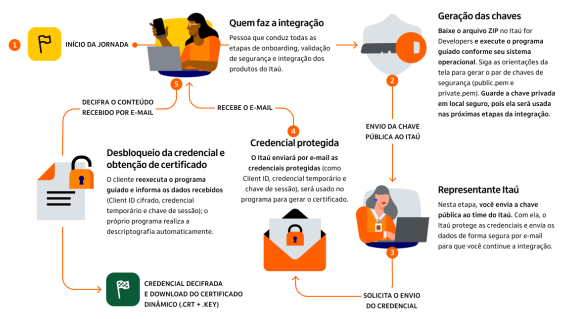
Como obter as credenciais e gerar o certificado dinâmico?

Confira neste guia como obter as suas credenciais e gerar o certificado dinâmico

Vai integrar com os produtos PISP, Corban Digital ou Extrato (Account Statement)?
A geração de credenciais e certificado é feita pela área logada do portal. Clique e veja o guia.

Atenção: Antes de iniciar o processo de obtenção das credenciais e geração do certificado dinâmico, é importante garantir que todos os pré-requisitos foram atendidos. Para isso, acesse Pré-requisitos para se conectar ao ambiente Itaú.


1
INÍCIO DA JORNADA
2
ENVIO DA CHAVE PÚBLICA AO ITAÚ
3
SOLICITA O ENVIO DO CREDENCIAL
4
RECEBE O E-MAIL
5
DECIFRA O CONTEÚDO RECEBIDO POR E-MAIL
CREDENCIAL DECIFRADA E DOWNLOAD DO CERTIFICADO DINÂMICO (.CRT + .KEY)

Quem faz a integração

Pessoa que conduz todas as etapas de onboarding, validação de segurança e integração dos produtos do Itaú.

Geração das chaves

Baixe o arquivo ZIP no Itaú for Developers **e execute o programa guiado conforme seu sistema operacional**. Siga as orientações da tela para gerar o par de chaves de segurança (public.pem e private.pem). **Guarde a chave privada em local seguro, pois ela será usada nas próximas etapas da integração.**

Representante Itaú

Nesta etapa, **você envia a chave pública ao time do Itaú.** Com ela, o Itaú protege as credenciais e envia os dados de forma segura por e-mail para que você continue a integração.

Credencial protegida

O Itaú enviará por e-mail as credenciais protegidas (como Client ID, credencial temporário e chave de sessão), será usado no programa para gerar o certificado.

Desbloqueio da credencial e obtenção de certificado

O cliente **reexecuta o programa guiado e informa os dados recebidos** (Client ID cifrado, credencial temporário e chave de sessão); o próprio programa realiza a descriptografia automaticamente.

Este manual tem como objetivo orientar o usuário na obtenção de credenciais e na geração do Certificado Dinâmico, ferramenta essencial para a autenticação junto à API do Itaú. Disponível em cinco linguagens de programação — Java, Python, Node.js, Go e C# e também por meio de um script de execução para Windows via Git Bash (`executar-gitbash.sh`), o programa oferece flexibilidade para que você escolha a tecnologia que melhor se adapta ao seu ambiente, garantindo a mesma funcionalidade completa em todas as opções. A seguir, apresentaremos os passos necessários para configurar e utilizar a ferramenta de forma eficiente e segura.

O que é isso?

Este programa ajuda você a obter o **Certificado Dinâmico** exigido para usar as APIs do Itaú. Ele faz todo o trabalho técnico por você, sendo necessário somente seguir as instruções na tela.

Você não precisa saber programar para usar este programa.

Resumo rápido: o que vai acontecer?

O programa vai guiar você por **4 etapas**:

Etapa	O que acontece	O que você precisa fazer
1	O programa gera suas chaves de segurança	Apenas digitar `1` e Enter
2	O programa gera seu certificado digital	Colar 3 valores que você recebeu por e-mail
3	O programa testa se tudo está funcionando	Apenas digitar `3` e Enter
4	Renovar o certificado (só quando necessário)	Informar um token de renovação

Ao final, você terá o **certificado digital** e as **credenciais** necessárias para integrar com as APIs do Itaú.

Como executar o programa

Existem **3 formas** de executar o processo. Escolha a que for mais fácil para você:

	Opção 1: Script de execução	Opção 2: Executar pelo terminal	Opção 3: Linha de comando
Dificuldade	Fácil	Médio	Avançado
Para quem	Todos	Quem já usa o terminal	Quem conhece openssl e curl
Precisa de openssl?	Não	Não	Sim
Menu interativo?	Sim	Sim	Não
Salva progresso?	Sim	Sim	Não

> ⚠ Importante: Caso nenhuma das 3 formas de execução do processo seja adequada ao seu cenário, consulte a documentação Certificado dinâmico (demais produtos).

Opção 1: Script de execução (Recomendado)

Esta é a forma mais simples de executar o processo. Você executa um único arquivo e o programa faz o resto.

Vantagens:

- Basta um duplo clique (Windows) ou um único comando (Linux/macOS)
- Menu para escolher a linguagem, sem precisar digitar comandos
- Verificação automática da instalação da linguagem escolhida
- Não é necessário instalar `openssl`
- Salva o progresso automaticamente
- Tratamento de erros com mensagens claras
- Logs de cada execução para facilitar o suporte
- Mascaramento automático de dados sensíveis

Passo a passo

Passo 1: Baixe arquivos

Faça o download do arquivo .zip com a estrutura de pastas.

Download do arquivo .zip

> ⚠ **Importante:**
>
> **É seguro usar este programa?**
>
> **Sim.** O programa:
>
> - **Não envia** suas chaves privadas para nenhum lugar;
> - **Mascara** informações sensíveis nos logs e na tela;
> - **Não armazena o Client Secret** em arquivo, ele aparece apenas uma vez na tela;
> - Salva todos os arquivos **localmente** no seu computador;
> - As chaves privadas (`private.pem` e `ARQUIVO_CHAVE_PRIVADA.key`) **nunca devem ser compartilhadas**.
>
> Caso não possa realizar o download do `.zip`, disponibilizamos os códigos dos scripts na seção Scripts. Basta copiá-los e salvar em um arquivo de acordo com as extensões e estrutura de pastas a seguir.

Estrutura de pastas

Após descompactar o arquivo, a estrutura de pastas será:

```
certificado-dinamico/
|-- executar.bat              <- Duplo clique para executar no Windows
|-- executar-gitbash.sh       <- Executar no Windows (Git Bash)
|-- executar.sh               <- Executar no Linux
|-- executar.zsh              <- Executar no macOS
|-- java/
|   |-- CertificadoDinamico.java
|-- python/
|   |-- certificado_dinamico.py
|-- node/
|   |-- certificadoDinamico.js
|-- go/
|   |-- main.go
|-- csharp/
|   |-- CertificadoDinamico.cs
|-- output/                   <- Criada automaticamente na primeira execucao
    |-- estado_certificado.json   <- Progresso salvo (etapas 1 e 2)
    |-- etapa1/
    |   |-- private.pem           <- Sua chave privada (NAO compartilhe)
    |   |-- public.pem            <- Sua chave publica (envie ao Analista de operacoes)
```

```
 |-- etapa2/
 |   |-- client_id.txt                  <- Client ID decifrado
 |   |-- token_temporario.txt           <- Token temporario decifrado
 |   |-- ARQUIVO_REQUEST_CERTIFICADO.csv  <- Pedido de certificado
 |   |-- ARQUIVO_CHAVE_PRIVADA.key        <- Chave privada do certificado
 |   |-- certificado.crt  <- Certificado assinado
 |   |-- certificado.pfx  <- (Opcional) PFX protegido por senha
 |-- etapa4/
 |   |-- certificado.crt  <- Certificado renovado
 |   |-- certificado.pfx  <- (Opcional) PFX protegido por senha
 |-- logs/
     |-- certificado_dinamico_YYYYMMDD_HHMMSS.log  <- Log de cada execucao
```

Passo 2: Descompacte o arquivo ZIP

Windows:

1. Localize o arquivo `.zip` na pasta de Downloads.
2. Clique com o botão direito no arquivo.
3. Selecione **"Extrair tudo..."**.
4. Escolha uma pasta de destino e clique em **"Extrair"**.

macOS:

1. Dê um duplo clique no arquivo `.zip` na pasta de Downloads.

Linux:

1. Clique com o botão direito no arquivo e selecione **"Extrair aqui"**, ou execute no terminal:

```
unzip certificado-dinamico.zip
```

Passo 3: Entre na pasta do projeto

> ⚠ **Importante:** Você precisa estar **dentro da pasta** `certificado-dinamico` para executar os scripts.

Windows:

1. Abra a pasta `certificado-dinamico` no Explorador de Arquivos (dê um duplo clique na pasta).

macOS / Linux:

1. Abra o Terminal.
2. Navegue até a pasta usando o comando `cd`. Por exemplo, se a pasta estiver em Downloads:

```
cd ~/Downloads/certificado-dinamico
```

Passo 4: Execute o script correspondente ao seu sistema operacional

Sistema	Como executar
Windows	Dê um duplo clique no arquivo `executar.bat`
Windows (Git Bash)	No terminal, execute: `chmod +x executar-gitbash.sh && ./executar-gitbash.sh`
Linux	No terminal, execute: `chmod +x executar.sh && ./executar.sh`
macOS	No terminal, execute: `chmod +x executar.zsh && ./executar.zsh`

> 💡 **Nota:** o comando `chmod +x` só precisa ser executado uma única vez. Ele dá permissão para o script ser executado.

Passo 5: Escolha a linguagem

O programa exibirá um menu como este:

```
=============================================
  Certificado Dinamico Itau - Launcher
=============================================

  Escolha a linguagem para executar:

  1. Java
  2. Python
  3. Node.js
  4. Go
  5. C# (.NET)

  0. Sair

Escolha (1-5, recomendado: 1):
```

Digite o número da linguagem desejada e pressione **Enter**. O programa verificará se a linguagem está instalada antes de continuar.

Dica: Se você não sabe qual escolher, digite `1` (Java), em caso de erro, tente digitar `2` (Python).

Passo 6: Siga as etapas do menu principal

Após escolher a linguagem, o programa exibirá o **menu principal** com as 4 etapas. Basta digitar o número da etapa e pressionar **Enter**. O programa sugere automaticamente a próxima etapa a ser executada.

Confira o detalhamento de cada etapa na seção Guia completos das etapas.

Opção 2: Executar pelo Terminal

Se você já sabe abrir o terminal e executar comandos, pode rodar o programa diretamente, sem usar o script launcher. O resultado é o mesmo da Opção 1.

> ⚠ **Importante:** Entre na pasta da linguagem escolhida dentro do projeto antes de executar.

Java (Recomendado)

```
cd java
javac CertificadoDinamico.java
java CertificadoDinamico
```

> 💡 **Dica**: O comando `javac` (compilação) precisa ser executado uma única vez.

Python

```
cd python
pip install requests
pip install cryptography
python certificado_dinamico.py
```

> 💡 **Dica**: No Linux/macOS pode ser necessario usar `python3` e `pip3`.

Node.js

```
cd node
node certificadoDinamico.js
```

Go

```
cd go
go run main.go
```

C# (.NET)

```
cd csharp
dotnet-script CertificadoDinamico.cs
```

> 💡 **Dica**: Se o `dotnet-script` não estiver disponível, crie um arquivo `CertificadoDinamico.csproj` com o seguinte conteúdo e use `dotnet run`:
>
> ```
> <Project Sdk="Microsoft.NET.Sdk">
>   <PropertyGroup>
>     <OutputType>Exe</OutputType>
>     <TargetFramework>net8.0</TargetFramework>
>   </PropertyGroup>
> </Project>
> ```
>
> ```
> dotnet run
> ```

Após executar, o **menu principal** será exibido. Confira o Guia completo das etapas.

Opção 3: Linha de comando com OpenSSL e cURL

Para usuários avançados. Exige `openssl` e `curl` instalados. As etapas a seguir correspondem as do menu scripts.

Etapa 1 - Gerar chaves (openssl)

Gerar a chave privada:

```
openssl genpkey -algorithm RSA -out private.pem -pkeyopt rsa_keygen_bits:2048
```

Extrair a chave pública:

```
openssl rsa -in private.pem -pubout -out public.pem
```

Exibir a chave pública:

```
cat public.pem
```

Windows: Use `type public.pem` em vez de `cat`.

Copie **todo** o conteúdo exibido (incluindo `-----BEGIN PUBLIC KEY-----` e `-----END PUBLIC KEY-----`) e envie ao Analista de Operações Itaú.

Etapa 2 - Gerar certificado (openssl + curl)

Após receber as credenciais criptografadas por e-mail:

Descriptografar a chave de sessão:

```
echo 'CHAVE_DE_SESSAO_CIFRADA' | base64 -d | openssl rsautl -decrypt -inkey private.pem -out session_key.bin
```

Descriptografar o Client ID:

```
echo 'CLIENT_ID_CIFRADO' | base64 -d | openssl enc -aes-256-cbc -d -K $(xxd -p session_key.bin) -iv 0 -out client_
```

Descriptografar o Token temporário:

```
echo 'TOKEN_TEMPORARIO_CIFRADO' | base64 -d | openssl enc -aes-256-cbc -d -K $(xxd -p session_key.bin) -iv 0 -out
```

⚠ **Importante**: Substitua os valores entre aspas pelos valores que você recebeu por e-mail.

Gerar CSR:

```
openssl req -new -newkey rsa:2048 -nodes \
  -keyout ARQUIVO_CHAVE_PRIVADA.key \
  -out ARQUIVO_REQUEST_CERTIFICADO.csr \
  -sha512 \
  -subj "/CN=SEU_CLIENT_ID/OU=SUA_ORGANIZACAO/L=SAO PAULO/ST=SP/C=BR"
```

⚠ **Importante**: Substitua `SEU_CLIENT_ID` pelo valor decifrado e `SUA_ORGANIZACAO` pelo nome da sua empresa.

Enviar CSR ao Itaú:

```
curl -X POST 'https://sts.itau.com.br/seguranca/v1/certificado/solicitacao' \
  -H 'Content-Type: text/plain' \
  -H "Authorization: Bearer $(cat token.txt)" \
  -d "$(cat ARQUIVO_REQUEST_CERTIFICADO.csr)"
```

A resposta conterá o certificado assinado e o `client_secret`. Salve o certificado:

```
echo '-----BEGIN CERTIFICATE-----
... conteudo do certificado retornado ...
-----END CERTIFICATE-----' > certificado.crt
```

⚠ **Importante:** O Client Secret e equivalente a uma **Senha**. Ele é exibido **uma única vez**. Copie e guarde em local seguro (ex.: gerenciador de senhas). Não compartilhe com ninguém.

Etapa 3 - Testar token OAuth (curl)

```
curl -X POST 'https://sts.itau.com.br/as/token.oauth2' \
  --cert certificado.crt \
  --key ARQUIVO_CHAVE_PRIVADA.key \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'grant_type=client_credentials&client_id=SEU_CLIENT_ID&client_secret=SEU_CLIENT_SECRET'
```

⚠ **Importante:** Substitua `SEU_CLIENT_ID` e `SEU_CLIENT_SECRET` pelos valores obtidos na etapa 2.

Resultado: A resposta contera o `access_token`. Este e o token de acesso que deve ser informado no header `Authorization: Bearer <access_token>` das requisições as APIs do Itaú. O token tem validade de **300 segundos (5 minutos)**.

Nota: Esta etapa é para **testes**. Na sua aplicação final, você deverá implementar esse fluxo OAuth diretamente no seu código, renovando o token a cada 5 minutos.

Etapa 4 - Renovar certificado (curl)

Renova o certificado dinâmico quando ele está próximo da expiração (menos de 30 dias).

```
curl -X POST 'https://sts.itau.com.br/seguranca/v1/certificado/renovacao' \
  --cert certificado.crt \
  --key ARQUIVO_CHAVE_PRIVADA.key \
  -H 'Content-Type: text/plain' \
  -H "Authorization: Bearer SEU_TOKEN_STS" \
  -d "$(cat ARQUIVO_REQUEST_CERTIFICADO.csr)"
```

Importante: Substitua `SEU_TOKEN_STS` por um token válido para autenticação no STS Itaú.

Guia completo das etapas

Esta seção detalha **exatamente** o que acontece e o que você precisa fazer em cada etapa do menu interativo (Opções 1 e 2).

Como funciona o menu

Ao abrir o programa, você pode conferir o menu principal:

```
=============================================
  Certificado Dinamico Itau - Menu Principal
=============================================

  Progresso atual: Etapa 0 de 3 concluida(s)

 1. Gerar par de chaves e exibir chave publica  [PROXIMA >>] <------ voce esta aqui
  2. Descriptografar credenciais, gerar e enviar certificado  [PENDENTE]
  3. Testar geracao de token                      [PENDENTE]
  4. Renovar certificado

  0. Sair
  9. Recomecar do zero (limpa progresso)
```

Como ler o menu:

- **[CONCLUIDA]** = Você já fez esta etapa
- **[PROXIMA >>] < voce esta aqui** = Esta é a próxima etapa que você deve fazer
- **[PENDENTE]** = Você ainda não pode fazer esta etapa (precisa terminar as anteriores primeiro)

Dica: O programa sugere automaticamente a próxima etapa. Se você pressionar **Enter** (sem digitar nada), ele executará a etapa sugerida.

Etapa 1 - Gerar par de chaves de segurança

Objetivo: Criar um par de chaves criptográficas (uma "chave privada" e uma "chave pública") e mostrar a chave pública para você enviar ao Analista de operações Itaú.

O que você precisa fazer:

1. Digite `1` e pressione **Enter**
2. Pronto! O programa gera as chaves e mostra a chave pública na tela

O que você vai ver na tela:

```
[PASSO 1.1] Gerando par de chaves RSA 2048...
Chave privada salva em /caminho/completo/output/etapa1/private.pem
Chave publica salva em /caminho/completo/output/etapa1/public.pem

[PASSO 1.2] Exibindo chave publica...

=============================================
  Chave Publica Gerada
=============================================
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A...
(varias linhas de texto)
...
-----END PUBLIC KEY-----

>> Etapa 1 concluida. Progresso salvo.
```

```
  O que fazer agora:
  1. Copie o conteudo da chave publica exibido acima
  2. Envie a chave publica para o Analista de operacoes Itau
  3. Aguarde o e-mail do Itau com as credenciais cifradas
  4. Quando receber o e-mail, execute a etapa 2
```

Próximo passo (Fora do programa):

Copie **todo** o texto da chave pública (incluindo as linhas `-----BEGIN PUBLIC KEY-----` e `-----END PUBLIC KEY-----`) e envie para o **Analista de operações Itaú**. Ele usará essa chave para gerar as credenciais criptografadas que você usará na etapa 2.

> ⚠ **Atenção**: Aguarde, o analista de operações Itaú precisará de algum tempo para processar. Você receberá um **e-mail** com 3 valores criptografados. Só então poderá continuar para a etapa 2.

Arquivos gerados:

Arquivo	Onde fica	Para que serve
`private.pem`	`output/etapa1/`	Chave privada. **NÃO compartilhe este arquivo com ninguém.**
`public.pem`	`output/etapa1/`	Chave pública. Envie ao Analista de operações Itaú.

Etapa 2 - Gerar o certificado digital

Objetivo: Utilizar as credenciais recebidas por e-mail para gerar o certificado digital.

Antes de começar, certifique-se de ter

- O **e-mail do Itaú** contendo os três valores criptografados:
  - Client ID criptografado
  - Token temporário criptografado
  - Chave de sessão criptografada

O que você precisa fazer:

1. Digite `2` e pressione **Enter**.
2. O programa solicitará três valores. Para cada um, **copie do e-mail e cole no terminal**:

Valor solicitado	Origem para copiar
`Client ID cifrado:`	Copie o valor "Client ID criptografado" do e-mail
`Token temporário cifrado:`	Copie o valor "Token temporário criptografado" do e-mail
`Chave de sessão cifrada:`	Copie o valor "Chave de sessão criptografada" do e-mail

> 💡 **Dica**
> **Como colar no terminal:**
> - **Windows (Prompt de Comando):** Clique com o botão **direito** do mouse.
> - **Windows (PowerShell/Terminal):** `Ctrl + V`.
> - **macOS:** `Cmd + V`.
> - **Linux:** `Ctrl + Shift + V`.

3. O programa solicitará os dados da sua empresa. **Todos os campos são obrigatórios**. Caso você pressione Enter sem digitar nada, o programa solicitará o preenchimento novamente.

Campo	Exemplo
`Nome da Empresa:`	Minha Empresa LTDA
`Cidade:`	SAO PAULO
`Estado - UF:`	SP
`País:`	BR

4. O programa realizará todo o processo automaticamente (descriptografar, gerar CSR, enviar ao Itaú).

O que acontece ao final:

O programa exibirá uma tela com os arquivos gerados e, **muito importante**, o **Client Secret**:

```
==============================================
  ATENCAO - CLIENT SECRET
==============================================

O Client Secret e equivalente a uma SENHA. Trate-o com o mesmo cuidado.
O valor do Client Secret sera exibido APENAS NESTE MOMENTO.
Copie e salve em um local seguro. Este valor NAO sera armazenado pelo programa.

  Client Secret: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

```
==============================================
Guarde essas informacoes em local seguro. O Client ID e o Client Secret sao equivalentes a senhas
e serao necessarios para gerar tokens OAuth e consumir as APIs do Itau.
NAO compartilhe essas credenciais. Quem tiver acesso a elas podera acessar as APIs em seu nome.

O Itau nao se responsabiliza pelo armazenamento local das credenciais.
A guarda correta e de responsabilidade exclusiva do cliente.
```

> ⚠️ **Atenção:** O **Client Secret** é equivalente a uma **senha** e aparece **apenas uma única vez** neste momento. Se você não copiar neste momento, precisará refazer todo o processo. **Copie e guarde em local seguro imediatamente** (ex.: gerenciador de senhas). **Não compartilhe, Não envie por e-mail e Não deixe exposto em código-fonte.** Quem tiver acesso ao Client ID e Client Secret poderá acessar as APIs em seu nome.

Arquivos gerados:

Arquivo	Onde fica	Para que serve
`client_id.txt`	`output/etapa2/`	Client ID decifrado
`token_temporario.txt`	`output/etapa2/`	Token temporário decifrado
`ARQUIVO_REQUEST_CERTIFICADO.csr`	`output/etapa2/`	Pedido de certificado (CSR)
`ARQUIVO_CHAVE_PRIVADA.key`	`output/etapa2/`	Chave privada do certificado. **Não compartilhe.**
`certificado.crt`	`output/etapa2/`	Certificado digital assinado pelo Itaú
`certificado.pfx`	`output/etapa2/`	(Opcional) Certificado + chave privada em formato PKCS#12 protegido por senha

> 💡 **PFX (opcional):** Após a etapa 2, o programa perguntará se você deseja gerar um arquivo `.pfx`. O PFX combina o certificado e a chave privada em um único arquivo protegido por senha. É útil para plataformas que trabalham com esse formato (ex: .NET, Java KeyStore, IIS, Azure). Se não precisar, basta responder `N`.

Validade do certificado:

O programa exibirá na tela as informações de validade:

```
==============================================
  VALIDADE DO CERTIFICADO
==============================================
  Validade: 365 dias
  Data de expiracao: DD/MM/AAAA
  Periodo de renovacao: de DD/MM/AAAA ate DD/MM/AAAA

  O processo de renovacao (etapa 4) pode ser realizado a partir de
  30 dias antes da expiracao ate um dia antes da data de expiracao do certificado.
```

> ⚠️ **Importante**: O certificado é válido por **365 dias** a partir da data de geração. A **renovação pode ser feita a partir de 30 dias antes da expiração** até um dia antes da data de expiração. Anote a data de expiração para não perder o prazo!
> **A responsabilidade de renovar o certificado dentro do prazo é exclusivamente sua.** Certificados expirados impossibilitam o acesso às APIs do Itaú e exigem a geração de um novo certificado do zero.

Etapa 3 - Testar o certificado

Objetivo: Verificar se o certificado e as credenciais estão funcionando corretamente, gerando um token de acesso de teste.

O que você precisa fazer:

1. Digite `3` e pressione **Enter**.
2. Se você acabou de executar a etapa 2 (sem fechar o programa), o Client Secret será usado automaticamente.
3. Se você reiniciou o programa, ele pedirá: `Informe o Client Secret:` - cole o valor que você guardou na etapa 2.

O que acontece:

O programa usa o certificado e as credenciais para gerar um **token de acesso OAuth**. Se tudo estiver correto, o programa informará:

```
==============================================
  Credenciais e certificado VALIDADOS!
==============================================

Access Token:
eyJhbGciOiJSUzI1NiIs... (token longo)

  Validade: 300 segundos (5 minutos)
```

```
  Uso: Informe este token no header 'Authorization: Bearer <access_token>'
       das requisicoes as APIs do Itau.

As credenciais (client_id e client_secret) e o certificado digital estao
validados e funcionais. Voce ja pode utiliza-los em suas integracoes.
```

> 💡 **Nota**: Esta etapa é apenas para **teste**. Na sua aplicação final, você deverá implementar esse fluxo OAuth no seu código, renovando o token a cada 5 minutos.

Etapa 4 - Renovar o certificado

Objetivo: Renovar o certificado digital quando ele estiver próximo de expirar.

Quando usar: Somente quando o certificado estiver perto da data de vencimento (menos de 30 dias). Esta opção está **sempre disponível** no menu, independente do progresso das outras etapas.

O que você precisa fazer:

1. Digite `4` e pressione **Enter**.
2. Informe o **Token STS** para renovação quando solicitado.
3. Se o arquivo CSR da etapa 2 for encontrado automaticamente, o programa o utiliza. Caso contrário, pedirá o caminho do arquivo.

Arquivo gerado:

Arquivo	Onde fica	Para que serve
`certificado.crt`	`output/etapa4/`	Novo certificado digital renovado
`certificado.pfx`	`output/etapa4/`	(Opcional) Certificado renovado + chave privada em formato PKCS#12 protegido por senha

> 💡 **PFX (opcional):** Assim como na etapa 2, o programa perguntara se você deseja gerar o arquivo `.pfx` para o certificado renovado.

Validade do certificado renovado:

Após a renovação, o programa exibirá as novas datas de validade:

```
===============================================
  VALIDADE DO CERTIFICADO RENOVADO
===============================================
  Validade: 365 dias
  Data de expiracao: DD/MM/AAAA
  Periodo de renovacao: de DD/MM/AAAA ate DD/MM/AAAA

  O processo de renovacao (etapa 4) pode ser realizado a partir de
  30 dias antes da expiracao ate um dia antes da data de expiracao do certificado.
```

> ⚠ **Importante:** O certificado renovado também é válido por **365 dias**. Anote a nova data de expiração.
> **A responsabilidade pela renovação do certificado dentro do prazo é exclusivamente sua.** Certificados expirados impedem o acesso às APIs do Itaú e exigem a geração de um novo certificado do zero.

Opção 9 - Recomeçar do zero

Se você precisar refazer todo o processo desde o início, digite `9` no menu. **Isso apaga todos os arquivos gerados e o progresso salvo.**

> ⚠ **Cuidado:** Esta ação é **irreversível**. Todos os arquivos (chaves, certificados, estado) serão excluídos permanentemente.

Opção 0 - Sair

Encerra o programa. O progresso das **etapas 1 e 2** é salvo automaticamente, na próxima vez que abrir o programa, ele continuará de onde parou.

> 💡 **Nota:** A **etapa 3** (teste de token) não fica salva. Se você fechar e reabrir o programa, precisará reexecutar a etapa 3. Isso é normal, ela é apenas um teste.

Perguntas frequentes (FAQ)

Preciso saber programar?

Não. O programa é interativo e guia você por cada etapa. Você só precisa digitar números e pressionar Enter. A linguagem (Java, Python, etc.) é usada internamente — você não precisa entender nada sobre ela.

Preciso descompactar o arquivo ZIP?

Sim. O programa não funciona de dentro do arquivo `.zip`. Descompacte primeiro. Confira as Descompacte o arquivo zip.

Preciso estar dentro da pasta para executar?

Sim. O programa precisa encontrar os arquivos internos. No Windows, abra a pasta no Explorador e dê duplo clique no `executar.bat`. No Linux/macOS, use `cd` para navegar até a pasta.

Qual a diferença entre as 3 opções de execução?

- **Opção 1 (Script de execução):** Mais fácil. Duplo clique e siga o menu. Para a maioria dos usuários.
- **Opção 2 (Terminal):** Você executa direto pela linguagem. Para quem já usa terminal.
- **Opção 3 (openssl/curl):** Você executa cada comando manualmente. Para usuários avançados ou auditoria.

Preciso copiar e colar código?

Não. Você não precisa copiar nenhum código de programação. O único momento em que precisará colar algo é na **etapa 2**, quando o programa pede as credenciais criptografadas que você recebeu por e-mail.

O que fazer se o programa fechar sem querer?

Não se preocupe. O progresso das **etapas 1 e 2** é salvo automaticamente. Basta abrir o programa novamente e ele continuará de onde parou. Apenas a **etapa 3** (teste de token) precisará ser reexecutada, pois não fica salva.

Posso usar qualquer uma das 5 linguagens?

Sim. Todas fazem exatamente a mesma coisa. Escolha a que já estiver instalada. Se não souber, tente **Java** (opção 1). Se der erro, tente **Python** (opção 2).

O que fazer se aparecer "comando não encontrado"?

A linguagem não está instalada ou não está no *Path*. Confira a seção Pré-requisitos e instale. Ou simplesmente tente outra linguagem no menu.

O que é o Client Secret e por que ele só aparece uma vez?

O **Client Secret** é equivalente a uma **Senha** da sua aplicação. Assim como uma senha de banco, **quem tiver acesso a ele poderá acessar as APIs em seu nome**. Por segurança, o programa exibe esse valor **apenas uma vez** na etapa 2 e **não o salva em nenhum arquivo**. Se você perder esse valor, precisará refazer o processo (opção 9 para recomeçar do zero). O mesmo vale para o **Client ID**, o **certificado** e a **chave privada**, todos são equivalentes a senhas e **Não devem ser compartilhados**.

O que fazer se a etapa 2 retornar erro 403?

O erro 403 significa "acesso negado". Geralmente indica que o **token expirou**. Solicite novas credenciais ao Analista de Operações Itaú e execute novamente a etapa 2.

O que fazer se aparecer "NullPointerException" na resposta?

Isso pode indicar que os dados enviados estavam incompletos. Use a opção `9` para recomeçar do zero e refaça todas as etapas.

Onde ficam os logs?

Na pasta `output/logs/`. O arquivo de log é criado automaticamente a cada execução. Informações sensíveis (tokens, senhas) são mascaradas nos logs, então é seguro compartilhá-los com o suporte.

Como renovar o certificado?

Use a opção `4` no menu. Ela está sempre disponível. Utilize quando o certificado estiver próximo da expiração (menos de 30 dias).

Posso executar em outro computador?

Sim, desde que a linguagem esteja instalada. Porém, os arquivos gerados ficam na pasta `output/` do computador original. Se precisar transferir, copie **toda** a pasta `certificado-dinamico/` (incluindo `output/`).

É seguro usar este programa?

Sim. O programa:

- **Não envia** suas chaves privadas para nenhum lugar;
- **Mascara** informações sensíveis nos logs e na tela;
- **Não armazena o Client Secret** em arquivo, ele aparece apenas uma vez na tela;
- Salva todos os arquivos **localmente** no seu computador;
- As chaves privadas (`private.pem` e `ARQUIVO_CHAVE_PRIVADA.key`) **nunca devem ser compartilhadas**.

> **Lembre-se:** Client ID, Client Secret, certificado e chave privada são **equivalentes a senhas**. Não compartilhe, não envie por e-mail e não deixe exposto em código-fonte. Trate essas credenciais com o mesmo cuidado que você trata suas senhas bancárias.
> **O Itaú não se responsabiliza pelo armazenamento local das credenciais. A guarda correta é de responsabilidade exclusiva do cliente.**

Resolução de problemas

"javac não é reconhecido como comando interno ou externo"

O Java não está instalado ou não está no PATH. Instale em https://adoptium.net/ e reinicie o terminal.

"python não é reconhecido como comando interno ou externo"

O Python não está instalado ou não está no PATH. Instale em https://www.python.org/downloads/ e marque **"Add Python to PATH"** durante a instalação.

"Erro ao descriptografar as credenciais"

Os valores criptografados copiados do e-mail podem estar incorretos ou incompletos. Certifique-se de copiar o texto **completo** de cada campo, sem espaços extras no início ou final.

"Status 403 - Acesso negado"

O token provavelmente expirou. Solicite novas credenciais ao Analista de Operações Itaú e refaça a etapa 2.

"Status 500 - Erro interno do servidor"

Erro no lado do Itaú. Tente novamente em alguns minutos.

O programa não encontra os arquivos

Verifique se você está executando o programa de dentro da pasta `certificado-dinamico`. Os caminhos dos arquivos são relativos a essa pasta.

O script `.bat` não executa ao dar duplo clique

Tente: clique com o botão direito > **"Executar como administrador"**. Ou abra o Prompt de Comando, navegue até a pasta e execute `executar.bat`.

Suporte

Em caso de dúvidas ou problemas:

- Verifique os logs em `output/logs/` para detalhes do erro.

> Para dúvidas que não forem esclarecidas por essa documentação, entre em contato conosco diretamente pelos canais de suporte combinados com o seu representante comercial Itaú.

Scripts

> **Importante:** Os códigos e comandos a seguir são os mesmos que você encontrará no arquivo .zip para download. Para utilizá-los é necessário copiar cada arquivo e salvá-lo com a extensão correspondente, além de criar a estrutura de pastas conforme disponibilizado em Estrutura de pastas.

Executável Windows Executável Mac Executável Linux

```
@echo off
chcp 65001 >nul 2>&1
setlocal enabledelayedexpansion
title Certificado Dinamico Itau

set "SCRIPT_DIR=%~dp0"

echo.
echo =============================================
echo   Certificado Dinamico Itau
echo =============================================
echo.
echo   Escolha a linguagem para executar:
echo.
echo   1. Java (recomendado)
echo   2. Python
echo   3. Node.js
echo   4. Go
echo   5. C# (.NET)
echo.
echo   0. Sair
echo.

set "escolha=1"
set /p escolha="Escolha (1-5, padrao: 1 - Java): "

if "%escolha%"=="0" goto opt_sair
if "%escolha%"=="1" goto opt_java
if "%escolha%"=="2" goto opt_python
if "%escolha%"=="3" goto opt_node
if "%escolha%"=="4" goto opt_go
if "%escolha%"=="5" goto opt_csharp
goto opt_invalida

:opt_sair
echo Saindo...
exit /b 0

:opt_java
echo.
echo Verificando pre-requisitos para Java...
echo.
set "PREREQ_OK=1"
where javac >nul 2>&1
if errorlevel 1 (
    echo   [ERRO] javac nao encontrado. Instale o JDK.
    set "PREREQ_OK=0"
) else (
    for /f "tokens=*" %%v in ('javac -version 2^>^&1') do echo   [OK] %%v
)
where java >nul 2>&1
if errorlevel 1 (
    echo   [ERRO] java nao encontrado. Instale o JDK.
    set "PREREQ_OK=0"
) else (
    for /f "tokens=*" %%v in ('java -version 2^>^&1') do echo   [OK] %%v
)
if exist "%SCRIPT_DIR%java\CertificadoDinamico.java" (
    echo   [OK] Arquivo CertificadoDinamico.java encontrado
) else (
    echo   [ERRO] Arquivo CertificadoDinamico.java nao encontrado
    set "PREREQ_OK=0"
)
echo.
if "%PREREQ_OK%"=="0" (
    echo Pre-requisitos nao atendidos. Instale o JDK e adicione ao PATH.
    echo Download: https://adoptium.net/
    pause
    exit /b 1
)
echo Pre-requisitos atendidos. Iniciando programa...
echo.
pushd "%SCRIPT_DIR%java"
if not exist "CertificadoDinamico.class" (
    echo Compilando CertificadoDinamico.java...
    javac CertificadoDinamico.java
    if errorlevel 1 (
        echo [ERRO] Falha na compilacao.
        popd
        pause
        exit /b 1
    )
)
java CertificadoDinamico
popd
goto fim

:opt_python
echo.
echo Verificando pre-requisitos para Python...
echo.
set "PREREQ_OK=1"
set "PYTHON_CMD="
```

```
REM 1) Tentar primeiro o Python Launcher (py), que vem com a instalacao oficial
where py >nul 2>&1
if not errorlevel 1 (
    py -3 --version >nul 2>&1
    if not errorlevel 1 set "PYTHON_CMD=py -3"
)

REM 2) Fallback: python3 no PATH, ignorando o stub do Microsoft Store (WindowsApps)
if "!PYTHON_CMD!"=="" (
    for /f "delims=" %%P in ('where python3 2^>nul') do (
        if "!PYTHON_CMD!"=="" (
            echo %%P| findstr /i "WindowsApps" >nul
            if errorlevel 1 set "PYTHON_CMD="%%P""
        )
    )
)

REM 3) Fallback: python no PATH, ignorando o stub do Microsoft Store (WindowsApps)
if "!PYTHON_CMD!"=="" (
    for /f "delims=" %%P in ('where python 2^>nul') do (
        if "!PYTHON_CMD!"=="" (
            echo %%P| findstr /i "WindowsApps" >nul
            if errorlevel 1 set "PYTHON_CMD="%%P""
        )
    )
)

if "!PYTHON_CMD!"=="" (
    echo   [ERRO] python nao encontrado ^(ou apenas o stub da Microsoft Store esta no PATH^)
    echo          Desabilite os aliases em: Configuracoes ^> Aplicativos ^> Aliases de execucao do aplicativo
    echo          ou instale o Python em https://www.python.org/downloads/
    set "PREREQ_OK=0"
) else (
    !PYTHON_CMD! --version > "%TEMP%\pyver.txt" 2>&1
    set /p PYVER=<"%TEMP%\pyver.txt"
    del "%TEMP%\pyver.txt" >nul 2>&1
    echo   [OK] !PYVER!
)
if not "!PYTHON_CMD!"=="" (
    !PYTHON_CMD! -c "import requests" >nul 2>&1
    if errorlevel 1 (
        echo   [AVISO] Biblioteca 'requests' nao encontrada - sera instalada automaticamente
    ) else (
        echo   [OK] Biblioteca 'requests' disponivel
    )
    !PYTHON_CMD! -c "import cryptography" >nul 2>&1
    if errorlevel 1 (
        echo   [AVISO] Biblioteca 'cryptography' nao encontrada - sera instalada automaticamente
    ) else (
        echo   [OK] Biblioteca 'cryptography' disponivel
    )
)
if exist "%SCRIPT_DIR%python\certificado_dinamico.py" (
    echo   [OK] Arquivo certificado_dinamico.py encontrado
) else (
    echo   [ERRO] Arquivo certificado_dinamico.py nao encontrado
    set "PREREQ_OK=0"
)
echo.
if "!PREREQ_OK!"=="0" (
    echo Pre-requisitos nao atendidos. Instale o Python e adicione ao PATH.
    echo Download: https://www.python.org/downloads/
    pause
    exit /b 1
)
echo Pre-requisitos atendidos. Iniciando programa...
echo.
pushd "%SCRIPT_DIR%python"
echo Instalando dependencias...

REM Em ambientes corporativos com SSL inspection (ex: proxy Itau), o pip precisa de:
REM  - usar o proxy correto (a variavel HTTPS_PROXY pode apontar para porta inativa)
REM  - confiar nos hosts do PyPI para nao falhar na verificacao do certificado MITM
set "PIP_PROXY_ARG="
if defined HTTP_PROXY set "PIP_PROXY_ARG=--proxy %HTTP_PROXY%"
set "PIP_TRUSTED=--trusted-host pypi.org --trusted-host files.pythonhosted.org --trusted-host pypi.python.org"

!PYTHON_CMD! -m pip install %PIP_PROXY_ARG% %PIP_TRUSTED% requests --quiet
!PYTHON_CMD! -m pip install %PIP_PROXY_ARG% %PIP_TRUSTED% cryptography --quiet
!PYTHON_CMD! certificado_dinamico.py
popd
goto fim

:opt_node
echo.
echo Verificando pre-requisitos para Node.js...
echo.
set "PREREQ_OK=1"
where node >nul 2>&1
```

```
if errorlevel 1 (
    echo   [ERRO] node nao encontrado
    set "PREREQ_OK=0"
) else (
    for /f "tokens=*" %%v in ('node --version 2^>^&1') do echo   [OK] Node.js %%v
)
if exist "%SCRIPT_DIR%node\certificadoDinamico.js" (
    echo   [OK] Arquivo certificadoDinamico.js encontrado
) else (
    echo   [ERRO] Arquivo certificadoDinamico.js nao encontrado
    set "PREREQ_OK=0"
)
echo.
if "%PREREQ_OK%"=="0" (
    echo Pre-requisitos nao atendidos. Instale o Node.js e adicione ao PATH.
    echo Download: https://nodejs.org/
    pause
    exit /b 1
)
echo Pre-requisitos atendidos. Iniciando programa...
echo.
pushd "%SCRIPT_DIR%node"
node certificadoDinamico.js
popd
goto fim

:opt_go
echo.
echo Verificando pre-requisitos para Go...
echo.
set "PREREQ_OK=1"
where go >nul 2>&1
if errorlevel 1 (
    echo   [ERRO] go nao encontrado
    set "PREREQ_OK=0"
) else (
    for /f "tokens=3" %%v in ('go version 2^>^&1') do echo   [OK] Go %%v
)
if exist "%SCRIPT_DIR%go\main.go" (
    echo   [OK] Arquivo main.go encontrado
) else (
    echo   [ERRO] Arquivo main.go nao encontrado
    set "PREREQ_OK=0"
)
echo.
if "%PREREQ_OK%"=="0" (
    echo Pre-requisitos nao atendidos. Instale o Go e adicione ao PATH.
    echo Download: https://go.dev/dl/
    pause
    exit /b 1
)
echo Pre-requisitos atendidos. Iniciando programa...
echo.
pushd "%SCRIPT_DIR%go"
go run main.go
popd
goto fim

:opt_csharp
echo.
echo Verificando pre-requisitos para C# .NET...
echo.
set "PREREQ_OK=1"
where dotnet >nul 2>&1
if errorlevel 1 (
    echo   [ERRO] dotnet nao encontrado
    set "PREREQ_OK=0"
) else (
    for /f "tokens=*" %%v in ('dotnet --version 2^>^&1') do echo   [OK] .NET SDK %%v
)
if exist "%SCRIPT_DIR%csharp\CertificadoDinamico.cs" (
    echo   [OK] Arquivo CertificadoDinamico.cs encontrado
) else (
    echo   [ERRO] Arquivo CertificadoDinamico.cs nao encontrado
    set "PREREQ_OK=0"
)
echo.
if "%PREREQ_OK%"=="0" (
    echo Pre-requisitos nao atendidos. Instale o .NET SDK e adicione ao PATH.
    echo Download: https://dotnet.microsoft.com/download
    pause
    exit /b 1
)
echo Pre-requisitos atendidos. Iniciando programa...
echo.
pushd "%SCRIPT_DIR%csharp"
if not exist "CertificadoDinamico.csproj" (
    echo ^<Project Sdk="Microsoft.NET.Sdk"^> > CertificadoDinamico.csproj
    echo   ^<PropertyGroup^> >> CertificadoDinamico.csproj
    echo     ^<OutputType^>Exe^</OutputType^> >> CertificadoDinamico.csproj
    echo     ^<TargetFramework^>net8.0^</TargetFramework^> >> CertificadoDinamico.csproj
    echo   ^</PropertyGroup^> >> CertificadoDinamico.csproj
    echo ^</Project^> >> CertificadoDinamico.csproj
)
dotnet run
```

```
popd
goto fim

:opt_invalida
echo Opcao invalida.
pause
exit /b 1

:fim
pause
exit /b 0
```

Scripts de certificado dinâmico

C# GO Java Node Python

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Security;
using System.Runtime.CompilerServices;
using System.Security.Authentication;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Text.Json;
using System.Text.RegularExpressions;

public class CertificadoDinamico
{
    public static RSA GerarParDeChaves(string caminhoChavePrivada = "private.pem", string caminhoChavePublica = "pu
    {
        RSA rsa = RSA.Create(2048);

        string privateKeyPem = rsa.ExportPkcs8PrivateKeyPem();
        File.WriteAllText(caminhoChavePrivada, privateKeyPem + Environment.NewLine);
        Console.WriteLine($"Chave privada salva em {Path.GetFullPath(caminhoChavePrivada)}");

        string publicKeyPem = rsa.ExportSubjectPublicKeyInfoPem();
        File.WriteAllText(caminhoChavePublica, publicKeyPem + Environment.NewLine);
        Console.WriteLine($"Chave publica salva em {Path.GetFullPath(caminhoChavePublica)}");

        return rsa;
    }

    private static RSA CarregarChavePrivada(string caminhoChavePrivada)
    {
        string pemContent = File.ReadAllText(caminhoChavePrivada);
        RSA rsa = RSA.Create();
        rsa.ImportFromPem(pemContent);
        return rsa;
    }

    // Carrega certificado + chave privada para uso em mTLS de forma compativel com
    // o SChannel (stack TLS nativa do Windows). X509Certificate2.CreateFromPem
    // associa a chave em memoria, mas no Windows ela nao fica persistida no
    // formato esperado pelo SChannel, o que causa o handshake mTLS a falhar com
    // "The SSL connection could not be established". A solucao padrao e
    // reexportar para PFX em memoria e reimportar.
    private static X509Certificate2 CarregarCertificadoMtls(string caminhoCRT, string caminhoKey)
    {
        string certPem = File.ReadAllText(caminhoCRT);
        string keyPem = File.ReadAllText(caminhoKey);
        using var certComChave = X509Certificate2.CreateFromPem(certPem, keyPem);
        byte[] pfxBytes = certComChave.Export(X509ContentType.Pkcs12);
        return new X509Certificate2(pfxBytes, (string)null, X509KeyStorageFlags.Exportable);
    }

    private static byte[] DescriptografarRsa(string caminhoChavePrivada, string dadosCifradosBase64)
    {
        using (RSA rsa = CarregarChavePrivada(caminhoChavePrivada))
        {
            byte[] dadosCifrados = Convert.FromBase64String(dadosCifradosBase64);
            return rsa.Decrypt(dadosCifrados, RSAEncryptionPadding.Pkcs1);
        }
    }

    private static string DescriptografarAes(byte[] chaveAes, string textoCifradoBase64)
    {
        byte[] textoCifrado = Convert.FromBase64String(textoCifradoBase64);
        byte[] iv = new byte[16];

        using (Aes aes = Aes.Create())
        {
```

```
                aes.Key = chaveAes;
                aes.IV = iv;
                aes.Mode = CipherMode.CBC;
                aes.Padding = PaddingMode.PKCS7;

                using (ICryptoTransform decryptor = aes.CreateDecryptor())
                {
                    byte[] textoDecifrado = decryptor.TransformFinalBlock(textoCifrado, 0, textoCifrado.Length);
                    return Encoding.UTF8.GetString(textoDecifrado);
                }
            }
        }

        public static (string clientId, string token) DescriptografarCredenciais(
            string clientIdCifrado,
            string tokenCifrado,
            string chaveSessaoCifrada,
            string caminhoChavePrivada)
        {
            Console.WriteLine("\n=========================================");
            Console.WriteLine("    Processo de Descriptografia           ");
            Console.WriteLine("=========================================");

            byte[] chaveSessaoDecifrada = DescriptografarRsa(caminhoChavePrivada, chaveSessaoCifrada);

            string clientIdDecifrado = DescriptografarAes(chaveSessaoDecifrada, clientIdCifrado);
            Console.WriteLine($"\nClient ID decifrado com a chave de sessao AES:\n[ {Mascarar(clientIdDecifrado)} ]");

            string tokenDecifrado = DescriptografarAes(chaveSessaoDecifrada, tokenCifrado);
            Console.WriteLine($"\nToken decifrado com a chave de sessao AES:\n[ {Mascarar(tokenDecifrado)} ]");

            return (clientIdDecifrado, tokenDecifrado);
        }

        public static void GerarCSR(
            string clientId,
            string organizacao,
            string cidade,
            string estado,
            string pais,
            string caminhoCSR = "ARQUIVO_REQUEST_CERTIFICADO.csr",
            string caminhoChavePrivadaCSR = "ARQUIVO_CHAVE_PRIVADA.key")
        {
            using (RSA rsaCSR = RSA.Create(2048))
            {
                string privateKeyPem = rsaCSR.ExportPkcs8PrivateKeyPem();
                File.WriteAllText(caminhoChavePrivadaCSR, privateKeyPem + Environment.NewLine);
                Console.WriteLine($"\nChave privada do CSR salva em {Path.GetFullPath(caminhoChavePrivadaCSR)}");

                string subjectName = $"CN={clientId}, OU={organizacao}, L={cidade}, ST={estado}, C={pais}";

                CertificateRequest csr = new CertificateRequest(
                    new X500DistinguishedName(subjectName),
                    rsaCSR,
                    HashAlgorithmName.SHA512,
                    RSASignaturePadding.Pkcs1);

                byte[] csrDer = csr.CreateSigningRequest();
                string csrBase64 = Convert.ToBase64String(csrDer, Base64FormattingOptions.InsertLineBreaks);
                string csrPem = $"-----BEGIN CERTIFICATE REQUEST-----\n{csrBase64}\n-----END CERTIFICATE REQUEST-----\n

                File.WriteAllText(caminhoCSR, csrPem);
                Console.WriteLine($"CSR salvo em {Path.GetFullPath(caminhoCSR)}");
            }
        }

        public static string EnviarCertificado(
            string token,
            string caminhoCSR = "ARQUIVO_REQUEST_CERTIFICADO.csr",
            string caminhoCRT = "certificado.crt")
        {
            string conteudoCSR = File.ReadAllText(caminhoCSR);

            using var client = new HttpClient();
            var requestMsg = new HttpRequestMessage(HttpMethod.Post, "https://sts.itau.com.br/seguranca/v1/certificado/
            requestMsg.Content = new StringContent(conteudoCSR, Encoding.ASCII, "text/plain");
            requestMsg.Headers.Accept.ParseAdd("*/*");
            requestMsg.Headers.AcceptEncoding.ParseAdd("gzip");
            requestMsg.Headers.AcceptEncoding.ParseAdd("deflate");
            requestMsg.Headers.AcceptEncoding.ParseAdd("br");
            requestMsg.Headers.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", token);

            string responseSecret = "";

            Log("INFO", "=== DETALHES DA REQUISICAO HTTP (Enviar CSR) ===");
            Log("INFO", $"URL: https://sts.itau.com.br/seguranca/v1/certificado/solicitacao");
            Log("INFO", "Metodo: POST");
            Log("INFO", $"Headers: Content-Type=text/plain, Accept=*/*, Accept-Encoding=gzip deflate br, Authorization=
            string csrPreview = conteudoCSR.Length > 200 ? conteudoCSR.Substring(0, 200) + "..." : conteudoCSR;
            Log("INFO", $"Body (CSR): {csrPreview}");
            Log("INFO", $"Tamanho do body: {conteudoCSR.Length} bytes");
            try
            {
                var httpResponse = client.SendAsync(requestMsg).Result;
                string response = httpResponse.Content.ReadAsStringAsync().Result;
```

```
            Log("INFO", "=== DETALHES DA RESPOSTA HTTP (Enviar CSR) ===");
            Log("INFO", $"Status code: {(int)httpResponse.StatusCode}");
            string respHeaders = "";
            foreach (var header in httpResponse.Headers)
                respHeaders += $"{header.Key}={string.Join(", ", header.Value)}; ";
            Log("INFO", $"Response headers: {respHeaders}");
            string corpoResp = response.Length > 500 ? response.Substring(0, 500) : response;
            Log("INFO", $"Response body (primeiros 500 chars): {corpoResp}");

            if (!httpResponse.IsSuccessStatusCode)
            {
                string detalhe = DetalharErroHttp((int)httpResponse.StatusCode);
                throw new Exception($"Erro na geracao do certificado. Status: {(int)httpResponse.StatusCode}\n  Pos
            }

            StreamWriter fileCertificate = File.CreateText(caminhoCRT);
            var linhas = response.Split(Environment.NewLine.ToCharArray());

            var uuidPattern = new Regex(@"[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{1
            foreach (var item in linhas)
            {
                if (item.Contains("Secret:"))
                {
                    var match = uuidPattern.Match(item);
                    responseSecret = match.Success ? match.Value : item;
                }
                else
                {
                    fileCertificate.WriteLine(item);
                }
            }

            fileCertificate.Close();
            Console.WriteLine($"\nCertificado salvo em {Path.GetFullPath(caminhoCRT)}");
            Console.WriteLine($"Client Secret: {Mascarar(responseSecret)}");

            return responseSecret;
        }
        catch (AggregateException aex) when (aex.InnerException is HttpRequestException hrex)
        {
            throw new Exception("Erro ao solicitar o certificado.\r\nDetalhe: " + hrex.Message);
        }
        catch (Exception ex) when (!(ex is AggregateException))
        {
            throw new Exception("Erro ao solicitar o certificado.\r\nDetalhe: " + ex.Message);
        }
    }

    public static string ObterTokenOAuth(string clientId, string clientSecret, string caminhoCRT, string caminhoKey
    {
        var certificate = CarregarCertificadoMtls(caminhoCRT, caminhoKey);

        var handler = new HttpClientHandler();
        handler.ClientCertificates.Add(certificate);
        handler.ServerCertificateCustomValidationCallback = (message, cert, chain, errors) => true;

        using var client = new HttpClient(handler);

        string body = $"grant_type=client_credentials&client_id={Uri.EscapeDataString(clientId)}&client_secret={Uri

        Log("INFO", "=== DETALHES DA REQUISICAO HTTP (Token OAuth) ===");
        Log("INFO", "URL: https://sts.itau.com.br/api/oauth/token");
        Log("INFO", "Metodo: POST");
        Log("INFO", "Headers: Content-Type=application/x-www-form-urlencoded");
        Log("INFO", $"Certificado mTLS: {caminhoCRT}");
        Log("INFO", $"Chave mTLS: {caminhoKey}");
        Log("INFO", $"Body: grant_type=client_credentials&client_id={Mascarar(clientId)}&client_secret={Mascarar(cl

        var content = new StringContent(body, Encoding.UTF8, "application/x-www-form-urlencoded");
        var response = client.PostAsync("https://sts.itau.com.br/api/oauth/token", content).Result;

        string responseBody = response.Content.ReadAsStringAsync().Result;

        Log("INFO", "=== DETALHES DA RESPOSTA HTTP (Token OAuth) ===");
        Log("INFO", $"Status code: {(int)response.StatusCode}");
        string respHeaders6 = "";
        foreach (var header in response.Headers)
            respHeaders6 += $"{header.Key}={string.Join(", ", header.Value)}; ";
        Log("INFO", $"Response headers: {respHeaders6}");
        string corpoResp6 = responseBody.Length > 500 ? responseBody.Substring(0, 500) : responseBody;
        Log("INFO", $"Response body (primeiros 500 chars): {corpoResp6}");

        if (!response.IsSuccessStatusCode)
        {
            string detalhe = DetalharErroHttp((int)response.StatusCode);
            throw new Exception($"Erro ao obter token OAuth. Status: {(int)response.StatusCode}\n  Possivel causa:
        }

        return responseBody;
    }
```

```
    public static string RenovarCertificado(
        string tokenSTS,
        string caminhoCSR = "ARQUIVO_REQUEST_CERTIFICADO.csr",
        string caminhoCRT = "certificado.crt",
        string caminhoCertMtls = "",
        string caminhoKeyMtls = "")
    {
        string conteudoCSR = File.ReadAllText(caminhoCSR);

        HttpClientHandler handler = new HttpClientHandler();
        handler.ServerCertificateCustomValidationCallback = (message, cert, chain, errors) => true;
        if (!string.IsNullOrEmpty(caminhoCertMtls) && !string.IsNullOrEmpty(caminhoKeyMtls))
        {
            var clientCert = CarregarCertificadoMtls(caminhoCertMtls, caminhoKeyMtls);
            handler.ClientCertificates.Add(clientCert);
            Log("INFO", $"mTLS cert: {caminhoCertMtls}");
            Log("INFO", $"mTLS key: {caminhoKeyMtls}");
        }
        using var client = new HttpClient(handler);
        var requestMsg = new HttpRequestMessage(HttpMethod.Post, "https://sts.itau.com.br/seguranca/v2/certificado/
        requestMsg.Content = new StringContent(conteudoCSR, Encoding.ASCII, "text/plain");
        string correlationId = Guid.NewGuid().ToString();
        requestMsg.Headers.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", tokenSTS
        requestMsg.Headers.Add("x-itau-correlationID", correlationId);

        Log("INFO", "=== DETALHES DA REQUISICAO HTTP (Renovacao) ===");
        Log("INFO", $"URL: https://sts.itau.com.br/seguranca/v2/certificado/renovacao");
        Log("INFO", "Metodo: POST");
        Log("INFO", $"Headers: Content-Type=text/plain, Authorization=Bearer {Mascarar(tokenSTS)}, x-itau-correlati
        string csrPreview5 = conteudoCSR.Length > 200 ? conteudoCSR.Substring(0, 200) + "..." : conteudoCSR;
        Log("INFO", $"Body (CSR): {csrPreview5}");
        Log("INFO", $"Tamanho do body: {conteudoCSR.Length} bytes");
        try
        {
            var httpResponse = client.SendAsync(requestMsg).Result;
            string response = httpResponse.Content.ReadAsStringAsync().Result;

            Log("INFO", "=== DETALHES DA RESPOSTA HTTP (Renovacao) ===");
            Log("INFO", $"Status code: {(int)httpResponse.StatusCode}");
            string respHeaders5 = "";
            foreach (var header in httpResponse.Headers)
                respHeaders5 += $"{header.Key}={string.Join(", ", header.Value)}; ";
            Log("INFO", $"Response headers: {respHeaders5}");
            string corpoResp5 = response.Length > 500 ? response.Substring(0, 500) : response;
            Log("INFO", $"Response body (primeiros 500 chars): {corpoResp5}");

            if (!httpResponse.IsSuccessStatusCode)
            {
                string detalhe = DetalharErroHttp((int)httpResponse.StatusCode);
                throw new Exception($"Erro na renovacao do certificado. Status: {(int)httpResponse.StatusCode}\n  P
            }

            File.WriteAllText(caminhoCRT, response);
            Console.WriteLine($"\nCertificado renovado salvo em {Path.GetFullPath(caminhoCRT)}");

            return response;
        }
        catch (AggregateException aex) when (aex.InnerException is HttpRequestException hrex)
        {
            throw new Exception("Erro ao renovar o certificado.\r\nDetalhe: " + hrex.Message);
        }
        catch (Exception ex) when (!(ex is AggregateException))
        {
            throw new Exception("Erro ao renovar o certificado.\r\nDetalhe: " + ex.Message);
        }
    }

    // Caminho do arquivo fonte em tempo de compilacao (equivalente a __file__ em Python,
    // __dirname em Node.js e runtime.Caller(0) em Go). Necessario porque
    // AppDomain.CurrentDomain.BaseDirectory aponta para o diretorio do binario
    // (csharp/bin/Debug/netX.X/) e nao para a pasta csharp/ onde o codigo esta.
    private static string GetThisFilePath([CallerFilePath] string path = "") => path;

    // sobe de csharp/ para certificado-dinamico/
    private static readonly string _dirRaiz = ResolverDirRaiz();
    private static readonly string PastaSaida = Path.Combine(_dirRaiz, "output");
    private static readonly string PastaEtapa1 = Path.Combine(PastaSaida, "etapa1");
    private static readonly string PastaEtapa2 = Path.Combine(PastaSaida, "etapa2");
    private static readonly string PastaEtapa4 = Path.Combine(PastaSaida, "etapa4");
    private static readonly string PastaLogs = Path.Combine(PastaSaida, "logs");
    private static readonly string ArquivoEstado = Path.Combine(PastaSaida, "estado_certificado.json");

    private static StreamWriter logWriter = null;
```

```
    private static string ResolverDirRaiz()
    {
        string thisFile = GetThisFilePath();
        string thisDir = Path.GetDirectoryName(thisFile);
        if (!string.IsNullOrEmpty(thisDir) && Directory.Exists(thisDir))
        {
            return Path.GetFullPath(Path.Combine(thisDir, ".."));
        }
        // Fallback caso o caminho compilado nao exista mais (ex: binarios copiados para outra maquina).
        // Assume a estrutura padrao csharp/bin/Debug/netX.X/ e sobe 4 niveis ate certificado-dinamico/.
        return Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "..", "..", "..", ".."));
    }

    private static string DetalharErroHttp(int statusCode)
    {
        switch (statusCode)
        {
            case 400: return "Requisicao invalida. O CSR pode estar malformado ou os dados enviados estao incorreto
            case 401: return "Nao autorizado. O token pode estar invalido, expirado ou nao foi informado corretamen
            case 403: return "Acesso negado. O token nao tem permissao para esta operacao ou esta expirado. Tente e
            case 404: return "Endpoint nao encontrado. Verifique se a URL do STS Itau esta correta.";
            case 500: return "Erro interno do servidor STS Itau. Tente novamente mais tarde.";
            case 502: return "Bad Gateway. O servidor STS Itau esta temporariamente indisponivel. Tente novamente m
            case 503: return "Servico indisponivel. O servidor STS Itau esta em manutencao. Tente novamente mais ta
            default: return $"Erro HTTP {statusCode}. Verifique a documentacao do STS Itau.";
        }
    }

    private static string Mascarar(string valor)
    {
        if (string.IsNullOrWhiteSpace(valor)) return "******";
        valor = valor.Trim();
        if (valor.Length <= 6) return "******";
        return valor.Substring(0, 3) + "******" + valor.Substring(valor.Length - 3);
    }

    // Formata uma excecao incluindo a cadeia de InnerException, util para diagnosticar
    // erros como AggregateException onde a causa real fica em InnerException.
    private static string FormatarExcecao(Exception ex)
    {
        var sb = new StringBuilder();
        var atual = ex;
        int nivel = 0;
        while (atual != null && nivel < 5)
        {
            if (nivel > 0) sb.Append(" --> ");
            sb.Append(atual.GetType().Name).Append(": ").Append(atual.Message);
            atual = (atual is AggregateException ae && ae.InnerExceptions.Count > 0)
                ? ae.InnerExceptions[0]
                : atual.InnerException;
            nivel++;
        }
        return sb.ToString();
    }

    private static string ConfigurarLog()
    {
        Directory.CreateDirectory(PastaLogs);
        string timestamp = DateTime.Now.ToString("yyyyMMdd_HHmmss");
        string arquivoLog = Path.Combine(PastaLogs, $"certificado_dinamico_{timestamp}.log");
        logWriter = new StreamWriter(arquivoLog, true, Encoding.UTF8) { AutoFlush = true };
        return arquivoLog;
    }

    private static void Log(string nivel, string mensagem)
    {
        if (logWriter == null) return;
        string ts = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
        logWriter.WriteLine($"{ts} [{nivel}] {mensagem}");
    }

    private static readonly Dictionary<int, string> EtapasDesc = new Dictionary<int, string>
    {
        { 1, "Gerar par de chaves e exibir chave publica" },
        { 2, "Descriptografar credenciais, gerar e enviar certificado" },
        { 3, "Testar geracao de token" }
    };

    // Chaves cujo valor deve ser serializado como numero JSON (sem aspas).
    // Demais chaves sao tratadas como strings.
    private static readonly HashSet<string> ChavesNumericasEstado = new HashSet<string>
    {
        "etapaConcluida"
    };

    private static Dictionary<string, string> CarregarEstado()
    {
        var estado = new Dictionary<string, string> { { "etapaConcluida", "0" } };
        foreach (string pasta in new[] { PastaSaida, PastaEtapa1, PastaEtapa2, PastaEtapa4, PastaLogs })
            Directory.CreateDirectory(pasta);
        if (File.Exists(ArquivoEstado))
        {
```

```
                try
                {
                    string json = File.ReadAllText(ArquivoEstado);
                    var parsed = JsonSerializer.Deserialize<Dictionary<string, JsonElement>>(json);
                    if (parsed != null)
                    {
                        estado.Clear();
                        foreach (var kv in parsed)
                        {
                            estado[kv.Key] = kv.Value.ValueKind == JsonValueKind.String
                                ? kv.Value.GetString()
                                : kv.Value.GetRawText();
                        }
                    }
                }
                catch {  }
            }
            return estado;
        }

        private static void SalvarEstado(Dictionary<string, string> estado)
        {
            var sb = new StringBuilder();
            sb.Append("{\n");
            int i = 0;
            int total = estado.Count;
            foreach (var kv in estado)
            {
                string chave = kv.Key;
                string valor = kv.Value ?? string.Empty;
                sb.Append("  ").Append(JsonSerializer.Serialize(chave)).Append(": ");
                if (ChavesNumericasEstado.Contains(chave) && long.TryParse(valor, out long n))
                {
                    sb.Append(n);
                }
                else
                {
                    sb.Append(JsonSerializer.Serialize(valor));
                }
                if (i < total - 1) sb.Append(",");
                sb.Append("\n");
                i++;
            }
            sb.Append("}");
            File.WriteAllText(ArquivoEstado, sb.ToString());
        }

        private static int GetEtapaConcluida(Dictionary<string, string> estado)
        {
            if (estado.TryGetValue("etapaConcluida", out string val) && int.TryParse(val, out int n))
                return n;
            return 0;
        }

        private static string GetEstadoStr(Dictionary<string, string> estado, string chave, string padrao)
        {
            return estado.TryGetValue(chave, out string val) ? val : padrao;
        }

        private static string StatusEtapa(int n, int etapaConcluida)
        {
            if (n <= etapaConcluida) return "[CONCLUIDA]";
            if (n == etapaConcluida + 1) return "[PROXIMA >>] <------ voce esta aqui";
            return "[PENDENTE]";
        }

        private static int ExibirMenu(Dictionary<string, string> estado)
        {
            int etapaConcluida = GetEtapaConcluida(estado);

            Console.WriteLine("\n=============================================");
            Console.WriteLine("  Certificado Dinamico Itau - Menu Principal ");
            Console.WriteLine("=============================================");
            Console.WriteLine($"\n  Progresso atual: Etapa {etapaConcluida} de 3 concluida(s)\n");

            Console.WriteLine($"  1. {EtapasDesc[1]} {StatusEtapa(1, etapaConcluida)}");

            Console.WriteLine($"  2. {EtapasDesc[2]} {StatusEtapa(2, etapaConcluida)}");

            Console.WriteLine($"  3. {EtapasDesc[3]} {StatusEtapa(3, etapaConcluida)}");

            Console.WriteLine("  4. Renovar certificado");

            Console.WriteLine("\n  0. Sair");
            Console.WriteLine("  9. Recomecar do zero (limpa progresso)");

            return etapaConcluida + 1 <= 3 ? etapaConcluida + 1 : 3;
        }

        static void Main(string[] args)
        {
            string arquivoLog = ConfigurarLog();
            Log("INFO", "Programa iniciado");
            Console.WriteLine($"  Log desta execucao: {arquivoLog}");
```

```
        var estado = CarregarEstado();
        string clientSecretMemoria = "";
        while (true)
        {
            int proxima = ExibirMenu(estado);

            Console.Write($"\nEscolha a etapa (padrao: {proxima}): ");
            string escolhaStr = Console.ReadLine()?.Trim() ?? "";
            int escolha;
            if (string.IsNullOrEmpty(escolhaStr))
            {
                escolha = proxima;
            }
            else if (!int.TryParse(escolhaStr, out escolha))
            {
                Console.WriteLine("Opcao invalida.");
                continue;
            }

            try
            {
                Log("INFO", $"Opcao escolhida pelo usuario: {escolha}");

                if (escolha == 0)
                {
                    Log("INFO", "Programa encerrado pelo usuario");
                    Console.WriteLine("\nSaindo. Seu progresso foi salvo.");
                    break;
                }
                else if (escolha == 9)
                {
                    Log("INFO", "Limpeza de progresso e arquivos gerados iniciada");
                    foreach (string pasta in new[] { PastaEtapa1, PastaEtapa2, PastaEtapa4 })
                    {
                        if (Directory.Exists(pasta))
                        {
                            foreach (string arquivo in Directory.GetFiles(pasta))
                            {
                                File.Delete(arquivo);
                            }
                            Console.WriteLine($"  Arquivos removidos da pasta: {Path.GetFullPath(pasta)}");
                            Log("INFO", $"Arquivos removidos da pasta: {Path.GetFullPath(pasta)}");
                        }
                    }
                    if (File.Exists(ArquivoEstado))
                    {
                        File.Delete(ArquivoEstado);
                        Console.WriteLine($"  Removido: {Path.GetFullPath(ArquivoEstado)}");
                        Log("INFO", $"Removido: {Path.GetFullPath(ArquivoEstado)}");
                    }
                    Console.WriteLine("\nProgresso e arquivos gerados removidos.");
                    Log("INFO", "Limpeza concluida");
                    estado = new Dictionary<string, string>();
                    clientSecretMemoria = "";
                    continue;
                }
                else if (escolha == 1)
                {
                    Log("INFO", "Inicio da Etapa 1: Gerar par de chaves e exibir chave publica");
                    Directory.CreateDirectory(PastaEtapa1);
                    Console.WriteLine("\n[PASSO 1.1] Gerando par de chaves RSA 2048...");
                    string caminhoPrivada = Path.Combine(PastaEtapa1, "private.pem");
                    string caminhoPublica = Path.Combine(PastaEtapa1, "public.pem");
                    GerarParDeChaves(caminhoPrivada, caminhoPublica);

                    Log("INFO", $"Chave privada gerada: {Path.GetFullPath(caminhoPrivada)}");
                    Log("INFO", $"Chave publica gerada: {Path.GetFullPath(caminhoPublica)}");

                    Console.WriteLine("\n[PASSO 1.2] Exibindo chave publica...");
                    string conteudoChavePublica = File.ReadAllText(caminhoPublica);
                    Console.WriteLine("\n=============================================");
                    Console.WriteLine("  Chave Publica Gerada                       ");
                    Console.WriteLine("=============================================");
                    Console.WriteLine(conteudoChavePublica);

                    estado["etapaConcluida"] = Math.Max(GetEtapaConcluida(estado), 1).ToString();
                    estado["caminhoChavePrivada"] = Path.GetFullPath(caminhoPrivada);
                    estado["caminhoChavePublica"] = Path.GetFullPath(caminhoPublica);
                    SalvarEstado(estado);
                    Log("INFO", "Etapa 1 concluida com sucesso");
                    Console.WriteLine("\n>> Etapa 1 concluida. Progresso salvo.");
                    Console.WriteLine("\n  O que fazer agora:");
                    Console.WriteLine("  1. Copie o conteudo da chave publica exibido acima");
                    Console.WriteLine("  2. Envie a chave publica para o Analista de operacoes Itau");
                    Console.WriteLine("  3. Aguarde o e-mail do Itau com as credenciais cifradas");
                    Console.WriteLine("  4. Quando receber o e-mail, execute a etapa 2");
                }
                else if (escolha == 2)
                {
                    if (GetEtapaConcluida(estado) < 1)
                    {
                        Console.WriteLine("\n[AVISO] A etapa 1 (gerar chaves) precisa ser concluida antes.");
                        Log("WARN", "Tentativa de executar Etapa 2 sem concluir Etapa 1");
                        continue;
                    }
```

```
Log("INFO", "Inicio da Etapa 2: Descriptografar credenciais, gerar e enviar certificado");
Directory.CreateDirectory(PastaEtapa2);

Console.WriteLine("\n[PASSO 2.1] Descriptografar credenciais recebidas por e-mail...");
string caminhoChavePrivada = Path.GetFullPath(GetEstadoStr(estado, "caminhoChavePrivada", "priv
Console.WriteLine($"  Chave privada: {caminhoChavePrivada}");
Log("INFO", $"Chave privada utilizada: {Path.GetFullPath(caminhoChavePrivada)}");

Console.Write("Client ID cifrado: ");
string clientIdCifrado = Console.ReadLine().Trim();
Console.Write("Token temporario cifrado: ");
string tokenCifrado = Console.ReadLine().Trim();
Console.Write("Chave de sessao cifrada: ");
string chaveSessaoCifrada = Console.ReadLine().Trim();
Log("INFO", $"Client ID cifrado: {Mascarar(clientIdCifrado)}");
Log("INFO", $"Token cifrado: {Mascarar(tokenCifrado)}");
Log("INFO", $"Chave de sessao cifrada: {Mascarar(chaveSessaoCifrada)}");

var (clientId, tokenDecifrado) = DescriptografarCredenciais(
    clientIdCifrado, tokenCifrado, chaveSessaoCifrada, caminhoChavePrivada);

string caminhoClientId = Path.Combine(PastaEtapa2, "client_id.txt");
File.WriteAllText(caminhoClientId, clientId);
Console.WriteLine($"  Client ID salvo em {Path.GetFullPath(caminhoClientId)}");
Log("INFO", $"Client ID decifrado: {Mascarar(clientId)}");

string caminhoToken = Path.Combine(PastaEtapa2, "token_temporario.txt");
File.WriteAllText(caminhoToken, tokenDecifrado);
Console.WriteLine($"  Token temporario salvo em {Path.GetFullPath(caminhoToken)}");
Log("INFO", $"Token decifrado: {Mascarar(tokenDecifrado)}");

Console.WriteLine("\n[PASSO 2.2] Gerando CSR (Certificate Sign Request)...");
Console.WriteLine($"  Client ID: {Mascarar(clientId)}");

string organizacao = "";
while (string.IsNullOrEmpty(organizacao))
{
    Console.Write("Nome da Empresa: ");
    organizacao = Console.ReadLine().Trim();
    if (string.IsNullOrEmpty(organizacao)) Console.WriteLine("  Campo obrigatorio. Informe o no
}
string cidade = "";
while (string.IsNullOrEmpty(cidade))
{
    Console.Write("Cidade: ");
    cidade = Console.ReadLine().Trim();
    if (string.IsNullOrEmpty(cidade)) Console.WriteLine("  Campo obrigatorio. Informe a cidade.
}
string estadoUF = "";
while (string.IsNullOrEmpty(estadoUF))
{
    Console.Write("Estado - UF: ");
    estadoUF = Console.ReadLine().Trim();
    if (string.IsNullOrEmpty(estadoUF)) Console.WriteLine("  Campo obrigatorio. Informe o estad
}
string pais = "";
while (string.IsNullOrEmpty(pais))
{
    Console.Write("Pais: ");
    pais = Console.ReadLine().Trim();
    if (string.IsNullOrEmpty(pais)) Console.WriteLine("  Campo obrigatorio. Informe o pais.");
}
Log("INFO", $"Client ID: {Mascarar(clientId)}");
Log("INFO", $"Organizacao: {organizacao}, Cidade: {cidade}, Estado: {estadoUF}, Pais: {pais}");

string caminhoCSR3 = Path.Combine(PastaEtapa2, "ARQUIVO_REQUEST_CERTIFICADO.csr");
string caminhoChaveCSR = Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIVADA.key");
GerarCSR(clientId, organizacao, cidade, estadoUF, pais, caminhoCSR3, caminhoChaveCSR);

Console.WriteLine("\n[PASSO 2.3] Enviando CSR ao STS Itau...");
Log("INFO", $"Token: {Mascarar(tokenDecifrado)}");
Log("INFO", $"CSR: {Path.GetFullPath(caminhoCSR3)}");
Log("INFO", "Enviando requisicao para https://sts.itau.com.br/seguranca/v1/certificado/solicita

string caminhoCRT = Path.Combine(PastaEtapa2, "certificado.crt");
string secret = EnviarCertificado(tokenDecifrado, caminhoCSR3, caminhoCRT);
clientSecretMemoria = secret;

Console.WriteLine("\n============================================");
Console.WriteLine("  Certificado gerado com sucesso!            ");
Console.WriteLine("============================================");
Console.WriteLine("Arquivos gerados:");
Console.WriteLine($"  - {Path.GetFullPath(Path.Combine(PastaEtapa1, "private.pem"))} (chave pri
Console.WriteLine($"  - {Path.GetFullPath(Path.Combine(PastaEtapa1, "public.pem"))} (chave publ
Console.WriteLine($"  - {Path.GetFullPath(Path.Combine(PastaEtapa2, "client_id.txt"))} (Client
Console.WriteLine($"  - {Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIVADA.key"
```

```
    Console.WriteLine($"  - {Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_REQUEST_CERTIFICAD
    Console.WriteLine($"  - {Path.GetFullPath(Path.Combine(PastaEtapa2, "certificado.crt"))} (certi
    Console.WriteLine("\n================================================");
    Console.WriteLine("  ATENCAO - CLIENT SECRET                       ");
    Console.WriteLine("================================================");
    Console.WriteLine("\nO Client Secret e equivalente a uma SENHA. Trate-o com o mesmo cuidado.");
    Console.WriteLine("O valor do Client Secret sera exibido APENAS NESTE MOMENTO.");
    Console.WriteLine("Copie e salve em um local seguro. Este valor NAO sera armazenado pelo progra
    Console.WriteLine($"\n  Client Secret: {secret}");
    Console.WriteLine("\n================================================");
    Console.WriteLine("Guarde essas informacoes em local seguro. O Client ID e o Client Secret sao
    Console.WriteLine("e serao necessarios para gerar tokens OAuth e consumir as APIs do Itau.");
    Console.WriteLine("NAO compartilhe essas credenciais. Quem tiver acesso a elas podera acessar a
    Console.WriteLine("\nO Itau nao se responsabiliza pelo armazenamento local das credenciais.");
    Console.WriteLine("A guarda correta e de responsabilidade exclusiva do cliente.");

    var expiracao = DateTime.Now.AddDays(365);
    var inicioRenovacao = expiracao.AddDays(-30);
    var fimRenovacao = expiracao.AddDays(-1);
    Console.WriteLine("\n=================================================");
    Console.WriteLine("  VALIDADE DO CERTIFICADO                        ");
    Console.WriteLine("=================================================");
    Console.WriteLine("  Validade: 365 dias");
    Console.WriteLine($"  Data de expiracao: {expiracao:dd/MM/yyyy}");
    Console.WriteLine($"  Periodo de renovacao: de {inicioRenovacao:dd/MM/yyyy} ate {fimRenovacao:d
    Console.WriteLine("\n  O processo de renovacao (etapa 4) pode ser realizado a partir de");
    Console.WriteLine("  30 dias antes da expiracao ate um dia antes da data de expiracao do certif
    Console.WriteLine("\n  ATENCAO: A responsabilidade de renovar o certificado dentro do prazo");
    Console.WriteLine("  e exclusivamente sua. Certificados expirados impossibilitam o acesso");
    Console.WriteLine("  as APIs do Itau e exigem a geracao de um novo certificado do zero.");

    estado["etapaConcluida"] = Math.Max(GetEtapaConcluida(estado), 2).ToString();
    estado["caminhoClientId"] = Path.GetFullPath(caminhoClientId);
    estado["caminhoChavePrivada"] = caminhoChavePrivada;
    estado["caminhoCSR"] = Path.GetFullPath(caminhoCSR3);
    estado["caminhoCRT"] = Path.GetFullPath(caminhoCRT);
    SalvarEstado(estado);
    Log("INFO", $"Certificado salvo: {Path.GetFullPath(caminhoCRT)}");
    Log("INFO", $"Client Secret: {Mascarar(secret)}");
    Log("INFO", "Etapa 2 concluida com sucesso");
    Console.WriteLine("\n>> Etapa 2 concluida. Progresso salvo.");
    Console.WriteLine("\n  O que fazer agora:");
    Console.WriteLine("  1. Guarde o Client ID e o Client Secret em local seguro - eles sao equival
    Console.WriteLine($"  2. O certificado (.crt) e a chave privada (.key) estao salvos na pasta {P
    Console.WriteLine("  3. Execute a etapa 3 para validar que tudo esta funcionando corretamente")
    Console.WriteLine("  4. Apos validar, utilize o client_id, client_secret, certificado e chave p
    Console.WriteLine("\n  LEMBRE-SE: Client ID, Client Secret, certificado e chave privada sao equ
    Console.WriteLine("  NAO compartilhe, NAO envie por e-mail e NAO deixe exposto em codigo-fonte.
    Console.WriteLine("  O Itau NAO se responsabiliza pelo armazenamento local. A guarda e responsa

    Console.Write("\nDeseja gerar o arquivo PFX (PKCS#12)? O PFX combina o certificado e a chave pr
    string respPfx2 = Console.ReadLine().Trim();
    if (respPfx2.Equals("s", StringComparison.OrdinalIgnoreCase))
    {
        Console.Write("  Informe a senha para proteger o PFX: ");
        string senhaPfx2 = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(senhaPfx2))
        {
            Console.WriteLine("  Senha nao informada. PFX nao gerado.");
            Log("WARN", "PFX nao gerado - senha nao informada");
        }
        else
        {
            string caminhoPfx2 = Path.Combine(PastaEtapa2, "certificado.pfx");
            string caminhoKeyPfx2 = Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIVA
            var certPfx = X509Certificate2.CreateFromPemFile(caminhoCRT, caminhoKeyPfx2);
            File.WriteAllBytes(caminhoPfx2, certPfx.Export(X509ContentType.Pfx, senhaPfx2));
            Console.WriteLine($"  PFX gerado: {Path.GetFullPath(caminhoPfx2)}");
            Log("INFO", $"PFX gerado: {Path.GetFullPath(caminhoPfx2)}");
        }
    }
}
else if (escolha == 3)
{
    if (GetEtapaConcluida(estado) < 2)
    {
        Console.WriteLine("\n[AVISO] A etapa 2 (gerar e enviar certificado) precisa ser concluida a
        Log("WARN", "Tentativa de executar Etapa 3 sem concluir Etapa 2");
        continue;
    }
                        Log("INFO", "Inicio da Etapa 3: Testar geracao de token");
                        Console.WriteLine("\n[PASSO 3] Testando geracao de token...");
    string caminhoClientId7 = GetEstadoStr(estado, "caminhoClientId", Path.Combine(PastaEtapa2, "cl
    if (!File.Exists(caminhoClientId7))
    {
        throw new Exception($"Arquivo de client_id nao encontrado: {caminhoClientId7}. Execute as e
    }
    string clientIdOAuth = File.ReadAllText(caminhoClientId7).Trim();
    string clientSecretOAuth;
    if (!string.IsNullOrEmpty(clientSecretMemoria))
    {
        clientSecretOAuth = clientSecretMemoria;
        Console.WriteLine($"  Client Secret: {Mascarar(clientSecretOAuth)} (obtido da sessao atual)
    }
    else
```

```
                    {
                        Console.Write("  Informe o Client Secret: ");
                        clientSecretOAuth = Console.ReadLine().Trim();
                    }
                    string caminhoCRTOAuth = GetEstadoStr(estado, "caminhoCRT", Path.Combine(PastaEtapa2, "certific
                    string caminhoKeyOAuth = Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIVADA.key"

                    Console.WriteLine("\n  Arquivos utilizados na requisicao:");
                    Console.WriteLine($"  - {Path.GetFullPath(caminhoClientId7)} (Client ID: {Mascarar(clientIdOAut
                    Console.WriteLine($"  - {Path.GetFullPath(caminhoCRTOAuth)} (certificado assinado)");
                    Console.WriteLine($"  - {Path.GetFullPath(caminhoKeyOAuth)} (chave privada do certificado)");

                    string respostaOAuth = ObterTokenOAuth(clientIdOAuth, clientSecretOAuth, caminhoCRTOAuth, camin
                    Log("INFO", "Token OAuth obtido com sucesso");
                    Log("INFO", "Etapa 3 concluida com sucesso");

                    string accessToken = "";
                    try
                    {
                        var jsonDoc = System.Text.Json.JsonDocument.Parse(respostaOAuth);
                        if (jsonDoc.RootElement.TryGetProperty("access_token", out var atProp))
                            accessToken = atProp.GetString() ?? "";
                    }
                    catch {}

                    Console.WriteLine("\n==============================================");
                    Console.WriteLine("  Credenciais e certificado VALIDADOS!        ");
                    Console.WriteLine("==============================================");
                    Console.WriteLine("\nAccess Token:");
                    Console.WriteLine(string.IsNullOrEmpty(accessToken) ? respostaOAuth : accessToken);
                    Console.WriteLine("\n  Validade: 300 segundos (5 minutos)");
                    Console.WriteLine("  Uso: Informe este token no header 'Authorization: Bearer <access_token>' d
                    Console.WriteLine("\nAs credenciais (client_id e client_secret) e o certificado digital estao v
                    Console.WriteLine("Voce ja pode utiliza-los em suas integracoes.");
                    Console.WriteLine("\n>> Etapa 3 concluida.");
                    Console.WriteLine("\n  O que fazer agora:");
                    Console.WriteLine("  1. Suas credenciais e certificado estao validados e prontos para uso");
                    Console.WriteLine("  2. Na sua aplicacao, implemente o fluxo OAuth usando:");
                    Console.WriteLine("     - Client ID e Client Secret (da etapa 2)");
                    Console.WriteLine($"     - Certificado .crt e chave privada .key (da pasta {Path.GetFullPath(Pa
                    Console.WriteLine("  3. O token gerado tem validade de 5 minutos - sua aplicacao deve renova-lo
                    Console.WriteLine("  4. Lembre-se de renovar o certificado antes da data de expiracao (etapa 4)
                }
                else if (escolha == 4)
                {
                    Log("INFO", "Inicio da Etapa 4: Renovar certificado");
                    Directory.CreateDirectory(PastaEtapa4);

                    string caminhoCRTValidacao = GetEstadoStr(estado, "caminhoCRT", Path.Combine(PastaEtapa2, "cert
                    if (File.Exists(caminhoCRTValidacao))
                    {
                        try
                        {
                            var certObj = new X509Certificate2(caminhoCRTValidacao);
                            DateTime dataExpiracao = certObj.NotAfter;
                            int diasRestantes = (int)(dataExpiracao - DateTime.Now).TotalDays;
                            Log("INFO", $"Certificado encontrado: {Path.GetFullPath(caminhoCRTValidacao)}");
                            Log("INFO", $"Data de expiracao: {dataExpiracao:dd/MM/yyyy} | Dias restantes: {diasRest
                            Console.WriteLine($"\n  Certificado encontrado: {Path.GetFullPath(caminhoCRTValidacao)}
                            Console.WriteLine($"  Data de expiracao: {dataExpiracao:dd/MM/yyyy} ({diasRestantes} di
                            if (diasRestantes > 30)
                            {
                                Console.WriteLine("\n  [AVISO] O certificado ainda nao esta no periodo de renovacao
                                Console.WriteLine("  A renovacao e permitida nos ultimos 30 dias antes da expiracao
                                Console.WriteLine($"  Faltam {diasRestantes - 30} dias para o inicio do periodo de 
                                Log("WARN", $"Certificado fora do periodo de renovacao. Dias restantes: {diasRestan
                                Console.Write("\n  Deseja prosseguir mesmo assim? (s/N): ");
                                string prosseguir = Console.ReadLine().Trim();
                                if (!prosseguir.Equals("s", StringComparison.OrdinalIgnoreCase))
                                {
                                    Console.WriteLine("  Renovacao cancelada pelo usuario.");
                                    Log("INFO", "Renovacao cancelada pelo usuario - fora do periodo");
                                    continue;
                                }
                                Log("INFO", "Usuario optou por prosseguir com a renovacao fora do periodo");
                            }
                            else if (diasRestantes < 0)
                            {
                                Console.WriteLine($"\n  [AVISO] O certificado ja esta EXPIRADO ha {Math.Abs(diasRes
                                Log("WARN", $"Certificado expirado ha {Math.Abs(diasRestantes)} dias");
                            }
                            else
                            {
                                Console.WriteLine("  Certificado dentro do periodo de renovacao.");
                            }
                        }
                        catch (Exception exCert)
                        {
                            Log("WARN", $"Nao foi possivel ler o certificado para validacao: {exCert.Message}");
                            Console.WriteLine("\n  [AVISO] Nao foi possivel ler o certificado para validar a data d
                            Console.WriteLine("  Prosseguindo com a renovacao...");
                        }
                    }
                    else
```

```
                    {
                        Log("INFO", "Certificado nao encontrado para validacao de periodo. Prosseguindo com a renov
                        Console.WriteLine($"\n  [INFO] Certificado nao encontrado em: {Path.GetFullPath(caminhoCRTV
                        Console.WriteLine("  Nao foi possivel validar o periodo de renovacao (30 dias antes da expi
                        Console.WriteLine("  Prosseguindo com a renovacao...");
                    }

                    Console.WriteLine("\n============================================");
                    Console.WriteLine("  Renovacao de Certificado - Escolha a opcao ");
                    Console.WriteLine("============================================");
                    Console.WriteLine("\n  A. Renovacao completa (gera novo CSR e nova chave privada) - recomendada
                    Console.WriteLine("     - Gera nova chave privada e novo CSR antes de renovar");
                    Console.WriteLine("     - Recomendado para maior seguranca ou quando a chave pode ter sido comp
                    Console.WriteLine("\n  B. Renovacao simples (reutiliza o CSR existente)");
                    Console.WriteLine("     - Mais rapido, mantem a mesma chave privada do certificado");
                    Console.WriteLine("     - Recomendado quando nao ha necessidade de trocar a chave");

                    Console.Write("\nEscolha a opcao (A ou B): ");
                    string opcaoRenov = Console.ReadLine().Trim().ToUpper();
                    if (opcaoRenov != "A" && opcaoRenov != "B")
                    {
                        Console.WriteLine("Opcao invalida. Escolha A ou B.");
                        continue;
                    }

                    // Funcao local para obter token
                    (string token, string certPath, string keyPath) ObterTokenRenovacao()
                    {
                        Console.WriteLine("\n============================================");
                        Console.WriteLine("  Como deseja obter o token para renovacao?  ");
                        Console.WriteLine("============================================");
                        Console.WriteLine("  1. Automatico (via mTLS com certificado atual)");
                        Console.WriteLine("  2. Manual (informar token manualmente)");
                        Console.Write("\nEscolha (1 ou 2): ");
                        string opcaoToken = Console.ReadLine().Trim();

                        if (opcaoToken == "1")
                        {
                            Log("INFO", "Obtencao de token automatica via mTLS");
                            Console.WriteLine("\n[PASSO 2.1] Obtendo token automaticamente via mTLS...");
                            string caminhoClientIdToken = GetEstadoStr(estado, "caminhoClientId", Path.Combine(Past
                            string clientIdToken;
                            if (File.Exists(caminhoClientIdToken))
                            {
                                clientIdToken = File.ReadAllText(caminhoClientIdToken).Trim();
                                Console.WriteLine($"  Client ID encontrado: {Mascarar(clientIdToken)}");
                            }
                            else if (estado.ContainsKey("clientId") && !string.IsNullOrEmpty(estado["clientId"]))
                            {
                                clientIdToken = estado["clientId"];
                                Console.WriteLine($"  Client ID recuperado do estado salvo: {Mascarar(clientIdToken
                            }
                            else
                            {
                                Console.WriteLine($"  [AVISO] Arquivo de Client ID nao encontrado em: {Path.GetFull
                                clientIdToken = "";
                                while (string.IsNullOrEmpty(clientIdToken))
                                {
                                    Console.Write("  Informe o Client ID: ");
                                    clientIdToken = Console.ReadLine().Trim();
                                    if (string.IsNullOrEmpty(clientIdToken)) Console.WriteLine("  Campo obrigatorio
                                }
                            }
                            string clientSecretToken;
                            if (!string.IsNullOrEmpty(clientSecretMemoria))
                            {
                                clientSecretToken = clientSecretMemoria;
                                Console.WriteLine($"  Client Secret: {Mascarar(clientSecretToken)} (obtido da sessa
                            }
                            else
                            {
                                clientSecretToken = "";
                                while (string.IsNullOrEmpty(clientSecretToken))
                                {
                                    Console.Write("  Informe o Client Secret: ");
                                    clientSecretToken = Console.ReadLine().Trim();
                                    if (string.IsNullOrEmpty(clientSecretToken)) Console.WriteLine("  Campo obrigat
                                }
                            }
                            string caminhoCRTToken = GetEstadoStr(estado, "caminhoCRT", Path.Combine(PastaEtapa2, "
                            if (!File.Exists(caminhoCRTToken))
                            {
                                Console.WriteLine($"  [AVISO] Certificado nao encontrado em: {Path.GetFullPath(cami
                                caminhoCRTToken = "";
```

```
            while (string.IsNullOrEmpty(caminhoCRTToken) || !File.Exists(caminhoCRTToken))
            {
                Console.Write("  Informe o caminho do certificado (.crt): ");
                caminhoCRTToken = Console.ReadLine().Trim();
                if (string.IsNullOrEmpty(caminhoCRTToken) || !File.Exists(caminhoCRTToken)) Con
            }
        }
        Console.WriteLine($"  Certificado: {Path.GetFullPath(caminhoCRTToken)}");
        string caminhoKeyToken = Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIV
        if (!File.Exists(caminhoKeyToken))
        {
            Console.WriteLine($"  [AVISO] Chave privada nao encontrada em: {caminhoKeyToken}");
            caminhoKeyToken = "";
            while (string.IsNullOrEmpty(caminhoKeyToken) || !File.Exists(caminhoKeyToken))
            {
                Console.Write("  Informe o caminho da chave privada (.key): ");
                caminhoKeyToken = Console.ReadLine().Trim();
                if (string.IsNullOrEmpty(caminhoKeyToken) || !File.Exists(caminhoKeyToken)) Con
            }
        }
        Console.WriteLine($"  Chave privada: {Path.GetFullPath(caminhoKeyToken)}");
        string respostaToken = ObterTokenOAuth(clientIdToken, clientSecretToken, caminhoCRTToke
        string tokenObtido = "";
        try
        {
            using var doc = JsonDocument.Parse(respostaToken);
            if (doc.RootElement.TryGetProperty("access_token", out JsonElement atEl))
                tokenObtido = atEl.GetString() ?? "";
        }
        catch { }
        if (string.IsNullOrEmpty(tokenObtido))
            throw new Exception("Nao foi possivel obter o token automaticamente. Tente a opcao
        Console.WriteLine($"  Token obtido com sucesso: {Mascarar(tokenObtido)}");
        Log("INFO", $"Token obtido automaticamente via mTLS: {Mascarar(tokenObtido)}");
        return (tokenObtido, caminhoCRTToken, caminhoKeyToken);
    }
    else
    {
        Log("INFO", "Obtencao de token manual");
        Console.Write("Token STS para renovacao: ");
        string tokenManual = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(tokenManual)) throw new Exception("Token STS e obrigatorio par
        Console.WriteLine("\n  Para a requisicao de renovacao, e necessario o certificado e a c
        string caminhoCertMtls = GetEstadoStr(estado, "caminhoCRT", Path.Combine(PastaEtapa2, "
        if (!File.Exists(caminhoCertMtls))
        {
            Console.WriteLine($"  [AVISO] Certificado nao encontrado em: {Path.GetFullPath(cami
            caminhoCertMtls = "";
            while (string.IsNullOrEmpty(caminhoCertMtls) || !File.Exists(caminhoCertMtls))
            {
                Console.Write("  Informe o caminho do certificado (.crt): ");
                caminhoCertMtls = Console.ReadLine().Trim();
                if (string.IsNullOrEmpty(caminhoCertMtls) || !File.Exists(caminhoCertMtls)) Con
            }
        }
        Console.WriteLine($"  Certificado: {Path.GetFullPath(caminhoCertMtls)}");
        string caminhoKeyMtls = Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIVA
        if (!File.Exists(caminhoKeyMtls))
        {
            Console.WriteLine($"  [AVISO] Chave privada nao encontrada em: {caminhoKeyMtls}");
            caminhoKeyMtls = "";
            while (string.IsNullOrEmpty(caminhoKeyMtls) || !File.Exists(caminhoKeyMtls))
            {
                Console.Write("  Informe o caminho da chave privada (.key): ");
                caminhoKeyMtls = Console.ReadLine().Trim();
                if (string.IsNullOrEmpty(caminhoKeyMtls) || !File.Exists(caminhoKeyMtls)) Conso
            }
        }
        Console.WriteLine($"  Chave privada: {Path.GetFullPath(caminhoKeyMtls)}");
        return (tokenManual, caminhoCertMtls, caminhoKeyMtls);
    }
}

// Funcao local para exibir validade
void ExibirValidadeRenovacao(string crtPath)
{
    var expiracaoRenov = DateTime.Now.AddDays(365);
    var inicioRenov = expiracaoRenov.AddDays(-30);
    var fimRenov = expiracaoRenov.AddDays(-1);
    Console.WriteLine("\n==================================================");
    Console.WriteLine("  VALIDADE DO CERTIFICADO RENOVADO                ");
    Console.WriteLine("==================================================");
    Console.WriteLine("  Validade: 365 dias");
    Console.WriteLine($"  Data de expiracao: {expiracaoRenov:dd/MM/yyyy}");
    Console.WriteLine($"  Periodo de renovacao: de {inicioRenov:dd/MM/yyyy} ate {fimRenov:dd/MM
    Console.WriteLine("\n  O processo de renovacao (etapa 4) pode ser realizado a partir de");
    Console.WriteLine("  30 dias antes da expiracao ate um dia antes da data de expiracao do ce
    Console.WriteLine("\n  ATENCAO: A responsabilidade de renovar o certificado dentro do prazo
    Console.WriteLine("  e exclusivamente sua. Certificados expirados impossibilitam o acesso")
    Console.WriteLine("  as APIs do Itau e exigem a geracao de um novo certificado do zero.");
    Log("INFO", $"Certificado renovado: {Path.GetFullPath(crtPath)}");
    Log("INFO", "Etapa 4 concluida com sucesso");
    Console.WriteLine("\n>> Etapa 4 concluida. Certificado renovado.");
}
```

```
// Funcao local para oferecer PFX
void OferecerPfxRenovacao(string crtPath, string keyPath)
{
    Console.Write("\nDeseja gerar o arquivo PFX (PKCS#12)? O PFX combina o certificado e a chav
    string respPfx = Console.ReadLine().Trim();
    if (respPfx.Equals("s", StringComparison.OrdinalIgnoreCase))
    {
        Console.Write("  Informe a senha para proteger o PFX: ");
        string senhaPfx = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(senhaPfx))
        {
            Console.WriteLine("  Senha nao informada. PFX nao gerado.");
            Log("WARN", "PFX nao gerado - senha nao informada");
        }
        else
        {
            string caminhoPfx = Path.Combine(PastaEtapa4, "certificado.pfx");
            var certPfx = X509Certificate2.CreateFromPemFile(crtPath, keyPath);
            File.WriteAllBytes(caminhoPfx, certPfx.Export(X509ContentType.Pfx, senhaPfx));
            Console.WriteLine($"  PFX gerado: {Path.GetFullPath(caminhoPfx)}");
            Log("INFO", $"PFX gerado: {Path.GetFullPath(caminhoPfx)}");
        }
    }
}

if (opcaoRenov == "A")
{
    Log("INFO", "Opcao A: Renovacao completa (novo CSR e nova chave privada)");
    Console.WriteLine("\n[PASSO 1] Gerando nova chave privada e novo CSR...");
    string caminhoClientId4b = GetEstadoStr(estado, "caminhoClientId", Path.Combine(PastaEtapa2
    string clientId4b;
    if (File.Exists(caminhoClientId4b))
    {
        clientId4b = File.ReadAllText(caminhoClientId4b).Trim();
    }
    else if (estado.ContainsKey("clientId") && !string.IsNullOrEmpty(GetEstadoStr(estado, "clie
    {
        clientId4b = GetEstadoStr(estado, "clientId", "");
        Console.WriteLine("  Client ID recuperado do estado salvo.");
    }
    else
    {
        Console.WriteLine("  Nao foi possivel recuperar o Client ID automaticamente.");
        Console.WriteLine("  Informe o Client ID manualmente para continuar.");
        clientId4b = "";
        while (string.IsNullOrEmpty(clientId4b))
        {
            Console.Write("Client ID: ");
            clientId4b = Console.ReadLine().Trim();
            if (string.IsNullOrEmpty(clientId4b)) Console.WriteLine("  Campo obrigatorio. Infor
        }
    }
    Console.WriteLine($"  Client ID: {Mascarar(clientId4b)}");

    string organizacao4 = "";
    while (string.IsNullOrEmpty(organizacao4))
    {
        Console.Write("Nome da Empresa: ");
        organizacao4 = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(organizacao4)) Console.WriteLine("  Campo obrigatorio. Informe
    }
    string cidade4 = "";
    while (string.IsNullOrEmpty(cidade4))
    {
        Console.Write("Cidade: ");
        cidade4 = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(cidade4)) Console.WriteLine("  Campo obrigatorio. Informe a ci
    }
    string estadoUF4 = "";
    while (string.IsNullOrEmpty(estadoUF4))
    {
        Console.Write("Estado - UF: ");
        estadoUF4 = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(estadoUF4)) Console.WriteLine("  Campo obrigatorio. Informe o
    }
    string pais4 = "";
    while (string.IsNullOrEmpty(pais4))
    {
        Console.Write("Pais: ");
        pais4 = Console.ReadLine().Trim();
        if (string.IsNullOrEmpty(pais4)) Console.WriteLine("  Campo obrigatorio. Informe o pais
    }

    string caminhoCSR4b = Path.Combine(PastaEtapa4, "ARQUIVO_REQUEST_CERTIFICADO.csr");
    string caminhoChaveCSR4b = Path.Combine(PastaEtapa4, "ARQUIVO_CHAVE_PRIVADA.key");
    GerarCSR(clientId4b, organizacao4, cidade4, estadoUF4, pais4, caminhoCSR4b, caminhoChaveCSR
    Log("INFO", $"Novo CSR gerado: {Path.GetFullPath(caminhoCSR4b)}");
    Log("INFO", $"Nova chave privada gerada: {Path.GetFullPath(caminhoChaveCSR4b)}");

    Console.WriteLine("\n[PASSO 2] Obtendo token para renovacao...");
    var resultadoToken = ObterTokenRenovacao();
    string tokenSTS = resultadoToken.token;
    string caminhoCertMtlsA = resultadoToken.certPath;
    string caminhoKeyMtlsA = resultadoToken.keyPath;
```

```
                        Console.WriteLine("\n[PASSO 3] Enviando novo CSR para renovacao...");
                        string caminhoCRT = Path.Combine(PastaEtapa4, "certificado.crt");
                        Log("INFO", $"Token STS: {Mascarar(tokenSTS)}");
                        Log("INFO", $"CSR: {Path.GetFullPath(caminhoCSR4b)}");
                        Log("INFO", "Enviando requisicao para https://sts.itau.com.br/seguranca/v2/certificado/reno
                        RenovarCertificado(tokenSTS, caminhoCSR4b, caminhoCRT, caminhoCertMtlsA, caminhoKeyMtlsA);

                        ExibirValidadeRenovacao(caminhoCRT);
                        Console.WriteLine("\n  O que fazer agora:");
                        Console.WriteLine($"  1. O novo certificado esta salvo na pasta {Path.GetFullPath(PastaEtap
                        Console.WriteLine("  2. Substitua o certificado antigo E a chave privada pelos novos na sua
                        Console.WriteLine($"  3. A nova chave privada (.key) esta em {Path.GetFullPath(caminhoChave
                        Console.WriteLine("  4. Teste a geracao de token (etapa 3) para confirmar que o novo certif

                        OferecerPfxRenovacao(caminhoCRT, Path.GetFullPath(caminhoChaveCSR4b));
                    }
                    else
                    {
                        Log("INFO", "Opcao B: Renovacao simples (reutiliza CSR existente)");
                        Console.WriteLine("\n[PASSO 1] Localizando CSR existente...");
                        string caminhoCSR = GetEstadoStr(estado, "caminhoCSR", "");
                        if (string.IsNullOrEmpty(caminhoCSR) || !File.Exists(caminhoCSR))
                        {
                            string caminhoCSRPadrao = Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_REQUEST_C
                            if (File.Exists(caminhoCSRPadrao))
                            {
                                caminhoCSR = caminhoCSRPadrao;
                            }
                            else
                            {
                                Console.Write("Caminho do arquivo CSR: ");
                                caminhoCSR = Console.ReadLine().Trim();
                                if (string.IsNullOrEmpty(caminhoCSR) || !File.Exists(caminhoCSR)) throw new Excepti
                            }
                        }
                        Console.WriteLine($"  CSR encontrado: {Path.GetFullPath(caminhoCSR)}");

                        Console.WriteLine("\n[PASSO 2] Obtendo token para renovacao...");
                        var resultadoTokenB = ObterTokenRenovacao();
                        string tokenSTS = resultadoTokenB.token;
                        string caminhoCertMtlsB = resultadoTokenB.certPath;
                        string caminhoKeyMtlsB = resultadoTokenB.keyPath;

                        Console.WriteLine("\n[PASSO 3] Enviando CSR para renovacao...");
                        string caminhoCRT = Path.Combine(PastaEtapa4, "certificado.crt");
                        Log("INFO", $"Token STS: {Mascarar(tokenSTS)}");
                        Log("INFO", $"CSR: {Path.GetFullPath(caminhoCSR)}");
                        Log("INFO", "Enviando requisicao para https://sts.itau.com.br/seguranca/v2/certificado/reno
                        RenovarCertificado(tokenSTS, caminhoCSR, caminhoCRT, caminhoCertMtlsB, caminhoKeyMtlsB);

                        ExibirValidadeRenovacao(caminhoCRT);
                        Console.WriteLine("\n  O que fazer agora:");
                        Console.WriteLine($"  1. O novo certificado esta salvo na pasta {Path.GetFullPath(PastaEtap
                        Console.WriteLine("  2. Substitua o certificado antigo pelo novo na sua aplicacao");
                        Console.WriteLine("  3. A chave privada (.key) da etapa 2 continua sendo a mesma - nao muda
                        Console.WriteLine("  4. Teste a geracao de token (etapa 3) para confirmar que o novo certif

                        string caminhoKeyPfx = Path.GetFullPath(Path.Combine(PastaEtapa2, "ARQUIVO_CHAVE_PRIVADA.ke
                        OferecerPfxRenovacao(caminhoCRT, caminhoKeyPfx);
                    }
                }
                else
                {
                    Console.WriteLine("Opcao invalida.");
                    continue;
                }
            }
            catch (Exception ex)
            {
                string detalheErro = FormatarExcecao(ex);
                Log("ERROR", detalheErro);
                Log("WARN", "Etapa NAO foi marcada como concluida devido ao erro");
                Console.WriteLine($"\n[ERRO] {detalheErro}");
                Console.WriteLine("A etapa NAO foi marcada como concluida. Verifique o erro acima e tente novamente
            }

            Console.Write("\nPressione ENTER para voltar ao menu...");
            Console.ReadLine();
        }

        if (logWriter != null) logWriter.Close();
    }
}
```

‹ Confira os pré-requisitos

Boas práticas com suas credenciais ›

© 2026 Itaú Unibanco Holding S.A. CNPJ: 60.872.504/0001-23