**Relatório PTC-Lab2**



Carlos Coelho

Cláudio Dias

José Costa

Rui Pinto

**1. Aplicação para lista de tarefas com arquitetura cliente/servidor**

**Introdução**

O desenvolvimento de um servidor de lista de tarefas utilizando sockets em Python. O objetivo é permitir que clientes possam adicionar, listar, completar e contar tarefas pendentes remotamente, interagindo com um servidor centralizado (localhost).

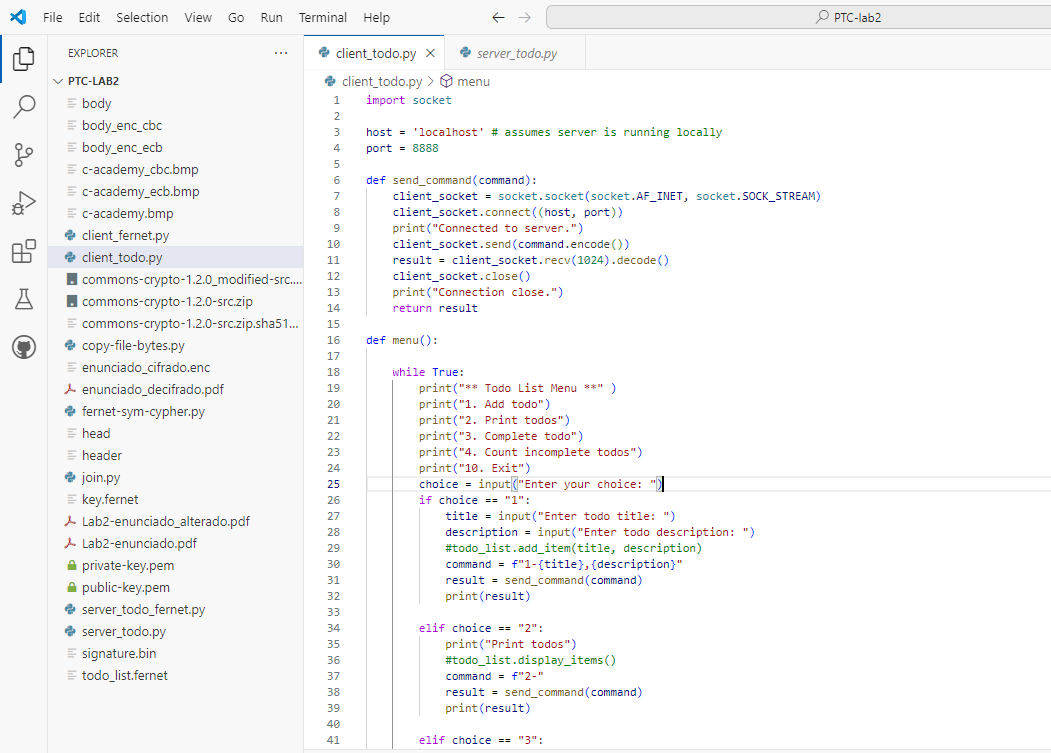
**Objetivos**

* Criar um servidor que faça a gestão de uma lista de tarefas.
* Implementar comunicação entre cliente e servidor via sockets.
* Permitir persistência dos dados para manter as tarefas salvas entre execuções.
* Garantir segurança e estabilidade no processamento de comandos.

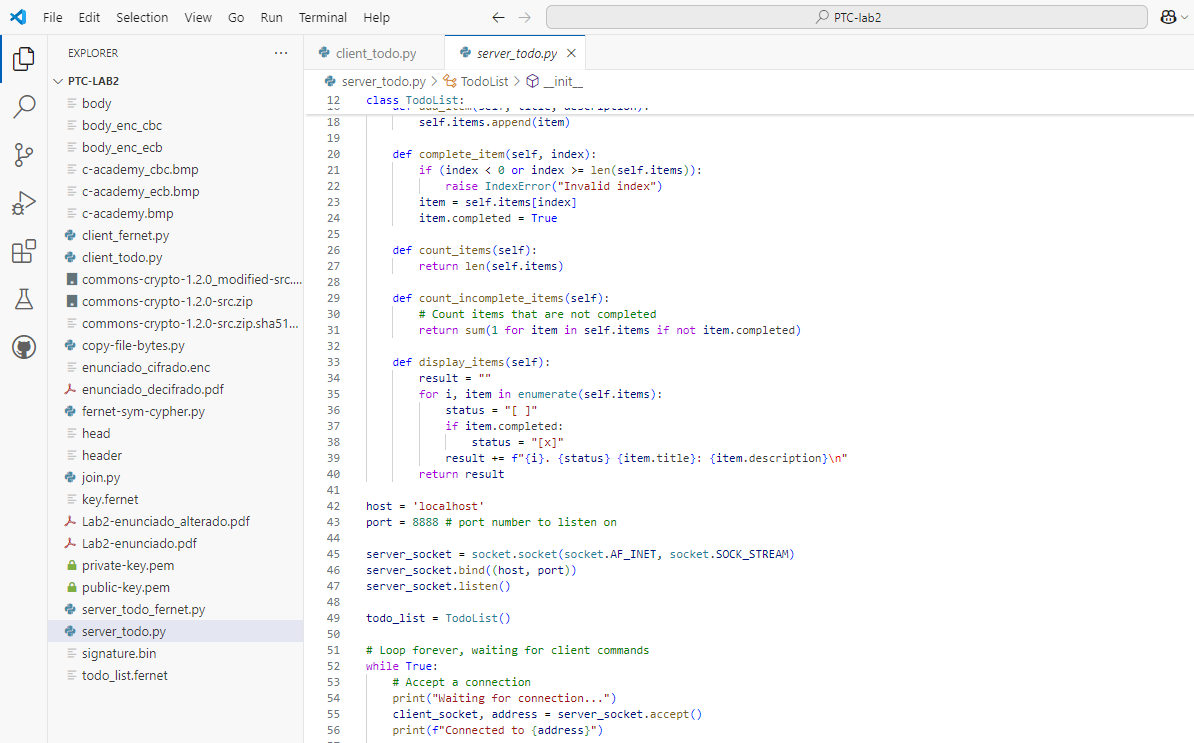
**Estrutura**

O projeto é composto por dois ficheiros principais:

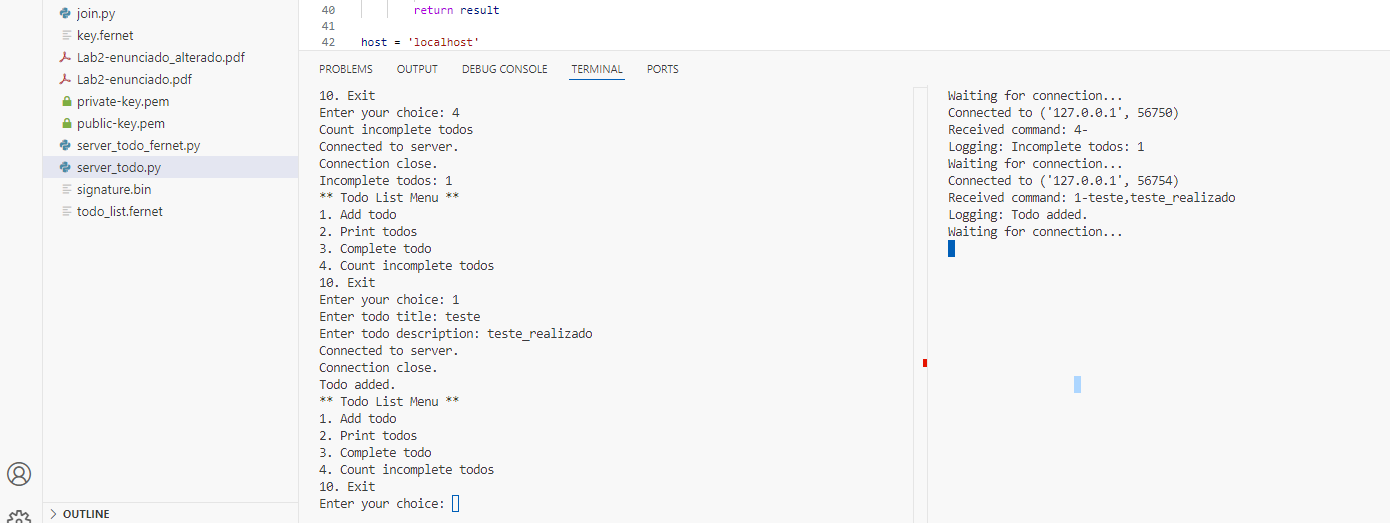
* server\_todo.py: Responsável por gerir as tarefas e responder às solicitações dos clientes.



* client\_todo.py: Permite que os utilizadores enviem comandos ao servidor.



**Funcionalidades do código**



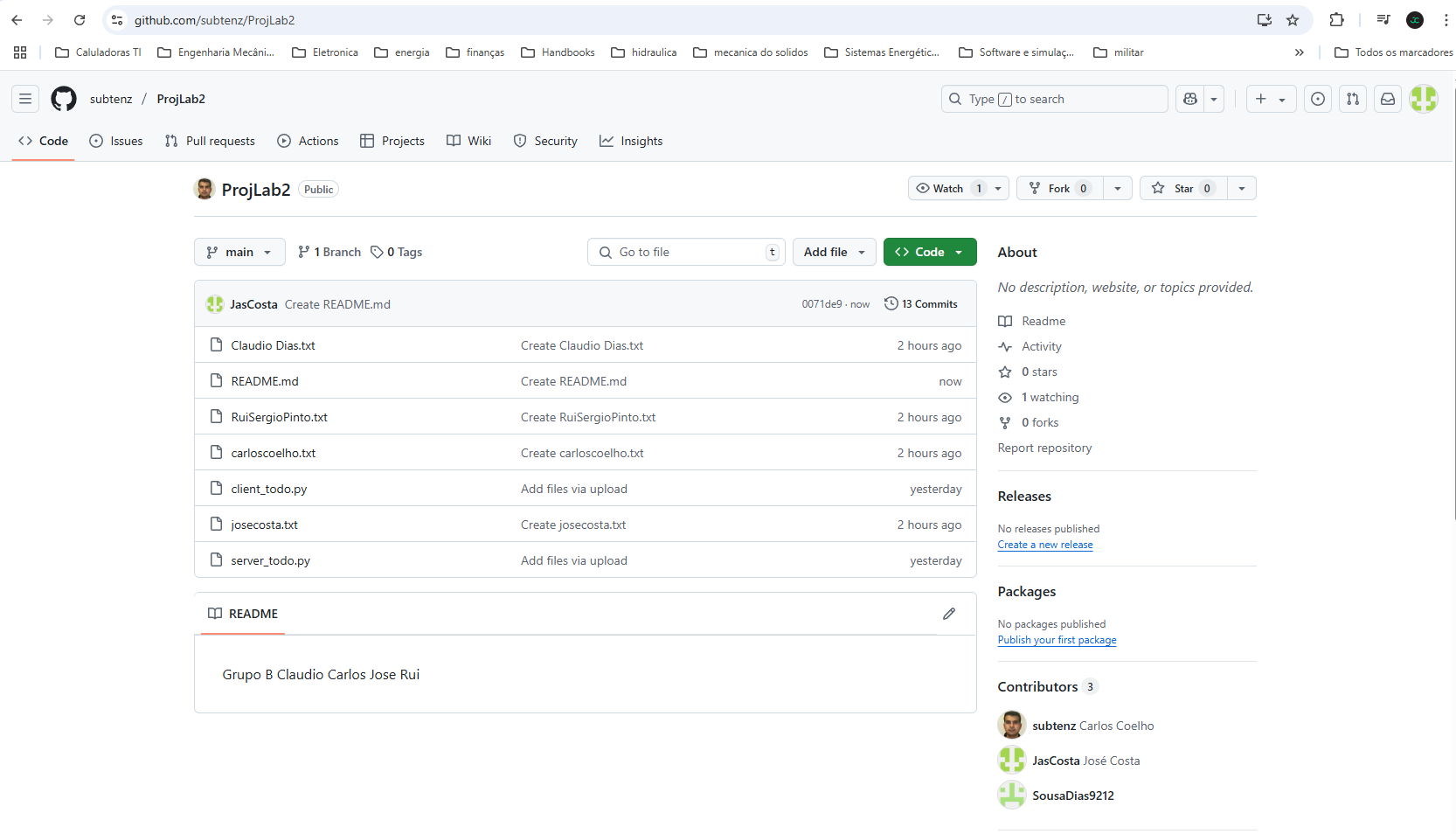
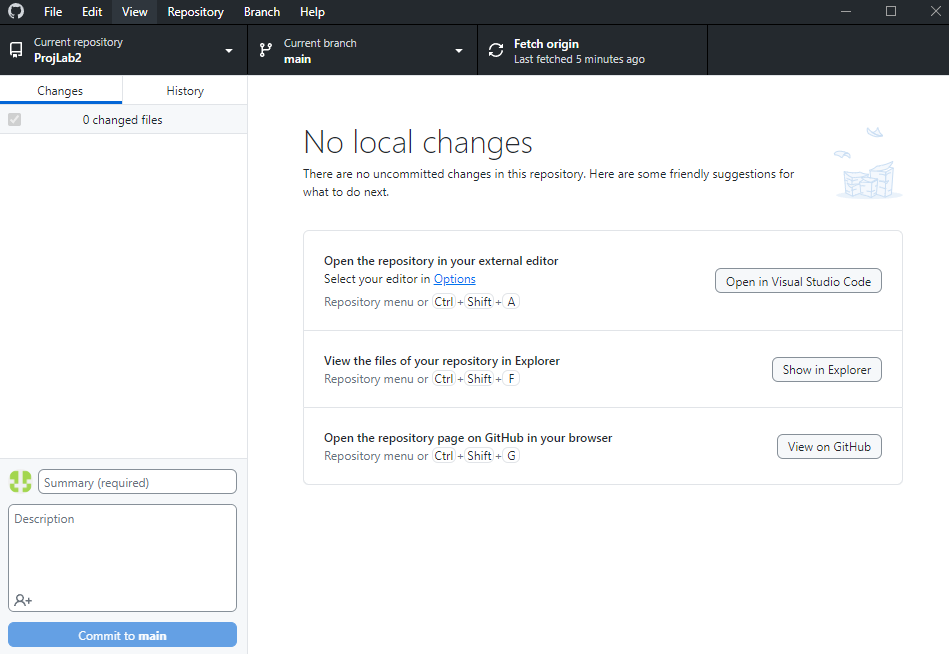
**Conclusão**

O sistema permite a gestão eficiente de tarefas através de comunicação cliente-servidor, garantindo a interatividade e o armazenamento das informações.

**2. GitHub**

**Relatório de Desenvolvimento e Colaboração no GitHub**

Configuração e colaboração de um grupo utilizando a plataforma GitHub e a aplicação GitHub Desktop para desenvolvimento de um projeto.

**3. Criptografia com OpenSSL**

**a) Cálculo de hash**

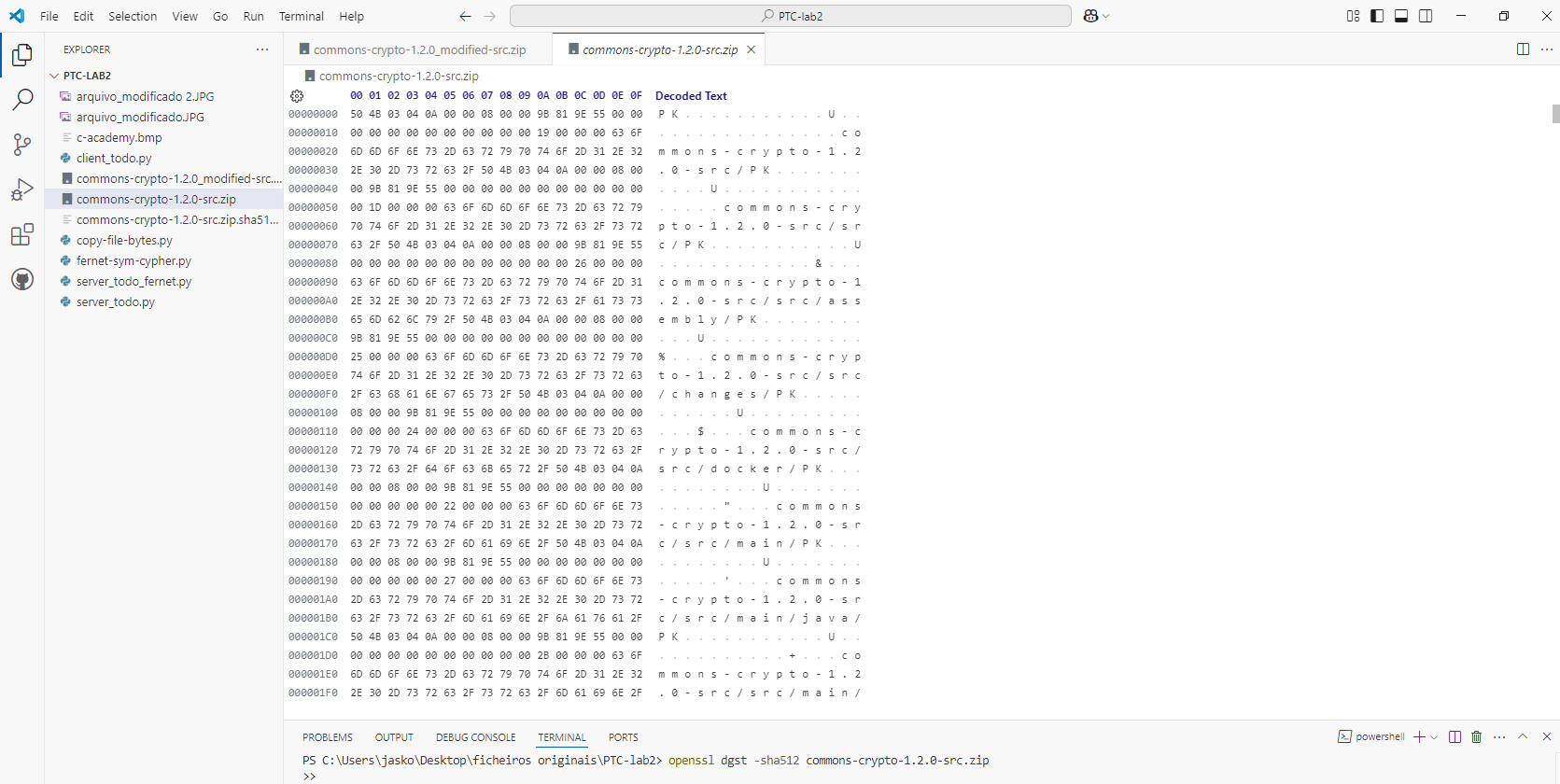
**Objetivo**

O objetivo do exercício foi:

* Calcular o hash SHA512 do arquivo de código-fonte da biblioteca Apache Commons Crypto.
* Confirmar que o hash gerado é igual ao valor divulgado no site oficial.
* Modificar um ou mais bytes do arquivo e observar a alteração no valor do hash.

**Procedimentos**

**Download do ficheiro -** nome do ficheiro**:** [**commons-crypto-1.2.0-src.zip**](https://dlcdn.apache.org/commons/crypto/source/commons-crypto-1.2.0-src.zip)

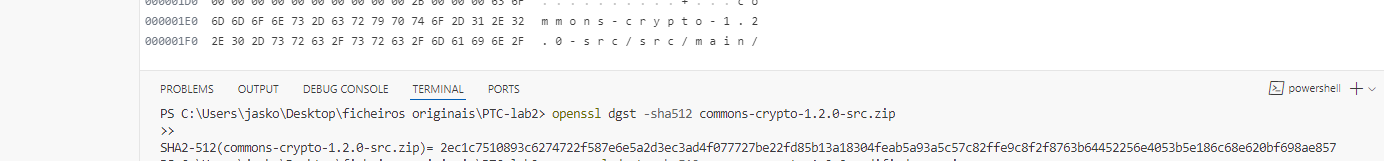


**Cálculo do Hash SHA512 Original**

* Com o terminal do VS Code aberto (dentro da pasta onde o ficheiro se encontra), foi executado o seguinte comando:

***openssl dgst -sha512 commons-crypto-1.2.0-src.zip***

* O resultado obtido foi semelhante a:

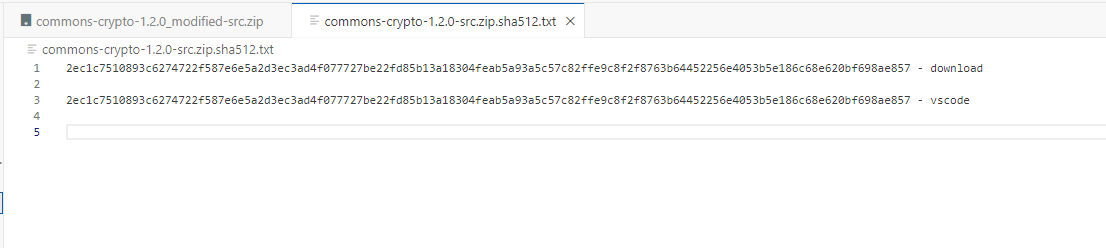


**Comparação com o Valor Anunciado**

* O valor do hash foi comparado com o divulgado no arquivo disponível em:

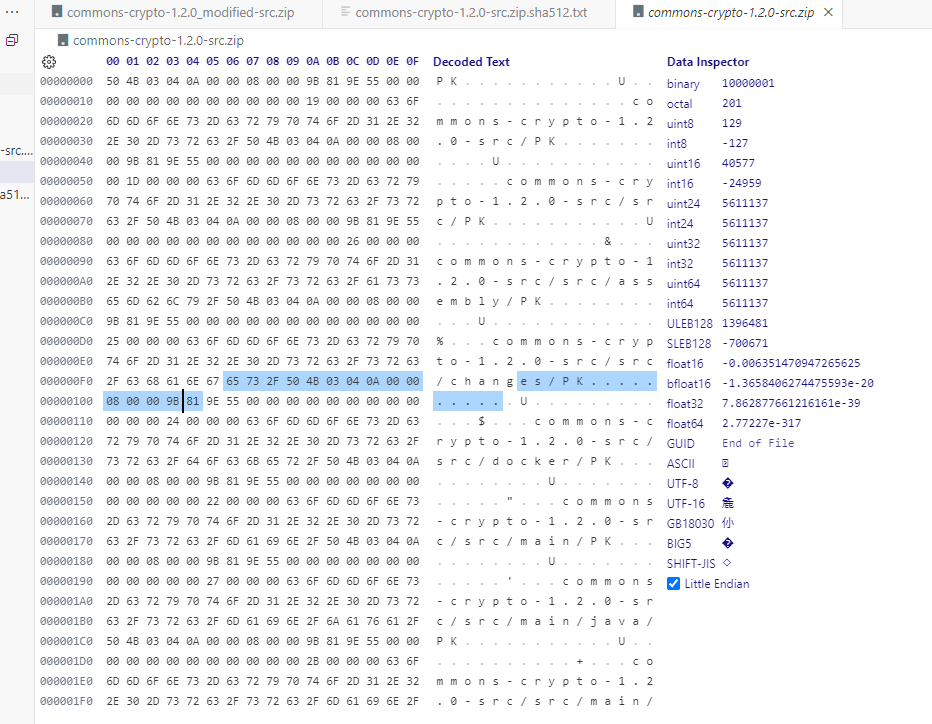
***commons-crypto-1.2.0-src.zip.sha512.txt***

* A confirmação de que os valores coincidem como revela na imagem abaixo:



**Modificação do Arquivo**

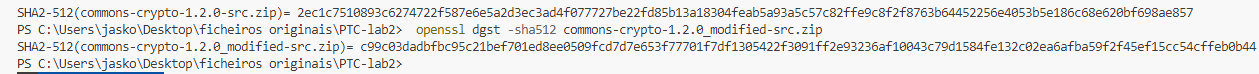
* Para demonstrar a variação do hash à menor alteração no arquivo, foi realizada uma modificação *de bytes.*
* Utilizou-se a extensão **Hex Editor** do VS Code para abrir e editar o arquivo:



* No terminal do VS Code, foi executado o comando para o arquivo modificado:

***openssl dgst -sha512 commons-crypto-1.2.0-src-modified.zip***

* O hash gerado foi diferente do hash original, conforme espectável, em que mesmo uma alteração mínima modifica completamente o valor do hash.



**Conclusão**

* Calcular e verificar o hash SHA512 utilizando o OpenSSL.
* Validar a integridade de um arquivo comparando o hash gerado com um hash de referência.
* Evidenciar a sensibilidade dos algoritmos hash a alterações mínimas em ficheiros.
* As capturas de tela anexadas confirmam os passos realizados e os resultados obtidos.

**b) Cálculo de Message Authentication Code (MAC)1. Objetivo**

Este exercício teve como finalidade:

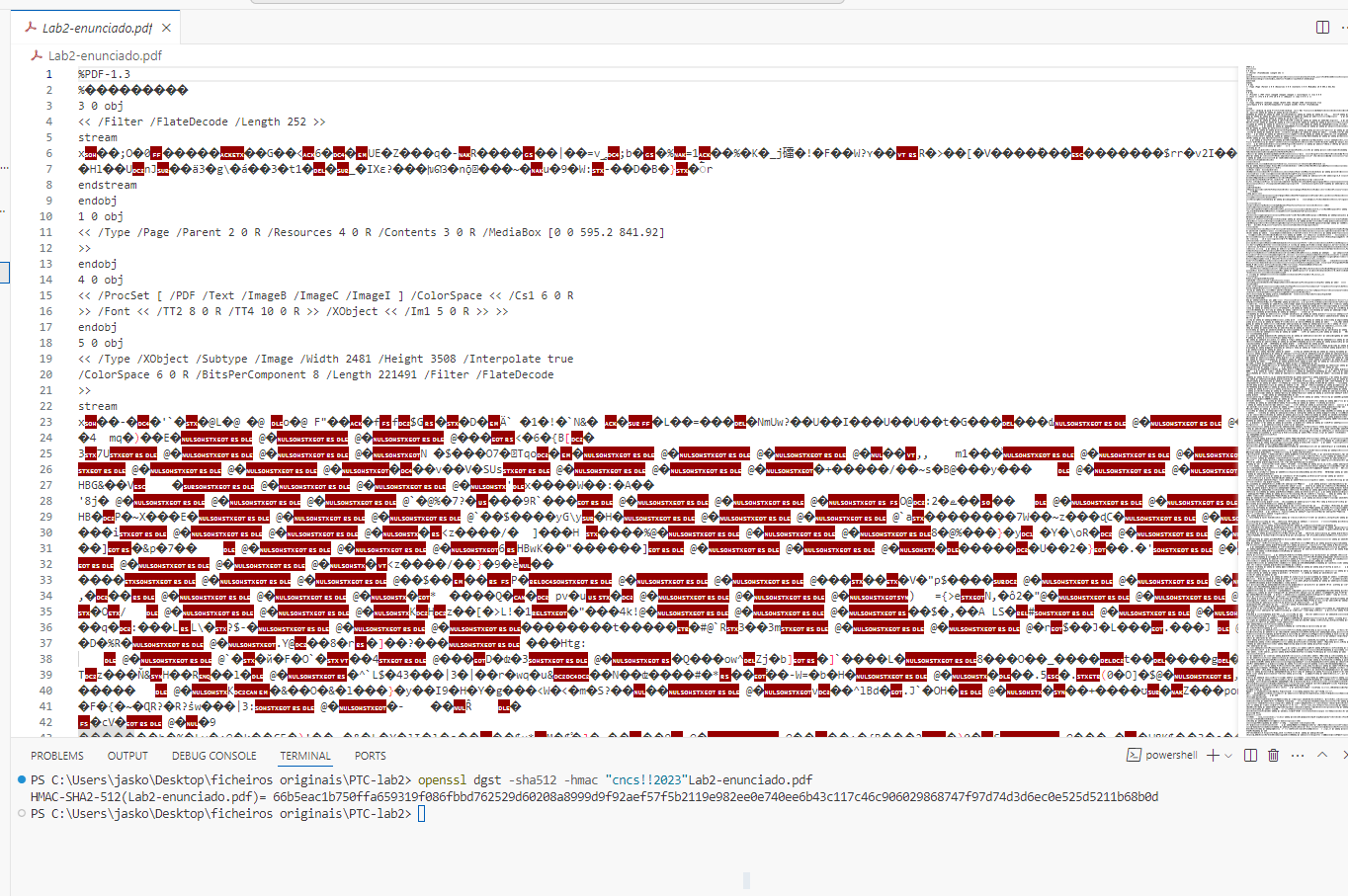
* Calcular o MAC do PDF do enunciado utilizando o algoritmo HMAC-SHA512.
* Utilizar uma chave (“***cncs!!2023***”) escolhida pelo grupo para o cálculo.

**Procedimentos**

* **Ficheiro:** PDF do enunciado do exercício.
* **Chave utilizada:** O grupo definiu a chave *"cncs!!2023"* .

**Cálculo do MAC com OpenSSL**

* No VS Code navegar até o diretório onde se encontra o PDF do enunciado.
* Executar o seguinte comando para calcular o MAC utilizando HMAC-SHA512:



Onde:

* *ha512* define o algoritmo hash a ser utilizado.
* *hmac* **"cncs!!2023"** especifica a chave a ser usada para o cálculo do MAC.
* *Lab2-enunciado.pdf* é o nome do ficheiro PDF do enunciado.

**Conclusão**

O exercício demonstrou a aplicação prática do cálculo de MAC utilizando o OpenSSL.

Com a execução do comando, foi possível:

* Gerar o MAC para o PDF do enunciado.
* Confirmar a integridade e autenticidade do documento utilizando uma chave definida pelo grupo.

**c) Cifra e decifra simétrica**

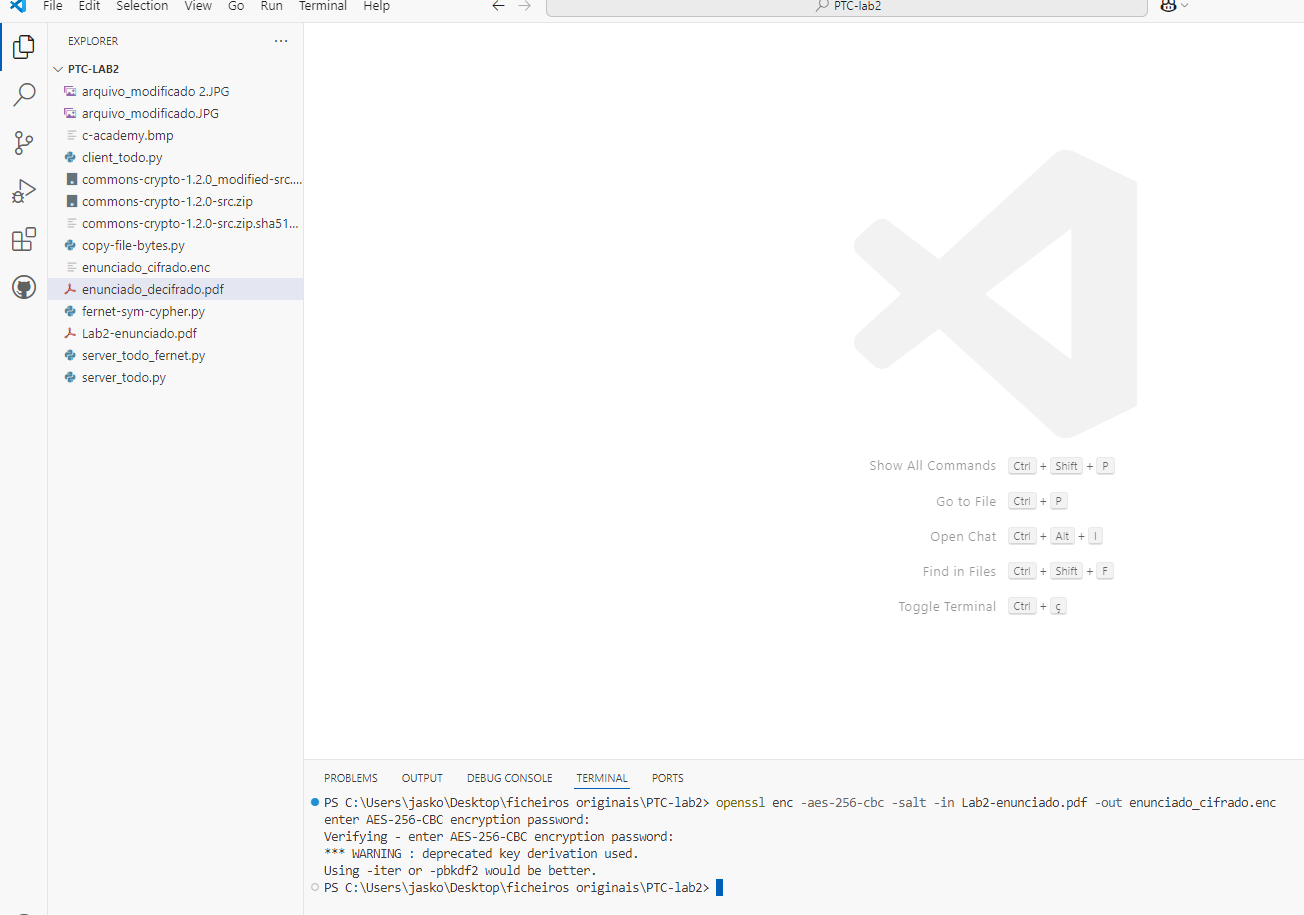
**Objetivo**

* Cifrar o ficheiro PDF usando uma chave criada pelo grupo.
* Distribuir o ficheiro cifrado entre os membros do grupo.
* Garantir que os membros conseguem decifrar o ficheiro e verificar sua integridade.

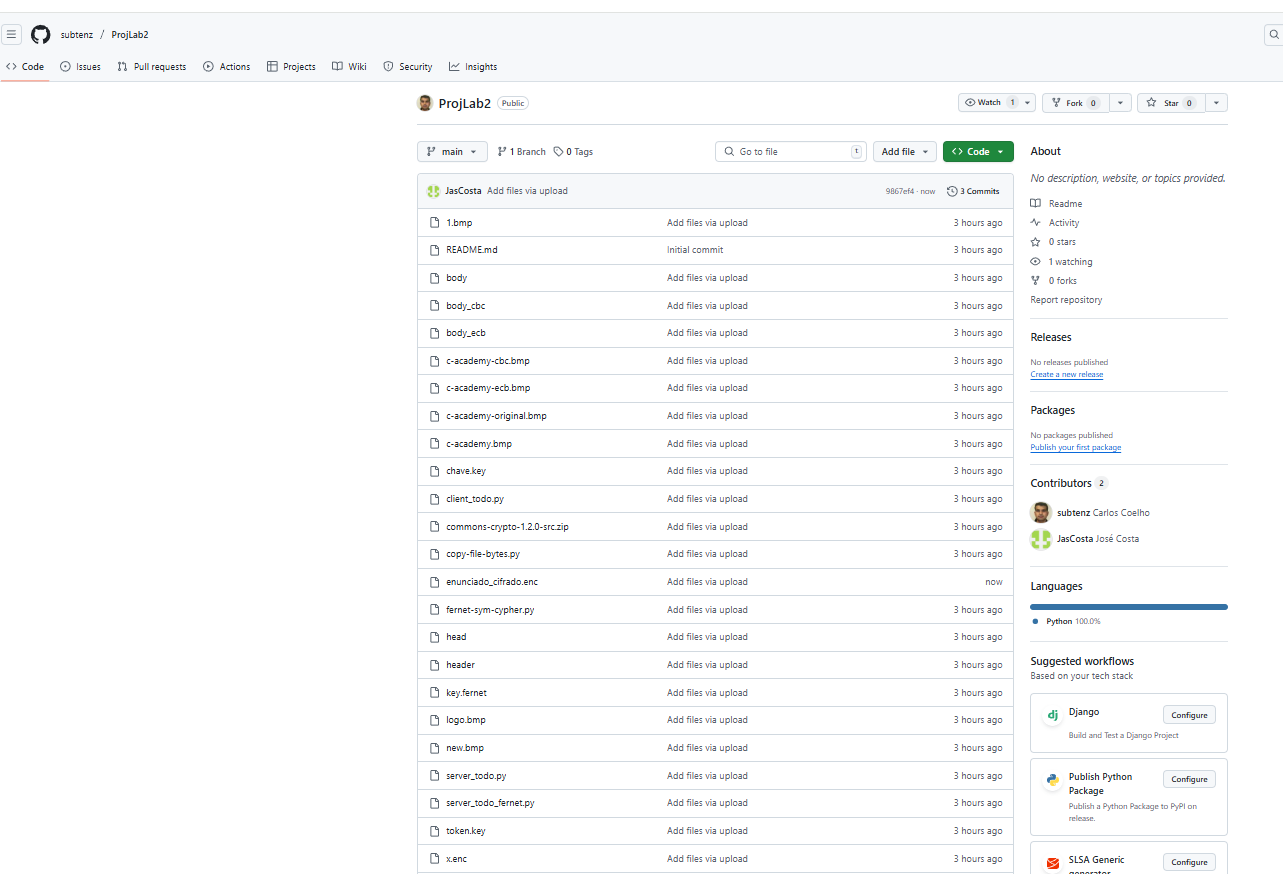
**Procedimentos**

* O ficheiro **"Lab2-enunciado.pdf"** foi cifrado utilizando o comando:

***openssl enc -aes-256-cbc -salt -in Lab2-enunciado.pdf -out enunciado\_cifrado.enc***



* A chave de cifra utilizada foi: **"grupob2025!"**
* O ficheiro cifrado **"***enunciado\_cifrado.enc***"** foi gerado e compartilhado no repositório do grupo



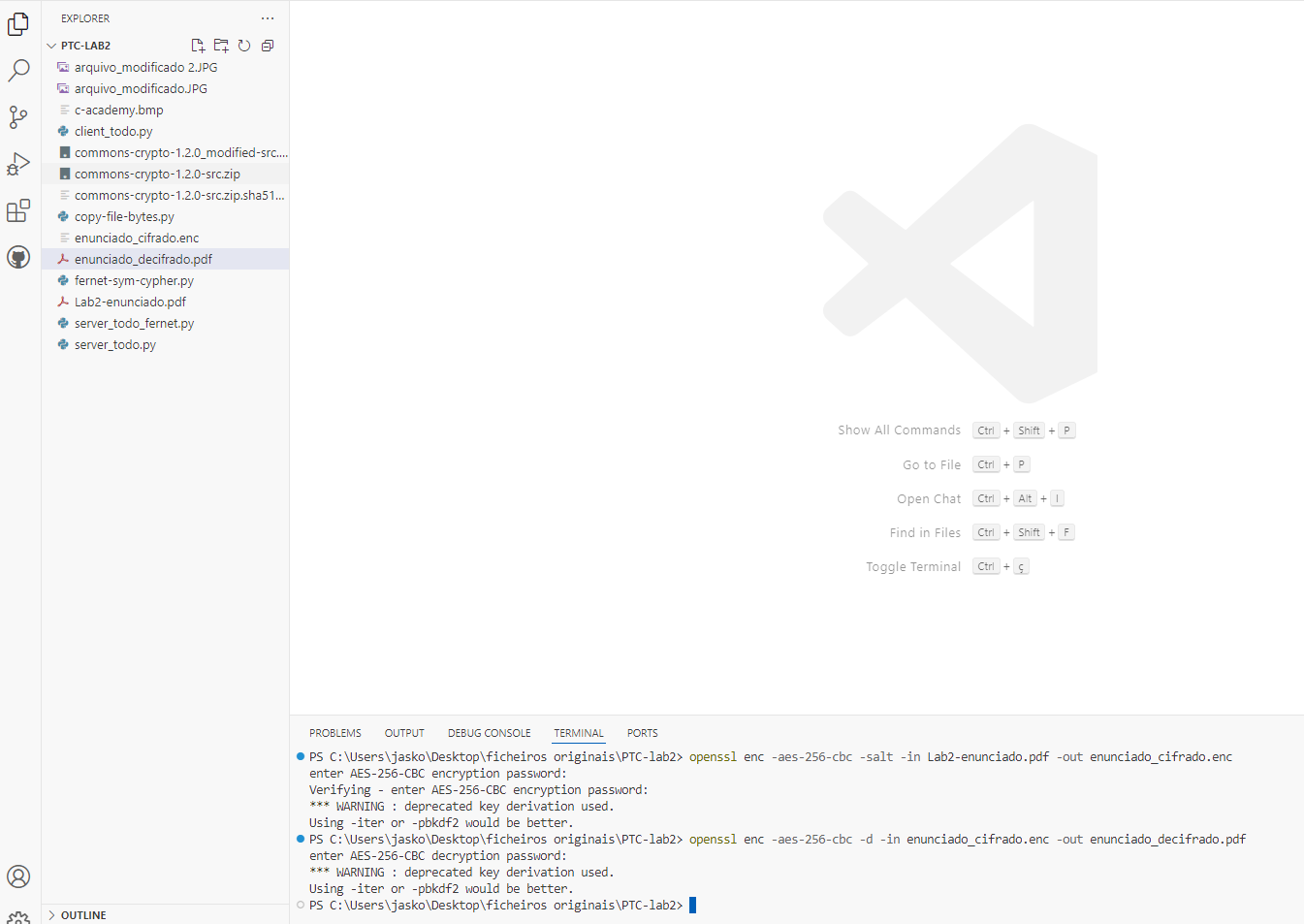
Ficheiro compartilhado no Github

**Decifra do Ficheiro**

* Os membros do grupo baixaram o ficheiro cifrado e executaram o comando para recuperar o PDF original:

***openssl enc -aes-256-cbc -d -in enunciado\_cifrado.enc -out enunciado\_decifrado.pdf***

* Foi necessário introduzir a chave secreta do grupo para completar a decifra.



Esse aviso indica que o OpenSSL está a usar um método antigo de chave, o que pode não ser tão seguroquanto métodos mais atuais. Para evitar esse aviso e melhorar a segurança, usa o parâmetro o **Password-Based Key Derivation Function 2 (PBKDF2)**.

**Conclusão**

O exercício demonstrou a aplicação prática de cifra e decifra de ficheiros usando OpenSSL. Através deste processo, foi possível:

* Proteger o ficheiro PDF contra acessos não autorizados.
* Garantir que apenas os membros do grupo que possuem a chave pudessem recuperar o conteúdo original.
* Validar a integridade do ficheiro após a decifra.

**d) Modos de operação**

**Objetivo**

O algoritmo de cifra simétrica AES (Advanced Encryption Standard) para cifrar o corpo de uma imagem BMP (c-academy.bmp) com dois modos de operação diferentes: **AES-ECB (Electronic Codebook)** e **AES-CBC (Cipher Block Chaining)**. O objetivo foi cifrar o corpo da imagem enquanto mantinha intato o cabeçalho, utilizando a ferramenta OpenSSL para a cifra e Python para a manipulação dos ficheiros.

A cifra simétrica AES é amplamente utilizada pela sua segurança e eficiência, sendo usada em diferentes modos de operação. O modo ECB cifra blocos de dados de forma independente, enquanto o CBC encadeia os blocos de dados, tornando o processo mais seguro.

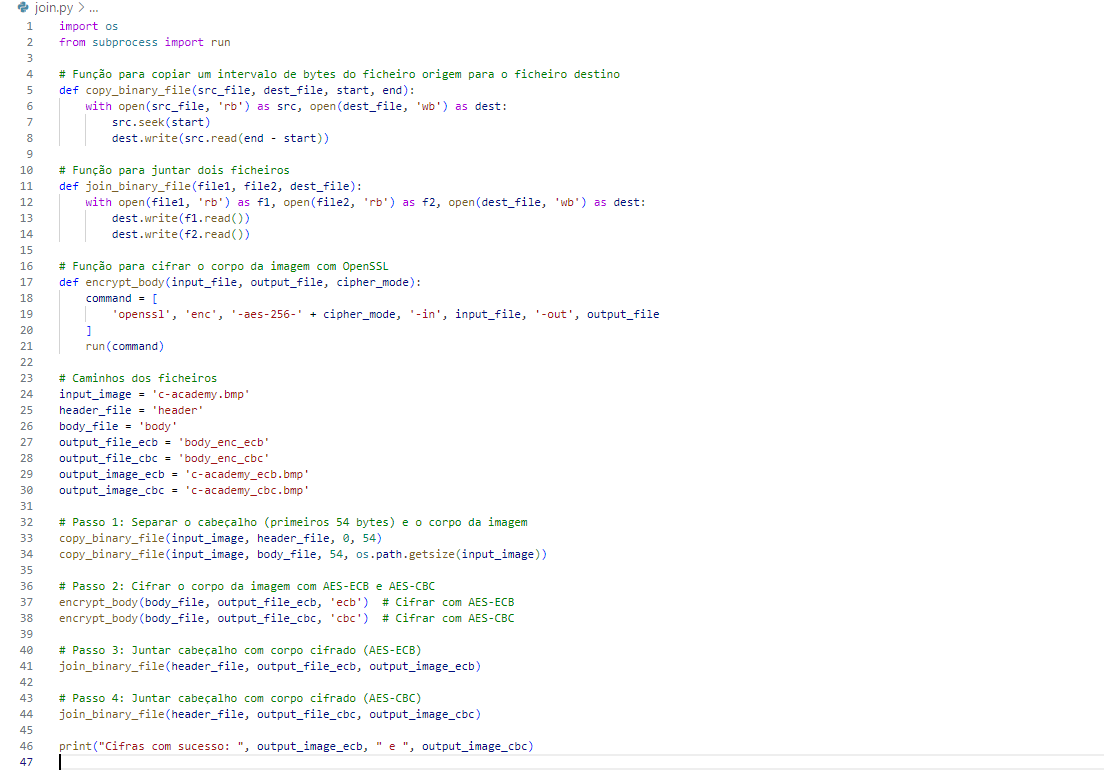
**Procedimento**

Primeiramente, a imagem original c-academy.bmp foi dividida em dois segmentos:

* **Cabeçalho (Primeiros 54 bytes):** A primeira parte da imagem, contendo as informações de formato e metadados.
* **Corpo da Imagem (Restante dos Bytes):** A segunda parte da imagem, que contém os dados da imagem propriamente dita.

**Cifra do Corpo da Imagem**

* O corpo da imagem foi cifrado utilizando o algoritmo AES com os modos de operação ECB e CBC, utilizando a ferramenta OpenSSL e python.



**Juntando Cabeçalho e Corpo Cifrado**

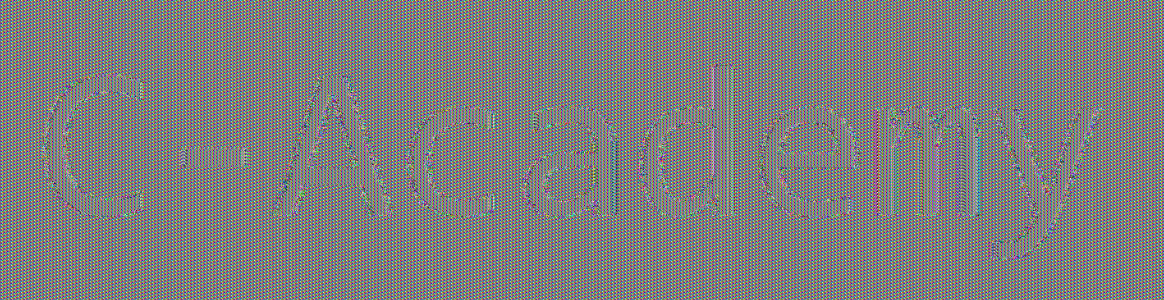
Após a cifra dos corpos com os dois modos de operação, os arquivos resultantes foram reunidos com o cabeçalho original para formar as imagens cifradas:

**Resultados**

Os dois modos de operação geraram imagens cifradas com características distintas:

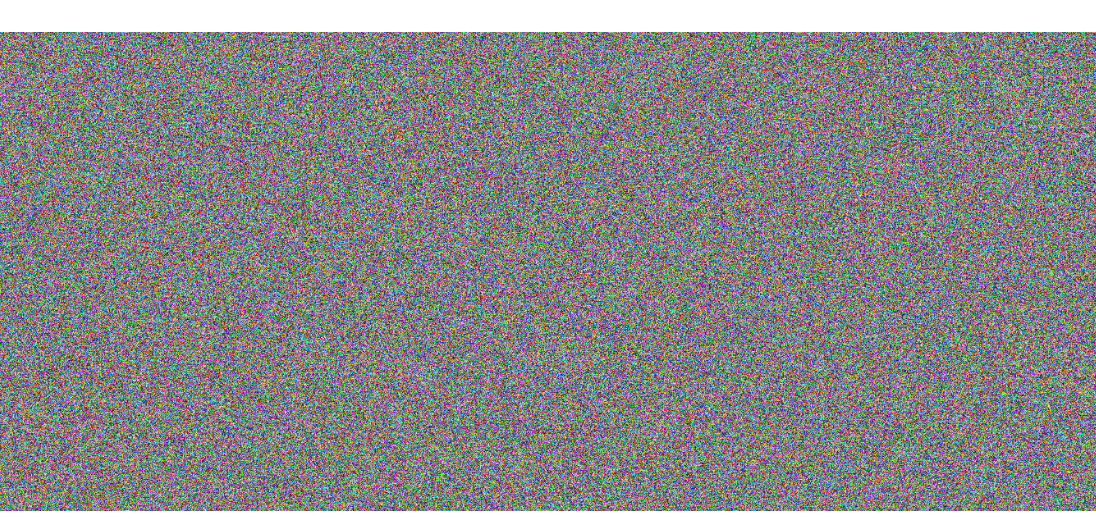
**AES-ECB**:

* + O modo ECB cifra blocos de dados de forma independente, podendo resultar na repetição de padrões visuais na imagem cifrada.
  + É menos seguro pois revela informações sobre o padrão da imagem original.



**AES-CBC**:

* + O modo CBC cifra cada bloco baseado no bloco anterior, tornando a cifragem mais segura.
  + A imagem cifrada apresenta ruído aleatório, sem revelar padrões da imagem original.



**Diferença entre os Resultados**

A principal diferença está no comportamento dos blocos de dados:

* + - **AES-ECB**: Mantém padrões visíveis na imagem cifrada, tornando-o vulnerável a ataques de análise de padrão.
    - **AES-CBC**: Oferece maior segurança, pois cada bloco de dados depende do anterior, dificultando a identificação da imagem original.

**Conclusão**

Foi demonstrado como a cifra simétrica AES pode ser aplicada a uma imagem usando dois modos de operação: **AES-ECB** e **AES-CBC**. O modo **AES-CBC** oferece uma segurança superior devido ao encadeamento entre os blocos de dados, enquanto o modo **AES-ECB** é mais simples, mas menos seguro. Ambos os arquivos cifrados foram gerados com sucesso e adicionados ao repositório GitHub, garantindo a integridade dos resultados.

Nota: a rotina que está a ser utilizado abre o OpenSSL diretamente na máquina devido à configuração do ambiente.

**e) Criptografia assimétrica**

**Objetivo**

A criptografia assimétrica com a geração de chaves RSA, assinatura digital de um ficheiro PDF e verificação da autenticidade da assinatura.

Os ficheiros utilizados para a experiência são:

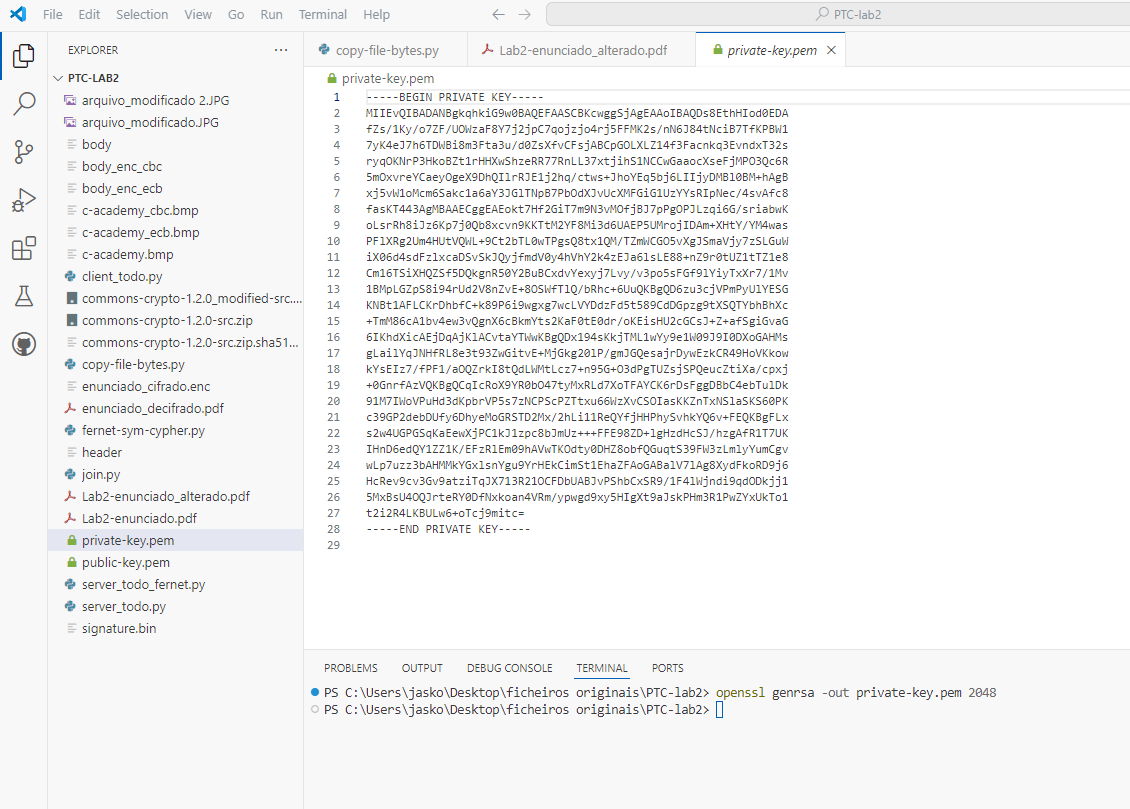
* + **Lab2-enunciado.pdf** (original)
  + **Lab2-enunciado\_alterado.pdf** (modificado).

**Procedimento**

**Gerar um par de chaves RSA (2048 bits)**

Como gerar uma chave privada:

**openssl genrsa -out private-key.pem 2048**



**Criar uma assinatura digital do ficheiro PDF**

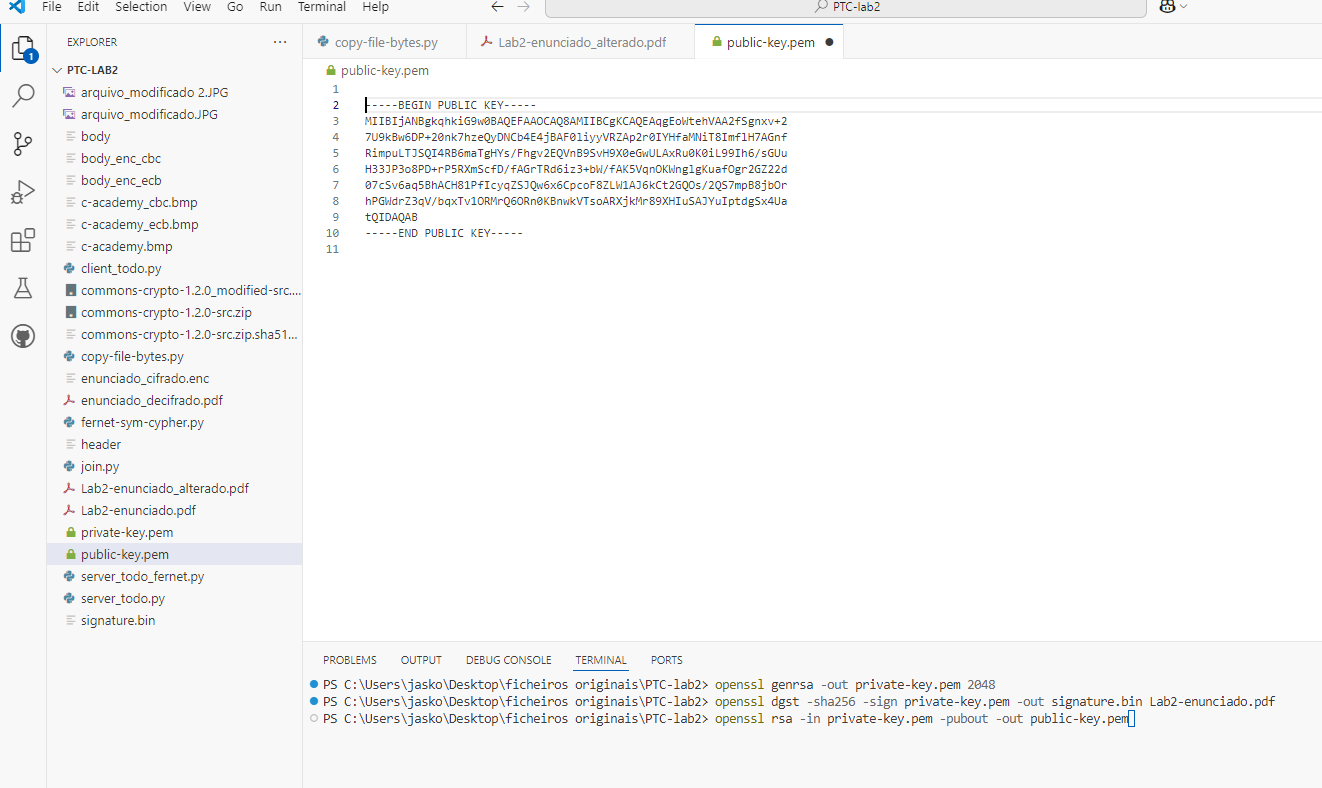
Assinamos o ficheiro Lab2-enunciado.pdf utilizando a chave privada:

**openssl dgst -sha256 -sign private-key.pem -out signature.bin Lab2-enunciado.pdf**

**Extrair a chave pública da chave privada**

A chave pública, utilizamos:

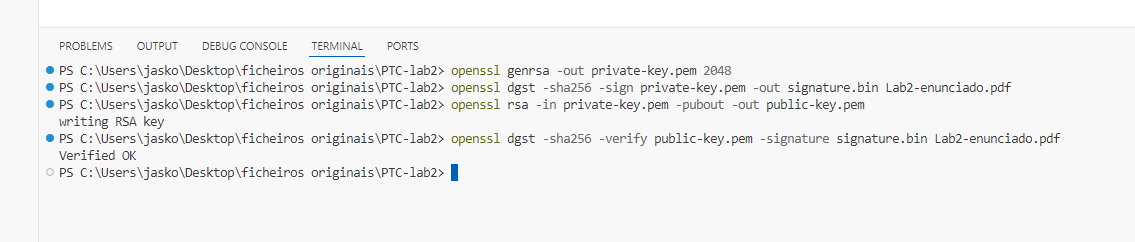
**openssl rsa -in private-key.pem -pubout -out public-key.pem**



**Verificar a assinatura digital**

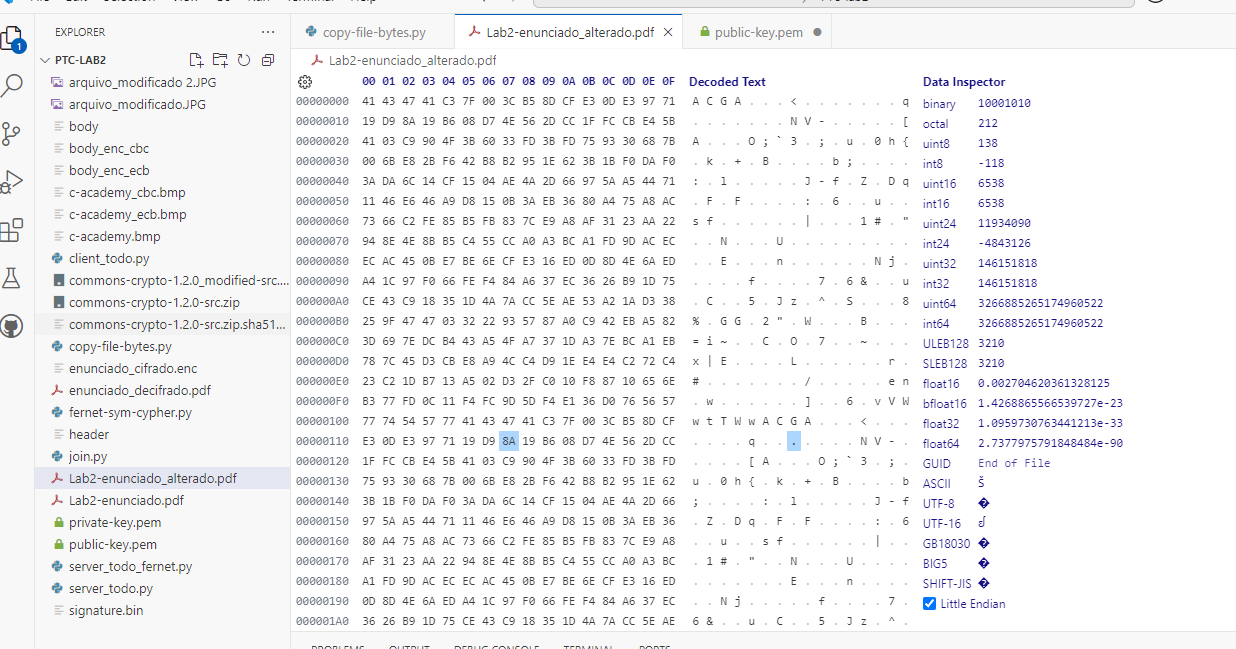
Com a chave pública, verificamos a assinatura do ficheiro original:

**openssl dgst -sha256 -verify public-key.pem -signature signature.bin Lab2-enunciado.pdf**



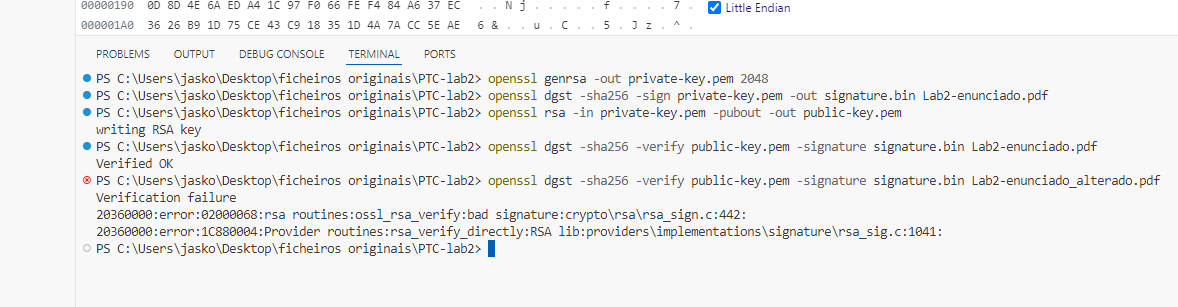
**Modificar um byte do ficheiro PDF**

* Abrimos o ficheiro Lab2-enunciado.pdf no Visual Studio Code (VS Code) em modo hexadecimal.
* Alteramos um byte aleatório.
* Guardamos o ficheiro modificado como Lab2-enunciado\_alterado.pdf.



**Repetir a verificação da assinatura**

**openssl dgst -sha256 -verify public-key.pem -signature signature.bin Lab2-enunciado\_alterado.pdf**



A verificação falha porque o ficheiro foi alterado, invalidando a assinatura.

**Conclusão**

* A assinatura digital é única para o ficheiro original.
* Qualquer alteração no ficheiro invalida a assinatura.
* A verificação falha quando um ou mais bytes é modificado, provando a integridade da criptografia assimétrica
* Este método é amplamente utilizado para garantir a autenticidade e integridade de documentos digitais.

**4. (extra) Criptografia em Python**

Implementar um sistema simples de gerenciamento de tarefas (todo list) que utiliza comunicação via sockets e criptografia para garantir a segurança das informações armazenadas. O sistema é composto por um servidor e um cliente.

**Servidor**: O servidor gere a lista de tarefas, permitindo adicionar, completar e visualizar tarefas, além de criptografar e descriptografar os dados.

**Cliente**: O cliente se conecta ao servidor para enviar comandos (como adicionar tarefas, listar tarefas e completar tarefas) e receber respostas.

**Arquitetura do Sistema**

O sistema é composto por dois componentes principais:

**Servidor**

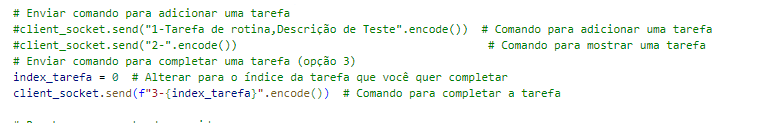
O servidor escuta na porta 8888 e aguarda conexões do cliente.

O servidor recebe comandos do cliente e executa ações como adicionar tarefas, listar tarefas e completar tarefas.

As tarefas são armazenadas e cifradas utilizando o algoritmo de criptografia simétrica Fernet, garantido pela biblioteca cryptography.

**Cliente**

O cliente envia comandos ao servidor por meio de um socket TCP, permitindo a interação com o sistema.



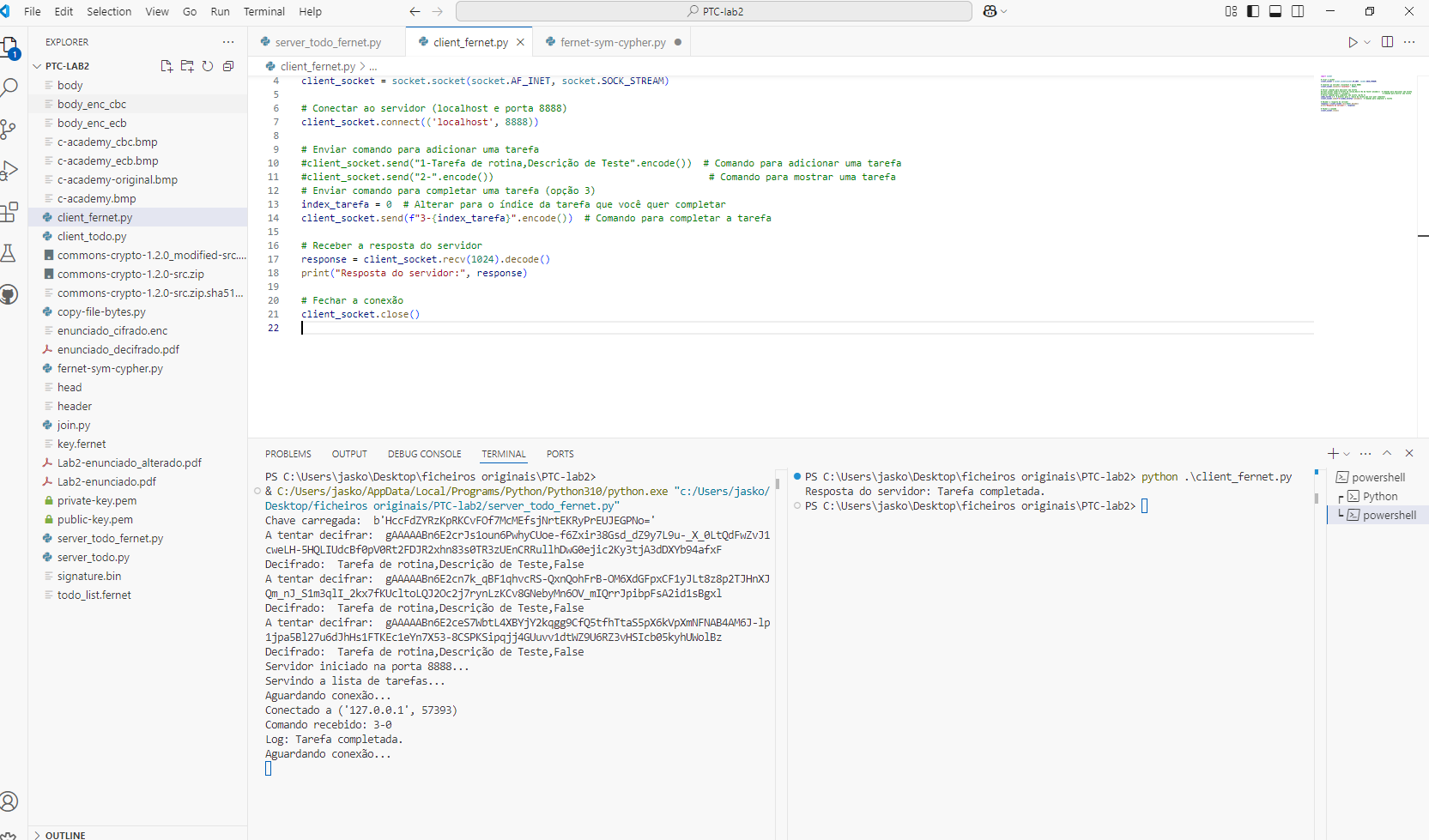
1-Envia um comando para adicionar uma tarefa com título "Tarefa de rotina" e descrição "Descrição de Teste".

2- Solicita ao servidor que envie a lista de todas as tarefas.

3- Envia o comando para marcar a tarefa no índice especificado (neste caso, 0), como completada.

O cliente pode enviar comandos para adicionar uma nova tarefa, listar todas as tarefas ou completar uma tarefa especificando o índice.

O cliente exibe as respostas recebidas do servidor.



**Conclusão**

O sistema é capaz de adicionar, listar e completar tarefas, garantindo a segurança das informações através da criptografia.