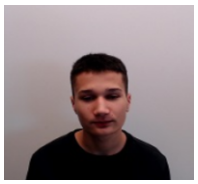
Universidade do Minho
Departamento de Informática

Criptografia e Segurança da Informação

Tecnologias de Segurança - TP3

Grupo 6
João Andrade Rodrigues (PG57879)
Benjamim Meleiro Rodrigues (PG57511)
Martim José Amaro Redondo (PG57889)


PG57879


PG57511


PG57889

30 de maio de 2025
Ano Letivo 2024/25

Conteúdo

1 Introdução

A gestão segura da informação é um desafio central em sistemas que lidam com dados sensíveis, especialmente em contextos com diferentes níveis de acesso. Os sistemas de segurança multi-nível garantem que a informação é apenas acessível por utilizadores autorizados, conforme o seu nível de autorização (clearance) e a classificação da informação.

O modelo Bell-LaPadula (BLP), centrado na proteção da confidencialidade, define duas regras principais: "no read up"e "no write down", impedindo que a informação flua para níveis inferiores. Contudo, a sua aplicação prática revela limitações, como a rigidez na definição de níveis e a ausência de mecanismos de partilha controlada entre classificações.

Este trabalho propõe a conceção e implementação de uma solução funcional baseada no BLP, adaptada a um sistema real. A solução inclui controlo de acesso multi-nível, gestão de utilizadores e domínios, autenticação com certificados digitais, comunicação segura via TLS com autenticação mútua, e encriptação dos conteúdos armazenados.

2 Modelo e Espaço de Desenho

2.1 Modelo de Segurança Implementado

A solução desenvolvida baseia-se numa adaptação prática do modelo Bell-LaPadula, estruturando o sistema em torno de níveis de segurança e domínios de informação. Foram definidos quatro níveis hierárquicos de segurança, ordenados por grau de confidencialidade: Top Secret, Secret, Confidential e Public, correspondentes aos valores numéricos de 0 a 3. Estes níveis aplicam-se tanto a utilizadores (clearance) como a ficheiros (classificação).

A informação é também compartimentada por domínios, representando áreas funcionais ou temáticas como Projects, Operations, Intelligence e General. Cada domínio pode existir em qualquer dos níveis definidos, formando instâncias como, por exemplo, Projects ao nível Secret ou General ao nível Public. A associação entre domínios e níveis é configurável e validada no sistema.

Os utilizadores são caracterizados por três propriedades fundamentais: o seu nível de segurança (clearance), a lista de domínios base atribuídos (que são interpretados no contexto do seu nível), e dois indicadores booleanos — is_admin e is_trusted. O atributo is_admin confere permissões administrativas totais sobre o sistema, enquanto is_trusted permite a estes utilizadores contornar algumas das restrições mais rígidas do modelo BLP clássico, permitindo-lhes escrever para níveis inferiores de segurança.

2.2 Regras de Acesso

As regras de acesso implementadas seguem os princípios fundamentais do modelo Bell-LaPadula, com algumas extensões para acomodar casos de uso práticos. Em conformidade com o modelo clássico, o sistema garante que os utilizadores podem ler ficheiros do seu nível ou de níveis inferiores (read down), mas não podem ler ficheiros classificados acima do seu nível de segurança.

Relativamente à escrita, a regra base do BLP — write up only — é respeitada, o que significa que um utilizador pode escrever apenas em ficheiros ou domínios do seu nível de segurança ou superiores. Esta regra evita a possibilidade de fuga de informação sensível para níveis mais baixos, mas impõe limitações consideráveis na comunicação descendente.

Para mitigar essa limitação, foi introduzida a noção de utilizadores de confiança (is_trusted), os quais podem escrever para domínios em níveis inferiores (write down), assumindo-se que realizam previamente uma filtragem da informação sensível antes da partilha. Este comportamento permite uma maior flexibilidade, mantendo ao mesmo tempo um controlo rigoroso sobre quem pode descer informação entre níveis.

Por fim, os utilizadores com privilégios administrativos (is_admin) estão isentos das restrições do modelo BLP, podendo aceder e modificar informação em qualquer nível e domínio, bem como criar, atualizar e remover utilizadores, níveis e domínios.

2.3 Adaptações ao BLP

De forma a tornar o modelo Bell-LaPadula aplicável num ambiente prático e dinâmico, foram introduzidas várias adaptações à sua formulação original. Em primeiro lugar, a solução implementa sessões de utilizador temporizadas, com um sistema de controlo de integridade baseado em HMAC (Hash-based Message Authentication Code). Este mecanismo garante que a sessão ativa está associada a um utilizador válido, e que os dados críticos de sessão não foram adulterados.

Além disso, a comunicação entre o cliente e o servidor é assegurada por TLS com autenticação mútua, com os certificados gerados por uma Autoridade Certificadora própria. No momento de autenticação, o servidor atribui ao cliente uma chave de sessão simétrica, usada para encriptar e assinar os dados trocados durante a sessão, reforçando a confidencialidade e a integridade da comunicação.

Por fim, foi também incluída uma camada de gestão dinâmica de utilizadores e domínios, acessível apenas a administradores. Esta funcionalidade permite criar ou remover utilizadores, definir níveis de segurança e configurar quais os domínios base válidos em cada nível. Esta flexibilidade, ausente no modelo original, é essencial para suportar um sistema operativo realista

e adaptável.

3 Arquitetura da Solução

3.1 Componentes

A solução está organizada em torno de uma arquitetura cliente-servidor, composta por múltiplos ficheiros que interagem entre si para implementar o modelo de controlo de acesso baseado no Bell-LaPadula.

O ficheiro server.py constitui o servidor principal, implementado com o framework Flask. Este expõe uma API REST segura, onde cada endpoint está protegido por mecanismos de autenticação e autorização. O servidor mantém o estado da sessão do utilizador e verifica a validade das operações com base nas regras do modelo.

Do lado do cliente, o ficheiro client.py implementa uma interface em linha de comandos (CLI), permitindo ao utilizador interagir com o servidor através de comandos como login, listagem de ficheiros, upload e download. O cliente suporta comunicação via TLS com autenticação mútua, bem como encriptação e verificação de integridade dos dados transmitidos, utilizando uma chave de sessão simétrica.

A gestão de dados persistentes é assegurada pelo módulo `database_manager.py`, que encapsula a lógica de acesso e validação sobre a base de dados TinyDB. Este componente gere utilizadores, ficheiros, níveis de segurança e domínios de forma estruturada, assegurando a coerência das regras de segurança.

A base de dados encontra-se fisicamente armazenada no ficheiro `blp_database.json`, onde são registadas todas as entidades do sistema. Os conteúdos dos ficheiros são encriptados antes de serem armazenados, utilizando uma chave Fernet, que se encontra guardada em encryption_key.json.

3.2 Fluxo de funcionamento

O funcionamento do sistema inicia-se com o processo de autenticação. O utilizador fornece as suas credenciais ao cliente, que as envia de forma segura ao servidor. Após validação, o servidor atribui uma chave de sessão simétrica ao cliente, codificada em Base64. Esta chave é usada em todas as interações subsequentes para encriptar os dados e gerar assinaturas HMAC, garantindo a confidencialidade e integridade da comunicação.

Durante a sessão, o utilizador pode executar várias operações, como listar ficheiros acessíveis, transferir ficheiros ou realizar uploads. Cada ficheiro está associado a um nível de segurança e a um domínio base. O servidor aplica as regras de acesso do modelo BLP para validar cada pedido, assegurando que o utilizador tem permissão para a ação solicitada.

Os administradores têm acesso a endpoints adicionais que permitem a gestão dinâmica de utilizadores, domínios e níveis de segurança. Estas operações são protegidas por verificações de privilégio e apenas podem ser executadas por utilizadores com a flag is_admin.

O servidor mantém ainda um controlo de atividade da sessão, aplicando um timeout após um período de inatividade, o que obriga a um novo login. Todas as interações são validadas com base nos dados armazenados na sessão e assinados com HMAC, garantindo que os pedidos não foram adulterados durante o transporte.

3.3 Segurança da Comunicação

A comunicação entre cliente e servidor é protegida através de TLS com autenticação mútua, garantindo que ambas as partes são autenticadas antes de qualquer troca de dados. Os certificados digitais necessários são gerados com o script generate_certificates.py, que cria uma autoridade certificadora (CA) local e emite certificados válidos para o servidor e para os clientes.

Após o estabelecimento da ligação segura, o servidor envia ao cliente uma chave de sessão simétrica, utilizada para encriptar os dados das mensagens e gerar assinaturas HMAC. Estas assinaturas permitem ao destinatário verificar a integridade e autenticidade dos dados recebidos, protegendo o sistema contra ataques de falsificação ou modificação de mensagens.

As sessões têm uma validade limitada no tempo, sendo automaticamente terminadas após um período de inatividade, prevenindo abusos ou uso indevido de sessões esquecidas. A verificação da integridade da sessão é feita com base num hash dos dados essenciais do utilizador, armazenado na própria sessão e comparado em cada pedido.

Importa salientar que, embora o TLS assegure a ligação segura entre o terminal e o servidor, este nível de proteção não é suficiente por si só para garantir que utilizadores subsequentes no mesmo terminal não possam aceder a dados sensíveis de sessões anteriores. Para mitigar este risco, a nossa solução implementa três medidas adicionais cruciais:

- **Refresh automático de sessões:** a sessão do utilizador é validada e renovada em cada pedido, assegurando que apenas sessões ativas e válidas permanecem operacionais.
- **Limpeza de rastros após logout:** após um utilizador terminar sessão, os dados da sessão são eliminados tanto no lado do cliente como do servidor, removendo qualquer possível cache ou rastro de dados.
- **Encriptação dos payloads nas rotas administrativas:** todas as operações sensíveis (como configuração de domínios ou permissões) requerem que os dados sejam encriptados com a chave de sessão e va-

lidados com HMAC, evitando que qualquer acesso local ao terminal permita interferência com estas ações críticas.

Este conjunto de medidas garante que a solução implementada não só respeita os princípios do modelo Bell-LaPadula, como também incorpora boas práticas de segurança na comunicação, gestão de sessões e proteção de terminais partilhados.

4 Implementação

4.1 Principais classes e funções

A implementação da solução foi realizada integralmente em Python, com uma separação clara entre os componentes de cliente, servidor e base de dados. A estrutura modular facilita a manutenção do código e permite testar e evoluir cada parte do sistema de forma independente.

A classe DatabaseManager, presente no módulo homónimo, é responsável por toda a gestão da base de dados. Esta classe permite criar utilizadores, níveis de segurança e domínios, bem como armazenar ficheiros encriptados. Além disso, inclui funções para hashing seguro de palavras-passe com PBKDF2-HMAC, e para verificação de permissões de acesso com base nas regras do modelo Bell-LaPadula. O acesso à base de dados é feito através da biblioteca TinyDB, e o armazenamento dos dados é mantido legível graças a uma implementação personalizada de JSONStorage com indentação.

O BLPClient, implementado em client.py, fornece uma interface de linha de comandos que permite ao utilizador interagir com o servidor. Esta classe trata da encriptação e desencriptação dos dados, da geração e verificação de assinaturas HMAC, e do envio de pedidos autenticados. O cliente é também responsável por gerir o estado da sessão e apresenta as respostas de forma informativa, com feedback visual diferenciado (cores e ícones).

No servidor, o ficheiro server.py define a API utilizando o framework Flask. Estão implementados decoradores de autenticação e autorização, que garantem que apenas utilizadores autenticados e autorizados acedem a determinadas rotas. O servidor gere as sessões com um timeout e aplica verificações de integridade com HMAC, de modo a garantir que as sessões não foram comprometidas. As rotas implementadas cobrem operações de login, logout, verificação de estado, listagem e manipulação de ficheiros, utilizadores, domínios e níveis.

4.2 Encriptação de conteúdos

A proteção da confidencialidade dos ficheiros é assegurada através da encriptação dos conteúdos antes de serem armazenados. Para isso, foi utilizado o módulo Fernet da biblioteca cryptography, que implementa encriptação simétrica com AES-128 em modo CBC e padding PKCS7, assegurando tanto a

confidencialidade como a integridade dos dados. A chave simétrica utilizada para esta encriptação é gerada automaticamente na criação da base de dados, caso não exista já uma chave guardada, e é posteriormente armazenada no ficheiro encryption_key.json. Esta chave é lida tanto pelo servidor como pelas ferramentas auxiliares de população e teste da base de dados.

Os conteúdos dos ficheiros são, assim, encriptados com a chave Fernet antes de serem inseridos na base de dados, o que garante que a informação armazenada não pode ser acedida directamente mesmo que o ficheiro blp_database.json seja comprometido.

Além da encriptação, todas as mensagens trocadas entre cliente e servidor durante uma sessão são assinadas com HMAC-SHA256, utilizando uma chave de sessão temporária atribuída pelo servidor após o login. Esta assinatura garante que os dados não foram modificados durante a transmissão e permite rejeitar pedidos forjados ou corrompidos. As mensagens encriptadas e assinadas são codificadas em Base64 para transporte.

4.3 Mecanismo de Auditoria e Registo de Logs

A solução implementa um sistema de auditoria detalhado que permite monitorizar e registar todas as ações críticas realizadas pelos utilizadores e administradores da plataforma. Este mecanismo visa garantir transparência, facilitar a deteção de comportamentos indevidos e fornecer suporte para futuras análises forenses, em alinhamento com os princípios do modelo de segurança Bell-LaPadula.

O sistema de logging baseia-se na biblioteca logging do Python, com configuração personalizada para escrever os registos num ficheiro geral e, adicionalmente, num ficheiro individual por utilizador.

O ficheiro principal de logs (blp_audit.log) armazena todos os eventos relevantes, como autenticações, leituras e escritas de ficheiros, operações administrativas e tentativas de acesso não autorizado. Cada registo inclui um timestamp, o nível de severidade (INFO, WARNING, ERROR), e uma mensagem descritiva que inclui o nome do utilizador e a ação realizada.

Para reforçar a auditabilidade individual e facilitar a análise de comportamento por utilizador, a aplicação cria também um ficheiro de log separado por utilizador autenticado. Estes ficheiros têm o formato `logs/user_<nome>.log` e contêm apenas as ações realizadas por esse utilizador.

As ações auditadas incluem, entre outras:

Categoria de Ação Auditada	Descrição do Evento
Autenticação	Sucesso ou falha na autenticação de um utilizador
Listagem de recursos	Consulta dos ficheiros acessíveis ao utilizador autenticado
Operações sobre ficheiros	Leitura, escrita (append) e eliminação de ficheiros
Gestão de domínios	Criação de domínios base e configuração de permissões por nível
Violações de políticas de acesso	Tentativas de acesso não autorizado, incluindo quebras do modelo BLP (ex.: escrita para níveis inferiores por utilizadores não confiáveis)

Tabela 1: Principais categorias de ações auditadas no sistema

A severidade dos eventos é usada para distinguir a natureza dos mesmos: ações esperadas são registadas como INFO, violações de permissões como WARNING, e erros internos ou exceções como ERROR.

A função log_user_action(action: str, level: str = "info") centraliza este comportamento, obtendo automaticamente o nome do utilizador autenticado e escrevendo a mensagem tanto no log geral como no log individual, quando aplicável.

Este sistema de registo permite não só cumprir os requisitos de segurança do sistema, mas também suportar a responsabilização dos utilizadores, deteção de incidentes e manutenção de um histórico de ações.

Este mecanismo de auditoria reforça a conformidade com os princípios do modelo Bell-LaPadula, nomeadamente o controlo rigoroso de acessos por nível de classificação e o princípio de mínimo privilégio.

5 Execução do Sistema

Para executar o sistema, devem ser seguidos os seguintes passos em dois terminais distintos:

- **Terminal 1**: `make clean-all`, `make`, `make certificates`, `make server`
- **Terminal 2**: `make interactive-tests` (executa os testes automáticos) ou `make client` (acesso manual via CLI)

6 Casos de Teste e Validação

6.1 Testes Automatizados com Cliente CLI

A validação da solução desenvolvida foi realizada através de um conjunto abrangente de testes automáticos e manuais. Para os testes automáticos, foi desenvolvido um ficheiro `test_client.py` que simula interações reais com o cliente `client.py` em modo linha de comandos, utilizando as bibliotecas `pexpect` e `unittest` . Estes testes reproduzem a experiência de um utilizador real, garantindo que cada funcionalidade responde corretamente aos inputs fornecidos.

Os testes abrangem os seguintes cenários:

- Autenticação com credenciais válidas e inválidas;
- Listagem de ficheiros disponíveis por utilizador;
- Upload de ficheiros com controlo de permissões;
- Append de conteúdo a ficheiros existentes;
- Acesso a funcionalidades administrativas;
- Restrições de acesso com base nos níveis de segurança e domínios atribuídos.

Cada um destes testes foi desenhado para confirmar tanto o comportamento esperado (acesso permitido quando autorizado) como a rejeição de acessos indevidos, testando de uma forma completa as permissões e restrições impostas pelo modelo Bell-LaPadula.

6.2 Testes Manuais Complementares

Complementarmente, foram realizados testes manuais a todos os comandos disponíveis no cliente, abrangendo operações de leitura, escrita, criação e eliminação de ficheiros, bem como a administração de utilizadores, domínios e níveis de segurança. Estes testes permitiram validar o sistema em contextos variados e com diferentes perfis de utilizador.

Durante os testes, foi verificado que:

- Apenas utilizadores autenticados com credenciais corretas conseguiam aceder ao sistema;
- Ficheiros só eram acessíveis a utilizadores com clearance igual ou superior ao nível de classificação e pertencentes ao domínio associado;
- A escrita em domínios de nível inferior estava restrita a utilizadores com a flag `is_trusted`;

- Apenas administradores podiam aceder a funcionalidades de gestão de utilizadores, níveis e domínios;
- Tentativas de acesso não autorizado resultavam em mensagens claras de erro e sem efeitos colaterais.

6.3 Resultados Observados

Tanto os testes automáticos como os manuais confirmaram o funcionamento correto do sistema, demonstrando o cumprimento das regras definidas pelo modelo de segurança Bell-LaPadula e as suas adaptações implementadas. Cada utilizador teve acesso apenas às funcionalidades compatíveis com o seu perfil, e o sistema comportou-se de forma robusta, segura e previsível em todos os casos testados.

Para visualizar os resultados dos testes automáticos e manuais, deverá ser consultada a área de Anexos, onde é apresentada uma captura de ecrã representativa para cada funcionalidade testada.

7 Limitações e Melhorias Futuras

Apesar da solução implementada cumprir os objetivos principais e apresentar uma adaptação funcional do modelo de segurança Bell-LaPadula (BLP), foram identificadas algumas limitações que, no futuro, poderão ser mitigadas ou resolvidas.

1. **Limitações identificadas**:
   (a) **Ausência de mecanismos de expurgação de conteúdo**: O sistema não verifica nem filtra o conteúdo dos ficheiros em função do seu nível de classificação. Isto significa que dados sensíveis eventualmente propagados por erro não são automaticamente removidos ou protegidos.
   (b) **Interface limitada à linha de comandos (CLI)**: A interação com o sistema é feita exclusivamente por consola, o que pode dificultar a utilização por utilizadores menos experientes ou comprometer a usabilidade em contextos mais exigentes.
   (c) **Gestão de chave simétrica em texto simples**: A chave de sessão simétrica, utilizada para encriptação de mensagens entre cliente e servidor, é temporariamente armazenada em plain text no lado do cliente, o que representa um risco potencial em caso de comprometimento do terminal.
   (d) **Falso erro na operação de eliminação de ficheiros**: O sistema devolve um erro aparente durante o processo de remoção de

ficheiros, mesmo a operação sendo concluída com sucesso. Esta anomalia ainda não foi totalmente diagnosticada.

2. **Melhorias propostas**:

   (a) **Extensão para suporte à leitura partilhada justificada**: Seria interessante permitir que utilizadores autorizados pudessem justificar o acesso a ficheiros de maior nível, fomentando um modelo mais flexível sem comprometer a segurança.

   (b) **Implementação de revogação de utilizadores e domínios**: Atualmente, uma vez atribuídos, os domínios e permissões mantêm-se indefinidamente. O suporte a revogação ativa permitiria uma gestão mais dinâmica e segura dos privilégios.

   (c) **Rotação periódica de chaves de sessão**: A chave de sessão utilizada entre cliente e servidor poderia ser rotacionada automaticamente após um determinado número de interações, reforçando a confidencialidade das comunicações.

   (d) **Investigação e resolução do falso erro na eliminação de ficheiros**: Deverá ser investigada a origem exata deste comportamento inesperado, de forma a garantir um feedback claro e coerente para o utilizador, assegurando que a operação de eliminação de ficheiros reflete corretamente o seu resultado.

Estas limitações não comprometem o funcionamento básico do sistema, mas representam oportunidades importantes de evolução para versões futuras mais robustas, utilizáveis e seguras.

8 Conclusão

A implementação desenvolvida constitui uma aplicação funcional do modelo Bell-LaPadula, adaptado à realidade de gestão segura de ficheiros com base em domínios e classificações. O sistema apresenta uma arquitetura clara, com separação entre cliente e servidor, autenticação mútua, encriptação das comunicações, controlo de acessos e auditoria.

Os testes realizados confirmaram o cumprimento das regras do modelo (no read up, no write down) e da atribuição correta de permissões por domínio. A gestão de sessões com verificação de integridade e desconexão por inatividade demonstra preocupação com boas práticas de segurança.

Apesar de possíveis melhorias, o grupo considera que realizou um bom trabalho, cumprindo os objetivos propostos com uma solução funcional, segura e coerente com os princípios do modelo BLP.

9 Anexos

9.1 Estrutura da Base de Dados (blp_database.json)

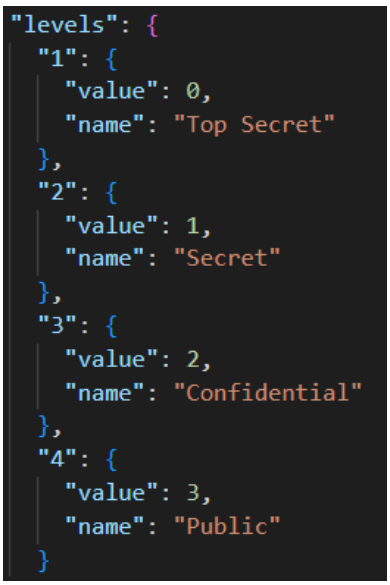


Figura 1: Levels

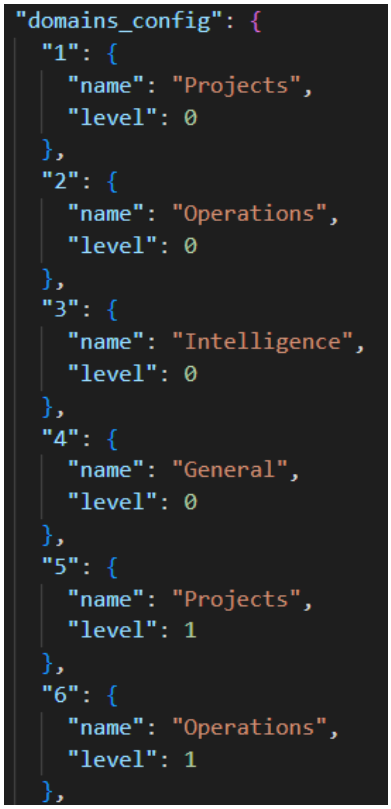


Figura 2: Domains

Figura 3: Users

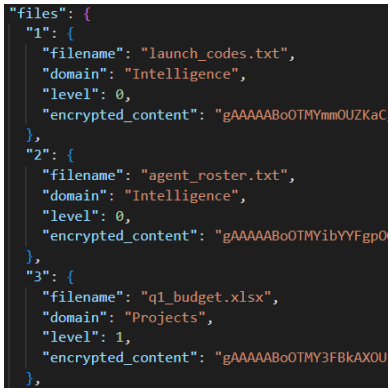


Figura 4: Files

9.2 Funcionamento da Aplicação

Esta secção inclui exemplos visuais do funcionamento da aplicação, obtidos a partir da execução dos diferentes componentes e endpoints.

9.2.1 Autenticação de Utilizador

```
=== BLP Client ===
1. Login
0. Exit

Enter choice: 1
Username: admin
Password:
ℹ️ Info: Secure session established with server.
✅ Success: Logged in as admin

=== BLP Client - Logged in as admin ===
1. User Info
2. List Files
3. Retrieve File
4. Upload File
5. Append to File
6. Delete File
7. List Domains
8. List Security Levels
9. Configure Session Refresh
10. Manage Downloaded Files
11. Logout
0. Exit

Enter choice:
```

Figura 5: Autenticação com sucesso

```
=== BLP Client ===
1. Login
0. Exit

Enter choice: 1
Username: director
Password:
❌ Error: HTTP error from server: 401 UNAUTHORIZED
❌ Error: Invalid credentials

=== BLP Client ===
1. Login
0. Exit

Enter choice:
```

Figura 6: Tentativa de autenticação falhada

9.2.2 Listagem de Ficheiros Acessíveis

```
Available Files:
Filename                       Domain          Level
------------------------------------------------------------
  agent_roster.txt             Intelligence    Top Secret
  employee_reviews.pdf         Operations      Confidential
  launch_codes.txt             Intelligence    Top Secret
  project_phoenix.doc          Projects        Secret
  public_announcement.txt      General         Public
  q1_budget.xlsx               Projects        Secret
  research_paper_draft.txt     General         Confidential
```

Figura 7: Utilizador Admin

```
Available Files:
Filename                       Domain          Level
------------------------------------------------------------
  employee_reviews.pdf         Operations      Confidential
  public_announcement.txt      General         Public
  research_paper_draft.txt     General         Confidential
```

Figura 8: Utilizador HR Manager

9.2.3 Comando Append to File

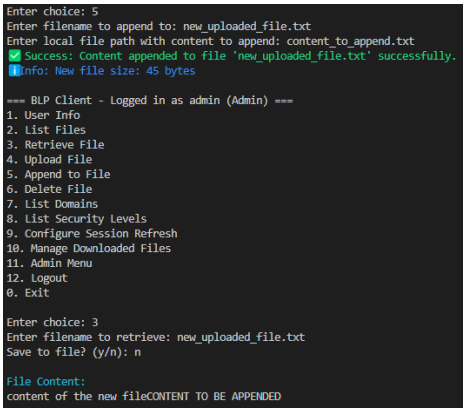


Figura 9: Operação de `append` bem-sucedida

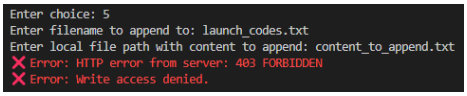


Figura 10: Operação de `append` falhada

9.2.4 Comando Retrieve File

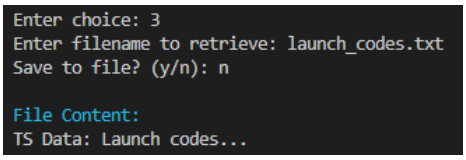


Figura 11: Leitura de ficheiro com sucesso

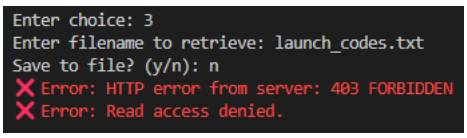


Figura 12: Leitura de ficheiro com falhada

9.2.5 Comando Upload File

```
Enter choice: 4
Enter filename for server storage: new_uploaded_file.txt
Enter base domain name: Intelligence
Enter security level (default: 0): 0
Enter local file path to upload: new_uploaded_file.txt
✅ Success: File 'new_uploaded_file.txt' uploaded successfully to domain 'Intelligence' at level 0.

=== BLP Client - Logged in as admin (Admin) ===
1. User Info
2. List Files
3. Retrieve File
4. Upload File
5. Append to File
6. Delete File
7. List Domains
8. List Security Levels
9. Configure Session Refresh
10. Manage Downloaded Files
11. Admin Menu
12. Logout
0. Exit

Enter choice: 2

Available Files:
Filename                    Domain          Level
-----------------------------------------------------
  agent_roster.txt          Intelligence    Top Secret
  employee_reviews.pdf      Operations      Confidential
  launch_codes.txt          Intelligence    Top Secret
  new_uploaded_file.txt     Intelligence    Top Secret
  project_phoenix.doc       Projects        Secret
  public_announcement.txt   General         Public
  q1_budget.xlsx            Projects        Secret
  research_paper_draft.txt  General         Confidential
```

Figura 13: Upload de ficheiro bem-sucedido

9.2.6 Comando Delete File

```
Enter choice: 6
Enter filename to delete: launch_codes.txt
Are you sure you want to delete 'launch_codes.txt'? (y/n): y
✅ Success: File 'launch_codes.txt' deleted successfully.

=== BLP Client - Logged in as admin (Admin) ===
1. User Info
2. List Files
3. Retrieve File
4. Upload File
5. Append to File
6. Delete File
7. List Domains
8. List Security Levels
9. Configure Session Refresh
10. Manage Downloaded Files
11. Admin Menu
12. Logout
0. Exit

Enter choice: 2

Available Files:
Filename                    Domain          Level
-----------------------------------------------------
  agent_roster.txt          Intelligence    Top Secret
  employee_reviews.pdf      Operations      Confidential
  project_phoenix.doc       Projects        Secret
  public_announcement.txt   General         Public
  q1_budget.xlsx            Projects        Secret
  research_paper_draft.txt  General         Confidential
  upload_test3.txt          Intelligence    Top Secret
```

Figura 14: Eliminação de ficheiro com sucesso

```
Enter choice: 6
Enter filename to delete: agent_roster.txt
Are you sure you want to delete 'agent_roster.txt'? (y/n): y
❌ Error: HTTP error from server: 403 FORBIDDEN
❌ Error: Delete access denied.
```

Figura 15: Eliminação de ficheiro falhada

9.2.7 Listagem dos Domains e dos Security Levels

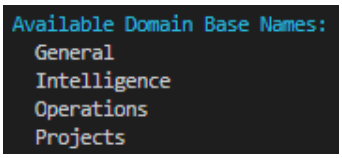


Figura 16: Domains

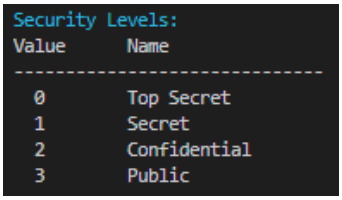


Figura 17: Security Levels

9.2.8 Configuração da Sessão

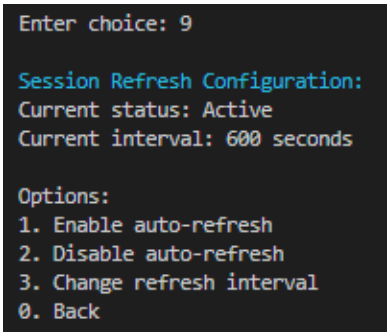


Figura 18: Session Refresh Configuration

9.2.9 Gestão de Downloads

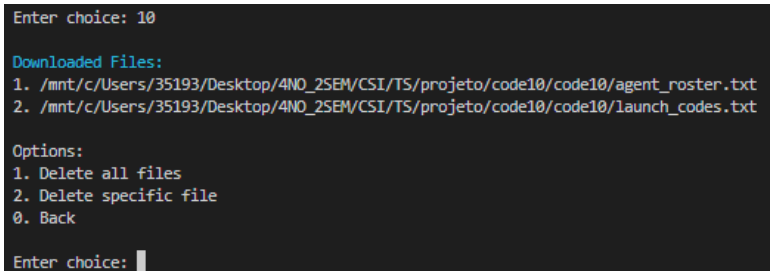


Figura 19: Manage Downloaded Files

9.2.10 Exemplos de Acesso Negado

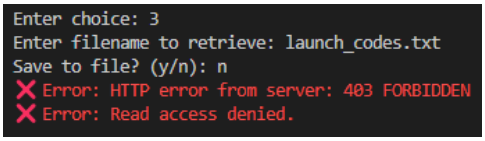


Figura 20: Tentativa de leitura negada (violação de política BLP)

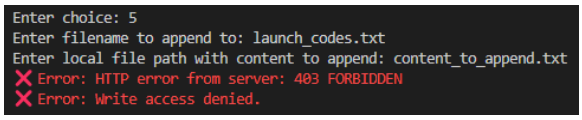


Figura 21: Tentativa de escrita negada (violação de política BLP)

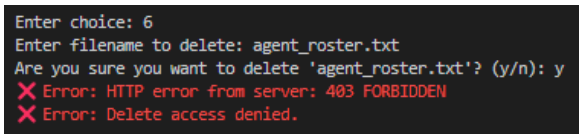


Figura 22: Tentativa de eliminação de um ficheiro negada (violação de política BLP)

9.2.11 Operações Administrativas

```
Enter choice: 11

=== Admin Menu ===
1. List Users
2. Create User
3. Delete User
4. List Domain Configurations
5. Create Domain Configuration
6. Delete Domain Configuration
0. Back to Main Menu

Enter choice:
```

Figura 23: Menu do administrador

```
Enter choice: 1

Users:
Username            Level      Admin    Trusted  Domains
--------------------------------------------------------------------------------
admin               Top Secret Yes      Yes      General, Intelligence, Operations, Projects
analyst_alpha       Top Secret No       No       General, Intelligence
colonel_smith       Secret     No       Yes      General, Operations
diplomat_jones      Secret     No       No       General, Projects
director            Top Secret No       Yes      General, Intelligence, Operations
hr_manager          Confidential No       No       General, Operations
intern_alice        Public     No       No       General, Projects
public_user         Public     No       No       General
researcher_doe      Confidential No       Yes      General, Intelligence, Projects
```

Figura 24: Listagem dos Utilizadores do Sistema

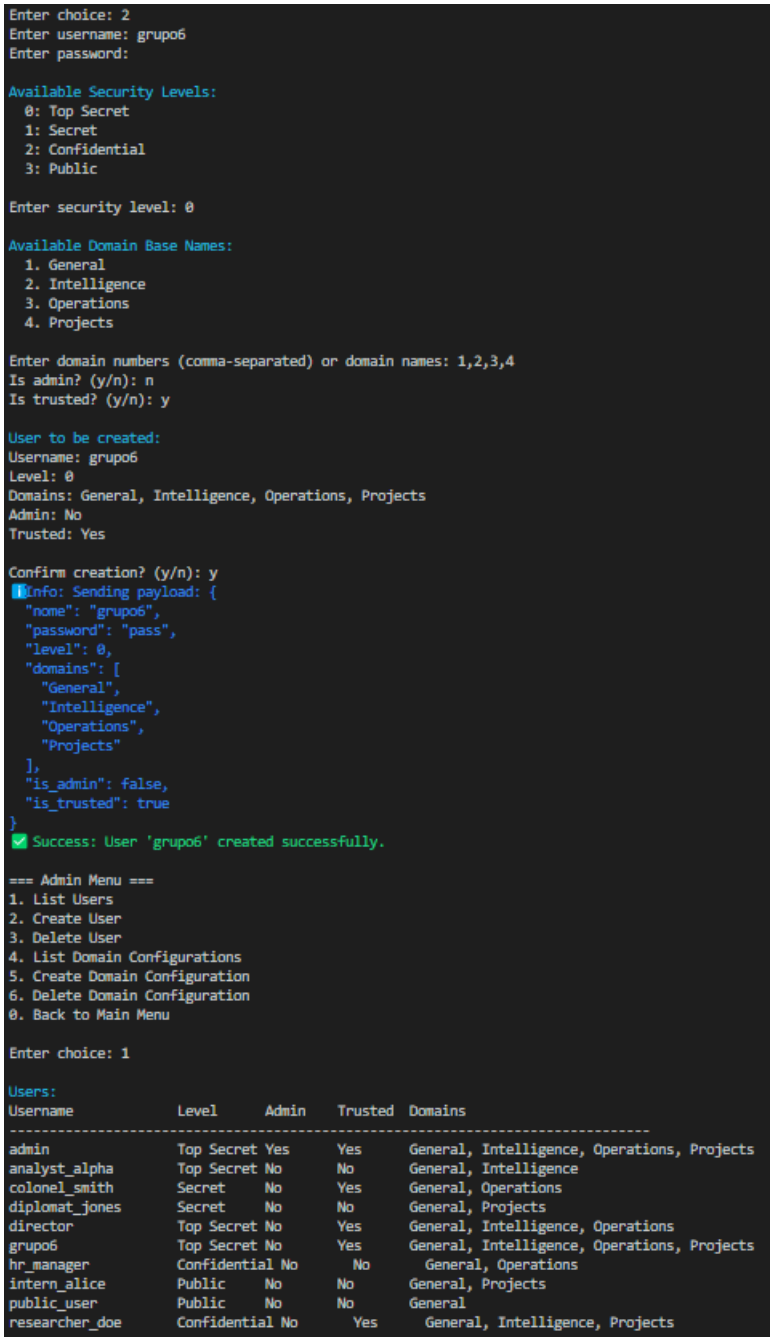


Figura 25: Criação de um Utilizador

```
Enter choice: 3

Users:
  1. admin (Level: 0, Admin: Yes)
  2. analyst_alpha (Level: 0, Admin: No)
  3. colonel_smith (Level: 1, Admin: No)
  4. diplomat_jones (Level: 1, Admin: No)
  5. director (Level: 0, Admin: No)
  6. grupo6 (Level: 0, Admin: No)
  7. hr_manager (Level: 2, Admin: No)
  8. intern_alice (Level: 3, Admin: No)
  9. public_user (Level: 3, Admin: No)
  10. researcher_doe (Level: 2, Admin: No)

Enter user number or username to delete: 6
Are you sure you want to delete user 'grupo6'? (y/n): y
❌ Error: HTTP error from server: 500 INTERNAL SERVER ERROR
❌ Error: Server error during user deletion.

=== Admin Menu ===
1. List Users
2. Create User
3. Delete User
4. List Domain Configurations
5. Create Domain Configuration
6. Delete Domain Configuration
0. Back to Main Menu

Enter choice: 1

Users:
Username            Level      Admin    Trusted  Domains
--------------------------------------------------------------------------------
admin               Top Secret Yes      Yes      General, Intelligence, Operations, Projects
analyst_alpha       Top Secret No       No       General, Intelligence
colonel_smith       Secret     No       Yes      General, Operations
diplomat_jones      Secret     No       No       General, Projects
director            Top Secret No       Yes      General, Intelligence, Operations
hr_manager          Confidential No       No       General, Operations
intern_alice        Public     No       No       General, Projects
public_user         Public     No       No       General
researcher_doe      Confidential No       Yes      General, Intelligence, Projects
```

Figura 26: Eliminação de um Utilizador

```
Enter choice: 4

Domain Configurations:
Base Name           Level
------------------------------
  General           0
  General           1
  General           2
  General           3
  Intelligence      0
  Intelligence      1
  Intelligence      2
  Intelligence      3
  Operations        0
  Operations        1
  Operations        2
  Operations        3
  Projects          0
  Projects          1
  Projects          2
  Projects          3
```

Figura 27: Listar configurações dos Domains

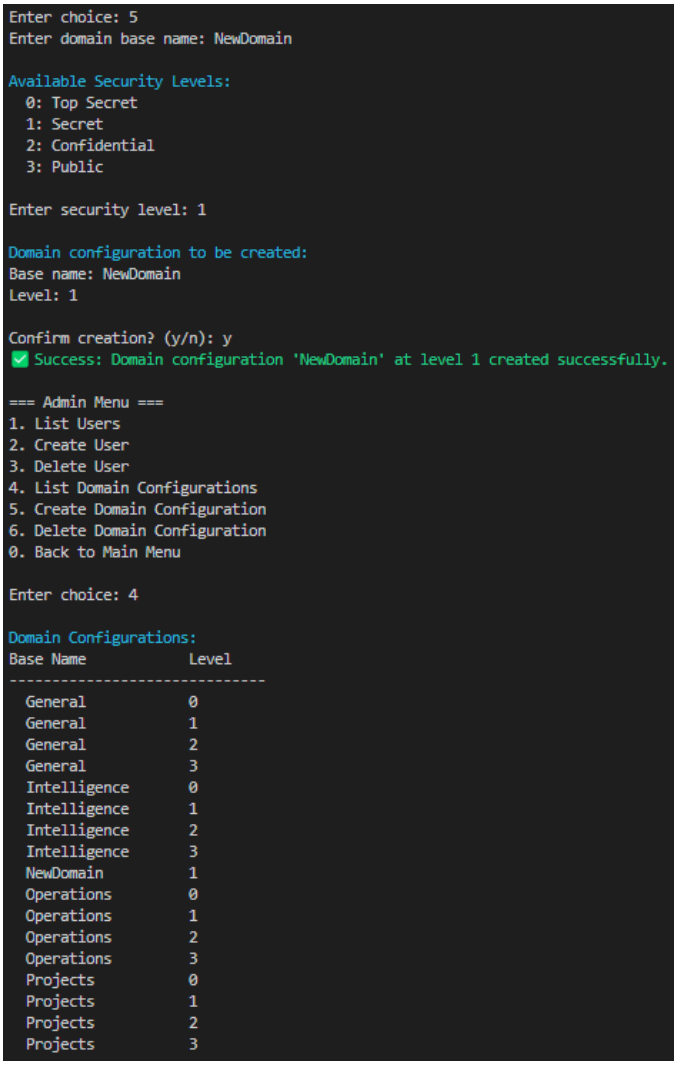


Figura 28: Criação de um Domain

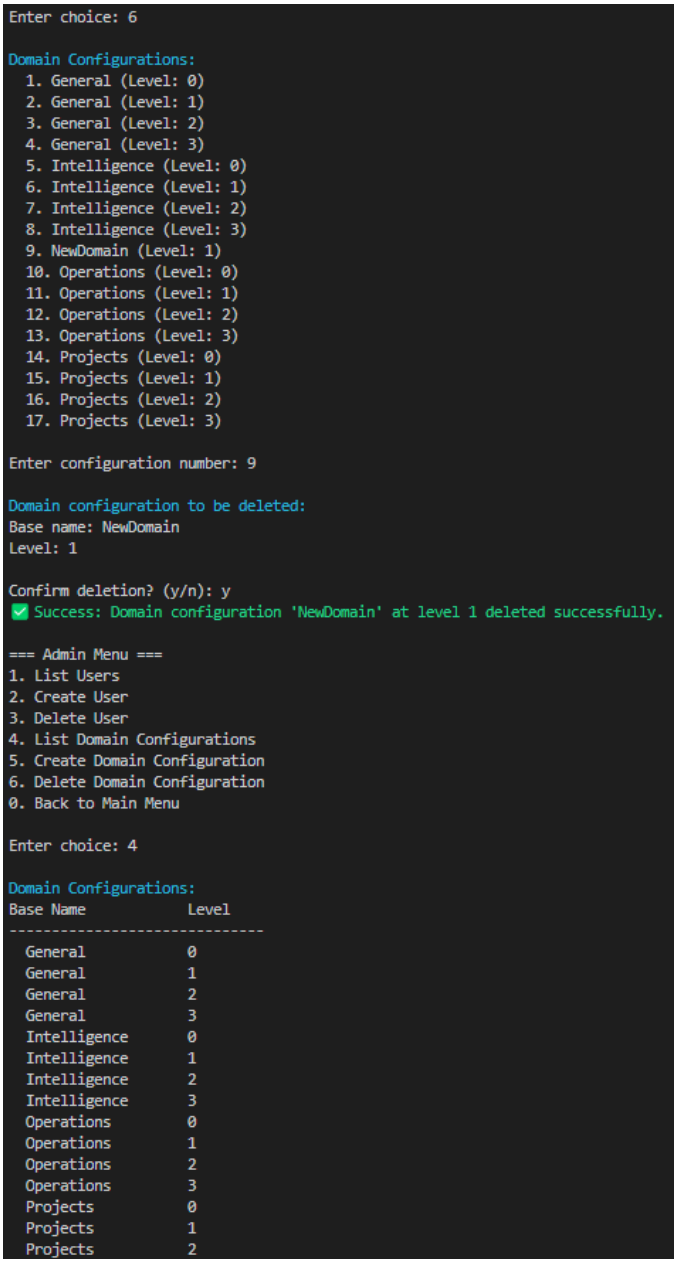


Figura 29: Eliminação de um Domain

9.2.12 Exemplos de Logs

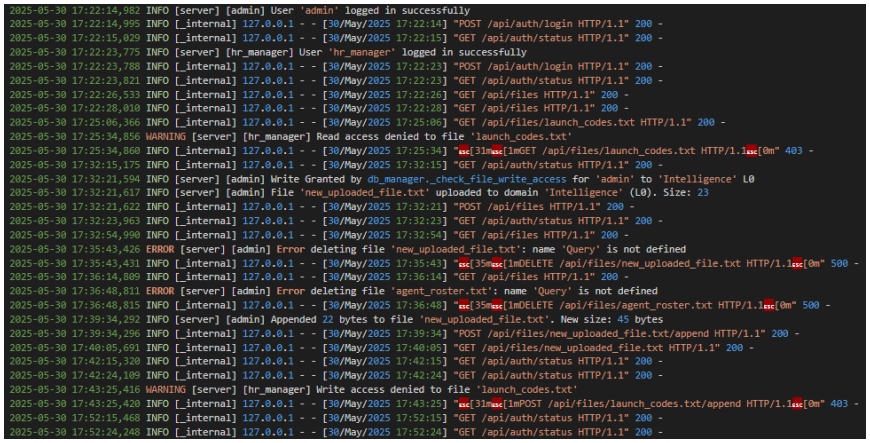


Figura 30: Ficheiro de log geral (`blp_audit.log`)

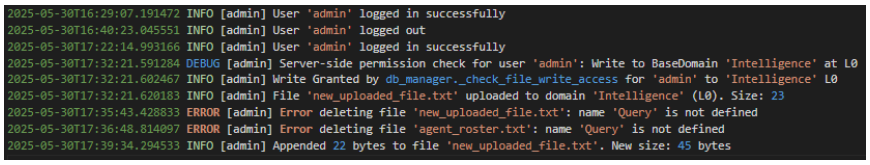


Figura 31: Ficheiro de log individual do utilizador `admin`

9.3 Testes automáticos (/tests/test.py)

```python
class TestClientListFiles(unittest.TestCase):
    def test_login_and_list_files(self):
        print("\nA testar login e listagem de ficheiros")
        cmd = "python3 client.py --ca-cert ./certs/ca.crt --client-cert ./certs/client.crt --client-key ./certs/client.key"
        child = pexpect.spawn(cmd, encoding='utf-8', timeout=10)

        child.expect("Enter choice:")
        child.sendline("1")  # Login

        child.expect("Username:")
        child.sendline("admin")
        child.expect("Password:")
        child.sendline("admin123")

        child.expect("Success: Logged in as admin")
        print("✅ Login bem-sucedido como admin")

        child.expect("Enter choice:")
        child.sendline("2")  # List files

        child.expect("Enter choice:")
        output = child.before

        self.assertIn("Available Files:", output)
        print("✅ Listagem de ficheiros executada com sucesso")

        child.sendline("0")
        child.close()
        print("✅ Teste de listagem concluído com sucesso\n")
```

Figura 32: Estrutura dos testes automáticos

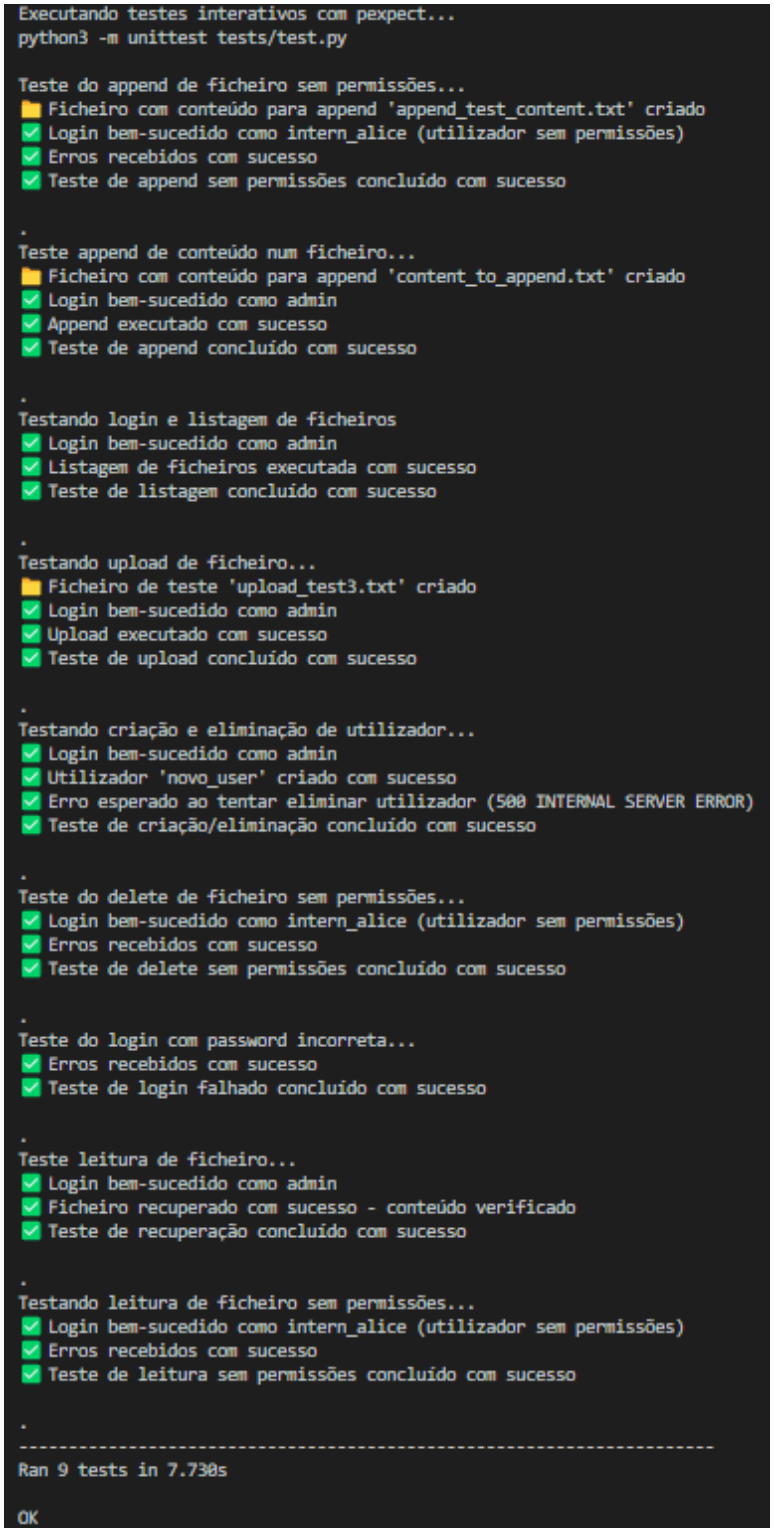


Figura 33: Resultado dos testes automáticos