## Engenharia de Software Seguro

# Relatório - Laboratório 3

Testes (unitários e integração) e aplicação cliente multiplataforma

Grupo 6
Elementos: Hugo Nunes, Miguel Teixeira
Repositório GitHub: c-academy-ess/api-todo-list-manager-grupo-6
Tag(s): lab3
Data: 06-02-2026

## Sumário

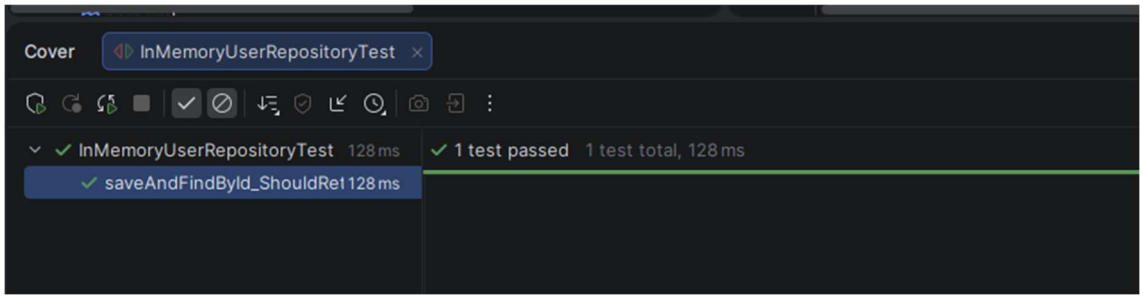
# 1. Testes unitários (JUnit + Mockito) e cobertura

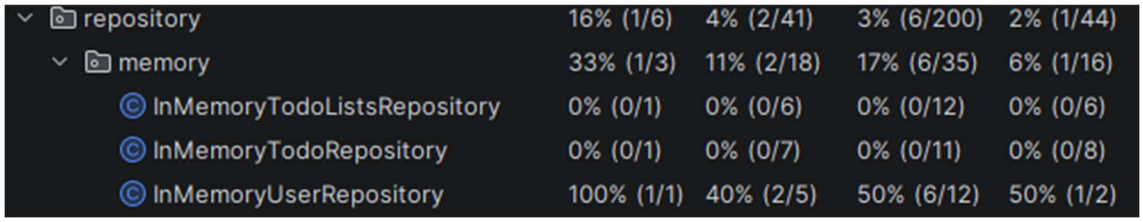
## 1.1 Dependências e estrutura do projeto

O projeto Maven inclui dependências de teste para JUnit Jupiter e Mockito. Os testes devem ficar em src/test/java, seguindo a convenção Maven.

## 1.2 Teste base: InMemoryUserRepositoryTest e cobertura

Foi executado o teste fornecido para o repositório em memória (saveAndFindById_ShouldReturnSavedUser) com medição de coverage.

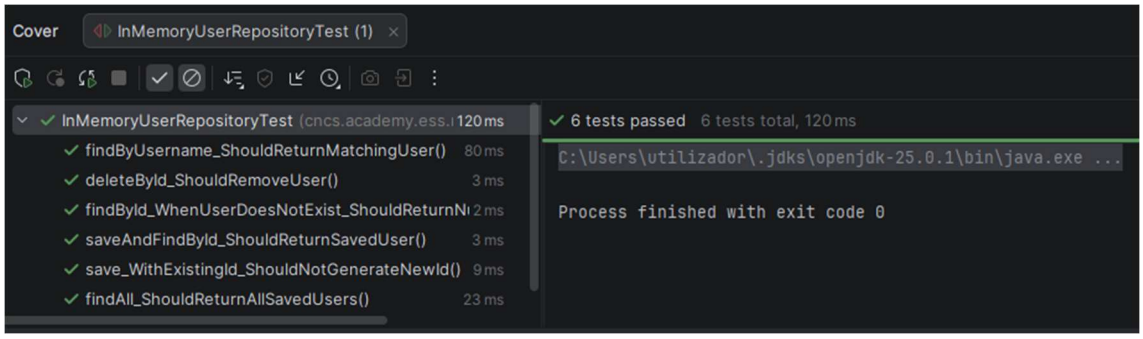


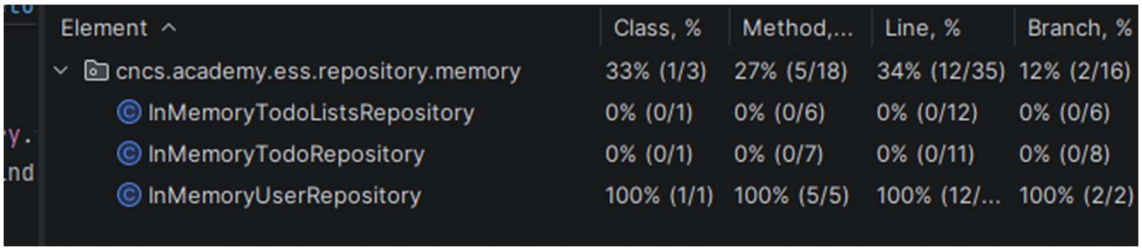




## 1.3 Testes adicionais para aumentar a cobertura

Foram acrescentados testes para cobrir cenários adicionais e aumentar a cobertura do repositório em memória, por exemplo:

- findByUsername_ShouldReturnUserWhenExists
- findByUsername_ShouldReturnNullWhenMissing
- deleteById_ShouldRemoveUser
- save_ShouldAssignIncrementalId
- [outros testes relevantes]





| Element | Class, % | Method,... | Line, % | Branch, % |
|---|---|---|---|---|
| cncs.academy.ess.repository.memory | 33% (1/3) | 27% (5/18) | 34% (12/35) | 12% (2/16) |
| InMemoryTodoListsRepository | 0% (0/1) | 0% (0/6) | 0% (0/12) | 0% (0/6) |
| InMemoryTodoRepository | 0% (0/1) | 0% (0/7) | 0% (0/11) | 0% (0/8) |
| InMemoryUserRepository | 100% (1/1) | 100% (5/5) | 100% (12/... | 100% (2/2) |

### 1.4 Porque usar mocks no TodoUserService e não no InMemoryUserRepository

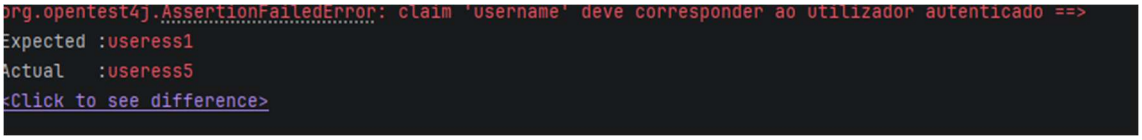
No teste do InMemoryUserRepository pretende-se validar o comportamento real de persistência em memória (estado interno, geração de IDs, pesquisas), pelo que não faz sentido mockar a própria classe sob teste.

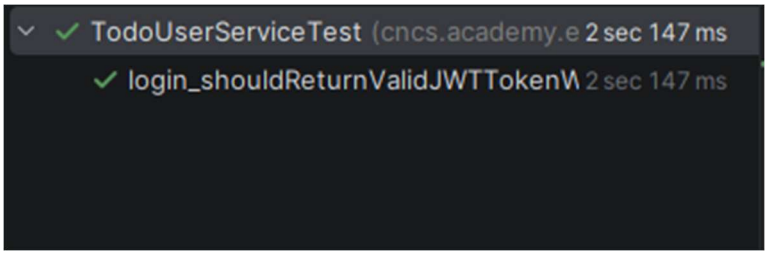
Já no TodoUserService, o objetivo é testar regras de negócio (ex: login, criação de utilizadores) de forma isolada e determinística, sem depender do repositório concreto, da base de dados, ou de detalhes externos. Mocks permitem simular respostas do UserRepository, forçar erros e validar interações.

### 1.5 Teste com Mockito ao método login e validação do JWT

Foi implementado o teste login_shouldReturnValidJWTTokenWhenCredentialsMatch. O teste prepara um UserRepository mock para devolver um utilizador existente e invoca login com credenciais corretas. Depois valida:

(1) o retorno começa por 'Bearer'
(2) o restante é um JWT válido, com assinatura e claims esperadas (issuer, username, issuedAt, expiresAt).



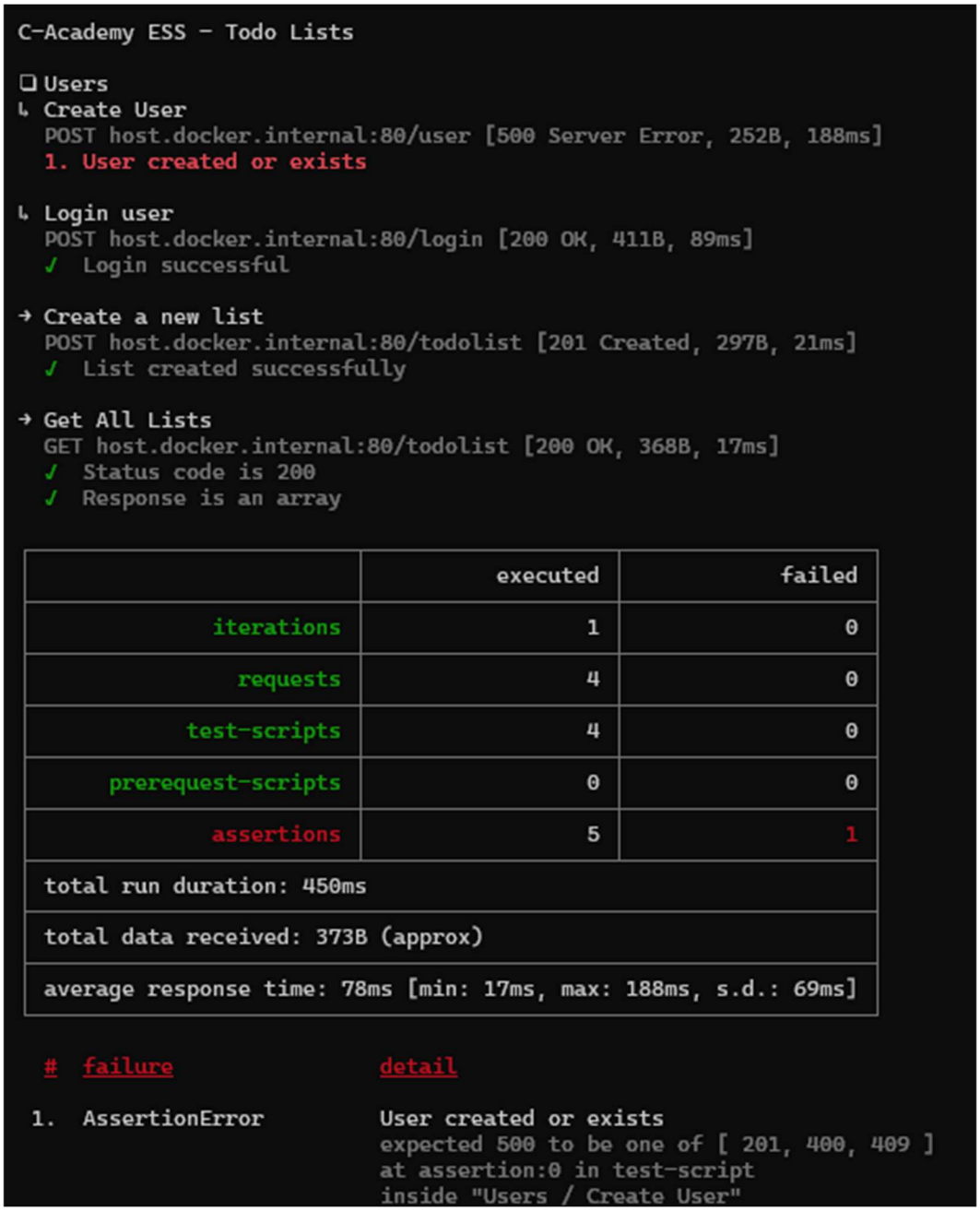




## 2. Testes de integração (Postman + Newman via Docker)

### 2.1 Execução de coleções Postman com Newman

As coleções Postman dos laboratórios anteriores foram adaptadas para execução em linha de comandos usando Newman, via imagem Docker postman/newman. Foi necessário substituir 'localhost' por 'host.docker.internal' e mapear o diretório da coleção para dentro do contentor.

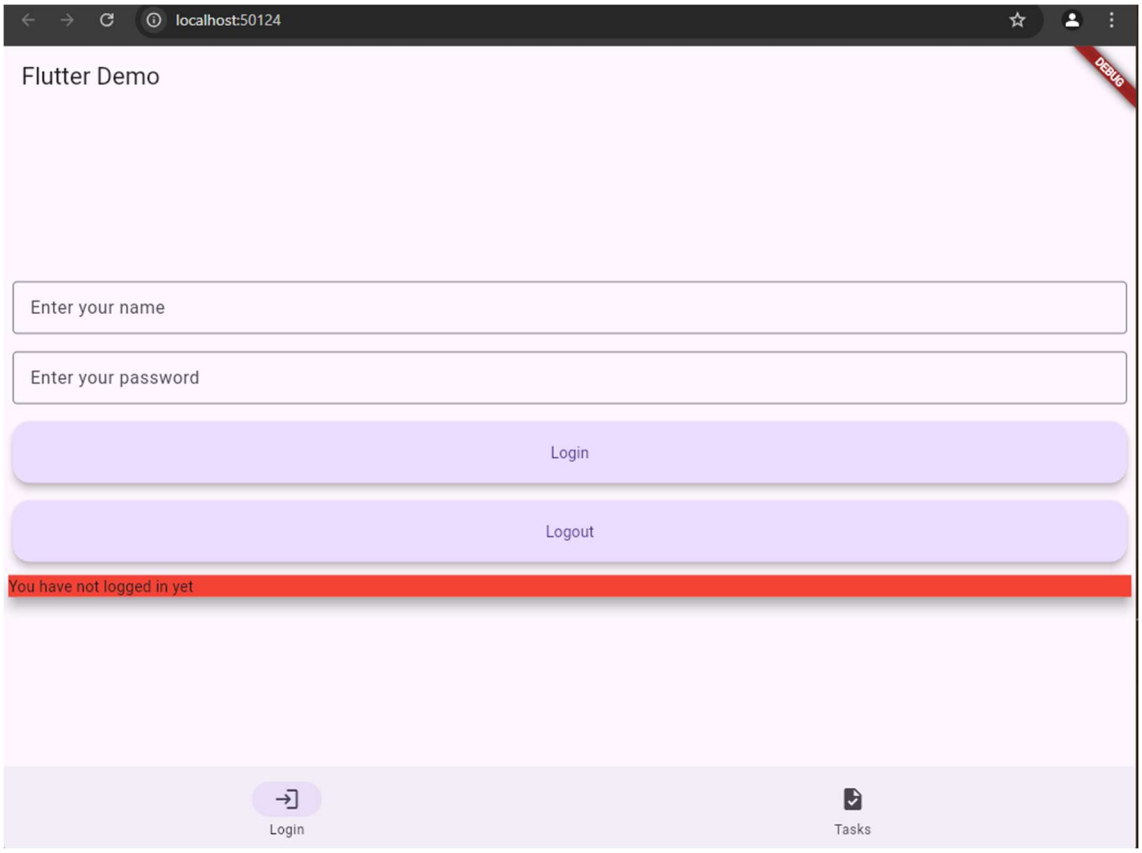
```
# Executar a coleção (no diretório onde está o ficheiro .json)
docker run -v ${PWD}:/etc/newman -t postman/newman run newman-v1.2.json
```




## 3. Aplicação cliente Flutter

### 3.1 Compilação e execução em web

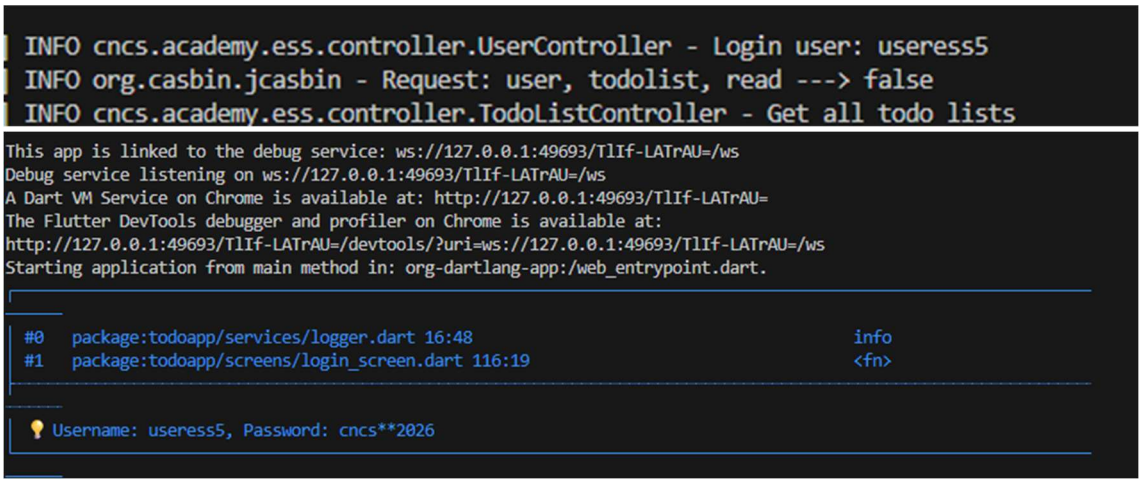
O ambiente Flutter foi instalado e a aplicação base (todoapp) foi compilada e executada em modo web. Foi registado o ecrã inicial de login.

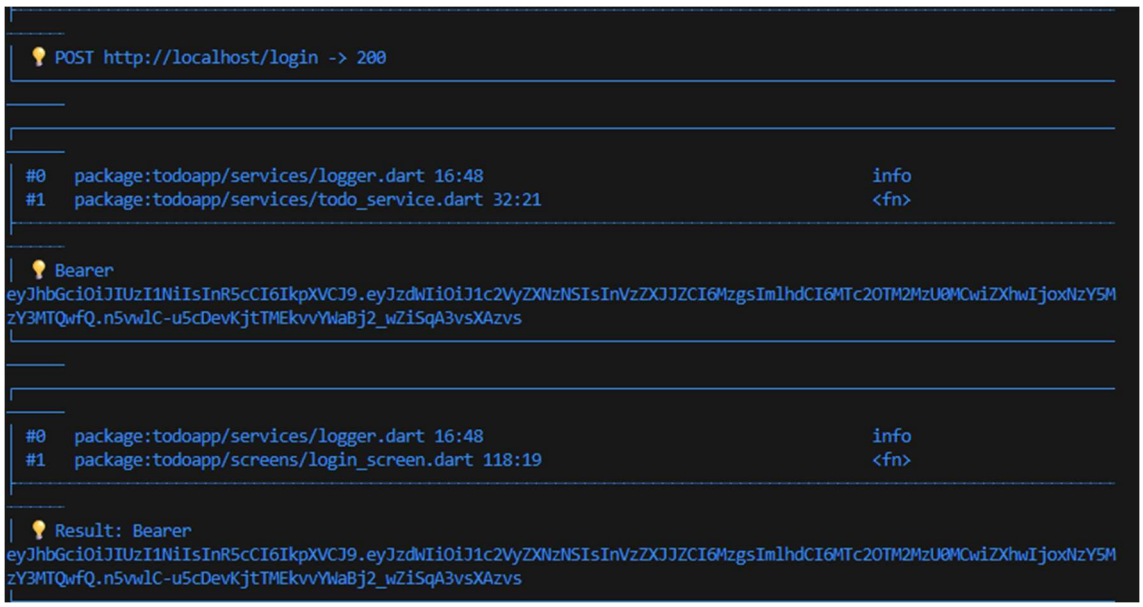


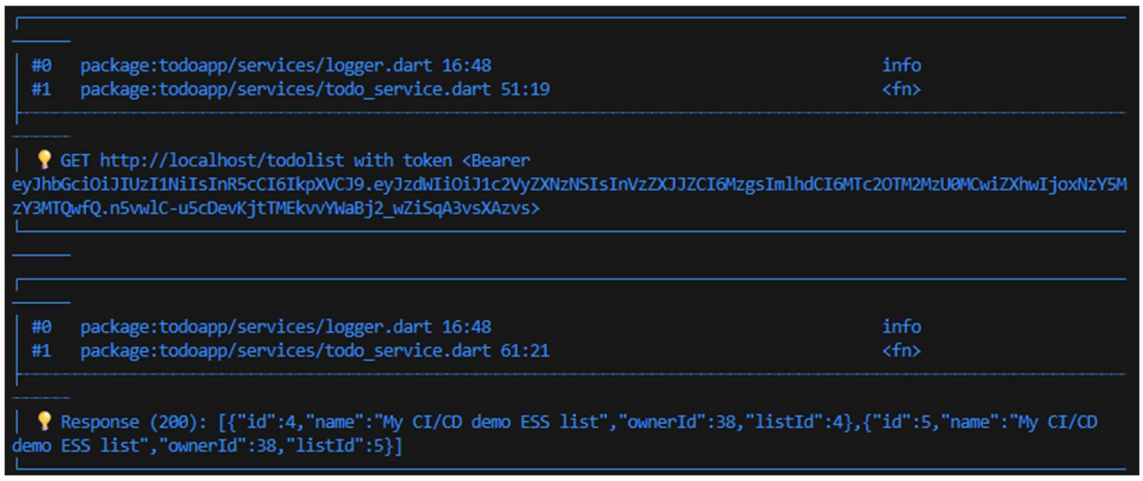

## 3.2 Implementação de todo_service.dart

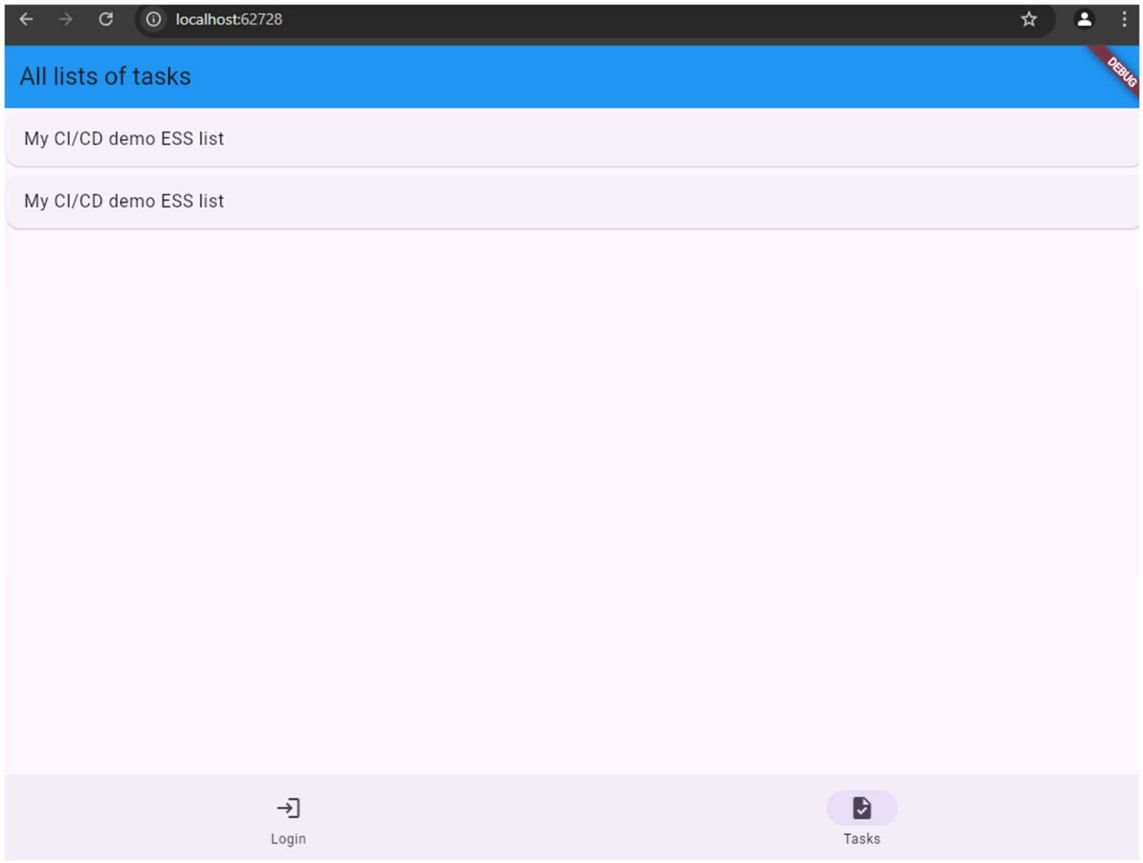
Foram completados os métodos da classe TodoService para chamar a API:

- Usar a variável de ambiente HOST para compor o endereço base
- Reutilizar o token guardado após login no header Authorization
- Mapear respostas JSON para os modelos TodoListModel e TodoModel




POST http://localhost/login -> 200
#0 package:todoapp/services/logger.dart 16:48 info
#1 package:todoapp/services/todo_service.dart 32:21 <fn>
Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyZXNzNSIsInVzZXJJZCI6MzgsImlhdCI6MTc2OTM2MzU0MCwiZXhwIjoxNzY5MzY3MTQwfQ.n5vwlC-u5cDevKjtTMEkvvYWaBj2_wZiSqA3vsXAzvs
#0 package:todoapp/services/logger.dart 16:48 info
#1 package:todoapp/screens/login_screen.dart 118:19 <fn>
Result: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyZXNzNSIsInVzZXJJZCI6MzgsImlhdCI6MTc2OTM2MzU0MCwiZXhwIjoxNzY5MzY3MTQwfQ.n5vwlC-u5cDevKjtTMEkvvYWaBj2_wZiSqA3vsXAzvs


#0 package:todoapp/services/logger.dart 16:48 info
#1 package:todoapp/services/todo_service.dart 51:19 <fn>
GET http://localhost/todolist with token <Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyZXNzNSIsInVzZXJJZCI6MzgsImlhdCI6MTc2OTM2MzU0MCwiZXhwIjoxNzY5MzY3MTQwfQ.n5vwlC-u5cDevKjtTMEkvvYWaBj2_wZiSqA3vsXAzvs>
#0 package:todoapp/services/logger.dart 16:48 info
#1 package:todoapp/services/todo_service.dart 61:21 <fn>
Response (200): [{"id":4,"name":"My CI/CD demo ESS list","ownerId":38,"listId":4},{"id":5,"name":"My CI/CD demo ESS list","ownerId":38,"listId":5}]




### 3.3 Armazenamento seguro do token (flutter_secure_storage)

A biblioteca flutter_secure_storage pode ser usada para guardar o token num armazenamento seguro do sistema (Keychain no iOS, Keystore no Android). O token é gravado após login e lido antes de chamadas futuras, evitando que fique apenas em memória. Em web, a estratégia deve ser avaliada com cuidado (armazenamentos web são mais expostos).

## Referências


- José Simão. Engenharia de Software Seguro - Laboratório 3 (Janeiro 2026).
- JUnit Jupiter (documentação oficial).
- Mockito (documentação oficial).
- Postman Newman (documentação oficial).
- Flutter (documentação oficial).