

Plano de Teste  
Relatório de Teste  
André Moura Pedroso

**Planejamento e Relatório de Teste para um Sistema de  
Armazenamento de Dados para a empresa Chapter**

SENAI/SP  
Jun - 2022

## Sumário

INTRODUÇÃO .....	1
1.Modelo .....	1
2.Resumo sobre o sistema .....	1
3.Funcionalidades do sistema .....	1
ESCOPO .....	1
1.Funcionalidades do sistema .....	1
2.Testes .....	2
OBJETIVOS .....	2
1.Objetivo Geral .....	2
2.Objetivo Específico 1 .....	2
3.Objetivo Específico 2 .....	2
ESPECIFICAÇÃO DO PROJETO DE TESTE .....	2
1.Requisitos de teste .....	2
1.1.Teste de Integração 1 – Retorno de Usuário Inválido. ....	2
1.2.Teste de Integração 2 – Retorno de Token no método de Login. ....	3
ESPECIFICAÇÃO DO PROCEDIMENTO DE TESTES .....	3
1.Ferramentas .....	3
2.Estratégia de Teste .....	4
2.1.Projeto SENAI-UC14-E4 (ChapterApi) .....	4
2.2.Projeto TestXunitChapter .....	5
3.Sistema utilizado .....	6
4.Equipe .....	7
CRONOGRAMA .....	8
DESENHO DE TESTE .....	9
1.TestXunitChapter .....	9
1.1.Classe LoginControllerTest .....	9
RELATÓRIO DE TESTE .....	11
1.Diário de Teste .....	11
1.1. Instalação e preparação do ambiente .....	11
1.2. Preparação XUnit .....	12
1.3. Classe LoginControllerTest – Teste de Integração 1 - Método LoginController_Retornar_Usuario_Invalido .....	13
1.4. Classe LoginControllerTest – Teste de Integração 2 - Método Login_Controller_Retornar_Token .....	13

1.5. Execução do Teste.....	14
1.6. Correção de falhas .....	15
1.7. Execução do Teste 2, Organização dos resultados e Análise dos dados .....	15
2. Relatório de incidentes .....	16
3. Resumo de teste.....	16
3.1. Teste de Integração 1 – Retorno de Usuário Inválido – Caso de Teste 1 .....	16
3.2. Teste de Integração 2 – Retorno de Token no método de Login – Caso de Teste 1. ....	17
CONCLUSÃO.....	19

# **INTRODUÇÃO**

## **1.Modelo**

Armazenamento de Dados de Livros para a Biblioteca Chapter.

## **2.Resumo sobre o sistema**

O sistema se trata do armazenamento de dados de livros que vão estar tombados em uma Biblioteca. Os livros vão possuir informações digitalizadas como título, quantidade de páginas e disponibilidade. Haverá também um controle de usuários que poderão ler ou alterar as informações de acordo com o seu tipo.

## **3.Funcionalidades do sistema**

Usuários comuns vão poder visualizar as informações para fazer uma solicitação. Usuários administradores vão poder cadastrar, alterar e deletar as informações.

# **ESCOPO**

## **1.Funcionalidades do sistema**

O sistema apresenta a entidade Livros para conter dados que terão os atributos Id, Título, QuantidadePaginas e Disponivel. Usuários comuns poderão ler as informações. Usuários administradores poderão ler, cadastrar, alterar e deletar as informações. Usuários terão os atributos Id, Email, Senha e Tipo. O Login precisará do Email e Senha que serão comparados com os usuários cadastrados no banco de dados. Caso o usuário esteja cadastrado será gerada uma chave de identificação.

## 2. Testes

Os testes deste plano vão verificar apenas o sistema de Login de usuários.

### **OBJETIVOS**

#### 1. Objetivo Geral

Testar as funcionalidades do sistema que se referem ao sistema de Login de usuários.

#### 2. Objetivo Específico 1

Testar o retorno de usuário inválido.

#### 3. Objetivo Específico 2

Testar a geração de uma chave de identificação de usuário conhecida como token.

### **ESPECIFICAÇÃO DO PROJETO DE TESTE**

#### 1. Requisitos de teste

##### 1.1. Teste de Integração 1 – Retorno de Usuário Inválido.

Um repositório será simulado com dados “mocados”, desta forma, o banco não precisará ser consultado fazendo uma verificação apenas dos códigos. Os dados de Login serão informados na elaboração do teste para que o resultado da aplicação do método que está utilizando o repositório simulado retorne uma resposta de não autorizado (401).

### *1.1.1.Caso de teste 1 e Cenário esperado*

Email = "email@email.com",

Senha = "123",

Retorno = Unauthorized (401) response

### 1.2.Teste de Integração 2 – Retorno de Token no método de Login.

Um repositório será simulado para que o banco de dados não precise ser consultado, fazendo uma verificação apenas nos códigos. Os dados utilizados serão:

Email = "email@email.com";

Senha = "1234";

Tipo = "0";

Os dados para identificação de usuário serão fornecidos conforme o caso de uso 1, serão aplicados no método de Login para geração de um token que deverá ser comparado com a configuração feita nos códigos para a geração de uma chave.

### *1.2.1.Caso de teste 1 e Cenário esperado*

Email = "email@email.com";

Senha = "1234";

Retorno = token.

## **ESPECIFICAÇÃO DO PROCEDIMENTO DE TESTES**

### 1.Ferramentas

- a) Visual Studio 2022.
- b) Moq.

- c) .NET versão 6.0.
- d) Projeto de Teste do xUnit.

## 2.Estratégia de Teste

O Projeto a ser testado é uma API desenvolvida para o sistema Armazenamento de Dados de Livros para a Biblioteca Chapter. O que será testado se refere as funcionalidades de Login que utiliza a entidade de armazenamento de dados de Usuários com os atributos Email, Senha e Tipo. O Login recebe as informações de Email e Senha que quando confirmadas com um usuário cadastrado retorna uma chave de identificação para acesso à Biblioteca (token).

### 2.1.Projeto SENAI-UC14-E4 (ChapterApi)

O projeto pode ser consultado no link: Disponível em: <<https://github.com/Andpedroso/SENAI-UC14-E4>> Acessado em 13 jun de 2022.

#### 2.1.1.Classe LoginContoller

Contém os métodos de verificação de usuário cadastrado para geração de chave de identificação, caso o usuário não esteja cadastrado o método retorna usuário inválido, caso esteja cadastrado, um token é gerado para acessar as informações da Biblioteca.

#### 2.1.2.Classe LoginViewModel

Contém o modelo de preenchimento de usuário para Login com Email e Senha. Ambos devem ser campos obrigatórios.

#### 2.1.3.Classe Usuario

Modelo de usuário cadastrado na entidade do banco, contém os atributos Id, Email, Senha e Tipo.

## 2.2. Projeto TestXunitChapter

### 2.2.1. Classe *LoginControllerTest*

Esta classe deverá ser pública para não restringir o acesso a outros projetos da mesma solução e irá conter o Teste de Integração 1 e 2.

#### 2.2.1.1. *Teste de Integração 1 – Retorno de Usuário Inválido.*

O método `LoginController_Retornar_Usuario_Invalido` contém no Arrange um repositório fake criado com um método de simular dados que seriam do banco de dados. Os dados de Login serão de Email e Senha e estão definidos no Caso de Teste 1 (1.1.1) do Teste de Integração 1. A classe `LoginController` que contém os métodos analisados no teste, usará as informações do repositório fake. No Act o controller definido com a classe `LoginController` no Arrange vai utilizar os dados de Email e Senha definidos no Arrange. O Assert vai utilizar um método que retorna `Unauthorized (401)` caso o resultado definido no Act seja o esperado.

#### 2.2.1.2. *Teste de Integração 2 – Retorno de Token no Método de Login*

O Método `Login_Controller_Retornar-Token` contém um repositório fake com a estrutura da entidade `Usuários` simulados (Email, Senha e Tipo). Os dados do usuário do repositório fake estão definidos no Caso de Teste 1 (1.2.1) do Teste de Integração 2 e vão ser passados no Arrange. A configuração responsável pela validação de usuário da classe `LoginController` será utilizada com os dados simulados para a classe `LoginViewModel` que receberia as informações de Login de usuário. A classe `LoginController` que contém os métodos analisados no teste, usará as informações do repositório fake. No Act o controller definido com a classe `LoginController` no Arrange vai utilizar os dados de Email e Senha definidos no Arrange, quando validado o token será gerado e



deverá ser utilizado no método da JwtSecurityTokenHandler para gerar o token. O Assert vai utilizar um método para verificar se o que está definido na configuração de validação do método está de acordo com o token gerado no Act.

### 3. Sistema utilizado

**Processador:** Intel(R) Core(TM) i5-3337U CPU @ 1.80 GHz

**RAM instalada:** 8.00 GB (utilizável: 7.90 GB)

**Tipo de sistema:** Sistema operacional de 64 bits, processador x64

#### 4. Equipe

**André Moura Pedroso**

---

36 anos.

Desenvolvedor Full Stack.

---

Análise e Desenvolvimento de Sistemas – Faculdade Descomplica.

Desenvolvedor Full Stack – SENAI.

---

Experiências:

- App para Android (Java e Flutter/Dart);
- Games (Desktop e Android) com Pixel Art (Java);
- Web Pages (Angular/Typescript);
- APIs (C#);
- Banco de Dados (SQL/Oracle e Microsoft).

## CRONOGRAMA

Instalação e preparação do ambiente	07/06/2022
Preparação xUnit	07/06/2022
Classe LoginController – Teste de Integração 1 – Retorno de Usuário Inválido.	07/06/2022
Classe LoginController – Teste de Integração 2 – Retorno de Token no Método de Login.	07/06/2022
Execução do Teste	07/06/2022
Correção de falhas	07/06/2022
Execução do Teste 2	07/06/2022
Organização dos resultados	10/06/2022
Análise dos dados	10/06/2022
Relatório de incidentes	10/06/2022
Resumo de teste	10/06/2022
Conclusão	10/06/2022

## DESENHO DE TESTE

### 1.TestXunitChapter

Projeto criado com o xUnit para o desenvolvimento dos testes necessários para o projeto.

#### 1.1.Classe LoginControllerTest

Classe criada para todos os métodos de testes do Teste de Integração 1 e 2.

LoginControllerTest.cs

```
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Moq;
using SENAI_UC14_E4.Controllers;
using SENAI_UC14_E4.Interfaces;
using SENAI_UC14_E4.Models;
using SENAI_UC14_E4.ViewModels;

namespace TestXunitChapter.Controllers
{
    public class LoginControllerTest
    {
        //TESTE DE INTEGRAÇÃO 1 - RETORNO DE USUÁRIO INVÁLIDO
        [Fact]
        public void LoginController_Retornar_Usuario_Invalido()
        {
            //Arrange

            var fakeRepository = new Mock<IUsuarioRepository>();

            fakeRepository.Setup(x => x.Login(It.IsAny<string>(),
            It.IsAny<string>())).Returns((Usuario)null);

            LoginViewModel dadosLogin = new LoginViewModel();

            dadosLogin.Email = "email@email.com";

            dadosLogin.Senha = "123";

            var controller = new LoginController(fakeRepository.Object);
```

```

        //Act

        var resultado = controller.Login(dadosLogin);

        //Assert

        Assert.IsType<UnauthorizedObjectResult>(resultado);
    }

    //TESTE DE INTEGRAÇÃO 2 - RETORNO DE TOKEN NO MÉTODO DE LOGIN
    [Fact]
    public void Login_Controller_Retornar_Token()
    {
        //Arrange

        Usuario usuarioRetorno = new Usuario();

        usuarioRetorno.Email = "email@email.com";

        usuarioRetorno.Senha = "1234";

        usuarioRetorno.Tipo = "0";

        var fakeRepository = new Mock<IUsuarioRepository>();

        fakeRepository.Setup(x => x.Login(It.IsAny<string>(),
        It.IsAny<string>()))).Returns(usuarioRetorno);

        string issuerValidacao = "chapter.webapi";

        LoginViewModel dadosLogin = new LoginViewModel();

        dadosLogin.Email = "email@email.com";

        dadosLogin.Senha = "1234";

        var controller = new LoginController(fakeRepository.Object);

        //Act

        OkObjectResult resultado =
        (OkObjectResult)controller.Login(dadosLogin);

        string token = resultado.Value.ToString().Split(' ')[3];

        var jwtHandler = new JwtSecurityTokenHandler();

        var tokenJwt = jwtHandler.ReadJwtToken(token);

        //Assert

        Assert.Equal(issuerValidacao, tokenJwt.Issuer);
    }
}

```

# RELATÓRIO DE TESTE

## 1. Diário de Teste

### 1.1. Instalação e preparação do ambiente

A Figura 1, 2 e 3 representam o resultado da preparação do ambiente considerando as ferramentas necessárias para o projeto. Após a criação de um projeto de teste na mesma solução, foi necessário referenciar o projeto com o projeto a ser testado.

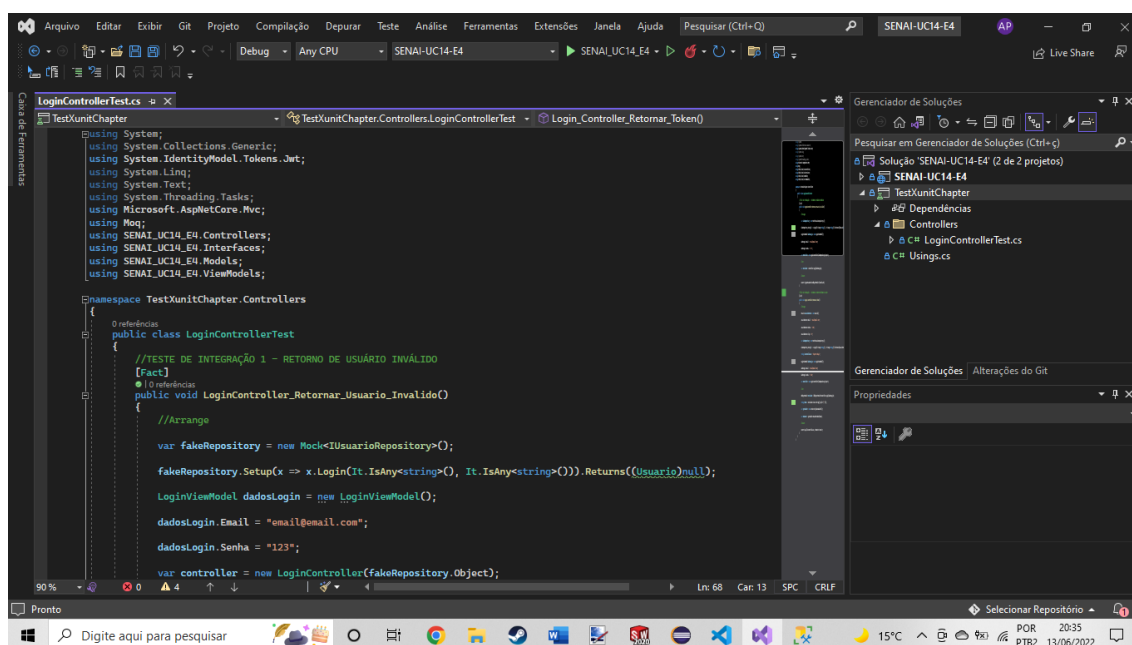


Figura 1 – Visual Studio 2022 instalado e rodando.

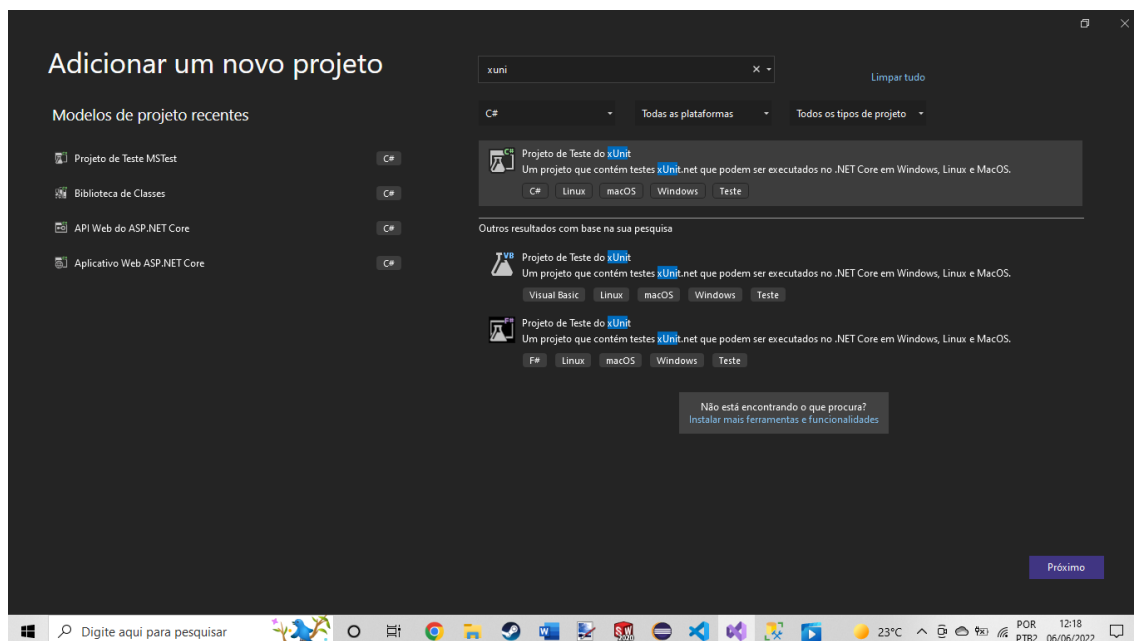


Figura 2 – criando projeto do xUnit.

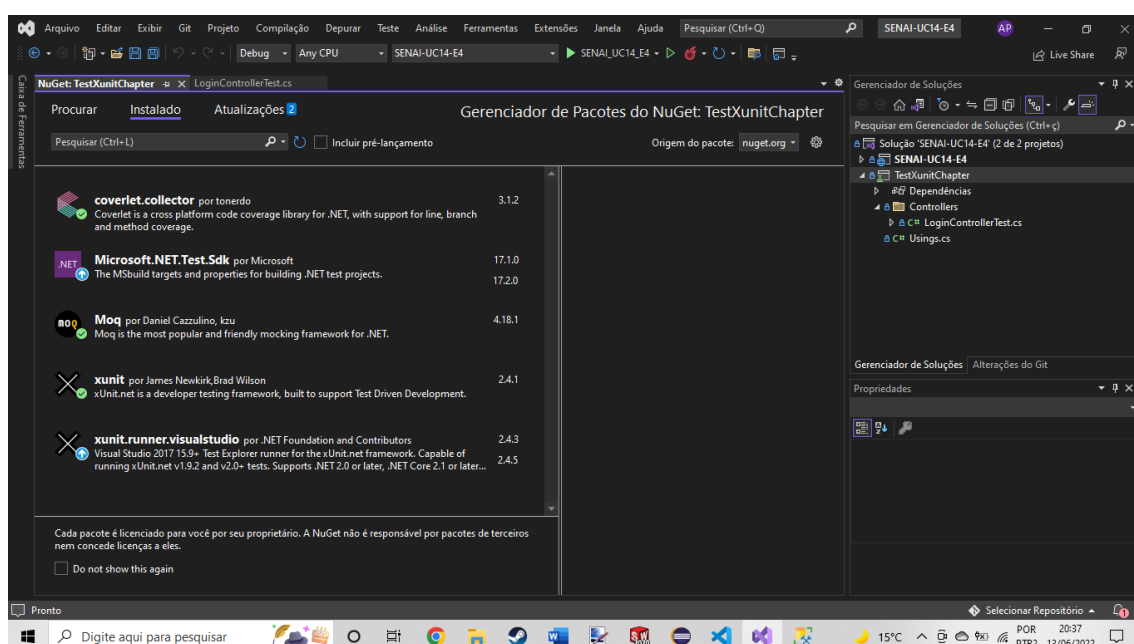


Figura 3 – Moq e extensões do xUnit.

## 1.2. Preparação XUnit

O projeto de teste do xUnit foi criado conforme pode ser visualizado na Figura 4. Contém a classe pública `LoginControllerTest`. A classe `LoginControllerTest` contém o Teste de Integração 1 e 2. O Teste de Integração

1 contém o método `LoginController_Retornar_Usuario_Invalido`. O Teste de Integração 2 contém o método `Login_Controller_Retornar_Token`.

### 1.3. Classe `LoginControllerTest` – Teste de Integração 1 - Método `LoginController_Retornar_Usuario_Invalido`

O método `LoginController_Retornar_Usuario_Invalido` foi desenhado da seguinte forma:

```
public void LoginController_Retornar_Usuario_Invalido()
{
    //Arrange

    var fakeRepository = new Mock<IUsuarioRepository>();

    fakeRepository.Setup(x => x.Login(It.IsAny<string>(),
    It.IsAny<string>()))).Returns((Usuario)null);

    LoginViewModel dadosLogin = new LoginViewModel();
    dadosLogin.Email = "email@email.com";
    dadosLogin.Senha = "123";

    var controller = new LoginController(fakeRepository.Object);

    //Act

    var resultado = controller.Login(dadosLogin);

    //Assert

    Assert.IsType<UnauthorizedObjectResult>(resultado);
}
```

### 1.4. Classe `LoginControllerTest` – Teste de Integração 2 - Método `Login_Controller_Retornar_Token`

O método `Login_Controller_Retornar_Token` foi desenhado da seguinte forma:

```
public void Login_Controller_Retornar_Token()
{
    //Arrange

    Usuario usuarioRetorno = new Usuario();
    usuarioRetorno.Email = "email@email.com";
```



```

        usuarioRetorno.Senha = "1234";
        usuarioRetorno.Tipo = "0";
        var fakeRepository = new Mock<IUsuarioRepository>();
        fakeRepository.Setup(x => x.Login(It.IsAny<string>(),
        It.IsAny<string>()))).Returns(usuarioRetorno);

        string issuerValidacao = "chapter.webapi";
        LoginViewModel dadosLogin = new LoginViewModel();
        dadosLogin.Email = "email@email.com";
        dadosLogin.Senha = "1234";
        var controller = new LoginController(fakeRepository.Object);

        //Act
        OkObjectResult resultado =
        (OkObjectResult)controller.Login(dadosLogin);

        string token = resultado.Value.ToString().Split(' ')[3];
        var jwtHandler = new JwtSecurityTokenHandler();
        var tokenJwt = jwtHandler.ReadJwtToken(token);

        //Assert
        Assert.Equal(issuerValidacao, tokenJwt.Issuer);
    }

```

### 1.5. Execução do Teste

Ocorreu um erro no método de teste `Login_Controller_Retornar_Token` (Figura 4).

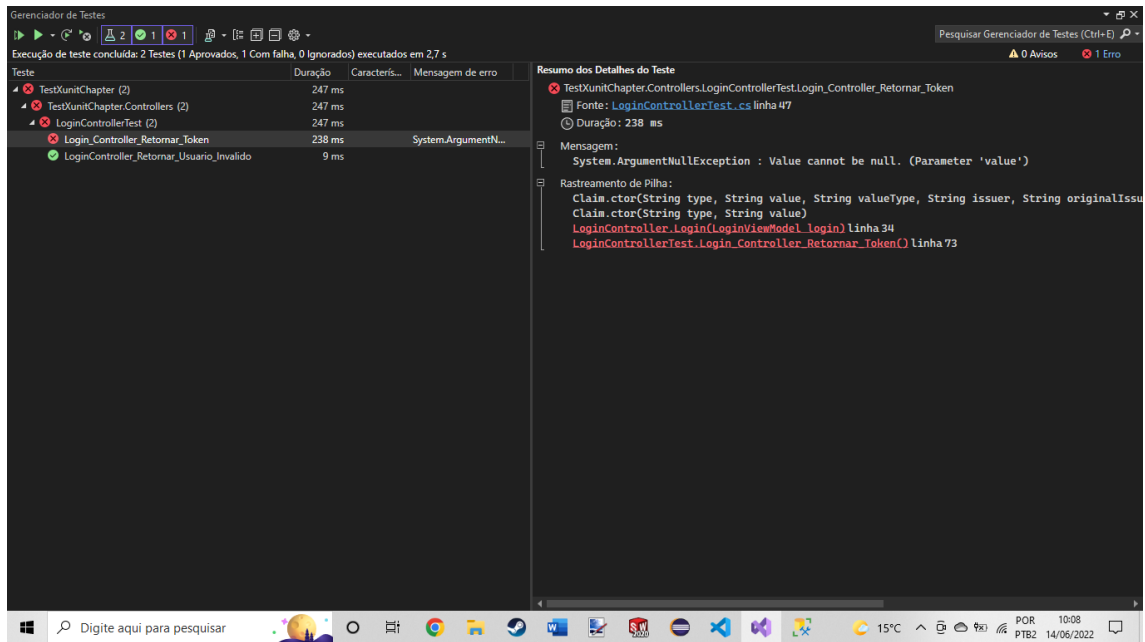


Figura 4 – teste realizado com comportamento inesperado.

## 1.6. Correção de falhas

Com a depuração dos códigos foi identificado o erro e corrigido (Figura 5).

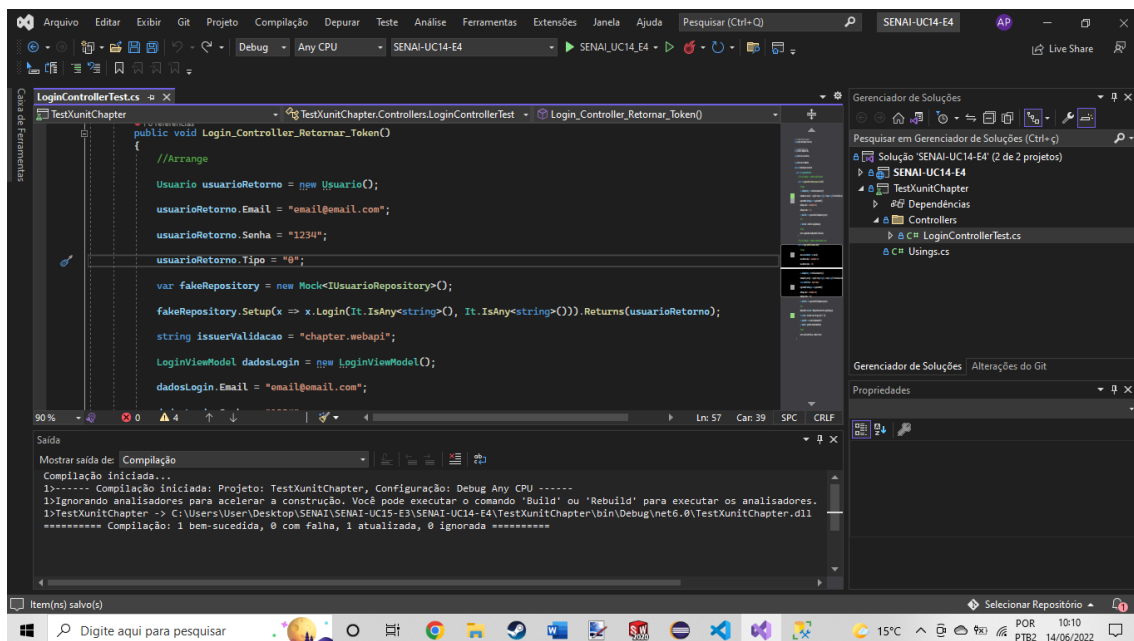


Figura 5 – erro na estrutura de usuário, faltou o atributo Tipo.

## 1.7. Execução do Teste 2, Organização dos resultados e Análise dos dados

Ao executar o teste depois da correção feita, o resultado do teste foi o esperado. A Figura 6 representa o resultado dos testes que ao todo foram 2. Ambos os testes foram aprovados. Os métodos de teste verificaram a funcionalidade dos métodos avaliados no projeto analisado e retornaram aprovação.

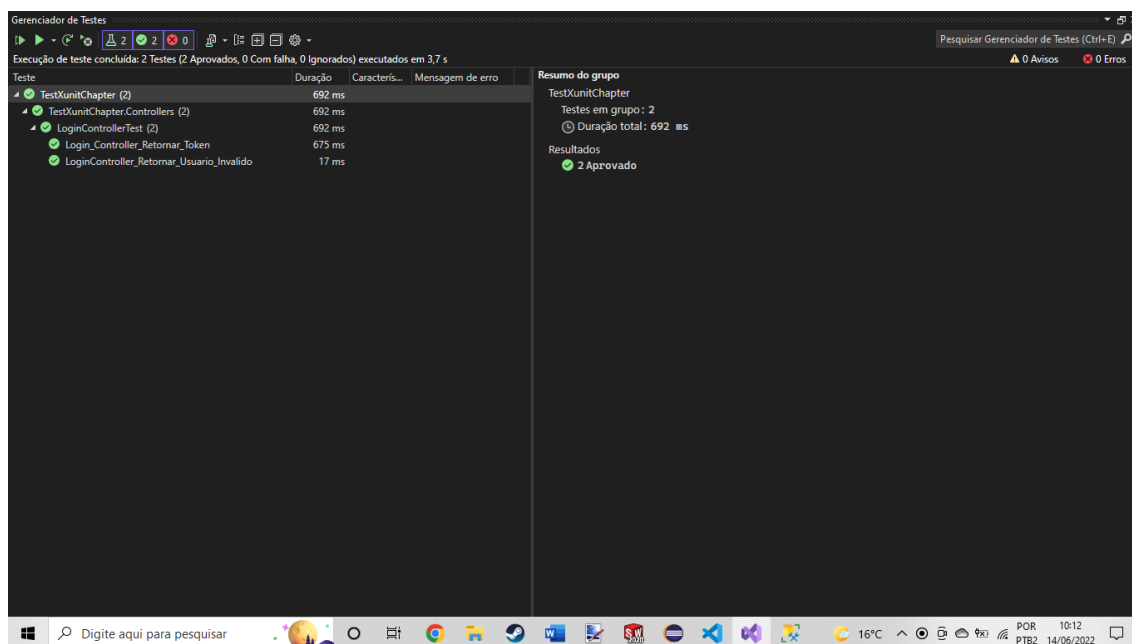


Figura 6 – resultado geral dos testes aplicados no projeto.

## 2. Relatório de incidentes

O incidente ocorreu na execução do método `Login_Controller_Retornar_Token` que avalia o método de geração de Token da Classe `LoginController` do projeto avaliado. Foi verificado que os dados de usuários precisam de três atributos: Email, Senha e Tipo a serem passados para o repositório fake criado para o teste. A correção pode ser verificada na Figura 5.

## 3. Resumo de teste

### 3.1. Teste de Integração 1 – Retorno de Usuário Inválido – Caso de Teste 1

Descrição: Email = “email@email.com”, Senha = “123”.

Resultado Esperado: Unauthorized (401) response.

Resultado do Teste: Figura 7.

Nota: O método testado foi analisado pelo método de teste que retornou aprovação.

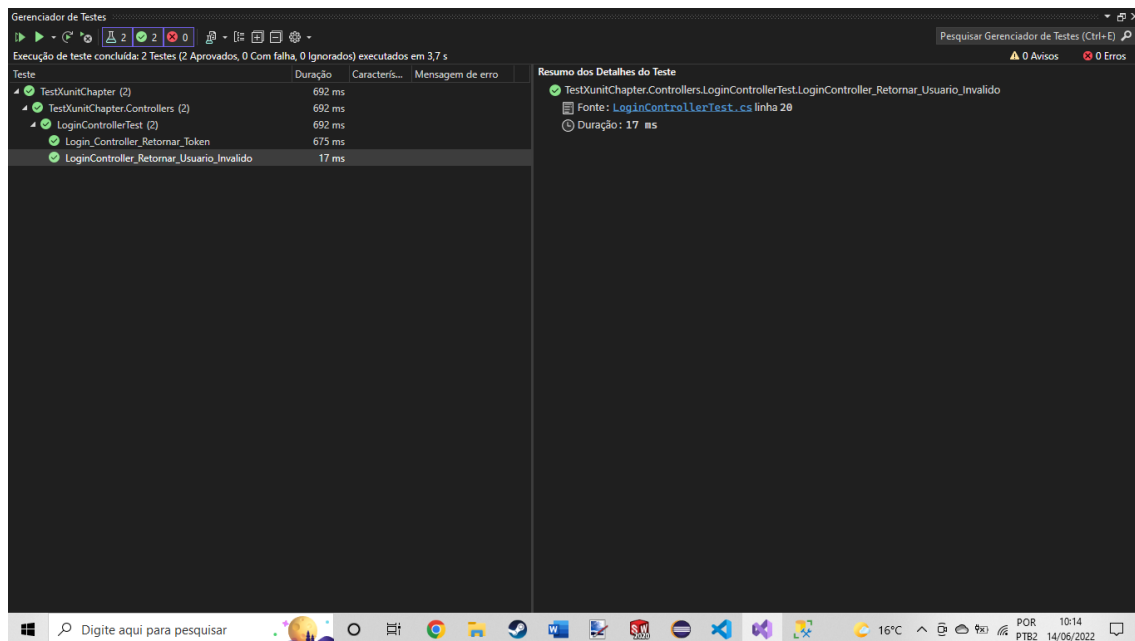


Figura 7 – resultado do teste do Caso de Teste 1 do Teste de Integração 1

### 3.2. Teste de Integração 2 – Retorno de Token no método de Login – Caso de Teste 1.

Descrição: Email = "email@email.com", Senha = "1234".

Resultado Esperado: token.

Resultado do Teste: Figura 8.

Nota: O método testado foi analisado pelo método de teste que retornou aprovação.

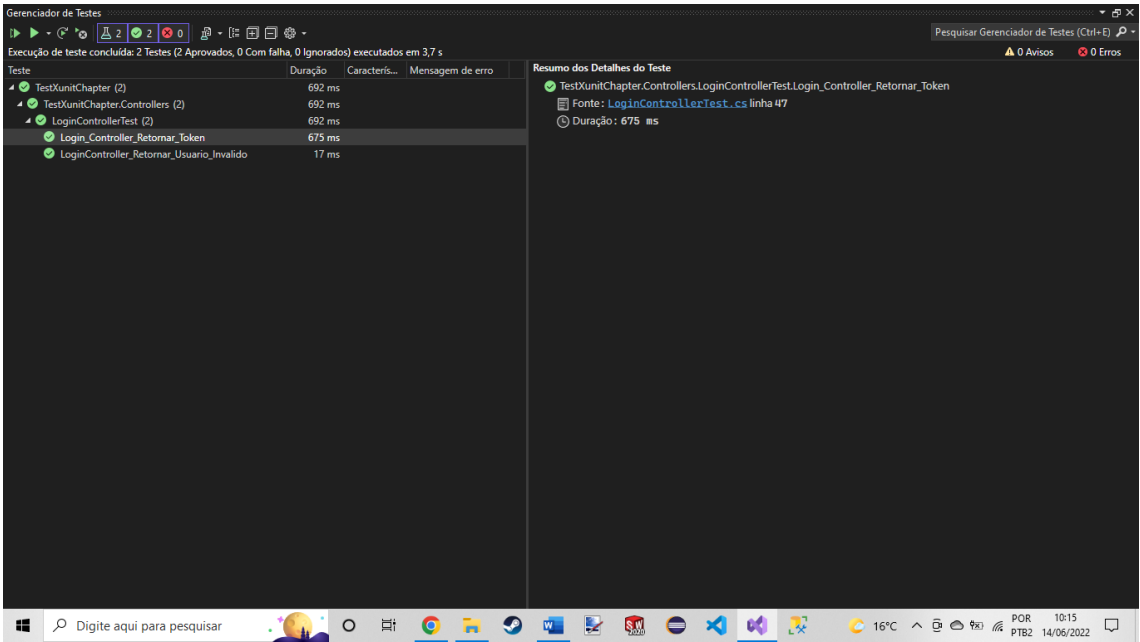


Figura 8 – resultado do teste do Caso de Teste 1 do Teste de Integração 2

## **CONCLUSÃO**

Com a execução e análise dos testes, foi constatado que os métodos avaliados para o projeto Armazenamento de Dados de Livros para a Biblioteca Chapter para a empresa Chapter estão desenvolvidos corretamente gerando resultados esperados na sua aplicação.