TEST IMPLEMENTATION

Enoiu Andra Maria CEN3.1b

TESTE AUTOMATE

- Pentru testele automate am implemetat pentru partea de admin si partea de user.
- Clasele pentru Home Page si Login Page sunt folosite in ambele implementari.
- Clasa de Home Page ne duce pe pagina principala al site-ului dupa cum se vede si dupa url.
- Are de asemenea implementata si GetLoginPage care prin apasarea butonului de loginButton, acesta ne va duce la pagina de Login.
- Clasa de Login Page ia toate field-urile necesare pentru completare din fromular (email si parola).
- Mai departe in Login Page este implementata functia Login in care se iau datele transmise si acestea sunt trimise spre completare prin “SendKeys”.

```
class HomePage
    {
        private IWebDriver webDriver;

        public HomePage(IWebDriver webDriver)
        {
            this.webDriver = webDriver;
            PageFactory.InitElements(webDriver, this);
        }

        [FindsBy(How = How.LinkText, Using = "Login")]
        private IWebElement loginButton;

        public LoginPage GetLoginPage()
        {
            loginButton.Click();
            return new LoginPage(webDriver);
        }

        public void GoToPage()
        {
            webDriver.Navigate().GoToUrl("https://localhost:44336/");
        }
    }
```

```
class LoginPage
    {
        private IWebDriver webDriver;

        [FindsBy(How = How.Id, Using = "exampleInputEmail1")]
        private IWebElement emailTextBox;

        [FindsBy(How = How.Id, Using = "exampleInputPassword1")]
        private IWebElement passwordTextBox;

        [FindsBy(How = How.XPath, Using =
"/html/body/div/main/center/section/form/center/div[1]/button")]
        private IWebElement loginButton;

        public LoginPage(IWebDriver driver)
        {
            webDriver = driver;
            PageFactory.InitElements(driver, this);
        }

        public void Login(string userName, string password)
        {
            emailTextBox.Clear();
            emailTextBox.SendKeys(userName);

            passwordTextBox.Clear();
            passwordTextBox.SendKeys(password);

            loginButton.Click();
        }
    }
```

TESTE AUTOMATE

```
namespace TenantsAss.AutomatedTest
{
    [TestClass]
    public class AdminPageTest
    {

        private IWebDriver webDriver;

        [TestInitialize]
        public void Initialize()
        {
            webDriver = new ChromeDriver();
        }

        [TestMethod]
        public void AdminAddInvoice()
        {
            string userName = "user1@user.com";
            string apartmentNo = "2";
            string apartmentId = "4";
            string price = "123";
            string dueDate = "05-24-2021-\t-10-30PM";
            string status = "Unpaid";
            string description = "Hello!You have a new invoice.Please, do not forget to pay.";

            HomePage homePage = new HomePage(webDriver);
            homePage.GoToPage();

            LoginPage loginPage = homePage.GetLoginPage();
            loginPage.Login("admin@admin.com", "Admin123456@");

            AdminPage adminPage = new AdminPage(webDriver);
            adminPage.GoToPage();

            AddInvoice addInvoice = adminPage.GoToAddInvoice();
            addInvoice.Create(userName, apartmentNo, apartmentId, price, dueDate, status, description);

            Assert.IsTrue(adminPage.InvoiceExists(apartmentNo));
        }

        [TestCleanup]
        public void Cleanup()
        {
            webDriver.Close();
        }
    }
}
```

- Pentru partea de admin se poate vedea ca se initializeaza un nou Chrome Driver dupa care se intra pe pagina de Home al site-ului.
- De pe pagina de Home se selecteaza link-ul pentru pagina de Login si dupa se intra pe aceasta.
- Pe pagina de Login se vor introduce automat datele pentru logare (email si parola) specifice adminului.
- Dupa ce acesta s-a logat cu success atunci o sa intre pe pagina pentru managerierea Invoice-urilor de unde se face click pe butonul de Create New Invoice.
- In formularul pentru Create Invoice se vor introduce automat datele necesare si se apasa butonul Create.
- Dupa ce toate datele au fost salvate cu success atunci ne intoarcem la tabelul de Invoices si se verifica daca s-a adaugat Invoice-ul respectiv dupa numarul apartamentului.
- Daca testul se termina fara nicio eroare si pagina de Chrome se inchide automat, atunci testul a decurs fara nicio problema.

TESTE AUTOMATE

```
namespace TenantsAss.AutomatedTest
{
    [TestClass]
    public class UserPageTest
    {
        private IWebDriver webDriver;

        [TestInitialize]
        public void Initialize()
        {
            webDriver = new ChromeDriver();
        }

        [TestMethod]
        public void UserPayInvoice()
        {
            string status = "Paid";

            HomePage homePage = new HomePage(webDriver);
            homePage.GoToPage();

            LoginPage loginPage = homePage.GetLoginPage();
            loginPage.Login("user@user.com", "User123456@");

            UserPage userPage = new UserPage(webDriver);
            userPage.GoToPage();

            PayInvoice payInvoice = userPage.GoToPayInvoice();
            payInvoice.Pay(status);

            Assert.IsTrue(userPage.PaimentExists(status));
        }

        [TestCleanup]
        public void Cleanup()
        {
            webDriver.Close();
        }
    }
}
```

- Pentru partea de user se poate observa ca se intra, la fel, pe pagina Home dupa care pe pagina pentru Login.
- Aici se introduc automat datele pentru logare specifice user-ului (email si parola).
- Dupa logarea cu success a user-ului se va intra pe pagina pentru Pay Invoce.
- Aici se alege plata unui invoice, iar dupa ce aceasta actiune a decurs fara probleme, atunci se verifica dupa status daca este ‘Paid’.

UNIT TESTE

- Pentru partea de Unit Teste am ales sa fac pentru Apartment (care poate avea mai multe Invoice-uri) si pentru Building (care poate avea mai multe apartamente).
- Pentru Apartment Logic Test se face un nou building cu datele necesare.
- Dupa aceasta se vor adauga mai multe apartamente, aceste la randul lor cu toate datele necesare.
- Apoi se apeleaza functia GetApartmentByNumber prin care ne ducem in lista de apartamente si numaram cate apartamente au numarul dat de noi.
- Intr-un final se testeaza daca numarul de apartamente gasite este cel asteptat de noi.

```
namespace TenantsAss.AppLogic.Tests
{
    [TestClass]
    public class ApartmentLogicTest
    {
        [TestMethod]
        public void GetApartmentByNumber_Return_CountOfApartmentsWithThatNumber()
        {
            //Arrange
            Building building = new Building()
            {
                BuildingId = 5,
                StreetName = "Street 5",
                StreetNo = "No.5",
                BuildingNo = "55"
            };

            List<Apartment> apartments = new List<Apartment>
            {
                new Apartment{ApartmentId = 10, ApartmentNo = 11, UserId = Guid.NewGuid(),
BuildingId = 5, BuildingNo = "55"},
                new Apartment{ApartmentId = 11, ApartmentNo = 33, UserId = Guid.NewGuid(),
BuildingId = 5, BuildingNo = "55"},
                new Apartment{ApartmentId = 12, ApartmentNo = 11, UserId = Guid.NewGuid(),
BuildingId = 5, BuildingNo = "55"},
                new Apartment{ApartmentId = 13, ApartmentNo = 22, UserId = Guid.NewGuid(),
BuildingId = 5, BuildingNo = "55"},
                new Apartment{ApartmentId = 14, ApartmentNo = 11, UserId = Guid.NewGuid(),
BuildingId = 5, BuildingNo = "55"},
            };

            foreach (var apartment in apartments)
                building.Apartments.Add(apartment);

            //Act
            var retList = building.GetApartmentByNumber(11);

            //Assert
            Assert.AreEqual(3, retList.Count);
        }
    }
}
```

UNIT TESTE

- Pentru Invoice Logic Test am facut nou nou apartament cu datele necesare.
- Apoi am adaugat o noua lista de invoice-uri.
- In aceasta lista se cauta cu ajutorul functiei GetInvoicePrice si se numara cate invoice-uri au pretul mai mare decat cel dat de noi.
- La final se testeaza daca numarul de invoice-uri este cel la care ne asteptam.

```
namespace TenantsAss.AppLogic.Tests
{
    [TestClass]
    public class InvoiceLogicTest
    {
        [TestMethod]
        public void GetInvoiceByPrice_Return_IfInvoicePriceIsGreater()
        {
            //Arrange
            Apartment apartment = new Apartment()
            {
                ApartmentId = 3,
                ApartmentNo = 44,
                UserId = Guid.NewGuid(),
                BuildingId = 1,
                BuildingNo = "No.1"
            };

            List<Invoice> invoices = new List<Invoice>
            {
                new Invoice{InvoiceId = 1, UserName = "user", ApartmentNo = 44, Price = 100, Status =
"Paid", ApartmentId = 3},
                new Invoice{InvoiceId = 2, UserName = "user", ApartmentNo = 44, Price = 250, Status =
"Paid", ApartmentId = 3},
                new Invoice{InvoiceId = 3, UserName = "user", ApartmentNo = 44, Price = 300, Status =
"Paid", ApartmentId = 3},
                new Invoice{InvoiceId = 4, UserName = "user", ApartmentNo = 44, Price = 190, Status =
"Paid", ApartmentId = 3},
                new Invoice{InvoiceId = 5, UserName = "user", ApartmentNo = 44, Price = 110, Status =
"Paid", ApartmentId = 3},
            };

            foreach (var invoice in invoices)
                apartment.Invoices.Add(invoice);

            //Act
            var retList = apartment.GetInvoiceByPrice(200);

            //Assert
            Assert.AreEqual(2, retList.Count);
        }
    }
}
```