

using Microsoft.VisualStudio.TestTools.UnitTesting;

using RoadsOfRussiaDLL.Desktop;

using RoadsOfRussiaDLL.Document;

using RoadsOfRussiaDLL.Mobile;

using System;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

namespace UnitTest

{

[TestClass]

public class Documents

{

private DocumentController documentsController;

[TestInitialize]

public void Initialize()

{

documentsController = new DocumentController();

}

[TestMethod]

public async Task GetJwtTokenWithRightLogin()

{

var result = await documentsController.Authorization("admin", "admin");

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetJwtTokenFalseLogin()

{

var result = await documentsController.Authorization("nick", "admin");

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetDocumentsWithAuthorization()

{

var jwtToken = await documentsController.Authorization("admin", "admin");

var result = await documentsController.GetDocuments(jwtToken);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetDocumentsWithoutAuthorization()

{

var jwtToken = await documentsController.Authorization("", "");

var result = await documentsController.GetDocuments(jwtToken);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetCommentWithAuthorization()

{

var jwtToken = await documentsController.Authorization("admin", "admin");

var result = await documentsController.GetComments(1, jwtToken);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetCommentWithoutAuthorization()

{

var jwtToken = await documentsController.Authorization("", "");

var result = await documentsController.GetComments(1, jwtToken);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task PostCommentWithAuthorization()

{

var jwtToken = await documentsController.Authorization("admin", "admin");

var result = await documentsController.SetComment(1, "test", 1, jwtToken);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task PostCommentWithoutAuthorization()

{

var jwtToken = await documentsController.Authorization("", "");

var result = await documentsController.SetComment(1, "test", 1, jwtToken);

Assert.IsNotNull(result);

}

}

[TestClass]

public class Desktop

{

private DesktopController desktopController;

[TestInitialize]

public void Initialize()

{

desktopController = new DesktopController();

}

[TestMethod]

public async Task GetEmployees()

{

var result = await desktopController.GetEmployeesAsync(1);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetDivision()

{

var result = await desktopController.GetDivisionsAsync();

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetPost()

{

var result = await desktopController.GetPostAsync();

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetDirecorAndAssistent()

{

var result = await desktopController.GetDirecorAndAssistent(1);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetTraningCalendar()

{

var result = await desktopController.GetTraningCalendar(26);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetVacationCalendar()

{

var result = await desktopController.GetVacationCalendar(26);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetTemporaryAbsenceCalendar()

{

var result = await desktopController.GetVacationCalendar(26);

Assert.IsNotNull(result);

}

[TestMethod]

public async Task PostTraningCalendar()

{

var result = await desktopController.PostTraningCalendar(1, "test", "test", DateTime.Now, DateTime.Now);

Assert.IsTrue(result);

var lastCalendar = await desktopController.GetTraningCalendar(1);

await desktopController.RemoveTraningCalendar(lastCalendar.LastOrDefault().IDCalendar);

}

[TestMethod]

public async Task PostVacationCalendar()

{

var result = await desktopController.PostVacationCalendar(1, "test", "test", DateTime.Now, DateTime.Now);

Assert.IsTrue(result);

var lastCalendar = await desktopController.GetVacationCalendar(1);

await desktopController.RemoveVacationCalendar(lastCalendar.LastOrDefault().IDCalendar);

}

[TestMethod]

public async Task PostTemporaryAbsenceCalendar()

{

var result = await desktopController.PostTemporaryAbsenceCalendar(1, "test", "test", DateTime.Now, DateTime.Now);

Assert.IsTrue(result);

var lastCalendar = await desktopController.GetTemporaryAbsenceCalendar(1);

await desktopController.RemoveTemporaryAbsenceCalendar(lastCalendar.LastOrDefault().IDCalendar);

}

[TestMethod]

public async Task RemoveExistTraningCalendar()

{

await desktopController.PostTraningCalendar(1, "test", "test", DateTime.Now, DateTime.Now);

var calendar = await desktopController.GetTraningCalendar(1);

var result = await desktopController.RemoveTraningCalendar(calendar.LastOrDefault().IDCalendar);

Assert.IsTrue(result);

}

[TestMethod]

public async Task RemoveNotExistTraningCalendar()

{

var result = await desktopController.RemoveTraningCalendar(-1);

Assert.IsTrue(result);

}

[TestMethod]

public async Task RemoveExistVacationCalendar()

{

await desktopController.PostVacationCalendar(1, "test", "test", DateTime.Now, DateTime.Now);

var calendar = await desktopController.GetVacationCalendar(1);

var result = await desktopController.RemoveVacationCalendar(calendar.LastOrDefault().IDCalendar);

Assert.IsTrue(result);

}

[TestMethod]

public async Task RemoveNotExistVacationCalendar()

{

var result = await desktopController.RemoveVacationCalendar(-1);

Assert.IsTrue(result);

}

[TestMethod]

public async Task RemoveExistTemporaryAbsenceCalendar()

{

await desktopController.PostTemporaryAbsenceCalendar(1, "test", "test", DateTime.Now, DateTime.Now);

var calendar = await desktopController.GetTemporaryAbsenceCalendar(1);

var result = await desktopController.RemoveTemporaryAbsenceCalendar(calendar.LastOrDefault().IDCalendar);

Assert.IsTrue(result);

}

[TestMethod]

public async Task RemoveNotExistTemporaryAbsenceCalendar()

{

var result = await desktopController.RemoveTemporaryAbsenceCalendar(-1);

Assert.IsTrue(result);

}

}

[TestClass]

public class Mobile

{

private MobileController mobileController;

[TestInitialize]

public void Initialize()

{

mobileController = new MobileController();

}

[TestMethod]

public async Task GetNews()

{

var result = await mobileController.GetNews();

Assert.IsNotNull(result);

}

[TestMethod]

public async Task GetEvents()

{

var result = await mobileController.GetEvents();

Assert.IsNotNull(result);

}

[TestMethod]

public async Task ExistNewsPositiveVote()

{

var result = await mobileController.NewsPositiveVote(1);

Assert.IsTrue(result);

}

[TestMethod]

public async Task NotExistNewsPositiveVote()

{

var result = await mobileController.NewsPositiveVote(-1);

Assert.IsTrue(result);

}

[TestMethod]

public async Task ExistNewsNegativeVote()

{

var result = await mobileController.NewsNegativeVote(1);

Assert.IsTrue(result);

}

[TestMethod]

public async Task NotExistNewsNegativeVote()

{

var result = await mobileController.NewsNegativeVote(-1);

Assert.IsTrue(result);

}

}

}