FORMATO PRUEBA DE SOFTWARE

- **Denominación del Programa de Formación:** Tecnología en Análisis y Desarrollo de Sistemas de información
- **Código del Programa de Formación:** 228106/ Versión 102
- **Nombre del Proyecto:** Software para el sector empresarial en el departamento del huila.
- **Fase del Proyecto:** Ejecución
- **Actividad de Proyecto:** Construir la capa de datos, lógica de negocios y presentación del sistema de información aplicando estándares de calidad y buenas prácticas de ergonomía.
- **Competencia:** Construir el sistema que cumpla con los requisitos de la solución informática.
- **Resultados de aprendizaje a alcanzar:** Ejecutar y documentar las pruebas del software, aplicando técnicas de ensayo-error, de acuerdo con el plan diseñado y los procedimientos establecidos por la empresa

INTRODUCCIÓN

El propósito del siguiente documento es realizar seguimiento a cada uno de los proyectos formativos que se están desarrollando a lo largo del tecnólogo ADSO por medio de la implementación de pruebas unitarias.

Las pruebas unitarias nos permiten asegurarnos de que los métodos individuales del código de un proyecto funcionen correctamente, así como la integración de todo el proyecto.

Se busca en los requerimientos funcionales: encontrar errores relacionados con la funcionalidad del sistema y en los no funcionales, fallas de rendimiento, seguridad, usabilidad, fiabilidad.

Existen 4 tipos de pruebas:

- **Pruebas unitarias.** Diseñadas para probar una parte pequeña y específica de funcionalidad. Ej. Un método.

- **Pruebas de integración.** Diseñadas para probar la interacción entre los distintos componentes de un sistema.

- **Pruebas de sistema**. Diseñadas para probar el sistema en su totalidad como si de una caja negra se tratase.

- **Pruebas de aceptación.** Diseñadas para verificar que el sistema cumple con los requisitos exigidos por el usuario.

DOCUMENTOS DE REFERENCIA	
Versión:	**Título:**
V1	Autentificación

Detalle de la prueba

Fecha de realización:	**Duración de la prueba: 148 ms**
Requerimientos Funcional de la prueba	**Validar usuario**
Objetivo	**Credenciales**
Tipo de Prueba	**Prueba unitaria**
Datos de entrada de la prueba	**[TestMethod]** **public async Task ValidationUser_CasoFeliz_JwtYExpiracion_AAA()** **{** **// ==========** **// PREPARAR (Arrange)** **// ==========** **var login = new CredencialesDto** **{** **Email = "user@test.com",** **Password = "P@ssw0rd"** **};** **// AuthDto es el que recibes desde Data (GenerateToken).** **// Ajusta nombres de propiedades si en tu proyecto difieren.** **var esperado = new AuthDto** **{** **Token = "jwt-123",** **Expiracion = DateTime.UtcNow.AddMinutes(60)** **};** **_authMock** **.Setup(a => a.AuthApp(**

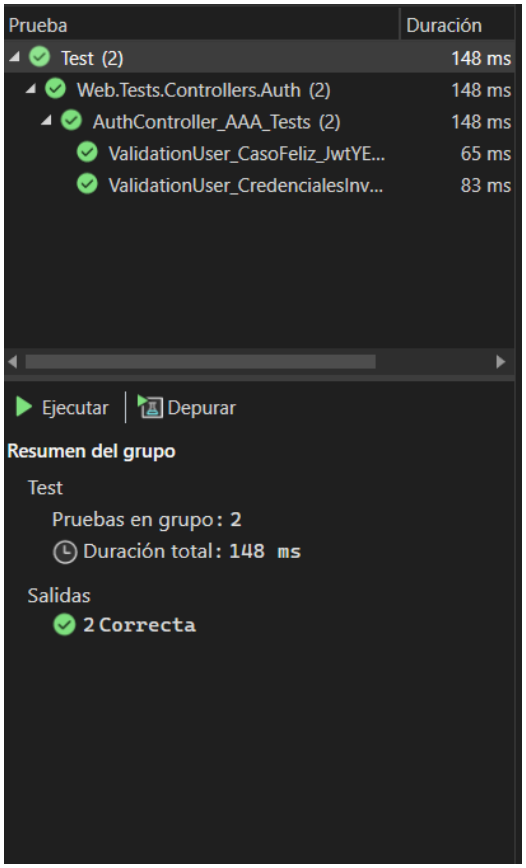
	It.Is<CredencialesDto>(c => c.Email == login.Email && c.Password == login.Password)))
	.ReturnsAsync(esperado);
	
	// ==========
	// PROBAR (Act)
	// ==========
	var actionResult = await _controller.ValidationUser(login);
	
	// ==========
	// VERIFICAR (Assert)
	// ==========
	var ok = actionResult as OkObjectResult;
	Assert.IsNotNull(ok, "Debe responder 200 OK");
	Assert.AreEqual(200, ok!.StatusCode);
	
	var payload = ok.Value as AuthDto;
	Assert.IsNotNull(payload, "El payload debe ser AuthDto");
	Assert.AreEqual(esperado.Token, payload!.Token, "Token incorrecto");
	
	// Comparación con tolerancia por diferencias de reloj
	var delta = Math.Abs((payload.Expiracion - esperado.Expiracion).TotalSeconds);
	Assert.IsTrue(delta < 3, $"ExpiresAt distinto (delta {delta:N2}s)");
	
	_authMock.Verify(a => a.AuthApp(It.IsAny<CredencialesDto>()), Times.Once);
	_authMock.VerifyNoOtherCalls();
	}		

Procedimiento de Prueba	MSTEST ASP.NET CORE pruebas en Visual Estudio		
Datos de salida - Resultado Esperado	Se registra de manera exitosa, y retorna los objetos correspondientes		
Resultado Obtenido Prueba Exitosa	**¿Prueba Exitosa?**	**Si (x)**	**No ( )**

Realizado por:	**Firma**	**Fecha**

JESUS FERNANDO CARVAJAL		31/10/2025
Aprobado por:	**Firma**	**Fecha**

Adjunto:



DOCUMENTOS DE REFERENCIA	
Versión:	**Título:**
V1	Peticiones principales de formularios

Detalle de la prueba

Fecha de realización:	**Duración de la prueba: 4,5 s**
Requerimientos Funcional de la prueba	**Todos los métodos principales**
Objetivo	**Funcionalidad de métodos principales**
Tipo de Prueba	**Prueba unitaria**
Datos de entrada de la prueba	**[TestClass]** **public class FormControllerTests** **{** **private Mock<IQueryServices<Form, FormDto>> _qMock = default!;** **private Mock<ICommandService<Form, FormDto>> _cMock = default!;** **private FormController _controller = default!;** **[TestInitialize]** **public void Setup()** **{** **_qMock = new Mock<IQueryServices<Form, FormDto>>(MockBehavior.Strict);** **_cMock = new Mock<ICommandService<Form, FormDto>>(MockBehavior.Strict);** **_controller = new FormController(_qMock.Object, _cMock.Object);** **}** **[TestMethod]** **public async Task GetAll_OK_AAA()** **{**

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        // PREPARAR
        int? status = 1;
        var esperado = new List<FormDto>
        {
            new FormDto { Id = 10, Name =
"Dashboard", Path = "/home", Order = 1 },
            new FormDto { Id = 11, Name =
"Users",    Path = "/users", Order = 2 }
        };

        _qMock.Setup(q =>
q.GetAllServices(status)).ReturnsAsync(es
perado);

        // PROBAR
        var action = await
_controller.GetAll(status);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);

CollectionAssert.AreEquivalent(esperado,
(System.Collections.ICollection)ok.Value!);

        _qMock.Verify(q =>
q.GetAllServices(status), Times.Once);
        _qMock.VerifyNoOtherCalls();
        _cMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task GetById_OK_AAA()
    {
        // PREPARAR
        var id = 42;
        var dto = new FormDto { Id = id, Name
= "Reports", Path = "/reports" };
        _qMock.Setup(q =>
q.GetByIdServices(id)).ReturnsAsync(dto);

        // PROBAR
        var action = await
_controller.GetById(id);

        // VERIFICAR
        var ok = action as OkObjectResult;
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(dto, ok.Value);

        _qMock.Verify(q =>
q.GetByIdServices(id), Times.Once);
        _qMock.VerifyNoOtherCalls();
        _cMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Create_CreatedAtAction_AAA()
    {
        // PREPARAR
        var input = new FormDto { Name =
"Settings", Description = "Config", Path =
"/settings", Order = 3 };
        var created = new FormDto { Id = 77,
Name = input.Name, Description =
input.Description, Path = input.Path,
Order = input.Order };

        _cMock.Setup(c =>
c.CreateServices(input)).ReturnsAsync(cre
ated);

        // PROBAR
        var action = await
_controller.Create(input);

        // VERIFICAR
        var createdAt = action as
CreatedAtActionResult;
        Assert.IsNotNull(createdAt, "Debe
devolver CreatedAtAction");

Assert.AreEqual(nameof(_controller.GetB
yId), createdAt!.ActionName);

Assert.IsNotNull(createdAt.RouteValues);
        Assert.AreEqual(77,
createdAt.RouteValues!["id"]);
        Assert.AreEqual(created,
createdAt.Value);

        _cMock.Verify(c =>
c.CreateServices(input), Times.Once);
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Update_OK_True_AAA()
    {
        // PREPARAR
        var dto = new FormDto { Id = 5, Name
= "Profile", Path = "/me" };
        _cMock.Setup(c =>
c.UpdateServices(dto)).ReturnsAsync(true
);

        // PROBAR
        var action = await
_controller.Update(dto);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.UpdateServices(dto), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Delete_OK_True_AAA()
    {
        // PREPARAR
        var id = 9;
        _cMock.Setup(c =>
c.DeleteServices(id)).ReturnsAsync(true);

        // PROBAR
        var action = await
_controller.Delete(id);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.DeleteServices(id), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
DeleteLogica_OK_True_AAA()
    {
        // PREPARAR
        var id = 9;
        var status = 0;
        _cMock.Setup(c =>
c.DeleteLogicalServices(id,
status)).ReturnsAsync(true);

        // PROBAR
        var action = await
_controller.DeleteLogica(id, status);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.DeleteLogicalServices(id, status),
Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
PartialUpdate_OK_RetornaDto_AAA()
    {
        // PREPARAR
        var patch = new FormDto { Id = 15,
Name = "Renamed", Order = 99 };
        var result = new FormDto { Id = 15,
Name = "Renamed", Order = 99, Path =
"/renamed" };
```

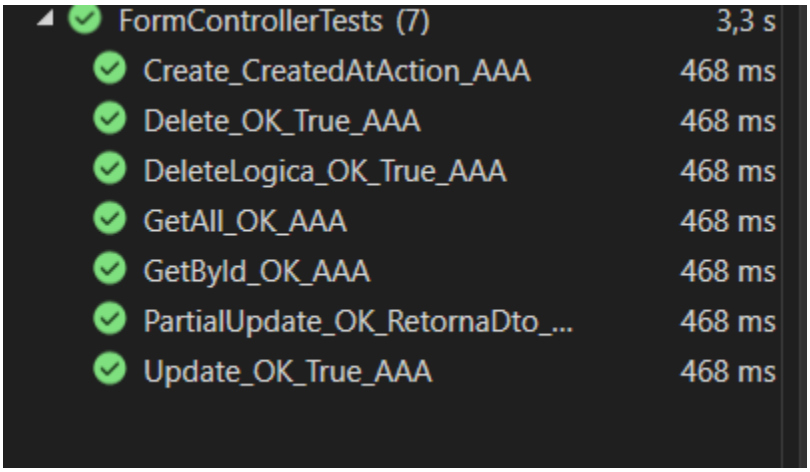

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

	_cMock.Setup(c => c.PathServices(patch)).ReturnsAsync(result);
	
	// PROBAR
	var action = await _controller.PartialUpdate(patch);
	
	// VERIFICAR
	var ok = action as OkObjectResult;
	Assert.IsNotNull(ok);
	Assert.AreEqual(200, ok!.StatusCode);
	Assert.AreEqual(result, ok.Value);
	
	_cMock.Verify(c => c.PathServices(patch), Times.Once);
	_cMock.VerifyNoOtherCalls();
	_qMock.VerifyNoOtherCalls();
	}
	}		

Procedimiento de Prueba	**MSTEST ASP.NET CORE pruebas en Visual Estudio**		
Datos de salida - Resultado Esperado	**Se registra de manera éxitos, la creación, consulta, actualización y eliminación de un registro de formulario**		
Resultado Obtenido Prueba Exitosa	**¿Prueba Exitosa?**	**Si (x)**	**No ( )**

Realizado por:	**Firma**	**Fecha**
JESUS FERNANDO CARVAJAL		31/10/2025
Aprobado por:	**Firma**	**Fecha**

Adjuntos:



Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

DOCUMENTOS DE REFERENCIA	
Versión:	**Título:**
V1	Peticiones principales de eps (Parameters)

Detalle de la prueba

Fecha de realización:	**Duración de la prueba: 400 ms**
Requerimientos Funcional de la prueba	**Todos los métodos principales**
Objetivo	**Funcionalidad de métodos principales**
Tipo de Prueba	**Prueba de creación, consulta y validación**
Datos de entrada de la prueba	**[TestClass]** **public class EpsControllerTests** **{** **private Mock<IQueryServices<Eps, EpsDto>> _qMock = default!;** **private Mock<ICommandService<Eps, EpsDto>> _cMock = default!;** **private EpsController _controller = default!;** **[TestInitialize]** **public void Setup()** **{** **_qMock = new Mock<IQueryServices<Eps, EpsDto>>(MockBehavior.Strict);** **_cMock = new Mock<ICommandService<Eps, EpsDto>>(MockBehavior.Strict);** **_controller = new EpsController(_qMock.Object, _cMock.Object);** **}** **[TestMethod]** **public async Task GetAll_OK_AAA()** **{**

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        // PREPARAR
        int? status = 1;
        var esperado = new List<EpsDto>
        {
            new EpsDto { Id = 10, Name =
"Sura" },
            new EpsDto { Id = 11, Name =
"Sanitas" }
        };
        _qMock.Setup(q =>
q.GetAllServices(status)).ReturnsAsync(es
perado);

        // PROBAR
        var action = await
_controller.GetAll(status);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);

CollectionAssert.AreEquivalent(esperado,
(System.Collections.ICollection)ok.Value!);

        _qMock.Verify(q =>
q.GetAllServices(status), Times.Once);
        _qMock.VerifyNoOtherCalls();
        _cMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task GetById_OK_AAA()
    {
        // PREPARAR
        var id = 42;
        var dto = new EpsDto { Id = id, Name =
"Coomeva" };
        _qMock.Setup(q =>
q.GetByIdServices(id)).ReturnsAsync(dto);

        // PROBAR
        var action = await
_controller.GetById(id);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(dto, ok.Value);

        _qMock.Verify(q =>
q.GetByIdServices(id), Times.Once);
        _qMock.VerifyNoOtherCalls();
        _cMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Create_CreatedAtAction_AAA()
    {
        // PREPARAR
        var input = new EpsDto { Name =
"Nueva EPS" };
        var created = new EpsDto { Id = 77,
Name = "Nueva EPS" };
        _cMock.Setup(c =>
c.CreateServices(input)).ReturnsAsync(cre
ated);

        // PROBAR
        var action = await
_controller.Create(input);

        // VERIFICAR
        var createdAt = action as
CreatedAtActionResult;
        Assert.IsNotNull(createdAt, "Debe
devolver CreatedAtAction");

Assert.AreEqual(nameof(_controller.GetB
yId), createdAt!.ActionName);

Assert.IsNotNull(createdAt.RouteValues);
        Assert.AreEqual(77,
createdAt.RouteValues!["id"]);
        Assert.AreEqual(created,
createdAt.Value);

        _cMock.Verify(c =>
c.CreateServices(input), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
    public async Task
Update_OK_True_AAA()
    {
        // PREPARAR
        var dto = new EpsDto { Id = 5, Name =
"Actualizada EPS" };
        _cMock.Setup(c =>
c.UpdateServices(dto)).ReturnsAsync(true
);

        // PROBAR
        var action = await
_controller.Update(dto);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.UpdateServices(dto), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Delete_OK_True_AAA()
    {
        // PREPARAR
        var id = 9;
        _cMock.Setup(c =>
c.DeleteServices(id)).ReturnsAsync(true);

        // PROBAR
        var action = await
_controller.Delete(id);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.DeleteServices(id), Times.Once);
        _cMock.VerifyNoOtherCalls();
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
DeleteLogica_OK_True_AAA()
    {
        // PREPARAR
        var id = 9;
        var status = 0;
        _cMock.Setup(c =>
c.DeleteLogicalServices(id,
status)).ReturnsAsync(true);

        // PROBAR
        var action = await
_controller.DeleteLogica(id, status);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200, ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.DeleteLogicalServices(id, status),
Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
PartialUpdate_OK_RetornaDto_AAA()
    {
        // PREPARAR
        var patch = new EpsDto { Id = 15,
Name = "EPS Patch" };
        var result = new EpsDto { Id = 15,
Name = "EPS Patch" };

        _cMock.Setup(c =>
c.PathServices(patch)).ReturnsAsync(resul
t);

        // PROBAR
        var action = await
_controller.PartialUpdate(patch);
```

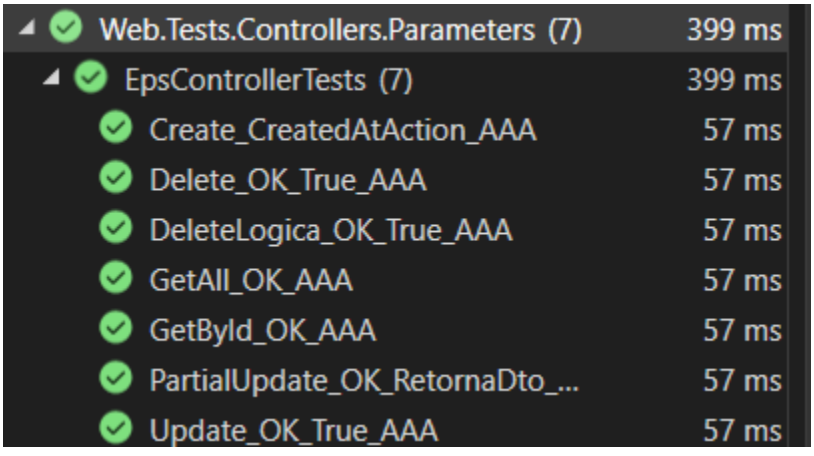

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

	// VERIFICAR
	var ok = action as OkObjectResult;
	Assert.IsNotNull(ok);
	Assert.AreEqual(200, ok!.StatusCode);
	Assert.AreEqual(result, ok.Value);
	
	_cMock.Verify(c => c.PathServices(patch), Times.Once);
	_cMock.VerifyNoOtherCalls();
	_qMock.VerifyNoOtherCalls();
	}
	}

Procedimiento de Prueba	**MSTEST ASP.NET CORE pruebas en Visual Estudio**
Datos de salida - Resultado Esperado	**Se registra de manera exitosa, y retorna los objetos correspondientes**
Resultado Obtenido Prueba Exitosa	**¿Prueba Exitosa?** / **Si (x)** / **No ( )**

Realizado por:	**Firma**	**Fecha**
JESUS FERNANDO CARVAJAL		31/10/2025
Aprobado por:	**Firma**	**Fecha**

Adjuntos



DOCUMENTOS DE REFERENCIA	
Versión:	**Título:**
V1	Peticiones principales de Agenda (Business)

Detalle de la prueba

Fecha de realización:	**Duración de la prueba: 400 ms**
Requerimientos Funcional de la prueba	**Todos los métodos principales**
Objetivo	**Funcionalidad de métodos principales**
Tipo de Prueba	**Prueba de creación, consulta y validación**
Datos de entrada de la prueba	**[TestClass]** **public class AgendaControllerTests** **{** **private Mock<IQueryServices<Agenda, AgendaDto>> _qMock = default!;** **private Mock<ICommandService<Agenda, AgendaDto>> _cMock = default!;** **private AgendaController _controller = default!;** **[TestInitialize]** **public void Setup()** **{** **_qMock = new Mock<IQueryServices<Agenda, AgendaDto>>(MockBehavior.Strict);** **_cMock = new Mock<ICommandService<Agenda, AgendaDto>>(MockBehavior.Strict);** **_controller = new AgendaController(_qMock.Object, _cMock.Object);** **}** **[TestMethod]**

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
    public async Task GetAll_OK_AAA()
    {
        // PREPARAR
        int? status = 1;
        var esperado = new List<AgendaDto>
        {
            new AgendaDto { Id = 1, Name =
"Agenda Ventas", Description =
"Reuniones comerciales" },
            new AgendaDto { Id = 2, Name =
"Agenda Soporte", Description = "Tickets
críticos" }
        };
        _qMock.Setup(q =>
q.GetAllServices(status)).ReturnsAsync(es
perado);

        // PROBAR
        var action = await
_controller.GetAll(status);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200,
ok!.StatusCode);

CollectionAssert.AreEquivalent(esperado,
(System.Collections.ICollection)ok.Value!);

        _qMock.Verify(q =>
q.GetAllServices(status), Times.Once);
        _qMock.VerifyNoOtherCalls();
        _cMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task GetById_OK_AAA()
    {
        // PREPARAR
        var id = 42;
        var dto = new AgendaDto { Id = id,
Name = "Agenda Operaciones",
Description = "Guardias" };
        _qMock.Setup(q =>
q.GetByIdServices(id)).ReturnsAsync(dto);

        // PROBAR
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        var action = await
_controller.GetById(id);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200,
ok!.StatusCode);
        Assert.AreEqual(dto, ok.Value);

        _qMock.Verify(q =>
q.GetByIdServices(id), Times.Once);
        _qMock.VerifyNoOtherCalls();
        _cMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Create_CreatedAtAction_AAA()
    {
        // PREPARAR
        var input = new AgendaDto { Name =
"Nueva Agenda", Description =
"Descripción X" };
        var created = new AgendaDto { Id =
77, Name = "Nueva Agenda", Description
= "Descripción X" };
        _cMock.Setup(c =>
c.CreateServices(input)).ReturnsAsync(cre
ated);

        // PROBAR
        var action = await
_controller.Create(input);

        // VERIFICAR
        var createdAt = action as
CreatedAtActionResult;
        Assert.IsNotNull(createdAt, "Debe
devolver CreatedAtAction");

Assert.AreEqual(nameof(_controller.GetB
yId), createdAt!.ActionName);

Assert.IsNotNull(createdAt.RouteValues);
        Assert.AreEqual(77,
createdAt.RouteValues!["id"]);
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        Assert.AreEqual(created,
createdAt.Value);

        _cMock.Verify(c =>
c.CreateServices(input), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Update_OK_True_AAA()
    {
        // PREPARAR
        var dto = new AgendaDto { Id = 5,
Name = "Agenda Actualizada", Description
= "Nueva desc" };
        _cMock.Setup(c =>
c.UpdateServices(dto)).ReturnsAsync(true
);

        // PROBAR
        var action = await
_controller.Update(dto);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200,
ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.UpdateServices(dto), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
Delete_OK_True_AAA()
    {
        // PREPARAR
        var id = 9;
        _cMock.Setup(c =>
c.DeleteServices(id)).ReturnsAsync(true);

        // PROBAR
```


Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
        var action = await
_controller.Delete(id);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200,
ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.DeleteServices(id), Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
DeleteLogica_OK_True_AAA()
    {
        // PREPARAR
        var id = 9;
        var status = 0;
        _cMock.Setup(c =>
c.DeleteLogicalServices(id,
status)).ReturnsAsync(true);

        // PROBAR
        var action = await
_controller.DeleteLogica(id, status);

        // VERIFICAR
        var ok = action as OkObjectResult;
        Assert.IsNotNull(ok);
        Assert.AreEqual(200,
ok!.StatusCode);
        Assert.AreEqual(true, ok.Value);

        _cMock.Verify(c =>
c.DeleteLogicalServices(id, status),
Times.Once);
        _cMock.VerifyNoOtherCalls();
        _qMock.VerifyNoOtherCalls();
    }

    [TestMethod]
    public async Task
PartialUpdate_OK_RetornaDto_AAA()
```

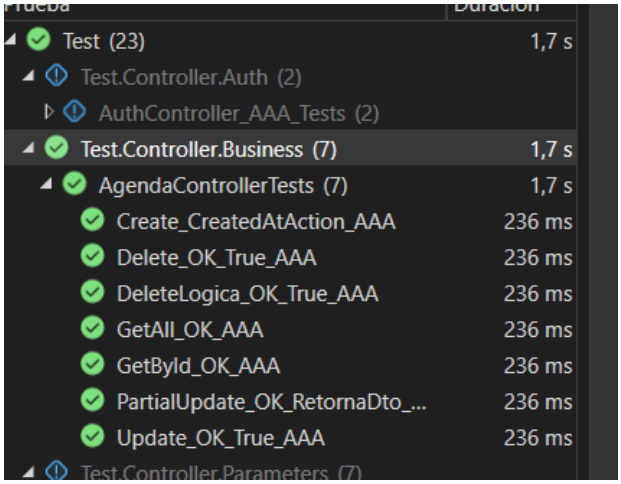

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

	{
	// PREPARAR
	var patch = new AgendaDto { Id = 15, Name = "Agenda Patch", Description = "Desc Patch" };
	var result = new AgendaDto { Id = 15, Name = "Agenda Patch", Description = "Desc Patch" };
	
	_cMock.Setup(c => c.PathServices(patch)).ReturnsAsync(result);
	
	// PROBAR
	var action = await _controller.PartialUpdate(patch);
	
	// VERIFICAR
	var ok = action as OkObjectResult;
	Assert.IsNotNull(ok);
	Assert.AreEqual(200, ok!.StatusCode);
	Assert.AreEqual(result, ok.Value);
	
	_cMock.Verify(c => c.PathServices(patch), Times.Once);
	_cMock.VerifyNoOtherCalls();
	_qMock.VerifyNoOtherCalls();
	}
	}

Procedimiento de Prueba	**MSTEST ASP.NET CORE pruebas en Visual Estudio**
Datos de salida - Resultado Esperado	**Se registra de manera exitosa, y retorna los objetos correspondientes**
Resultado Obtenido Prueba Exitosa	**¿Prueba Exitosa?** | **Si (x)** | **No ( )**

Realizado por:	**Firma**	**Fecha**
JESUS FERNANDO CARVAJAL		31/10/2025
Aprobado por:	**Firma**	**Fecha**

Adjuntos:



Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

DOCUMENTOS DE REFERENCIA	
Versión:	**Título:**
V1	integración prueba de Personas

Detalle de la prueba

Fecha de realización:	**Duración de la prueba: 900ms**
Requerimientos Funcional de la prueba	**Todos los métodos principales**
Objetivo	**Funcionalidad de métodos principales**
Tipo de Prueba	**Prueba de integración**
Datos de entrada de la prueba	**[TestMethod]** **public async Task InsertComplete_Inserta_Guarda_Devuelve_AsNoTracking_Con_DataBasic_Y_FKs()** **{** **// Arrange** **var person = NewPerson(** **rhId: 1, epsId: 1, municipalityId: 1, materialStatusId: 1,** **address: "Cra 45 #67-89", birth: new DateTime(1988, 5, 20), stratum: 4** **);** **// Act** **var result = await _sut.InsertComplete(person);** **_createdPersonIds.Add(result.Id); // para limpiar al final** **// Assert** **Assert.IsTrue(result.Id > 0);** **Assert.IsNotNull(result.DataBasic);** **Assert.AreEqual("Cra 45 #67-89", result.DataBasic!.Adress);** **Assert.AreEqual(new DateTime(1988, 5, 20), result.DataBasic!.BrithDate);**

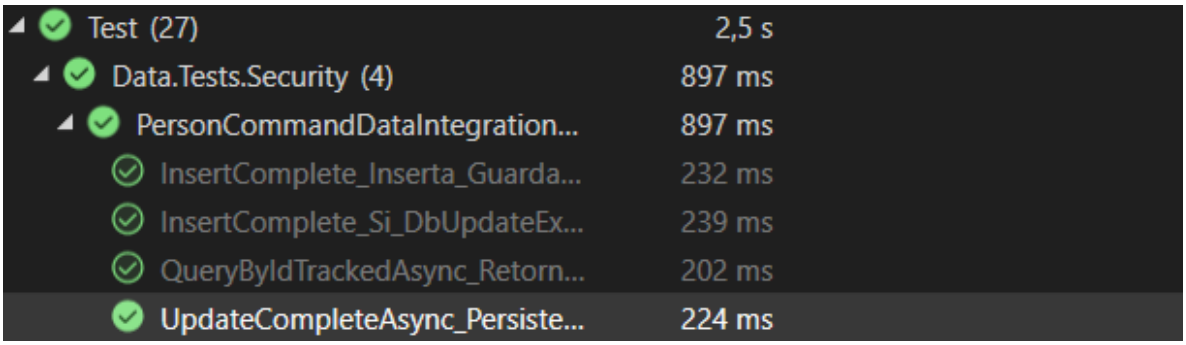
Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

	Assert.AreEqual(4, result.DataBasic!.StratumStatus);
	Assert.AreEqual(1, result.DataBasic!.RhId);
	Assert.AreEqual(1, result.DataBasic!.EpsId);
	Assert.AreEqual(1, result.DataBasic!.MunisipalityId);
	Assert.AreEqual(1, result.DataBasic!.MaterialStatusId);
	
	// Debe venir sin tracking (consulta AsNoTracking en el método)
	Assert.AreEqual(EntityState.Detached, _ctx.Entry(result).State);
	
	// Verificación en DB
	var fromDb = await _ctx.Set<Person>()
	.Include(p => p.DataBasic)
	.AsNoTracking()
	.FirstOrDefaultAsync(p => p.Id == result.Id);
	
	Assert.IsNotNull(fromDb);
	Assert.IsNotNull(fromDb!.DataBasic);
	Assert.AreEqual(result.Id, fromDb.DataBasic!.PersonId);
	}		

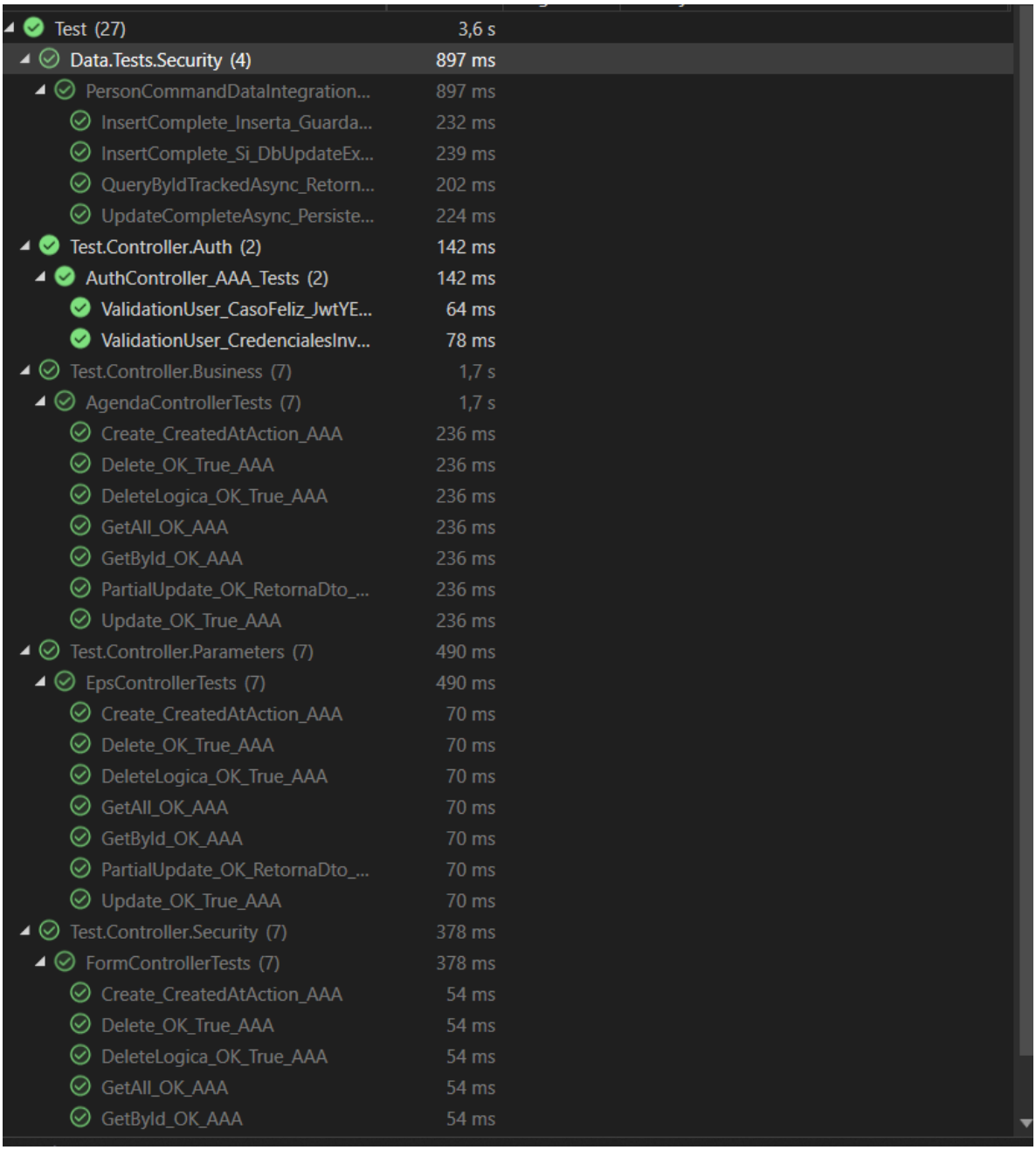
Procedimiento de Prueba	**MSTEST ASP.NET CORE pruebas en Visual Estudio**		
Datos de salida - Resultado Esperado	**Se registra de manera exitosa, y retorna los objetos correspondientes**		
Resultado Obtenido Prueba Exitosa	**¿Prueba Exitosa?**	**Si (x)**	**No ( )**

Realizado por:	**Firma**	**Fecha**
JESUS FERNANDO CARVAJAL		31/10/2025
Aprobado por:	**Firma**	**Fecha**

Adjunto:



Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com



Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com