UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

Carrera: Ingeniería en Sistemas Computacionales

Materia: Metodología de desarrollo de sistemas

Entrega: 14 Resultados del Sprint 1

Trabajo: En equipo

Profesor: Juan Muñoz López

Alumnos:

Rubén Eduardo Dávila Flores	ID 261740
Frida Ximena Escamilla Ramírez	ID 321538
Robert Everett Fillingham Gaeta	ID 260982
César Reyes Torres	ID 261177
Estefeen Sandoval Rodríguez	ID 261527

Semestre: 9°C

Fecha: 25 de noviembre de 2025

Tabla de contenido

Introducción

Este documento marca el cierre formal de nuestro Sprint 1 y tiene como objetivo presentar la evidencia tangible de que el incremento de software desarrollado es funcional, robusto y cumple con los criterios de aceptación acordados.

Más allá de simplemente mostrar código, en las siguientes secciones detallamos la exhaustiva estrategia de aseguramiento de calidad (QA) que ejecutamos. Presentamos los resultados de tres niveles de pruebas: unitarias, para validar la lógica interna de nuestros servicios y controladores; de integración, para asegurar la correcta comunicación entre la API, la base de datos y los mecanismos de seguridad; y de aceptación, donde validamos que los flujos de negocio satisfacen las necesidades del usuario final.

Finalmente, incluimos los resultados de nuestra retrospectiva, un ejercicio de honestidad donde analizamos no solo el *producto*, sino el *proceso*, identificando los aciertos técnicos y, sobre todo, las fricciones operativas que debemos corregir para la siguiente iteración.

Desarrollo

Pruebas Unitarias

1. Resumen General de las Pruebas

Métrica	Resultado
Total de pruebas ejecutadas	77
Pruebas exitosas	77
Pruebas fallidas	0
Tasa de éxito	100%
Tiempo total de ejecución	10.7 segundos
Framework utilizado	xUnit 2.9.2
Plataforma	.NET 9.0
Fecha de ejecución	25 de noviembre de 2025

Este conjunto de pruebas confirma que el backend del proyecto opera correctamente en todos los módulos probados, sin errores funcionales ni lógicos.

2. Arquitectura y Alcance de las Pruebas

Las pruebas unitarias se desarrollaron bajo un enfoque **AAA (Arrange-Act-Assert)**, utilizando:

- **Framework**: xUnit
- **Mocking**: Moq
- **Assertions**: FluentAssertions
- **Cobertura**: Controladores y Servicios

Estructura de pruebas cubiertas:

- Controladores (Controllers): 47 pruebas
- Servicios (Services): 30 pruebas

3. Resultados por Componente

3.1 AuthController (6 pruebas)

- Autenticación por usuario y por correo
- Validación de credenciales incorrectas
- Registro de usuarios
- Manejo de duplicados

Estado: 6/6 pruebas exitosas
Tasa de éxito: 100%

3.2 ProjectController (14 pruebas)

- Creación de proyectos
- Identificadores duplicados
- Consulta individual y general
- Filtrado por estatus
- Archivado y desarchivado
- Cálculo de progreso

Estado: 14/14 pruebas exitosas

3.3 IterationController (13 pruebas)

- Validación de fechas
- Validación de nombres únicos
- Obtención por proyecto y fase
- Cálculo de progreso
- Datos burndown

Estado: 13/13 pruebas exitosas

3.4 MicroincrementController (14 pruebas)

- CRUD completo
- Validación de URLs
- Control de autoría

- Filtros por tipo e iteración

Estado: 14/14 pruebas exitosas

3.5 AuthService (8 pruebas)

- Login con diferentes flujos
- Validación de hash
- Manejo de datos duplicados
- Generación de JWT

Estado: 8/8 pruebas exitosas

3.6 IterationService (10 pruebas)

- Lógica de negocio de iteraciones
- Validación de fases
- Validación de fechas
- Cálculo de progreso

Estado: 10/10 pruebas exitosas

3.7 MicroincrementService (12 pruebas)

- Validación de tipo
- Consultas filtradas
- Autorización por usuario
- Manejo de excepciones

Estado: 12/12 pruebas exitosas

4. Métricas de Calidad

Distribución de pruebas por módulo

- **Controladores:** 61%
- **Servicios:** 39%

Distribución por tipo de operación

Tipo de operación	Cantidad
Create	16
Read/Get	24
Update	10
Delete	14
Otros	13

Estas métricas muestran que se cubren todos los caminos principales de operación del sistema.

5. Tecnologías utilizadas

Tecnología	Uso
xUnit	Pruebas unitarias
Moq	Simulación de dependencias
FluentAssertions	Validación fluida
BCrypt	Hashing seguro
coverlet	Cobertura de código

Pruebas de Integración

1. Descripción General de las Pruebas de Integración

Las pruebas de integración se enfocaron en validar el funcionamiento completo de la API REST, evaluando cómo interactúan entre sí los distintos componentes del backend cuando se ejecutan flujos reales.

Estas pruebas permiten asegurar que los controladores, servicios, repositorios, autenticación y base de datos funcionan de manera conjunta de forma correcta.

Componentes integrados probados

- Controladores (Controllers)
- Servicios de negocio (Services)
- Repositorios (Repositories)
- Base de datos (DbContext)
- Middleware de autenticación JWT
- Filtros de validación y autorización

Enfoque utilizado

Se empleó WebApplicationFactory<Program> para levantar un servidor real en memoria, permitiendo probar los endpoints mediante solicitudes HTTP completas, incluyendo:

- Flujos completos de login
- Rutas protegidas con JWT
- Validaciones de negocio reales
- Interacción real con base de datos InMemory
- Ejecución end-to-end de CRUD en todos los módulos

2. Módulos Evaluados

Módulo	Controlador	Funciones Evaluadas
Autenticación	AuthController	Registro, login, generación de tokens, validación JWT
Proyectos	ProjectController	CRUD, fases OpenUP, progreso, archivado
Iteraciones	IterationController	CRUD, asignación de fases, burndown, progreso
Microincrementos	MicroincrementController	CRUD, filtrado, evidencias, permisos de autor

---

3. Resultados por Módulo

3.1 Pruebas de Autenticación – AuthController (8 pruebas)

Validan el registro, inicio de sesión y manejo de credenciales incorrectas.

ID	Prueba	Descripción	Resultado
AUTH-01	Registro válido	Registra usuario correctamente	Correcto
AUTH-02	Email duplicado	Rechaza registro con email existente	Correcto
AUTH-03	Username duplicado	Valida duplicidad de username	Correcto
AUTH-04	Datos inválidos	Rechaza campos incompletos	Correcto
AUTH-05	Login correcto	Genera token JWT	Correcto
AUTH-06	Email incorrecto	Retorna Unauthorized	Correcto
AUTH-07	Password incorrecta	Retorna Unauthorized	Correcto

AUTH-08	Usuario inexistente	Retorna Unauthorized	Correcto

3.2 Pruebas de Proyectos – ProjectController (12 pruebas)

Evalúan la creación, consulta, filtrado, progreso y archivado de proyectos.

ID	Prueba	Descripción	Resultado
PROJ-01	Crear proyecto válido	Genera proyecto con fases OpenUP	Correcto
PROJ-02	Crear sin login	Requiere autenticación	Correcto
PROJ-03	Identificador duplicado	Rechaza duplicados	Correcto
PROJ-04	Obtener proyectos	Lista proyectos del usuario	Correcto
PROJ-05	Filtrar proyectos	Filtra por estatus	Correcto
PROJ-06	Obtener por ID	Devuelve detalles	Correcto
PROJ-07	Proyecto inexistente	Retorna NotFound	Correcto
PROJ-08	Progreso	Calcula % de avance	Correcto
PROJ-09	Actualizar	Cambia datos correctamente	Correcto
PROJ-10	Archivar	Cambia estatus a archivado	Correcto
PROJ-11	Obtener fases	Retorna fases OpenUP	Correcto
PROJ-12	Flujo completo CRUD	Ciclo de vida completo	Correcto

3.3 Pruebas de Iteraciones – IterationController (14 pruebas)

Valida creación, actualización, eliminación y progreso.

ID	Prueba	Descripción	Resultado
ITER-01	Crear válida	Crea iteración	Correcto
ITER-02	Sin autenticación	Rechaza acceso	Correcto
ITER-03	Nombre duplicado	Valida nombre único	Correcto
ITER-04	Fechas inválidas	Rechaza fechas incorrectas	Correcto
ITER-05	Obtener por ID	Devuelve detalles	Correcto
ITER-06	ID inválido	Retorna NotFound	Correcto
ITER-07	Iteraciones por proyecto	Lista completas	Correcto
ITER-08	Iteraciones por fase	Filtra correctamente	Correcto
ITER-09	Actualizar	Cambia datos	Correcto
ITER-10	Eliminar válida	Elimina correctamente	Correcto
ITER-11	Eliminar con dependencias	Rechaza borrado	Correcto
ITER-12	Progreso	Calcula avance	Correcto
ITER-13	Burndown	Devuelve datos	Correcto
ITER-14	Flujo completo CRUD	Correcto	Correcto

3.4 Pruebas de Microincrementos – MicroincrementController (15 pruebas)

Valida gestión completa de microincrementos, filtrado y permisos.

ID	Prueba	Descripción	Resultado
MICRO-01	Crear válido	Crea microincremento	Correcto
MICRO-02	Sin autenticación	Retorna Unauthorized	Correcto
MICRO-03	Iteración inexistente	Rechaza creación	Correcto
MICRO-04	Tipo Technical	Crea correctamente	Correcto
MICRO-05	Tipo Other	Crea correctamente	Correcto

MICRO-06	Obtener por ID	Devuelve detalles	Correcto
MICRO-07	ID inválido	NotFound	Correcto
MICRO-08	Obtener todos	Lista completa	Correcto
MICRO-09	Filtrar por tipo	Filtra correctamente	Correcto
MICRO-10	Actualizar válido	Guarda cambios	Correcto
MICRO-11	Actualizar evidencia	Modifica link	Correcto
MICRO-12	Actualizar por otro usuario	Rechaza	Correcto
MICRO-13	Eliminar válido	Elimina correctamente	Correcto
MICRO-14	Eliminar por otro usuario	Rechaza	Correcto
MICRO-15	Flujo completo	Correcto	Correcto

4. Resumen Global de Resultados

4.1 Estadísticas generales

Métrica	**Valor**
Total de pruebas de integración	49
Pruebas exitosas	49
Pruebas fallidas	0
Porcentaje de éxito	**100%**
Tiempo de ejecución	~35 segundos

4.2 Cobertura por módulo

AuthController → 8/8 (100%)

ProjectController: → 12/12 (100%)

IterationController → 14/14 (100%)

MicroincrementController → 15/15 (100%)

4.3 Cobertura funcional garantizada

Funcionalidad	Cobertura
Autenticación y JWT	Correcta
CRUD de proyectos	Correcta
CRUD de iteraciones	Correcta
CRUD de microincrementos	Correcta
Fases OpenUP	Correcta
Validaciones de negocio	Correcta
Datos de burndown	Correcta
Progreso por módulo	Correcta
Manejo de errores HTTP	Correcta

5. Entorno de Pruebas

5.1 Servidor de pruebas

Se utilizó una clase CustomWebApplicationFactory para:

- Crear una instancia real del servidor ASP.NET Core
- Aislar la base de datos con InMemory
- Cargar datos iniciales (roles, fases OpenUP, usuarios de prueba)

5.2 Datos sembrados

- Roles: Admin, Product Owner, Developer, Analyst
- Plantilla OpenUP
- Fases: Inception, Elaboration, Construction, Transition

Pruebas de Aceptación

1. Descripción de las pruebas de aceptación

Las pruebas de aceptación tuvieron como propósito verificar que el sistema OpenUP Tool cumpliera con los criterios de aceptación definidos en las historias de usuario, evaluando la funcionalidad desde la perspectiva del usuario final.

1.1 Objetivo

Validar que el sistema permite a los usuarios:

- Registrarse e iniciar sesión de forma segura.
- Crear y gestionar proyectos bajo la metodología OpenUP.
- Organizar el trabajo en iteraciones asociadas a las fases del proyecto.
- Registrar microincrementos de trabajo diario con diferentes tipos y evidencias.
- Visualizar el progreso del proyecto y de las iteraciones de manera clara.

1.2 Alcance

Las pruebas de aceptación cubrieron los siguientes módulos

Módulo
Autenticación
Proyectos
Iteraciones
Microincrementos
Progreso

1.3 Metodología de pruebas

Se utilizó la técnica de Pruebas de Aceptación de Usuario (UAT), siguiendo el enfoque Given–When–Then:

- Given (Dado): Se establece el contexto inicial (usuario registrado, proyecto creado, iteración existente, etc.).
- When (Cuando): El usuario realiza una acción específica (registrarse, crear proyecto, registrar microincremento, consultar progreso, etc.).

- Then (Entonces): Se valida el resultado esperado de acuerdo con los criterios de aceptación definidos en cada historia de usuario.

Las pruebas se estructuraron en casos de prueba de aceptación (UA-01 a UA-06), que agrupan escenarios como:

- UA-01: Registro e inicio de sesión (validación de cuentas nuevas, emails/usuarios duplicados y credenciales inválidas).
- UA-02: Creación de proyectos OpenUP (proyecto con las 4 fases, rol de Admin y listado de proyectos).
- UA-03: Gestión de iteraciones (fechas válidas, nombres únicos y filtrado por fase).
- UA-04: Registro de microincrementos (tipos Functional/Technical/Documentation, evidencias y permisos de edición).
- UA-05: Visualización de progreso (avance del proyecto, iteraciones y datos para burndown).
- UA-06: Flujo end-to-end desde el registro hasta la consulta del progreso del proyecto.

2. Resultados de las pruebas de aceptación

2.1 Estadísticas generales

Métrica	**Valor**
Total de casos de prueba	28
Casos aprobados	28
Casos fallidos	0
Porcentaje de aprobación	**100%**
Estado global de la UAT	APROBADO
Fecha de ejecución	25 de noviembre de 2025

2.2 Resultados por historia de usuario

Todas las historias de usuario incluidas en el alcance fueron aprobadas:

- **UA-01 – Autenticación:** 5/5 casos aprobados (registro, validación de duplicados y login con JWT).
- **UA-02 – Proyectos:** 4/4 casos aprobados (creación con template OpenUP, rol Admin, identificador único y listado).
- **UA-03 – Iteraciones:** 5/5 casos aprobados (creación, fechas válidas, nombres únicos y filtros por fase).
- **UA-04 – Microincrementos:** 6/6 casos aprobados (tipos de trabajo, evidencias y permisos del autor).
- **UA-05 – Progreso:** 3/3 casos aprobados (progreso de proyecto, progreso de iteración y datos de burndown).
- **UA-06 – Flujo end-to-end:** 5/5 pasos aprobados (registro → login → creación de proyecto → iteración → microincremento → consulta de progreso).

2.3 Cobertura de requisitos

Los principales **requisitos funcionales** quedaron validados:

- RF-01 / RF-02: Registro de usuarios y autenticación JWT.
- RF-03 / RF-04 / RF-05: Gestión de proyectos con metodología OpenUP y sus fases.
- RF-06 / RF-07 / RF-08: Gestión de iteraciones y microincrementos con tipos diferenciados.
- RF-09 / RF-10: Cálculo y visualización del progreso, así como generación de datos de burndown.

Además, se verificaron criterios no funcionales básicos:

- Tiempo de respuesta adecuado en operaciones CRUD.
- Mensajes claros en respuestas de la API.
- Protección de endpoints sensibles mediante autenticación.

Resultados de la retrospectiva del Sprint

1. **¿Qué problemas detectamos en el procedimiento que seguimos?**

- Confusión de responsabilidades entre miembros del equipo.
- Asignación individual de tareas sin coordinación.
- Trabajo excesivo en curso al mismo tiempo.
- Falta de claridad sobre con qué historia de usuario iniciar.

2. **¿Cómo pensamos corregirlos?**

- Definir roles claros para cada integrante para evitar que varias personas trabajen en lo mismo o que tareas importantes queden sin dueño.
- Establecer un límite de 3 historias simultáneas para mantener el enfoque.
- Determinar prioridades desde el inicio del sprint para que todos sepan qué historia se debe comenzar primero y por qué.
- Trabajar por parejas en tareas complejas para evitar errores.

3. **¿Qué problemas detectamos en los entregables que produjimos?**

- Incompatibilidad de versiones entre tokens JWT y Swagger.
- Errores de integración entre frontend y backend.

4. **¿Cómo pensamos corregirlos?**

- Documentar versiones exactas de cada dependencia para evitar conflictos en entornos distintos.
- Crear un contrato de API formal que describa estructuras, rutas y reglas, alineando a frontend y backend.
- Documentar cómo se genera, valida y usa el token JWT para que todos sigan el mismo proceso.
- Implementar pruebas de integración que detecten fallos antes de unir cambios al código principal.

5. **¿Qué problemas detectamos en la forma en que funciona el equipo?**

- Dificultad para que cada integrante se enfoque en un solo tema.
- Curva de aprendizaje prolongada.

6. **¿Cómo pensamos corregirlos?**

- Asignar especialización por áreas para que cada integrante domine una parte del sistema.
- Mantener documentación actualizada para evitar que el equipo pierda tiempo investigando de nuevo decisiones ya tomadas.
- Dividir tareas grandes en unidades pequeñas para facilitar avance y reducir la percepción de complejidad.

7. **¿Qué otros problemas detectamos?**

- Se olvidó cómo el diseño de la base de datos se relacionaba con las historias de usuario.
- Falta de tiempo para completar todas las actividades planificadas.

8. **¿Cómo pensamos corregirlos?**

- Añadir comentarios en A.sql que indiquen qué parte de la estructura de la base de datos responde a cada historia de usuario.
- Reducir el alcance del sprint cuando sea necesario, priorizando únicamente lo indispensable.

9. **¿Qué mejoras encontramos que podemos implementar en el próximo Sprint?**

- Mejor organización del equipo.
- Límite de 3 historias en progreso.
- Estrategia de branches ordenada.
- Comunicación centralizada.
- Mejora continua mediante documentación y demos internas.

10. **¿Cómo lo haremos?**

- Asignando roles definidos para que cada integrante sepa qué área lidera y qué decisiones le corresponden.
- Documentando versiones, decisiones técnicas y cambios clave para mantener coherencia entre todos los desarrolladores.

- Actualizando diagramas y referencias técnicas para conservar la visión completa del sistema.
- Estimando de manera más realista y dividiendo tareas grandes, asegurando una carga de trabajo alcanzable durante el sprint.

Conclusión

El cierre del Sprint 1 nos deja con un balance técnico sobresaliente y lecciones invaluables sobre nuestra dinámica de trabajo.

Desde la perspectiva del producto, los resultados son contundentes: hemos logrado un 100% de éxito en las 154 pruebas ejecutadas (77 unitarias, 49 de integración y 28 de aceptación). Esto confirma que el *backend* del sistema es sólido, seguro y capaz de soportar la lógica completa de la metodología OpenUP sin errores funcionales. La arquitectura base en .NET ha demostrado ser estable y estar lista para escalar.

Sin embargo, nuestra retrospectiva revela que la excelencia técnica contrastó con desafíos organizacionales. Reconocemos abiertamente que sufrimos problemas de coordinación, como la duplicidad de esfuerzos y conflictos de versiones entre *frontend* y *backend*, derivados de no tener roles suficientemente claros y abarcar demasiado trabajo simultáneo.