SERVICIO NACIONAL DE APRENDIZAJE SENA

Actividad: GA1-220501093-AA1-EV01. Taller. Metodologías de desarrollo de software.

Programa: Tecnología en Análisis y Desarrollo de Software

Proyecto formativo: Construcción de software integrador de tecnologías orientadas a servicios

Fase: Análisis

Proyecto: E-Commerce Sport+ S.A.S.

Instructor: Jhon Edison Nuñez Garzón

Aprendiz: Julián David Macías Garcés

Fecha: 1 de diciembre de 2025

Centro de la Tecnología del Diseño y la Productividad Empresarial — Regional Cundinamarca (Centro de Girardot)

Taller. Metodologías de Desarrollo de Software (GA1-220501093-AA1-EV01)

I. Fundamentos de las Metodologías de Desarrollo de Software

1. ¿Qué es y de qué se compone una Metodología?

Una Metodología de Desarrollo de Software es un marco de trabajo estructurado que establece un conjunto de reglas, procesos, técnicas, actividades y artefactos que un equipo debe seguir de manera sistemática para concebir, diseñar, construir, probar y mantener un sistema informático. Su propósito es asegurar un trabajo organizado, trazable y de alta calidad.

Composición Clave:

- **Fases del Ciclo de Vida:** Etapas secuenciales o iterativas (ej. Análisis, Diseño, Implementación, Pruebas).
- **Roles y Responsabilidades:** Definición clara de las funciones del equipo (ej. Desarrollador, Analista, *Scrum Master*).
- **Artefactos (Entregables):** Documentos o productos generados en cada fase (ej. ERS, Diagramas UML, Código fuente).

Utilidad (2 Datos que Demuestran su Valor):

1. **Gestión de Riesgos y Aumento de Calidad:** Permiten la identificación temprana de errores y ambigüedades (como se hizo en la fase de validación de requisitos), reduciendo el costo y la complejidad de las correcciones en etapas avanzadas de la implementación.
2. **Transparencia y Trazabilidad:** Proporcionan un mapa de ruta que facilita el seguimiento del progreso (*monitoring*) y establece una conexión verificable (*trazabilidad*) entre los requisitos del cliente y el producto de software final.

II. Clasificación y Comparación de Marcos de Trabajo

2. Características de Marcos de Trabajo: Ágil vs. Tradicional

La elección de un marco de trabajo depende de la naturaleza, la complejidad y el nivel de certeza de los requisitos del proyecto.

Característica	Marco de Trabajo Tradicional (Ej. Cascada, RUP)	Marco de Trabajo Ágil (Ej. Scrum, XP, Kanban)
Cambios	**Resistencia al Cambio**. Se gestionan formalmente y son costosos, especialmente después de la fase de Análisis.	**Aceptación del Cambio**. Adaptación rápida y continua a nuevos requisitos del cliente o del mercado.
Planificación	**Predictiva**. Se enfoca en la planificación exhaustiva y detallada al inicio del proyecto.	**Adaptativa**. Se planifica continuamente en ciclos cortos (Iteraciones o *Sprints*).
Entrega	**Entrega Única y Tardía**. El producto funcional se entrega al final de todas las fases.	**Entregas Frecuentes e Incrementales**. El software funcional se entrega en ciclos cortos de 2 a 4 semanas.
Interacción	Mínima interacción con el cliente durante las fases intermedias.	**Colaboración Continua** con el cliente (o *Product Owner*).
Énfasis	Énfasis en la Documentación completa y formal.	Énfasis en el Software Funcional que aporte valor al negocio.

3. Clasificación de Metodologías

Categoría	Metodología	Descripción Breve
Tradicional	**Modelo en Cascada**	Secuencial y lineal. Una fase debe completarse antes de comenzar la siguiente, adecuada para proyectos donde los requisitos son estables y bien entendidos.

Categoría	Metodología	Descripción Breve
Tradicional	**Proceso Racional Unificado (RUP)**	Modelo iterativo e incremental basado en la arquitectura y los casos de uso, con fases bien definidas (Incepción, Elaboración, Construcción, Transición).
Ágil	**Scrum**	Marco de trabajo liviano que organiza el desarrollo en ciclos cortos (Sprints). Se enfoca en la gestión de producto mediante la priorización del *Product Backlog*.
Ágil	**Programación Extrema (XP)**	Metodología centrada en la excelencia técnica, promueve prácticas de ingeniería como el Desarrollo Guiado por Pruebas (TDD) y la Programación en Parejas.
Ágil	**Desarrollo Rápido de Aplicaciones (RAD)**	Utiliza ciclos de desarrollo cortos y rápidos enfocados en la construcción de prototipos funcionales para obtener *feedback* temprano.

III. Aplicación al Proyecto E-Commerce Sport+ S.A.S.

4. Selección y Justificación de la Metodología

Metodología Seleccionada: Scrum

Justificación:

El proyecto **E-Commerce Sport+ S.A.S.**, al ser una plataforma digital, opera en un entorno de **alta volatilidad** y **cambios frecuentes** (nuevas tendencias, cambios de inventario, exigencias de seguridad).

1. **Adaptabilidad al Mercado (Riesgo):** Scrum permite al negocio adaptarse rápidamente a los cambios. El *Product Backlog* se re-prioriza continuamente

para enfocar el esfuerzo de desarrollo en las funcionalidades que ofrecen el mayor valor de negocio en ese momento.

2. **Validación Continua (Usabilidad y Rendimiento):** Dado que la Usabilidad (RNF-01) y el Rendimiento (RNF-05) son requisitos críticos para un e-commerce, Scrum obliga a realizar entregas funcionales al final de cada *Sprint* (Revisión del *Sprint*), permitiendo al cliente validar la UI/UX y detectar problemas de rendimiento antes del lanzamiento final.
3. **Entrega Rápida de Valor:** Facilita la implementación temprana de un Producto Mínimo Viable (MVP) (ej. un catálogo con carrito de compras básico) que pueda empezar a generar ingresos mientras se desarrollan funcionalidades secundarias.

5. Establecimiento del Plan de Actividades bajo la Metodología

La Fase de Análisis (incluyendo Requisitos, ERS y Validación) ya fue completada. Por lo tanto, el plan de Scrum se enfoca en las fases restantes (Diseño, Construcción y Pruebas).

El trabajo se organizará en Sprints que integrarán las tareas de Diseño y Construcción:

Fase del Proyecto	**Sprint Sugerido**	**Objetivo del Sprint (Sprint Goal)**	**Entregables Clave (Artefactos)**
Diseño Lógico y Arquitectónico	**Sprint 1**	Definir la estructura de datos y la arquitectura técnica del e-commerce.	**Modelo de Clases/DER** (Estructura de la base de datos). Documento de **Arquitectura y Tecnología**.
Diseño Dinámico y Base de Datos	**Sprint 2**	Modelar los flujos críticos de negocio y empezar la codificación de la persistencia.	**Diagramas de Secuencia** (Compra y Pagos). **Esquema de Base de Datos** implementado.

Fase del Proyecto	Sprint Sugerido	Objetivo del Sprint (Sprint Goal)	Entregables Clave (Artefactos)
Construcción e Interfaz (UI/UX)	**Sprint 3**	Implementar las interfaces de usuario principales y la funcionalidad básica.	**Mockups de Alta Fidelidad** (Vistas principales). **Código** funcional de Registro y Catálogo de Productos.
Pruebas y Revisión	**Fin de Cada Sprint**	Evaluar el incremento de *software* y validar el cumplimiento de los RNF (Usabilidad, Seguridad).	Demo funcional y Retroalimentación del *Product Owner*.

IV. Profundización de la Aplicación de Scrum en Sport+ S.A.S.

6. Integración de Requisitos No Funcionales (RNF) en Scrum

RNF Crítico	Ubicación en Scrum	Descripción de la Aplicación en Sport+ S.A.S.
RNF-07 Seguridad (Protección de datos y pagos)	**Definition of Done (DoD)**	Cada historia de usuario que implique manejo de datos sensibles (registro, *checkout*) DEBE incluir el cifrado de contraseñas (hashing) y la validación de *inputs* contra inyección de código como parte obligatoria del *Definition of Done* antes de ser aceptada.

RNF Crítico	Ubicación en Scrum	Descripción de la Aplicación en Sport+ S.A.S.
RNF-01 Usabilidad (Experiencia de usuario)	**Refinamiento del *Product Backlog***	Las historias de usuario de las interfaces críticas (ej. Ficha de producto, Carrito) deben ser desglosadas en tareas específicas que incluyan el diseño de **Mockups de Alta Fidelidad** (Actividad 4) y la revisión por un experto en UX.
RNF-05 Rendimiento (Tiempos de respuesta)	**Pruebas del *Sprint***	Se incluirán pruebas de carga (simulando 100 usuarios concurrentes) en los *endpoints* críticos (búsqueda de producto, finalización de compra) al finalizar cada *Sprint* para garantizar que la plataforma cumpla con los tiempos de respuesta inferiores a 3 segundos.

7. Estructura Inicial del *Product Backlog* (Épicas y Features)

Muestra cómo la ERS se traduce en el formato ágil. Define las categorías más grandes de trabajo (Épicas) y desglosa en funcionalidades (Features), las cuales luego se convertirán en Historias de Usuario para los *Sprints*.

Jerarquía Ágil	Título (Épica o Feature)	Origen ERS / Descripción
ÉPICA 1	**Gestión de Ventas y *Checkout***	Cubre todos los requisitos para que un cliente pueda comprar (RF-01, RF-02, RF-03).

Jerarquía Ágil	**Título (Épica o Feature)**	**Origen ERS / Descripción**
Feature	Flujo de Carrito de Compras	Permite añadir/eliminar productos y calcular el subtotal.
Feature	Integración de Pasarela de Pago	Conexión con *PayU/ePayco* para procesar la transacción y generar notificación (RNF-09).
ÉPICA 2	**Gestión de Producto e Inventario**	Cubre los requisitos para que el Administrador mantenga el catálogo y el stock (RF-04, RF-05).
Feature	CRUD de Productos	Permite al administrador crear, leer, actualizar y eliminar productos.
Feature	Alerta de Stock Mínimo	Notifica al administrador cuando el *stock* de un producto sea inferior a 10 unidades.

8. Gestión de Riesgos bajo la Óptica de Scrum

Riesgo del E-Commerce	**Mitigación con Scrum**	**Frecuencia de Mitigación**
Cambio de Requisitos/Tendencias del Mercado: La moda deportiva cambia y los clientes quieren nuevas funcionalidades.	**Priorización del *Product Backlog***	**Continuo** (antes de cada *Sprint*). El *Product Owner* ajusta el orden de las Historias de Usuario, asegurando que el equipo trabaje en lo más valioso.
Fallo en la Integración de Pagos: La pasarela no funciona o hay un error de comunicación.	**Desarrollo Iterativo y Revisión del *Sprint***	**Fin de cada *Sprint***. La integración se aborda en una iteración temprana para ser probada y validada por el cliente antes de que se construya el resto del sistema sobre ella.
Baja Adopción del Usuario (Mala UX): La interfaz es confusa y los clientes abandonan la compra.	**Revisión del *Sprint* y *Feedback* Constante**	**Fin de cada *Sprint***. Se obtiene *feedback* directo del cliente o *stakeholders* sobre el *software* funcional entregado, permitiendo correcciones inmediatas de usabilidad.

V. Conclusión

La selección del marco de trabajo Scrum para el proyecto E-Commerce Sport+ S.A.S. no es solo una elección metodológica, sino una decisión estratégica que garantiza la adaptabilidad y la entrega continua de valor en un entorno digital dinámico.

La exhaustiva fase de Análisis ya completada (la cual generó el ERS, el mapa de procesos y la validación de requisitos) sienta una base sólida para el desarrollo ágil. Scrum, mediante la división del trabajo en *Sprints* y el uso del *Product Backlog*, permite que los riesgos asociados a los cambios del mercado o a los Requisitos No Funcionales (como la Usabilidad y la Seguridad) se aborden y mitiguen de forma incremental, no al final.

En resumen, la aplicación de Scrum transforma la rigidez de un plan lineal en un proceso de aprendizaje constante, asegurando que cada incremento de *software* (desde el modelado de datos hasta la implementación de interfaces) esté perfectamente alineado con los objetivos de negocio de Sport+ S.A.S. y cumpla con los más altos estándares de calidad y rendimiento definidos en la ingeniería de *software*.

VI. Referencias Bibliográficas

Fuentes Académicas y Técnicas

IEEE Computer Society. (2018). *ISO/IEC/IEEE 29148:2018 - Systems and software engineering - Life cycle processes - Requirements engineering*. IEEE.

Pressman, R. S. (2010). *Ingeniería del software: Un enfoque práctico* (7.ª ed.). McGraw-Hill.

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. Scrum.org.

Sommerville, I. (2011). *Ingeniería del software* (9.ª ed.). Pearson Educación.

Fuentes SENA

Servicio Nacional de Aprendizaje (SENA). (2024). *Fundamentos de análisis y diseño de software*. (Documento de apoyo para la formación).

Servicio Nacional de Aprendizaje (SENA). (2025). *Metodologías de desarrollo de software*. (Material de apoyo de la actividad GA1-220501093-AA1).