Trabajo Práctico 1: Introducción a Node.js y Metodología Scrum en la Educación

Índice

---

1. Metodología Scrum aplicada a la educación

1.1 Introducción a Scrum

Scrum es una metodología ágil diseñada para organizar y gestionar el desarrollo de proyectos complejos. Su enfoque permite a los equipos trabajar de forma colaborativa y eficiente para alcanzar objetivos comunes a través de ciclos cortos de trabajo llamados **sprints.**

> 📘 **Documentación respaldatoria:**
> Scrum es un marco de trabajo ágil que se enfoca en la mejora continua y en la entrega de productos funcionales a través de iteraciones regulares. Está estructurado en ceremonias, roles y artefactos que aseguran un flujo de trabajo eficiente y colaborativo.

1.2 Las ceremonias clave en Scrum

Scrum se estructura alrededor de cuatro ceremonias fundamentales que aseguran el progreso continuo y el ajuste del equipo en cada sprint:

1. Sprint Planning (Planificación del Sprint)

- 🗓️ **Cuándo:** Al inicio de cada sprint, generalmente los lunes.
- 🔍 **Qué es:** Una reunión donde se decide el trabajo que se realizará durante la duración del Sprint.
- 🎯 **Objetivo:** Definir un plan claro para todo el equipo sobre qué tareas se realizarán y quién será responsable de cada una.

2. Daily Stand-up (Reunión diaria)

- 🗓️ **Cuándo:** Todos los días, internamente en el equipo y con reuniones oficiales con el docente los Lunes Miercoles y Viernes.
- 🕐 **Duración:** 5-10 minutos. Se puede extender
- 🔍 **Qué es:** Cada miembro del equipo responde tres preguntas clave:
  1. ¿Qué hice ayer?
  2. ¿Qué haré hoy? ¿Qué tengo planeado resolver hasta la proxima daily?
  3. ¿Hay algo que me está bloqueando?

  El docente utilizara este espacio como reunión de seguimiento y resolución de problemas. Se resolveran las dudas que hayan sido cargadas en el formulario de consultas.
- 🎯 **Objetivo:** Mantener a todos alineados y resolver obstáculos rápidamente.

3. Sprint Review (Revisión del Sprint)

- 🗓️ **Cuándo:** Desde el 2do. Sprint. Se realiza despues de las Daily a personas seleccionadas.
- 🔍 **Qué es:** La persona presenta el trabajo completado del Sprint anterior, recibe retroalimentación del docente.
- 🕐 **Duración:** 5-15 minutos. Se puede extender
- 🎯 **Objetivo:** Validar el trabajo realizado y generar aprendizaje a partir de la retroalimentación recibida.

4. Sprint Retrospective (Retrospectiva)

- 🗓️ **Cuándo:** Después de la Sprint Review.
- 🔍 **Qué es:** Reflexión interna del equipo sobre el trabajo realizado. ¿Qué salió bien? ¿Qué se puede mejorar?
- 🎯 **Objetivo:** Identificar oportunidades de mejora para el próximo sprint.

1.3 Roles en Scrum

En este módulo, los equipos de trabajo estarán organizados en grupos de 4 personas, cada una con roles definidos:

- **Product Owner y Stakeholder**: Responsable de liderar la planificación y organización del equipo. Este rol lo desarrollaremos como Docentes, les asignan los requerimientos a cumplir y una vez presentado evaluaremos si cumplen el requisito solicitado.
- **Desarrolladores**: Trabajan en las tareas asignadas, participando activamente en las reuniones diarias y en la ejecución del plan. Colaboran con los miembros del equipo.

1.4 Uso de Trello para la gestión ágil de proyectos

Para organizar las tareas de cada equipo, utilizaremos **Trello**, una herramienta de gestión de proyectos visual y fácil de usar. Cada equipo contará con un tablero dividido en las siguientes columnas:

1. **Backlog del Sprint**: Tareas pendientes de realizar durante el sprint.
2. **En progreso**: Tareas en las que el equipo está trabajando actualmente.
3. **Bloqueado/En espera**: Tareas que no pueden completarse por alguna razón y necesitan atención.
4. **Completado**: Tareas terminadas.

---

2. Introducción a Node.js

2.1 ¿Qué es Node.js?

Node.js es un entorno de ejecución de JavaScript que permite ejecutar código del lado del servidor, más allá del navegador. Está construido sobre el motor V8 de Google Chrome, y se caracteriza por ser altamente eficiente y escalable, gracias a su arquitectura basada en eventos y su manejo de entrada/salida no bloqueante.

Según el libro *Node.js for Beginners (2024)*, Node.js permite crear aplicaciones con capacidad de manejar grandes volúmenes de solicitudes simultáneas sin necesidad de crear hilos adicionales, lo que lo convierte en una excelente opción para aplicaciones en tiempo real.

2.2 Arquitectura de Node.js

Node.js se basa en un **modelo de un solo hilo** (single-threaded) y un **bucle de eventos** (event loop), lo que permite manejar operaciones asíncronas sin bloquear el flujo de ejecución. Este modelo es ideal para aplicaciones con muchas operaciones de entrada/salida, como servidores web o APIs.

2.3 Instalación de Node.js y configuración del entorno de desarrollo

Instalación de Node.js

1. **Descarga**: Visita nodejs.org y descarga la versión LTS.
2. **Instalación**: Sigue los pasos del instalador, aceptando la configuración predeterminada.
3. **Verificación**: Abre la terminal y ejecuta `node -v` para verificar la instalación.

Instalación de Visual Studio Code

1. **Descarga**: Desde code.visualstudio.com, descarga la versión para tu sistema operativo.
2. **Instalación**: Sigue los pasos del instalador.
3. **Configuración inicial**: Instala extensiones recomendadas como **Node.js Extension Pack**, **ESLint** y **Prettier**.

2.4 Casos de uso de Node.js

Node.js es ideal para:

- **Aplicaciones en tiempo real**: Como chats, juegos en línea, aplicaciones colaborativas.
- **APIs RESTful**: Ideal para servir datos JSON en aplicaciones web o móviles.
- **Automatización de tareas**: Scripts y herramientas de línea de comandos.
- **Aplicaciones de streaming**: Plataformas de video y audio en tiempo real.

2.5 Ventajas y desventajas de Node.js

Ventajas:

- **Escalabilidad**: Maneja múltiples conexiones simultáneas de manera eficiente.
- **Un solo lenguaje**: Permite utilizar JavaScript tanto en el frontend como en el backend.
- **Ecosistema robusto**: Gran cantidad de módulos disponibles en npm para acelerar el desarrollo.

Desventajas:

- **CPU intensiva**: No es ideal para aplicaciones que requieren un uso intensivo del procesador.
- **Callback Hell**: Aunque se puede mitigar con Promesas y `async/await`, el uso excesivo de callbacks puede complicar el código.

2.6 Comparativa con otras tecnologías

Node.js vs. Python (Flask, Django):

- **Rendimiento**: Node.js suele ser más eficiente en I/O gracias a su arquitectura no bloqueante.
- **Lenguaje**: JavaScript vs. Python; la elección depende de las habilidades del equipo.

Node.js vs. Java (Spring Boot):

- **Concurrencia**: Node.js usa un solo hilo y bucle de eventos, mientras que Java utiliza múltiples hilos para manejar la concurrencia.
- **Ecosistema**: Java tiene un ecosistema más maduro para aplicaciones empresariales, pero Node.js ofrece un desarrollo más rápido y ligero.

2.7 Ejemplos prácticos y detallados

2.7.1 Ejemplo: Servidor básico en Node.js

Objetivo: Crear un servidor que responda con "Hola Mundo" a las solicitudes HTTP.

```
import http from 'http';

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola Mundo\n');
});

server.listen(port, hostname, () => {
console.log(`Servidor corriendo`);
});
```