CUC- TECNICATURA EN DESARROLLO Y CALIDAD DE SOFTWARE - Proyecto Final 3° Año - Ing. Luis Antonio Juárez

Ejemplo de proyecto de software para la Tecnicatura en Desarrollo y Calidad de Software de la Universidad del Norte Santo Tomás de Aquino:

Nombre del proyecto: Sistema de gestión de guardias médicas

Objetivos:
- Desarrollar un sistema web que permita gestionar la asignación de guardias médicas en diferentes centros de salud.
- Aplicar los conocimientos adquiridos en la tecnicatura sobre desarrollo de software, bases de datos, Frontend, Backend, control de calidad, etc.
- Trabajar en equipo siguiendo metodologías ágiles para la gestión del proyecto.

Descripción:
El sistema permitirá registrar médicos, centros de salud y disponibilidad de guardias. Los administradores podrán asignar guardias respetando reglas como límites de horas por médico, incompatibilidades entre médicos y centros, etc. Los médicos podrán consultar su calendario de guardias asignadas.

Tecnologías a utilizar:
- Frontend: HTML, CSS, JavaScript, frameworks como Angular o React
- Backend: Java, Spring, Hibernate
- Base de datos: SQL
- Herramientas de control de calidad: TestLink, Selenium, JUnit

Fases del proyecto:

1. Análisis de requerimientos
   - Relevar las necesidades con los stakeholders
   - Especificar casos de uso
   - Definir criterios de aceptación

2. Diseño
   - Diseño de la arquitectura
   - Modelo de datos
   - Prototipos de interfaz de usuario

3. Codificación
   - Setup del ambiente de desarrollo
   - Implementación iterativa de los módulos Frontend y Backend
   - Scripting de casos de prueba

4. Pruebas

- Pruebas unitarias
- Pruebas de integración
- Pruebas de sistema
- Pruebas de aceptación

5. Despliegue
   - Preparación del ambiente de producción
   - Migraciones de datos
   - Capacitación de usuarios

6. Mantenimiento
   - Corrección de defectos
   - Gestión de cambios y nuevos requerimientos

Se trabajará con la metodología Scrum, con entregas incrementales cada seman. El proyecto completo tendría una duración estimada de 4 meses.

Este proyecto permitiría aplicar los conocimientos adquiridos sobre programación, bases de datos, gestión de proyectos, pruebas y aseguramiento de calidad e involucrará trabajo en equipo, tal como lo requiere el perfil de la Tecnicatura.

Respaldo y recuperación de datos:
- Se implementarán rutinas de respaldo (backup) de la base de datos y del código fuente cada 24 horas
- Los respaldos se almacenarán en un repositorio remoto para su recuperación en caso de fallas
- Se definirá un plan de recuperación ante desastres para restablecer el sistema rápidamente

Pruebas y control de calidad:
- Se aplicarán pruebas unitarias con JUnit para validar los componentes individuales
- Las pruebas de integración con Selenium verificarán el correcto ensamblaje de los módulos
- Las pruebas de sistema con casos de prueba end-to-end en TestLink validarán los requerimientos
- Se realizarán pruebas de desempeño, seguridad, usabilidad previo al pase a producción
- Se utilizarán herramientas de análisis de cobertura de código y control de calidad estática
- Se definirán métricas y umbrales de calidad aceptables

Metodología Scrum:
- El dueño del producto (Product Owner) priorizará y mantendrá el backlog del producto
- Se realizarán Sprints de 2 semanas con entregas parciales
- Habrá reuniones diarias breves de sincronización (daily standup)
- Al iniciar cada Sprint se realizará la reunión de planificación del Sprint

- Al finalizar cada Sprint se realizará la revisión de la incrementa y la retrospectiva

Herramientas de comunicación del equipo:
- Trello para la gestión del backlog, tareas y tablero Scrum
- Microsoft Teams para videollamadas, mensajería y compartir documentación
- GitHub o similar para el control de versiones del código
- Confluence/SharePoint para mantener el conocimiento del proyecto
- Correo electrónico para comunicaciones oficiales

De esta forma se complementa la propuesta inicial con buenas prácticas de respaldo, pruebas exhaustivas siguiendo un proceso de calidad, aplicación de la metodología Scrum y utilización de las herramientas de comunicación y colaboración adecuadas para un trabajo coordinado del equipo.

La metodología Scrum es un enfoque ágil para el desarrollo de software que se basa en la iteración y la colaboración continua entre los miembros del equipo. Aquí tienes una explicación simple de cada componente de Scrum, junto con un ejemplo:

1. Roles en Scrum:

- **Product Owner:**

  - Responsabilidad: Representa los intereses del cliente y define las funcionalidades del producto.
  - Ejemplo: Ana es la Product Owner de un proyecto para desarrollar una aplicación de gestión de proyectos. Ella trabaja estrechamente con el equipo de desarrollo para priorizar las características más importantes para los usuarios.

- **Scrum Master:**

  - Responsabilidad: Facilita el proceso Scrum y elimina obstáculos que puedan afectar al equipo.
  - Ejemplo: Juan es el Scrum Master en el equipo de desarrollo. Su papel es asegurarse de que todas las reuniones Scrum se realicen correctamente y ayudar al equipo a superar cualquier problema que pueda surgir durante el desarrollo.

- **Equipo de Desarrollo:**

- Responsabilidad: Grupo de profesionales encargados de convertir los requisitos en incrementos potencialmente entregables.
- Ejemplo: El equipo de desarrollo incluye a desarrolladores, diseñadores y testers. Trabajan juntos para implementar las funcionalidades definidas por el Product Owner.

2. Artefactos en Scrum:

- **Product Backlog:**

- Descripción: Lista priorizada de todas las funcionalidades deseadas del producto.
- Ejemplo: El Product Backlog de la aplicación de gestión de proyectos incluye elementos como "crear tareas", "asignar responsables" y "generar informes".

- **Sprint Backlog:**

- Descripción: Lista de tareas específicas seleccionadas del Product Backlog para abordar durante un sprint.
- Ejemplo: Para el Sprint 1, el equipo selecciona tareas del Product Backlog, como "diseñar interfaz de usuario" y "implementar base de datos".

- **Incremento del Producto:**

- Descripción: La versión actualizada y potencialmente entregable del producto después de completar un sprint.
- Ejemplo: Al final del Sprint 1, el equipo ha implementado las funciones de diseño de interfaz de usuario y base de datos, lo que constituye el Incremento del Producto.

3. Eventos en Scrum:

- **Sprint Planning Meeting:**

- Descripción: Reunión al inicio de cada sprint para planificar las tareas.
- Ejemplo: En el Sprint Planning Meeting, el equipo discute y selecciona las tareas del Product Backlog que abordarán durante el próximo sprint.

- **Daily Scrum:**

- Descripción: Reunión diaria corta para que el equipo de desarrollo sincronice actividades.
- Ejemplo: Durante la Daily Scrum, cada miembro del equipo comparte lo que hizo ayer, lo que hará hoy y cualquier obstáculo que esté enfrentando.

- **Sprint Review:**

- Descripción: Reunión al final de cada sprint para revisar el incremento y ajustar el Product Backlog si es necesario.
- Ejemplo: Al final del Sprint 1, el equipo presenta el Incremento del Producto y el Product Owner decide agregar nuevas funcionalidades al Product Backlog.

- **Sprint Retrospective:**

- Descripción: Reunión al final de cada sprint para reflexionar sobre el proceso y mejorar.
- Ejemplo: Después del Sprint 1, el equipo se reúne para discutir lo que funcionó bien y las áreas que podrían mejorarse, implementando cambios para el próximo sprint.

Scrum es un marco ágil flexible y adaptable, y estos componentes ayudan a los equipos a colaborar eficientemente y a entregar valor de manera regular.

1. Gestión de proyectos □

Estos software permiten organizar tareas, establecer plazos y hacer seguimiento del progreso del equipo.

- **Trello** → Gestión visual de tareas con tableros y listas. Ideal para metodologías ágiles como Kanban.
- **Jira** → Software avanzado para la gestión de proyectos ágiles, enfocado en equipos de desarrollo. Permite gestionar sprints y errores.
- **Asana** → Organización de tareas y proyectos con diagramas de Gantt y flujos de trabajo colaborativos.
- **Microsoft Project** → Herramienta profesional para planificación de proyectos, cronogramas y gestión de recursos.
- **Notion / Confluence** → Espacios colaborativos para documentación, notas y gestión de información.

---

2. Desarrollo y programación □

Software y herramientas utilizadas para escribir, ejecutar y gestionar el código del proyecto.

- **Visual Studio Code** → Editor de código liviano y potente con soporte para múltiples lenguajes y extensiones.
- **XAMPP** → Paquete que incluye Apache, MySQL y PHP, ideal para correr servidores locales en desarrollo web.
- **MySQL Workbench** → Herramienta para diseñar, gestionar y administrar bases de datos MySQL.
- **Node.js con nvm** → Plataforma para ejecutar JavaScript del lado del servidor, con nvm para gestionar versiones.
- **Laravel 7** → Framework PHP para desarrollo backend con arquitectura MVC.
- **Angular** → Framework frontend de Google para construir aplicaciones web dinámicas y escalables.

3. Comunicación y colaboración □

Estas herramientas facilitan la comunicación entre los miembros del equipo y la colaboración en tiempo real.

- **Slack / Microsoft Teams** → Chats grupales, llamadas y gestión de canales de comunicación en equipos.
- **Google Drive / OneDrive** → Almacenamiento en la nube para compartir documentos, imágenes y archivos del proyecto.
- **Zoom / Google Meet** → Herramientas para reuniones en línea, ideal para revisiones de código y planificación de tareas.

---

4. Control de versiones □

Esenciales para gestionar el código fuente, colaborar y evitar conflictos entre desarrolladores.

- **Git** → Sistema de control de versiones distribuido.
- **GitHub / GitLab / Bitbucket** → Plataformas de repositorios remotos para alojar el código del proyecto y gestionar versiones.

---

5. Pruebas y despliegue □

Herramientas para probar el código antes de su lanzamiento y gestionar entornos de producción.

- **Postman / Insomnia** → Software para probar APIs REST y realizar pruebas de integración.
- **Docker** → Virtualización de entornos de desarrollo y producción para garantizar consistencia en la ejecución del software.

---

6. Diseño y prototipado □

Software para diseñar interfaces y prototipos interactivos.

- **Figma** → Herramienta colaborativa en la nube para diseño UI/UX y prototipado de interfaces.

- **Adobe XD** → Alternativa potente para crear prototipos interactivos con integraciones a otros productos de Adobe.