Universidad Corporativa Coppel | Programas Académicos

Ingeniería en Desarrollo de Software

Actividad 1 | Herramienta para la gestión de proyectos

Ingeniería de software 1

Ingeniería en Desarrollo de Software

academia global

TUTOR: Eduardo Israel Castillo García

ALUMNO: Kevin Iván Flores Pedraza

FECHA: 06/10/24

Índice

Introducción 3

Descripción 3

Justificación 5

Desarrollo 5

Contextualización: 5
Actividad: 5
Plan de proyecto 5
Calendarización del proyecto 10
Control de versiones de software. 11

Conclusión 13

Referencias 13

Introducción

La administración de proyectos de software es un proceso que implica la planificación, organización y control de todas las actividades necesarias para desarrollar un producto de software. Su objetivo principal es asegurar que el proyecto se entregue a tiempo, dentro del presupuesto y con la calidad esperada. Para ello, los gestores de proyectos utilizan diversas metodologías y herramientas, como Scrum, Kanban o el enfoque en cascada, adaptadas a las necesidades del equipo y del proyecto en cuestión. Estos enfoques ayudan a gestionar riesgos, coordinar equipos, asignar tareas y controlar el progreso.

Además, la administración de proyectos de software involucra la comunicación constante con las partes interesadas, incluyendo desarrolladores, clientes y otros actores clave. La interacción eficaz con todos estos grupos asegura que los requisitos del proyecto se comprendan correctamente y que cualquier cambio o desafío se aborde de manera ágil. A lo largo del proyecto, se emplean técnicas de monitoreo y evaluación para medir el rendimiento y hacer los ajustes necesarios, minimizando el riesgo de fallos y maximizando el valor del software desarrollado.

Descripción

La administración de proyectos de software es el proceso de planificar, organizar, dirigir y controlar las actividades necesarias para desarrollar un producto de software de acuerdo con los objetivos específicos, como tiempo, costo, calidad y alcance. Involucra la coordinación de recursos humanos, técnicos y financieros, con el fin de cumplir con los requerimientos del cliente o usuario, y garantizar que el software sea funcional, eficiente y escalable.

Características principales de la administración de proyectos de software:

1. Administradores de Proyectos de Software:

- Liderazgo y coordinación: Los administradores de proyectos son responsables de liderar el equipo de desarrollo, definir prioridades y asignar tareas. Son los encargados de facilitar la comunicación entre el equipo técnico y las partes interesadas, como clientes o ejecutivos.
- Gestión de riesgos: Identifican y mitigan riesgos que puedan impactar el éxito del proyecto, como retrasos, sobrecostos o problemas técnicos.

- Control del proyecto: Supervisan el progreso del proyecto, aseguran el cumplimiento del cronograma, y ajustan los recursos cuando es necesario para cumplir con los plazos.

2. Métodos para la planificación:

- Metodologías ágiles: Enfoques como Scrum y Kanban son comunes en la gestión de proyectos de software. Se centran en iteraciones cortas (sprints), retroalimentación constante y flexibilidad para adaptarse a cambios rápidos.
- Enfoque en cascada: Es un método más tradicional que sigue un proceso lineal de etapas bien definidas (requisitos, diseño, desarrollo, pruebas, implementación). Aunque es menos flexible que las metodologías ágiles, sigue siendo útil en proyectos donde los requisitos son claros y no cambian mucho.
- Modelo híbrido: Combina elementos de los enfoques ágil y en cascada, dependiendo de las necesidades y características específicas del proyecto.

3. Herramientas disponibles:

- Gestión de tareas y colaboración: Herramientas como Trello, Jira o Asana permiten asignar y hacer seguimiento de tareas en tiempo real, ofreciendo visibilidad del progreso.
- Control de versiones: Herramientas como GitHub o GitLab son cruciales para gestionar el código fuente, permitiendo a los desarrolladores colaborar en el mismo proyecto sin conflictos.
- Monitoreo de proyectos: Microsoft Project, Monday.com o ClickUp permiten planificar cronogramas, asignar recursos, y realizar un seguimiento detallado de la evolución del proyecto en tiempo real.
- Automatización y CI/CD: Herramientas como Jenkins o CircleCI ayudan a automatizar pruebas, integración continua y despliegue, mejorando la eficiencia del proceso de desarrollo.

Estas características permiten a los administradores mantener un control preciso y flexible sobre los proyectos de software, garantizando que se entreguen de manera eficiente y cumpliendo los objetivos del negocio.

Justificación

La correcta planificación de un proyecto de software es fundamental porque permite establecer una hoja de ruta clara para alcanzar los objetivos propuestos. Una buena planificación asegura que los requisitos del proyecto se entiendan correctamente desde el inicio, evitando malentendidos o cambios drásticos durante el desarrollo. Al definir claramente los alcances, plazos y recursos necesarios, se puede coordinar eficientemente al equipo de trabajo, asignar las tareas adecuadas a cada miembro y prever posibles desafíos o riesgos. Esto reduce la probabilidad de retrasos y sobrecostos, aumentando la probabilidad de éxito.

Además, una planificación adecuada mejora la comunicación entre las partes interesadas, como el equipo de desarrollo, los clientes y otros actores involucrados. Permite establecer expectativas realistas y factibles, lo que facilita el seguimiento del progreso del proyecto y la toma de decisiones rápidas cuando surgen problemas. También ayuda a controlar los cambios de alcance, que pueden desviar al proyecto si no se gestionan correctamente, asegurando que el producto final cumpla con los requisitos iniciales y se entregue dentro del presupuesto y los plazos previstos.

Desarrollo

Contextualización:

En la actividad anterior se analizó el proyecto solicitado por la empresa y se promovió la selección del marco de trabajo y la manera de trabajar de forma colaborativa, ahora se pretende simular el proceso de gestión del proyecto y el control de versiones de software, por tal es importante identificar que método o técnica puede funcionar de mejor manera para la empresa.

Actividad:

Con base al método de desarrollo de software clarificar:

- Plan del proyecto
- Calendarización del proyecto
- Seleccionar un software para el control de versiones y justificar su elección.

Plan de proyecto

a) Datos generales

Visión general	El Sistema de Información y Planificación de Recursos Empresariales (ERP) desarrollado por ITPower para ExpoFull S.A. tiene como objetivo centralizar y optimizar la gestión de los más de 50 puntos de venta distribuidos a nivel nacional. ExpoFull, especializada en la venta de ropa y calzado, busca integrar todas sus operaciones comerciales, logísticas y administrativas en una única plataforma, facilitando así la toma de decisiones informadas y mejorando la eficiencia de sus procesos. El ERP permitirá a la empresa gestionar sus ventas, compras, inventarios, empleados y clientes de manera eficiente, generando reportes detallados para el control y análisis en tiempo real.
Alcance	El sistema ERP abarca los principales procesos comerciales y operativos de ExpoFull, incluyendo la gestión del punto de venta, compras, inventarios, clientes y empleados. A través de una Pantalla de Acceso centralizada, los usuarios accederán a los siguientes módulos: • Punto de Venta (POS): Procesamiento de ventas en las tiendas físicas, con integración de pagos, descuentos y gestión de inventarios en tiempo real. • Módulo de Compras: Control de órdenes de compra, seguimiento de proveedores, y optimización de la adquisición de mercancías. • Registro de Clientes: Base de datos para gestionar la información de los clientes, historial de compras, y programas de fidelización. • Registro de Empleados: Control y seguimiento del personal, gestión de horarios, asistencia y datos de contacto. • Generación de Reportes: Análisis detallado de ventas, compras e inventarios para la toma de decisiones estratégicas. • Gestión de Inventarios: Monitoreo de existencias, control de niveles de stock y gestión de inventarios en cada punto de venta.
objetivos	• Centralizar la información: Integrar todos los procesos operativos, comerciales y administrativos de ExpoFull en una única plataforma accesible desde los más de 50 puntos de venta. • Mejorar la eficiencia operativa: Automatizar y optimizar las tareas relacionadas con ventas, compras, inventarios y la gestión de empleados, reduciendo tiempos de respuesta y mejorando la productividad. • Facilitar la toma de decisiones: Proporcionar reportes y análisis en tiempo real de las ventas, compras e inventarios, permitiendo a ExpoFull realizar ajustes estratégicos rápidos y basados en datos. • Garantizar la escalabilidad: Diseñar el sistema para que sea capaz de soportar el crecimiento futuro de la empresa, tanto en términos de puntos de venta como de volumen de operaciones. • Mejorar la experiencia del cliente: Ofrecer un sistema robusto que permita a ExpoFull gestionar mejor las relaciones con sus clientes, incluyendo la creación de programas de fidelización y una atención más personalizada.

b) Roles y responsabilidades

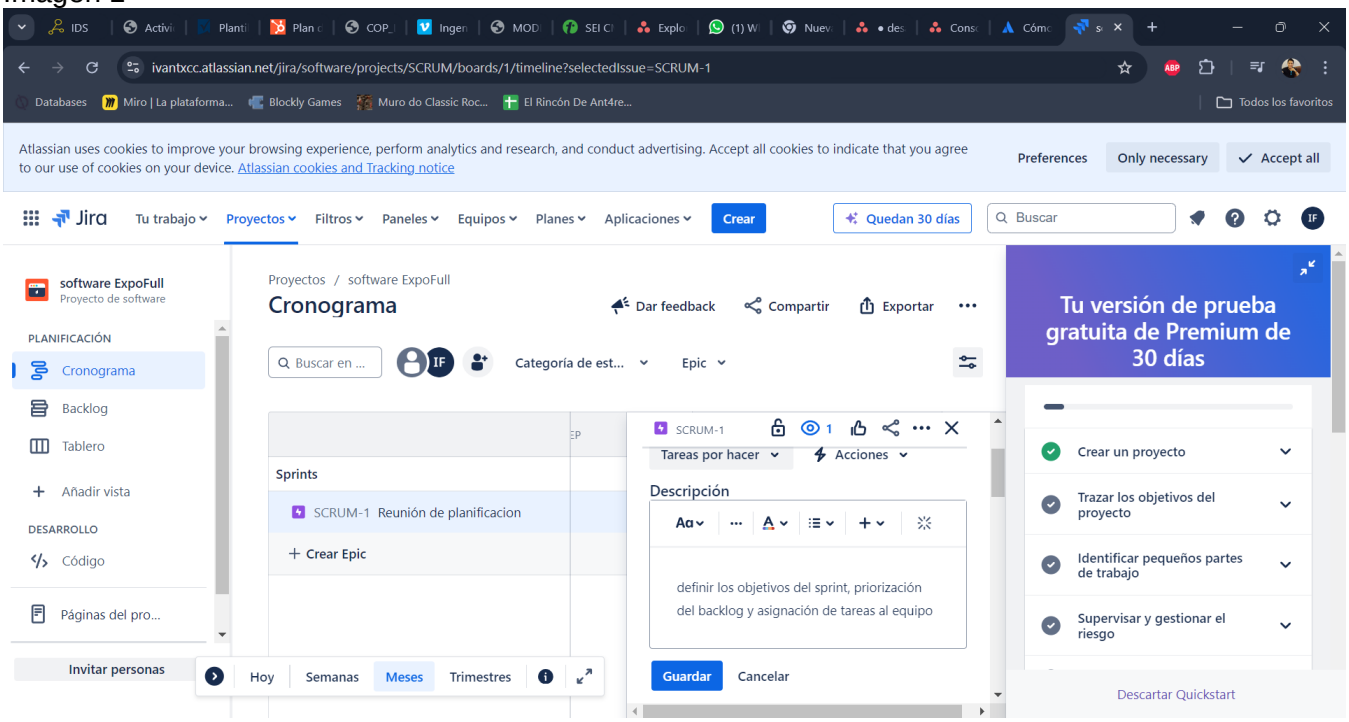
Nombre	Rol	Responsabilidades
Carlos Martínez	Product Owner	• Definir y priorizar las funcionalidades del ERP. • Ser el enlace principal entre ExpoFull y el equipo de desarrollo. • Asegurarse de que el sistema ERP cumpla con las expectativas y necesidades del cliente.
Lucía Gómez	Scrum Master	• Facilitar las reuniones Scrum (daily stand-ups, sprint planning, sprint review, retrospectivas). • Eliminar impedimentos que afecten al equipo de desarrollo. • Garantizar que el equipo siga los principios ágiles.
Juan Pérez	Desarrollador Backend	• Implementar la lógica del servidor, bases de datos, y APIs del ERP. • Desarrollar y gestionar el Módulo de Gestión de Inventarios y el Módulo de Compras. • Asegurar la integración con otros sistemas de ExpoFull.
Ana Torres	Desarrollador Frontend	• Diseñar y desarrollar las interfaces de usuario para

		los módulos del ERP. • Implementar el Punto de Venta (POS) y la Pantalla de Acceso. • Asegurar la usabilidad y accesibilidad del sistema en todos los dispositivos.
Roberto Sánchez	Ingeniero de Base de Datos	• Diseñar y gestionar la base de datos para almacenar la información de clientes, empleados, ventas, compras e inventarios. • Optimizar el rendimiento de las consultas y garantizar la seguridad de los datos.
María López	Analista de Negocios	• Recolectar y analizar los requisitos del cliente. • Crear documentación detallada para el equipo de desarrollo. • Asegurar que el ERP se alinee con los procesos de negocio de ExpoFull.
Javier Ramírez	Tester/QA	• Crear y ejecutar casos de prueba para garantizar que el ERP funcione correctamente. • Realizar pruebas de integración, usabilidad, y seguridad. • Identificar y reportar

		errores, asegurando que sean corregidos antes del lanzamiento.
Laura Fernández	Especialista DevOps	• Gestionar el entorno de desarrollo, integración y despliegue continuo (CI/CD). • Supervisar la infraestructura y servidores donde se desplegará el ERP. • Automatizar los procesos de prueba y despliegue.
Fernando Castillo	Especialista de Seguridad	• Implementar medidas de seguridad para proteger los datos del ERP. • Monitorear vulnerabilidades y realizar auditorías de seguridad. • Garantizar el cumplimiento de normativas de privacidad de datos.

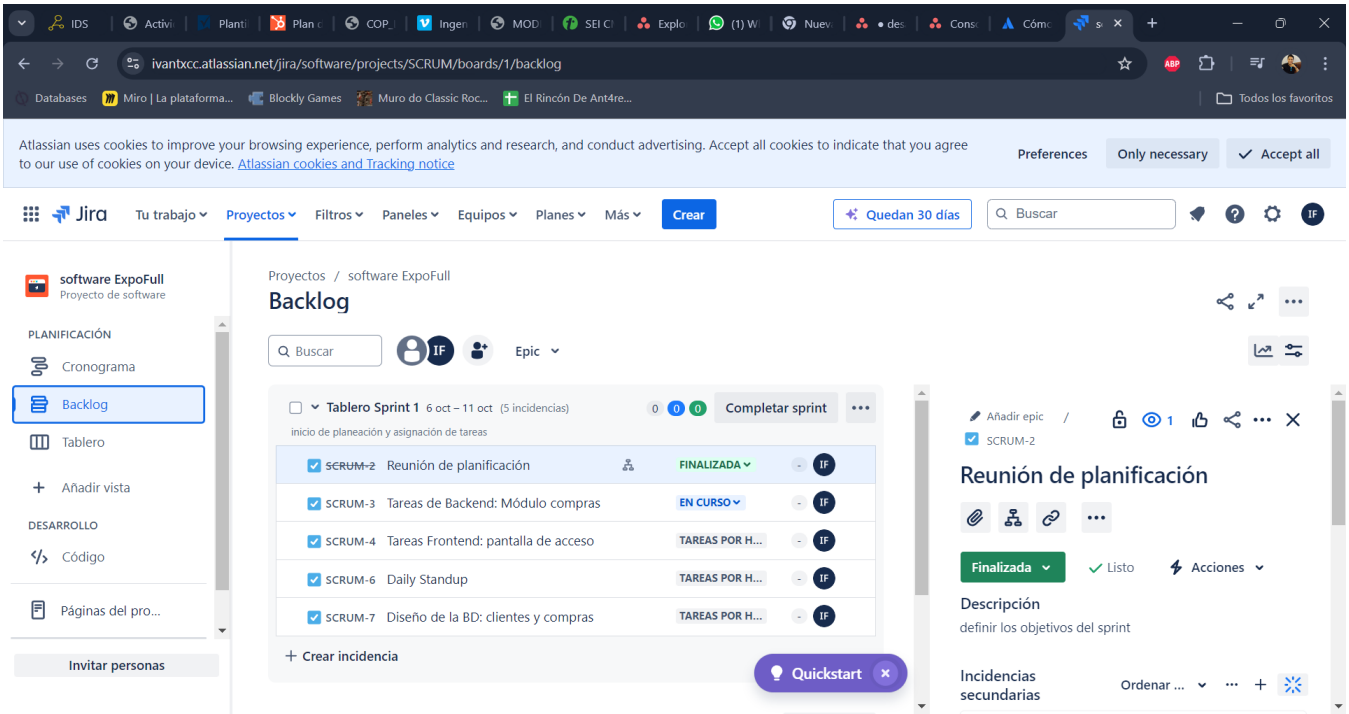
Calendarización del proyecto

Imagen 1



Nota: se comienza con la programación de las actividades, siendo el inicio la propia planificación.

Imagen 2



Nota: se muestra el proceso de las tareas finalizadas y en curso.

Control de versiones de software.

El control de versiones es fundamental en el desarrollo de software, ya que permite gestionar los cambios en el código de manera organizada. Existen varias herramientas populares para este propósito, cada una con características específicas que pueden adaptarse a diferentes necesidades:

1. Git

- Popularidad: El más utilizado a nivel mundial.
- Características: Distribuido, permite múltiples flujos de trabajo, fusión de ramas (branching) y manejo de conflictos eficiente.
- Ventajas: Integración con plataformas como GitHub, GitLab y Bitbucket. Soporta grandes proyectos con múltiples colaboradores.
- Uso: Ideal para equipos grandes y pequeños, con gran soporte y documentación.
- Interfaz gráfica: Puedes usarlo desde la línea de comandos o interfaces gráficas como Sourcetree, GitKraken o GitHub Desktop.

2. Mercurial

- Popularidad: Menos usado que Git, pero aún relevante en ciertos proyectos (por ejemplo, fue usado por Mozilla).
- Características: Similar a Git en su naturaleza distribuida, pero con una curva de aprendizaje un poco más suave.
- Ventajas: Mejor manejo de rendimiento con grandes repositorios. Ofrece simplicidad y es fácil de configurar.
- Uso: Recomendado para usuarios que buscan un flujo de trabajo distribuido y fácil de usar.

3. Subversion (SVN)

- Popularidad: Antes de Git, Subversion era el estándar para control de versiones centralizado.
- Características: Centralizado, con todos los datos almacenados en un servidor único.
- Ventajas: Excelente para proyectos que requieren un control de acceso más estricto y no necesitan la flexibilidad de un sistema distribuido.
- Uso: Aún popular en empresas grandes y proyectos con requerimientos estrictos de seguridad o

auditoría.

4. Perforce (Helix Core)

- Popularidad: Usado en grandes empresas, especialmente en videojuegos y sectores industriales.
- Características: Sistema de control de versiones centralizado, que maneja eficientemente archivos grandes y numerosos usuarios.
- Ventajas: Excelente para grandes organizaciones y proyectos con necesidades complejas de rendimiento.
- Uso: Adecuado para entornos corporativos donde la velocidad y la colaboración entre grandes equipos son críticas.

5. Bazaar

- Popularidad: Menos popular hoy en día, pero solía ser relevante.
- Características: Similar a Git y Mercurial, es un sistema distribuido que ofrece flexibilidad.
- Ventajas: Integración con Launchpad (la plataforma de desarrollo de Ubuntu), fácil de usar y aprender.
- Uso: Adecuado para proyectos más pequeños o usuarios que prefieren la simplicidad.

Git es la mejor opción para control de versiones porque es un sistema distribuido que permite a cada colaborador tener una copia completa del repositorio, lo que aumenta la flexibilidad y la eficiencia en proyectos colaborativos. A diferencia de los sistemas centralizados, Git permite a los usuarios trabajar de manera local sin depender de una conexión constante con un servidor, lo que facilita el trabajo en entornos con conectividad limitada. Además, su capacidad para manejar ramas (branches) y fusiones (merges) de manera sencilla y eficaz permite a los equipos desarrollar múltiples características en paralelo sin interrumpir el flujo principal del proyecto.

Otro factor clave es la extensa comunidad de soporte y el ecosistema de herramientas integradas con Git, como **GitHub**, **GitLab** y **Bitbucket**, que facilitan la colaboración y la integración continua (CI/CD). Git se ha convertido en el estándar en la industria del software gracias a su velocidad, eficiencia con proyectos grandes, y su robustez para manejar complejas estructuras de desarrollo. Además, la curva de aprendizaje es asequible con una amplia gama de documentación y tutoriales disponibles, lo que lo hace accesible tanto para principiantes como para desarrolladores experimentados.

Conclusión

La planificación de proyectos de software es una fase crucial dentro de la ingeniería de software, ya que establece las bases para el éxito del proyecto. Involucra la definición clara de objetivos, alcance, requisitos y la estimación de tiempos y recursos. Una planificación efectiva ayuda a anticipar problemas potenciales, asignar prioridades y asegurar que se sigan los estándares de calidad a lo largo del proceso de desarrollo. La calendarización adecuada es fundamental para garantizar que cada fase del proyecto, desde el diseño hasta la implementación y pruebas, se complete dentro de los plazos acordados. Esto no solo mejora la eficiencia, sino que también mantiene al equipo alineado con los hitos clave, lo que minimiza los riesgos de retrasos y sobrecostos.

La asignación de roles también es esencial para el éxito de cualquier proyecto de software. Al definir claramente las responsabilidades de cada miembro del equipo, se optimiza el flujo de trabajo y se evita la duplicación de esfuerzos o la confusión sobre las tareas. Los roles deben asignarse según las habilidades y fortalezas individuales, lo que permite una mayor especialización y colaboración efectiva. Finalmente, el control de versiones juega un papel vital en la gestión de cambios en el código a lo largo del proyecto. Herramientas como Git permiten llevar un registro detallado de las modificaciones, facilitando la colaboración entre varios desarrolladores y asegurando que los errores se puedan revertir o corregir sin afectar el trabajo de todo el equipo. Juntas, estas prácticas forman una estructura sólida para gestionar proyectos de software complejos de manera eficiente y con éxito.

Referencias

Domina las 7 etapas de desarrollo de software y planea eficazmente. (s. f.). https://www.isol.mx/mentoring/7-etapas-de-desarrollo-de-software

¿Qué es el SDLC? - Explicación del ciclo de vida del desarrollo de software - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/sdlc/

Andres. (2022, 29 mayo). Planificación de proyecto de software - Andres - Medium. Medium. https://medium.com/@andrescarruyo9/planificaci%C3%B3n-de-proyecto-de-software-529a520ac9a5

Talbert, M. (2024, 3 febrero). Cómo crear y gestionar el calendario de un proyecto [2024] • Asana. Asana. https://asana.com/es/resources/project-calendar