ESCUELA POLITÉCNICA NACIONAL

CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE

PROYECTO

Integrantes: Julián Andrés Camacho Zúñiga, Dilan Mateo Calvache Heredia

Curso: GR2SW

Fecha: 14 de diciembre de 2025

Índice:

ESCUELA POLITÉCNICA NACIONAL

CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE

1. Definición del proyecto

1.1 Nombre del proyecto

EconoMe

1.2 Descripción del proyecto

EconoMe es una plataforma web orientada a la gestión de finanzas personales, cuyo objetivo principal es ayudar a los usuarios a llevar un mejor control de sus ingresos, gastos y ahorro de manera clara, organizada y accesible. La aplicación está pensada para personas que desean mejorar su educación financiera y tomar decisiones económicas más informadas en su vida diaria.

La plataforma se estructura en módulos funcionales, de esta manera, EconoMe permite al usuario tener una visión general de su situación económica y detectar oportunidades de mejora en sus hábitos de consumo. Al tratarse de un proyecto en crecimiento, EconoMe está diseñado para ser escalable y mantenible, permitiendo la incorporación futura de nuevos módulos.

2. Plan maestro de SCM

El Plan Maestro de Gestión de la Configuración del Software (SCM) para el proyecto EconoMe tiene como objetivo principal mantener el proyecto bajo control durante todo su ciclo de vida. Para ello, se definirán de manera clara las reglas, procesos y buenas prácticas que regulan cómo se crea, modifica, versiona, prueba y libera el software.

A través de este plan se busca asegurar que todos los cambios realizados en el sistema estén debidamente controlados, documentados y trazables, evitando pérdidas de información, errores por versiones incorrectas o conflictos entre desarrolladores. Además, el SCM permitirá garantizar la calidad del software, facilitar el trabajo colaborativo y preparar el proyecto para su crecimiento futuro, manteniendo siempre un entorno de desarrollo ordenado y predecible.

2.1 Fase de planificación y diseño

2.1.1 Configuración del Entorno y Repositorio

Para el proyecto EconoMe, la configuración del entorno y del repositorio es un aspecto clave dentro del Plan Maestro de SCM, ya que permitirá establecer un control adecuado

desde el inicio del desarrollo y facilitar el trabajo colaborativo entre los integrantes del equipo.

2.1.1.1 Herramientas

Se utilizará Git como sistema de control de versiones, debido a su amplio uso en la industria y a su capacidad para gestionar cambios de manera distribuida. Como plataforma de alojamiento del repositorio se empleará GitHub, ya que ofrece herramientas integradas para la gestión de repositorios, control de accesos, Pull Requests, Issues y automatización de procesos mediante Integración Continua (CI).

2.1.1.2 Estructura del repositorio

El repositorio del proyecto llevará el nombre EconoMe y se organizará bajo un enfoque de monorepositorio, en el cual se almacenará todo el código fuente de la plataforma web, junto con la documentación y otros artefactos relevantes del proyecto. Esta decisión permite centralizar la información, facilitar el control de versiones y simplificar la gestión del proyecto en sus primeras etapas, considerando que se trata de una aplicación web con módulos estrechamente relacionados.

2.1.1.3 Política de ramas (Branching Strategy)

La rama principal del repositorio se denominará main y representará siempre una versión estable del proyecto. Esta rama estará protegida, por lo que no se permitirá realizar push directos sobre ella. Todos los cambios deberán integrarse mediante Pull Requests, los cuales serán revisados antes de su fusión. Esta política garantiza un mayor control sobre el código, reduce el riesgo de errores en la versión principal y asegura que únicamente cambios validados formen parte del estado oficial del proyecto.

2.1.2 Definición del Flujo de Trabajo (Workflow)

El flujo de trabajo define la forma en que los desarrolladores interactúan con el repositorio y gestionan los cambios en el proyecto EconoMe. Establecer un workflow claro permite mantener el orden, evitar conflictos entre versiones y asegurar que cada modificación siga un proceso controlado antes de integrarse al sistema principal.

2.1.2.1 Convención de ramas de trabajo

Las ramas de trabajo se crearán a partir de la rama principal main y seguirán una convención de nombres estandarizada para facilitar su identificación. Se utilizarán los siguientes prefijos:

- feature/nombre-feature: para el desarrollo de nuevas funcionalidades o módulos.
- bugfix/descripcion-bug: para la corrección de errores no críticos detectados durante el desarrollo.
- hotfix/descripcion-bug: para la solución de errores críticos que requieran una corrección inmediata.

Esta convención permite identificar rápidamente el propósito de cada rama y mejora la trazabilidad de los cambios realizados en el proyecto.

2.1.2.2 Política de integración de cambios

Todo cambio en el código deberá integrarse exclusivamente mediante un Pull Request (PR) hacia la rama main. No se permitirá la fusión directa de ramas sin revisión previa. Cada Pull Request deberá describir claramente el cambio realizado y estará sujeto a revisión antes de su aprobación.

Esta política garantiza que los cambios sean evaluados de forma controlada, promoviendo la calidad del código, el trabajo colaborativo y la estabilidad del proyecto EconoMe a lo largo de su desarrollo.

2.1.3 Gestión de Artefactos (No-Código)

La gestión de artefactos no relacionados directamente con el código fuente es una parte fundamental del SCM, ya que permite mantener un control adecuado sobre toda la información que apoya el desarrollo del proyecto EconoMe, como requisitos, documentación y diseños. Versionar estos elementos junto con el proyecto garantiza coherencia, trazabilidad y fácil acceso para todo el equipo.

2.1.3.1 Versionamiento de artefactos no-código

Además del código fuente, todos los artefactos relevantes del proyecto serán gestionados y versionados utilizando el mismo repositorio de GitHub. Esto incluye documentación, especificaciones y enlaces a recursos externos. De esta forma, cualquier cambio en los

artefactos quedará registrado, permitiendo conocer su evolución a lo largo del tiempo y asegurando que siempre estén alineados con el estado actual del proyecto.

2.1.3.2 Requisitos

Los requisitos funcionales y no funcionales del sistema se documentarán en archivos Markdown ubicados dentro de una carpeta /docs en el repositorio. Esta decisión facilita el versionamiento de los requisitos, su revisión mediante Pull Requests y su actualización conforme el proyecto evoluciona. Adicionalmente, se podrá utilizar GitHub Issues para registrar historias de usuario, mejoras o errores, manteniendo la trazabilidad entre requisitos y cambios en el código.

2.1.3.3 Diseños (Mockups/UI)

Los diseños de la interfaz de usuario y los mockups del proyecto se desarrollarán en Figma. El enlace al proyecto de Figma se documentará de forma permanente en el archivo README.md del repositorio, asegurando que todos los integrantes del equipo tengan acceso a la versión más actualizada de los diseños. En caso de ser necesario, capturas o exportaciones relevantes de los diseños podrán almacenarse dentro de la carpeta /docs como material de apoyo.

2.2 Fase de codificación y pruebas

2.2.1 Gestión de Cambios en Acción

La gestión de cambios durante la fase de desarrollo del proyecto EconoMe es fundamental para asegurar que cada modificación realizada en el sistema sea controlada, revisada y correctamente integrada. Para ello, se establecen reglas claras relacionadas con el uso de Pull Requests y la trazabilidad de los cambios.

2.2.1.1 Proceso de Pull Request (PR) y Definición de Hecho

Todo cambio realizado en el código deberá ser integrado mediante un Pull Request (PR) hacia la rama main. Para que un PR sea aprobado y fusionado, deberá cumplir con la siguiente Definición de Hecho (Definition of Done):

- El código debe estar completo y cumplir con los requisitos funcionales asociados.

- El Pull Request debe contar con la revisión y aprobación de al menos un compañero (peer review).
- Todas las pruebas automáticas configuradas en el sistema de Integración Continua (CI) deben ejecutarse correctamente sin errores.
- El PR debe incluir una descripción clara del cambio realizado.

Estas condiciones garantizan que únicamente código validado y revisado forme parte de la versión principal del proyecto.

2.2.1.2 Trazabilidad

Para asegurar la trazabilidad entre los cambios de código y las tareas del proyecto, cada commit deberá referenciar explícitamente el Issue o tarea asociada. Los mensajes de commit seguirán un formato descriptivo que incluya el identificador correspondiente, por ejemplo:

git commit -m "Agrega funcionalidad de registro de gastos (closes #12)".

De esta manera, se mantiene una relación clara entre los requerimientos, las tareas registradas y las modificaciones realizadas en el código, facilitando el seguimiento y el control del proyecto EconoMe.

2.2.2 Integración Continua

La Integración Continua (CI) en el proyecto EconoMe tiene como finalidad garantizar que los nuevos cambios incorporados al sistema no afecten negativamente al código existente. Para ello, se implementará un proceso automatizado que permita validar cada modificación de manera consistente y controlada.

2.2.2.1 Automatización del entorno con Docker

Se utilizará Docker para estandarizar el entorno de ejecución del proyecto, asegurando que las pruebas y validaciones se realicen siempre bajo las mismas condiciones, independientemente del equipo del desarrollador. Mediante contenedores Docker se definirá el entorno necesario para la aplicación, incluyendo dependencias, versiones de librerías y configuraciones requeridas para la ejecución de pruebas.

2.2.2.2 Pipeline de Integración Continua con Jenkins

El proceso de Integración Continua se gestionará mediante Jenkins, el cual estará configurado para ejecutarse automáticamente ante cada push a las ramas de trabajo (feature/, bugfix/, hotfix/) y ante la creación o actualización de un Pull Request. Jenkins utilizará los contenedores Docker para construir el proyecto y ejecutar las validaciones correspondientes.

Dentro del pipeline de CI se incluirán las siguientes etapas:

- Construcción de la aplicación dentro de un contenedor Docker.
- Ejecución de pruebas unitarias para verificar el correcto funcionamiento de los módulos.
- Ejecución de herramientas de análisis de estilo y calidad de código (linter).

Solo cuando todas estas etapas se completen de manera exitosa, el Pull Request podrá ser aprobado y fusionado a la rama main.

La combinación de Docker y Jenkins permite al proyecto EconoMe contar con un proceso de Integración Continua robusto, reproducible y escalable, reduciendo errores por diferencias de entorno y manteniendo la calidad del software durante todo el desarrollo.

2.2.3 Gestión de Línea Base (Baselining)

La gestión de la línea base en el proyecto EconoMe permite identificar y controlar versiones estables del sistema en momentos clave de su desarrollo. Establecer líneas base facilita el seguimiento del progreso del proyecto y proporciona puntos de referencia confiables a los cuales se puede regresar en caso de errores o cambios no deseados.

2.2.3.1 Definición de hitos y versiones base

Los hitos importantes del proyecto se marcarán mediante el uso de tags en Git, los cuales representarán versiones específicas y estables del sistema. Estos tags se crearán cuando uno o varios módulos del proyecto hayan sido completados, revisados y validados mediante pruebas.

Por ejemplo, cuando un módulo funcional relevante, como el registro y categorización de gastos, esté completamente implementado y listo para ser probado por los usuarios, se creará un tag como v1.0-beta, indicando que se trata de una versión preliminar estable.

2.2.3.2 Control y uso de las líneas base

Cada línea base quedará asociada a una versión específica del código que haya superado el proceso de revisión y validación definido en el flujo de trabajo. Estas versiones servirán como referencia para pruebas, demostraciones y futuras etapas de desarrollo, permitiendo mantener un control claro sobre la evolución del proyecto EconoMe.

La utilización de líneas base contribuye a una gestión más ordenada del proyecto, reduce riesgos ante cambios inesperados y facilita la planificación de nuevas versiones del sistema.

2.3 Fase de despliegue y mantenimiento

2.3.1 Gestión de Releases y Despliegue

La gestión de releases y el despliegue del proyecto EconoMe permiten controlar cómo las versiones del sistema pasan desde el entorno de desarrollo hasta el entorno de producción, asegurando estabilidad y trazabilidad en cada entrega.

2.3.1.1 Proceso de despliegue a producción

El código llega a producción a partir de la rama main, la cual siempre contiene una versión estable del sistema. El proceso de despliegue será semi-automatizado. Una vez que un Pull Request es aprobado y fusionado a main, se genera una versión candidata a release. Posteriormente, el despliegue a producción se realizará de forma manual, permitiendo validar el estado del sistema antes de ponerlo a disposición de los usuarios finales.

Este enfoque reduce el riesgo de errores en producción y brinda mayor control sobre las versiones liberadas.

2.3.1.2 Esquema de versionamiento

Para el proyecto EconoMe se utilizará el Versionamiento Semántico, siguiendo el esquema MAJOR.MINOR.PATCH. Este esquema permite identificar de manera clara el tipo de cambios incluidos en cada versión y facilita la comunicación entre los desarrolladores y los usuarios del sistema. La aplicación de este esquema de versionamiento permitirá al proyecto EconoMe mantener un control claro sobre sus releases y facilitará su mantenimiento y evolución a largo plazo.

2.3.1.2.1 Criterios de incremento de versiones

- PATCH: se incrementará cuando se realicen correcciones de errores o ajustes menores que no afecten la funcionalidad principal del sistema.
- MINOR: se incrementará cuando se agreguen nuevas funcionalidades o módulos que sean compatibles con versiones anteriores.
- MAJOR: se incrementará cuando se introduzcan cambios significativos que rompan la compatibilidad con versiones anteriores del sistema.

2.3.2 Plan de Mantenimiento Proactivo

El Plan de Mantenimiento Proactivo del proyecto EconoMe tiene como objetivo garantizar la estabilidad, seguridad y evolución continua del sistema una vez que ha sido desplegado. Para ello, se establecen estrategias claras para gestionar los cuatro tipos de mantenimiento, permitiendo responder de manera organizada tanto a problemas inmediatos como a necesidades futuras. La aplicación de estos cuatro tipos de mantenimiento permitirá que el proyecto EconoMe se mantenga confiable, seguro y preparado para su evolución a largo plazo.

2.3.2.1 Mantenimiento correctivo

El mantenimiento correctivo se enfocará en la resolución de errores detectados en el sistema, especialmente aquellos considerados críticos. Ante la identificación de un bug urgente, se creará una rama hotfix/descripcion-bug a partir de la rama main, en la cual se realizará la corrección correspondiente. Una vez solucionado el problema y verificado su funcionamiento, la rama se fusionará nuevamente en main y se generará una nueva versión etiquetada como PATCH. Este proceso permite corregir errores sin afectar el desarrollo normal del proyecto.

2.3.2.2 Mantenimiento adaptativo

El mantenimiento adaptativo permitirá que EconoMe se mantenga compatible con los cambios del entorno tecnológico. De forma periódica, cada seis meses, se creará una tarea específica para revisar la compatibilidad de la plataforma con nuevas versiones de navegadores web, frameworks o servicios externos utilizados por el sistema. Asimismo,

se evaluará la necesidad de actualizar APIs de terceros, asegurando que la aplicación continúe funcionando correctamente ante cambios externos.

2.3.2.3 Mantenimiento perfectivo

El mantenimiento perfectivo estará orientado a la mejora continua del sistema y a la reducción de la deuda técnica. Para ello, se destinará tiempo de manera planificada a actividades de refactoring, optimización del código y mejora del rendimiento. Como política del proyecto, el último viernes de cada mes se dedicará a tareas de limpieza de código, optimización de procesos y mejora de la estructura interna del sistema, sin alterar su funcionalidad principal.

2.3.2.4 Mantenimiento preventivo

El mantenimiento preventivo buscará anticiparse a posibles problemas futuros, especialmente relacionados con la seguridad y la estabilidad del sistema. Para este fin, se utilizarán herramientas automatizadas como Dependabot, integradas en GitHub, que permitirán analizar de forma periódica las dependencias del proyecto. Estas herramientas alertarán sobre vulnerabilidades de seguridad y propondrán actualizaciones mediante Pull Requests, reduciendo el riesgo de fallos o ataques en el futuro.

3. Referencias

- Geekflare, Una guía completa para el plan de gestión de la configuración, Geekflare, may. 2024. [En línea]. Disponible: https://geekflare.com/es/configuration-management-plan/
- Secureframe, Cómo crear un plan de gestión de configuración y por qué es importante, Secureframe, sep. 2023. [En línea]. Disponible: https://secureframe.com/es-es/blog/configuration-management-plan
- SemVer.org, Semantic Versioning 2.0.0, 2025. [En línea]. Disponible: https://semver.org/lang/es/