Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Planeación de sistemas de software

Gpo 105

Plan de Calidad

Diego Dávila Hernández A01285584

Fernando Morán Fougerat A01284623

Imanol Armando González Solís A00835759

Ramiro Alejandro Garza Villarreal A01178167

Rogelio Garza Rendón A01571384

Campus Monterrey

3 may 2025

Índice

Descripción del Proyecto

- **Historial de versiones:**

Versión	Fecha	Descripción
1.0	9 de Marzo del 2025	Versión inicial del documento

- Información del proyecto
- **Nombre del Proyecto:** CodeCourses.
- **Tecnologías:** React (Frontend), Go (Backend), PostgreSQL (Base de Datos), Judge0, SonarQube.
- **Arquitectura:** Monolítica.
- **Objetivo:**

Crear una plataforma de retos de programación basada en los problemas de la plataforma CodeForces, donde los usuarios puedan resolver problemas de código con un editor integrado, recibir validaciones automáticas y obtener recompensas.

- **Metas:**
  - Facilitar el aprendizaje de habilidades en programación y la eficiencia de los trabajadores.
  - Proveer retroalimentación inmediata sobre código y calidad de las soluciones.
  - Implementar un sistema de recompensas y progreso basado en XP y currency
- **Alcance del Plan**

Este proyecto está dirigido a empleados de TechMahindra interesados en fortalecer sus habilidades de programación a través de retos interactivos.

- **Resumen ejecutivo**

La plataforma permite a los usuarios acceder a problemas de programación, resolverlos dentro de un editor de código y recibir evaluaciones automáticas con Judge0 y SonarQube. Se busca fomentar el aprendizaje y la mejora continua mediante recompensas y un sistema de progreso.

Plan de Pruebas

- Historias de Usuario

1. **HU-001 - Historia de usuario 1: Login**

   Título: Login de usuario

   Descripción: Como usuario y trabajador de la empresa, quiero poder ingresar a la plataforma

   Criterios de aceptación:

   - El usuario debe ingresar su correo y contraseña de la empresa
   - Se debe mostrar un mensaje de error si el correo o la contraseña son incorrectos o inválidos

2. **HU-002 - Historia de usuario 2: Resolver problemas de programación**

   Título: Problemas de programación

   Descripción: Como usuario, quiero poder acceder a los problemas de programación para resolverlos

   Criterios de aceptación:

   - El usuario debe haber ingresado correctamente
   - El usuario debe entrar a la sección de “Problemas de programación”
   - El usuario debe seleccionar el problema que quiere resolver

3. **HU-003 - Historia de usuario 3: Filtrar problemas de programación**

   Título: Filtros en la lista de problemas

   Descripción: Como usuario, quiero poder filtrar los problemas de programación por dificultad y etiquetas para encontrar los que más se adapten a mi nivel.

   Criterios de aceptación:

   - El usuario debe ver opciones de filtro por dificultad y etiquetas en la sección de problemas.
   - Al seleccionar un filtro, la lista de problemas debe actualizarse automáticamente.
   - Debe existir la opción de quitar los filtros para volver a ver todos los problemas.

4. **HU-004 - Historia de usuario 4: Ver detalles de un problema**

   Título: Visualización del problema

   Descripción: Como usuario, quiero ver la descripción completa de un problema antes de intentar resolverlo.

   Criterios de aceptación:

   - Al hacer clic en un problema, se debe abrir una página con su descripción.
   - La página debe mostrar la dificultad, las etiquetas y el enlace al problema original en CodeForces.
   - Debe haber un botón para regresar a la lista de problemas.

5. **HU-005 - Historia de usuario 5: Escribir y ejecutar código**

   Título: Editor de código integrado

   Descripción: Como usuario, quiero escribir mi solución en un editor de código integrado y ejecutarla para comprobar si funciona correctamente.

   Criterios de aceptación:

   - El usuario debe poder escribir código en un editor dentro de la página del problema.
   - Al hacer clic en "Ejecutar", el código debe enviarse a Judge0 para su evaluación.
   - Los resultados de la ejecución (errores o salida esperada) deben mostrarse en la interfaz.

6. **HU-006 - Historia de usuario 6: Validar código con estándares de calidad**

   Título: Evaluación de calidad del código

   Descripción: Como usuario, quiero que mi código sea analizado en cuanto a estilo y buenas prácticas para mejorar su calidad.

   Criterios de aceptación:

   - Al enviar el código, este debe ser evaluado por SonarQube.
   - Si hay errores de calidad, el usuario debe recibir un mensaje con las recomendaciones.
   - Si el código cumple con los estándares, se debe marcar como “completo”.

7. **HU-007 - Historia de usuario 7: Recibir recompensas por resolver problemas**

   Título: Sistema de recompensas

   Descripción: Como usuario, quiero recibir XP y currency cuando se resuelve correctamente un problema para motivarme a seguir aprendiendo.

   Criterios de aceptación:

   - Si el código es válido y pasa todas las pruebas, se debe actualizar la XP y currency del usuario.
   - El usuario debe recibir una notificación de éxito tras resolver un problema.
   - El progreso del usuario debe actualizarse en su perfil.

8. **HU-008 - Historia de usuario 8: Ver avances**

   Título: Dashboard avances

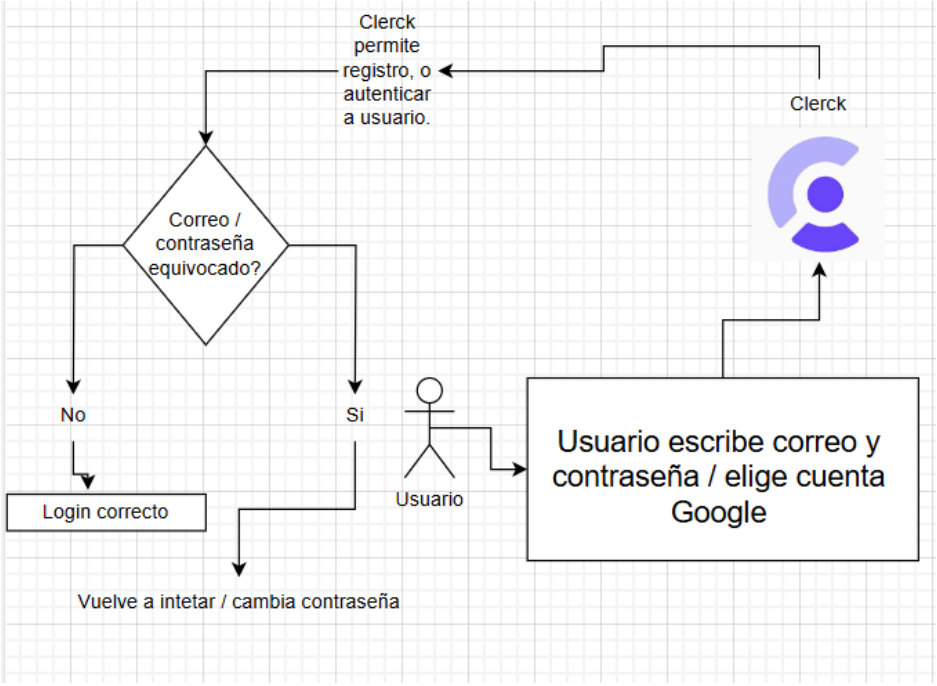
   Descripción: Como usuario quieres ver el avance de personas en tu organización, y tu propio.

   Criterios de aceptación:

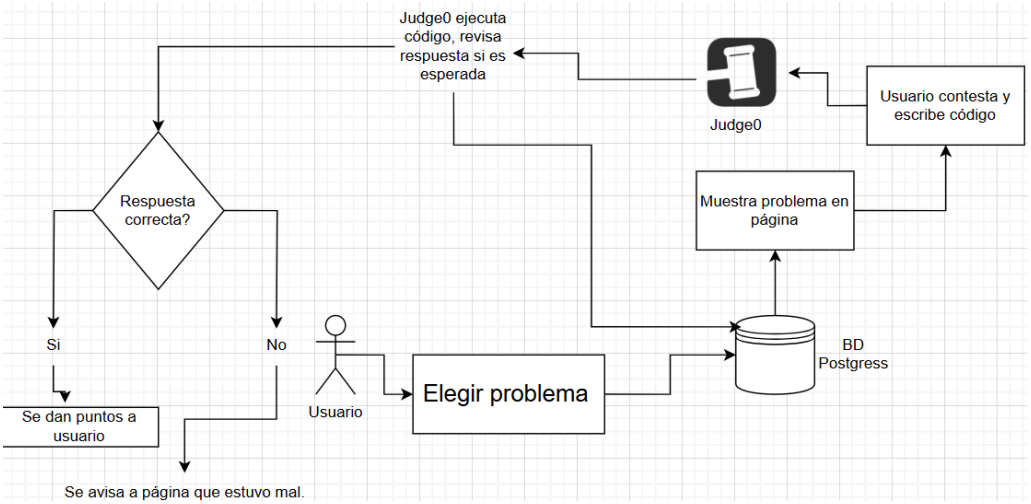
   - Ver tu porcentaje de aceptación, problemas resueltos, y promedio de tu equipo.
   - Poder ver información de usuarios compañeros.

- **Casos de Uso**

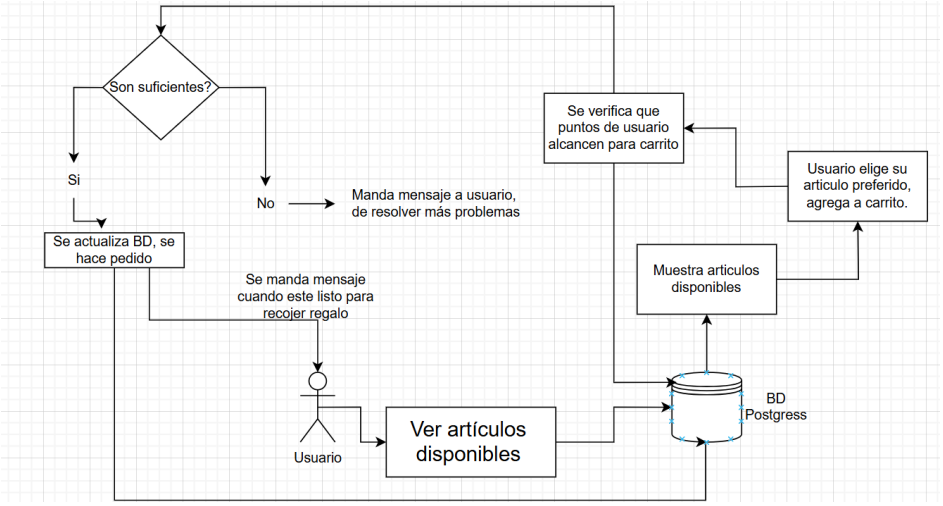
1. Autenticación de usuario



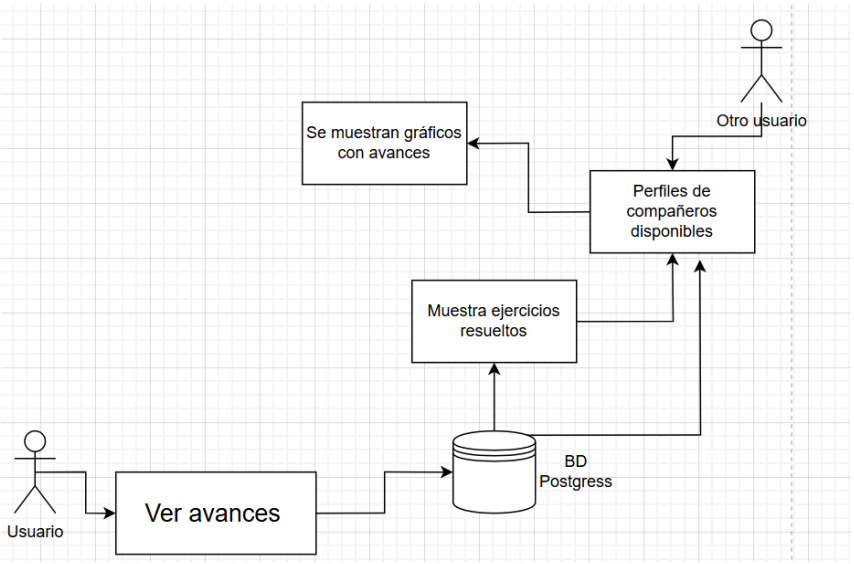
2. Resolver un problema



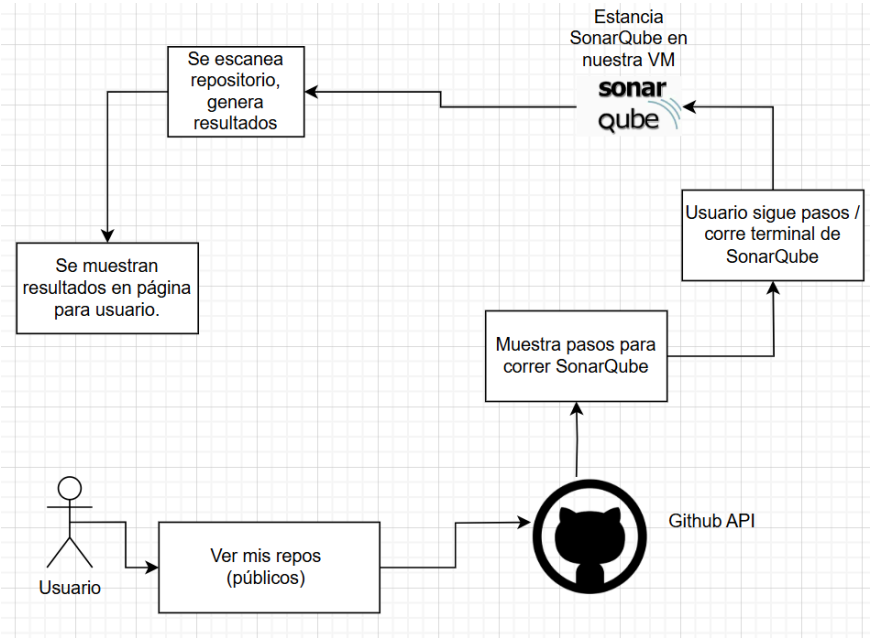
3. Comprar merch



4. Ver avances



5. Ver calidad en repositorios



- Alcance de las Pruebas

El plan de pruebas cubre los siguientes aspectos:

- **Pruebas de desempeño**: Se analizará la rapidez de respuesta del sistema.
- **Pruebas de estrés**: Se evaluará la capacidad de la plataforma bajo condiciones extremas.
- **Pruebas de carga**: Se medirá el rendimiento con múltiples usuarios simultáneos.
- **Pruebas funcionales**: Se validará el correcto funcionamiento de cada componente.
- **Pruebas unitarias**: Se probarán funciones individuales del sistema, sobretodo desde el lado del módulo de datos.
- **Pruebas de aceptación**: Se comprobará que la plataforma cumpla con los requisitos del usuario final.

- Pruebas de desempeño

ID	Nombre	Descripción	Criterios de Éxito
PD-001	Tiempo de respuesta del login	Medir el tiempo que tarda el sistema en autenticar a	La autenticación no debe tardar más de 3

		un usuario.	segundos.
PD-002	Carga del dashboard	Evaluar el tiempo de carga del dashboard.	El dashboard debe cargarse en menos de 4 segundos.
PD-003	Evaluación de código en Judge0	Medir el tiempo de procesamiento del código en Judge0.	La evaluación debe completarse en menos de 15 segundos.

- Pruebas de estrés

ID	Nombre	Descripción	Criterios de Éxito
PE-001	Uso de plataforma cuando hay mucha información a desplegar.	Cargar el dashboard de problemas cuando hay más de 100 en la base de datos.	La plataforma debe mantener tiempos de respuesta aceptables.
PE-002	Envío masivo de código a Judge0	Mandar varias solicitudes de evaluación en paralelo.	Judge0 debe seguir funcionando sin caídas abruptas.

- Pruebas de carga

ID	Nombre	Descripción	Criterios de Éxito
PL-001	Simulación de 50 usuarios registrados	Medir el desempeño de la página cuando hay varios usuarios registrados.	La plataforma debe mantener tiempos de respuesta aceptables.
PL-002	Peticiones simultáneas a la base de datos	Enviar consultas masivas a PostgreSQL.	La base de datos no debe ralentizarse significativamente.

PL-003	Simulación de 1,000 usuarios	Enviar muchas solicitudes al servidor al mismo tiempo.	La plataforma no debe colapsar al enviar todas las solicitudes.

- Pruebas Funcionales

ID	Nombre	Descripción	Criterios de Éxito
PF-001	Autenticación de usuario	Validar que los usuarios puedan iniciar sesión correctamente.	El usuario debe autenticarse con credenciales válidas y recibir un mensaje de error si los datos son incorrectos.
PF-002	Filtrado de problemas	Comprobar que los filtros de dificultad y etiquetas funcionen correctamente.	La lista de problemas debe actualizarse dinámicamente al aplicar filtros y permitir eliminar filtros sin errores.
PF-003	Evaluación de código	Verificar que el código enviado a Judge0 se procese adecuadamente.	Los resultados deben ser precisos y mostrarse correctamente al usuario.
PF-004	Registro de nuevos usuarios	Validar que el sistema permita la creación de nuevas cuentas.	El usuario debe recibir un correo de confirmación tras registrarse y poder iniciar sesión posteriormente.
PF-005	Envío de código	Comprobar que los usuarios puedan enviar sus soluciones de código correctamente.	La plataforma debe procesar y evaluar el código sin errores.
PF-006	Generación de reportes de actividad	Validar que el sistema genere reportes sobre la actividad del usuario.	Los reportes deben incluir número de problemas resueltos, XP ganado y estadísticas relevantes.

PF-07	Historial de intentos	Evaluar que los usuarios puedan consultar sus envíos anteriores de código.	Los intentos deben mostrarse en orden cronológico con detalles del resultado de cada uno.
PF-08	Respuesta a errores de ejecución	Comprobar que el sistema muestre mensajes adecuados cuando el código presenta errores de sintaxis o lógica.	Los mensajes deben ser claros y ayudar al usuario a corregir el problema.

- Pruebas Unitarias

ID	Nombre	Descripción	Criterios de Éxito
PU-001	Validación de datos de login	Verificar que la función de validación de correo y contraseña funcione correctamente.	La función debe rechazar formatos de correo inválidos y contraseñas que no cumplan con los requisitos de seguridad.
PU-002	Verificación de filtros	Comprobar que las funciones de filtrado de problemas devuelvan los resultados esperados.	Los filtros deben retornar exactamente los problemas que cumplen con los criterios seleccionados.
PU-003	Cálculo de recompensas	Validar que el algoritmo de cálculo de XP y currency funcione según los parámetros establecidos.	Las recompensas deben calcularse correctamente según la dificultad del problema resuelto.
PU-004	Integración con Judge0	Comprobar que la función de envío de código a Judge0	La función debe establecer conexión con Judge0, enviar

		funcione adecuadamente.	el código y recibir respuesta sin errores.
PU-005	Análisis de código con SonarQube	Verificar que la integración con SonarQube evalúe correctamente los estándares de calidad del código.	SonarQube debe detectar problemas de calidad en el código y generar reportes precisos.
PU-006	Actualización de progreso	Validar que las funciones de actualización de estadísticas del usuario modifiquen correctamente la base de datos.	Los datos de progreso del usuario deben actualizarse de manera precisa en la base de datos.
PU-007	Función de ordenamiento de problemas	Comprobar que la función para ordenar problemas por dificultad, fecha o popularidad funcione correctamente.	Los problemas deben ordenarse según el criterio seleccionado sin excepción.
PU-008	Función de búsqueda	Verificar que la búsqueda de problemas por palabra clave funcione adecuadamente.	La función debe retornar todos los problemas que contengan la palabra clave en su título o descripción.

- Pruebas de Aceptación

ID	Nombre	Descripción	Criterios de Éxito
PA-001	Registro con Google	Como usuario nuevo, quiero poder registrarme usando mi cuenta de Google para acceder	El usuario puede registrarse con un solo clic utilizando su cuenta corporativa de Google y acceder

		rápidamente a la plataforma.	inmediatamente a la plataforma.
PA-002	Navegación intuitiva de problemas	Como usuario, quiero navegar fácilmente por la lista de problemas disponibles para encontrar lo que necesito.	Los usuarios pueden filtrar, ordenar y buscar problemas sin dificultad y encuentran la información relevante en menos de 3 clics.
PA-003	Editor de código funcional	Como usuario, quiero un editor de código con resaltado de sintaxis y autocompletado para facilitar la escritura de soluciones.	El editor debe soportar múltiples lenguajes, tener resaltado de sintaxis y ofrecer sugerencias de autocompletado.
PA-004	Retroalimentación clara sobre soluciones	Como usuario, quiero recibir feedback detallado sobre mi código para entender por qué funciona o falla.	El sistema debe mostrar mensajes claros sobre errores de sintaxis, fallos en casos de prueba y sugerencias de mejora.
PA-005	Visualización de progreso	Como usuario, quiero ver mi progreso y compararlo con otros miembros del equipo para mantener la motivación.	El dashboard debe mostrar claramente estadísticas personales y comparativas con otros usuarios.
PA-006	Sistema de recompensas atractivo	Como usuario, quiero recibir recompensas por resolver problemas para mantener mi interés en la	Los usuarios deben recibir notificaciones visibles sobre las recompensas obtenidas y ver su acumulación en el

		plataforma.	perfil.
PA-007	Experiencia de usuario optimizada en móviles	Como usuario, quiero acceder a la plataforma desde mi dispositivo móvil para practicar en cualquier momento.	La interfaz debe ser responsiva y mantener todas sus funcionalidades en dispositivos móviles.
PA-008	Tiempo de respuesta satisfactorio	Como usuario, espero que la plataforma responda rápidamente a mis interacciones para no perder tiempo esperando.	Todas las operaciones básicas deben completarse en menos de 3 segundos y las evaluaciones de código en menos de 15 segundos.

Estándares aplicados

Para este proyecto, estaremos siguiendo diferentes estándares para poder garantizar la calidad del software, usaremos los siguientes estándares:

- **ISO/IEC 25010** - Modelo de calidad del software que define características como funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.
- **ISO/IEC 12207** - Estándar que establece procesos para el desarrollo y mantenimiento de software.
- **ISO/IEC 29119:** Establece procesos para el diseño, ejecución y documentación de pruebas.
- **ISO/IEC 9001:** Certifica que los procesos cumplen estándares de calidad internacionales.

También estaremos incorporando niveles de modelos de gestión como por ejemplo:

- **CMMI - Nivel 3**: Los procesos estarán definidos y documentados durante todas las etapas del desarrollo, mejoras continuas mediante métricas de calidad y desempeño y usaremos práctica de gestión de riesgos y medición de calidad en el ciclo de vida del software.

Equipo de Calidad

- Roles de cada miembro del equipo:
  - Diego Dávila Hernández: Validación de pruebas de desempeño.
  - Fernando Morán Fougerat: Validación de pruebas de estrés.
  - Imanol Armando González Solís: Validación de pruebas de carga.
  - Rogelio Garza Rendón: Validación de pruebas funcionales
  - Ramiro Alejandro Garza Villarreal: Validación de pruebas de aceptación y unitarias.

Procesos para auditar

- Riesgos

	1 Insignificante	2 Menor	3 Moderada	4 Importante	5 Catastrófica
5 **Muy Probable**	5 El LLM que utilizamos se equivoca al modificar un problema	10	15 Alguna dependencia no es compatible con las demás tecnologías que utilizamos	20	25
4 **Probable**	4	8 Inventario de la tienda inexacto	12 Los usuarios no se sienten cómodos con la dificultad de los problemas	16 Sobrecarga de la plataforma por falta de recursos para mejorar el servidor	20 Algún servicio externo se cae durante la presentación al socio
3 **Posible**	3 Error de presición en alguna de las tags de un problema	6	9 Cantidad de dinero digital inexacto	12 Se presenta algún bug en la tienda que hace que impide a los usuarios redimir sus recompenzas	15 Corrupción de la base de datos debido a un error en el sistema de respaldo, afectando a todos los usuarios.
2 **No es probable**	2	4	6 Los problemas de programación no se refrescan correctamente, mostrando desafíos obsoletos o eliminados	8 Se elimina accidentalmente información en la base de datos	10 Cierre forzado de la plataforma por problemas legales o de derechos de autor.
1 **Muy improbable**	1	2	3	4 Desincronización con Judge0, causando que el código enviado no se evalúe correctamente o tome demasiado tiempo.	5 Ataque cibernético que tumbe el proyecto por completo

Los anteriores son los principales riesgos que encontramos que son relevantes para la plataforma, el proceso para auditar los riesgos consistirá en monitorear cada uno de estos sobretodo aquellos que tienen una relación probabilidad-impacto superior,

de esa manera enfocamos los esfuerzos del equipo en aquello que representa un mayor riesgo para el proyecto.

- Software

A continuación, mostraremos los pasos que recomendamos para verificar y prevenir problemas en nuestro Software.

1. **Gestión de Requisitos y Documentación**
    a. Evaluar nuestros requisitos funcionales y no funcionales en equipo.
    b. Hablar con el socio formador, para ver su retroalimentación.
    c. Revisar que las especificaciones de usuario estén alineadas con el software entregado.
2. **Pruebas y Calidad del Código**
    a. Revisar métricas de calidad del código con la herramienta SonarQube.
    b. Hacer pruebas con usuarios finales (programadores).
3. **Seguridad del Software**
    a. Validar que no haya contraseñas hardcodeadas en el código

De igual manera, se tendrá un proceso agendado (aproximadamente cada mes) para revisar y actualizar nuestras versiones de dependencias, para asegurarnos de tener versiones actualizadas para no tener vulnerabilidades conocidas.

Referencias

1. CodeForces. (2024). *CodeForces Problem Set*. https://codeforces.com/problemset
2. Go. (2024). *The Go Programming Language*. https://golang.org/
3. International Organization for Standardization. (2011). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) (ISO/IEC 25010:2011). https://www.iso.org/standard/35733.html
4. International Organization for Standardization. (2015). Quality management systems — Requirements (ISO 9001:2015). https://www.iso.org/standard/62085.html
5. International Organization for Standardization. (2017). Systems and software engineering — Software life cycle processes (ISO/IEC 12207:2017). https://www.iso.org/standard/63712.html
6. International Organization for Standardization. (2022). Software and systems engineering — Software testing (ISO/IEC 29119:2022). https://www.iso.org/standard/79439.html

7. Judge0. (2023). *Judge0 API Documentation*. https://judge0.com/
8. PostgreSQL. (2024). *PostgreSQL Documentation*. https://www.postgresql.org/docs/
9. React. (2024). *React Documentation*. https://react.dev/
10. Software Engineering Institute. (2018). CMMI for Development, Version 2.0. Carnegie Mellon University. https://cmmiinstitute.com/cmmi
11. SonarSource. (2024). *SonarQube Documentation*. https://docs.sonarqube.org/