Plan de gestión de riesgos

**1. Introducción**

Este documento describe el **Plan de Gestión de Riesgos** para el proyecto, siguiendo un enfoque sistemático para la identificación, evaluación y mitigación de riesgos que puedan impactar el desarrollo del software.

El propósito de este plan es asegurar que se tomen **medidas preventivas y correctivas** adecuadas para minimizar los efectos negativos de posibles riesgos.

**2. Proceso de Gestión de Riesgos**

**2.1 Identificación de Riesgos**

Los riesgos se identifican mediante:

* **Revisiones periódicas** en reuniones del equipo.
* **Reportes ad-hoc** si se detecta un riesgo durante el desarrollo del proyecto.
* **Evaluaciones de expertos** en calidad de software y pruebas.

Cada riesgo identificado se documentará en el **Registro de Riesgos**.

**2.2 Evaluación y Clasificación de Riesgos**

Cada riesgo se evaluará en función de dos factores:

1. **Impacto** (en el cronograma o en los resultados del proyecto).
2. **Probabilidad** (posibilidad de ocurrencia del riesgo).

Se asignará un nivel de riesgo multiplicando **Impacto x Probabilidad**, lo que permitirá priorizar acciones correctivas.

**2.3 Plan de Mitigación y Respuesta**

Para cada riesgo identificado, se establecerán estrategias de mitigación, que pueden incluir:

* **Evitación del riesgo:** Cambios en el diseño o proceso para eliminar la amenaza.
* **Reducción del impacto:** Implementación de medidas que minimicen los efectos adversos.
* **Plan de contingencia:** Estrategias alternativas en caso de que el riesgo ocurra.

**2.4 Monitoreo y Seguimiento**

El estado de los riesgos se actualizará periódicamente y se revisará en reuniones de seguimiento del proyecto.

**3. Formato de Análisis de Riesgos (Registro de Riesgos)**

| **ID** | **Descripción del riesgo** | **Impacto (1-5)** | **Probabilidad (1-5)** | **Nivel de riesgo (Impacto \* Probabilidad)** | **Estrategia de mitigación** | **Plan de contingencia** | **Estado** | **Responsable** | **Observaciones Finales** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R-001 | Incompatibilidad con los archivos heredados para la importación de información, | 5 | 2 | 10 | Análisis técnico y pruebas de integración con tests previos de los archivos | Implementar soluciones de interpretación o conversores de datos si es necesario | Mitigado | Miranda | Hubieron ciertos problemas al momento de la implementación pero fueron mitigados al instante |
| R-002 | Fallos en la integración con APIs | 2 | 1 | 2 | Incluir pruebas unitarias en el proceso de llamadas de la APIs | Desarrollar la API maneje distintos formatos de respuesta | Mitigado | Alejandro | Se implementaron las apis con precaución y haciendo pruebas unitarias al momento por lo cual fue mitigado |
| R-003 | Baja velocidad en consultas a la base de datos | 3 | 2 | 6 | Optimización de queries individualizadas | Implementar debugging en la consola del navegador | Mitigado | Alejandro | Utilizamos la IA para optimizar las queries antes de implementarlas por lo cual logramos mitigarlo |
| R-004 | Vulnerabilidades en la autenticación y autorización de usuarios | 5 | 3 | 15 | Implementación de un servicio de autenticación como el de supabase auth | Uso de doble factor de autenticación. | Mitigado | Alejandro | Se dio uso con supabase para la autenticación y se realizaron pruebas para evitar todo tipo de vulnerabilidades comunes |
| R-005 | Fallo en el versionado del código que cause pérdida de funcionalidades | 4 | 1 | 4 | Uso de Git con ramas bien definidas (main, develop, feature) y revisión de código con pull requests | Restauración desde backups automáticos y rebase controlado en Git | Mitigado | Miranda | Se usaron PR lo cual ayudó a prevenir los errores y alguna vez sucedió un error pero se uso la restauración |
| R-006 | Caídas del servidor debido a alta demanda inesperada | 5 | 2 | 10 | Escalado automático en AWS | Implementación de balanceo de carga y servidores de respaldo | En proceso | Estefania | Seguimos al pendiente pero al momento no ha sucedido nada y se ha vigilado el trafico pero no hay indicadores de que pueda caer. |
| R-007 | Errores en la configuración de seguridad de la base de datos | 5 | 2 | 10 | Implementación de políticas de seguridad con roles y permisos mínimos necesarios. | Registro de logs y restauración desde backups en caso de errores. | Mitigado | Alejandro | Al inicio hubo un par de errores por la configuración pero se implementaron los cambios progresivamente para evitar errores inesperados |
| R-008 | Errores tipográficos en el código que causan fallos | 1 | 5 | 5 | Uso de linters como ESLint (JavaScript) o Pylint (Python) para detectar errores antes de la ejecución. | Pruebas manuales en funcionalidades clave antes del despliegue. | Mitigado | Sofia | Al incio nos aseguramos de instalar extensiones de codigo para checar errores para poder estar al tanto de cualquier typo |
| R-009 | No cumplir con tiempos de entrega estimados | 2 | 3 | 6 | Estimaciones realistas con la técnica de "Planning Poker" y control de tiempos con Jira o Trello. | Priorización de tareas críticas y ajuste del alcance si es necesario. | Mitigado | Estefania | Nos organizamos por sprints y a pesar de que hubo un retraso asignamos prioridades y tareas para dentro de un plazo por acuerdo a todos los integrantes por lo cual terminamos antes de lo esperado. |
| R-010 | Falta de pruebas en diseño responsivo | 3 | 3 | 9 | Uso de la practica de diseño MDF (Mobile Design First) | Desarrollar dando uso a las funciones de diseño responsivo | Mitigado | Sofia | Se implementó un diseño MDF lo cual nos ayudó a tener el diseño responsivo y al acabar el frontend se realizaron pruebas al UI/UX por lo cual nos aseguramos de que todo fuera responsivo. |

**4. Criterios de Evaluación de Riesgos**

**4.1 Riesgos Mitigados Exitosamente**

* **R-001 - Incompatibilidad con archivos heredados**: Se implementó exitosamente un análisis técnico previo y pruebas de integración exhaustivas. Aunque se presentaron algunos problemas iniciales durante la implementación, el equipo logró resolverlos de manera inmediata aplicando las estrategias de mitigación planificadas, evitando retrasos significativos en el proyecto.
* **R-002 - Fallos en la integración con APIs**: El riesgo fue completamente mitigado mediante la implementación de pruebas unitarias rigurosas durante el proceso de desarrollo. El equipo adoptó un enfoque preventivo, realizando pruebas continuas al momento de integrar cada API, lo que permitió detectar y resolver problemas antes de que afectaran el sistema.
* **R-003 - Baja velocidad en consultas a la base de datos**: Se logró una mitigación exitosa utilizando inteligencia artificial para optimizar las queries antes de su implementación. Esta estrategia innovadora permitió al equipo anticipar y resolver problemas de rendimiento de manera proactiva, superando las expectativas iniciales de la estrategia de mitigación planificada.
* **R-004 - Vulnerabilidades en autenticación y autorización**: Se implementó exitosamente Supabase Auth como servicio de autenticación, cumpliendo con la estrategia de mitigación establecida. El equipo realizó pruebas exhaustivas de seguridad para prevenir vulnerabilidades comunes, garantizando un sistema robusto y seguro para los usuarios.
* **R-005 - Fallo en el versionado del código**: El uso de Git con ramas bien definidas y pull requests demostró ser altamente efectivo. Aunque se presentó un incidente menor durante el desarrollo, el sistema de respaldo y restauración funcionó correctamente, validando la efectividad tanto de la estrategia de mitigación como del plan de contingencia.
* **R-007 - Errores en la configuración de seguridad de la base de datos**: Inicialmente se presentaron algunos errores de configuración, pero el equipo implementó los cambios de seguridad de manera progresiva y controlada. Las políticas de seguridad con roles y permisos mínimos se establecieron exitosamente, eliminando las vulnerabilidades identificadas.
* **R-008 - Errores tipográficos en el código**: La implementación temprana de extensiones de código y linters resultó altamente efectiva. El equipo se aseguró de instalar las herramientas necesarias desde el inicio del proyecto, lo que permitió detectar y prevenir errores tipográficos de manera automática.
* **R-009 - No cumplir con tiempos de entrega**: A pesar de experimentar un retraso inicial, el equipo logró reorganizarse efectivamente utilizando metodología de sprints. La priorización de tareas críticas y el ajuste colaborativo del cronograma no solo permitieron recuperar el tiempo perdido, sino que el proyecto se completó antes de lo esperado.
* **R-010 - Falta de pruebas en diseño responsivo**: La implementación del enfoque Mobile Design First (MDF) fue exitosa. El equipo realizó pruebas exhaustivas de UI/UX al finalizar el frontend, asegurando que todas las funcionalidades fueran completamente responsivas en diferentes dispositivos.

### **4. 2 Riesgos Vigentes al Cierre del Proyecto**

* **R-006 - Caídas del servidor debido a alta demanda**: Este riesgo permanece en estado de monitoreo activo al cierre del proyecto. Aunque no se han presentado incidencias y el tráfico se mantiene dentro de parámetros normales, el equipo continúa vigilando los indicadores de rendimiento. Se recomienda mantener el monitoreo continuo y tener preparadas las estrategias de escalado automático en AWS para implementación futura si la demanda aumenta.

### **4.3 Lecciones Aprendidas**

* **Efectividad del plan inicial**: El plan de gestión de riesgos demostró ser altamente efectivo, con un 90% de los riesgos identificados siendo completamente mitigados. Las estrategias preventivas resultaron más exitosas que las reactivas.
* **Herramientas y técnicas más útiles**:
  + La integración de IA para optimización de queries fue una innovación exitosa no contemplada inicialmente
  + Las pruebas unitarias continuas resultaron fundamentales para la calidad del código
  + El enfoque Mobile Design First superó las expectativas para el diseño responsivo
  + La metodología de sprints y priorización colaborativa fue clave para la gestión del tiempo
* **Mejoras para futuros proyectos**:
  + Implementar monitoreo automatizado de servidor desde las fases tempranas del desarrollo
  + Considerar el uso de IA como herramienta estándar para optimización de código
  + Establecer puntos de control más frecuentes para riesgos de alta criticidad
  + Documentar mejor los procesos de mitigación exitosos para reutilización

### **5. Resumen Ejecutivo**

Al cierre del proyecto, de los 10 riesgos identificados inicialmente:

* **9 riesgos** fueron completamente mitigados
* **0 riesgos** fueron parcialmente mitigados
* **1 riesgo** permanece vigente bajo monitoreo
* **0 riesgos nuevos** se identificaron durante el desarrollo

El nivel de riesgo general del proyecto se considera **BAJO** al momento del cierre, con un impacto total mínimo en el cronograma y funcionalidades. La gestión proactiva de riesgos contribuyó significativamente al éxito del proyecto, permitiendo incluso completar la entrega antes del tiempo estimado original.