

# Fundamentos de la Calidad del Software



## Equipo de Trabajo

Penélope Noreña Ramos  
Hernán Darío Pérez Higuita  
William Pérez Muñoz

## Servicio Nacional de Aprendizaje - SENA

**Centro de Gestión de Mercados, Logística y Tecnologías de la  
Información**

**Regional Distrito Capital**

**Tecnología en Análisis y Desarrollo de Software**

**Ficha: 2758315**

**2024**

# Tabla de contenido

1. [Introducción](#)
2. [Objetivos](#)
  - [2.1 Generales](#)
  - [2.2 Específicos](#)
3. [Fases del Desarrollo de Software](#)
  - [3.1 Visualización de las Fases](#)
4. [Estado de Requerimientos](#)
  - [4.1 Requerimientos Funcionales y No Funcionales](#)
5. [Casos de Prueba](#)
  - [5.1 Distribución de Casos de Prueba por Funcionalidad](#)
  - [5.2 Estado de Ejecución de Casos de Prueba](#)
  - [5.3 Ciclos de Prueba](#)
6. [Métricas y Seguimiento](#)
  - [6.1 Métricas Clave](#)
7. [Documentación de Defectos y Mejoras](#)
8. [Lecciones Aprendidas](#)
  - [8.1 Gestión del Proyecto](#)
  - [8.2 Aspectos Técnicos](#)
  - [8.3 Mejores Prácticas Identificadas](#)
9. [Gestión de Riesgos](#)
  - [9.1 Riesgos Materializados](#)
10. [Herramientas Utilizadas](#)
11. [Recomendaciones](#)
12. [Conclusiones del Aprendizaje](#)
13. [Glosario de Términos](#)
14. [Referencias](#)
15. [Historial de Cambios](#)
16. [Próximos Pasos](#)
17. [Anexos](#)

# 1. Introducción

El proyecto **Casino la Fortuna** es una solución de software contable diseñada para establecimientos pequeños y medianos. Este sistema busca optimizar los procesos financieros y administrativos, garantizando una experiencia tecnológica moderna, accesible y confiable.

La implementación de prácticas de calidad y la documentación detallada del desarrollo aseguran un producto escalable, funcional y alineado con las necesidades del cliente.

## 2. Objetivos

### 2.1 Generales

- Automatizar procesos financieros para mejorar la eficiencia operativa.
- Desarrollar un sistema integral que cumpla con normativas locales y estándares de calidad.
- Garantizar la estabilidad y usabilidad mediante pruebas continuas.

### 2.2 Específicos

1. Diseñar una arquitectura modular que permita futuras expansiones.
2. Implementar funcionalidades clave como gestión de clientes, transacciones y reportes.
3. Ejecutar pruebas exhaustivas para detectar y corregir errores.
4. Documentar cada fase del desarrollo para facilitar el mantenimiento.

## 3. Fases del Desarrollo de Software

El desarrollo del software Casino la Fortuna sigue un enfoque iterativo influenciado por metodologías como RAD (Rapid Application Development), que permite entregar prototipos funcionales tempranos, recibir retroalimentación, y realizar ajustes rápidos. Este proceso se organiza en las siguientes etapas clave:

1. Análisis de Requisitos: Identificación y priorización de necesidades del cliente.

2. Diseño del Sistema: Creación de prototipos interactivos y definición de la arquitectura del sistema.
3. Desarrollo: Implementación de módulos funcionales, frontend y backend, con operaciones CRUD (Create, Read, Update, Delete) para la gestión eficiente de datos.
4. Pruebas: Identificación y corrección de errores para garantizar estabilidad y calidad.
5. Despliegue: Publicación del sistema en entornos reales y validación final.
6. Mantenimiento: Monitoreo, actualización y mejora continua del software.

## 3.1 Visualización de las Fases

Para una mejor comprensión, se presenta una infografía que resume estas etapas:

# Fases del Desarrollo de Software

El desarrollo de software sigue un conjunto de pasos metodológicos que aseguran la creación de soluciones tecnológicas efectivas, escalables y confiables. A continuación, las etapas clave que guían este proceso.

1

### Análisis de Requisitos

#### Objetivo

Entender qué necesita el cliente.

#### Actividades

- Entrevistas con usuarios clave
- Definición y priorización de requisitos
- Creación de historias de usuario

## Resultado

Documento detallado de requisitos

2

## Diseño del Sistema

### Objetivo

Planificar la arquitectura y diseño del software.

### Actividades

- Diseño de bases de datos
- Creación de prototipos y mockups
- Elaboración de diagramas UML

### Resultado

Prototipos y arquitectura definida

3

## Desarrollo

### Objetivo

Construir el software mediante la programación.

### Actividades

- Implementación de frontend y backend
- Integración de servicios y APIs
- Gestión de control de versiones

## Resultado

Código funcional y documentado



## Pruebas

### Objetivo

Detectar y corregir errores antes del despliegue.

### Actividades

- Pruebas unitarias y funcionales
- Simulación de casos de uso reales
- Corrección de errores detectados

### Resultado

Software estable y validado



## Despliegue

### Objetivo

Poner el software en producción.

### Actividades

- Configuración del entorno de producción
- Pruebas finales en servidores reales
- Implementación de medidas de seguridad

## Resultado

Software disponible para el cliente final



## Mantenimiento

### Objetivo

Garantizar la operación continua del software.

### Actividades

- Resolución de problemas reportados
- Actualizaciones regulares
- Monitoreo del rendimiento

### Resultado

Software estable y actualizado

# 4. Estado de Requerimientos

Esta sección detalla el avance en el cumplimiento de los requerimientos funcionales y no funcionales, proporcionando una visión clara del progreso del proyecto.

- **Funcionales:** Describen las acciones y funcionalidades que el sistema debe cumplir.
- **No funcionales:** Abarcan aspectos como rendimiento, seguridad y escalabilidad.

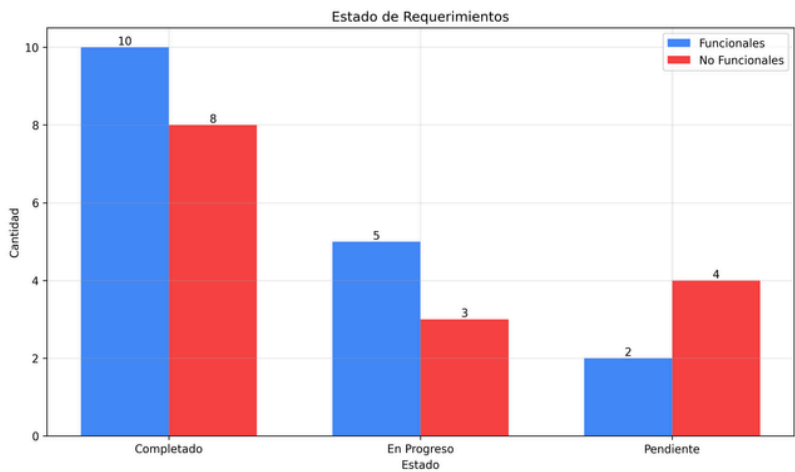
El siguiente cuadro muestra el estado actual de los requerimientos:

## 4.1 Requerimientos Funcionales y No Funcionales

| Estado      | Requerimientos Funcionales | Requerimientos No Funcionales |
|-------------|----------------------------|-------------------------------|
| Completado  | 10                         | 8                             |
| En Progreso | 5                          | 3                             |
| Pendiente   | 2                          | 4                             |
| Total       | 17                         | 15                            |

### Gráfico: Estado de Requerimientos

Representación gráfica del estado actual de los requerimientos funcionales y no funcionales.





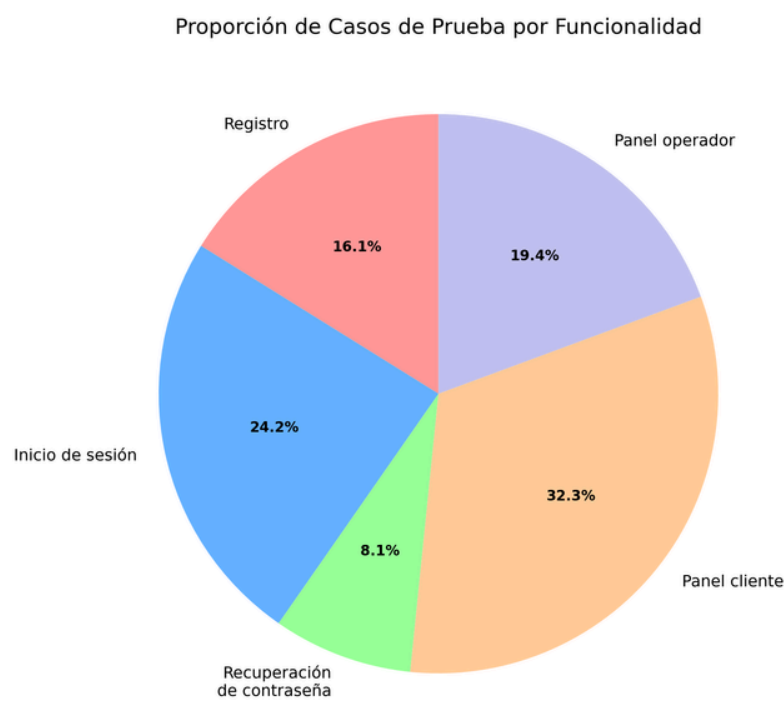
# 5. Casos de Prueba

## 5.1 Distribución de Casos de Prueba por Funcionalidad

| Funcionalidad              | Casos de Prueba | Porcentaje |
|----------------------------|-----------------|------------|
| Registro                   | 10              | 16.1%      |
| Inicio de sesión           | 15              | 24.2%      |
| Recuperación de contraseña | 5               | 8.1%       |
| Panel cliente              | 20              | 32.3%      |
| Panel operador             | 12              | 19.3%      |
| Total                      | 62              | 100%       |

### Gráfico: Proporción de Casos de Prueba

El gráfico muestra cómo se distribuyen los casos de prueba diseñados entre las funcionalidades clave.

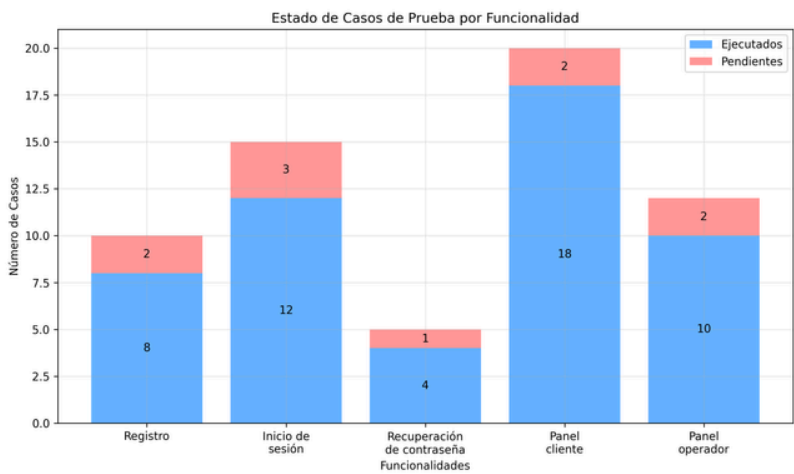


## 5.2 Estado de Ejecución de Casos de Prueba

| Funcionalidad              | Casos Ejecutados | Casos Pendientes | Total |
|----------------------------|------------------|------------------|-------|
| Registro                   | 8                | 2                | 10    |
| Inicio de sesión           | 12               | 3                | 15    |
| Recuperación de contraseña | 4                | 1                | 5     |
| Panel cliente              | 18               | 2                | 20    |
| Panel operador             | 10               | 2                | 12    |
| Total                      | 52               | 10               | 62    |

### Gráfico: Estado de Casos de Prueba

Este gráfico muestra los casos ejecutados y pendientes por funcionalidad, destacando áreas críticas.



## 5.3 Ciclos de Prueba

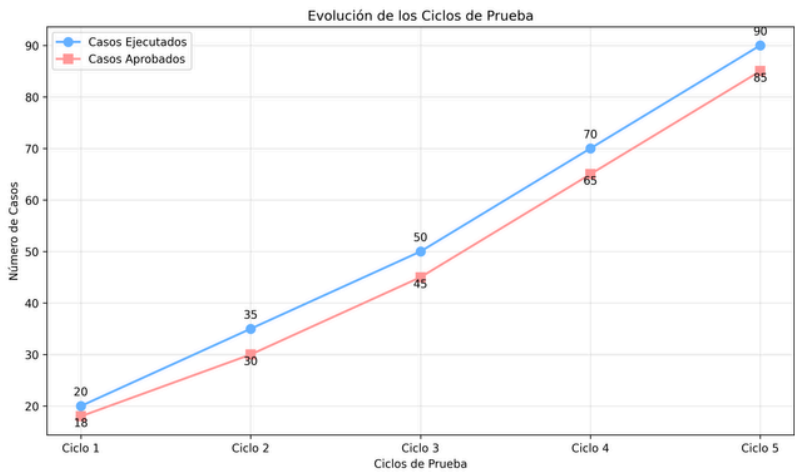
### 5.3.1 Evolución de los Ciclos de Prueba

| Ciclo   | Casos Ejecutados | Casos Aprobados |
|---------|------------------|-----------------|
| Ciclo 1 | 20               | 18              |
| Ciclo 2 | 35               | 30              |
| Ciclo 3 | 50               | 45              |

| Ciclo   | Casos Ejecutados | Casos Aprobados |
|---------|------------------|-----------------|
| Ciclo 4 | 70               | 65              |
| Ciclo 5 | 90               | 85              |

### 5.3.2 Gráficos de Seguimiento

- Gráfico: Proporción de Casos de Prueba
- Gráfico: Estado de Casos de Prueba
- Gráfico: Evolución de Ciclos de Prueba



## 6. Métricas y Seguimiento

### 6.1 Métricas Clave

- **Cobertura de Pruebas:** Se ha logrado ejecutar el 83.9% de los casos planificados, lo que indica un progreso significativo en la validación de funcionalidades.
- **Tasa de Aprobación:** Una alta tasa del 94.4% refleja la calidad y estabilidad del sistema en desarrollo.
- **Evolución Positiva:** El incremento constante en los casos aprobados por ciclo sugiere una mejora continua en la calidad del software.

# 7. Documentación de Defectos y Mejoras - Casino La Fortuna

## 7.1 Análisis de Defectos y Mejoras

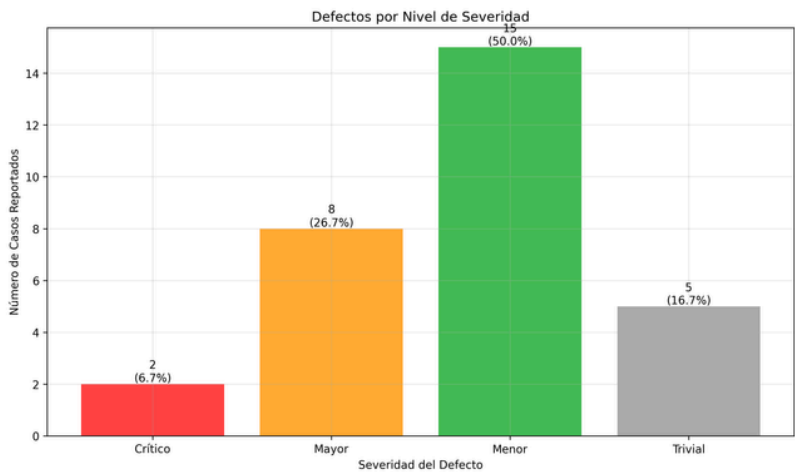
### 7.1.1 Defectos Reportados

Los defectos reportados durante el proceso de pruebas se han clasificado según su severidad, priorizando aquellos que impactan más en la funcionalidad del sistema.

| Severidad | Cantidad | Descripción                              | Estado                      |
|-----------|----------|--|-----------------------------|
| Crítico   | 2        | Impacto severo en funcionalidad core     | 2 Resueltos                 |
| Mayor     | 8        | Afecta significativamente la operación   | 6 Resueltos, 2 En Progreso  |
| Menor     | 15       | Impacto limitado en funcionalidad        | 10 Resueltos, 5 En Progreso |
| Trivial   | 5        | Problemas cosméticos o mejoras sugeridas | 3 Resueltos, 2 Pendientes   |
| Total     | 30       |  | 21 Resueltos (70%)          |

### Distribución de Defectos por Severidad

Este gráfico ilustra la proporción de defectos según su severidad, destacando el enfoque del equipo en resolver defectos críticos y mayores.



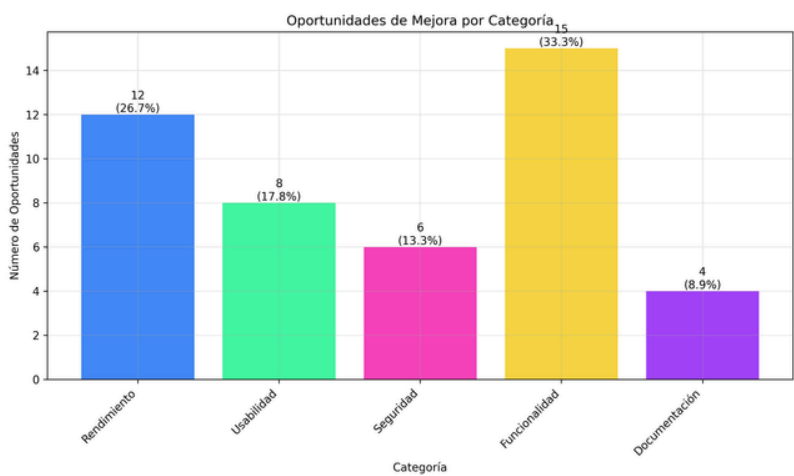
## 7.1.2 Oportunidades de Mejora Identificadas

Se han identificado 45 oportunidades de mejora distribuidas en las siguientes categorías:

| Categoría     | Cantidad | Descripción                                     | Prioridad |
|---------------|----------|---|-----------|
| Rendimiento   | 12       | Optimización de tiempos de respuesta y recursos | Alta      |
| Usabilidad    | 8        | Mejoras en la experiencia de usuario            | Media     |
| Seguridad     | 6        | Fortalecimiento de controles y validaciones     | Alta      |
| Funcionalidad | 15       | Expansión y refinamiento de características     | Media     |
| Documentación | 4        | Mejoras en guías y documentación técnica        | Baja      |
| Total         | 45       |   |           |

### Distribución de Oportunidades de Mejora

El gráfico muestra las áreas prioritarias para la mejora continua, enfocándose en rendimiento y funcionalidad.



## 7.1.3 Análisis de Tendencias

### Tasa de Resolución de Defectos:

- 70% de defectos resueltos (21 de 30)
- 100% de defectos críticos resueltos
- 75% de defectos mayores resueltos
- 67% de defectos menores resueltos
- 60% de defectos triviales resueltos

## Distribución de Esfuerzo:

- **40%** en corrección de defectos
- **35%** en implementación de mejoras
- **25%** en prevención y documentación

## 7.1.4 Recomendaciones Prioritarias

- 1. Corto Plazo:**
  - Resolver los 2 defectos mayores pendientes, identificados como críticos en el gráfico "Defectos por Severidad".
  - Implementar mejoras de rendimiento críticas.
- 2. Mediano Plazo:**
  - Abordar defectos menores pendientes.
  - Fortalecer pruebas automatizadas.
- 3. Largo Plazo:**
  - Desarrollar un plan de mejora continua.
  - Establecer métricas de calidad preventivas.

# 8. Lecciones Aprendidas

## 8.1 Gestión del Proyecto

| Área          | Lección Aprendida                                    | Impacto  | Acciones de Mejora                              |
|---------------|--|----------|---|
| Planificación | La estimación inicial no incluyó tiempo para pruebas | Alto     | Incluir actividades de testing en el cronograma |
| Comunicación  | Las reuniones frecuentes mejoraron la organización   | Positivo | Mantener la práctica en futuros proyectos       |
| Desarrollo    | Pruebas tempranas detectaron errores críticos        | Positivo | Implementar testing desde el inicio             |
| Documentación | La falta de estándares iniciales generó confusión    | Medio    | Crear plantillas básicas desde el inicio        |

## 8.2 Aspectos Técnicos

| Aspecto       | Aprendizaje   | Recomendación                 |
|---------------|---|-------------------------------|
| Modularidad   | Diseñar módulos pequeños facilitó la implementación | Seguir el enfoque modular     |
| Base de Datos | Consultas mal optimizadas afectaron el rendimiento  | Revisar y optimizar consultas |
| Seguridad     | Validaciones básicas evitaron accesos no deseados   | Integrar pruebas de seguridad |
| Interfaz      | Diseños simples ayudaron a usuarios finales         | Priorizar usabilidad          |

## 8.3 Mejores Prácticas Identificadas

1. **Desarrollo:**

- Dividir tareas grandes en entregas pequeñas
- Usar un control de versiones como Git para rastrear cambios
- Mantener el código organizado y documentado

2. **Testing:**

- Realizar pruebas manuales iniciales en módulos clave
- Validar cada funcionalidad antes de avanzar al siguiente módulo
- Documentar los resultados de las pruebas

3. **Gestión:**

- Documentar decisiones importantes
- Asignar tareas claras a cada miembro del equipo
- Mantener una comunicación constante

# 9. Gestión de Riesgos

## 9.1 Riesgos Materializados

| Riesgo                   | Impacto | Plan de Contingencia Activado              | Resultado   |
|--------------------------|---------|--|-------------|
| Retraso en tareas        | Medio   | Dividir tareas complejas en subtareas      | Mitigado    |
| Errores en validaciones  | Alto    | Revisar requerimientos y validar diseño    | Resuelto    |
| Problemas de integración | Alto    | Uso de documentación y pruebas adicionales | En Progreso |

# 10. Herramientas Utilizadas

El desarrollo del proyecto se apoyó en herramientas que facilitaron el diseño, implementación y pruebas. Las herramientas principales incluyen:

## 10.1 Desarrollo y Control de Versiones

| Categoría            | Herramienta        | Uso                               |
|----------------------|--------------------|-----------------------------------|
| IDE                  | Visual Studio Code | Desarrollo del frontend y backend |
| Control de Versiones | Git                | Rastreo de cambios en el código   |
| Repositorio          | GitHub             | Alojamiento del proyecto          |

## 10.2 Testing y Calidad

| Categoría      | Herramienta | Uso                      |
|----------------|-------------|--------------------------|
| Testing Manual | Navegadores | Validación de interfaces |
| Documentación  | Google Docs | Registro de pruebas      |



# 10.3 Gestión y Documentación

| Categoría         | Herramienta | Uso                                 |
|-------------------|-------------|-------------------------------------|
| Gestión de Tareas | Trello      | Seguimiento de actividades          |
| Comunicación      | WhatsApp    | Coordinación rápida entre el equipo |

# 10.4 Métricas de Uso de Herramientas

| Herramienta | Usuarios Activos | Frecuencia de Uso | Satisfacción |
|-------------|------------------|-------------------|--------------|
| VS Code     | 3                | Diaria            | 90%          |
| GitHub      | 3                | Semanal           | 85%          |
| Trello      | 3                | Semanal           | 80%          |

# 11. Recomendaciones

## Corto Plazo

- Mejorar la planificación inicial para incluir pruebas
- Revisar los módulos de seguridad y realizar ajustes básicos
- Establecer estándares de documentación básicos

## Mediano Plazo

- Implementar una plantilla estándar para la documentación
- Automatizar tareas repetitivas donde sea posible
- Mejorar las prácticas de control de versiones

## Largo Plazo

- Aprender a usar herramientas más avanzadas como Jenkins para CI/CD

- Explorar herramientas de monitoreo y métricas
- Implementar prácticas de desarrollo más avanzadas

## 12. Conclusiones del Aprendizaje

### 1. Logros Principales:

- Implementación exitosa de un sistema modular
- Desarrollo de habilidades en control de versiones
- Mejora en la organización y documentación

### 2. Áreas de Mejora:

- Profundizar en prácticas de testing
- Mejorar la estimación de tiempos
- Fortalecer conocimientos en seguridad

### 3. Próximos Pasos en el Aprendizaje:

- Explorar frameworks de testing
- Estudiar metodologías ágiles
- Practicar desarrollo basado en pruebas

## 13. Glosario de Términos

### Términos de Desarrollo

#### **RAD (Rapid Application Development)**

Metodología de desarrollo ágil que enfatiza el desarrollo rápido y la entrega iterativa de prototipos funcionales. Permite obtener retroalimentación temprana del cliente y realizar ajustes continuos durante el proceso de desarrollo.

#### **CRUD (Create, Read, Update, Delete)**

Operaciones básicas de persistencia de datos que debe realizar un sistema de información:

- Create: Crear o añadir nuevos registros

- Read: Leer o recuperar datos existentes
- Update: Actualizar o modificar registros
- Delete: Eliminar o borrar datos

## Frontend

Parte de la aplicación con la que interactúa directamente el usuario. Incluye la interfaz gráfica, formularios y todos los elementos visuales de la aplicación.

## Backend

Parte del sistema que procesa la lógica de negocio y gestiona los datos. Opera en el servidor y maneja las solicitudes del frontend, procesando y almacenando información en la base de datos.

## Términos de Calidad y Testing

### Defecto crítico

Error que impide completamente la operación del sistema, causando fallos que hacen imposible el uso de funcionalidades core del software.

### Defecto mayor

Error que afecta significativamente la funcionalidad pero permite operación limitada del sistema. Impacta severamente la experiencia del usuario pero no detiene completamente la operación.

### Defecto menor

Error que no impide la operación principal del sistema. Puede causar inconvenientes menores pero no afecta las funcionalidades principales.

### Defecto trivial

Problema cosmético o de baja prioridad que no afecta la funcionalidad. Generalmente relacionado con la interfaz de usuario o elementos no críticos.

# Términos Específicos del Proyecto

## Panel cliente

Interfaz específica para usuarios finales del Casino la Fortuna, diseñada para facilitar las operaciones cotidianas de los clientes del establecimiento.

## Panel operador

Interfaz administrativa para gestión del sistema, que permite el control y supervisión de todas las operaciones realizadas en la plataforma.

## Herramientas

### Git

Sistema de control de versiones distribuido que permite el seguimiento de cambios en el código fuente durante el desarrollo de software.

### Visual Studio Code

IDE (Entorno de Desarrollo Integrado) utilizado para el desarrollo del frontend y backend del proyecto.

## 14. Referencias

1. Documentación técnica inicial del proyecto.
2. Normativas contables aplicables.
3. Estándares de calidad ISO/IEC 25010.
4. Guía de mejores prácticas en testing de software.

# 15. Historial de Cambios

| Fecha      | Versión | Descripción                        | Autor     |
|------------|---------|------------------------------------|-----------|
| 2024-01-15 | 1.0     | Versión inicial                    | Equipo QA |
| 2024-02-01 | 1.1     | Actualización de métricas          | Equipo QA |
| 2024-02-15 | 1.2     | Inclusión de análisis de defectos  | Equipo QA |
| 2024-03-01 | 1.3     | Actualización de estado y métricas | Equipo QA |

# 16. Próximos Pasos

- 1. Completar los 10 casos pendientes.
- 2. Revisar defectos identificados.
- 3. Planificar y ejecutar el ciclo 6 de pruebas.
- 4. Preparar documentación de usuario final.

# 17. Anexos

## Referencias

- 1. Documentación técnica inicial.
- 2. Estándares de seguridad.