


PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

## NOMBRE DEL PROYECTO

Fin4Youth

## INTEGRANTES

Alejandro Córdoba Ríos

Juan Pablo Hoyos López

Juan Camilo Naranjo Cuartas

Adrián David Perdomo Echeverri

## ÁREA TEMÁTICA

Validación y Verificación de Software

Ingeniería de Sistemas

Facultad de Ingenierías


## LUGAR

Universidad de Medellín

Medellín, Colombia

## SEMESTRE

2024-1


PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

## HOJA DE CONTROL

<b>Sistema</b>	Sistema de servicios financieros digitales Fin4Youth		
<b>Entregable</b>	Plan Detallado de Pruebas		
<b>Autor</b>	F4Y Team		
<b>Versión</b>	1.0	<b>Fecha versión</b>	19/02/2024


## REGISTRO DE CAMBIOS

Versión	Causa del cambio	Responsable del cambio	Fecha del cambio
1.0	Versión inicial	F4Y Team	19/02/2024

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

## TABLA DE CONTENIDO

HOJA DE CONTROL .....	2
REGISTRO DE CAMBIOS.....	2
TABLA DE CONTENIDO .....	3
1. INTRODUCCIÓN .....	4
2. DESCRIPCIÓN DEL PRODUCTO .....	5
2.1. F4Y SERVER.....	6
2.2. F4Y CLIENT.....	6
2.3. HISTORIAS DE USUARIO .....	7
3. PLAN DE PRUEBAS.....	8
3.1. OBJETIVO DEL PLAN DE PRUEBAS.....	8
3.2. ALCANCE DE LAS PRUEBAS .....	9
3.3. DESCRIPCIÓN DE USUARIOS .....	9
3.4. ESTRATEGIAS Y TÉCNICAS DE PRUEBAS .....	10
3.4.1. ESTRATEGIA DE CAJA NEGRA .....	10
3.4.2. ESTRATEGIA DE CAJA BLANCA.....	12
3.5. TIPOS DE PRUEBAS .....	13
3.5.1. PRUEBAS UNITARIAS.....	13
3.5.2. PRUEBAS DE ACEPTACIÓN .....	13
3.5.3. PRUEBAS DE SEGURIDAD .....	14
3.6. EJECUCIÓN DE PRUEBAS .....	14
3.6.1. CRONOGRAMA DE EJECUCIÓN.....	14
3.6.2. RECURSOS NECESARIOS .....	15
3.6.3. AMBIENTES DE PRUEBAS .....	16
3.6.4. CRITERIOS DE ACEPTACIÓN DE PRUEBAS .....	17
3.6.5. REGISTRO DE RESULTADOS DE PRUEBAS.....	18
3.7. ENTREGABLES DE PRUEBAS .....	20

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		


## 1. INTRODUCCIÓN

En el ámbito del desarrollo de software, la Validación y Verificación (V&V) desempeña un papel fundamental para garantizar la calidad y fiabilidad de los productos de software. La Validación se refiere al proceso de asegurarse de que el software cumpla con los requisitos especificados por el cliente y las necesidades del usuario final, mientras que la Verificación implica la evaluación y confirmación de que el software se desarrolló de acuerdo con los estándares y procedimientos establecidos.

El presente trabajo se centra en la Validación y Verificación de un software previamente creado, con el objetivo de evaluar su funcionamiento, identificar posibles defectos y asegurar su adecuación a los requerimientos definidos. Para ello, se emplearán diversas técnicas y herramientas de prueba para examinar exhaustivamente el software en diferentes aspectos, tales como funcionalidad, rendimiento, seguridad y usabilidad.

Durante el desarrollo de este trabajo, se seguirá un enfoque sistemático y metodológico para llevar a cabo las actividades de Validación y Verificación. Esto incluirá la planificación y diseño de pruebas, la ejecución de pruebas en diferentes escenarios y condiciones, el análisis de los resultados obtenidos y la elaboración de informes detallados sobre el estado del software y las acciones correctivas recomendadas.

En resumen, el objetivo principal de este trabajo es garantizar la calidad y confiabilidad del software mediante la Validación y Verificación exhaustiva del

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

mismo. Se espera que los resultados obtenidos contribuyan a mejorar la experiencia del usuario, minimizar los riesgos asociados al uso del software y asegurar su correcto funcionamiento en el entorno para el que fue diseñado.


## 2. DESCRIPCIÓN DEL PRODUCTO

F4Y (Fin4Youth) es una aplicación web que simula un sistema financiero que ofrece servicios de ahorro y crédito, destinado inicialmente para jóvenes de 18 a 28 años en Colombia. Permite a los usuarios ahorrar su dinero por medio de una cuenta de ahorros, separar y aprovisionar su dinero usando bolsillos, y también ofrece una forma de hacer inversiones seguras con CDTs (Certificados de Depósito a Término).



*Logo de F4Y (Fin4Youth)*

Fin4Youth sale de "Finance For Youth" que en español significa "Finanzas para jóvenes".

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

La aplicación consta de los siguientes artefactos de software:

## 2.1. F4Y SERVER


Consiste en una aplicación backend y API, desarrollada con NodeJS y ExpressJS, usando el lenguaje de programación JavaScript. Se usa la arquitectura MVC (Modelo-Vista-Controlador) con el fin de tener de forma más modular y limpia la lógica de la aplicación.

Repositorio de GitHub: <https://github.com/lejito/f4y-server>

## 2.2. F4Y CLIENT

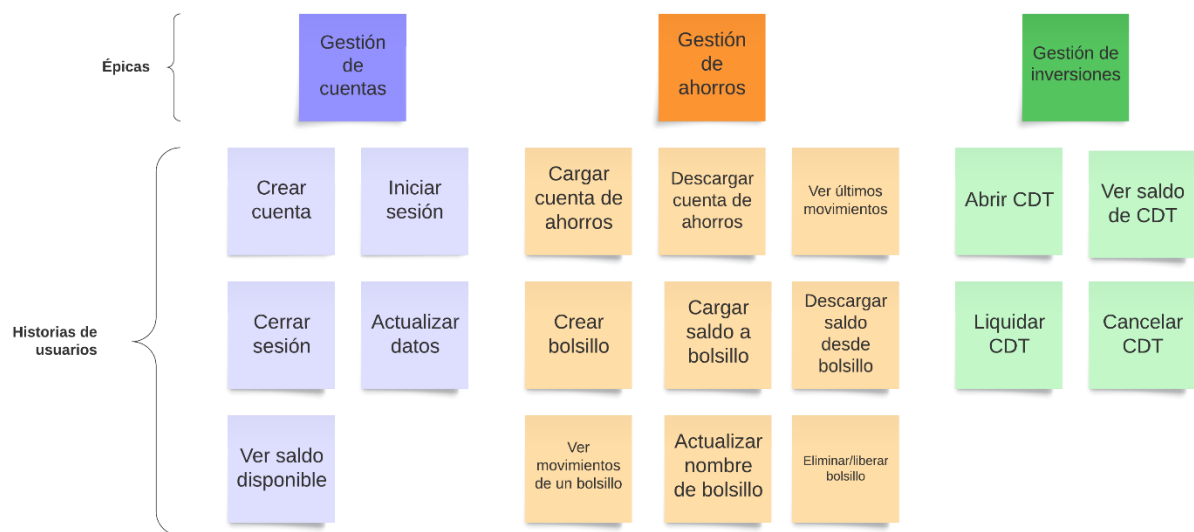
Consiste en una aplicación frontend, desarrollada con Angular 16, con el lenguaje de programación TypeScript. Se hace división por módulos, componentes y servicios para tener las responsabilidades de la aplicación separadas y hacer más escalable y mantenible la aplicación.

Repositorio de GitHub: <https://github.com/lejito/f4y-client>

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

### 2.3. HISTORIAS DE USUARIO


A continuación, se muestra el mapa de historias de usuario donde se podrán visualizar las funcionalidades que van a ser objeto de pruebas:



*Mapa de historias de usuario*

Matriz de historias de usuario:

[https://drive.google.com/file/d/1Dgnc1cWm9ZqKerR7may8dBLSIFzzndof/view?usp=drive\\_link](https://drive.google.com/file/d/1Dgnc1cWm9ZqKerR7may8dBLSIFzzndof/view?usp=drive_link)

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

### 3. PLAN DE PRUEBAS


Proyecto	Tipo de proyecto
Fin4Youth (F4Y)	Sistema de servicios financieros digitales
Documentos de evaluación relacionados	Equipo del proyecto
<a href="#">F4Y_Matriz_HU.pdf</a>	Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri

#### 3.1. OBJETIVO DEL PLAN DE PRUEBAS

Este documento tiene como finalidad entregar las pautas y definir la estrategia que se seguirá para llevar a cabo la certificación del software **Fin4Youth (F4Y)**.

El objetivo general del plan es establecer la cronología y condiciones para la aplicación de pruebas, y de este modo obtener, un sistema que pueda ser completado con una recepción total de los interesados y entrar en operación con la totalidad de las funcionalidades requeridas para su funcionamiento. Se busca validar y verificar el funcionamiento del software para garantizar su calidad y confiabilidad, identificando los posibles defectos y asegurar la adecuación a los requerimientos definidos.



PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		


### 3.2. ALCANCE DE LAS PRUEBAS

Las pruebas cubrirán tanto la aplicación backend (F4Y Server) como la aplicación frontend (F4Y Client). Se evaluarán aspectos como funcionalidad, rendimiento, seguridad y usabilidad del software. Al ser las dos aplicaciones necesarias para configurar un sistema o un “todo” con el que se logra cumplir los requerimientos funcionales, es necesario también, aparte de pruebas exhaustivas a cada aplicación, hacer pruebas de integración para verificar que el software en su totalidad funciona correctamente.

### 3.3. DESCRIPCIÓN DE USUARIOS

A continuación, se mostrará, por medio de la técnica Mad Libs, una tabla que lista a los principales usuarios interesados en el negocio al que pertenece el software:

Usuario	+	Necesidad del cambio	+	Revelación
Un joven universitario	Necesita	Ahorrar dinero	Porque	Debe financiar sus estudios
Un joven emprendedor	Necesita	Realizar una inversión	Porque	Quiere generar ganancias para ampliar la cobertura de su empresa
Un joven empleado	Necesita	Generar un ahorro programado	Porque	Así asegura su futuro
Un joven deportista	Necesita	Crear una cuenta de ahorros	Porque	Quiere independizarse financieramente

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		


### 3.4. ESTRATEGIAS Y TÉCNICAS DE PRUEBAS

Para la implementación del plan de pruebas al sistema de servicios financieros Fin4Youth, se considera de gran importancia considerar ciertas estrategias útiles para probar la eficiencia y eficacia del software:

#### 3.4.1. ESTRATEGIA DE CAJA NEGRA


La estrategia de caja negra consiste en realizar pruebas en las cuáles no interesa el proceso qué ocurre internamente y el código escrito en las funciones de software; solo a través de los datos de entrada y salida se determina que está funcionando correctamente. Esta estrategia encaja perfectamente en aquellas secciones más transaccionales de la aplicación, dónde la lógica es simple y se quiere corroborar que siempre con las mismas entradas se obtengan los resultados acertados de acuerdo con las reglas de negocio.

Con esta estrategia nos concentraremos en dos técnicas, la primera será la de conjetura de errores, donde haciendo uso de la experiencia de todo el equipo desarrollando, planearemos los errores más típicos para así atacar lo más probable que falle en el programa. La siguiente técnica es en modo de contingencia para poder dar con aquellas situaciones que se salen de la experticia que tenemos y así poder asegurar que tenemos un producto bueno; en este caso usaremos el análisis de valores límites. Con esta técnica, apegándonos a los requisitos funcionales y sus criterios de aceptación, buscaremos darle al sistema los datos más improbables de entrada y así ver si el comportamiento de salida es el supuesto de acuerdo con lo documentado para el proyecto.

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

A continuación, se presenta una tabla donde se nombran las historias de usuario que más se adaptan a ser verificadas mediante esta estrategia, justificando:

Historia de usuario	Justificación
Crear cuenta	El proceso de crear una cuenta carece de mucha lógica interna, pero tiene muchos campos definidos con validaciones establecidas en las reglas de negocio, las cuales son importantes de asegurar.
Iniciar sesión	Es una lógica simple, pero es vital, ya que es la encargada de validar si un usuario está registrado para poder utilizar la plataforma y guardar sus datos y configuraciones. Es importante asegurarse que solo con los datos correctos es posible iniciar sesión.
Actualizar datos	Es común en plataformas como la de proyecto, tener que actualizar datos de un usuario para que se encuentren al día con la realidad de la persona, pero es altamente probable que permitir a un usuario realizar actualizaciones en sus datos, no mantengan la integridad de las validaciones usadas al crear la cuenta. Por esto, se debe probar que, al realizar este proceso, solo sea posible hacer efectivo un cambio si se mantienen las reglas del campo que se intenta actualizar.
Ver saldo disponible	Es saldo es algo común, pero íntimo. Debe ser totalmente seguro para el usuario al ser un dato de alto valor y privado, y es obligatorio que solo el dueño de la cuenta, el personal autorizado y nadie más pueda ver el saldo disponible en la cuenta. Acá se debe comprobar que con los datos de entrada de un usuario específico solo sea posible, para él, ver su propio estado de cuenta.
Crear bolsillo	Es una funcionalidad de ahorro creado para que los usuarios puedan separar dinero de sus cuentas principales y hacerlos inaccesibles por lo menos directamente y forzarlos a sacar el dinero de bolsillo de nuevo a la cuenta para usarlo. Es cómo tal un proceso sencillo pero crítico, ya que existen flujos de dinero únicamente de forma interna en la cuenta de usuario y, por ello, es obligatorio comprobar que los límites estén establecidos correctamente en la lógica, para que no


PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

	sea posible ingresar más dinero de que se tiene actualmente en la cuenta, ni regresar a la cuenta más dinero de que se tiene en el bolsillo.
Eliminar y liberar bolsillo	Se debe probar que, al eliminar el bolsillo, solo el dinero qué había reservado en el bolsillo, se encuentre nuevamente en la cuenta.

### 3.4.2. ESTRATEGIA DE CAJA BLANCA

Esta estrategia consiste en darle un mayor peso a la lógica que existe en el código. A diferencia de la estrategia de la caja negra, que solo centra en los resultados, esta estrategia se enfoca en que cada una de las rutas que existen en la lógica sea probada. Estableciendo los casos de uso correctos y los valores iniciales para cada camino posible que se crea en la lógica, esta estrategia se adapta mejor a aquellos segmentos en los cuales la lógica que se realiza para procesar los datos es más profunda que en otros puntos, y es necesario usar una estrategia que nos permita ir a fondo y probar todos los casos probables para asegurar que la integridad de cada camino sea el documentado al levantar los requisitos funcionales.

La estrategia de caja blanca la usaremos por el simple hecho de que hay partes del software donde es posible identificar múltiples caminos y resultados posibles a partir de dichos caminos, lo que nos lleva a “quitar el velo” y dejar ver la abstracción que la estrategia de caja negra no nos dejaría observar. Esta estrategia se usará para funcionalidades selectas donde se identifiquen dicha variedad de casos a partir de distintos valores de entrada.

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

### 3.5. TIPOS DE PRUEBAS

Se contemplan los siguientes tipos de pruebas que se realizarán al software, a manera general:

#### 3.5.1. PRUEBAS UNITARIAS


Las pruebas unitarias tienen como objetivo verificar la funcionalidad y estructura de cada componente individualmente del sistema una vez que ha sido codificado.

Es una prueba técnica que permitirá:

- Verificar que los módulos del sistema estén libres de errores.
- Que todos los caminos lógicos principales deben ejecutarse correctamente en cada módulo de la aplicación.
- Todas las transacciones deben ser probadas.
- Todos los tipos de registro de entrada válido deben ser procesados.
- Todos los tipos de registro de entrada inválido deben ser también procesados correctamente.

#### 3.5.2. PRUEBAS DE ACEPTACIÓN

El objetivo de las pruebas de aceptación es validar que la implementación del sistema Fin4Youth cumpla con el funcionamiento esperado y permitir al usuario de dicho sistema determinar su aceptación, desde el punto de vista de su funcionalidad y de su rendimiento.

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

Estas pruebas corresponden a la ejecución de las siguientes pruebas por parte de los usuarios funcionales o el cliente:

- Pruebas funcionales.
- Pruebas de usabilidad.
- Pruebas de configuración.

### 3.5.3. PRUEBAS DE SEGURIDAD

El propósito de esta prueba es establecer y mantener la integridad del sistema de información mediante dos enfoques:


- **Pruebas de seguridad de la aplicación:** donde se verifica que un actor solo pueda acceder a las funciones y datos que el usuario tiene permitido.
- **Pruebas de seguridad del sistema:** donde se verifica que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla.

## 3.6. EJECUCIÓN DE PRUEBAS

### 3.6.1. CRONOGRAMA DE EJECUCIÓN

A continuación, se presentan las fases del cronograma de ejecución del plan de pruebas:

Fase	Actividades
Fase 1: Preparación	<ul style="list-style-type: none"> <li>• Configuración del entorno de pruebas.</li> <li>• Elaboración de casos de prueba.</li> <li>• Preparación de datos de prueba.</li> </ul>
Fase 2: Ejecución	<ul style="list-style-type: none"> <li>• Pruebas unitarias.</li> </ul>

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

	<ul style="list-style-type: none"> <li>• Pruebas de integración.</li> <li>• Pruebas funcionales.</li> <li>• Pruebas de rendimiento.</li> <li>• Pruebas de seguridad.</li> </ul>
Fase 3: Cierre	<ul style="list-style-type: none"> <li>• Análisis de resultados.</li> <li>• Reporte de defectos.</li> <li>• Preparación de informes.</li> <li>• Revisión y aprobación.</li> <li>• Planificación de acciones correctivas.</li> </ul>

### 3.6.2. RECURSOS NECESARIOS


Se listan las siguientes herramientas que se consideran necesarias para llevar a cabo el proceso de pruebas de la aplicación Fin4Youth:

#### Hardware:

- Servidores de prueba para el backend F4Y Server.
- Dispositivos de prueba para la aplicación frontend F4Y Client (inicialmente se requiere de un computador portátil o de escritorio).

#### Software:

- SO y plataformas compatibles con NodeJS, Angular como Windows, macOS o Linux.
- Herramientas de desarrollo (IDEs) como Visual Studio Code para editar y depurar el código.
- Herramientas de pruebas automatizadas como Jasmine, Jest o Supertest para automatizar tanto pruebas del frontend como el backend.

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

- Herramientas de gestión de proyectos como Jira, para gestionar las tareas relacionadas con las pruebas y el desarrollo.

#### Personal humano:


- Equipo de desarrollo: desarrolladores de software responsables de implementar cambios y correcciones basadas en los resultados de las pruebas.
- Equipo de QA: Ingenieros de pruebas encargados de diseñar, ejecutar y analizar las pruebas, así como reportar y gestionar los defectos encontrados.
- Líder de proyecto: Persona encargada de coordinar y supervisar las actividades de prueba, así como comunicar los resultados y tomar decisiones basadas en ellos.

#### 3.6.3. AMBIENTES DE PRUEBAS

La siguiente tabla muestra los ambientes definidos para el proceso de pruebas:

Ambiente	Descripción	Configuración
Desarrollo	Este ambiente se utiliza para el desarrollo y pruebas iniciales de nuevas funcionalidades. Es un entorno controlado donde los desarrolladores pueden trabajar en el código y realizar pruebas unitarias antes de integrar los cambios en el repositorio principal.	<ul style="list-style-type: none"> <li>• Servidor de Base de Datos: Configurado con una base de datos de prueba PostgreSQL para almacenar datos de prueba.</li> <li>• Servidor Backend (F4Y Server): Configurado para ejecutar el servidor NodeJS y ExpressJS.</li> <li>• Servidor Frontend (F4Y Client): Configurado para ejecutar la aplicación Angular (versión 16) en modo de desarrollo.</li> </ul>




PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

Pruebas	Este ambiente se utiliza para pruebas exhaustivas de integración y funcionalidad antes del lanzamiento a producción. Es una réplica del entorno de producción, pero con datos de prueba y configuraciones específicas para pruebas.	<ul style="list-style-type: none"> <li>• Servidor de Base de Datos: Réplica de la base de datos de producción con datos de prueba anonimizados o generados artificialmente.</li> <li>• Servidor Backend (F4Y Server): Réplica del servidor de producción con la última versión del código implementado.</li> <li>• Servidor Frontend (F4Y Client): Réplica del servidor frontend con la aplicación Angular compilada y optimizada para pruebas.</li> </ul>
Producción	Este ambiente es donde se despliega la versión final del software y se pone a disposición de los usuarios finales. Es crítico mantener la estabilidad y disponibilidad de este ambiente para garantizar una experiencia de usuario óptima.	<ul style="list-style-type: none"> <li>• Servidor de Base de Datos: Configurado con una base de datos de producción y copias de seguridad regulares para garantizar la integridad de los datos.</li> <li>• Servidor Backend (F4Y Server): Configurado con las últimas actualizaciones y parches de seguridad aplicados.</li> <li>• Servidor Frontend (F4Y Client): Configurado para servir la aplicación Angular de forma segura y eficiente a los usuarios finales.</li> </ul>

### 3.6.4. CRITERIOS DE ACEPTACIÓN DE PRUEBAS

Se exponen los criterios de aceptación para cada uno de los componentes que se consideran importantes para la calidad de la aplicación:

Componente	Criterio	Métrica de calidad
Funcionalidad	Todas las funcionalidades principales de la aplicación deben funcionar según lo especificado en las	Se espera que al menos el 95% de las pruebas de funcionalidad automatizadas pasen con éxito.

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		


	historias de usuario y los requisitos del cliente.	
Rendimiento	La aplicación debe ser capaz de manejar un volumen esperado de usuarios y transacciones sin experimentar tiempos de respuesta excesivos o fallas del sistema.	El tiempo de respuesta promedio de la aplicación no debe superar los 3 segundos bajo una carga de 100 usuarios concurrentes.
Seguridad	La aplicación debe proteger la información del usuario y garantizar la integridad de los datos mediante medidas de seguridad adecuadas.	No se deben encontrar vulnerabilidades críticas durante las pruebas de seguridad.
Usabilidad	La aplicación debe ser intuitiva y fácil de usar para los usuarios finales, minimizando la necesidad de asistencia o capacitación adicional.	Al menos el 80% de los usuarios piloto deben calificar la aplicación con una puntuación de satisfacción de 7 o superior en una escala del 1 al 10.

### 3.6.5. REGISTRO DE RESULTADOS DE PRUEBAS

Se presenta la forma en que se registrarán los resultados de las pruebas efectuadas a los artefactos de software:

#### Metodología de registro:

- Todos los resultados de las pruebas serán registrados de manera detallada en un sistema de seguimiento de problemas o en una hoja de cálculo compartida, accesible para todos los miembros del equipo de desarrollo y QA.
- Se utilizará un formato estándar para registrar los resultados de las pruebas, que incluirá información detallada sobre cada prueba realizada, incluyendo el

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		


caso de prueba, el resultado esperado, el resultado real, y cualquier observación o comentario relevante.

### Formato de reporte de defectos:

- Los defectos identificados durante las pruebas serán documentados utilizando un formato estandarizado de reporte de defectos.
- Cada reporte de defecto incluirá los siguientes campos:
  - Título: Una descripción breve pero descriptiva del defecto.
  - Descripción: Detalles sobre el defecto, incluyendo cómo reproducirlo y su impacto en la aplicación.
  - Prioridad: La importancia relativa del defecto en relación con otros, generalmente clasificado como crítico, mayor o menor.
  - Estado: El estado actual del defecto, como Nuevo, Asignado, En Progreso, Resuelto, Verificado, o Cerrado.
  - Responsable: El miembro del equipo responsable de corregir el defecto.
  - Fecha de Creación y Última Actualización: Fechas importantes relacionadas con la creación y el progreso del defecto.

### Seguimiento de acciones correctivas:

- Se asignará a un miembro del equipo la responsabilidad de seguir y documentar las acciones correctivas para cada defecto identificado durante las pruebas.

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		


- Se establecerá un proceso para verificar y validar la resolución de cada defecto antes de cerrarlo en el sistema de seguimiento de problemas.
- Se mantendrá un historial completo de todas las acciones correctivas tomadas para cada defecto, incluyendo cambios en el código, pruebas adicionales realizadas, y cualquier otra actividad relevante.

#### Informe de resultados de pruebas:

- Al finalizar las pruebas, se elaborará un informe completo que resuma los resultados de las pruebas, incluyendo estadísticas sobre el número de pruebas realizadas, número de defectos encontrados, y métricas de calidad como la tasa de éxito de las pruebas.
- El informe también incluirá una evaluación general de la calidad del software, destacando los aspectos positivos y las áreas de mejora identificadas durante las pruebas.
- Se proporcionarán recomendaciones para futuras iteraciones del software, basadas en los hallazgos de las pruebas y las lecciones aprendidas durante el proceso.

### 3.7. ENTREGABLES DE PRUEBAS

De acuerdo con el tipo de pruebas ejecutadas puede que el entregable del mismo sea diferente. En la siguiente tabla se señalan los diferentes entregables por cada tipo de pruebas:

PLAN DETALLADO DE PRUEBAS			Versión: 1.0
Validación y Verificación de Software		Alejandro Córdoba Ríos Juan Pablo Hoyos López Juan Camilo Naranjo Cuartas Adrián David Perdomo Echeverri	
	Proyecto: Fin4Youth		

Tipo de pruebas	Entregables
Pruebas funcionales	Se entregará un documento de pruebas de funcionales, que incluye resultados de la ejecución de los scripts de pruebas y análisis de los defectos encontrados durante el proceso de pruebas y solicitud de las correcciones recibidas.
Pruebas de aceptación	Resumen de validación de la prueba.
Pruebas de seguridad	Resultado de pruebas funcionales de seguridad.