

UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA  
FACULTAD DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN



JOSÉ ROBERTO IBOY SICUIR	5190-10-4883
JULIO ESTIVEN DUBÓN MORALES	5190-19-679
MANUEL EDUARDO ORDOÑEZ SILVA	5190-19-4050
CARLOS EDUARDO GALINDO MIRANDA	5190-19-5184
FERNANDO ISAAC PICHE CIFUENTES	5190-19-1099

## ÍNDICE

ÍNDICE.....	1
Introducción.....	5
CAPÍTULO I.....	6
Generalidad.....	6
1.1   Calidad de software .....	6
1.1.1   La calidad del software:.....	6
✓   Funcionalidad:.....	6
✓   Fiabilidad:.....	6
✓   Usabilidad: .....	6
✓   Rendimiento: .....	6
✓   Mantenibilidad: .....	6
1.2   Pruebas de software .....	6
✓   Pruebas funcionales:.....	7
✓   Pruebas de regresión:.....	7
✓   Pruebas de rendimiento: .....	7
✓   Pruebas de seguridad: .....	7
✓   Pruebas de usabilidad: .....	7
1.3   Proceso de pruebas.....	7
1.4   Niveles de prueba.....	7
Pruebas Unitarias:.....	7
Pruebas de Integración: .....	7
Pruebas de Sistema: .....	8
Pruebas de Aceptación: .....	8
1.5   Tipos de pruebas .....	8

1.5.1	Pruebas funcionales .....	8
1.5.2	Pruebas no funcionales .....	8
1.6	Alcances.....	10
1.7	Viabilidad.....	<b>¡Error! Marcador no definido.</b>
1.7.1	Viabilidad operativa .....	<b>¡Error! Marcador no definido.</b>
1.7.2	Viabilidad financiera .....	<b>¡Error! Marcador no definido.</b>
1.7.3	Viabilidad tecnológica.....	<b>¡Error! Marcador no definido.</b>
CAPÍTULO II.....		11
2.	Plan de pruebas de calidad de software: .....	11
2.1.	Pruebas de aceptación.....	11
2.1.2.	Criterios de aceptación .....	11
2.1.3.	Criterios de aceptación para la página de Login .....	11
2.1.4.	Criterios de aceptación para la lista de pacientes .....	12
2.1.5.	Criterios de aceptación para la edición y eliminación de pacientes .....	12
2.1.6.	Criterios de aceptación para el Registro de Nuevos Pacientes.....	12
2.1.7.	Criterios de Aceptación para el Registro de Nuevos Usuarios.....	12
2.1.8.	Casos de prueba .....	13
	Casos de Prueba para la Página de Login.....	13
	Caso de Prueba 1: Inicio de Sesión Exitoso .....	13
	Caso de Prueba 2: Inicio de Sesión Fallido .....	13
	Caso de Prueba 3: Restricción de Acceso No Autorizado .....	13
	Casos de Prueba para la Lista de Pacientes .....	13
	Caso de Prueba 4: Visualización de la Lista de Pacientes.....	14
	Casos de Prueba para la Edición y Eliminación de Pacientes .....	14
	Caso de Prueba 5: Edición de Información de Paciente .....	14
	Caso de Prueba 6: Eliminación de Paciente .....	14
	Casos de Prueba para el Registro de Nuevos Pacientes .....	14
	Caso de Prueba 7: Registro de Nuevo Paciente.....	14
	Casos de Prueba para el Registro de Nuevos Usuarios .....	15
	Caso de Prueba 8: Registro de Nuevo Usuario.....	15

Ejecución de casos de prueba .....	15
Caso de Prueba 1: Inicio de Sesión Exitoso .....	15
Caso de Prueba 2: Inicio de Sesión Fallido .....	15
Caso de Prueba 3: Restricción de Acceso No Autorizado .....	17
Caso de Prueba 4: Visualización de la Lista de Pacientes.....	18
Caso de Prueba 5: Edición de Información de Paciente.....	20
Caso de Prueba 6: Eliminación de Paciente .....	24
Caso de Prueba 7: Registro de Nuevo Paciente.....	28
Caso de Prueba 8: Registro de Nuevo Usuario.....	32
2.2. Pruebas de Extremo a Extremo: .....	36
2.2.1. Proceso de Login y Veterinarios .....	36
2.2.2. Registrar Veterinario .....	38
2.2.3. Olvide mi contraseña:.....	40
2.2.4. Confirmar nueva contraseña.....	41
2.2.5. Testing de errores: .....	42
Confirmar nueva contraseña:.....	42
Cypress esta validando el texto y el color de los mensajes de error.....	42
2.2.6. Errores Registro de Veterinarios: .....	47
Error Olvide mi contraseña: .....	51
Pacientes .....	53
Registrar .....	53
Editar: .....	55
Eliminar: .....	56
2.3. Pruebas Unitarias:.....	57
2.3.1. Registrar usuario veterinario: .....	58
2.3.2. Autenticación por token: .....	59
2.3.3. Autenticar usuario.....	60
2.3.4. Olvidar contraseña.....	60
2.3.5. Crear un nuevo paciente .....	64
2.3.6. Modificación de paciente.....	65

2.3.7. Eliminar paciente.....	67
Conclusiones.....	68
Glosario .....	71
Anexos.....	74
Anexo A.....	74
Anexo B. Pruebas de aceptacion .....	77
Anexo C.....	77
Anexo D. Aplicación Web.....	77
Bibliografía.....	78

## **Introducción**

En el dinámico mundo del desarrollo de software, el aseguramiento de la calidad es una piedra angular para garantizar que los productos y sistemas cumplan con los estándares de excelencia que los usuarios finales esperan. Entre las numerosas prácticas que contribuyen a este proceso, las pruebas de software desempeñan un papel crucial. Las pruebas son un componente esencial en la validación y verificación de un software, permitiendo a los equipos de desarrollo identificar defectos, errores y problemas potenciales antes de que lleguen a los usuarios. En este contexto, este artículo se adentrará en la importancia y características clave de las pruebas de software como herramienta fundamental para el aseguramiento de la calidad.

La manera en que se garantiza esta calidad es mediante un proceso de pruebas riguroso y bien planificado. Sin embargo, el éxito de las pruebas depende en gran medida de un elemento clave: definir un alcance adecuado. El alcance de las pruebas se refiere a la delimitación precisa de qué aspectos del software se evaluarán y bajo qué condiciones se llevarán a cabo las pruebas. En esencia, es como trazar un mapa detallado que guía a los equipos de prueba a través de un territorio digital en constante cambio. En este artículo, exploraremos la importancia del alcance de las pruebas en el desarrollo de software, junto con algunas recomendaciones clave para definirlo de manera efectiva.

# CAPÍTULO I

## Generalidad

### 1.1 Calidad de software

#### 1.1.1 La calidad del software:

Se refiere a la medida en que el software cumple con los requisitos y las expectativas del usuario, y está libre de defectos que puedan afectar su funcionamiento. La gestión de la calidad del software implica planificar, controlar y asegurar que el software se desarrolle y entregue de manera consistente y conforme a los estándares de calidad establecidos. Algunos aspectos clave de la calidad del software incluyen:

✓ **Funcionalidad:**

El software debe cumplir con todas las funciones y características requeridas.

✓ **Fiabilidad:**

El software debe ser confiable y consistente en su comportamiento.

✓ **Usabilidad:**

El software debe ser fácil de usar y proporcionar una buena experiencia de usuario.

✓ **Rendimiento:**

El software debe funcionar de manera eficiente y responder rápidamente.

✓ **Mantenibilidad:**

El software debe ser fácil de mantener y actualizar.

### 1.2 Pruebas de software

Las pruebas de software son el proceso de evaluación de un sistema o una aplicación de software para identificar defectos, errores o problemas que puedan afectar su funcionamiento.

Estas pruebas se realizan para asegurarse de que el software cumple con los requisitos especificados y funcione correctamente. Algunos tipos comunes de pruebas de software incluyen:

- ✓ **Pruebas funcionales:** Verifican que las funciones y características del software en función en según lo previsto.
- ✓ **Pruebas de regresión:** Se realizan para asegurarse de que las modificaciones en el código no afecten negativamente a las funciones existentes.
- ✓ **Pruebas de rendimiento:** Evalúan el rendimiento, la escalabilidad y la capacidad de respuesta del software bajo diferentes cargas y condiciones.
- ✓ **Pruebas de seguridad:** Buscan vulnerabilidades y riesgos de seguridad en el software para garantizar la protección de datos y sistemas.
- ✓ **Pruebas de usabilidad:** Evalúan la facilidad de uso y la experiencia del usuario para garantizar que el software sea intuitivo y accesible.

### 1.3 Proceso de pruebas

Los procesos de desarrollo de software son los procedimientos y enfoques que una organización sigue para crear, mantener y mejorar el software. Estos procesos son esenciales para garantizar una gestión eficiente y coherente del proyecto de desarrollo de software.

### 1.4 Niveles de prueba

#### Pruebas Unitarias:

- Tipo: Funcionales
- Propósito: Verificar que las unidades individuales de código, como funciones o métodos, funcionen correctamente.
- Herramientas: Frameworks de pruebas unitarias como JUnit, NUnit, pytest, etc.

#### Pruebas de Integración:

- Tipo: Funcionales
- Propósito: Evaluar la interacción entre diferentes componentes o módulos del sistema.

- Herramientas: Dependiendo del lenguaje y la tecnología, pueden utilizarse herramientas específicas o pruebas manuales.

### **Pruebas de Sistema:**

- Tipo: Funcionales
- Propósito: Probar el sistema completo para asegurarse de que cumple con los requisitos funcionales especificados.
- Herramientas: Pruebas manuales y automatizadas, herramientas de gestión de pruebas.

### **Pruebas de Aceptación:**

- Tipo: Funcionales
- Propósito: Evaluar si el sistema cumple con los criterios de aceptación definidos por el cliente o las partes interesadas.
- Herramientas: Pruebas manuales y automatizadas, a veces utilizando herramientas específicas de aceptación como Cucumber o SpecFlow.

## **1.5 Tipos de pruebas**

### **1.5.1 Pruebas funcionales**

Las pruebas funcionales en el contexto del testing de software son un tipo de prueba que se centra en verificar si el software cumple con las funciones y características especificadas en su diseño y requisitos. Estas pruebas evalúan si el software realiza correctamente las acciones que se esperan de él y si responde adecuadamente a diversas entradas.

### **1.5.2 Pruebas no funcionales**

Las pruebas de software no funcionales son las que se hacen desde una perspectiva totalmente diferente a las pruebas automatizadas. Este tipo de plan de pruebas son un medio de control de calidad, que se realiza en aplicaciones de software para asegurarse de que todo funciona bien y poder saber en qué circunstancias podrían fallar.

Las pruebas no funcionales de software nos permiten conocer qué riesgos corre el producto y nos dicen si tiene un mal desempeño o un bajo rendimiento en los entornos de producción.

En ese sentido, las pruebas de software no funcionales se hacen con el fin de obtener información. Permiten explicar lo que soporta el producto y si cumple con las expectativas de los clientes.

Tipos de pruebas no funcionales (performance)

#### **1.5.2.1 Pruebas de carga**

Estas pruebas se hacen con el objetivo de determinar y validar la respuesta de la aplicación cuando esta está sometida a una carga de un cierto número de usuarios o de peticiones.

Ejemplo: Verificar si el producto puede soportar la carga de 100 usuarios de forma simultánea. Este resultado se compara con el volumen esperado.

#### **1.5.2.2 Pruebas de rendimiento**

El principal objetivo de este tipo de pruebas no funcionales es calcular la respuesta de la aplicación con diferentes medidas de usuario o peticiones.

Ejemplo: conocer cuál es la respuesta al procesar el ingreso de 10, 100 y 1000 usuarios de forma parametrizada. Este resultado se compara con el resultado esperado.

#### **1.5.2.3 Pruebas de estrés**

Estas pruebas se realizan para encontrar el número de usuarios, peticiones o tiempos que la aplicación puede soportar. Este tipo de pruebas no funcionales son muy semejantes a las pruebas de carga y rendimiento, pero se diferencian en que debemos superar los límites esperados en el ambiente de producción o los límites que fueron determinados en las pruebas.

Ejemplo: encontrar la cantidad de usuarios que soporta de manera simultánea hasta que la aplicación deja de responder (cuelgue o time out), haciéndolo de forma correcta según todas las peticiones. (Morales, 2021)

## 1.6 Alcances

El alcance de las pruebas en el contexto del desarrollo de software se refiere a la definición precisa de qué aspectos del software se evaluarán y bajo qué condiciones. Establecer un alcance adecuado es fundamental para asegurar que las pruebas sean efectivas y se centren en los aspectos más relevantes del software.

Definir un alcance adecuado es esencial para asegurarse de que las pruebas sean efectivas y cumplan con los objetivos del proyecto de desarrollo de software. Esto permite que las pruebas se enfoquen en los aspectos críticos del software y ayuden a garantizar su calidad y funcionamiento adecuado.

## CAPÍTULO II

### 2. Plan de pruebas de calidad de software:

#### 2.1. Pruebas de aceptación

En estas pruebas se tiene como objetivo presentar el proceso de pruebas de aceptación realizadas en el sistema de registro de pacientes para una veterinaria. El sistema, diseñado para facilitar la gestión de pacientes y usuarios en un entorno veterinario, ha sido sometido a un riguroso proceso de pruebas con el fin de garantizar su funcionalidad, seguridad y cumplimiento de los requisitos establecidos.

El propósito de esta sección es proporcionar una descripción detallada de las pruebas de aceptación realizadas en el sistema, los criterios de aceptación definidos, los casos de prueba utilizados y los resultados obtenidos. Además, se busca obtener la aprobación del cliente o usuario final para la implementación del sistema en un entorno de producción.

En las secciones siguientes, se detallarán los criterios de aceptación, los casos de prueba, el proceso de ejecución de pruebas y los resultados obtenidos durante el proceso de pruebas de aceptación.

##### 2.1.2. Criterios de aceptación

Los criterios de aceptación establecen los estándares que el sistema de registro de pacientes para veterinaria debe cumplir para considerarse exitoso y listo para su implementación. Estos criterios se han definido en colaboración con el cliente y se basan en los requisitos y expectativas del sistema.

##### 2.1.3. Criterios de aceptación para la página de Login

- El sistema debe proporcionar una página de inicio de sesión segura que solicite un nombre de usuario y una contraseña.
- Un usuario debe poder iniciar sesión con éxito utilizando credenciales válidas.
- Se debe mostrar un mensaje de error adecuado si las credenciales ingresadas son incorrectas.

- El acceso a la página de inicio de sesión debe estar restringido a usuarios autorizados.
- Después del inicio de sesión, el usuario debe ser redirigido a la página principal del sistema.

#### **2.1.4. Criterios de aceptación para la lista de pacientes**

- El sistema debe mostrar una lista de pacientes registrados en la veterinaria.
- La lista de pacientes debe incluir información básica, como el nombre del paciente, la especie y la fecha de registro.
- Los usuarios deben poder buscar pacientes por nombre o especie.
- Los usuarios autorizados deben tener la capacidad de editar la información de un paciente existente.

#### **2.1.5. Criterios de aceptación para la edición y eliminación de pacientes**

- Se debe permitir la actualización de datos, como el nombre, la especie, la fecha de nacimiento y el historial médico.
- Los usuarios deben confirmar antes de eliminar un paciente existente.
- Despues de la eliminación, el paciente debe ser removido de la lista de pacientes de manera adecuada.

#### **2.1.6. Criterios de aceptación para el Registro de Nuevos Pacientes**

- Los usuarios registrados deben poder registrar nuevos pacientes en el sistema.
- Se debe recopilar información esencial, como el nombre, la especie, la fecha de nacimiento y el historial médico.
- El sistema debe asignar automáticamente un número de identificación único a cada nuevo paciente registrado.

#### **2.1.7. Criterios de Aceptación para el Registro de Nuevos Usuarios**

- Los datos requeridos para el registro de usuarios incluyen nombre, dirección de correo electrónico y contraseña.

- Los nuevos usuarios registrados deben poder iniciar sesión en el sistema.

### **2.1.8. Casos de prueba**

Los casos de prueba se han diseñado para evaluar el cumplimiento de los criterios de aceptación establecidos para el sistema de registro de pacientes de la veterinaria. A continuación, se presentan los casos de prueba para las diferentes funcionalidades del sistema.

#### **Casos de Prueba para la Página de Login**

En esta sección, se detallan los casos de prueba relacionados con la funcionalidad de inicio de sesión en el sistema. Estos casos de prueba se centran en evaluar la capacidad del sistema para autenticar a los usuarios y permitirles acceder al sistema de manera segura.

##### **Caso de Prueba 1: Inicio de Sesión Exitoso**

- Acción: Ingresar un nombre de usuario y una contraseña válidos.
- Resultado Esperado: El sistema permite el acceso y redirige al usuario a la página principal.

##### **Caso de Prueba 2: Inicio de Sesión Fallido**

- Acción: Ingresar un nombre de usuario o una contraseña incorrectos.
- Resultado Esperado: El sistema muestra un mensaje de error adecuado y no permite el acceso.

##### **Caso de Prueba 3: Restricción de Acceso No Autorizado**

- Acción: Intentar acceder a la página de inicio de sesión sin autenticación.
- Resultado Esperado: El sistema redirige al usuario no autenticado a la página de inicio de sesión.

#### **Casos de Prueba para la Lista de Pacientes**

En esta sección, se presentan los casos de prueba relacionados con la gestión de la lista de pacientes en el sistema. Estos casos de prueba se enfocan en verificar la capacidad del sistema para mostrar a los pacientes registrados.

#### **Caso de Prueba 4: Visualización de la Lista de Pacientes**

- Acción: Iniciar sesión y acceder a la lista de pacientes.
- Resultado Esperado: El sistema muestra la lista de pacientes registrados.

#### **Casos de Prueba para la Edición y Eliminación de Pacientes**

En esta sección, se describen los casos de prueba relacionados con la edición y eliminación de información de pacientes existentes en el sistema. Estos casos de prueba evalúan la capacidad del sistema para realizar estas operaciones de manera efectiva.

#### **Caso de Prueba 5: Edición de Información de Paciente**

- Acción: Seleccionar un paciente y editar su información.
- Resultado Esperado: El sistema permite la edición de los datos del paciente y guarda los cambios correctamente.

#### **Caso de Prueba 6: Eliminación de Paciente**

- Acción: Seleccionar un paciente y solicitar su eliminación.
- Resultado Esperado: El sistema muestra una confirmación y elimina al paciente de la lista.

#### **Casos de Prueba para el Registro de Nuevos Pacientes**

En esta sección, se detallan los casos de prueba que se centran en el proceso de registro de nuevos pacientes en el sistema. Estos casos de prueba verifican la capacidad del sistema para capturar y almacenar información de pacientes recién registrados.

#### **Caso de Prueba 7: Registro de Nuevo Paciente**

- Acción: Registrar un nuevo paciente con información válida.
- Resultado Esperado: El sistema registra el nuevo paciente y lo muestra en la lista.

## **Casos de Prueba para el Registro de Nuevos Usuarios**

En esta sección, se presentan los casos de prueba relacionados con el registro de nuevos usuarios, especialmente por parte de los administradores del sistema. Estos casos de prueba evalúan la capacidad del sistema para gestionar el registro de usuarios de manera eficiente.

### **Caso de Prueba 8: Registro de Nuevo Usuario**

- Acción: registrarse como un nuevo usuario.
- Resultado Esperado: El sistema registra al nuevo usuario y permite su inicio de sesión.

### **Ejecución de casos de prueba**

En esta sección se llevarán a cabo los casos de prueba previamente definidos para evaluar el sistema. Cada caso de prueba se ejecutará siguiendo los pasos detallados y se registrarán los resultados obtenidos.

#### **Caso de Prueba 1: Inicio de Sesión Exitoso**

Pasos:

1. Abra el navegador web e ingrese la URL del sistema de registro de pacientes.
2. En la página de inicio de sesión, ingrese el nombre de usuario y la contraseña.
3. Haga clic en el botón "Iniciar Sesión".

Resultado de la ejecución:

- El sistema permitió el acceso y redirigió al usuario a la página principal del sistema.
- No se identificaron problemas durante la ejecución.
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.
- No es necesario adjuntar capturas de pantalla.

#### **Caso de Prueba 2: Inicio de Sesión Fallido**

Pasos:

1. Abra el navegador web e ingrese la URL del sistema de registro de pacientes.
2. En la página de inicio de sesión, ingrese un nombre de usuario incorrecto, como "usuario\_incorrecto", y una contraseña incorrecta, como "contraseña\_incorrecta".

3. Haga clic en el botón "Iniciar Sesión".

Resultados de la ejecución:

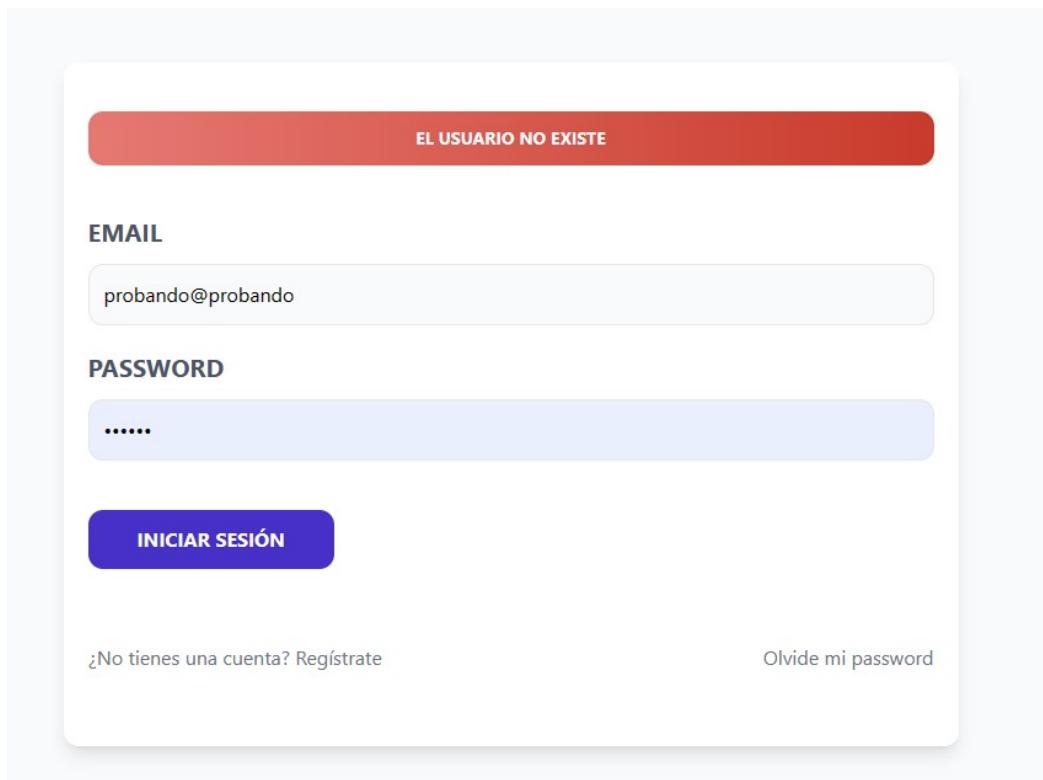
- El sistema no permitió el acceso y mostró un mensaje de error indicando que las credenciales ingresadas son incorrectas.
- Se adjuntan capturas de pantalla para mostrar la página de inicio de sesión y el mensaje de error.
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

Capturas de pantalla

Imagen 1: Inicio de sesión con credenciales incorrectas



## Imagen 2: Acceso denegado



## Caso de Prueba 3: Restricción de Acceso No Autorizado

Pasos:

1. Abra el navegador web e ingrese la URL del sistema de registro de pacientes.
2. Verifique que no haya una sesión de usuario iniciada y que no se haya proporcionado ninguna credencial de inicio de sesión.
3. Intente acceder directamente a la página de inicio principal escribiendo la URL en la barra de direcciones.

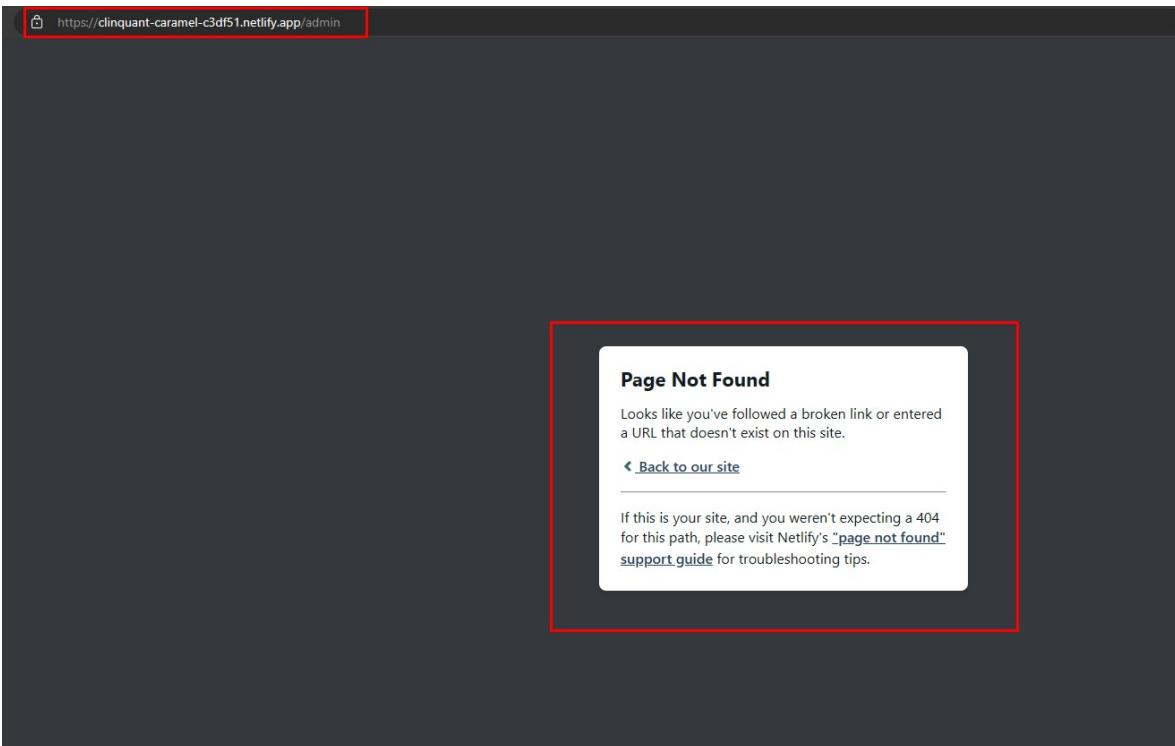
Resultados de la ejecución:

- El sistema detectó que no se había iniciado sesión y redirigió al usuario a la página de inicio de sesión o a una página de acceso restringido.

- Se adjuntan capturas de pantalla para mostrar la página correspondiente y cualquier mensaje de error si es aplicable.
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

#### Capturas

Imagen 1: acceso denegado



#### Caso de Prueba 4: Visualización de la Lista de Pacientes

##### Pasos:

1. Abra el navegador web e ingrese la URL del sistema de registro de pacientes.
2. En la página de inicio de sesión, ingrese un nombre de usuario válido y una contraseña válida.
3. Haga clic en el botón "Iniciar Sesión" para acceder al sistema.

4. Una vez en la página principal del sistema, le listara el listado de pacientes.

Resultados de la ejecución:

- El sistema permitió el acceso después de una autenticación exitosa y mostró la lista de pacientes registrados.
- Se adjunta una captura de pantalla que muestra la lista de pacientes en la página principal.
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

Capturas de pantalla:

Imagen 1: página de login

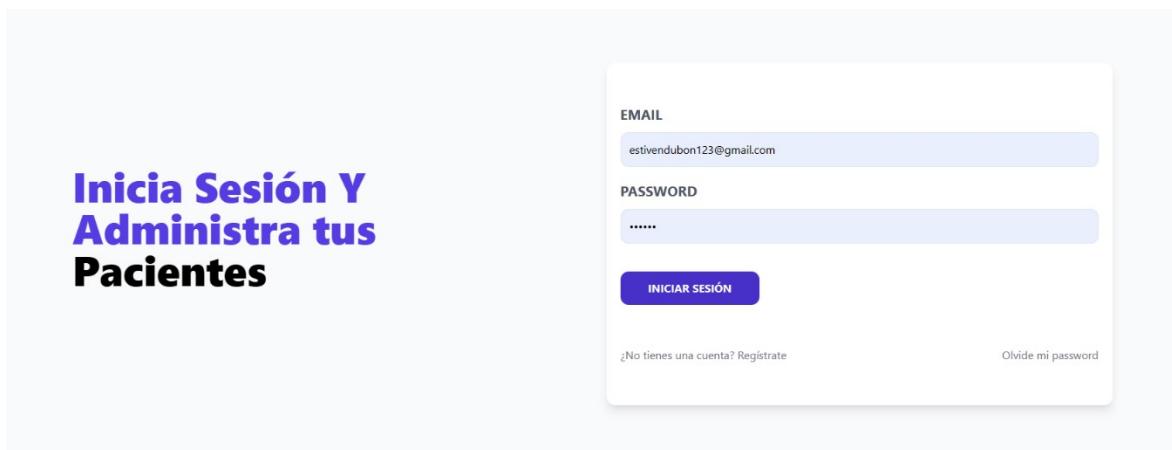


Imagen 2: login exitoso y listado de pacientes

**NOMBRE:** Teo

**PROPIETARIO:** Tato

**EMAIL:** aherandez@gmail.com

**FECHA DE ALTA:** 10 de septiembre de 2020

**SINTOMAS:** dolor de panza

**EDITAR**

**ELIMINAR**

**NOMBRE:** Tortuga

**PROPIETARIO:** Carlos

**EMAIL:** correo@correo.com

**FECHA DE ALTA:** 13 de septiembre de 2023

**SINTOMAS:** No come

**EDITAR**

**ELIMINAR**

## Caso de Prueba 5: Edición de Información de Paciente

**Pasos:**

1. Abra el navegador web e ingrese la URL del sistema de registro de pacientes.
2. Inicie sesión con un nombre de usuario válido y una contraseña válida.
3. Acceda a la lista de pacientes registrados.
4. Seleccione un paciente de la lista al que desee editar.
5. En la página de detalles del paciente, ubique la opción de edición o modificación de la información del paciente.
6. Edite uno o varios campos de información del paciente, como el nombre, el síntoma, el propietario, etc.
7. Guarde los cambios realizados en la información del paciente.

Resultados de la ejecución:

- El sistema permitió la edición de la información del paciente seleccionado.
- Se adjuntan capturas de pantalla para mostrar la página de detalles del paciente antes de la edición, la página de edición de la información del paciente y la página de detalles del paciente después de la edición.
- Los cambios realizados se reflejaron correctamente en la información del paciente.
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

Capturas de pantalla:

Captura de Pantalla 1: Datos del paciente antes de la edición.

**NOMBRE MASCOTA**

Tortuga

**NOMBRE PROPIETARIO**

Carlos

**EMAIL PROPIETARIO**

correo@correo.com

**FECHA ALTA**

dd/mm/aaaa

**SÍNTOMAS**

No come

**GUARDAR CAMBIOS**

Captura de Pantalla 2: Campos modificados.

**NOMBRE MASCOTA**

Tortuga

**NOMBRE PROPIETARIO**

Carlos

**EMAIL PROPIETARIO**

carlos@prueba.com

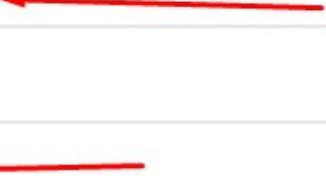
**FECHA ALTA**

06/10/2023

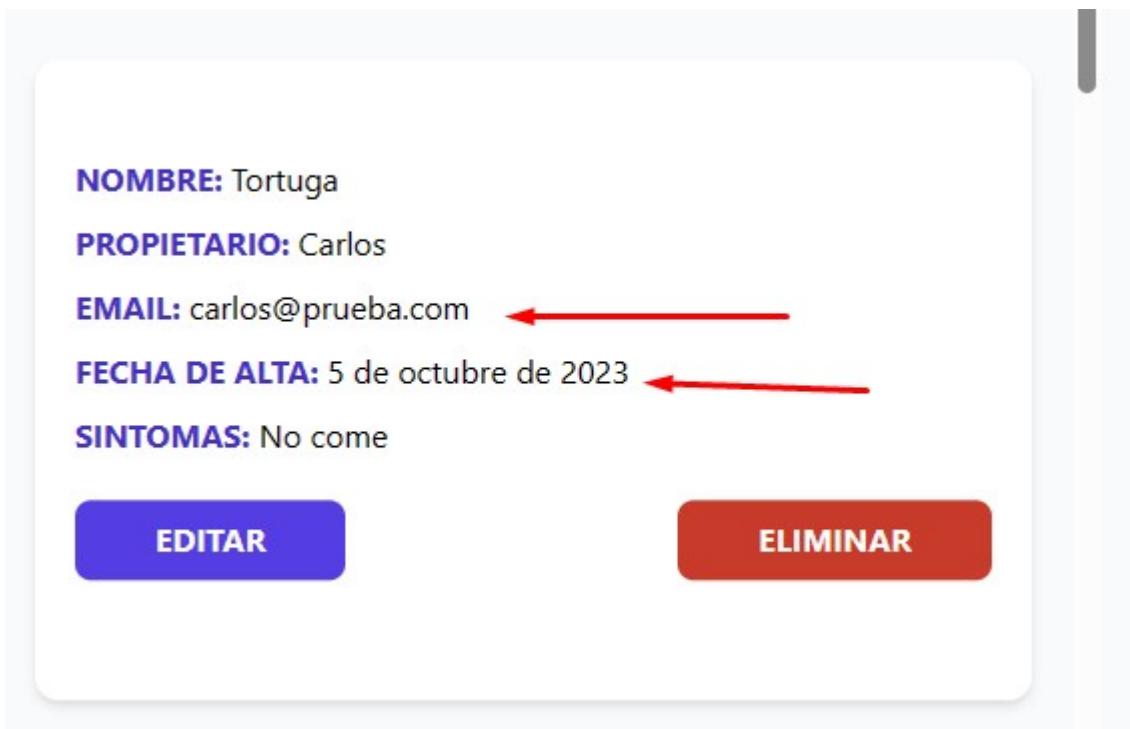
**SÍNTOMAS**

No come

**GUARDAR CAMBIOS**



Captura de Pantalla 3: Datos modificados



### Caso de Prueba 6: Eliminación de Paciente

Pasos:

1. Acceda a la lista de pacientes registrados.
2. Seleccione un paciente de la lista que desee eliminar.
3. En la página de detalles del paciente, ubique la opción de eliminación o eliminación de paciente.
4. Confirme la eliminación del paciente cuando se le solicite.

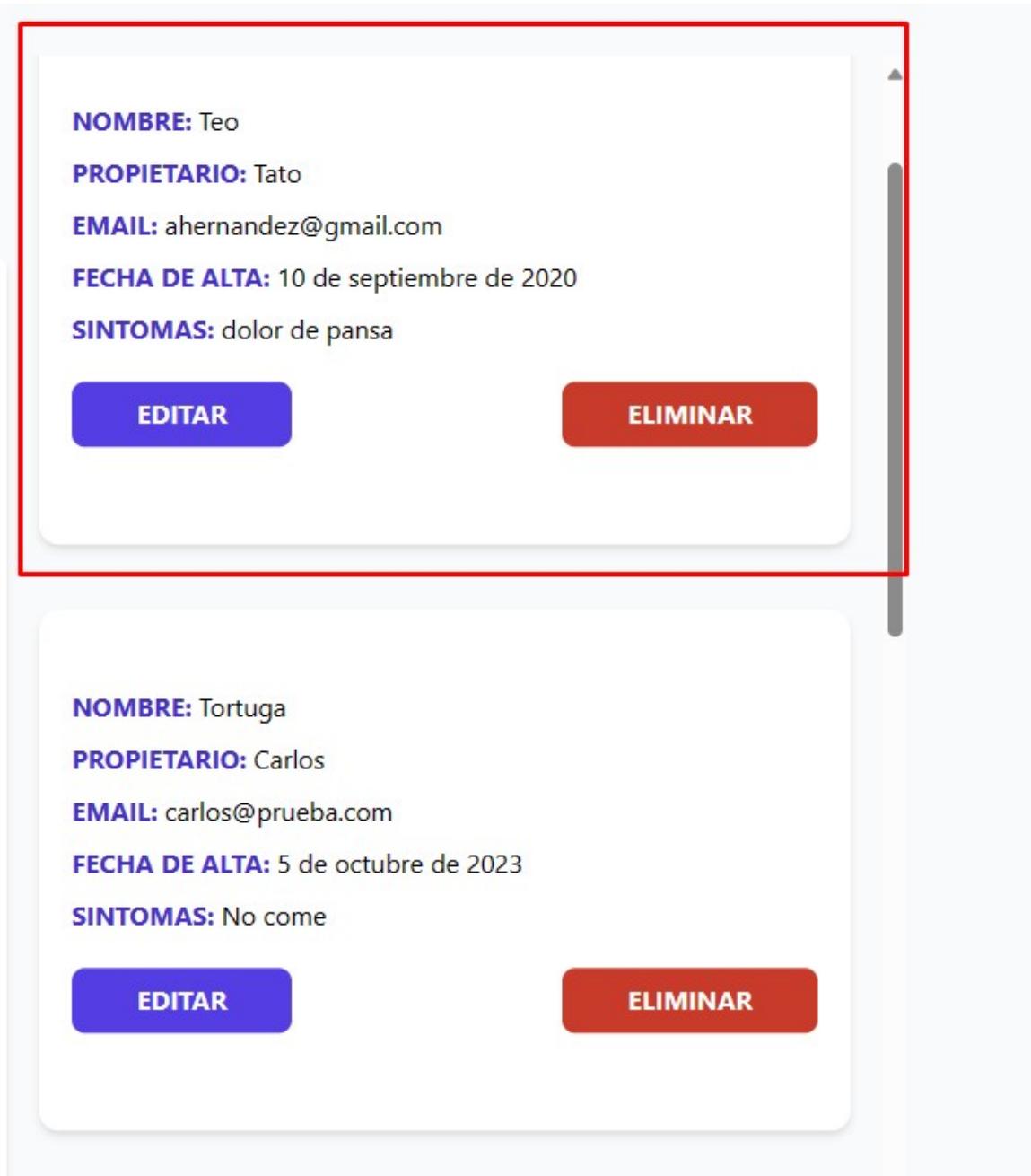
Resultados de la ejecución:

- El sistema permitió la eliminación del paciente seleccionado después de la confirmación.
- Se adjuntan capturas de pantalla para mostrar la página de detalles del paciente antes de la eliminación, la confirmación de eliminación del paciente y la lista de pacientes después de la eliminación.
- El paciente seleccionado se eliminó correctamente de la lista de pacientes.

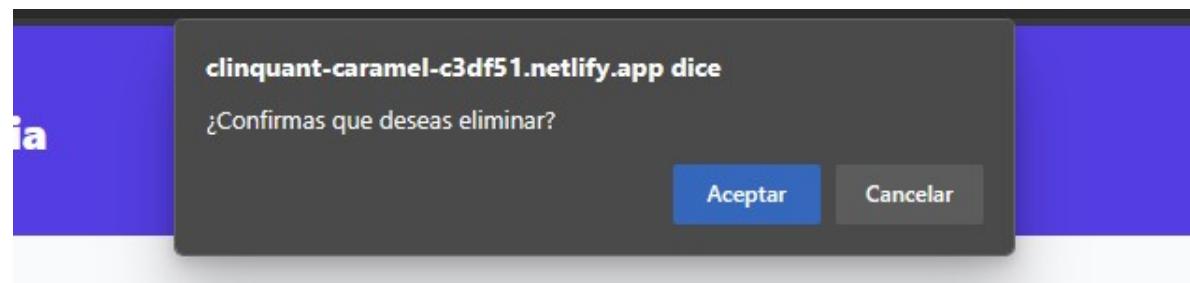
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

### Capturas de pantalla

Captura de Pantalla 1: Paciente a eliminar



Captura de Pantalla 2: confirmación de eliminación



Captura de Pantalla 3: listado de pacientes actualizado

# Listado Pacientes

Administra tus **Pacientes y Citas**

**NOMBRE:** Tortuga

**PROPIETARIO:** Carlos

**EMAIL:** carlos@prueba.com

**FECHA DE ALTA:** 5 de octubre de 2023

**SINTOMAS:** No come

**EDITAR**

**ELIMINAR**

**NOMBRE:** Cerberus

**PROPIETARIO:** Fitzwilliam Darcy

**EMAIL:** orgulloYprejuicio@gmail.com

**FECHA DE ALTA:** 8 de septiembre de 2023

**SINTOMAS:** Lele pancha jajaja

**EDITAR**

**ELIMINAR**

## **Caso de Prueba 7: Registro de Nuevo Paciente**

Pasos:

1. Acceda a la sección de registro de nuevos pacientes en el sistema.
2. Complete los campos requeridos del formulario de registro de nuevo paciente, incluyendo nombre, nombre del propietario, email propietario, fecha de alta y los síntomas.
3. Haga clic en el botón “Aregar Paciente” para completar el proceso de registro.

Resultados de la ejecución:

1. El sistema permitió el registro de un nuevo paciente después de completar el formulario de registro.
2. Se adjuntan capturas de pantalla para mostrar el formulario de registro de nuevo paciente antes de enviar los datos y la confirmación de registro o la lista de pacientes después de registrar al nuevo paciente.
3. El paciente recién registrado se muestra correctamente en la lista de pacientes.
4. El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

Capturas de pantalla:

Captura de Pantalla 1: Formulario para registrar el paciente.

# **Administrador de Pacientes**

Añade tus pacientes **Administralos**

**NOMBRE MASCOTA**

**NOMBRE PROPIETARIO**

**EMAIL PROPIETARIO**

**FECHA ALTA**

CALENDAR

**SÍNTOMAS**

**AGREGAR PACIENTE**

Captura de Pantalla 2: Formulario con datos.

# Administrador de Pacientes

Añade tus pacientes [Administralos](#)

**NOMBRE MASCOTA**

tortuga

**NOMBRE PROPIETARIO**

carlos galindo

**EMAIL PROPIETARIO**

carlo@galindo.com

**FECHA ALTA**

05/10/2023



**SÍNTOMAS**

No come

**AGREGAR PACIENTE**

Captura 3: listado de pacientes actualizado.

# Listado Pacientes

Administra tus **Pacientes y Citas**

**NOMBRE:** tortuga

**PROPIETARIO:** carlos galindo

**EMAIL:** carlo@galindo.com

**FECHA DE ALTA:** 4 de octubre de 2023

**SINTOMAS:** No come

**EDITAR**

**ELIMINAR**

**NOMBRE:** salchicha

**PROPIETARIO:** julio dubon

**EMAIL:** correo@gmail.com

**FECHA DE ALTA:** 4 de octubre de 2023

**SINTOMAS:** Dolos de panza para el perro salchicha

**EDITAR**

**ELIMINAR**

## **Caso de Prueba 8: Registro de Nuevo Usuario**

Pasos:

1. Abra el navegador web e ingrese la URL del sistema de registro de pacientes.
2. Acceda al enlace “¿No tienes una cuenta? Regístrate”
3. Complete los campos requeridos del formulario de registro de nuevo usuario, incluyendo nombre, dirección de correo electrónico, nombre de usuario y contraseña.
4. Haga clic en el botón "Iniciar sesión" para completar el proceso de registro de nuevo usuario.
5. Revisar la bandeja de correos del email para validar la cuenta creada.
6. Hacer click sobre el enlace para validar la cuenta.
7. Iniciar sesión con la cuenta creada

Resultados de la ejecución:

- El sistema permitió el registro de un nuevo usuario por parte del administrador después de completar el formulario de registro.
- Se adjuntan capturas de pantalla para mostrar el formulario de registro de nuevo usuario antes de enviar los datos y la confirmación de registro o la lista de usuarios después de registrar al nuevo usuario.
- El nuevo usuario puede iniciar sesión en el sistema.
- El caso de prueba se ejecutó exitosamente de acuerdo con los resultados esperados.

Capturas de pantalla:

Captura 1: Enlace para registrarse.

**EMAIL**

**PASSWORD**

**INICIAR SESIÓN**

¿No tienes una cuenta? Regístrate

Olvide mi password

Captura 2: Formulario de registro

**NOMBRE**

**EMAIL**

**PASSWORD**

**REPETIR PASSWORD**

**INICIAR SESIÓN**

Ya tienes una cuenta? Inicia Sesión

Olvide mi password

Captura 3: correo de validación de cuenta



Captura 4: enlace para validar la cuenta

**Comprueba tu cuenta en APV**

From: <>  
To: <julio@gmail.com>

Show Headers

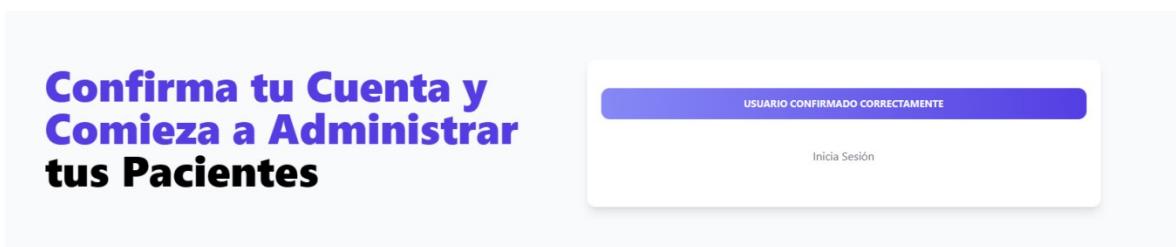
HTML    HTML Source    Text    Raw    Spam Analysis    HTML Check 2    Tech Inf

A close-up photograph of a brown and black dog's face, looking slightly to the side.

Hola julio dubon,  
¡Tu cuenta en APV está lista para ser verificada!  
Para completar el proceso, simplemente haz clic en el siguiente enlace:  
[Comprobar Cuenta](#)

Si no has creado esta cuenta, por favor ignora este mensaje.

Captura 5: confirmación de cuenta

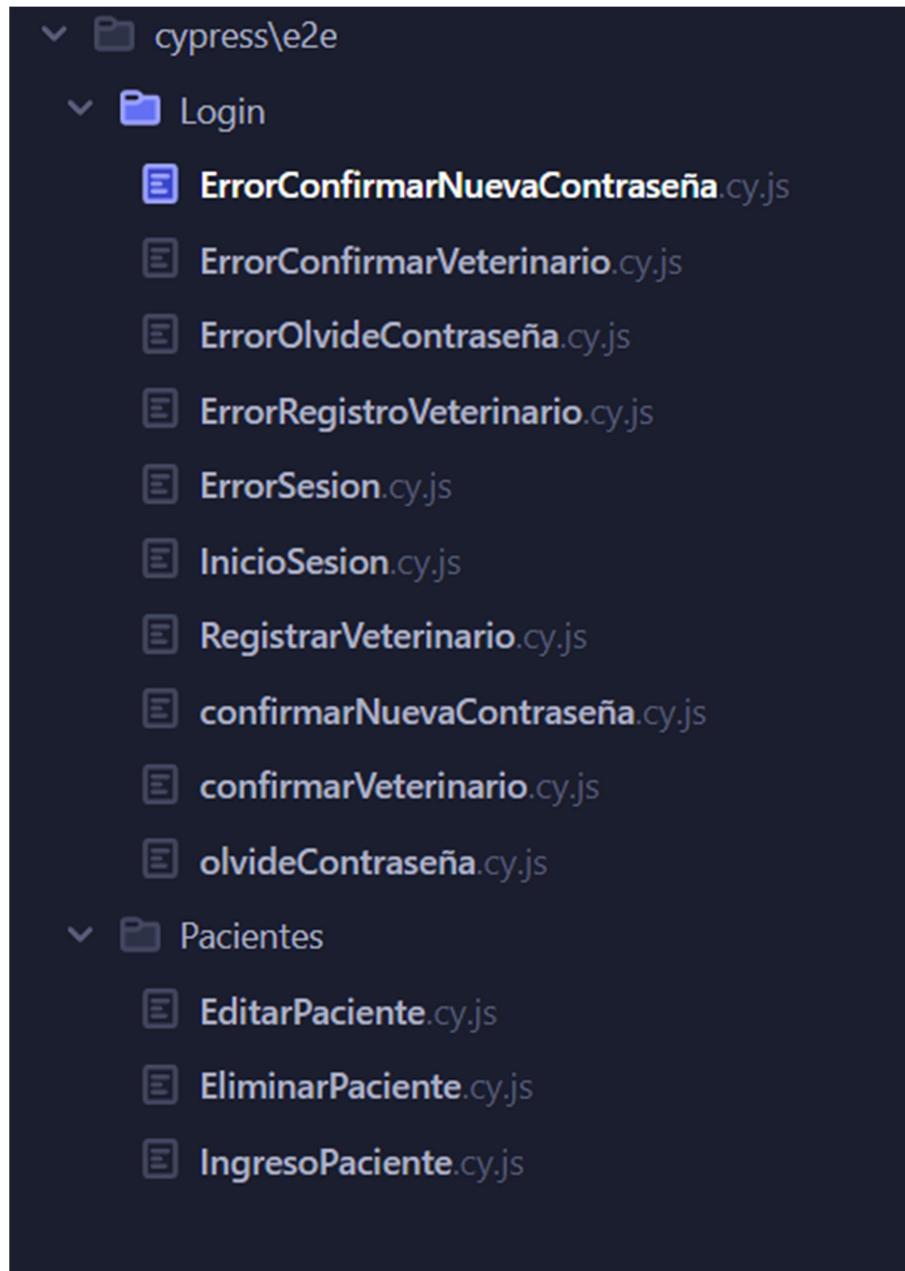


Captura 6: inicio de sesión con la nueva cuenta

A login form with fields for "EMAIL" containing "julio@gmail.com" and "PASSWORD" containing ".....". A blue button labeled "INICIAR SESIÓN" is at the bottom. Below the form are links for "¿No tienes una cuenta? Regístrate" and "Olvide mi password".

EMAIL	julio@gmail.com
PASSWORD	.....
<b>INICIAR SESIÓN</b>	
¿No tienes una cuenta? <a href="#">Regístrate</a>	
<a href="#">Olvide mi password</a>	

## 2.2. Pruebas de Extremo a Extremo:



### 2.2.1. Proceso de Login y Veterinarios

#### Inicio sesión:

Este proceso se logró digitando por medio de Cypress el correo y contraseña.

# Inicia Sesión Y Administra tus Pacientes

EMAIL

manuel@manuel.com

PASSWORD

.....|

INICIAR SESIÓN

¿No tienes una cuenta? Regístrate

Olvide mi password

## ✓ Administrador de Citas

### ✓ Inicio de sesion

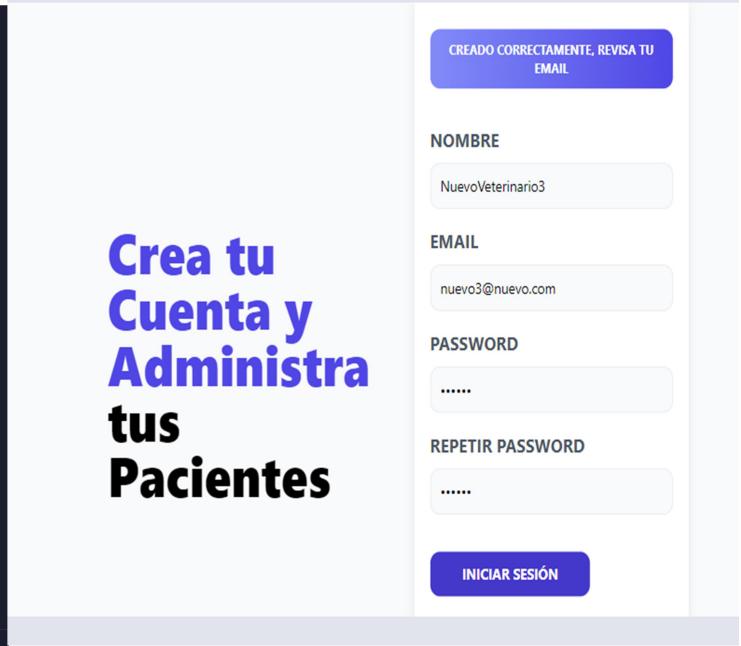
#### ▼ TEST BODY

```
1 visit http://localhost:5173/
2 get :nth-child(1) > .border
3 -type manuel@manuel.com
4 get :nth-child(2) > .border
5 -type 123456
6 get .bg-indigo-700
7 -click
(xhr) POST 200 http://localhost:4000/api/veterinarios/login
(new url) http://localhost:5173/admin
(xhr) GET 200 http://localhost:4000/api/pacientes
```

## 2.2.2. Registrar Veterinario

Se hace el ingreso para registrar un nuevo Veterinario y además se verifica que salgan los mensajes de éxito.

```
1 visit http://localhost:5173/
2 get [href="/registrar"]
3 -click
4 (new url) http://localhost:5173/registrar
5 get :nth-child(1) > .border
6 -type NuevoVeterinario3
7 get :nth-child(2) > .border
8 -type nuevo3@nuevo.com
9 get :nth-child(3) > .border
10 -type 123456
11 get :nth-child(4) > .border
12 -type 123456
13 get .bg-indigo-700
14 -click
15 (xhr) POST 200 http://localhost:4000/api/veterinarios
16 get .from-indigo-400
17 invoke .text()
18 -assert expected Creado Correctamente, revisa tu email to equal **Creado Correctamente,
revisa tu email**
19 get .from-indigo-400
20 -assert expected <div.from-indigo-400.to-indigo-600.bg-gradient-to-r.text-center.p-
3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-indigo-600
```



Confirmar Veterinario: Este proceso se logra a través de un token que se envía por medio del correo, por lo que es necesario ingresar al enlace mediante el token que se mandó.

El usuario no está confirmado por lo que el testing es capaz de validar eso.

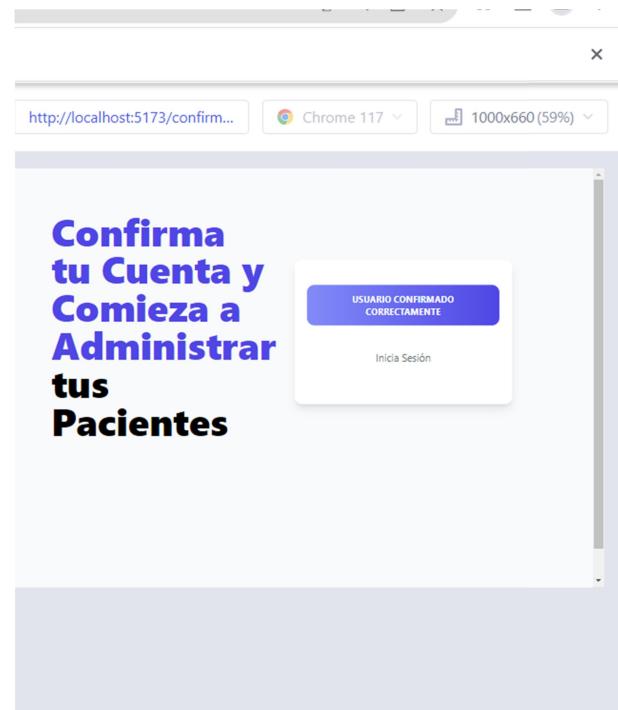
```
_id: ObjectId('6520a5e1b63e6119ef18f935')
nombre: "NuevoVeterinario3"
password: "$2b$10$xNJhNd9BJx/iYRmMEW/8LeH.jG22GlQGpihuSUnUdxUmvPzp987Rq"
email: "nuevo3@nuevo.com"
telefono: null
web: null
token: "1hc3chcc6j4hvehvdlko"
confirmado: false
__v: 0
```

Agregamos el token:

```
1 //> <reference types="cypress"/>
2
3 const token = '1hc3chcc6j4hvehvd1ko';
4
5 describe("Administrador de Citas", () => {
6     it("Inicio de sesion", () => {
7         cy.visit(`http://localhost:5173/confirmar/${token}`);
8
9         cy.get('.from-indigo-400').invoke("text").should("equal", "Usuario confirmado Correctamente");
10        cy.get('.from-indigo-400').should("have.class", "to-indigo-600");
11
12        cy.wait(5000)
13
14        cy.get('.block').click()
15
16    });
17});
```

✓ Inicio de sesion

- TEST BODY
  - 1 visit http://localhost:5173/confirmar/1hc3chcc6j4hvehvd1ko
  - 2 get .from-indigo-400
    - (xhr) GET 200 http://localhost:4000/api/v1/confirmar/1hc3chcc6j4hvehvd1ko
  - 3 invoke .text()
  - 4 - assert expected Usuario confirmado Correctamente\*\*  
Correctamente\*\*
  - 5 get .from-indigo-400
    - assert expected <div.from-indigo-400.to-indigo-600.rounded-xl.uppercase.text-white.font-bold.t
  - 6 - assert expected <div.from-indigo-400.to-indigo-600.rounded-xl.uppercase.text-white.font-bold.t
  - 7 wait 5000
  - 8 get .block
    - click
  - 9 (new url) http://localhost:5173/



### 2.2.3. Olvide mi contraseña:

```
✓ Administrador de Citas
  ✓ Olvide Contraseña
    TEST BODY
    1 visit http://localhost:5173/
    2 get [href="/olvide-password"]
    3 -click
      (new url) http://localhost:5173/olvide-password
    4 get :nth-child(1) > .border
    5 -type cambio@cambio.com
    6 get .bg-indigo-700
    7 -click
    8 get .from-indigo-400
      (xhr) POST 200 http://localhost:4000/api/veterinarios/olvide-password
    9 invoke .text()
   10 -assert expected Hemos enviado un email con las instrucciones to equal **Hemos enviado un  
email con las instrucciones**
```



```

1  _id: ObjectId('65207a42b63e6119ef18f8b0')
2  [+] nombre: "cambio"
3  password: "$2b$10$1R1joikMiML0Ef0XztVZ1OU/4b8RU6/3o4e4VhlRfRpFHp9Gfezce"
4  email: "cambio@cambio.com"
5  telefono: null
6  web: null
7  [+] token: '1hc3okaciqrmpcopdbg'
8  confirmado: false
9  __v: 0

```

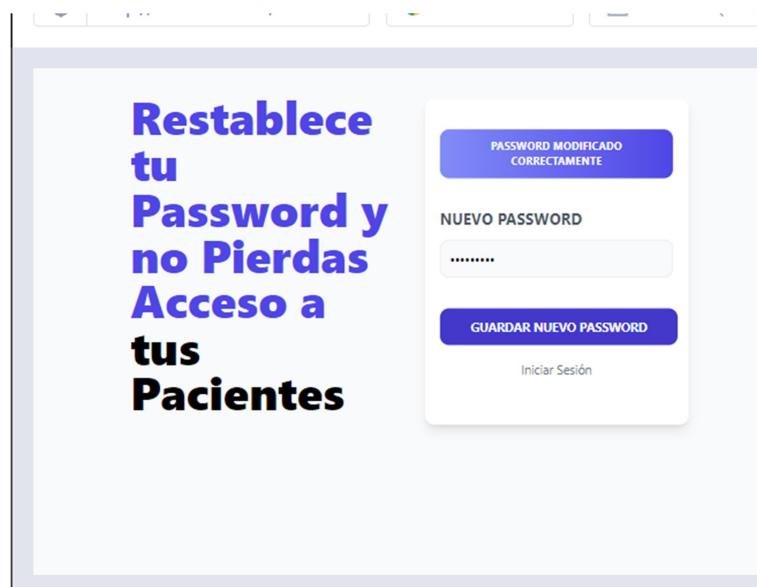
4d. ObjectId('65207a42b63e6119ef18f8b0')

## 2.2.4. Confirmar nueva contraseña

```

MERN_APV_fronend > cypress > e2e > Login > ▲ confirmarNuevaContraseña.cy.js > [x] token
1  /// <reference types="cypress"/>
2  [!]
3  const token = '1hc3okaciqrmpcopdbg';
4
5  describe("Administrador de Citas", () => {
6    it("Inicio de sesion", () => {
7      cy.visit(`http://localhost:5173/olvide-password/${token}`);
8
9      cy.get('.from-indigo-400').invoke("text").should("equal", "Coloca tu Nuevo Password");
10     cy.get('.from-indigo-400').should("have.class", "to-indigo-600");
11
12     cy.wait(5000)

```



## 2.2.5. Testing de errores:

Confirmar nueva contraseña:

Cypress esta validando el texto y el color de los mensajes de error.



```
3 invoke .text()
4 - assert expected Coloca tu Nuevo Password to equal **Coloca tu Nuevo Password**
5 get .from-indigo-400
6 - assert expected <div.from-indigo-400.to-indigo-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-indigo-600
7 wait 5000
8 get .border
9 -type 123
10 get .bg-indigo-700
11 -click
12 get .from-red-400
13 invoke .text()
14 - assert expected El Password debe ser mínimo de 6 caracteres to equal **El Password
debe ser mínimo de 6 caracteres**
15 get .from-red-400
```

Error  
inicio  
sesión:

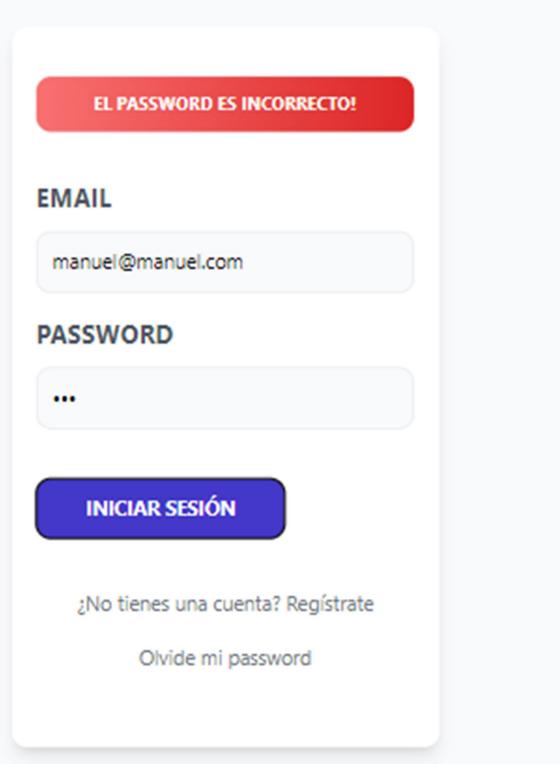
```
1 visit http://localhost:5173/
2 get :nth-child(2) > .border
3 -type 123456
4 get .bg-indigo-700
5 -click
6 get .from-red-400
7 invoke .text()
8 -[assert] expected Todos los Campos son Obligatorios to equal **Todos los Campos son Obligatorios**
9 get .from-red-400
10 -[assert] expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
```





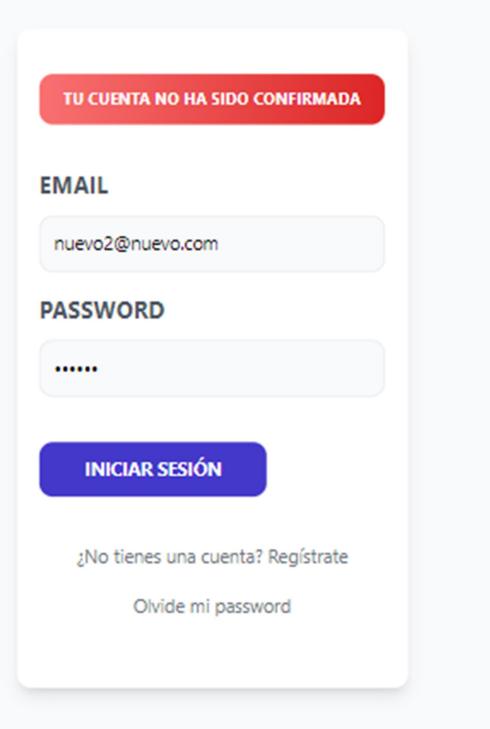
```
xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
wait 10000
get :nth-child(1) > .border
-type chayanne@chayanne.com
get :nth-child(2) > .border
-type 123456
get .bg-indigo-700
-click
get .from-red-400
invoke .text()
-assert expected El usuario no existe to equal **El usuario no existe**
(xhr) ● POST 404 http://localhost:4000/api/veterinarios/login
get .from-red-400
-assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
```

# Inicia Sesión Y Administra tus Pacientes



```
21 get .from-red-400
22 -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-
xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
23 wait 10000
24 get :nth-child(1) > .border
25 -clear
26 get :nth-child(2) > .border
27 -clear
28 get :nth-child(1) > .border
29 -type manuel@manuel.com
30 get :nth-child(2) > .border
31 -type 123
32 get .bg-indigo-700
33 -click
34 get .from-red-400
35 invoke .text()
36 -assert expected El password es incorrecto! to equal **El password es incorrecto!**
(xhr) POST 403 http://localhost:4000/api/veterinarios/login
37 get .from-red-400
38 -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-
```

# Inicia Sesión Y Administra tus Pacientes



```
38 -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
39 wait 10000
40 get :nth-child(1) > .border
41 -clear
42 get :nth-child(2) > .border
43 -clear
44 get :nth-child(1) > .border
45 -type nuevo2@nuevo.com
46 get :nth-child(2) > .border
47 -type 123456
48 get .bg-indigo-700
49 -click
50 get .from-red-400
51 invoke .text()
52 -assert expected Tu Cuenta no ha sido confirmada to equal **Tu Cuenta no ha sido
      confirmada**
      (xhr) POST 403 http://localhost:4000/api/veterinarios/login
53 get .from-red-400
54 -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
```

## 2.2.6. Errores Registro de Veterinarios:

Crea tu Cuenta y Administra tus Pacientes

USUARIO YA REGISTRADO

NOMBRE  
NuevoVeterinario3

EMAIL  
nuevo3@nuevo.com

PASSWORD  
.....

REPETIR PASSWORD  
.....

INICIAR SESIÓN

Vs tiene una cuenta? Inicia Sesión

```
1 visit http://localhost:5173/
2 get [href="/registrar"]
3 -click
(new url) http://localhost:5173/registrar
4 get :nth-child(1) > .border
5 -type NuevoVeterinario3
6 get :nth-child(2) > .border
7 -type nuevo3@nuevo.com
8 get :nth-child(3) > .border
9 -type 123456
10 get :nth-child(4) > .border
11 -type 123456
12 get .bg-indigo-700
13 -click
(xhr) POST 400 http://localhost:4000/api/veterinarios
14 get .from-red-400
15 invoke .text()
16 -assert expected Usuario ya registrado to equal **Usuario ya registrado**
17 get .from-red-400
18 -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
```



```
- assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
wait 10000
get :nth-child(1) > .border
-clear
get :nth-child(2) > .border
-clear
get .bg-indigo-700
-click
get .from-red-400
invoke .text()
assert expected Hay campos vacios to equal **Hay campos vacios**
get .from-red-400
assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to have class to-red-600
wait 10000
```



```
40 get :nth-child(5) > .border
41 -type 12345
42 get :nth-child(4) > .border
43 -type 12345
44 get .bg-indigo-700
45 -click
46 get .from-red-400
47 invoke .text()
48 -assert expected El password es muy corto, agrega minimo 6
  caracteres to equal **El password es muy corto, agrega minimo 6
  caracteres**
49 get .from-red-400
50 -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-
  center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to
  have class to-red-600
```

# Crea tu Cuenta y Administra tus Pacientes

LOS PASSWORD NO SON IGUALES

NOMBRE

NuevoVeterinario32

EMAIL

nuevo32@nuevo.com

PASSWORD

.....

REPETIR PASSWORD

.....

```
69  -click
70  get .from-red-400
71  invoke .text()
72  -assert expected Los Password no son iguales to equal **Los
    Password no son iguales**
73  get .from-red-400
74  -assert expected <div.from-red-400.to-red-600.bg-gradient-to-r.text-
    center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to
    have class to-red-600
```

Error Olvide mi contraseña:



```
1 visit http://localhost:5173/
2 get [href="/olvide-password"]
3 -click
  (new url) http://localhost:5173/olvide-password
4 get .bg-indigo-700
5 -click
6 wait 5000
7 get .from-red-400
8 invoke .text()
9 -assert expected El Email es obligatorio to equal **El Email es
obligatorio**
10 get :nth-child(1) > .border
11 -type cambio@cambi.com
12 get .bg-indigo-700
13 -click
14 get .from-red-400
15 invoke .text()
16 -assert expected El usuario no existe to equal **El usuario no
existe**
```

**Restablece  
tu  
Password y  
no Pierdas  
Acceso a  
tus  
Pacientes**

HUBO UN ERROR CON EL ENLACE

**Restablece  
tu  
Password y  
no Pierdas  
Acceso a  
tus  
Pacientes**

EL PASSWORD DEBE SER MÍNIMO DE 6  
CARACTERES

NUEVO PASSWORD

Password

GUARDAR NUEVO PASSWORD

```

1 visit http://localhost:5173/olvide-
password/1hc3qbbrh1tdib3ke7eg
2 get .bg-indigo-700
(xhr) GET 200
http://localhost:4000/api/veterinarios/olvide-
password/1hc3qbbrh1tdib3ke7eg
3 -click
4 get .from-red-400
5 invoke .text()
6 -assert expected El Password debe ser mínimo de 6
caracteres to equal **El Password debe ser mínimo de 6
caracteres**
7 wait 5000

```

## Pacientes

### Registrar

Añade tus pacientes [Administrados](#)

Administra tus [Pacientes y Citas](#)

<b>NOMBRE MASCOTA</b>	Cerberus
<b>NOMBRE PROPIETARIO</b>	Fitzwilliam Darcy
<b>EMAIL PROPIETARIO</b>	orgulloYprejuicio@
<b>FECHA ALTA</b>	8 de septiembre de 2023
<b>SÍNTOMAS</b>	Lele pancha jajaja

[EDITAR](#)

```
1 visit http://localhost:5173/
2 get :nth-child(1) > .border
3 -type manuel@manuel.com
4 get :nth-child(2) > .border
5 -type 123456
6 get .bg-indigo-700
7 -click
8 get #nombre
  (xhr) POST 200
  http://localhost:4000/api/veterinarios/login
  (new url) http://localhost:5173/admin
  (xhr) GET 200 http://localhost:4000/api/pacientes
9 -type Cerberus
10 get #propietario
11 -type Fitzwilliam Darcy
12 get #email
13 -type orgulloYprejuicio@gmail.com
14 get #fecha
15 -type 2023-09-09
16 get #sintomas
17 -type Lele pancha jajaja
18 get .bg-white > .bg-indigo-600
19 -click
  (xhr) POST 200 http://localhost:4000/api/pacientes
```

## Editar:

**NOMBRE MASCOTA**  
Cerberus

**NOMBRE PROPIETARIO**  
Fitzwilliam Darcy

**EMAIL PROPIETARIO**  
orgulloYprejuicio@gmail.com

**FECHA ALTA**  
dd/mm/aaaa

**SÍNTOMAS**  
Lele pancha jajaja

**GUARDAR CAMBIOS**

**NOMBRE:**  
Cerberus

**PROPIETARIO:**  
Fitzwilliam  
Darcy

**EMAIL:**  
orgulloYprejuicio@

**FECHA DE ALTA:** 8 de septiembre de 2023

**SÍNTOMAS:**  
Lele pancha  
jajaja

```
EditarPaciente.cy.js
00:23

15 -clear
16 get '#email'
17 -clear
18 get '#fecha'
19 -clear
20 get '#sintomas'
21 -clear
22 wait 5000
23 get '#nombre'
24 -type ZEUS
25 get '#propietario'
26 -type Darcy
27 get '#email'
28 -type orgullo@gmail.com
29 get '#fecha'
30 -type 2023-09-09
31 get '#sintomas'
32 -type Lele pancha jejeje
33 wait 5000
34 get '.bg-white > .bg-indigo-600'
35 -click
(xhr) PUT 200 http://localhost:4000/api/pacientes/6520ade2b63e6119ef18f95f
36 get '.from-indigo-400'
37 -assert expected <div.from-indigo-400.to-indigo-600.bg-gradient-to-r.text-center.p-3.rounded-xl.uppercase.text-white.font-bold.text-sm.mb-10> to be visible
```

## Eliminar:

The screenshot shows a web application interface for managing patients. On the left, there's a main form titled "Pacientes" with fields for "NOMBRE MASCOTA", "NOMBRE PROPIETARIO", "EMAIL PROPIETARIO", "FECHA ALTA", and "SÍNTOMAS". On the right, a modal dialog displays the patient details for deletion: "NOMBRE: Firulais", "PROPIETARIO: Claudia", "EMAIL: mordonezsilva@gmail.com", "FECHA DE ALTA: 12 de octubre de 2023", "SÍNTOMAS: LELE PANCHA", and an "EDITAR" button.

NOMBRE:	Firulais
PROPIETARIO:	Claudia
EMAIL:	mordonezsilva@gmail.com
FECHA DE ALTA:	12 de octubre de 2023
SÍNTOMAS:	LELE PANCHA

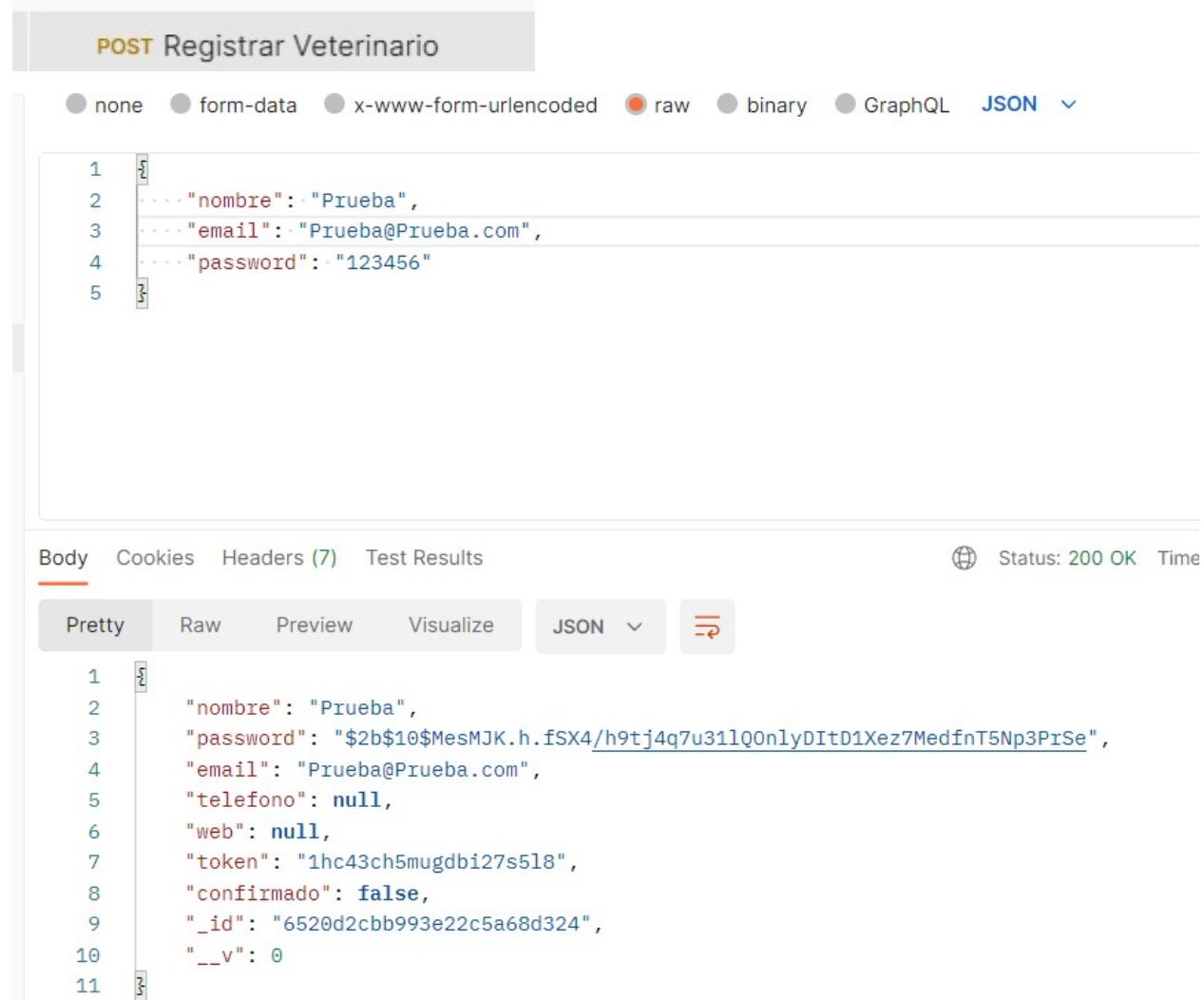
```
TEST BODY
1  visit http://localhost:5173
2  get :nth-child(1) > .border
3  -type manuel@manuel.com
4  get :nth-child(2) > .border
5  -type 123456
6  get .bg-indigo-700
7  -click
(xhr)  POST 200 http://localhost:4000/api/veterinarios/login
8  wait 5000
(new url) http://localhost:5173/admin
(xhr)  GET 200 http://localhost:4000/api/pacientes
9  get :nth-child(4) > .flex > .bg-red-600
10 -click
(confirm) ¿Confirmas que deseas eliminar?
(xhr)  DELETE 200 http://localhost:4000/api/pacientes/6520ade2b63e6119ef18f95f
11  wait 5000
```

### 2.3. Pruebas Unitarias:

▼ FullStack JS	
	<b>POST</b> Registrar Veterinario
	<b>GET</b> Confirma una cuenta via Token
	...
	<b>POST</b> Autenticar Usuario
	<b>GET</b> Obtener Perfil
	<b>POST</b> Resetear Email - Comprobar si usuario existe y gene...
	<b>GET</b> Valida el Token cuando el usuario cambia su password
	<b>POST</b> Almacena el nuevo password
	<b>POST</b> Crea un nuevo paciente
	<b>GET</b> Obtiene los pacientes de un veterinario
	<b>GET</b> Obtiene un Paciente
	<b>PUT</b> Actualizar paciente
	<b>DEL</b> Elimina un paciente

### 2.3.1. Registrar usuario veterinario:

- Se le envía datos al api de registrar
- Se guarda los datos correctamente en la base de datos



POST Registrar Veterinario

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "nombre": "Prueba",  
3   "email": "Prueba@Prueba.com",  
4   "password": "123456"  
5 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time

Pretty Raw Preview Visualize JSON

```
1 {  
2   "nombre": "Prueba",  
3   "password": "$2b$10$MesMJK.h.fSX4/h9tj4q7u31lQOnlyDItD1Xez7MedfnT5Np3PrSe",  
4   "email": "Prueba@Prueba.com",  
5   "telefono": null,  
6   "web": null,  
7   "token": "1hc43ch5mugdbi27s5l8",  
8   "confirmado": false,  
9   "_id": "6520d2cbb993e22c5a68d324",  
10  "__v": 0  
11 }
```

### 2.3.2. Autenticación por token:

- Se le envía el token por URL al api para validar.
- Despliega un mensaje de “Usuario confirmado Correctamente”.

The screenshot shows a Postman request configuration and its resulting JSON response. The request is a GET to `{{API_URL}}/veterinarios/confirmar/1hc43ch5mugdbi27s5l8`. The response body is a JSON object with a single key-value pair: "msg": "Usuario confirmado Correctamente".

**POST Registrar Veterinario**

**GET Confirma una cuenta via Token**

**GET** `{{API_URL}}/veterinarios/confirmar/1hc43ch5mugdbi27s5l8`

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {  
2   "msg": "Usuario confirmado Correctamente"  
3 }
```

### 2.3.3. Autenticar usuario

- Se envía una confirmación tipo booleana.

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** {{API\_URL}}/veterinarios/login
- Body (JSON):**

```
1 { "email": "Prueba@Prueba.com",  
2   "password": "123456"}  
3  
4
```
- Response Status:** 200 OK
- Response Time:** 137 ms
- Response Size:** 499 B
- Response Body (Pretty JSON):**

```
1 {  
2   "_id": "6520d2cbb993e22c5a68d324",  
3   "nombre": "Prueba",  
4   "email": "Prueba@Prueba.com",  
5   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
6       eyJpZCI6IjY1MjBkMmNiYjk5M2UyMmM1YTY4ZDMyNCIsImhdCI6MTY5NjY1MDEwMCwiZXhwIjoxNjk5MjQyMTAwfQ.  
       kNaKFKOYYcnGuvNAnmdy-MGBBMGoH-HoT3u13KMZCK4"
```

### 2.3.4. Olvidar contraseña

- Se envía el correo electrónico del usuario al api para cambiar la contraseña.
- Despliega el mensaje “Hemos enviado un email con las instrucciones”.

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** {{API\_URL}}/veterinarios/resetearEmail
- Body (JSON):**

```
1 { "email": "Prueba@Prueba.com"}  
2  
3
```

```
1  {
2    "email": "Prueba@Prueba.com"
3 }
```

Body Cookies Headers (7) Test Results

Pretty

Raw

Preview

Visualize

JSON ▾



```
1  {
2    "msg": "Hemos enviado un email con las instrucciones"
3 }
```

- Luego de seguir las instrucciones enviadas al correo electrónico proporcionado se confirma el acceso al cambio de contraseña.
- El mensaje que despliegue es “Tokén valido y el usuario existente”

**GET** Valida el Token cuando el usuario cambia su password

GET

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

This request does not have a body

Body Cookies Headers (7) Test Results Status: 200 OK Time: 1s

Pretty Raw Preview Visualize JSON 

```
1 {  
2   "msg": "Token valido y el usuario existe"  
3 }
```

**POST** Almacena el nuevo password

The screenshot shows the Postman application interface. At the top, there is a header bar with the method **POST**, a dropdown menu, and the URL **`{{API_URL}}`/veterinarios/olvide-password/1hc4498qpbd6ig9mor1**. Below the header, there are tabs for **Params**, **Authorization**, **Headers (8)**, **Body** (which is selected), **Pre-request Script**, **Tests**, and **Settings**. Under the **Body** tab, there are options for **none**, **form-data**, **x-www-form-urlencoded**, **raw** (selected), **binary**, **GraphQL**, and **JSON**. The **raw** section contains the following JSON payload:

```
1 {  
2   "password": "123456"  
3 }
```

Below the body, there are tabs for **Body**, **Cookies**, **Headers (7)**, and **Test Results**. The **Test Results** tab is selected, showing the response status, time, and size. The status is **200 OK**, time is **221 ms**, and size is **278 B**. The response body is displayed in **Pretty** format:

```
1 {  
2   "msg": "Password modificado correctamente"  
3 }
```

### 2.3.5. Crear un nuevo paciente

Se le envía los datos del paciente para agregar a la api.

Despliega un mensaje: “Se agrego correctamente el paciente”

The screenshot shows a Postman interface for creating a new patient. The title bar says "POST Crea un nuevo paciente". The URL is {{API\_URL}}/pacientes. The method is POST. The body is set to raw JSON with the following data:

```
1 {
2   "nombre": "EQUISDE",
3   "propietario": "Manuel",
4   "email": "correo@correo",
5   "sintomas": "Le cuesta caminar"
6 }
```

The response status is 200 OK. The response body is:

```
1 {
2   "nombre": "EQUISDE",
3   "propietario": "Manuel",
4   "email": "correo@correo",
5   "fecha": "2023-10-07T03:51:59.536Z",
6   "sintomas": "Le cuesta caminar",
7   "_id": "6520d6d5decfc72f81cbdd1c",
8   "veterinario": "6520d2cbb993e22c5a68d324",
9   "createdAt": "2023-10-07T03:56:05.044Z",
10  "updatedAt": "2023-10-07T03:56:05.044Z",
11  "__v": 0
12 }
```

### 2.3.6. Modificación de paciente

Se le envía un id al api para buscar el paciente a buscar.

**GET** Obtiene un Paciente

GET [{{API\\_URL}}/pacientes/6520d6d5decfc72f81cbdd1c](#)

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Tok... Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [authorization ↗](#)

Token eyJhbGciOiJIUzI1N

Body Cookies Headers (7) Test Results Status: 200 OK Time

Pretty Raw Preview Visualize JSON Copy

```
1 {  
2   "_id": "6520d6d5decfc72f81cbdd1c",  
3   "nombre": "EQUISDE",  
4   "propietario": "Manuel",  
5   "email": "correo@correo",  
6   "fecha": "2023-10-07T03:51:59.536Z",  
7   "sintomas": "Le cuesta caminar",  
8   "veterinario": "6520d2cbb993e22c5a68d324",  
9   "createdAt": "2023-10-07T03:56:05.044Z",  
10  "updatedAt": "2023-10-07T03:56:05.044Z",  
11  "__v": 0  
12 }
```

- Luego de traer los datos del paciente se hace el cambio y luego los mando al api para confirma el cambio

**PUT** Actualizar paciente

none form-data X-www-form-urlencoded IdW Binary GraphQL JSON

```

1  {
2    "nombre": "Glufis actualizada mi perrita"
3  }

```

Body Cookies Headers (7) Test Results Status: 2

Pretty Raw Preview Visualize JSON

```

1  {
2    "_id": "6520d6d5decfc72f81cbdd1c",
3    "nombre": "Glufis actualizada mi perrita",
4    "propietario": "Manuel",
5    "email": "correo@correo",
6    "fecha": "2023-10-07T03:51:59.536Z",
7    "sintomas": "Le cuesta caminar",
8    "veterinario": "6520d2cbb993e22c5a68d324",
9    "createdAt": "2023-10-07T03:56:05.044Z",
10   "updatedAt": "2023-10-07T04:00:59.648Z",
11   "__v": 0
12 }

```

### 2.3.7. Eliminar paciente

- Se le envía el id del paciente a eliminar al api.
- Despliega un mensaje “Paciente eliminado”.

The screenshot shows a Postman collection named "FullStack JS / Elimina un paciente". The request is a **DELETE** to `{{API_URL}}/pacientes/6520d6d5decfc72f81cbdd1c`. The Authorization tab is selected, showing a Bearer Token type. A note says: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)". The token value is partially visible as `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...`. The response status is 200 OK with a size of 263 B.

```
1   "msg": "Paciente eliminado"
```

## **Conclusiones**

Las pruebas de software son un componente vital del aseguramiento de la calidad, y su importancia no puede ser subestimada. Al aplicar pruebas sistemáticas y efectivas, las organizaciones de desarrollo pueden lograr los siguientes beneficios:

- Detección temprana de defectos: Las pruebas permiten identificar y corregir problemas en las etapas iniciales del ciclo de desarrollo, lo que reduce significativamente los costos y los riesgos asociados con la corrección de errores más adelante.
- Mejora de la confiabilidad: Las pruebas rigurosas garantizan que el software funcione como se espera, lo que aumenta la confiabilidad y la satisfacción del usuario final.
- Cumplimiento de requisitos: Las pruebas funcionales aseguran que el software cumpla con los requisitos y especificaciones establecidos, evitando desviaciones no deseadas.
- Optimización del rendimiento: Las pruebas de rendimiento y carga ayudan a identificar cuellos de botella y problemas de escalabilidad, lo que garantiza que el software funcione de manera eficiente incluso bajo cargas pesadas.
- Aumento de la seguridad: Las pruebas de seguridad permiten identificar vulnerabilidades y riesgos de seguridad, lo que protege al software y a los datos de posibles amenazas.

En última instancia, las pruebas de software son una inversión esencial para garantizar que los productos y sistemas cumplan con los estándares de calidad requeridos en un entorno cada vez más competitivo. Al incorporar pruebas de manera integral en el ciclo de desarrollo, las organizaciones pueden ofrecer software confiable y de alto rendimiento que satisfaga las expectativas de los usuarios y promueva el éxito a largo plazo de sus proyectos.

## **Recomendaciones**

La calidad del testing de software es un componente esencial en el desarrollo de aplicaciones exitosas y confiables. Para lograr esta calidad, es fundamental seguir buenas prácticas en diferentes tipos, clases y procesos de pruebas. A continuación, se presenta un análisis que destaca algunas de estas buenas prácticas:

### **1. Buenas Prácticas en Tipos de Pruebas:**

- **Pruebas Funcionales:** Las pruebas funcionales verifican que las funciones y características del software funcionen correctamente según lo especificado en los requisitos. Una buena práctica es diseñar casos de prueba exhaustivos que cubran todas las funcionalidades y escenarios posibles.
- **Pruebas de Rendimiento:** Las pruebas de rendimiento evalúan cómo el software se comporta bajo diferentes cargas y condiciones. Para lograr una calidad óptima, es esencial simular condiciones del mundo real y definir objetivos de rendimiento medibles.
- **Pruebas de Seguridad:** En las pruebas de seguridad, es crucial utilizar prácticas de prueba de penetración y buscar vulnerabilidades. Se deben abordar todos los vectores de ataque posibles y aplicar parches de seguridad adecuados.

### **2. Buenas Prácticas en Clases de Pruebas:**

- **Pruebas de Aceptación del Usuario:** La participación activa de los usuarios finales en las pruebas de aceptación es una buena práctica. Esto garantiza que el software cumpla con sus expectativas y requisitos específicos.
- **Pruebas de Regresión:** Automatizar las pruebas de regresión es una práctica eficaz para garantizar que las modificaciones no introduzcan nuevos errores en las funciones existentes. Esto ahorra tiempo y recursos a largo plazo.
- **Pruebas de Usabilidad:** Realizar pruebas de usabilidad con usuarios reales puede ayudar a identificar problemas de experiencia del usuario y mejorar la interfaz de usuario para una mejor satisfacción.

### **3. Buenas Prácticas en Procesos de Pruebas:**

- **Planificación de Pruebas:** Una buena planificación es esencial. Definir un alcance claro, establecer objetivos medibles y documentar casos de prueba detallados son pasos críticos.
- **Gestión de Defectos:** Mantener un sistema eficaz de gestión de defectos es esencial para rastrear y priorizar problemas. Esto ayuda a garantizar que se aborden los problemas críticos primero.
- **Automatización de Pruebas:** La automatización de pruebas puede acelerar significativamente el proceso de pruebas. Sin embargo, es fundamental seleccionar las pruebas adecuadas para la automatización y mantener los scripts actualizados.
- **Monitoreo Continuo:** Una vez que el software está en producción, el monitoreo continuo y las pruebas de regresión automatizadas pueden ayudar a detectar problemas en tiempo real y garantizar que el software siga siendo confiable.

## **Glosario**

**Alcance de Pruebas:** Los límites y objetivos específicos que definen qué se evaluará en las pruebas de software.

**API:** son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos.

**Automatización de Pruebas:** El uso de herramientas y scripts para ejecutar pruebas de software de manera automática.

**Calidad de Software:** La medida en que el software cumple con requisitos y expectativas, y está libre de defectos.

**Casos de Prueba:** Escenarios específicos diseñados para evaluar el software durante las pruebas.

**Compatibilidad:** La capacidad del software para funcionar correctamente en diferentes plataformas y dispositivos.

**Defecto:** Un error o problema en el software que causa un comportamiento no deseado.

**Ejecución de Pruebas:** El proceso de realizar pruebas para evaluar el software.

**Escalabilidad:** La capacidad del software para adaptarse a un aumento en la carga o usuarios sin degradación del rendimiento.

**Funcionalidad:** Las capacidades y características que ofrece el software a los usuarios.

**Gestión de Defectos:** El proceso de identificación, seguimiento y priorización de problemas encontrados durante las pruebas.

**Interfaz de Usuario:** El punto de interacción entre el usuario y el software.

**Mantenibilidad:** La facilidad con la que se pueden realizar cambios y actualizaciones en el software.

**Monitoreo Continuo:** La supervisión constante del software en producción para detectar problemas en tiempo real.

**Planificación de Pruebas:** La preparación y definición de las estrategias y objetivos para las pruebas de software.

**Pruebas de Aceptación del Usuario (UAT):** Las pruebas realizadas por usuarios finales para verificar si el software cumple con sus requisitos.

**Pruebas de Carga:** Las pruebas que evalúan cómo el software se comporta bajo cargas pesadas.

**Pruebas de Funcionalidad:** Las pruebas que verifican que las funciones y características del software funcionen según lo previsto.

**Pruebas de Integración:** Las pruebas que evalúan la interoperabilidad entre módulos o componentes del software.

**Pruebas de Rendimiento:** Las pruebas que evalúan la velocidad y eficiencia del software bajo diferentes condiciones.

**Pruebas de Regresión:** Las pruebas que verifican que las modificaciones no afecten negativamente las funciones existentes.

**Pruebas de Seguridad:** Las pruebas que buscan vulnerabilidades y riesgos de seguridad en el software.

**Pruebas de Usabilidad:** Las pruebas que evalúan la facilidad de uso y experiencia del usuario.

**Requisitos de Software:** Las especificaciones y condiciones que el software debe cumplir.

**Rendimiento:** La capacidad del software para funcionar de manera eficiente y responder rápidamente.

**Revisión de Código:** La inspección y análisis del código fuente para detectar errores y mejoras.

**Validación:** La confirmación de que el software cumple con los requisitos especificados.

**Verificación:** La confirmación de que el software se ajusta a las especificaciones y está libre de defectos.

**Vulnerabilidad:** Una debilidad o brecha en el software que puede ser explotada por amenazas de seguridad.

**UAT (User Acceptance Testing):** Siglas de "Pruebas de Aceptación del Usuario", que son pruebas realizadas por usuarios finales.

**UI (User Interface):** Siglas de "Interfaz de Usuario", que es la parte del software con la que interactúa el usuario.

## **Anexos**

### **Anexo A.**

#### Pruebas de stress realizadas

La prueba se realizó de forma secuencial en un servidor con 512 MB de RAM, el número de usuarios fue subiendo y en el ultimo minuto no se envia ninguna solicitud de conexión para probar la recuperación del sistema. La aplicación web fue llevada a condiciones extremas para revisar su comportamiento, el mismo fue el esperado, el sistema funcionó de forma lenta, sin embargo sus capacidades sobrepasan al requerimiento del cliente, ya que ellos no tendrán tantas peticiones como las pruebas realizadas.

La capacidad maxima de usuarios conectados es de 5000 sin embargo para este requerimiento únicamente se necesitan 4 usuarios conectados.

Se evaluó que el sistema se recupera en 1 minuto después de llegar a su capacidad maxima.

El tiempo de respuesta promedio para la página es de 2.17 ms, para este caso no teníamos un requerimiento exacto de tiempo de respuesta sin embargo es un tiempo considerable bueno para el usuario.

En general, la aplicación web cumplió con los requisitos de rendimiento y disponibilidad. Sin embargo, se identificaron algunas áreas de mejora, como la reducción del tiempo de respuesta para la página de inicio y la implementación de medidas de mitigación contra ataques de denegación de servicio.

```

import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
  stages: [
    { duration: '1m', target: 1 }, // below normal load
    { duration: '1m', target: 100 },
    { duration: '1m', target: 200 }, // normal load
    { duration: '1m', target: 200 },
    { duration: '1m', target: 300 }, // around the breaking point
    { duration: '1m', target: 300 },
    { duration: '1m', target: 400 }, // beyond the breaking point
    { duration: '1m', target: 400 },
    { duration: '1m', target: 0 }, // scale down. Recovery stage.
  ],
};

export default function () {
  let res = http.get('https://cliquant-caramel-c3df51.netlify.app/');
  check(res, { '304 Not Modified': (r) => r.status == 304 });

  // Pausa de 1 segundo entre cada solicitud
  sleep(1);
}

execution: local
script: stress2.js
output: -

scenarios: (100.00%) 1 scenario, 200 max VUs, 4m30s max duration (incl. graceful stop):
  * default: Up to 200 looping VUs for 4m0s over 4 stages (gracefulRampDown: 30s, gracefulStop: 30s)

WARN[0259] Request Failed error="Get \"https://cliquant-caramel-c3df51.netlify.app/\": dial tcp 44.217.161.11:443: connectex: Se produjo un error en la conexión establecida ya que el host conectado no ha podido responder."
ERROR[0259] ReferenceError: check is not defined
running at file:///C:/Users/EquipoSSD/Downloads/stress2.js:21:8(12) executor=ramping-vus scenario=default source=stacktrace
WARN[0259] Request Failed error="Get \"https://cliquant-caramel-c3df51.netlify.app/\": dial tcp 54.161.234.33:443: connectex: Se produjo un error en la conexión establecida ya que el host conectado no ha podido responder."
ERROR[0259] ReferenceError: check is not defined
running at file:///C:/Users/EquipoSSD/Downloads/stress2.js:21:8(12) executor=ramping-vus scenario=default source=stacktrace

data_received.....: 1.0 MB 4.1 kB/s
data_sent.....: 84 kB 324 B/s
http_req_blocked.....: avg=2.95ms min=0s med=0s max=327.88ms p(90)=0s p(95)=0s
http_req_connecting.....: avg=1.35ms min=0s med=0s max=130.98ms p(90)=0s p(95)=0s
http_req_duration.....: avg=391.63ms min=0s med=65.71ms max=21.72s p(90)=106.02ms p(95)=111.1ms
  { expected_response:true }.....: avg=77.59ms min=51.75ms med=70.59ms max=592.05ms p(90)=108.79ms p(95)=111.85ms
http_req_failed.....: 45.04% @ 1210 @ 1476
http_req_receiving.....: avg=671.72us min=0s med=0s max=15.74ms p(90)=1.08ms p(95)=5.5ms
http_req_sending.....: avg=69.34μs min=0s med=0s max=10.08ms p(90)=0s p(95)=512.29μs
http_req_tls_handshaking.....: avg=1.56ms min=0s med=0s max=169.66ms p(90)=0s p(95)=0s
http_req_waiting.....: avg=391.09ms min=0s med=62.45ms max=21.72s p(90)=104.4ms p(95)=110.01ms
http_reqs.....: 2686 10.37145/s
iteration.duration.....: avg=0.53s min=57.29ms med=111.1ms max=21.74s p(90)=21.06s p(95)=21.07s
iterations.....: 2686 10.37145/s
vus.....: 1 min=1 max=200
vus_max.....: 200 min=200 max=200

```

```

WARN[0140] Request Failed error="Get \"https://clinquant-caramel-c3df51.netlify.app\": dial tcp 35.169.59.174:443: connectex: Se respondió adecuadamente tras un periodo de tiempo, o bien se produjo un error en la conexión establecida ya que el host conectado no ha podido responder."
WARN[0141] Request Failed error="Get \"https://clinquant-caramel-c3df51.netlify.app\": dial tcp 44.217.161.11:443: connectex: Se respondió adecuadamente tras un periodo de tiempo, o bien se produjo un error en la conexión establecida ya que el host conectado no ha podido responder."
  █ status was 200
  █ 76% - █ 1297 / █ 404

checks.....: 76.24% █ 1297      █ 404
data_received.....: 1.1 MB 7.7 kB/s
data_sent.....: 97 KB 686 B/s
http_req_blocked.....: avg=162.6ms min=0s med=0s max=15.26s p(90)=0s p(95)=0s
http_req_connecting.....: avg=158.22ms min=0s med=0s max=15.14s p(90)=0s p(95)=0s
http_req_duration.....: avg=798.45ms min=0s med=69.38ms max=21.65s p(90)=110.97ms p(95)=152.16ms
  { expected response:true }
  avg=79.21ms min=50.81ms med=70.49ms max=416.2ms p(90)=109.03ms p(95)=112.8ms
http_req_failed.....: 23.75% █ 484      █ 1297
http_req_receiving.....: avg=3.56ms min=0s med=825.6μs max=19.86ms p(90)=11.95ms p(95)=13.86ms
http_req_sending.....: avg=58.03μs min=0s med=0s max=922.6μs p(90)=232.6μs p(95)=520μs
http_req_tls_handshaking.....: avg=4.37ms min=0s med=0s max=157.98ms p(90)=0s p(95)=0s
http_req_waiting.....: avg=794.83ms min=0s med=66.59ms max=21.65s p(90)=107.56ms p(95)=151.89ms
http_reqs.....: 1701 12.015987/s
iteration_duration.....: avg=5.96s min=1.05s med=1.08s max=22.67s p(90)=22.04s p(95)=22.05s
iterations.....: 1701 12.015987/s
vus.....: 1 min=1 max=100
vus_max.....: 100 min=100 max=100

running (2m21.6s), 000/100 VUs, 1701 complete and 0 interrupted iterations
default █ [=====] 000/100 VUs 2m0s

respondió adecuadamente tras un periodo de tiempo, o bien se produjo un error en la conexión establecida ya que el host conectado no ha podido responder.

  █ status was 200
  █ 0% - █ 0 / █ 459

checks.....: 0.00% █ 0      █ 459
data_received.....: 0 B 0 B/s
data_sent.....: 0 B 0 B/s
http_req_blocked.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_connecting.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_duration.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_failed.....: 100.00% █ 459      █ 0
http_req_receiving.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_sending.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_reqs.....: 459 3.232932/s
iteration_duration.....: avg=22.05s min=22.02s med=22.05s max=22.12s p(90)=22.06s p(95)=22.07s
iterations.....: 459 3.232932/s
vus.....: 5 min=2 max=100
vus_max.....: 100 min=100 max=100

running (2m22.0s), 000/100 VUs, 459 complete and 0 interrupted iterations
fault █ [=====] 000/100 VUs 2m0s

```

```
C:\Users\EquipoSSD\Downloads>k6 run stress2.js

[ \ \ \ \ \ ] [ \ \ \ \ \ ] .io

execution: local
  script: stress2.js
  output: - 

scenarios: (100.00%) 1 scenario, 1 max VUs, 1m30s max duration (incl. graceful stop):
  * default: Up to 1 looping VUs for 1m0s over 1 stages (gracefulRampDown: 30s, gracefulStop: 30s)

  □ 304 Not Modified
  □ 0% - □ 0 / □ 56

checks.....: 0.00% □ 0      □ 56
data_received.....: 39 kB 650 B/s
data_sent.....: 2.8 kB 46 B/s
http_req_blocked.....: avg=4.55ms min=0s med=0s max=255.07ms p(90)=0s p(95)=0s
http_req_connecting.....: avg=1.13ms min=0s med=0s max=63.57ms p(90)=0s p(95)=0s
http_req_duration.....: avg=72.2ms min=63.35ms med=70.59ms max=91.53ms p(90)=78.76ms p(95)=80.18ms
{ expected_response:true }.....: avg=72.2ms min=63.35ms med=70.59ms max=91.53ms p(90)=78.76ms p(95)=80.18ms
http_req_failed.....: 0.00% □ 0      □ 56
http_req_receiving.....: avg=6.91ms min=0s med=6.33ms max=15.3ms p(90)=11.45ms p(95)=13.68ms
http_req_sending.....: avg=58.94μs min=0s med=0s max=581.5μs p(90)=215.3μs p(95)=535.25μs
http_req_tls_handshaking.....: avg=2.45ms min=0s med=0s max=137.27ms p(90)=0s p(95)=0s
http_req_waiting.....: avg=65.22ms min=62.72ms med=64.38ms max=80.45ms p(90)=65.28ms p(95)=74.61ms
http_reqs.....: 56 0.926065/s
iteration_duration.....: avg=1.07s min=1.06s med=1.07s max=1.33s p(90)=1.08s p(95)=1.08s
iterations.....: 56 0.926065/s
vus.....: 1 min=1 max=1
vus_max.....: 1 min=1 max=1

running (9m20.1s), 000/400 VUs, 6718 complete and 0 interrupted iterations
default □ [=====] 000/400 VUs 9m0s
```

## **Anexo B. Pruebas de aceptacion**

## Anexo C.

## Anexo D. Aplicación Web

## Bibliografía

Morales, V. M. (20 de mayo de 2021). *pragma*. Obtenido de  
<https://www.pragma.co/es/blog/conoce-que-son-las-pruebas-no-funcionales-de-software>

5 tipos de pruebas o testing de software. (2022, septiembre 28). *KeepCoding Bootcamps*.  
<https://keepcoding.io/blog/tipos-de-pruebas-o-testing-de-software/>

Rodgers, N. (2022, diciembre 13). *Diferencias entre testing funcional y no funcional*. Blog de  
Testing y Calidad de Software | Abstracta Chile; Abstracta Chile.  
<https://cl.abstracta.us/blog/diferencias-testing-funcional-no-funcional/>

Singureanu, C. (s/f). *Pruebas no funcionales: proceso, herramientas, tipos y mucho más*.  
Recuperado el 7 de octubre de 2023, de <https://www.zaptest.com/es/pruebas-no-funcionales-que-es-tipos-enfoques-herramientas-y-mas>

*Tipos de testing de software*. (s/f). ..Ed.team. Recuperado el 7 de octubre de 2023, de  
<https://ed.team/blog/tipos-de-testing-de-software>

Vive. (2022, agosto 26). *La importancia de las pruebas de software*. UNIR.  
<https://www.unir.net/ingenieria/revista/pruebas-software/>