MODELADO SISTEMA DE MASCOTAS

Descripción general del sistema:

Se necesita diseñar una aplicación web que permita gestionar servicios de atención a mascotas

Consideremos estos flujos de datos:

1. Agendar una cita de cuidado de mascotas

CASOS DE USO

Los casos de uso dentro del diseño orientado a objetos consisten en abstraer a los actores o usuarios haciendo uso de un sistema de información, cada usuario tendrá alguna tarea.

1. **Caso de uso "Agendar una cita de cuidado de mascota"**

Actores:

- Usuario (dueño de la mascota)
- Sistema web de cuidado de mascotas
- Base de datos
- Veterinario

Flujo de eventos

- El usuario inicia sesión en la aplicación.
- El usuario selecciona la opción de programar una nueva cita.
- El sistema muestra las opciones de servicios y veterinarios disponibles.
- El usuario elige un servicio y un veterinario.
- El sistema muestra el calendario del veterinario seleccionado.
- El usuario elige una fecha y hora disponibles.

- El sistema confirma la disponibilidad y guarda la cita en la base de datos.
- El sistema envía una notificación de confirmación al usuario y al veterinario

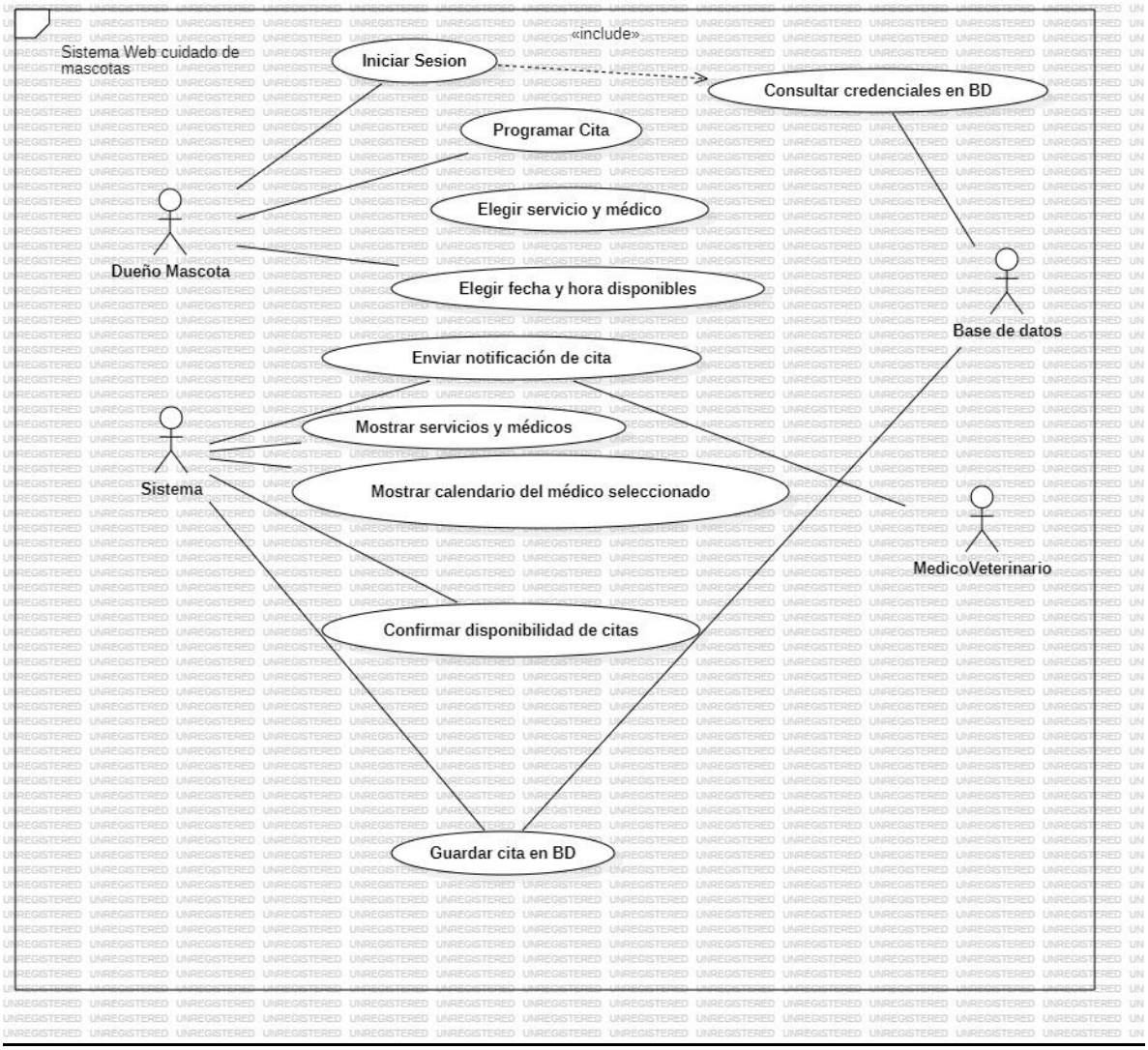


Diagrama de Secuencia

1. **Usuario**:

- Inicia sesión en la aplicación.
- Solicita programar una nueva cita.
- Selecciona el servicio y el veterinario.
- Elige una fecha y hora disponibles.

2. **Sistema Web de Cuidado de Mascotas:**

- Valida las credenciales del usuario con la base de datos.
- Muestra las opciones de servicios y veterinarios disponibles.
- Muestra el calendario del veterinario seleccionado.
- Confirma la disponibilidad de la fecha y hora.
- Guarda la cita en la base de datos.
- Envía notificaciones de confirmación al usuario y al veterinario.

3. **Base de Datos:**

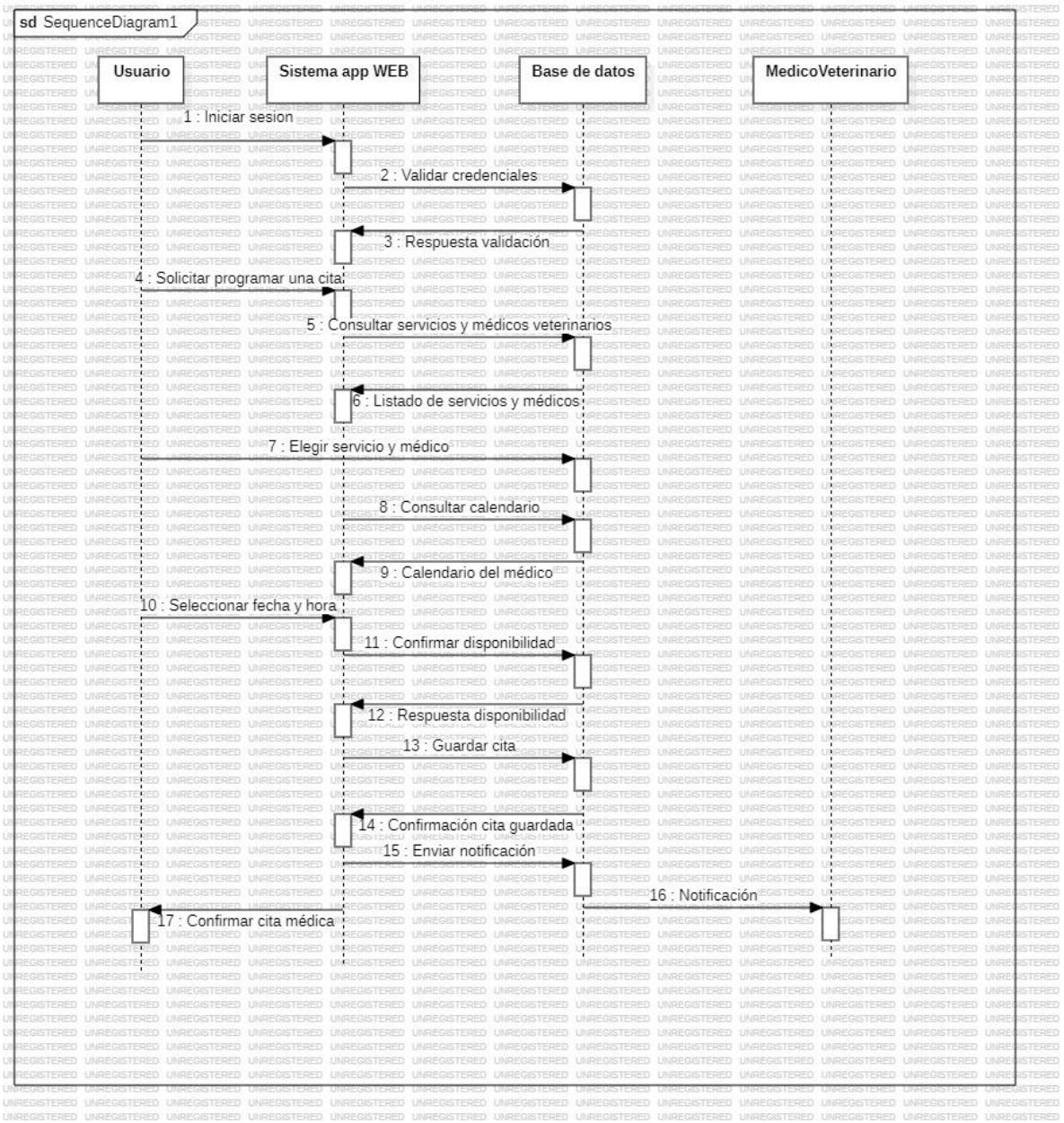
- Verifica las credenciales del usuario.
- Recupera las opciones de servicios y veterinarios.
- Recupera la disponibilidad del veterinario.
- Guarda la información de la cita.

4. Médico Veterinario

- Recibe la notificación de una cita programada

DIAGRAMA DE SECUENCIA.

1. Agendar una cita de cuidado de mascotas.



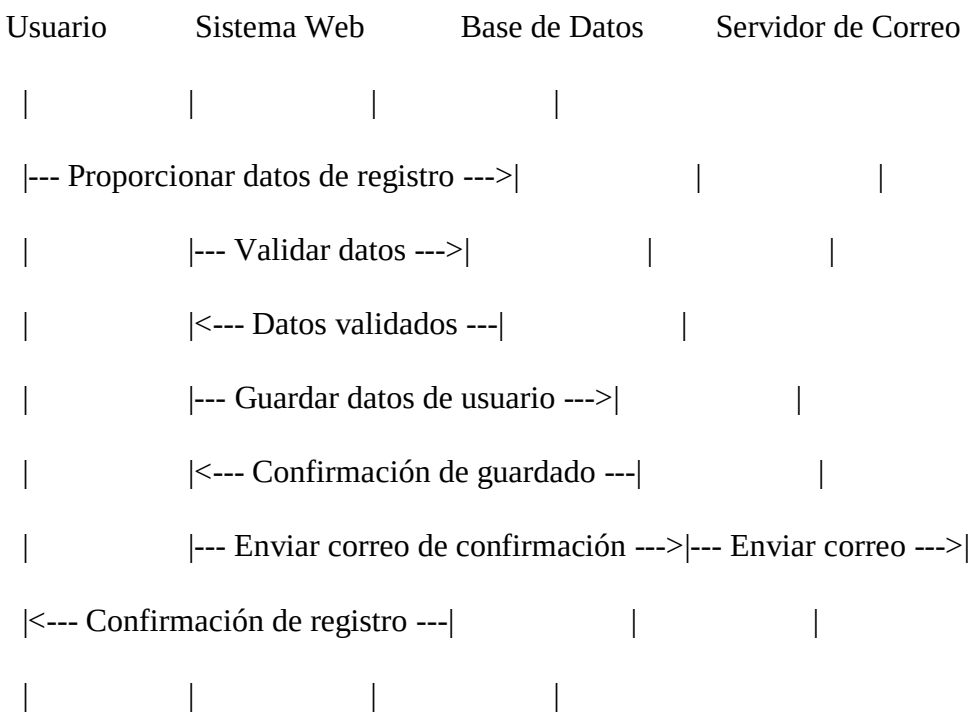
2. Caso de Uso Registro de Usuario

Descripción: Proceso por el cual un nuevo usuario se registra en la aplicación.

Diagrama de Secuencia

1. **Usuario**:
   - Proporciona datos de registro.
   - Confirma la información.
2. **Sistema Web de Cuidado de Mascotas**:
   - Valida la información del registro.
   - Guarda los datos en la base de datos.
   - Envía un correo de confirmación al usuario.
3. **Base de Datos**:
   - Guarda los datos del nuevo usuario.
4. **Servidor de Correo**:
   - Envía un correo de confirmación al usuario.

DIAGRAMA DE SECUENCIA



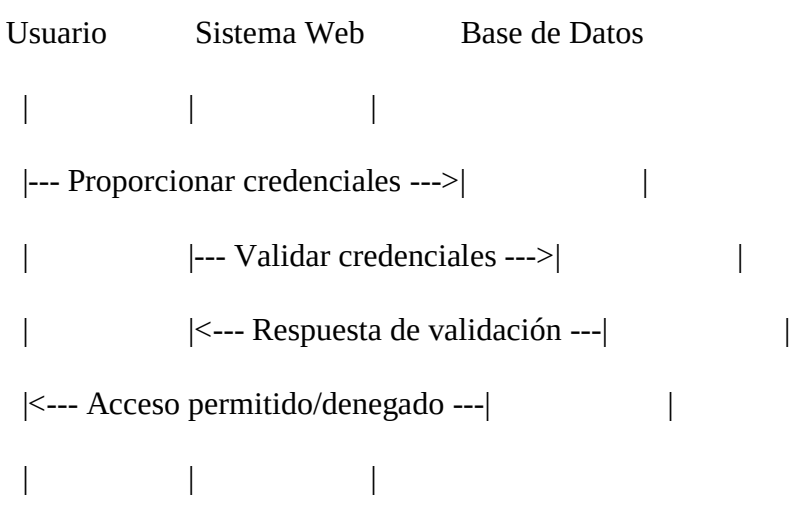
3. Inicio de Sesión

Descripción: Proceso por el cual un usuario existente inicia sesión en la aplicación.

Diagrama de Secuencia

1. **Usuario**:
   - Proporciona credenciales de inicio de sesión.
2. **Sistema Web de Cuidado de Mascotas**:
   - Valida las credenciales con la base de datos.
   - Otorga acceso si las credenciales son correctas.
3. **Base de Datos**:
   - Verifica las credenciales del usuario.

DIAGRAMA DE SECUENCIA

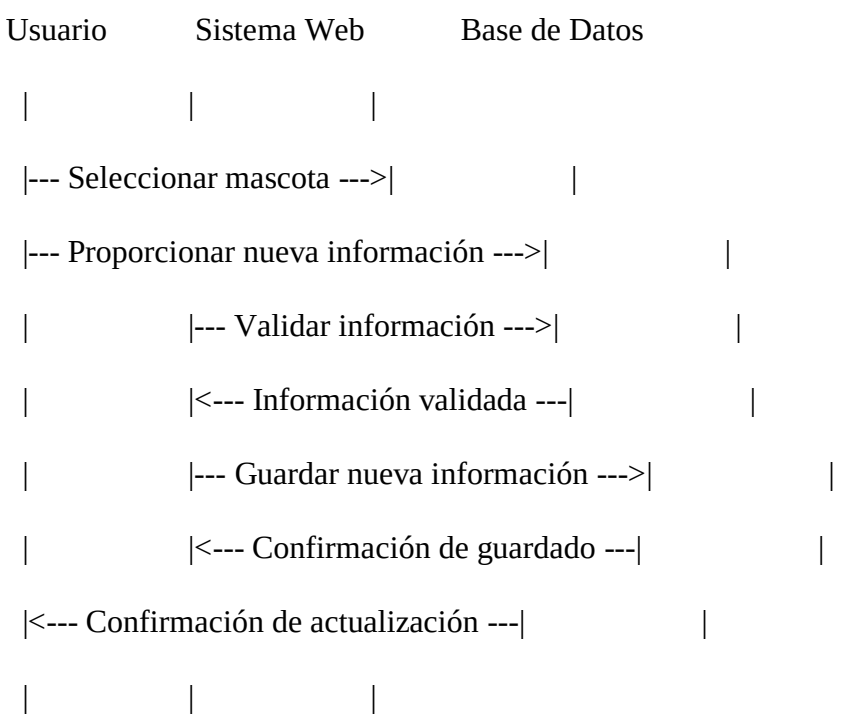


4. Actualización de Perfil de Mascota

Descripción: Proceso por el cual un usuario actualiza la información del perfil de su mascota.

Diagrama de Secuencia

1. **Usuario**:
   - Selecciona la mascota cuyo perfil quiere actualizar.
   - Proporciona la nueva información de la mascota.
2. **Sistema Web de Cuidado de Mascotas**:
   - Valida la nueva información.
   - Guarda la información actualizada en la base de datos.
3. **Base de Datos**:
   - Guarda la información actualizada de la mascota.

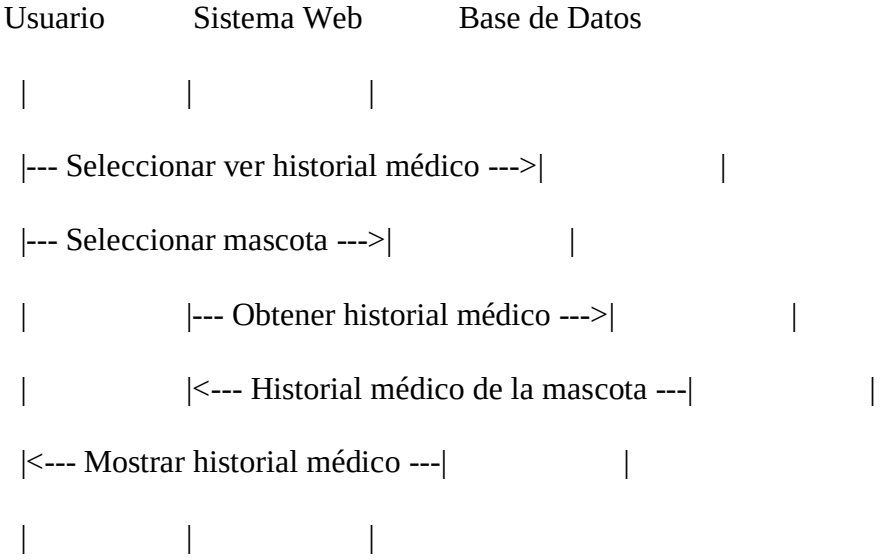


5. Consulta de Historial Médico de Mascota

Descripción: Proceso por el cual un usuario consulta el historial médico de su mascota.

Diagrama de Secuencia

1. **Usuario**:
   - Selecciona la opción de ver historial médico.
   - Selecciona la mascota.
2. **Sistema Web de Cuidado de Mascotas**:
   - Recupera el historial médico de la base de datos.
   - Muestra el historial médico al usuario.
3. **Base de Datos**:
   - Proporciona el historial médico de la mascota seleccionada.

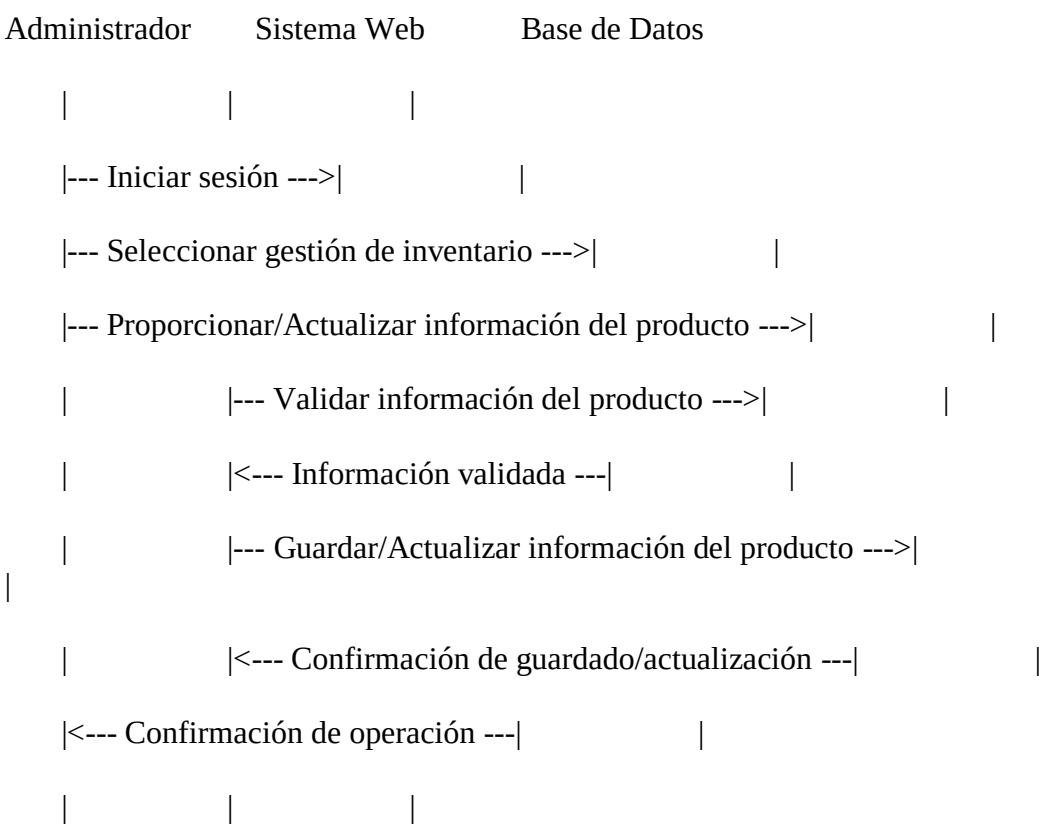


6. Gestión de Inventario de Productos (para administración)

Descripción: Proceso por el cual un administrador gestiona el inventario de productos.

Diagrama de Secuencia

1. **Administrador**:
   - Inicia sesión en la aplicación.
   - Selecciona la opción de gestión de inventario.
   - Proporciona información de nuevos productos o actualiza existentes.
2. **Sistema Web de Cuidado de Mascotas**:
   - Valida la información del producto.
   - Guarda o actualiza la información del producto en la base de datos.
3. **Base de Datos**:
   - Guarda o actualiza la información del producto.



Clases Principales

1. **Usuario**
   - **Atributos**: id, nombre, apellido, email, contraseña, dirección, teléfono
   - **Métodos**: iniciarSesion(), registrarse(), actualizarPerfil()
2. **Mascota**
   - **Atributos**: id, nombre, especie, raza, edad, peso, historialMedico
   - **Métodos**: actualizarPerfil(), verHistorialMedico()
3. **Cita**
   - **Atributos**: id, fecha, hora, servicio, veterinario, mascota, estado
   - **Métodos**: programarCita(), cancelarCita(), confirmarCita()
4. **Veterinario**
   - **Atributos**: id, nombre, especialidad, horario
   - **Métodos**: verCalendario(), actualizarDisponibilidad()
5. **Servicio**
   - **Atributos**: id, nombre, descripción, precio
   - **Métodos**: verDetalles()
6. **HistorialMedico**
   - **Atributos**: id, fecha, descripcion, tratamiento, mascota
   - **Métodos**: agregarEntrada(), verHistorial()
7. **Producto**
   - **Atributos**: id, nombre, descripción, precio, cantidadEnStock
   - **Métodos**: actualizarInventario(), verDetalles()
8. **Administrador**
   - **Atributos**: id, nombre, email, contraseña
   - **Métodos**: gestionarUsuarios(), gestionarProductos(), gestionarCitas()

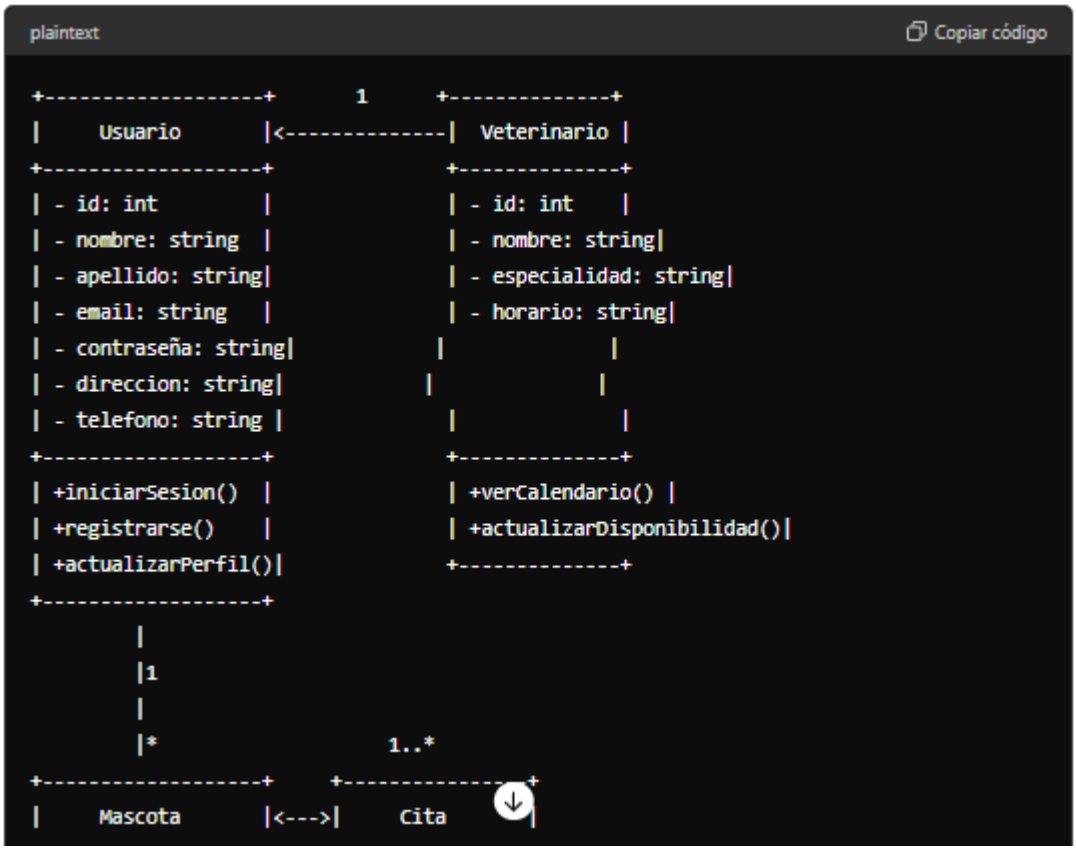
Relaciones

- **Usuario** tiene una o más **Mascotas**.
- **Mascota** puede tener uno o más **HistorialMedico**.
- **Usuario** puede programar varias **Citas**.
- **Cita** se asocia con una **Mascota**, un **Veterinario** y un **Servicio**.
- **Veterinario** puede proporcionar varios **Servicios**.
- **Administrador** puede gestionar **Usuarios**, **Productos** y **Citas**.

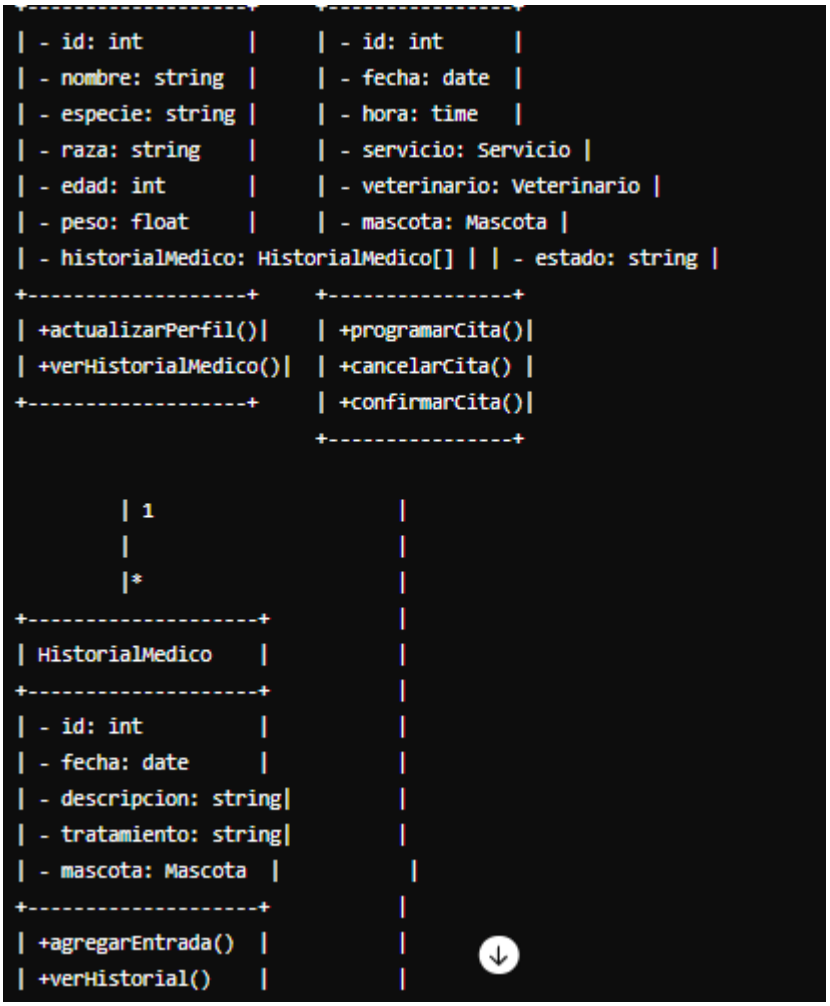
Diagrama de Clases

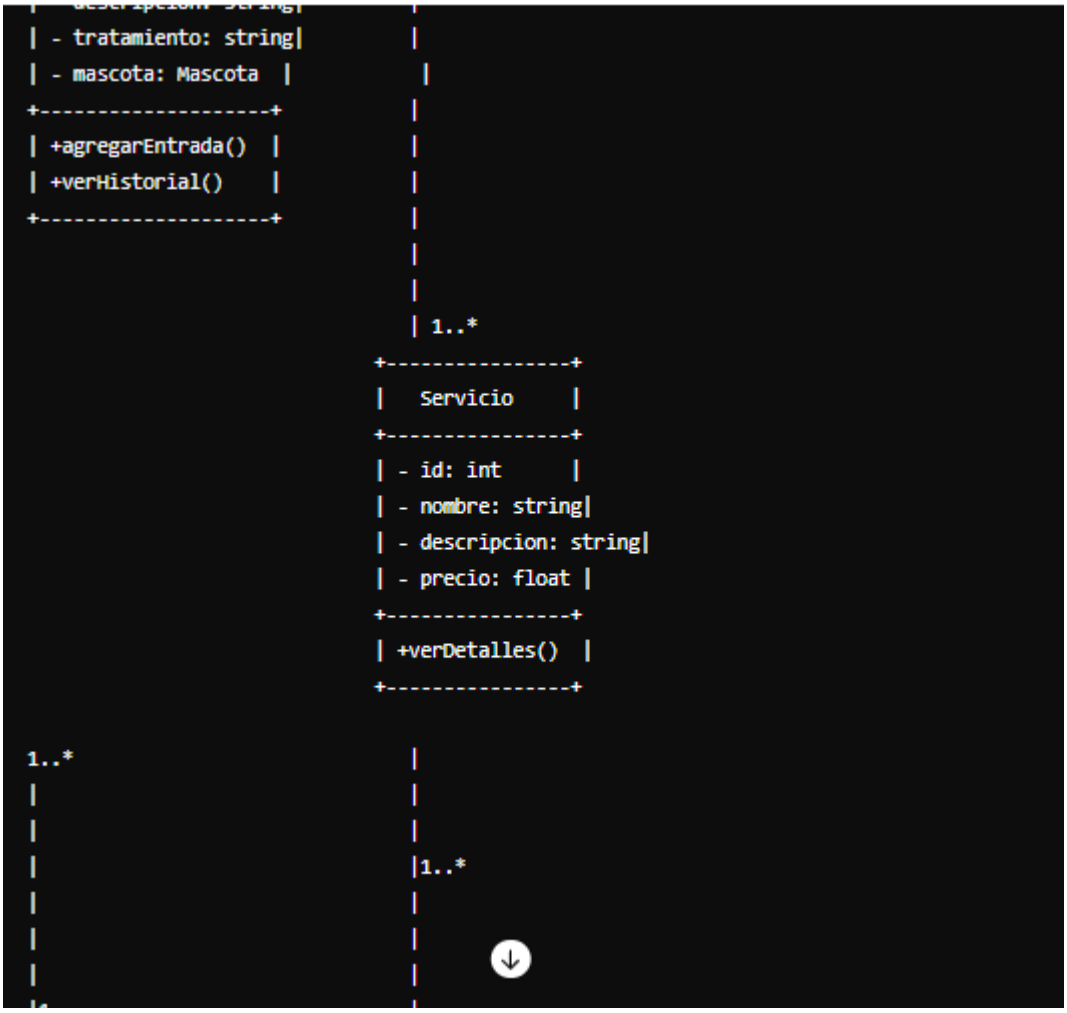
Aquí está la representación visual de estas clases y sus relaciones:

plaintext Copiar código

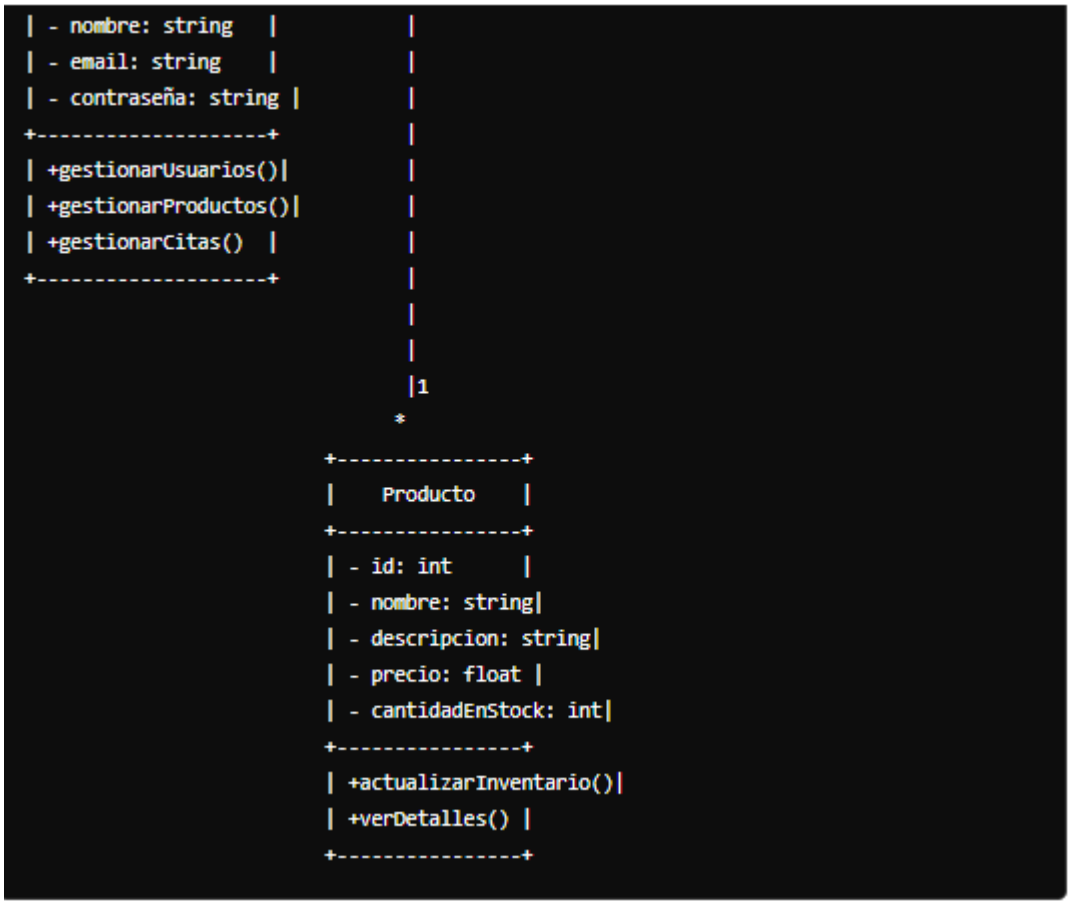



+iniciarSesion()
+registrarse()
+actualizarPerfil()
+verCalendario()
+actualizarDisponibilidad()
1
*
1..*
Mascota
Cita
- id: int
- nombre: string
- especie: string
- raza: string
- edad: int
- peso: float
- historialMedico: HistorialMedico[]
+actualizarPerfil()
+verHistorialMedico()
- id: int
- fecha: date
- hora: time
- servicio: Servicio
- veterinario: Veterinario
- mascota: Mascota
- estado: string
+programarCita()
+cancelarCita()
+confirmarCita()
1
*


- id: int
- nombre: string
- especie: string
- raza: string
- edad: int
- peso: float
- historialMedico: HistorialMedico[]
+actualizarPerfil()
+verHistorialMedico()
- id: int
- fecha: date
- hora: time
- servicio: Servicio
- veterinario: Veterinario
- mascota: Mascota
- estado: string
+programarCita()
+cancelarCita()
+confirmarCita()
1
*
HistorialMedico
- id: int
- fecha: date
- descripcion: string
- tratamiento: string
- mascota: Mascota
+agregarEntrada()
+verHistorial()


- tratamiento: string
- mascota: Mascota
+agregarEntrada()
+verHistorial()
1..*
Servicio
- id: int
- nombre: string
- descripcion: string
- precio: float
+verDetalles()
1..*
1..*


+----------------+
| +verDetalles() |
+----------------+
1..*
|1..*
|1
|1
+--------------------+
| Administrador |
+--------------------+
| - id: int |
| - nombre: string |
| - email: string |
| - contraseña: string |
+--------------------+
| +gestionarUsuarios()|
| +gestionarProductos()|
| +gestionarCitas() |
+--------------------+



- **Usuario**: Representa a los dueños de las mascotas que utilizan la aplicación. Pueden tener varias mascotas y programar citas.
- **Mascota**: Representa a las mascotas de los usuarios. Tienen un historial médico y pueden tener varias citas.
- **HistorialMedico**: Guarda el historial médico de una mascota.
- **Cita**: Representa una cita programada entre una mascota y un veterinario para un servicio específico.
- **Veterinario**: Representa a los veterinarios que proporcionan servicios.
- **Servicio**: Representa los servicios ofrecidos por los veterinarios.
- **Producto**: Representa los productos disponibles para compra o gestión de inventario.

- **Administrador**: Representa a los administradores del sistema que gestionan usuarios, productos y citas.

DICCIONARIO DE DATOS

Describe las estructuras de datos utilizadas, sus atributos, tipos de datos y cualquier otra información relevante. A continuación, se presenta el diccionario de datos para las principales entidades del sistema descritas en el diagrama de clases:

Tabla: Usuario

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único del usuario
nombre	VARCHAR(50)	Nombre del usuario
apellido	VARCHAR(50)	Apellido del usuario
email	VARCHAR(100)	Dirección de correo electrónico del usuario
contraseña	VARCHAR(255)	Contraseña del usuario
direccion	VARCHAR(255)	Dirección del usuario
telefono	VARCHAR(20)	Número de teléfono del usuario

Tabla: Mascota

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único de la mascota
nombre	VARCHAR(50)	Nombre de la mascota
especie	VARCHAR(50)	Especie de la mascota (e.g., perro, gato)
raza	VARCHAR(50)	Raza de la mascota
edad	INT	Edad de la mascota
peso	FLOAT	Peso de la mascota en kilogramos
usuario_id	INT (FK)	Identificador del dueño (Usuario)

Tabla: HistorialMedico

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único del historial médico
fecha	DATE	Fecha de la entrada en el historial médico
descripcion	TEXT	Descripción de la condición o visita
tratamiento	TEXT	Descripción del tratamiento proporcionado
mascota_id	INT (FK)	Identificador de la mascota a la que pertenece

Tabla: Cita

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único de la cita
fecha	DATE	Fecha de la cita
hora	TIME	Hora de la cita
servicio_id	INT (FK)	Identificador del servicio asociado a la cita
veterinario_id	INT (FK)	Identificador del veterinario asociado a la cita
mascota_id	INT (FK)	Identificador de la mascota asociada a la cita
estado	VARCHAR(20)	Estado de la cita (e.g., programada, confirmada, cancelada)

Tabla: Veterinario

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único del veterinario
nombre	VARCHAR(50)	Nombre del veterinario
especialidad	VARCHAR(100)	Especialidad del veterinario
horario	VARCHAR(255)	Horario de atención del veterinario

Tabla: Servicio

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único del servicio
nombre	VARCHAR(100)	Nombre del servicio
descripcion	TEXT	Descripción del servicio
precio	FLOAT	Precio del servicio

Tabla: Producto

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único del producto
nombre	VARCHAR(100)	Nombre del producto
descripcion	TEXT	Descripción del producto
precio	FLOAT	Precio del producto
cantidadEnStock	INT	Cantidad de productos en stock

Tabla: Administrador

Nombre del Atributo	Tipo de Datos	Descripción
id	INT (PK)	Identificador único del administrador
nombre	VARCHAR(50)	Nombre del administrador
email	VARCHAR(100)	Dirección de correo electrónico del administrador
contraseña	VARCHAR(255)	Contraseña del administrador

Relaciones entre Tablas

- **Usuario** a **Mascota**: 1 a N (Un usuario puede tener muchas mascotas).
- **Mascota** a **HistorialMedico**: 1 a N (Una mascota puede tener muchos registros en su historial médico).
- **Usuario** a **Cita**: 1 a N (Un usuario puede programar muchas citas).
- **Mascota** a **Cita**: 1 a N (Una mascota puede tener muchas citas).
- **Veterinario** a **Cita**: 1 a N (Un veterinario puede atender muchas citas).
- **Servicio** a **Cita**: 1 a N (Un servicio puede estar asociado con muchas citas).
- **Administrador** a **Usuario/Producto/Cita**: Relaciones administrativas para la gestión.

Notas Adicionales

- Las claves primarias (PK) aseguran la unicidad de cada registro en la tabla.
- Las claves foráneas (FK) se utilizan para establecer relaciones entre tablas, garantizando la integridad referencial.
- Los tipos de datos se seleccionan para asegurar que la información se almacene de manera eficiente y precisa.

Tablas y Relaciones

1. **Tabla: Usuario**
   - **id** (INT, PK)
   - **nombre** (VARCHAR(50))
   - **apellido** (VARCHAR(50))
   - **email** (VARCHAR(100), UNIQUE)
   - **contraseña** (VARCHAR(255))
   - **direccion** (VARCHAR(255))
   - **telefono** (VARCHAR(20))
2. **Tabla: Mascota**
   - **id** (INT, PK)
   - **nombre** (VARCHAR(50))
   - **especie** (VARCHAR(50))
   - **raza** (VARCHAR(50))
   - **edad** (INT)
   - **peso** (FLOAT)
   - **usuario_id** (INT, FK, references Usuario(id))
3. **Tabla: HistorialMedico**
   - **id** (INT, PK)
   - **fecha** (DATE)
   - **descripcion** (TEXT)
   - **tratamiento** (TEXT)
   - **mascota_id** (INT, FK, references Mascota(id))
4. **Tabla: Cita**
   - **id** (INT, PK)
   - **fecha** (DATE)
   - **hora** (TIME)
   - **servicio_id** (INT, FK, references Servicio(id))
   - **veterinario_id** (INT, FK, references Veterinario(id))
   - **mascota_id** (INT, FK, references Mascota(id))
   - **estado** (VARCHAR(20))
5. **Tabla: Veterinario**
   - **id** (INT, PK)
   - **nombre** (VARCHAR(50))
   - **especialidad** (VARCHAR(100))
   - **horario** (VARCHAR(255))
6. **Tabla: Servicio**
   - **id** (INT, PK)
   - **nombre** (VARCHAR(100))
   - **descripcion** (TEXT)
   - **precio** (FLOAT)
7. **Tabla: Producto**
   - **id** (INT, PK)
   - **nombre** (VARCHAR(100))
   - **descripcion** (TEXT)
   - **precio** (FLOAT)
   - **cantidadEnStock** (INT)

8. **Tabla: Administrador**
    - **id** (INT, PK)
    - **nombre** (VARCHAR(50))
    - **email** (VARCHAR(100), UNIQUE)
    - **contraseña** (VARCHAR(255))

Relaciones entre Tablas

1. **Usuario-Mascota**
    - Un usuario puede tener muchas mascotas.
    - `Mascota.usuario_id` es una clave foránea que referencia `Usuario.id`.
2. **Mascota-HistorialMedico**
    - Una mascota puede tener muchos registros en su historial médico.
    - `HistorialMedico.mascota_id` es una clave foránea que referencia `Mascota.id`.
3. **Mascota-Cita**
    - Una mascota puede tener muchas citas.
    - `Cita.mascota_id` es una clave foránea que referencia `Mascota.id`.
4. **Veterinario-Cita**
    - Un veterinario puede atender muchas citas.
    - `Cita.veterinario_id` es una clave foránea que referencia `Veterinario.id`.
5. **Servicio-Cita**
    - Un servicio puede estar asociado con muchas citas.
    - `Cita.servicio_id` es una clave foránea que referencia `Servicio.id`.

Diagramas Entidad-Relación (ERD)

En un diagrama entidad-relación, las tablas y las relaciones se pueden representar de la siguiente manera:

Entidades

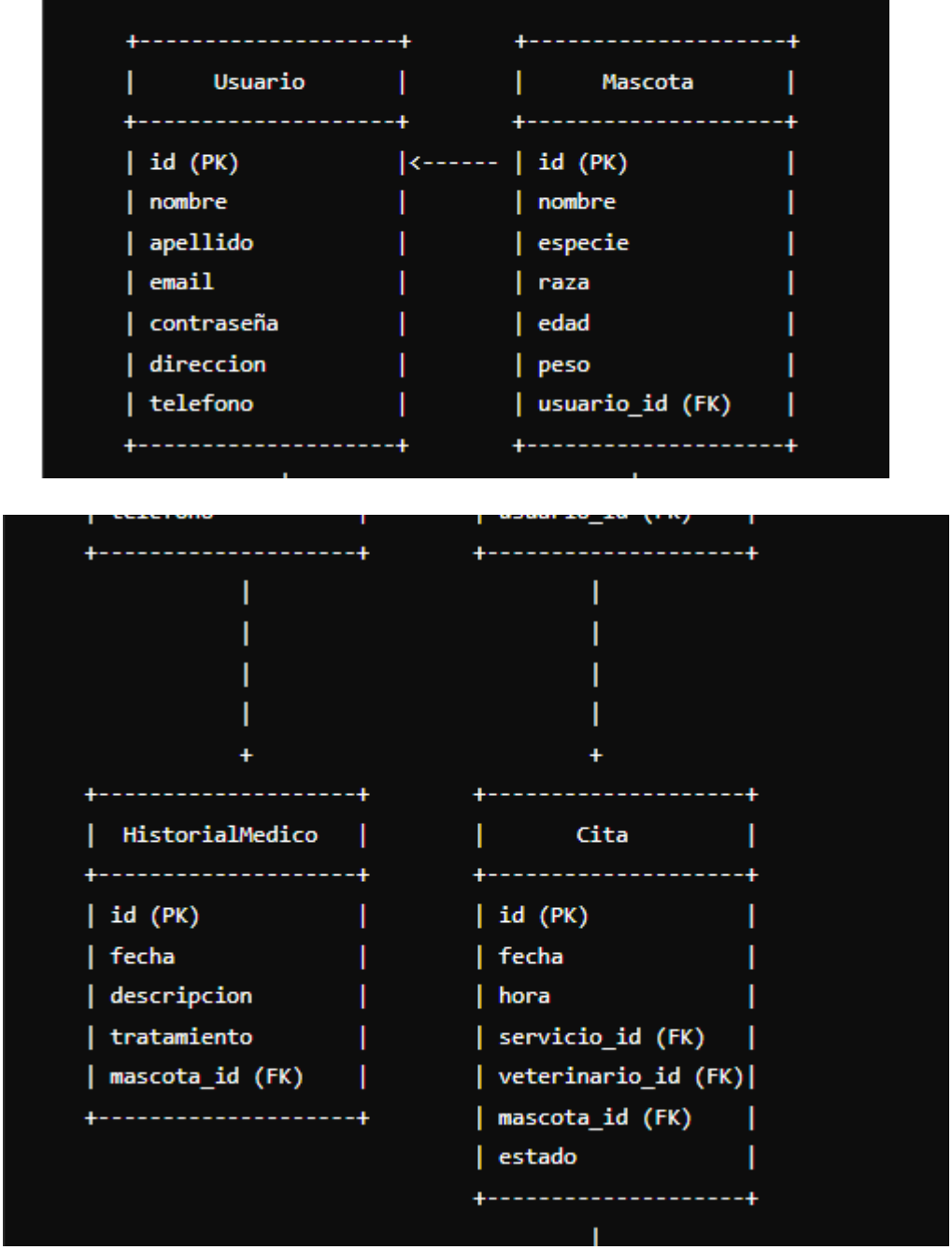
- **Usuario**
  - Atributos: id (PK), nombre, apellido, email, contraseña, direccion, telefono
- **Mascota**
  - Atributos: id (PK), nombre, especie, raza, edad, peso, usuario_id (FK)
- **HistorialMedico**
  - Atributos: id (PK), fecha, descripcion, tratamiento, mascota_id (FK)
- **Cita**
  - Atributos: id (PK), fecha, hora, servicio_id (FK), veterinario_id (FK), mascota_id (FK), estado
- **Veterinario**
  - Atributos: id (PK), nombre, especialidad, horario
- **Servicio**
  - Atributos: id (PK), nombre, descripcion, precio
- **Producto**
  - Atributos: id (PK), nombre, descripcion, precio, cantidadEnStock
- **Administrador**
  - Atributos: id (PK), nombre, email, contraseña

Relaciones

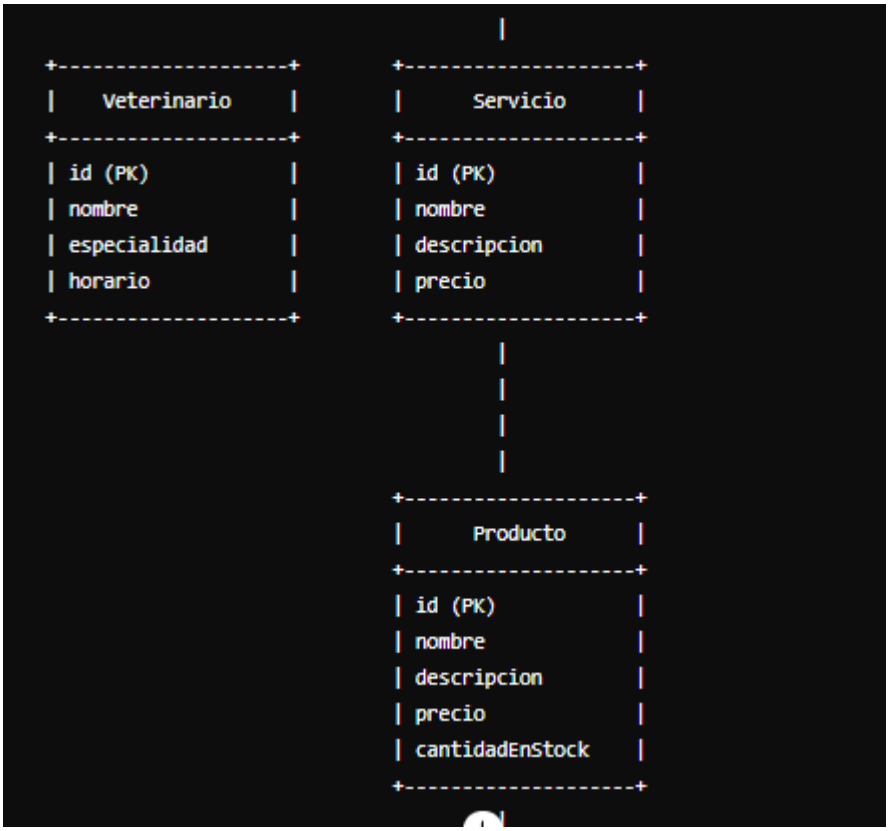
1. **Usuario-Mascota**: 1 a N
2. **Mascota-HistorialMedico**: 1 a N
3. **Mascota-Cita**: 1 a N
4. **Veterinario-Cita**: 1 a N
5. **Servicio-Cita**: 1 a N

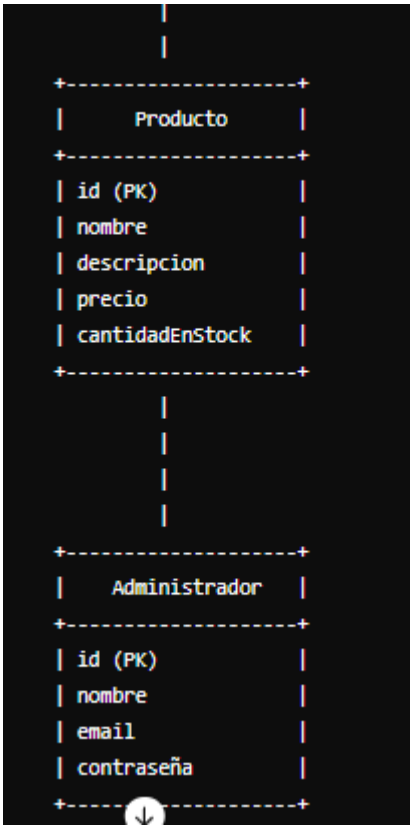
Modelo Relacional Visual

plaintext




mascota_id (FK)
estado
Veterinario
id (PK)
nombre
especialidad
horario
Servicio
id (PK)
nombre
descripcion
precio


Veterinario
id (PK)
nombre
especialidad
horario
Servicio
id (PK)
nombre
descripcion
precio
Producto
id (PK)
nombre
descripcion
precio
cantidadEnStock


Producto
id (PK)
nombre
descripcion
precio
cantidadEnStock
Administrador
id (PK)
nombre
email
contraseña

CASOS DE USO

Actores Principales

1. **Usuario (Dueño de la Mascota)**
2. **Veterinario**
3. **Administrador**
4. **Visitante (Usuario no registrado)**

Casos de Uso

1. Registro y Autenticación

Actor: Visitante

- **Caso de uso: Registrarse**
  - Descripción: El visitante se registra en el sistema proporcionando sus datos personales.
  - Precondiciones: El visitante no debe estar registrado.
  - Flujo principal:
    1. El visitante accede al formulario de registro.
    2. El visitante completa el formulario con sus datos personales.
    3. El visitante envía el formulario.
    4. El sistema valida los datos y crea una nueva cuenta de usuario.
  - Postcondiciones: El visitante se convierte en usuario registrado y puede iniciar sesión.
- **Caso de uso: Iniciar sesión**
  - Descripción: El usuario se autentica en el sistema proporcionando sus credenciales.
  - Precondiciones: El usuario debe estar registrado.
  - Flujo principal:
    1. El usuario accede al formulario de inicio de sesión.
    2. El usuario ingresa su email y contraseña.
    3. El usuario envía el formulario.
    4. El sistema valida las credenciales y otorga acceso.
  - Postcondiciones: El usuario tiene acceso a las funcionalidades del sistema según su rol.

2. Gestión de Perfil

Actor: Usuario

- **Caso de uso: Actualizar perfil**
  - Descripción: El usuario actualiza su información personal.
  - Precondiciones: El usuario debe estar autenticado.

- Flujo principal:
    1. El usuario accede a la página de perfil.
    2. El usuario modifica los datos deseados.
    3. El usuario guarda los cambios.
    4. El sistema valida y actualiza la información.
- Postcondiciones: La información del perfil del usuario se actualiza en el sistema.

3. Gestión de Mascotas

Actor: Usuario

- **Caso de uso: Registrar mascota**
    - Descripción: El usuario registra una nueva mascota en el sistema.
    - Precondiciones: El usuario debe estar autenticado.
    - Flujo principal:
        1. El usuario accede al formulario de registro de mascota.
        2. El usuario completa el formulario con los datos de la mascota.
        3. El usuario envía el formulario.
        4. El sistema valida los datos y registra la mascota.
    - Postcondiciones: La nueva mascota se registra en el sistema.
- **Caso de uso: Actualizar perfil de mascota**
    - Descripción: El usuario actualiza la información de una mascota registrada.
    - Precondiciones: El usuario debe estar autenticado y tener mascotas registradas.
    - Flujo principal:
        1. El usuario accede a la página de perfil de la mascota.
        2. El usuario modifica los datos deseados.
        3. El usuario guarda los cambios.
        4. El sistema valida y actualiza la información.
    - Postcondiciones: La información de la mascota se actualiza en el sistema.

4. Gestión de Citas

Actor: Usuario, Veterinario

- **Caso de uso: Programar cita**
    - Descripción: El usuario programa una cita con un veterinario.
    - Precondiciones: El usuario debe estar autenticado y tener mascotas registradas.
    - Flujo principal:
        1. El usuario accede a la página de programación de citas.
        2. El usuario selecciona la mascota, el servicio y el veterinario.
        3. El usuario selecciona la fecha y hora disponibles.
        4. El usuario confirma la cita.
        5. El sistema valida y registra la cita.
    - Postcondiciones: La cita se programa y se notifica al usuario y al veterinario.
- **Caso de uso: Cancelar cita**
    - Descripción: El usuario cancela una cita programada.
    - Precondiciones: El usuario debe estar autenticado y tener citas programadas.

- Flujo principal:
    1. El usuario accede a la página de citas programadas.
    2. El usuario selecciona la cita a cancelar.
    3. El usuario confirma la cancelación.
    4. El sistema actualiza el estado de la cita.
- Postcondiciones: La cita se cancela y se notifica al usuario y al veterinario.

- **Caso de uso: Confirmar cita**
    - Descripción: El veterinario confirma la asistencia a una cita programada.
    - Precondiciones: El veterinario debe estar autenticado y tener citas programadas.
    - Flujo principal:
        1. El veterinario accede a la página de citas programadas.
        2. El veterinario selecciona la cita a confirmar.
        3. El veterinario confirma la asistencia.
        4. El sistema actualiza el estado de la cita.
    - Postcondiciones: La cita se confirma y se notifica al usuario.

5. Gestión de Servicios

Actor: Administrador

- **Caso de uso: Gestionar servicios**
    - Descripción: El administrador agrega, actualiza o elimina servicios ofrecidos.
    - Precondiciones: El administrador debe estar autenticado.
    - Flujo principal:
        1. El administrador accede a la página de gestión de servicios.
        2. El administrador selecciona agregar, actualizar o eliminar un servicio.
        3. El administrador completa la acción correspondiente.
        4. El sistema valida y aplica los cambios.
    - Postcondiciones: Los servicios se gestionan en el sistema.

6. Gestión de Productos

Actor: Administrador

1. **Caso de uso: Gestionar productos**
    - Descripción: El administrador agrega, actualiza o elimina productos disponibles.
    - Precondiciones: El administrador debe estar autenticado.
    - Flujo principal:
        1. El administrador accede a la página de gestión de productos.
        2. El administrador selecciona agregar, actualizar o eliminar un producto.
        3. El administrador completa la acción correspondiente.
        4. El sistema valida y aplica los cambios.
    - Postcondiciones: Los productos se gestionan en el sistema.

7. Gestión de Usuarios

Actor: Administrador

- **Caso de uso: Gestionar usuarios**
  - Descripción: El administrador gestiona los usuarios registrados en el sistema.
  - Precondiciones: El administrador debe estar autenticado.
  - Flujo principal:
    1. El administrador accede a la página de gestión de usuarios.
    2. El administrador selecciona agregar, actualizar o eliminar un usuario.
    3. El administrador completa la acción correspondiente.
    4. El sistema valida y aplica los cambios.
  - Postcondiciones: Los usuarios se gestionan en el sistema.

Resumen de Casos de Uso

1. **Registro y Autenticación**
   - Registrarse
   - Iniciar sesión
2. **Gestión de Perfil**
   - Actualizar perfil
3. **Gestión de Mascotas**
   - Registrar mascota
   - Actualizar perfil de mascota
4. **Gestión de Citas**
   - Programar cita
   - Cancelar cita
   - Confirmar cita
5. **Gestión de Servicios**
   - Gestionar servicios
6. **Gestión de Productos**
   - Gestionar productos
7. **Gestión de Usuarios**
   - Gestionar usuarios

DIAGRAMA DE OBJETOS

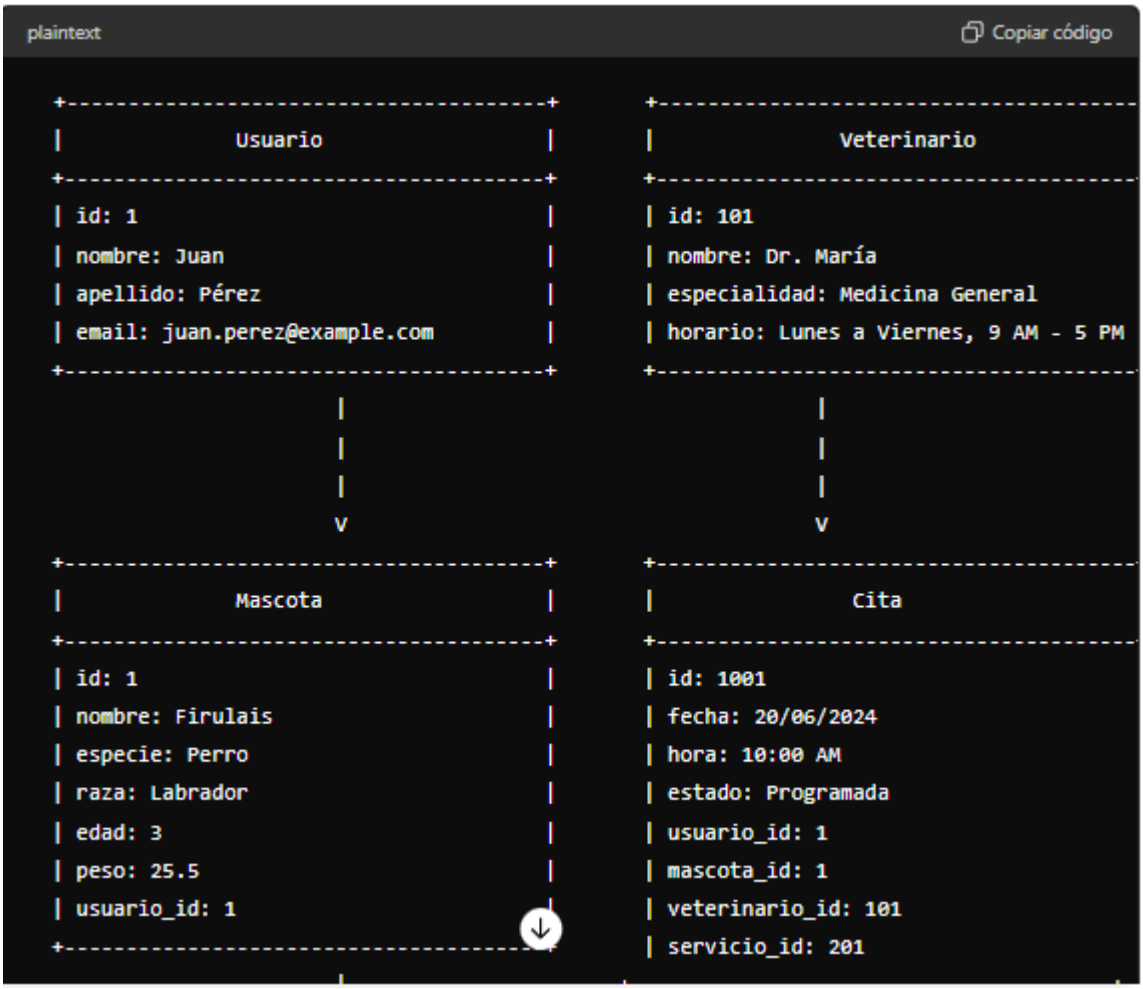
El diagrama de objetos muestra una instantánea de las instancias de clases y sus relaciones en un momento específico durante la ejecución del sistema. Para la aplicación de software web de cuidado de mascotas, el diagrama de objetos puede reflejar cómo interactúan los objetos de las principales clases en un escenario particular, como la programación de una cita.

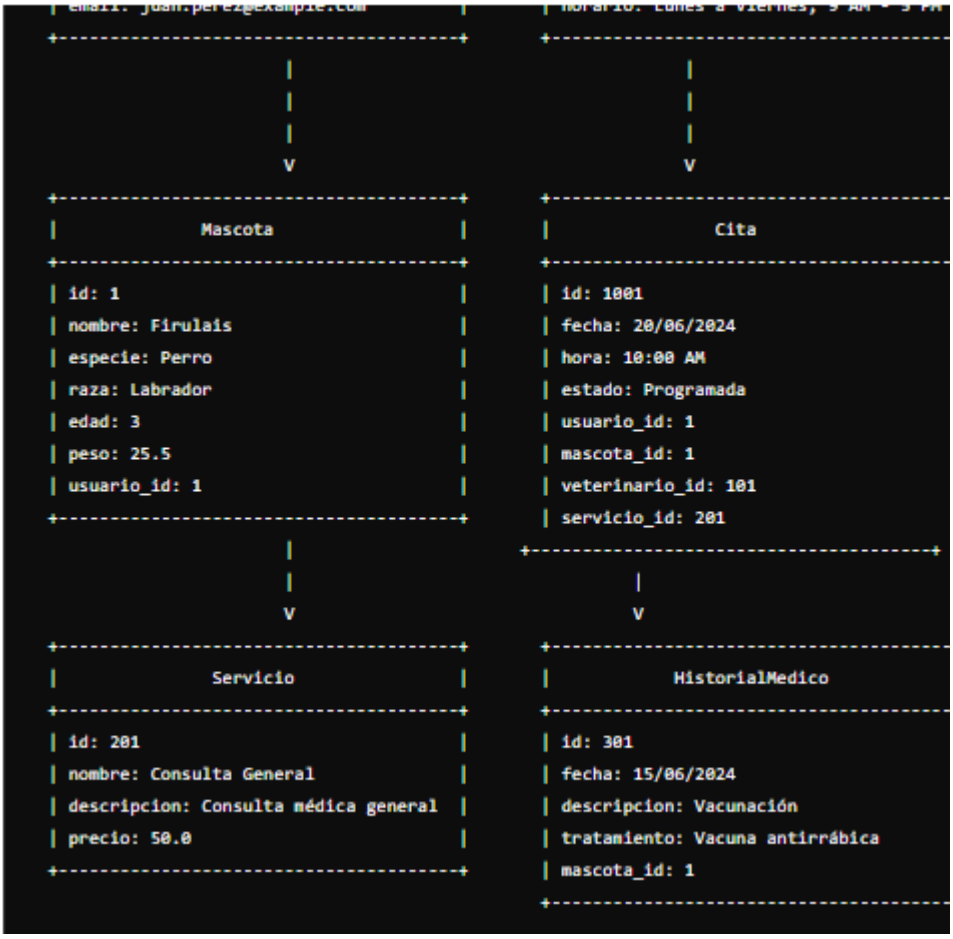
Escenario: Programación de una Cita

Descripción del Escenario:

- Un usuario llamado Juan Pérez tiene una mascota llamada Firulais.
- Juan Pérez programa una cita para Firulais con el veterinario Dr. María López para un servicio de consulta general.
- La cita está programada para el 20 de junio de 2024 a las 10:00 AM.

plaintext Copiar código





Descripción del Diagrama de Objetos

1. **Usuario: Juan Pérez**
   - `id: 1`
   - `nombre: Juan`
   - `apellido: Pérez`
   - `email: juan.perez@example.com`
2. **Mascota: Firulais**
   - `id: 1`
   - `nombre: Firulais`
   - `especie: Perro`
   - `raza: Labrador`
   - `edad: 3`
   - `peso: 25.5`
   - `usuario_id: 1` (Juan Pérez es el dueño)
3. **Cita:**
   - `id: 1001`
   - `fecha: 20/06/2024`
   - `hora: 10:00 AM`
   - `estado: Programada`
   - `usuario_id: 1` (Juan Pérez)
   - `mascota_id: 1` (Firulais)
   - `veterinario_id: 101` (Dr. María López)

    - `servicio_id: 201` (Consulta General)
4. **Veterinario: Dr. María López**
    - `id: 101`
    - `nombre: Dr. María`
    - `especialidad: Medicina General`
    - `horario: Lunes a Viernes, 9 AM - 5 PM`
5. **Servicio: Consulta General**
    - `id: 201`
    - `nombre: Consulta General`
    - `descripcion: Consulta médica general`
    - `precio: 50.0`
6. **HistorialMedico:**
    - `id: 301`
    - `fecha: 15/06/2024`
    - `descripcion: Vacunación`
    - `tratamiento: Vacuna antirrábica`
    - `mascota_id: 1` (Firulais)

Notas Adicionales

- El diagrama de objetos representa una instantánea específica, en este caso, enfocada en el proceso de programación de una cita para una mascota.
- Cada instancia de objeto contiene atributos relevantes, mostrando cómo se relacionan entre sí.
- Las relaciones se indican mediante líneas que conectan las instancias de objetos correspondientes, reflejando las claves foráneas y las asociaciones en el modelo de datos.

Este diagrama de objetos ayuda a visualizar el estado del sistema en un momento dado,

DISEÑO DE FRONT-END

Diseño Responsivo

- **Tecnologías:** Utiliza frameworks CSS como Bootstrap o Tailwind CSS para asegurarte de que la aplicación se vea bien en dispositivos móviles, tabletas y escritorios.
- **Flexbox y Grid:** Emplea Flexbox y CSS Grid para crear layouts flexibles y adaptables.

2. Interfaz de Usuario Intuitiva

- **Navegación Clara:** Diseña una barra de navegación clara y sencilla que permita a los usuarios acceder rápidamente a las principales secciones del sitio (Perfil, Mascotas, Citas, Servicios, Productos).
- **Breadcrumbs:** Implementa breadcrumbs para que los usuarios siempre sepan en qué parte de la aplicación se encuentran.

3. Componentes Reutilizables

- **Librerías de Componentes:** Utiliza librerías de componentes como React, Vue o Angular para crear componentes reutilizables (formularios, tarjetas de perfil de mascotas, calendarios de citas).
- **Modularidad:** Mantén tu código modular y bien organizado para facilitar el mantenimiento y la escalabilidad.

4. Accesibilidad

- **Contraste y Tipografía:** Asegúrate de que haya suficiente contraste entre el texto y el fondo, y utiliza tipografías legibles.
- **Aria-Labels:** Usa atributos ARIA para mejorar la accesibilidad para usuarios que utilizan lectores de pantalla.
- **Navegación por Teclado:** Asegúrate de que la aplicación sea completamente navegable mediante teclado.

5. Feedback del Usuario

- **Mensajes de Error y Confirmación:** Proporciona mensajes claros y útiles cuando ocurran errores o se completen acciones exitosamente.
- **Indicadores de Carga:** Utiliza spinners o barras de progreso para indicar que se está procesando una acción.

6. Formulario de Gestión de Mascotas y Citas

- **Validación en Tiempo Real:** Implementa validación en tiempo real para mejorar la experiencia del usuario al completar formularios.
- **Autocompletar y Sugerencias:** Usa autocompletar y sugerencias en campos de formulario para agilizar la entrada de datos.

7. Panel de Control del Usuario

- **Resumen Rápido:** Proporciona un dashboard que resuma información clave, como próximas citas, historial médico reciente de mascotas y productos recomendados.
- **Notificaciones:** Implementa un sistema de notificaciones para recordar citas, vacunas pendientes o nuevos productos disponibles.

8. Integración de Mapas

- **Localización de Veterinarios:** Usa APIs de mapas como Google Maps para mostrar la ubicación de veterinarios cercanos.
- **Direcciones:** Proporciona direcciones y opciones de ruta para facilitar a los usuarios llegar a las citas.

9. Seguridad

- **Protección de Datos:** Asegúrate de que todos los datos personales y de mascotas estén protegidos mediante HTTPS y prácticas de seguridad sólidas.
- **Autenticación:** Implementa autenticación segura con opciones de recuperación de contraseña y verificación en dos pasos si es necesario.

10. Estética y Marca

- **Consistencia Visual:** Mantén un diseño visual coherente con la identidad de la marca, utilizando una paleta de colores, fuentes y estilos visuales consistentes.
- **Imágenes y Multimedia:** Usa imágenes de alta calidad y vídeos cuando sea relevante para enriquecer la experiencia del usuario.

11. Interacción Social

- **Compartir en Redes Sociales:** Permite a los usuarios compartir información sobre sus mascotas, citas o productos en redes sociales.
- **Comentarios y Reseñas:** Implementa un sistema de comentarios y reseñas para servicios y productos.

12. Optimización del Rendimiento

- **Carga Rápida:** Optimiza las imágenes y los recursos estáticos para mejorar los tiempos de carga.
- **Caching:** Implementa técnicas de caching para reducir el tiempo de espera del usuario.

13. Soporte y Ayuda

- **FAQs y Tutoriales:** Proporciona una sección de preguntas frecuentes y tutoriales en vídeo para ayudar a los usuarios a navegar por la aplicación.

- **Chat en Vivo:** Considera la implementación de un sistema de chat en vivo para soporte inmediato.

14. Feedback del Usuario

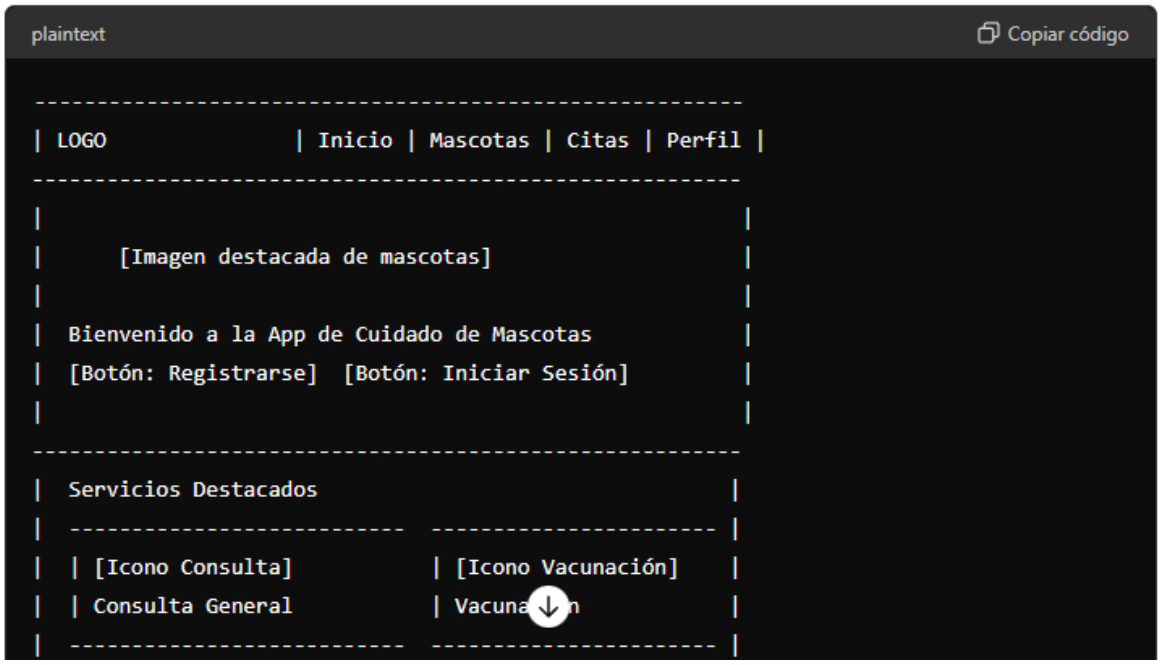
- **Encuestas y Sugerencias:** Implementa encuestas y formularios de sugerencias para recopilar feedback continuo de los usuarios y mejorar la aplicación.

Ejemplo de Tecnologías y Herramientas para el Front-End

- **Frameworks:** React.js, Vue.js, Angular
- **CSS:** Bootstrap, Tailwind CSS, SASS
- **Gestión del Estado:** Redux (para React), Vuex (para Vue)
- **Validación de Formularios:** Formik (para React), Vuelidate (para Vue)
- **Autenticación:** JWT, OAuth

MAQUETAS

PAGINA PRINCIPAL



MAPA DE NAVEGACIÓN

1. **Inicio**
    - Página de bienvenida
    - Información sobre la aplicación
    - Botones de registro e inicio de sesión
2. **Perfil**
    - Información personal
    - Actualizar perfil
    - Historial de actividades
3. **Mascotas**
    - Lista de mascotas
    - Registrar nueva mascota
    - Perfil de mascota (detalles, historial médico, etc.)
4. **Citas**
    - Programar cita
    - Lista de citas programadas
    - Historial de citas
    - Cancelar cita
5. **Servicios**
    - Lista de servicios ofrecidos
    - Detalles de cada servicio
    - Precios
6. **Productos**
    - Catálogo de productos
    - Detalles del producto
    - Agregar al carrito/comprar
7. **Administración**
    - Gestión de usuarios
    - Gestión de servicios
    - Gestión de productos
8. **Soporte**
    - Preguntas frecuentes
    - Contacto
    - Chat en vivo
9. **Configuración**
    - Configuración de cuenta
    - Preferencias
    - Notificaciones
10. **Ayuda**
    - Tutoriales
    - Documentación

Recomendaciones Específicas

1. Claridad y Simplicidad

- **Menú Principal:** Mantén el menú de navegación principal claro y sencillo, con categorías principales claramente definidas.
- **Submenús:** Utiliza submenús para agrupar secciones relacionadas y evitar sobrecargar el menú principal.

2. Consistencia

- **Diseño Consistente:** Asegúrate de que todos los elementos de navegación (botones, enlaces, menús) mantengan un estilo consistente en todas las páginas.
- **Terminología Uniforme:** Usa términos consistentes para las acciones y secciones para evitar confusión.

3. Acceso Rápido

- **Barra de Búsqueda:** Incluye una barra de búsqueda visible en todas las páginas para permitir a los usuarios encontrar rápidamente lo que necesitan.
- **Atajos y Enlaces Rápidos:** Proporciona accesos directos a las funciones más utilizadas (programar cita, ver perfil de mascota, etc.).

4. Retroalimentación del Usuario

- **Indicadores Visuales:** Usa indicadores visuales (colores, iconos) para mostrar al usuario en qué sección se encuentra.
- **Confirmaciones y Alertas:** Proporciona confirmaciones y alertas visuales para acciones importantes como programar o cancelar citas.

5. Adaptabilidad

- **Responsive Design:** Asegúrate de que el mapa de navegación sea completamente responsive, adaptándose a diferentes tamaños de pantalla.
- **Menús Desplegables:** Utiliza menús desplegables que funcionen bien en dispositivos móviles y de escritorio.

6. Optimización de la Experiencia de Usuario

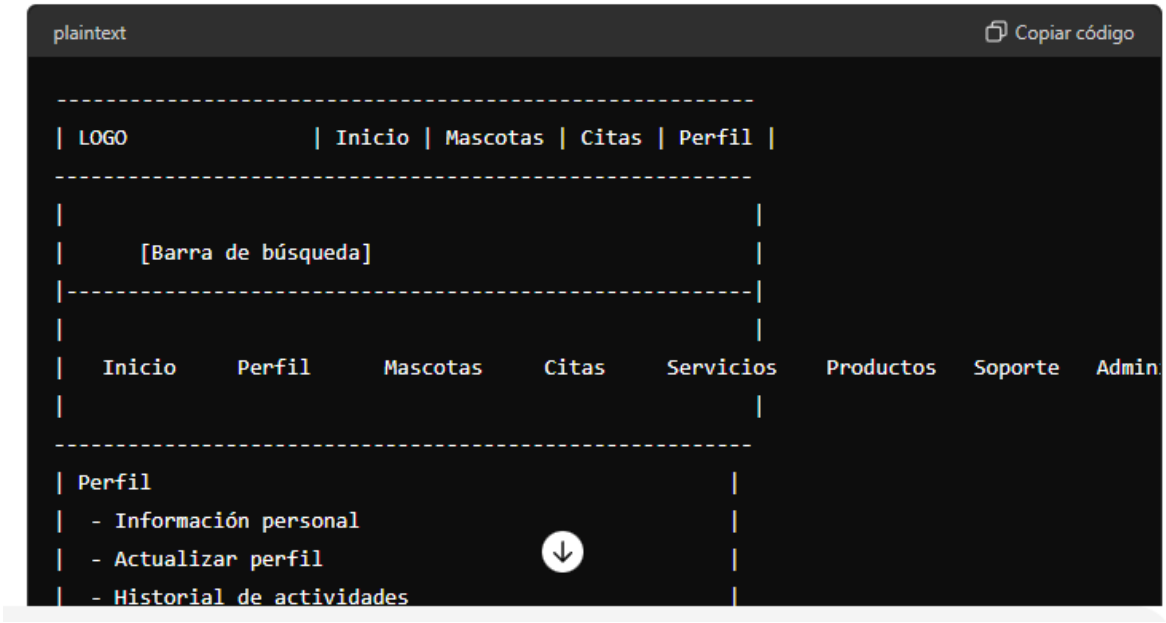
- **Feedback de los Usuarios:** Recoge y analiza feedback de los usuarios para identificar áreas de mejora en la navegación.
- **Pruebas de Usabilidad:** Realiza pruebas de usabilidad para asegurarte de que los usuarios pueden navegar fácilmente por el sitio.

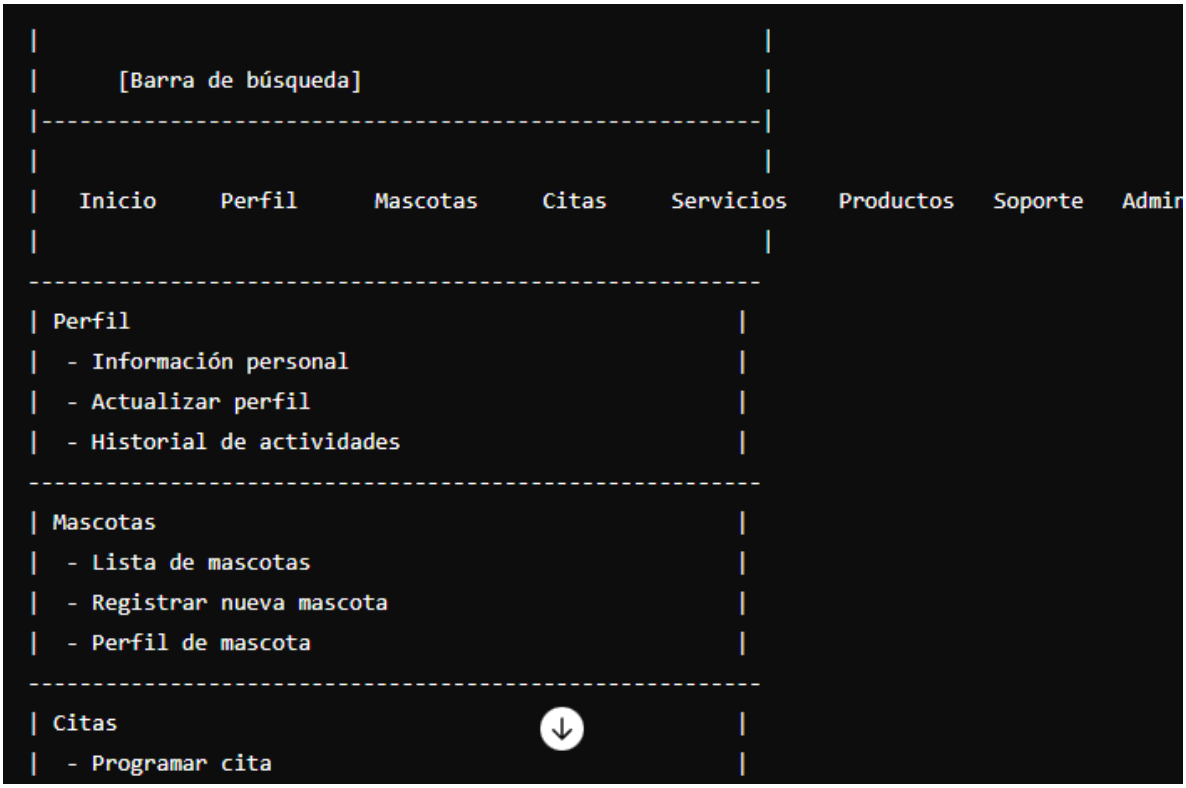
7. Elementos de Navegación Importantes

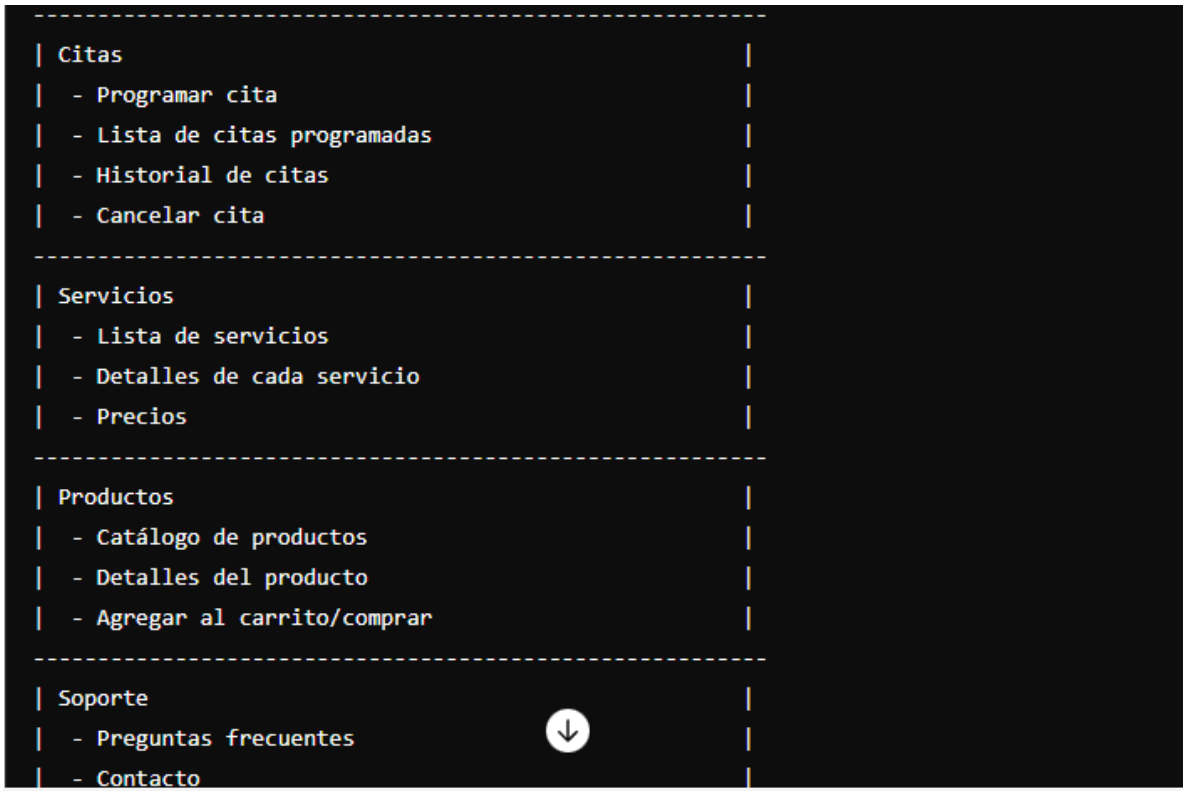
- **Barra de Navegación Superior:** Para las principales secciones del sitio.
- **Sidebar:** Para enlaces a funciones específicas y detalles adicionales, especialmente en páginas de perfil y administración.
- **Footer:** Para enlaces a información secundaria como términos y condiciones, política de privacidad y contacto.

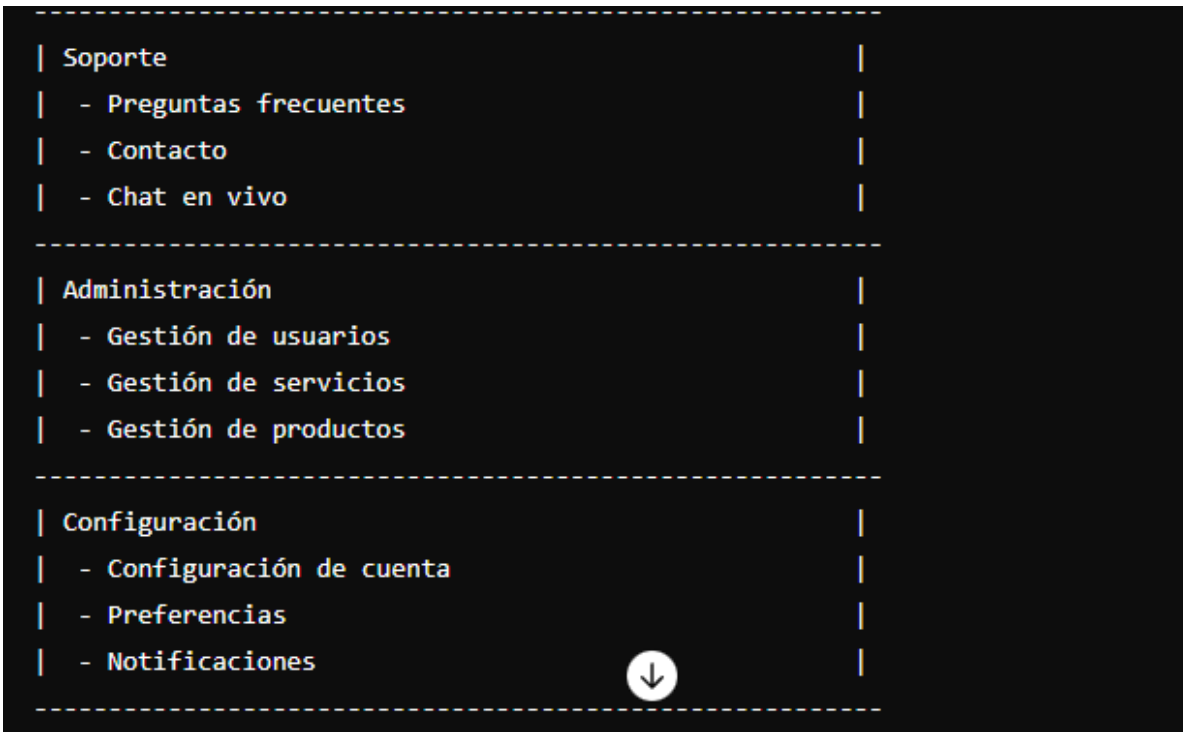
Ejemplo Visual de un Menú de Navegación

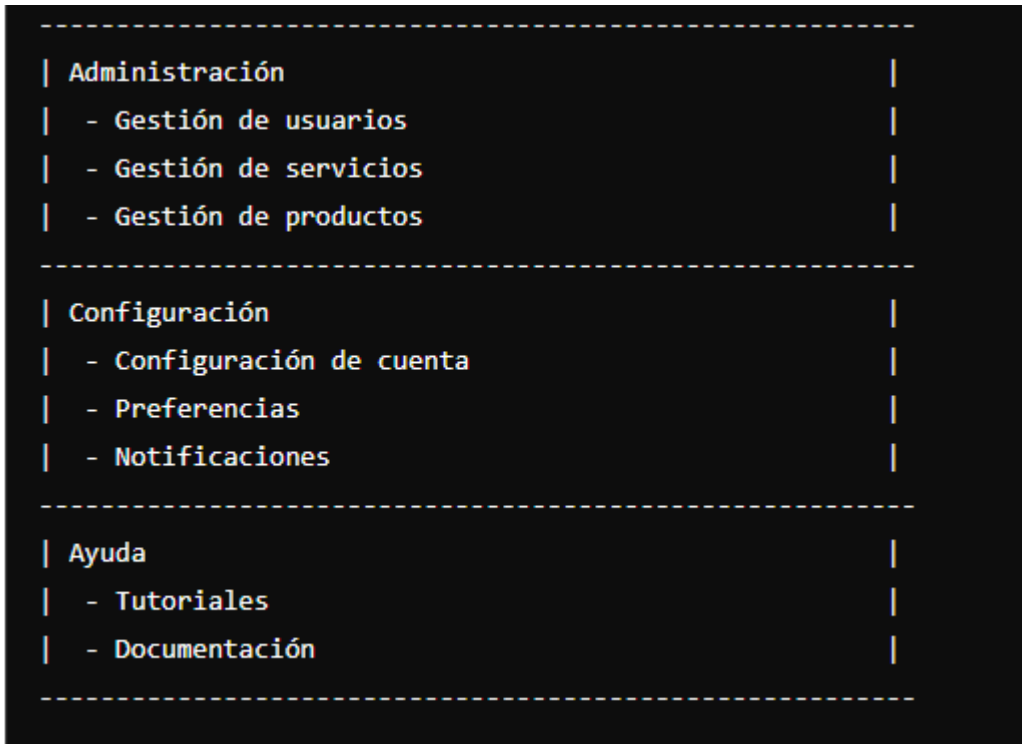
plaintext Copiar código






Citas
- Programar cita
- Lista de citas programadas
- Historial de citas
- Cancelar cita
Servicios
- Lista de servicios
- Detalles de cada servicio
- Precios
Productos
- Catálogo de productos
- Detalles del producto
- Agregar al carrito/comprar
Soporte
- Preguntas frecuentes
- Contacto


Soporte
- Preguntas frecuentes
- Contacto
- Chat en vivo
Administración
- Gestión de usuarios
- Gestión de servicios
- Gestión de productos
Configuración
- Configuración de cuenta
- Preferencias
- Notificaciones



Implementación Tecnológica

Para implementar este mapa de navegación, considera usar tecnologías y herramientas como:

- **React Router (para React) o Vue Router (para Vue):** Para gestionar la navegación en aplicaciones de una sola página.
- **CSS Frameworks:** Bootstrap o Tailwind CSS para facilitar el diseño responsivo y la consistencia de estilos.
- **Librerías de Componentes UI:** Material-UI (para React), Vuetify (para Vue) para componentes pre-construidos y estilizados.

ARQUITECTURAS

Diseñar una arquitectura sólida y escalable es crucial para el éxito de una aplicación de software web de cuidado de mascotas. Aquí tienes algunas recomendaciones importantes para la arquitectura de tu aplicación:

1. Arquitectura de Microservicios

- **Modularidad:** Divide tu aplicación en servicios independientes (usuarios, mascotas, citas, servicios, productos) que se pueden desarrollar, desplegar y escalar de forma independiente.
- **Comunicación:** Usa APIs RESTful o gRPC para la comunicación entre microservicios.
- **Base de Datos:** Cada microservicio puede tener su propia base de datos para garantizar la independencia y reducir el acoplamiento.

2. Arquitectura de N-Tier (Multicapa)

- **Presentación (Front-End):** Esta capa incluye la interfaz de usuario que interactúa con los usuarios.
- **Lógica de Negocio (Back-End):** Contiene la lógica de la aplicación y procesa las solicitudes de los usuarios.
- **Acceso a Datos:** Gestiona las interacciones con la base de datos.

3. Arquitectura Serverless

- **Funciones Lambda:** Usa servicios como AWS Lambda para ejecutar código en respuesta a eventos sin necesidad de gestionar servidores.
- **Servicios Gestionados:** Emplea servicios gestionados como AWS API Gateway para APIs, AWS S3 para almacenamiento de archivos y AWS DynamoDB para bases de datos NoSQL.

4. Arquitectura Basada en Eventos

- **Mensajería:** Implementa sistemas de mensajería como Apache Kafka o RabbitMQ para manejar la comunicación asíncrona entre servicios.
- **Eventos:** Diseña tu aplicación para que reaccione a eventos, lo que permite una mayor escalabilidad y flexibilidad.

5. Arquitectura de Contenedores

- **Docker:** Usa Docker para encapsular cada microservicio en un contenedor, garantizando consistencia en los entornos de desarrollo, prueba y producción.
- **Orquestación:** Utiliza Kubernetes para la orquestación de contenedores, gestionando el despliegue, la escalabilidad y la operación de los contenedores.

6. Seguridad y Autenticación

- **OAuth2 y JWT:** Implementa OAuth2 para la autenticación y autorización. Usa JSON Web Tokens (JWT) para gestionar las sesiones de los usuarios.
- **Cifrado:** Asegúrate de que todas las comunicaciones estén cifradas utilizando HTTPS/TLS.
- **Control de Acceso:** Implementa roles y permisos para gestionar el acceso a diferentes partes de la aplicación.

7. Desarrollo API-First

- **Swagger/OpenAPI:** Diseña tus APIs antes de desarrollar la lógica de negocio. Usa herramientas como Swagger para documentar y probar tus APIs.
- **Versionado de APIs:** Gestiona el versionado de tus APIs para garantizar la compatibilidad hacia atrás.

8. Escalabilidad y Desempeño

- **Autoescalado:** Configura autoescalado para tus servicios en la nube para manejar incrementos en la carga de trabajo.
- **Cache:** Usa caching con herramientas como Redis o Memcached para reducir la carga en la base de datos y mejorar la velocidad de respuesta.
- **Balanceo de Carga:** Implementa balanceadores de carga para distribuir el tráfico de manera uniforme entre tus servidores.

9. Monitorización y Logging

- **Logging Centralizado:** Implementa un sistema de logging centralizado con herramientas como ELK Stack (Elasticsearch, Logstash, Kibana) para rastrear y analizar logs.
- **Monitorización:** Usa soluciones de monitorización como Prometheus y Grafana para monitorear el desempeño de tu aplicación y detectar problemas en tiempo real.
- **Alertas:** Configura alertas para notificarte sobre problemas críticos de rendimiento o disponibilidad.

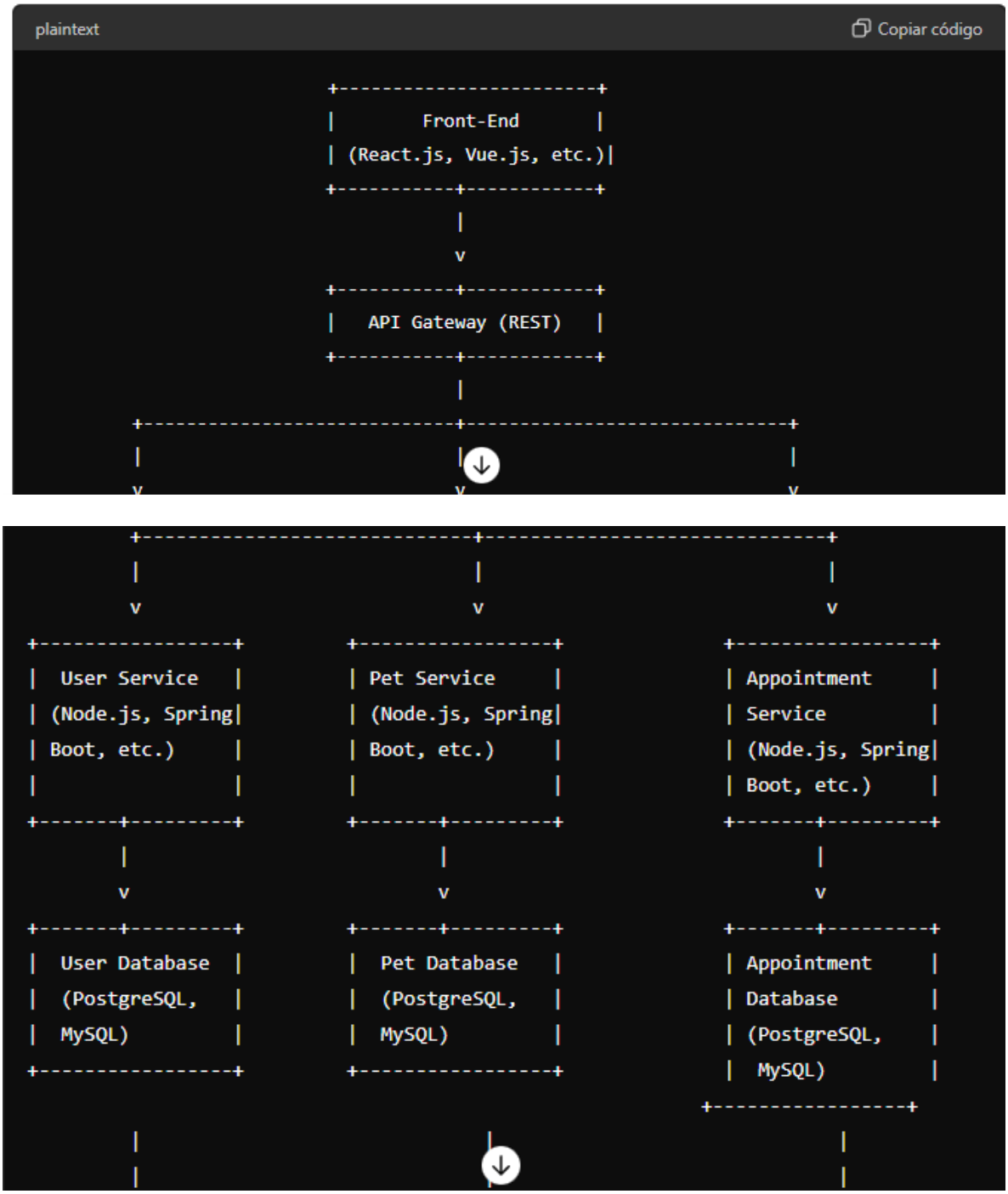
10. Despliegue Continuo e Integración Continua (CI/CD)

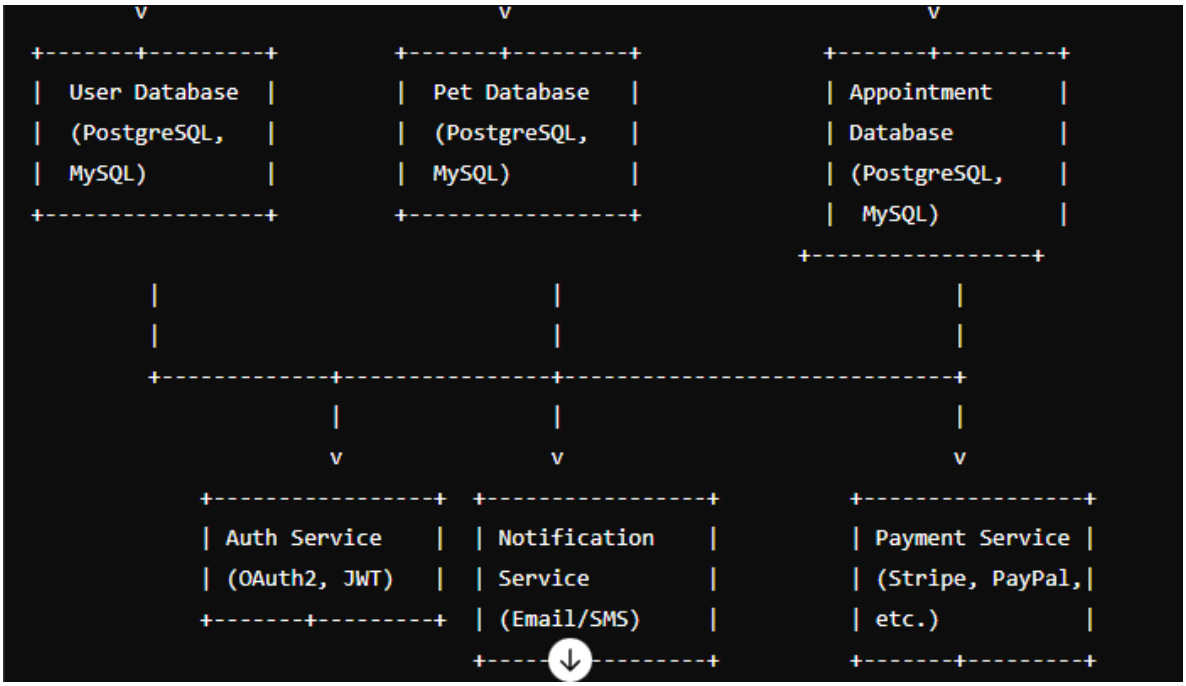
- **Pipelines de CI/CD:** Implementa pipelines de CI/CD con herramientas como Jenkins, GitLab CI, o GitHub Actions para automatizar el proceso de integración y despliegue.
- **Pruebas Automatizadas:** Asegúrate de que cada cambio en el código pase por un conjunto de pruebas automatizadas antes de ser desplegado.

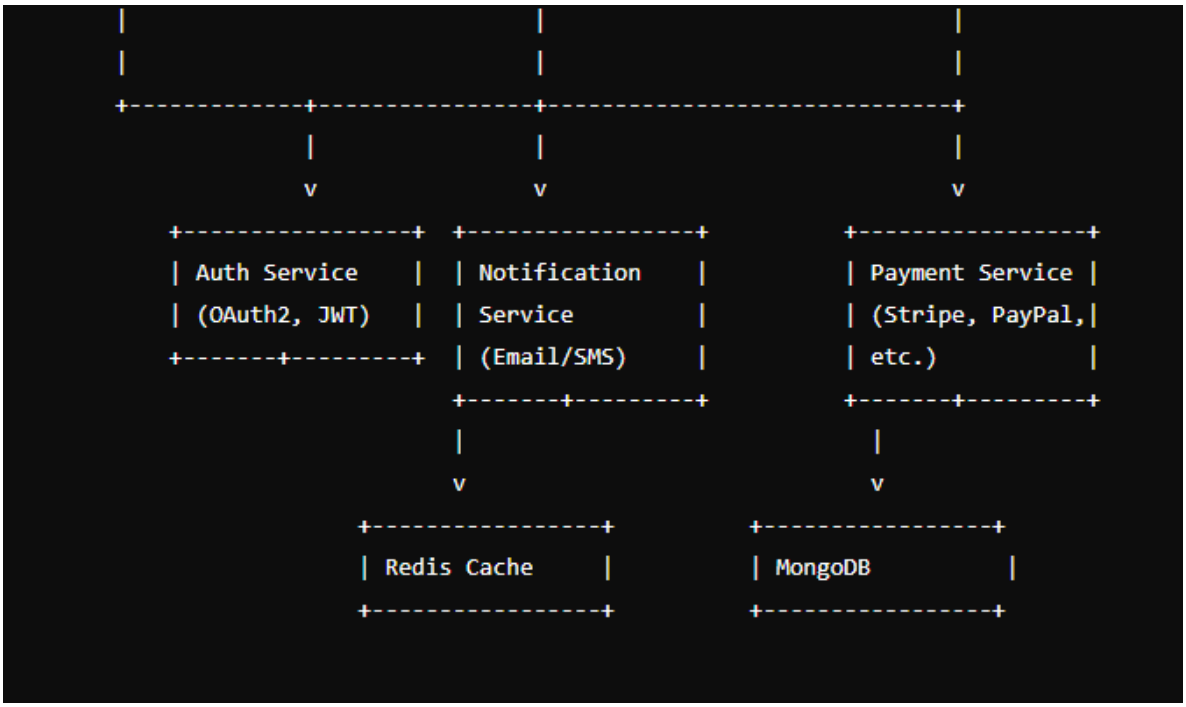
11. Almacenamiento de Datos

- **Relacional y NoSQL:** Usa bases de datos relacionales como PostgreSQL o MySQL para datos estructurados y bases de datos NoSQL como MongoDB o DynamoDB para datos no estructurados.
- **Backups:** Implementa políticas de backup y recuperación para proteger los datos contra pérdidas.

plaintext Copiar código




User Database (PostgreSQL, MySQL)
Pet Database (PostgreSQL, MySQL)
Appointment Database (PostgreSQL, MySQL)
Auth Service (OAuth2, JWT)
Notification Service (Email/SMS)
Payment Service (Stripe, PayPal, etc.)


Auth Service (OAuth2, JWT)
Notification Service (Email/SMS)
Payment Service (Stripe, PayPal, etc.)
Redis Cache
MongoDB

Descripción del Diagrama

1. **Front-End:**
    - Construido con frameworks como React.js o Vue.js para proporcionar una interfaz de usuario interactiva y responsiva.
2. **API Gateway:**
    - Actúa como un punto de entrada único para todas las solicitudes de cliente, dirigiéndolas a los servicios correspondientes.
3. **Microservicios:**
    - **User Service:** Gestiona la información de los usuarios, autenticación y autorización.
    - **Pet Service:** Gestiona la información de las mascotas, historial médico y detalles.
    - **Appointment Service:** Gestiona la programación y el historial de citas.
    - **Notification Service:** Maneja las notificaciones por correo electrónico o SMS.
    - **Payment Service:** Procesa los pagos para servicios y productos.
4. **Bases de Datos:**
    - Cada servicio tiene su propia base de datos, permitiendo una mayor independencia y escalabilidad.
5. **Servicios Complementarios:**
    - **Auth Service:** Utiliza OAuth2 y JWT para autenticación y autorización.
    - **Redis Cache:** Almacena en caché datos frecuentemente accedidos para mejorar el rendimiento.
    - **MongoDB:** Almacena datos no estructurados y documentos.

Implementación Tecnológica

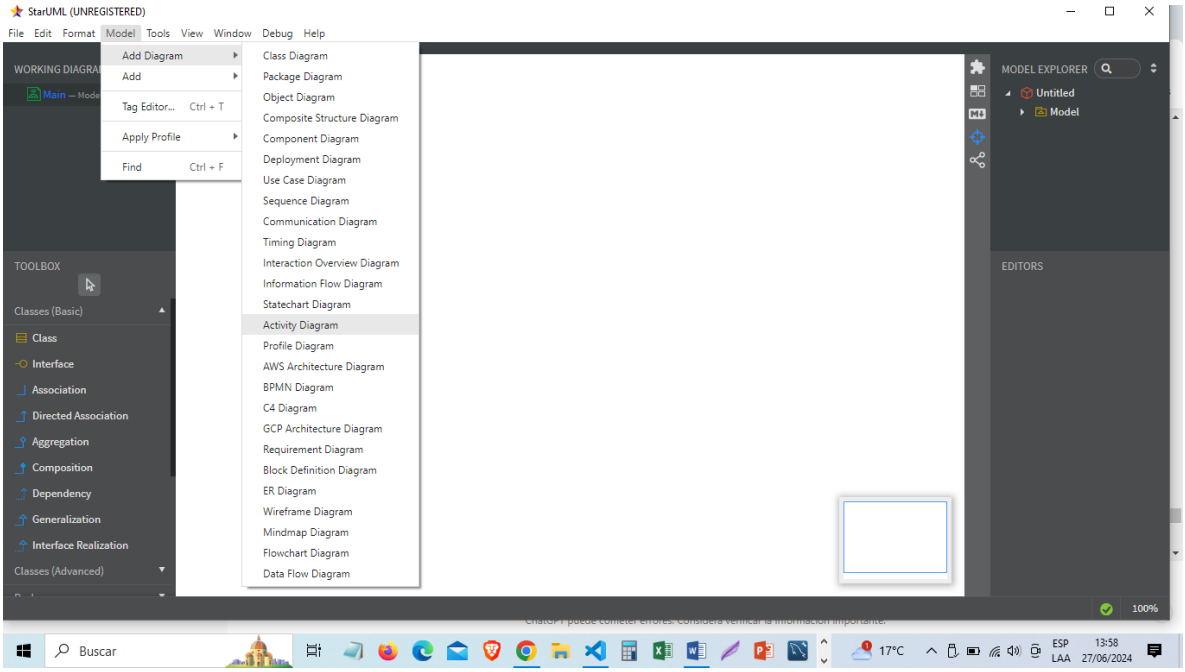
- **Frameworks Front-End:** React.js, Vue.js
- **Lógica de Negocio:** Node.js, Spring Boot
- **Bases de Datos:** PostgreSQL, MySQL, MongoDB
- **API Gateway:** Kong, AWS API Gateway
- **Cache:** Redis
- **Mensajería:** RabbitMQ, Apache Kafka
- **Orquestación de Contenedores:** Kubernetes
- **Monitorización:** Prometheus, Grafana
- **CI/CD:** Jenkins, GitLab CI, GitHub Actions

ACTIVIDADES DEL SISTEMA

DIAGRAMA DE ACTIVIDADES

Las actividades en una aplicación de software representan los flujos de datos y procesos dentro del sistema. Los diagramas UML de actividades son parte esencial de la etapa de diseño para modelar aquellos flujos de datos complejos, para el efecto tomemos algunos casos sencillos:

(Con StarUML)



Entre sus componentes básicos están:

Estado inicial y estado final – En el diagrama de actividades, los puntos de inicio y fin de un proceso son identificados por medio de su estado inicial y estado final.

Actividad o estado de acción – Consta de una única actividad que inicia una secuencia de acciones. Por ejemplo, cuando un usuario inicia sesión en su cuenta del sistema.

Acciones – Dentro de la actividad se ejecuta una acción o paso, bien sea a través del sistema o usuario para hacer una tarea. Continuando con el ejemplo anterior, un tipo de acción puede ser cuando el usuario verifica los datos de su perfil.

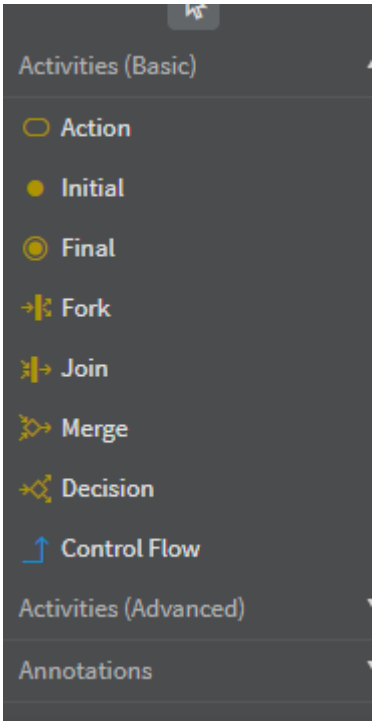
Objetos – Son materiales o datos, creados o que se utilizan en una actividad.

Decisiones – Son tomadas antes de continuar con otra acción o actividad. Las respuestas deben ser obligatoriamente ‘sí’ o ‘no’.

Sincronización – Consta de dos nodos, el nodo de bifurcación desde donde se ramifican los flujos concurrentes y el nodo de unión que recoge los flujos concurrentes en un solo flujo.

Señales – Indican la ejecución de acciones externas al sistema que pueden modificar la actividad. Un ejemplo claro sobre esto, es durante el pago en línea, cuando el usuario recibe como señal un código OTP para autorizar la transacción.

Carril – Se utilizan para agrupar por medio de columnas o categorías actividades relacionadas que son ejecutadas por actores diferentes.



ACTIVIDAD REGISTRO DE USUARIO

Pasos

1. Inicio
2. Introducir datos de registro de usuarios
3. Validar datos (datos completos y coherentes)
4. Crear una cuenta de Usuario
5. Confirmar Cuenta
6. Fin

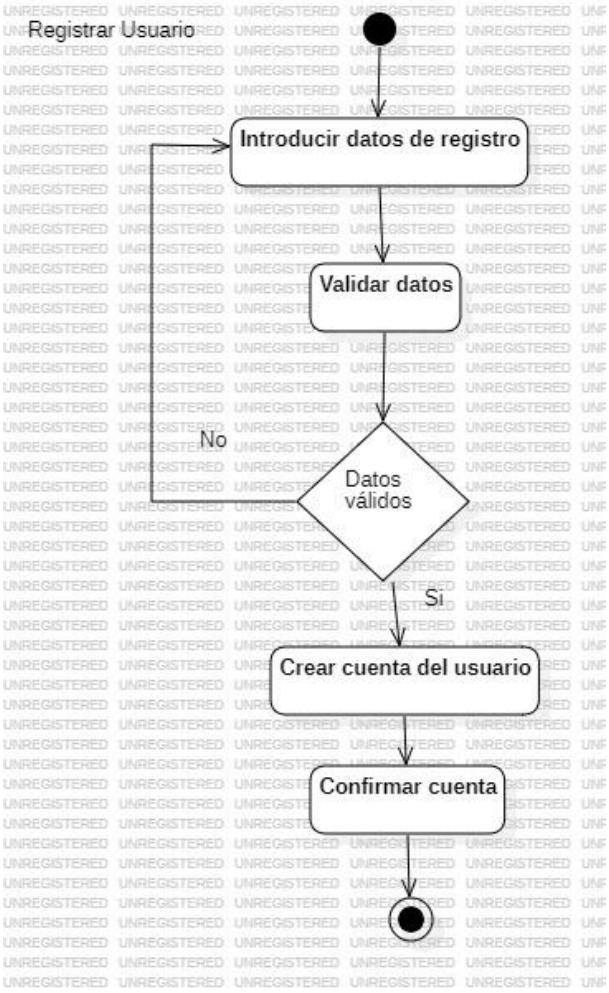


Figura1. Diagrama de flujo de actividades Registrar Usuario

ACTIVIDAD INICIAR SESIÓN

Pasos.

1. Aportar credenciales de usuario
2. Verificar credenciales
3. Crear una nueva sesión en el sistema
4. Abrir página o interfaz perfil del usuario

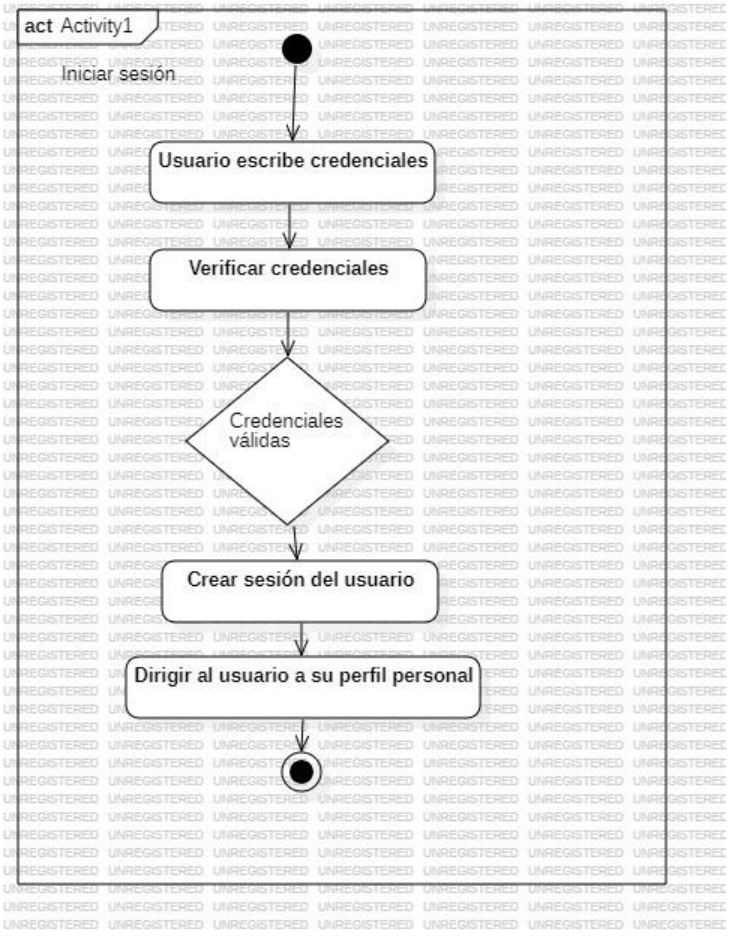


Figura 2. Diagrama de actividades Iniciar sesión en el sistema

ACTIVIDAD PROGRAMAR CITA CON VETERINARIO

Pasos.

1. Usuario inicia sesión

2. Selecciona fecha y hora de servicio

3. Verificar disponibilidad de servicios o médicos

4. Confirmar disponibilidad de servicios o médicos

5. Guardar datos de la cita

6. Confirmar cita al usuario

7. Notificar al veterinario

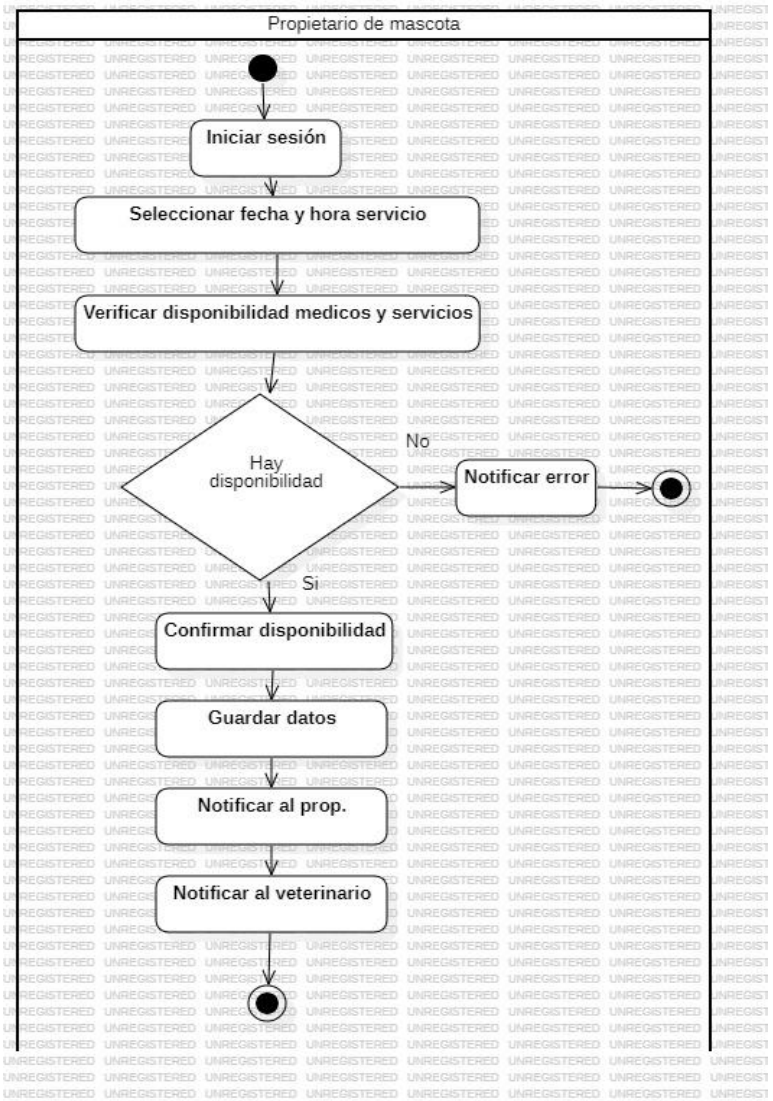


Figura 3. Diagrama de secuencia Programar cita de servicio médico

ACTIVIDAD AGREGAR MASCOTA

Pasos del usuario

1. Iniciar sesión
2. Aportar datos de la mascota
3. Validar los datos de la mascota
4. Guardar los datos

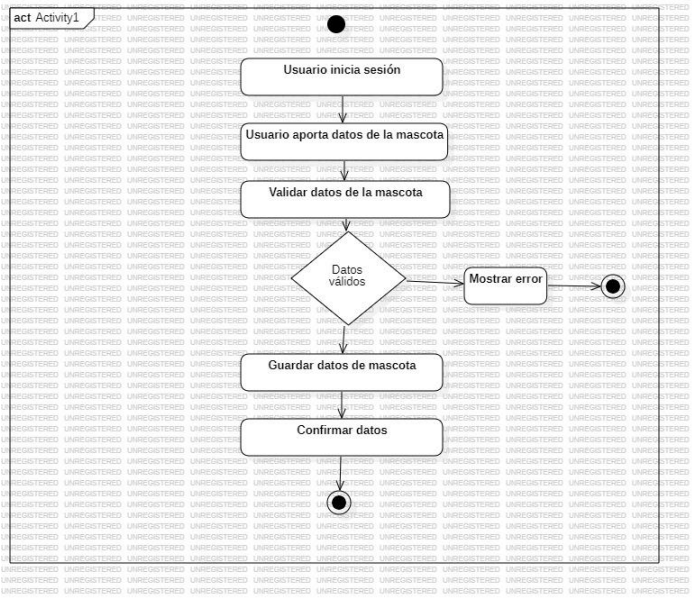


Figura 4. Agregar una mascota

ACTIVIDAD VISUALIZAR HISTORIAL MEDICO

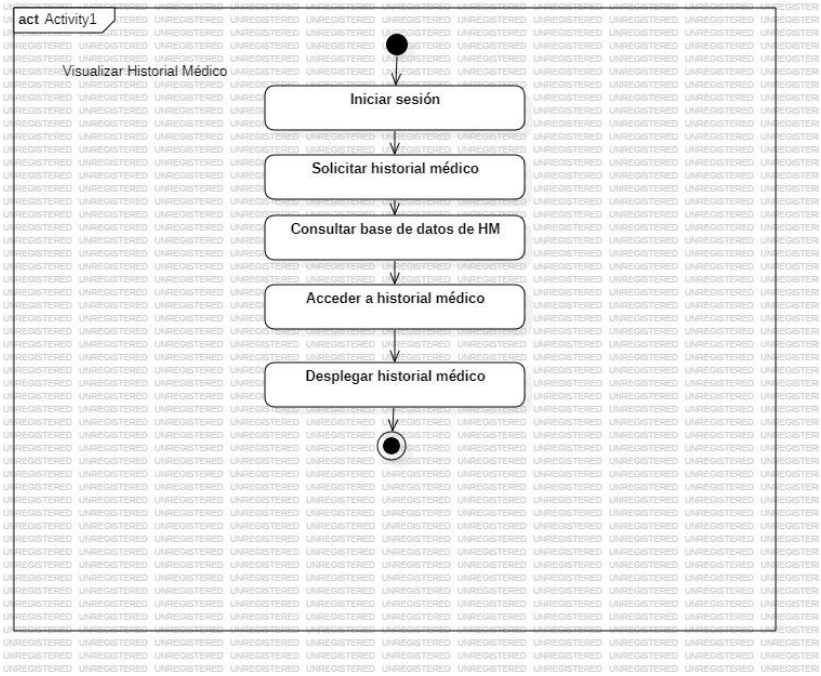


Figura 5. Diagrama de actividad Visualizar historial médico

METODOLOGIA DE DESARROLLO

METODOLOGÍAS ÁGILES CON SCRUM

Para emplear metodologías ágiles usando el marco de trabajo SCRUM es preciso saber de conceptos manejados en ese medio.

SCRUM es concepto de marco de trabajo que se usa para laborar en equipo, establecer metas con compromisos colaborativos, priorizados de entregas incrementales de producto definidas. No se usa solo en proyectos de software sino en otras actividades industriales manejables por etapas como la construcción y la producción.

- **Sprint**. Un Sprint es una unidad de tiempo establecida para cumplir una meta de trabajo
- **Historia de usuario.** Conceptos de los usuarios en torno a lo que esperan recibir de un producto de software
- **Épica**. Conjunto de historias de usuario relacionadas
- **Backlog**. Lista priorizada de los trabajos por hacer en el proyecto

Actores de un equipo Scrum

- **Product Owner.** Persona encargada de optimizar el valor del producto (de software en este caso). Representa a los clientes del producto, es el portavoz de todos los interesados del proyecto (stakeholders).
- **Scrum Master**. Es una persona que lidera la comunicación del equipo de trabajo facilitando las reuniones, resolviendo impedimentos.
- **Developmet Team**. Es el equipo de personas compuesto entre 3 y 9 expertos en el producto que se encargan de su desarrollo.

HISTORIAS DE USUARIO PROPUESTAS

1. REGISTRO Y VALIDACION DE USUARIOS

Historia	Descripción	Para
US1	Como usuario, quiero registrarme en la aplicación.	Poder acceder a sus funcionalidades más importantes.
US2	Como usuario, quiero iniciar sesión en el sistema.	Acceder a mi perfil y gestionar mis mascotas.
US3	Como usuario, quiero restablecer mi contraseña en caso de un olvido	Gestionar la recuperación de mis datos.
Criterios de aceptación	• El usuario debe poder introducir sus datos personales, su email y contraseña. • El sistema debe validar que el email no esté registrado previamente. • El sistema debe enviar un correo de confirmación al email del usuario. • El usuario debe ver un mensaje de confirmación tras completar el registro	
Tareas relacionadas	• Crear formulario de registro en una interfaz gráfica. • Implementar validación de datos en el backend del sistema. • Integrar sistema de envío de correos. • Desarrollar la lógica de confirmación del registro.	
Prioridad	Alta	

2. GESTION DE LAS MASCOTAS

Historia	Descripción	Para
US4	Como usuario, quiero añadir una nueva mascota a mi perfil	Administrar su información.
US5	Como usuario, quiero editar la información de mis mascotas.	Mantener sus datos actualizados según necesidades.
US6	Como usuario, quiero eliminar una mascota de mi perfil.	Actualizar datos en caso de no contar con la mascota
Criterios de aceptación	• El usuario debe ingresar con sus credenciales al sistema • El usuario debe tener acceso a todos los datos de sus mascotas	
Tareas relacionadas	• Autenticación de usuarios en el sistema	

	• Acceso a datos relacionados del usuario logueado
Prioridad	Alta

3. AGENDA O PROGRAMACION DE CITAS

Historia	**Descripción**	**Para**
US7	Como usuario, quiero programar una cita con el médico veterinario.	Llevar a mi mascota a una revisión médico veterinaria regular o en caso de ser necesario.
US8	Como usuario, quiero ver la disponibilidad de los veterinarios.	Conocer y seleccionar una fecha y hora adecuada.
US9	Como usuario, quiero recibir recordatorios de mis citas.	Recibir la información oportuna de los compromisos con la mascota.
Criterios de aceptación	•	
Tareas relacionadas	•	
Prioridad	Alta	

4. HISTORIAL MEDICO

Historia	Descripción	Para
US10	Como usuario, quiero ver el historial médico de mis mascotas.	Estar informado de las visitas y tratamientos recibidos o pendientes
US11	Como usuario, quiero añadir notas al historial médico de mi mascota.	Mantener un registro detallado.
Criterios de aceptación	•	
Tareas relacionadas	•	
Prioridad	Alta	

5. NOTIFICACIONES

Historia	Descripción	Para
UH12	Como usuario, quiero recibir notificaciones de vacunas y citas	Procurar la salud de la mascota.
Criterios de aceptación	•	
Tareas relacionadas	•	
Prioridad	Alta	

6. PERFIL DEL USUARIO

Historia	Descripción	Para
UH13	Como usuario, quiero actualizar mi información personal.	Mantener mi perfil personal al día.
Criterios de aceptación	•	
Tareas relacionadas	•	
Prioridad	Alta	

PLANIFICACIÓN DE SPRINTS

El **sprint** es una forma ordenada de hacer las tareas del proyecto de una forma priorizada de mayor a menor, suele ocupar entre 2 y 4 semanas.

Sprint 1: Registro y Autenticación

Historia 1: Registro de usuario

Historia 2: Inicio de sesión

Historia 3: Restablecimiento de contraseña

Sprint 2: Gestión de Mascotas

Historia 4: Añadir mascota

Historia 5: Editar información de mascota

Historia 6: Eliminar mascota

Sprint 3: Programación de Citas

Historia 7: Programar cita

Historia 8: Ver disponibilidad de veterinarios

Historia 9: Recordatorios de citas

Sprint 4: Historial Médico

Historia 10: Ver historial médico

Historia 11: Añadir notas al historial médico

Sprint 5: Notificaciones y Perfil de Usuario

Historia 12: Notificaciones de vacunas y citas

Historia 13: Actualizar información de usuario

Product Owner (PO): Profesional responsable de definir las historias de usuario y priorizar el Product Backlog.

Scrum Master: Facilita el proceso SCRUM, elimina impedimentos y asegura que el equipo sigue las prácticas ágiles.

Development Team: Equipo multifuncional que desarrolla la aplicación. Incluye desarrolladores de software, diseñadores, analistas, diseñadores gráficos, líderes de calidad, probadores (tester), entre otros.

Eventos SCRUM

Sprint Planning: Reunión al inicio de cada sprint para planificar el trabajo a realizar.

Daily Standup: Reuniones diarias cortas para discutir el progreso, planificar el trabajo del día y resolver impedimentos.

Sprint Review: Reunión al final de cada sprint para demostrar el trabajo completado al Product Owner y a los stakeholders.

Sprint Retrospective: Reunión para reflexionar sobre el sprint que acaba de terminar e identificar mejoras para el próximo sprint.

Software de apoyo administrativo

- **Jira**: Para la gestión de proyectos y seguimiento de tareas.
- **Trello**: Para gestionar el backlog y el progreso de las historias de usuario.

DESPLIEGUE DE LA APLICACIÓN

El despliegue de una aplicación es el proceso de implementar o instalar el software desarrollado en los servidores, equipos o redes de datos necesarios para que esté disponible al usuario final, si la aplicación es de tipo escritorio local se deberá contar con la infraestructura de equipos y redes de datos para que pueda ser usado finalmente. Si es una aplicación web que se ejecute en un navegador en internet habrá que dispones de servicios de conexión de equipos en redes conectadas a internet y gestionar software de comunicaciones para la implementación de los servicios.

Para el efecto nuestro consideramos una aplicación web con arquitectura de cliente servidor en varias capas y se necesita de protocolos de comunicaciones en red como HTTP, HTTPS y TCP para las conexiones entre equipos remotos. Los temas de redes y comunicaciones son rigurosos y están fuera del alcance de este proyecto y solo mencionaremos tecnologías como los protocolos, los equipos y software necesario que intervienen en el lado cliente y servidor de la aplicación.

https://www.youtube.com/watch?v=NSB0ATJUavA

DESCRIPCION DE DESPLIEGUE PARA UN SOFTWARE WEB

Frontend: es un concepto usado en las aplicaciones que hacen referencia a la parte del sistema de información que el usuario final utiliza. Aquí se determina que se necesita un tipo de equipo de cómputo que tenga un navegador web compatible con el estándar HTML5, caso de Google Chrome, Mozilla Firefox y otros más: El cliente web es el equipo y software que accede a la aplicación.

Backend: Corresponde a los equipos y software donde residen las bases de datos y el software de la aplicación.

Servidor de Aplicaciones: Un equipo que ejecuta la lógica del negocio de la aplicación usando Python u otro lenguaje del lado backend (Python con la ayuda de frameworks como por ejemplo, Flask o Django).

API RESTful: Las API proporcionan código para comunicar dos aplicaciones diferentes entre si a través de puertos lógicos conocidos como *endpoints* para interactuar con el frontend y manejar solicitudes.

Bases de Datos

MySQL: Base de datos relacional para almacenar datos estructurados como información de usuarios y citas.

MongoDB: Base de datos NoSQL para almacenar datos no estructurados como el historial médico de las mascotas.

Infraestructura

Servidor Web: Maneja solicitudes HTTP/HTTPS y las redirige al servidor de aplicaciones.

Balanceador de Carga: Distribuye el tráfico entrante entre múltiples servidores de aplicaciones para escalabilidad y alta disponibilidad.

Redes y Firewalls: Aseguran la comunicación segura y eficiente entre los componentes.

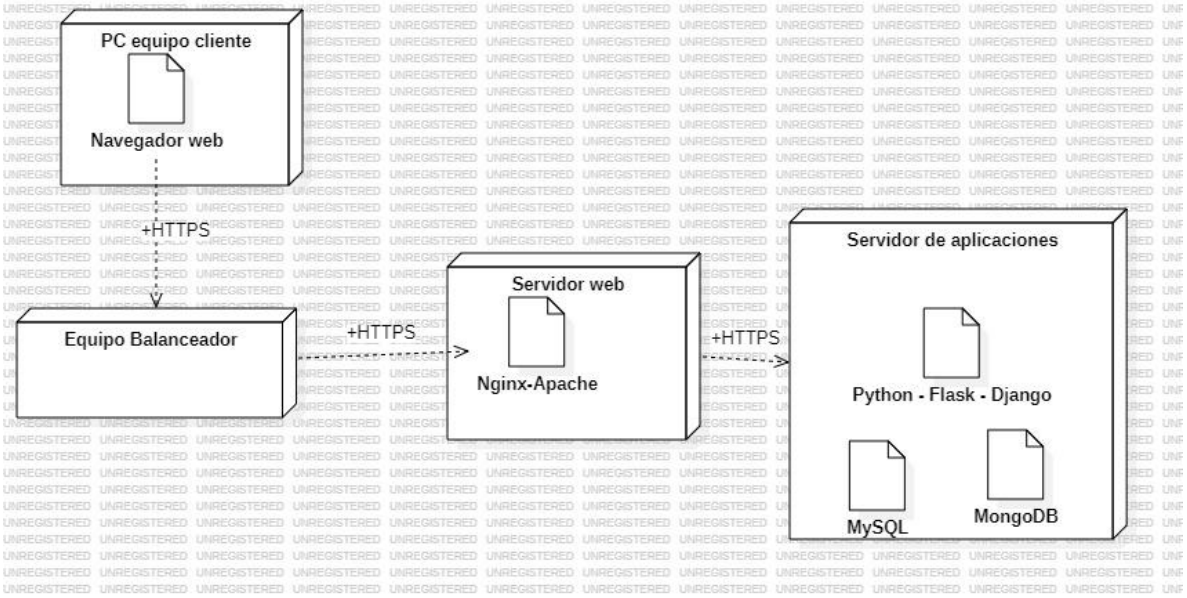


Figura 6. Diagrama de despliegue software cuidado de mascotas con Python MySQL y MongoDB.

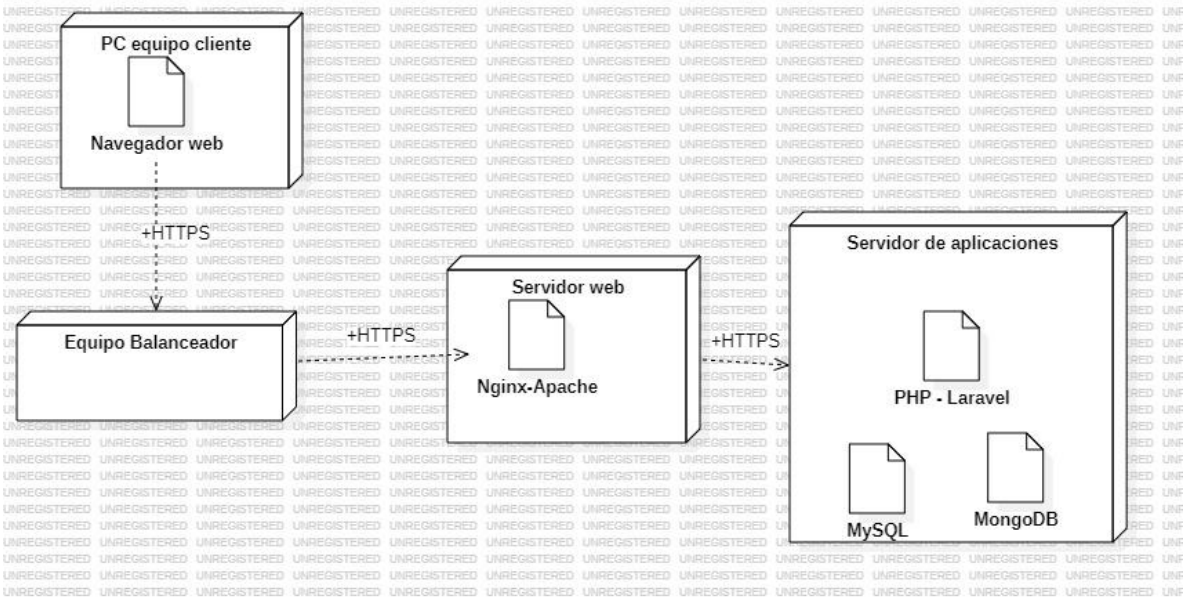


Figura 7. Diagrama de despliegue software cuidado de mascotas con PHP, Laravel 10 MySQL y MongoDB.

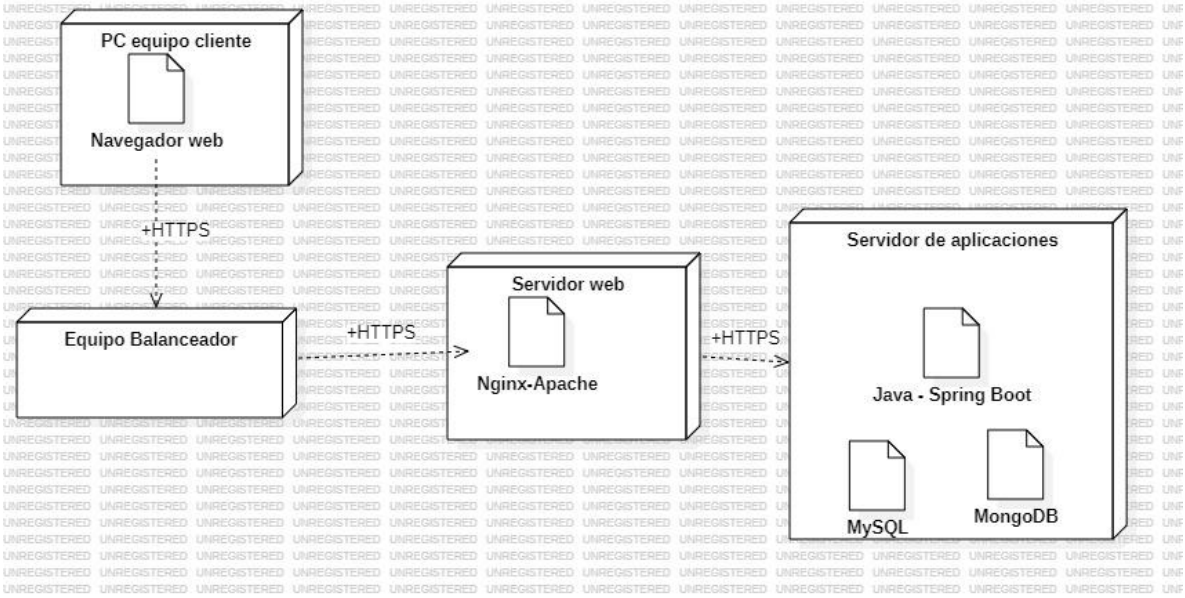


Figura 7. Diagrama de despliegue software cuidado de mascotas con Java – Spring Boot MySQL y MongoDB.

Bibliografía:

colaboradores de Wikipedia. (2024, 24 enero). Lenguaje unificado de modelado. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

colaboradores de Open AI. (2024, 24 enero). Modelado de aplicación web para cuidado de mascotas.

https://chatgpt.com/c/68da6e20-7f7a-4a35-aec9-d2fcae66e263

Ang, J. (2022, mayo 16). Cómo crear un diagrama de actividades [+Ejemplos]. Venngage Blog. https://es.venngage.com/blog/diagrama-de-actividades/

Tutorial de diagrama de actividades UML. (s. f.). Lucidchart. https://www.lucidchart.com/pages/es/tutorial-diagrama-de-actividades-uml

Scrum: roles y responsabilidades. (s. f.). Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html

Diagrama de despliegue UML: Qué es y cómo hacerlo | Miro. (s. f.). https://miro.com/. https://miro.com/es/diagrama/que-es-diagrama-despliegue-uml/