Universidad de Costa Rica

Escuela de Ciencias de Computación e Informática

CI-0128 Proyecto Integrador de Ingeniería de Software y Bases de Datos

Profesores:

María Isabel Murillo Quintana

Luis Antonio Mata Reyes

Conceptualización

Integrantes:

Alejandro Pacheco Rojas C05765

Ernesto Delgado Páez C22554

Paola Feng Wu C22884

Valentino Vidaurre Rodríguez C28377

II Semestre 2024

Definición del problema

La clínica veterinaria El Bigote está en un rápido periodo de crecimiento, por lo que requiere de una aplicación web y nuestro equipo ha sido contratado para desarrollarla. El Bigote se ha vuelto la veterinaria más popular del pueblo, y la voz ha empezado a correrse, por lo que no sería sorpresa si en unos meses deciden abrir nuevas clínicas, o empezar a recibir distintos tipos de animales (por ahora solo permiten perros y gatos).

Dado que es una veterinaria de pueblo, El Bigote no pretende dejar a ninguno de sus clientes atrás. Desde don Jairo, vecino que a sus 72 años está perdiendo la vista y pasa sus días con su perrita Lulú, hasta María, la brillante informática que está en tercer año de su doctorado en inteligencia artificial cuya única compañía es su gato Misifus. Es por esto que la aplicación debe de tener una opción de voice-over que ayude a las personas con visibilidad limitada, mientras que para usuarios más experimentados, debe de ofrecer opciones de búsqueda avanzadas.

Va a existir una entidad con poder “supremo” llamado Dueño. Dicho Dueño debe de poder crear instancias de la entidad Veterinaria. Dentro de una Veterinaria pueden existir instancias de cinco entidades: Administrador, Veterinario, Cliente, Mascota y Cita. El Dueño será el encargado de crear y borrar tanto entidades Veterinaria como administradores de ellas, además de “mover” un Administrador o Veterinario de una Veterinaria a otra en caso de ser necesario. El Dueño además debe de poder realizar cualquier función que esté disponible para cualquier otra entidad.

Cada una de las entidades de la Veterinaria posee permisos y funciones adaptados a sus necesidades. Todas deben de poder iniciar sesión menos Mascota y Cita. La persona Administrador es la encargada de exactamente una instancia de Veterinaria. Toda Veterinaria debe de tener al menos un Administrador. Podrá agregar veterinarios, modificar los servicios ofrecidos y la tarifa a cobrar por ellos, supervisar la información de todas las personas registradas en la veterinaria, así como añadir y/o eliminar tanto Cliente (con su mascota) como Veterinario. También podrá ver la información de las citas agendadas por el cliente. Por su parte los

Veterinario deben de poder verificar los historiales e informaciones personales de sus pacientes, así como modificarlos. También pueden ver las citas que han sido agendadas por sus clientes. Finalmente, las entidades Cliente por su parte podrán agendar citas, ver el historial médico de sus mascotas, realizar pagos, comprobar, modificar citas y su información personal. La entidad Cita no tendrá permisos al igual que Mascota.

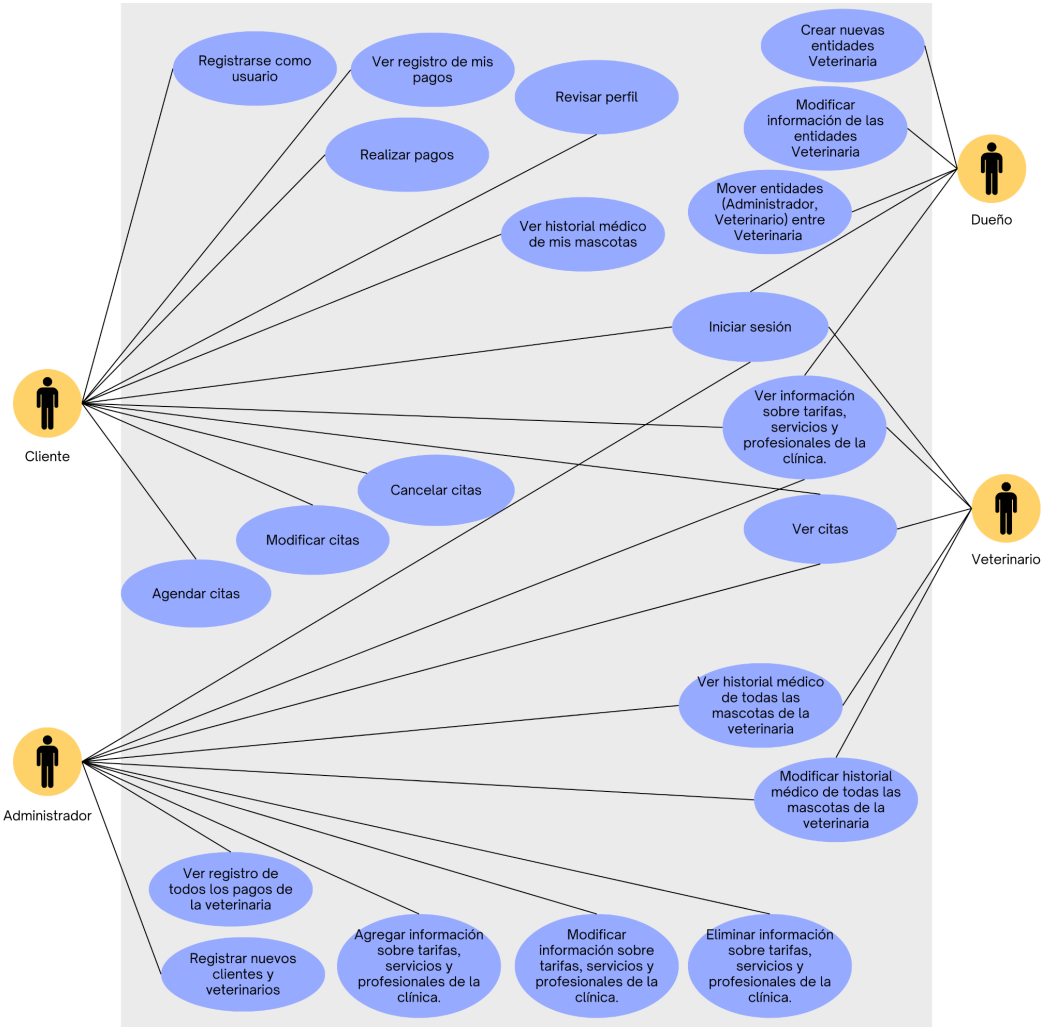
En cuanto al apartado técnico, para backend se va a utilizar Oracle para la base de datos, y se utilizará Django como framework. Por otro lado, para el frontend se va a utilizar JavaScript, CSS, HTML y Bootstrap. Preliminarmente la página va a estar alojada en una computadora de la ECCI, aunque se contempla que esto puede cambiar.

Alcance del proyecto

Objetivo:

- Desarrollo de una aplicación web con gestión de usuarios, citas, pagos e historial médico. Además proporcionará administración de clínicas, veterinarios y clientes.
- Implementación de una base de datos para el almacenamiento de información.
- Interfaz visualmente atractiva y accesible desde los navegadores Firefox, Chrome, Edge y sistemas operativos iOS y Android.
- Integración de pagos en línea mediante APIs seguras.
- Implementación de características de seguridad web y usabilidad.
- Realizar reportes de pruebas y controles de calidad.

Funcionalidad del producto:



Recursos:

- Equipo (4 personas): 9 horas de trabajo por 16 semanas

Cronograma:

- 11 de septiembre: fin de iteración 0 (user personas, conceptualización, pila de producto, prototipo)
- a definir: otras iteraciones
- 30 de noviembre: fin de lecciones

Entregables:

- Una aplicación web que cumpla con los requerimientos especificados.
- Documentación en la forma de documentos pdf con las evidencias y explicaciones de los entregables solicitados para cada iteración.

Fuera del alcance:

- El equipo no va implementar de forma manual los aspectos de seguridad. En vez, se van a integrar APIs para cumplir con estos requerimientos.
- Desarrollo de una aplicación móvil o de desktop.
- Integración con sistemas externos no mencionados.
- Mantenimiento o actualizaciones posteriores al lanzamiento inicial.
- Desarrollo de manejo de *payroll*, es decir, no se encargará de gestionar el salario pagado a los trabajadores de las veterinarias.

Requisitos del sistema

Requisitos funcionales

Gestión de Usuarios:

1. **RF1**: El sistema debe permitir a los usuarios clientes registrarse proporcionando: nombre, primer apellido, correo electrónico y contraseña. En caso de no proporcionar al menos uno de los anteriores, no permite terminar el registro.
2. **RF2**: Los usuarios registrados deben poder iniciar sesión ingresando su correo y contraseña. Si el usuario no tiene disponible la contraseña de su cuenta, se le enviará un correo al email con el que creó su cuenta con un enlace para verificar su identidad. Desde dicho email podrá cambiar la contraseña de su cuenta.
3. **RF3:** Debe haber diferentes roles de usuario: dueño, administrador, veterinario y cliente, con diferentes privilegios de acceso: el dueño tendrá permisos de agregar y modificar cualquier información de cualquier veterinaria, el administrador tiene acceso a la información comercial de su veterinaria asignada y las personas afiliadas a ellas, el veterinario tiene acceso a la información médica de su veterinaria y las mascotas afiliadas a ella, y el cliente solo a la información de su propio perfil.

Gestión de Citas:

4. **RF4:** Los clientes deben poder buscar y solicitar citas disponibles en función de (al menos una de:) fecha, tipo de consulta (nueva, seguimiento y revisión) y disponibilidad del veterinario.
5. **RF5:** Los clientes deben poder cancelar citas. Si cancelan una cita con menos de 24 horas de aviso se les hará un cobro de $5 por penalización.
6. **RF6:** Los clientes deben de poder reagendar o modificar los detalles de citas antes de las 24 horas previas a la cita. Al pasarse dentro del rango de 24 horas, no podrán realizar cambios.
7. **RF7:** Los clientes podrán llegar a la veterinaria sin citas y, según disponibilidad, ser atendidos en casos de emergencias.

8. **RF8:** El sistema debe enviar recordatorios por correo electrónico a los clientes 24 horas antes de sus citas.
9. **RF9:** Los veterinarios deberán poder acceder a la información de las citas que fueron agendadas por los clientes.
10. **RF10:** Los administradores de las veterinarias deberán poder acceder a la información de las citas que fueron agendadas por los clientes.

Gestión de Pagos:

11. **RF11:** Los usuarios deben de poder realizar pagos en línea con las tarjetas aceptadas (Visa, Mastercard y American Express) de manera segura mediante un API que contenga *transaction*, *tokenization* y *encryption* que conforma al PCI DSS. Algunas opciones son las de: Stripe, PayPal, Square, Authorize.net.
12. **RF12:** El sistema debe de mantener un registro de los pagos realizados por los usuarios mediante un log de facturas, números de comprobantes (si lo incluye el API), y fecha del pago.
13. **RF13:** Se debe de asociar cada pago a una cita o a un servicio rendido mediante un número de referencia.
14. **RF14:** El administrador debe poder ver el historial de pagos de toda la veterinaria (es decir, de los diferentes clientes que atendieron).
15. **RF15:** En caso de que un cliente pague en efectivo, se debe de permitir a los administradores ingresar una nueva entrada al registro de pagos con los detalles correspondientes (factura, comprobante, monto, nombre, fecha, medio).
16. **RF16:** Los usuarios podrán ver únicamente su propio historial de pago. Incluye el monto, la fecha, el medio de pago (tipo de tarjeta o efectivo si fué hecho directamente en la veterinaria) y una breve descripción del servicio brindado.

Gestión de Historial Médico:

17. **RF17:** Los veterinarios deben poder registrar y consultar mediante un filtro y un buscador el historial médico de las mascotas. Las cinco categorías grandes son (de la parte médica) las vacunas, tratamientos, diagnósticos, (y de la parte administrativa) dueño y fecha.

18. **RF18:** El filtro debe de poder aislar procedimientos según las 5 gran categorías pero también las siguientes subcategorías: tratamiento repetido (cuando tiene que ir cada cierto tiempo), tratamiento único, diagnóstico crónico, diagnóstico agudo (no es crónico), diagnóstico grave, diagnóstico leve, diagnóstico de riesgo medio, tratamientos/vacunas preventivas, tratamientos/vacunas en respuesta de un diagnóstico, cirugía, tratamientos no invasivos (no quirúrgicos) y medicamentos administrados.
19. **RF19:** Los resultados del filtro serán paginados y se usará la paginación Django. En caso de no ser posible, se usará la de Bootstrap.

Gestión de Información:

20. **RF20:** El sistema debe permitir a los administradores agregar nueva información sobre servicios, tarifas, trabajadores (médicos, asistentes, recepcionistas) de la clínica.
21. **RF21:** Los administradores podrán, en lo posible, deshabilitar/eliminar información obsoleta sobre la veterinaria (servicios, tarifas, profesionales).
22. **RF22:** Se debe permitir a los administradores modificar información sobre la veterinaria (servicios, tarifas, profesionales) mientras todavía sea marcado como vigente.
23. **RF23:** Se debe de mantener registro de la existencia de cualquier información marcada como obsoleta por razones legales. Similarmente para la información vigente, se respaldará cada mes y se mantendrá un registro vigente de los últimos 10 años.
24. **RF24:** Los administradores podrán filtrar los clientes de su veterinaria por pagos independientes, pagos totales, servicios brindados, fecha y mascota.

<u>Requisitos no funcionales:</u>

Usabilidad:

1. **RNF1:** La interfaz de usuario debe ser accesible mediante opciones de agrandar la letra, tener opción de *voiceover*, y las imágenes deben de tener texto alternativo. La página debe de ser accesible desde iOS versión 17.6.1, Android versión 14 y desde los navegadores Firefox, Chrome, Edge.

2. **RNF2:** El tiempo de carga de las páginas debe de ser menor de 5 segundos para una experiencia de usuario fluida. Si no carga dentro del tiempo especificado, la página debe de mostrar un mensaje de error diciendo "tuvimos un problema de carga lo sentimos".

Seguridad:

3. **RNF3:** La aplicación debe implementar two-factor authentication mediante un usuario y contraseña (contiene al menos 8 caracteres, un número y un carácter especial) al igual que un código enviado al correo o un verificador móvil para proteger la información confidencial de los usuarios (los datos de pago y detalles personales). La autenticación se pide una sola vez al iniciar sesión.
4. **RNF4:** Se debe de implementar SSL, validación de input y least-privilege access para prevenir ataques de inyección de SQL

Plan de calidad

Los estándares que se seguirán en el proyecto serán:

1. Se utilizará el linter sonarlint como herramienta de análisis estático.
2. Se seguirá la convención Camel Case para nombrar variables y funciones.
3. Los nombres de las variables y funciones serán en inglés.
4. Seguiremos los principios de SOLID (Single Responsibility, Open/Closed, Liskov's Substitution, Interface Segregation, Dependency Inversion).
5. Los nombres de las variables serán singulares, compuestos por al menos un sustantivo y puede que contengan algún adjetivo, mientras que los nombres de las funciones deben de contener algún verbo.
6. En el caso de SQL las palabras reservadas siempre irán en mayúsculas y las tablas en plural.

La manera en que nos aseguraremos de respetar estos y garantizar que se cumpla con los requisitos así cómo la calidad será a través de:

1. Revisiones de los pushes por parte de otro(s) integrantes del grupo (pair programming), no se harán pushes cooperativos.
2. Reuniones semanales con los product owners conforme se va avanzado en el desarrollo para verificar que las funcionalidades vayan conforme la visión del cliente.
3. Utilizaremos la plataforma GitUCR para mantener un control de versiones.
4. En cuanto al propio uso de la plataforma GitUCR habrán ramas divididas por funcionalidades (historias de usuario) y una de "desarrollo" donde se estarán probando antes de integrar. Una vez que una funcionalidad se considere "terminada" se integrará a la rama principal.
5. Utilizaremos la plataforma Jira para mantener un seguimiento de cómo va el proyecto.
6. Se seguirá la metodología Scrum.