Instituto Tecnológico de Costa Rica

Proyecto #2

Lenguajes de Programación

Jose Pablo Quirós Hidalgo - 2020054485

Alberto Zumbado Abarca - 2020095172

Profesora: María Auxiliadora Mora.

Tabla de contenidos.

---

incio

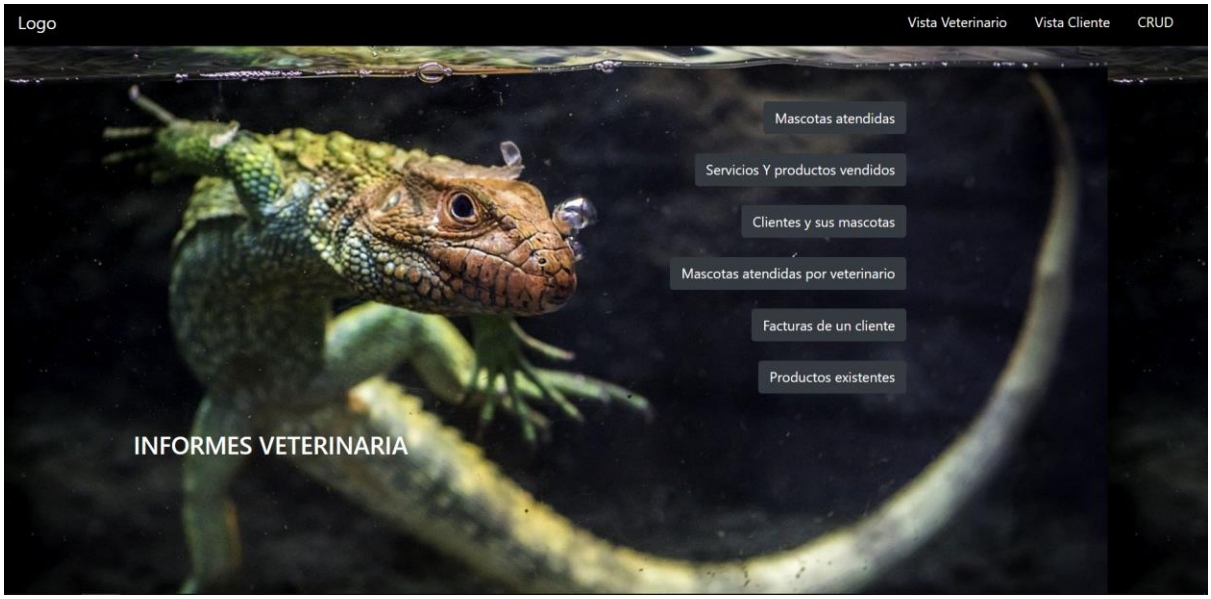
ID	Tipo de Mascota	Nombre	Fecha	Cedula Dueno
1	hamster siberano	Alex Lopez	2020-10-12	Beto Zumbado
2	pastor alemán	Leo Moreira	2020-09-11	Beto Zumbado
3	zaguate	Barlon Sequeira	2020-02-09	JP Quiros
4	conejo	Patrick Pemberton	2020-10-10	JP Quiros
5	Hamster	Rolado Fonseca	2020-09-11	Pablo Antonio Gabas
6	Gato	Rata Calva	2020-11-08	Pablo Antonio Gabas
7	Perro	Manchas	2020-11-12	Alejandro Alpizar

(Muestra del select de la base de datos)

Descripción del problema:

Se solicitó una aplicación para el manejo de datos de un hospital veterinario y tienda de mascotas con requerimientos como específicos de cada entidad del sistema (cliente/dueño, veterinario, mascota, facturas, expedientes), y que contara con varios métodos tanto de ingreso de datos, como de listado (mascotas atendidas, clientes y sus mascotas... etc).

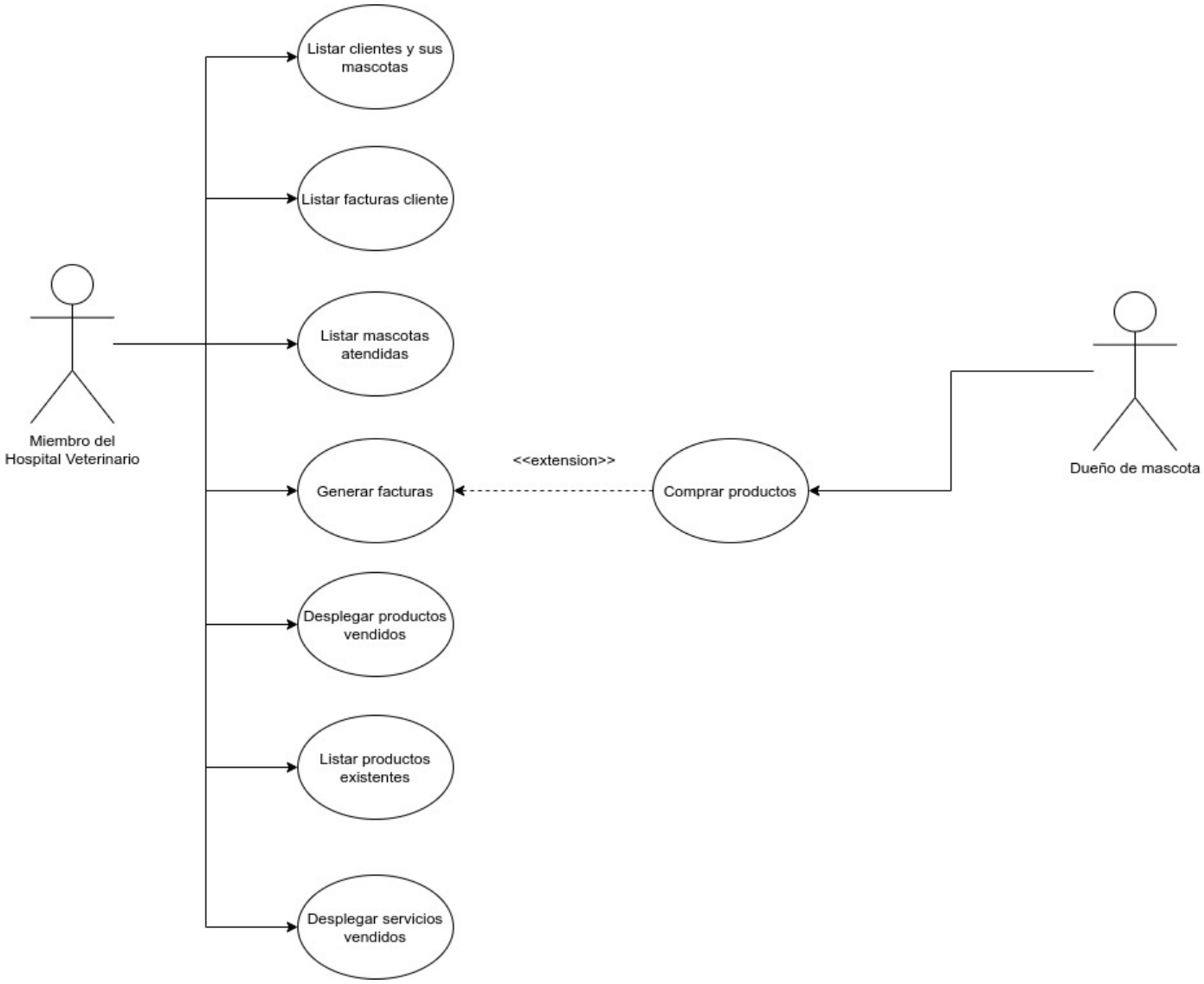
De igual forma, un cliente tendría la funcionalidad de adquirir productos y generar esas respectivas facturas en la base de datos.



(Imagen de referencia para la idea de lo llegaría a ser la interfaz gráfica)

Nuestra solución se basa en distintos menús, con las distintas funcionalidades dependiendo de la entidad con la que interactúe el sistema (veterinario o cliente), y una vista para generar las operaciones CRUD en le sistema.

Diagrama de casos de uso



Para cuestiones de caso de uso, tomamos todos los métodos de la vista del hospital como una sola entidad, donde cualquier persona que tenga acceso como veterinario podría ejecutar todos esos métodos.

Para el caso del método del cliente, el hecho de comprar productos generaría dentro de la entidad del hospital una factura con los datos de la compra.

Para el caso de las operaciones CRUD, se hará una vista aparte con todas las entidades en la base datos.

Descripciones de casos de uso:

Caso de Uso: Listar cliente y sus mascotas	
Actor Primario:	- Miembro del hospital veterinario
Actor Secundario:	- Dueño de mascota
Pre-condiciones:	- El miembro debe estar registrado y autenticado en el sistema como miembro del hospital veterinario.
Post-condiciones:	- El sistema despliega en el servidor web la información de todos los clientes del hospital veterinario.
Flujo principal:	1. El miembro selecciona la funcionalidad de “Listar Clientes” 2. El sistema accede a la base de datos y selecciona toda la información respectiva de los clientes. 3. El sistema mediante las funcionalidades de Java Spring, despliega la información de todos los clientes y sus mascotas, de la selección de la base de datos hecha previamente.
Flujos Alternos:	3.(b) El sistema muestra un mensaje de “No existen clientes registrados en el sistema”

Caso de Uso: Listar facturas cliente	
Actor Primario:	- Miembro del hospital veterinario
Actor Secundario:	- Dueño de mascota
Pre-condiciones:	- El miembro debe estar registrado y autenticado en el sistema como miembro del hospital veterinario.
Post-condiciones:	- El sistema despliega en el servidor web la información sobre las facturas del cliente seleccionado.
Flujo principal:	1. El miembro selecciona la funcionalidad de “Listar facturas” 2. El sistema pasa a la pantalla donde se ingresan las credenciales del cliente. 3. El miembro ingresa las credenciales del cliente (identificación). 4. El miembro selecciona el botón de “desplegar”. 5. El sistema accede a la base de datos y selecciona toda la información de las facturas respectivas del cliente. 6. El sistema mediante las funcionalidades de Java Spring, despliega en otra pantalla la información de todas las facturas del cliente respectivo.

Flujos Alternos:	5. (b) El sistema muestra un mensaje de “No existe ese cliente registrado en el sistema”

Caso de Uso: Listar mascotas atendidas	
Actor Primario:	- Miembro del hospital veterinario
Actor Secundario:	
Pre-condiciones:	- El miembro debe estar registrado y autenticado en el sistema como miembro del hospital veterinario.
Post-condiciones:	- El sistema despliega en el servidor web la información sobre todas las mascotas atendidas por el hospital.
Flujo principal:	1. El miembro selecciona la funcionalidad de “Listar mascotas atendidas” 2. El sistema accede a la base de datos y selecciona toda la información respectiva de las mascotas atendidas por el hospital. 3. El sistema mediante las funcionalidades de Java Spring, despliega la información de todas las mascotas, de la selección de la base de datos hecha previamente.
Flujos Alternos:	3.(b) El sistema muestra un mensaje de “No existen mascotas registradas en el sistema”

Caso de Uso: Desplegar productos vendidos	
Actor Primario:	- Miembro del hospital veterinario
Actor Secundario:	
Pre-condiciones:	- El miembro debe estar registrado y autenticado en el sistema como miembro del hospital veterinario.
Post-condiciones:	- El sistema despliega en el servidor web la información sobre todas los productos vendidos, durante un periodo de tiempo especificado.
Flujo principal:	1. El miembro selecciona la funcionalidad de “Listar productos vendidos”. 2. El sistema pasa a la pantalla donde se ingresa el lapso de tiempo de los productos a mostrar. 3. El sistema accede a la base de datos y selecciona toda la información respectiva de los productos vendidos en el lapso de tiempo ingresado anteriormente. 4. El sistema mediante las funcionalidades de Java Spring, despliega la información de los productos

	vendidos, de la selección de la base de datos hecha previamente.
Flujos Alternos:	4.(b) El sistema muestra un mensaje de “No existen productos vendidos en ese lapso de tiempo”

Caso de Uso: Desplegar servicios vendidos	
Actor Primario:	- Miembro del hospital veterinario
Actor Secundario:	
Pre-condiciones:	- El miembro debe estar registrado y autenticado en el sistema como miembro del hospital veterinario.
Post-condiciones:	- El sistema despliega en el servidor web la información sobre todas los servicios vendidos, durante un periodo de tiempo especificado.
Flujo principal:	5. El miembro selecciona la funcionalidad de “Listar servicios vendidos”. 6. El sistema pasa a la pantalla donde se ingresa el lapso de tiempo de los servicios a mostrar. 7. El sistema accede a la base de datos y selecciona toda la información respectiva de los servicios vendidos en el lapso de tiempo ingresado anteriormente. 8. El sistema mediante las funcionalidades de Java Spring, despliega la información de los servicios vendidos, de la selección de la base de datos hecha previamente.
Flujos Alternos:	4.(b) El sistema muestra un mensaje de “No existen servicios vendidos en ese lapso de tiempo”

Caso de Uso: Listar productos existentes	
Actor Primario:	- Miembro del hospital veterinario
Actor Secundario:	
Pre-condiciones:	- El miembro debe estar registrado y autenticado en el sistema como miembro del hospital veterinario.
Post-condiciones:	- El sistema despliega en el servidor web la información sobre los productos vendidos por el hospital.
Flujo principal:	1. El miembro selecciona la funcionalidad de “Listar productos existentes”

	2. El sistema accede a la base de datos y selecciona toda la información respectiva de los productos vendidos por el hospital. 3. El sistema mediante las funcionalidades de Java Spring, despliega la información de todos los productos, de la selección de la base de datos hecha previamente.
Flujos Alternos:	3.(b) El sistema muestra un mensaje de “No existen productos existentes en el sistema”

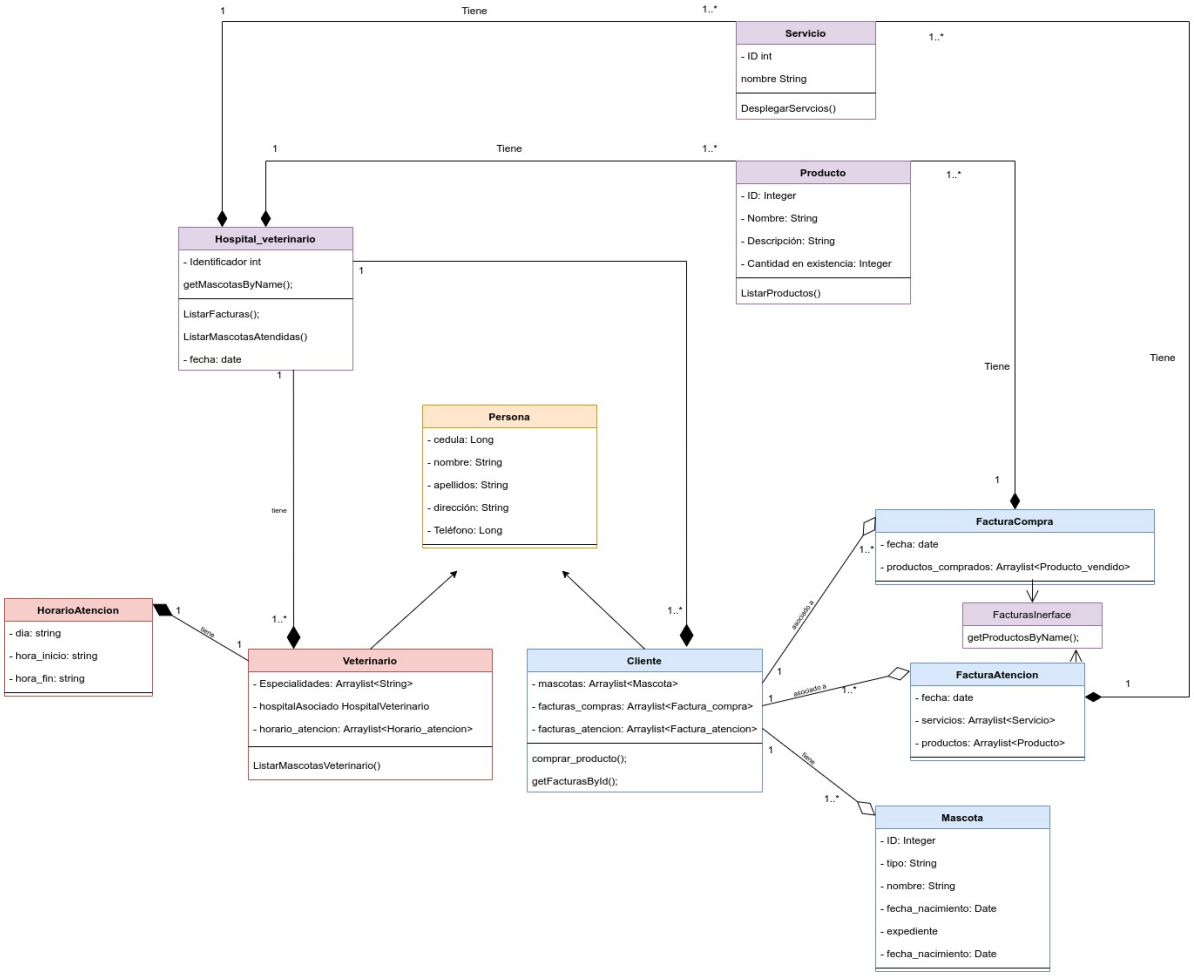
Para lo que es la conexión y la llamada a la base de datos, a los servicios entre entidades del modelo vista controlador, se implementarán distintas interfaces que permitas desarrollar estas funcionalidades de una forma más modular y que realizar las distintas funcionalidades para mismos objetivos, como lo es caso de las operaciones de la base de datos, ya que se hará uso del Spring con la dependencia JPA Repository, para poder acceder a la base de datos sin necesidad de crear nuevas conexiones, nada más estos acceden a la interface en su método e implementación específicos.

Las tres demencias claves para realizar la implementación con Spring Boot serían:

- Spring Web (trae integrado el MVC)
- Thymyleaf para la comunicación con los archivos de marcado HTML
- JPA Repository: Interactúa con la base de datos de una forma más eficaz.

Para las muestras de la base de datos se hizo uso de Boostrap para hacer esas vistas muchos más agradables a la vista.

Diagrama de clases



A nivel de implementación, vimos pertinente realizar distintas composiciones entre clases, para hacer la comunicación de los atributos más sencilla entre estos.

Gracias a la ayuda de las dependencias mencionadas anteriormente, la inyección de los distintas clases y la comunicación y los beans de Spring se hace más sencilla.

A la hora de implementar para mantener un mejor orden, se tomó la decisión de separar las clases que tienen mayor relación dentro de un mismo paquete, con la finalidad de no tener gran cantidad de archivos Java regados en un mismo directorio, y así tener una mejor modularización.

Análisis de resultados:

El proyecto se concretó con éxito, se logró implementar de manera óptima el modelo, cumpliendo con cada requerimiento solicitado y de manera ordenada y eficiente. Se logró crear cada objeto solicitado en el sistema y sus respectivos métodos, en nuestro caso para guardar los datos decidimos hacerlo por medio de una base de Datos en MySQL, de la cual por medio del uso de Spring Data se importaban los datos.

Las únicas operaciones que no fueron posibles fue implementar todo el CRUD, con las operaciones de insert, update y delete. Con la implementación de la biblioteca JPA, se tiene

la implementación para poder realizar cada operación, pero no se hubiera realizado con interacción al usuario con HTML y Thymyleaf, por lo que se omitió por estos motivos.

Esto se debió más que todo por cuestiones de tiempo, y no hubo tiempo para la implementación. Sin embargo, las interfaces respectivas y los servicios contienen las implementaciones necesarias para realizarlo desde código, mas no como interacción de usuario desde la pagina web.

Se utilizó el framework de Java Spring que era de los principales objetivos y se aprendió mucho más de este.

Conclusiones Personales:

José Pablo Quirós:

Este proyecto me pareció sumamente enriquecedor, me dio una vista más cercana a desarrollo empresarial por medio de Spring y sus herramientas. Este acercamiento me dio mucho conocimiento para poder abordar este tipo de proyectos con más confianza. Por mi pate siento que según lo especificado en este proyecto pudimos terminar todo de la mejor manera.

Alberto Zumbado:

El hecho de poder aprender los conceptos necesarios de lo que se refiere al MVC, e implementarlo en un marco de trabajo como lo es Spring me parece lo más provechoso de este trabajo. Me parece muy interesante como ya en el mundo laboral si puede llegar a hacer uso de estas herramientas que desenvuelven el desarrollo de una forma más ágil y rápida, con herramientas eficaces que ayudan a configurar dependencias que posiblemente desde java nativo serían sumamente complicadas.