

EXAMEN INTEGRADOR

LABORATORIO DE COMPUTACIÓN II



CARRERA TÉCNICO UNIVERSITARIO EN PROGRAMACIÓN

PROFESOR ASIGNADO ANGEL SIMÓN

ALUMNOS

BERTELLO ANA, CARABAJAL MARCELO,

MASINO FACUNDO, TOLA LUCAS



Índice



\supset	INTRODUCCIÓN	03
\supset	METODOLOGÍA DE TRABAJO	03
\supset	DESCRIPCIÓN DEL SISTEMA	04
\supset	SALIDAS DEL SISTEMA	06
\supset	INFORMES	06
\supset	INTERFAZ GRÁFICA	07
\supset	DIAGRAMA DE CLASES	09
\nearrow	CONCLUSIONES	10



Introducción

GESTION INTEGRAL DE CLINICA VETERINARIA

Vete++ tiene como foco agrupar la gestión de las operaciones claves de una clínica veterinaria.

Esto incluye el registro de los turnos de atención, la administración de clientes y pacientes, y el control de stock de suministros veterinarios y productos de petshop.

En lo que respecta al servicio veterinario en sí, Vete ++ permite l: mantener actualizado un registro de las vacunas administradas y además facilitar la información para que se envíen alertas a los clientes con el fin de asegurar la revacunación de sus animales de compañia.

Vete ++ no solo está pensando para facilitar la toma de decisiones sino para hacerle la vida más grata a esos seres tan especiales.

Metodología de Trabajo

DESARROLLO MODERNO

Para el desarrollo del sistema se decidió utilizar la plataforma **GitHub** que nos permite utilizar el sistema de control de versiones Git para trabajar de forma colaborativa.

De esta manera podemos controlar, comparar y revisar los distintos cambios que cada miembro del equipo realice de manera asíncrona.

Los commits que cada integrante del equipo realice van a ser subidos a la rama "develop" y una vez que se considere que el desarrollo llegó a una etapa estable se hará un merge a la rama "main"







Descripción del sistema



1. Registro de turnos

- ✓El sistema permitirá a los clientes solicitar y programar citas para sus mascotas de manera eficiente.
- Se registrarán la fecha y hora del turno, el cliente (dueño de la mascota) y los detalles de la mascota, incluyendo nombre, especie, raza, edad y cualquier historial médico relevante.
- Se generará un numero de turno único para cada cita, lo que facilitará la organización y el seguimiento.

2. Administración de clientes

- ✓El sistema mantendrá un registro completo de los clientes de la clínica, incluyendo su información de contacto, como nombre, dirección, teléfono y correo electrónico.
- Se asignará un id único a cada cliente para una identificación rápida y precisa.
- Los clientes tendrán la capacidad de programar nuevos turnos, verificar el estado de salud de sus mascotas y acceder a su historial médico en línea.





Descripción del sistema

3. Control de stock

- ✓El sistema gestionará un inventario de medicamentos y suministros veterinarios.
- Se registrarán los detalles de cada artículo, incluyendo nombre, cantidad de stock, precio, fecha de vencimiento y proveedor.
- Se implementará un sistema de alertas para notificar al personal de la clínica cuando un artículo este bajo en stock, lo que facilitará la reordenación oportuna.

4. Calendario de vacunación

- ✓ El sistema permitirá programar y registrar las vacunas administradas a las mascotas.
- Se mantendrá un historial de las vacunas aplicadas, incluyendo la fecha, el tipo de vacuna, el lote y la próxima fecha de vacunación recomendada.





Salidas del sistema

El sistema proporcionara diversas salidas, entre ellas:

- ✓ Listado de turnos programados.
- ✓Detalles de clientes y sus mascotas.
- ✓Estado del stock de medicamentos y suministros.
- Estadísticas de finanzas
- ✓ Historia clínica de una mascota

Informes

El sistema generará informes para ayudar en la toma de decisiones y la gestión de la clínica, incluyendo:

- Estadísticas de atención medica por periodo.
- ✓Reportes financieros, como ingresos por servicios y ventas de medicamentos.
- ✓ Vacunaciones pendientes
- Estadísticas de turnos por especialidad.







Interfaz Gráfica

La interfaz gráfica del sistema será del tipo Consola utilizando la librería "rlutil.h" para dar formato y color al texto y tablas.



EXAMEN INTEGRADOR - LABORATORIO DE COMPUTACION II GRUPO 10 - "VETE++"









(c)

Person

- □ personld: int
- □ name: string
- lastName: string
- setName(name: string): void
- setLastname(lastname: string): void
- getName(): string
- getLastname(): string
- load(): void



Client

- clientld: int
- address: string
- phone: string
- email: string
- setId(id: int): void
- setAddress(address: string): void
- setPhone(phone: string): void
- setEmail(email: string): void
- getAddress(): string
- getPhone(): string
- getEmail(): string
- getld(): int
- addPet():void
- deletePet(): void
- queryHistory(): void
- makeAppointment(): void
- queryPets(): void
- load(): void



- Vet
- vetld: int
- speciality: string
- setIdVet(idVet: int): void
- setSpeciality(speciality: string): void
- getIdVet(): int
- getSpeciality(): string
- queryHistory(): void
- assignTreatment(): void
- issueDiagnosis(): void
- load(): void

Appointment appld: int date: Date □ time: Time urgencyCat: int reason: string attended: bool clientld: int petId: int setAppld(id: int): void setDate(date: Date): void setTime(time: Time): void setUrgencyCat(cat: int): void setReason(reason: string): void setAttended(attended: bool): void getAppld(): int

getDate(): DategetTime(): Time

getUrgencyCat(): int

getReason(): string

getAttended(): bool

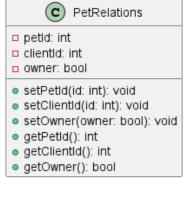
load(): void







(C) VetVisits



(C) Product

- productld: int
- productName: string
- description: string
- quantity: int
- stock: int
- price: float
- brand: string
- productType:string
- productCategory: Category
- setProductId(id: int): void
- setName(name:string): void
- setDescription(description: string): void
- setQuantity(qty: int): void
- setStock(stock:int): void
- setPrice(price:float): void
- setBrand(brand: string): void
- setProductType(type: string): void
- setProductCategory(cat: Category): void
- getProductId(): int
- getName(): string
- getQuantity(): int
- getStock(): int
- getPrice(): float
- getBrand():string
- getProductType(): string
- getProductCategory(): Category
- addProduct(): void
- DeleteProduct(): void
- updateStock(): void

(C) ProductCategory

- catld: int
- catName: string
- setCatId(id: int): void
- setCatName(name: string): void
- getCatId(): int
- getCatName(): string
- load(): void



Date

- day: int
- month:int
- year: int
- setDay(day: int): void
- setMonth(month: int): void
- setYear (year: int): void
- getDay(): int
- getMonth(): int
- getYear(): int



Time

- hour: int
- min:int
- sec: int
- setHour(hour: int):void
- setMin(min: int): void
- setSec (sec: int): void
- getHour(): int
- getMin(): int
- getSec(): int

(c)

Sale

- saleld: int
- □ totalAmount: float
- paymentMethod: string
- □ saleDate: Date
- □ saleTime: Time
- quantity: int
- □ isVisit: bool
- transactionId: int
- setSaleId(id:int): void
- setAmount(amount: float): void
- setPaymentMethod(method: string): void
- setDate(date: Date): void
- setTime(time: Time): void
- setQuantity(qty: int): void
- setIsVisit(isVisit: bool): void
- setTransactionId(id: int): void
- getSaleId(): int
- getAmount(): float
- getPaymentMethod(): string
- getDate(): Date
- getTime(): Time
- getQuantity(): int
- getIsVisit(): Bool
- getTransactionId(): int
- load(): void

C

Buy

- buyld: int
- □ totalAmount: float
- paymentMethod: string
- buyDate: Fecha
- buyTime: Time
- productid: int
- quantity: int
- transactionId: int
- setBuyld(item:Inventory): void
- setAmount(importe: int): void
- setpaymentMethod (method : string): void
- setDate (date: Date):void
- setTime (time: Time): void
- setProductId (ProductId: int): void
- setQuantity(qty: int): void
- setTransactionId (id: int): void
- getBuyld(): int
- getTotalAmount(): float
- getPaymentMethod():string
- getbuyDate(): Date
- getbuyTime(): Time
- getProductId (): int
- getQuiantity (): int
- getTransaction (): int
- load(): void



ItemSale

- productid: int
- quantity: int
- unitPrice: float
- saleld: int
- setProductId(productId: int): void
- setQuantity(quantity int): void
- setUnitPrice(total: float): void
- setSaleId(saleId: int) void
- getProductId(): int
- getQuantity():int
- getUnitPrice(): float
- getSaleId(): int
- load(): void

□ idTransaction: int

- date: Datetime: Time
- description: stringamount: float
- setIdTransaction(idTrx: int): void
- setDate(date: Date): void
- setTime (time : Time): void
- setDescription(desc: string): void
- setAmount(amount: float): void
- getIdTransaction(): int
- getDate(): Fecha
- getTime(): Time
- getDescription(): string
- getAmount(): float



- □ saleId:int
- paymentMethod: string
- setSaleId(saleId int): void
- setpaymentMethod(method:string): void
- getSaleId():int
- getPaymentMethod(): string

c Expense

- buyld: int
- paymentMethod: string
- setBuyld(id: int): void
- setPaymentMethod(method: string)
- getBuyld(): int
- getPaymentMethod(): string

Conclusiones



El sistema proporcionará a la clínica veterinaria una herramienta eficiente para gestionar sus operaciones diarias, desde la programación de turnos hasta el control de stock y la administración de clientes.

También garantizará el seguimiento adecuado de las vacunas y proporcionará datos valiosos para la toma de decisiones estratégicas y la mejora de la atención al cliente. Su implementación puede aumentar la eficiencia y la calidad de los servicios veterinarios ofrecidos.

Muchas gracias

facundo.masino@alumnos.frgp.utn.edu.ar



marcelocarabajal@alumnos.frgp.utn.edu.ar



ana.bertello@alumni.frgp.utn.edu.ar

