

## 1. Informe final ଫଫ଼rଖ3଼ମ୍ବାnforme final Proyecto APT

El objetivo de este informe es que describas los aspectos más relevantes de tu Proyecto APT. Es importante que fundamentes las decisiones que tuviste que tomar a lo largo del proceso.

A continuación, encontrarás distintos campos que deberás completar con la información solicitada, los que dan cuenta del resumen de tu proyecto APT y sus principales resultados.

Nombre del proyecto	VetCare Pro - Sistema de Gestión Veterinaria
Área (s) de desempeño(s)	<ul> <li>Desarrollo de Aplicaciones Web</li> <li>Gestión de Proyectos Informáticos</li> <li>Bases de Datos</li> </ul>
Competencias	<ol> <li>Desarrollar una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento</li> <li>Construir modelos de datos para soportar los requerimientos de la organización</li> <li>Gestionar proyectos informáticos, ofreciendo alternativas para la toma de decisiones</li> <li>Realizar pruebas de certificación de productos y procesos utilizando buenas prácticas</li> </ol>

Contenidos del informe final		
1. Relevancia del proyecto APT	Problema que busca solucionar:  VetCare Pro aborda la gestión manual e ineficiente en clínicas veterinarias pequeñas y medianas de Chile. Muchas clínicas manejan citas, historiales médicos y pagos en papel o con sistemas obsoletos, generando errores, pérdida de información y mala experiencia para los clientes.  Relevancia profesional:  La digitalización del sector salud veterinaria es una demanda creciente. Como ingeniero informático, desarrollar soluciones web para sectores en proceso de transformación digital es una oportunidad laboral concreta y necesaria en el mercado actual.  Ubicación: Clínicas veterinarias en zonas urbanas de Chile, principalmente Región Metropolitana (clínicas pequeñas con 1-5 veterinarios).  Población impactada:  • Veterinarios que necesitan gestionar pacientes y citas eficientemente • Clientes que buscan acceso online a información de sus mascotas • Administradores que requieren control de pagos y facturación	



Aporte de valor:  Guía3. Inf Automotinación progenda miesta y facturación. APT  Historiales médicos digitales centralizados  Asignaturate Carpestionne en reviver Carre Pro -  Sistema de Gestión Veterinaria  Objetivo General:  Desarrollar un sistemo web de gestión para clínicos veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados)  2. Desarrollar base de datos relacional con PostgreSQL y Prisma  3. Crear sistema de autenticación con rolas (Admin, Veteriaria, Cliente)  4. Implementar APIs REST para operaciones CRUD  5. Desarrollar integra responsive con React/Next.js  6. Integrar sistema de pagos anline con Stripe  7. Crear portal del cliente para visualización de información  8. Implementar historioles médicos digitales  9. Desarrollar antificaciones automáticas por email  10. Desplegar el sistema en producción  Metodología útilizada:  Metodología úgil con Scrum adaptado, dividido en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups  2. Setup: Configuración de entarno y herramientas  3. Backend: Prisma Schema y APIs REST  4. Autenticación: Sistema DNT con roles  5. Fronten: Osabbador y componentes React  6. Funcionalidades core: Citos, pagos, portal cliente  7. Avanadosi: Historiales médicos y notificaciones  8. Testing: Pruebas y optimización  9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:  Esta metodología fue adecuada porque permitió entregas incrementales, facilitó ajustes		
Asignatură e da postione networte e Pro - Sistema de Gestión Veterinaria  Objetivo General:  Desorrollor un sistema web de gestión para clínicas veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de formo eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollor base de adotas relacional con PostgreSQ1 y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinaria, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollor interfaz responsive con React/Next js 6. Integrar sistema de pagos online con Stripe 7. Crear portol del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología diflizada:  Metodología diflizada:  Metodología dor Scrum adaptado, dividido en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientos 3. Backent; Prisma Schema y APIs REST 4. Autenticación: Sistema IMT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Clas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing monual continuo.  Pertinencia:		Aporte de valor:
Asignatură e da postionie nedicos digitales centralizados Asignatură e da postionie nedicore previdet e de Pro - Sistema de Gestión Veterinaria  Objetivo General:  Desarrollar un sistema web de gestión para clínicas veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de adotas relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfar responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portol del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología difilizada:  Metodología difilizada:  Metodología difilizada:  Metodología vitilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientos 3. Backent: Prisma Schema y APIs REST 4. Autenticación: Sistema IMT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Clas, pagos, portal cliente 7. Avanzadas: Historiales médicas y notificaciones 8. Testing: Pruebas y optimización 9. Depley: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Automatización de agendamiento y facturación
Asignatură e da postionie nedicos digitales centralizados Asignatură e da postionie nedicore previdet e de Pro - Sistema de Gestión Veterinaria  Objetivo General:  Desarrollar un sistema web de gestión para clínicas veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de adotas relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfar responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portol del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología difilizada:  Metodología difilizada:  Metodología difilizada:  Metodología vitilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientos 3. Backent: Prisma Schema y APIs REST 4. Autenticación: Sistema IMT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Clas, pagos, portal cliente 7. Avanzadas: Historiales médicas y notificaciones 8. Testing: Pruebas y optimización 9. Depley: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Guía3. Informe sun al Pirroyecto APT
Sistema de Gestión Veterinaria  Objetivo General:  Desarrollar un sistema web de gestión para clínicas veterinarias que permita administrar citas, pacientes, historiales médicas y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autentiacción con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticos por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología útilizada:  Metodología útilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientos 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema iMT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citos, pagos, portal cliente 7. Avanzadas: Historiales médicas y notificaciones 8. Testing: Pruebas y aptimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semandles y testing manual continuo.  Pertinencia:		<ul> <li>Historiales médicos digitales centralizados</li> </ul>
Sisterna de Gestión Veterinaria  Objetivo General:  Desarrollar un sistema web de gestión para clínicos veterinarios que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autentiacción con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsíve con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones outomáticos por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología útilizada:  Metodología útilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citos, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y aptimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo Iterativo (implementar → probar → corregir → mejorar), con revisiones semandes y testing manual continuo.  Pertinencia:		Asignatura Capstone - Vetcare Pro -
Objetivo General:  Desarrollar un sistema web de gestión para clinicas veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema IWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.		
Desarrollar un sistema web de gestión para clínicas veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar bose de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autentiacción con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de apagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología utilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema IMT con roles 5. Frontend: Dashboard y componentes React 6. Funcionolidades corre citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Sistema de Gestión Veterinaria
Desarrollar un sistema web de gestión para clínicas veterinarias que permita administrar citas, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar bose de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autentiacción con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de apagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología utilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema IMT con roles 5. Frontend: Dashboard y componentes React 6. Funcionolidades corre citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
citos, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prismo Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebos y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Objetivo General:
citos, pacientes, historiales médicos y pagos de forma eficiente.  Objetivos Específicos:  1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prismo Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebos y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Desarrollar un sistema web de gestión para clínicas veterinarias que permita administrar
1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología útilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
1. Implementar arquitectura fullstack escalable (frontend/backend separados) 2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología útilizada:  Metodología útilizada:  Metodología útilizada:  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
2. Desarrollar base de datos relacional con PostgreSQL y Prisma 3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema WT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Objetivos Especificos:
3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente) 4. Implementar APIs REST para operaciones CRUD 5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología útilizada:  Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Implementar arquitectura fullstack escalable (frontend/backend separados)
<ul> <li>4. Implementar APIs REST para operaciones CRUD</li> <li>5. Desarrollar interfaz responsive con React/Next.js</li> <li>6. Integrar sistema de pagos online con Strup</li> <li>7. Crear portal del cliente para visualización de información</li> <li>8. Implementar historiales médicos digitales</li> <li>9. Desarrollar notificaciones automáticas por email</li> <li>10. Desplegar el sistema en producción</li> <li>Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):</li> <li>1. Planificación: Análisis de requerimientos y diseño de mockups</li> <li>2. Setup: Configuración de entorno y herramientas</li> <li>3. Backend: Prisma Schema y APIs REST</li> <li>4. Autenticación: Sistema JWT con roles</li> <li>5. Frontend: Dashboard y componentes React</li> <li>6. Funcionalidades core: Citas, pagos, portal cliente</li> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>	2. Objetivos	2. Desarrollar base de datos relacional con PostgreSQL y Prisma
5. Desarrollar interfaz responsive con React/Next.js 6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología ágil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		3. Crear sistema de autenticación con roles (Admin, Veterinario, Cliente)
6. Integrar sistema de pagos online con Stripe 7. Crear portal del cliente para visualización de información 8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
<ul> <li>7. Crear portal del cliente para visualización de información</li> <li>8. Implementar historiales médicos digitales</li> <li>9. Desarrollar notificaciones automáticas por email</li> <li>10. Desplegar el sistema en producción</li> <li>Metodología ágil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):</li> <li>1. Planificación: Análisis de requerimientos y diseño de mockups</li> <li>2. Setup: Configuración de entorno y herramientas</li> <li>3. Backend: Prisma Schema y APIs REST</li> <li>4. Autenticación: Sistema JWT con roles</li> <li>5. Frontend: Dashboard y componentes React</li> <li>6. Funcionalidades core: Citas, pagos, portal cliente</li> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		
8. Implementar historiales médicos digitales 9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología útilizada:  Metodología ágil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
9. Desarrollar notificaciones automáticas por email 10. Desplegar el sistema en producción  Metodología utilizada:  Metodología ágil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):  1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
<ul> <li>Metodología utilizada:</li> <li>Metodología ágil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):</li> <li>1. Planificación: Análisis de requerimientos y diseño de mockups</li> <li>2. Setup: Configuración de entorno y herramientas</li> <li>3. Backend: Prisma Schema y APIs REST</li> <li>4. Autenticación: Sistema JWT con roles</li> <li>5. Frontend: Dashboard y componentes React</li> <li>6. Funcionalidades core: Citas, pagos, portal cliente</li> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		
<ul> <li>Metodología úgil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):         <ol> <li>Planificación: Análisis de requerimientos y diseño de mockups</li> <li>Setup: Configuración de entorno y herramientas</li> <li>Backend: Prisma Schema y APIs REST</li> <li>Autenticación: Sistema JWT con roles</li> <li>Frontend: Dashboard y compentes React</li> <li>Funcionalidades core: Citas, pagos, portal cliente</li> <li>Avanzadas: Historiales médicos y notificaciones</li> <li>Testing: Pruebas y optimización</li> <li>Deploy: Producción y documentación</li> </ol> </li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		
<ul> <li>Metodología ágil con Scrum adaptado, dividida en 9 sprints de 2 semanas (18 semanas totales):         <ol> <li>Planificación: Análisis de requerimientos y diseño de mockups</li> <li>Setup: Configuración de entorno y herramientas</li> <li>Backend: Prisma Schema y APIs REST</li> <li>Autenticación: Sistema JWT con roles</li> <li>Frontend: Dashboard y componentes React</li> <li>Funcionalidades core: Citas, pagos, portal cliente</li> <li>Avanzadas: Historiales médicos y notificaciones</li> <li>Testing: Pruebas y optimización</li> <li>Deploy: Producción y documentación</li> </ol> </li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		10. Despiegar ei sistema en producción
1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		Metodología utilizada:
1. Planificación: Análisis de requerimientos y diseño de mockups 2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		
<ol> <li>Planificación: Análisis de requerimientos y diseño de mockups</li> <li>Setup: Configuración de entorno y herramientas</li> <li>Backend: Prisma Schema y APIs REST</li> <li>Autenticación: Sistema JWT con roles</li> <li>Frontend: Dashboard y componentes React</li> <li>Funcionalidades core: Citas, pagos, portal cliente</li> <li>Avanzadas: Historiales médicos y notificaciones</li> <li>Testing: Pruebas y optimización</li> <li>Deploy: Producción y documentación</li> </ol> Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo. Pertinencia:		
2. Setup: Configuración de entorno y herramientas 3. Backend: Prisma Schema y APIs REST 4. Autenticación: Sistema JWT con roles 5. Frontend: Dashboard y componentes React 6. Funcionalidades core: Citas, pagos, portal cliente 7. Avanzadas: Historiales médicos y notificaciones 8. Testing: Pruebas y optimización 9. Deploy: Producción y documentación  Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.  Pertinencia:		totuics).
<ol> <li>Backend: Prisma Schema y APIs REST</li> <li>Autenticación: Sistema JWT con roles</li> <li>Frontend: Dashboard y componentes React</li> <li>Funcionalidades core: Citas, pagos, portal cliente</li> <li>Avanzadas: Historiales médicos y notificaciones</li> <li>Testing: Pruebas y optimización</li> <li>Deploy: Producción y documentación</li> </ol> Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo. Pertinencia:	3. Metodología	1. Planificación: Análisis de requerimientos y diseño de mockups
<ul> <li>4. Autenticación: Sistema JWT con roles</li> <li>5. Frontend: Dashboard y componentes React</li> <li>6. Funcionalidades core: Citas, pagos, portal cliente</li> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		, , , ,
<ul> <li>5. Frontend: Dashboard y componentes React</li> <li>6. Funcionalidades core: Citas, pagos, portal cliente</li> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		
<ul> <li>6. Funcionalidades core: Citas, pagos, portal cliente</li> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		
<ul> <li>7. Avanzadas: Historiales médicos y notificaciones</li> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		
<ul> <li>8. Testing: Pruebas y optimización</li> <li>9. Deploy: Producción y documentación</li> <li>Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo.</li> <li>Pertinencia:</li> </ul>		· · · · · · · · · · · · · · · · · · ·
Procedimiento: Desarrollo iterativo (implementar → probar → corregir → mejorar), con revisiones semanales y testing manual continuo. Pertinencia:		
revisiones semanales y testing manual continuo.  Pertinencia:		
revisiones semanales y testing manual continuo.  Pertinencia:		December 1 - Comment of the section
Pertinencia:		
Esta metodología fue adecuada porque permitió entregas incrementales, facilitó ajustes		Pertinencia:
Esta metodologia jue daecudaa porque permitio entregas incrementales, facilito ajustes		Esta matadología fue adocuada norque normitió entragas ingrementales facilitá silutes
rápidos ante problemas técnicos, y proporcionó visibilidad constante del progreso		

mediante el roadmap visual.





Actividades principales:

### Guía3. Informesefinale Pur, oxectour AP el Tentorno

(PostgreSQL, Git, Vercel)

Asignaturaringa: pstoppele bas Vento (Agree y Dimogrién JWT Sprint 5: Desarrollo de interfaz principal (Dashboard, Login, componentes

## Sisterna.des Siste

médicos, PDFs y emails

- Sprint 8: Testing y optimización
- Sprint 9: Deploy en producción y documentación

#### Facilitadores:

4. Desarrollo

- Documentación técnica clara de Next.js, Prisma y Stripe
- Arquitectura bien definida desde el inicio
- Planificación realista con buffer para imprevistos

#### Dificultades y cómo las abordé:

- Stripe con CLP: Investigué documentación sobre monedas sin decimales y ajusté el códiao
- Rutas Next.js 13: Depuré estructura de carpetas siguiendo convenciones del App Router
- 3. **Middleware multi-rol:** Dediqué tiempo extra a diseñar sistema de permisos escalable
- 4. **Configuración de Resend:** Seguí tutoriales y realicé pruebas incrementales

**Ajuste realizado:** Adelanté el sistema de notificaciones del Sprint 7 al Sprint 6 para integrarlo mejor con Stripe.

5. Evidencias

- 1. Roadmap visual con 9 sprints y progreso completado
- 2. Código fuente en repositorio Git organizado
- 3. Base de datos PostgreSQL con Prisma Schema implementado
- Capturas de pantalla: Dashboard, sistema de citas, portal cliente, pagos, historiales médicos
- 5. Sistema en producción desplegado en Vercel
- 6. Documentación técnica: README y guías de usuario

## 6. Intereses y proyecciones profesionales

#### Aporte del proyecto:

VetCare Pro me ayudó a confirmar mi interés en desarrollo fullstack y descubrir afinidad particular con backend y arquitectura de sistemas. Aprendí que disfruto diseñar bases de datos eficientes, resolver problemas de integración y trabajar en proyectos con impacto social real.

#### Cambios en intereses:

Mis intereses se mantienen en desarrollo de software pero ahora más enfocados. Pasé de querer "hacer aplicaciones web" a interesarme específicamente en desarrollo backend con arquitecturas escalables y proyectos del sector salud.





Intereses a profundizar:

# Guía3. Infath Mantico Provección de Gestión Veterinaria Asignatura de Gestión Veterinaria

Me proyecto como desarrollador fullstack junior en una empresa tech, idealmente en salud digital. Busco trabajar en equipos ágiles, proyectos con impacto real, y eventualmente especializarme en arquitectura backend. Este proyecto me dio confianza en mis habilidades técnicas y me siento preparado para desafíos profesionales reales.