1) Prototipado dinámico completo

en el archivo src/main/resources/templates/index.html se observa un prototipado dinámico completo. Se utilizan componentes visuales de Bootstrap (navbar, tarjetas, modales, tablas, carrusel) y datos dinámicos con Thymeleaf (th:text, th:each, etc.), lo que permite que la interfaz se adapte y muestre información proveniente del backend. Esto cumple con el requisito de prototipado dinámico completo.

2) Framework front-end (Bootstrap) con componentes: formularios, validación, ventanas modales, barras de navegación, tablas, carrusel, entre otros.

Sí, en el archivo src/main/resources/templates/index.html se están utilizando los siguientes componentes de Bootstrap:

Barras de navegación: <nav class="navbar ...">
Ventanas modales: Hay tres modales (modalUno, modalDos, modalTres) usando clases modal y atributos data-bs-toggle="modal".
Tablas: Se usa una tabla con clases table table-hover para mostrar próximas citas.
Carrusel: Se incluye con th:replace="~{fragments/header :: carrusel}" (el carrusel está en un fragmento aparte).
Formularios: En este archivo no hay formularios visibles, pero podrían estar en otras vistas.
Validación: No se observa validación de formularios en este archivo.
Otros componentes: Tarjetas (card), botones (btn), iconos (bi), y diseño responsivo (container, row, col-*).
Resumen:
Se usan la mayoría de los componentes requeridos (barra de navegación, modales, tablas, carrusel, tarjetas, botones). Falta ver formularios y validación en este archivo, pero podrían estar presentes en otras vistas del proyecto.

3) Framework backend (Spring Boot)

Sí, el proyecto está usando el framework backend Spring Boot. Esto se confirma por la estructura del proyecto, las dependencias en los logs y la clase principal HappypetsApplication que inicia con Spring Boot.

4) Spring Web: @Controller, @GetMapping, @PostMapping

Las anotaciones @Controller, @GetMapping y @PostMapping de Spring Web se utilizan en las clases del paquete de controladores, normalmente ubicadas en src/main/java/tu/paquete/controller

5) Thymeleaf: 10 Atributos, 5 expresiones

En el archivo src/main/resources/templates/index.html se utilizan los siguientes atributos de **Thymeleaf (prefijo th:):**

th:href
th:src
th:replace
th:text
th:each

Cantidad de usos en este archivo:

th:href: 5 veces (en enlaces y hoja de estilos)
th:src: 3 veces (en imágenes)
th:replace: 2 veces (para incluir fragmentos)
th:text: 7 veces (para mostrar variables y datos)
th:each: 1 vez (para iterar sobre próximas citas)

Resumen de expresiones:

${...}: Expresión de variable de contexto (por ejemplo, ${citasHoy}, ${proximasCitas}).
#temporals.format(...): Expresión de utilidad para formatear fechas.
Ejemplos concretos:

${citasHoy}
${totalPacientes}
${totalMedicos}
${totalConsultorios}
${proximasCitas}
${cita.paciente.nombre + ' ' + cita.paciente.apellido}
${cita.medico.nombre}
${cita.medico.apellido}
${cita.medico.especialidad.nombre}
${cita.consultorio.codigo}
#temporals.format(cita.fechaHora, 'dd/MM/yyyy HH:mm')
Estas expresiones permiten mostrar datos dinámicos y recorrer listas en la plantilla.

6) Spring Data JPA: 5 clases relacionales en el modelo y en total 10 anotaciones
Se esta usando Spring Data JPA, las clases relacionales suelen estar en el paquete model o entity
Las clases relacionales típicas para una clínica serían: Cita, Paciente, Medico, Consultorio, Especialidad.

Las 10 anotaciones más comunes de JPA que puedes encontrar en estas clases son:

@Entity — Marca la clase como entidad JPA.
@Table — Especifica la tabla en la base de datos.
@Id — Indica el campo clave primaria.
@GeneratedValue — Estrategia de generación de la clave primaria.
@Column — Configura columnas específicas.
@ManyToOne — Relación muchos a uno.
@OneToMany — Relación uno a muchos.
@JoinColumn — Especifica la columna de unión para relaciones.
@ManyToMany — Relación muchos a muchos.
@OneToOne — Relación uno a uno.
Dónde están en el proyecto:

7) CRUD completo.

8) Spring Validator: 10 validaciones en las clases del modelo y 1 validación
Las validaciones de Spring Validator suelen implementarse usando anotaciones de Bean Validation (javax.validation o jakarta.validation) en las clases del modelo, ubicadas normalmente en
Las 10 validaciones más comunes y su uso típico serían:
@NotNull — El campo no puede ser nulo.
@NotBlank — El campo no puede estar vacío (para Strings).
@NotEmpty — El campo no puede estar vacío (colecciones o Strings).
@Size — Tamaño mínimo/máximo de cadenas o colecciones.
@Email — Formato de correo electrónico válido.
@Min — Valor mínimo para números.
@Max — Valor máximo para números.
@Pattern — Expresión regular para validar el formato.
@Past — Fecha debe ser en el pasado.
@Future — Fecha debe ser en el futuro.
Dónde se usan:
Estas anotaciones se colocan sobre los atributos de las entidades, por ejemplo en la clase Paciente:

PERSONALIZADA
en el archivo src/main/java/com/grupo2/happypets/validation/UniqueDni.java se define una validación específica personalizada mediante la anotación @UniqueDni.
Esta anotación se utiliza para validar que el campo DNI sea único en la base de datos, y está asociada a la clase validadora UniqueDniValidator.