Documento Proyecto Final de Verificación y Validación

Elaborado por:

Mendoza Severo Gustavo Yussif

Mongeote Tlachy Daniel

Torres Osorio Alesis de Jesús

Universidad Veracruzana

Xalapa, Ver; a viernes 8 de Noviembre del 2024.

Tabla de Contenido

Control de cambios

Nombre	Fecha	Motivo del cambio	Versión
EquipoEscribeUnNombre	08/11/2024	Creación del documento hasta el pt.5	1.0

1. Propósito

En esta sección se describirá el propósito, objetivos y alcance del esfuerzo de Verificación y Validación, enfatizando el alcance de la V&V y los resultados que se esperán obtener.

1.1. Objetivos de validación y verificación de software

Interesados:

a. **Profesora de V&V (Elizabeth Murrieta Sangabriel):** Responsable de supervisar y guiar el proceso de V&V, asegurando que se sigan las metodologías adecuadas, revisando la planificación de pruebas y validando que los resultados cumplan con los estándares de calidad establecidos para el sistema HealthDivineSys.

b. **Equipo de V&V (Equipo EscribeUnNombre):** Encargado de ejecutar las pruebas y verificaciones del sistema, diseñando y aplicando casos de prueba para todos los módulos, documentando resultados y asegurando que el sistema cumpla con los requerimientos funcionales y no funcionales especificados en el proyecto.

Sistema:

- El sistema debe garantizar un acceso rápido a la información (menos de 2 segundos de tiempo de respuesta), mantener una disponibilidad del 98%, proporcionar una interfaz intuitiva para los nutricionistas, y asegurar la confidencialidad de los datos mediante cifrado y autenticación robusta, todo esto mientras facilita la comunicación entre nutricionistas y pacientes, mejora la precisión en el seguimiento del progreso, y permite la generación y envío de documentación en formato PDF.

Operacionales:

- El sistema busca mejorar la eficiencia en la gestión de datos de pacientes al automatizar procesos que tradicionalmente eran manuales, permitiendo el registro y actualización de información personal, planes alimenticios, citas y evaluaciones nutricionales.

1.2. Alcance

Como base para el enfoque de este proyecto se utilizará el estándar IEEE std. 1012-2016, específicamente los apartados 8 y 9 donde se describen las fases y las tareas que se emplean más adelante. Cabe destacar, que no se aplicará la priorización que describe el estándar debido a pequeño alcance de este proyecto dados los artefactos con los que contamos.

Requisitos

Se asegura de que los requisitos sean claros, completos y verificables, incluyendo características necesarias, contexto de uso y restricciones.

	Artefacto		**Técnica**
Documento de Diseño		CU01 – Dar de alta paciente	
		CU03 – Modificar padecimientos del paciente	
		CU04 – Crear plan alimenticio del paciente	
		CU06 – Programar cita	

	Requisitos Funcionales	CU08 – Mostrar calendario de citas mensuales	**Inspección**
		CU09 – Crear nuevo registro de diagnóstico	
		CU11 – Añadir imagen de proceso del paciente	
		CU12 – Visualizar historial del paciente	
		CU14 – Exportar plan alimenticio a PDF	
	Requisitos No Funcionales	**Desempeño** RNF – 03: El sistema deberá proporcioanr al nutricionista una interfaz intuitiva y fácil de navegar. RNF – 07: El tiempo de respuesta del sistema para la carga de imágenes no deberá exceder los 3 segundos para garantizar una experiencia de usuario fluida.	
		Seguridad RNF – 02: El sistema deberá garantizar la confidencialidad de la información del paciente, restringiendo el acceso solo a usuarios autorizados. RNF – 08: Después de tres intentos fallidos de inicio de sesión, el sistema aplicará automáticamente un tiempo de espera antes de permitir nuevos intentos.	

Los entregables serán: Reporte de Inspección TSP en formato INS, Reporte de Junta TSP en formato MTG y el Checklist.

Diseño

Se busca asegurar que los requisitos del sistema especifiquen todas las características necesarias, restricciones y requisitos de rendimiento de manera clara y verificable.

	Artefacto		**Técnica**
Documento de Diseño	**Prototipos** **Diagramas de Robustez** **Diagramas de Secuencia**	CU01 – Dar de alta paciente	**RTF**
		CU03 – Modificar padecimientos del paciente	
		CU04 – Crear plan alimenticio del paciente	
		CU06 – Programar cita	
		CU08 – Mostrar calendario de citas mensuales	
		CU09 – Crear nuevo registro de diagnóstico	
		CU11 – Añadir imagen de proceso del paciente	
		CU12 – Visualizar historial del paciente	
		CU14 – Exportar plan alimenticio a PDF	

Los entregables serán: Reporte de RTF, Reporte de Junta TSP en formato MTG y el Checklist.

Implementación

Busca asegurar que las actividades de implementación produzcan un elemento del sistema que cumpla con los requisitos del sistema y que implementen correctamente la definición de diseño.

	Artefacto		Técnica
Código fuente, Manual de usuario	**Código Pruebas unitarias**	CU01 – Dar de alta paciente	**Walkthrough**
		CU03 – Modificar padecimientos del paciente	
		CU04 – Crear plan alimenticio del paciente	
		CU06 – Programar cita	
		CU08 – Mostrar calendario de citas mensuales	
		CU09 – Crear nuevo registro de diagnóstico	
		CU11 – Añadir imagen de proceso del paciente	
		CU12 – Visualizar historial del paciente	
		CU14 – Exportar plan alimenticio a PDF	

Los entregables serán: Reporte de Recorrido y el Checklist.

2. Documentos de referencia

En esta sección se incluyen todos los artefactos con los que se cuenta para el proceso de la V&V, con un enlace para consultarlos cuando se requiera como apoyo.

Fase	URL	Artefactos
Requisitos	Documento de Diseño	9 Requerimientos Funcionales 2 Requerimientos No Funcionales
Diseño	Documento de Diseño	9 Prototipos 9 Diagramas de Robustez 9 Diagramas de Secuencia
Implementación	Código Fuente	Código del sistema 9 Pruebas unitarias respecto a cada caso de uso

3. Definiciones

En esta sección se detalla la terminología requerida para el correcto entendimiento del documento.

Término	Definición
Artefacto	Se refiere a los productos generados durante el desarrollo de un proyecto, tangibles o intangibles.
Bug o Falla	Se refiere a problemas o defectos que causan que un programa no se comporte como se espera o como fue diseñado originalmente.
Error	Se refiere a un defecto, fallo o problema en el código, diseño, o en la lógica del software que causa un comportamiento no deseado o incorrecto en el programa.
Code smell	Se usa para referirse a ciertos patrones o características en el código fuente que pueden indicar la presencia de problemas subyacentes.

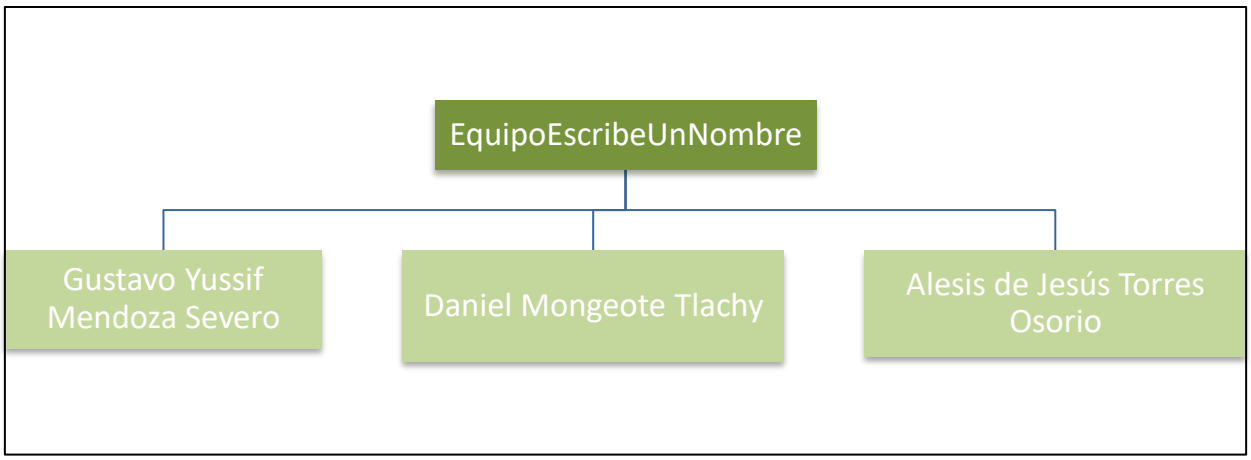
Calidad	Se refiere al grado en que un producto, servicio o proceso cumple con las expectativas, requisitos o estándares establecidos y satisface las necesidades y expectativas de los clientes o usuarios.
Hardware	Se refiere a todos los componentes físicos y tangibles de un sistema informático.
RTF (Revisión Técnica Formal)	Proceso estructurado y meticuloso utilizado en el desarrollo de software y otros proyectos tecnológicos para evaluar y analizar un producto o artefacto técnico, como el diseño de un sistema, el código fuente de un programa, la documentación técnica o cualquier otro entregable relevante.
Inspección	Es una técnica formal de revisión sistemática que implica el examen visual detallado de los productos del desarrollo (como código fuente, documentación o diseños) siguiendo una lista de verificación o criterios predefinidos.
Walkthrough (Recorrido)	Es una técnica no formal de revisión sistemática cuyo objetivo es identificar posibles errores, inconsistencias, problemas de diseño o deficiencias en el producto en una etapa temprana del proceso de desarrollo.
Verificación	El proceso que evalúa un sistema o componente para determinar en qué grado los productos generados en una fase del desarrollo satisfacen las condiciones establecidas al inicio de esta fase, en las etapas previas (las entradas).
Validación	El proceso que evalúa un sistema o componente durante o al final del proceso de desarrollo, para determinar en qué medida satisface la especificación de requisitos del producto y las expectativas que el cliente deposita en ellos.
Stakeholders	Partes interesadas que pueden influir o verse afectadas por el proyecto.

4. Descripción general de Verificación y Validación

En esta sección se describen las actividades generales de lo realizado en el proceso V&V.

4.1.1 Organización

En este subapartado se muestra el organigrama de la organización, representando el nombre de cada miembro del equipo.



4.1.2 Stakeholders

En este apartado se describen los principales interesados en el desarrollo, despliegue y cumplimiento de la V&V del sistema.

Stakeholder	Descripción	Requisitos sobre la V&V	Justificación
Equipo de Desarrollo de Software Katherine Bautista Márquez, José Márquez Reyes Rodríguez	Equipo responsable del desarrollo e implementación del sistema.	Verificar que el sistema cumpla con las funcionalidades solicitadas para garantizar una buena calidad.	Como desarrolladores principales, son los responsables directos de asegurar que el código y las funcionalidades cumplan con los requisitos establecidos y mantengan altos estándares de calidad.
Profesor de Desarrollo de Software Mario Alberto Hernández Pérez	Académico encargado de supervisar y guía del proceso de desarrollo del software.	Verificar que el software es útil y cumple con sus objetivos.	Su experiencia y conocimiento son fundamentales para evaluar la efectividad y utilidad práctica del software desde una perspectiva académica y profesional.
Equipo EscribeUnNombre Gustavo Yussif Mendoza Severo, Daniel Mongeote Tlachy & Alesis de Jesús Torres Osorio	Equipo encargado de la verificación y validación del proyecto.	Validar y Verificar el cumplimiento de la V&V durante todo el desarrollo que tuvo el sistema por parte del Equipo de Desarrollo de Software.	Su rol como equipo independiente de V&V es crucial para garantizar una evaluación objetiva y completa del proceso de desarrollo y el producto final.
Profesora de V&V de Software Elizabeth Murrieta Sangabriel	Académica experta en verificación y validación de software.	Revisar que el proceso de V&V sea apropiado y realizado de manera adecuada.	Su supervisión asegura que las prácticas de V&V se apliquen correctamente y cumplan con los estándares académicos y profesionales requeridos.

4.2. Calendario Maestro

En esta sección se ilustra las tareas y/o actividades a llevarse a cabo durante las etapas del proceso de desarrollo de software a verificar y validar. Así mismo, se representarán las fechas y tiempos sobre los cuales serán llevadas a cabo estas actividades.

	Nombre	Duracion	Inicio	Terminado
1	**Análisis y planificación de V&V**	**12 days?**	**23/09/24 08:00 AM**	**8/10/24 05:00 PM**
2	Estimación de recursos	3 days?	23/09/24 08:00 AM	25/09/24 05:00 PM
3	Desarrollo del EDT	2 days?	26/09/24 08:00 AM	27/09/24 05:00 PM
4	**Elaboración del Plan de V&V**	**7 days?**	**30/09/24 08:00 AM**	**8/10/24 05:00 PM**
5	Definición de roles y responsabilidades	3 days?	30/09/24 08:00 AM	2/10/24 05:00 PM
6	Establecimiento de cronograma de actividades	2 days?	5/10/24 08:00 AM	8/10/24 05:00 PM
7	**Requisitos (inspección)**	**29 days?**	**17/09/24 08:00 AM**	**25/10/24 05:00 PM**
8	**Verificación de Requisitos Funcionales**	**22 days?**	**17/09/24 08:00 AM**	**16/10/24 05:00 PM**
9	CU01 – Dar de alta paciente	5 days?	7/10/24 08:00 AM	11/10/24 05:00 PM
10	CU03 – Modificar padecimientos del paciente	3 days?	12/10/24 08:00 AM	16/10/24 05:00 PM
11	CU04 – Crear plan alimenticio del paciente	1 day?	17/09/24 08:00 AM	17/09/24 05:00 PM
12	CU06 – Programar cita	1 day?	18/09/24 08:00 AM	18/09/24 05:00 PM
13	CU08 – Mostrar calendario de citas mensuales	1 day?	19/09/24 08:00 AM	19/09/24 05:00 PM
14	CU09 – Crear nuevo registro de diagnóstico	1 day?	20/09/24 08:00 AM	20/09/24 05:00 PM
15	CU11 – Añadir imagen de proceso del paciente	1 day?	21/09/24 08:00 AM	23/09/24 05:00 PM
16	CU12 – Visualizar historial del paciente	1 day?	22/09/24 08:00 AM	23/09/24 05:00 PM
17	CU14 – Exportar plan alimenticio a PDF	1 day?	23/09/24 08:00 AM	23/09/24 05:00 PM
18	**Verificación de Requisitos No Funcionales**	**6 days?**	**17/10/24 08:00 AM**	**24/10/24 05:00 PM**
19	**Desempeño**	**1 day?**	**17/10/24 08:00 AM**	**17/10/24 05:00 PM**
20	RNF-03: El sistema deberá proporcionar al nutricionista una interfaz intuitiva y.	1 day?	17/10/24 08:00 AM	17/10/24 05:00 PM
21	RNF-07: El tiempo de respuesta del sistema para la carga de imágenes no deb.	1 day?	17/10/24 08:00 AM	17/10/24 05:00 PM
22	**Seguridad**	**1 day?**	**24/10/24 08:00 AM**	**24/10/24 05:00 PM**
23	RNF-02: El sistema deberá garantizar la confidencialidad de la información del	1 day?	24/10/24 08:00 AM	24/10/24 05:00 PM
24	RNF-08: Después de tres intentos fallidos de inicio de sesión, el sistema aplica.	1 day?	24/10/24 08:00 AM	24/10/24 05:00 PM
25	**Entregables**	**1 day?**	**25/10/24 08:00 AM**	**25/10/24 05:00 PM**
26	Reporte de inspección TSP formato INS	1 day?	25/10/24 08:00 AM	25/10/24 05:00 PM
27	Reporte de junta TSP formato MTQ	1 day?	25/10/24 08:00 AM	25/10/24 05:00 PM
28	Checklist	1 day?	25/10/24 08:00 AM	25/10/24 05:00 PM
29	**Diseño (RTF)**	**36 days?**	**23/09/24 08:00 AM**	**11/11/24 05:00 PM**
30	**Inspección de prototipos de interfaz, Diagramas de robustez y secuencia**	**11 days?**	**27/10/24 09:00 AM**	**11/11/24 05:00 PM**
31	CU01 – Dar de alta paciente	3 days?	27/10/24 09:00 AM	30/10/24 05:00 PM
32	CU03 – Modificar padecimientos del paciente	1.875 days?	31/10/24 09:00 AM	1/11/24 05:00 PM
33	CU04 – Crear plan alimenticio del paciente	0 days?	2/11/24 09:00 AM	4/11/24 05:00 PM
34	CU06 – Programar cita	0.875 days?	4/11/24 09:00 AM	4/11/24 05:00 PM
35	CU08 – Mostrar calendario de citas mensuales	0.875 days?	5/11/24 09:00 AM	5/11/24 05:00 PM
36	CU09 – Crear nuevo registro de diagnóstico	0.875 days?	6/11/24 09:00 AM	6/11/24 05:00 PM
37	CU11 – Añadir imagen de proceso del paciente	0.875 days?	7/11/24 09:00 AM	7/11/24 05:00 PM
38	CU12 – Visualizar historial del paciente	0.875 days?	8/11/24 09:00 AM	8/11/24 05:00 PM
39	CU14 – Exportar plan alimenticio a PDF	0 days?	9/11/24 09:00 AM	11/11/24 05:00 PM
40	**Entregables**	**36 days?**	**23/09/24 08:00 AM**	**11/11/24 05:00 PM**
41	Reporte de RTF	3.875 days?	4/11/24 09:00 AM	7/11/24 05:00 PM
42	Reporte de junta TSP formato MTQ	1.875 days?	8/11/24 09:00 AM	11/11/24 05:00 PM
43	Checklist	1 day?	23/09/24 08:00 AM	23/09/24 05:00 PM
44	**Implementación (Recorrido)**	**14.875 day...**	**11/11/24 09:00 AM**	**29/11/24 05:00 PM**
45	**Revisión de códigoy pruebas unitarias**	**7.875 days?**	**11/11/24 09:00 AM**	**20/11/24 05:00 PM**
46	CU01 – Dar de alta paciente	0.875 days?	11/11/24 09:00 AM	11/11/24 05:00 PM
47	CU03 – Modificar padecimientos del paciente	0.875 days?	12/11/24 09:00 AM	12/11/24 05:00 PM
48	CU04 – Crear plan alimenticio del paciente	0.875 days?	13/11/24 09:00 AM	13/11/24 05:00 PM
49	CU06 – Programar cita	0.875 days?	14/11/24 09:00 AM	14/11/24 05:00 PM
50	CU08 – Mostrar calendario de citas mensuales	0.875 days?	15/11/24 09:00 AM	15/11/24 05:00 PM
51	CU09 – Crear nuevo registro de diagnóstico	0 days?	16/11/24 09:00 AM	18/11/24 05:00 PM
52	CU11 – Añadir imagen de proceso del paciente	0.875 days?	18/11/24 09:00 AM	18/11/24 05:00 PM
53	CU12 – Visualizar historial del paciente	0.875 days?	19/11/24 09:00 AM	19/11/24 05:00 PM
54	CU14 – Exportar plan alimenticio a PDF	0.875 days?	20/11/24 09:00 AM	20/11/24 05:00 PM
55	**Entregables**	**6.875 days?**	**21/11/24 09:00 AM**	**29/11/24 05:00 PM**
56	Checklist	2.875 days?	21/11/24 09:00 AM	25/11/24 05:00 PM
57	Reporte de recorrido	3.875 days?	26/11/24 09:00 AM	29/11/24 05:00 PM

4.3. Resumen de recursos

En este sub apartado se especifican los recursos con los que se cuentan para realizar la V&V.

4.3.1 Recursos Humanos

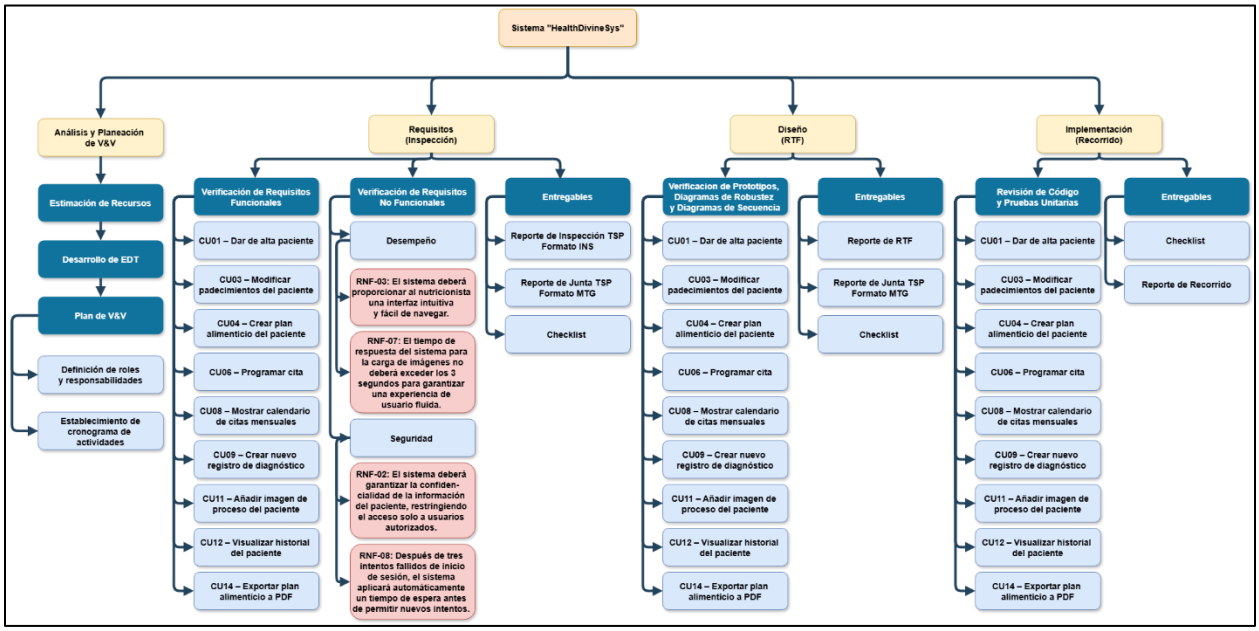
Integrante	Roles	Cantidad de tiempo disponible (hrs)
Gustavo Yussif	Moderador, Líder de revisión y Revisor	240
Daniel Mongeote	Anotador, Cronometrista y Revisor	240
Alesis Torres	Productor, Documentador y Revisor	240

4.3.2 Recursos técnicos

Recurso	Cantidad	Unidad de medida	Justificación
Computadora Personal/Laptop	3	Cantidad	Cada integrante del equipo de desarrollo cuenta con este recurso para realizar los documentos necesarios de las tareas de la V&V,
Internet	3	Meses	Cada integrante del equipo de desarrollo cuenta con este recurso para enviar los documentos necesarios a través de la red.
Luz	3	Meses	Cada integrante del equipo de desarrollo cuenta con este recurso para utilizar sus respectivos recursos electrónicos.

4.3.3 Estructura del Desglose de Trabajo

En este apartado se presenta el diagrama correspondiente a la *Estructura del Desglose de Trabajo*, la cual detalla el desglose de actividades por realizar para aplicar la V&V en el sistema.



4.4. Responsabilidades

A continuación, se detallan las responsabilidades asignadas a cada miembro del equipo de V&V según las diferentes técnicas de revisión a implementar (Inspección, Revisión Técnica Formal y Recorrido), así como la descripción específica de cada rol para asegurar una ejecución efectiva del proceso de verificación y validación.

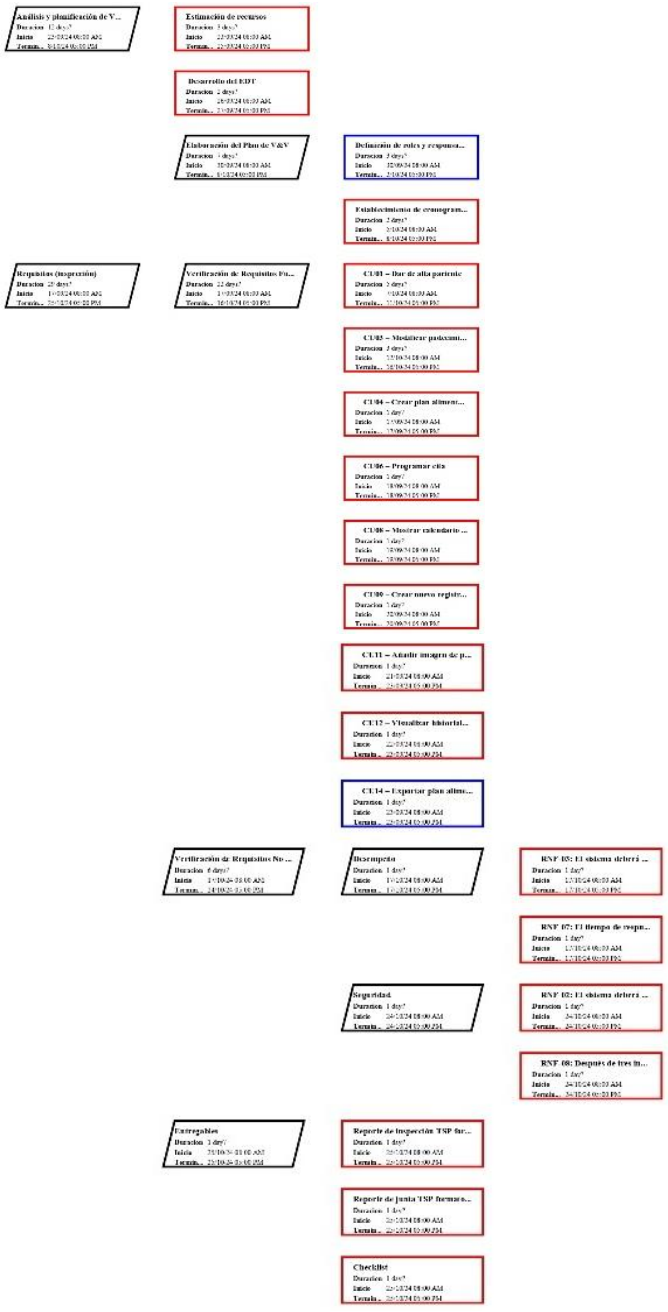
Técnica	Rol	Responsable
	Productor	Alesis de Jesús Torres Osorio

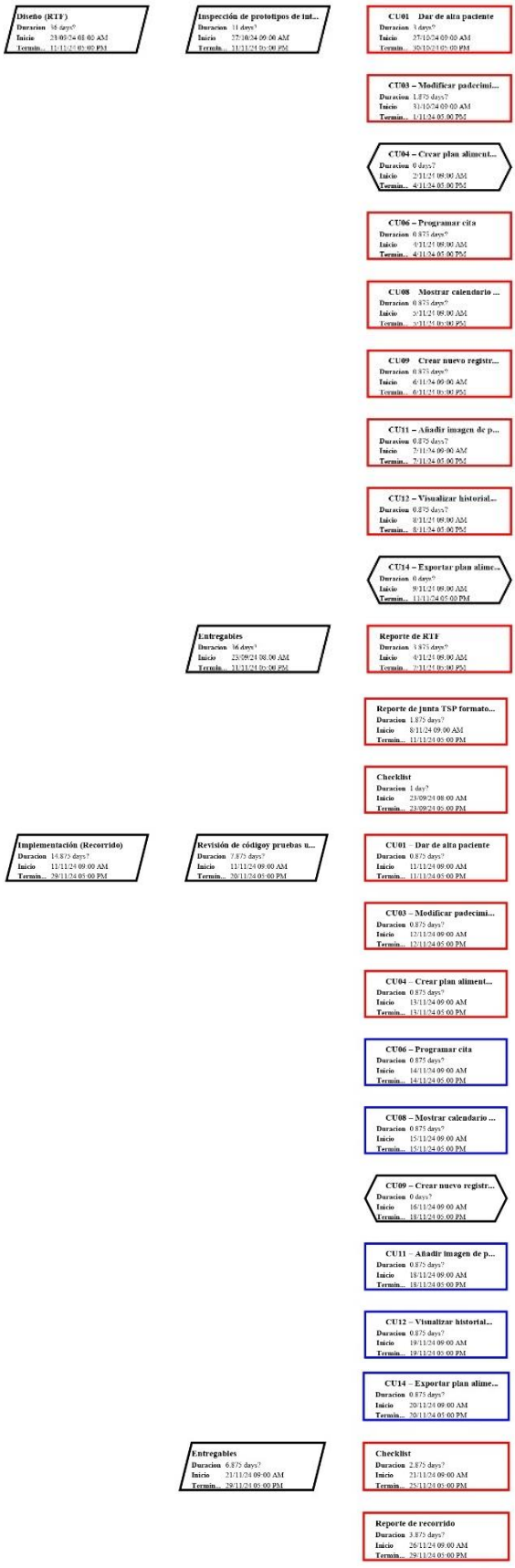
Inspección	Moderador	Gustavo Yussif Mendoza Severo
	Anotador	Daniel Mongeote Tlachy
	Cronometrista	Daniel Mongeote Tlachy
Revisión Técnica Formal	Responsible de la decision	Gustavo Yussif Mendoza Severo
	Líder de revisión	Gustavo Yussif Mendoza Severo
	Documentador	Alesis de Jesús Torres Osorio
	Revisores	Gustavo Yussif Mendoza Severo, Daniel Mongeote Tlachy & Alesis de Jesús Torres Osorio
Walkthroug	Autor	Katherine Bautista Márquez & José Márquez Reyes Rodríguez
	Revisor	Gustavo Yussif Mendoza Severo, Daniel Mongeote Tlachy & Alesis de Jesús Torres Osorio

Rol	Responsabilidades
Productor	Responsable de garantizar que el producto a inspeccionar se encuentre listo para la inspección y resuelve los problemas que se vayan identificando de la manera más rápida posible.
Moderador	Encargado de la repartición de tareas de inspección dentro del equipo de trabajo, garantizando que cada miembro cuente con las habilidades y el tiempo para realizarlas.
Anotador	Responsable de tomar notas durante la sesión de revisión formal. Esto incluye registrar los problemas identificados, las discusiones relevantes, las decisiones tomadas y cualquier acción de seguimiento requerida.
Cronometrista	Responsable de controlar el tiempo durante las sesiones de revisión, asegurando que se mantenga el cronograma establecido y que las discusiones no se extiendan más allá del tiempo asignado.
Responsable de la decisión	Se encarga de determinar si los objetivos de la revisión fueron dados a conocer.
Líder de revisión	Asegura que las tareas administrativas pertenecientes a la revisión sean completadas.
Documentador	Documenta anomalías, objetos de acción, decisiones, y recomendaciones del equipo de revisión.
Revisor	Se encarga de examinar el producto o documentación en busca de defectos, verificar el cumplimiento de estándares y proporcionar retroalimentación constructiva durante la revisión.
Autor	Es la persona que creó o mantiene el producto que está siendo revisado. Es responsable de explicar el producto y responder preguntas durante la revisión, así como de implementar las correcciones necesarias identificadas durante el proceso.

4.4.1 Diagrama de Red

En este apartado se muestra el Diagrama de red, la cual representa visualmente la secuencia y las dependencias entre las actividades del proyecto.




Diseño (RTF)
Duracion 16 days?
Inicio 23/09/24 08:00 AM
Termin... 11/11/24 05:00 PM
Inspección de prototipos de int...
Duracion 11 days?
Inicio 27/10/24 09:00 AM
Termin... 11/11/24 05:00 PM
CU01 Dar de alta paciente
Duracion 3 days?
Inicio 27/10/24 09:00 AM
Termin... 29/10/24 05:00 PM
CU03 – Modificar padecimi...
Duracion 1.875 days?
Inicio 31/10/24 09:00 AM
Termin... 1/11/24 05:00 PM
CU04 – Crear plan alimenti...
Duracion 0 days?
Inicio 2/11/24 09:00 AM
Termin... 4/11/24 05:00 PM
CU06 – Programar cita
Duracion 0.875 days?
Inicio 4/11/24 09:00 AM
Termin... 4/11/24 05:00 PM
CU08 Mostrar calendario ...
Duracion 0.875 days?
Inicio 5/11/24 09:00 AM
Termin... 5/11/24 05:00 PM
CU09 Crear nuevo registr...
Duracion 0.875 days?
Inicio 6/11/24 09:00 AM
Termin... 6/11/24 05:00 PM
CU11 – Añadir imagen de p...
Duracion 0.875 days?
Inicio 7/11/24 09:00 AM
Termin... 7/11/24 05:00 PM
CU12 – Visualizar historial...
Duracion 0.875 days?
Inicio 8/11/24 09:00 AM
Termin... 8/11/24 05:00 PM
CU14 – Exportar plan alime...
Duracion 0 days?
Inicio 8/11/24 09:00 AM
Termin... 11/11/24 05:00 PM
Entregables
Duracion 16 days?
Inicio 23/09/24 08:00 AM
Termin... 11/11/24 05:00 PM
Reporte de RTF
Duracion 3.875 days?
Inicio 4/11/24 09:00 AM
Termin... 7/11/24 05:00 PM
Reporte de junta TSP formato...
Duracion 1.875 days?
Inicio 8/11/24 09:00 AM
Termin... 11/11/24 05:00 PM
Checklist
Duracion 1 day?
Inicio 23/09/24 08:00 AM
Termin... 23/09/24 05:00 PM
Implementación (Recorrido)
Duracion 14.875 days?
Inicio 11/11/24 09:00 AM
Termin... 29/11/24 05:00 PM
Revisión de códigoy pruebas u...
Duracion 7.875 days?
Inicio 11/11/24 09:00 AM
Termin... 20/11/24 05:00 PM
CU01 – Dar de alta paciente
Duracion 0.875 days?
Inicio 11/11/24 09:00 AM
Termin... 11/11/24 05:00 PM
CU03 – Modificar padecimi...
Duracion 0.875 days?
Inicio 12/11/24 09:00 AM
Termin... 12/11/24 05:00 PM
CU04 – Crear plan aliment...
Duracion 0.875 days?
Inicio 13/11/24 09:00 AM
Termin... 13/11/24 05:00 PM
CU06 – Programar cita
Duracion 0.875 days?
Inicio 14/11/24 09:00 AM
Termin... 14/11/24 05:00 PM
CU08 – Mostrar calendario ...
Duracion 0.875 days?
Inicio 15/11/24 09:00 AM
Termin... 15/11/24 05:00 PM
CU09 – Crear nuevo registr...
Duracion 0 days?
Inicio 18/11/24 09:00 AM
Termin... 18/11/24 05:00 PM
CU11 – Añadir imagen de p...
Duracion 0.875 days?
Inicio 18/11/24 09:00 AM
Termin... 18/11/24 05:00 PM
CU12 – Visualizar historial...
Duracion 0.875 days?
Inicio 19/11/24 09:00 AM
Termin... 19/11/24 05:00 PM
CU14 – Exportar plan alime...
Duracion 0.875 days?
Inicio 20/11/24 09:00 AM
Termin... 20/11/24 05:00 PM
Entregables
Duracion 6.875 days?
Inicio 21/11/24 09:00 AM
Termin... 29/11/24 05:00 PM
Checklist
Duracion 2.875 days?
Inicio 21/11/24 09:00 AM
Termin... 25/11/24 05:00 PM
Reporte de recorrido
Duracion 3.875 days?
Inicio 26/11/24 09:00 AM
Termin... 29/11/24 05:00 PM

4.5. Herramientas, técnicas y métodos aplicados al proceso de V&V

Usaremos "Check list" o también conocidas como listas de cotejo, basándonos en el estándar IEEE Std 1012-2016 como rubricas para la evaluación de los artefactos a los que se le apliquen dicha técnica. La plantilla se encuentra en el Anexo A de este documento.

Técnica	Descripción
RTF	Su objetivo es descubrir errores en la función, la lógica o la implementación de cualquier producto del software, verificar que satisface sus especificaciones, que se ajusta a los estándares establecidos, señalando las posibles desviaciones detectadas. Es un proceso de revisión riguroso, su objetivo es llegar a detectar lo antes posible, potenciales defectos o desviaciones en los productos que se van generando a lo largo del desarrollo.
Inspección	Se centra en la identificación de defectos en un documento o producto de software, en un proceso formal y estructurado. Durante las inspecciones se analizan y verifican los requisitos del sistema, los modelos de diseño, el código fuente del programa e incluso las pruebas de sistema propuestas.
Walkthrough	Su principal objetivo es encontrar conflictos en el producto que se revisa, de forma que puedan plantearse alternativas y los participantes (revisores) aumenten su conocimiento del producto en cuestión.

5. Procesos de la verificación y validación

5.1. Procesos, actividades y tareas de la Verificación y Validación de Software

5.1.1 Verificación y Validación de requerimientos de software

En esta sub sección, se especifica cómo es que será verificada y validada la fase de requerimientos de software correspondiente al proyecto.

5.1.2 Verificación y Validación de diseño de software

En esta sub sección, se muestra cómo es que será verificada y validada la fase de diseño de software correspondiente al proyecto. Esto a través de técnicas, métodos y documentos de referencia ya mencionados.

6. Reporte de requisitos y validación

6.1. Reporte de tareas

En esta sub sección, se realizarán reportes de aquellas tareas o actividades que le corresponden a los diferentes miembros con respecto a la V&V. Esto, para identificar tareas pendientes, tareas finalizadas en qué tiempo se llevaron a cabo, etc.

Inspección

CheckList

Nombre	Lista de verificación	**Fecha**	26/11/2024
Equipo	EscribeUnNombre	**Instructor/Gerente**	Elizabeth Murrieta Sangabriel
Parte/Nivel	Requisitos	**Ciclo**	-
Hora inicio	19:47	**Hora fin**	20:41

Propósito	Lista de verificación para los documentos de Visión y Alcance y Requisitos funcionales.
General	Revisar el documento completo para cada categoría de lista, no trate de revisar más de una categoría a la vez. A medida que vaya completando cada etapa de revisión, marque el elemento en el cuadro de laderecha. Completar la lista de verificación para un archivo antes de examinar los próximos.

Verificación de Requisitos Funcionales

Gustavo Yussif Mendoza Severo			
Pregunta	**Sí**	**No**	**Comentarios**
CU01. ¿El proceso para dar de alta un paciente incluye campos obligatorios como nombre, edad y contacto?	X		Los campos básicos están definidos, pero falta validar datos como formato de fecha y números telefónicos.
CU01. ¿Se contempla un manejo adecuado de errores al registrar un paciente?		X	No se detalla cómo se gestionan errores de conexión o fallos al guardar los datos en el sistema.
CU03. ¿La modificación de padecimientos del paciente considera la validación de los datos ingresados?	X		Aunque valida los datos ingresados, no menciona restricciones específicas para evitar errores comunes.
CU03. ¿Se permite visualizar un historial de modificaciones de los padecimientos?		X	No hay una forma clara de consultar quién realizó cambios ni cuándo se efectuaron.

CU04. ¿El requisito para crear un plan alimenticio considera la personalización según el perfil del paciente?	X		Ofrece opciones personalizadas, pero no detalla si se puede ajustar el plan según cambios en tiempo real.
CU04. ¿Es posible editar o eliminar un plan alimenticio una vez creado?		X	Falta incluir un flujo de trabajo específico para actualizar o descartar un plan creado erróneamente.
CU06. ¿El proceso para programar citas incluye validaciones de disponibilidad del horario?	X		Permite validar horarios disponibles, pero no incluye integración con calendarios externos.
CU08. ¿El calendario mensual permite la visualización de múltiples citas en un día?	X		Muestra citas múltiples, pero no ofrece una vista resumida que facilite la gestión diaria.
CU09. ¿La creación de un registro de diagnóstico incluye campos obligatorios como descripción y fecha?	X		Los campos son claros, pero no se menciona un mecanismo para adjuntar imágenes relacionadas.
CU11. ¿El sistema permite la carga de imágenes de diferentes formatos?	X		Compatible con formatos populares, pero falta soporte para documentos como PDF con múltiples páginas.
CU12. ¿El historial del paciente se puede filtrar por fechas o tipo de registros?		X	No hay herramientas de filtrado, lo que dificulta localizar información específica rápidamente.
CU14. ¿El proceso de exportación a PDF incluye un formato legible y con buena estructura visual?	X		Los PDF exportados son claros, pero podrían incluir un índice para facilitar la navegación.

Verificación de Requisitos No Funcionales

Gustavo Yussif Mendoza Severo			
Pregunta	**Sí**	**No**	**Comentarios**
RNF-03. ¿El sistema ofrece una interfaz intuitiva y fácil de usar para el nutriólogo?	X		El diseño es funcional, pero no incluye accesibilidad para usuarios con dificultades visuales.
RNF-03. ¿Se ha probado la interfaz con usuarios finales para validar la usabilidad?		X	No se mencionan pruebas de usuario finales ni ajustes basados en su retroalimentación.
RNF-07. ¿El tiempo de respuesta para cargar imágenes cumple con el límite de 3 segundos?	X		El sistema cumple con el tiempo de respuesta, pero no se especifica cómo afectará la carga de red lenta.
RNF-07. ¿Se incluye alguna especificación de rendimiento bajo cargas de usuarios simultáneos?		X	No hay información sobre pruebas de estrés con múltiples usuarios trabajando simultáneamente.
RNF-02. ¿El sistema garantiza la confidencialidad de los datos sensibles del paciente?	X		La seguridad es sólida, pero no se detalla cómo se auditan los accesos o los datos compartidos.
RNF-02. ¿Se especifica cómo se gestionarán las credenciales de acceso al sistema?		X	Falta información sobre gestión de contraseñas y políticas de expiración de sesiones.

RNF-08. ¿El sistema bloquea automáticamente el acceso después de varios intentos fallidos?	X		Se implementa bloqueo tras tres intentos fallidos, pero no especifica si el tiempo de bloqueo es ajustable.
RNF-08. ¿Se notifica al usuario sobre el bloqueo por intentos fallidos?		X	No se envía un mensaje al usuario bloqueado, lo que podría generar confusión.

Verificación de Requisitos Funcionales

Daniel Mongeote Tlachy			
Pregunta	**Sí**	**No**	**Comentarios**
CU01. ¿El proceso para dar de alta un paciente incluye campos obligatorios como nombre, edad y contacto?	X		Se especifican los campos mínimos requeridos, pero falta validar el formato de algunos datos.
CU01. ¿Se contempla un manejo adecuado de errores al registrar un paciente?		X	No se describe cómo se notifican errores como duplicidad o campos vacíos.
CU03. ¿La modificación de padecimientos del paciente considera la validación de los datos ingresados?	X		Sí, se menciona la validación de entrada, pero no se incluyen ejemplos de valores esperados.
CU03. ¿Se permite visualizar un historial de modificaciones de los padecimientos?		X	No se menciona un registro histórico de cambios, lo cual puede ser relevante para auditoría.
CU04. ¿El requisito para crear un plan alimenticio considera la personalización según el perfil del paciente?	X		Se menciona que el plan es personalizado con base en datos como peso y actividad física.
CU04. ¿Es posible editar o eliminar un plan alimenticio una vez creado?		X	No se incluye información sobre la edición o eliminación de planes existentes.
CU06. ¿El proceso para programar citas incluye validaciones de disponibilidad del horario?	X		Sí, se describe que el sistema verifica disponibilidad antes de confirmar.
CU08. ¿El calendario mensual permite la visualización de múltiples citas en un día?	X		Se indica que puede mostrar varias citas, pero falta un límite máximo por día.
CU09. ¿La creación de un registro de diagnóstico incluye campos obligatorios como descripción y fecha?	X		Los campos son suficientes, pero falta incluir datos opcionales como adjuntos médicos.
CU11. ¿El sistema permite la carga de imágenes de diferentes formatos?	X		Es compatible con formatos comunes como JPG y PNG, pero no menciona PDF o TIFF.
CU12. ¿El historial del paciente se puede filtrar por fechas o tipo de registros?		X	No se menciona ningún mecanismo de filtrado para facilitar búsquedas específicas.
CU14. ¿El proceso de exportación a PDF incluye un formato legible y con buena estructura visual?	X		Se detalla que los planes son exportados con encabezados claros y tablas bien formateadas.

Verificación de Requisitos No Funcionales

Daniel Mongeote Tlachy

Pregunta	Sí	No	Comentarios
RNF-03. ¿El sistema ofrece una interfaz intuitiva y fácil de usar para el nutriólogo?	X		La interfaz es clara y ofrece menús desplegables para navegar fácilmente entre opciones.
RNF-03. ¿Se ha probado la interfaz con usuarios finales para validar la usabilidad?		X	No se menciona si se realizaron pruebas de usabilidad con usuarios reales.
RNF-07. ¿El tiempo de respuesta para cargar imágenes cumple con el límite de 3 segundos?	X		En las pruebas realizadas, los tiempos fueron consistentes dentro del límite especificado.
RNF-07. ¿Se incluye alguna especificación de rendimiento bajo cargas de usuarios simultáneos?		X	No se especifica cómo el sistema manejará cargas altas de trabajo.
RNF-02. ¿El sistema garantiza la confidencialidad de los datos sensibles del paciente?	X		Se menciona cifrado de datos y restricciones de acceso por roles de usuario.
RNF-02. ¿Se especifica cómo se gestionarán las credenciales de acceso al sistema?		X	No se incluye información sobre mecanismos como autenticación de dos factores o recuperación.
RNF-08. ¿El sistema bloquea automáticamente el acceso después de varios intentos fallidos?	X		El sistema se bloquea después de tres intentos y requiere intervención administrativa.
RNF-08. ¿Se notifica al usuario sobre el bloqueo por intentos fallidos?		X	No se especifica un mensaje de notificación al usuario afectado.

Verificación de Requisitos Funcionales

Alesis de Jesús Torres Osorio			
Pregunta	**Sí**	**No**	**Comentarios**
CU01. ¿El proceso para dar de alta un paciente incluye campos obligatorios como nombre, edad y contacto?	X		Los datos obligatorios están cubiertos, pero no se permite registrar datos opcionales como el seguro médico.
CU01. ¿Se contempla un manejo adecuado de errores al registrar un paciente?		X	No hay mensajes claros en caso de errores de duplicidad o formatos incorrectos.
CU03. ¿La modificación de padecimientos del paciente considera la validación de los datos ingresados?	X		La validación de datos está documentada, pero no se mencionan restricciones específicas por tipo de campo.
CU03. ¿Se permite visualizar un historial de modificaciones de los padecimientos?		X	El sistema no muestra quién hizo cambios ni cuándo, lo que limita la trazabilidad.
CU04. ¿El requisito para crear un plan alimenticio considera la personalización según el perfil del paciente?	X		Los planes son personalizables, pero no se menciona si el sistema sugiere ajustes según progresos.
CU04. ¿Es posible editar o eliminar un plan alimenticio una vez creado?		X	No se menciona cómo manejar planes obsoletos o si se requiere autorización para eliminarlos.
CU06. ¿El proceso para programar citas incluye validaciones de disponibilidad del horario?	X		Valida horarios, pero no permite definir tiempos adicionales para preparación o pausas.
CU08. ¿El calendario mensual permite la visualización de múltiples citas en un día?	X		No se detalla cómo mostrar citas simultáneas de diferentes nutriólogos en la misma vista.

CU09. ¿La creación de un registro de diagnóstico incluye campos obligatorios como descripción y fecha?	X		Los campos obligatorios están presentes, pero falta un campo para agregar comentarios personalizados.
CU11. ¿El sistema permite la carga de imágenes de diferentes formatos?	X		Solo menciona soporte para imágenes estándar, pero sería útil admitir escaneos de documentos en PDF.
CU12. ¿El historial del paciente se puede filtrar por fechas o tipo de registros?		X	No hay filtros avanzados ni opciones de exportar búsquedas específicas del historial.
CU14. ¿El proceso de exportación a PDF incluye un formato legible y con buena estructura visual?	X		Los formatos PDF son básicos y carecen de elementos visuales como logotipos o gráficos.

Verificación de Requisitos No Funcionales

Alesis de Jesús Torres Osorio			
Pregunta	**Sí**	**No**	**Comentarios**
RNF-03. ¿El sistema ofrece una interfaz intuitiva y fácil de usar para el nutriólogo?	X		La navegación es adecuada, pero se podrían incluir accesos directos a funciones comunes.
RNF-03. ¿Se ha probado la interfaz con usuarios finales para validar la usabilidad?		X	No hay evidencia de pruebas con usuarios que evalúen la experiencia real del sistema.
RNF-07. ¿El tiempo de respuesta para cargar imágenes cumple con el límite de 3 segundos?	X		Aunque se cumple el tiempo promedio, no se realizaron pruebas con archivos muy grandes.
RNF-07. ¿Se incluye alguna especificación de rendimiento bajo cargas de usuarios simultáneos?		X	No se indica si el sistema soportará adecuadamente múltiples usuarios cargando imágenes simultáneamente.
RNF-02. ¿El sistema garantiza la confidencialidad de los datos sensibles del paciente?	X		Se garantizan los permisos de acceso, pero falta una política clara de respaldo de los datos.
RNF-02. ¿Se especifica cómo se gestionarán las credenciales de acceso al sistema?		X	No se describe un sistema para evitar contraseñas débiles ni para reforzar el cambio periódico.
RNF-08. ¿El sistema bloquea automáticamente el acceso después de varios intentos fallidos?	X		El bloqueo es funcional, pero falta especificar cómo reactivar la cuenta de forma segura.
RNF-08. ¿Se notifica al usuario sobre el bloqueo por intentos fallidos?		X	No se envía ninguna alerta.

Reporte de Inspección TSP – Formato INS

Nombre	Reporte de Inspección TSP	**Fecha**	28/11/2024
Proyecto	HealthDivineSystem	**Equipo**	EscribeUnNombre
Producto	HealthDivineSystem	**Fase**	Única
Moderador	Alesis de Jesús Torres Osorio	**Generador**	-

Tamaño del producto:	257			**Medida de Tamaño:**	LOC
Defectos Totales para A	6	**Defectos Totales para B**	9	**C (# comunes)**	5

Defectos Totales (AB/C)	10.8	**Núm. Descub. (A+B-C)**	10	**Número de Pendientes**	0
Tiempo de la Junta:	27	**Horas Tot. de Inspec.**	0	**Tasa General:**	0.92

Nombre	**Defectos**		**Datos de Preparación**			**Rendimiento Esperado**
	Mayor	**Menor**	**Tam.**	**Tpo**	**Tasa**	
Gustavo Yussif Mendoza Severo	4	1	257	LOC	0.65	0.65
Daniel Mongeote Tlachy	2	5	257	LOC	0.79	0.76
Alesis de Jesús Torres Osorio	2	4	257	LOC	0.65	0.65

Datos de Defectos		**Defectos**		**Ing. (encontrando los def. princip.)**					
No	**Descripción del Defecto**	**May**	**Men**	**Gustavo**	**Daniel**	**Alesis**	**A**	**B**	**C**
1	No se valida el formato correcto del nombre al dar de alta un paciente.	X		X				X	
2	Error al manejar campos duplicados o vacíos durante el registro de pacientes.	X		X				X	
3	Faltan ejemplos de valores esperados al modificar padecimientos.		X		X				X
4	No se registra un historial de modificaciones de padecimientos.	X			X	X	X	X	X
5	No se permite editar o eliminar planes alimenticios una vez creados.	X				X		X	
6	Fallo en la validación de disponibilidad al programar citas.	X		X				X	
7	El calendario no establece un límite máximo de citas por día.	X		X				X	
8	No se permiten adjuntar documentos médicos opcionales en los registros de diagnóstico.	X			X		X		
9	Falta soporte para formatos adicionales de imagen, como PDF o TIFF.		X			X			X
10	No hay mecanismo de filtrado por fechas o tipo en el historial del paciente.		X		X	X	X		X
11	La exportación a PDF tiene errores en la estructura visual en algunos encabezados.		X		X		X		
12	No se realizaron pruebas de usabilidad con usuarios finales.		X	X				X	
13	Rendimiento insuficiente bajo cargas de usuarios simultáneos.		X		X		X		
14	Falta de autenticación de dos factores para las credenciales de acceso.		X		X	X	X	X	X
15	No se notifica al usuario sobre bloqueos automáticos tras varios intentos fallidos de acceso.		X			X		X	
Totales en clase		9					6	9	5
Total de Defectos Únicos		14							

Reporte de Junta TSP – Formato MTG

Moderador:	Elizabeth Murrieta Sangabriel	**Lugar:**	Aula 105 de la FEI	**Fecha:**	28/11/2024
Fecha de la junta:	28/11/2024	**Hora Inicio:**	17:30	**Hora Fin:**	19:05

Tema/Propósito	Revisión de los requisitos del Sistema HealthDivineSystema para la EE de Desarrollo de Software-

Asistentes

Nombre	Rol
Alesis de Jesús Torres Osorio	Responsable de la decisión
Alesis de Jesús Torres Osorio	Líder de revisión
Daniel Mongeote Tlachy	Documentador
Equipo EscribeUnNombre	Revisores

Agenda

Tiempos (min.)			Temas	Líder de Presentacion
Plan	**Inicio**	**Fin**		
10	17:30	17:40	Planificación de la revisión	Alesis de Jesus Torres Osorio
10	17:40	17:50	Presentación de los procedimientos para la revisión	Alesis de Jesus Torres Osorio
5	17:50	17:55	Preparación de la revisión	Daniel Mongeote Tlachy
10	17:55	18:05	Presentación de los recursos del software a revisar	Daniel Mongeote Tlachy
5	18:05	18:10	Junta de la revisión: Apertura	Alesis de Jesus Torres Osorio
30	18:10	18:40	Examinación de los productos	Daniel Mongeote Tlachy
20	18:40	19:00	Señalización de defectos	Gustavo Yussif Mendoza Severo
5	19:00	19:05	Junta de la revisión: Conclusión	Alesis de Jesus Torres Osorio

Decisiones, Acciones e Informes Clave

Qué	Quién	Cuando
El documento presenta una base sólida para el proyecto, cubriendo la mayoría de los aspectos fundamentales esperados en un documento de visión y alcance. Sin embargo, existen áreas de mejora significativas. La ausencia de casos de uso o escenarios de interacción es una omisión notable, ya que estos son cruciales para entender cómo los usuarios interactuarán con el sistema. Además, aunque se han identificado los principales riesgos del proyecto, la falta de un plan de mitigación para estos riesgos es una debilidad importante que debería abordarse.	Alesis de Jesús Torres Osorio	Jueves 28/11/2024

RTF

CheckList

Nombre	Lista de verificación	**Fecha**	29/11/2024
Equipo	EscribeUnNombre	**Instructor/Gerente**	Elizabeth Murrieta Sangabriel
Parte/Nivel	Requisitos	**Ciclo**	-
Hora inicio	12:45	**Hora fin**	14:21

Propósito	Lista de verificación para los entregables de Prototipos, Diagramas de Robustez y Diagramas de Secuencia del sistema, localizados en el documento de Visión y Alcance y Requisitos funcionales.
General	Revisar el documento completo para cada categoría de lista, no trate de revisar más de una categoría a la vez. A medida que vaya completando cada etapa de revisión, marque el elemento en el cuadro de laderecha. Completar la lista de verificación para un archivo antes de examinar los próximos.

Checklist para Prototipos

Gustavo Yussif Mendoza Severo				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El prototipo muestra claramente los campos obligatorios para dar de alta un paciente?	X		Los campos obligatorios están marcados con un asterisco y resaltados visualmente en rojo.
CU01	¿El diseño incluye mensajes de error para entradas incorrectas al registrar datos de pacientes?		X	No se especifican mensajes claros para datos inválidos; esto podría confundir a los usuarios.
CU03	¿El prototipo tiene un diseño intuitivo para modificar padecimientos del paciente?	X		Los campos editables están bien posicionados y tienen etiquetas descriptivas.
CU03	¿Los campos editables están correctamente diferenciados de los no editables en el prototipo?	X		Los campos no editables tienen un color gris, diferenciándose claramente de los editables.
CU04	¿El prototipo permite personalizar planes alimenticios con datos dinámicos como calorías y horarios?	X		La interfaz permite ajustar calorías y horarios de forma flexible a través de controles deslizantes.
CU04	¿Se proporciona una vista previa del plan alimenticio antes de guardarlo en el prototipo?		X	No se incluye una opción para previsualizar el plan antes de confirmarlo.
CU06	¿El prototipo incluye un calendario visual para programar citas?	X		Se utiliza un diseño estilo calendario mensual, fácil de entender y manejar.
CU06	¿El diseño muestra advertencias en caso de solapamiento de citas al programar?		X	No se menciona un mecanismo para notificar conflictos de horarios al usuario.
CU08	¿El prototipo permite visualizar todas las citas de un mes en un formato compacto y organizado?	X		Las citas se presentan en un formato de lista para cada día, manteniendo un diseño ordenado.
CU09	¿Se presentan ejemplos de datos en el prototipo del registro de diagnóstico para facilitar el uso?		X	No se incluyen datos de ejemplo, lo que dificulta la orientación de nuevos usuarios.
CU11	¿El prototipo ofrece un botón claro para cargar imágenes del paciente?	X		El botón tiene un ícono de cámara que lo hace fácilmente identificable.

CU11	¿Se muestra el progreso de carga de imágenes en el prototipo?		X	No hay una barra de progreso para indicar el estado de la carga de imágenes.
CU12	¿El prototipo incluye opciones de búsqueda y filtrado en el historial del paciente?	X		Las opciones de filtrado son claras e incluyen rangos de fecha y palabras clave.
CU14	¿El prototipo tiene una interfaz que confirma la correcta exportación del plan alimenticio a PDF?	X		Una ventana emergente confirma que el archivo se exportó exitosamente con un botón de descarga.

Checklist para Prototipos

Daniel Mongeote Tlachy				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El prototipo muestra claramente los campos obligatorios para dar de alta un paciente?	X		Los campos obligatorios están identificados con un texto adicional que explica su importancia.
CU01	¿El diseño incluye mensajes de error para entradas incorrectas al registrar datos de pacientes?		X	Los errores solo se muestran al intentar guardar, lo que podría frustrar a los usuarios.
CU03	¿El prototipo tiene un diseño intuitivo para modificar padecimientos del paciente?	X		Las opciones son accesibles mediante un menú desplegable fácil de ubicar.
CU03	¿Los campos editables están correctamente diferenciados de los no editables en el prototipo?	X		Los campos editables tienen bordes resaltados, lo que mejora la experiencia visual del usuario.
CU04	¿El prototipo permite personalizar planes alimenticios con datos dinámicos como calorías y horarios?	X		Los datos dinámicos se ajustan automáticamente con base en las preferencias seleccionadas.
CU04	¿Se proporciona una vista previa del plan alimenticio antes de guardarlo en el prototipo?		X	La falta de vista previa puede generar errores en los planes antes de ser guardados.
CU06	¿El prototipo incluye un calendario visual para programar citas?	X		El calendario es interactivo, con opciones para cambiar entre vistas semanales y mensuales.
CU06	¿El diseño muestra advertencias en caso de solapamiento de citas al programar?		X	El prototipo no indica claramente cuándo dos citas se solapan en el mismo horario.
CU08	¿El prototipo permite visualizar todas las citas de un mes en un formato compacto y organizado?	X		Las citas se organizan por colores dependiendo de su estado (confirmadas o pendientes).
CU09	¿Se presentan ejemplos de datos en el prototipo del registro de diagnóstico para facilitar el uso?		X	Se podría incluir un ejemplo precargado como referencia para nuevos usuarios.
CU11	¿El prototipo ofrece un botón claro para cargar imágenes del paciente?	X		El botón tiene texto adicional que aclara su función (e.g., "Cargar Imagen del Paciente").
CU11	¿Se muestra el progreso de carga de imágenes en el prototipo?		X	La carga ocurre en segundo plano sin informar al usuario sobre el estado de la operación.
CU12	¿El prototipo incluye opciones de búsqueda y filtrado en el historial del paciente?	X		Las opciones de búsqueda permiten filtrar resultados según palabras clave y fechas.

CU14	¿El prototipo tiene una interfaz que confirma la correcta exportación del plan alimenticio a PDF?	X		Una notificación emergente informa al usuario que la exportación fue exitosa.

Checklist para Prototipos

Alesis de Jesús Torres Osorio				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El prototipo muestra claramente los campos obligatorios para dar de alta un paciente?	X		Los campos están identificados con un borde en rojo y una leyenda que indica su obligatoriedad.
CU01	¿El diseño incluye mensajes de error para entradas incorrectas al registrar datos de pacientes?		X	La ausencia de validación en tiempo real podría dificultar la detección temprana de errores.
CU03	¿El prototipo tiene un diseño intuitivo para modificar padecimientos del paciente?	X		La edición de padecimientos es simple gracias a un menú con opciones predeterminadas.
CU03	¿Los campos editables están correctamente diferenciados de los no editables en el prototipo?	X		Los campos no editables aparecen en gris, lo que mejora la identificación visual.
CU04	¿El prototipo permite personalizar planes alimenticios con datos dinámicos como calorías y horarios?	X		Los ajustes dinámicos son claros y permiten personalizar fácilmente los planes.
CU04	¿Se proporciona una vista previa del plan alimenticio antes de guardarlo en el prototipo?		X	No se cuenta con una vista previa que permita verificar los datos antes de guardarlos.
CU06	¿El prototipo incluye un calendario visual para programar citas?	X		La funcionalidad del calendario es básica pero cumple con los requisitos mínimos.
CU06	¿El diseño muestra advertencias en caso de solapamiento de citas al programar?		X	No se destacan conflictos de horario de forma automática, lo cual es un área de mejora.
CU08	¿El prototipo permite visualizar todas las citas de un mes en un formato compacto y organizado?	X		Las citas aparecen organizadas por días, pero podrían optimizarse para pantallas más pequeñas.
CU09	¿Se presentan ejemplos de datos en el prototipo del registro de diagnóstico para facilitar el uso?		X	Un registro de muestra puede ayudar a los usuarios a comprender mejor la funcionalidad.
CU11	¿El prototipo ofrece un botón claro para cargar imágenes del paciente?	X		El botón incluye un ícono intuitivo que mejora la accesibilidad visual.
CU11	¿Se muestra el progreso de carga de imágenes en el prototipo?		X	Una barra de progreso podría mejorar la experiencia del usuario al cargar imágenes.
CU12	¿El prototipo incluye opciones de búsqueda y filtrado en el historial del paciente?	X		La búsqueda incluye múltiples filtros que agilizan el acceso a la información.
CU14	¿El prototipo tiene una interfaz que confirma la correcta exportación del plan alimenticio a PDF?	X		Una notificación confirma que el PDF se generó correctamente.

Checklist para Diagramas de Robustez

Gustavo Yussif Mendoza Severo				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El diagrama incluye las entidades necesarias para dar de alta un paciente (e.g., base de datos, validadores)?	X		Se incluyen todas las entidades requeridas para la operación básica, lo que asegura la integridad de los datos.
CU03	¿El diagrama de robustez considera validaciones para modificar campos como nombre o fecha de diagnóstico?	X		El diagrama establece una estructura lógica para verificar la validez de los datos antes de su modificación.
CU03	¿Se muestra cómo el sistema maneja errores al intentar modificar un padecimiento inexistente?		X	El diagrama no muestra el proceso exacto para la gestión de errores en estos casos específicos.
CU04	¿El diagrama incluye reglas de negocio para calcular valores nutricionales automáticamente?	X		Se define claramente cómo se calculan los valores, pero podría beneficiarse de ejemplos más específicos.
CU06	¿El diagrama contempla verificaciones de disponibilidad en el calendario para programar citas?	X		La verificación de disponibilidad está bien representada, pero sería útil añadir validaciones adicionales en situaciones complejas.
CU06	¿Se incluye un actor que confirma los detalles de la cita programada?		X	No se representa un actor adicional que verifique los detalles antes de la confirmación final.
CU08	¿El diagrama representa el flujo para obtener un listado de citas mensuales?	X		El flujo está claramente diagramado, pero falta un proceso de optimización para consultas con gran cantidad de datos.
CU11	¿El diagrama detalla cómo se procesan imágenes para verificar su formato antes de cargarlas?	X		Los procesos de validación de imágenes están bien detallados, garantizando que solo se suban formatos compatibles.
CU12	¿Se considera un mecanismo para filtrar registros en el historial del paciente?	X		El filtro está bien implementado, aunque se podrían agregar opciones adicionales de personalización.
CU14	¿El diagrama incluye la conversión de datos en un formato PDF?		X	No se muestra claramente cómo se realiza la conversión de los datos a formato PDF en el diagrama.

Checklist para Diagramas de Robustez

Daniel Mongeote Tlachy				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El diagrama incluye las entidades necesarias para dar de alta un paciente (e.g., base de datos, validadores)?	X		Las entidades necesarias están bien representadas y permiten una fácil integración con el sistema.
CU03	¿El diagrama de robustez considera validaciones para modificar campos como nombre o fecha de diagnóstico?	X		Se consideran validaciones eficientes para evitar registros erróneos o inconsistentes.
CU03	¿Se muestra cómo el sistema maneja errores al intentar modificar un padecimiento inexistente?		X	No se aborda el flujo de errores para este tipo de situaciones, lo cual podría afectar la robustez.
CU04	¿El diagrama incluye reglas de negocio para calcular valores nutricionales automáticamente?	X		El diagrama presenta las reglas de negocio de manera clara, pero faltan ejemplos específicos.

CU06	¿El diagrama contempla verificaciones de disponibilidad en el calendario para programar citas?	X		La verificación es exhaustiva y permite manejar varias situaciones de disponibilidad.
CU06	¿Se incluye un actor que confirma los detalles de la cita programada?		X	La falta de un actor para la confirmación hace que el proceso sea menos seguro.
CU08	¿El diagrama representa el flujo para obtener un listado de citas mensuales?	X		El flujo es intuitivo y permite al usuario obtener un listado organizado de citas.
CU11	¿El diagrama detalla cómo se procesan imágenes para verificar su formato antes de cargarlas?	X		El proceso de validación es adecuado, pero podría beneficiarse de un paso de revisión adicional.
CU12	¿Se considera un mecanismo para filtrar registros en el historial del paciente?	X		Los filtros en el historial se implementan correctamente, pero podrían mejorarse con opciones avanzadas.
CU14	¿El diagrama incluye la conversión de datos en un formato PDF?		X	No se muestra cómo se realiza la conversión de datos a PDF, lo que es un área de mejora.

Checklist para Diagramas de Robustez

Alesis de Jesús Torres Osorio				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El diagrama incluye las entidades necesarias para dar de alta un paciente (e.g., base de datos, validadores)?	X		El diagrama cubre todas las entidades necesarias, lo que asegura un proceso claro de alta.
CU03	¿El diagrama de robustez considera validaciones para modificar campos como nombre o fecha de diagnóstico?	X		Las validaciones están correctamente representadas, pero se podrían añadir más validaciones de formato.
CU03	¿Se muestra cómo el sistema maneja errores al intentar modificar un padecimiento inexistente?		X	No se muestra el flujo para manejar este tipo de errores específicos, lo que es una omisión importante.
CU04	¿El diagrama incluye reglas de negocio para calcular valores nutricionales automáticamente?	X		Las reglas son claras, pero la lógica de negocio podría beneficiarse de una descripción más profunda.
CU06	¿El diagrama contempla verificaciones de disponibilidad en el calendario para programar citas?	X		Las verificaciones están correctamente representadas, garantizando una asignación de citas eficiente.
CU06	¿Se incluye un actor que confirma los detalles de la cita programada?		X	El actor encargado de la confirmación falta, lo que podría llevar a confusión en la implementación.
CU08	¿El diagrama representa el flujo para obtener un listado de citas mensuales?	X		El flujo de citas mensuales está detallado, pero podría mejorarse con un sistema de paginación.
CU11	¿El diagrama detalla cómo se procesan imágenes para verificar su formato antes de cargarlas?	X		El proceso de validación de formatos está presente, pero debería incluir pasos de recuperación en caso de error.
CU12	¿Se considera un mecanismo para filtrar registros en el historial del paciente?	X		Se proporciona un sistema de filtrado eficiente, pero no se detallan todos los filtros posibles.
CU14	¿El diagrama incluye la conversión de datos en un formato PDF?		X	Falta representar el flujo de conversión a PDF, lo que es esencial para la exportación de documentos.

Checklist para Diagramas de Secuencia

Gustavo Yussif Mendoza Severo				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El diagrama de secuencia detalla el flujo completo para dar de alta un paciente, desde la entrada hasta el almacenamiento?	X		El flujo es completo, pero se podrían especificar con más detalle las interacciones entre actores y el sistema.
CU01	¿Se muestran las interacciones con la base de datos al registrar un nuevo paciente?	X		El diagrama muestra claramente las interacciones, con un buen enfoque en la gestión de datos.
CU03	¿El diagrama incluye mensajes de error en caso de fallos al modificar datos del paciente?		X	No se visualizan los mensajes de error que podrían surgir al intentar modificar los datos del paciente.
CU04	¿El diagrama detalla cómo se calculan y actualizan las calorías en el plan alimenticio del paciente?	X		Los cálculos están representados, pero sería útil incluir ejemplos de entrada y salida de datos.
CU06	¿El diagrama incluye la interacción entre el usuario y el calendario para programar citas?	X		Se representan las interacciones de forma fluida y fácil de seguir, mostrando el flujo adecuado.
CU06	¿Se representa una confirmación al usuario después de agendar una cita correctamente?	X		La confirmación se muestra claramente en el diagrama, lo que facilita la experiencia del usuario.
CU08	¿El diagrama muestra cómo se obtienen todas las citas para un mes y se visualizan al usuario?	X		El proceso de visualización está bien explicado, pero podría mejorarse para optimizar la búsqueda.
CU09	¿Se detalla el flujo para agregar un nuevo registro de diagnóstico?	X		El flujo es claro y cubre todos los pasos esenciales para la creación de un registro.
CU11	¿El diagrama incluye el manejo de errores al cargar una imagen incompatible?		X	No se abordan los posibles errores derivados de imágenes incompatibles.
CU12	¿El diagrama de secuencia representa cómo se filtra el historial por rango de fechas?	X		El filtrado está bien documentado, pero podría mejorar con una opción más dinámica de rango de fechas.
CU14	¿El diagrama muestra las interacciones necesarias para exportar un archivo PDF?		X	No se representa el proceso completo para exportar datos en formato PDF.

Checklist para Diagramas de Secuencia

Daniel Mongeote Tlachy				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El diagrama de secuencia detalla el flujo completo para dar de alta un paciente, desde la entrada hasta el almacenamiento?	X		El flujo de alta de paciente está completo, aunque algunos pasos podrían ser más detallados.
CU01	¿Se muestran las interacciones con la base de datos al registrar un nuevo paciente?	X		Se documentan correctamente las interacciones con la base de datos, garantizando la integridad de los datos.

CU03	¿El diagrama incluye mensajes de error en caso de fallos al modificar datos del paciente?		X	No se representan los mensajes de error que el sistema debería enviar en caso de fallos.
CU04	¿El diagrama detalla cómo se calculan y actualizan las calorías en el plan alimenticio del paciente?	X		Se detalla cómo las calorías se calculan automáticamente, pero puede mejorarse con más condiciones.
CU06	¿El diagrama incluye la interacción entre el usuario y el calendario para programar citas?	X		La interacción está bien representada, con una clara división entre los pasos del usuario y el sistema.
CU06	¿Se representa una confirmación al usuario después de agendar una cita correctamente?	X		La confirmación del agendamiento es explícita, asegurando que el usuario está informado de la acción realizada.
CU08	¿El diagrama muestra cómo se obtienen todas las citas para un mes y se visualizan al usuario?	X		El diagrama muestra el proceso de recuperación de citas, pero sería útil incluir cómo manejar citas eliminadas.
CU09	¿Se detalla el flujo para agregar un nuevo registro de diagnóstico?	X		El flujo es claro y cubre las acciones necesarias para agregar un diagnóstico completo.
CU11	¿El diagrama incluye el manejo de errores al cargar una imagen incompatible?		X	No se detalla un flujo adecuado para manejar errores de carga de imágenes incompatibles.
CU12	¿El diagrama de secuencia representa cómo se filtra el historial por rango de fechas?	X		El diagrama representa correctamente el proceso de filtrado por fechas, con detalles claros de cada paso.
CU14	¿El diagrama muestra las interacciones necesarias para exportar un archivo PDF?		X	Falta el proceso completo para la exportación de un archivo PDF, lo cual necesita más detalles.

Checklist para Diagramas de Secuencia

Alesis de Jesús Torres Osorio				
Caso de Uso	**Pregunta**	**Sí**	**No**	**Comentarios**
CU01	¿El diagrama de secuencia detalla el flujo completo para dar de alta un paciente, desde la entrada hasta el almacenamiento?	X		El flujo cubre todas las etapas desde la captura de datos hasta la creación del registro.
CU01	¿Se muestran las interacciones con la base de datos al registrar un nuevo paciente?	X		Se muestra claramente cómo interactúan las entidades con la base de datos, asegurando coherencia en la información.
CU03	¿El diagrama incluye mensajes de error en caso de fallos al modificar datos del paciente?		X	No se presentan los mensajes de error específicos para fallos al modificar los datos del paciente.
CU04	¿El diagrama detalla cómo se calculan y actualizan las calorías en el plan alimenticio del paciente?	X		El flujo de cálculo de calorías está bien descrito, pero podría beneficiarse de más detalles en los cálculos intermedios.
CU06	¿El diagrama incluye la interacción entre el usuario y el calendario para programar citas?	X		Se detallan claramente las interacciones entre el sistema y el calendario del usuario.
CU06	¿Se representa una confirmación al usuario después de agendar una cita correctamente?	X		El diagrama muestra una confirmación visual y auditiva al usuario después de la acción.
CU08	¿El diagrama muestra cómo se obtienen todas las citas para un mes y se visualizan al usuario?	X		El proceso está claramente representado, pero se pueden agregar optimizaciones para consultas más rápidas.

CU09	¿Se detalla el flujo para agregar un nuevo registro de diagnóstico?	X		El flujo cubre todos los pasos necesarios para la adición de un nuevo diagnóstico, incluyendo validaciones.
CU11	¿El diagrama incluye el manejo de errores al cargar una imagen incompatible?		X	No se observa un manejo adecuado para errores derivados de imágenes incompatibles.
CU12	¿El diagrama de secuencia representa cómo se filtra el historial por rango de fechas?	X		El diagrama muestra cómo se realiza el filtrado por fecha, pero faltan detalles sobre la optimización de esta búsqueda.
CU14	¿El diagrama muestra las interacciones necesarias para exportar un archivo PDF?		X	El proceso de exportación a PDF no está bien detallado en el diagrama, lo que deja lugar a ambigüedades.

Reporte de Junta TSP – Formato MTG

Moderador:	Elizabeth Murrieta Sangabriel	**Lugar:**	Aula 105 de la FEI	**Fecha:**	29/11/2024
Fecha de la junta:	29/11/2024	**Hora Inicio:**	14:30	**Hora Fin:**	16:05

Tema/Propósito:	Revisión de los prototipos, diagramas de secuencia y diagramas de robustez del Sistema HealthDivineSystema para la EE de Desarrollo de Software.

Asistentes

Nombre	Rol
Gustavo Yussif Mendoza Severo	Responsable de la decisión
Gustavo Yussif Mendoza Severo	Líder de revisión
Alesis de Jesús Torres Osorio	Documentador
Equipo EscribeUnNombre	Revisores

Agenda

Tiempos (min.)			Temas	Líder de Presentacion
Plan	**Inicio**	**Fin**		
10	14:30	14:40	Planificación de la revisión	Gustavo Yussif Mendoza Severo
10	14:40	14:50	Presentación de los procedimientos para la revisión	Gustavo Yussif Mendoza Severo
5	14:50	14:55	Preparación de la revisión	Alesis de Jesús Torres Osorio
10	14:55	15:05	Presentación de los recursos del software a revisar	Alesis de Jesús Torres Osorio
5	15:05	15:10	Junta de la revisión: Apertura	Gustavo Yussif Mendoza Severo
30	15:10	15:40	Examinación de los productos	Alesis de Jesús Torres Osorio
20	15:40	16:00	Señalización de defectos	Alesis de Jesús Torres Osorio
5	16:00	16:05	Junta de la revisión: Conclusión	Gustavo Yussif Mendoza Severo

Decisiones, Acciones e Informes Clave

Qué	Quién	Cuando
El documento establece una base sólida para el proyecto, abarcando la mayoría de los aspectos esenciales que se esperan en un documento de visión y alcance. No obstante, hay áreas clave que requieren mejora. La ausencia de casos de uso o escenarios de interacción es una omisión significativa, ya que estos son fundamentales para comprender cómo los usuarios interactuarán con el sistema. Además, aunque se han identificado los principales riesgos del proyecto, la falta de un plan de mitigación para abordarlos representa una debilidad importante que debe ser corregida.	Gustavo Yussif Mendoza Severo	Viernes 29/11/2024

Reporte Final de Revisión Técnica Formal

1. Producto revisado

1.1. Nombre y Versión del Producto revisado

Producto	Versión
Documento de Visión y Alcance del Sistema HealthDivineSystem para las EE de Desarrollo de Software	1.0

1.2. Participantes de la revisión

Miembro	Rol
Daniel Mongeote Tlachy	Documentador, Revisor
Gustavo Yussif Mendoza Severo	Responsable de decisión, Revisor
Alesis de Jesús Torres Osorio	Líder de revisión, Revisor

1.3. Técnica utilizada

Revisión Técnica Formal.

2. Objetivos de la RTF

- Evaluar la consistencia y completitud de los casos de uso implementados.
- Identificar oportunidades de mejora en la interfaz, validaciones y representación gráfica.

- Asegurar que las funcionalidades cumplan con los estándares definidos en el documento de requisitos.

3. Problemas detectados

3.1. Prototipos

<CU-01 Dar de alta un paciente>

- Problema: No se especifican mensajes claros para entradas inválidas. Esto podría confundir a los usuarios.
- Sugerencia: Incluir mensajes de error detallados para cada posible caso de entrada inválida, asegurando que el usuario pueda corregir los errores fácilmente.

<CU-04 Crear plan alimenticio>

- Problema: Falta una opción de vista previa del plan alimenticio antes de confirmarlo.
- Sugerencia: Implementar una funcionalidad de previsualización para que el nutricionista pueda revisar los datos antes de guardarlos definitivamente.

<CU-06 Programar citas>

- Problema: No se incluye un mecanismo para notificar conflictos de horarios al usuario.
- Sugerencia: Añadir validaciones en tiempo real que detecten solapamientos de citas y muestren advertencias claras al usuario.

<CU-11 Añadir imágenes>

- Problema: No hay una barra de progreso para indicar el estado de la carga de imágenes.
- Sugerencia: Incorporar una barra de progreso que indique visualmente la etapa de carga para mejorar la experiencia del usuario.

3.2. Diagramas de Robustez

<CU-01 Dar de alta un paciente>

- Problema: No se muestra cómo el sistema maneja errores al registrar datos duplicados.
- Sugerencia: Agregar una validación explícita en el diagrama que detalle el manejo de casos donde el paciente ya está registrado.

<CU-04 Crear plan alimenticio>

- Problema: La representación del flujo de validación es insuficiente para reflejar escenarios de error.
- Sugerencia: Incluir pasos detallados sobre cómo el sistema maneja entradas inválidas y errores de cálculo de valores nutricionales.

<CU-14 Exportar a PDF>

- Problema: No se detalla cómo se realiza la conversión de datos a formato PDF.
- Sugerencia: Diagramar el proceso completo de exportación, incluyendo interacciones con bibliotecas externas o servicios específicos de generación de PDF.

3.3. Diagramas de Secuencia

<CU-06 Programar citas>

- Problema: Falta representar cómo el sistema alerta al usuario sobre horarios inválidos o citas duplicadas.
- Sugerencia: Añadir interacciones que incluyan mensajes de error para horarios solapados, validaciones, y una confirmación de la acción corregida.

<CU-09 Crear diagnóstico>

- Problema: No se consideran errores al subir imágenes incompatibles.
- Sugerencia: Incluir en el diagrama un flujo alternativo que valide formatos y tamaños de imágenes antes de subirlas.

<CU-14 Exportar a PDF>

- Problema: El proceso para exportar datos no está completamente especificado.
- Sugerencia: Representar la interacción del sistema con los datos, cómo se formatean y se envían al módulo de exportación a PDF.

4. Evaluación

4.1. Estado Actual del Producto

El sistema Health Divine Sys presenta un avance significativo en su desarrollo. Los prototipos, diagramas de robustez y de secuencia abordan las funcionalidades principales del sistema, alineándose con los objetivos del proyecto. La estructura base de los entregables incluye la gestión de pacientes, creación de planes alimenticios personalizados y manejo de citas.

No obstante, se han identificado áreas clave de mejora, entre las que destacan:

- Falta de mensajes de error claros en prototipos para guiar al usuario en escenarios de entrada inválida.
- Incompleta representación gráfica de procesos críticos en diagramas, como validaciones y gestión de errores.
- Ausencia de flujos detallados para funcionalidades específicas como la exportación de planes a PDF.

4.2. Acciones para Tomar

4.2.1 Prototipos:

- Implementar mensajes de error claros en todos los escenarios de entrada inválida.
- Añadir funcionalidades de previsualización en CU-04 (Planes alimenticios) y CU-14 (Exportar a PDF).

4.2.2 Diagramas de Robustez:

- Incluir flujos detallados de validaciones y manejo de errores, especialmente en CU-01, CU-04 y CU-14.
- Representar de forma explícita el proceso de generación y validación de exportación de datos.

4.2.3. Diagramas de Secuencia:

- Completar los flujos alternativos relacionados con errores en subida de imágenes y conflictos de horarios en CU-06 y CU-11.
- Representar las interacciones del sistema con los datos y módulos externos para procesos críticos.

5. Próxima Revisión del Producto

Se recomienda realizar la siguiente revisión en un plazo de 4 semanas, una vez que las acciones identificadas hayan sido implementadas. Durante esta revisión se evaluará:

- La efectividad de los cambios en los prototipos y diagramas.
- La alineación de los entregables con los objetivos del proyecto.
- La cobertura de validaciones y flujos alternativos en todos los casos de uso relevantes.

Walkthrough

CheckList

Nombre	Lista de verificación	**Fecha**	07/12/2024
Equipo	EscribeUnNombre	**Instructor/Gerente**	Elizabeth Murrieta Sangabriel
Parte/Nivel	Construcción	**Ciclo**	-
Hora inicio	15:02	**Hora fin**	16:48

Propósito	Lista de verificación para el codigo fuente de los siguientes casos de uso: CU01 – Dar de alta paciente, CU03 – Modificar padecimientos del paciente, CU04 – Crear plan alimenticio del paciente, CU06 – Programar cita, CU08 – Mostrar calendario de citas mensuales, CU09 – Crear nuevo registro de diagnóstico, CU11 – Añadir imagen de proceso del paciente, CU12 – Vusyualizar historial del paciente y CU14 – Exportar plan alimenticio a PDF.
General	Revisar el código fuente completo para cada categoría de lista, no trate de revisar más de una categoría a la vez. A medida que vaya completando cada etapa de revisión, marque el elemento en el cuadro de laderecha. Completar la lista de verificación para un archivo antes de examinar los próximos.

Gustavo Yussif Mendoza Severo			
Caso de Uso	**Pregunta**	**Sí**	**No**
CU01 – Dar de alta paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		

	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		X
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU03 – Modificar padecimientos del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		

	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	

	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU04 – Crear plan alimenticio del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X

	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU06 – Programar cita	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	

	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU08 – Mostrar calendario de citas mensuales	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	

	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.		X
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU09 – Crear nuevo registro de diagnóstico	Calidad del Código		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	

	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.		X
	Las consultas a bases de datos o servicios externos están optimizadas.		X
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		

	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU11 – Añadir imagen de proceso del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.		X
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		

	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.		X
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU12 – Visualizar historial del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.		X
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X

	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.		X
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU14 – Exportar plan alimenticio a PDF	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X

	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.	X	
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	

Daniel Mongeote Tlachy			
Caso de Uso	**Pregunta**	**Sí**	**No**
CU01 – Dar de alta paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X

	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		X
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU03 – Modificar padecimientos del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	

	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU04 – Crear plan alimenticio del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	

	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU06 – Programar cita	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	

	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		

	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU08 – Mostrar calendario de citas mensuales	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		

	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.		X
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU09 – Crear nuevo registro de diagnóstico	Calidad del Código		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X

	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.		X
	Las consultas a bases de datos o servicios externos están optimizadas.		X
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU11 – Añadir imagen de proceso del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X

	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.		X
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.		X
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	

CU12 – Visualizar historial del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.		X
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	

	El código ha sido revisado por un compañero antes de ser integrado.		X
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU14 – Exportar plan alimenticio a PDF	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.	X	
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	

	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	

Alesis de Jesús Torres Osorio			
Caso de Uso	**Pregunta**	**Sí**	**No**
CU01 – Dar de alta paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	

	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		X
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU03 – Modificar padecimientos del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	

	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		

	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU04 – Crear plan alimenticio del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		

	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU06 – Programar cita	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X

	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU08 – Mostrar calendario de citas mensuales	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X

	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.		X
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	

CU09 – Crear nuevo registro de diagnóstico	Calidad del Código		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.		X
	Las consultas a bases de datos o servicios externos están optimizadas.		X
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	

	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU11 – Añadir imagen de proceso del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.		X
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	

	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.		X
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU12 – Visualizar historial del paciente	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	
	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.		X
	El flujo lógico del programa es claro y funciona según lo esperado.		X
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X

	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.		X
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		
	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	
CU14 – Exportar plan alimenticio a PDF	**Calidad del Código**		
	El código sigue las convenciones y estándares de codificación establecidos.	X	
	Los nombres de variables, métodos y clases son claros y representativos de su propósito.	X	
	Se eliminaron todos los comentarios de depuración y código innecesario o redundante.		X
	Se utilizaron buenas prácticas como la separación de responsabilidades y principios SOLID.	X	
	Estructura del Código		
	El código está modularizado y es reutilizable.	X	

	Se minimizó la duplicación de código mediante la implementación de funciones o clases.		X
	Cada archivo de código tiene una única responsabilidad y tamaño razonable.	X	
	Las dependencias entre módulos están correctamente gestionadas y minimizadas.		X
	Funcionalidad		
	Todas las funciones cumplen con los requerimientos especificados.	X	
	El código maneja todas las entradas válidas y no válidas correctamente.	X	
	Las excepciones y errores se manejan adecuadamente.	X	
	El flujo lógico del programa es claro y funciona según lo esperado.	X	
	Pruebas Unitarias		
	Todas las funciones críticas están cubiertas por pruebas unitarias.		X
	Las pruebas unitarias tienen un porcentaje de éxito alto (idealmente > 90%).		X
	Se probaron casos límite, de borde y de error.		X
	Las pruebas están documentadas y son fáciles de reproducir.		X
	Eficiencia		
	El código evita operaciones costosas innecesarias.		X
	Los algoritmos implementados son eficientes en términos de tiempo y memoria.	X	
	Las consultas a bases de datos o servicios externos están optimizadas.	X	
	Seguridad		
	Las entradas del usuario están validadas y sanitizadas.	X	
	No se utilizan datos sensibles directamente en el código.	X	
	Las claves API, contraseñas o datos sensibles están protegidos y no en texto plano.	X	
	El código cumple con las prácticas de seguridad (ej.: OWASP).	X	
	Comentarios y Documentación		
	Cada método o función tiene comentarios claros que describen su propósito.		X
	Las partes complejas del código están documentadas para facilitar su comprensión.		X
	Los comentarios están actualizados y no generan confusión.		X
	El código incluye documentación técnica sobre la estructura general y el propósito.		X
	Gestión de Versiones		
	Todos los cambios están registrados en el sistema de control de versiones.	X	
	Los commits son claros, con mensajes descriptivos y específicos.	X	
	El código ha sido revisado por un compañero antes de ser integrado.	X	
	No hay conflictos de código pendientes.	X	
	Compatibilidad		
	El código es compatible con todas las versiones de las dependencias definidas.	X	
	El código se probó en todos los entornos objetivo (desarrollo, pruebas, producción).		X
	Las configuraciones dependientes del entorno están correctamente separadas del código fuente.	X	
	Rendimiento		

	El tiempo de ejecución del código es aceptable y cumple con los requisitos.	X	
	Se realizaron pruebas de carga o estrés para identificar cuellos de botella.		X
	El código está preparado para escalar según sea necesario.	X	

Reporte de Recorrido

Producto:	Código Fuente	**Fecha:**			07/12/2024
Autor:	HealthDivineSystem	**Lugar:**	Xalapa. Veracruz		
Fecha de Recorrido:	07/12/2024	**Hora Inicio:**	19:55	**Hora Fin:**	21:25

Tema/Propósito	Recorrido de Construcción (Código fuente)

Asistentes

Nombre	Rol
HealthDivineSystem	Autor
Gustavo Yussif Mendoza Severo	Revisor
Daniel Mongeote Tlachy	Revisor
Alesis de Jesús Torres Osorio	Revisor

Agenda

Tiempos (min.)			Temas
Plan	**Inicio**	**Fin**	
20	19:55	20:15	Revisión Preliminar
10	20:15	20:25	Presentación del Producto
25	20:25	20:50	Recorrido del Producto
25	20:50	21:15	Señalización de Defectos
10	21:15	21:25	Conclusión
90 min			

Decisiones, Acciones e Información Clave

Qué	Quién	Cuándo
Refactorizar el código para que cumpla con las convenciones y estándares establecidos.	EquipoEscribeUnNombre	Antes del cierre de la iteración actual.

Renombrar variables, métodos y clases para que sean claros y representativos.	EquipoEscribeUnNombre	Durante la próxima iteración.
Eliminar comentarios de depuración y código redundante presente en el proyecto.	EquipoEscribeUnNombre	Durante la fase de depuración actual.
Aplicar principios SOLID y prácticas de separación de responsabilidades.	EquipoEscribeUnNombre	En la fase de diseño refactorizado.
Modularizar el código para mejorar la reutilización y evitar duplicación.	EquipoEscribeUnNombre	Durante la siguiente fase de implementación.
Revisar dependencias entre módulos y reducir el acoplamiento.	EquipoEscribeUnNombre	En la revisión técnica próxima.
Escribir pruebas unitarias para funciones críticas no cubiertas.	EquipoEscribeUnNombre	Durante el próximo ciclo de pruebas.
Optimizar consultas a bases de datos y servicios externos.	EquipoEscribeUnNombre	Antes del despliegue en producción.
Validar y sanear las entradas de usuario para mejorar la seguridad.	EquipoEscribeUnNombre	En la iteración actual de desarrollo.
Actualizar y clarificar los comentarios del código.	EquipoEscribeUnNombre	Durante la revisión del código.
Resolver conflictos de código pendientes en el repositorio.	EquipoEscribeUnNombre	Antes del próximo commit.

6.2. Reporte de anomalías

En esta sub sección se presenta un reporte de aquellos errores, defectos o deficiencias que puedan surgir dentro de las etapas de la V&V.

Nombre	Reporte de anomalías	**Fecha**	29/11/2024
Equipo	EscribeUnNombre	**Instructor/Gerente**	Elizabeth Murrieta Sangabriel
Parte/Nivel	-	**Ciclo**	-
Hora inicio	18:32	**Hora fin**	19:54

Propósito	Identificacion de los posibles errores cometidos durante la ejecucion de VyV
General	En el proceso de **Verificación y Validación (VyV)** de software, suelen surgir errores, defectos o deficiencias debido a diversos factores inherentes al desarrollo y control de calidad. A continuación, se detallan los problemas más comunes que pueden aparecer durante las etapas de VyV

En Verificación:

1. **Especificaciones incompletas o ambiguas**: Dificultan verificar el cumplimiento del software.
2. **Cobertura insuficiente**: No se prueban todas las áreas críticas.
3. **Errores en configuración del entorno**: Resultados incorrectos por configuraciones defectuosas.
4. **Pruebas manuales ineficaces**: Falta de documentación o experiencia adecuada.
5. **Dependencia excesiva en automatización**: Omite problemas como usabilidad y accesibilidad.

En Validación:

1. **Falta de interacción con usuarios reales**: Pruebas alejadas de escenarios reales.
2. **Pruebas funcionales incompletas**: Algunas funcionalidades no se validan correctamente.
3. **Descuidos en pruebas no funcionales**: Omisión de rendimiento, seguridad o compatibilidad.
4. **Errores en pruebas de aceptación**: Malentendidos con usuarios finales.
5. **Inconsistencias entre documentación y software**: Confusión para los usuarios.

En Procesos y Herramientas:

1. **Falta de métricas claras**: Dificultad para evaluar calidad y avance.
2. **Comunicación deficiente entre equipos**: Genera malentendidos y redundancias.
3. **Herramientas de prueba inadecuadas**: Limitan la detección de defectos.
4. **Restricciones de tiempo y presupuesto**: Reduce profundidad de pruebas.

En Manejo de Defectos:

1. **Priorización incorrecta**: Defectos críticos pasan desapercibidos.
2. **Retrasos en corrección**: Ineficiencias en el seguimiento de defectos.
3. **Reintroducción de errores**: Cambios no validados generan nuevas fallas.

Factores Humanos:

1. **Falta de experiencia o capacitación**: Omisión de defectos importantes.

2. **Presión por plazos**: Pruebas apresuradas y superficiales.
3. **Fatiga o desmotivación**: Reduce la efectividad del equipo.

Revisión Técnica Formal (RTF):

1. **Preparación inadecuada**: Participantes no revisan previamente los documentos a analizar.
2. **Falta de roles definidos**: No se asignan moderador, autor y revisores claramente.
3. **Documentación incompleta**: Artefactos o especificaciones faltantes dificultan la revisión.
4. **Foco excesivo en detalles menores**: Se ignoran problemas críticos por centrarse en cuestiones triviales.
5. **Ausencia de criterios claros**: No se establecen estándares objetivos para evaluar la calidad.
6. **Falta de seguimiento**: Los problemas detectados no se documentan ni se corrigen adecuadamente.

Inspección:

1. **Selección de artefactos irrelevantes**: Se inspeccionan elementos que no son prioritarios.
2. **Equipo insuficientemente capacitado**: Los inspectores carecen del conocimiento técnico necesario.
3. **Tiempo insuficiente para análisis**: Se apresura la inspección, pasando por alto defectos críticos.
4. **Falta de herramientas de apoyo**: No se utilizan herramientas para detectar defectos automatizables.
5. **Comunicación deficiente**: Los participantes no comparten observaciones o conclusiones de manera efectiva.
6. **Documentación de defectos vaga o incompleta**: No se describen claramente los problemas detectados, dificultando su resolución.

Recorrido:

1. **Falta de objetivos claros**: No se define qué se espera lograr en el recorrido.
2. **Participantes no preparados**: Asisten sin haber revisado previamente los documentos o código.
3. **Discusiones desorganizadas**: Se desvían hacia temas no relacionados con los objetivos del recorrido.
4. **Falta de moderador**: No se asigna un facilitador para dirigir la sesión de manera efectiva.
5. **Problemas sin registrar**: Defectos identificados no se documentan formalmente para su seguimiento.
6. **Enfoque limitado a una sola perspectiva**: No se consideran aspectos de usuarios, rendimiento o mantenimiento.

6.3. Reporte Final de Verificación y Validación

1. Introducción

El presente documento describe los resultados obtenidos durante la evaluación del sistema en el contexto del proceso de verificación y validación. El análisis realizado busca identificar áreas de mejora críticas y proponer acciones correctivas para garantizar el cumplimiento de los requisitos establecidos y una experiencia de usuario robusta.

2. Resultados Generales

2.1 Avances Significativos

Se han logrado avances importantes en la implementación de las siguientes funcionalidades principales:

- Gestión de pacientes: Incluye alta, edición y eliminación de registros.
- Planes alimenticios personalizados: Diseño automático de planes adaptados a las necesidades del usuario.
- Programación de citas: Sistema para gestionar horarios y disponibilidad.

2.2 Áreas Críticas de Mejora

A pesar de los avances, se detectaron áreas que requieren atención inmediata:

- Claridad de mensajes al usuario: Los mensajes de error y retroalimentación necesitan ser más descriptivos.
- Validaciones de datos: Algunas validaciones son insuficientes o inexistentes.
- Representación gráfica de procesos: Los diagramas carecen de detalles en flujos clave.

Los artefactos evaluados muestran una alineación parcial con los requisitos establecidos, pero requieren ajustes para cumplir con los estándares de calidad.

3. Evaluación de Artefactos

3.1 Prototipos

Se identificaron las siguientes deficiencias:

- **CU-01:** Falta de mensajes de error para entradas incorrectas. Se recomienda que los errores se muestren de manera inmediata al usuario y no solo al intentar guardar, para mejorar la experiencia.
- **CU-04:** Ausencia de una opción de previsualización en la creación de planes alimenticios. Esto podría generar errores al guardar.
- **CU-06:** No se notifican conflictos de horarios al programar citas, lo que podría llevar a solapamientos inadvertidos.

3.2 Diagramas de Robustez

Los diagramas presentan flujos incompletos en los siguientes casos:

- **CU-01:** Validación de errores al dar de alta pacientes. Aunque se consideran las interacciones con la base de datos, se podría mejorar la especificación de interacciones detalladas entre actores y sistema.
- **CU-04:** Cálculos automáticos de planes alimenticios. Las reglas de negocio son claras, pero se beneficiarían de ejemplos de entrada y salida.
- **CU-14:** Generación de PDFs. No se representa el proceso completo para convertir y exportar los datos.

3.3 Diagramas de Secuencia

En los diagramas se omiten flujos alternativos y detalles en procesos críticos:

- **CU-11:** Manejo de errores en la carga de imágenes. No se abordan los errores derivados de formatos incompatibles o fallas durante la carga.
- **CU-06:** Resolución de conflictos de horarios. La falta de un actor encargado de confirmar los detalles podría generar ambigüedades en la implementación.
- **CU-14:** Exportación de datos a PDF. El proceso de conversión no está representado en su totalidad, lo que dificulta la comprensión del flujo completo.

4. Acciones Correctivas Propuestas

4.1 Validaciones Específicas

Se recomienda implementar validaciones que aborden los problemas detectados en las siguientes áreas:

- Mostrar mensajes de error detallados para entradas inválidas de forma inmediata.
- Notificar automáticamente a los usuarios en casos de conflictos de horarios al programar citas.
- Incorporar una opción de previsualización en la creación de planes alimenticios.

4.2 Flujos Alternativos

Es necesario incluir flujos alternativos en los diagramas para manejar errores y contingencias:

- Diseñar soluciones para errores en la carga y validación de imágenes antes de ser procesadas.
- Incorporar mecanismos para gestionar conflictos detectados en la programación de citas.
- Detallar los pasos en la conversión y exportación a formato PDF, incluyendo mensajes de confirmación al usuario.

4.3 Mejoras en Documentación

Actualizar la documentación técnica para incluir:

- Diagramas completos y detallados de robustez y secuencia que representen todos los flujos y escenarios posibles.
- Guías claras y ejemplos prácticos para la implementación y verificación de cada caso de uso, considerando también casos de borde y errores.

6. Conclusión

Los resultados de esta evaluación evidencian un progreso significativo en el desarrollo del sistema, destacando logros en funcionalidades clave como la gestión de pacientes y la programación de citas. No obstante, persisten áreas críticas que requieren atención inmediata, especialmente en aspectos relacionados con la validación de datos, la claridad de mensajes al usuario y la representación gráfica de procesos.

Las acciones correctivas propuestas buscan no solo garantizar el cumplimiento de los requisitos establecidos, sino también optimizar la experiencia del usuario mediante una mejora integral de la documentación técnica y la implementación de flujos alternativos. Estas mejoras contribuirán a reforzar la robustez del sistema y asegurar su escalabilidad.

Con una revisión programada en cuatro semanas, se espera que los ajustes implementados logren avances sustanciales en la calidad del sistema, consolidando un producto alineado con los estándares de calidad y las expectativas de los usuarios.