PROYECTO MEDICAL AID

Product Backlog

Integrantes

Raimundo Estévez
Soledad Inostroza
Marcel Brard

Fecha	versión	Elaborado por
20-09-2024	1.0	Raimundo Estévez Soledad Inostroza Marcel Brard

Tabla de Contenido

Product Backlog

ID	Historia de Usuario	Prioridad	Puntos de Historia	Estado
HU-0001	**Nombre de la Historia:** Creación de la base de datos **Como:** Desarrollador del sistema **Quiero:** Crear una base de datos que soporte las entidades necesarias **Para:** Garantizar que la información se almacene, actualice y consulte de manera eficiente y segura. **Criterios de Aceptación:** 1. La base de datos debe estar estructurada para soportar las entidades principales del sistema, incluyendo relaciones adecuadas (por ejemplo, usuarios, diagnósticos, síntomas). 2. Las consultas a la base de datos deben ser optimizadas para garantizar tiempos de respuesta rápidos. 3. La base de datos debe contar con índices, claves primarias y foráneas para mantener la integridad de los datos. **Definición de Terminado:**	**Alta**	**6**	**Completado**

	1. La base de datos está implementada y soporta todas las entidades establecidas. 2. Las relaciones, claves primarias y foráneas están configuradas correctamente. 3. Las consultas han sido optimizadas y cumplen con los tiempos de respuesta requeridos. 4. Se han realizado pruebas de integridad y rendimiento en la base de datos. 5. La base de datos almacena y recupera la información correctamente. **Tareas a realizar:** 1. Identificar las entidades, atributos y relaciones necesarias para el sistema. 2. Diseño del modelo entidad relación (ER): Crear un diagrama ER que represente las entidades y sus relaciones. 3. Definición de las claves primarias y foráneas: Asegurar que cada entidad tenga su clave primaria, y definir las relaciones entre tablas con claves foráneas. 4. Crear base de datos con motor PostgreSQL 5. Implementar las tablas y relaciones necesarias en PostgreSQL 6. Realizar pruebas para asegurar que las entidades y relaciones funcionen correctamente.			

	7. Documentar los resultados y ajustar la base de datos según los hallazgos.			
HU-0002	**Nombre de la Historia:** Crear cuenta de usuario **Como:** Un nuevo usuario registrado **Quiero:** Crear una cuenta en el sistema **Para :** Poder acceder a las funcionalidades del sistema **Criterios de Aceptación:** 1. Un usuario puede crear una cuenta en el sistema. 2. No se generan errores, como creación de usuarios duplicados ni explotación de vulnerabilidades. 3. Las Notificaciones de error deben funcionar correctamente. **Definición de Terminado:** 1. La funcionalidad de registro de usuario está completamente implementada. 2. Las validaciones del frontend y backend funcionan correctamente. 3. La contraseña se almacena de forma segura utilizando BCRYPT. 4. Se han realizado pruebas unitarias, de integración y de seguridad para asegurar que no existan vulnerabilidades 5. El correo de confirmación es enviado correctamente. **Tareas a realizar:**	**Media**	**5**	**Completado**

	1. Implementar un formulario donde el usuario ingrese los datos necesarios para crear la cuenta. 2. Implementar validaciones en el frontend. 3. Crear Endpoint en el backend para crear usuarios. 4. Implementar validaciones en el backend 5. Implementar hasheo de la contraseña con BCRYPT 6. Implementar manejo de errores en el Endpoint 7. Implementar envío de respuesta con el resultado de la consulta en el Endpoint 8. Validar el correcto funcionamiento del endpoint 9. Configurar la solicitud HTTP par la creación del usuario desde el frontend 10. Manejar la respuesta en el frontend (200: Redirigir a página de inicio. Error: Crear alerta notificando el tipo de error) 11. Implementar medidas contra ataques de fuerza bruta, ej. El usuario puede intentar el ingreso de contraseña 3 veces como máximo. 12. Implementar medidas ante SQL Injection en los campos (Baja Prioridad) 13. Aplicar HTTPS (Baja Prioridad) 14. Enviar correo de confirmación al usuario (Opcional)			
HU-0003	**Nombre de la Historia:** Ver cuenta de usuario	**Media**	**6**	**Por Hacer**

	Como: Un usuario registrado **Quiero:** Poder ver mi cuenta en el sistema **Para qué:** Revisar o verificar mi información personal **Criterios de Aceptación:** 1. El sistema puede consultar la información del usuario activo. 2. El sistema puede consultar la información de todos los usuarios. 3. El sistema muestra correctamente la información del usuario activo. 4. El sistema muestra correctamente la información de todos los usuarios registrados 5. La comunicación entre el frontend y backend maneja correctamente los errores y los despliega en la vista. 6. Método sólo puede ser utilizado por usuarios registrados. **Definición de Terminado:** 1. El perfil de usuario se muestra y se puede consultar la información correctamente. 2. La información de todas los usuarios es accesible para los administradores. 3. Las pruebas de usabilidad confirman que la funcionalidad de consulta de usuario es correcta. 4. Se ha validado que la información del usuario activo se muestra de forma precisa y sin errores. 5. Se valida que método solo puede ser usado por los usuarios registrados.			

	6. La comunicación entre el frontend y backend maneja correctamente los errores y los despliega en la vista. **Tareas a realizar:** 1. Crear vista de perfil de usuario. 2. Crear lista de administración de usuarios. 3. Crear Endpoint GetUserByID: Obtención de datos de usuario desde la base de datos. 4. Crear Endpoint GetAllUsers: Obtención de datos de todos los usuarios desde la base de datos. 5. Implementar método de autenticación de usuario en los Endpoints. 6. Implementar manejo de errores en las Endpoints. 7. Implementar envío de respuesta con el resultado de la consulta en los Endpoints. 8. Validar el correcto funcionamiento de los Endpoints. 9. Implementar consulta HTTP al Endpoint GetUserByID en la vista perfil de usuario. 10. Implementar consulta HTTP al Endpoint GetAllUsers en la vista administración de usuarios. 11. Mostrar la información del usuario obtenida del Endpoint GetuserByID dentro de la vista perfil de usuario. 12. Mostrar la información de los usuarios obtenida del Endpoint GetAllUsers dentro de la vista de			

	administración de usuario en forma de tabla. 13. Gestionar errores de carga de datos en vista perfil de usuario. 14. Gestionar errores de carga de datos en vista de la administración de usuarios. 15. Aplicar HTTPS (Baja Prioridad).			
HU-0004	**Nombre de la Historia:** Actualizar cuenta de usuario **Como:** Un usuario registrado **Quiero:** Editar mi cuenta **Para qué:** Actualizar mi información personal **Criterios de Aceptación:** 1. Un usuario puede generar su información en el sistema. 2. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. 3. Método solo puede ser utilizado por usuarios registrados. **Definición de Terminado:** 1. La funcionalidad de edición de usuario está implementada. 2. Las validaciones de datos y seguridad están en funcionamiento. 3. La contraseña se actualiza correctamente usando BCRYPT. 4. Las pruebas de usabilidad y funcionalidad son exitosas.	Media		No Iniciado

	5. Se valida que método implementado solo puede ser usado por los usuarios registrados. **Tareas a realizar:** 1. Crear vista con formulario para editar usuario 2. Implementar validaciones en el frontend 3. Crear endpoint en el backend para editar usuario 4. Implementar método de autenticación de usuario en el Endpoint. 5. Implementar validaciones en el backend 6. Implementar hasheo de la contraseña con BCRYPT 7. Implementar manejo de errores en los Endpoints. 8. Generar respuesta con el resultado de la solicitud 9. Validar el correcto funcionamiento del endpoint 10. Configurar la solicitud HTTP para la edición del usuario desde el frontned 11. Manejar la respuesta en el frontend (200: Redirigir a vista perfil de usuario mostrando los datos actualizados, Error: Crear alerta notificando el tipo de error) 12. Implementar medidas ante SQL Injection en los campos (Baja Prioridad) 13. Aplicar HTTPS (Baja Prioridad)			

HU-0005	**Nombre de la Historia:** Borrar cuenta de usuario **Como:** Un administrador **Quiero:** Poder eliminar una cuenta de usuario **Para que:** Gestionar usuarios que deben ser eliminados **Criterios de Aceptación:** 1. Un administrador puede eliminar usuarios en el sistema. 2. La eliminación es lógica. 3. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. 4. Método sólo puede ser utilizado por usuarios registrados. 5. Método sólo puede ser utilizado por administradores registrados. **Definición de terminado:** 1. La eliminación lógica de usuarios está implementada. 2. Solo los administradores autorizados pueden acceder a la función. 3. Se han implementado medidas de seguridad para evitar eliminaciones accidentales. 4. Las pruebas han validado que el usuario se elimina correctamente y no afecta otros registros. **Tareas a realizar:** 1. Implementar botón de eliminar usuario en la vista de administración de usuarios.	Media		No Iniciado

	2. Crear Endpoint para la eliminación de usuario en el backend. (Esta debe aplicar una eliminación lógica, no física) 3. Implementar método de autenticación de usuario en el Endpoint. 4. Implementar manejo de errores en el Endpoint. 5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Validar el correcto funcionamiento del Endpoint. 7. Implementar alerta para confirmar la eliminación de un usuario en el botón de eliminar usuario. (Así se evitan eliminaciones accidentales) 8. Implementar consulta HTTP al Endpoint de eliminación de usuario en botón de eliminar usuario. 9. Manejar la respuesta en el frontend (200: Actualizar página para mostrar la lista de usuarios actualizado, Error: Crear alerta notificando el tipo de error)			
HU-0006	**Nombre de la Historia:** Búsqueda de usuarios registrados **Como:** Un administrador **Quiero:** Buscar usuarios por identificación o nombre **Para qué:** Gestionar rápidamente los registros de usuarios en el sistema **Criterios de Aceptación:** 1. Módulo de administración de usuarios creado. 2. Sólo los usuarios con rol de administrador pueden acceder.	**Baja**		**No iniciado**

	3. Se puede realizar búsqueda de los usuarios por id o nombre. 4. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. 5. Método sólo puede ser utilizado por administradores registrados. **Definición de Terminado:** 1. La funcionalidad de búsqueda de usuarios por ID o nombre está implementada. 2. La interfaz de administración permite a los administradores realizar búsquedas correctamente. 3. Las pruebas de usabilidad confirman que la búsqueda es eficiente y sin errores. 4. Los resultados de la búsqueda son precisos y se muestran correctamente en el frontend. 5. Se valida que sólo los usuarios con rol de administrador pueden acceder al módulo. **Tareas a realizar:** 1. Implementar el formulario de búsqueda de usuarios en la vista administración de usuario 2. Crear Endpoint con la lógica de búsqueda en el backend, para recibir las solicitudes de búsqueda (por ID o nombre) 3. Implementar método de autenticación de usuario en el Endpoint. 4. Implementar manejo de errores en el Endpoint.			

	5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Implementar consulta HTTP al en la vista administración de usuarios. 7. Mostrar la información del usuario obtenida del Endpoint dentro de la vista administración de usuarios. 8. Gestionar errores y excepciones, como usuarios no encontrados o problemas en la base de datos.			
HU-0007	**Nombre de la Historia:** Autenticar cuenta de usuario **Como:** Un usuario registrado **Quiero: Autenticarme en el sistema con mis credenciales** **Para qué:** Acceder de forma segura a mi información y funcionalidades **Criterios de Aceptación:** 1. El sistema posee vista de login 2. El envío de la contraseña está cifrado. 3. Creación de token 4. Las Notificaciones de error funcionan correctamente. **Definición de Terminado:** 1. La vista de login y autenticación está implementada. 2. La contraseña se transmite y valida de forma segura. 3. El sistema maneja errores y autenticación de tokens correctamente.	Alta	5	Por Hacer

	4. Las pruebas de seguridad confirman que no existen vulnerabilidades en la autenticación. **Tareas a realizar:** 1. Diseñar vista de login con formulario de inicio de sesión, incluir campos para introducir nombre de usuario y contraseña. 2. Crear Endpoint con la lógica de autenticación, para recibir las solicitudes de búsqueda (por ID o nombre) 3. Implementar uso de Bcrypt para el hasheo de las contraseñas y compararla con la guardada en base de datos para validar el login. 4. Implementar manejo de errores en el Endpoint. 5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Validar el correcto funcionamiento del Endpoint. 7. Implementar consulta HTTP en la vista de login. Enviar las credenciales al backend cuando el usuario pulse el botón de inicio de sesión. 8. Mostrar mensaje de error en caso de credenciales incorrectas o problemas con autenticación. 9. Implementar gestión de tokens. 10. Implementar la funcionalidad de cierre de sesión, eliminando el token almacenado.			

HU-0008	**Nombre de la Historia:** Recuperación de contraseña **Como:** Un usuario que olvidó su contraseña **Quiero:** Poder recuperar la contraseña mediante un enlace enviado a mi correo **Para qué:** Volver a acceder a mi cuenta sin problemas **Criterios de Aceptación:** 1. El sistema posee vista de recuperación de contraseña 2. El sistema envía correo con link o código al correo ingresado. 3. El sistema restablece la contraseña del usuario al utilizar el link o Código. 4. El sistema restablece la contraseña hasheada utilizando Bcrypt. 5. Las Notificaciones de error funcionan correctamente. **Definición de Terminado:** 1. La funcionalidad de recuperación de contraseñas funciona correctamente. 2. El enlace o código enviado al correo restablece la contraseña con éxito. 3. El sistema hashea la nueva contraseña usando BCRYPT. 4. Las pruebas de usabilidad y funcionalidad validan que el proceso es claro y seguro. 5. Se valida que las notificaciones de error son recibidas correctamente. **Tareas a realizar:** 1. Diseñar vista de recuperación de contraseña con formulario de	Baja		No Iniciado

	recuperación de contraseña. Debe incluir un campo donde el usuario pueda introducir su dirección de correo electrónico. 2. Crear Endpoint con la lógica de recuperación de contraseña. Se debe enviar correo con un código o link para que el usuario restablezca su contraseña 3. Implementar manejo de errores en el Endpoint. 4. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 5. Validar el correcto funcionamiento del Endpoint. 6. Enviar la solicitud de recuperación de contraseña al backend al pulsar el botón de "Olvide mi contraseña". 7. Gestionar errores recibidos en la respuesta del endpoint en vista perfil de recuperación de contraseña. 8. Mostrar un mensaje confirmando que se ha enviado el correo con el enlace de recuperación. 9. Crear un nuevo formulario que permita introducir y confirmar su nueva contraseña una vez acceda desde el enlace. 10. Implementar validaciones para asegurar que las contraseñas nuevas cumplan con los requisitos de seguridad. 11. Informar al usuario si el proceso de restablecimiento fue exitoso o si hubo algún problema.			

	12. Hashear con Bcrypt y actualizar la nueva contraseña en la base de datos después de la validación del token. 13. Enviar un correo adicional al usuario informando que su contraseña ha sido restablecida.			
HU-0009	**Nombre de la Historia:** Crear diagnóstico **Como:** Un usuario **Quiero:** Registrar un diagnóstico en el sistema **Para que:** Que el sistema genere un diagnóstico basado en mis síntomas **Criterios de Aceptación:** 1. Módulo de diagnóstico creado. 2. Un usuario puede crear diagnóstico al ingresar síntomas. 3. Se registra ubicación del usuario, y fecha y hora del diagnóstico realizado. 4. Método sólo puede ser utilizado por usuarios registrados. 5. Las Notificaciones de error funcionan correctamente. 6. No se generan errores, como creación de usuarios duplicados ni explotación de vulnerabilidades. **Definición de Terminado:** 1. La funcionalidad para registrar diagnósticos basada en síntomas está implementada. 2. El sistema guarda correctamente la fecha, hora y ubicación del diagnóstico. 3. Las validaciones y manejo de errores funcionan correctamente.	**Alta**	**15**	**En Progreso**

	4. El diagnóstico se almacena en la base de datos sin problemas. 5. El módulo de diagnóstico fue integrado correctamente y cumple con el objetivo. 6. Se valida que el método sólo puede utilizado por usuario registrados. **Tareas a realizar:** 1. Dentro de la vista de Diagnósticos: Crear un botón para agregar más campos de síntomas. 2. Implementar el acceso a la ubicación del usuario mediante la api de geolocalización del navegador (Con consentimiento). 3. Implementar validaciones de los campos del formulario de la vista. 4. Crear Endpoint en el backend para recibir los síntomas del formulario y enviarlos al Endpoint del modelo de clasificación de síntomas. 5. Implementar método de autenticación de usuario en el Endpoint. 6. Implementar validaciones en el backend. 7. Implementar manejo de errores en el Endpoint. 8. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 9. Validar el correcto funcionamiento del endpoint. 10. Configurar la solicitud HTTP para enviar al backend los datos de los síntomas ingresados por el usuario en la vista de Diagnósticos. 11. Capturar la fecha y hora de la solicitud. 12. Manejar la respuesta en el frontend (200: Mostrar diagnóstico generado,			

	Error: Crear alerta notificando el tipo de error) 13. Almacenar el diagnóstico y los síntomas ingresados en la base de datos, vinculados con el usuario para su historial.			
HU-0010	**Nombre de la Historia:** Ver síntomas **Como:** Un usuario **Quiero:** Poder consultar los registros de síntoma en el sistema **Para que:** Revisar los síntomas que se han registrado **Criterios de Aceptación:** 1. Módulo de diagnóstico creado. 2. Un usuario puede ver los síntomas para realizar el diagnóstico, 3. Método sólo puede ser utilizado por usuarios registrados. **Definición de Terminado:** 1. Los usuarios pueden ver sus síntomas registrados. 2. La consulta de síntomas desde el backend funciona correctamente. 3. Se han realizado pruebas de usabilidad que confirman la funcionalidad. 4. La vista de diagnósticos se muestra de manera correcta sin errores de carga. **Tareas a realizar:** 1. Crear vista de Diagnósticos. Debe incluir formulario para registrar	**Alta**	**3**	**En progreso**

	síntomas. Debe incluir un sólo campo de síntoma. 2. Crear Endpoint en el backend para enviar todos los síntomas registrados en el sistema y enviarlos al frontend. 3. Implementar método de autenticación de usuario en el Endpoint. 4. Implementar manejo de errores en los Endpoints. 5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Validar el correcto funcionamiento de los Endpoints. 7. Implementar consulta HTTP al Endpoint en la vista de Diagnósticos. 8. Mostrar la información de los síntomas obtenida del Endpoint dentro del campo de síntoma en forma de lista dentro de la vista de Diagnósticos. 9. Gestionar errores de carga de datos en vista de Diagnósticos.			
HU-0011	**Nombre de la Historia:** Eliminar síntoma **Como:** Un usuario **Quiero:** Poder eliminar un síntoma que seleccioné mal **Para que:** Corregir los síntomas a ingresar dentro del formulario de diagnóstico **Criterios de Aceptación:** 1. Módulo de diagnóstico creado.	**Alta**	**6**	**Completado**

	2. Un usuario puede eliminar un síntoma antes de realizar el diagnóstico. 3. Método sólo puede ser utilizado por usuarios registrados. **Definición de Terminado:** 1. El usuario puede eliminar síntomas antes de enviar el diagnóstico. 2. El botón de eliminación de síntomas está implementado y funciona correctamente. 3. Las pruebas unitarias confirman que no existen errores de funcionalidad. 4. El módulo de diagnósticos fue desarrollado y cumple con la funcionalidad establecida. **Tareas a realizar:** 1. Dentro de la vista de Diagnósticos: Crear un botón para eliminar un campo de síntoma. 2. Crear lógica de eliminación del campo de síntoma en el frontend. 3. Implementar la lógica de eliminación del campo en el botón para eliminar el campo. 4. Validar el correcto funcionamiento del botón.			
HU-0012	**Nombre de la Historia:** Búsqueda de diagnósticos **Como:** Un usuario **Quiero:** Buscar los diagnósticos por nombre o fecha	**Media**		**No Iniciado**

	Para que: Encontrar los diagnósticos recibidos y revisarlos sin tener que navegar por todos los registros. **Criterios de Aceptación:** 1. Módulo de diagnóstico creado. 2. Se puede realizar búsqueda de los diagnósticos por fecha o nombre. 3. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. 4. Método sólo puede ser utilizado por usuarios registrados. **Definición de Terminado:** 1. La búsqueda por nombre o fecha está implementada. 2. Las pruebas de usabilidad muestran que la funcionalidad es clara y eficiente. 3. Los resultados se muestran correctamente y no existen problemas de rendimiento. 4. El módulo de diagnósticos fue desarrollado y cumple con la funcionalidad establecida. **Tareas a realizar:** 1. Implementar el formulario de búsqueda de usuarios en la vista de Diagnósticos. 2. Crear Endpoint con la lógica de búsqueda en el backend, para recibir las solicitudes de búsqueda (por fecha o nombre) 3. Implementar método de autenticación de usuario en el Endpoint.			

	4. Implementar manejo de errores en el Endpoint. 5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Implementar consulta HTTP al en la vista de Diagnósticos. 7. Mostrar la información del usuario obtenida del Endpoint dentro de la vista de Diagnósticos. 8. Gestionar errores y excepciones, como diagnósticos no encontrados o problemas en la base de datos.			
HU-0013	**Nombre de la Historia:** Seguimiento de diagnósticos **Como:** Un administrador **Quiero:** Tener acceso a una interfaz de dashboard que muestra un resumen de los síntomas registrados en el sistema **Para qué:** Monitorear y analizar las tendencias de los diagnósticos en tiempo real y detectar posibles patrones o problemas. **Criterios de Aceptación:** 1. Módulo de dashboard. 2. Se puede visualizar gráfico de mapa con la cantidad de diagnósticos realizados por zona. 3. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. 4. Método sólo puede ser utilizado por administradores registrados. **Definición de Terminado:** 1. El dashboard con gráficos de diagnósticos está implementado.	**Alta**		**No Iniciado**

	2. La funcionalidad de filtro y visualización de datos en tiempo real está lista. 3. Se han realizado pruebas de usabilidad y se confirman los resultados de diagnóstico por zona. 4. El método sólo puede ser utilizado por administradores registrados. 5. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. **Tareas a realizar:** 1. Crear la vista del dashboard. Debe incluir al menos un gráfico de mapa. Utilizar librerías como Chart.js o D3.js. 2. Crear Endpoint con la lógica de obtención de la data de síntomas en el backend. 3. Agregar lógica para procesar, agrupar y obtener estadística de los datos (En caso de ser necesario). 4. Implementar método de autenticación de usuario en el Endpoint. 5. Implementar manejo de errores en el Endpoint. 6. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 7. Implementar consulta HTTP al en la vista del dashboard. 8. Implementar manejo de filtros en la vista del dashboard (En caso de ser necesario).			
HU-0014	**Nombre de la Historia:** Suscripción de noticias **Como:** Un usuario	**Baja**		**No Iniciado**

	Quiero: Registrar mi correo electrónico para suscribirme a noticias **Para qué:** Recibir información relevante y actualizaciones de la plataforma. **Criterios de Aceptación:** 1. Módulo de noticias creado. 2. Un usuario puede suscribir su correo. 3. Método sólo puede ser utilizado por usuarios tanto registrados como no registrados. 4. Las Notificaciones de error funcionan correctamente. 5. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. **Definición de Terminado:** 1. El módulo de suscripción está implementado. 2. Las validaciones y manejo de errores funcionan correctamente. 3. Los correos de suscripción se almacenan y procesan de forma correcta. 4. El módulo de noticias fue creado y cumple con su objetivo. 5. El método sólo puede ser utilizado por usuarios registrados. 6. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades.			

	Tareas a realizar: 1. Dentro de la vista de inicio: Se debe crear un formulario de suscripción, solo debe incluir un campo para ingresar el correo y un botón para enviar. 2. Implementar validaciones en el frontend del campo. 3. Crear Endpoint en el backend para recibir el correo y registrarlo en la base de datos. 4. Implementar validaciones en el backend. 5. Implementar manejos de errores en el Endpoint. 6. Implementar envío de respuestas con el resultado de la consulta en el Endpoint. 7. Validar el correcto funcionamiento del Endpoint. 8. Configurar la solicitud HTTP para enviar la dirección de correo al backend. 9. Manejar la respuesta en el frontend (200: Crear alerta de confirmación. Error: Crear alerta notificando el tipo de error).			
HU-0015	**Nombre de la Historia:** Histórico de Diagnósticos **Como:** Un usuario **Quiero:** Visualizar mis diagnósticos anteriores con sus detalles correspondientes **Para que:** Poder hacer un seguimiento de mi estado de salud a lo largo del tiempo.	**Alta**		**No Iniciado**

	Criterios de Aceptación: 1. Módulo de diagnóstico creado. 2. Se puede visualizar todos los diagnósticos realizados por el usuario activo en una tabla. 3. No se generan errores, como problemas de comunicación entre el backend y el frontend. 4. Método sólo puede ser utilizado por usuarios registrados. **Definición de Terminado:** 1. Los usuarios pueden ver su historial de diagnósticos. 2. Los datos se muestran correctamente en una tabla sin errores. 3. Se han realizado pruebas de usabilidad que confirman la funcionalidad. 4. El método sólo puede ser utilizado por usuarios registrados. **Tareas a realizar:** 1. Crear vista de visualización de diagnósticos. (Se deben mostrar los diagnósticos en forma de tabla) 2. Crear Endpoint en el backend para recibir los diagnósticos registrados por el usuario. 3. Implementar método de autenticación de usuario en el Endpoint. 4. Implementar manejo de errores en el Endpoint.			

	5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Validar el correcto funcionamiento del endpoint. 7. Configurar la solicitud HTTP para recibir los diagnósticos registrados por el usuario. 8. Manejar la respuesta en el frontend (200: Mostrar listado de diagnósticos, Error: Crear alerta notificando el tipo de error)			
HU-0016	**Nombre de la Historia:** Crear alertas por anomalías en diagnósticos **Como:** Un administrador **Quiero:** Poder generar alertas cuando se detecten anomalías en los diagnósticos **Para qué:** Identificar y predecir posibles brotes de enfermedades. **Criterios de Aceptación:** 1. Módulo de alertas creado. 2. Un administrador puede crear alertas. 3. Método sólo puede ser utilizado por administradores registrados. 4. Las Notificaciones de error funcionan correctamente. 5. No se generan errores, como creación de usuarios duplicados ni explotación de vulnerabilidades. **Definición de Terminado:** 1. La funcionalidad de alertas está implementada.	Alta		No Iniciado

	2. Se valida que el administrador puede crear y enviar alertas. 3. Se confirma que el método sólo puede ser utilizado por los administradores registrados. 4. Las validaciones y manejo de errores funcionan correctamente. 5. Se han realizado pruebas unitarias y de integración con éxito. **Tareas a realizar:** 1. Implementar una sección dentro de la vista de seguimiento de diagnósticos donde el administrador pueda crear y gestionar alertas en el sistema. 2. Crear Endpoint en el backend para creación de alertas. 3. Implementar método de autenticación de usuario en el Endpoint. 4. Implementar validaciones en el Endpoint. 5. Implementar manejo de errores en el Endpoint. 6. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 7. Validar el correcto funcionamiento del Endpoint.			
HU-0017	**Nombre de la Historia:** Ver alertas **Como:** Un usuario **Quiero:** Consultar las alertas que se hayan registrado	**Media**		**No Iniciado**

	Para que: Estar al tanto de posibles riesgos o anomalías. **Criterios de Aceptación:** 1. Módulo de alertas creado. 2. Un usuario puede ver las alertas creadas previamente en el icono de alertas. 3. Método sólo puede ser utilizado por administradores registrados. 4. Las Notificaciones de error funcionan correctamente. 5. No se generan errores, como creación de usuarios duplicados ni explotación de vulnerabilidades. **Definición de Terminado:** 1. El sistema de alertas funciona correctamente. 2. Los administradores pueden ver las alertas registradas y los usuarios son notificados. 3. Se confirma que método sólo puede ser utilizado por los administradores registrados en el sistema. 4. Se valida que las notificaciones de error funcionan correctamente. 5. Se han validado las pruebas de usabilidad para asegurar que las alertas se despliegan sin errores. **Tareas a realizar:** 1. Implementar icono de alertas en el navbar.			

	2. implementar sistema de notificación en el frontend. 3. Configurar la solicitud HTTP para consultar el Endpoint y recuperar las alertas. 4. Desplegar las notificaciones al clicar en el icono de alertas. 5. Al seleccionar una alerta, se debe mostrar la información detallada como la causa, síntomas relacionados, fecha y recomendaciones a seguir. 6. Implementar notificaciones visibles para alertas recientes o de alta prioridad cuando el usuario acceda al sistema.			
HU-0018	**Nombre de la Historia:** Búsqueda de alertas **Como:** Un usuario **Quiero:** Buscar alertas por criterios como nombre, fecha o ubicación **Para que:** Encontrar información relevante rápidamente y estar informado sobre posibles anomalías. **Criterios de Aceptación:** 1. Módulo de alertas creado. 2. Un usuario puede ver las alertas creadas previamente en forma de tabla. 3. Método sólo puede ser utilizado por administradores registrados. 4. Las Notificaciones de error funcionan correctamente.	Media		No Iniciado

	5. No se generan errores, como creación de usuarios duplicados ni explotación de vulnerabilidades. **Definición de Terminado:** 1. La búsqueda de alertas por criterios específicos está implementada. 2. El módulo de alertas fue desarrollado de acuerdo a las especificaciones establecidas por el equipo. 3. Los usuarios interesados pueden ver las alertas que fueron enviadas por el admin. 4. Los usuarios pueden filtrar las alertas por nombre, fecha o ubicación. 5. Se valida que las notificaciones de error funcionen correctamente. 6. Las pruebas de usabilidad confirman que la funcionalidad es precisa y eficiente. 7. Este método sólo puede ser utilizado (modificado) por los administradores. **Tareas a realizar:** 1. Implementar vista de alertas de usuario. 2. Implementar un formulario donde el usuario pueda ingresar criterios de búsqueda como nombre de la alerta, fecha. o ubicación. 3. Permitir al usuario filtrar las alertas.			

	4. Renderizar la lista de alertas que coincidan con los criterios de búsqueda ingresados.			
HU-0019	**Nombre de la Historia:** Control de accesos y permisos **Como:** Un administrador **Quiero:** Gestionar los permisos de acceso y edición de los registros de usuarios y diagnósticos **Para que:** Garantizar que solo los usuarios autorizados puedan modificar o ver la información relevante según su nivel de acceso. **Criterios de Aceptación:** 1. Módulo de administración de usuarios creado. 2. Sólo los usuarios con rol de administrador pueden acceder. 3. Se puede ver, editar y eliminar usuarios. 4. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. **Definición de Terminado:** 1. La funcionalidad de gestión de permisos está implementada. 2. Sólo los administradores pueden acceder, asignar roles y permisos. 3. Sólo los administradores pueden ver, editar y eliminar usuarios.	Alta		No Iniciado

	4. Las pruebas de usabilidad y seguridad confirman que el acceso está restringido correctamente. 5. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. **Tareas a realizar:** 1. Implementar vista de administración de usuarios para asignar o modificar permisos de acceso y edición para usuarios. 2. Permitir que el administrador seleccione un usuario y asigne roles (por ejemplo: administrador, usuario regular) con diferentes niveles de acceso. 3. Mostrar los permisos asignados a cada usuario, incluyendo acceso. (esta información se despliega en forma de tabla). 4. Mostrar notificaciones o mensajes cuando los permisos sean modificados exitosamente. 5. Asegurar que la estructura de la base de datos refleje los roles y permisos asociados a cada usuario, permitiendo o restringiendo el acceso a ciertas funcionalidades o datos. 6. Manejar errores de permiso no válidos o problemas al actualizar roles, (uso de alerta).			

HU-0020	**Nombre de la Historia:** Diseño de la interfaz (Fácil aprendizaje, Interfaz responsive, Diseño amigable e intuitivo, y Compatibilidad Multiplataforma) **Como:** Un usuario **Quiero:** Una interfaz amigable, intuitiva y adaptable **Para que:** Poder utilizar el sistema en cualquier dispositivo de manera fluida y sin dificultades. **Criterios de Aceptación:** 1. La interfaz debe ser responsive, adaptándose correctamente a diferentes tamaños de pantalla como móviles, tablets y escritorios sin pérdida de funcionalidad o claridad. 2. Los elementos de la interfaz (botones, formularios, menús) deben ajustarse y ser fáciles de usar tanto en dispositivos táctiles como en equipos de escritorio. 3. La navegación entre las diferentes secciones debe ser clara, rápida y sencilla, con un menú accesible e intuitivo que mantenga la consistencia en todas las plataformas 4. La interfaz debe ser compatible con los principales navegadores (Chrome, Firefox, Safari, Edge) y funcionar sin problemas en los sistemas operativos más comunes (iOS, Android, Windows, macOS). 5. Deben realizarse pruebas de usabilidad con usuarios en diferentes dispositivos para garantizar que la	**Alta**	**6**	**En progreso**

	interfaz sea entendible, fácil de usar y no presente dificultades. **Definición de Terminado:** 1. La interfaz es responsive y se adapta correctamente a todos los dispositivos. 2. Los elementos de la interfaz (botones, formularios, menús) se ajustan y son fáciles de usar. 3. Las pruebas de usabilidad confirman que el diseño es intuitivo y funcional en múltiples plataformas. 4. La interfaz es compatible con los principales navegadores sin pérdida de funcionalidad. **Tareas a realizar:** 1. Implementar un diseño adaptativo, que se ajuste a diferentes tamaños de pantalla (móvil, tablet, escritorio). 2. Crear una barra de navegación clara y accesible, que permita al usuario moverse fácilmente por el sistema. 3. Se deben incorporar elementos interactivos que sean fáciles de usar, como botones grandes, formularios claros y campos de entrada con suficiente espacio. 4. La implementación del diseño debe ser limpio y minimalista. 5. Se deben realizar pruebas de usabilidad para asegurar que el sistema es fácil de entender y usar.			

	6. Se debe configurar el backend para manejar solicitudes de diferentes tipos de dispositivos. 7. Compatibilidad cross-browser, el sistema debe funcionar correctamente en diferentes navegadores (Chrome, Firefox, Safari, Edge). 8. Implementar un sistema de autenticación que funcione sin problemas en dispositivos móviles y de escritorio. 9. Incluir tutoriales o guías de usuario interactivos que explican cómo usar el sistema. 10. Agregar animaciones ligeras para transiciones y botones mejorando la fluidez del sistema sin sobrecargar la interfaz.			
HU-0021	**Nombre de la Historia:** Capacidad del sistema (Tiempo de carga, Tiempo de respuesta, Capacidad de usuarios y Disponibilidad del sistema 24/7) **Como:** Un usuario **Quiero:** Poder usar el sistema en cualquier momento y que cargue rápidamente, responda a mis acciones en menos de 2 segundos, y mantenga un rendimiento óptimo, incluso cuando haya hasta 1000 usuarios concurrentes **Para qué:** Utilizar la aplicación de manera fluida, sin interrupciones o demoras, sin importar cuántas personas están usando el sistema simultáneamente.	Alta		No Iniciado

	Criterios de Aceptación: 1. Todos los elementos se cargan en un máximo de 7 segundos. 2. El tiempo de respuesta toma 2 segundos como máximo. 3. El aplicativo logra estar disponible durante 1 semana seguida. 4. El aplicativo supera las pruebas de carga. **Definición de Terminado:** 1. El sistema carga los elementos en un máximo de 7 segundos. 2. El sistema responde en menos de 2 segundos bajo una carga de hasta 1000 usuarios concurrentes. 3. Todas las pruebas de rendimiento y carga han sido aprobadas. 4. El sistema ha estado disponible 24/7 durante una semana sin interrupciones. **Tareas a realizar:** 1. Configurar múltiples instancias del backend utilizando herramientas de escalado como Kubernetes o Docker Swam. 2. Usar balanceador de carga (NGINX o AWS Elastic Load Balancer). 3. Implementar índices en las tablas más consultadas y optimizar las consultas SQL. 4. Implementar sistema de caché para almacenar resultados de consultas			

	más frecuentes (Redis o Memcached). 5. Minimizar el tamaño de los archivos JavaScript, CSS e imágenes con herramientas como Gzip para comprimir. 6. Implementación de lazy loading. 7. Reducir el número de solicitudes con técnicas como batching o polling eficiente. 8. Pruebas de rendimiento del frontend con Lighthouse o WebPageTest.			
HU-0022	**Nombre de la Historia:** Código estructurado (Documentación de código y Sistema modular) **Como:** Un desarrollador **Quiero:** Que el código esté bien estructurado en módulos y documentado **Para que:** Asegurar que el sistema sea fácil de mantener, modificar y expandir en el futuro. **Criterios de Aceptación:** 1. El código del aplicativo se carga en GitHub. 2. Se implementa arquitectura MVC. 3. Los archivos tienen nombres representativos y reconocibles. 4. Se implementan comentarios en todo el código relevante. 5. Se realiza manual técnico. **Definición de Terminado:** 1. El código de la aplicación Medical AID fue guardado en GitHub.	**Alta**	**6**	**En Progreso**

	2. El código está estructurado según el patrón MVC. 3. Toda la documentación del código está completada y se ha generado un manual técnico. 4. El código ha sido revisado y cumple con las normas de calidad de software. **Tareas a realizar:** 1. Crear un repositorio en GitHub. 2. Implementar la arquitectura modular. 3. Organizar el código en módulos o componentes pequeños y reutilizables, cada uno encargado de una funcionalidad específica (autenticación, gestión de usuarios, procesamiento de diagnósticos) 4. Utilizar una convención de nombres coherente para los módulos, funciones, y clases, y agruparlas lógicamente por función o área del sistema. 5. Utilizar sistemas de gestión de dependencias como npm o pip para mantener un control claro de las bibliotecas externas utilizadas en cada módulo, asegurando que los módulos no dependan de forma rígida entre sí. 6. Mantener el código de la lógica de negocio separado de la lógica de presentación (ej. separar el backend del frontend), asegurando que los cambios en uno no afecten al otro.			

	7. Incluir comentarios claros y concisos para explicar la funcionalidad de bloques de código complejos o poco evidentes, siguiendo buenas prácticas de comentario. 8. Utilizar comentarios para documentar la finalidad de cada función, clase o método, explicando sus parámetros, valores de retorno y posibles excepciones. 9. Documentación de la API, proporcionar documentación detallada de los endpoints, parámetros, respuestas y códigos de error. 10. Escribir una guía detallada para que otros desarrolladores sepan cómo instalar, configurar y ejecutar el sistema, incluyendo dependencias y pasos de configuración inicial. (Este es el manual técnico) 11. Mantener una estructura clara en el repositorio del código, con carpetas adecuadamente organizadas y archivos de configuración.			
HU-0023	**Nombre de la Historia:** Encriptación de datos **Como:** Un desarrollador **Quiero:** Qué los datos sensibles estén cifrados durante la transmisión **Para que:** Garantizar la seguridad de los datos. **Criterios de Aceptación:**	**Alta**		**No Iniciado**

	1. Se implementa método de configuración HTTPS. 2. Se implementa certificado SSL. 3. Se implementa cifrado TLS en datos de comunicación. 4. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. 5. Se encriptan los datos sensibles como las contraseñas. 6. Se aprueban las pruebas de pentesting. **Definición de Terminado:** 1. La comunicación entre el frontend y el backend utiliza HTTPS. 2. El certificado de SSL fue implementado. 3. El cifrado TLS en datos de comunicación fue implementado. 4. Los datos sensibles como las contraseñas fueron encriptadas. 5. Se han pasado las pruebas de pentesting para verificar la seguridad. 6. No se generan errores, como problemas de comunicación entre el backend y el frontend ni explotación de vulnerabilidades. **Tareas a realizar:** 1. Implementar HTTPS, asegurar que todas las solicitudes realizadas desde el frontend al backend se realicen a través de HTTPS, para garantizar que			

	los datos se transmitan de manera segura. 2. Configurar HTTPS en el servidor para aceptar solo conexiones HTTPS, garantizando que todos los datos transmitidos entre el frontend y el backend estén cifrados. 3. Verificar que el certificado SSL del servidor esté correctamente configurado y sea válido para evitar ataques de tipo "Man-in-the-middle." 4. Cifrado en formularios, asegurarse de que los datos sensibles (por ejemplo, contraseñas, información médica) ingresados en los formularios estén cifrados antes de ser enviados, utilizando mecanismos como el cifrado del lado del cliente, si es necesario. 5. Uso de tokens de acceso seguros, asegurar que los tokens de autenticación (por ejemplo, JWT) estén cifrados y firmados, y se transmitan de forma segura a través de HTTPS. 6. Cifrado de datos en tránsito, utilizar protocolos seguros como TLS (Transport Layer Security) para cifrar todos los datos en tránsito entre el servidor y los clientes, incluyendo llamadas API y respuestas. 7. Implementación de HSTS (HTTP Strict Transport Security): Configurar HSTS para obligar a los navegadores a realizar siempre solicitudes a través			

	de HTTPS, evitando cualquier posible degradación a HTTP. 8. Encriptar datos sensibles antes de enviarlos: Cifrar información especialmente sensible (como contraseñas o datos médicos) antes de enviarla a través de la red, garantizando que esté protegida durante toda la transmisión. 9. Pruebas de seguridad (Penetration testing): Realizar pruebas de penetración para asegurarse de que el cifrado y otras medidas de seguridad no tengan vulnerabilidades que podrían ser explotadas.			
HU-0024	**Nombre de la Historia:** Pruebas (Pruebas de Usabilidad y Pruebas de Integración) **Como:** Un desarrollador **Quiero:** Realizar pruebas de usabilidad e integración **Para que:** Asegurar que el sistema sea fácil de usar y funcione correctamente. **Criterios de Aceptación:** 1. Se generan casos de uso y escenarios más relevantes. 2. Se obtiene feedback informativo y útil de las pruebas de usabilidad. 3. Se aplican mejoras a partir del feedback. 4. La aplicación funciona correctamente en los navegadores más populares.	Alta		No Iniciado

	5. Se aprueban las pruebas de integración con un 80% de aprobación. **Definición de Terminado:** 1. Los casos de uso y los escenarios más relevantes fueron generados. 2. Se obtuvo feedback informativo y útil de las pruebas de usabilidad e integración. 3. Las pruebas de integración han sido aprobadas con un 80% de éxito. 4. El feedback de los usuarios ha sido implementado para mejorar la experiencia. 5. La aplicación funciona correctamente en los navegadores más populares. **Tareas a realizar:** 1. Identificar casos de uso y escenarios clave que representen las principales interacciones del usuario con el sistema (ej, autentificación, registro de síntomas, visualización de alertas). 2. Observar cómo los usuarios navegan por la aplicación, registran síntomas, consultan diagnósticos o realizan otra tareas. 3. Recoger comentarios de los usuarios sobre la factibilidad de uso, claridad de la interfaz y sugerencias para mejorar la experiencia. 4. Analizar los datos obtenidos de las pruebas de usabilidad, identificando			

	patrones comunes en patrones comunes en problemas de navegación, tiempo de respuesta, o funcionalidades difíciles de entender. 5. Identifica los componentes clave del sistema que deben ser integrados, como frontend y backend, APIs, base de datos y servicios externos. 6. Desarrollar casos de prueba para asegurarse de que los módulos individuales interactúan correctamente entre si (ej, envío de datos de registro de síntomas del frontend al backend y verificación en la base de datos). 7. Implementar pruebas de integración automatizadas, para validar el funcionamiento de los componentes de forma continua. 8. Verificación de comunicación entre servicios, asegurarse de que los servicios externos o internos se comunican correctamente entre si. 9. Pruebas de errores y excepciones, para asegurar que el sistema maneja adecuadamente las excepciones y fallos en la integración de componentes. 10. Evaluar el rendimiento de las integraciones bajo diferentes cargas de trabajo, asegurando que no haya cuellos de botella o tiempo de espera excesivos entre módulos. 11. Realizar pruebas de usabilidad e integración en diferentes dispositivos (móvil, tablet, escritorio) y			

	navegadores (Chrome, Firefox, Safari, Edge) para garantizar compatibilidad.			
HU-0025	**Nombre de la Historia:** Alertas y notificaciones automáticas **Como:** Un usuario **Quiero:** Recibir alertas y notificaciones cuando ocurran problemas de validación **Para qué:** Estar informado de problemas o errores en tiempo real. **Criterios de Aceptación:** 1. Todos los endpoint manejan todos los errores detectados. 2. Todos los endpoint envían respuestas necesarias para generar alertas. 3. Se utiliza la información de respuesta para generar alerta en caso de errores para todas las interacciones con endpoint. **Definición de Terminado:** 1. Los endpoint son capaces de manejar todos los errores detectados. 2. Los endpoint realizan envíos de respuestas para generar alertas. 3. Se utiliza la información de respuesta para generar alerta en caso de errores para todas las interacciones con endpoint. **Tareas a realizar:**	Alta		No Iniciado

	1. Validar el correcto manejo de errores en los endpoint del backend. 2. Validar el correcto envío de respuestas en los endpoint del backend. 3. Capturar los errores en todos las consultas HTTP dentro del frontend. 4. Implementar notificaciones con SweetAlert2 y/o React-Toastify en las consultas HTTP. (SweetAlert2 para errores del frontend y React-Toastify para errores del backend). 5. Utilizar la información de los errores capturados en las notificaciones. Considerar tanto errores del frontend como errores del backend.			
HU-0026	**Nombre de la Historia:** Formulario intuitivo **Como:** Un usuario **Quiero:** Acceder a formularios intuitivos **Para que:** Me permitan ingresar síntomas de manera fácil y organizada. **Criterios de Aceptación:** 1. Se implementa formulario intuitivo con los síntomas agrupados por categoría. **Definición de Terminado:** 1. El formulario de síntomas está implementado con funcionalidades de agregar y eliminar síntomas. 2. El formulario es intuitivo y las pruebas de usabilidad confirman su facilidad de uso.	Alta		No Iniciado

	3. El sistema redirige correctamente cuando el usuario necesita ayuda adicional. **Tareas a realizar:** 1. Validar que el formulario de la vista de diagnóstico tenga un sólo campo de síntoma y las funcionalidades correctas (Agregar síntoma, borrar síntoma y diagnosticar). 2. Añadir botón de: ¿No sabes lo que tienes?. 3. Implementar vista de formulario intuitivo. 4. Implementar redirección en el botón "¿No sabes lo que tienes?" para dirigir a la vista de formulario intuitivo. 5. Implementar diseño de canva del formulario intuitivo. 6. Configurar la solicitud HTTP para enviar los síntomas al backend ingresados por el usuario en la vista de formulario intuitivo. 7. Capturar la fecha y hora de la solicitud. 8. Manejar la respuesta en el frontend (200: Mostrar diagnóstico generado, Error: Crear alerta notificando el tipo de error).			
HU-0027	**Nombre de la Historia:** Integración del Modelo de ML **Como:** Un desarrollador **Quiero:** Integrar el modelo de machine learning en el sistema	**Alta**	**7**	**Completado**

	Para que: Poder utilizar el modelo de machine learning dentro de la aplicación web. **Criterios de Aceptación:** 1. Se crea backend en Python disponibilizado modelo de machine learning. 2. Se genera comunicación entre Node.js con Python. 3. Se generan clasificaciones de síntomas correctamente. 4. El frontend recibe el resultado del diagnóstico **Definición de Terminado:** 1. El modelo de machine learning está correctamente integrado en el backend. 2. La comunicación entre Node.js y la API de Python funciona sin errores. 3. Las predicciones de síntomas son precisas y el frontend muestra correctamente los resultados. **Tareas a realizar:** 1. Desarrollar un endpoint de API en Python para consumir el modelo de ML. (Se puede usar la librería Flask para el desarrollo y la página Replit para el despliegue en línea). 2. Cargar el modelo en el backend. (En caso de haberse implementado localmente).			

	3. Implementar comunicación entre Node.js y la API de Python. 4. Implementar lógica de predicción en el backend para poder procesar los síntomas recibidos y entregarlos al modelo. 5. Implementar lógica de respuesta con el resultado. 6. Implementar manejo de errores en el endpoint. 7. Se valida el correcto funcionamiento del endpoint.			
HU-0028	**Nombre de la Historia:** Respaldo **Como:** Un administrador **Quiero:** Que el sistema cuente con una función que respalde automáticamente la base de datos actual **Para que:** Garantizar la protección de la información y poder restaurar los datos en caso de pérdida de datos. **Criterios de Aceptación:** 1. Se genera script de respaldo. 2. Se aplica la ejecución automática con frecuencia definida al script. 3. Se configura acceso a respaldos. 4. Se implementa monitoreo. **Definición de Terminado:** 1. El sistema de respaldo automático está implementado y funcionando según la frecuencia definida.	Alta		No Iniciado

	2. El acceso a los respaldos está restringido solo a usuarios autorizados. 3. Se ha configurado un sistema de monitoreo que alerta sobre errores en los respaldos. **Tareas a realizar:** 1. Determinar frecuencia de respaldos. 2. Determinar tipo de respaldo (Completo, incremental o diferencial). 3. Crear script de respaldo de base de datos (pg_dump para postgresql). 4. Programar tarea para ejecutar script automáticamente con la frecuencia definida 5. Configurar permisos de acceso (Sólo usuarios autorizados pueden acceder a los respaldos) 6. Implementar monitoreo. Al detectar fallos o errores se debe enviar una alerta al administrador.			
HU-0029	**Nombre de la Historia:** Crear modelo de machine learning **Como:** Un desarrollador **Quiero:** Crear un modelo de machine learning **Para que:** Generar diagnósticos precisos a partir de los síntomas ingresados por los usuarios. **Criterios de Aceptación:** 1. Se genera modelo de machine learning.	**Alta**	**9**	**Completado**

	2. Accuracy del modelo mayor a 95%. 3. Se aplica la metodología CRISP-DM para la generación del modelo. 4. Se exporta correctamente el modelo. **Definición de Terminado:** 1. El modelo de machine learning ha sido entrenado y validado, con una precisión superior al 95%. 2. El modelo está exportado correctamente y listo para su integración. 3. La metodología CRISP-DM es aplicada para la generación del modelo. **Tareas a realizar:** 1. Recolectar los datos. 2. Analizar los datos. 3. Limpiar y transformar los datos. 4. Entrenar el modelo de Machine Learning. 5. Validar y evaluar el modelo. 6. Guardar el modelo con pickle o jotblib.			
HU-0030	**Nombre de la Historia:** Crear vista de inicio **Como:** Un desarrollador **Quiero:** Crear una interfaz para la vista del inicio **Para que:** Mostrar una interfaz de inicio al usuario que ingrese al sistema. **Criterios de aceptación:**	**Alta**	**2**	**En progreso**

	1. Se crea vista de inicio. 2. Diseño de mockups. 3. Se incorpora estado de usuario. 4. La interfaz es clara y fácil de navegar. **Definición de Terminado:** 1. La vista de inicio está implementada y es responsive. 2. El diseño de mockups es respetado e implementado. 3. El estado de los usuarios es incorporado. 4. Las pruebas de usabilidad confirman que la interfaz es clara y fácil de navegar. **Tareas a realizar:** 1. Definir la estructura de la vista. 2. Crear componentes principales (Header, Footer, etc). 3. Incorporar elementos visuales (Imágenes, íconos, etc). 4. Aplicar estilos CSS/SCSS. 5. Hacer la vista responsive. 6. Añadir lógica de navegación. 7. Incorporar el estado del usuario (Mostrar iniciar sesión o cerrar sesión).			
HU-0031	**Nombre de la Historia:** Diseño de Mockup y Prototipos **Como:** Un diseñador de interfaz de usuario (UI) **Quiero:** Crear mockups y prototipos interactivos del sistema	**Alta**	**6**	**Completado**

	Para que: Los desarrolladores puedan visualizar e implementar el diseño del sistema. **Criterios de Aceptación:** 1. Los mockups deben representar todas las pantallas clave del sistema con sus componentes y disposición. 2. El prototipo interactivo debe permitir la navegación entre pantallas simulando las interacciones principales del usuario. 3. Los mockups y prototipos deben ser compartidos con los interesados para su revisión y aprobación antes de proceder con la fase de desarrollo. 4. El diseño debe ser consistente con la guía de estilo y branding de la aplicación. **Definición de Terminado:** 1. Los mockups representaron todas las pantallas clave del sistema con sus componentes y disposición. 2. Los mockups y prototipos interactivos están completados y aprobados por el equipo de interesados. 3. Los desarrolladores han recibido el diseño final para su implementación. 4. El prototipo interactivo simula correctamente las interacciones principales del usuario.			

	Tareas a realizar: 1. Recopilar los requisitos funcionales. 2. Diseño de Mockups, crear versiones de alta fidelidad de las pantallas clave, incluyendo colores, tipografía, iconos y otros elementos visuales. 3. Compartir los mockups y prototipos con el equipo para recibir retroalimentación y realizar ajustes. 4. Una vez aprobado, entregar el diseño final a los desarrolladores para la implementación.			
HU-0032	**Nombre de la Historia:** Documentación final **Como:** Un diseñador de interfaz de usuario (UI) **Quiero:** Crear documentación final del proyecto. **Para que:** Facilitar la entrega del proyecto, asegurando su calidad y correcto funcionamiento, evaluar el éxito del proyecto y extraer aprendizajes obtenidos tras el desarrollo del proyecto que puedan aplicarse a futuro. **Criterios de aceptación:** 1. El manual técnico debe contener una descripción detallada de la arquitectura del sistema, las tecnologías utilizadas, la estructura del código, y las instrucciones para la instalación y despliegue del proyecto. 2. El manual de usuario debe incluir instrucciones claras y concisas sobre cómo utilizar la aplicación, con	Media		No Iniciado

	capturas de pantalla y ejemplos de las principales funcionalidades. 3. El informe debe documentar los resultados de las pruebas unitarias, de integración y de usabilidad, incluyendo los casos de prueba, resultados, y cualquier problema encontrado. 4. Debe incluir una descripción detallada de cómo se integran los distintos módulos del sistema, los servicios externos utilizados, y cualquier dependencia entre ellos. 5. Debe documentar el proceso de desarrollo, los objetivos alcanzados, las dificultades encontradas, y las lecciones aprendidas durante el desarrollo del proyecto. 6. Toda la documentación debe estar subida al repositorio del proyecto, organizada y disponible para su consulta. 7. La documentación debe ser revisada y aprobada por los miembros del equipo y los interesados antes de su entrega final. **Definición de Terminado:** 1. El manual técnico está completo y documenta la arquitectura del sistema, las tecnologías utilizadas, la estructura del código y las instrucciones detalladas para la instalación y despliegue del proyecto. 2. El manual de usuario está elaborado, contiene capturas de pantalla y			

	ejemplos que guían a los usuarios en el uso de las funcionalidades clave de la aplicación. 3. El informe de pruebas documenta todos los casos de prueba (unitarios, de integración y de usabilidad) y contiene los resultados, identificando problemas críticos y su resolución. 4. El informe de integración describe cómo los módulos del sistema interactúan entre sí, incluyendo las dependencias y los servicios externos utilizados. 5. Se ha completado el informe de cierre, documentando el cumplimiento de los objetivos, las dificultades encontradas y las lecciones aprendidas. 6. Toda la documentación ha sido subida y organizada correctamente en el repositorio del proyecto, accesible para su consulta por todos los miembros del equipo. 7. La documentación ha sido revisada por el equipo y aprobada por los interesados, cumpliendo con los estándares de calidad y los requisitos del proyecto. **Tareas a realizar:** 1. Desarrollar manual técnico. 2. Desarrollar manual de usuarios. 3. Desarrollar manual de despliegue. 4. Desarrollar informe de cierre de proyecto. 5. Desarrollar informe de pruebas.			

	6. Desarrollar informe de integración.			
HU-0033	**Nombre de la Historia:** Documentación de seguridad **Como:** Un desarrollador **Quiero:** Crear informe de pruebas unitarias y de integración **Para que:** Asegurar la calidad del software y su correcto funcionamiento antes de la entrega final. **Criterios de Aceptación:** 1. El informe debe incluir el resultado de las pruebas unitarias realizadas, con detalles de los casos de prueba. 2. El informe debe incluir el resultado de las pruebas de integración, con una descripción de los módulos o servicios probados. 3. Debe estar documentada la cobertura de pruebas, especificando el porcentaje de código cubierto. 4. El informe debe ser validado y revisado por el equipo de QA. 5. Los problemas encontrados durante las pruebas deben estar documentados y categorizados (bug menor, crítico, etc.). 6. Se debe generar una conclusión sobre la calidad del software basándose en los resultados de las pruebas. 7. El informe debe estar disponible en el repositorio de documentación del proyecto.	**Alta**		**No Iniciado**

	Definición de Terminado: 1. El informe de pruebas unitarias y de integración ha sido completado y revisado por el equipo de QA. 2. Se ha documentado la cobertura de pruebas y los problemas encontrados durante las pruebas. 3. El informe está disponible en el repositorio de documentación del proyecto. **Tareas a realizar:** 1. Definir los casos de prueba unitarios e implementarlos. 2. Ejecutar las pruebas unitarias y documentar los resultados. 3. Definir los casos de prueba de integración y prepararlos para su ejecución. 4. Ejecutar las pruebas de integración y documentar los resultados. 5. Generar un informe con los resultados de las pruebas (unitarias e integración), incluyendo la cobertura de código. 6. Revisar el informe con el equipo de QA para recibir retroalimentación y hacer ajustes si es necesario. 7. Subir el informe final al repositorio de documentación del proyecto.			
HU-0034	**Nombre de la Historia:** Agregar Síntoma **Como:** Un usuario **Quiero:** Agregar un síntoma en el formulario de síntomas	**Alta**	**10**	**Completado**

	Para que: Pueda seleccionar los síntomas antes de realizar el diagnóstico **Criterios de aceptación:** 1. Un usuario puede agregar un síntoma en el formulario de diagnóstico. 2. No se generan errores, como la obtención de datos erróneos. 3. Las Notificaciones de error funcionan correctamente. **Definición de Terminado:** 1. La funcionalidad de agregar un síntoma está completamente implementada. 2. Las validaciones del frontend y backend funcionan correctamente. 3. No permite agregar más de 16 síntomas. 4. Al agregar un nuevo síntoma, el nuevo listado no contiene síntomas previamente seleccionados. **Tareas a realizar:** 1. Implementar la vista de formulario donde el usuario pueda realizar un diagnóstico. 2. Implementar validaciones en el frontend. 3. Crear Endpoint en el backend para devolver un listado con los síntomas registrados en la BBDD. 4. Implementar validaciones en el backend.			

	5. Implementar envío de respuesta con el resultado de la consulta en el Endpoint. 6. Validar el correcto funcionamiento del endpoint. 7. Configurar la solicitud HTTP para la creación del usuario desde el frontend. 8. Manejar la respuesta en el frontend (200: Agregar síntoma, Error: Crear alerta notificando el tipo de error)			
HU-0035	**Nombre de la Historia:** Administración de usuarios. **Como:** Un desarrollador **Quiero:** Poder gestionar los permisos y estados de los usuarios del sistema. **Para:** Tener control sobre los usuarios del sistema. **Criterios de Aceptación:** ● Se debe permitir asignar y revocar roles de usuario (administrador, usuario estándar, etc.). ● Se debe poder activar y desactivar cuentas de usuarios. ● Se debe permitir actualizar los permisos de los usuarios (lectura, escritura, edición, eliminación). ● Los cambios en los permisos y estados de los usuarios deben reflejarse en tiempo real. ● Se debe registrar un historial de cambios en los permisos y estados de los usuarios.	**Alta**	**5**	**En Progreso**

	● La interfaz debe ser clara y permitir realizar estas acciones de forma sencilla. ● Los usuarios sin los permisos adecuados no deben poder acceder a funcionalidades restringidas. ● Los roles y permisos deben ser almacenados de forma segura y poder ser consultados eficientemente. **Definición de Terminado:** ● La funcionalidad de gestión de usuarios debe estar implementada en el backend. ● Se deben realizar pruebas unitarias y de integración que verifiquen la correcta asignación de permisos y estados. ● La interfaz gráfica debe permitir gestionar usuarios de forma intuitiva. ● La documentación técnica debe estar actualizada y detallar el proceso de gestión de usuarios. ● Se ha validado que las modificaciones en permisos y estados funcionan correctamente en todos los módulos del sistema. ● La historia ha sido revisada por el equipo y aprobada en la revisión del sprint. **Tareas a realizar:**			

	1. Implementar la vista de administración de usuarios. 2. Utilizar endpoint de ver cuenta de usuario para obtener los datos de los usuarios. 3. Desplegar los datos de los usuarios en una tabla. 4. Utilizar endpoint de borrar cuenta de usuario para eliminar lógicamente el registro de un usuario en el icono de eliminar (Icono de basurero). 5. Crear Endpoint para la modificación del usuario en el backend. 6. Validar el correcto funcionamiento del Endpoint. 7. Implementar formulario de modificación de usuario, se debe acceder al apretar el icono de modificación (Icono de lapiz). (Es el mismo del registro de usuario) 8. Implementar consulta HTTP al endpoint de actualización cuenta de usuario para poder modificar el registro de un usuario al enviar el formulario de modificación de usuario. 9. Crear Endpoint para la modificación del rol del usuario en el backend. 10. Implementar consulta HTTP al endpoint de modificación del rol de usuario para poder modificar el rol de un usuario en un combobox dentro de la tabla de usuarios.			

	11. Crear Endpoint para la obtención de los datos de un usuario en particular en el backend. 12. Implementar consulta HTTP al endpoint de obtención de datos de un usuario para poder visualizar el registro de un único usuario.			