

Actividad 6 – Entrega final

Camilo Andrés Ramírez Muñoz

Carolina Flórez Mesa


Mateo Grajales Jaramillo

Ingeniería de software – Corporación Universitaria Iberoamericana

Proyecto de software

José Castro

Diciembre 2023


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

DESCRIPCIÓN DEL PROBLEMA

En la actualidad, la gestión eficiente de los registros de pacientes, la asignación de citas médicas y la prescripción de medicamentos en nuestra empresa prestadora de salud se enfrenta a una serie de desafíos y limitaciones que impactan directamente en la calidad de los servicios ofrecidos y en la experiencia de los pacientes. Estos desafíos incluyen:


- **Registro de Pacientes Descentralizado:** Actualmente, la información de los pacientes se encuentra dispersa en múltiples sistemas y formatos, lo que dificulta la obtención de un historial médico completo y preciso. Esto puede llevar a errores en la atención médica y a retrasos en la toma de decisiones clínicas.
- **Asignación Manual de Citas:** El proceso de asignación de citas médicas se realiza de manera manual, lo que genera ineficiencias, conflictos de horarios y largos tiempos de espera para los pacientes. Esto afecta la satisfacción del paciente y la productividad del personal.
- **Prescripción de Medicamentos en Papel:** La prescripción de medicamentos se realiza en papel, lo que aumenta el riesgo de errores en la interpretación de las recetas y dificulta la coordinación con las farmacias. Además, no se cuenta con un seguimiento efectivo de los tratamientos.
- **Falta de Visualización Geográfica:** La empresa no dispone de una herramienta que permita a los pacientes y al personal ubicar de manera fácil y precisa la ubicación de los diversos consultorios y centros médicos. Esto puede causar confusión y retrasos en las visitas programadas.

Estos problemas afectan la calidad de la atención médica que brindamos a nuestros pacientes, generan ineficiencias en la operación del hospital y pueden llevar a situaciones críticas en la

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

atención médica. Para abordar estos desafíos de manera efectiva y mejorar la calidad de nuestros servicios, proponemos la creación de un sistema de manejo utilizando Spring Boot y servicios REST, junto con un front en Angular. Este sistema automatizará y optimizará la administración de registros de pacientes, asignación de citas y prescripción de medicamentos, y como futura mejora, permitirá la integración con Google Maps para una mejor ubicación geográfica de nuestros consultorios.

La implementación de este sistema resolverá los problemas mencionados y permitirá una atención médica más eficiente, precisa y satisfactoria para nuestros pacientes, al tiempo que mejorará la gestión interna del hospital y la comunicación con los pacientes.


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

ALCANCE DEL PROYECTO

El proyecto tiene como objetivo principal diseñar, desarrollar e implementar un sistema de manejo completo que aborde los desafíos identificados en la administración de registros de pacientes, asignación de citas médicas y prescripción de medicamentos en nuestra empresa prestadora de salud. El alcance del proyecto se define de la siguiente manera:

Funcionalidades clave

- **Registro de Pacientes:** El sistema permitirá la creación, actualización y gestión centralizada de registros de pacientes, incluyendo información médica relevante, historiales y datos de contacto.
- **Asignación de Citas Médicas:** Se desarrollará un módulo para la asignación automatizada de citas médicas, teniendo en cuenta la disponibilidad de los médicos y las preferencias de los pacientes.
- **Prescripción de Medicamentos:** Se implementará un sistema de prescripción electrónica de medicamentos que facilite la generación de recetas médicas, su seguimiento y la coordinación con las farmacias.
- **Integración con Google Maps (Futuro Desarrollo):** Como opción de desarrollo futuro, se permitirá la integración con Google Maps para mostrar la ubicación geográfica de los diversos consultorios y centros médicos de la empresa, facilitando la navegación para los pacientes.
- **Interfaz de Usuario Frontend:** Se creará una interfaz de usuario amigable utilizando Angular para que el personal médico y administrativo pueda interactuar con el sistema de manera intuitiva.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01


- **Seguridad y Acceso Controlado:** Se implementarán medidas de seguridad para proteger la información sensible de los pacientes, incluyendo autenticación de usuarios y control de acceso basado en roles.
- **Generación de Reportes:** Se desarrollará la capacidad de generar informes y estadísticas sobre la gestión de pacientes, citas médicas y prescripciones de medicamentos para facilitar la toma de decisiones y el seguimiento de la atención médica.

Exclusiones del Alcance

- El proyecto no incluye la adquisición de hardware adicional, como servidores o dispositivos móviles.
- No se incluye la integración completa con Google Maps en esta fase inicial, ya que se considera una mejora futura.

Criterios de Aceptación del Proyecto

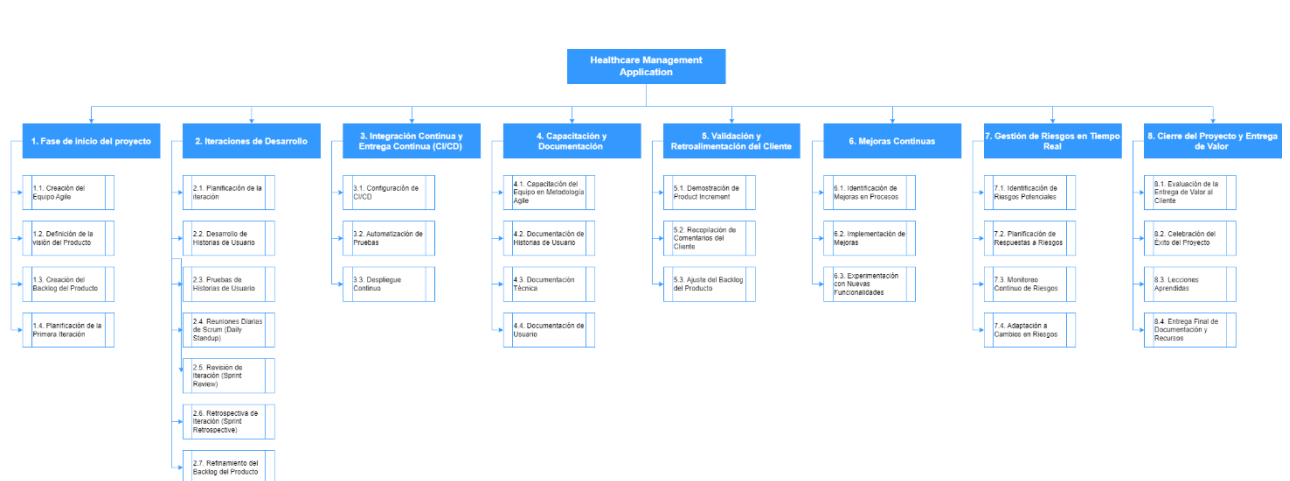
- Todas las funcionalidades mencionadas se implementarán de manera satisfactoria y serán probadas exhaustivamente.
- El sistema estará en producción y en uso por el personal médico y administrativo.
- Se proporcionará capacitación al personal para la correcta utilización del sistema.
- Se entregará documentación técnica y de usuario.
- Se establecerá un proceso de soporte y mantenimiento continuo para asegurar la estabilidad y la evolución del sistema.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Este alcance del proyecto define las funcionalidades clave que se deben abordar y los criterios para considerar el proyecto como exitoso. Cualquier aspecto adicional o cambios en el alcance deben ser gestionados a través de un proceso de control de cambios.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

ESTRUCTURA DEL DESGLOSE (EDT)



<https://laiberocol->

my.sharepoint.com/:i:/g/personal/cramir75_ibercoeduco/EZFslz_E2FpLiYKrC-

[X5eVIBwVb7HheA3o45JK7LArAXIA?e=dCJ56n](https://my.sharepoint.com/:i:/g/personal/cramir75_ibercoeduco/EZFslz_E2FpLiYKrC-X5eVIBwVb7HheA3o45JK7LArAXIA?e=dCJ56n)

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

INTRODUCCIÓN

En un contexto de atención médica en constante transformación, la gestión eficiente de registros de pacientes, la asignación de citas médicas y la prescripción de medicamentos se han convertido en elementos cruciales para elevar la calidad de la atención y mejorar la experiencia del paciente. En línea con nuestro compromiso inquebrantable de alcanzar estándares superiores en la atención médica y de proporcionar soluciones innovadoras, hemos emprendido un proyecto estratégico destinado a transformar profundamente la operación de nuestra empresa prestadora de servicios de salud.

El objetivo central de este proyecto es concebir y poner en marcha un sistema de gestión integral que aborde de manera efectiva los desafíos previamente mencionados. Este sistema no solo se propone optimizar nuestra eficiencia operativa, sino también fortalecer la relación con nuestros pacientes, brindándoles una atención más precisa y centrada en sus necesidades individuales.

A lo largo de las secciones que siguen, exploraremos en detalle los objetivos específicos que respaldan esta iniciativa y describiremos con profundidad cómo planeamos alcanzarlos. Asimismo, analizaremos la importancia y la justificación de este proyecto, subrayando cómo contribuirá al éxito continuo de nuestra empresa y al bienestar tanto de nuestros pacientes como de nuestros principales stakeholders.

Este proyecto representa un paso de gran trascendencia en nuestra búsqueda constante de la mejora de los servicios de atención médica y la entrega de resultados de la más alta calidad. Estamos decididos a forjar un sistema que no solo afronte los desafíos actuales, sino que también sienta las bases para un futuro de excelencia en el ámbito de la atención médica.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

OBJETIVOS

Objetivo General

Diseñar, desarrollar e implementar un sistema ágil e integrado de gestión que optimice la administración de registros de pacientes, agilice la asignación de citas médicas y mejore la prescripción de medicamentos en nuestra empresa prestadora de salud. Este sistema se orienta a elevar la calidad de la atención médica y la experiencia del paciente, asegurando un flujo eficiente y seguro de información clínica, así como una atención más efectiva y personalizada.

Objetivos Específicos


1. Automatizar la Gestión de Pacientes:

Desarrollar una funcionalidad robusta para la creación, actualización y gestión centralizada de registros de pacientes, garantizando la integridad y seguridad de los datos mediante medidas de encriptación y copias de seguridad periódicas.

2. Optimizar la Asignación de Citas Médicas:

Implementar un sistema de asignación automatizada de citas médicas que considere la disponibilidad de los médicos y las preferencias de los pacientes, reduciendo los tiempos de espera y conflictos de horarios a través de algoritmos de asignación eficientes.

3. Facilitar la Prescripción Electrónica de Medicamentos:

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Crear un módulo de prescripción electrónica de medicamentos que permita a los médicos generar recetas de manera eficiente, asegurando la legibilidad y el seguimiento efectivo de los tratamientos, integrando bases de datos actualizadas de medicamentos.

4. Diseñar una Interfaz de Usuario Intuitiva:

Diseñar una interfaz de usuario atractiva y fácil de usar utilizando Angular, centrada en la usabilidad y la accesibilidad para brindar una experiencia fluida tanto para el personal médico como administrativo, considerando sus retroalimentaciones.

5. Garantizar la Seguridad y Privacidad de los Datos:


Implementar medidas de seguridad sólidas para proteger la información sensible de los pacientes, incluyendo autenticación de dos factores, control de acceso basado en roles y cifrado de datos sensibles.

6. Generar Información y Estadísticas Relevantes:

Desarrollar la capacidad de generar informes y estadísticas sobre la gestión de pacientes, citas médicas y prescripciones de medicamentos para facilitar la toma de decisiones y la mejora continua, asegurando la precisión y relevancia de la información.

7. Preparar para la Integración con Google Maps (Opción Futura):

Investigar y planificar la posible integración con Google Maps para proporcionar a pacientes y personal una forma eficiente de ubicar los consultorios y centros médicos de la empresa, evaluando los beneficios y la viabilidad de esta integración.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

8. Capacitar al Personal y Documentar el Sistema:


Proporcionar capacitación adecuada al personal para utilizar el sistema eficazmente y generar documentación técnica y de usuario completa para una comprensión y uso efectivos del sistema, facilitando su adopción y correcto manejo.

9. Validar y Ajustar Continuamente:

Obtener retroalimentación del cliente y del equipo durante cada iteración para ajustar y mejorar el sistema de manera continua, experimentando con nuevas funcionalidades y mejoras según sea necesario para adaptarse a las necesidades cambiantes.

10. Lograr un Cierre Exitoso del Proyecto:


Evaluar la entrega de valor al cliente y celebrar el éxito del proyecto, documentando lecciones aprendidas y asegurando una entrega final completa de documentación y recursos para garantizar la continuidad operativa y futuras optimizaciones.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

JUSTIFICACIÓN

Este proyecto reviste una importancia fundamental para elevar tanto la calidad como la eficiencia de los servicios de atención médica que ofrecemos como empresa proveedora de salud. Nos encontramos ante desafíos sustanciales en la gestión de registros de pacientes, asignación de citas médicas y prescripción de medicamentos, aspectos que inciden directamente en la experiencia de nuestros pacientes y en la operatividad eficaz. La justificación de este proyecto se fundamenta en los siguientes beneficios inmediatos y a largo plazo:

1. **Mejora Sustancial de la Calidad de la Atención:** La implementación de este sistema posibilitará a nuestros profesionales de la salud acceder a información precisa y actualizada de los pacientes, lo que redundará en una atención médica más fundamentada y efectiva.
2. **Reducción Significativa de Errores:** La prescripción electrónica de medicamentos reducirá drásticamente los errores en la interpretación de recetas, mejorando sustancialmente la seguridad del paciente y la coordinación con las farmacias.
3. **Optimización Eficiente de Recursos:** La automatización en la asignación de citas disminuirá los tiempos de espera y potenciará la eficiencia operativa al eliminar duplicaciones innecesarias de tareas.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

4. **Incremento de la Satisfacción del Paciente:** Al proporcionar una experiencia más eficiente y centrada en el paciente, incrementaremos significativamente la satisfacción del paciente y su confianza en nuestros servicios.

Estos beneficios representan no solo mejoras inmediatas, sino que establecen los cimientos para un futuro de atención médica óptima, precisa y satisfactoria. Estamos comprometidos a abordar estos desafíos de manera estratégica para asegurar que nuestros pacientes reciban la atención que merecen y que nuestra empresa opere de manera eficiente y eficaz en este entorno sanitario dinámico.

Alcance a Corto Plazo (3-4 meses)

Durante los próximos 3 a 4 meses, nos enfocaremos en:

- **Desarrollo del Sistema Básico:** Implementar las funcionalidades esenciales, incluyendo el registro de pacientes, la asignación de citas médicas y la prescripción electrónica de medicamentos.
- **Desarrollo de una Interfaz de Usuario Funcional:** Crear una interfaz de usuario mínima pero funcional para permitir que el personal médico y administrativo comience a utilizar el sistema.
- **Seguridad y Privacidad Iniciales:** Establecer medidas de seguridad y control de acceso básicos.
- **Pruebas y Validación:** Realizar pruebas rigurosas para garantizar el funcionamiento correcto del sistema.
- **Capacitación Inicial:** Proporcionar capacitación inicial al personal para utilizar el sistema de manera eficaz.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Alcance a Medio Plazo (5-10 meses)

En los próximos 5 a 10 meses, ampliaremos el proyecto para:

- **Integración con Google Maps:** Investigar, planificar y posiblemente implementar la integración con Google Maps.
- **Mejoras en la Interfaz de Usuario:** Basadas en la retroalimentación del usuario, mejoraremos la interfaz de usuario para una experiencia más intuitiva.
- **Expansión de Funcionalidades:** Agregar nuevas características que mejoren aún más la gestión de la atención médica y la eficiencia operativa.


Alcance a Largo Plazo (12+ meses)

A largo plazo, continuaremos evolucionando el sistema:

- **Mejoras Continuas:** Identificar oportunidades de mejora en procesos y funcionalidades y realizar ajustes en consecuencia.
- **Investigación y Desarrollo:** Mantenernos al tanto de las tendencias tecnológicas y las necesidades cambiantes del sector de la salud para garantizar la relevancia continua del sistema.
- **Expansión y Adaptación:** Estar preparados para adaptarnos a los cambios en el entorno de atención médica y expandir el sistema según sea necesario.
- **Soporte y Mantenimiento Permanente:** Proporcionar soporte y mantenimiento continuo para garantizar la estabilidad y disponibilidad del sistema.

Respuesta a los Stakeholders sobre la necesidad del proyecto

En pos de potenciar nuestra operatividad y elevar la calidad asistencial para nuestros pacientes, nos complace presentar una solución innovadora que transformará la gestión de atención

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

médica. Este proyecto se centra en optimizar procesos cruciales, como el registro de pacientes, asignación de citas y prescripción de medicamentos, para adaptarnos al dinámico entorno de la atención médica actual.

Al buscar mejorar la eficiencia operativa y facilitar un acceso más efectivo a nuestros servicios, esta solución no solo fortalecerá la interacción entre nuestro personal médico y administrativo, sino que también garantizará la seguridad y privacidad de los datos sensibles. Representa un hito significativo en nuestra misión de mejora continua y promete un impacto positivo tanto en la experiencia del paciente como en la eficacia de nuestros servicios. Estamos ansiosos por los resultados que traerá y por cómo allanará el camino hacia un futuro donde la excelencia en la atención médica esté aún más al alcance.

Descripción de la Solución Propuesta:

Nuestra solución consiste en la implementación de un sistema de gestión de atención médica que hará que todo sea más eficiente y efectivo. Aquí hay una descripción sencilla de lo que este sistema hará por ustedes:


- **Registro de Pacientes Simplificado:** Con este sistema, registrar y actualizar la información de nuestros pacientes será más rápido y preciso. Tendrán acceso inmediato a su historial médico, lo que les permitirá brindar una atención más personalizada y de calidad.
- **Citas Médicas sin Esperas:** El sistema automatizará la asignación de citas médicas, lo que significa que los pacientes ya no tendrán que esperar largos periodos para obtener una cita. Esto mejorará la satisfacción del paciente y optimizará la agenda de sus médicos.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

- **Recetas Médicas Más Seguras:** La prescripción electrónica de medicamentos reducirá los errores en las recetas y garantizará que los pacientes reciban los tratamientos adecuados. Además, las farmacias podrán preparar los medicamentos de manera más eficiente.
- **Facilidad de Uso:** La interfaz de usuario del sistema ha sido diseñada pensando en la simplicidad y la facilidad de uso. El personal médico y administrativo no necesitará una capacitación extensa para utilizarlo de manera efectiva.
- **Seguridad de Datos:** Se implementarán medidas de seguridad sólidas para proteger la información de los pacientes. Esto garantiza que los datos médicos estén seguros y accesibles solo para personal autorizado.


Beneficios Clave para la Empresa:

1. **Optimización de la Eficiencia:** Al mejorar nuestros procesos, lograremos una utilización más efectiva de los recursos, resultando en ahorro de tiempo y costos operativos.
2. **Experiencia Mejorada para el Paciente:** Los pacientes experimentarán una atención más rápida y personalizada, lo que elevará su satisfacción y fomentará su lealtad hacia nuestra empresa.
3. **Reducción de Errores:** Al disminuir los errores en las recetas y en la gestión de citas, garantizaremos una atención más segura y confiable para nuestros valiosos pacientes.
4. **Mayor Competitividad mediante Tecnología Innovadora:** La adopción de tecnología de vanguardia nos mantendrá relevantes y competitivos en el mercado de la atención médica, respaldando así nuestro posicionamiento en la industria.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01


Estamos comprometidos con la implementación exitosa de esta solución y esperamos que esta mejora transforme su empresa prestadora de salud para mejor. Estamos seguros de que estos cambios repercutirán positivamente en los pacientes y en el éxito continuo de su organización.

Si tienen alguna pregunta o inquietud, no duden en comunicarse con nosotros.


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

MATRIZ DE RIESGOS

Riesgo	Probabilidad	Impacto	Descripción	Estrategia de Mitigación
Cambios en los Requisitos	Alta	Alto	Cambios significativos en los requisitos por parte de los stakeholders pueden retrasar el proyecto.	Realizar una comunicación constante con los stakeholders y documentar los requisitos.
Problemas en la Adopción del Sistema	Mediana	Alto	La resistencia al cambio entre el personal médico podría afectar la implementación.	Proporcionar capacitación y apoyo continuo, involucrar al personal en el proceso.
Fallos en la Seguridad del Sistema	Baja	Alto	Vulnerabilidades de seguridad podrían poner en riesgo la integridad de los datos de los pacientes.	Implementar medidas de seguridad robustas y realizar auditorías de seguridad.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Problemas en la Integración con Google Maps	Mediana	Mediano	Dificultades técnicas pueden surgir al integrar con Google Maps para mostrar ubicaciones.	Realizar pruebas exhaustivas de la integración y tener un plan de respaldo.
Falta de Recursos	Mediana	Mediano	La escasez de personal o recursos técnicos podría afectar la ejecución del proyecto.	Realizar una planificación de recursos cuidadosa y tener un equipo de respaldo disponible.
Cambios en la Regulación de Salud	Baja	Mediano	Cambios en las regulaciones de salud pueden requerir ajustes en el sistema.	Mantenerse al tanto de las regulaciones y estar preparado para adaptarse.
Interrupciones en el Proveedor de Servicios en la Nube	Baja	Alto	Si se utiliza un proveedor de servicios en la nube,	Tener un plan de contingencia y respaldo en

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

			interrupciones en su servicio pueden afectar la disponibilidad del sistema.	caso de interrupciones.
Fallos en el Hardware o Software Subyacente	Mediana	Mediano	Problemas en la infraestructura subyacente pueden afectar la operatividad del sistema.	Realizar un monitoreo continuo de la infraestructura y tener acuerdos de soporte.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

CRONOGRAMA

Fase de Inicio del Proyecto

- Semana 1-2: Creación del Equipo Agile
 - Selección y Capacitación del Equipo Agile
 - Designación de Roles y Responsabilidades
- Semana 3-4: Definición de la Visión del Producto
 - Creación de una Declaración de Visión del Producto
 - Identificación de Metas y Objetivos Clave
- Semana 5-6: Creación del Backlog del Producto
 - Reuniones de Brainstorming para Identificar Funcionalidades
 - Priorización de Elementos en el Backlog
- Semana 7-8: Planificación de la Primera Iteración
 - Seleccionar Historias de Usuario para la Primera Iteración
 - Definir Criterios de Aceptación

Iteraciones de Desarrollo (Iteración de 2 Semanas Cada Una)

- Iteración 1-2
 - Planificación de Iteración
 - Desarrollo de Historias de Usuario
 - Pruebas de Historias de Usuario
 - Reuniones Diarias de Scrum (Daily Standup)
- Iteración 3-4
 - Revisión de Iteración (Sprint Review)
 - Retrospectiva de Iteración (Sprint Retrospective)

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01


- Refinamiento del Backlog del Producto
- Planificación de la Siguiente Iteración
- Iteración 5-6
 - Desarrollo y Pruebas Continuas
 - Integración Continua y Entrega Continua (CI/CD)
 - Refinamiento del Backlog del Producto

Capacitación y Documentación

- Semana 17-18: Capacitación del Equipo en Metodología Agile
 - Sesiones de Capacitación en Agile
 - Entrenamiento en Herramientas y Prácticas Agile
- Semana 19-20: Documentación de Historias de Usuario
 - Creación de Documentación Detallada para Historias de Usuario
 - Generación de Documentación Técnica
- Semana 21-22: Documentación de Usuario
 - Creación de Manuales de Usuario
 - Preparación de Recursos de Soporte

Validación y Retroalimentación del Cliente

- Semana 23-24: Demostración de Product Increment
 - Presentación de las Funcionalidades Implementadas
 - Recopilación de Comentarios del Cliente
- Semana 25-26: Ajuste del Backlog del Producto
 - Priorización de Nuevas Funcionalidades
 - Modificación del Backlog para Futuras Iteraciones

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Mejoras Continuas


- Semana 27-28: Identificación de Mejoras en Procesos
 - Evaluación de la Eficiencia de las Prácticas Agile
 - Identificación de Oportunidades de Mejora
- Semana 29-30: Implementación de Mejoras
 - Implementación de Cambios en los Procesos
 - Experimentación con Nuevas Prácticas Agile

Gestión de Riesgos en Tiempo Real

- Ongoing: Identificación y Monitoreo Continuo de Riesgos
 - Identificación de Riesgos Potenciales
 - Planificación de Respuestas a Riesgos
 - Monitoreo Continuo de Riesgos
 - Adaptación a Cambios en Riesgos

Cierre del Proyecto y Entrega de Valor

- Semana 40-42: Evaluación de la Entrega de Valor al Cliente
 - Evaluación de Resultados del Proyecto
 - Comparación con Objetivos Iniciales
- Semana 43: Celebración del Éxito del Proyecto
 - Reconocimiento del Equipo por los Logros
 - Agradecimiento a los Stakeholders
- Semana 44-45: Lecciones Aprendidas y Entrega Final de Documentación y Recursos
 - Identificación de Lecciones Aprendidas

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

- Entrega de Documentación Final
- Cierre Formal del Proyecto

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

PRESUPUESTO


Tasa de Cambio Estimada: 1 USD = 3,900 COP

Fase de Inicio del Proyecto

- Creación del Equipo Agile: \$3,000 - \$5,000 (total para el equipo)
 - En pesos colombianos: 11,700,000 - 19,500,000 COP (total para el equipo)
- Definición de la Visión del Producto: \$1,000 - \$1,500
 - En pesos colombianos: 3,900,000 - 5,850,000 COP
- Creación del Backlog del Producto: \$800 - \$1,200
 - En pesos colombianos: 3,120,000 - 4,680,000 COP
- Planificación de la Primera Iteración: \$500 - \$800
 - En pesos colombianos: 1,950,000 - 3,120,000 COP

Iteraciones de Desarrollo (Iteración de 2 Semanas Cada Una)

- Costos de Personal de Desarrollo y Pruebas: \$3,000 - \$5,000 (por cada iteración de 2 semanas, total para el equipo)
 - En pesos colombianos: 11,700,000 - 19,500,000 COP (por cada iteración de 2 semanas, total para el equipo)
- Costos de Reuniones Diarias de Scrum: \$200 - \$400 (por cada iteración de 2 semanas)
 - En pesos colombianos: 780,000 - 1,560,000 COP (por cada iteración de 2 semanas)
- Refinamiento del Backlog del Producto: \$500 - \$800 (por cada iteración de 2 semanas)
 - En pesos colombianos: 1,950,000 - 3,120,000 COP (por cada iteración de 2 semanas)

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Capacitación y Documentación

- Capacitación del Equipo en Metodología Agile: \$800 - \$1,500
 - En pesos colombianos: 3,120,000 - 5,850,000 COP
- Documentación de Historias de Usuario: \$500 - \$800
 - En pesos colombianos: 1,950,000 - 3,120,000 COP

Validación y Retroalimentación del Cliente

- Demostración de Product Increment: \$500 - \$800
 - En pesos colombianos: 1,950,000 - 3,120,000 COP
- Ajuste del Backlog del Producto: \$200 - \$400
 - En pesos colombianos: 780,000 - 1,560,000 COP

Mejoras Continuas


- Identificación de Mejoras en Procesos: \$500 - \$800
 - En pesos colombianos: 1,950,000 - 3,120,000 COP
- Implementación de Mejoras: \$300 - \$600
 - En pesos colombianos: 1,170,000 - 2,340,000 COP

Gestión de Riesgos en Tiempo Real

- Identificación y Monitoreo Continuo de Riesgos: \$200 - \$400
 - En pesos colombianos: 780,000 - 1,560,000 COP

Cierre del Proyecto y Entrega de Valor


- Evaluación de la Entrega de Valor al Cliente: \$500 - \$800
 - En pesos colombianos: 1,950,000 - 3,120,000 COP

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

- Celebración del Éxito del Proyecto: \$300 - \$600
 - En pesos colombianos: 1,170,000 - 2,340,000 COP
- Lecciones Aprendidas y Entrega Final de Documentación y Recursos: \$200 - \$400
 - En pesos colombianos: 780,000 - 1,560,000 COP

Total Estimado del Presupuesto en Dólares: \$13,000 - \$21,800 (para todo el proyecto)

Total Estimado del Presupuesto en Pesos Colombianos: 50,700,000 - 85,020,000 COP (para todo el proyecto)

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

PROTOTIPO

Este apartado proporciona una descripción detallada del prototipo que se está desarrollando para nuestro proyecto. El prototipo es una representación temprana de nuestro producto final y está diseñado para demostrar sus características y funcionalidades clave.


El prototipo contiene una versión preliminar del código que se utilizará en el producto final, tenga en cuenta que al ser un prototipo algunas funcionalidades pueden no estar incluidas. El código utilizado en este prototipo es de una etapa temprana y usado como una vista previa, el código del producto final será pulido siguiendo las mejores prácticas de la industria y será revisado rigurosamente para asegurar su calidad.

El prototipo está diseñado para realizar una breve demostración de las características del software. Esto incluye una vista previa de la interfaz de usuario, la funcionalidad del sistema y la interacción del usuario.

El apartado de Google Maps, como se especificó en las páginas anteriores, es una implementación planeada a futuro. Por lo tanto, en este prototipo no se encuentra funcional.

Proceso de Instalación del Prototipo

Es esencial que su estación de trabajo tenga instalado y en funcionamiento el sistema de gestión de bases de datos MySQL, operando específicamente en el puerto 3306. Se requiere la creación previa de una base de datos con las siguientes especificaciones:

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Nombre de la base de datos: healthcare_db

Usuario: root

Contraseña del usuario: 12345678

Además, es necesario instalar Node.js en su equipo. Para ello, puede seguir las instrucciones proporcionadas en el sitio web oficial de Node.js (Introduction to Node.js, s.f.).

Asimismo, se requiere la instalación de React en su equipo. Puede seguir las instrucciones proporcionadas en el sitio web oficial de React para este propósito (React, s.f.).

El back-end del proyecto, denominado healthcare.zip, debe ser descargado desde el siguiente repositorio: <https://github.com/camilodvl/proyecto>

Para el desarrollo de este proyecto, recomendamos el uso del entorno de desarrollo integrado (IDE) IntelliJ. Una vez descargado el archivo healthcare.zip, extraiga todos los archivos y abra el proyecto desde IntelliJ, el puerto 8080 debe estar disponible en su estación de trabajo.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Finalmente, ejecute el proyecto desde IntelliJ para iniciar el servidor de back-end.

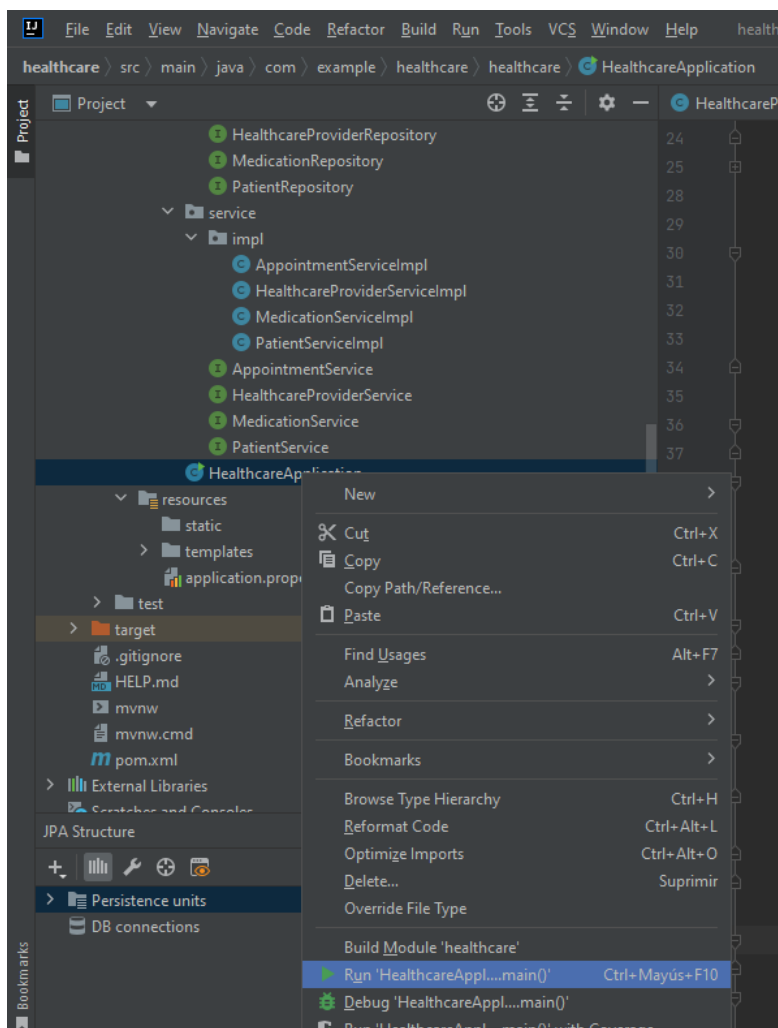



Ilustración 1 Captura de pantalla IntelliJ proyecto

Para la implementación del interfaz de usuario, es imperativo que se tenga instalado el sistema de control de versiones Git en su estación de trabajo. A continuación, se debe proceder a clonar el repositorio del proyecto utilizando el siguiente comando:

```
git clone https://github.com/camilodvl/proyecto-front
```

Este comando descargará una copia del repositorio especificado en el directorio actual de trabajo.

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

Posteriormente, es necesario instalar las dependencias del proyecto. Esto se puede lograr ejecutando el siguiente comando en la raíz del directorio del proyecto:

```
npm install
```

Este comando instalará todas las dependencias necesarias para el proyecto, tal como se especifica en el archivo package.json. Una vez que todas las dependencias se hayan instalado correctamente, se puede iniciar la aplicación de interfaz de usuario con el siguiente comando:

```
npm start
```

Este comando iniciará el servidor de desarrollo y abrirá automáticamente la aplicación en el navegador predeterminado. En este punto, debería poder ver la página principal del proyecto cargada en su navegador. Tenga presente que el proyecto corre en el puerto 3000, por lo cual este debe estar disponible en su estación de trabajo.

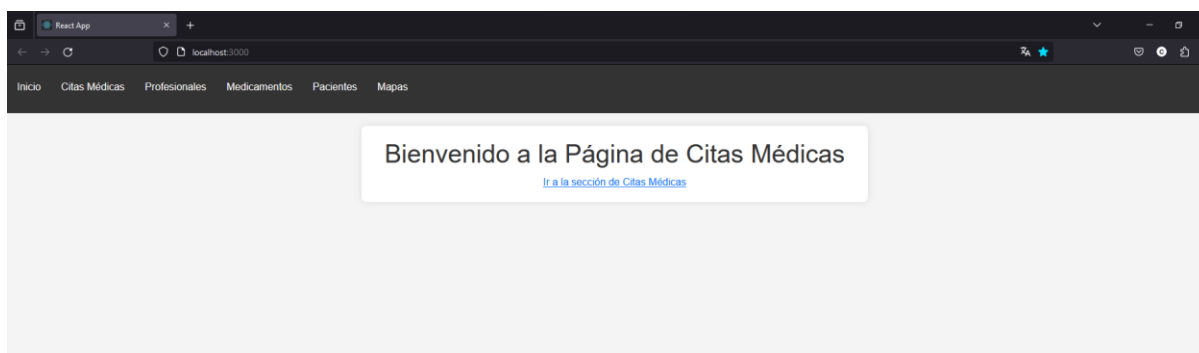


Ilustración 2 Captura de página inicial

Tenga en cuenta

Los modelos incorporados en el back-end del prototipo mantienen relaciones con otros modelos, lo que implica correspondientemente una relación en la base de datos. Por lo tanto, al realizar pruebas con el prototipo, es necesario crear registros en el sistema teniendo en cuenta las dependencias de cada uno. Se debe seguir el siguiente orden:

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

1. Paciente
2. Profesional
3. Citas

La introducción de registros en un orden diferente puede provocar fallos inesperados en el prototipo. Por ejemplo, para poder crear una cita, es necesario haber creado previamente un profesional, el cual será referenciado en la cita.

Es importante recordar que este es un prototipo y, como tal, puede presentar ciertas limitaciones. Estos fallos serán controlados en su totalidad en el producto final, donde se implementarán mecanismos de manejo de errores más robustos y completos.

A continuación, encontrará un video del prototipo con sus respectivas pruebas:

<https://youtu.be/aXdpHft8yHI>

Código

Se realizó el código del prototipo en java para el back-end y js para el front-end (React)


```
package com.example.healthcare.healthcare.model;
```

```
import java.util.Date;
```

```
import jakarta.persistence.Entity;
```

```
import jakarta.persistence.GeneratedValue;
```

```
import jakarta.persistence.GenerationType;
```


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import jakarta.persistence.Id;
```

```
import jakarta.persistence.JoinColumn;
```

```
import jakarta.persistence.ManyToOne;
```

```
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
```

```
@Entity
```

```
@Data
```

```
@NoArgsConstructor
```

```
public class Appointment {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private Date appointmentDate;
```


```
    // Define a relationship between Appointment and Patient
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "patient_id")
```

```
    private Patient patient;
```

```
    // Define a relationship between Appointment and HealthcareProvider
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

@ManyToOne

@JoinColumn(name = "provider_id")

private HealthcareProvider healthcareProvider;

// Add more fields as needed

}

package com.example.healthcare.healthcare.model;

import java.util.List;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.OneToMany;


import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Data

@NoArgsConstructor

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

public class HealthcareProvider {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;


    private String name;

    private String specialty;


    // Define a relationship between HealthcareProvider and Appointment

    // @OneToMany(mappedBy = "healthcareProvider")

    //private List<Appointment> appointments;


}

```

```

package com.example.healthcare.healthcare.model;

```

```

import jakarta.persistence.Entity;


import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import jakarta.persistence.ManyToOne;
```

```
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
```

```
@Entity
```

```
@Data
```

```
@NoArgsConstructor
```

```
public class Medication {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private String name;
```

```
    private String dosage;
```

```
    // Define a relationship between Medication and Patient
```


```
    @ManyToOne
```

```
    @JoinColumn(name = "patient_id")
```

```
    private Patient patient;
```

```
    // Add more fields as needed
```

```
}
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
package com.example.healthcare.healthcare.model;
```

```
import java.util.Date;
```

```
import jakarta.persistence.Entity;
```

```
import jakarta.persistence.GeneratedValue;
```

```
import jakarta.persistence.GenerationType;
```

```
import jakarta.persistence.Id;
```

```
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
```

```
@Entity
```

```
@Data
```

```
@NoArgsConstructor
```


```
public class Patient {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private String firstName;
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

    private String lastName;

    private Date dateOfBirth;

}

package com.example.healthcare.healthcare.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import com.example.healthcare.healthcare.model.Appointment;

import com.example.healthcare.healthcare.service.AppointmentService;

import java.util.List;


import java.util.Optional;

@RestController

@RequestMapping("/api/appointments")

public class AppointmentController {

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

@Autowired

@Qualifier("appointmentServiceImpl")

private AppointmentService appointmentService;

@CrossOrigin(origins = "http://localhost:3000")

@GetMapping

```
public List<Appointment> getAllAppointments() {
    return appointmentService.getAllAppointments();
}
```

@CrossOrigin(origins = "http://localhost:3000")

@GetMapping("/{id}")


```
public ResponseEntity<Appointment> getAppointmentById(@PathVariable Long id) {
    Optional<Appointment> appointment = appointmentService.getAppointmentById(id);
    return appointment.map(value -> new ResponseEntity<>(value, HttpStatus.OK))
        .orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT_FOUND));
}
```

@CrossOrigin(origins = "http://localhost:3000")

@PostMapping

```
public ResponseEntity<Appointment> createAppointment(@RequestBody Appointment
appointment) {
```

```
    Appointment createdAppointment = appointmentService.createAppointment(appointment);
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        return new ResponseEntity<>(createdAppointment, HttpStatus.CREATED);

    }

    @PutMapping("/{id}")

    public ResponseEntity<Appointment> updateAppointment(@PathVariable Long id, @RequestBody
Appointment appointment) {

        Appointment    updatedAppointment    =    appointmentService.updateAppointment(id,
appointment);

        if (updatedAppointment != null) {

            return new ResponseEntity<>(updatedAppointment, HttpStatus.OK);

        } else {

            return new ResponseEntity<>(HttpStatus.NOT_FOUND);

        }

    }

}

    @DeleteMapping("/{id}")

    public ResponseEntity<Void> deleteAppointment(@PathVariable Long id) {


        appointmentService.deleteAppointment(id);

        return new ResponseEntity<>(HttpStatus.NO_CONTENT);

    }

}

package com.example.healthcare.healthcare.controller;
```


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

@Controller

public class GoogleMapsController {

    @Value("${google.maps.api.key}")

    private String googleMapsApiKey;

    @GetMapping("/maps")

    public String showMap(Model model) {

        model.addAttribute("apiKey", googleMapsApiKey);

        return "maps"; // Return a Thymeleaf or HTML template for displaying the map


    }

}

package com.example.healthcare.healthcare.controller;

import org.springframework.beans.factory.annotation.Autowired;

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import com.example.healthcare.healthcare.model.HealthcareProvider;

import com.example.healthcare.healthcare.service.HealthcareProviderService;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/providers")

public class HealthcareProviderController {

    @Autowired

    @Qualifier("healthcareProviderServiceImpl")

    private HealthcareProviderService providerService;


    @CrossOrigin(origins = "http://localhost:3000")

    @GetMapping

    public List<HealthcareProvider> getAllProviders() {

        return providerService.getAllProviders();

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

}

@GetMapping("/{id}")

public ResponseEntity<HealthcareProvider> getProviderById(@PathVariable Long id) {

Optional<HealthcareProvider> provider = providerService.getProviderById(id);

return provider.map(value -> new ResponseEntity<>(value, HttpStatus.OK))

.orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT_FOUND));

}

@CrossOrigin(origins = "http://localhost:3000")

@PostMapping

public ResponseEntity<HealthcareProvider> createProvider(@RequestBody
HealthcareProvider provider) {

HealthcareProvider createdProvider = providerService.createProvider(provider);

return new ResponseEntity<>(createdProvider, HttpStatus.CREATED);

}


@PutMapping("/{id}")

public ResponseEntity<HealthcareProvider> updateProvider(@PathVariable Long id,
@RequestBody HealthcareProvider provider) {

HealthcareProvider updatedProvider = providerService.updateProvider(id, provider);

if (updatedProvider != null) {

return new ResponseEntity<>(updatedProvider, HttpStatus.OK);

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

    } else {

        return new ResponseEntity<>(HttpStatus.NOT_FOUND);

    }

}

@CrossOrigin(origins = "http://localhost:3000")

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteProvider(@PathVariable Long id) {

    providerService.deleteProvider(id);

    return new ResponseEntity<>(HttpStatus.NO_CONTENT);

}

}

package com.example.healthcare.healthcare.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.http.HttpStatus;


import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import com.example.healthcare.healthcare.model.Medication;

import com.example.healthcare.healthcare.service.MedicationService;

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import java.util.List;
```

```
import java.util.Optional;
```

```
@RestController
```

```
@RequestMapping("/api/medications")
```

```
public class MedicationController {
```

```
    @Autowired
```

```
    @Qualifier("medicationServiceImpl")
```

```
    private MedicationService medicationService;
```

```
    @CrossOrigin(origins = "http://localhost:3000")
```

```
    @GetMapping
```

```
    public List<Medication> getAllMedications() {
```

```
        return medicationService.getAllMedications();
```

```
    }
```


```
    @GetMapping("/{id}")
```

```
    public ResponseEntity<Medication> getMedicationById(@PathVariable Long id) {
```

```
        Optional<Medication> medication = medicationService.getMedicationById(id);
```

```
        return medication.map(value -> new ResponseEntity<>(value, HttpStatus.OK))
```

```
            .orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT_FOUND));
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

}

```
@CrossOrigin(origins = "http://localhost:3000")
```

```
@PostMapping
```

```
public ResponseEntity<Medication> createMedication(@RequestBody Medication medication)
{
    Medication createdMedication = medicationService.createMedication(medication);

    return new ResponseEntity<>(createdMedication, HttpStatus.CREATED);
}
```

```
@CrossOrigin(origins = "http://localhost:3000")
```

```
@PutMapping("/{id}")
```

```
public ResponseEntity<Medication> updateMedication(@PathVariable Long id, @RequestBody
Medication medication) {

    Medication updatedMedication = medicationService.updateMedication(id, medication);

    if (updatedMedication != null) {


        return new ResponseEntity<>(updatedMedication, HttpStatus.OK);

    } else {

        return new ResponseEntity<>(HttpStatus.NOT_FOUND);

    }
}
```

```
@CrossOrigin(origins = "http://localhost:3000")
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteMedication(@PathVariable Long id) {

    medicationService.deleteMedication(id);

    return new ResponseEntity<>(HttpStatus.NO_CONTENT);

}

}

```

```

package com.example.healthcare.healthcare.controller;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.example.healthcare.healthcare.model.Patient;
import com.example.healthcare.healthcare.service.PatientService;

import java.util.List;
import java.util.Optional;

@RestController

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
@RequestMapping("/api/patients")
```

```
public class PatientController {
```

```
    @Autowired
```

```
    @Qualifier("patientServiceImpl")
```

```
    private PatientService patientService;
```

```
    @CrossOrigin(origins = "http://localhost:3000")
```

```
    @GetMapping
```

```
    public List<Patient> getAllPatients() {
```

```
        return patientService.getAllPatients();
```

```
    }
```

```
    @GetMapping("/{id}")
```

```
    public ResponseEntity<Patient> getPatientById(@PathVariable Long id) {
```

```
        Optional<Patient> patient = patientService.getPatientById(id);
```

```
        return patient.map(value -> new ResponseEntity<>(value, HttpStatus.OK))
```

```
            .orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT_FOUND));
```


```
    }
```

```
    @CrossOrigin(origins = "http://localhost:3000")
```

```
    @PostMapping
```

```
    public ResponseEntity<Patient> createPatient(@RequestBody Patient patient) {
```

```
        Patient createdPatient = patientService.createPatient(patient);
```


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        return new ResponseEntity<>(createdPatient, HttpStatus.CREATED);

    }

    @PutMapping("/{id}")

    public ResponseEntity<Patient> updatePatient(@PathVariable Long id, @RequestBody Patient
patient) {

        Patient updatedPatient = patientService.updatePatient(id, patient);

        if (updatedPatient != null) {

            return new ResponseEntity<>(updatedPatient, HttpStatus.OK);

        } else {

            return new ResponseEntity<>(HttpStatus.NOT_FOUND);

        }

    }

    @DeleteMapping("/{id}")

    public ResponseEntity<Void> deletePatient(@PathVariable Long id) {


        patientService.deletePatient(id);

        return new ResponseEntity<>(HttpStatus.NO_CONTENT);

    }

}

package com.example.healthcare.healthcare.repository;
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import com.example.healthcare.healthcare.model.Appointment;
```

```
public interface AppointmentRepository extends JpaRepository<Appointment, Long> {

    // You can add custom query methods here if needed

}
```

```
package com.example.healthcare.healthcare.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
```


```
import com.example.healthcare.healthcare.model.HealthcareProvider;
```

```
public interface HealthcareProviderRepository extends JpaRepository<HealthcareProvider,
Long> {

    // You can add custom query methods here if needed

}
```

```
package com.example.healthcare.healthcare.repository;
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import com.example.healthcare.healthcare.model.Medication;
```

```
public interface MedicationRepository extends JpaRepository<Medication, Long> {
```

```
    // You can add custom query methods here if needed
```

```
}
```

```
package com.example.healthcare.healthcare.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import com.example.healthcare.healthcare.model.Patient;
```

```
public interface PatientRepository extends JpaRepository<Patient, Long> {
```


```
    // You can add custom query methods here if needed
```

```
}
```

```
package com.example.healthcare.healthcare.service;
```

```
import java.util.List;
```

```
import java.util.Optional;
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import com.example.healthcare.healthcare.model.Appointment;
```

```
public interface AppointmentService {
```

```
    List<Appointment> getAllAppointments();
```

```
    Optional<Appointment> getAppointmentById(Long id);
```

```
    Appointment createAppointment(Appointment appointment);
```

```
    Appointment updateAppointment(Long id, Appointment appointment);
```

```
    void deleteAppointment(Long id);
```


```
}
```

```
package com.example.healthcare.healthcare.service;
```

```
import java.util.List;
```

```
import java.util.Optional;
```

```
import com.example.healthcare.healthcare.model.HealthcareProvider;
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
public interface HealthcareProviderService {

    List<HealthcareProvider> getAllProviders();

    Optional<HealthcareProvider> getProviderById(Long id);

    HealthcareProvider createProvider(HealthcareProvider provider);

    HealthcareProvider updateProvider(Long id, HealthcareProvider provider);

    void deleteProvider(Long id);

}
```

```
package com.example.healthcare.healthcare.service;
```


```
import java.util.List;
```

```
import java.util.Optional;
```

```
import com.example.healthcare.healthcare.model.Medication;
```

```
public interface MedicationService {
```

```
    List<Medication> getAllMedications();
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
Optional<Medication> getMedicationById(Long id);
```

```
Medication createMedication(Medication medication);
```

```
Medication updateMedication(Long id, Medication medication);
```

```
void deleteMedication(Long id);
```

```
}
```

```
package com.example.healthcare.healthcare.service;
```

```
import java.util.List;
```


```
import java.util.Optional;
```

```
import com.example.healthcare.healthcare.model.Patient;
```

```
public interface PatientService {
```

```
    List<Patient> getAllPatients();
```

```
    Optional<Patient> getPatientById(Long id);
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
Patient createPatient(Patient patient);
```

```
Patient updatePatient(Long id, Patient patient);
```

```
void deletePatient(Long id);
```

```
}
```

```
package com.example.healthcare.healthcare;
```

```
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
```


```
public class HealthcareApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(HealthcareApplication.class, args);
```

```
    }
```

```
}
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import React, { useState } from 'react';
```

```
import axios from 'axios';
```

```
const AppointmentForm = () => {
```

```
  const [appointmentDate, setAppointmentDate] = useState('');
```

```
  const [patientId, setPatientId] = useState('');
```

```
  const [providerId, setProviderId] = useState('');
```

```
  const handleSubmit = (event) => {
```

```
    event.preventDefault();
```

```
    const appointment = { appointmentDate, patient: { id: patientId }, healthcareProvider:
    { id: providerId } };

```


```
    axios.post('http://localhost:8080/api/appointments', appointment)
```

```
      .then(response => {
```

```
        console.log(response.data);
```

```
        window.location.href = 'http://localhost:3000/appointments';
```

```
      })
```


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        .catch(error => {

            console.error('There was an error!', error);

        });

    };

    return (

        <div className='container'>

<h2>Crear cita</h2>

<form onSubmit={handleSubmit}>

    <div>

        <label htmlFor="appointmentDate">Fecha de la cita:</label>

        <input type="date" id="appointmentDate" value={appointmentDate} onChange={(event) =>
setAppointmentDate(event.target.value)} />

    </div>

    <div>

        <label htmlFor="patientId">Id del paciente:</label>

        <input type="text" id="patientId" value={patientId} onChange={(event) =>
setPatientId(event.target.value)} />

    </div>


    <div>

        <label htmlFor="providerId">Id del profesional:</label>

        <input type="text" id="providerId" value={providerId} onChange={(event) =>
setProviderId(event.target.value)} />

    </div>

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
        <button type="submit">Crear cita</button>

    </form>

</div>

);

};
```

```
export default AppointmentForm;
```

```
import React, { useState } from 'react';
```

```
import axios from 'axios';
```

```
function CreateMedication() {
```

```
    const [name, setName] = useState('');
```


```
    const [dosage, setDosage] = useState('');
```

```
    const handleNameChange = (event) => {
```

```
        setName(event.target.value);
```

```
    };
```

```
    const handleDosageChange = (event) => {
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

    setDosage(event.target.value);

  });

const handleSubmit = (event) => {

  event.preventDefault();

  const medication = { name, dosage };

  axios.post('http://localhost:8080/api/medications', medication)

    .then(response => {

      console.log(response.data);

      window.location.href = 'http://localhost:3000/medication';

    })

    .catch(error => {

      console.error('There was an error!', error);

    });

});

return (

  <div className='container'>


    <form onSubmit={handleSubmit}>

      <label>

        Nombre:

        <input type="text" value={name} onChange={handleNameChange} />

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

    </label>

    <label>

      Dosis:

      <input type="text" value={dosage} onChange={handleDosageChange} />

    </label>

    <button type="submit">Crear medicamento</button>

  </form>

</div>

);
}

```

```
export default CreateMedication;
```

```
import React, { useState } from 'react';
```

```
import axios from 'axios';
```

```
function CreatePatient() {
```


```
  const [patient, setPatient] = useState({
```

```
    firstName: '',
```

```
    lastName: '',
```

```
    dateOfBirth: ''
```

```
  });
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

const handleInputChange = (event) => {

  setPatient({

    ...patient,

    [event.target.name]: event.target.value

  });

};

const handleSubmit = (event) => {

  event.preventDefault();

  axios.post('http://localhost:8080/api/patients', patient)

    .then(response => {

      console.log(response.data);

      window.location.href = 'http://localhost:3000/patients';

    });

};


return (

  <div className='container'>

    <form onSubmit={handleSubmit}>

      <label htmlFor="firstName">Nombre:</label>

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

<input

  type="text"

  name="firstName"

  value={patient.firstName}

  onChange={handleInputChange}

/>

<label htmlFor="lastName">Apellido:</label>

<input

  type="text"

  name="lastName"

  value={patient.lastName}

  onChange={handleInputChange}

/>

<label htmlFor="dateOfBirth">Fecha de nacimiento:</label>

<input

  type="date"

  name="dateOfBirth"

  value={patient.dateOfBirth}


  onChange={handleInputChange}

/>

<button type="submit">Crear paciente</button>

</form>

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

</div>

);

}

export default CreatePatient;

import React, { useState } from 'react';

import axios from 'axios';

const HealthcareProviderForm = () => {

const [name, setName] = useState('');

const [specialty, setSpecialty] = useState('');

const handleSubmit = (event) => {

event.preventDefault();

const provider = { name, specialty };


axios.post('http://localhost:8080/api/providers', provider)

.then(response => {

console.log(response.data);

window.location.href = 'http://localhost:3000/providers';

}))

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        .catch(error => {

            console.error('There was an error!', error);

        });

    };

    return (

        <div className='container'>

            <form onSubmit={handleSubmit}>

                <div>

                    <label htmlFor="name">Nombre:</label>

                    <input      type="text"      id="name"      value={name}      onChange={(event)      =>
setName(event.target.value)} />

                </div>

                <div>

                    <label htmlFor="specialty">Especialidad:</label>

                    <input    type="text"    id="specialty"    value={specialty}    onChange={(event)    =>
setSpecialty(event.target.value)} />

                </div>

                <button type="submit">Crear Profesional</button>


            </form>

        </div>

    );

};

```


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
export default HealthcareProviderForm;
```

```
const Indexap =()=>{
```

```
  return(
```

```
    <div className="container">
```

```
      <h1>Bienvenido a la Página de Citas Médicas</h1>
```

```
      <a href="http://localhost:3000/appointments">Ir a la sección de Citas Médicas</a>
```

```
    </div>
```

```
  )
```


```
}
```

```
export default Indexap
```

```
import React, { useEffect, useState } from "react";
```

```
import axios from "axios";
```

```
import "../css/Appointments.css"
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

const AppointmentList = () => {

  const [appointments, setAppointments] = useState([]);

  useEffect(() => {

    axios

      .get("http://localhost:8080/api/appointments")

      .then((response) => {

        setAppointments(response.data);

      })

      .catch((error) => {

        console.error("There was an error!", error);

      });

  }, []);

  return (

    <div className="container">

      <h2>Lista de citas</h2>


      <table className="table">

        <thead>

          <tr>

            <th scope="col">Nombre Paciente</th>

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        <th scope="col">Fecha de la cita</th>

        <th scope="col">Profesional</th>

    </tr>

</thead>

<tbody>

    {appointments.map((appointment, index) => (

        <tr key={index}>

            <th>{appointment.patient.firstName} {appointment.patient.lastName}</th>

            <th>{new Date(appointment.appointmentDate).toLocaleDateString()}</th>

            <th>{appointment.healthcareProvider.name}</th>

        </tr>


    ))}

</tbody>

</table>

<a className="ref" href="http://localhost:3000/set/appointment">Registrar Cita</a>

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

</div>

);

};

export default AppointmentList;

import React, { useState, useEffect } from 'react';

import axios from 'axios';

function MedicationList() {

const [medications, setMedications] = useState([]);

useEffect(() => {

axios.get('http://localhost:8080/api/medications')

.then(response => {

setMedications(response.data);


});

}, []);

const deleteMedication = (id) => {

axios.delete(`http://localhost:8080/api/medications/\${id}`)

.then(() => {

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        setMedications(medications.filter(medication => medication.id !== id));

    });

};

return (

    <div className='container'>

        <h2>Lista de Medicamentos</h2>

        <table className='table'>

            <thead>

                <tr>

                    <th>Nombre</th>

                    <th>Dosis</th>

                    <th></th>

                </tr>

            </thead>

            <tbody>

                {medications.map(medication => (


                    <tr key={medication.id}>

                        <td>{medication.name}</td>

                        <td>{medication.dosage}</td>

                        <td>

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        <button          className='btn          btn-danger'          onClick={()          =>
deleteMedication(medication.id)}}>Eliminar</button>

      </td>

    </tr>

  )}}

</tbody>

</table>

    <a          className="ref"          href="http://localhost:3000/set/medication">Registrar
Medicamentos</a>

  </div>

);

}

```

```
export default MedicationList;
```


```
import React, { useEffect, useState } from "react";
```

```
import axios from "axios";
```

```
import "./css/Appointments.css"
```

```
const PatientList = () => {
```

```
  const [patients, setPatients] = useState([]);
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
useEffect(() => {

  axios

    .get("http://localhost:8080/api/patients")

    .then((response) => {

      setPatients(response.data);

    })

    .catch((error) => {

      console.error("There was an error!", error);

    });

}, []);
```

```
return (

  <div className="container">

    <h2>Lista de pacientes</h2>

    <table className="table">

      <thead>

        <tr>


          <th scope="col">Id</th>

          <th scope="col">Nombre</th>

          <th scope="col">Apellido</th>

          <th scope="col">Fecha de nacimiento</th>

          <th></th>
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

    </tr>

  </thead>

  <tbody>

    {patients.map((patient, index) => (

      <tr key={index}>

        <th>{patient.id}</th>

        <th>{patient.firstName}</th>

        <th>{patient.lastName}</th>

        <th>{new Date(patient.dateOfBirth).toLocaleDateString()}</th>

      </tr>

    ))}

  </tbody>

</table>

<a className="ref" href="http://localhost:3000/set/patient">Registrar Paciente</a>

</div>

);


};

export default PatientList;

import React, { useState, useEffect } from 'react';

import axios from 'axios';

```


	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import './css/Appointments.css'
```

```
function ProviderList() {
```

```
  const [providers, setProviders] = useState([]);
```

```
  useEffect(() => {
```

```
    axios.get('http://localhost:8080/api/providers')
```

```
      .then(response => {
```

```
        setProviders(response.data);
```

```
      });
```

```
  }, []);
```

```
  const deleteProvider = (id) => {
```

```
    axios.delete(`http://localhost:8080/api/providers/${id}`)
```

```
      .then(() => {
```

```
        setProviders(providers.filter(provider => provider.id !== id));
```


```
      });
```

```
  };
```

```
  return (
```

```
    <div className='container'>
```

```
      <h2>Lista de profesionales</h2>
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
<table className="table">
```

```
  <thead>
```

```
    <tr>
```

```
      <th scope="col">Id</th>
```

```
      <th scope="col">Nombre</th>
```

```
      <th scope="col">Especialidad</th>
```

```
      <th></th>
```

```
    </tr>
```

```
  </thead>
```

```
  <tbody>
```

```
    {providers.map(provider => (
```

```
      <tr key={provider.id}>
```

```
        <td>
```

```
          {provider.id}
```

```
        </td>
```

```
        <td>
```

```
          {provider.name}
```

```
        </td>
```

```
        <td>
```

```
          {provider.specialty}
```

```
        </td>
```

```
        <td>
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

        <button      className='btn      btn-danger'      onClick={()      =>
deleteProvider(provider.id)}>Eliminar</button>

        </td>

    </tr>

    )}}

</tbody>

</table>

    <a      className="ref"      href="http://localhost:3000/set/provider">Registrar
profesional</a>

    </div>

    );

}

export default ProviderList;

import React from 'react';

import { Map, GoogleApiWrapper } from 'google-maps-react';


const mapStyles = {

    width: '100%',

    height: '100%'

};

```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
export class MapContainer extends React.Component {

  render() {

    return (

      <Map

        google={this.props.google}

        zoom={14}

        style={mapStyles}

        initialCenter={{ lat: -1.2884, lng: 36.8233 }}

      />

    );

  }

}
```

```
export default GoogleApiWrapper({

  apiKey: 'AIzaSyCnUMDIoSFrXD6cJ5uuzbWztzW1WgquI5Q'


})(MapContainer);
```

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

import CreatePatient from "../components/CreatePatient";

import PatientList from "../components/ListPatients";

import HealthcareProviderForm from "../components/CreateProvider";
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```
import AppointmentForm from "../components/CreateAppointment";
```

```
import AppointmentList from "../components/ListAppointment";
```

```
import Indexap from "../components/Indexap";
```

```
import ProviderList from "../components/ListProviders";
```

```
import { MapContainer } from "../components/Maps";
```

```
import MedicationList from "../components/ListMedication";
```

```
import CreateMedication from "../components/CreateMedication";
```

```
function App() {
```

```
  return (
```

```
    <>
```

```
    <BrowserRouter>
```

```
      <Routes>
```

```
        <Route path="/" element={<Indexap/>}/>
```

```
        <Route path="/set/patient" element={<CreatePatient/>}/>
```

```
        <Route path="/patients" element={<PatientList/>}/>
```


```
        <Route path="/providers" element={<ProviderList/>}/>
```

```
        <Route path="/set/provider" element={<HealthcareProviderForm/>}/>
```

```
        <Route path="/set/appointment" element={<AppointmentForm/>}/>
```

```
        <Route path="/appointments" element={<AppointmentList/>}/>
```

```
        <Route path="/maps" element={<MapContainer/>}/>
```

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01

```

    <Route path='/medication' element={<MedicationList/>}/>

    <Route path='/set/medication' element={<CreateMedication/>}/>


  </Routes>

</BrowserRouter>

</>

);
}

export default App;

```

Referencias

Introduction to Node.js. (s.f.). *How to install nodejs*. Obtenido de Nodejs:

<https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>

Omaña, M. C. (2012). *Manufactura Esbelta: Una contribución para el desarrollo de software*.


Red Enlace. Obtenido de <https://elibro.net/es/ereader/biblioibero/98547/>

Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería de software un enfoque practico*.

McGraw-Hill Interamericana. Obtenido de <http://www.ebooks7->

[24.com.ibero.basesdedatosezproxy.com/?il=16414](http://www.ebooks7-24.com.ibero.basesdedatosezproxy.com/?il=16414)

React. (s.f.). *React*. Obtenido de Instalación - React: <https://es.react.dev/learn/installation>

	Documento de formulación de proyecto	Fecha de actualización 29/09/2023
		Version: 01
		Código: IB-01