



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Curso: Ingeniería de Software 1 (2016701)

Profesor: Oscar Eduardo Alvarez Rodriguez

Estudiante(s):

Nelson Ivan Castellanos Betancourt

Angel Santiago Avendaño Cañon

David Sebastian Hurtado Sanchez

Nicolas Zuluaga Galindo

1. LEVANTAMIENTO DE REQUERIMIENTOS

La idea surgió al identificar que el sector salud ofrece un gran potencial para mejorar la atención al usuario y el manejo de la información mediante tecnología, lo que redundaría en mayor productividad para los profesionales de la salud y mayores utilidades para las entidades. Al inicio del semestre, cada integrante del equipo propuso diversas temáticas y, tras exponer motivaciones y beneficios, se realizó una votación para definir el proyecto. Coincidimos en que el sistema de salud actual presenta largos tiempos de espera y herramientas informáticas insuficientes; por ello, optimizar el servicio de urgencias podría significar la diferencia entre la vida y la muerte, además de ser un factor diferenciador en un mercado competitivo. Los usuarios potenciales esperan soluciones como auto-triage, medición de ocupación o generación de estadísticas, con miras a acortar tiempos de espera y mejorar la atención. Para nosotros, esta iniciativa representa una oportunidad de aprender el desarrollo organizado de software, manejar recursos y equipo, y familiarizarnos con tecnologías y metodologías vigentes en el entorno laboral.

A partir de la información obtenida de la entrevista realizada a Gemini, se identificó que el cliente es un médico con experiencia en el área de urgencias. Él requiere una página web integral y eficiente para el área de urgencias del centro médico. Este espacio servirá como una página web que permitirá mejorar la comunicación, optimizar los procesos y mejorar la atención del paciente. Así mismo, se busca que este sitio web sea la tarjeta de presentación online del centro médico.

A continuación se realiza una breve descripción de los requerimientos identificados según el usuario:

1. Usuarios externos

1.1. Pacientes

- 1.1.1. Información del hospital:** El paciente debe tener la capacidad de acceder a información general sobre los servicios de urgencias, ubicación, horarios y especialidades.
- 1.1.2. Pre Registro y auto-triage:** Permitir a los pacientes registrar sus datos personales, el contacto de emergencia y el motivo de consulta antes de llegar al centro médico. Adicionalmente, tener un sistema de

preguntas guiadas que ayude a los pacientes a determinar la gravedad de su situación.

- 1.1.3. Consulta de tiempo de espera: A partir del pre-registro y de la evaluación médica, generar una estimación del tiempo de espera de cada paciente según la gravedad de sus síntomas.
- 1.1.4. Preguntas frecuentes: Generar una sección de preguntas frecuentes frente al proceso en la sección de urgencias.
- 1.1.5. Campañas de salud: Encontrar blog e información de los temas actuales relacionadas con el sector salud (virosis, cuidados general,vacunación,etc)
- 1.2. Acompañantes
 - 1.2.1. Información del hospital: El paciente debe tener la capacidad de acceder a información general sobre los servicios de urgencias, ubicación, horarios y especialidades.
 - 1.2.2. Notificaciones estado paciente: Recibir actualizaciones sobre el estado del paciente de manera virtual (se debe cumplir las normativas de privacidad en el manejo de los datos)
 - 1.2.3. Información de la patología del paciente: Encontrar información de la patología del paciente y recomendaciones para su manejo.
 - 1.2.4. Campañas de salud: Encontrar blog e información de los temas actuales relacionadas con el sector salud (virosis, cuidados general,vacunación,etc)
- 1.3. Público general
 - 1.3.1. Información del hospital: El paciente debe tener la capacidad de acceder a información general sobre los servicios de urgencias, ubicación, horarios y especialidades.
 - 1.3.2. Campañas de salud: Encontrar blog e información de los temas actuales relacionadas con el sector salud (virosis, cuidados general,vacunación,etc)

2. Usuarios Internos - Personal de la salud y

- 2.1. Médicos de urgencias
 - 2.1.1. Sistema de mensajería interna: Poder comunicarse con otros miembros del equipo a través de un sistemas de mensajería en la intranet,
 - 2.1.2. Servidor con información relevante del hospital: Acceso a protocolos, guías clínicas, procedimientos o cualquier material que sea de interés para los médicos y enfermeros.
 - 2.1.3. Visualizador de turnos y horarios del personal: Visualizador que permita identificar los profesionales que se van a encontrar de turno en el centro médico.
 - 2.1.4. Historias clínicas: Al momento de la consulta, se debe contar con toda la historia clínica del paciente para realizar una valoración médica o un procedimiento.
 - 2.1.5. Interconsultas según especialidad: En caso de presentarse un caso en donde un médico general no tenga las conocimientos necesarios, poder comunicarse con el médico especialista de turno.
- 2.2. Enfermeros

- 2.2.1. Servidor con información relevante del hospital: Acceso a protocolos, guías clínicas, procedimientos o cualquier material que sea de interés para los médicos y enfermeros.
- 2.2.2. Visualizador de turnos y horarios del personal: Visualizador que permita identificar los profesionales que se van a encontrar de turno en el centro médico.
- 2.2.3. Historias clínicas: Al momento de la consulta, se debe contar con toda la historia clínica del paciente para realizar una valoración médica o un procedimiento.
- 2.2.4. Asignación de camas y recursos: Interfaz que permita identificar la ocupación de las camas, consultorios y espacios de procedimientos en el centro médico.
- 2.3. Personal administrativo
 - 2.3.1. Gestionar información de la página web: Actualizar la información correspondiente a la información del centro médicos, campañas de salud, entre otras.
 - 2.3.2. Gestionar usuarios y permisos: Actualizar los permisos a los diferentes usuarios que entran a la intranet.
 - 2.3.3. Monitorear estadísticas del sistemas: Interfaz gráfica para visualizar los pacientes, nivel de urgencia, cantidad de procedimientos, entre otras.
 - 2.3.4. Notificaciones estado paciente: Recibir y generar actualizaciones sobre el estado del paciente de manera virtual.

Lista de funcionalidades

1. Información relacionada al hospital: (1.1.1,1.1.14,1.15)

La funcionalidad de información sobre el hospital, ayuda proporcionando a los usuarios externos (Como pacientes, acompañantes y público general) detalles clave como los servicios ofrecidos en urgencias, horarios de atención, ubicación del o los centros médicos y las especialidades y tratamientos disponibles. Esta sección actúa asimismo como una guía inicial para los usuarios que buscan interactuar con el centro médico, ya sea para solicitar atención o simplemente para informarse sobre los servicios

La funcionalidad es crucial, dado que establece la primera impresión que los usuarios tienen del centro médico. Debe ser una página clara, accesible, bien estructurada, y que ayude al usuario con información clara, reduciendo así el número de llamadas o visitas innecesarias relacionadas con dudas generales, optimizando los recursos administrativos

2. Pre Registro y auto triage: (1.1.2)

El pre-registro permite a los pacientes ingresar sus datos personales, contacto de emergencia y motivo de consulta antes de llegar al centro médico. A su vez, el auto-triage utiliza un sistema de preguntas guiadas para evaluar la gravedad de los síntomas del paciente, categorizándolos en niveles de urgencia. La herramienta

permite ahorrar tiempo tanto al paciente como al personal de salud, al agilizar los procesos administrativos y de clasificación en la entrada del centro médico.

La importancia de la funcionalidad radica en la capacidad de optimizar los tiempos de atención y garantizar que los casos más graves reciban la prioridad adecuada. Al permitir que los pacientes realicen este paso desde el hogar o cambio al hospital, reduce la saturación en las áreas de recepción y se aumenta la eficiencia del flujo de trabajo interno, además de mejorar la experiencia del paciente al darles un papel activo en su proceso de atención

3. Consulta en tiempo real: (1.1.3)

La consulta en tiempo real hace uso de la información recopilada en el proceso de pre-registro y auto-triage para proporcionar a los pacientes una estimación del tiempo de espera. El cálculo se basa en la gravedad de los síntomas, la cantidad de pacientes en el sistema y la disponibilidad del personal médico. La herramienta ofrece transparencia y establece expectativas claras para los pacientes que buscan atención en el área de urgencias.

Es importante para la mejora de la experiencia del paciente, al reducir la incertidumbre asociada con los tiempos de espera en urgencias, además de fomentar la organización y priorización dentro del hospital, permitiendo al personal enfocarse en casos críticos mientras los pacientes más urgentes son informados de su posición en la fila.

4. Notificaciones: (1.2.2 -1.2.3)

Las notificaciones están diseñadas para mantener informados a los acompañantes sobre el estado del paciente de manera virtual. Estas notificaciones pueden incluir actualizaciones sobre la evaluación médica inicial, tiempos de espera, cambios en el estado de salud o la finalización de procedimiento. Este sistema permite a los acompañantes estar al tanto sin necesidad de permanecer físicamente en el hospital.

Es imperativo mejorar la comunicación entre el hospital y las familias de los pacientes, En situaciones críticas, la tranquilidad de recibir actualizaciones en tiempo real puede marcar una gran diferencia en la experiencia de los acompañantes, además de reducir la congestión en las áreas de espera del centro médico, al disminuir la necesidad de interacción física con el personal administrativo

5. Sistema de mensajería interna: (2.1.1-2.1.5)

Un sistema de mensajería interna tiene permite la comunicación directa entre los miembros del personal del centro de salud, por medio de la intranet del edificio. Los médicos, enfermeros y otros empleados pueden hacer uso de la herramienta para coordinar procedimientos, intercambiar información sobre casos específicos y enviar alertas en tiempo real. La mensajería puede incluir características como chats grupales por especialidad o departamento, al igual que notificaciones de alta prioridad.

Esto permite agilizar la comunicación en un entorno donde cada segundo cuenta. Al reducir la necesidad de interacciones presenciales o llamadas telefónicas, el personal puede centrarse en el cuidado del paciente

6. Turnos y horarios: (2.1.3)

La funcionalidad de visualización de turnos y horarios permite al personal médico y administrativo consultar fácilmente la distribución de las jornadas laborales. Los usuarios pueden ver qué profesionales están de turno, sus horarios y las áreas en las que estarán asignados. Esta herramienta también puede incluir notificaciones sobre cambios en los turnos o recordatorios.

Esta funcionalidad es importante porque asegura una gestión eficiente de los recursos humanos del hospital, reduciendo conflictos o duplicaciones en las asignaciones de personal. Además, mejora la planificación individual de los profesionales, ayudándoles a organizar sus actividades dentro y fuera del hospital.

7. Historias clínicas: (2.1.4)

El acceso a historias clínicas permite al personal médico consultar toda la información relevante sobre los pacientes en tiempo real. Desde antecedentes médicos y resultados de exámenes hasta procedimientos previos realizados en el hospital, este sistema centraliza y organiza los datos clínicos para facilitar el diagnóstico y la toma de decisiones.

La importancia de esta funcionalidad radica en su capacidad de mejorar la calidad del cuidado médico. Al tener acceso inmediato a información completa y precisa, los profesionales pueden realizar evaluaciones más acertadas y reducir los riesgos asociados con errores o falta de información. También permite la continuidad del cuidado, especialmente en casos recurrentes o de larga duración.

8. Medida de ocupación: (2.2.4)

La funcionalidad de medida de ocupación proporciona una vista en tiempo real sobre la disponibilidad de camas, consultorios y otros recursos dentro del hospital. Este sistema permite al personal identificar rápidamente qué áreas están ocupadas y cuáles tienen capacidad para recibir nuevos pacientes.

Esta funcionalidad es clave para optimizar la gestión operativa del hospital. Al ofrecer datos precisos sobre la ocupación, el personal puede tomar decisiones informadas sobre la asignación de recursos, reducir tiempos de espera y evitar la sobrecarga de ciertas áreas. También ayuda a prevenir el colapso de los servicios en momentos de alta demanda.

9. Estadísticas del sistema: (2.3.4)

La funcionalidad de estadísticas del sistema permite al personal administrativo y a los directivos acceder a un panel con datos relevantes sobre el funcionamiento del área de urgencias. Este panel puede incluir métricas como el número de pacientes atendidos, tiempos promedio de espera, niveles de urgencia y cantidad de procedimientos realizados en un periodo determinado.

Esta funcionalidad es fundamental para evaluar y mejorar la eficiencia del hospital. Al analizar los datos recopilados, los administradores pueden identificar cuellos de botella, planificar mejoras en los procesos y justificar decisiones estratégicas basadas en evidencia. También permite el cumplimiento de indicadores clave de rendimiento (KPIs) en el sector salud.

2. ANÁLISIS DE REQUERIMIENTOS

Listado de requisitos funcionales:

1. **Información general y de contacto:** Acceso a información general (contacto, ubicación, horarios, servicios ofrecidos).
2. **Pre-registro online:** Formulario para que los pacientes puedan registrar sus datos personales y motivo de consulta antes de llegar al hospital. También disponible para que el personal interno registre pacientes que no realizaron el pre-registro.
3. **Auto-triage preliminar:** Sistema de preguntas guiadas que ayude a los pacientes a determinar la gravedad de su situación y si necesitan atención urgente.
4. **Consulta de tiempos de espera estimado:** Información actualizada en tiempo real o lo más aproximado posible sobre los tiempos de espera para ser atendido.
5. **Sistema de autenticación para personal interno y administradores:** Sistema de autenticación seguro para el personal del servicio de urgencias.
6. **Notificaciones externas dirigidas a familiares de pacientes:** Recibir actualizaciones sobre el estado del paciente.
7. **Notificaciones internas en tiempo real para el personal de urgencias:** El personal interno recibe notificaciones sobre información importante.
8. **Gestión de notificaciones:** Crear nuevas notificaciones y verificar el estado de antiguas notificaciones.
9. **Admisión de pacientes:** Llevar un registro del ingreso de los pacientes. El personal del centro médico realiza la admisión de los pacientes, con y sin pre-registro.
10. **Salida de pacientes:** Registrar los detalles relevantes de la salida de un paciente, por parte del personal del centro médica. Actualiza automáticamente la disponibilidad de recursos del centro médico.
11. **Mensajería interna:** Sistema de mensajería interna para facilitar la comunicación y coordinación del equipo de urgencias.
12. **Acceso a protocolos y guías:** Repositorio digital de protocolos de actuación, guías clínicas y procedimientos estandarizados.
13. **Consulta y gestión de turnos y horarios:** Visualización y gestión de los turnos del personal de urgencias..
14. **Acceso a historias clínicas:** Acceder a información de historias clínicas de los pacientes a través de la integración con otros sistemas.
15. **Medida de ocupación del centro médico:** Proporciona una vista en tiempo real sobre la disponibilidad de camas, consultorios y otros recursos dentro del hospital.
16. **Reportes y estadísticas:** Acceso a datos relevantes sobre el funcionamiento del servicio.

17. **Blog y campañas de salud (Portal de educación para pacientes):** Recursos educativos sobre temas de salud relacionados con las urgencias más comunes.
18. **Encuestas de satisfacción:** Formulario para que los pacientes puedan evaluar la atención recibida en urgencias.
19. **Integración con sistemas hospitalarios:** Conexión con el sistema de historias clínicas electrónicas para facilitar el flujo de información, tales como laboratorio, farmacia.
20. **Preguntas frecuentes:** Respuestas claras y concisas a las preguntas más comunes sobre el servicio de urgencias.
21. **Solicitud de citas post-urgencias:** Integración con el sistema de citas del hospital.

Listado de requisitos no funcionales:

1. **Diseño y usabilidad:**
 - a. Diseño limpio, minimalista y profesional.
 - b. Navegación intuitiva.
 - c. Responsividad: adaptable a cualquier dispositivo.
 - d. Cumplimiento de WCAG para accesibilidad.
2. **Rendimiento y escalabilidad:**
 - a. Actualización de tiempos de espera en tiempo real o cada 15 minutos.
 - b. Capacidad para soportar un número elevado de usuarios simultáneos.
3. **Seguridad:**
 - a. Protección de datos sensibles conforme a normativas internacionales
 - b. Respaldo periódico y mecanismos de recuperación
4. **Mantenimiento y actualización:**
 - a. Facilidad para que el personal administrativo gestione el contenido

Listado de funcionalidades clasificadas con el método MoSCoW:

| <u>Funcionalidad</u> | <u>Must</u> | <u>Should</u> | <u>Could</u> | <u>Won't</u> |
|--------------------------------------|-------------|---------------|--------------|--------------|
| Información del centro médico | <u>X</u> | | | |
| Pre-registro | <u>X</u> | | | |
| Auto-triage | | <u>X</u> | | |
| Consulta en tiempo real | | <u>X</u> | | |
| Sistema de autenticación | <u>X</u> | | | |
| Notificaciones para acompañantes | | | | <u>X</u> |
| Notificaciones y alertas al personal | <u>X</u> | | | |

| <u>Funcionalidad</u> | <u>Must</u> | <u>Should</u> | <u>Could</u> | <u>Won't</u> |
|--|--------------------|----------------------|---------------------|---------------------|
| Gestión de notificaciones | <u>X</u> | | | |
| Admisión de pacientes | <u>X</u> | | | |
| Salida de pacientes | <u>X</u> | | | |
| Sistema de mensajería interna | | <u>X</u> | | |
| Acceso a protocolos y guías | <u>X</u> | | | |
| Turnos y horarios | | <u>X</u> | | |
| Historias clínicas | | <u>X</u> | | |
| Medida de ocupación del centro médico | <u>X</u> | | | |
| Estadísticas del sistema | | <u>X</u> | | |
| Blog y campañas de salud | | | <u>X</u> | |
| Encuestas de satisfacción | | | <u>X</u> | |
| Integración con sistemas hospitalarios | | | <u>X</u> | |
| FAQ | | | <u>X</u> | |
| Solicitud de citas post-urgencias | | | | <u>X</u> |

Nota: Para establecer la prioridad de estas funcionalidades se tuvo en cuenta la opinión del cliente (Gemini), quién especificó que algunas de las funcionalidades debían tener una prioridad específica en el proyecto.

Descripción MVP

El MVP del sistema de gestión para un centro médico optimiza la administración y la comunicación interna, mejorando la calidad de la atención y la eficiencia operativa. Se priorizan funcionalidades esenciales, como la gestión de notificaciones que alerta al personal, así como la admisión y el alta de pacientes para agilizar su flujo dentro del centro. El pre-registro de pacientes reduce los tiempos de espera, mientras que la medición de la ocupación del centro médico permite asignar recursos de manera más eficiente. También se integra un sistema de autenticación para el personal de salud y administrativo, garantizando un acceso seguro y el acceso a protocolos y guías médicas asegura una toma de decisiones adecuada. Estas funciones digitalizan procesos críticos, optimizan las operaciones y mejoran la experiencia del paciente. Gracias a su arquitectura modular, este MVP establece las bases para futuras mejoras, como el auto-triage y la teleconsulta, consolidando una solución escalable y altamente eficiente para el centro médico.

3. ANÁLISIS GESTIÓN DE SOFTWARE

ESTIMACIÓN DE RECURSOS

Recursos humanos:

Equipo principal:

1. Desarrolladores backend: 2 junior
2. Desarrolladores frontend: 2 junior
3. Diseñadores UX/UI: 1
4. QA Tester: 1
5. Director de proyecto: 1 senior

Equipo de apoyo:

1. Redactor de contenido: 1
2. Consultor médico
3. Especialista Legal.

Estimación de tiempo:

| Funcionalidad | Complejidad | Días (Fibonacci) | Justificación |
|-------------------------------|-------------|------------------|--|
| Información del centro médico | Baja | 3 | La implementación consiste en la creación de una sección estática con información básica. Esto implica el diseño de una página HTML simple y su conexión a un sistema de gestión de contenidos (CMS). No requiere integración con bases de datos o lógica compleja. |
| Pre-registro | Media | 5 | Incluye el diseño de un formulario interactivo que capture datos personales y de contacto. Requiere validaciones de entrada, integración con una base de datos para almacenar información, y pruebas de usabilidad para garantizar que los pacientes puedan usarlo fácilmente. |
| Auto-triage | Alta | 5 | Implica el desarrollo de un sistema de |

| Funcionalidad | Complejidad | Días (Fibonacci) | Justificación |
|---------------------------|-------------|------------------|---|
| | | | preguntas guiadas con lógica condicional para determinar la gravedad de los síntomas. Esto requiere la implementación de algoritmos de clasificación básicos, validación de las decisiones tomadas por el sistema y un diseño amigable para el usuario. |
| Consulta en tiempo real | Media | 3 | Este sistema calcula tiempos estimados de espera basándose en datos existentes como el número de pacientes y su nivel de urgencia. Requiere integración con la base de datos y una lógica para calcular prioridades, pero su desarrollo es relativamente directo. |
| Sistema de autenticación | Media | 5 | Incluye la creación de un sistema de registro e inicio de sesión con niveles de acceso diferenciados (pacientes, médicos, administrativos). Requiere la implementación de encriptación para las contraseñas y pruebas de seguridad. |
| Gestión de notificaciones | Media | 5 | Incluye la creación de un modelo, validación de reglas del sistema y revisión de arquitectura una correcta implementación compatible con el sistema que se encarga de procesar las notificaciones. |
| Notificaciones para | Alta | 8 | Requiere desarrollar un |

| Funcionalidad | Complejidad | Días (Fibonacci) | Justificación |
|--------------------------------------|-------------|------------------|---|
| acompañantes | | | sistema de notificaciones en tiempo real con cumplimiento de normativas de privacidad (como GDPR o HIPAA). Esto incluye integrar servicios de mensajería o notificaciones push. |
| Notificaciones y alertas al personal | Media | 5 | Implica la creación de un sistema de alertas altamente confiable y en tiempo real. Esto incluye la configuración de canales de comunicación internos, prioridades de notificación y la integración con el flujo de trabajo hospitalario. |
| Admisión de pacientes | Media | 3 | Esta funcionalidad requiere de realizar una interacción con otras funcionalidades, Pre-registro y Ocupación del centro médico. Modelar y diseñar la arquitectura resiliente que garantice el funcionamiento correcto que impacta en otras partes del sistema requiere de mínimo 3 días. |
| Salida de pacientes | Media | 3 | Igual que la anterior, requiere interactuar con otra funcionalidad, Ocupación del centro médico. Su complejidad es similar al ingreso de pacientes y requiere de garantizar un funcionamiento correcto, porque impacta en las métricas de ocupación de los recursos del centro médico. |

| Funcionalidad | Complejidad | Días (Fibonacci) | Justificación |
|---------------------------------------|-------------|------------------|---|
| Sistema de mensajería interna | Alta | 13 | Requiere desarrollar una plataforma de chat segura y eficiente para la comunicación interna del hospital. Incluye características como chats grupales, notificaciones y manejo de archivos adjuntos, además de garantizar la seguridad de la información. |
| Acceso a protocolos y guías | Media | 8 | Esta funcionalidad requiere desarrollar un repositorio centralizado con acceso restringido por permisos, interfaz de búsqueda avanzada y capacidad de actualización en tiempo real. |
| Turnos y horarios | Media | 3 | El desarrollo incluye la creación de un calendario dinámico que permita consultar los horarios de los profesionales. Esto requiere integración con una base de datos y un diseño adaptable. |
| Historias clínicas | Alta | 8 | Requiere integrar un sistema para almacenar y visualizar historias clínicas. Esto implica manejo de datos sensibles, autorización de acceso y capacidad de actualización en tiempo real. |
| Medida de ocupación del centro médico | Alta | 5 | Incluye el diseño de una herramienta para monitorear la ocupación de camas y consultorios en tiempo real. Requiere integración con dispositivos y bases de |

| Funcionalidad | Complejidad | Días (Fibonacci) | Justificación |
|--|-------------|------------------|---|
| | | | datos, además de visualizaciones claras y actualizaciones frecuentes. |
| Estadísticas del sistema | Media | 8 | El desarrollo del panel de estadísticas incluye la recolección, procesamiento y visualización de datos clave mediante gráficos. Esto implica trabajar con una base de datos y bibliotecas de visualización. |
| Blog y campañas de salud | Media | 8 | Implica la creación de un sistema que permita al personal administrativo publicar contenido dinámico en el sitio web. Requiere un sistema de gestión de contenidos y una interfaz amigable. |
| Encuestas de satisfacción | Baja | 2 | Se puede implementar utilizando herramientas estándar como formularios HTML y una base de datos simple para registrar respuestas. |
| Integración con sistemas hospitalarios | Alta | 13 | Implica la conexión con otros sistemas ya existentes, lo que requiere trabajo en APIs, mapeo de datos y sincronización segura. |
| FAQ | Baja | 2 | Consiste en la creación de una sección estática con preguntas frecuentes, que no requiere lógica compleja. |
| Solicitud de citas post-urgencias | Baja | 3 | Requiere un formulario simple y la integración con el sistema de gestión de citas |

| Funcionalidad | Complejidad | Días (Fibonacci) | Justificación |
|---------------|-------------|------------------|---------------|
| | | | existente. |
| Total | | 118 | |

Tecnologías a utilizar:

Frontend:

- Diseño y maquetación: Figma.
- Framework: React.js.
- Lenguajes requeridos: HTML5, CSS3, JavaScript.

Backend:

- Framework: Django Rest Framework
- Base de datos: PostgreSQL/SQLite + MongoDB
- Almacenamiento de objetos: S3 API
- Lenguajes requeridos: Python, SQL.
- Correo electrónico (Transaccional): Resend/AWS SES

Herramientas colaborativas:

- Github para el control de versiones.
- Jira/Trello para gestión de tareas.

Infraestructura:

- Hosting: Vercel, CloudRun
- Base de datos: Supabase/Turso
- Almacenamiento de objetos: Cloudflare R2/AWS S3
- Notificaciones: Firebase, Resend

Estimación de costo:

1. Desarrolladores Backend (Junior)

- **Total horas por desarrollador:** 515 horas/persona.
- **Justificación:**
 - El backend es responsable de implementar la lógica central del sistema, la conexión a la base de datos y las APIs que permiten que las funcionalidades del frontend operen correctamente.
 - Funcionalidades críticas como el auto-triage, historias clínicas, y la integración con sistemas hospitalarios requieren lógica avanzada y trabajo intensivo.
 - El tiempo incluye diseño de arquitectura, desarrollo, pruebas unitarias y depuración de código.
 - Parte del tiempo también se destina a la comunicación con el frontend para asegurar una integración eficiente.

2. Desarrolladores Frontend (Junior)

- **Total horas por desarrollador:** 305 horas/persona.
- **Justificación:**
 - El frontend se enfoca en implementar interfaces interactivas y amigables, basadas en los diseños proporcionados por el diseñador UX/UI.
 - Funcionalidades clave como el pre-registro, auto-triage y las notificaciones requieren interfaces personalizadas con validaciones y lógica básica en el cliente.
 - Se incluyen ajustes según la retroalimentación del QA tester y soporte al backend para integrar las APIs en las vistas.
 - Este tiempo cubre desarrollo, pruebas de interfaz y ajustes finales.

3. Diseñador UX/UI

- **Total horas:** 150 horas.
- **Justificación:**
 - Se concentra en la fase inicial del proyecto, creando wireframes, prototipos y diseños de alta fidelidad para las funcionalidades principales.
 - Durante el desarrollo, ofrece soporte a los desarrolladores frontend para garantizar que los diseños se implementen correctamente.
 - En las fases finales, revisa las interfaces para asegurar la consistencia visual y mejora la experiencia del usuario según retroalimentación.

4. QA Tester

- **Total horas:** 150 horas.
- **Justificación:**
 - Diseña y ejecuta pruebas funcionales, de integración y de regresión para cada funcionalidad.
 - Documenta los errores encontrados y coordina con los desarrolladores para resolverlos.
 - Valida el cumplimiento de criterios de aceptación antes de la entrega final.
 - Su tiempo incluye la creación de casos de prueba, ejecución manual o automatizada y generación de reportes de calidad.

5. Director de Proyecto (Senior)

- **Total horas:** 200 horas.
- **Justificación:**
 - Lidera la planificación y coordinación del proyecto, asegurando que los desarrolladores, el QA tester, y el diseñador trabajen alineados.
 - Realiza revisiones periódicas del progreso y valida entregables intermedios.
 - Dedicar tiempo a la comunicación con el cliente para aclarar requisitos, validar entregas y gestionar cambios.
 - Supervisa el presupuesto y ajusta las prioridades para cumplir los plazos.

7. Consultor Médico

- **Total horas:** 100 horas.
- **Justificación:**
 - Participa en la definición de funcionalidades médicas, como el auto-triage y las notificaciones del estado del paciente.
 - Valida la precisión y pertinencia de las preguntas y flujos médicos implementados.
 - Ofrece asesoría durante las pruebas y revisiones finales.

8. Especialista Legal

- **Total horas:** 50 horas.
- **Justificación:**
 - Asesora sobre el cumplimiento de normativas legales, como GDPR y HIPAA, en funcionalidades sensibles (historias clínicas, notificaciones).
 - Revisa los términos y condiciones del sitio web y garantiza que se sigan buenas prácticas de privacidad y manejo de datos.
 - Participa en reuniones clave relacionadas con regulaciones.

| Rol | Horas Totales | Tarifa por hora (COP) | Subtotal (COP) |
|-----------------------------------|--------------------------|-----------------------|----------------|
| Desarrolladores Backend (Junior) | 515/persona (x2) = 1,030 | 25,000 | 25,750,000 |
| Desarrolladores Frontend (Junior) | 305/persona (x2) = 610 | 25,000 | 15,250,000 |
| Diseñador UX/UI | 150 | 40,000 | 6,000,000 |
| QA Tester | 150 | 38,000 | 5,700,000 |
| Director de Proyecto (Senior) | 200 | 70,000 | 14,000,000 |
| Redactor de Contenido | 100 | 30,000 | 3,000,000 |
| Consultor Médico | 100 | 80,000 | 8,000,000 |
| Especialista Legal | 50 | 70,000 | 3,500,000 |

Total : 81,200,000

Planificación de Sprints del Proyecto


Sprint 1 (Día 1 - 10) - Configuración del Entorno y Autenticación

Objetivo: Configurar la infraestructura base del sistema y establecer el sistema de autenticación.

Tareas:

1. Creación del repositorio en GitHub y configuración de CI/CD.
2. Configuración del backend con Django Rest Framework (DRF).
3. Configuración de PostgreSQL como base de datos principal.
4. Implementación de CloudRun para el despliegue del backend.
5. Creación del sistema de autenticación y permisos (JWT, roles, usuarios).
6. Integración de Resend para correos y Firebase para notificaciones push.
7. Configuración de S3 API para almacenamiento de objetos.

 **Duración:** 10 días

 **Funcionalidades cubiertas:** ✓ Configuración de infraestructura (CloudRun, PostgreSQL, S3 API)

✓ Sistema de autenticación y permisos

Sprint 2 (Día 11 - 21) - Pre-registro y Auto-triage

Objetivo: Implementar el pre-registro de pacientes y el sistema de auto-triage.

Tareas:

1. Desarrollo del formulario de pre-registro en línea.
2. Implementación del sistema de auto-triage con preguntas guiadas.
3. Validaciones y conexión con la base de datos.
4. Pruebas iniciales y ajuste de flujos.

 **Duración:** 11 días

 **Funcionalidades cubiertas:** ✓ Pre-registro

✓ Auto-triage

Sprint 3 (Día 22 - 34) - Información del Hospital y Notificaciones

Objetivo: Proporcionar información relevante y permitir la comunicación con usuarios.

Tareas:

1. Desarrollo de la sección de información del hospital.
2. Implementación del sistema de notificaciones internas para el personal.
3. Creación de API para gestionar notificaciones.
4. Pruebas e integración con backend.

 **Duración:** 12 días

 **Funcionalidades cubiertas:** ✓ Información del hospital
✓ Notificaciones internas

Sprint 4 (Día 35 - 47) - Consulta en Tiempo Real y Turnos Médicos

Objetivo: Implementar el sistema de turnos y consulta de tiempos de espera.

Tareas:

1. Desarrollo del sistema de turnos y horarios del personal médico.
2. Implementación de consulta en tiempo real de tiempos de espera.
3. Integración de notificaciones para cambios de turno.

 **Duración:** 12 días

 **Funcionalidades cubiertas:** ✓ Consulta en tiempo real
✓ Turnos y horarios

Sprint 5 (Día 48 - 61) - Medida de Ocupación y Gestión de Camas

Objetivo: Optimizar la asignación de recursos hospitalarios.

Tareas:

1. Implementación del monitor de ocupación del hospital.
2. Desarrollo del gestor de asignación de camas y espacios.
3. Alertas para ocupación crítica.

 **Duración:** 13 días

 **Funcionalidades cubiertas:** ✓ Medida de ocupación hospitalaria


Sprint 6 (Día 62 - 74) - Historias Clínicas y Mensajería Interna

Objetivo: Gestionar el historial clínico del paciente y mejorar la comunicación del personal.

Tareas:

1. Desarrollo del sistema de historias clínicas con integración a la base de datos.
2. Implementación del sistema de mensajería interna.

 **Duración:** 12 días

 **Funcionalidades cubiertas:** ✓ Acceso a historias clínicas
✓ Sistema de mensajería interna

Sprint 7 (Día 75 - 87) - Gestión de Notificaciones y Alertas al Personal

Objetivo: Implementar un sistema robusto de alertas médicas en tiempo real.

Tareas:

1. Integración de alertas automatizadas para urgencias.
2. Creación de un panel de notificaciones para el personal médico.



Duración: 12 días



Funcionalidades cubiertas: ✓ Notificaciones al personal

✓ Gestión avanzada de alertas

Sprint 8 (Día 88 - 100) - Estadísticas y Reportes

Objetivo: Implementar herramientas de análisis y visualización de datos.

Tareas:

1. Desarrollo del panel de estadísticas hospitalarias.
2. Implementación de reportes de atención médica y tiempos de respuesta.



Duración: 12 días



Funcionalidades cubiertas: ✓ Estadísticas y reportes

Sprint 9 (Día 101 - 118) - Integraciones y Entrega Final

Objetivo: Conectar el sistema con bases de datos hospitalarias y preparar la entrega final.

Tareas:

1. Integración con sistemas de historias clínicas electrónicas.
2. Pruebas finales y optimización.



Duración: 17 días



Funcionalidades cubiertas: ✓ Integración con sistemas hospitalarios

✓ Pruebas finales y entrega

4. DISEÑO Y ARQUITECTURA

Decisión de arquitectura

Arquitectura elegida: se opta por una arquitectura del tipo cliente-servidor, en 3 capas (presentación, lógica de negocio y datos) implementada de la siguiente manera:

- **Frontend (capa de presentación):** se desarrollará utilizando React.js. Esta capa se encarga de la interacción con el usuario (paciente, personal médico, administrativos) y consume la API REST expuesta por la capa de Backend.
- **Backend (capa de lógica de negocio):** Se utiliza Django Rest Framework para la implementación de esta capa. Gestiona la autenticación, autorización y provee servicios mediante API REST.

Se optó por DRF porque responde a la necesidad de contar con un framework robusto, seguro y confiable que permite la integración con bases de datos relacionales de manera sencilla, además de contar con herramientas que automatizan la experiencia de desarrollo, como lo son las vistas genéricas.

- **Base de datos (capa de datos):** Se planea utilizar SQLite como base de datos, la decisión se tomó en base a la capa gratuita que ofrece Turso. SQLite ofrece las opciones suficientes para almacenar la información del sistema y el esquema entidad-relación se desarrolló pensando en ser compatible con este sistema de base de datos. Por otro lado, se contempla la posibilidad de utilizar MongoDB para almacenar información no estructurada o que requiera de escalabilidad horizontal; sin embargo por ahora no lo consideramos necesario para el MVP.

Infraestructura y servicios: se contratarán servicios externos para el despliegue de los servicios.

- **Hosting:** El hosting se divide en 2 secciones principales: frontend y backend. Para el frontend se considera Vercel como la opción más viable, es una forma sencilla de tener el control sobre el tráfico del sitio web y además cuenta con herramientas útiles que facilitan la experiencia de desarrollo. Por otro lado para el backend consideramos que la opción más viable para un sistema que no tendrá un alto tráfico es CloudRun, un servicio de Google Cloud para el despliegue de contenedores Docker. Ambos servicios pertenecen a la categoría Serverless y su escalabilidad es automática.
- **Hosting de base de datos:** Turso es una opción económica y con una capa gratuita generosa para el desarrollo de este proyecto. Inicialmente se contempla una cantidad de usuarios moderada por lo que este servicio es ideal para el proyecto, su costo inicial es 0. También cuenta con una gran escalabilidad.
- **Correo electrónico:** el servicio de notificaciones para el sistema en desarrollo cuenta con una funcionalidad para enviar notificaciones a través de correo electrónico, es una característica fundamental en el sistema. Optamos por el servicio de Resend, no requiere de grandes configuraciones para implementarlo, cuenta con una documentación sencilla y una API por la cual hacer uso del servicio con diferentes lenguajes de programación.

Esquema de la arquitectura:

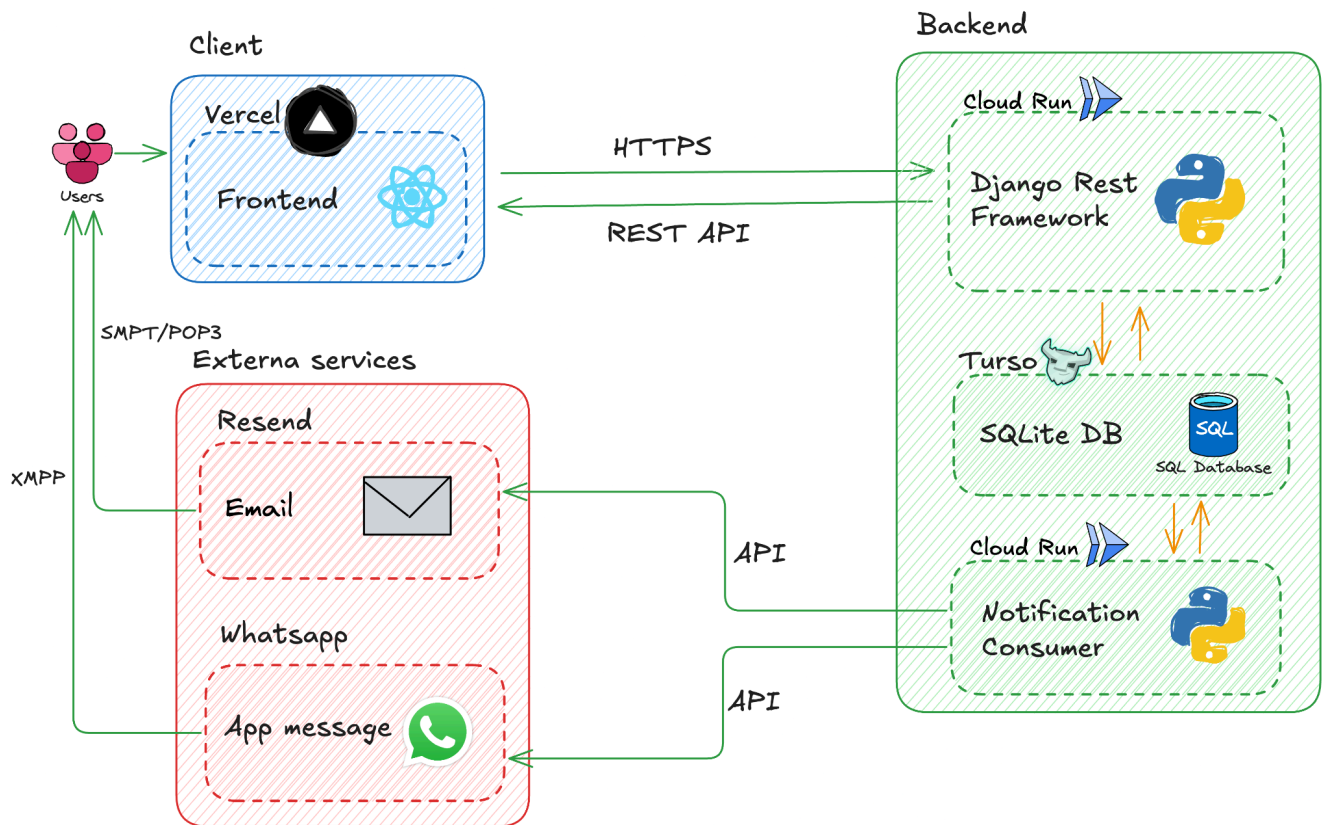


Imagen creada con <https://excalidraw.com/>

Diseño de base de datos

Modelo entidad-relación: El modelo relacional se ha diseñado para almacenar información crítica del centro médico. Esto incluye usuarios, pacientes, roles, permisos, registros de ingreso y salida de pacientes, protocolos, administración de recursos y su estado de ocupación. A continuación se presenta el diagrama **Entidad-Relación**.



Imagen generada con <https://app.chartdb.io/>

Notas:

- Este diagrama fue creado a partir de sentencias SQL, no incluye esquemas generados por Django, cómo pueden ser los de migraciones, entre otros.
- Se definieron triggers en las tablas para mantener los campos de `updated_at` actualizados, garantizando así la integridad temporal de los datos. Este aspecto no se ve reflejado en el diagrama.

Decisiones en el diseño del esquema

Para asegurar un sistema robusto y eficiente que soporte las operaciones críticas del negocio se implementaron las siguientes decisiones de diseño:

- **Normalización:** El esquema fue normalizado para evitar redundancia de datos y minimizar inconsistencias, lo que permite mantener una única fuente de información esencial. Al separar los datos en entidades bien definidas y relacionarlas mediante tablas asociativas, se mejora la calidad de la información, lo que se traduce en una mayor confiabilidad y eficiencia en los procesos administrativos y en la atención de los pacientes.

- **Índices y claves:** El uso de claves primarias auto-incrementales y claves foráneas garantizan la integridad de las relaciones entre las entidades (integridad referencial). Se creó un índice sobre el campo `id_number` en la tabla `Patients` para acelerar la búsqueda y recuperación de datos, optimizando tiempos de respuesta que impactan en la experiencia de usuario.
- **Triggers:** Se implementaron varios triggers para la actualización del campo `updated_at` en varias de las tablas (es obligatorio crear los triggers uno a uno en SQLite), con el objetivo de actualizar automáticamente este campo y facilitar el seguimiento de cambios.

¿Por qué SQL?

Se optó por SQL por su capacidad para garantizar la integridad transaccional y la consistencia en el manejo de datos estructurados y sensibles. SQL nos permite establecer relaciones complejas entre entidades a través de las claves primarias y foráneas, lo que resulta ser esencial para un sistema donde la precisión, consistencia y seguridad de la información son fundamentales.

5. PATRONES DE DISEÑO

Producer/Consumer Pattern

Permite desacoplar la creación de notificaciones (Producer) del procesamiento y envío de las mismas (Consumer). Permite al sistema mejorar su escalabilidad al tener un acoplamiento más bajo, haciéndolo más modular y robusto.

La necesidad nace de querer mantener el flujo de la API lo más enfocada posible en la responsabilidad que tiene, de modo que el flujo de trabajo no fuese bloqueado por crear notificaciones e intentar varias veces su procesamiento y envío, en procesos que pueden llegar a tener Timeout o Idle Time, según el servicio externo que se consume para enviar la notificación.

Implementación:

Para la implementación se separó el modelo en común que usa DRF para registrar las notificaciones, de modo que el Producer (DRF) y el Consumer (notification_consumer) hacen uso de un mismo modelo, evitando duplicidad de código.

El Producer crea las notificaciones usando el modelo compartido Notifications y les asigna un estado pending.

El Consumer utiliza un bucle que consulta periódicamente las notificaciones pendientes y las procesa, al procesarlas actualiza el estado según la respuesta obtenida de su procesamiento, además de gestionar los reintentos.

Pull request relacionada: [Refactorización del Consumer de Notificaciones: Implementación del Patrón Strategy y Clase Consumer](#)

Diagrama UML:

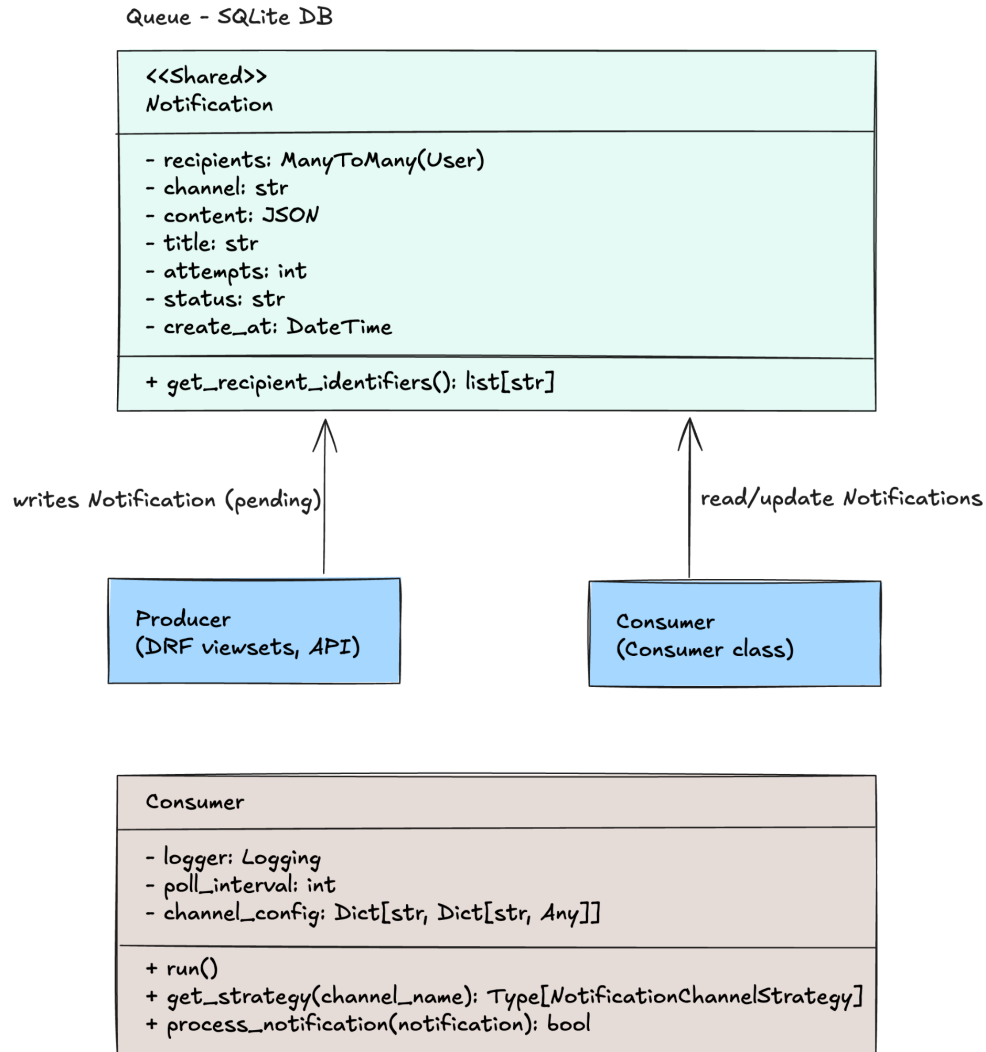


Imagen creada con <https://excalidraw.com/>

Strategy Pattern

Nos permite manejar diferentes lógicas de envío de notificaciones según el canal (email, whatsapp, etc), evitando hacer una única implementación para cualquier estructura. Dado que cada canal requiere de sus propias validaciones y configuración específicas, este patrón se ajusta perfectamente a la necesidad separando la lógica y mejorando la cohesión de cada submódulo (channel).

Implementación:

Se creó una base clase abstracta base NotificationChannel, la cuál actúa como interfaz y que define el contrato send, método que debe ser implementado por toda clase que la implemente (herede, por tratarse de python). En el flujo principal del programa se creó un diccionario el cuál contiene las configuraciones específicas para cada canal, y otro para obtener las instancias de los diferentes canales disponibles. Una vez obtenida una lista de notificaciones para procesar, a través de la función process_notification se decide

la estrategia y se procede con el envío de la notificación, usando el contrato de la interfaz send.

Diagrama UML:

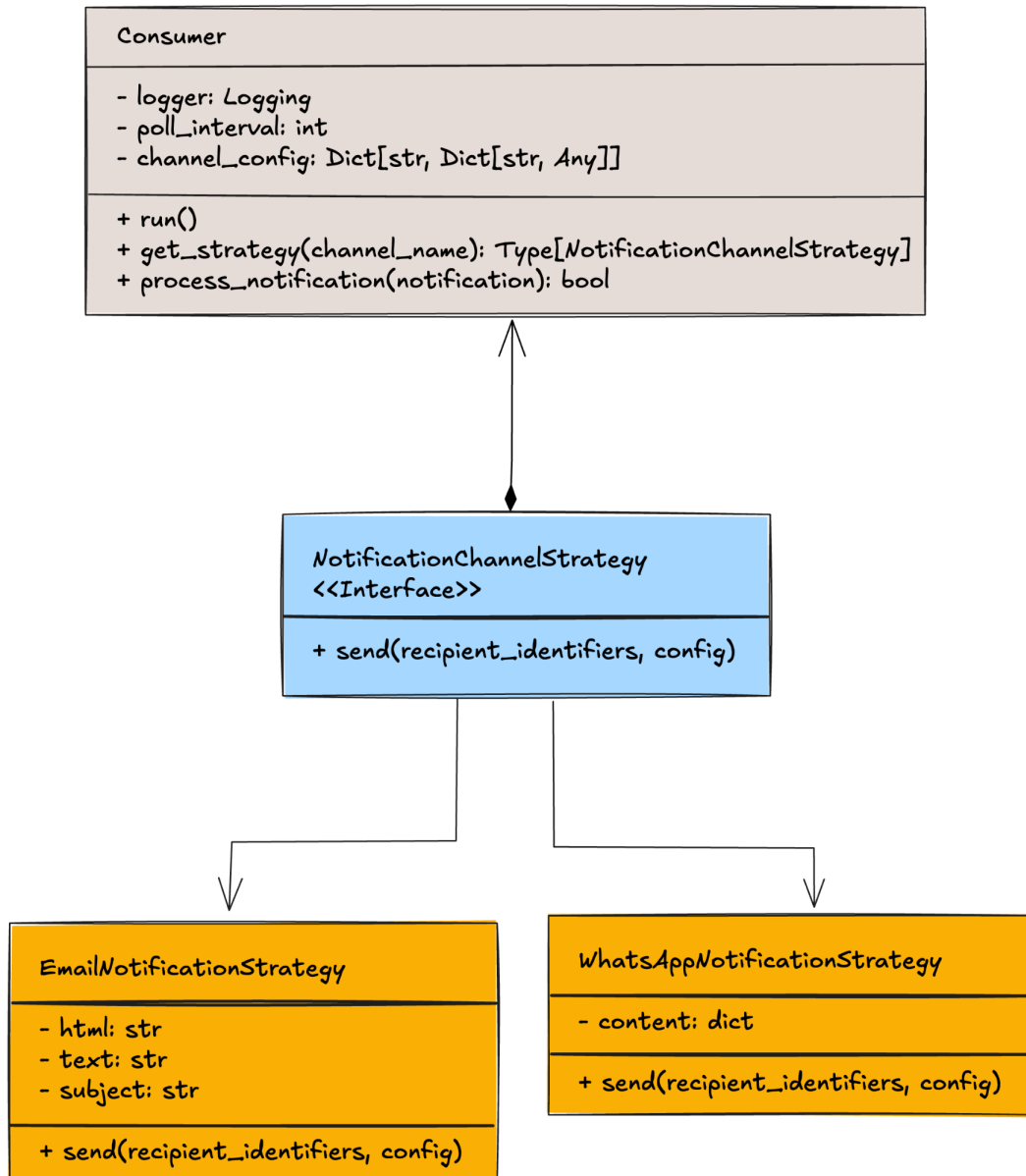


Imagen creada con <https://excalidraw.com/>

Usados en Django Rest Framework por defecto:

Mixin Pattern

Permite definir un flujo predefinido que se encarga de manejar las peticiones HTTP, la validación y transformación de los datos (a través de serializers). Se utilizó bastante en el backend, con las viewsets genéricas que provee Django Rest Framework, de modo que no fue necesario escribir toda la lógica, una a una.

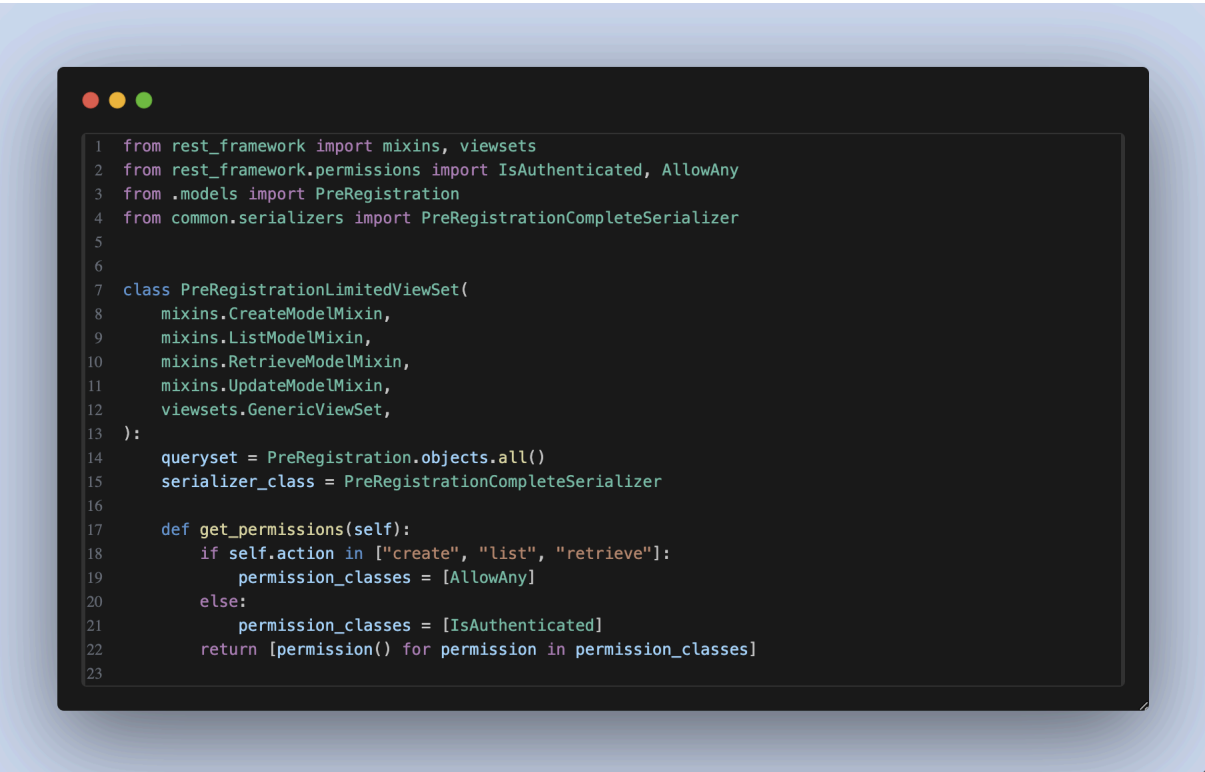
DRF maneja las operaciones CRUD de forma estandarizada, nos provee de un conjunto de vistas genéricas que se encargan de estas operaciones CRUD, evitandonos escribir código para cada tipo de operación CRUD. Para evitar escribir vistas completas para modificar partes muy específicas, el framework nos ofrece la opción de sobrescribir métodos específicos, cómo `create()`, `update()`, etc.

<https://freedium.cfd/https://nurettinabaci.com/mixin-design-pattern-2ed064719603>

<https://whiztal.io/mixins-in-django-and-django-rest-framework/>

Implementación (uso):

Django nos ofrece una implementación, nosotros nos encargamos de usarla:



```
1 from rest_framework import mixins, viewsets
2 from rest_framework.permissions import IsAuthenticated, AllowAny
3 from .models import PreRegistration
4 from common.serializers import PreRegistrationCompleteSerializer
5
6
7 class PreRegistrationLimitedViewSet(
8     mixins.CreateModelMixin,
9     mixins.ListModelMixin,
10    mixins.RetrieveModelMixin,
11    mixins.UpdateModelMixin,
12    viewsets.GenericViewSet,
13 ):
14     queryset = PreRegistration.objects.all()
15     serializer_class = PreRegistrationCompleteSerializer
16
17     def get_permissions(self):
18         if self.action in ["create", "list", "retrieve"]:
19             permission_classes = [AllowAny]
20         else:
21             permission_classes = [IsAuthenticated]
22         return [permission() for permission in permission_classes]
```

Adapter Pattern

Usado en el proyecto para adaptar los datos salidos de los modelos y procesados por los serializers. Definimos un tipo de salida estándar para la API, donde mínimo se responde con un JSON que indica el estado status y un mensaje message, de modo que el patrón Adapter nos permite estandarizar la salida de la API según lo planeado lo planeado.

Implementación (uso):

Para la implementación se creó una clase personalizada para responder con un objeto JSON a nuestra medida. DRF provee de unos renders ya predefinidos, cómo el `JSONRenderer`, del cuál creamos una subclase para estructurar la salida de la petición, basándonos en el código de estado HTTP que ya fue resuelto por las viewsets.

```
1 from rest_framework.renderers import JSONRenderer
2
3
4 class CustomJSONRenderer(JSONRenderer):
5     """
6     Render the response in JSON format with a custom structure:
7     - In successful responses (status < 400):
8       {"status": "success", "data": <data>}
9     - In error responses (status >= 400):
10      {"status": "error", "message": <message>}
11     """
12
13 def render(self, data, accepted_media_type=None, renderer_context=None):
14     response = renderer_context.get("response", None)
15
16     if response is None:
17         return super().render(data, accepted_media_type, renderer_context)
18
19     status_code = response.status_code
20
21     if status_code >= 400:
22         message = None
23         if isinstance(data, dict):
24             message = data.get("detail", data)
25         else:
26             message = data
27         custom_response = {"status": "error", "message": message}
28     else:
29         custom_response = {"status": "success", "data": data}
30
31     return super().render(custom_response, accepted_media_type, renderer_context)
32
```