Documentación de Proyecto Fonasa y Automatización Atención Pacientes - Inexoos

Autor: Jaime Campillay Rojas

Fecha: 21/08/2024

Índice

1. Resumen Ejecutivo

- **Descripción breve del proyecto**
  - El proyecto aborda la problemática de la atención de pacientes en FONASA (Fondo Nacional de Salud) y su automatización. El objetivo es clasificar a los pacientes por prioridad, considerando factores como el rango de edad, las dietas asignadas, si son fumadores o no, peso y estatura. Además, se gestiona la información de los pacientes, los distintos especialistas y los hospitales donde se atienden los pacientes de la región Metropolitana de Chile.
- **Objetivos principales a alcanzar**
  - Automatizar el orden de atención a los pacientes.
  - Clasificar el tipo de consulta según la prioridad y el rango de edad.
  - Controlar la cantidad de consultas ocupadas y disponibles.
  - Optimizar los tiempos de atención al paciente.
- **Una visión general del proyecto**
  - El proyecto busca mejorar la calidad en los tiempos de atención al paciente mediante la automatización del orden de atención, además de generar estadísticas generales sobre el desempeño del sistema.

2. Contexto del Proyecto

- **Propósito del documento**
  - Este documento tiene como objetivo detallar de manera precisa los objetivos, el alcance, la visión, la estructura y el desarrollo del proyecto de automatización y optimización de la atención de pacientes en FONASA, simplificando los procesos con el fin de ofrecer un valor agregado tanto para FONASA como para sus pacientes.
- **Explicación detallada del contexto de salud pública y la problemática que motiva el proyecto**
  - En el contexto actual, la rapidez y la capacidad de respuesta son elementos críticos, casi obligatorios, para mantener la competitividad en las propuestas de valor. Consciente de esta necesidad, FONASA requiere reconfigurar sus operaciones para optimizar los tiempos de respuesta. Esto implica automatizar y mejorar la gestión de la atención a los pacientes mediante una mayor visibilidad de los estados de las consultas y la consolidación de la información en un sistema único y de fácil uso. Esta medida permitirá realizar un análisis detallado, una evaluación precisa y la identificación de oportunidades de mejora para futuras propuestas.

3. Objetivos y Resultados Esperados

1. Automatizar el orden de atención de los pacientes según su nivel de prioridad.
2. Simplificar la gestión de atención a los pacientes para una ejecución más eficiente.
3. Mejorar la visibilidad y el seguimiento del estado de las consultas.
4. Consolidar la información en un sistema único para facilitar el análisis y la identificación de mejoras.
5. Optimizar los tiempos de respuesta y la eficiencia en la atención a los pacientes.
6. Incrementar la satisfacción de los pacientes.
7. Establecer métricas de rendimiento.

4. Antecedentes que llevaron a FONASA a enmarcarse en la Transformación Digital mediante la automatización del orden y clasificación de la atención de los pacientes

1. Dificultades para ordenar y gestionar las solicitudes de consultas de los pacientes.
2. La necesidad de mejorar los tiempos de atención a los pacientes.
3. Falta de visibilidad y seguimiento de las solicitudes de consulta.
4. Necesidad de mejorar la eficiencia operativa y la calidad del servicio.
5. Requerimiento de ofrecer un valor agregado a los pacientes mediante procesos más ágiles y la clasificación por prioridad de atención.
6. Dificultades para conocer el estado de las consultas donde atienden los especialistas.
7. Necesidad de consolidar la información en un sistema centralizado que permita un análisis más preciso y facilite la toma de decisiones estratégicas.

5. Alcance del Proyecto

- **Funcionalidades incluidas en el proyecto:**

1. **Control de Pacientes Nuevos y Existentes:**

   - Visualización de los datos de pacientes existentes.
   - Edición de los datos de pacientes existentes.
   - Eliminación de pacientes existentes.
   - Creación de nuevos pacientes.
   - Control de la información general de los pacientes (nombre, apellido, edad, número de historia clínica).
   - Control de la información de pacientes niños (estatura y peso).
   - Control de la información de pacientes jóvenes (fumador/no fumador, años fumando en caso de ser fumador).
   - Control de la información de pacientes ancianos (si tiene dieta asignada).
   - Control de la información de hospitales (ID y nombre del hospital).
   - Control de la información de especialistas (nombre, especialidad, hospital donde trabaja, estado ocupado/desocupado).

2. **Generar Consultas:**

   - Generación de solicitud de consulta por hospital.
   - Asignación de especialidad de consulta (Pediatría, CGI, Urgencia).
   - Asignación del orden de atención según prioridad y orden de llegada.

3. **Lista de Espera:**

   - Visualización de la lista de espera.
   - Atención de pacientes.
   - Liberación de consulta.
   - Orden de atención de los pacientes.

4. **Monitoreo y Estado de Consulta:**

   - Visualización en tiempo real del estado de las solicitudes de consulta.
   - Filtros de estados de solicitudes por paciente, hospital, tipo de consulta y fecha.
   - Tablas con filtros por columna.
   - Actualización automatizada del estado de las solicitudes en tiempo real.

- **Funcionalidades que no están incluidas en el alcance del proyecto:**
    - Login / Registro de usuarios.
    - Asignación de roles y permisos (Paciente, Especialista, Administrador, otros).
    - Pruebas exhaustivas para garantizar un control adecuado en el manejo de errores y excepciones del sistema.

6. Tecnologías y Herramientas

- **Lista de las tecnologías y herramientas que se utilizarán en el desarrollo:**
    - Python/Flask
    - Vue.js 3
    - RestAPI
    - MySQL
    - HTML, JavaScript
    - Bootstrap
    - Git
    - GitHub (Repositorio)
    - APIs (Json)
    - Environ
- **Explicación del uso de tecnologías mencionadas y requeridas para el proyecto:**
    - **Python/Flask (Backend)**:
        1. Python es un lenguaje de programación versátil y poderoso, con una gran comunidad de desarrolladores y una amplia gama de bibliotecas y frameworks. Es conocido por su simplicidad, legibilidad y facilidad de uso.
        2. Flask es un microframework minimalista para Python, ideal para proyectos ágiles. Permite construir aplicaciones web rápidamente, proporcionando únicamente las herramientas esenciales necesarias, y dejando que el desarrollador tenga libertad sobre las decisiones de arquitectura.
        3. Flask es altamente escalable y modular, lo que facilita la adición y modificación de funcionalidades a medida que el proyecto crece, adaptándose bien tanto a proyectos pequeños como grandes.
        4. En este proyecto, Flask se utiliza principalmente para manejar la lógica del backend, el enrutamiento y las conexiones con la base de datos.
        5. Flask también facilita la creación de APIs que conectan el backend con el frontend, utilizando JSON para el intercambio de datos y gestionando la comunicación entre ambos lados de la aplicación.
    - **Vue.js 3 (Frontend)**:
        1. Vue.js es un framework progresivo de JavaScript utilizado para construir interfaces de usuario interactivas y dinámicas. Es fácil

de integrar en proyectos que ya utilizan HTML y JavaScript, y es altamente flexible y eficiente.
2. Vue.js 3 es la última versión del framework y trae mejoras en rendimiento, características avanzadas como Composition API, y una arquitectura más optimizada para aplicaciones a gran escala.
3. Vue.js permite el desarrollo de aplicaciones web modernas y reactivas, donde la actualización de la interfaz se realiza de manera eficiente sin recargar la página. Su sistema de componentes facilita la reutilización de código y la organización del frontend en unidades modulares.
4. En este proyecto, Vue.js se utiliza para crear una experiencia de usuario fluida y dinámica, manejando las vistas y la interacción con el backend de Flask a través de RestAPI.

- **MySQL (Base de Datos)**:

1. MySQL es un sistema de gestión de bases de datos relacional de código abierto, conocido por su fiabilidad, robustez y flexibilidad. Es ampliamente utilizado en aplicaciones web y se integra fácilmente con múltiples lenguajes de programación, incluyendo Python.
2. MySQL permite almacenar, gestionar y consultar grandes volúmenes de datos de manera eficiente. Es ideal para proyectos que requieren transacciones y alta consistencia de datos.
3. En este proyecto, MySQL se encarga de almacenar toda la información relacionada con los pacientes, especialistas, hospitales y las consultas generadas, garantizando la integridad y seguridad de los datos.
4. A través de la conexión con Flask, se realizarán operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para gestionar la información del sistema de manera centralizada.

- **Git**:

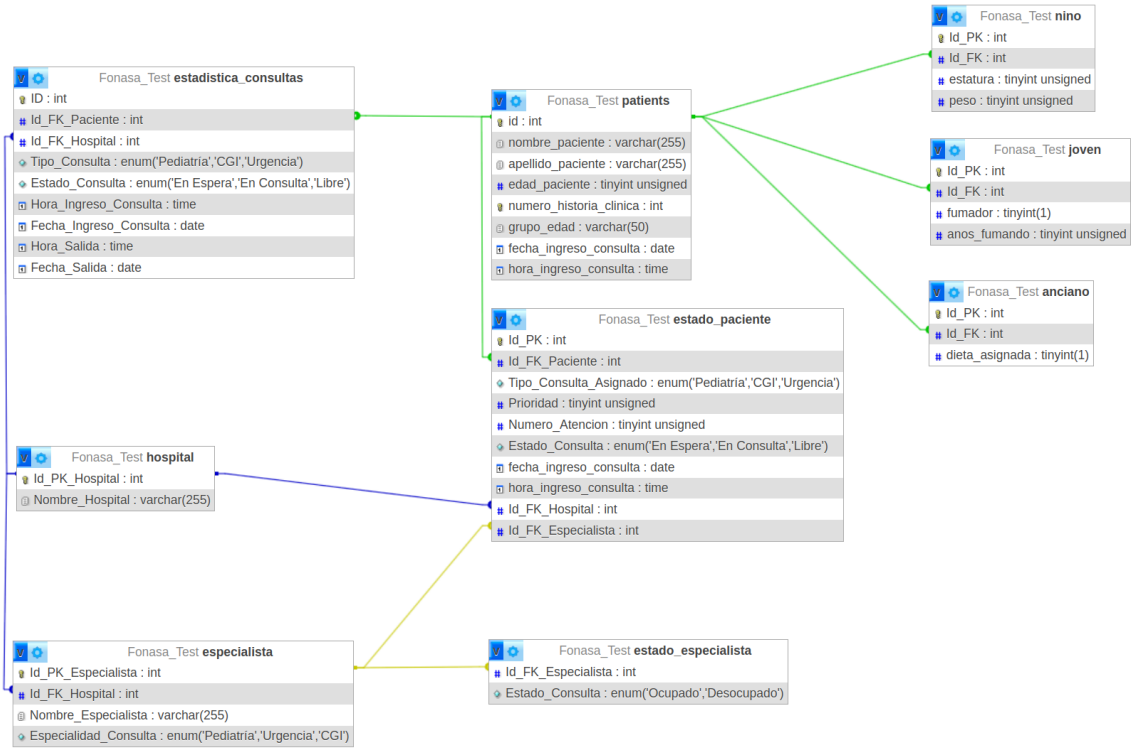
1. Git es un sistema de control de versiones distribuido que permite un seguimiento eficiente de los cambios en el código fuente y la colaboración entre equipos de desarrollo.
2. Facilita la gestión de ramas de desarrollo, fusiones de código y resolución de conflictos, lo que contribuye a una integración continua fluida y una entrega continua.
3. Su arquitectura descentralizada permite a los desarrolladores trabajar de forma independiente y colaborar en proyectos de manera distribuida sin depender de una conexión a Internet.

4. Git proporciona un historial completo de cambios en el código fuente, lo que facilita la auditoría y la reversión de cambios en caso de problemas.
5. Plataformas de alojamiento de código como GitHub ofrecen funcionalidades adicionales, como seguimiento de problemas, integración con herramientas de construcción y despliegue, y colaboración en proyectos de código abierto.
6. La popularidad y el soporte comunitario de Git y GitHub lo convierten en una opción confiable para la gestión de versiones y la colaboración en proyectos de software.

- **Environ**:

1. Su integración con frameworks y bibliotecas de Python, como Django y Flask, simplifica la gestión de la configuración y la segregación de entornos de desarrollo, prueba y producción.
2. Environ es una biblioteca de Python que simplifica la gestión flexible, segura y clara de variables de entorno en aplicaciones web, mejorando así la portabilidad, seguridad y mantenibilidad de la aplicación
3. Su compatibilidad con la convención de archivos .env y su uso de convenciones de nomenclatura estándar mejoran la legibilidad y la mantenibilidad del código.
4. Environ ofrece características adicionales, como la validación de tipos de datos y la gestión de valores predeterminados, que mejoran la robustez y la confiabilidad de la aplicación.

7. Tablas MySql del Proyecto



8. Estructura del Proyecto

- **Base Proyecto.**

```
├── docker-compose.yml
├── Fonasa_Inexoos_Flask **(Backend)**
│   ├── app.py
│   ├── data.csv
│   ├── Dockerfile
│   ├── env
│   ├── estadistica.py
│   ├── fonasa_data.sql
│   ├── fonasa.sql
│   ├── lista_espera.py
│   ├── patient_resource.py
│   ├── requirements.txt
│   └── solicitar_consulta.py
└── fonasa_inexoos_vue **(Frontend)**
    ├── babel.config.js
    ├── Dockerfile
    ├── jsconfig.json
    ├── node_modules
    ├── package.json
    ├── package-lock.json
    ├── public
    ├── README.md
    ├── src
    └── vue.config.js
```

- **Backend.**

```
Fonasa_Inexoos_Flask
├── app.py
├── Dockerfile
├── env **(Ambiente Virtual)**
├── .env **(Archivo para manejo información sensible)**
├── estadistica.py
├── fonasa_data.sql
├── fonasa.sql
├── lista_espera.py
├── patient_resource.py
├── requirements.txt
└── solicitar_consulta.py
```

- **Frontend.**

```
fonasa_inexoos_vue
├── node_modules ## Carpeta
├── babel.config.js
├── Dockerfile
├── jsconfig.json
├── package.json
├── package-lock.json
├── public
│   ├── favicon.ico
│   └── index.html
├── README.md
├── src ## (Carpeta con Archivos Proyecto)
│   ├── App.vue
│   ├── assets
│   ├── components
│   ├── main.js
│   ├── resources
│   ├── router
│   └── views
└── vue.config.js
```

Frontend (carpeta 'fonasa_inexoos_vue/src')

```
.
├── App.vue
├── assets
│   └── logo.png
├── components
│   ├── ExistingPatient.vue --> Archivo Proyecto
│   ├── HelloWorld.vue
│   └── NewPatient.vue --> Archivo Proyecto
├── main.js
├── resources --> Carpeta Estilos (Bootstrap)
│   └── css
├── router --> Carpeta archivo Enrutamiento
│   └── index.js
└── views
    ├── ConsultaView.vue  --> Archivo Proyecto
    ├── EstadisticaView.vue  --> Archivo Proyecto
    ├── HomeView.vue
    ├── ListaEsperaView.vue  --> Archivo Proyecto
    ├── PatientView.vue --> Archivo Proyecto
    └── TableView.vue --> Archivo Proyecto
```

09. Sobre Archivos del Proyecto

- **Archivos Frontend & Backend**
- app.py & patient_resource.py & TableView.vue:
    - Archivos asociados con los datos del paciente donde permite ver, editar y eliminar datos del paciente
- solicitar_consulta.py & ConsultaView.vue & ExistingPatient.vue & NewPatient.vue:
    - Permite Crear Paciente si este no existe
    - Permite hacer solicitud de atención
- estadistica.py & EstadisticaView.vue:
    - Visualizar cantidad de pacientes atendidos filtrado por fecha, tipo consulta y hospital
    - Se visualiza un gráfico (chart.js)
- lista_espera.py & ListaEsperaView.vue:
    - Clasifica el tipo de atención por cliente
    - Entrega un orden y prioridad de atención
    - Permite visualizar si un paciente esta en espera, en consulta o libre
    - Permite visualizar si un especialista está ocupado o desocupado
    - Permite indicar cuando un paciente está siendo atendido
- requirements.txt:
    - Archivos con las librerías que se ocupan en backend
- **Base Datos MySQL**
- fonasa.sql:
    - Creación de tablas que se ocuparán en el proyecto

- fonasa_data.sql:
    - Contiene dummy data (datos falsos) para incluir en el proyecto y visualizar este con datos