SESAMO

Aplicación Web de Formación en el Uso de “Mambrino”

(Con implementación de API)

2 DAW 2023-2024
Jhoel alexander Narváez Valarezo

2 DAW 2023-2024

Enlace Principal: https://github.com/jhax311/SESAMO-TFG

Enlace Web: https://sesamo.sytes.net

Enlace Memoria:

https://github.com/jhax311/SESAMO-TFG/blob/main/ TFG_JhoelNarvaez.pdf

https://github.com/jhax311/SESAMO-TFG/

Enlace Manual Aplicación:

https://github.com/jhax311/SESAMO-TFG/blob/main/ManualDeUso_SESAMO.pdf

Enlace Manual API: https://sesamo.sytes.net/sesamo

Usuarios:

- Administrador:
  - Usuarios: admin
  - Contraseña: admin
- Usuario administrativo
  - Usuario: usuario
  - Contraseña: usuario

Scripts para iniciar base de datos:

https://github.com/jhax311/SESAMO-TFG/blob/main/ConfiguracionBdd/iniciarBaseDatos.php

Contenido

1. INTRODUCCIÓN Y JUSTIFICACIÓN DEL PROYECTO

En el entorno sanitario, la eficiencia en la gestión de recursos y la calidad de atención al paciente son aspectos fundamentales que impactan directamente en la salud pública. Los sistemas de gestión hospitalaria juegan un papel crucial en la automatización y optimización de procesos en hospitales y centros de salud, cubriendo áreas como la admisión de pacientes, la gestión de historias clínicas y la administración de recursos. Sin embargo, una de las principales barreras para su implementación efectiva es la falta de formación adecuada del personal que utiliza estos sistemas.

En este contexto, el proyecto propone el desarrollo de una plataforma educativa que nace como un modelo de gemelo digital de la aplicación “Mambrino”, utilizada en Castilla-La Mancha, con esto se busca abordar la necesidad formativa mediante la replicación más fiel posible de las funciones principales de “Mambrino”.

El objetivo de este proyecto es desarrollar una aplicación educativa destinada a estudiantes de sanidad que les permita adquirir habilidades prácticas en la gestión hospitalaria y en el uso de la aplicación de Mambrino. Mediante esta plataforma, los usuarios podrán familiarizarse con los distintos procedimientos importantes con el fin de superar el choque entre los programas usados en el entorno académico y la realidad del entono laboral. La aplicación promueve una transición fluida y efectiva hacia el ámbito profesional.

La unificación del formato de la historia clínica facilita compartir datos entre todos los centros asistenciales, haciendo posible que un paciente pueda ser atendido en cualquier centro hospitalario de la Comunidad Autónoma con todas las garantías de que sus datos clínicos están accesibles. A pesar de su importancia, muchos profesionales de la salud no están adecuadamente formados en el uso de estas herramientas debido a su complejidad y al tiempo necesario para aprender a utilizarlas correctamente. La falta de formación en este ámbito puede resultar en ineficiencias y errores que afectan negativamente la calidad del servicio sanitario.

La implementación de soluciones tecnológicas en el sector sanitario es vital para mejorar la calidad de la atención y la eficiencia de los servicios de salud. El proyecto no solo mejorará su preparación profesional, sino que también contribuirá a la reducción de errores y al aumento de la eficiencia en los servicios sanitarios.

Con la implementación del proyecto, se espera mejorar significativamente la preparación de los futuros profesionales en sanidad y administración para el uso de sistemas de gestión hospitalaria. La aplicación no solo facilitará el aprendizaje de Mambrino, sino que también establecerá una base sólida para la adaptación a otros sistemas similares, contribuyendo a una mayor eficiencia y calidad en la atención sanitaria.

2. DESCRIPCIÓN DEL PROYECTO

2.1 CARACTERÍSTICAS GENERALES

La característica principal es conseguir una plataforma educativa diseñada para estudiantes de sanidad, que actúa como gemelo digital de la aplicación Mambrino. Tiene como propósito principal ofrecer una herramienta de formación que permita adquirir habilidades prácticas en la gestión hospitalaria. Como gemelo digital replica las principales funciones de Mambrino, permitiendo a los usuarios explorar y aprender aspectos clave en la formación del mismo. El proyecto está compuesto por dos partes esenciales: una API en el Back End encargada de la gestión de datos y una interfaz de cliente que explota la API para proporcionar toda la funcionalidad a la aplicación.

2.2 OBJETIVOS Y ALCANCE

Objetivos Específicos:

- Minimizar las diferencias entre los programas utilizados en el entorno educativo y la aplicación sanitaria Mambrino.
- Diseñar e implementar una API que reproduzca las funcionalidades principales de Mambrino, asegurando la compatibilidad con los sistemas existentes o por existir, tanto de escritorio como web.
- Desarrollar una interfaz de cliente intuitiva y fácil de usar que permita a los estudiantes interactuar con el gemelo digital de Mambrino de manera similar a como lo harían en el entorno real.
- Evaluar la efectividad del gemelo digital como herramienta de aprendizaje, mediante pruebas de usabilidad y comparaciones con el uso de la aplicación Mambrino en un entorno profesional.

Alcance del proyecto:

Se pretende lograr los siguientes objetivos específicos, abarcando diversas etapas de desarrollo y evaluación:

1. Análisis comparativo:
    a. Identificar las diferencias y similitudes entre los programas educativos y Mambrino
2. Desarrollo Back End:
    a. Creación de una API que simule las funcionalidades clave de Mambrino.
    b. Asegurar la compatibilidad con sistemas educativos actuales y futuros.
3. Desarrollo del Front End:
    a. Diseño de una interfaz de usuario que sea intuitiva y que refleje la experiencia de uso de Mambrino.

b. Implementación de funcionalidades que permitan a los estudiantes realizar tareas comunes en la gestión hospitalaria.

4. Pruebas y evaluación:
    a. Realización de pruebas de usabilidad con estudiantes para evaluar la efectividad de la plataforma.
    b. Comparación de los resultados de las pruebas con el uso real de Mambrino en un entorno profesional para medir la eficacia del aprendizaje.

Limitaciones del proyecto:

1. Alcance temporal: el proyecto está limitado por el tiempo disponible para su desarrollo y evaluación dentro del periodo académico.
2. Adaptabilidad: aunque se ha diseñado para ser lo más fiel a Mambrino, pueden existir limitaciones en la replicación exacta de todas sus funcionalidades debido a restricciones tecnológicas o de acceso a la información.

3. ANÁLISIS DEL SECTOR

3.1 PREVISIONES FUTURAS DEL SECTOR

El sector del desarrollo web en cuenca, al igual que en el resto de España, esta experimentado un crecimiento significativo. Este crecimiento es impulsado por la digitalización de las empresas, la adopción de tecnologías emergentes y la creciente demanda de servicios en línea. Se espera que la demanda de desarrolladores web continúe en aumento, debido a la necesidad de sitios web dinámicos, aplicaciones web y comercio electrónico.

El ciclo formativo prepara a los estudiantes un entorno profesional dinámico y en constante evolución. Las funciones en las que desempeñaran pueden ser diversos roles en el sector tecnológico, como:

- Programador web (Front-End, Back-End).
- Programador multimedia.
- Desarrollador de aplicaciones en entornos web.

El sector de la tecnología ha mostrado una evolución positiva, reflejada en varios indicadores clave:

- Las empresas tecnologías han aumentado, lo que incluye una cantidad de pequeñas y medianas empresas dedicadas al desarrollo web.
- El empleo en el sector TIC en Castilla-La Mancha ha crecido, con un aumento en la demanda de profesionales cualificados en desarrollo web y tecnologías relacionadas.

3.2NORMATIVA Y DOCUMENTACIÓN

Para asegurar la accesibilidad y seguridad de aplicaciones web, es esencial seguir ciertas normativas y documentación técnica específica.

ACCESIBILIDAD

- WCAG: indicaciones desarrollas por World wide web consortion, con el fin de hacer el contenido web más accesible para personas con discapacidades.

PROTECCIÓN DE DATOS

- GDPR: reglamentos sobre la recopilación, procesamiento y almacenamiento de datos personales, garantizando la privacidad y protección de los datos de los usuarios.

DOCUMENTACIÓN TÉCNICA

- OWASP: una serie de guías y recursos para el desarrollo seguro de aplicaciones web, abordando aspectos de seguridad y protección contra vulnerabilidades comunes.
- ISO 27001: Esta norma internacional proporciona los requisitos para establecer, implementar, mantener y mejorar un sistema de gestión de la seguridad de la información

4. PLAN DE EJECUCIÓN

4.1 DIAGRAMA DE GANTT

El plan de ejecución se organizará en varias fases que se distribuirán en el periodo de desarrollo del proyecto. Se utilizará un diagrama de Gantt para visualizar el cronograma y gestionar el progreso de las tareas.

4.2 PROCESO DE DESARROLLO SOFTWARE

FASE DE ANÁLISIS

TIPOS DE USUARIO

En esta fase se identificarán los diferentes tipos de usuarios que interactuarán con la aplicación.

- Profesor: Usuario con permisos para supervisar, evaluar y gestionar el uso de la aplicación por parte de los estudiantes.
- TCAE (Técnico en Cuidados Auxiliares de Enfermería): Usuario que simulará la realización de tareas auxiliares de enfermería.
- Administrativo: Usuario que gestionará la admisión de pacientes y la administración de recursos humanos.
- Auxiliar de Farmacia y Parafarmacia: Usuario que realizará tareas relacionadas con la gestión de medicamentos y productos de parafarmacia.
- Técnico en Imagen para el Diagnóstico: Usuario que simulará la realización de pruebas de imagen diagnóstica.
- Técnico en Laboratorio Clínico y Biomédico: Usuario que gestionará y realizará análisis clínicos y biomédicos.
- Administrador: encargado de la gestión de usuarios de la aplicación.

Actualmente no hay distinción entre ellos, pero con la idea de una futura implementación. El usuario administrador en cambio el único capaz de borrar, modificar usuarios.

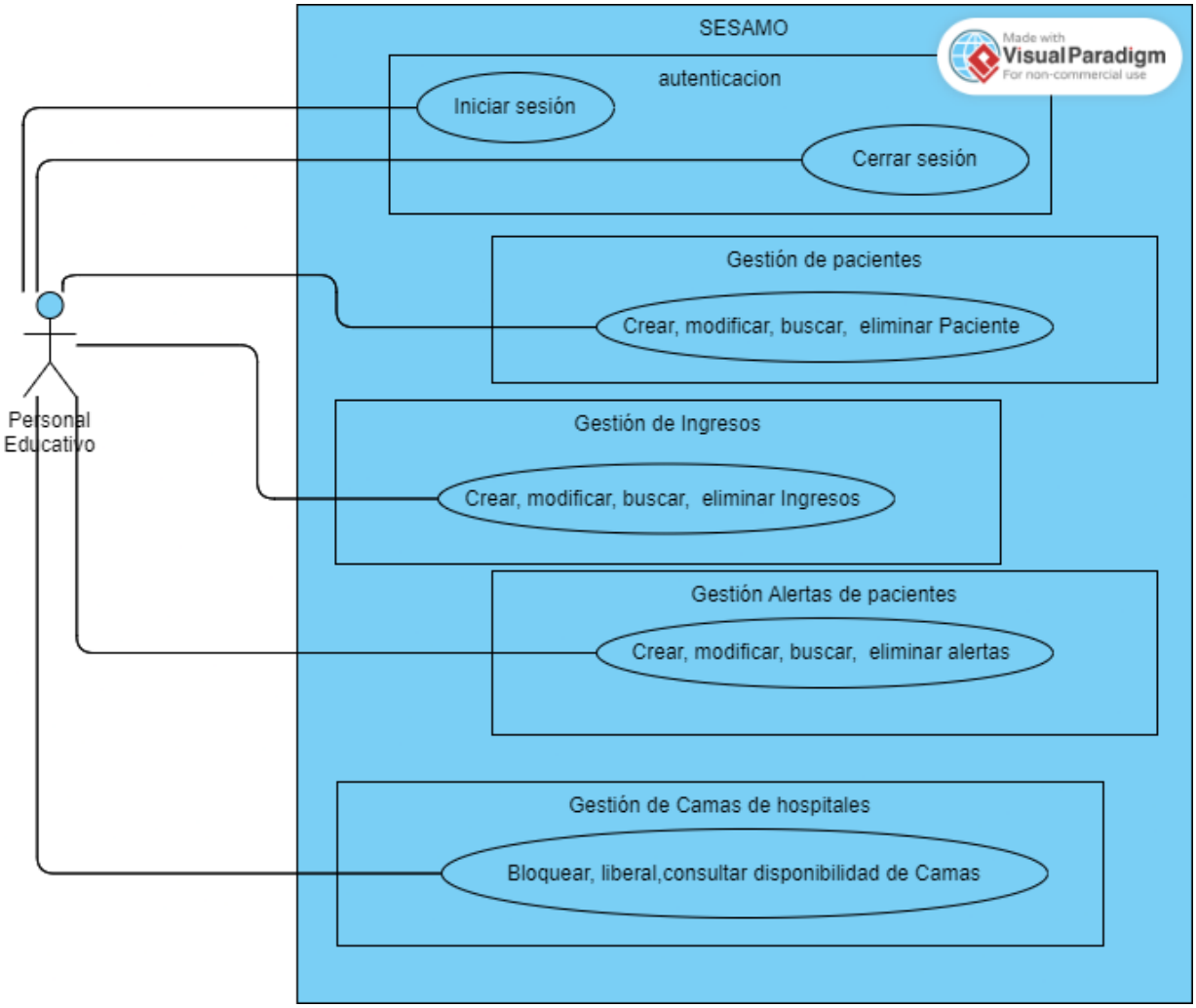
DESCRIPCIÓN DE REQUISITOS

1. Gestión de la admonición de pacientes:
    a. Capacidad de registro de nuevos pacientes en el sistema, incluyendo datos personales, historial médico, y motivos de ingreso.
    b. Seguimiento continuo de los pacientes durante su estancia en el centro de salud, incluyendo actualizaciones de alertas o cualquier cambio en su condición.
2. Manejo de historias clínicas:
    a. Acceso seguro y controlado a las historias clínicas de los pacientes, permitiendo a los usuarios ver y actualizar información médica.
    b. Capacidad para añadir nuevas entradas a las historias clínicas, como diagnósticos, tratamientos, y notas del personal médico.
3. Interfaz amigable e intuitiva:

a. Una interfaz de usuario diseñada para ser intuitiva y fácil de usar, con un enfoque en la experiencia del usuario para facilitar el aprendizaje.
b. Estructura de navegación clara y coherente que permita a los usuarios acceder rápidamente a las funciones principales.
c. La aplicación debe ser accesible desde diferentes dispositivos (escritorio, tabletas, y teléfonos móviles) y navegadores.

4. Reportes:
   a. Capacidad para generar reportes detallados sobre la admisión de pacientes, historial clínico, utilización de recursos, y otros aspectos relevantes.

CASOS DE USO



GUÍA DE ESTILO

La biblioteca principal usada para el desarrollo es Bootstrap.

Colores:

La paleta de colores se ha elegido para similar lo máximo posible a la aplicación original de Mambrino.

Iconos:

Se usarán iconos coherentes para representar acciones y funciones comunes, iconos claros y fácilmente reconocibles.

Formularios:

Para la implementación de formularios usaremos la librería de Angular Material UI, ya que vienen componentes predefinidos para implementarlos con la lógica.

Implementación como Single Page Application (SPA)

La aplicación será una SPA para mejorar la velocidad y la experiencia del usuario.

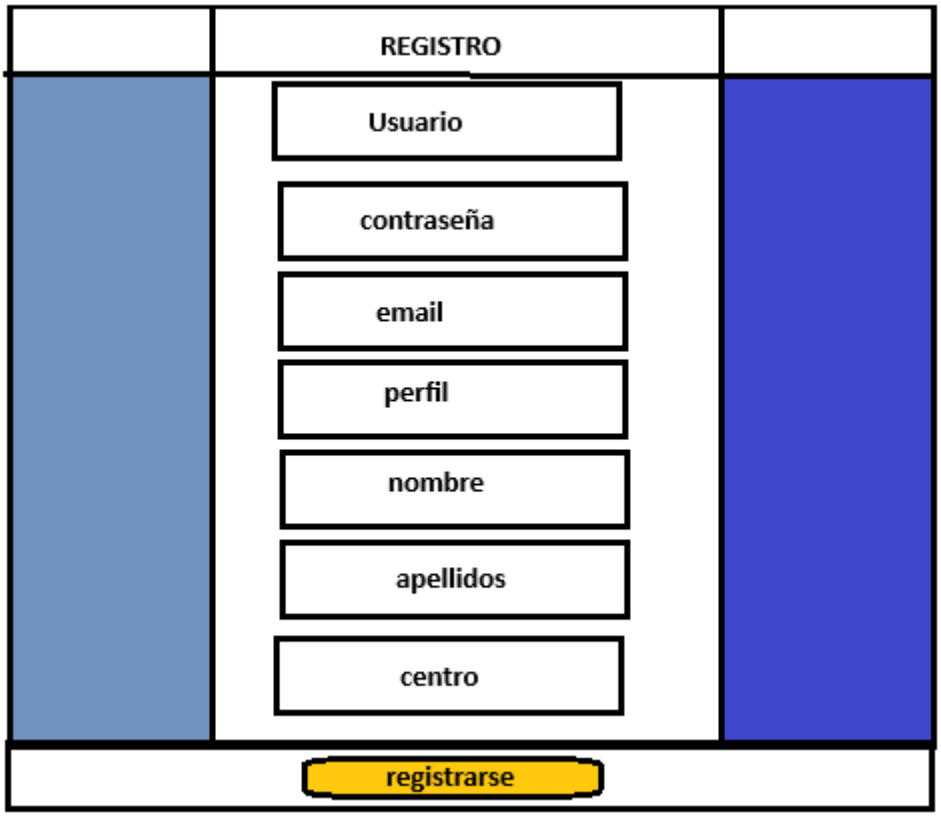
- Rutas y navegación:
  - Se implementa el enrutamiento en el lado del cliente
  - Cada sección de la aplicación se cargará dinámicamente sin recargar la pagina completa.
- Conexión con el servidor:
  - Comunicación asíncrona.

PROTOTIPO DEL SITIO WEB

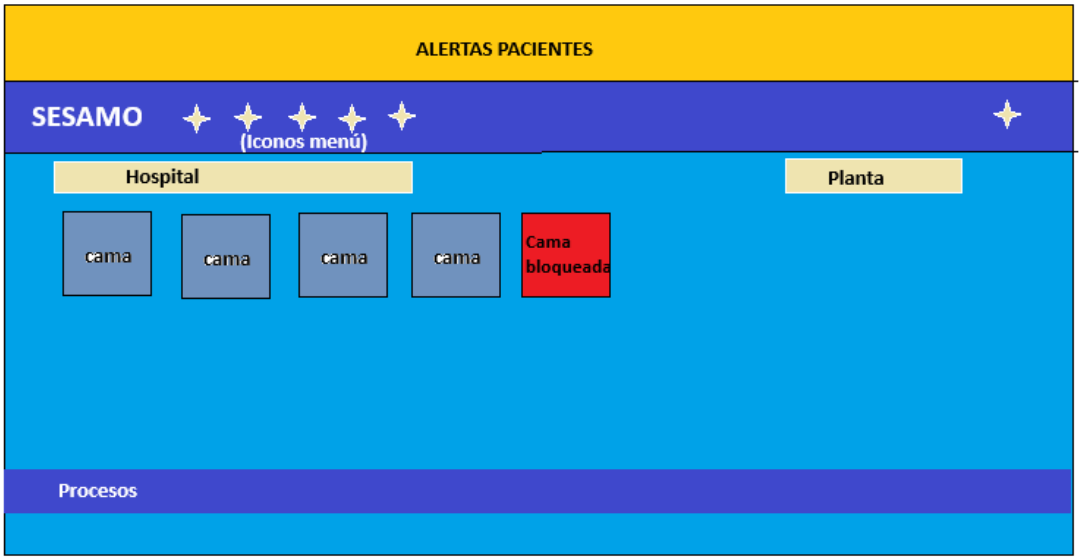
El prototipado de viene definido al diseño original de la aplicación Mambrino, con el fin de conseguir un gemelo digital lo más parecido posible.

INICIO DE SESIÓN

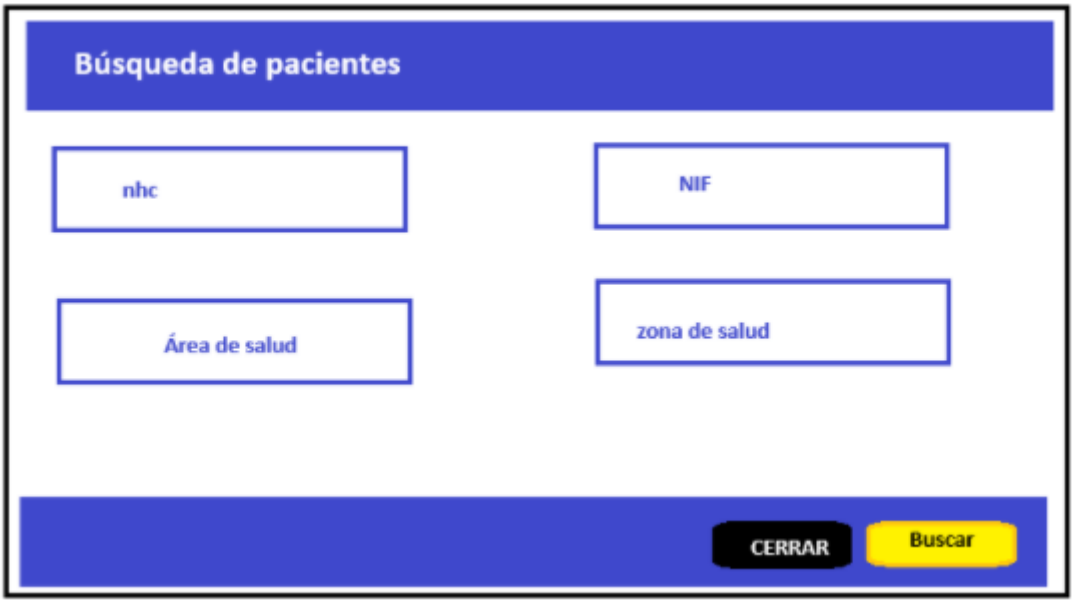
REGISTRO



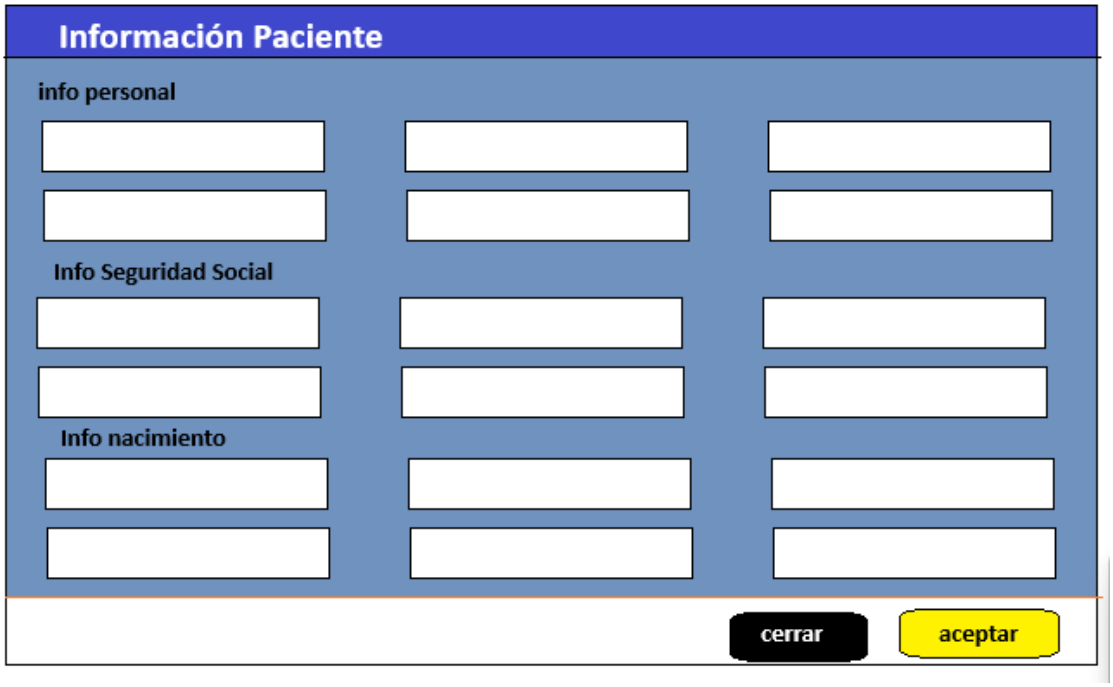
DASHBOARD



BUSCADOR DE PACIENTES

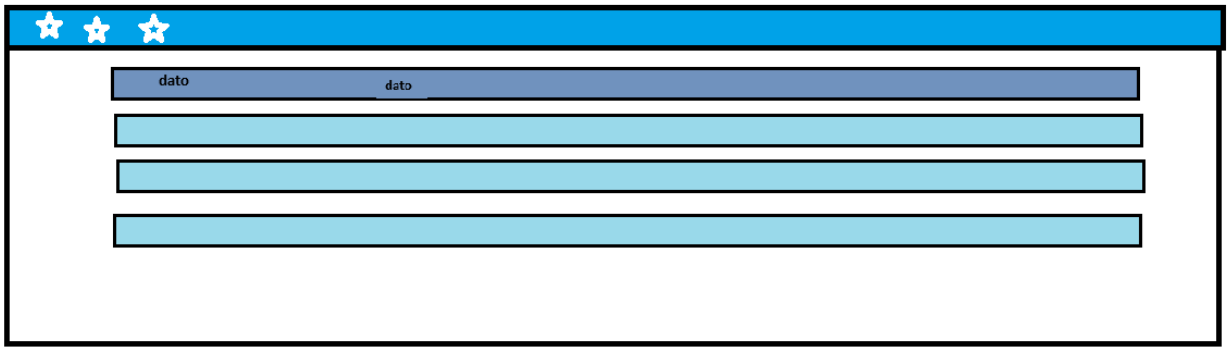


VER, MODIFICAR, AÑADIR PACIENTE

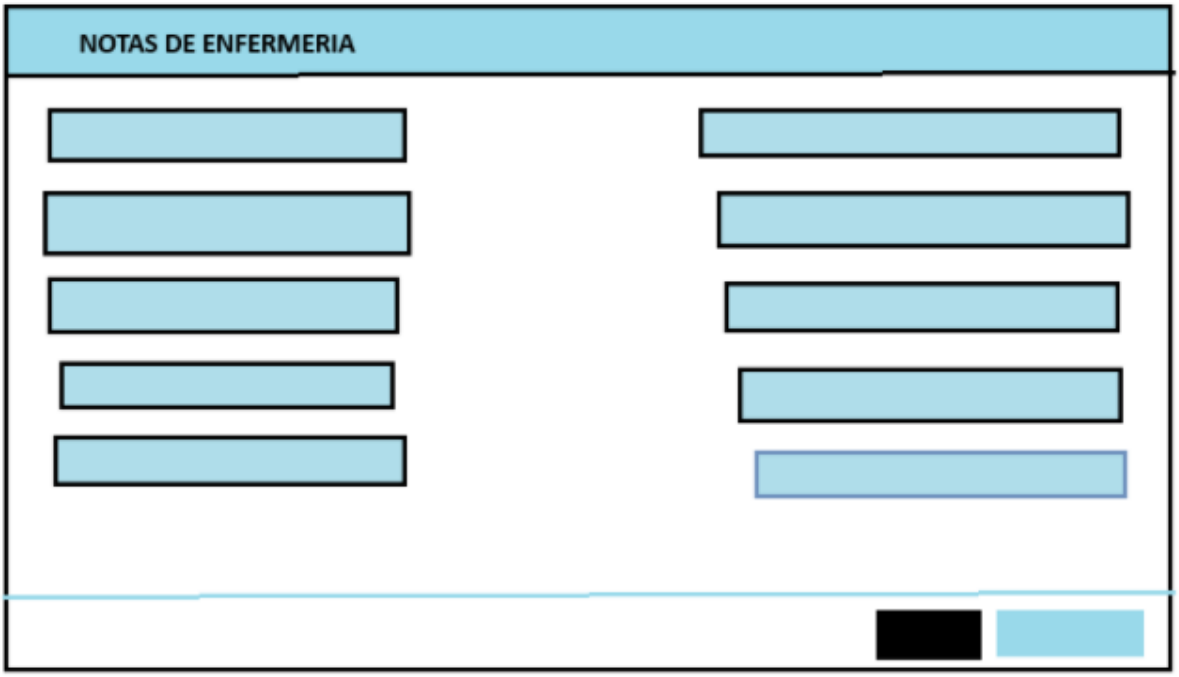


CREAR ALERTAS

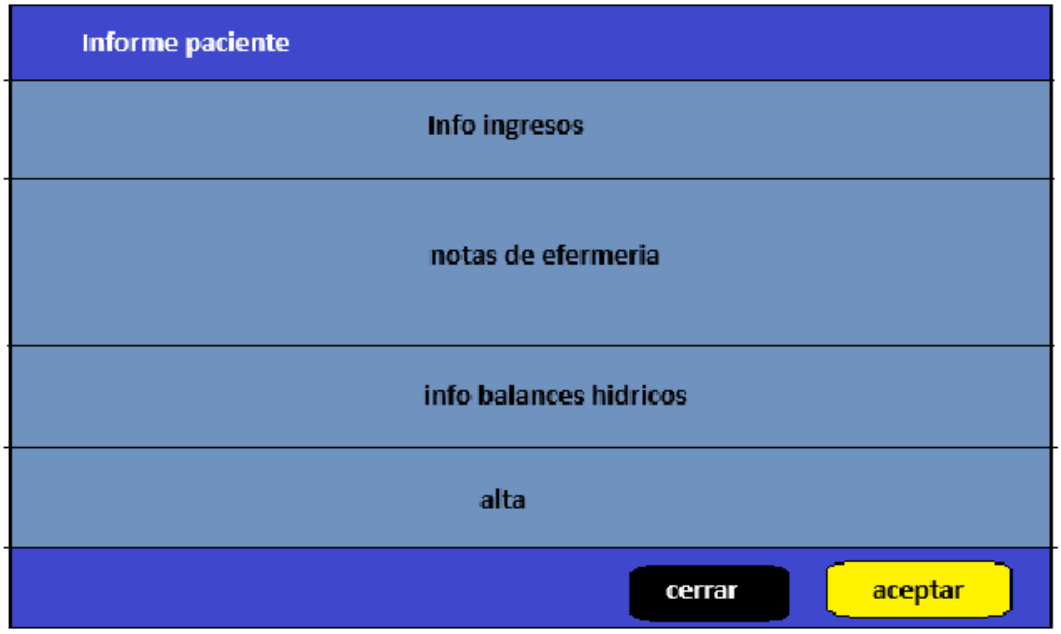
HOJA DE PRESCRIPCIÓN Y PATOLOGÍAS



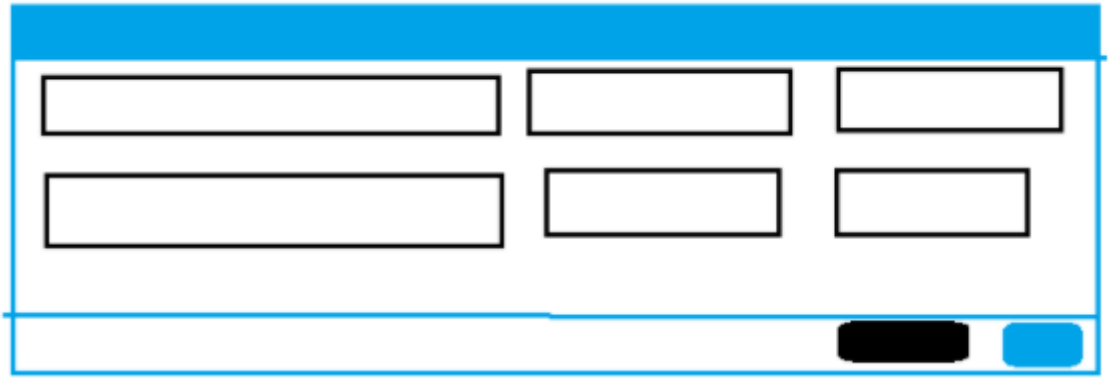
NOTAS DE ENFERMERÍA



VER INFORME

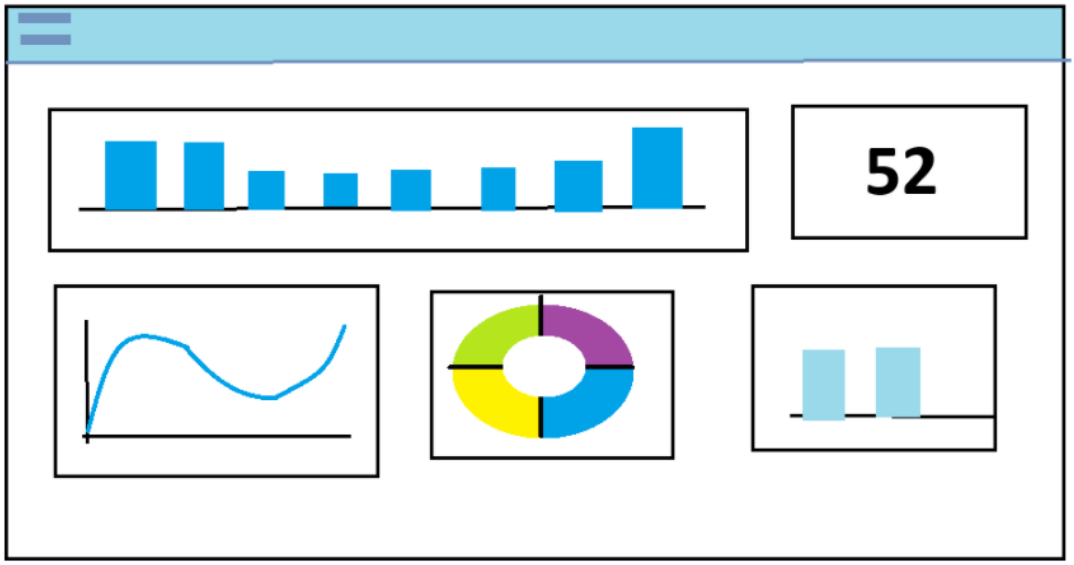


AÑADIR PATOLOGÍA Y HOJAS DE PRESCRIPCIÓN

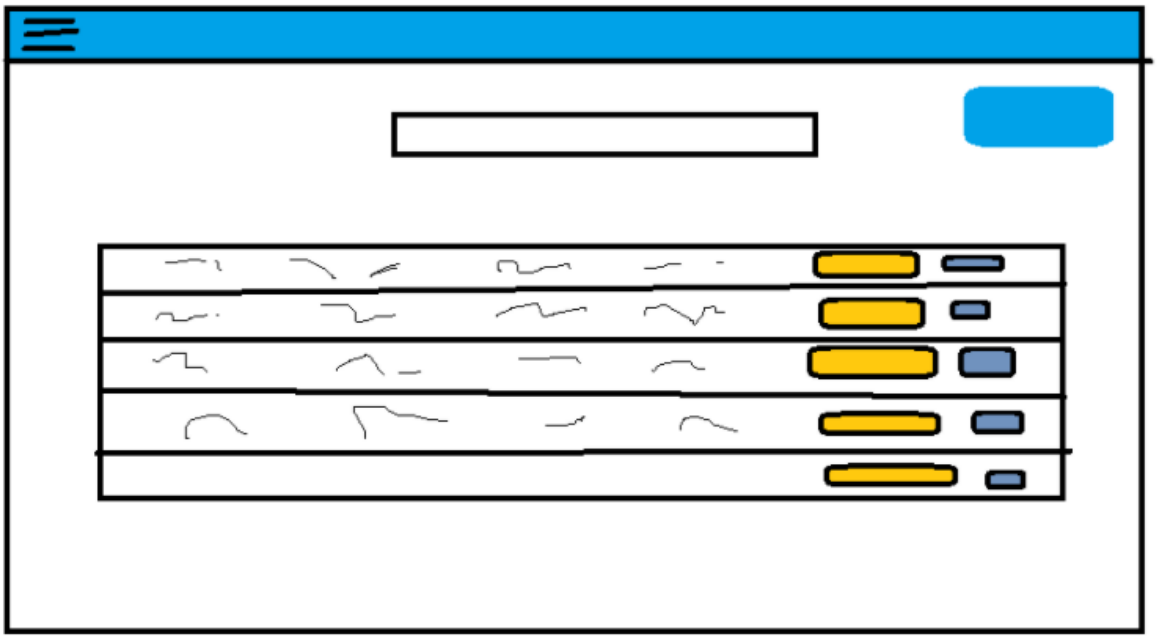


DASHBOARD ADMINISTRADOR

Mostrara estadísticas del uso de la API

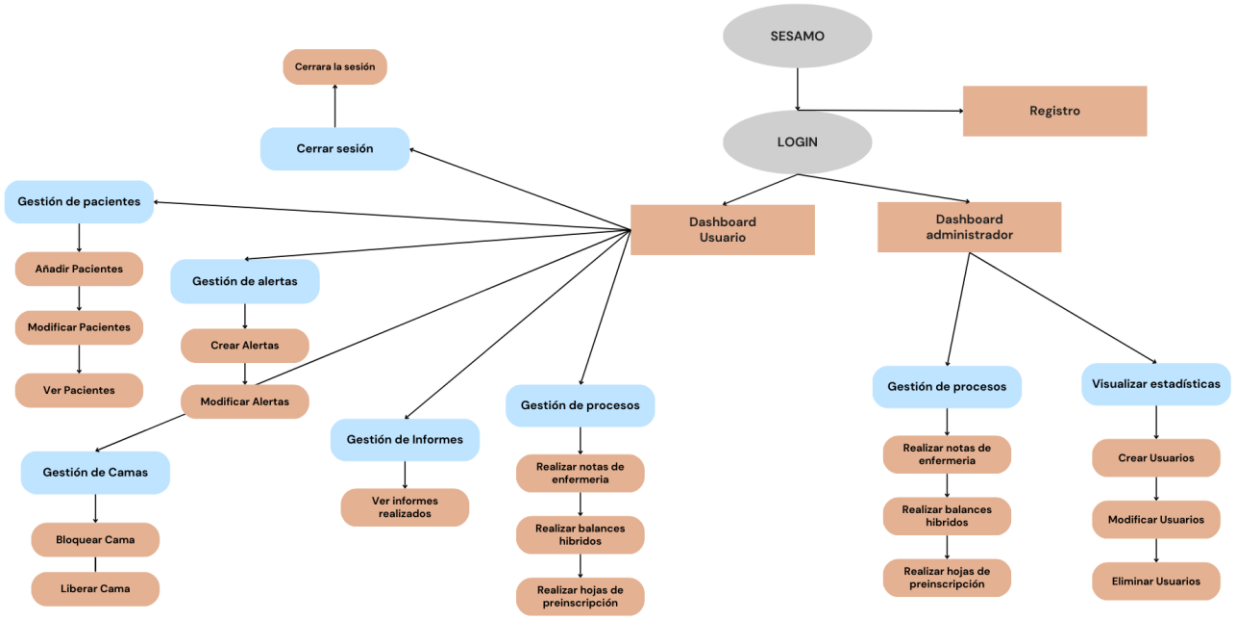


GESTIÓN USUARIOS (Dashboard Administrador)



MAPA DE NAVEGACIÓN

Según el rol del usuario podrá acceder a una parte u otra.



FASE DE DESARROLLO

La fase de desarrollo es una de las más críticas, durante esta fase de materializan las especificaciones y diseños definidos en las fases anteriores, de manera que la web diseñada se convierte en un sistema funcional.

En esta sección se detalla el proceso de desarrollo del proyecto, abarcando desde el análisis de requisitos hasta la implementación del código y la configuración de la infraestructura necesaria para su funcionamiento.

BASE DE DATOS

Partimos de una base de datos inicial que contine las entidades fundamentales necesarias para el funcionamiento básico de la aplicación. Esta base de datos fue previamente diseñada y utilizada en un proyecto anterior. Sin embargo, para satisfacer los requisitos específicos del proyecto, se han realizado una serie de modificaciones clave con el fin de expandir y adaptar la base de datos a las nuevas funcionalidades y necesidades del proyecto.

La base de datos tiene en cuenta las necesidades de aplicación original de Mambrino.

Para iniciar el diseño de la base de datos partimos de las tablas definidas, como pueden ser la tabla “usuarios”, que incluye todas las personas que tendrán acceso al sistema. Estos usuarios estarán asociados a un centro y categorizados bajo roles y ámbitos. Los ámbitos están asociados a diferentes zonas básicas conectadas a determinadas áreas de salud. Por lo que cada área de salud se relacionara con un determinado centro.

La tabla camas además de relacionarse con los usuarios y las áreas de salud, está conectada a la tabla centros con el fin de registrar la disponibilidad y estado de las camas de cada centro. La conexión de la misma con los ingresos y pacientes permiten registrar la admisión de un paciente en una cama especifica dentro de un centro a través de la tabla ingresos.

Los pacientes son la parte central de este sistema, ya que su gestión es el foco principal de la aplicación. Esta tabla está relacionada con varias otras: “ingresos” (para admisiones), “alta” (para almacenar los datos de alta de un paciente), “alertas” (para registrar alertas medicas de un paciente), “hoja_prescripcion” (gestiona las prescripciones médicas), “patologia” (almacena diagnósticos y síntomas) y “notas_enfermeria” (encargada de registrar las observaciones y tomas de datos realizadas por el personal de enfermería).

Para mejorar la funcionalidad se crearon dos tablas relacionadas entre sí, comunidades autónomas y provincias.

Partiendo de esta , se han implementado ciertos cambios con el fin de mejorar la funcionalidad de la aplicación web. Dado que el proyecto se desarrollará con una clara separación entre la parte del servidor y la parte del cliente, estos cambios has sido diseñados para optimizar tanto el rendimiento del servidor como la experiencia del usuario en el cliente.

Con el fin de tener un registro de los ingresos del paciente y de que estos no se eliminen al ser dado de alta al paciente, se agrega un nuevo campo que nos permite saber si el paciente se encuentra ingresado actualmente. De esta manera mantenemos los ingresos y cuando el paciente sea dado de alta solo tendremos que cambiar el valor de “ingresado”, además evitamos ingresar al paciente en camas distintas en el mismo periodo de tiempo.

Para mejorar la seguridad y gestión de usuarios de nuestra base de datos, se implementa la tabla “recuperarContraseña”, esta tabla diseñada para gestionar el proceso de recuperación de contraseñas, permite a los usuarios restablecer sus contraseñas de forma segura mediante la generación y verificación de tokens temporales. La tabla guarda el identificador del usuario que solicita la recuperación, un token único generado para el proceso y la expiración del mismo con el fin de evitar que sea reutilizado después de un tiempo específico.

Ejemplo:

id_usuario	token	expiracion
1	PYqjj565s	2024-06-10 13:18:36

Tabla “usuarios_token” se ha implementado para gestionar los tokens de autenticación de los usuarios. Estos tokens se utilizan para mantener sesiones seguras y verificar la identidad de los usuarios en cada interacción con el sistema.

La tabla almacena los tokens de sesión generados para los usuarios autenticados. Cada vez que un usuario inicia sesión, se genera un nuevo token y se almacena en esta tabla. Este token se envía al cliente y se utiliza para autenticar al usuario en futuras solicitudes sin necesidad de volver a ingresar sus credenciales. La columna Estado permite controlar la validez de los tokens, asegurando que los tokens expirados o revocados no puedan ser utilizados.

Ejemplo:

TokenId	UsuarioId	Token	Estado	fecha	Id_rol

1	1	Wertyuidsdfghj68685fdddggdjgdf	1	2024-06-11	3

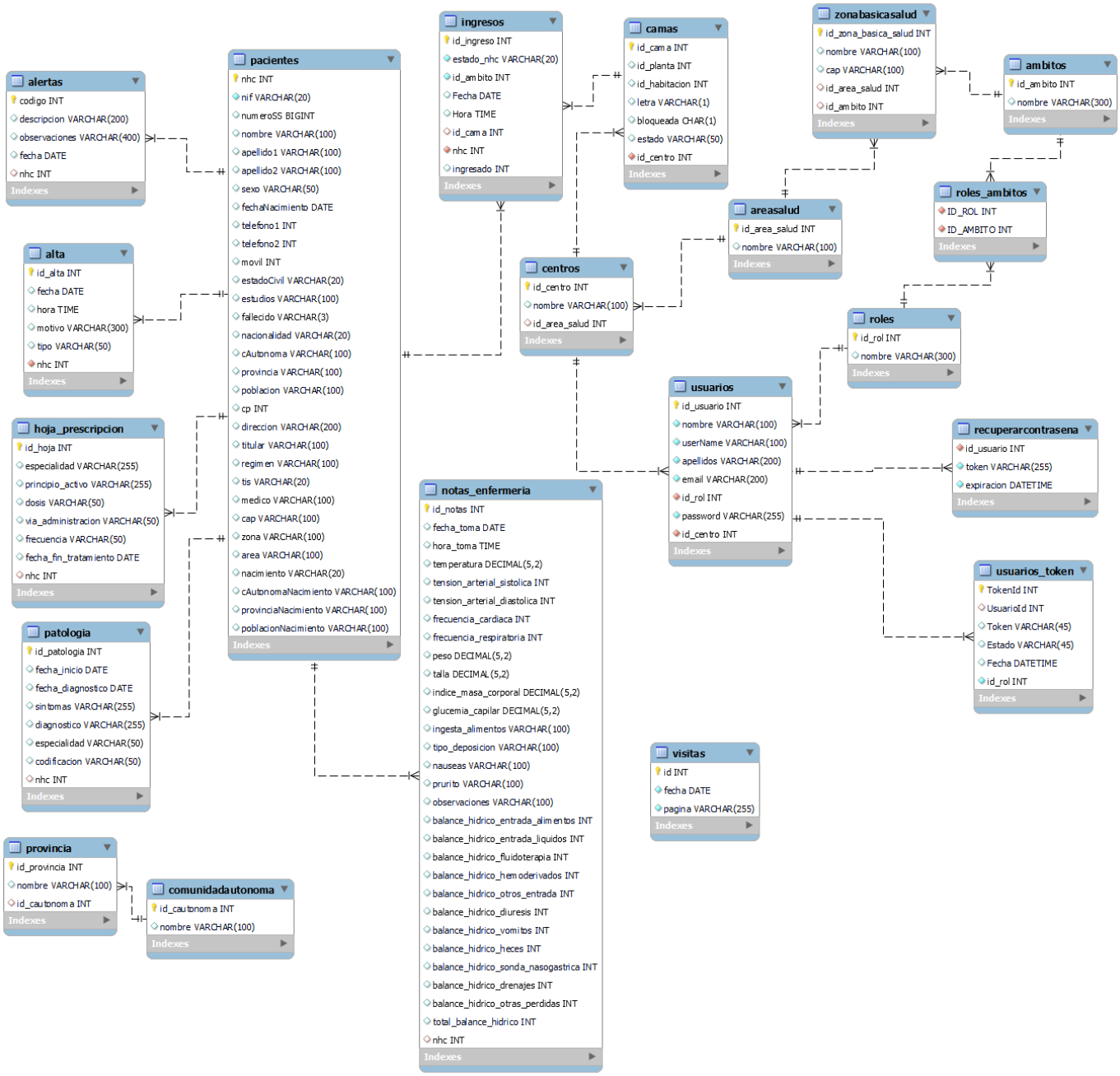
La implementación de estas tablas mejora significativamente la seguridad del sistema. Al implementar mecanismos de recuperación de contraseñas y gestión de tokens, se asegura que solo los usuarios autenticados puedan acceder al sistema y que puedan recuperar sus cuentas de manera segura.

La posibilidad de recuperar contraseñas y mantener sesiones autenticadas sin requerir credenciales en cada interacción mejora la experiencia del usuario, haciéndola más fluida y menos propensa a interrupciones.

Estos cambios reflejan un enfoque integral para mejorar tanto la seguridad como la funcionalidad de la aplicación web, garantizando una experiencia de usuario segura y eficiente.

También se agrega la tabla “visitas” con el fin de elaborar estadísticas del uso de la aplicación.

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL

(pk): claves primarias, (fk): claves foráneas y dirección

alertas (codigo (PK), descripcion, observaciones, fecha, nhc (FK hacia pacientes.nhc))

alta (id_alta (PK), fecha, hora, motivo, tipo, nhc (FK hacia pacientes.nhc))

ambitos (id_ambito (PK), nombre)

areaSalud (id_area_salud (PK), nombre)

camas (id_cama (PK), id_planta, id_habitacion, letra, bloqueada, estado, id_centro (FK hacia centros.id_centro))

centros (id_centro (PK), nombre, id_area_salud (FK hacia areaSalud.id_area_salud))

comunidadautonoma (id_cautonoma (PK), nombre)

hoja_prescripcion (id_hoja (PK), especialidad, principio_activo, dosis, via_administracion, frecuencia, fecha_fin_tratamiento, nhc (FK hacia pacientes.nhc))

ingresos (id_ingreso (PK), estado_nhc, id_ambito (FK hacia ambitos.id_ambito), Fecha, Hora, id_cama (FK hacia camas.id_cama), nhc (FK hacia pacientes.nhc), ingresado)

notas_enfermeria (id_notas (PK), fecha_toma, hora_toma, temperatura, tension_arterial_sistolica, tension_arterial_diastolica, frecuencia_cardiaca, frecuencia_respiratoria, peso, talla, indice_masa_corporal, glucemia_capilar, ingesta_alimentos, tipo_deposicion, nauseas, prurito, observaciones, balance_hidrico_entrada_alimentos, balance_hidrico_entrada_liquidos, balance_hidrico_fluidoterapia, balance_hidrico_hemoderivados, balance_hidrico_otros_entrada, balance_hidrico_diuresis, balance_hidrico_vomitos, balance_hidrico_heces, balance_hidrico_sonda_nasogastrica, balance_hidrico_drenajes, balance_hidrico_otras_perdidas, total_balance_hidrico, nhc (FK hacia pacientes.nhc))

pacientes (nhc (PK), nif, numeroSS, nombre, apellido1, apellido2, sexo, fechaNacimiento, telefono1, telefono2, movil, estadoCivil, estudios, fallecido, nacionalidad, cAutonoma, provincia, poblacion, cp, direccion, titular, regimen, tis, medico, cap, zona, area, nacimiento, cAutonomaNacimiento, provinciaNacimiento, poblacionNacimiento)

patologia (id_patologia (PK), fecha_inicio, fecha_diagnostico, sintomas, diagnostico, especialidad, codificacion, nhc (FK hacia pacientes.nhc))

provincia (id_provincia (PK), nombre, id_cautonoma (FK hacia comunidadautonoma.id_cautonoma))

recuperarContrasena (id_usuario (FK hacia usuarios.id_usuario), token, expiracion)

roles (id_rol (PK), nombre)

roles_ambitos (ID_ROL (FK hacia roles.id_rol), ID_AMBITO (FK hacia ambitos.id_ambito))

usuarios (id_usuario (PK), nombre, userName, apellidos, email, id_rol (FK hacia roles.id_rol), password, id_centro (FK hacia centros.id_centro))

usuarios_token (TokenId (PK), UsuarioId (FK hacia usuarios.id_usuario), Token, Estado, Fecha, id_rol (FK hacia roles.id_rol))

visitas (id (PK), fecha, pagina)

TIPOS DE DATOS

Int: para identificadores numéricos y claves primarias.

Varchar: Para cadenas de texto de longitud variable

Date: para fechas

Time: para horas

Char: para caracteres únicos

Decimal: para valores numéricos con decimales

SCRIPTS DE CREACIÓN DE TABLAS

Para crear las tablas se utilizan comando "CREATE TABLE IF NO EXISTS" junto con las definiciones de columnas y restricciones de claves primarias y foráneas.

Nada más inicializar la aplicación, se comprueba si existe la base de datos establecida en un fichero de configuración, en caso de no existir la crea y se ejecuta un script base de todas las tablas necesarias para el funcionamiento de la aplicación.

SERVIDOR

La lógica del servidor se implementa en PHP, necesitamos configurar el entorno PHP y luego desarrollar el backend utilizando PHP y MySQL para manejar la lógica de la base de datos.

Para la parte del servidor de la aplicación se divide en dos bloques, el primero se encarga de recibir las solicitudes HTTP (verificando si son admitidas, por ejemplo, la tabla comunidades solo permite solicitudes GET) y la parte que interactúa con la base de datos, estas instanciadas en clases diseñadas para realizar operaciones CRUD (crear, leer, actualizar, eliminar). Estas clases gestionan las acciones de los usuarios sobre las distintas entidades definidas en la base de datos.

A continuación, se describe la estructura básica del proyecto y las principales funciones que se desarrollarán para cada entidad.

ESTRUCTURA DEL PROYECTO:

/sesamo

|-- **clases**

| |-- **conexion**

| |-- conexion.php

| |-- config

| |-- alertas.class.php

| |-- ambitos.class.php

| |-- areaSalud.class.php

| |-- camas.class.php

| |-- centros.class.php

| |-- comunidades.class.php

| |-- ingresos.class.php

| |-- pacientes.class.php

| |-- recuperarPassword.class.php

| |-- visitas.class.php

| |-- perfil.class.php

| |-- provincias.class.php

| |-- respuestas.class.php

| |-- usuarios.class.php

| |-- zonas.class.php

|-- error

| -- error403.json

| -- error404.json

|-- alertas.php

|-- ambitos.php

|-- areaSalud.php

|-- camas.php

|-- centros.php

|-- comunidades.php

|-- index.php

|-- ingresos.php

|-- pacientes.php

|-- perfil.php

|-- provincias.php

|-- README.md

|-- usuarios.php

|-- recuperarPassword.php

|-- visitas.php

|-- zonas.php

FUNCIONES PHP A DESARROLLAR

- **Recursos identificados por URIs:**
    - Cada recurso (entidad, como usuarios, pacientes, etc.) se identifica de manera única mediante una URI. Por ejemplo, /sesamo/pacientes/nhc/1 podría ser la URI para solicitar información sobre el paciente con NHC 1.
- **Operaciones CRUD utilizando métodos HTTP:**
    - **GET:** Obtener información sobre un recurso.
        - **Ejemplo:** "GET /sesamo/pacientes/" para listar todos los pacientes
    - **POST:** Crear un nuevo recurso.
        - **Ejemplo** "POST /sesamo/pacientes/" para crear un nuevo paciente.
    - **PUT:** Modificar un recurso.
        - **Ejemplo:** "PUT /sesamo/pacientes/" para actualizar la información de un paciente.

- **DELETE:** Eliminar un recurso
  - **Ejemplo:** "DELETE /sesamo/pacientes/" para eliminar un paciente.

- **Representación de los recursos:**
  - Los recursos se representarán en formato JSON gracias a su simplicidad y facilidad de uso en aplicaciones web.
  - Ejemplo: {
    "id_cama": 1,
    "id_planta": 1,
    "id_habitacion": 1,
    "letra": "A",
    "bloqueada": "N",
    "estado": "Ocupada",
    "id_centro": 10
    }
- **Solicitudes sin estado:**
  - La solicitud del cliente debe contener toda la información necesaria para que el servidor deba entender y procesar la solicitud.
  - El servidor no almacenará información sobre el estado del cliente entre las solicitudes.
- **Código de Estado HTTP:**
  - La aplicación usa códigos de estado HTTP para e indicar el resultado de las solicitudes.
  - 200 OK : La solicitud fue exitosa.
  - 204 OK: La solicitud fue exitosa pero no hay datos para devolver.
  - 400 error: La solicitud es incorrecta, datos mal enviados.
  - 401 error: La solicitud requiere autenticación.
  - 404 error: El recurso solicitado no fue encontrado.
  - 405 error: El método HTTP no está permitido para
- **Entidades:**
  - Cada entidad o recurso tiene su propio fichero que es el encargado de gestionar las peticiones HTTP recibidas por el cliente. También se encarga de pasar los resultados a JSON y responder con el código de estado de la petición.
    - Ejemplo de la estructura básica:
      if ($_SERVER["REQUEST_METHOD"] == "GET") {
      //solicitudes GET
      } else if ($_SERVER["REQUEST_METHOD"] == "POST") {

```
//solicitudes post
} else if ($_SERVER["REQUEST_METHOD"] == "PUT") {
//solicitudes put
}else if ($_SERVER["REQUEST_METHOD"] == "DELETE") {
//solicitudes delete
} else {
//si no es ninguno, error 405
}
```

- Se conectan con las clases de su propia entidad para las operaciones CRUD

- **Clases de entidades:**
  - Cada entidad del sistema tiene su propia clase que maneja las operaciones CRUD relacionadas con la base de datos.
  - GET: son selects de la entidad.
  - POST: son inserts en la entidad.
  - PUT: son modificaciones de la entidad.
  - DELETE: elimina entidades del recurso.

Como hemos comentado las entidades poseen sus ficheros de recepción de solicitudes HTTP y el fichero que hace las acciones CRUD con la base de datos, las funciones principales son las siguientes:

Funciones principales	
Conexión	Esta será la clase y función principal de la aplicación en el lado servidor, todas las clases de entidades heredan de ella, ya que es la encargada de realizar la conexión con la base de datos. Además es la encargada de crear la base de datos y las tablas en caso de no existir al momento de iniciar la aplicación.
GET	Se encargan de listar todos los datos relacionados sobre una entidad, si realizamos una solicitud GET a /pacientes nos devuelve toda la información sobre los mismo. Se considera información que no necesita un token para visualizarla.
POST	Se encarga de crear un nuevo recurso sobre la entidad, de manera que, si realizamos una petición POST a /pacientes, generamos un nuevo paciente en la tabla. Para esta petición se necesita la información del paciente que se deberá pasar en el body de la petición y tener un token el cual conseguimos a través de la autenticación en la aplicación web.

PUT	Se encarga de realizar modificaciones sobre un recurso de la entidad, de manera que si hacemos una petición PUT a /pacientes con la información a modificar en el body y el token de autenticación nos permite actualizar ese paciente.
DELETE	Se encarga de eliminar recursos de una entidad, al hacer una petición DELETE a /pacientes con la información necesaria para identificar el recurso, nos permite eliminarlo de nuestra tabla.

Dentro de estas funciones hay subfunciones, por ejemplo, para comprobar que el paciente exista o no, comprobar si está ingresado, comprobar si cierta cama esta disponible y no está bloqueada etc. Todas estas subfunciones garantizan que las principales se ejecuten de manera correcta haciendo las comprobaciones necesarias para ello.

Además, para la recuperación de contraseñas se ha instalado las librerías necearías para la funcionalidad de envió de correos en PHP, en este caso “PHPMailer”

Por último, se ha desarrollado un script en Bash para gestionar la eliminación de tokens expirados de la base de datos, asegurando así la seguridad y eficiencia del sistema. Este script se conecta a la base de datos MySQL y elimina automáticamente los tokens que han superado el límite de su creación.

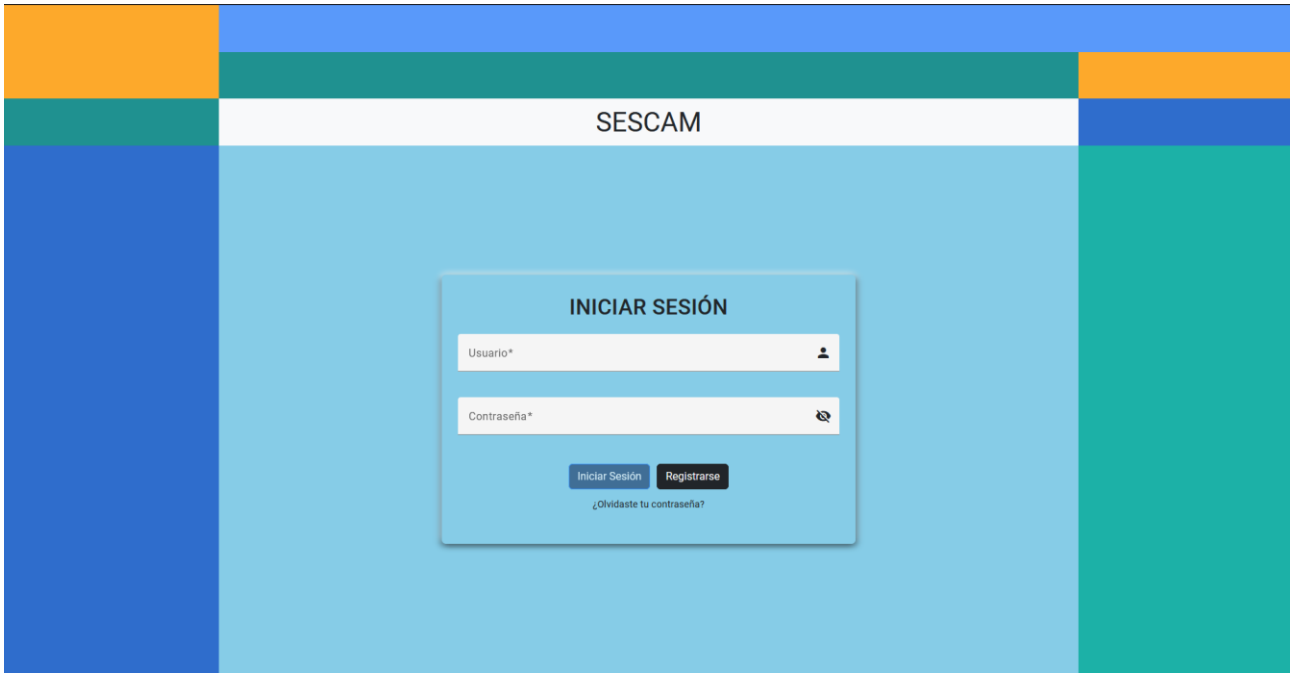
CLIENTE

Para la parte cliente de la aplicación, que constituye la interfaz de usuario, emplearemos tecnologías comunes como HTML, CSS, JavaScript y el framework de Angular para crear una interfaz de usuario dinámica. Esta interfaz permitirá a los usuarios interactuar de forma eficiente y fluida con el sistema, mientras realiza una comunicación asíncrona con el servidor.

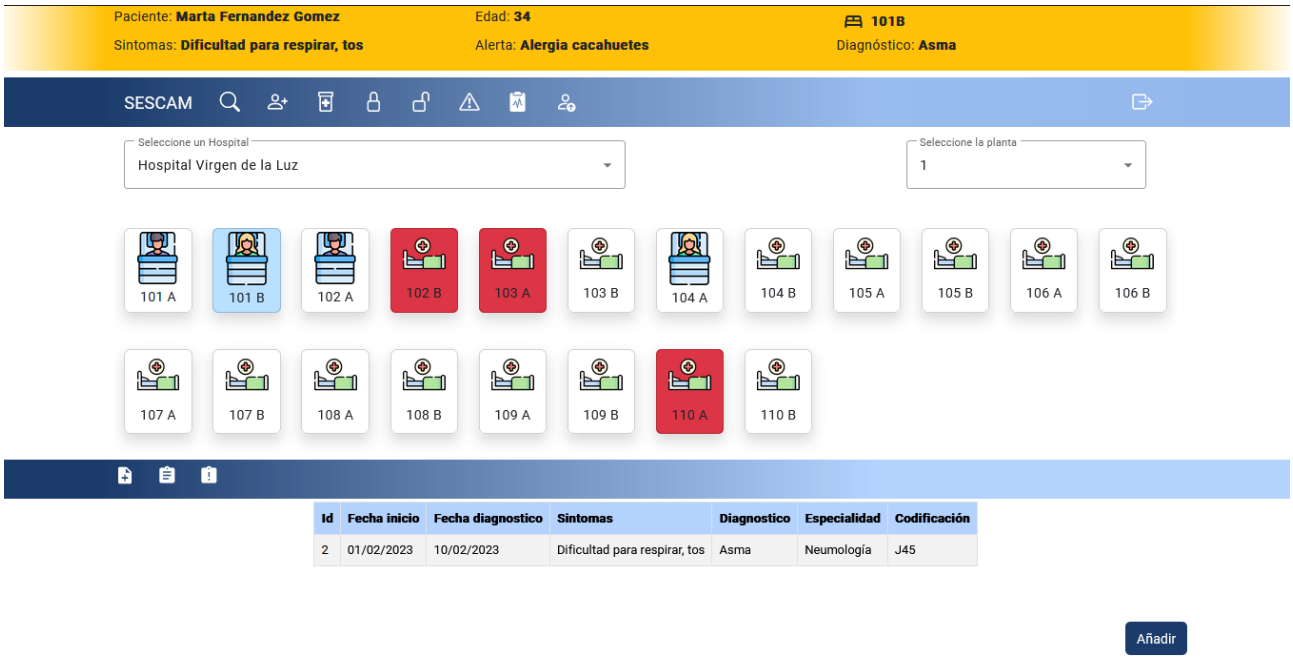
DISEÑO DE LA INTERFAZ

Como gemelo digital tiene que ser lo más parecida a la aplicación original de Mambrino además de ser muy intuitiva para los usuarios con menos conocimientos técnico.

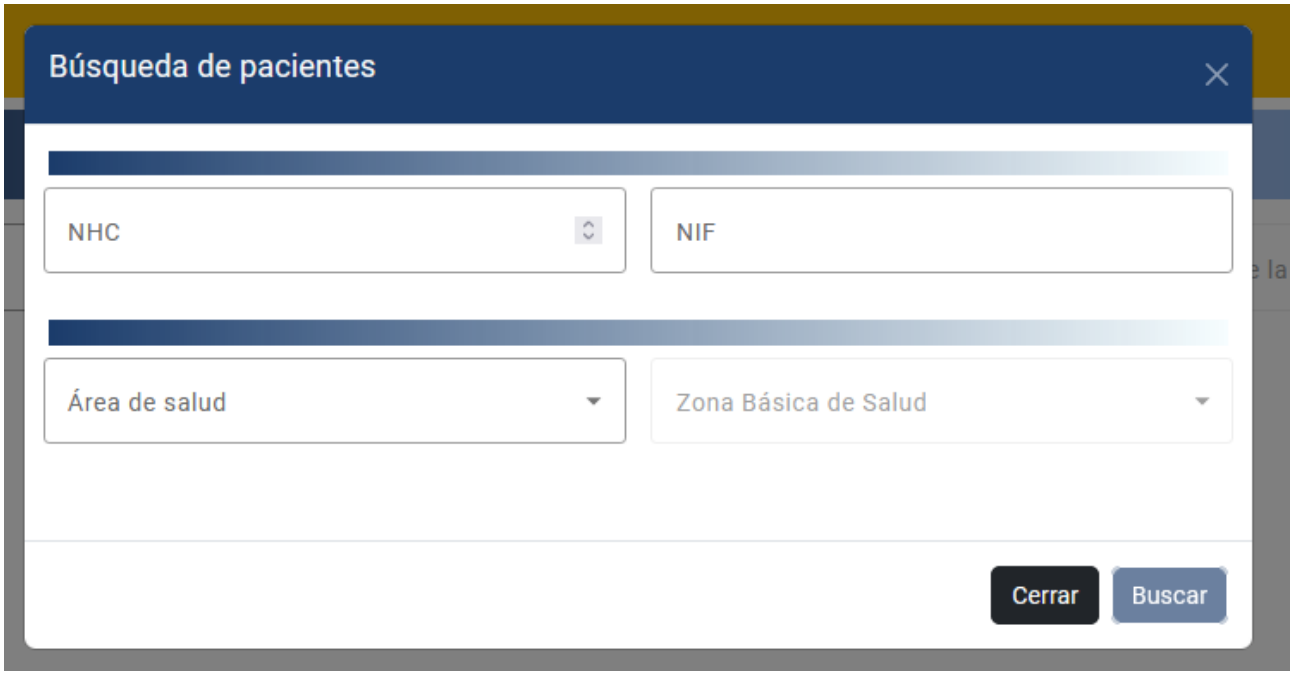
PANTALLA DE INICIO

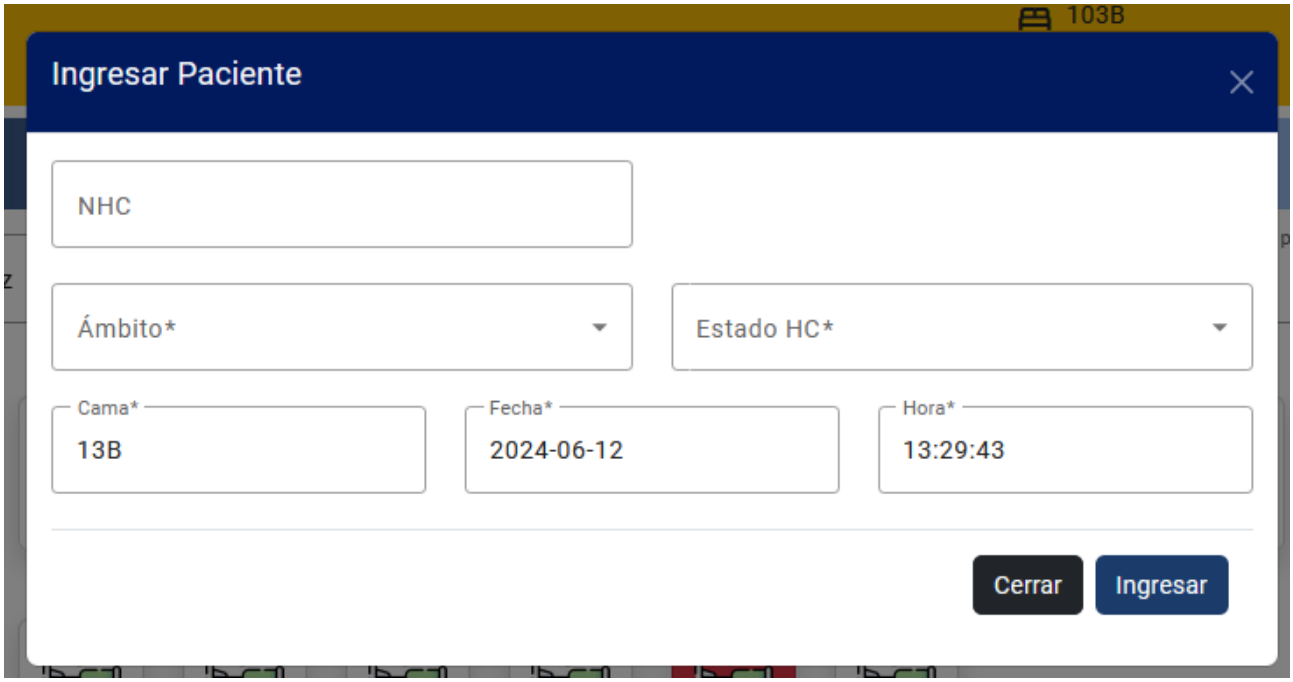


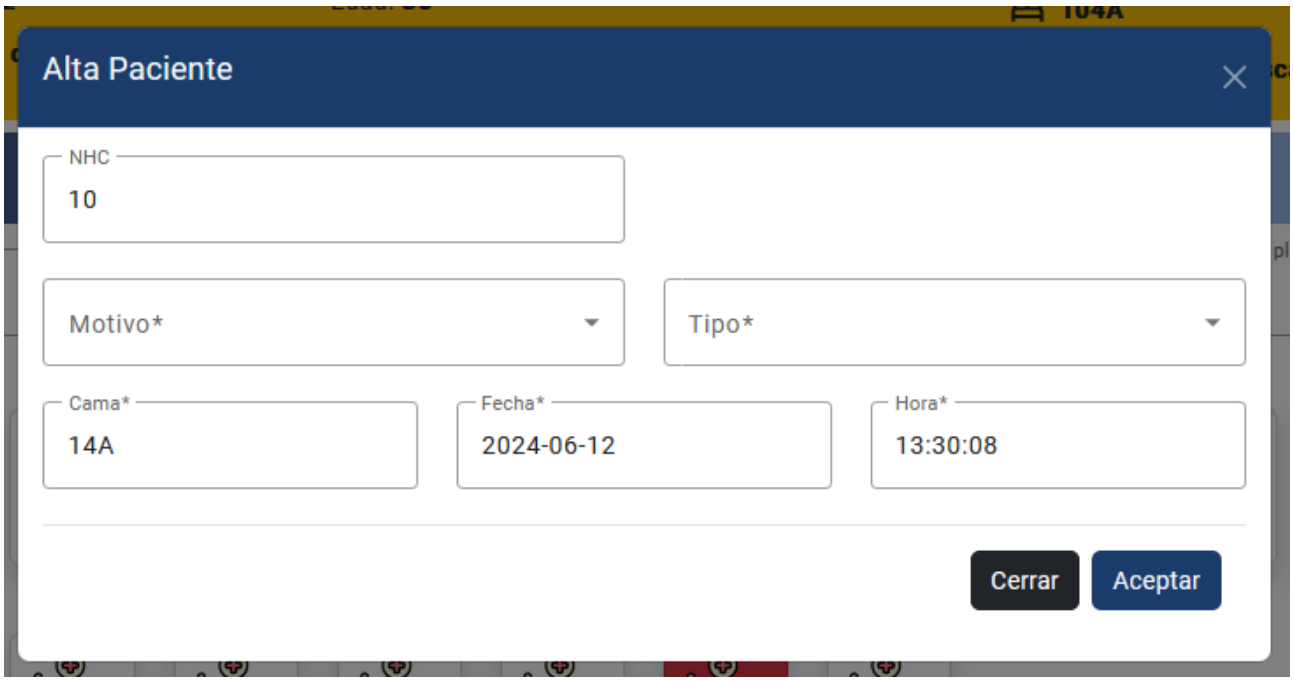
PANTALLA PRINCIPAL DE USUARIO

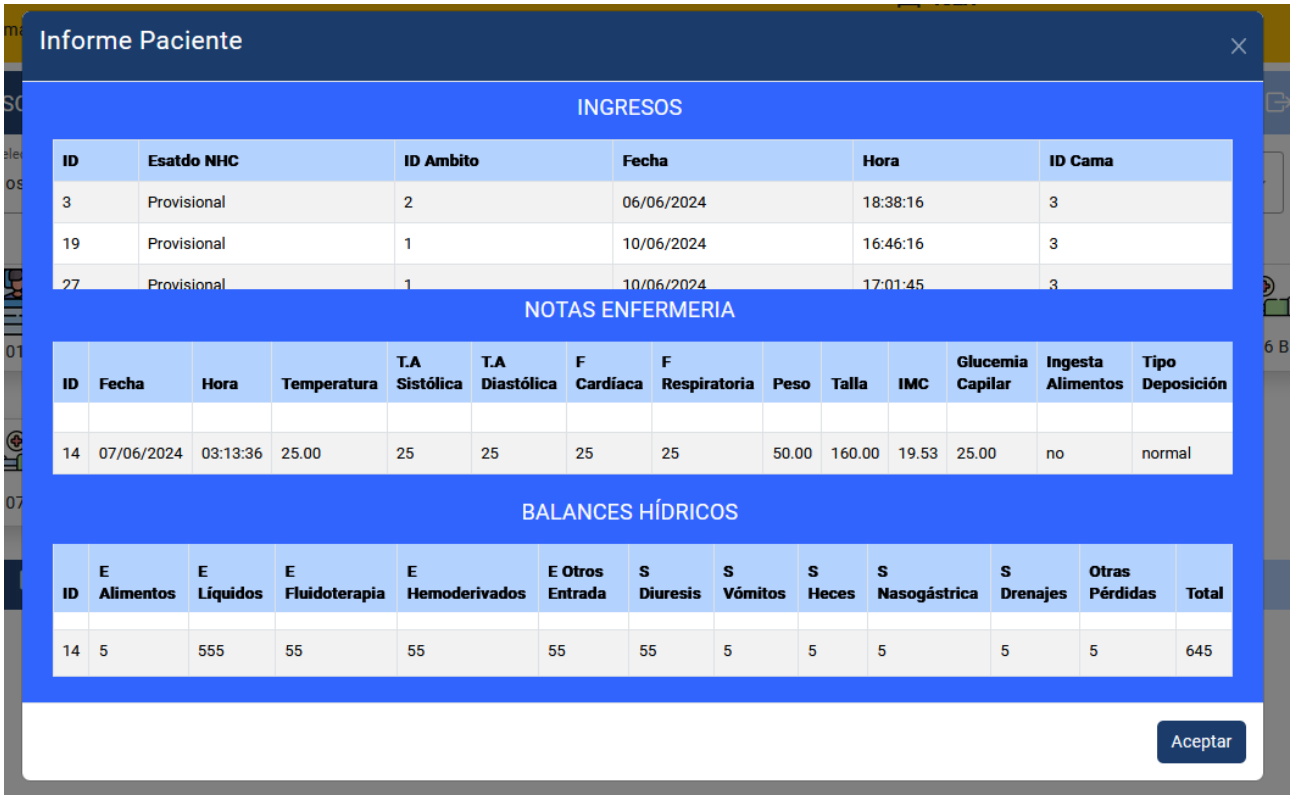


VENTANAS PARA LA GESTIÓN DE PACIENTES

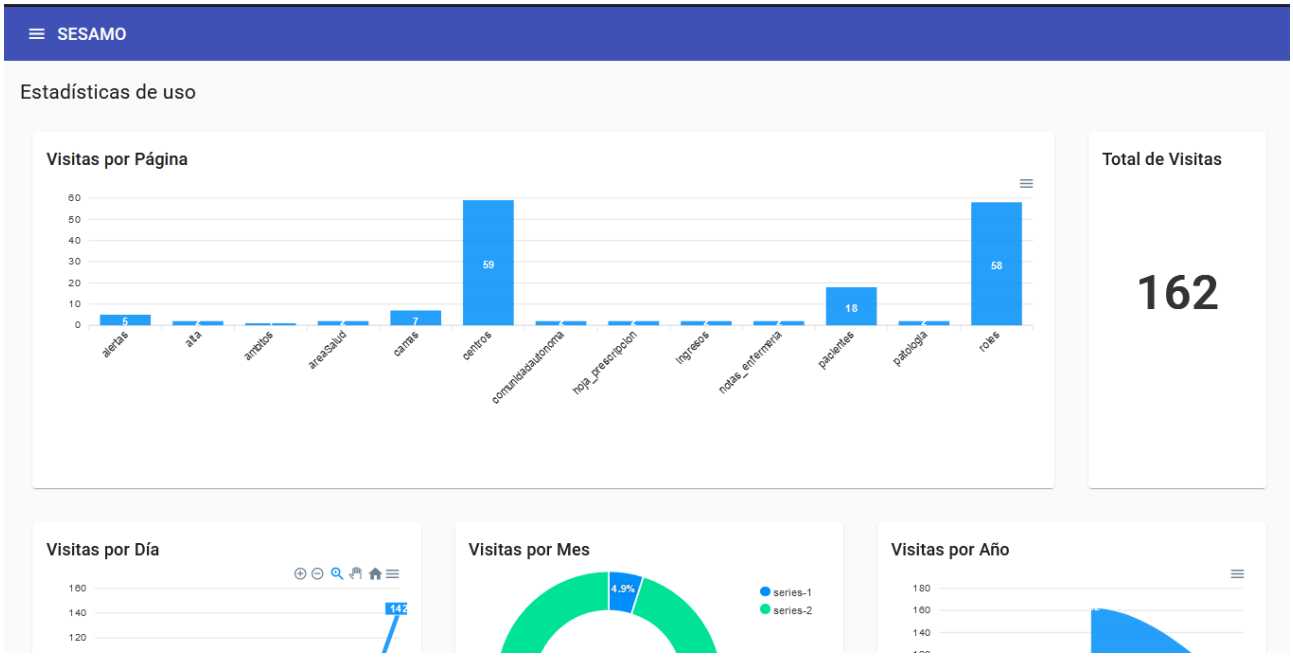




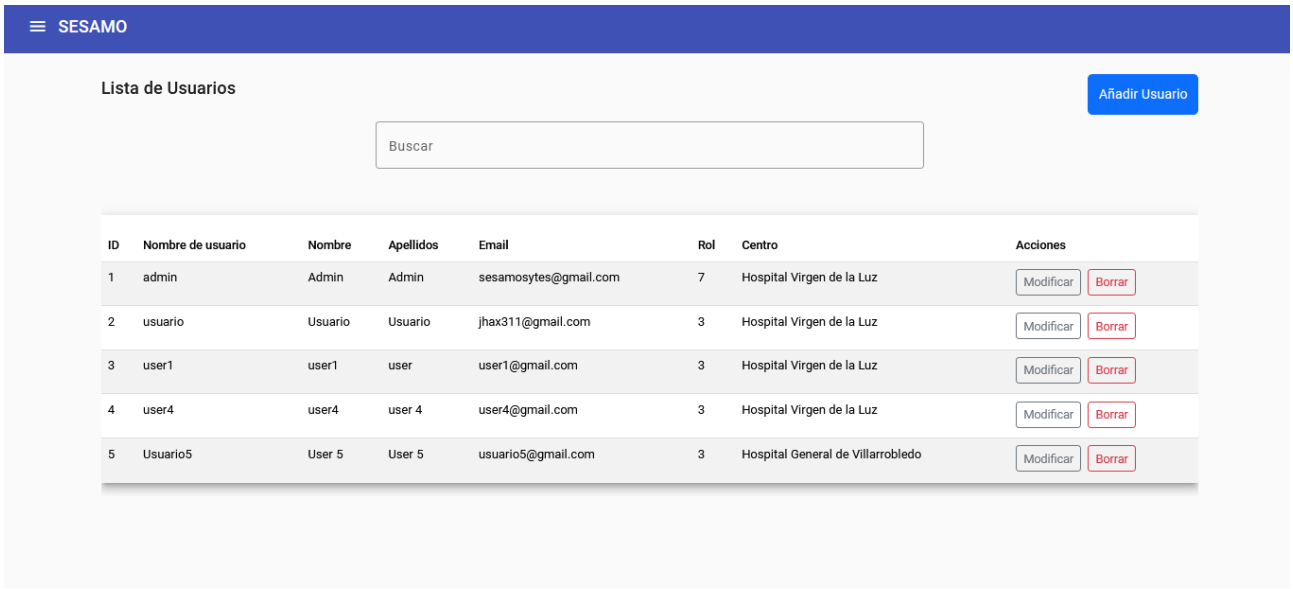

Alta Paciente
NHC
10
Motivo*
Tipo*
Cama*
14A
Fecha*
2024-06-12
Hora*
13:30:08
Cerrar
Aceptar


Informe Paciente
INGRESOS
ID | Esatdo NHC | ID Ambito | Fecha | Hora | ID Cama
3 | Provisional | 2 | 06/06/2024 | 18:38:16 | 3
19 | Provisional | 1 | 10/06/2024 | 16:46:16 | 3
27 | Provisional | 1 | 10/06/2024 | 17:01:45 | 3
NOTAS ENFERMERIA
ID | Fecha | Hora | Temperatura | T.A Sistólica | T.A Diastólica | F Cardíaca | F Respiratoria | Peso | Talla | IMC | Glucemia Capilar | Ingesta Alimentos | Tipo Deposición
14 | 07/06/2024 | 03:13:36 | 25.00 | 25 | 25 | 25 | 25 | 50.00 | 160.00 | 19.53 | 25.00 | no | normal
BALANCES HÍDRICOS
ID | E Alimentos | E Líquidos | E Fluidoterapia | E Hemoderivados | E Otros Entrada | S Diuresis | S Vómitos | S Heces | S Nasogástrica | S Drenajes | Otras Pérdidas | Total
14 | 5 | 555 | 55 | 55 | 55 | 55 | 5 | 5 | 5 | 5 | 5 | 645
Aceptar

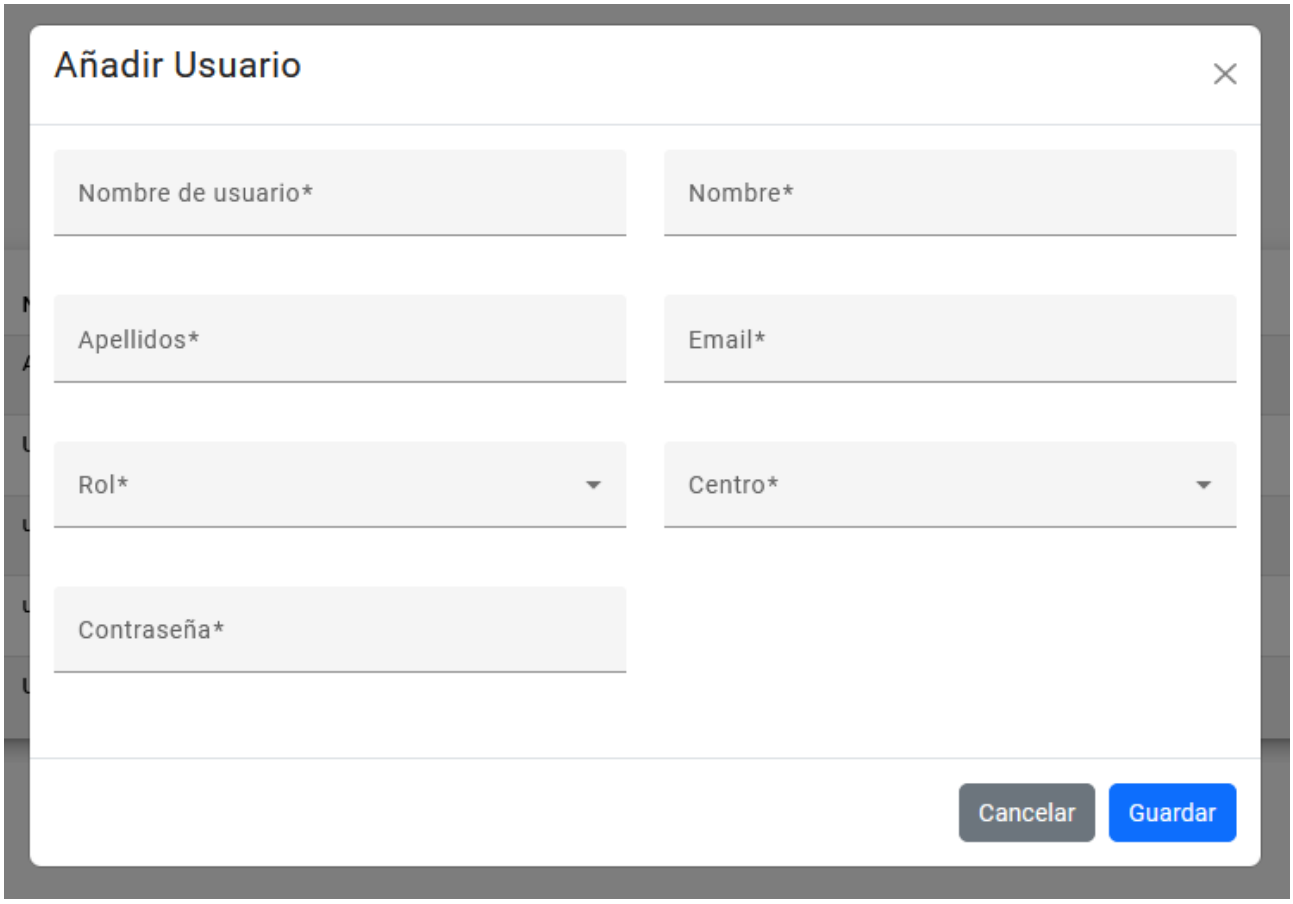
PARTE ADMINISTRADOR



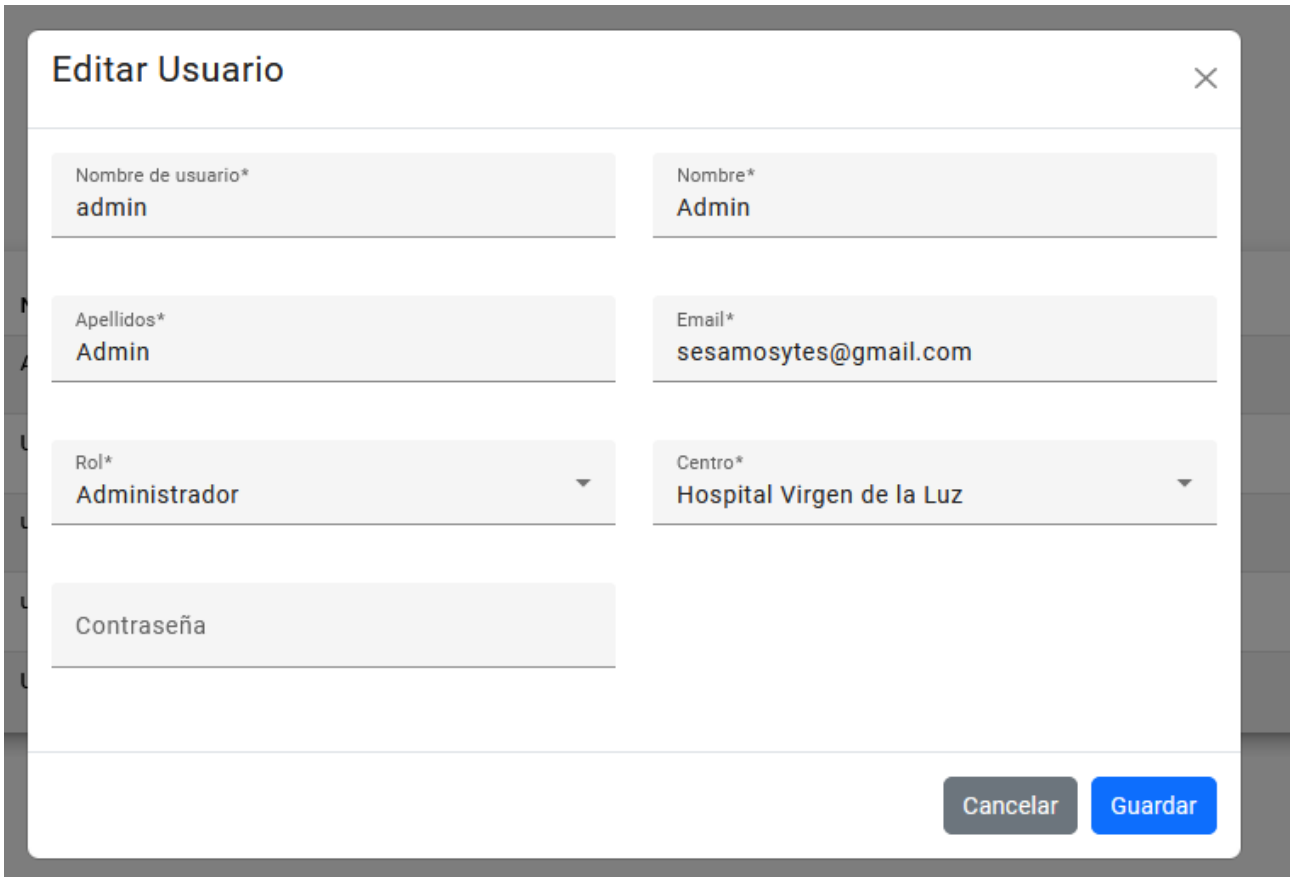
GESTIÓN DE USUARIOS (ADMINISTRADOR)



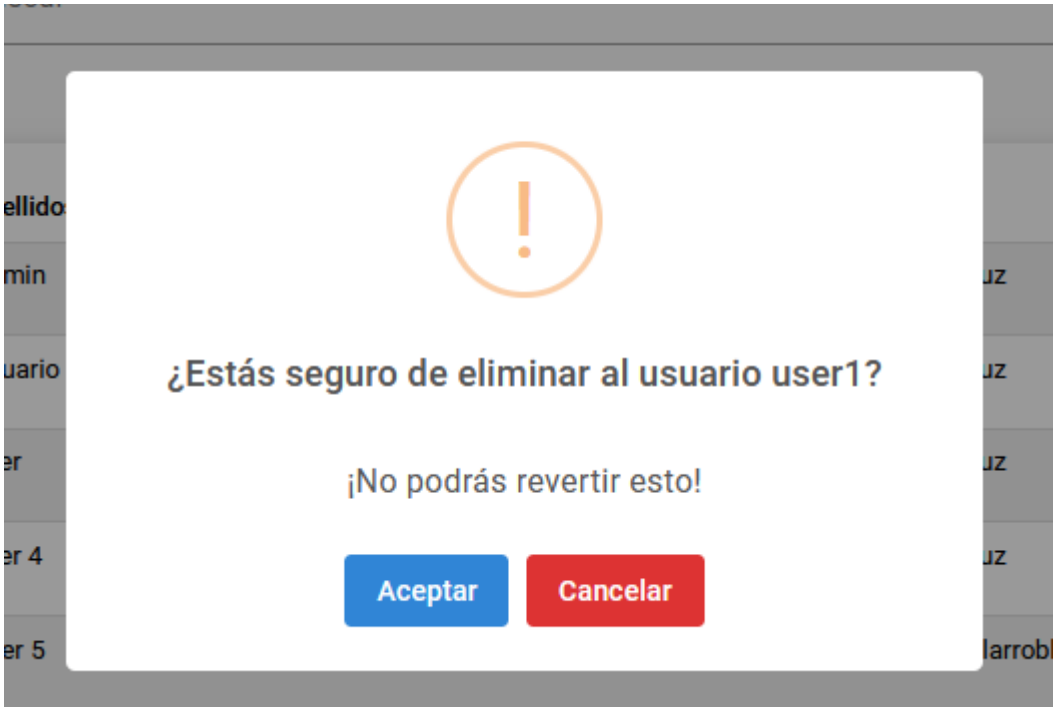
CREAR USUARIOS



MODIFICAR USUARIOS



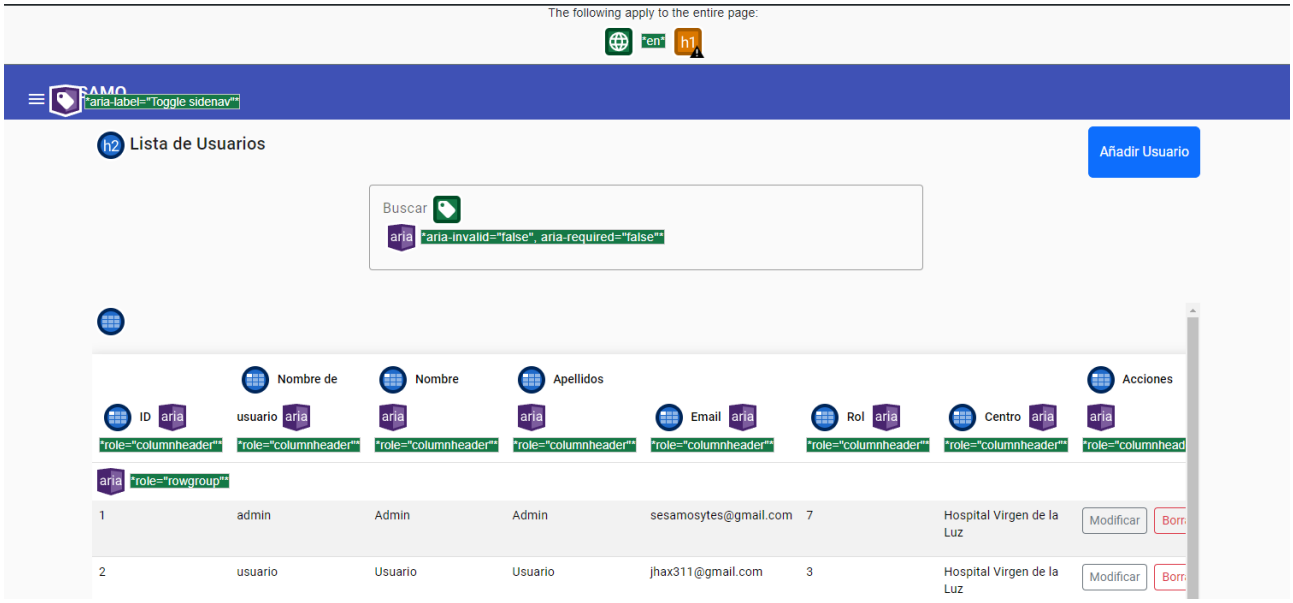
ELIMINAR USUARIOS



ACCESIBILIDAD

Para asegurar la accesibilidad todas las imágenes tendrán atributos “alt” descriptivos, los enlaces tendrán textos significados a su función, pondremos encabezados en tablas además los formularios tienen etiquetas y campos accesibles.

Aplicación comprobada por la herramienta de WAVE:



USABILIDAD

La usabilidad en un gemelo digital es fundamental para garantizar que la interfaz sea comprensible, eficiente y efectiva. En el contexto educativo, en este caso la gestión de la salud, es muy importante

para mejorar la absorción del conocimiento y la competencia en el uso de herramientas digitales de manera que se ofrece una transición más suave a la práctica profesional.

La aplicación web se encarga de:

- Asegurar que los usuarios puedan aprender las funcionalidades de Mambrino de manera rápida y directa.
- Reducir la carga visual mediante una interfaz intuitiva que evita al usuario dificultades innecesarias.
- Facilitar la ejecución de tareas con exactitud promoviendo resultado de alta calidad.
- Promover una experiencia que incentive al uso continuado y positivo de la plataforma
- Confirmar visualmente sobre las acciones generadas del usuario.
- Soporte para múltiples dispositivos, aunque la experiencia en escritorio es la para la que ha sido diseñada.

Desarrollo web

Para abordar el desarrollo de la parte cliente en el proyecto utilizaremos Angular, un framework moderno y poderoso para construcción de aplicaciones web dinámicas. Aquí se detallan los aspectos técnicos y las ventajas de utilizar Angular, junto con sus características para formularios, validación, gestión de eventos y comunicación con el servidor.

Gracias a Angular, el cual opera bajo una arquitectura basada en componentes, lo que facilita la reutilización y la modularidad. Esto es esencial para mantener el código organizado y facilitar la mantenibilidad y escalabilidad de la aplicación.

Angular permite que los cambios en el modelo de datos se reflejen automáticamente en la vista y viceversa, lo cual es crucial para las interfaces interactivas donde el estado debe actualizarse dinámicamente.

Lo más importante es que Angular nos permite desarrollar aplicaciones de una sola página, lo que mejora la experiencia del usuario al minimizar los tiempos de carga y proporcionar transiciones fluidas entre vistas sin recargar la página completa.

Formularios y validación

Los formularios son una parte critica del proyecto, ya que permiten la entrada y gestión de datos de manera eficiente:

- **Angular Material:** Usamos Angular Material para construir formularios que no solo son funcionales sino también estéticamente agradables. Esta biblioteca ofrece una amplia gama de componentes de interfaz de usuario diseñados con principios de material design.
- **Validación Integrada:** Angular proporciona validaciones predefinidas como *required*, *email*, *minLength,* y *maxLength*, que facilitan enormemente la implementación de controles de entrada robustos. Además, permite crear validaciones personalizadas para adaptarse a requisitos específicos, asegurando que los datos recogidos sean precisos y útiles.

Gestión de almacenamiento local

- **Local Storage:** Utilizamos el almacenamiento local para guardar datos de forma persistente o temporal. Esto mejora la experiencia del usuario al reducir la necesidad de solicitudes al servidor para cada acción, optimizando así el rendimiento y la respuesta de la aplicación.
    - Principalmente guardaremos el token recibido al autenticar el usuario y su rol.

Modificación del DOM

- Utilizamos directivas como ngIf o ngFor para manipular el DOM de manera declarativa, lo que simplifica el código y mejora su legibilidad.
- Angular facilita la gestión de eventos como clics y entradas de formulario utilizando tales como (click), (valueChange), lo que permite manejar iteraciones del usuario de manera eficiente.

Comunicación con el servidor:

- Para la comunicación entre el cliente y el servidor, utilizamos los servicios HTTP de Angular que permiten realizar operaciones CRUD sobre las entidades del sistema como pacientes e ingresos. Las solicitudes asíncronas mejoran la experiencia del usuario al permitir la actualización y consulta de datos sin recargar la página.
- La comunicación entre la parte y el cliente se realizará mediante la API. Esta API permite al cliente realizar operaciones CRUD sobre las diferentes entidades del sistema.

FASE DE DESPLIEGUE

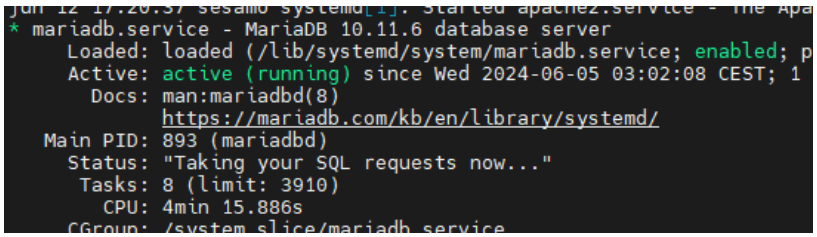
En la opción de despliegue elegida para el proyecto, se ha optado por el alojamiento propio utilizando una configuración LAMP en una Raspberry Pi. Esta sección detalla los pasos necesarios para poner en producción la aplicación, garantizando su funcionalidad y seguridad.

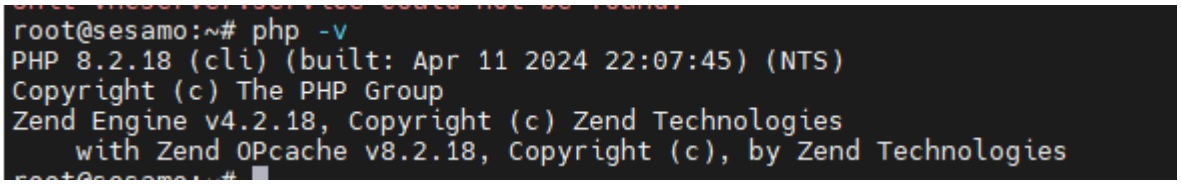
INSTALACIÓN LAMP

En esta fase, se procederá a configurar un entorno LAMP (Linux, Apache, MySQL, PHP) en una Raspberry Pi para proporcionar un servidor web robusto y eficiente capaz de alojar nuestra aplicación.

CONFIGURACIÓN DEL ENTORNO LAMP:

- **Raspberry Pi:** Como hardware base, se utiliza una Raspberry Pi debido a su bajo costo, eficiencia energética y suficiente capacidad para ejecutar aplicaciones web.
- **Linux (Raspbian):** Utilizamos Raspbian, un sistema operativo basado en Debian optimizado para la Raspberry Pi.
- **Apache:** Instalación y configuración del servidor web Apache para manejar las solicitudes HTTP. Se configura el módulo "mod_rewrite" para permitir reescrituras de URL, crucial para una aplicación SPA y para gestionar las rutas de la API
- **MySQL:** Instalación del sistema de gestión de base de datos MySQL, que albergará las tablas y datos de la aplicación.
- **PHP**: Instalación de PHP y configuración para interactuar adecuadamente con Apache y MySQL.
- **phpMyAdmin:** Instalación de phpMyAdmin para facilitar la gestión de la base de datos a través de una interfaz web amigable.





CREACIÓN DE LA BASE DE DATOS

En esta etapa, se crean y configuran la base de datos necesarias para el proyecto, asegurando que todas las estructuras de datos estén optimizadas y listas para su uso.

CREACIÓN Y CONFIGURACIÓN DE LA BASE DE DATOS

- Utilización de phpMyAdmin para la creación de la base de datos y tablas a través de los scripts SQL desarrollados en la fase de diseño de la base de datos.
- Asegurar la integridad y el rendimiento óptimo mediante la revisión de índices y claves foráneas.

CONEXIÓN Y CONFIGURACIÓN

Esta fase cubre la conexión de la aplicación con la base de datos y la implementación de mecanismos de seguridad para proteger el acceso a la aplicación y a los datos.

CONEXIÓN DE LA APLICACIÓN CON LA BASE DE DATOS

- Se configura un archivo de configuración en "/etc/sesamo" que guardara los parámetros de conexión a la base de datos.
- Pruebas de conexión para verificar que la configuración es correcta y que la aplicación puede comunicarse eficazmente con la base de datos.

MECANISMOS DE SEGURIDAD DE ACCESO

- Configuración de ".htaccess" para restringir el acceso a directorios sensibles, forzar el uso de HTTPS y mejorar la seguridad del servidor web.
- Implementación de políticas de seguridad en la base de datos, como usuarios y permisos específicos para limitar el acceso y la manipulación de la información sensible.

INSTALACIÓN DE SSL CON CERTBOT

Para asegurar que todas las comunicaciones con el servidor sean seguras, se instalará un certificado SSL utilizando Certbot.

Certbot es una herramienta de código abierto que simplifica la obtención y renovación de certificados SSL/TLS de Let's Encrypt, una autoridad de certificación gratuita. SSL (Secure Sockets Layer) y su sucesor TLS (Transport Layer Security) son protocolos que permiten cifrar las comunicaciones en una red informática, asegurando que los datos transferidos entre el servidor y los clientes permanezcan privados y protegidos.

Beneficios de SSL:

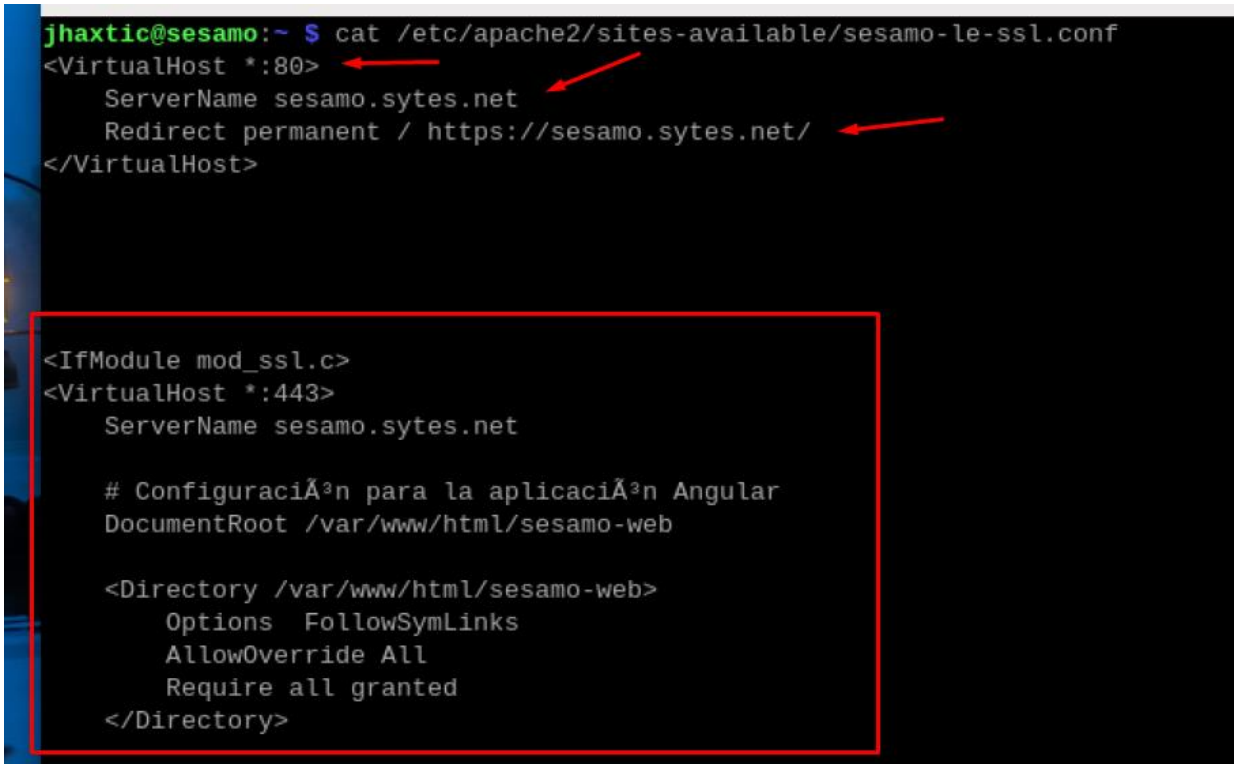
- **Seguridad:** Encripta la información para proteger los datos sensibles durante la transmisión.
- **Confianza:** Aumenta la confianza de los usuarios al ver el indicador de conexión segura en el navegador.
- **SEO:** Mejora el posicionamiento en los motores de búsqueda, ya que Google favorece los sitios seguros.

Cerbot automatiza el proceso de creación de certificados SSL. Ejecutando Cerbot, se obtiene e instala el certificado SSL para el dominio del servidor local.

sudo certbot –apache

CONFIGURACIÓN DE APACHE PARA USAR EL CERTIFICADO SSL

Una vez obtenido el certificado, se configura Apache para usarlo, habilitando HTTPS en el servidor. Además, se redirigen las peticiones HTTP a HTTPS para asegurar que todas ñas comunicaciones sean seguras.



CONFIGURACIÓN DE RED

Para hacer accesible la aplicación desde fuera de la red local, se realizarán los siguientes pasos de configuración de red.

REGISTRO EN UN PROVEEDOR DNS DINÁMIC

Para gestionar la IP dinámica del servidor, se utiliza DynDNS para registrar y monitorear la dirección IP y utilizamos No-IP para asociar el nombre de host de DynDNS con un dominio más amigable y accesible, en este caso nuestro dominio final es **sesamo.sytes.net**

Nombre de host ▲	Last Update	IP / Objetivo	Type
sesamo.sytes.net Active	Jun 12, 2024 03:41 PDT	84.127.56.24	A

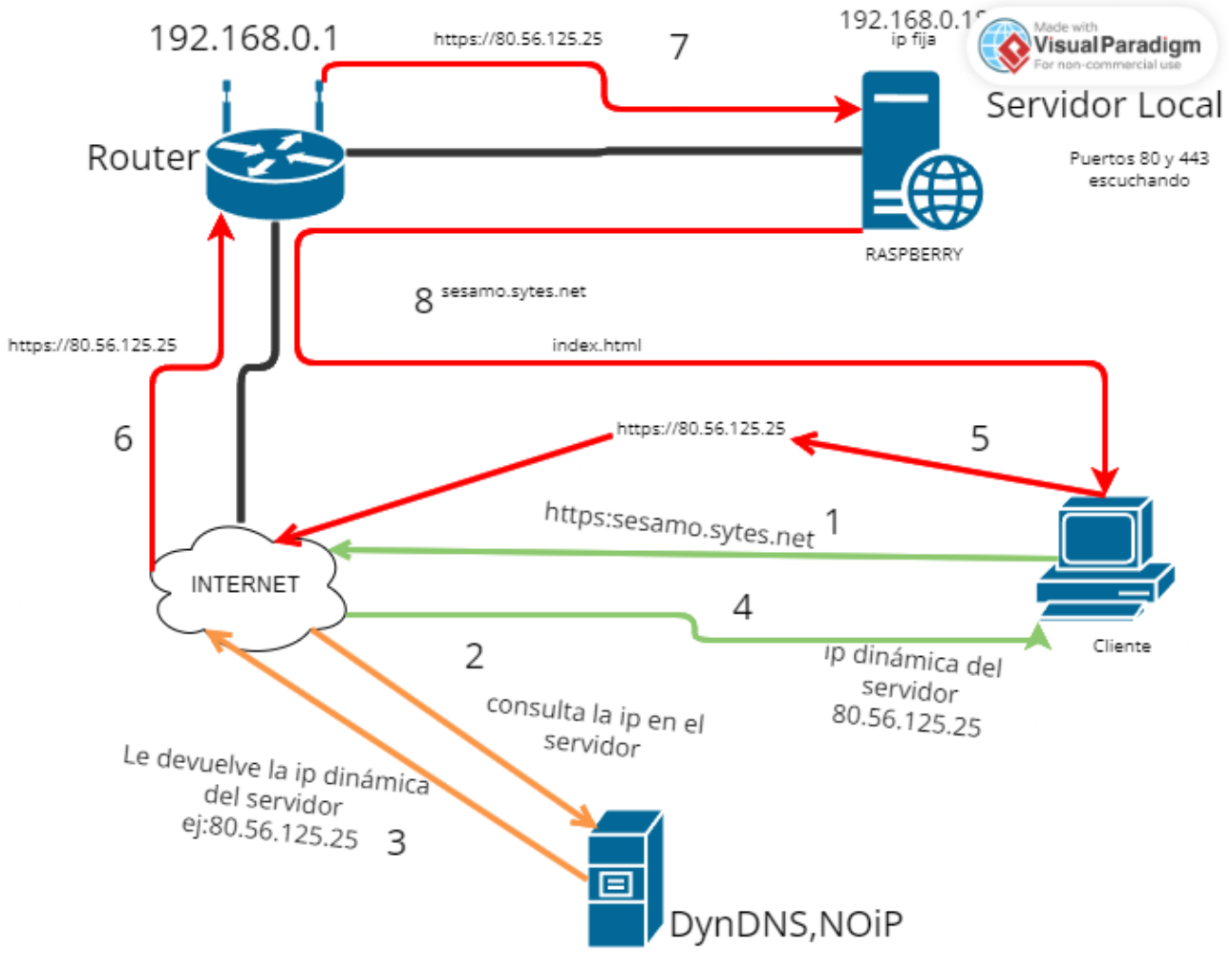
CONFIGURACIÓN DE PUERTOS EN EL ROUTER

En el router debemos configurar el DynDNS para que se actualice dinámicamente

Y realizaremos una configuración de redirección de puertos para dirigir las peticiones web al servidor Apache instalado localmente en la Raspberry pi. Se configuran los puertos 80 (HTTP) y 443 (HTTPS) para permitir el acceso web seguro.

Además, configuramos SSH para permitir la administración remota segura del servidor. Ya que Raspberry contiene interfaz gráfica, he configurado el puerto 5900 para permitir la conexión VNC, la cual estará activa en el desarrollo del proyecto con la intención de al final dejar solo SSH.

EL diagrama del funcionamiento del hosting web local, seria este:



En conclusión, el despliegue de la aplicación en una Raspberry Pi proporciona un entorno controlado y flexible que permite realizar modificaciones rápidas y pruebas directas. La implementación de SSL garantiza que todas las comunicaciones sean seguras, proporcionando confianza y seguridad tanto para los usuarios como para los administradores del sistema.

5. RECURSOS MATERIALES

En este apartado se detallan los recursos de hardware y software necesarios para llevar a cabo el proyecto , así como un presupuesto que indica el coste asociado a dichos recursos.

5.1 HARDWARE

- Raspberry Pi
  - Modelo: Raspberry Pi 4 Model B
  - precio: 60 €
  - Descripción: microordenador de bajo costo y alta eficiencia adecuado para ejecutar aplicaciones web.
- Alimentación y accesorios para la Raspberry Pi
  - Adaptador de corriente: 7.35 €

- Tarjeta microSD (32GB) : 15€
- Caja protectora: 5.95€
- Cable HDMI : 3.36€

- Periféricos:
  - Teclado, ratón (usando existentes): 0€

5.2 SOFTWARE

- Sistema operativo y utilidades
  - Raspbian OS: gratis
  - Cerbot: gratis
  - NoIP, DynDNS: gratis
- Software de desarrollo y herramientas
  - Apache: Gratis
  - MySQL: Gratis
  - PHP: Gratis
  - phpMyAdmin: Gratis
  - Angular (Framework): Gratis
  - PHPMailer: Gratis

5.3 PRESUPUESTO

El presupuesto total para el proyecto se desglosa a continuación, cubriendo tanto los costos de hardware como de software.

Recurso	Descripción	Precio
Raspberry Pi	Modelo Raspberry Pi 4 Model B	60
Adaptador de Corriente	Alimentación para la Raspberry Pi	7.35
Tarjeta MicroSD (32GB)	Almacenamiento	15
Caja Protectora	Protección para la Raspberry Pi	5.95
Cable HDMI	Conexión de video	3.36
Teclado, Ratón y Monitor	Uso de periféricos existentes	0
Sistema Operativo y Utilidades	Raspbian OS, Certbot, No-IP DynDNS	0
Software de Desarrollo y Herramientas	Apache, MySQL, PHP, phpMyAdmin, Angular, PHPMailer	0
Total		91.66€

6. RECURSOS HUMANOS

6.1 ORGANIZACIÓN

La organización representa los departamentos necesarios para desarrollar el proyecto al completo en un futuro, actualmente solo estaría compuesta por una persona.

DEPARTAMENTOS NECESARIOS

1. Dirección:
    a. Encargado de supervisar el proyecto en general, incluyendo la planificación estratégica y toma de decisiones clave. Coordina y guía todos los demás departamentos para asegurar que los objetivos del proyecto se cumplan de manera efectiva y eficiente.
2. Recursos Humanos:
    a. Responsable de la gestión del personal, incluyendo la contratación, capacitación, desarrollo y bienestar de los empleados.
3. Marketing:
    a. Desarrolla y ejecuta estrategias de marketing para promover los productos y servicios de la empresa, gestiona las relaciones públicas y maneja las campañas publicitarias.
4. Administración y economía:
    a. Maneja las finanzas, la contabilidad y otras funciones administrativas necesarias para el funcionamiento diario de la empresa. Se encarga de la gestión de presupuestos, nóminas y reportes financieros.
5. Programadores:
    a. Responsables del desarrollo técnico del proyecto. Se divide en subdepartamentos de Front-End y Back-End para asegurar una completa integración y funcionalidad de los productos.
        i. **Desarrolladores Front-End:** Implementan interfaces de usuario atractivas y funcionales utilizando tecnologías como HTML, CSS y JavaScript.
        ii. **Desarrolladores Back-End:** Desarrollan y mantienen la lógica del servidor, bases de datos y APIs utilizando lenguajes como PHP, Python o Node.js.
6. Administrador de Sistemas Informáticos:
    a. Gestiona la infraestructura tecnológica de la empresa, asegurando que todos los sistemas informáticos funcionen de manera eficiente y segura. Se encarga del mantenimiento de servidores, redes y sistemas de seguridad.

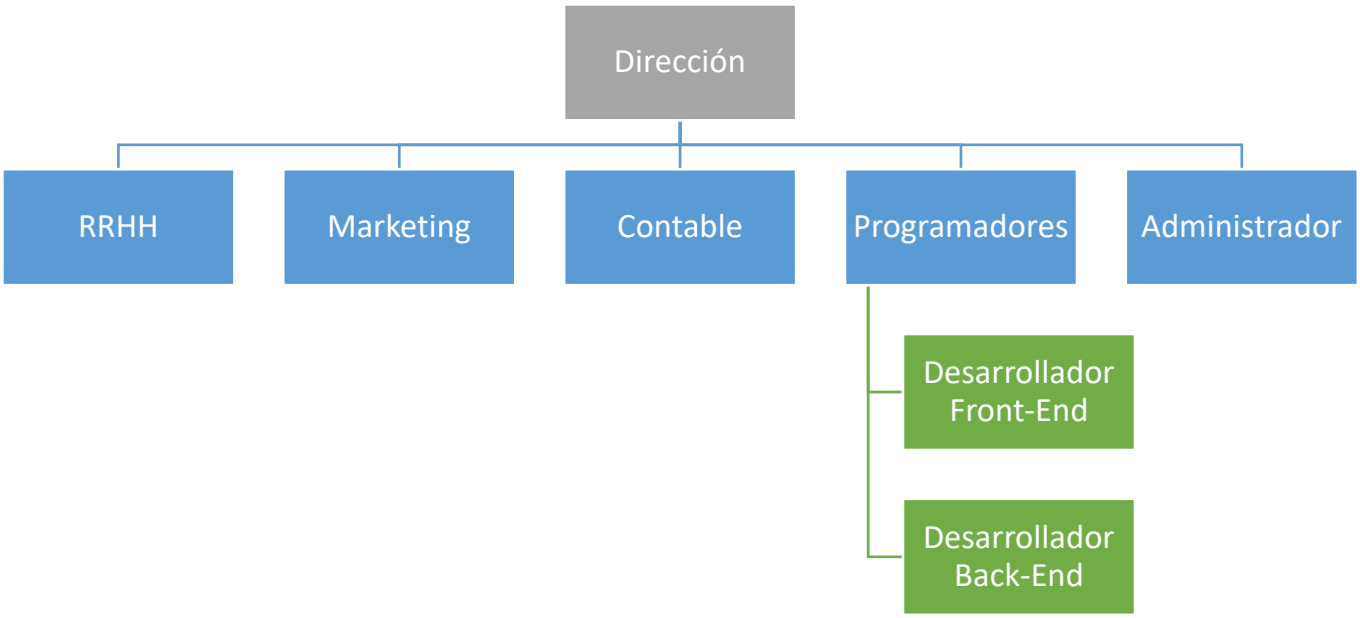
FUNCIONES

- Dirección:

- Liderazgo estratégico, toma de decisiones clave, supervisión de todos los departamentos y representación de la empresa ante terceros.

- RRHH:
  - Gestión de contratación, capacitación y desarrollo de empleados, manejo de conflictos laborales y aseguramiento del bienestar del personal.
- Marketing:
  - Desarrollo de estrategias de marketing, gestión de campañas publicitarias y análisis de mercado.
- Administración y economía:
  - Gestión de las finanzas de la empresa, incluyendo presupuestos, nóminas y reportes financieros.
- Programadores:
  - Jefe de desarrollo
    - Supervisión de los equipos de desarrollo, planificación de proyectos y aseguramiento de la calidad técnica.
  - Desarrolladores Front-End:
    - Implementación de interfaces de usuario, asegurando la funcionalidad y diseño atractivo.
  - Desarrolladores Back-End:
    - Desarrollo de la lógica del servidor, integración de bases de datos y APIs.
- Administradores de sistemas informáticos
  - Se encarga de la gestión de la infraestructura tecnológica, mantenimiento de servidores, redes y sistemas de seguridad.

ORGANIGRAMA



6.2 CONTRATACIÓN

Necesitamos cubrir las siguientes bacantes:

- Gerente de recursos humanos.
- Gerente de marketing.
- Contable.
- Jefe de desarrollo.
- Desarrollador Front-End.
- Desarrollador Back-End.
- Administrador de sistemas informáticos.

Estos puestos cambiaran en función del crecimiento de la empresa y la necesidad de cubrir nuevos proyectos con más personal.

PROFESIOGRAMA

	FACTORES	1	2	3	4
Datos objetivos	Edad				x
	Nivel cultural			x	
	Estudios realizados				x
	Inglés		x		
	Informática				x
	Imagen			x	
	Experiencia				x
Datos psicológicos	Personalidad	Seguridad			x
		Control		x	
		Expresivo		x	
		Comunicación			x
	Rasgos profesionales	Organización		x	
		Responsabilidad			x
		Iniciativa		x	

CONTRATACIÓN Y COSTES DE LOS TRABAJADORES

Concepto	Precio
Sueldo media jornada	920€/mes
Coste seguridad social a empresa	558 €/mes
Coste seguridad social a empleado	57,60 €/mes
Salario neto	797,40 €/mes
Coste mensual para el empresario	1478.9€/mes

6.3 PREVENCIÓN DE RIESGOS LABORALES

Durante el desarrollo del proyecto, los trabajadores están expuestos a varios riesgos laborales. Se detallan los principales identificados:

- Riesgos ergonómicos:
  - Debido a la prolongada utilización de computadoras, los trabajadores pueden experimentar dolores de espalda, cuello y muñecas.
  - Síndrome del Túnel Carpiano
  - Trabajar muchas horas frente a una pantalla puede causar tensión ocular, visión borrosa y dolores de cabeza.
- Riesgos Psicosociales:
  - La presión para cumplir con los plazos, la carga de trabajo y la gestión de múltiples proyectos pueden generar estrés.
  - La acumulación de estrés y la falta de descansos adecuados pueden llevar al agotamiento físico y mental.
  - Trabajo remoto o largas horas en un entorno de oficina pueden reducir la interacción social.
- Riesgos de Seguridad:
  - Riesgo de choques eléctricos por equipos eléctricos mal mantenidos.
  - Mal uso de equipos eléctricos puede provocar incendios.

Para una empresa de desarrollo de aplicaciones web, la elección de la forma de organizar la prevención de riesgos laborales debe basarse en varios factores, incluyendo el tamaño de la empresa, el presupuesto disponible, y la complejidad de las operaciones. Por ello se opta por un Servicio de Prevención Ajeno para organizar la prevención de riesgos laborales. Esta elección proporciona una solución equilibrada entre costo y efectividad, garantizando el acceso a expertos en prevención y cumpliendo con las normativas legales, sin desviar recursos significativos de las actividades principales de la empresa.

7. VIABILIDAD TÉCNICA

Para el desarrollo del proyecto se han utilizado diversas herramientas y tecnologías que aseguran la viabilidad técnica del proyecto. A continuación, se detallan las características tecnológicas involucradas y los recursos utilizados.

7.2 HERRAMIENTAS Y TECNOLOGÍAS USADAS

El estudio de viabilidad del proyecto demuestra que constan de un conjunto completo y robusto de herramientas y tecnologías para asegurar el éxito del desarrollo.[1]

8. VIABILIDAD ECONÓMICA-FINANCIERA

8.1 INVERSIONES Y GASTOS

Los recursos necesarios para la ejecución del proyecto no son elevados, se deben considerar los costos asociados a recursos humanos, materiales y otros gastos. A continuación, se presenta una determinación detallada de estos costos.

Recursos Materiales:

- Portátil:
  - Se usa el personal por lo que no generaría un costo.
- Servidor (RaspBerry Pi)
  - 91.66€
- Software y licencias
  - Todo el software, licencias, So son gratuitos
- GitHub.
  - Repositorios en la nube gratuitos

Con esto tendríamos un coste en equipos de 91.66€

Para comenzar, se utilizará el centro de empresas de cuenca, obteniendo ayuda para empezar el negocio.

- Despacho con internet y gastos incluidos
  - 108€/mes
- Fianza inicial
  - 100€
- Cuota de autónomos:
  - 80€ (tarifa plana, con posibilidad de extensión, siendo menor de 30 años y sin superar el SMI)
- Solicitud de nombre comercial
  - 127.88€
- Marketing y publicidad.:

[1] Ver Anexo 1

- 100€/mes

Esto nos da un total de inversiones de : 607.54€

8.2 FINANCIACIÓN

Para asegurar la viabilidad económica del proyecto, se deben considerar diversas fuentes de financiación, así como la amortización de los préstamos y bienes de inversión.

FUENTES

- Ahorros personales
  - Utilización de ahorros personales para cubrir los gastos iniciales y operativos.
- Autofinanciación
  - Reutilización de las ganancias generadas por la empresa para reinversión y expansión futura.
- FFF(Family, Friends Fools)
  - Financiación inicial a través de préstamos personales y contribuciones de familiares, amigos y conocidos que confían en el proyecto.
    - Monto estimado: 1200€
- Ayudas y subvenciones
  - Subvenciones específicas para menores de 30 años que inician su actividad empresarial.
  - Fondos y subvenciones para proyectos innovadores en tecnología y desarrollo web.

PLAN DE AMORTIZACIÓN

PRÉSTAMO INICIAL

Monto: 1200€

Interés: 0%

Plazo de amortización: 12 meses

- Cuota mensual
  - 100€ / mes
- Total, pagado en 1 año
  - 1200€

BIENES DE INVERSIÓN

Equipos y herramientas: 91,66€

Despacho: 32€/mes

Fianza: 100€

Total, de inversión inicial: 191,66€

Pensamos en una amortización en 1 año

- 191.66€ / 12 = 15,97€
- Pagado en un año al 100%

La financiación del proyecto se asegura mediante una combinación de fuentes propias y financiación FFF (Family, Friends, and Fools), que proporciona un monto inicial sin intereses para cubrir los costos de inversión y operativos. La amortización del préstamo inicial y de los bienes de inversión se ha calculado para asegurar que los costos sean manejables y sostenibles en el tiempo, permitiendo una ejecución exitosa del proyecto.

8.3 VIABILIDAD ECONÓMICA-FINANCIERA

Para evaluar la viabilidad económico-financiera del proyecto, se considera la previsión de tesorería, la cuenta de resultados y el balance patrimonial. A continuación, se detallan estos aspectos para el primer año de actividad.

PREVISIÓN DE TESORERÍA

La previsión de tesorería refleja el flujo estimado de entradas y salidas de efectivo durante el primer año de actividad.

Entradas de efectivo		
Concepto	Monto Mensual	Monto Anual
Ahorro personal		300
FFF		1200
subvenciones		2000
Ingresos por servicios	4000	48000
Total de entradas	4000	51500
Salidas de Efectivo		
Concepto	Monto Mensual	Monto Anual
RRHH	2000	24000
Equipos y herramientas	91,66	91,66
Despacho	32	384

Fianza		100
Cuota autónomos	80	960
Marketing y publicidad	100	1200
Solicitud nombre comercial		127,88
Amortización de préstamo FFF	100	1200
Total, Salidas	2403.66	28063.54

Cuenta de resultados

Concepto	**Monto anual**
Ingresos Totales	51500
Gastos Totales	28063.54
Resumen Neto	23436,46€

Balance patrimonial

Activo	
Efectivo	23436,46€
Equipos y Herramientas	91,66
Total	23528,12
Pasivo	
FFF	1200
Total	1200
Patrimonio neto	300
Ganancias	23436,46
Total, patrimonio Neto	23736.46

Balance general

Total Activo (€)	**Total Pasivo y Patrimonio Neto (€)**
23,528.12	1,200 + 23,736.46
23,528.12	**23,528.12**

El análisis de la viabilidad económico-financiera muestra una ganancia neta de 23,436.46 € en el primer año debido a los ingresos generados por servicios y una gestión eficiente de los costos

operativos y de inversión inicial. La inversión en marketing y herramientas permitirá la estabilización y crecimiento del negocio en años subsiguientes, con expectativas de ingresos crecientes y una mejor gestión de los costos operativos. Para garantizar la sostenibilidad del proyecto, es fundamental seguir estrategias de control de costos y maximización de ingresos.

9. PROPUESTA DE MEJORA

Para continuar con el desarrollo y optimización de la plataforma, se presentan las siguientes propuestas de mejora.

En primer lugar, es esencial completar el desarrollo de todas las funciones previstas para cada tipo de usuario. Asegurar que profesores, técnicos en cuidados auxiliares de enfermería, administrativos, auxiliares de farmacia y parafarmacia, técnicos en imagen para el diagnóstico, técnicos en laboratorio clínico y biomédico, y administradores puedan utilizar la plataforma de manera completa y eficiente es una prioridad. Esto permitirá una experiencia formativa más enriquecedora y cercana a la realidad profesional.

Además, es fundamental seguir añadiendo más procesos y funcionalidades que simulen con mayor precisión los escenarios y procedimientos hospitalarios reales. La integración de módulos adicionales para la gestión de inventarios, la programación de citas, el seguimiento de tratamientos y el análisis de datos clínicos podría ser muy beneficiosa. Estas mejoras no solo enriquecerán la plataforma, sino que también la harán más útil y aplicable en diversos contextos sanitarios.

Otra propuesta clave es integrar el feedback de los usuarios actuales es crucial para el éxito continuo. Recoger y analizar este feedback permitirá identificar áreas de mejora y posibles nuevas funcionalidades. Adaptar la plataforma a las necesidades y expectativas reales de los estudiantes y profesionales del sector sanitario ayudará a mantenerla relevante y efectiva.

10. CONCLUSIONES

En conclusión, el proyecto ha cumplido con los resultados principales establecidos, logrando los objetivos digitales de la aplicación Mambrino. A pesar de que el proyecto ya había sido desarrollado anteriormente, mi trabajo ha servido como una implementación y mejora del proyecto original, permitiendo su extensión y uso en múltiples entornos. Este enfoque facilita la creación de una red de usuarios que utilicen la aplicación con fines formativos, especialmente en el ámbito sanitario.

La implementación realizada no solo asegura la funcionalidad y estabilidad de la aplicación, sino que también mejora su escalabilidad y adaptabilidad a diferentes escenarios educativos. Este proyecto ha demostrado ser técnicamente viable, utilizando tecnologías robustas y eficientes que garantizan un rendimiento óptimo. Económicamente, el proyecto es viable debido a su capacidad de adaptación a diversos contextos y la posibilidad de escalarse sin incurrir en costes prohibitivos.

El despliegue de la aplicación en un entorno LAMP (Linux, Apache, MySQL, PHP) utilizando una Raspberry Pi ha sido una de las fases más gratificantes y exitosas del proyecto. Este enfoque ha permitido un entorno controlado y flexible que facilita la realización de modificaciones rápidas y pruebas directas. La implementación de SSL garantiza que todas las comunicaciones sean seguras, proporcionando confianza y seguridad tanto para los usuarios como para el administrador. La configuración de red, incluyendo el uso de DNS dinámico y la redirección de puertos, ha asegurado que la aplicación sea accesible de manera remota y segura.

La parte del servidor ha sido diseñada para manejar eficientemente las solicitudes y operaciones CRUD (Crear, Leer, Actualizar, Eliminar) mediante la implementación de una API robusta en PHP. Esta API garantiza una comunicación asíncrona y segura entre el cliente y el servidor, permitiendo una interacción fluida y en tiempo real. Las funciones PHP desarrolladas para la gestión de entidades como pacientes, usuarios y recursos médicos aseguran que el sistema sea capaz de manejar grandes volúmenes de datos y operaciones complejas sin comprometer el rendimiento.

Además, este proyecto me ha permitido reafirmar mis conocimientos, especialmente en el uso de Angular. Anteriormente, había tenido dificultades para comprender Angular por completo. Sin embargo, a través del desarrollo de este proyecto, he logrado consolidar mis conocimientos y habilidades en esta tecnología.

En resumen, la realización del proyecto no solo ha reforzado mis habilidades técnicas y de resolución de problemas, sino que también ha ampliado las posibilidades de uso de la aplicación como gemelo digital de Mambrino, contribuyendo a su potencial impacto positivo en el ámbito educativo y sanitario.

Estoy entusiasmado con el impacto potencial que tendrá esta aplicación en la formación de futuros profesionales en el ámbito sanitario. Creo firmemente que facilitará una transición más fluida entre el entorno académico y la práctica profesional, ayudando a los estudiantes a adquirir habilidades prácticas esenciales. Espero continuar mejorando y ampliando la funcionalidad de la aplicación en futuras versiones.

11. WEBGRAFÍA

- http://avie.es/centro-de-empresas-de-cuenca/

- https://ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176742&menu=ultiDatos&idp=1254735576692
- https://todofp.es/que-estudiar/loe/informatica-comunicaciones/des-aplicaciones-web.html
- https://certbot.eff.org/instructions?ws=apache&os=debianbuster&tab=standard
- https://www.youtube.com/watch?v=4mKY_yDq64g&t=87s
- https://www.youtube.com/watch?v=-UHoAPsZNOk&list=PLIbWwxXce3VoAMUd6R1yI6oR23zz7MkkE&index=2
- https://www.youtube.com/watch?v=sFJsR5VprNs&t=193s
- https://www.youtube.com/watch?v=D1g6RW4oJEs&t=1029s
- https://www.youtube.com/watch?v=jkdLRbrJj9M&t=1677s
- https://www.youtube.com/watch?v=xH5HAMPwz6Q
- https://www.youtube.com/watch?v=2tknxiydR0s&t=1028s

12. ANEXOS

Anexo 1: tecnologías y herramientas de desarrollo

En este anexo se resumen las tecnologías y herramientas empleadas para el desarrollo del proyecto, detallando su aplicación especifica en el desarrollo de la plataforma educativa.

- **PHP**
  - **Aplicación**: Desarrollo web en entorno servidor.
  - **Descripción**: PHP se utilizó para implementar la lógica del servidor, manejar las operaciones CRUD (crear, leer, actualizar, eliminar) y gestionar la interacción con la base de datos MySQL. Se configuró el entorno PHP en un servidor LAMP (Linux, Apache, MySQL, PHP) alojado en una Raspberry Pi.
- **Angular**
  - **Aplicación**: Desarrollo del frontend.
  - **Descripción**: Angular se empleó para desarrollar la interfaz de usuario de SESAMO como una Single Page Application (SPA). Se utilizaron componentes de Angular Material para construir una interfaz moderna, responsiva e intuitiva que replica las funcionalidades de la aplicación Mambrino.
- **MySQL**
  - **Aplicación**: Gestión de bases de datos.
  - **Descripción**: MySQL se usó para almacenar y gestionar todos los datos de la aplicación, incluyendo la información de usuarios, pacientes y otros datos relevantes.

La base de datos se diseñó para asegurar la integridad y eficiencia en el acceso a la información.

- **Bootstrap**
  - **Aplicación**: Diseño y estilos del frontend.
  - **Descripción**: Bootstrap facilitó la creación de una interfaz responsiva y estética. Se utilizaron sus componentes y clases CSS para asegurar una experiencia de usuario coherente y accesible en diferentes dispositivos.
- **Visual Studio Code**
  - **Aplicación**: Entorno de desarrollo integrado (IDE).
  - **Descripción**: Visual Studio Code fue la principal herramienta de desarrollo, utilizada para escribir, depurar y desplegar el código. Las extensiones para Angular, HTML, CSS, JavaScript y PHP mejoraron la productividad y eficiencia en el desarrollo.

- **Git y GitHub**
  - **Aplicación**: Control de versiones y repositorio de código.
  - **Descripción**: Git se utilizó para el control de versiones, facilitando la gestión y seguimiento de cambios en el código fuente. GitHub sirvió como repositorio centralizado, permitiendo la colaboración y el manejo del historial de versiones.

- **PHPMailer**
  - **Aplicación**: Envío de correos electrónicos.
  - **Descripción**: PHPMailer se implementó para la funcionalidad de recuperación de contraseñas, permitiendo el envío seguro de correos electrónicos con enlaces para restablecer las contraseñas de los usuarios.

- **Certbot**
  - **Aplicación**: Gestión de certificados SSL/TLS.
  - **Descripción**: Certbot se utilizó para obtener y renovar certificados SSL/TLS de Let’s Encrypt, asegurando que todas las comunicaciones con el servidor fueran seguras mediante HTTPS.

- **Bash**
  - **Aplicación**: Automatización de tareas y scripts.
  - **Descripción**: Se crearon scripts en Bash para la eliminación automática de tokens de autenticación y códigos de recuperación de contraseñas expirados, mejorando la seguridad y eficiencia del sistema.

- **Canvas y Visual Paradigm**
  - **Aplicación**: Diseño visual y diagramas.
  - **Descripción**: Estas herramientas se utilizaron para crear diagramas de flujo, modelos de entidad-relación y otros diseños visuales necesarios para la planificación y documentación del proyecto.

Anexo 2: Eliminación de tokens

Para eliminar los tokens generado para los usuarios que se autentican como los códigos de recuperación de contraseña, vamos a usar bash.

Tokens de usuarios:

- Borra los tokens de usuario cuya fecha es 3 días menor a la actual.

```
jhaxtic@sesamo:/etc/sesamo/bash $ cat tokenU.sh
#!/usr/bin/bash

# datos del servidor
DB_USER="root"
DB_PASSWORD="..1791.."
DB_NAME="sesamo"
DB_HOST="localhost"

# Ruta del archivo de log
LOG_FILE="/etc/sesamo/bash/eliminar_tokens_antiguos.log"

# eliminar tokens de usuarios que tengan más de 3 días
SQL_QUERY="DELETE FROM usuarios_token WHERE Fecha < NOW() - INTERVAL 3 DAY;"

# Ejecutar el comando SQL y registrar la salida y errores en el archivo de log
{
  echo "[$(date)] Ejecutando eliminación de tokens antiguos."
  mysql -u$DB_USER -p$DB_PASSWORD -h$DB_HOST $DB_NAME -e "$SQL_QUERY"
  echo "[$(date)] Completado."
} >> $LOG_FILE 2>&1
```

Códigos de recuperación:

- Borra los códigos de recuperación cuya fecha de expiración es menor que la actual (tiene en cuenta las horas, minutos y segundos)

```
jhaxtic@sesamo:/etc/sesamo/bash $ cat tokenR.sh
#!/usr/bin/bash

# datos del servidor
DB_USER="root"
DB_PASSWORD="..1791.."
DB_NAME="sesamo"
DB_HOST="localhost"
# ruta para lso og
LOG_FILE="/etc/sesamo/bash/eliminar_codigos_recuperacion.log"

# eliminar cdigos de recuperacion de mas de 30 min, se gaurdan, comprobamos con la ctual
SQL_QUERY="DELETE FROM recuperarContrasena WHERE expiracion < NOW();"

# vamos a registralos comands
{
  echo "[$(date)] Eliminando codigos de recuperacion antiguos."
  mysql -u$DB_USER -p$DB_PASSWORD -h$DB_HOST $DB_NAME -e "$SQL_QUERY"
  echo "[$(date)] Completado."
} >> $LOG_FILE 2>&1
```

Ambos generaran logs para tener un seguimiento

```
[jue 13 jun 2024 19:05:01 CEST] Completado.
[jue 13 jun 2024 19:06:01 CEST] Eliminando codigos de recuperacion antiguos.
[jue 13 jun 2024 19:06:01 CEST] Completado.
```

Y para automatizar estas tares, usaremos crontab, el escript de borrar códigos se debe ejecutar cada minuto y el de borrar tokens cada día a las 00:00