

UNIVERSIDAD DE MONTERREY

Bases de Datos Avanzada

The logo of the Universidad de Monterrey (UDEM) is displayed. It consists of the letters "UDEM" in a bold, dark blue, sans-serif font, centered on a bright yellow rectangular background.

UDEM

Anteproyecto Inicial

Diego Patricio Linares Navarrete - 632231

Ricardo Arath Sánchez Aguirre - 583928

“Damos nuestra palabra de que hemos realizado esta actividad con integridad académica.”

Profesor: Dr. Raúl Moráles Salcedo

Monterrey, N.L. 19 de Mayo de 2023

Problema 4

Una empresa de atención médica necesita una base de datos para almacenar información sobre pacientes, médicos, citas, tratamientos y facturación. Además, requiere un sistema web que permita a los pacientes programar citas, seleccionar especialidades y médicos, ver horarios, y ver su historial clínico, además de ver sus resultados de exámenes y hacer pagos en línea.

Nombre de la empresa: Hospital El Regio

ÍNDICE

SECTOR	4
ANÁLISIS DEL PROBLEMA	5
OBJETIVO GENERAL Y OBJETIVOS PARTICULARES	6
DETALLES Y REQUERIMIENTOS	7
FUNCIONALIDADES	9
DIAGRAMA ER Y DICCIONARIO DE DATOS	13
NORMALIZAR BASE DE DATOS	16
CREACIÓN BASE DE DATOS	17
DATOS INICIALES	19
SENTENCIAS SQL Y EXPLICACIÓN	20
PROCEDIMIENTOS ALMACENADOS Y EXPLICACIÓN	21
DESCRIPCIÓN DE LA APLICACIÓN DESARROLLADA	24
SCREENSHOTS	25
CONCLUSIONES	28

SECTOR

Este proyecto en el sector de la Salud involucra el desarrollo de una base de datos y un sistema web para mejorar la gestión de información médica y la prestación de servicios. Los pacientes podrán programar citas, seleccionar especialidades y médicos, acceder a su historial clínico, resultados de exámenes y realizar pagos en línea. Esto mejora la atención al paciente, automatiza procesos administrativos y garantiza la seguridad de los datos. El objetivo es proporcionar una experiencia eficiente y centrada en el paciente, mejorando la calidad de la atención médica.

ANÁLISIS DEL PROBLEMA

Una empresa de atención médica necesita una base de datos para almacenar información sobre pacientes, médicos, citas, tratamientos y facturación. Además, requiere un sistema web que permita a los pacientes programar citas, seleccionar especialidades y médicos, ver horarios, y ver su historial clínico, además de ver sus resultados de exámenes y hacer pagos en línea.

OBJETIVO GENERAL Y OBJETIVOS PARTICULARES

Objetivo General

Implementar en una página web sobre una empresa de atención médica, que incluya una base de datos segura para el almacenamiento y gestión de información de pacientes, médicos, citas, tratamientos y facturación, así como un sistema web accesible y fácil de usar que permita a los pacientes programar citas, seleccionar especialidades y médicos, ver horarios, su historial clínico, resultados de exámenes y hacer pagos en línea, mejorando así la eficiencia y la calidad de la atención médica prestada.

Objetivos Específicos

- Como usuario reservar citas
- Ver horarios de los diferentes médicos
- Ver médicos disponibles
- Ver resultados de exámenes
- Introducir su historial clínico
- Ver su historial clínico
- Poder realizar pagos en línea (ya sean citas o exámenes)

DETALLES Y REQUERIMIENTOS

REQUERIMIENTOS	DETALLES
Una base de datos	Se requiere una base de datos para almacenar la información de pacientes, médicos, citas, tratamientos y facturación. La base de datos debe ser segura y cumplir con los estándares de privacidad y protección de datos personales de la industria de la atención médica. Además, debe ser fácil de administrar y escalable en caso de que la empresa crezca.
Pagina web	Se necesita un sistema web que permita a los pacientes programar citas, seleccionar especialidades y médicos, ver horarios, ver su historial clínico y resultados de exámenes, y hacer pagos en línea. El sistema web debe ser fácil de usar y accesible desde cualquier dispositivo con conexión a internet. Además, debe ser seguro y proteger la privacidad de los pacientes.
Programar Citas	El sistema web debe permitir a los pacientes programar citas en línea

	seleccionando la especialidad médica y el médico deseado. También debe permitir a los pacientes ver los horarios disponibles y reservar una cita en el horario que les convenga.
Ver historial clínico y resultados de exámenes	El sistema web debe permitir a los pacientes ver su historial clínico y los resultados de los exámenes realizados. Además, los pacientes deben poder descargar y compartir estos registros con otros médicos si es necesario.
Realizar pagos en línea	El sistema web debe permitir a los pacientes realizar pagos en línea de forma segura y fácil. Debe aceptar diferentes formas de pago, como tarjetas de crédito, débito y transferencias bancarias.

REQUERIMIENTOS NO FUNCIONALES

Seguridad y privacidad	El sistema web y la base de datos deben cumplir con los estándares de seguridad y privacidad de la industria de la atención médica. Deben proteger la información personal y médica de los pacientes y garantizar que solo se compartan con fines médicos legítimos.
Estabilidad	El sistema web y la base de datos deben ser escalables para adaptarse al crecimiento futuro de la empresa de atención médica. Esto implica que la base de datos debe ser capaz de manejar grandes cantidades de información y el sistema web debe ser capaz de soportar un gran número de usuarios concurrentes.

FUNCIONALIDADES

Base de datos:

- Almacenamiento de información de pacientes, médicos, citas, tratamientos y facturación.
- Relaciones entre pacientes, médicos y citas (por ejemplo, un paciente puede tener múltiples citas con diferentes médicos).
- Funcionalidad de búsqueda para permitir a los usuarios encontrar información específica.
- Seguridad para proteger la información confidencial de pacientes y médicos.

Sistema web:

- Registro de pacientes para permitirles programar citas, seleccionar especialidades y médicos, ver horarios y ver su historial clínico.
- Interfaz de usuario intuitiva y fácil de usar para que los pacientes puedan navegar fácilmente por el sistema web.
- Funcionalidad de búsqueda para permitir a los pacientes encontrar médicos y especialidades específicas.
- Integración con el calendario de citas para permitir a los pacientes programar y cancelar citas.
- Acceso a resultados de exámenes y a facturación en línea.
- Seguridad para proteger la información confidencial de pacientes y médicos.
- Integración con sistemas de pago en línea para permitir a los pacientes pagar facturas y hacer transacciones financieras de manera segura y fácil.

FLUJOGRAMA INICIAL DEL SISTEMA CON UI

<https://www.figma.com/file/oMzFD1P2l3S5y3wZXztM0M/Proyecto-Hospital-El-Regio?node-id=0%3A1&t=pv5I6liGdSXgzpUE-1>

FLUJOGRAMA PARA PROBAR EN EL FIGMA**

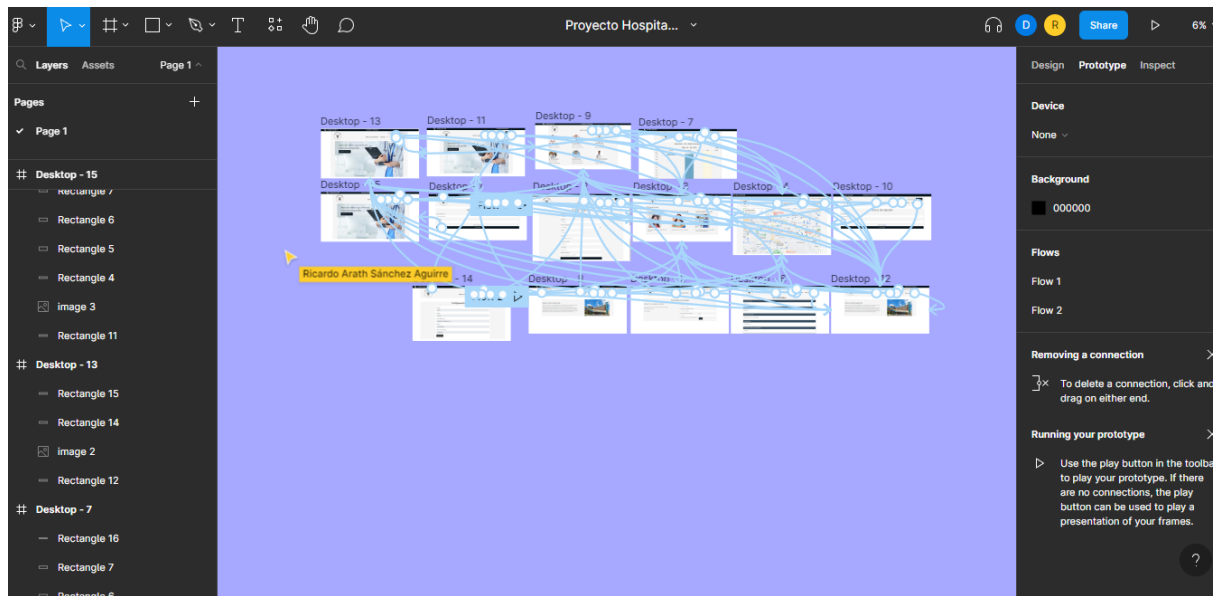
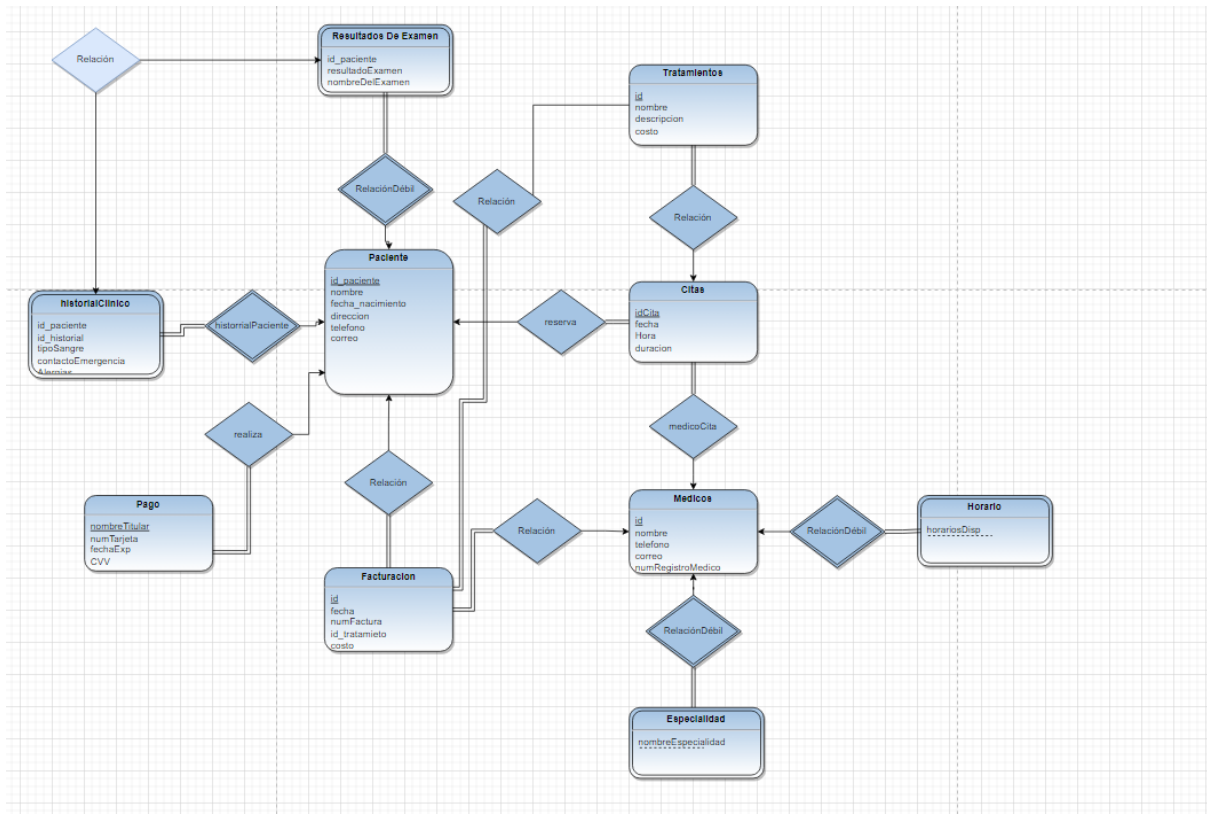


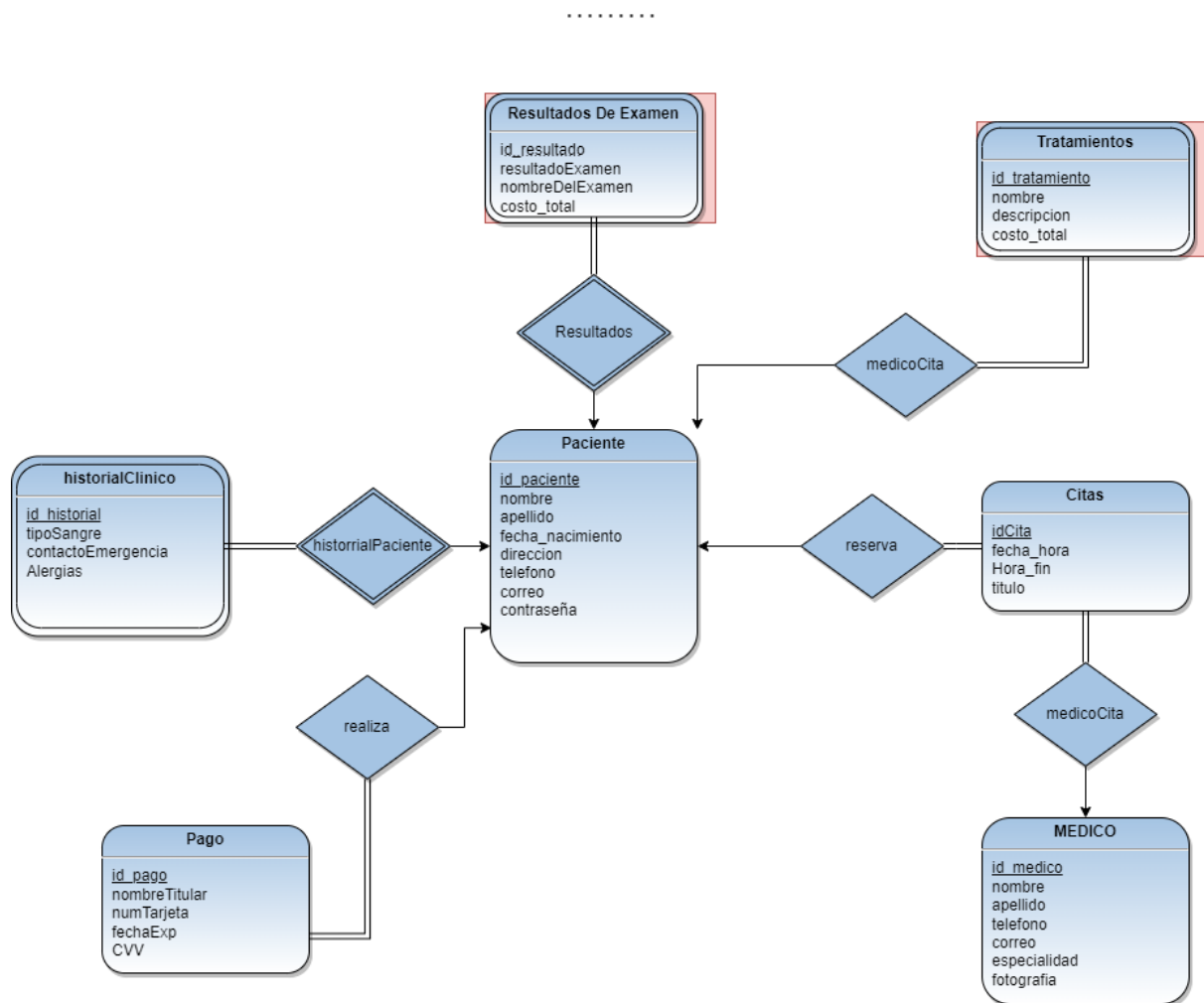
DIAGRAMA ER Y DICCIONARIO DE DATOS

DISEÑO INICIAL DEL DIAGRAMA E-R



DISEÑO FINAL DEL DIAGRAMA E-R

link: <https://app.diagrams.net/#G1DV9Ogi-LaYg-4lm4B2cox6LQaDoL8sD>



PACIENTE:

- **id_paciente:** (int) identificador único del paciente.
- **nombre:** (string) nombre completo del paciente.
- **apellido** (string) nombre completo del paciente.
- **fecha_nacimiento:** (date) fecha de nacimiento del paciente.
- **direccion:** (string) dirección del paciente.
- **telefono:** (string) número de teléfono del paciente.
- **correo:** (string) correo electrónico del paciente.
- **contraseña** (string) contraseña del paciente

MEDICO:

- **idMedico:** (int) identificador único del médico.
- **nombre:** (string) nombre completo del médico.
- **telefono:** (string) número de teléfono del médico.
- **correo:** (string) correo electrónico del médico.

- **especialidad (string)** especialidad médico
- **MedicoFoto (string)** foto del medico.

CITAS:

- **idCita: (int)** identificador único de la cita.
- **fecha_hora: (datetime)** fecha de la cita.
- **hora_fin: (datetime)** hora de la cita.
- **title: (string)** título cita
- **id_medico (int)** id medico
- **id_paciente (int)** id del paciente

HISTORIAL CLINICO:

- **id_historial : (int)** identificador único de el historial
- **id_paciente: (int)** identificador único del paciente.
- **enfermedadesPrevias: (string)** tipo de sangre del paciente
- **medicamentos: (string)** tipo de sangre del paciente
- **cirugias: (string)** tipo de sangre del paciente
- **antecedentes: (string)** tipo de sangre del paciente
- **tipoSangre: (string)** tipo de sangre del paciente
- **alergias:(string)** alergias a medicamentos etc

PAGO:

- **id_pago: (int)** identificador único del pago.
- **id_paciente: (int)** identificador único del paciente.
- **nombreTitular: (string)** nombre completo del titular de la tarjeta
- **numTarjeta(int):** numero de la tarjeta de debito o credito
- **fechaExp(int):** fecha de expiracion de la tarjeta
- **CVV(int):** numero de seguridad de la tarjeta

RESULTADOS_EXAMEN

- **id_resultado: (int)** identificador único del resultado.
- **id_paciente: (int)** identificador único del paciente.
- **resultadoExamen(string):** puede ser una breve descripcion de los resultados
- **nombreDelExamen(string):** sobre que fue el examen o el nombre del tratamiento
- **total(float)** precio final.

TRATAMIENTO

- **id_tratamiento:** (int) identificador único del resultado.
- **id_paciente:** (int) identificador único del paciente.
- **descripcion**(varchar) descripción
- **nombre**(varchar) nombre del tratamiento.
- **total**(float) precio final.

DEPENDENCIAS FUNCIONES

PACIENTE:

id_paciente -> {nombre, apellido, fecha_nacimiento, direccion, telefono, correo, contraseña}

MEDICO:

idMedico -> {nombre, telefono, correo, especialidad, MedicoFoto}

CITAS:

idCita -> {fecha_hora, hora_fin, title, id_medico, id_paciente}

HISTORIAL CLINICO:

id_historial -> {id_paciente, enfermedadesPrevias, medicamentos, cirugias, antecedentes, tipoSangre, alergias}

PAGO:

id_pago -> {id_paciente, nombreTitular, numTarjeta, fechaExp, CVV}

RESULTADOS_EXAMEN:

id_resultado -> {id_paciente, resultadoExamen, nombreDelExamen, total}

TRATAMIENTO:

id_tratamiento -> {id_paciente, descripcion, nombre, total}

NORMALIZAR BASE DE DATOS

Excel en el zip

CREACIÓN BASE DE DATOS

```
PACIENTE | CREATE TABLE `PACIENTE` (  
  `id_paciente` int(11) NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(255) NOT NULL,  
  `fecha_nacimiento` date DEFAULT NULL,  
  `direccion` varchar(255) NOT NULL,  
  `telefono` varchar(20) NOT NULL,  
  `correo` varchar(255) NOT NULL,  
  `contrasena` varchar(150) NOT NULL,  
  `apellido` varchar(255) NOT NULL,  
  PRIMARY KEY (`id_paciente`)  
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=latin1 |
```

```
MEDICO | CREATE TABLE `MEDICO` (  
  `id_medico` int(11) NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(255) NOT NULL,  
  `telefono` varchar(20) NOT NULL,  
  `especialidad` varchar(255) NOT NULL,  
  PRIMARY KEY (`id_medico`)  
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1 |
```

```
CITAS | CREATE TABLE CITAS (  
  id_cita INT(11) AUTO_INCREMENT NOT NULL,  
  id_medico INT(11) NOT NULL,  
  id_paciente INT(11) NOT NULL,  
  title VARCHAR(255) NOT NULL,  
  Fecha_Hora DATETIME NOT NULL,  
  Hora_Fin DATETIME NOT NULL,  
  PRIMARY KEY(id_cita),  
  FOREIGN KEY(id_medico) REFERENCES MEDICO(id_medico),  
  FOREIGN KEY(id_paciente) REFERENCES PACIENTE(id_paciente)  
);  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

```
FACTURACION | CREATE TABLE `FACTURACION` (  
  `id_factura` int(11) NOT NULL AUTO_INCREMENT,  
  `fecha` date NOT NULL,  
  `cliente_id` int(11) NOT NULL,  
  `producto_id` int(11) NOT NULL,  
  `cantidad` int(11) NOT NULL,  
  `precio_unitario` decimal(10,2) NOT NULL,  
  `total` decimal(10,2) NOT NULL,  
  PRIMARY KEY (`id_factura`),  
  KEY `cliente_id` (`cliente_id`),  
  CONSTRAINT `FACTURACION_ibfk_1` FOREIGN KEY (`cliente_id`) REFERENCES  
  `PACIENTE` (`id_paciente`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```



```
HISTORIAL_CLINICO | CREATE TABLE `HISTORIAL_CLINICO` (
  `id_historial` int(11) NOT NULL AUTO_INCREMENT,
  `id_paciente` int(11) NOT NULL,
  `fechaNacimiento` date DEFAULT NULL,
  `enfermedadesPrevias` varchar(255) DEFAULT NULL,
  `medicamentos` varchar(255) DEFAULT NULL,
  `cirugias` varchar(255) DEFAULT NULL,
  `antecedentes` varchar(255) DEFAULT NULL,
  `tipoSangre` varchar(10) DEFAULT NULL,
  `alergias` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id_historial`),
  KEY `fk_historial_paciente` (`id_paciente`),
  CONSTRAINT `fk_historial_paciente` FOREIGN KEY (`id_paciente`) REFERENCES
`PACIENTE` (`id_paciente`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1 |
```

```
HORARIO_MEDICO | CREATE TABLE `HORARIO_MEDICO` (
  `id_horario` int(11) NOT NULL,
  `id_medico` int(11) DEFAULT NULL,
  `diaSemana` varchar(10) DEFAULT NULL,
  `horaInicio` time DEFAULT NULL,
  `horaFin` time DEFAULT NULL,
  PRIMARY KEY (`id_horario`),
  KEY `id_medico` (`id_medico`),
  CONSTRAINT `HORARIO_MEDICO_ibfk_1` FOREIGN KEY (`id_medico`) REFERENCES
`MEDICO` (`id_medico`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

```
PAGO | CREATE TABLE `PAGO` (
  `id_pago` int(11) NOT NULL AUTO_INCREMENT,
  `id_paciente` int(11) DEFAULT NULL,
  `nombreTitular` varchar(255) DEFAULT NULL,
  `numTarjeta` int(11) DEFAULT NULL,
  `fechaExp` date DEFAULT NULL,
  `CVV` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_pago`),
  KEY `fk_pago_paciente` (`id_paciente`),
  CONSTRAINT `fk_pago_paciente` FOREIGN KEY (`id_paciente`) REFERENCES `PACIENTE`
(`id_paciente`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

```
RESULTADOS_EXAMEN | CREATE TABLE `RESULTADOS_EXAMEN` (
  `id_resultado` int(11) NOT NULL AUTO_INCREMENT,
  `id_paciente` int(11) DEFAULT NULL,
  `nombreDelExamen` varchar(255) DEFAULT NULL,
  `resultadoExamen` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id_resultado`),
  KEY `fk_resultados_paciente` (`id_paciente`),
  CONSTRAINT `fk_resultados_paciente` FOREIGN KEY (`id_paciente`) REFERENCES
`PACIENTE` (`id_paciente`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

```
TRATAMIENTO | CREATE TABLE `TRATAMIENTO` (
  `id_tratamiento` int(11) NOT NULL AUTO_INCREMENT,
```

```
`nombre` varchar(255) NOT NULL,  
`descripcion` varchar(255) NOT NULL,  
PRIMARY KEY (`id_tratamiento`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

DATOS INICIALES

```
INSERT INTO HISTORIAL_CLINICO (id_historial, id_paciente, fechaNacimiento,
enfermedadesPrevias, medicamentos, cirugias, antecedentes, tipoSangre, alergias)
VALUES (3, 3, '2023-05-12', 'Neumonia', 'Genoprazol, Ketorolaco, Paracetamol', 'Cirugia de
Riñon', 'Diabetes', 'A+', 'Alergia al Polvo');
```

```
INSERT INTO HISTORIAL_CLINICO (id_historial, id_paciente, fechaNacimiento,
enfermedadesPrevias, medicamentos, cirugias, antecedentes, tipoSangre, alergias)
VALUES (4, 6, '2023-05-02', 'las gomitas', 'paracetamol', 'cirugia de estomago', 'diabetes tipo
3', 'O negativa', 'Al polen');
```

SENTENCIAS SQL Y EXPLICACIÓN

```
"SELECT * FROM HISTORIAL_CLINICO WHERE id_paciente = %s"
```

En este caso, busca seleccionar todos los registros de la tabla "HISTORIAL_CLINICO" donde el valor de la columna "id_paciente" sea igual al valor proporcionado.

```
"SELECT nombre, id_medico FROM MEDICO WHERE id_medico = %s;"
```

En este caso, busca seleccionar los campos "nombre" e "id_medico" de la tabla "MEDICO" donde el valor de la columna "id_medico" sea igual al valor proporcionado.

```
"SELECT id_cita, id_medico, id_paciente, title, Fecha_Hora, Hora_Fin FROM CITAS  
WHERE id_medico = %s;":
```

En este caso, busca seleccionar los campos "id_cita", "id_medico", "id_paciente", "title", "Fecha_Hora" y "Hora_Fin" de la tabla "CITAS" donde el valor de la columna "id_medico" sea igual al valor proporcionado.

```
"SELECT id_tratamiento, nombre, descripcion, total FROM TRATAMIENTO WHERE  
id_paciente=%s"
```

Esta es la consulta SQL que se ejecutará. En este caso, busca seleccionar los campos "id_tratamiento", "nombre", "descripcion" y "total" de la tabla "TRATAMIENTO" donde el valor de la columna "id_paciente" sea igual al valor proporcionado.

```
"SELECT id_resultado, nombreDelExamen, resultadoExamen, total FROM  
RESULTADOS_EXAMEN WHERE id_paciente=%s"
```

Esta es la consulta SQL que se ejecutará. En este caso, busca seleccionar los campos "id_resultado", "nombreDelExamen", "resultadoExamen" y "total" de la tabla "RESULTADOS_EXAMEN" donde el valor de la columna "id_paciente" sea igual al valor proporcionado.

PROCEDIMIENTOS ALMACENADOS Y EXPLICACIÓN

getHistorialClinico

```
DELIMITER $$
```

```
CREATE PROCEDURE GetHistorialClinico(IN id_paciente INT)
```

```
BEGIN
  SELECT * FROM HISTORIAL_CLINICO WHERE id_paciente = id_paciente;
END $$
```

DELIMITER ;

DELIMITER \$\$

Este procedimiento recibe como entrada el ID de un paciente y devuelve todos los registros de su historial clínico almacenados en la tabla "HISTORIAL_CLINICO". Es útil para obtener toda la información médica relacionada con un paciente específico.

InsertIntoHistorialClínico

```
CREATE PROCEDURE InsertIntoHistorialClinico(IN p_id_paciente INT, IN
p_fechaNacimiento DATE, IN p_enfermedadesPrevias VARCHAR(255), IN p_medicamentos
VARCHAR(255), IN p_cirugias VARCHAR(255), IN p_antecedentes VARCHAR(255), IN
p_tipoSangre VARCHAR(255), IN p_alergias VARCHAR(255))
BEGIN
  INSERT INTO HISTORIAL_CLINICO(id_paciente, fechaNacimiento,
enfermedadesPrevias, medicamentos, cirugias, antecedentes, tipoSangre, alergias)
  VALUES(p_id_paciente, p_fechaNacimiento, p_enfermedadesPrevias, p_medicamentos,
p_cirugias, p_antecedentes, p_tipoSangre, p_alergias);
END $$
```

DELIMITER ;

Este procedimiento se encarga de insertar nuevos registros en la tabla "HISTORIAL_CLINICO". Recibe varios parámetros que representan la información médica de un paciente, como su ID, fecha de nacimiento, enfermedades previas, medicamentos, cirugías, antecedentes, tipo de sangre y alergias. Estos valores se insertan en la tabla correspondiente, permitiendo almacenar y mantener actualizado el historial clínico del paciente.

getConfig

DELIMITER \$\$

```
CREATE PROCEDURE GetPaciente(IN p_id_paciente INT)
BEGIN
  SELECT * FROM PACIENTE WHERE id_paciente = p_id_paciente;
END $$
```

DELIMITER ;

Este procedimiento se utiliza para obtener la configuración del sistema. Puede ser útil para obtener información relevante sobre la configuración actual del sistema, como opciones de personalización, valores predeterminados o ajustes específicos.

getPacienteCorreo

```
DELIMITER //
CREATE PROCEDURE GetPacienteCorreo(IN correo_param VARCHAR(255))
BEGIN
    SELECT id_paciente, correo, contrasena FROM PACIENTE WHERE correo =
correo_param;
END //
DELIMITER ;
```

Este procedimiento recibe como entrada un correo electrónico y busca en la tabla "PACIENTE" si existe algún registro con ese correo. Si se encuentra una coincidencia, se devuelve el ID del paciente, su correo electrónico y contraseña. Este procedimiento es útil para buscar información de un paciente en función de su dirección de correo electrónico.

GetAllMedicos

```
DELIMITER //
CREATE PROCEDURE GetAllMedicos()
BEGIN
    SELECT nombre, especialidad, telefono, id_medico FROM MEDICO;
END //
DELIMITER ;
```

Este procedimiento se encarga de obtener información de todos los médicos registrados en la tabla "MEDICO". Devuelve los nombres, especialidades, números de teléfono y IDs de todos los médicos almacenados en la base de datos. Es útil para obtener una lista completa de médicos y sus detalles asociados

DESCRIPCIÓN DE LA APLICACIÓN DESARROLLADA

La página es un sistema web de gestión médica relacionada con la atención médica. La página está diseñada para permitir a los usuarios (pacientes) realizar diversas acciones, como:

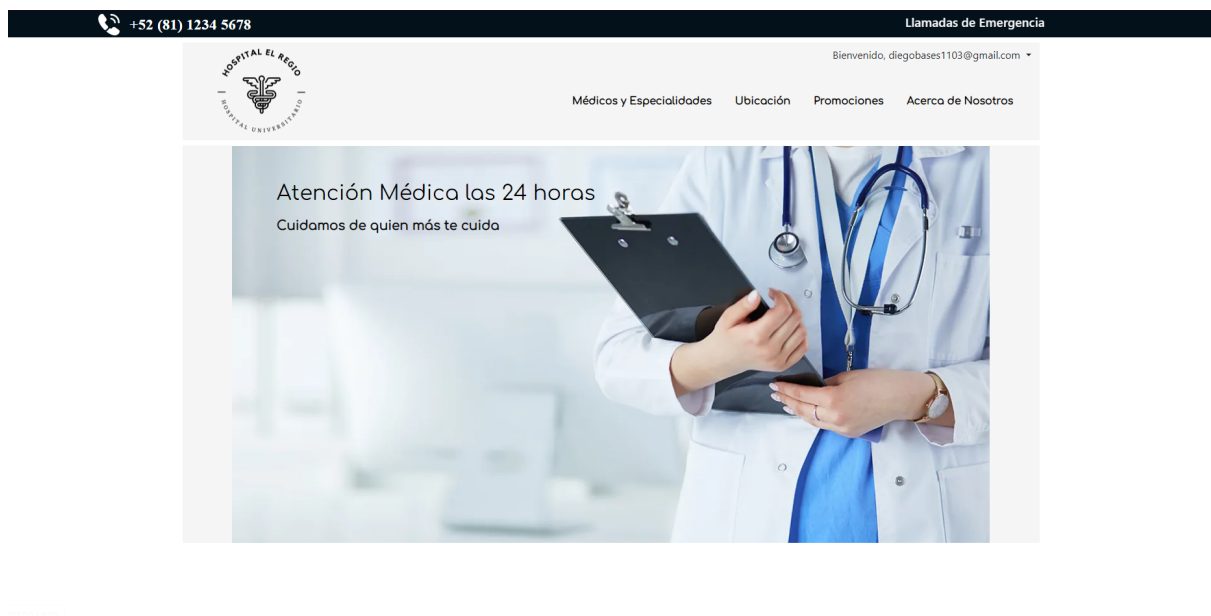
- Iniciar sesión y cerrar sesión.
- Registrarse como nuevos usuarios.
- Configurar y gestionar su perfil de usuario.
- Ver y registrar su historial médico.
- Realizar pagos relacionados con tratamientos médicos.
- Programar citas con médicos.
- Ver horarios de médicos y sus calendarios de citas.
- Ver resultados de exámenes médicos.
- Explorar información sobre médicos y promociones.
- Acceder a secciones de información adicional, como "Acerca de nosotros", "Ubicación" y "Promociones".

Además, la aplicación ofrece una interfaz intuitiva y fácil de usar, con un diseño moderno y atractivo. Utiliza tecnologías como Flask, Jinja2 y MySQL para la

gestión de la base de datos y la interacción con el backend. Se enfoca en brindar una experiencia conveniente y eficiente para los pacientes, permitiéndoles acceder y administrar su información médica de manera segura y cómoda desde cualquier lugar.

SCREENSHOTS

INDEX



UBICACIÓN

Promociones

 +52 (81) 1234 5678

Llamadas de Emergencia

Bienvenido, diegobases1103@gmail.com ▾

 Médicos y Especialidades Ubicación Promociones Acerca de Nosotros

Promociones

TARJETA BIENESTAR



20% de descuento al pagar con tarjeta "Banco del Bienestar".

EXAMEN DE SANGRE



Descuento del 15% en examen de sangre.

MEDICAMENTOS 2x1



2x1 en medicamentos. Ahorra y cuida tu salud en nuestro hospital. (50 palabras)

Pago

 +52 (81) 1234 5678

Llamadas de Emergencia

Bienvenido, diegobases1103@gmail.com ▾

 Médicos y Especialidades Ubicación Promociones Acerca de Nosotros

Formulario de Pago

Selecciona el pago a realizar

☐ Tratamiento 1 - Este es un tratamiento de ejemplo

☒ Examen de Cancer

Total: 149999


Nombre en la tarjeta*

Número de la tarjeta de crédito*


Fecha de expiración (MM/AA)*

CVV*

Por favor, ingrese la fecha en el formato MM/AA


+52 (81) 1234 5678

Llamadas de Emergencia



Bienvenido, diegobases1103@gmail.com

Médicos y Especialidades
Ubicación
Promociones
Acerca de Nosotros

Historial Médico

Fecha de Nacimiento

2023-05-12

Enfermedades Previas

Neumonía

Medicamentos Actuales

Genoprazol, Ketorolaco, Paracetamol

Cirugías Previas

Cirugía de Riñón

Antecedentes Familiares de Enfermedades

Acerca de Nosotros


+52 (81) 1234 5678

Llamadas de Emergencia



Bienvenido, diegobases1103@gmail.com

Médicos y Especialidades
Ubicación
Promociones
Acerca de Nosotros

Acerca de nosotros

Desde su fundación en 1934, "El Regio" ha permanecido a la vanguardia de la medicina en Nuevo León. A 89 años de su creación, hoy es líder en especialidades y subespecialidades, cuenta con infraestructura hospitalaria de última generación, y es reconocido por ofrecer atención médica de gran complejidad con alto sentido humano. Durante 2023, habilitamos 10 cubículos de terapia respiratoria de oxigenación por membrana extracorpórea.



CONCLUSIONES

En conclusión, nos sentimos muy entusiasmados por realizar este proyecto ya que en el futuro nos puede servir de mucho al ya tener algo de experiencia de La integración de una base de datos a una página web, esto requiere de habilidades de programación y gestión de bases de datos para

asegurar una funcionalidad adecuada y una experiencia del usuario satisfactoria. Al igual que entender mejor el funcionamiento de tecnologías como Python, Flask, aplicar el concepto de usar un framework y facilitar el proceso de diseño con Bootstrap y css, y Jinja 2 para utilizar los datos traídos desde el app.py y poder utilizarlos dentro de los templates.