Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

IMPLEMENTACIÓN DE *SOFTWARE* PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Juan José Ramos Campos

Asesorado por Msc. Lic. Marco Tulio Aldana Prillwitz

Guatemala, febrero 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE *SOFTWARE* PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JUAN JOSÉ RAMOS CAMPOS
ASESORADO POR MSC. LIC. MARCO TULIO ALDANA PRILLWITZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, FEBRERO 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. José Francisco Gómez Rivera (a. i.)
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Ing. Kevin Vladimir Armando Cruz Lorente
VOCAL V	Ing. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. José Francisco Gómez Rivera (a. i.)
EXAMINADORA	Inga. Floriza Felipa Ávila Pesquera de Medinilla
EXAMINADOR	Ing. Sergio Leonel Gómez Bravo
EXAMINADOR	Ing. Carlos Alfredo Azurdia Morales
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

IMPLEMENTACIÓN DE *SOFTWARE* PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería de Ciencias y Sistemas, con fecha 08 de febrero de 2022.

Juan José Ramos Campos

Guatemala, 27 de septiembre del 2023

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Unidad de Ejercicio Profesional Supervisado - EPS
Dirección
Ing. Oscar Argueta Hernández

Estimado director, por este medio hago de su conocimiento que el estudiante **JUAN JOSÉ RAMOS CAMPOS** quien se identifica con el Documento Personal de Identificación -DPI- **2757 88148 2208** extendido por el Registro Nacional de las Personas -RENAP- y número de registro estudiantil **201801262** de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala en la Carrera de Ingeniería en Ciencias y Sistemas. Ha finalizado su informe final del Ejercicio Profesional Supervisado -EPS-, titulado "**IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA".**

Por lo tanto, doy por aprobado su trabajo para continuar con los trámites respectivos. Sin otro particular, me es grato suscribirme

Muy cordialmente

"ID Y ENSEÑAD A TODOS"

Marco Tulio Aldana Prillwitz
Master in Bussiness Intelligence and Data Analytics
Colegio de Humanidades 30747

Msc. Lic. Marco Tulio Aldana Prillwitz
Asesor-Supervisor de EPS
Escuela de Ingeniería en Ciencias y Sistemas
Colegiado 30747

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala, 03 de octubre de 2023.
REF.EPS.D.320.10.2023.

Ing. Carlos Gustavo Alonzo
Director Escuela de Ingeniería Ciencias y Sistemas
Facultad de Ingeniería
Presente

Estimado Ingeniero Alonzo:

Por este medio atentamente le envío el informe final correspondiente a la práctica del Ejercicio Profesional Supervisado, (E.P.S) titulado **IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** que fue desarrollado por el estudiante universitario **Juan José Ramos Campos, Registro Académico 201801262 y CUI 2757 88148 2208** quien fue debidamente asesorado por el Ing. Marco Tulio Aldana Prillwitz y supervisado por la Inga. Floriza Felipa Ávila Pesquera de Medinilla.

Por lo que habiendo cumplido con los objetivos y requisitos de ley del referido trabajo y existiendo la aprobación del mismo por parte del Asesor y la Supervisora de EPS, en mi calidad de Director apruebo su contenido solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,
"Id y Enseñad a Todos"

Universidad de San Carlos de Guatemala
DIRECCIÓN
Unidad de Prácticas de Ingeniería y EPS
Facultad de Ingeniería

Ing. Oscar Argueta Hernández
Director Unidad de EPS

/ra

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala 16 de octubre de 2023

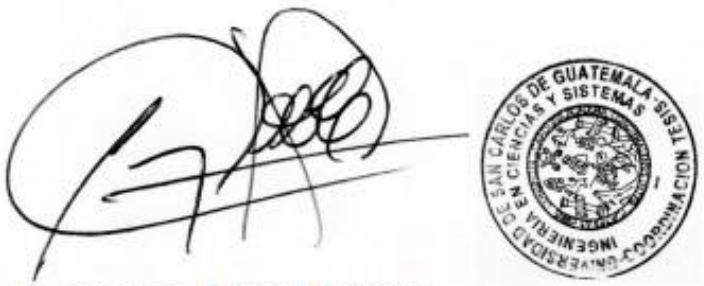
Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación-EPS del estudiante **JUAN JOSÉ RAMOS CAMPOS** carné **201801262** y CUI **2757 88148 2208,** titulado: **"IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,



Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LNG.DIRECTOR.021.EICCSS.2024

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, el visto bueno del Coordinador de área y la aprobación del área de lingüística del trabajo de graduación titulado: **IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**, presentado por: **Juan José Ramos Campos** , procedo con el Aval del mismo, ya que cumple con los requisitos normados por la Facultad de Ingeniería.

"ID Y ENSEÑAD A TODOS"

Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Msc. Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, febrero de 2024

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.067.2024

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** presentado por: **Juan José Ramos Campos,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO a.i.
Facultad de Ingeniería

Ing. José Francisco Gómez Rivera

Decano a.i.

Guatemala, febrero de 2024

JFGR/gaoc

ACTO QUE DEDICO A:

Dios

Quien me ha guiado por el camino correcto, me ha brindado la inteligencia, constancia, quien ha puesto a personas maravillosas que me han apoyado en la vida y me ha permitido lograr una meta más.

Mis padres

Daniel Ramos y Mirian Aracely Campos por ser personas muy trabajadoras, quienes se sacrificaron día y noche para que nada nos faltará a mí y a mis hermanos, nos educaron con mucho amor, para ser personas de bien. Quienes me han apoyado por varios años para que el día de hoy pueda decir: “soy un profesional”.

Mis hermanos

Daniel, Luis y Ana Ramos quienes me han acompañado durante toda esta trayectoria motivándome a no rendirme y me han apoyado de muchas maneras.

Mi novia

Alexandra Cotto por apoyarme incondicionalmente en esta última etapa de mi vida como estudiante y futuro profesional, estando para mí en los momentos donde más lo he necesitado como impulsora para no rendirme y conseguir mi superación personal.

Mis amigos José Morente, Gerson Beltetón, Didier Domínguez, Luis Velásquez, María Juárez, Julissa Velásquez, Luis García, Mario Subuyuj y demás amigos con los que conviví en cada etapa de la Universidad, quienes me brindaron su apoyo y ánimos para continuar al final de la carrera

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por brindarme la oportunidad de estudiar en esta gran casa de estudios y ser orgullosamente San Carlista.
Facultad de Ingeniería	Por brindarme la formación académica, profesional y las herramientas necesarias para ser una mejor persona y así contribuir a nuestro país Guatemala.
Facultad de Odontología	Por permitirme realizar mi Ejercicio Profesional Supervisado permitiendo concluir con mis estudios de manera exitosa.
Asesores	Msc. Lic. Marco Tulio Aldana Prillwitz, Dr. Jorge Orlando Ávila Morales e Inga. Floriza Felipa Ávila Pesquera de Medinilla por su orientación y asesoramiento a lo largo de todo mi proceso de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SIMBOLOS

Símbolo	Significado
$	Dólar
Q	Quetzal

GLOSARIO

Aplicación web **“**Es un *software* que se ejecuta en el navegador web. Las empresas tienen que intercambiar información y proporcionar servicios de forma remota. Utilizan aplicaciones web para comunicarse con los clientes cuando lo necesiten y de una forma segura” (Amazon Web Services, Inc, s.f., p. 1).

API Las *API* son mecanismos que permiten a dos componentes de *software* comunicarse entre sí mediante un conjunto de definiciones y protocolos.

Asíncrono Este término se refiere al concepto de que más de una cosa ocurre al mismo tiempo, o múltiples cosas relacionadas ocurren sin esperar a que la previa se haya completado.

JavaScript *JavaScript* es un lenguaje de programación que los desarrolladores utilizan para hacer páginas web interactivas. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de *JavaScript* pueden mejorar la experiencia del usuario de un sitio *web*. Como lenguaje de *scripting* del lado del servidor, se trata de una de las principales tecnologías de la *World Wide Web* (Amazon Web Services, Inc, s.f., p. 1)

JSON	*JSON* es un formato de texto que forma parte del sistema de *JavaScript* y que se deriva de su sintaxis, pero no tiene como objetivo la creación de programas, sino el acceso, almacenamiento e intercambio de datos. Usualmente es conocido como una alternativa al lenguaje *XML* (Hubspot, s.f., p. 4).
Microservicios	**“**Son un enfoque arquitectónico y organizativo para el desarrollo de *software* donde el *software* está compuesto por pequeños servicios independientes que se comunican a través de *API* bien definidas” (Amazon Web Services, Inc, s.f., p. 1).
Modelo de datos	**“**Un modelo de datos muestra la estructura lógica de una base de datos, incluidas las relaciones y limitaciones que determinan cómo se almacenan los datos y cómo se accede a ellos” (Lucidchart, s.f., p. 1).
Script	El *script* es un documento que contiene instrucciones, escritas en códigos de programación. El *script* es un lenguaje de programación que ejecuta diversas funciones en el interior de un programa de computadora.
XML	*XML* son las siglas de lenguaje de marcado extensible, este lenguaje permite definir y almacenar datos que pueden ser compartidos. *XML* admite el intercambio de información entre sistemas de computación, como sitios web, bases de datos y

aplicaciones de terceros. Las reglas predefinidas facilitan la transmisión de datos como archivos *XML* a través de cualquier red, ya que el destinatario puede usar esas reglas para leer los datos de forma precisa y eficiente (Amazon Web Services, s.f., p. 1).

RESUMEN

Actualmente estamos viviendo la era de la información, es decir, estamos en la época donde la tecnología ha generado un gran impacto en toda la sociedad, no solo en la manera en que vivimos, sino también en la manera en cómo nos educamos o trabajamos. Internet es la tecnología decisiva de la era de la información del mismo modo que el motor eléctrico fue el vector de la transformación tecnológica durante la era industrial.

Para esta época, la información se ha convertido en uno de los recursos más importantes que existen ya que todos los dispositivos son capaces de recopilarla, por ello, las empresas se enfocan cada vez más y más para obtener información relevante para proporcionar mejores servicios a sus clientes. Actualmente y con el auge de las tecnologías web, las empresas han decidido transferir su información a la nube para asegurarse de que dicha información esté plenamente protegida y disponible siempre que sea necesaria.

Por lo anteriormente expuesto, se quiere realizar una aplicación web para la Facultad de Odontología, más concretamente para uso del laboratorio de microbiología, que permita almacenar información de los pacientes que visitan dicho laboratorio y de los exámenes que el laboratorio provee, dicha aplicación será construida de manera que pueda ser escalable, es decir, que los exámenes de laboratorio puedan ir creciendo con el tiempo y a su vez se puedan generar más y más si es necesario.

La aplicación también contará con la opción de generar solicitudes para dichos exámenes de laboratorio y editarlas para colocar sus respectivos

resultados, generar pagos y facturas, un calendario donde se especifiquen las solicitudes vigentes, una sección de manejos de usuarios y roles y por último, una sección estadística para ver resultados de diferentes análisis estadísticos como por ejemplo la cantidad de pacientes en un determinado tiempo, las edades más comunes, entre otros.

Se planea también realizar un manual técnico donde se detalle el proceso para el desarrollo de la aplicación web, incluyendo la arquitectura, funciones, métodos y toda la lógica conllevar para la realización de la aplicación con el fin de que futuros desarrolladores puedan darle soporte en un futuro.

OBJETIVOS

General

Desarrollar una aplicación *web* intuitiva y fácil de usar capaz de llevar el control y gestión de los pacientes que ingresan al laboratorio de microbiología de la Facultad de Odontología de la Universidad de San Carlos de Guatemala, así como también llevar el control de los exámenes clínicos que el laboratorio provee y que los pacientes pueden realizar, generando las solicitudes y realizando los pagos correspondientes.

Específicos

1. Desarrollar una aplicación *web* interactiva, fácil de usar y funcional que permita la realización de todas las tareas necesarias para la correcta administración de los pacientes y exámenes del laboratorio.

2. Desarrollar un *API* basada en microservicios que permita la manipulación de información de manera correcta y que, a su vez, los microservicios sean independientes lo que permite un mejor mantenimiento.

3. Crear protocolos de contingencia para las diferentes partes que interactúan en el sistema, es decir, contar con una forma de verificar si algo está funcionando mal y si es así, reiniciarlo de manera automática.

4. Reestructurar la información existente de acuerdo con el nuevo modelo de datos para una mejor manipulación de esta.

INTRODUCCIÓN

Actualmente vivimos en la época más tecnológica que ha habido, la tecnología de la información ha cambiado la forma de vida de las personas, así como también, la forma de trabajo de estas. En el entendido de que las Tecnologías de la Información y Comunicación se refieren a sistemas y aplicaciones, se dice que esta disciplina importa porque les permite a las personas relacionarse mejor aún a distancia.

Para las empresas, organizaciones o instituciones, la información que manejan es su recurso más valioso, por lo que es de suma importancia que cuenten con altos protocolos y estándares para garantizar que su información sea correctamente resguardada y evitar pérdidas que puedan incurrir en problemas mucho mayores. Toda esa información producida y recibida representa una parte importante para los procesos que realizan, por lo que protegerlos y gestionarlos adecuadamente por medio del almacenamiento, representa una tarea importante.

Por lo anterior, el correcto almacenamiento de información se posiciona como la estrategia más necesaria de implementación en las empresas ya que permite entre otras cosas, garantizar el acceso a la información, conservar la información en óptimas condiciones para su consulta, mantener un orden lógico y práctico y por último ahorrar tiempos de búsqueda de esta, por lo que sus procesos de trabajo se ven optimizados y se vuelven más eficientes.

1. FASE DE INVESTIGACIÓN

1.1. Antecedentes de la organización

Previo al desarrollo del proyecto se realizó una investigación preliminar de la institución en la cual se iba a trabajar, en este caso la Facultad de Odontología, más concretamente el laboratorio de microbiología, esta decisión se tomó con la finalidad de tener un contexto más amplio de la institución es decir que hace, como lo hace y desde cuando lo hace.

Con esto se pretendía entender mejor la forma de trabajo de la institución y alinearlo con los requerimientos solicitados para el proyecto y, poder así, presentar un producto de alta calidad, funcional y que cumpla con las necesidades de esta.

1.1.1. Reseña histórica

Los estudios de odontología se iniciaron en Guatemala en forma organizada con la fundación del Instituto Dental como una dependencia de la Facultad de medicina, Cirugía y Farmacia, el 1 de mayo de 1895, por decreto legislativo No. 297. La universidad de San Carlos de Guatemala funcionaba en ese entonces bajo la dirección del Ministerio de Instrucción Pública. En 1926 al producirse la reorganización de la universidad, con la separación de la Facultad de Medicina y Cirugía de la de Farmacia, fue establecida la escuela de Odontología como una unidad de la Facultad de Ciencias Médicas. Posteriormente el 1 de abril de 1940, se creó la Facultad de Odontología por

decreto gubernativo No. 2336. Su junta directiva se instaló el 09 de abril y tuvo como sede el edificio que ocupaba anteriormente la Escuela Dental.

De esa manera, la Facultad de odontología desarrolló sus actividades hasta el año 1965, durante el cual se inició una modificación en su plan de estudios que tenía como una de sus principales características la realización sistemática, gradual y creciente de experiencias docentes con la comunidad, concluyendo con la realización del programa de Ejercicio Profesional Supervisado, que vino a constituir el 6º. Año de la carrera.

Como resultado de un seminario realizado en 1995, los participantes, profesores, autoridades y estudiantes, definieron la finalidad de la Facultad de Odontología de la manera siguiente:

> Orientar el proceso de enseñanza-aprendizaje hacia la formación de recursos humanos estomatológicos adecuados para Guatemala, con una base científica sólida y con capacidad para aplicar teórica y prácticamente el enfoque científico y tecnológico para la búsqueda de soluciones a los problemas del ejercicio de la profesión, bajo normas éticas y de servicio que, mediante la aplicación de medidas preventivas e integrales, logren un impacto eficaz en el mejoramiento de la salud bucal de la mayoría de guatemaltecos, contribuyendo con ello a elevar su calidad de vida. (Facultad de Odontología, 2008, p. 1)

Un nuevo seminario realizado en 1998 aprobó, además de lo anterior, la recomendación de que la Facultad de Odontología realice los esfuerzos necesarios para definir sus estudios de posgrado y efectúe las propuestas y programas correspondientes.

1.1.2. Misión

Nuestra misión es formar odontólogos capaces, con excelencia académica, rigor científico y comprometido con el desarrollo sostenible nacional y regional para la prevención, curación y solución de problemas estomatológicos en la población guatemalteca, cumpliendo con las políticas institucionales de la Universidad de San Carlos de Guatemala. (Facultad de Odontología, s.f., p. 1)

1.1.3. Visión

La Facultad de Odontología, identificada y comprometida con los fines, principios y políticas de la Universidad de San Carlos de Guatemala, la facultad forma profesionales en estomatología, altamente capacitados nacional e internacionalmente para la prevención, curación y control de enfermedades bucales de la población. (Facultad de Odontología, s.f., p. 2)

1.2. Descripción de las necesidades

El principal problema que existe dentro del laboratorio es que no se tiene un control regulado de los pacientes que llegan al mismo, existen formularios de inscripción, solicitudes de diferentes tipos y otros documentos pero únicamente en papel, no hay nada en digital, por lo tanto el control de los pacientes y sus expedientes se vuelve más complicado de manejar, por ejemplo si se pierde alguna ficha clínica, si no es archivada en el momento o es archivada en otro

lugar, casos como los anteriores pueden darse lo que causa que exista mucho riesgo de que la información de los pacientes o sus exámenes cuente con un margen de error.

Del caso anterior, al no tener nada en digital también deriva el problema de que pueda perderse información completa no solo de pacientes si no información importante que el laboratorio pueda manejar como por ejemplo las solicitudes mismas, si se perdiera una solicitud, no habría forma de indicar al paciente sus resultados ya que en dicha solicitud está la información básica del paciente.

Otro problema es que, al igual que los pacientes, no se tiene un registro regulado de los pagos realizados por los pacientes mismos, el laboratorio indica al paciente cuánto debe pagar por los exámenes solicitados, el paciente se dirige a alguna de las cajas de la facultad, realiza su pago y vuelve con un comprobante, así sin más, por lo tanto, se sabe que se realizó el pago más, sin embargo, el laboratorio no tiene ningún registro de este y no se sabe cuál es el estado financiero del mismo.

Por último, el laboratorio necesita conocer para hacer algunos análisis estadísticos referente a por ejemplo sus ingresos, las cantidades de pacientes en un periodo de tiempo, exámenes realizados, entre otros. pero debido a la forma de trabajar actual, realizar esto es una tarea muy complicada y requerirá mucho trabajo y tiempo, por lo cual, actualmente es imposible.

1.3. Priorización de las necesidades

Las necesidades del laboratorio de microbiología se priorizan basándose en la generación de valor que hacen los pacientes ya que de ahí parte todo lo

demás, es decir, las solicitudes, pagos, exámenes, y resto de información relevante para el correcto funcionamiento del laboratorio parte de la cantidad de pacientes.

El laboratorio de microbiología atiende a un mínimo de 600 personas anuales, pero debido a la pandemia del COVID-19 el laboratorio tuvo que cesar actividades como el resto de las instituciones o la universidad misma, esto permitió que los encargados de este pudieran enfocarse en recolectar la información del laboratorio y que dicha recolección de datos se tuviera una prioridad alta.

La generación de solicitudes de exámenes también tiene una prioridad alta ya que como principal función del laboratorio es proveer atención a sus pacientes, dichas solicitudes deben incluir el listado de exámenes solicitados y el listado de resultados obtenidos.

Por otro lado, la generación de pagos y la sección estadística tienen una prioridad media ya que son agregados extra que, aunque aportan valor, no es lo más importante dentro de las funcionalidades que deben ser desarrolladas.

Por último, los manuales de usuario y técnico tienen una prioridad media dentro del orden de desarrollo de los entregables, pero es indispensable para que desarrolladores en un futuro tengan una guía para dar soporte a la aplicación que será utilizada por el laboratorio de microbiología de la Facultad de Odontología de la Universidad de San Carlos de Guatemala.

2. FASE TÉCNICO PROFESIONAL

2.1. Descripción del proyecto

El proyecto consiste en la elaboración de una aplicación *web* capaz de llevar una correcta gestión de pacientes, solicitudes de exámenes y exámenes clínicos, entre otras funcionalidades descritas con anterioridad. Para el desarrollo del expediente clínico del paciente, se planea desarrollar un panel de administración que cuente primeramente con un módulo de Pacientes donde exista un formulario para la inscripción del paciente, en este se solicitaron datos generales del mismo como su nombre, teléfono, entre otros. Además, se desarrollará un módulo llamado Antecedente el cuál solicitará datos más específicos como por ejemplo enfermedades recientes o si toma medicamentos. Dichos pacientes pueden gestionarse de manera que permita modificar su información o eliminar a los mismos.

Como parte fundamental de la aplicación se desarrollará un módulo administrativo de Exámenes, dichos exámenes son los que actualmente el laboratorio provee a los pacientes, como Hematología completa o Perfil diabético, entre otros. Para la adición de exámenes únicamente se solicitará el nombre y el precio de este, esto permitirá que la aplicación pueda ser escalable ya que los doctores encargados del laboratorio pueden agregar más exámenes en un futuro y gestionarlos a voluntad. Cada examen a su vez contará con otros dos módulos principales para su uso, estos módulos son opcionales, pero aportan gran valor por su funcionalidad.

El primero modulo será el llamado Perfil clínico, los perfiles representarán por así decirlo un examen más pequeño que forma parte del examen padre desde donde fue creado. Cada perfil, al igual que su padre, necesitará un nombre y un precio opcional, el perfil indicará que laboratorio se hará con las muestras del paciente para, posteriormente, poder guardar los resultados obtenidos. Para entender mejor el flujo descrito con anterioridad se toma como ejemplo el examen llamado Química sanguínea, este examen cuenta con un perfil llamado Perfil renal y este perfil cuenta con un laboratorio llamado Ácido Úrico, por lo tanto, cuando se genere una solicitud para un paciente y esta solicitud incluya el perfil renal, el asistente de laboratorio encargado de realizar los exámenes de laboratorio deberá saber que en listado de resultados deberá venir uno para ácido úrico.

El segundo módulo importante es el llamado Campo resultado, estos campos son los nombres de cada resultado obtenido para cada examen, todos los exámenes cuentan con su lista de campos resultado. Para entender mejor lo descrito anteriormente, se tomará como ejemplo el examen Química sanguínea, este puede tener los campos llamados glucosa y colesterol total, mientras que el examen Hematología, cuenta con los campos llamados linfocitos y monocitos y sobre cada campo se guardaran los resultados obtenidos de la máquina de laboratorio.

El hecho de que no se cuenten con tablas específicas para cada examen laboratorio y sus campos, permite que, como se mencionó anteriormente, la aplicación pueda ser escalable ya que permite agregar o eliminar exámenes a voluntad, así como los resultados que se va a guardar de cada uno de ellos.

Otro módulo de Solicitudes de exámenes clínicos, también llamada ficha clínica, en dicho módulo, el encargado de atender a los pacientes que llegan al laboratorio le mostrará los diferentes perfiles de exámenes disponibles y el paciente podrá elegir cual o cuales se quiere realizar, cada perfil contará con un precio específico y al final se mostrará el total que el estudiante debe pagar, las solicitudes, al igual que los pacientes pueden ser modificadas o eliminadas a voluntad, así como ver su detalle siempre y cuando se cuente con los permisos necesarios.

También se contará con un módulo de Resultados el cual permitirá, como su nombre indica, gestionar los resultados de los de los diferentes exámenes que el paciente se haya realizado, es decir, crear resultados, modificarlo o eliminarlos de ser necesario, cada resultado tiene un detalle específico por lo que cada resultado es totalmente independiente del otro, será necesario desarrollar una funcionalidad que permita tener resultados para cada tipo de examen a realizar.

Se contará con un módulo de Pago donde se permitirá registrar los pagos realizados por los pacientes y observar el listado de pagos realizados, para ello se solicitará el número de boleta o comprobante, el monto pagado y la fecha, así como la referencia a la solicitud pagada y de manera opcional imágenes que sirvan de respaldo o comprobante del pago realizado.

Además de los pagos, se contará con otro módulo llamado Solicitud pago, dicho módulo se utilizará para generar solicitudes de modificación y eliminación de pago, esto para agregar más seguridad a los pagos ya que es información sensible y no se puede alterar dicha información de manera tan sencilla, por ello, solo los usuarios con permisos necesarios podrán aprobar dichas solicitudes para actualizar la información.

Otros módulos por ser desarrollados son Permisos, Roles y Usuarios, los usuarios serán las personas que hagan uso de la aplicación, cada usuario tendrá un rol asociado, los roles pueden ser por ejemplo doctor, encargado de laboratorio, administrador, recepcionista, entre otros., así mismo, cada rol tendrá asociado uno o varios permisos asignados, los permisos determinarán las acciones que un usuario puede realizar, por ejemplo un usuario recepcionista puede agregar pacientes pero no puede modificar la información de los resultados de exámenes.

Toda acción realizada dentro de la aplicación será almacenada utilizando un módulo de Historial, en dicho módulo se podrá observar todas las acciones que se han realizado en cada módulo de la aplicación, quién lo hizo y en qué momento, así como también cada módulo tendrá su historial propio donde también se podrán observar las acciones realizadas, por ejemplo, quien modificó a determinado paciente, quien creo determinada solicitud, entre otros.

Por último, se contará con el módulo Estadístico que permitirá, como su nombre indica, mostrar diferentes resultados estadísticos en formato de gráficas y tablas para que la información sea fácilmente interpretada por los interesados, dichos resultados estadísticos incluirán por ejemplo los exámenes más solicitados, cantidad de pacientes registrados por mes, ingreso monetario total del laboratorio, entre otros

2.2. Investigación preliminar para la solución del proyecto

Previo al inicio del desarrollo del proyecto se realizó una investigación rigurosa tanto del problema que se necesitaba resolver como las herramientas necesarias para hacerlo, esto con el fin de:

- Identificar el problema principal que se presenta en el laboratorio de microbiología y las causas de este.
- Identificar las necesidades por las cuales se requería la construcción de la aplicación.
- Seleccionar las metodologías más idóneas que mejor se adapte a la forma de trabajo del laboratorio para adaptarlas a la aplicación y facilitar dicho trabajo.
- Establecer directrices iniciales antes de comenzar con el desarrollo, es decir, como se trabajarán ciertas áreas o como se modelarán ciertos datos.
- Establecer que acciones se podrán realizar dentro de la aplicación y quienes podrán realizarlas.

Para llegar a lo anteriormente descrito se tomaron un conjunto de acciones entre las cuales figuran:

- Reuniones y toma de requerimientos con la institución interesada para conocer más detalles de su forma de trabajo y que es lo que esperan conseguir.
- Reuniones con el asesor de la escuela de Ciencias y Sistemas para recibir orientación y retroalimentación sobre las herramientas a utilizar y forma de trabajo propuesta.

2.3. Presentación de la solución al proyecto

Para obtener una mejor organización y distribución de cada parte del proyecto se optó por diseñarlo por fases donde en cada fase se desarrolló una parte importante del proyecto, cada fase se describe a continuación:

2.3.1. Fase 1: Modelo de datos

Esta fase incluye tanto la recolección de datos existentes como la construcción de un nuevo modelo o estructura que se adapte a las necesidades identificadas con anterioridad para la aplicación sea funcional y óptima.

Entre los principales modelos identificados están los siguientes:

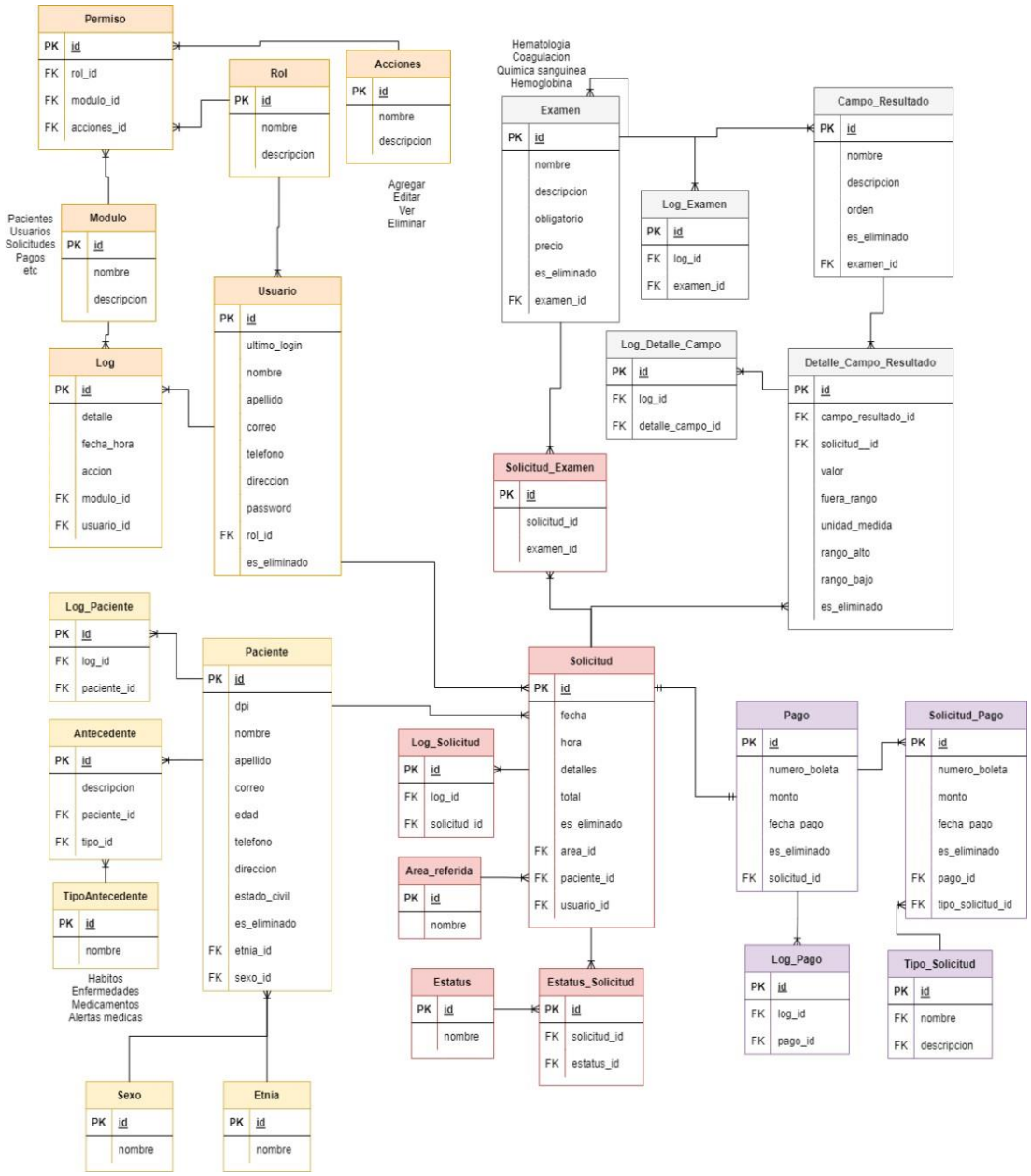
- Módulo: representa las diferentes áreas en las que se divide la aplicación sobre las cuales cada usuario podrá realizar una o varias acciones siempre y cuando tenga los permisos adecuados para hacerlo.
- Acciones: son, como su nombre lo indica, las acciones que se pueden realizar, entre las cuales están Crear datos, Modificar datos, Eliminar datos y Consultar datos.
- Rol: son los diferentes roles que puede tener la aplicación, cada rol tendrá asociados uno o varios permisos y cada usuario tendrá asociado un rol.
- Permiso: almacena los diferentes permisos sobre los que se basa la aplicación para permitir a los usuarios realizar acciones como por ejemplo crear pacientes, editar pacientes, entre otros.
- Usuario: almacena la información de las personas que harán uso de la aplicación y sus diferentes módulos, estas personas son los encargados del laboratorio, asistentes, recepcionista, doctores en general, entre otros.
- Log: guarda la información de las acciones realizadas por un usuario específico sobre un módulo específico, cuando fue realizada dicha acción y que repercusiones o beneficios tuvo sobre la información existente.
- Paciente: almacena la información de los pacientes que llegan al laboratorio, como su nombre, dirección, entre otros.

- Tipo antecedente: guarda la información de las categorías que puede tener los antecedentes clínicos proporcionada por los pacientes, como por ejemplo las enfermedades, hábitos, entre otros.
- Antecedente: guarda información extra de los pacientes que puede ser de utilidad para el laboratorio, por ejemplo, los medicamentos que toma, la dosis, cada cuanto, los hábitos que posee, entre otros.
- Sexo: guarda los sexos de las personas, es decir, masculino y femenino, su utilidad principal es realizar análisis estadístico de los mismos.
- Etnia: guarda las etnias de los pacientes, su utilidad principal es realizar análisis estadísticos de las mismas y poder determinar la etnia de pacientes que más ingresa al laboratorio.
- Estatus: indica el estatus que pueden tener las solicitudes, por ejemplo, rechazada, atendida, pagada, entre otros, el estatus va cambiando a medida que se vayan realizando cambios sobre la solicitud.
- Estatus Solicitud: guarda los diferentes estatus que pueden ser asociados a las solicitudes mientras estas se van procesando o siendo atendidas por los encargados.
- Área referida: guarda las diferentes áreas de los que un paciente puede ser referido como por ejemplo Periodoncia, Prótesis, Docentes, Estudiantes, entre otros.
- Solicitud: esta tabla almacena las diferentes solicitudes de exámenes que los pacientes se pueden realizar dentro del laboratorio. Es una de las secciones más importantes de la aplicación, se podría decir que es la parte central considerando que de las solicitudes parte muchas de las funcionalidades desarrolladas.
- Solicitud examen: almacena la información de que exámenes se realizarán por cada solicitud.
- Pago: esta tabla almacena la información de los pagos realizados por los pacientes para cada solicitud de examen realizada.

- Solicitud pago: esta tabla almacena solicitudes realizadas para los pagos, ya que los pagos son información sensible, no es posible manipular la información, así como así, por lo tanto, es necesario tener solicitudes para aprobar o rechazar un cambio en el mismo de ser necesario.
- Examen: esta es una de las tablas más importantes ya que guarda la información de los diferentes exámenes que puede realizar la facultad de odontología como por ejemplo Hematología completa, Hemoglobina, entre otros. No se guarda una tabla de examen específico para que la aplicación pueda ser escalable, es decir, sea posible agregar más laboratorios en un futuro.
- Campo resultado: almacena los diferentes campos que de los que se quiere guardar información de los resultados obtenidos, por ejemplo, del examen hematológico se espera guardar el campo A mientras que del examen coagulación se espera guardar el campo B, esto permite que cada examen tenga sus propios campos que pueden cambiar a lo largo del tiempo.
- Detalle campo resultado: guarda el detalle de cada campo resultado de cada examen para una solicitud específica.

Figura 1.

Modelo de datos propuesto



Nota. Modelo de dato inicial. Elaboración propia, realizado con Draw.io.

2.3.2. Fase 2: Construcción de aplicación *web*

La aplicación fue construida con la librería de *React* en su versión 18.2.0, escrita en *JavaScript*, está enfocada en la creación de componentes interactivos y reutilizables para interfaces de usuario que a su vez sean visualmente atractivos para los usuarios. (Doonamis, 2021)

Las vistas de la aplicación están divididas en cada uno de los módulos que conforman ésta, seccionando el proyecto completo de una manera que pueda ser explorada fácilmente para el desarrollo.

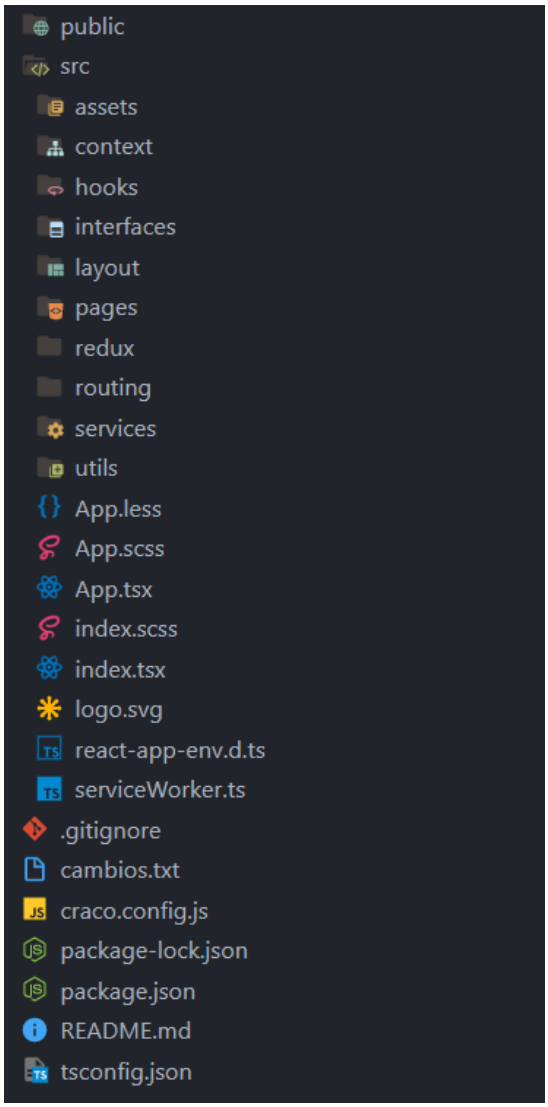
2.3.2.1. Arquitectura de la aplicación web

La característica que puede considerarse la más importante de *React* es que se basa en los llamados componentes, “un componente es una pieza de interfaz de usuario” (Doonamis, 2021, p. 5), que puede ir desde algo tan simple como un botón, una tabla o una imagen a algo mucho más elaborado que incluya funcionalidades y características propias y bien definidas para ese componente.

Cuando se diseña una aplicación utilizando *React*, lo que se crea son componentes independientes y reutilizables para obtener interfaces de usuario más complejas, con esto se consigue diseñar aplicaciones utilizando un paradigma llamado programación orientada a componentes en el que cada componente es una pieza con la que el usuario puede interactuar, se dice que los componentes están bien hechos cuando brindan la “posibilidad de utilizar un componente software en contextos distintos a aquellos para los que fue diseñado” (EcuRed, s.f. p. 9). Es decir, cuando un componente pueda utilizarse de muchas formas y que siga siendo igual de útil independientemente del contexto.

Figura 2.

Estructura de proyecto



Nota. Estructura de archivos y carpetas del código fuente. Elaboración propia, tomado de *Visual Studio Code.*

2.3.2.1.1. Routing

La aplicación incluye una implementación de *React Router DOM*, utilizando el modelo de enrutamiento programático el cual “es el proceso de mantener sincronizadas las *URL* con el contenido de una aplicación, permitiendo controlar el flujo de datos de la aplicación” (García, 2021, p. 10) para moverse entre las diferentes vistas que conforman a la misma.

Las rutas de las vistas deben localizarse dentro de la carpeta *routing* y debe tener una estructura como la siguiente:

Figura 3.

Estructura de rutas

```
import DashboardLayout from "./layouts/Dashboard";
import NewPage from "./pages/NewPage";

const routes = [
  {
    path: "pages",
    element: <DashboardLayout />,
    children: [
      {
        path: "new",
        element: <NewPage />,
      },
    ],
  },
];

export default routes;
```

Nota. Ejemplo de rutas de vistas. Elaboración propia, realizada en *Visual Studio Code.*

El ejemplo anterior dará como resultado que exista una vista en la ruta http://localhost:3000/pages/new a la que es posible acceder como si fuera una dirección de página web común.

Otra forma de moverse entre vistas es mediante programación haciendo uso del *hook useNavigate*, este recibe como parámetro la ruta de destino y se invoca luego de realizar una acción en una función, de la siguiente manera:

Figura 4.

Navegación por programación

```
import { useNavigate } from "react-router-dom";

function ExampleComponent() {
  const navigate = useNavigate();

  const handleSubmit = () => {
    navigate("/pages/new");
  };

  return (
    <form onSubmit={handleSubmit}>
      ...
    </form>
  );
}

export default ExampleComponent;
```

Nota. Ejemplo de movimiento entre rutas mediante programación. Elaboración propia, realizada en *Visual Studio Code*.

2.3.2.1.2. Autenticación

La autenticación provee a la aplicación de una capa de seguridad extra al asegurarse de que los usuarios cuenten con un *token* de acceso que permita principalmente comunicarse con el servidor, con esto se logra que solo los usuarios a los cuales el propio servidor les haya proporcionado un *token* puedan realizar peticiones al mismo.

Para agregar esa capa extra de seguridad se hace uso de *JWT* o *JSON Web Token* el cual es “un estándar abierto que define una forma compacta y autónoma de transmitir información de forma segura entre las partes como un objeto *JSON.*” (JWT.io, s.f., p. 1). Y así mismo “se pueden usar para pasar la identidad de los usuarios autenticados entre el proveedor de identidades y el servicio que solicita la autenticación” (Microsoft, s.f., p. 21)

Dentro de la aplicación, el sistema de seguridad lo habilita el *AuthProvider* que se encuentra dentro del contexto correspondiente, este proveedor se llama en la raíz de la aplicación para lo primero que haga es validar la existencia de un usuario activo o un *token* de acceso, de lo contrario solicitara que se inicie sesión.

Figura 5.

Llamada al proveedor de autenticación

```
import { AuthProvider } from "./contexts/JWTContext";

function App() {
  return (
    <AuthProvider>
      {content}
    </AuthProvider>;
  )
}
```

Nota. Ejemplo de llamada a proveedores. Elaboración propia, realizada en *Visual Studio Code.*

Para poder hacer uso del proveedor, se debe crear un *hook* personalizado que llame al proveedor desde el contexto donde fue creado.

Figura 6.

Hook de autenticación

```
import { AuthContext } from "../contexts/JWTContext";

const useAuth = () => {
  return useContext(AuthContext);
};
```

Nota. Uso del *hook*. Elaboración propia, realizada en *Visual Studio Code.*

Además de asegurarse de que exista una sesión activa, el proveedor, como su nombre indica, puede proveer otras características como retornar al usuario activo o bien retornar funciones para ser ejecutadas en cualquier lugar de la aplicación como se muestra a continuación:

Figura 7.

Recuperar usuario activo

```
import useAuth from '../hooks/useAuth';

const App = () => {
  const { user } = useAuth();

  return (
    <span>
      {user.displayName}
    </span>
  );
};
```

Nota. Llamada de propiedades de un proveedor. Elaboración propia, realizada en *Visual Studio Code.*

Figura 8.

Llamada de funciones desde el proveedor

```
import useAuth from '../hooks/useAuth';

const App = () => {
  const { signIn } = useAuth();

  return (
    <button onClick={() => signIn()}>
      Sign in
    </button>
  );
};
```

Nota. Llamada de función de un proveedor. Elaboración propia, realizada en *Visual Studio Code*.

2.3.2.1.3. Contextos

El contexto proporciona una forma de pasar datos a través del árbol de componentes sin tener que pasar propiedades manualmente en cada nivel. (Reactjs, s.f., p. 1).

En una aplicación típica de *React*, los datos se pasan de arriba hacia abajo (de padre a hijo) a través de *props* o propiedades, pero dicha forma puede ser poco práctica para ciertos tipos de propiedades que son requeridos por muchos componentes dentro de una aplicación. El contexto proporciona una forma de compartir información entre componentes sin tener que pasar explícitamente una propiedad a través de cada nivel del árbol. (Reactjs, s.f., p. 2).

El principal problema de compartir propiedades nivel por nivel es que el estar arrastrando dichas propiedades puede causar que se pierda la referencia de estas entre cada componente o bien, que cuando se haga uso de alguna propiedad ya no interactúe de la manera que se esperaba, para evitar toda esa problemática se hacen uso de los contextos.

El contexto está diseñado para compartir datos que pueden considerarse globales en un árbol de componentes en *React*, como el usuario autenticado actual, el tema o el idioma preferido (Reactjs, s.f., p. 3).

Y eso es precisamente lo que hacen los contextos desarrollados para la aplicación entre los que se encuentran:

- *ConstantesContext*: permite obtener ciertos valores desde el *API* para ser utilizados dentro de la aplicación, estos valores son constantes porque no

son alterables en ningún momento como por ejemplo el sexo, las etnias para pacientes, los módulos de aplicación, entre otros.

- *JWTContext*: es uno de los contextos más importantes de la aplicación, se utiliza para manejar los inicios de sesión y los usuarios registrados de la aplicación, Utiliza *json web tokens* para validar si hay un usuario activo y si su sesión es válida, de lo contrario lo envía al inicio de sesión para ingresar nuevamente. Este contexto tiene varias funciones, como iniciar sesión, crear una cuenta y recuperar contraseñas.
- *MenuContext*: provee el menú de usuario dependiendo de los permisos con lo que es cuente, es decir, si un usuario no tiene permisos para acceder a ciertas áreas, el menú no lo mostrará.
- *NotificacionContext*: se utiliza mostrar las diferentes notificaciones cuando se realiza alguna acción.
- *PermisosContext*: es otro de los contextos más importantes, ya que evalúa si un usuario tiene o no los permisos necesarios para realizar las acciones dentro de la aplicación, como por ejemplo modificar pacientes o crear exámenes, entre otros.

2.3.2.1.4. Hooks

Los *Hooks* son funciones que te permiten capturar el estado y otras características como por ejemplo el ciclo de vida de *React* desde componentes de función sin escribir una clase. (Reactjs, s.f.)

Para los contextos de la aplicación, se desarrollaron *hooks* personalizados que permiten, entre otras cosas, acceder a los diferentes elementos que existen dentro de los contextos, como funciones u objetos, entre los *hooks* principales se encuentran:

- *useAuth*: permite el acceso al *JWTContext*, y a su vez, a las propiedades y funciones definidas en ese contexto. Con este *hook* se puede acceder al usuario activo y a las funciones de sesión.
- *useConstantes*: permite acceder al *ConstantesContext* para obtener las constantes de aplicación como los sexos, por ejemplo.
- *useMenu*: se utiliza para llamar al menú principal, evalúa los permisos de usuario para mostrar las rutas o no.
- *usePermisos*: es de suma importancia ya que permite acceder a la función obtenerPermisos() del contexto *PermisosContext*, la función descrita, evalúa si el usuario posee o no los permisos necesarios para realizar acciones sobre un módulo en específico, recibe como parámetro el nombre de dicho módulo y retorna las diferentes acciones que el usuario puede realizar.

2.3.2.1.5. Servicios

Se le llama servicios a un conjunto de archivos cuya función es realizar diferentes peticiones al *API*, se conoce como petición a una instrucción que se espera que el servidor ejecute y retorne algún resultado de esta como por ejemplo guardar cierta información o consultarla.

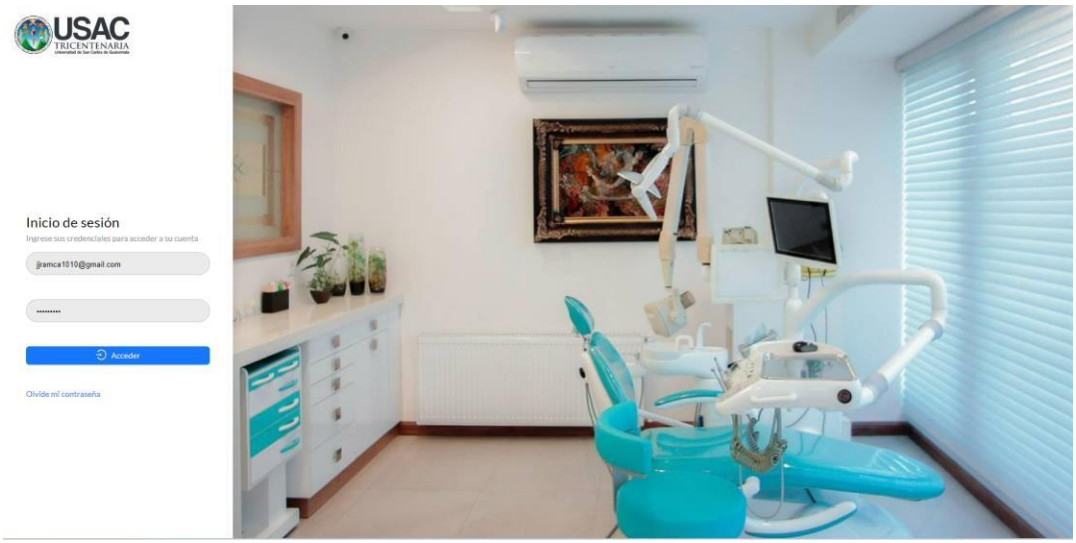
En lugar de realizar cada petición, como añadir pacientes o exámenes, consultar pacientes, modificar solicitudes, entre otros. desde diferentes partes de la aplicación, estos archivos adquieren la responsabilidad de centralizar el origen de las peticiones para cada módulo de la aplicación, con esto, se logra tener un mejor control de dónde se realiza cada solicitud y si es necesario cambiar el código, modificar la funcionalidad o agregar funcionalidad adicional, únicamente se edita el archivo correspondiente en lugar de buscar entre todo el código existente.

2.3.2.2. Páginas

Esta carpeta contiene todas las vistas principales de la aplicación. Las vistas no son componentes, una vista está construida con un conjunto de múltiples componentes que interactúan entre ellos, cada uno con una funcionalidad bien definida pero esencial en cada una de las vistas.

Figura 9.

Inicio de sesión de aplicación

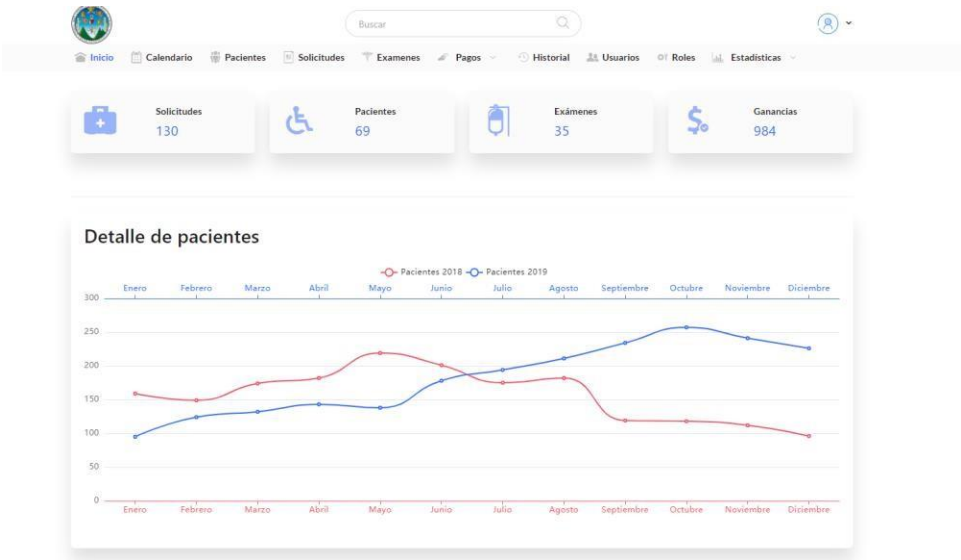


Nota. Vista de inicio de sesión. Elaboración propia, realizada con *React*.

Las vistas simularan lo que vendría siendo una página web independiente sin embargo este no es el caso, ya que todas las vistas forman parte de una misma página web que será la aplicación completa.

Figura 10.

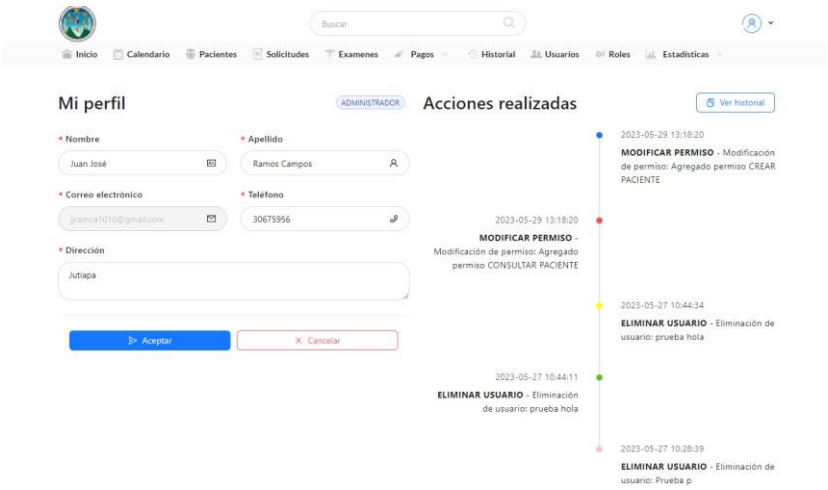
Dashboard de aplicación



Nota. Vista inicial de aplicación. Elaboración propia, realizada con *React*.

Figura 11.

Cuenta de usuario

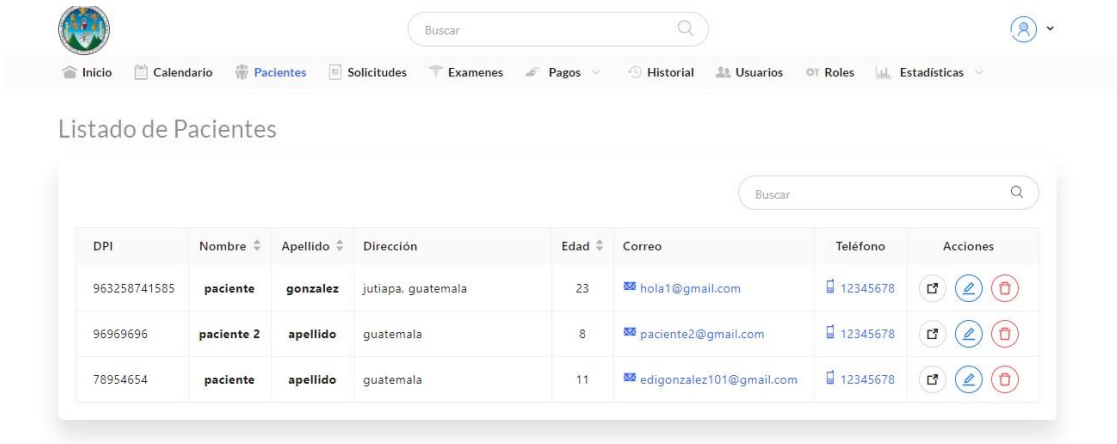


Nota. Vista de datos de usuario. Elaboración propia, realizada con *React*.

Todas las acciones de usuario estarán registradas en una bitácora especial que no es posible modificar de ninguna manera.

Figura 12.

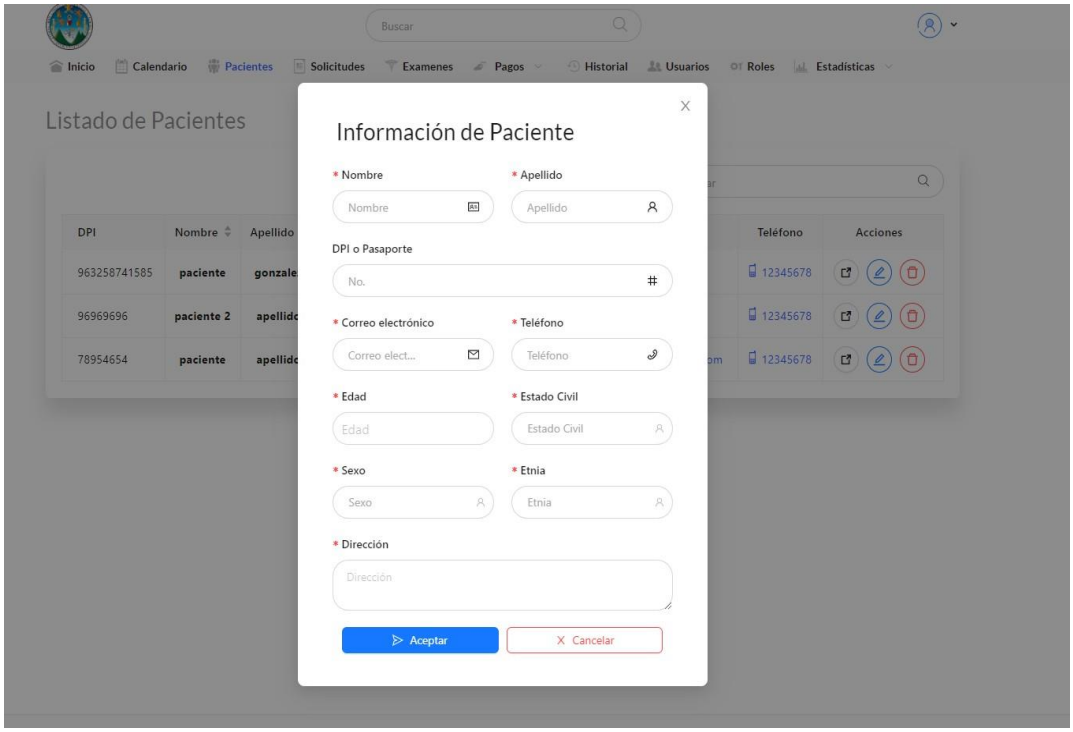
Listado general de pacientes



Nota. Vista de pacientes. Elaboración propia, realizada con *React*.

Figura 13.

Formulario de nuevo paciente

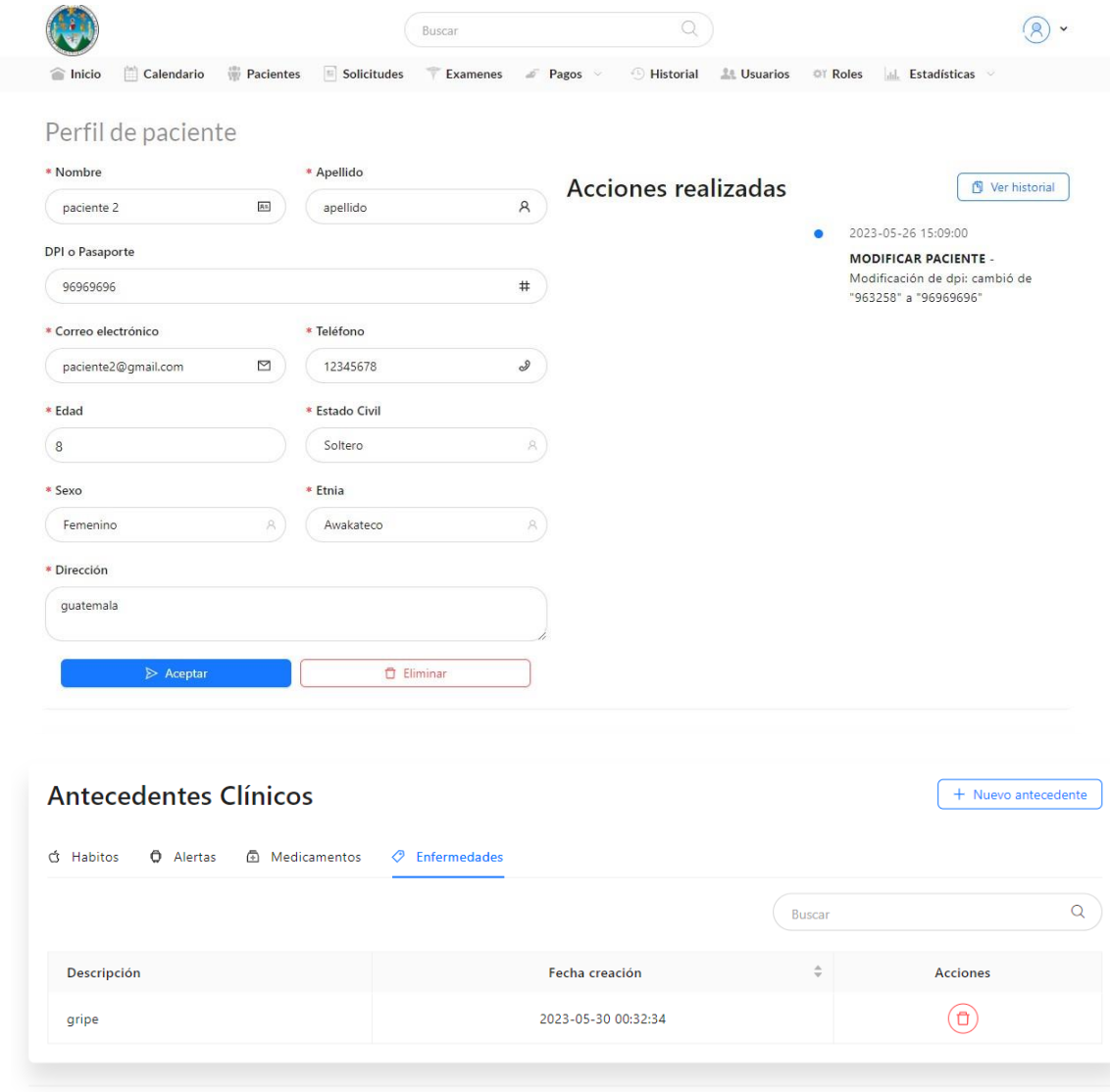


Nota. Vista de ingreso de pacientes. Elaboración propia, realizada con *React.*

Se garantiza que la información proporcionada sea correcta antes de almacenarla en la base de datos, esto es así para todos los casos, siempre que se ingrese nueva información de cualquier módulo, se cuenta con validaciones rigurosas para evitar enviar información incorrecta a la base de datos.

Figura 14.

Detalle de paciente



Nota. Vista de expediente clínico de pacientes. Elaboración propia, realizada con *React.*

Los antecedentes del paciente se agregan una vez este se haya creado de manera exitosa.

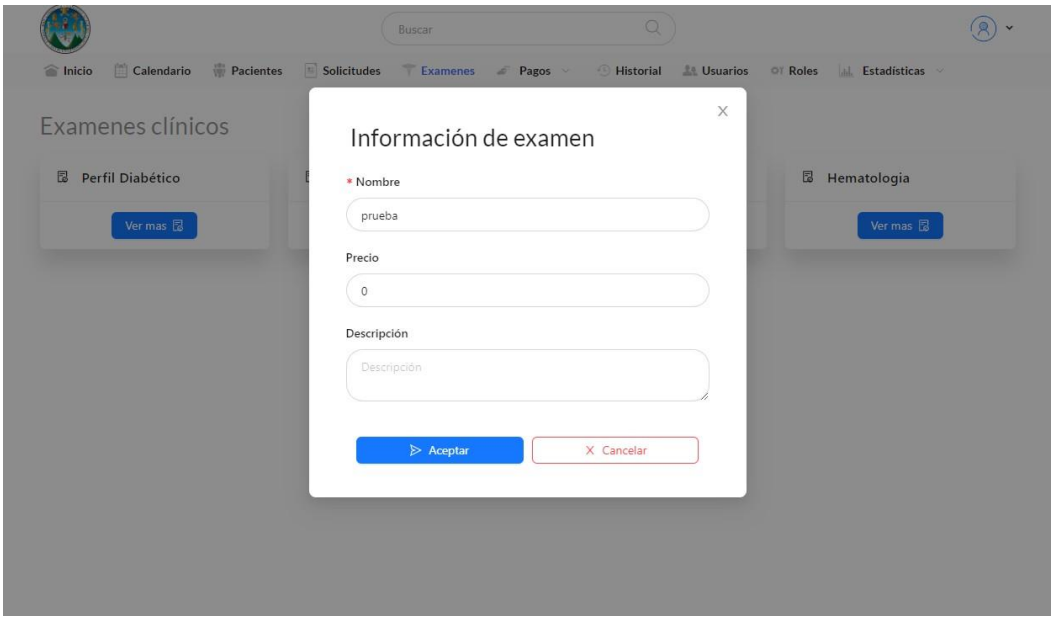
Figura 15.

Listado de exámenes

Nota. Vista de exámenes de laboratorio. Elaboración propia, realizada con React.

Figura 16.

Formulario de nuevo examen

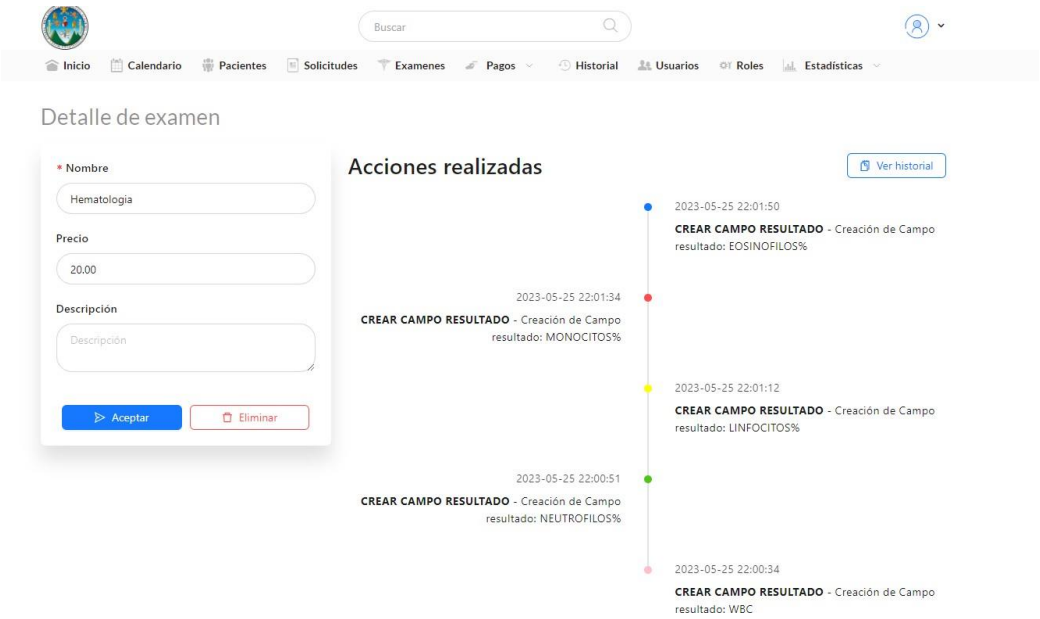


Nota. Vista de ingreso de examen. Elaboración propia, realizada con *React*.

El formulario valida que se ingrese un nombre, ya que es obligatorio un nombre como mínimo para crear el examen.

Figura 17.

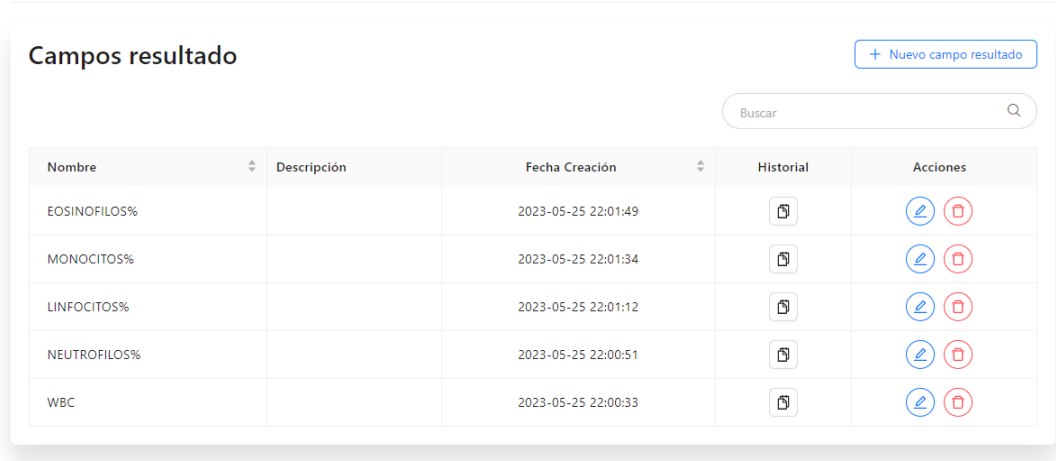
Detalle de examen



Nota. Vista de información de examen. Elaboración propia, realizada con *React*.

Figura 18.

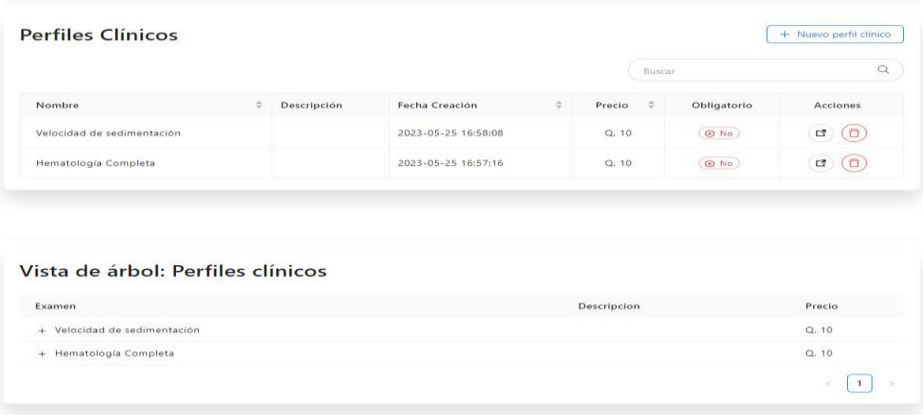
Campos resultado por examen



Nota. Vista de campo resultado de examen. Elaboración propia, realizada con React.

Figura 19.

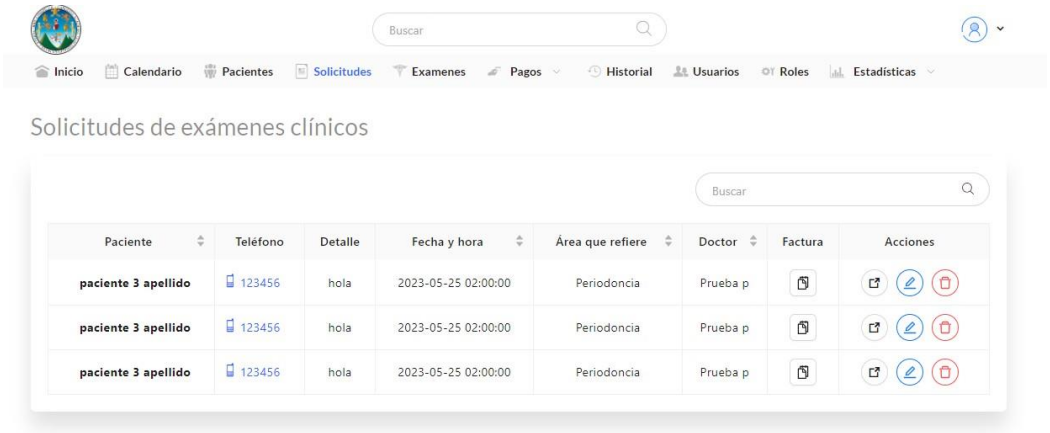
Perfiles clínicos por examen



Nota. Vistas adicionales de información de examen. Elaboración propia, realizada con React.

Figura 20.

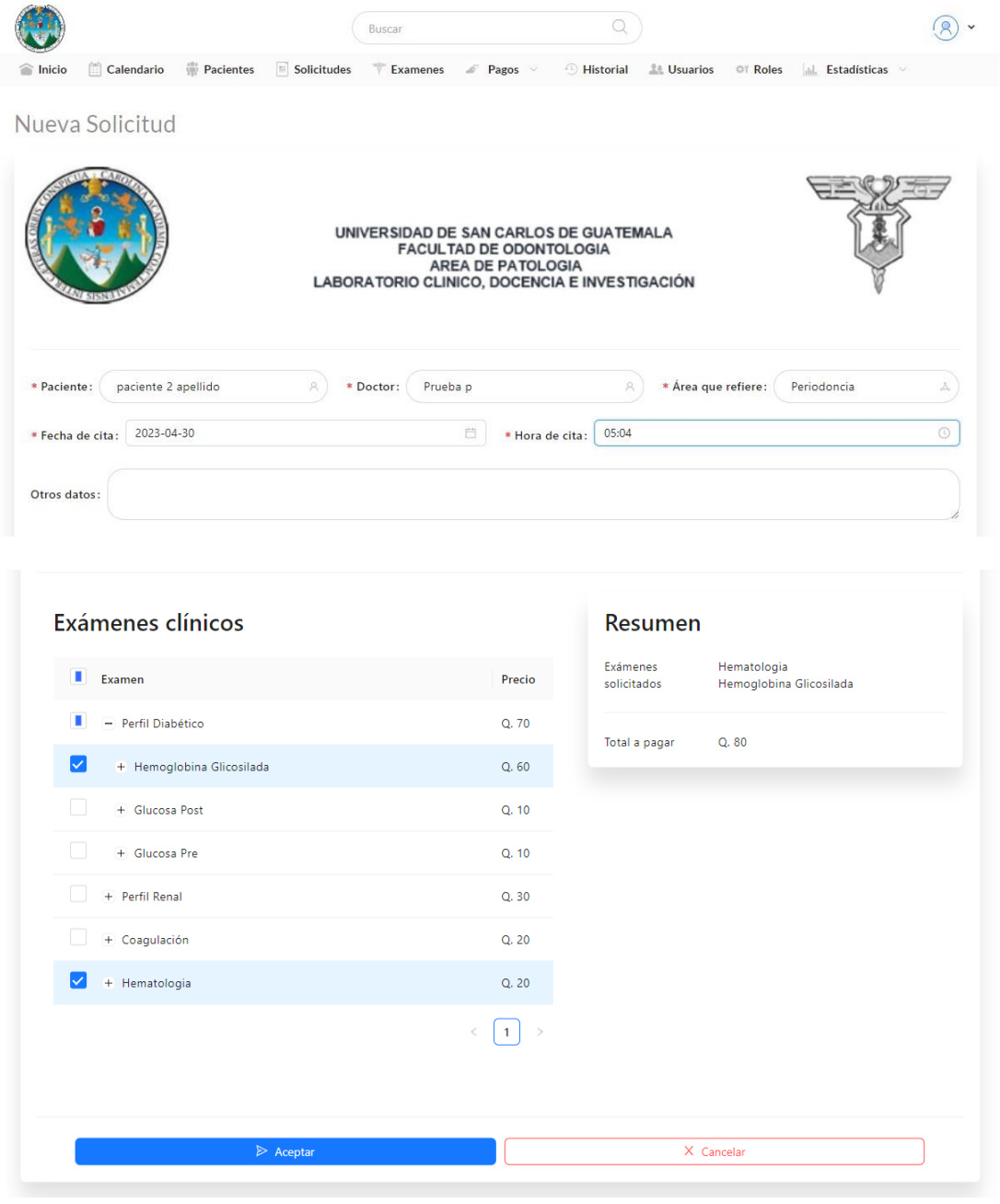
Lista de solicitudes de exámenes



Nota. Vista de solicitudes de exámenes clínicos. Elaboración propia, realizada con *React*.

Figura 21.

Nueva solicitud de exámenes clínicos

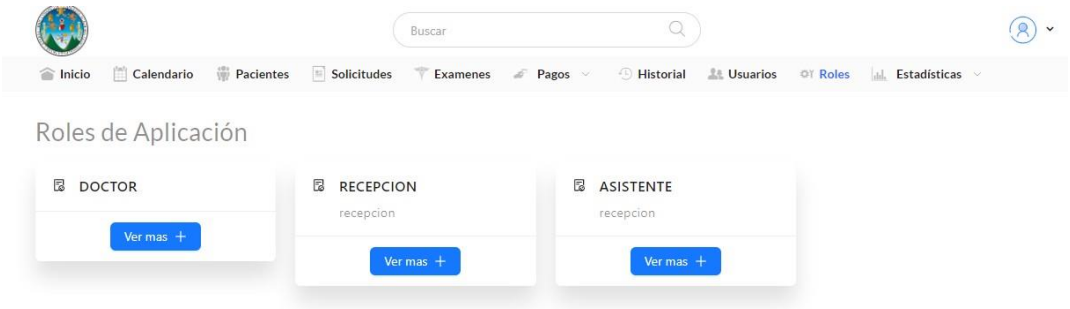


Nota. Vista de ingreso de solicitud. Elaboración propia, realizada con *React*.

Se deben seleccionar uno o más exámenes para poder crear la solicitud, de lo contrario el sistema no lo permitirá.

Figura 22.

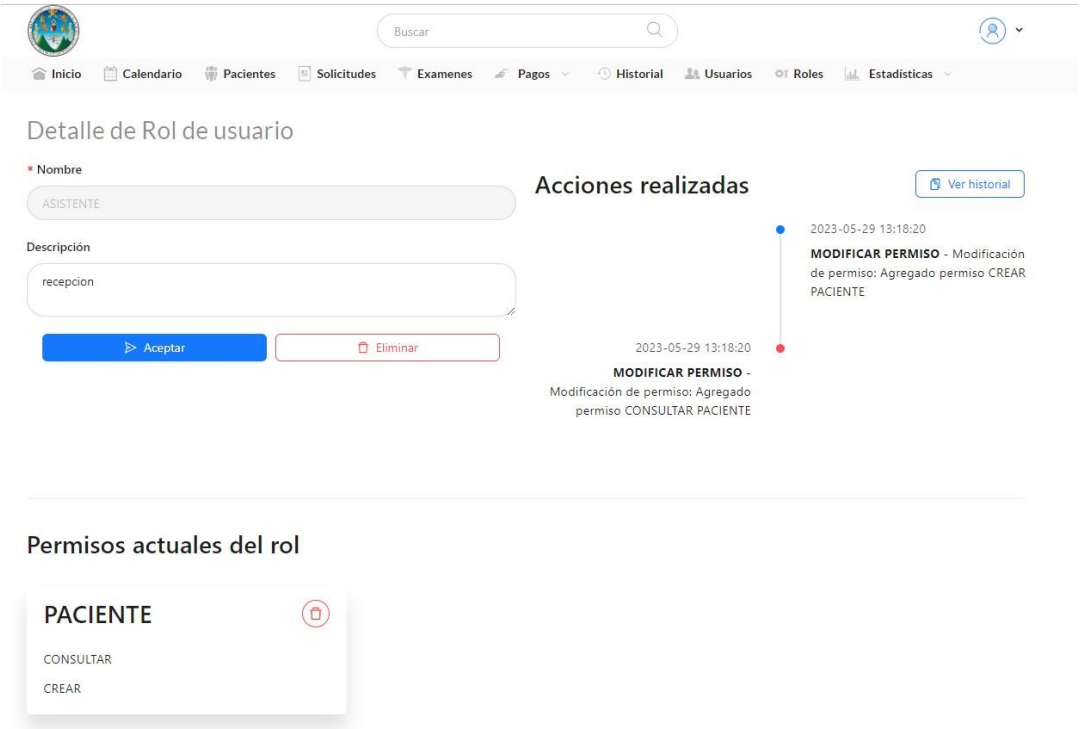
Roles de aplicación



Nota. Vista de roles de usuario. Elaboración propia, realizada con *React*.

Figura 23.

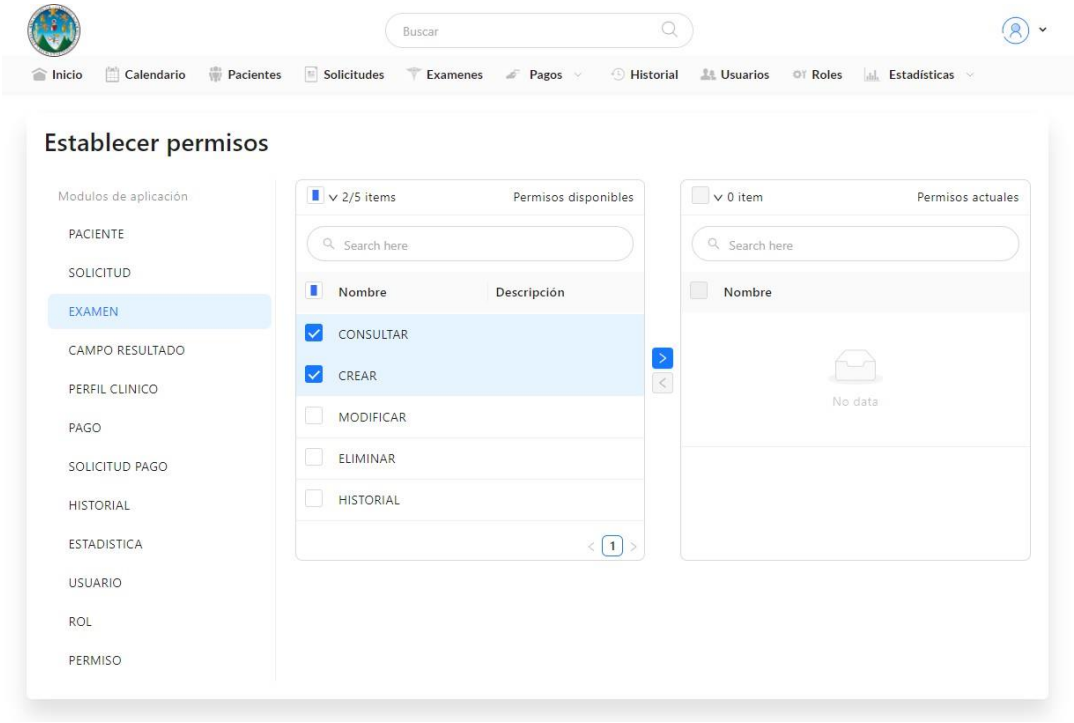
Detalle de rol de usuario



Nota. Vista de información de rol. Elaboración propia, realizada con *React*.

Figura 24.

Establecimiento de permisos de rol



Nota. Vista de permisos de rol. Elaboración propia, realizada con *React*.

2.3.3. Fase 3: Arquitectura del *API REST*

Se le conoce como servidor de aplicación a un *software* que proporciona un entorno para el almacenamiento, despliegue, ejecución y mantenimiento de aplicaciones *web*. Éste actúa como una plataforma intermedia entre los clientes y los servidores de bases de datos, permitiendo que las aplicaciones *web* se ejecuten de manera eficiente y escalable, además permite gestionar las solicitudes de los clientes y se encarga de procesarlas, interactuar la base de datos, generar respuestas y enviarlas de vuelta al cliente.

Por su parte un *API* significa “Interfaz de programación de aplicaciones”. “La palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas” (Amazon Web Services, Inc, s.f., p. 2).

Y la misma se define como “un mecanismo que permite a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos” (Amazon Web Services, Inc, s.f., p. 1).

“La arquitectura de las API suele explicarse en términos de cliente y servidor. La aplicación que envía la solicitud se llama cliente, y la que envía la respuesta se llama servidor” (Amazon Web Services, Inc, s.f., p. 3).

REST significa transferencia de estado representacional. *REST* define un conjunto de funciones como *GET, POST, PUT y DELETE*. que los clientes pueden utilizar para acceder a los datos del servidor. Los clientes y los servidores intercambian datos mediante *HTTP*.

La principal característica de la *API REST* es que no tiene estado. La ausencia de estado significa que los servidores no guardan los datos del cliente entre las solicitudes. Las solicitudes de los clientes al servidor son similares a las *URL* que se escriben en el navegador para visitar un sitio web. La respuesta del servidor son datos simples, sin la típica representación gráfica de una página web (Amazon Web Services, Inc, s.f., p. 8).

Para construir el *API* de la aplicación se utilizó el *framework* de *Django*, este es un *framework* de desarrollo *web* de alto nivel escrito en *Python* que proporciona un conjunto completo de herramientas y funcionalidades para

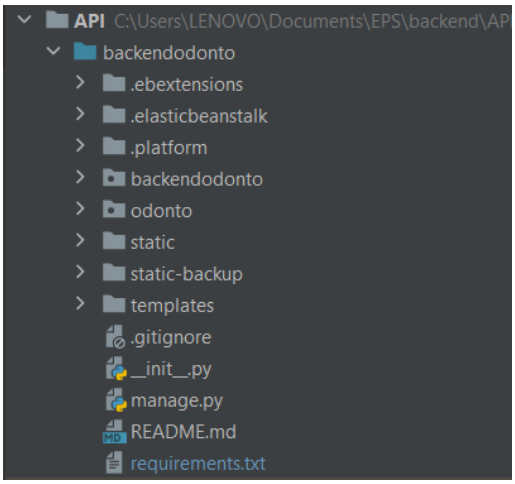
construir aplicaciones *web* escalables (Django, s.f.). Entre las principales características que tienen las aplicaciones diseñadas en *Django* se encuentran:

Una de las características principales de *Django* es su arquitectura sólida, la cual permite que las aplicaciones cuenten con características bien definidas como lo pueden ser:

- Escalabilidad: se refiere al crecimiento de la aplicación “permitiendo gestionar múltiples solicitudes de clientes al mismo tiempo” (Amazon Web Services, Inc, s.f., p. 38). de manera más eficiente sin comprometer el rendimiento, lo cual es justo lo que se espera conseguir dado que la aplicación del laboratorio fue pensada para ser integrada con máquinas en un futuro.
- Mantenibilidad: se facilita grandemente el mantenimiento de la aplicación a lo largo del tiempo, permitiendo realizar cambios y mejoras de manera más eficiente.
- Reutilización de código: *Django* promueve la reutilización de código, lo que significa que los componentes y módulos desarrollados previamente podrán ser utilizados en diferentes partes de la aplicación ahorrando tiempo al no ser necesario desarrollar código igual.
- Modularidad: significa que la aplicación se divide en módulos más pequeños e independientes, cada módulo está encargado de cierta funcionalidad que es ajena a la de los demás, pero de igual de indispensable que el resto.

Figura 25.

Arquitectura del API



Nota. Estructura general del *API*. Elaboración propia, realizada con *PyCharm*.

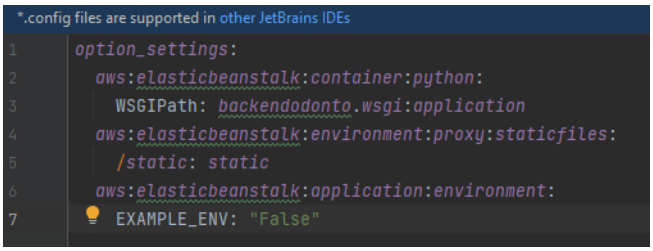
2.3.3.1. *.Ebextensions*

Esta carpeta contiene los archivos necesarios para configurar la aplicación dentro de *Elastic Beanstalk*, dicho servicio se explicará más adelante, pero a grandes rasgos representa el servidor de aplicación. Entre los archivos se encuentran:

- *django.conf*: este archivo contiene las configuraciones básicas para montar el *API*, es el archivo de configuración principal.

Figura 26.

Contenido del archivo django.conf

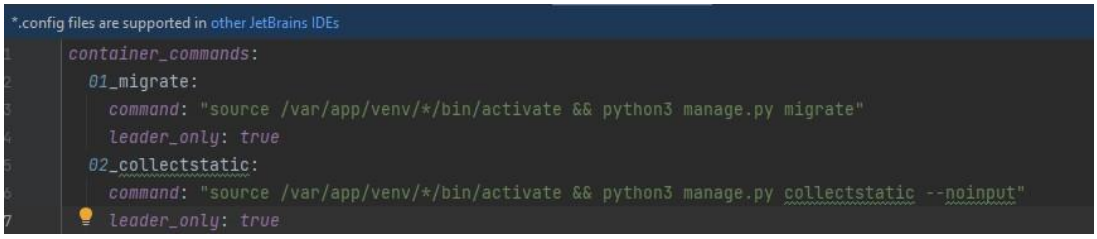


Nota. Estructura del archivo *django.conf*. Elaboración propia, realizada con *PyCharm*.

- *jobs.config*: este archivo contiene los trabajos o "*jobs*", un *job* es una instrucción que ejecuta algún comando o acción, en este caso, se indica que ejecute las migraciones realizadas, una migración es un cambio sobre la base de datos, por ejemplo, crear una tabla o modificar un campo, entre otros.

Figura 27.

Contenido del archivo jobs.config



Nota. Estructura del archivo *jobs.conf*. Elaboración propia, realizada con *PyCharm*.

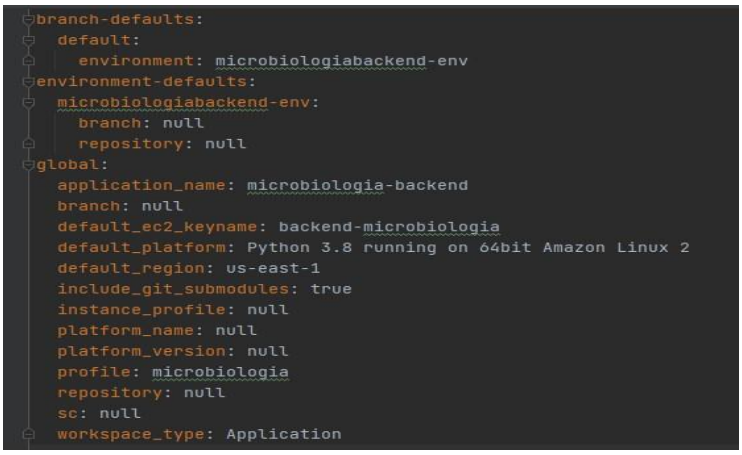
2.3.3.2. .*Elasticbeanstalk*

Esta carpeta contiene el archivo *config.yml* el cual es necesario para desplegar el API en *Elastic Beanstalk*, este es un archivo muy importante porque en él se especifica el servidor donde estará corriendo la aplicación, así como la plataforma o el lenguaje y el área.

Este archivo se utiliza principalmente para realizar el despliegue de la aplicación, dicho despliegue se explicará con mayor detalle más adelante, pero las principales partes de este archivo son el *enviroment*, para especificar el ambiente, *application_name* para definir el nombre de la aplicación y *default_ec2_keyname* para definir la máquina virtual donde estará alojado el servidor.

Figura 28.

Contenido del archivo de despliegue



Nota. Estructura básica del archivo de despliegue. Elaboración propia, realizada con *PyCharm*.

2.3.3.3. .*Backendodonto*

Es el folder raíz del proyecto de *Django*, en él se almacenan las configuraciones básicas del proyecto para hacerlo funcionar, entre los archivos se encuentran:

- *asgi.py*: el archivo es necesario para ejecutar una aplicación de *Django* utilizando un servidor web compatible con *ASGI* que son las siglas de *Asynchronous Server Gateway Interface* el cual es una especificación para comunicación entre servidores y aplicaciones *web*, dicha comunicación permite un procesamiento asíncrono y escalable de las solicitudes.
- *settings.py*: es un archivo de configuración fundamental de *Django* ya que contiene un conjunto de variables y ajustes que definen el comportamiento y configuración de la aplicación, algunas de las configuraciones pueden ser la base de datos, configuraciones de rutas, configuraciones de otras aplicaciones, entre otros.
- *urls.py*: en *Django* este archivo se usa para definir las rutas y las vistas de una aplicación *web*. Actúa como un compilado de rutas y permite configurar cómo se manejan las solicitudes de los usuarios y qué respuestas se generan para cada ruta en específico.

2.3.3.4. Odonto

Este folder contiene las diferentes clases, métodos y funciones que harán funcional la aplicación *web*, como por ejemplo guardar nuevos usuarios, pacientes, solicitudes, exámenes, entre otros.

Cada módulo contiene su folder especifico donde se encontrarán las clases que permiten realizar cada funcionalidad como se mencionó, por ejemplo, el folder de pacientes contendrá las funciones necesarias para la manipulación de datos de pacientes.

2.3.3.5. Static

La carpeta *static* se utiliza para almacenar archivos estáticos, como pueden ser hojas de estilo *CSS*, *scripts* de *JavaScript*, imágenes y otros recursos estáticos utilizados en una aplicación *web* que se utilizan durante el desarrollo de esta. Dentro de esta carpeta se pueden crear más carpetas para organizar de manera estructurada todos los archivos que se vayan a utilizar.

2.3.3.6. Templates

La carpeta *templates* se utiliza para almacenar las plantillas *HTML* que se utilizarán para mostrar contenido a los usuarios. Esta carpeta proporciona una ubicación organizada para las plantillas que mostrarán diferente contenido y funcionalidad, esto facilita la separación de la lógica de programación al momento de desarrollar.

2.3.3.7. *Manage.py*

Es un *script* de línea de comandos que se utiliza para administrar y ejecutar varias tareas relacionadas con un proyecto *Django*. Este archivo proporciona una forma conveniente de interactuar con el proyecto y ejecutar comandos de gestión, como iniciar el servidor de desarrollo, realizar migraciones y ejecutar pruebas.

2.3.4. Fase 4: Despliegue de aplicación

Se conoce como despliegue de aplicación al proceso de implementar y poner en funcionamiento una aplicación en un entorno operativo, generalmente este entorno suele ser un servidor, para que los usuarios finales puedan acceder a la aplicación y utilizar la misma. El proceso de despliegue implica una serie de pasos que van más allá de solamente escribir el código de la aplicación, incluye entre otras cosas la preparación del entorno, instalación de dependencias, gestión de recursos, optimización de rendimiento y principalmente la garantía de que la aplicación esté disponible en todo momento y funcione de manera correcta.

2.3.4.1. *Front end* en producción

Para el despliegue del *front end* se hizo uso de dos servicios, *Amazon S3* y *GitHub Actions*.

Amazon Simple Storage Service (S3) es un servicio de almacenamiento en la nube ofrecido por *Amazon Web Services (AWS).* "*Amazon S3* permite a las personas y las organizaciones almacenar y recuperar datos de manera escalable" (Amazon Web Services, Inc, s.f.).

Este servicio se utilizó principalmente para contener los archivos del código de la página *web*.

Por su parte, *GitHub Actions* es una plataforma de integración y despliegue que permite automatizar compilación, pruebas y despliegue. Es posible crear flujos de trabajo y crear y probar cada solicitud de cambios en un

repositorio o desplegar solicitudes de cambios directamente a producción (GitHub, s.f.).

> En Acciones de *GitHub*, un flujo de trabajo es un proceso automatizado que se configura en un repositorio de *GitHub*. Se puede compilar, probar, empaquetar, liberar o implementar cualquier proyecto de *GitHub* con un flujo de trabajo.
>
> Cada flujo de trabajo se compone de acciones individuales que se ejecutan después de que se produce un evento específico (por ejemplo, una solicitud de incorporación de cambios). Las acciones individuales son scripts empaquetados que automatizan las tareas de desarrollo de *software*. (Microsoft, s.f. p. 2)

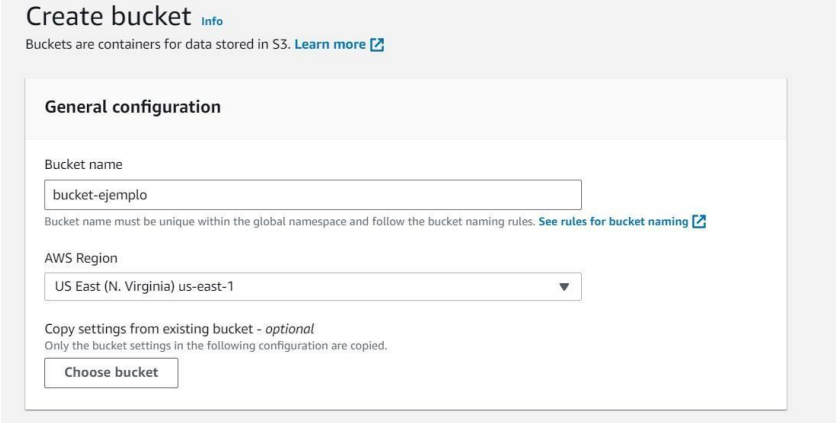
Ambas herramientas permiten realizar el despliegue de manera sencilla únicamente al subir los cambios al repositorio de versiones, una vez realizada esta acción empieza el proceso de despliegue donde las *GitHub Actions* realiza el compilado y publicación de los nuevos cambios.

El proceso para realizar el despliegue se explica a continuación:

- Se creó un *bucket* en *S3* como se muestra en la imagen:

Figura 29.

Creación de bucket en S3

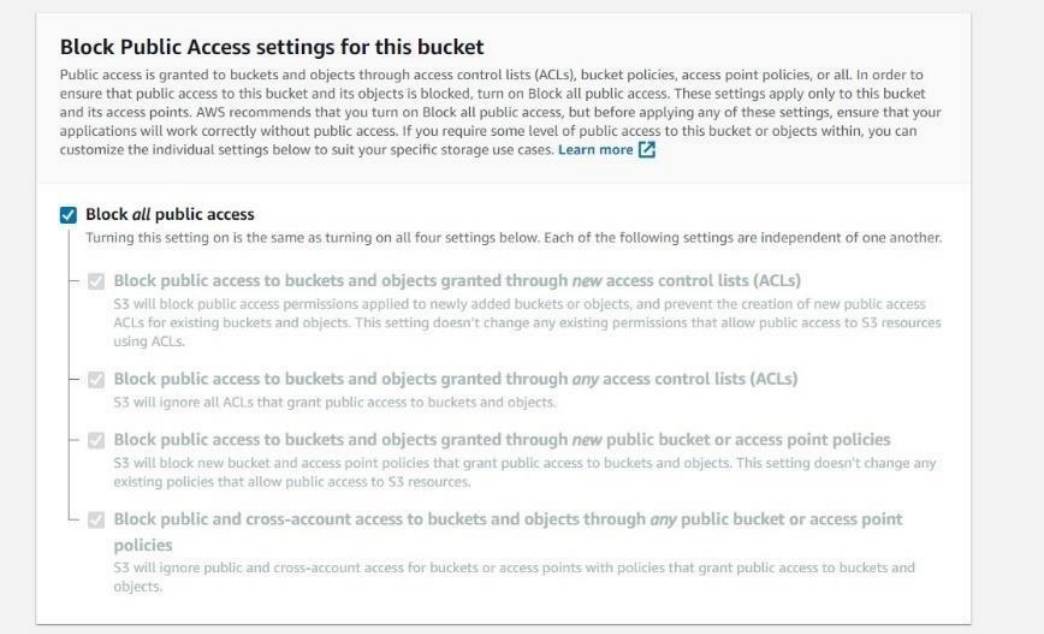


Nota. Ejemplo de creación. Consola de administración de *AWS*, 2023.

- Se desmarcó la casilla *Block all public access* que bloquea el acceso público, esto con la finalidad de poder acceder a la aplicación una vez terminado el proceso, con esto se consigue que el acceso a la aplicación únicamente lo puedan hacer los servicios o personas que cuenten con la autorización necesaria, dicha autorización es en base a usuarios permitidos, la cual se explicará más adelante.

Figura 30.

Bloqueo de acceso público del bucket



Nota. Permisos de *bucket*. Consola de administración de *AWS*, 2023.

- Posteriormente se agregaron políticas de acceso público al *bucket*, esta acción no se recomienda cuando se trabaja con *buckets*, pero para realizar el despliegue de la aplicación que se utilizará internamente no hay ningún problema.

- Para ello se dirigió al detalle del *bucket* haciendo clic en su nombre y se accedió a la sección de Permisos, luego a la sección de *Bucket policy* y se agregó el fragmento de código mostrado a continuación.

Figura 31.

Política de acceso del bucket

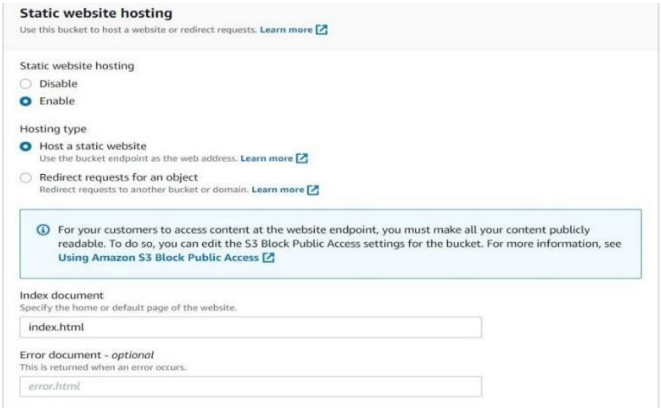
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*"
            ]
        }
    ]
}
```

Nota. Permisos de *bucket* para ser público. Consola de administración de *AWS*, 2023.

Con el código anterior, el *bucket* pasó a ser de dominio público. Posteriormente se habilitó el alojamiento de sitios *web* estáticos, con el fin del *bucket* reconozca el compilado como una aplicación y lo ejecute como una página web normal.

Figura 32.

Habilitar el alojamiento de sitios web estáticos

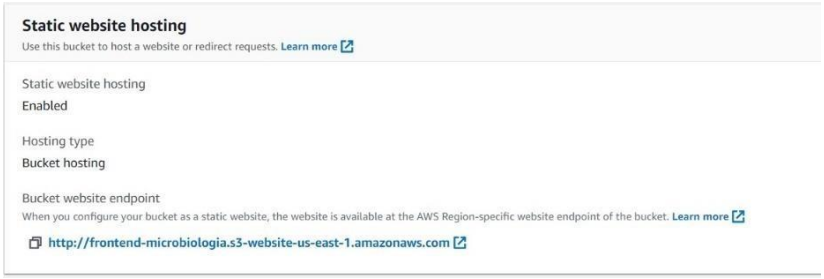


Nota. Establecer permisos de sitio *web*. Consola de administración de *AWS*, 2023.

El resultado quedó de la siguiente manera:

Figura 33.

Bucket creado

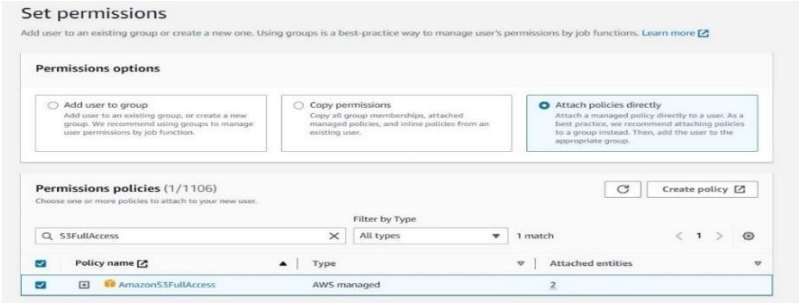


Nota. Resultado de creación de *bucket*. Consola de administración de *AWS*, 2023.

- El siguiente paso fue obtener autorización de *AWS* para acceder al *bucket* desde aplicaciones externas, esto porque, *GitHub Actions* intentará almacenar el compilado de la página web y sin los permisos necesarios, no podrá realizar dicha acción, para ello se creó un nuevo usuario y se le asignaron los permisos de acceso completo de administración de *buckets*.

Figura 34.

Permisos de acceso a S3 a usuarios de AWS

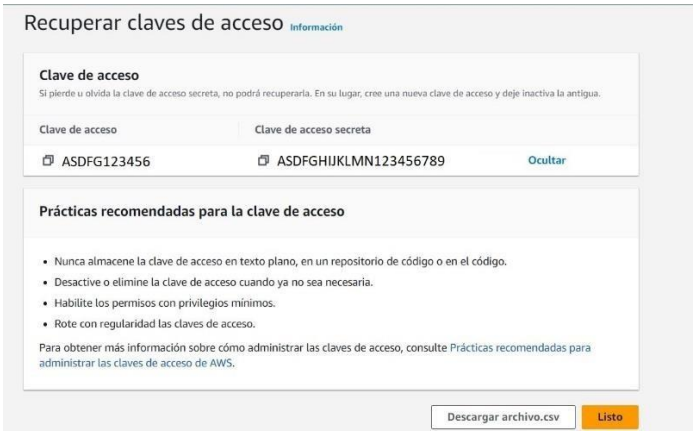


Nota. Establecimiento de permisos de usuario. Consola de administración de *AWS*, 2023.

- Una vez se creó el usuario, se procedió a crear las Credenciales de seguridad descritas con anterioridad, para ello se accedió a su detalle haciendo clic en su nombre y en la sección claves de acceso se obtuvieron dichas credenciales.

Figura 35.

Credenciales de acceso de usuario AWS

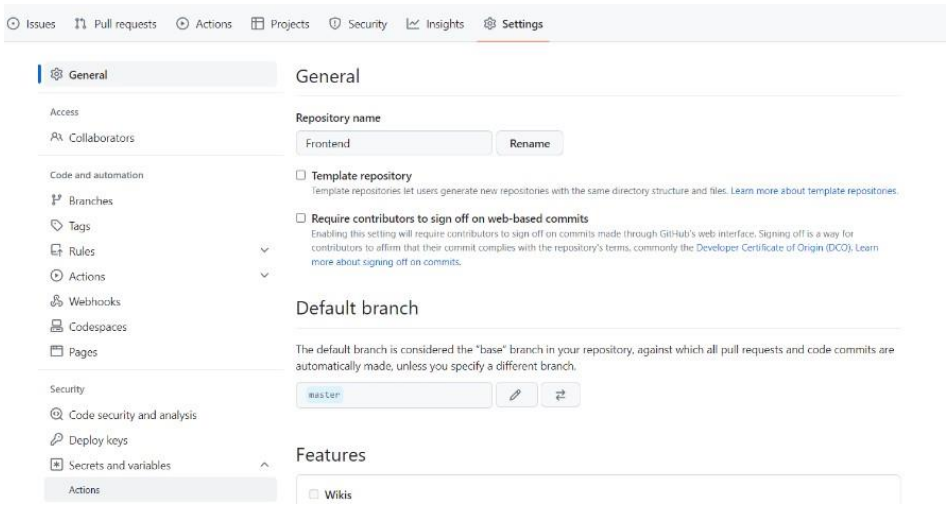


Nota. Credenciales de usuario. Consola de administración de *AWS*, 2023.

- Posteriormente dentro de las opciones del repositorio del *front end* del proyecto, se accedió a la sección de *Secrets and variables* y luego a *Actions* donde se agregaron las variables secretas para que el servicio las obtenga y ejecutara el despliegue, las variables secretas también se conocen como variables de entorno, estas se configuran de esta manera para no tener que colocar información sensible dentro del código de la aplicación.

Figura 36.

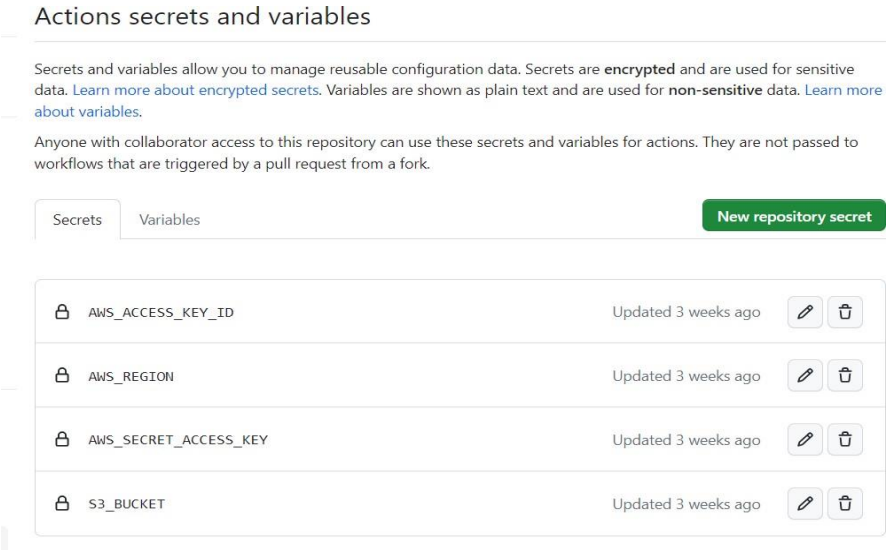
Opciones de repositorio



Nota. Configuración general de repositorio. Consola de administración de *GitHub*, 2023.

Figura 37.

Ingreso de Secrets and Variables



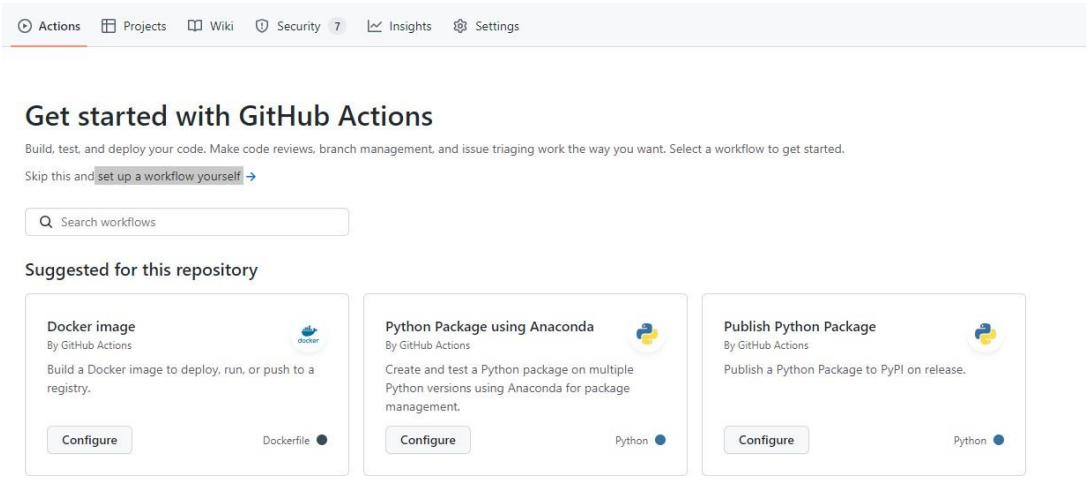
Nota. Variables secretas. Consola de administración de *GitHub*, 2023.

- Como ya se contaba tanto con el repositorio del proyecto configurado como con el *bucket* de *S3* se procedió a comenzar con las acciones de *GitHub*, para ello dentro del repositorio se accedió a la pestaña de *Actions* en la cual se hizo clic en el botón *set up a workflow yourself* para ser dirigido a una pestaña con un editor de código en el cual se colocó el código necesario para ejecutar las acciones correspondientes de despliegue.

  Las acciones incluían entre otras cosas, compilar la aplicación, acceder a *AWS* para obtener los permisos necesarios para guardar los compilados, almacenar el compilado, montarlo en una máquina virtual, entre otros.

Figura 38.

Configurar acciones de GitHub



Nota. Vista de GitHub Actions. Consola de administración de GitHub, 2023.

- Dentro del editor se colocó, en un archivo con extensión *yml***,** el código mostrado a continuación que contiene las diferentes acciones que *GitHub* utilizará para desplegar el *front end*.

Figura 39.

Archivo de acciones de GitHub

```yaml
# This is a basic workflow to help you get started with Actions

name: s3-depl

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the master branch
  push:
    branches: [ master ]

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest
    env:
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
      AWS_DEFAULT_REGION: ${{ secrets.AWS_REGION }}
      CI: false
    strategy:
      matrix:
        node-version: [18.9.0]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    # Steps represent a sequence of tasks that will be executed as part of the job
    steps:
      # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
      - uses: actions/checkout@v3
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v2
        with:
          node-version: ${{ matrix.node-version }}
          cache: 'npm'
      - run: npm install --force
      - run: npm run build
      - name: Deploy app build to S3 bucket
        run: aws s3 sync ./build/ s3://${{ secrets.S3_BUCKET }}
```

Nota. Ejemplo de archivo de despliegue de *GitHub Actions*. Elaboración propia.

Las secciones del código se explicarán a continuación:

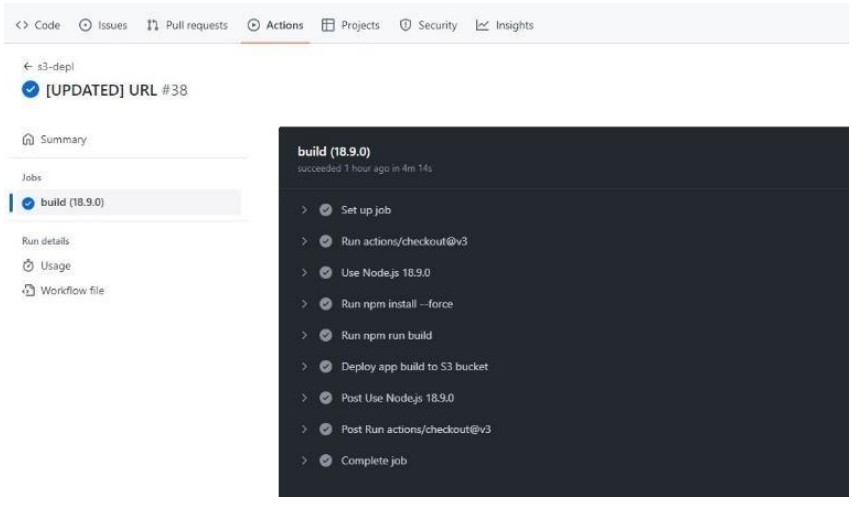
- *name*: define el nombre de una acción en medio de otras que pudieran existir.
- *on*: define un disparador de la acción, en este caso cuando se haga *push* en la rama *master*.
- *jobs*: son los trabajos o acciones que se ejecutarán.

- *steps*: un *job* contiene una secuencia de tareas denominadas pasos. Los pasos pueden ejecutar comandos, tareas de configuración o ejecutar acciones en el repositorio.
- *use Node.js*: descarga la versión de node js a utilizar e instala las dependencias y ejecuta el en el comando que crea el build de la página web.
- *Deploy app build to S3 bucket*: despliega la compilación recién creada en el bucket haciendo uso de las credenciales.

- Por último, cuando se guardaron los cambios de manera automática *GitHub* empezó a ejecutar las acciones descritas con anterioridad para poder desplegar la aplicación el *bucket* y que la página *web* pueda ser accedida desde la dirección correspondiente.

Figura 40.

Ejecución de GitHub Actions



Nota. Vista de *GitHub Actions* ejecutadas. Consola de administración de *GitHub*, 2023.

2.3.4.2. Base de datos en producción

La base de datos utilizada fue *PostgreSQL*, mientras que, para su almacenamiento y gestión se utilizó *Amazon RDS* el cual es un servicio de administración de bases de datos relacionales ofrecido por *Amazon Web Services*, que proporciona una manera sencilla de configurar, operar y escalar una base de datos en la nube, sin preocuparse por la infraestructura de esta.

Figura 41.

Diagrama de funcionamiento de Amazon RDS

Nota. Diagrama de funcionamiento de *Amazon RDS*, Obtenido de *Amazon Web Services* (https://aws.amazon.com/es/rds/) consultado el 26 de octubre de 2023. De dominio público.

Para la implementación de la base de datos se realizó lo siguiente:

- Se accedió al servicio de *Amazon RDS* desde la consola de administración de *AWS*.

Figura 42.

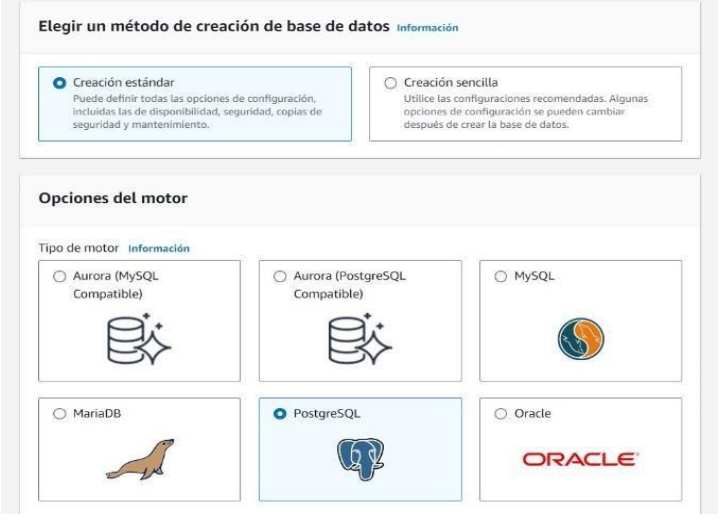
Servicio de Amazon RDS

Nota. Servicio de Amazon RDS. Consola de administración de AWS, 2023.

- Se seleccionó la opción de Creación estándar y se eligió *PostgreSQS* como motor de base de datos.

Figura 43.

Creación estándar de base de datos

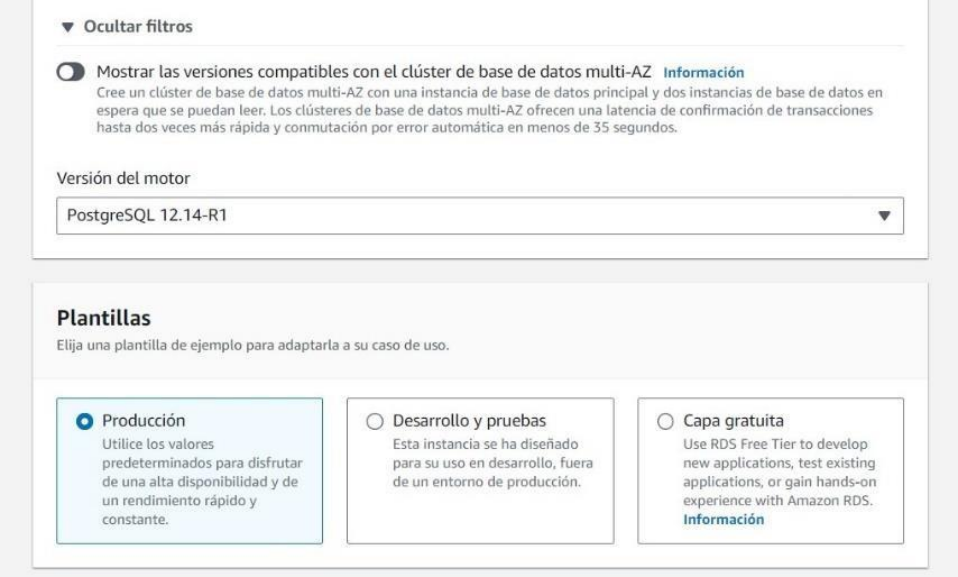


Nota. Ejemplo de creación de base de datos. Consola de administración de *AWS*, 2023.

- Se seleccionó la versión de *PostgreSQL* a utilizar, así como el tipo de plantilla, para este caso la plantilla Producción.

Figura 44.

Versión de base de datos

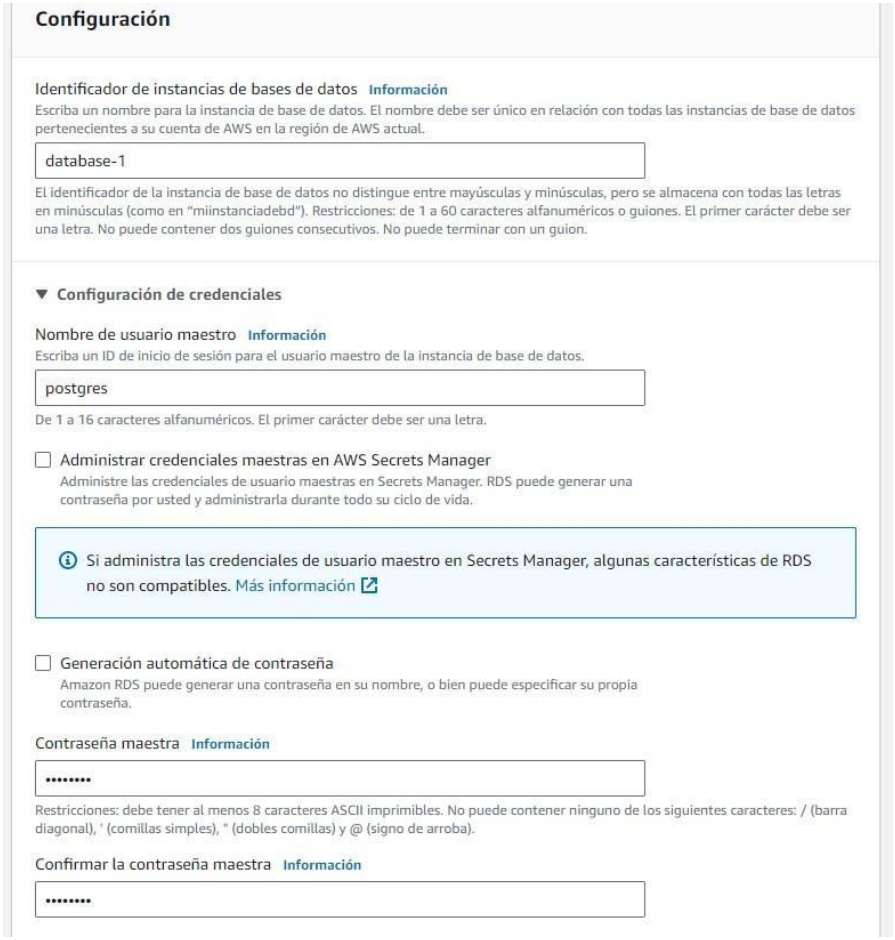


Nota. Selección de versión de base de datos. Consola de administración de AWS, 2023.

- Posteriormente, en la sección de Configuración se colocó un nombre a la base de datos y las credenciales del usuario maestro como lo son el nombre y la contraseña, dichas credenciales fueron necesarias para acceder a la base de datos desde el *API*.

Figura 45.

Credenciales de usuario maestro



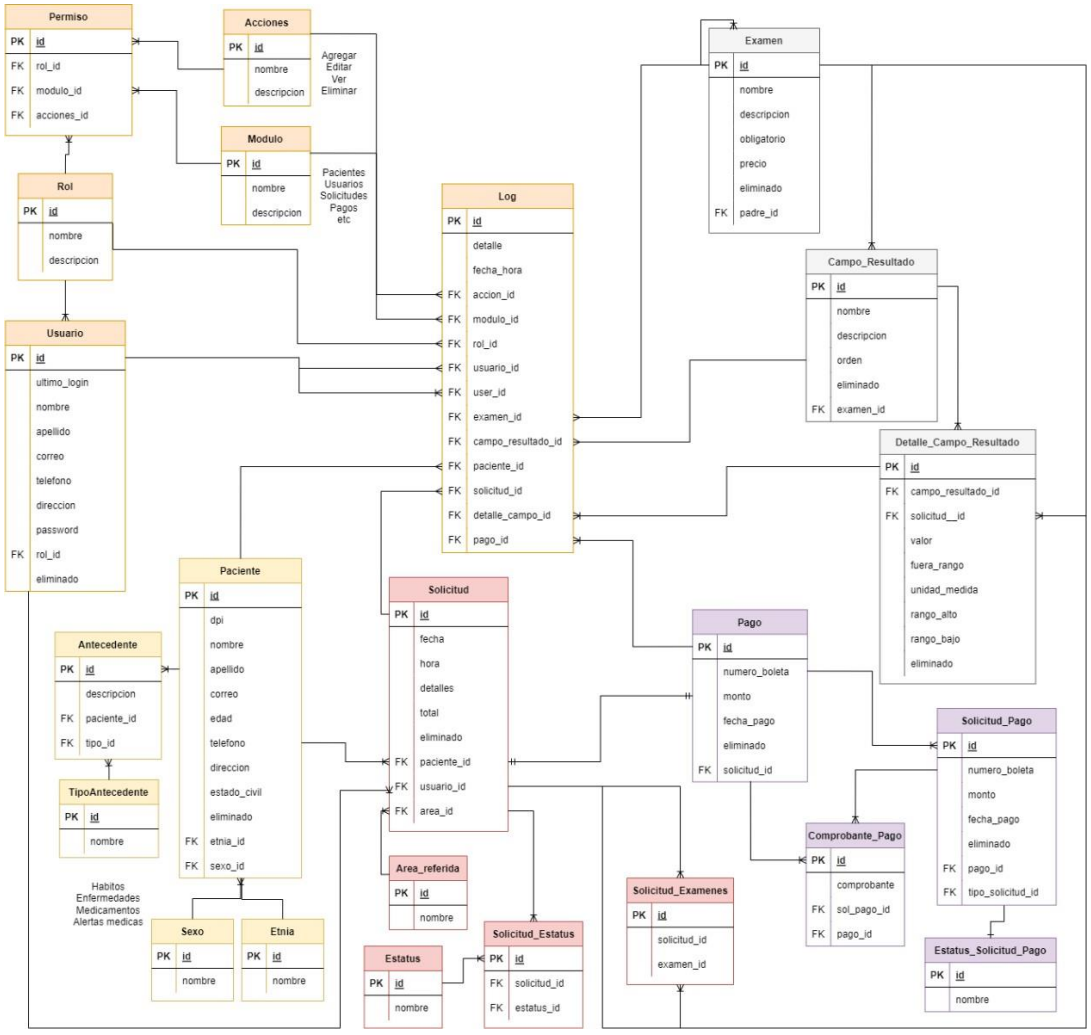
Nota. Establecimiento de usuario y contraseña. Consola de administración de AWS, 2023.

- Para el resto de las configuraciones, se quedaron los valores por defecto y una vez creada la base de datos se accedió a la sección de Conectividad y seguridad donde se pudieron obtener tanto el punto de enlace como el puerto y en conjunto con las credenciales del usuario maestro, la base de datos ya era funcional y accesible.

El modelo de datos final tuvo la estructura siguiente:

Figura 46.

Modelo de base de datos final



Nota. Diseño final de la base de datos. Elaboración propia, realizado con *Draw.io.*

Como se puede observar es muy parecido al modelo propuesto con la diferencia de que en lugar de que cada módulo tuviera su respectiva tabla de *log*, se centralizaron los *logs* en una única tabla que recibe el identificador del objeto

al que le pertenece ese *log*, así como el módulo al que pertenece el objeto y la acción realizada.

También se procedió a eliminar la tabla llamada Tipo Solicitud y se reemplazó con la llamada Estatus solicitud pago que maneja, como su nombre lo dice, los diferentes estatus que puede tener una solicitud de modificación, es decir si es rechazada o aceptada.

El último cambio significativo fue agregar la tabla Comprobante pago, asociada al pago la cual, como su nombre lo dice, almacena las fotos de la boleta de pago realizada por pacientes para cada solicitud.

2.3.4.3. *Back end* en producción

El servidor de producción o *API* está alojado en *AWS*, más concretamente se utilizó el servicio de *Elastic Beanstalk* para realizar las configuraciones necesarias para alojar el servidor. Este servicio se utiliza para implementar y escalar aplicaciones *web*.

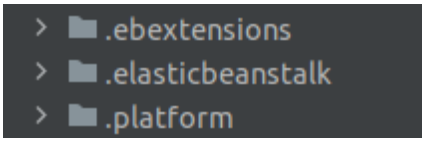
> Elastic Beanstalk permite implementar y administrar aplicaciones rápidamente en la nube de AWS sin tener que preocuparse por la infraestructura que las ejecuta. Elastic Beanstalk reduce la complejidad de la administración sin restringir la libertad de elección ni el control. Solo tiene que cargar la aplicación y Elastic Beanstalk gestionará de manera automática los detalles de aprovisionamiento de capacidad, balanceo de carga, escalado y monitorización de estado de la aplicación (Amazon Web Services, Inc. s.f. p. 2).

En términos simples, toma el código de la aplicación y lo despliega mientras provee la arquitectura de soporte y los recursos informáticos necesarios para que se ejecute el código.

Los archivos de configuración de *Elastic Beanstalk* se encuentran en las siguientes carpetas:

Figura 47.

Archivos de configuración de Elastic Beanstalk



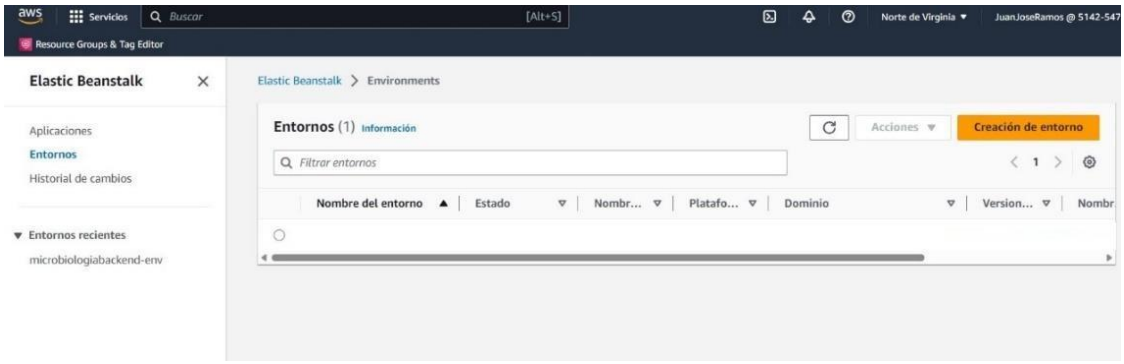
Nota. Carpetas principales de *Elastic Beanstalk*. Elaboración propia, realizado con *PyCharm*.

Los pasos para realizar el despliegue del servidor se explicarán a continuación:

- Primero se accedió al servicio de *Elastic Beanstalk* y se creó un nuevo entorno.

Figura 48.

Nuevo entorno Elastic Beanstalk

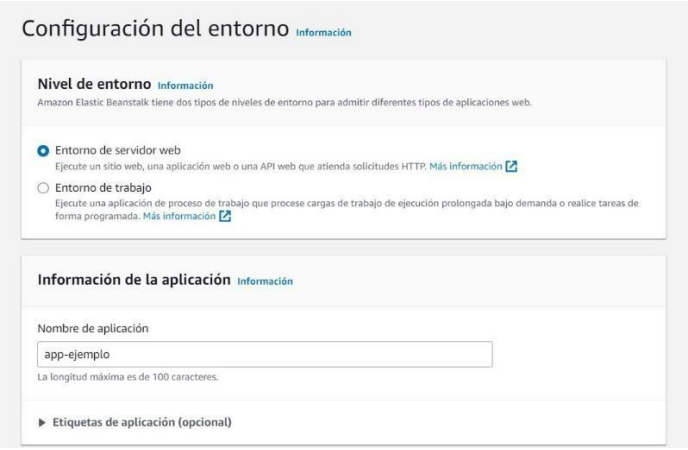


Nota. Vista de entornos. Consola de administración de AWS, 2023.

- Dentro de la página correspondiente se seleccionó Entorno de servidor *web* y se le dio un nombre a la aplicación para diferenciarlas del resto de aplicaciones que pudiese haber, este servicio es muy comúnmente utilizado para desplegar diferentes aplicaciones y no solo servidores por lo que colocar un nombre específico para el servidor es un paso muy importante debido a que puede haber otras aplicaciones ya creadas.

Figura 49.

Creación de aplicación de Elastic Beanstalk

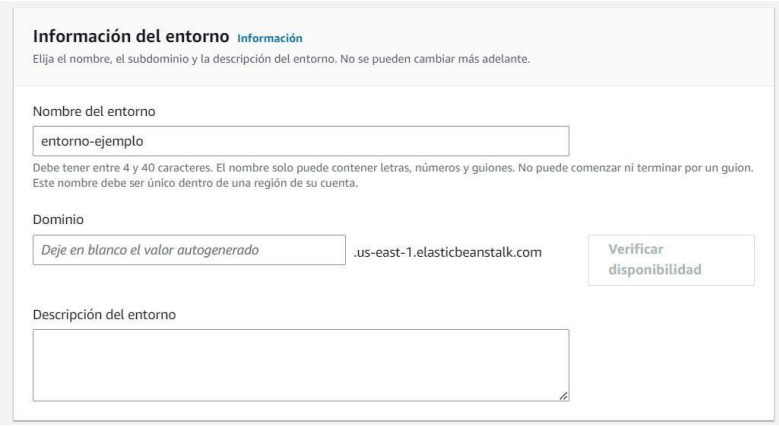


Nota. Ejemplo de creación de aplicación. Consola de administración de *AWS*, 2023.

- Posteriormente se le dio un nombre al entorno donde está almacenado el API, el dominio que tendrá el *API* se lo asignó automáticamente AWS.

Figura 50.

Creación del entorno de Elastic Beanstalk

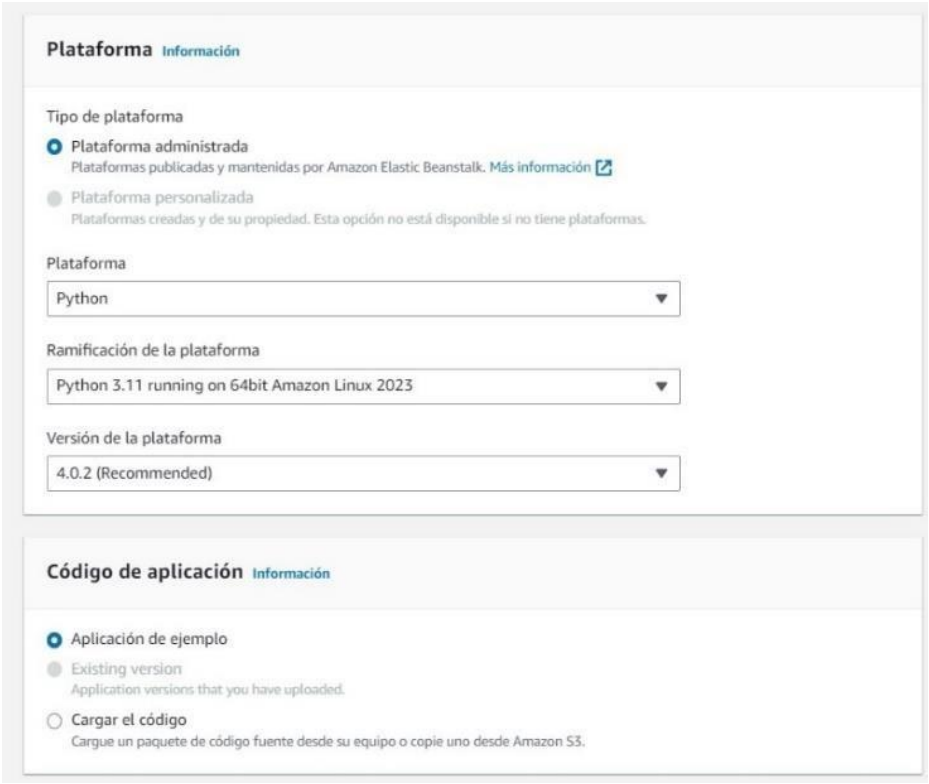


Nota. Ejemplo de creación de entorno. Consola de administración de AWS, 2023.

- Se seleccionó la versión de *Python* a utilizar y en la sección de Código de aplicación se seleccionó Aplicación de ejemplo**.**

Figura 51.

Versión de Python del entorno



Nota. Selección de versión de Python del entorno. Consola de administración de AWS, 2023.

- En la sección de Acceso al servicio, se realizaron algunas configuraciones adicionales antes de continuar con el mismo.

    - Primero se accedió al servicio de *EC2* y creó una instancia nueva la cual se utilizó como parte del entorno de ejecución gestionado por *Elastic Beanstalk*, cuando se despliega un proyecto, se crea de

manera automática un entorno de ejecución que incluye una o más instancias de *EC2* para ejecutar la aplicación. También se encarga de configurar y escalar las instancias según sean las necesidades del proyecto.

Figura 52.

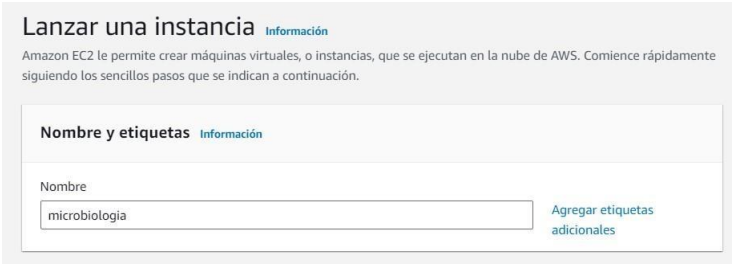
Nueva instancia de EC2

Nota. Creación de máquina virtual que almacenara el entorno. Consola de administración de *AWS*, 2023.

- Se le dio un nombre a la instancia, el nombre colocado hizo referencia a que servicio utilizó dicha instancia.

Figura 53.

Nombre de instancia de EC2

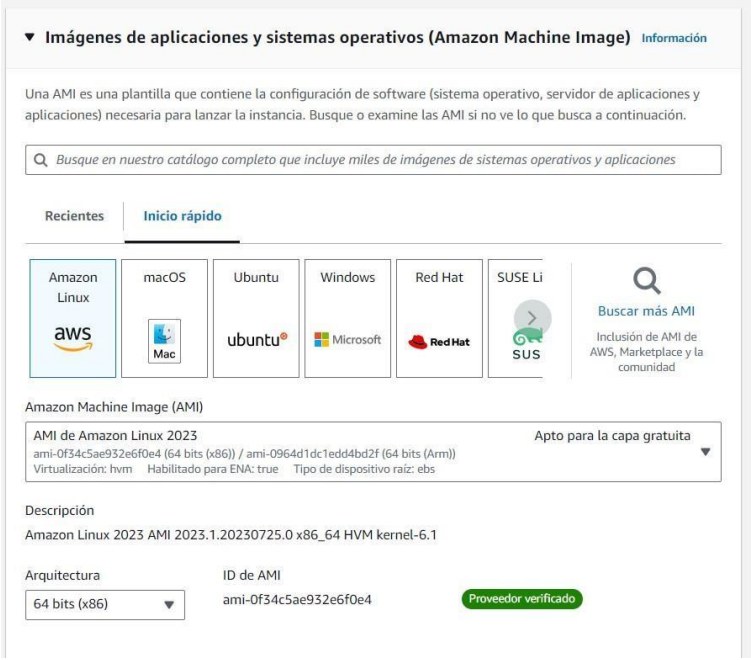


Nota. Ejemplo de nombre de máquina virtual. Consola de administración de *AWS*, 2023.

- Se seleccionó *Amazon Linux* como imagen del sistema operativo de la instancia, así como el tipo de arquitectura.

Figura 54.

Sistema operativo de instancia

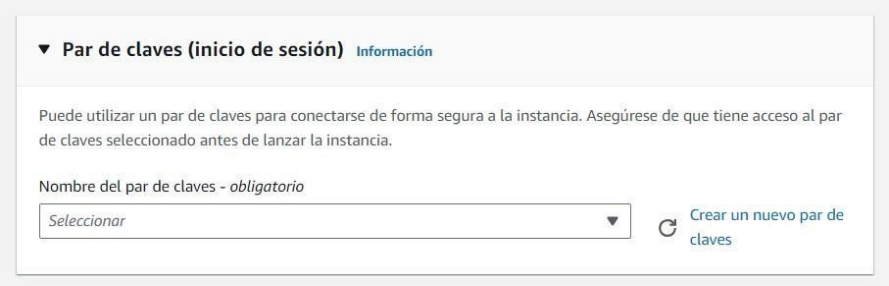


Nota. Selección de sistema operativo de máquina virtual. Consola de administración de *AWS*, 2023.

- Se crearon un nuevo par de claves de acceso para acceder a la instancia y el resto de las configuraciones se quedaron por defecto.

Figura 55.

Claves de acceso de instancia de EC2

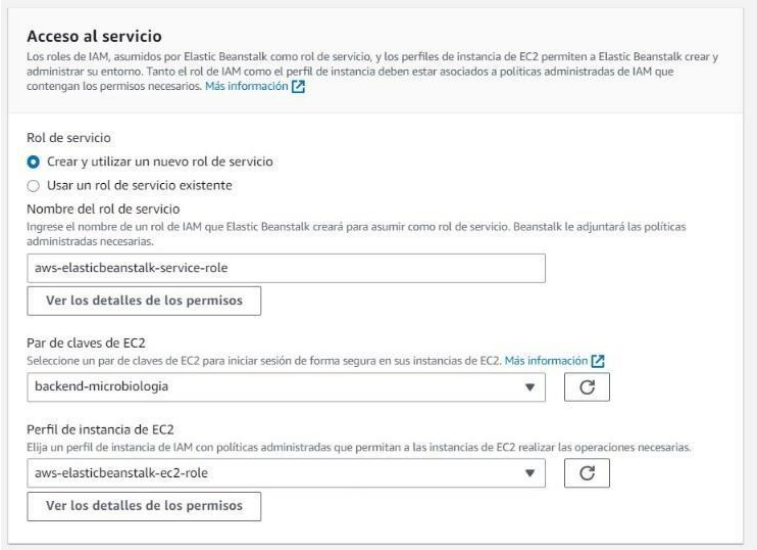


Nota. Selección de claves de acceso. Consola de administración de AWS, 2023.

- Regresando a la sección de Acceso al servidor, se seleccionó las claves generadas de la instancia anterior y se creó el entorno dejando el resto de las configuraciones por defecto.

Figura 56.

Acceso al servicio de Elastic Beanstalk

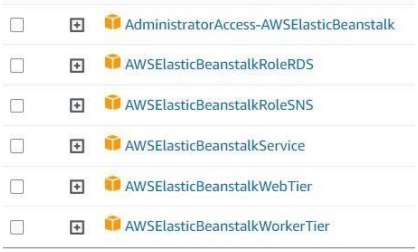


Nota. Detalles de instancia. Consola de administración de AWS, 2023.

- Por último, se creó un usuario de *IAM* que tuviera acceso completo al servicio de *Elastic Beanstalk*.

Figura 57.

Usuario de IAM



Nota. Asignación de permisos de usuario. Consola de administración de AWS, 2023.

Para luego colocar dichas credenciales en el archivo de configuración local *~/.aws/config*

Figura 58.

Archivo de configuración local de aws

```
[profile eb-cli]
aws_access_key_id = XXXXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXX

[profile eb-cli2]
aws_access_key_id = XXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXX
```

Nota. Credenciales de acceso locales. Elaboración propia.

Se actualizó el archivo *config.yml* para realizar el despliegue:

- El nombre del entorno debe ser actualizado al nombre nuevo, el actual es *microbiologiabackend-env*

Figura 59.

Archivo config.yml

```
branch-defaults:
  default:
    environment: microbiologiabackend-env
environment-defaults:
  microbiologiabackend-env:
    branch: null
    repository: null
```

Nota. Archivo de configuración yml. Elaboración propia.

- La sección de *application_name* y *default_ec2_keyname* se cambiaron por el nombre de la instancia de *EC2* y sus claves de acceso.

Figura 60.

Actualización de archivo config.yml

```
global:
  application_name: microbiologia-backend
  default_ec2_keyname: backend-microbiologia
```

Nota. Actualización de configuración de despliegue. Elaboración propia.

- Una vez cubiertos los requisitos anteriores fue posible publicar los cambios entrando a la raíz del proyecto */backend/API/backendodonto* y ejecutando el comando *eb deploy.*

2.3.4.4. Arquitectura final

La arquitectura final de la aplicación se explica a continuación:

Primero el desarrollador realiza los cambios o crea las funcionalidades correspondientes en el *front end* y envía los cambios a *GitHub*. Esto desencadena las *GitHub Actions* explicadas con anterioridad ejecutando las acciones descritas en el archivo *yml* donde a su vez valida los permisos para acceder a al *bucket* donde se almacenará la aplicación utilizando las credenciales del usuario de *AWS*, para luego realizar el *build* o compilado de la aplicación. Esto únicamente si se enviaron los cambios en la rama *máster* del repositorio de *GitHub*.

Posteriormente la acción de *GitHub* carga el compilado de la aplicación en *Amazon S3* e internamente invoca el servicio de *AWS CodeDeploy*, el cual “es un servicio que automatiza la implementación de aplicaciones en una variedad de entornos” (Amazon Web Services, Inc, s.f), este recupera los archivos almacenados en *S3* y desencadena el proceso de implementación en una instancia de *Amazon EC2* para luego devolver la ruta de los archivos almacenados en dicha instancia en formato *URL* para que el cliente pueda acceder a la página web.

Por su parte, el proceso de implementación del *back end* sigue un conjunto de pasos muy similar, primero, el desarrollador realiza los cambios correspondientes y envía el código a la consola de administración de *AWS* y este invoca el servicio de *AWS CodeBuild* el cual es un servicio que compilación de código.

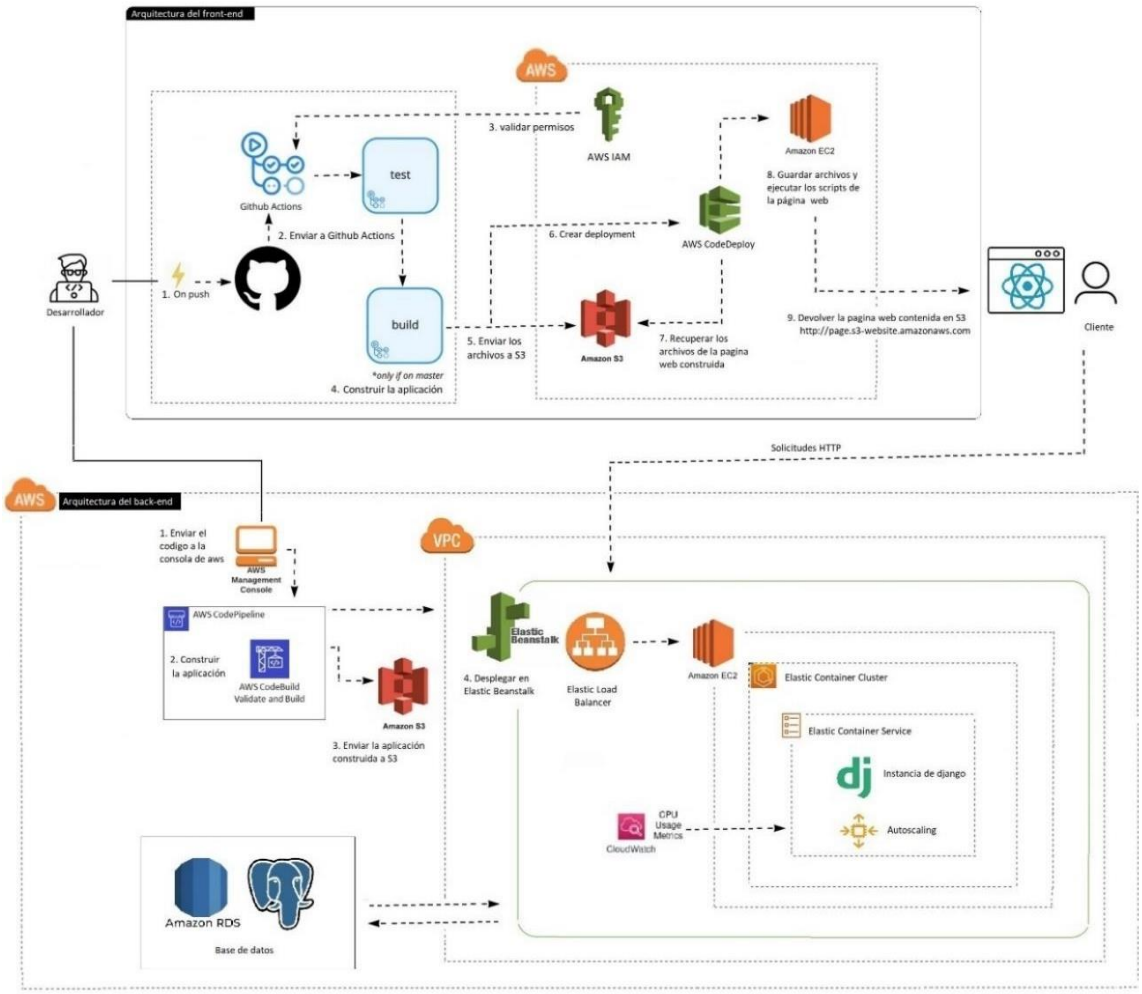
Una vez, el código este compilado y la aplicación construida, se guarda una copia de seguridad en un *bucket* de *Amazon S3* y ese mismo compilado se envía al entorno de ejecución de *Elastic Beanstalk*, el cual es el entorno donde se ejecutará la aplicación, este entorno puede ser de desarrollo, pruebas o producción y está provisto con varias opciones para configurar la capacidad de escalamiento, así como la infraestructura necesaria para hacer funcionar el *API*.

Dentro del entorno también se encuentra el balanceador de carga (*Load Balancer*) que *Elastic Beanstalk* configura de manera automática y cuya principal función es distribuir el tráfico entrante entre las instancias de la aplicación mejorando la disponibilidad y la escalabilidad. Así como también, se encuentra una instancia de *EC2* que *Elastic Beanstalk* utiliza para montar y ejecutar el *API* y por último contiene la aplicación misma la cual es un servidor de *Django*.

El *API* tiene configurada su conexión con el servicio de *Amazon RDS* para obtener los registros contenidos en la base de datos, realizar las correspondientes acciones y devolverle una respuesta a los usuarios que hagan uso de la aplicación.

Figura 61.

Arquitectura final de aplicación



Nota. Diagrama de arquitectura final de la aplicación. Elaboración propia, realizada en *Draw.io.*

.

2.4. Costos del proyecto

En la siguiente sección se listarán y detallarán los costos que se asocian al desarrollo del proyecto para el laboratorio de microbiología de la Facultad de Odontología, se tomaron en cuenta los recursos materiales, recursos humanos.

Tabla 1.

Presupuesto de aplicación del laboratorio de microbiología

Recursos	Cantidad	Costo unitario	Total
Equipo Lenovo Gaming 3 15IAH7	1	Q8,000.00	Q8,000.00
Internet 50 Mbs	6	Q350.00	Q2,100.00
Licencia Trimestral Filmora	2	Q155.00	Q310.00
Servicio Elastick Load Balancing mensual	6	Q280.00	Q1680.00
Relational Database Service mensual	6	Q160.00	Q960.00
TOTAL			**Q13,050.00**

Nota. Elaboración propia, realizada con Microsoft Word.

2.5. Beneficios del proyecto

Los principales beneficios que obtendrá el laboratorio de microbiología de la Facultad de Odontología es una aplicación web intuitiva, fácil de utilizar y de gran utilidad para la gestión de sus pacientes clínicos y sus resultados de laboratorio. Entre los principales beneficios que obtendrá el laboratorio se encuentran:

- Creación, modificación y eliminación de pacientes clínicos y antecedentes médicos.
- Creación, modificación y eliminación de exámenes clínicos con sus respectivos perfiles clínicos, es decir, exámenes más pequeños y sus respectivos campos de resultados para guardar la información correspondiente.

- Creación, modificación y eliminación de solicitudes de exámenes, con la opción de poder ingresar los resultados obtenidos y poder enviar por correo al paciente.
- Creación, modificación y eliminación de usuarios para que hagan uso del sistema, con la opción de poder enviar correo de verificación para validar el usuario en el sistema.
- Creación, modificación y eliminación de roles de usuario donde a cada rol se le asignan uno o varios permisos para que los usuarios con ese rol asignado puedan realizar las acciones correspondientes en la aplicación.
- Creación, modificación y eliminación de pagos y solicitudes de pagos para llevar un control de los ingresos del laboratorio.
- Automatización de la creación de una bitácora o historial general donde se listen todas las acciones realizadas por los usuarios.
- Automatización de la creación de una bitácora o historial por módulo donde se listen todas las acciones realizadas por los usuarios sobre ese modulo en específico.

Al igual que tendrán un manual tanto de usuario como de desarrollo que servirá como guía para futuros desarrollos que deban dar soporte a la aplicación del laboratorio de microbiología de la Universidad de San Carlos de Guatemala.

3. FASE ENSEÑANZA APRENDIZAJE

3.1. Capacitación propuesta

La capacitación propuesta se realizó con los encargados del laboratorio de microbiología de la Facultad de Odontología se desarrolló de la siguiente manera:

3.1.1. Capacitación con los doctores principales del laboratorio de microbiología

Se llevaron a cabo 5 reuniones con los interesados para presentar los avances correspondientes a las fechas de las reuniones, en la primera reunión se presentó avances acerca de los usuarios y roles del sistema que se trabajó dentro de la página del laboratorio. En la segunda reunión se presentaron avances respecto a los exámenes clínicos del laboratorio. En la tercera reunión se presentó avances referentes a pacientes y solicitudes de exámenes clínicos.

Durante la cuarta reunión se mostraron avances en lo que refiere a los pagos y al historial de acciones y por último, en la quinta reunión se mostraron avances en la parte de análisis estadístico solicitado por el propio laboratorio y se le dio finalización a la construcción de la aplicación.

3.2. Material elaborado

A continuación, se listará la documentación desarrollada y presentada a los encargados del laboratorio de microbiología.

3.2.1. Manual de usuario

Este manual consiste en una serie de videos donde se específica como hacer uso de la aplicación, dicho manual explica de manera detallada modulo por modulo y como hacer uso de este, es decir cuáles son los datos que se debe ingresar, como se deben ingresar, que datos se mostrarán, qué relación tiene con otros módulos, como un módulo afecta directa o indirectamente a otros, entre otros.

3.2.2. Manual técnico de *front end* y *back end*

Los manuales técnicos contienen información detallada sobre la estructura, funcionalidad y uso de la aplicación. En ellos se listan cuáles son los requisitos del sistema, instrucciones paso a paso sobre cómo instalar las aplicaciones en diferentes plataformas, detalles sobre cómo configurar la aplicación, incluyendo la conexión a bases de datos, configuración de seguridad. la arquitectura de cada aplicación, las versiones de las herramientas utilizadas y principalmente las dependencias que se necesitan para hacer funcionar las aplicaciones.

3.2.3. Manual de despliegue de *front end* y *back end*

Los manuales de despliegue proporcionan instrucciones detalladas sobre cómo implementar y configurar tanto el *front end* como el *back end* en un entorno

de producción. Los manuales incluyen especificaciones técnicas necesarias para el entorno de producción, incluyendo la base de datos utilizada, el sistema operativo, la plataforma utilizada, la versión del servidor *web*, entre otros. Así como también incluyen instrucciones sobre cómo preparar el entorno de producción incluyendo las configuraciones locales que se deben realizar para poder realizar el respectivo despliegue y por último el proceso de despliegue en el que se listan las instrucciones paso a paso sobre cómo implementar el *front end* y *back end* en su respectivo entorno incluyendo la configuración de permisos e instalación de dependencias.

CONCLUSIONES

1. Una aplicación escalable es un sistema diseñado para que esta pueda manejar de forma eficiente un aumento significativo de la carga de trabajo sin que afecte el rendimiento, calidad y disponibilidad, es decir, una aplicación escalable es aquella que puede crecer con el tiempo y adaptarse a medida que aumenten la necesidades o requerimientos sin necesidad de realizar cambios en la estructura o arquitectura de esta.

2. Un servidor de aplicación es un *software* diseñado exclusivamente para proveer un entorno donde las aplicaciones *web* puedan ejecutarse y operar de manera eficiente. Éste actúa como intermediario entre el servidor *web*, que maneja las solicitudes de los usuarios, y la aplicación en sí. Tiene como objetivo principal administrar la ejecución de la aplicación, así como proveer servicios que faciliten el funcionamiento de esta.

3. Con la finalidad de proporcionar apoyo a futuros desarrolladores de la Universidad de San Carlos de Guatemala, se crearon un conjunto de manuales de apoyo que detallan paso a paso la construcción, funcionamiento y despliegue del proyecto completo, ilustrándolo por medio de imágenes o videos para su correcta aplicación.

RECOMENDACIONES

1. Realizar actualizaciones de los manuales de despliegue, usuario y técnicos a medida que el proyecto vaya crezca, además de procurar siempre utilizar las herramientas más actualizadas y los procesos más eficientes, así como dar explicaciones detalladas y fáciles de seguir.

2. Procurar trabajar con sumo cuidado la cuenta de *Amazon Web Services* provista para el despliegue del proyecto ya que los servicios a utilizar tienen un costo, costo que incurre en el dueño de la cuenta por lo que no se debe jugar con los servicios, así como así y solo se deben hacer uso de ellos al momento de realizar el despliegue correspondiente.

3. Trabajar las incidencias encontradas, tanto del *front end* como del *back end,* en ramas diferentes a la rama *master* de los repositorios, esto para mantener el código de dicha rama intacto y con la última versión estable del proyecto.

4. Reutilizar código siempre que sea posible, así como limpiar constantemente el código que deje de ser de utilidad o se encuentre obsoleto.

5. Realizar mantenimiento del código en un futuro, tanto del *back end* como del *front end,* para mantener la aplicación limpia y lo más actualizada posible.

REFERENCIAS

Amazon Web Services, Inc. (marzo de 2023). *AWS CodeDeploy.* https://aws.amazon.com/es/codedeploy/.

Amazon Web Services, Inc. (marzo de 2023). *¿Qué es un API?* https://aws.amazon.com/es/what-is/api/.

Amazon Web Services, Inc. (marzo de 2023). *¿Qué es una aplicación web?* https://aws.amazon.com/es/what-is/web-application/.

Amazon Web Services, Inc. (marzo de 2023). *¿Qué es Amazon S3?* https://aws.amazon.com/es/pm/serv-s3/.

Amazon Web Services, Inc. (marzo de 2023). *¿Qué es AWS Elastic Beanstalk?* https://docs.aws.amazon.com/es_es/elasticbeanstalk/latest/dg/Welcome.html.

Amazon Web Services, Inc. (abril de 2023). *¿Qué es Django?* https://aws.amazon.com/es/what-is/django/.

Amazon Web Services, Inc. (mayo de 2023). *¿Qué es JavaScript?* https://aws.amazon.com/es/what-is/javascript/.

Amazon Web Services, Inc. (marzo de 2023). *¿Qué son los microservicios?* https://aws.amazon.com/es/microservices/.

Amazon Web Services, Inc. (mayo de 2023). *¿Qué es XML?* https://aws.amazon.com/es/what-is/xml/.

Doonamis. (mayo de 2017). *Para qué sirve React JS: beneficios y ejemplos.* https://www.doonamis.es/para-que-sirve-react-js-beneficios-para-tus-apps/.

Django. (junio de 2023). *Meet Django.* https://www.djangoproject.com/ .

EcuRed. (junio de 2023). *Programación orientada a componentes.* https://www.ecured.cu/Programación_orientada_a_componentes.

Facultad de Odontología. (2023). *CATALOGO DE ESTUDIOS 2008.* https://www.usac.edu.gt/catalogo/odontologia.pdf.

Facultad de Odontología. (2023). *Misión y Visión.* https://odontologia.usac.edu.gt/?page_id=509.

García, M. (20 de noviembre de 2023). *React – React Router. Medium.* https://mauriciogc.medium.com/react-react-router-db391b3def2a.

GitHub. (julio de 2023). *Documentación sobre las Acciones de GitHub.* https://docs.github.com/es/actions.

Hubspot. (mayo de 2023). *¿Qué es JSON?* https://blog.hubspot.es/website/que-es-json.

JWT.IO. (enero de 2023). *What is JSON Web Token?* https://jwt.io/introduction.

Lucidchart. (mayo de 2023). *¿Qué es un modelo de bases de datos?* https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos.

Microsoft. (abril de 2023). *¿Qué son las Acciones de GitHub para Azure?* https://learn.microsoft.com/es-es/azure/developer/github/github-actions.

Microsoft. (abril de 2023). *¿Qué es la administración de identidad y acceso (IAM)?* https://learn.microsoft.com/es-es/entra/fundamentals/introduction-identity-access-management.

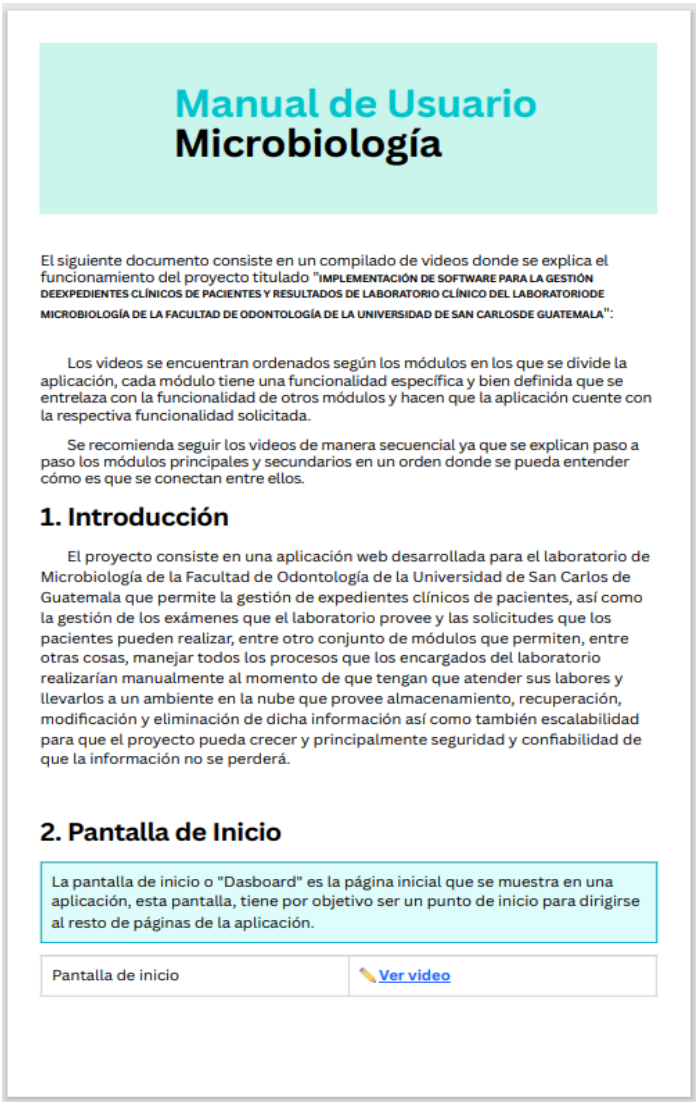
Reactjs. (abril de 2023). *Context.* https://legacy.reactjs.org/docs/context.html.

Reactjs. (abril de 2023). *Un vistazo a los Hooks.* https://es.legacy.reactjs.org/docs/hooks-overview.html.

APÉNDICES

Apéndice 1.

Manual de usuario – Introducción



Manual de Usuario
Microbiología

El siguiente documento consiste en un compilado de videos donde se explica el funcionamiento del proyecto titulado "IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DEEXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIODE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOSDE GUATEMALA":

Los videos se encuentran ordenados según los módulos en los que se divide la aplicación, cada módulo tiene una funcionalidad específica y bien definida que se entrelaza con la funcionalidad de otros módulos y hacen que la aplicación cuente con la respectiva funcionalidad solicitada.

Se recomienda seguir los videos de manera secuencial ya que se explican paso a paso los módulos principales y secundarios en un orden donde se pueda entender cómo es que se conectan entre ellos.

1. Introducción

El proyecto consiste en una aplicación web desarrollada para el laboratorio de Microbiología de la Facultad de Odontología de la Universidad de San Carlos de Guatemala que permite la gestión de expedientes clínicos de pacientes, así como la gestión de los exámenes que el laboratorio provee y las solicitudes que los pacientes pueden realizar, entre otro conjunto de módulos que permiten, entre otras cosas, manejar todos los procesos que los encargados del laboratorio realizarían manualmente al momento de que tengan que atender sus labores y llevarlos a un ambiente en la nube que provee almacenamiento, recuperación, modificación y eliminación de dicha información así como también escalabilidad para que el proyecto pueda crecer y principalmente seguridad y confiabilidad de que la información no se perderá.

2. Pantalla de Inicio

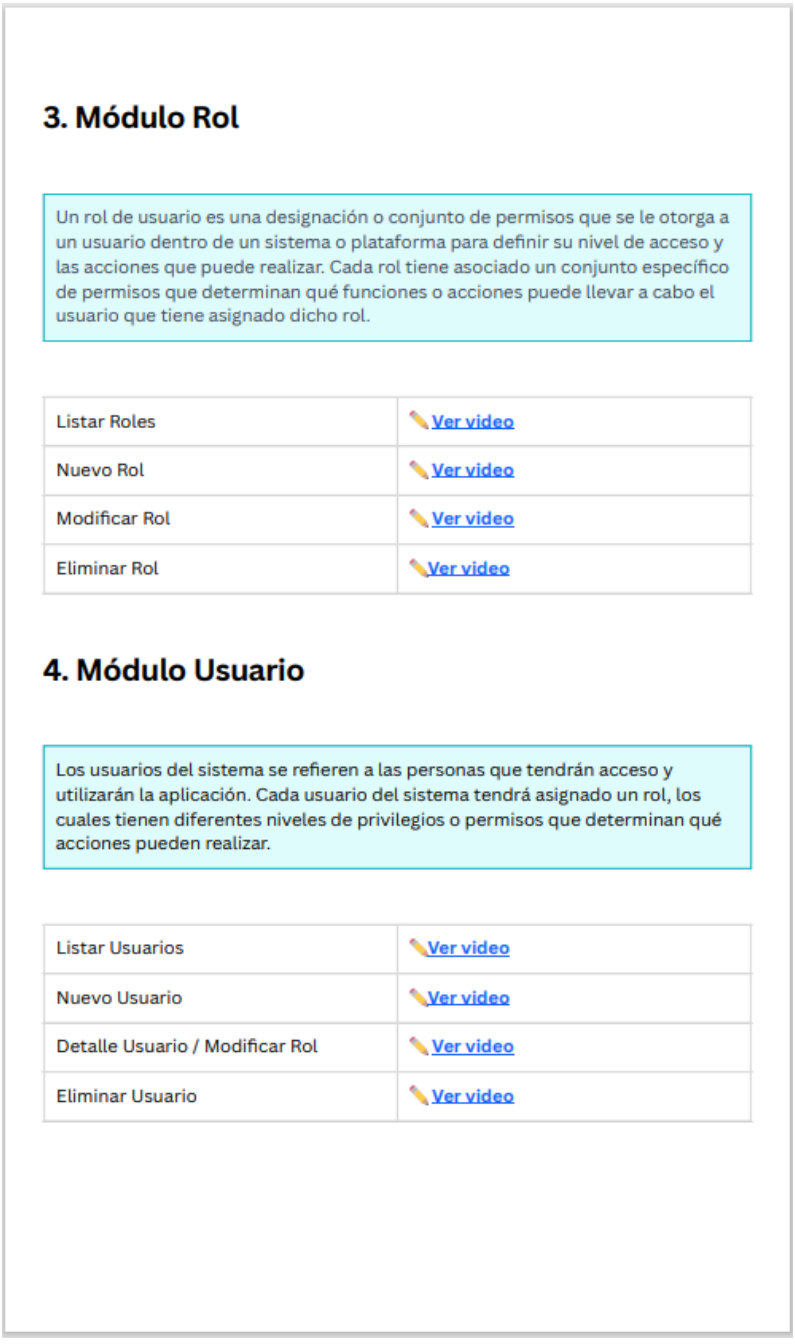
La pantalla de inicio o "Dasboard" es la página inicial que se muestra en una aplicación, esta pantalla, tiene por objetivo ser un punto de inicio para dirigirse al resto de páginas de la aplicación.

Pantalla de inicio	Ver video

Nota. Elaboración propia, realizado en *Microsoft Word*.

Apéndice 2.

Manual de usuario – Sección de Rol y Usuario



3. Módulo Rol

Un rol de usuario es una designación o conjunto de permisos que se le otorga a un usuario dentro de un sistema o plataforma para definir su nivel de acceso y las acciones que puede realizar. Cada rol tiene asociado un conjunto específico de permisos que determinan qué funciones o acciones puede llevar a cabo el usuario que tiene asignado dicho rol.

Listar Roles	✏️ Ver video
Nuevo Rol	✏️ Ver video
Modificar Rol	✏️ Ver video
Eliminar Rol	✏️ Ver video

4. Módulo Usuario

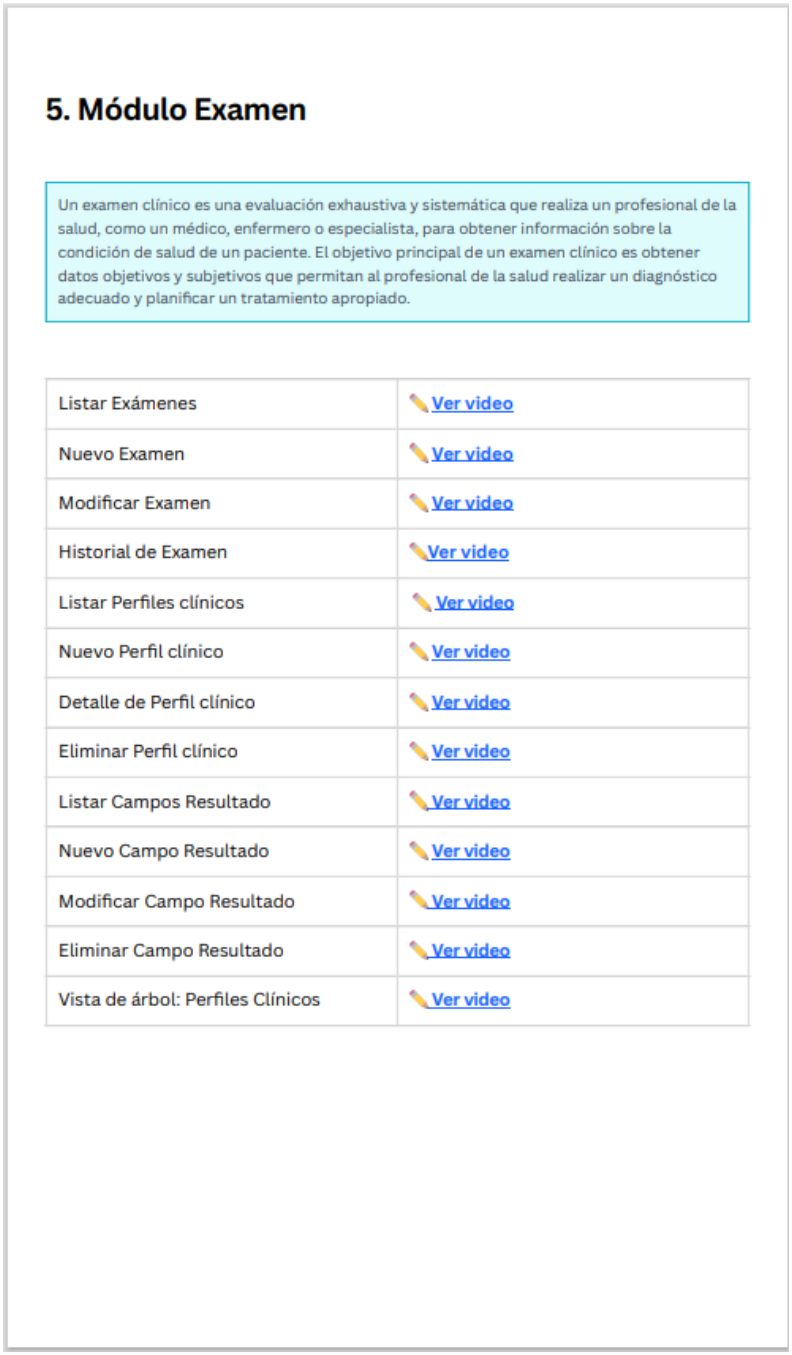
Los usuarios del sistema se refieren a las personas que tendrán acceso y utilizarán la aplicación. Cada usuario del sistema tendrá asignado un rol, los cuales tienen diferentes niveles de privilegios o permisos que determinan qué acciones pueden realizar.

Listar Usuarios	✏️ Ver video
Nuevo Usuario	✏️ Ver video
Detalle Usuario / Modificar Rol	✏️ Ver video
Eliminar Usuario	✏️ Ver video

Nota. Elaboración propia, realizado en *Microsoft Word*.

Apéndice 3.

Manual de usuario – Sección de Examen



5. Módulo Examen

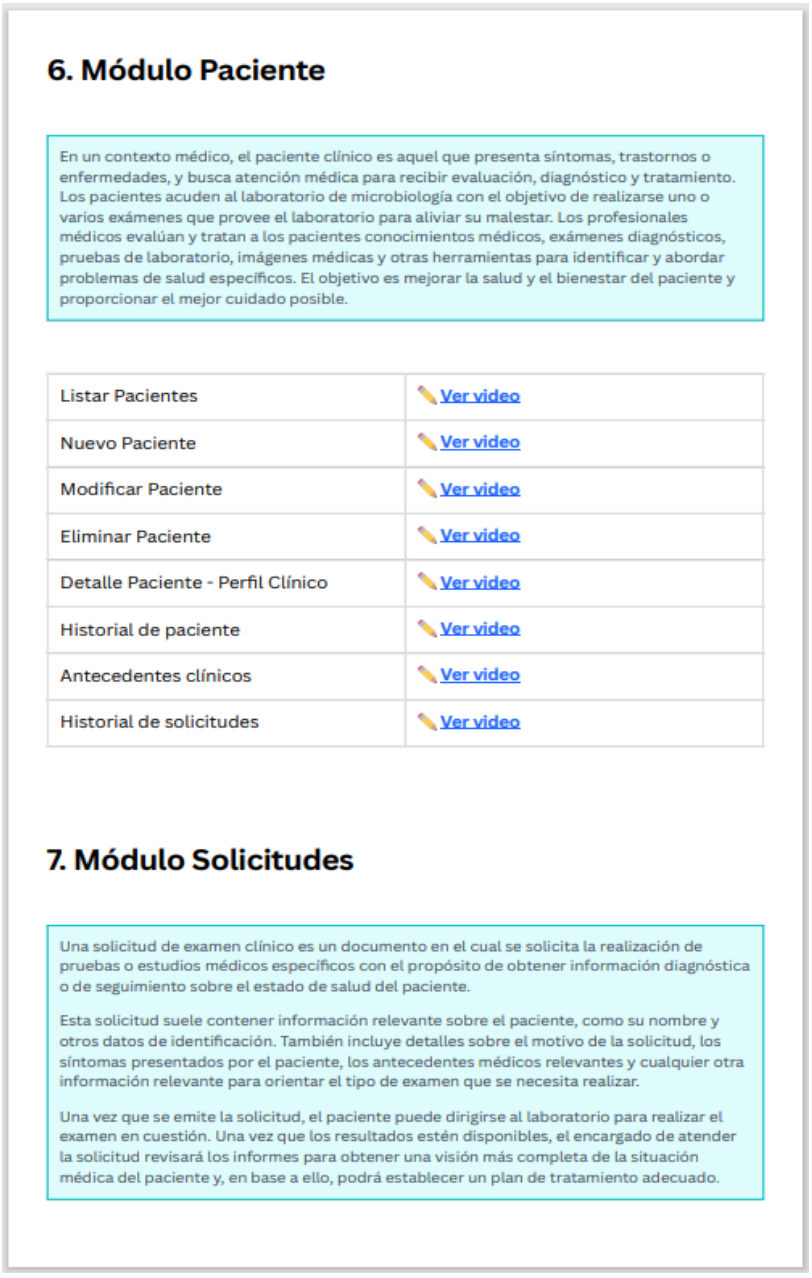
Un examen clínico es una evaluación exhaustiva y sistemática que realiza un profesional de la salud, como un médico, enfermero o especialista, para obtener información sobre la condición de salud de un paciente. El objetivo principal de un examen clínico es obtener datos objetivos y subjetivos que permitan al profesional de la salud realizar un diagnóstico adecuado y planificar un tratamiento apropiado.

Listar Exámenes	✏️ Ver video
Nuevo Examen	✏️ Ver video
Modificar Examen	✏️ Ver video
Historial de Examen	✏️ Ver video
Listar Perfiles clínicos	✏️ Ver video
Nuevo Perfil clínico	✏️ Ver video
Detalle de Perfil clínico	✏️ Ver video
Eliminar Perfil clínico	✏️ Ver video
Listar Campos Resultado	✏️ Ver video
Nuevo Campo Resultado	✏️ Ver video
Modificar Campo Resultado	✏️ Ver video
Eliminar Campo Resultado	✏️ Ver video
Vista de árbol: Perfiles Clínicos	✏️ Ver video

Nota. Elaboración propia, realizado en *Microsoft Word*.

Apéndice 4.

Manual de usuario – Sección de Paciente y Solicitud



6. Módulo Paciente

En un contexto médico, el paciente clínico es aquel que presenta síntomas, trastornos o enfermedades, y busca atención médica para recibir evaluación, diagnóstico y tratamiento. Los pacientes acuden al laboratorio de microbiología con el objetivo de realizarse uno o varios exámenes que provee el laboratorio para aliviar su malestar. Los profesionales médicos evalúan y tratan a los pacientes conocimientos médicos, exámenes diagnósticos, pruebas de laboratorio, imágenes médicas y otras herramientas para identificar y abordar problemas de salud específicos. El objetivo es mejorar la salud y el bienestar del paciente y proporcionar el mejor cuidado posible.

Listar Pacientes	✏️ Ver video
Nuevo Paciente	✏️ Ver video
Modificar Paciente	✏️ Ver video
Eliminar Paciente	✏️ Ver video
Detalle Paciente - Perfil Clínico	✏️ Ver video
Historial de paciente	✏️ Ver video
Antecedentes clínicos	✏️ Ver video
Historial de solicitudes	✏️ Ver video

7. Módulo Solicitudes

Una solicitud de examen clínico es un documento en el cual se solicita la realización de pruebas o estudios médicos específicos con el propósito de obtener información diagnóstica o de seguimiento sobre el estado de salud del paciente.

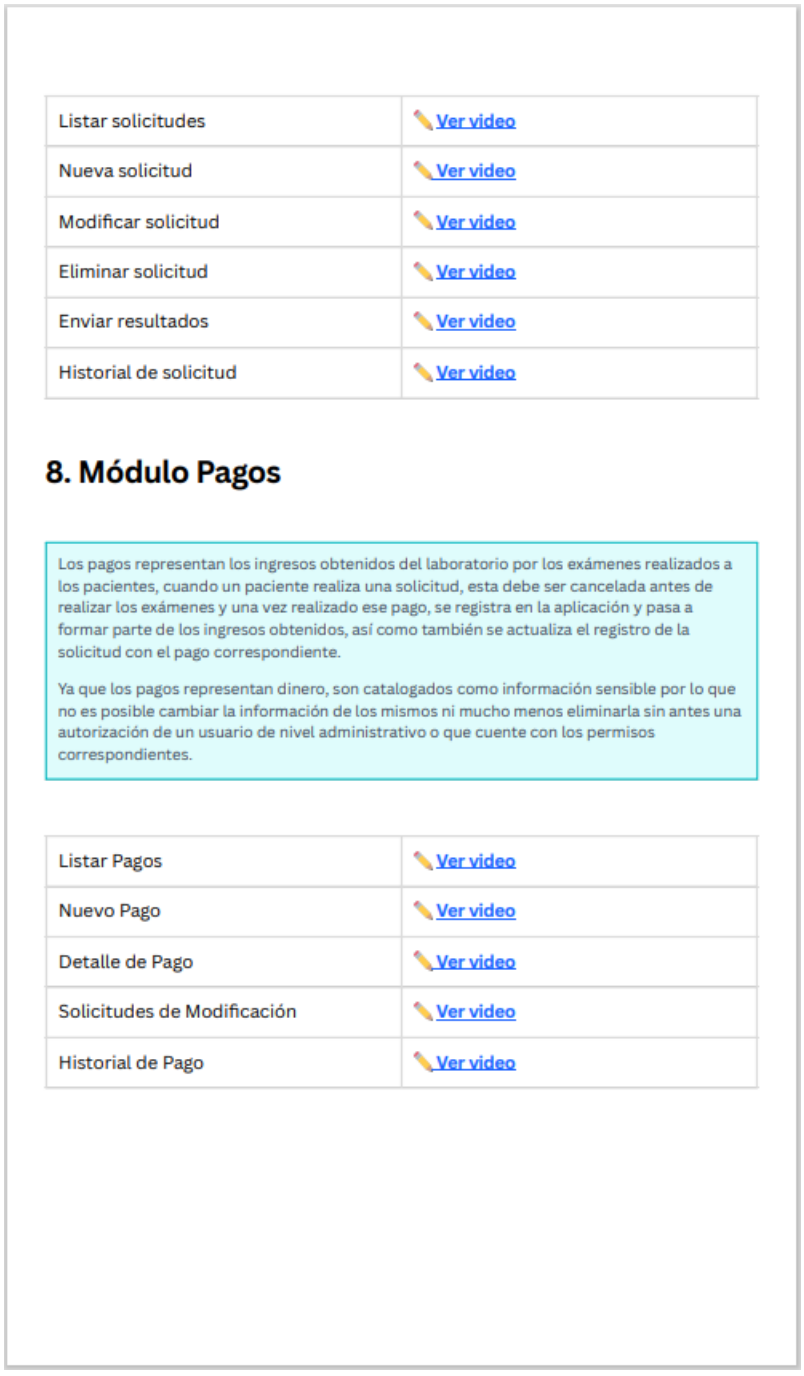
Esta solicitud suele contener información relevante sobre el paciente, como su nombre y otros datos de identificación. También incluye detalles sobre el motivo de la solicitud, los síntomas presentados por el paciente, los antecedentes médicos relevantes y cualquier otra información relevante para orientar el tipo de examen que se necesita realizar.

Una vez que se emite la solicitud, el paciente puede dirigirse al laboratorio para realizar el examen en cuestión. Una vez que los resultados estén disponibles, el encargado de atender la solicitud revisará los informes para obtener una visión más completa de la situación médica del paciente y, en base a ello, podrá establecer un plan de tratamiento adecuado.

Nota. Elaboración propia, realizado en Microsoft Word.

Apéndice 5.

Manual de usuario – Sección de Solicitudes y Pagos



Listar solicitudes	✏️ Ver video
Nueva solicitud	✏️ Ver video
Modificar solicitud	✏️ Ver video
Eliminar solicitud	✏️ Ver video
Enviar resultados	✏️ Ver video
Historial de solicitud	✏️ Ver video

8. Módulo Pagos

Los pagos representan los ingresos obtenidos del laboratorio por los exámenes realizados a los pacientes, cuando un paciente realiza una solicitud, esta debe ser cancelada antes de realizar los exámenes y una vez realizado ese pago, se registra en la aplicación y pasa a formar parte de los ingresos obtenidos, así como también se actualiza el registro de la solicitud con el pago correspondiente.

Ya que los pagos representan dinero, son catalogados como información sensible por lo que no es posible cambiar la información de los mismos ni mucho menos eliminarla sin antes una autorización de un usuario de nivel administrativo o que cuente con los permisos correspondientes.

Listar Pagos	✏️ Ver video
Nuevo Pago	✏️ Ver video
Detalle de Pago	✏️ Ver video
Solicitudes de Modificación	✏️ Ver video
Historial de Pago	✏️ Ver video

Nota. Elaboración propia, realizado en *Microsoft Word*.

Apéndice 6.

Manual de usuario – Sección de Historial y Estadística

9. Módulo Historial

El historial de usuario se refiere al registro o seguimiento de las acciones y actividades que un usuario específico ha realizado dentro de la aplicación a lo largo del tiempo. Este registro puede incluir información detallada sobre las interacciones del usuario, como las funciones utilizadas, las opciones seleccionadas y cualquier otro tipo de acción realizada dentro de la aplicación.

Historial general	✏️ Ver video

10. Módulo Estadístico

Un módulo estadístico en el contexto de la aplicación se está diseñado específicamente para realizar análisis y cálculos estadísticos de algunos de los módulos de la misma. Este módulo puede contiene funciones y herramientas que permiten recopilar, procesar y presentar datos en forma de estadísticas, gráficos, tablas y otros elementos visuales que ayudan a comprender patrones, tendencias y relaciones en los datos de pacientes, exámenes e ingresos.

Este módulo pretende ayudar a los especialistas del laboratorio a extraer información significativa de grandes conjuntos de datos y a tomar decisiones informadas y fundamentadas en análisis estadísticos sólidos.

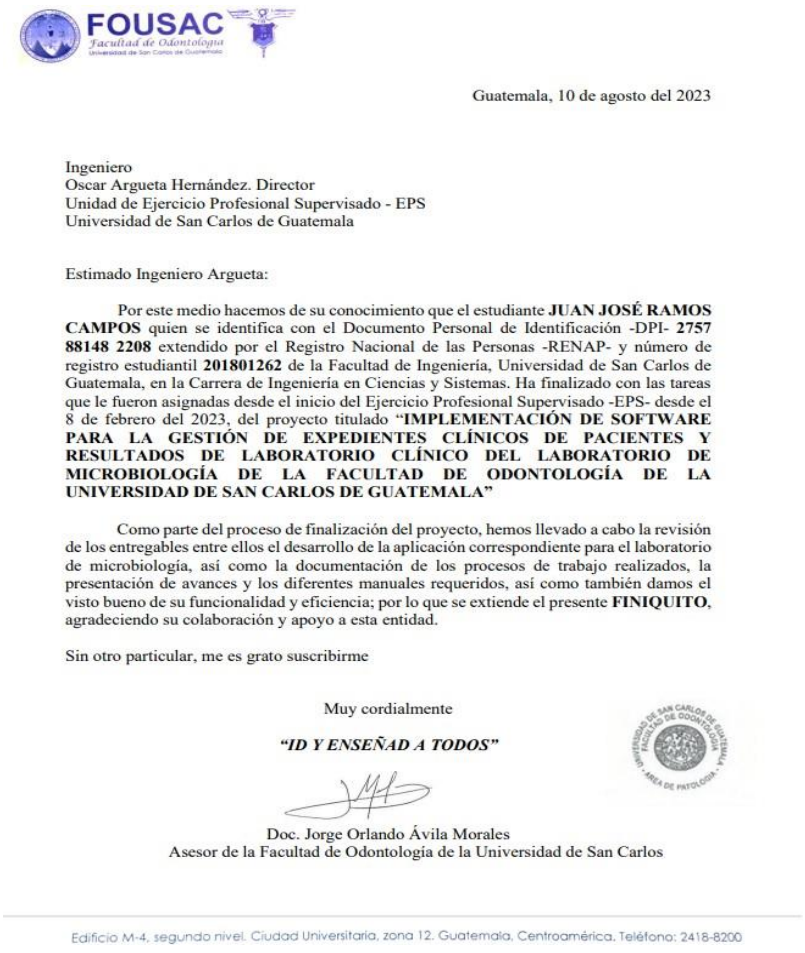
Listado de análisis	✏️ Ver video
Estadística de pacientes	✏️ Ver video
Estadística de edades	✏️ Ver video
Estadística de sexos	✏️ Ver video
Estadística de etnias	✏️ Ver video
Estadística de exámenes	✏️ Ver video
Estadística de ingresos	✏️ Ver video

Nota. Elaboración propia, realizado en *Microsoft Word*.

ANEXOS

Anexo 1.

Carta finiquito, firmada por el asesor de la Facultad de Odontología



FOUSAC
Facultad de Odontología
Universidad de San Carlos de Guatemala

Guatemala, 10 de agosto del 2023

Ingeniero
Oscar Argueta Hernández. Director
Unidad de Ejercicio Profesional Supervisado - EPS
Universidad de San Carlos de Guatemala

Estimado Ingeniero Argueta:

Por este medio hacemos de su conocimiento que el estudiante **JUAN JOSÉ RAMOS CAMPOS** quien se identifica con el Documento Personal de Identificación -DPI- **2757 88148 2208** extendido por el Registro Nacional de las Personas -RENAP- y número de registro estudiantil **201801262** de la Facultad de Ingeniería, Universidad de San Carlos de Guatemala, en la Carrera de Ingeniería en Ciencias y Sistemas. Ha finalizado con las tareas que le fueron asignadas desde el inicio del Ejercicio Profesional Supervisado -EPS- desde el 8 de febrero del 2023, del proyecto titulado "**IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**"

Como parte del proceso de finalización del proyecto, hemos llevado a cabo la revisión de los entregables entre ellos el desarrollo de la aplicación correspondiente para el laboratorio de microbiología, así como la documentación de los procesos de trabajo realizados, la presentación de avances y los diferentes manuales requeridos, así como también damos el visto bueno de su funcionalidad y eficiencia; por lo que se extiende el presente **FINIQUITO**, agradeciendo su colaboración y apoyo a esta entidad.

Sin otro particular, me es grato suscribirme

Muy cordialmente

"ID Y ENSEÑAD A TODOS"

Doc. Jorge Orlando Ávila Morales
Asesor de la Facultad de Odontología de la Universidad de San Carlos

Edificio M-4, segundo nivel. Ciudad Universitaria, zona 12. Guatemala, Centroamérica. Teléfono: 2418-8200

Nota. Carta de finalización de proyecto firmada por el asesor de institución. Facultad de Odontología.

Anexo 2.

Carta de finalización de informe de la Facultad de Odontología

FOUSAC
Facultad de Odontología
Universidad de San Carlos de Guatemala

Guatemala, 10 de agosto del 2023

Ingeniero
Oscar Argueta Hernández. Director
Unidad de Ejercicio Profesional Supervisado - EPS
Universidad de San Carlos de Guatemala

Estimado Ingeniero Argueta:

Por este medio hago de su conocimiento que el estudiante **JUAN JOSÉ RAMOS CAMPOS** quien se identifica con el Documento Personal de Identificación -DPI- **2757 88148 2208** extendido por el Registro Nacional de las Personas -RENAP- y número de registro estudiantil **201801262** de la Facultad de Ingeniería, Universidad de San Carlos de Guatemala, en la Carrera de Ingeniería en Ciencias y Sistemas. Ha culminado su informe final del Ejercicio Profesional Supervisado -EPS-, titulado "**IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**"

Por lo tanto, doy por aprobado su trabajo para continuar con los trámites respectivos. Sin otro particular me es grato suscribirme.

Muy cordialmente

"ID Y ENSEÑAD A TODOS"

Doc. Jorge Orlando Ávila Morales
Asesor de la Facultad de Odontología de la Universidad de San Carlos

Edificio M-4, segundo nivel. Ciudad Universitaria, zona 12. Guatemala, Centroamérica. Teléfono: 2418-8200

Nota. Carta de finalización de informe firmada por el asesor de institución. Facultad de Odontología.

Anexo 3.

Carta de finalización de proyecto de la Facultad de Odontología

FOUSAC
Facultad de Odontología
Universidad de San Carlos de Guatemala

Guatemala, 10 de agosto del 2023

Ingeniero
Oscar Argueta Hernández. Director
Unidad de Ejercicio Profesional Supervisado - EPS
Universidad de San Carlos de Guatemala

Estimado Ingeniero Argueta:

Por este medio hago de su conocimiento que el estudiante **JUAN JOSÉ RAMOS CAMPOS** quien se identifica con el Documento Personal de Identificación -DPI- **2757 88148 2208** extendido por el Registro Nacional de las Personas -RENAP- y número de registro estudiantil **201801262** de la Facultad de Ingeniería, Universidad de San Carlos de Guatemala, en la Carrera de Ingeniería en Ciencias y Sistemas. Ha culminado su proyecto del Ejercicio Profesional Supervisado -EPS-, titulado "**IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**"

Sin otro particular, me es grato suscribirme

Muy cordialmente

"ID Y ENSEÑAD A TODOS"

Doc. Jorge Orlando Ávila Morales
Asesor de la Facultad de Odontología de la Universidad de San Carlos

Edificio M-4, segundo nivel, Ciudad Universitaria, zona 12, Guatemala, Centroamérica, Teléfono: 2418-8200

Nota. Carta de finalización de proyecto firmada por el asesor de institución. Facultad de Odontología.

Anexo 4.

Carta de finalización de informe de la Escuela de Ciencias y Sistemas

Guatemala, 10 de agosto del 2023

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Unidad de Ejercicio Profesional Supervisado - EPS
Dirección
Ing. Oscar Argueta Hernández

Estimado director, por este medio hago de su conocimiento que el estudiante **JUAN JOSÉ RAMOS CAMPOS** quien se identifica con el Documento Personal de Identificación -DPI- **2757 88148 2208** extendido por el Registro Nacional de las Personas -RENAP- y número de registro estudiantil **201801262** de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala en la Carrera de Ingeniería en Ciencias y Sistemas. Ha finalizado su informe final del Ejercicio Profesional Supervisado -EPS-, titulado "**IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA".**

Por lo tanto, doy por aprobado su trabajo para continuar con los trámites respectivos. Sin otro particular, me es grato suscribirme

Muy cordialmente

"ID Y ENSEÑAD A TODOS"

Marco Tulio Aldana Prillwitz
Master in Bussiness Intelligence and Data Analytics
Colegio de Humanidades 30747

Msc. Lic. Marco Tulio Aldana Prillwitz
Asesor de EPS
Escuela de Ingeniería en Ciencias y Sistemas
Colegiado 30747

Nota. Carta de finalización de informe firmada por el asesor de escuela. Escuela de Ciencias y Sistemas.

Anexo 5.

Carta de finalización de proyecto de la Escuela de Ciencias y Sistemas

Guatemala, 10 de agosto del 2023

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Unidad de Ejercicio Profesional Supervisado - EPS
Dirección
Ing. Oscar Argueta Hernández

Estimado director, por este medio hago de su conocimiento que el estudiante **JUAN JOSÉ RAMOS CAMPOS** quien se identifica con el Documento Personal de Identificación -DPI- **2757 88148 2208** extendido por el Registro Nacional de las Personas -RENAP- y número de registro estudiantil **201801262** de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala en la Carrera de Ingeniería en Ciencias y Sistemas. Ha finalizado su proyecto del Ejercicio Profesional Supervisado -EPS-, titulado "**IMPLEMENTACIÓN DE SOFTWARE PARA LA GESTIÓN DE EXPEDIENTES CLÍNICOS DE PACIENTES Y RESULTADOS DE LABORATORIO CLÍNICO DEL LABORATORIO DE MICROBIOLOGÍA DE LA FACULTAD DE ODONTOLOGÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA".**

Sin otro particular, me es grato suscribirme

Muy cordialmente

"ID Y ENSEÑAD A TODOS"

Marco Tulio Aldana Prillwitz
Master in Business Intelligence and Data Analytics
Colegio de Humanidades 30747

Msc. Lic. Marco Tulio Aldana Prillwitz
Asesor de EPS
Escuela de Ingeniería en Ciencias y Sistemas
Colegiado 30747

Nota. Carta de finalización de proyecto firmada por el asesor de escuela. Escuela de Ciencias y Sistemas.