MANUAL TECNICO - APLICATIVO WEB PARA FACILITAR LA CITACIÓN EXPEDITA DE LOS PACIENTES A LAS ÁREAS DE TRIAGE, CONSULTORIOS MÉDICOS, Y ÁREA DE PROCEDIMIENTOS

RICARDO FELIPE MARIÑO PERDOMO

UNIVERSIDAD PILOTO DE COLOMBIA

SECCIONAL ALTO MAGDALENA

INGENIERÍA DE SISTEMAS

CLINICA JUAN N CORPAS

BOGOTA D.C.

2024

TABLA DE CONTENIDO.

1. Presentación

El manual técnico presente para el desarrollo del aplicativo web diseñado para agilizar la citación de pacientes a las áreas críticas de atención médica. Este proyecto, concebido con el propósito de optimizar la gestión hospitalaria, representa un avance significativo en la mejora de la eficiencia y la atención al paciente.

I. Objetivo del Manual Técnico:

El presente manual tiene como finalidad proporcionar una guía detallada sobre la implementación y mantenimiento del aplicativo web destinado a la facilitación de la citación expedita de pacientes en las áreas de triage, consultorios médicos y área de procedimientos. Está dirigido especialmente a los profesionales y estudiantes que estarán a cargo de la administración y operación del sistema.

II. Estructura del Manual Técnico:

El manual está organizado en secciones que abordan aspectos críticos del sistema, asegurando una comprensión integral para su correcta implementación y funcionamiento. A continuación, se describen brevemente las secciones principales:

1. Requisitos de Hardware y Software:
En esta sección, detallamos los componentes hardware esenciales para garantizar el rendimiento eficiente del sistema. Asimismo, se presentan los requisitos de software necesarios para su correcta operación. Proporcionamos una guía detallada para asegurar la compatibilidad y óptimo desempeño del aplicativo.

2. Software y Herramientas Utilizadas:
Ofrecemos una visión general de las interfaces y programas empleados en el sistema. Este apartado proporciona una comprensión completa de las herramientas utilizadas **en el desarrollo del aplicativo web, permitiendo a los usuarios familiarizarse con el entorno de trabajo.**

3. Configuración de la Base de Datos y Diccionario de Datos:
Aquí, se brindan instrucciones detalladas para la importación y gestión de la base de datos del sistema, incluyendo la visualización de las tablas pertinentes. Este apartado garantiza la integridad y consistencia de la información, fundamentales para el correcto funcionamiento del aplicativo.

Este manual representa una herramienta indispensable para aquellos involucrados en el despliegue y mantenimiento del aplicativo web, asegurando una implementación eficiente y un rendimiento óptimo del sistema.

2. Requisitos de hardware y software.

Requisitos de Hardware para el Servidor:

- Procesador: Mínimo Intel Xeon o equivalente, con capacidad multinúcleo para manejar múltiples solicitudes concurrentes.
- Memoria RAM: Al menos 16 GB de RAM para garantizar un rendimiento eficiente del sistema.
- Almacenamiento: Espacio de almacenamiento SSD (Recomendable) o en su defecto HDD, con capacidad suficiente para el sistema operativo, la aplicación web y la base de datos. Se recomienda un mínimo de 256 GB para asegurar una respuesta rápida y fluida.
- Conectividad: Conexión a internet de alta velocidad para garantizar la accesibilidad remota y la carga rápida de datos.
- Tarjeta de Red: Tarjeta de red Gigabit para facilitar la transferencia rápida de datos entre el servidor y los usuarios.
- Sistema Operativo: Se recomienda un sistema operativo Linux, como Ubuntu Server, para optimizar la estabilidad y seguridad del servidor.
- Otros: Copias de seguridad automáticas, redundancia de energía y sistema de enfriamiento eficiente para mantener la integridad del servidor.

Requisitos de Hardware para los Equipos de Cómputo de los Usuarios:

- Procesador: Mínimo Intel Core i5 o equivalente, para un rendimiento fluido al utilizar la aplicación web.
- Memoria RAM: Se recomienda al menos 8 GB de RAM para garantizar una experiencia de usuario sin interrupciones.
- Almacenamiento: Un disco duro SSD mejora significativamente la velocidad de carga de la aplicación y la respuesta general del sistema.
- Tarjeta de Red: Tarjeta de red integrada o externa para una conexión estable a la red.

- Navegador Web: Última versión de navegadores web modernos como Google Chrome, Mozilla Firefox o Microsoft Edge, para una compatibilidad óptima con las tecnologías utilizadas (React, JavaScript y Tailwind CSS).

Requisitos de Hardware para el Televisor Smart TV:

- Tipo de TV: Smart TV con capacidad para conectarse a internet.

- Sistema Operativo: Sistema operativo compatible con navegadores web modernos. Ejemplos incluyen webOS para LG Smart TVs o Tizen para Samsung Smart TVs.

- Navegador Web: Navegador integrado compatible con React, JavaScript y Tailwind CSS. Se recomienda el uso de navegadores como Google Chrome o Mozilla Firefox.

- Conectividad: Conexión a internet estable mediante WiFi o conexión Ethernet.

- Resolución: Se recomienda una resolución HD o superior para una visualización nítida de la aplicación web.

- Control Remoto: Un control remoto intuitivo (Opcional) para facilitar la navegación a través de la aplicación web.

Requisitos de software:

- Windows 10, 11, server
- Mac OS
- Linux

Requisitos de software Tv (SO):

- webOS: Utilizado en televisores LG Smart TV. webOS es conocido por su soporte a navegadores web modernos, por lo que debería ser compatible con SpeechSynthesis.

- Tizen: Utilizado en televisores Samsung Smart TV. Tizen generalmente es compatible con las últimas tecnologías web, incluyendo JavaScript, y debería ser capaz de ejecutar la librería SpeechSynthesis.

- Android TV: Algunos televisores Smart TV utilizan el sistema operativo Android TV. En este caso, la compatibilidad dependerá de la versión de Android y del navegador web utilizado.

Herramientas y ambientes de desarrollo

Node.js:

Node.js es un entorno de ejecución de JavaScript del lado del servidor que permite la construcción de aplicaciones web escalables y de alto rendimiento. Basado en el motor V8 de Google Chrome, Node.js es conocido por su eficiencia y capacidad para manejar operaciones asincrónicas. Es esencial para nuestro aplicativo web, ya que facilita la ejecución del código del lado del servidor y la gestión de paquetes con npm. Node.js utiliza un modelo de ejecución asíncrona, lo que significa que puede manejar operaciones simultáneas sin bloquear el hilo principal de ejecución.

React:

React es una biblioteca de JavaScript para construir interfaces de usuario interactivas y reutilizables. Destaca por su enfoque en la creación de componentes modulares, lo que facilita el desarrollo, mantenimiento y actualización de la interfaz de usuario de nuestro aplicativo web. Su capacidad para actualizar eficientemente la interfaz en tiempo real mejora significativamente la experiencia del usuario.

TailwindCSS:

Tailwind CSS es un marco (framework) de desarrollo de interfaz de usuario (UI) para la construcción de sitios web. A diferencia de otros frameworks CSS como Bootstrap o Foundation, Tailwind CSS no proporciona componentes prediseñados, sino que se centra en proporcionar utilidades de bajo nivel que permiten construir estilos personalizados de manera más eficiente.

Xampp:

Xampp proporciona un entorno de desarrollo local completo que incluye Apache, MySQL, PHP y Perl. Es crucial para nuestro proceso de desarrollo, ya que nos permite simular un entorno de servidor real en nuestros equipos. Xampp simplifica la configuración de servidores y bases de datos locales, acelerando el desarrollo y las pruebas de nuestro aplicativo web.

Visual Studio Code:

Visual Studio Code (VSCode) es un entorno de desarrollo integrado (IDE) altamente configurable y liviano. Su compatibilidad con una amplia gama de lenguajes de programación, extensiones y potentes herramientas de depuración hacen que sea la elección perfecta para el desarrollo de nuestro aplicativo web. VSCode mejora la productividad del desarrollador con características como autocompletado, sugerencias de código y control de versiones integrado.

Postman:

Postman es una herramienta de colaboración y desarrollo de API que simplifica la creación, prueba y documentación de servicios web. Su interfaz intuitiva permite realizar solicitudes HTTP, probar endpoints y automatizar flujos de trabajo. Postman es esencial para verificar la funcionalidad correcta de nuestros servicios web y garantizar una comunicación eficiente entre el frontend y el backend de nuestro aplicativo.

Integrar estas herramientas y entornos de desarrollo dentro del proyecto, asegura un desarrollo eficiente, un rendimiento óptimo y una gestión efectiva de los componentes esenciales de nuestro aplicativo web.

3. Instalación, configuración de dependencias y base de datos

La configuración del entorno de servidor local es un paso fundamental para el desarrollo y prueba eficientes de aplicaciones antes de su implementación en línea. Para instalar y configurar este entorno utilizando XAMPP, primero, accede al sitio web oficial de XAMPP y selecciona la versión correspondiente a tu sistema operativo: ya sea Windows, macOS o Linux.

Xampp: https://www.apachefriends.org/es/index.html

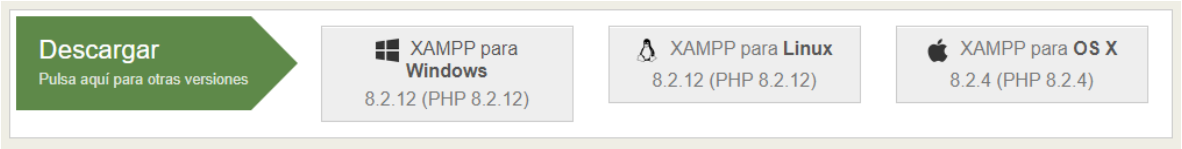

Figura 1 pagina xampp - Autoría propia

Una vez descargado, el proceso de configuración es intuitivo y accesible. XAMPP integra de manera fluida Apache como servidor web, MySQL como sistema de gestión de bases de datos, y PHP, junto con otros componentes esenciales. Esta integración simplifica significativamente el proceso de establecer un entorno de servidor local.

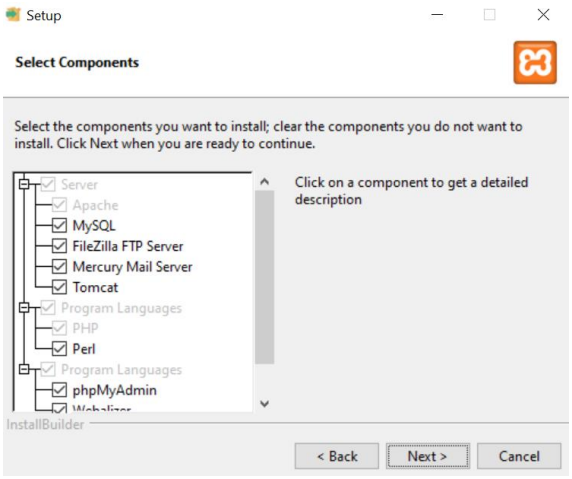

Figura 2 Instalacion xampp - Autoría propia

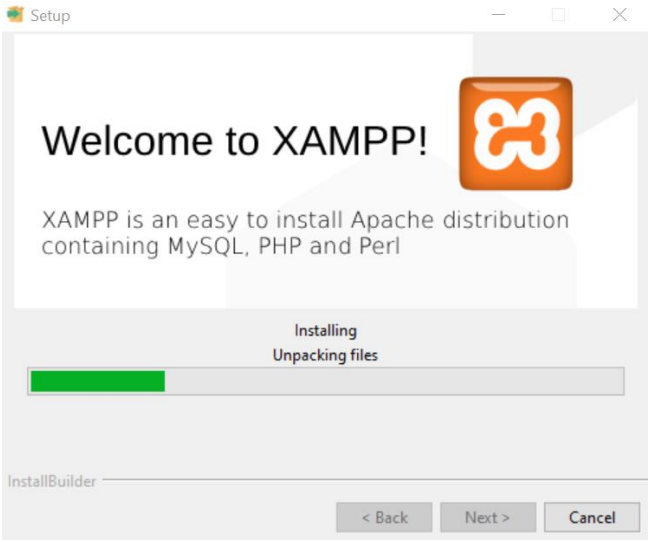

Figura 3 Instalacion xampp - Autoría propia

Una vez instalado correctamente el Xampp Control panel, podemos proceder a abrir el mismo, y encontraremos varios módulos, de los cuales solo usaremos Apache y MySql. Debemos dar click en los botones Start del apache y el MySql, se comenzara a ejecutar estos módulos que seleccionamos.

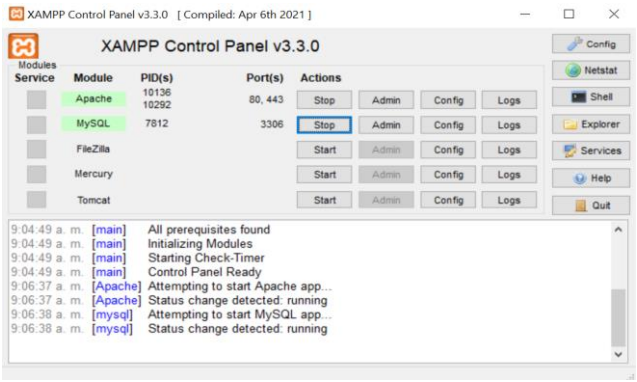

Figura 4 Xampp controlPanel - Autoría propia

Mientras se ejecuta estos módulos, debemos estar muy atento a la consola en caso de que ocurra algún error. Uno de los errores frecuentes al ejecutar MySql y apache es la configuración de puertos, para esto solo se debe acceder archivo de configuración del modulo que está generando conflicto. En este caso se realizará la explicación con apache.

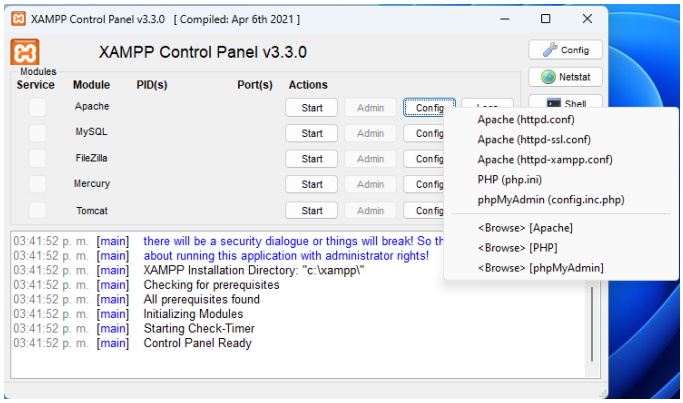

Figura 5 Boton config - Autoría propia

Una vez se pueda visualizar la ventana de config del apache, debemos dirigirnos a: “Apache (httpd.conf)”. Se abrirá un documento de texto y debemos buscar la siguiente sección para cambiar al puerto deseado en la siguiente línea, Ej: Listen 8088.

```
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 8088
```

Figura 6 Archivo configuración xampp - Autoría propia

Para abrir phpMyAdmin, donde encontraremos la vista administrar y crear bases de datos, deben estar los dos módulos ejecutándose (Apache y MySQL) y le damos clic a “Admin”

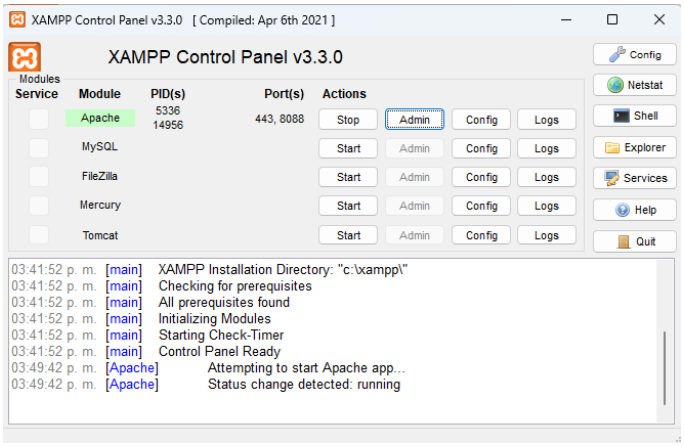

Figura 7 xampp botón Admin - Autoría propia

Nos aparecerá una página igual a la de la foto y nos dirigimos a phpMyAdmin.

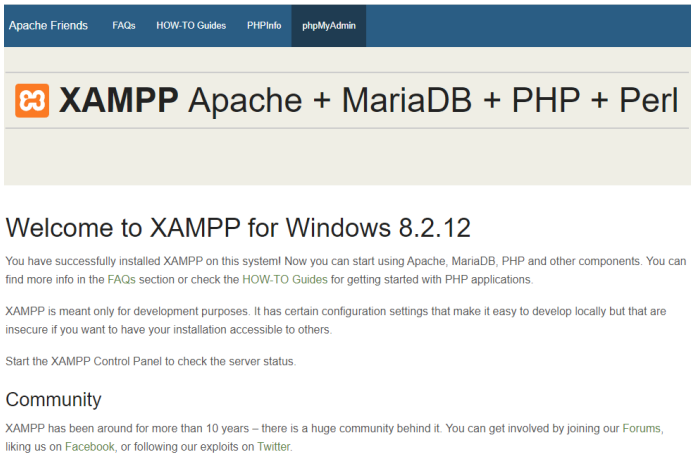

Figura 8 Dashboard Apache - Autoría propia

En este apartado podemos ver todas las bases de datos que tengamos en este momento, junto con las bases de datos por defecto del xampp.

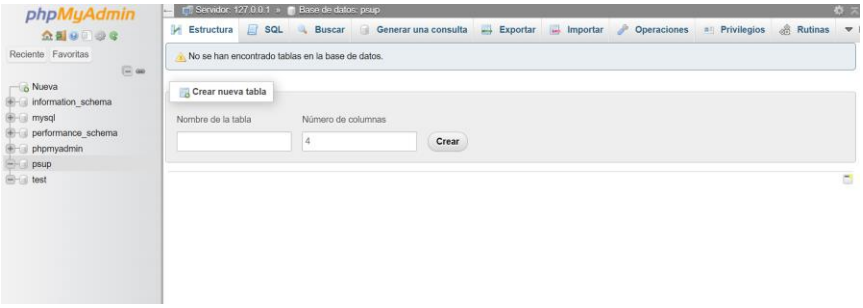

Figura 9 Admin PhpMyAdmin - Autoría propia

Para montar la base de datos de “llamadopacientes.sql” debemos crear primero una base de datos, de la siguiente forma. Nos dirigimos a “+Nueva” y Colocamos el nombre de la base de datos (Debe ser la misma que la del archivo)

Figura 10 Crear Base de datos nueva - Autoría propia

Una vez creada la base de datos debemos importar el archivo. Luego de importarlo damos click en el botón importar.

Figura 11 Importar base de datos - Autoría propia

Una vez importada, debe mostrar la lista de tablas de la base de datos. Luego, se podrá conectarla y configurarla con el proyecto en el archivo poolDb.js ubicado en el backend del proyecto.

Además de XAMPP, es esencial instalar otras dependencias cruciales para el desarrollo de nuestro aplicativo web. Node.js y React son necesarios para el desarrollo del lado del cliente, mientras que TailwindCSS mejora la eficiencia en el diseño y estilo de la interfaz de usuario.

Instalación de NodeJS.

Debemos descargar el instalador de NodeJS de la página oficial.

https://nodejs.org/en

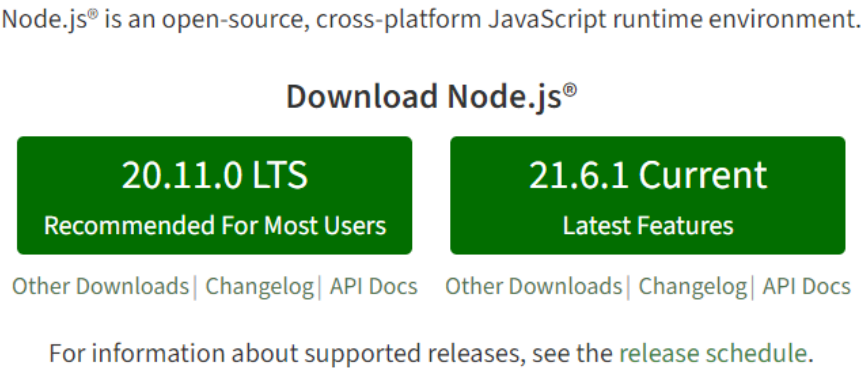

Figura 12 Pagina nodeJS- Autoría propia

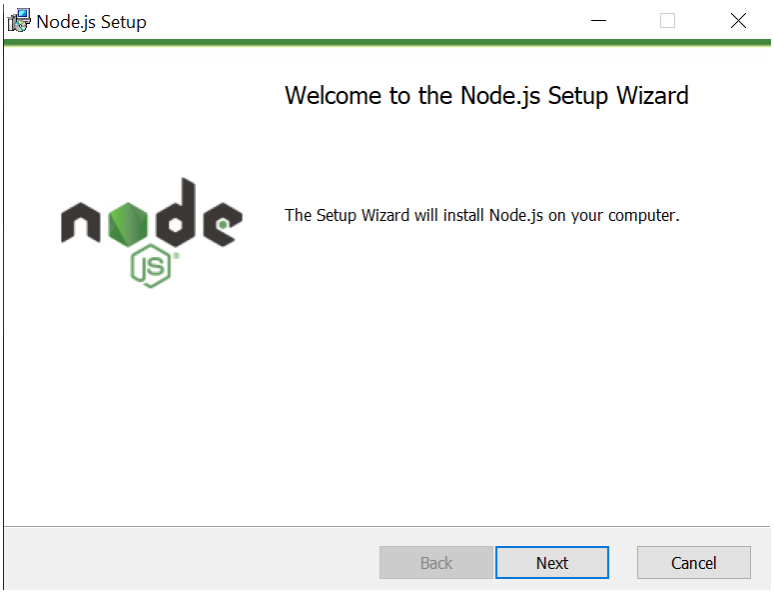

Figura 13 Instalador nodejs - Autoría propia

Figura 14 Instalador nodejs - Autoría propia

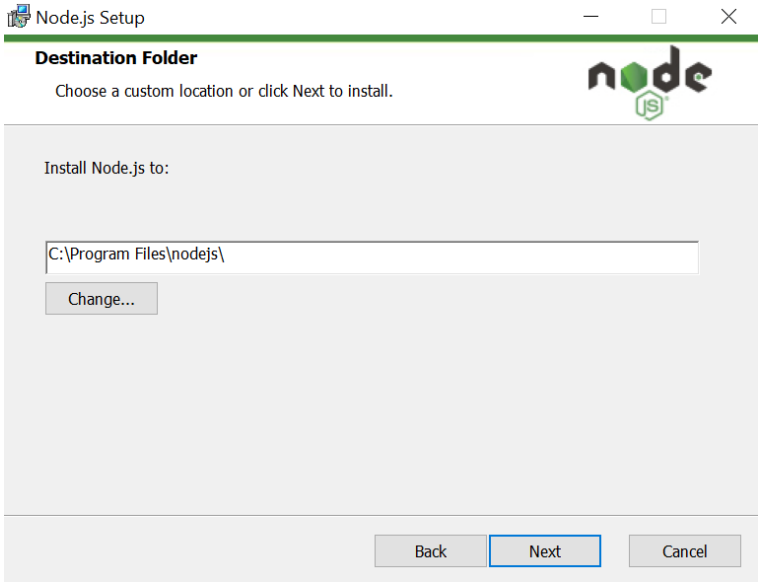

Figura 15 Instalador nodejs - Autoría propia

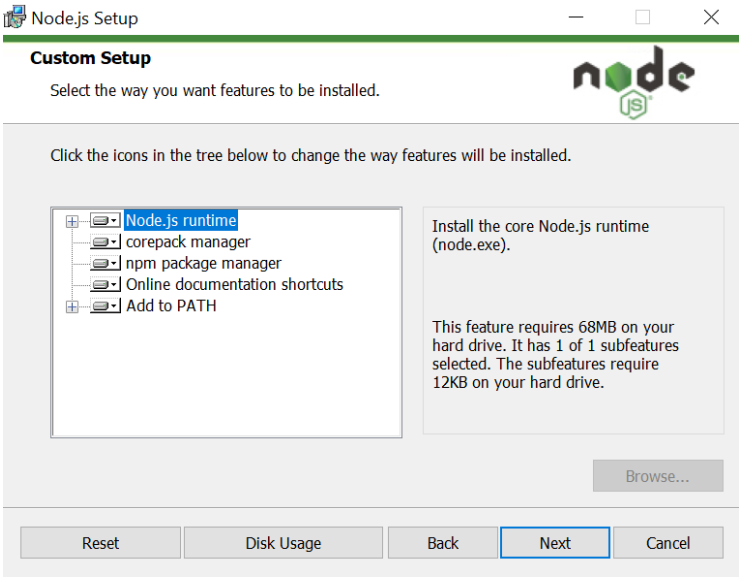

Figura 16 Instalador nodejs - Autoría propia

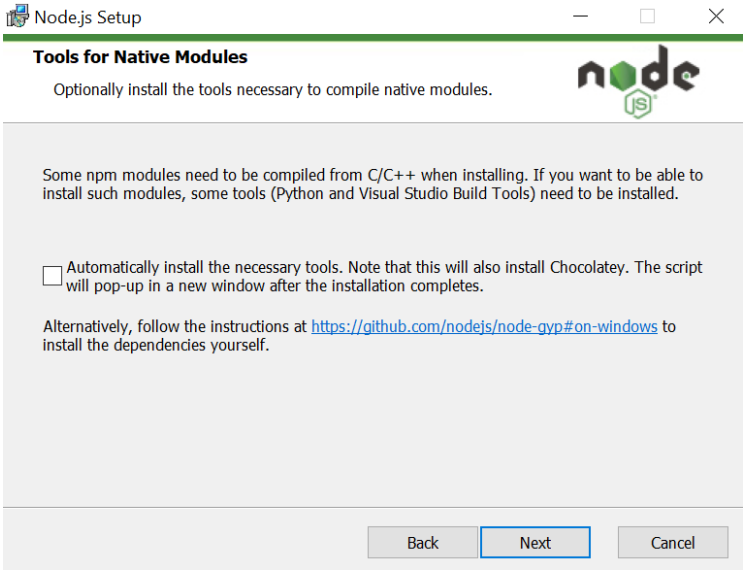

Figura 17 Instalador nodejs - Autoría propia

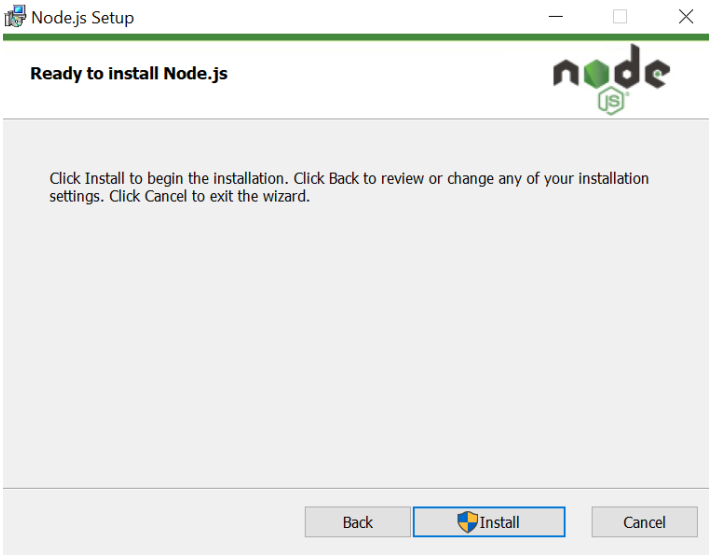

Figura 18 Instalador nodejs - Autoría propia

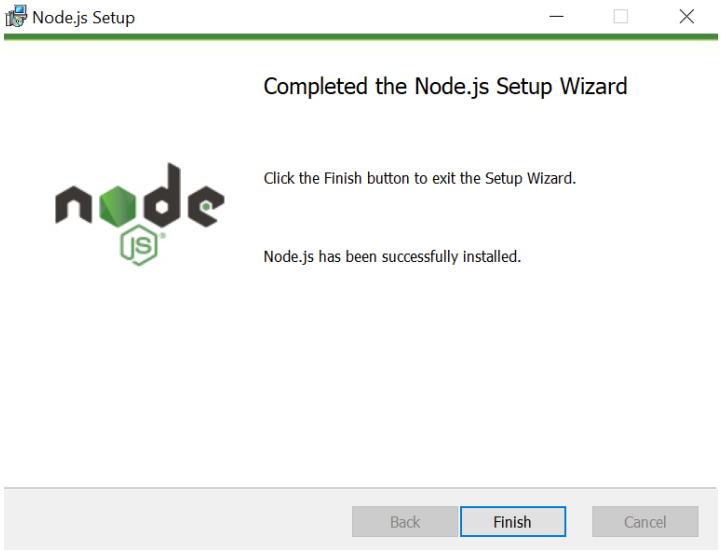

Figura 19 Instalador nodejs - Autoría propia

Instalación de Visual studio code, React y tailwindCss.

Debemos descargar el instalador del Visual studio code de la pagina oficial.

https://code.visualstudio.com/

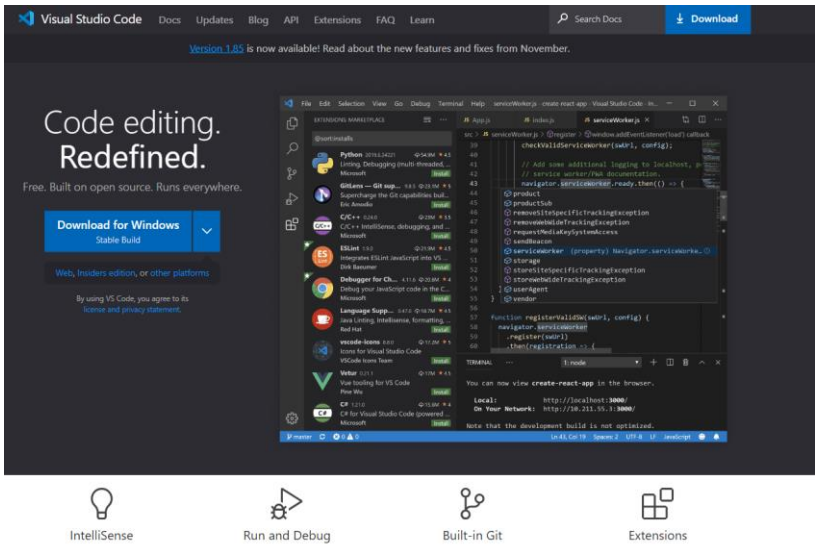

Figura 20 Pagina visual studio - Autoría propia

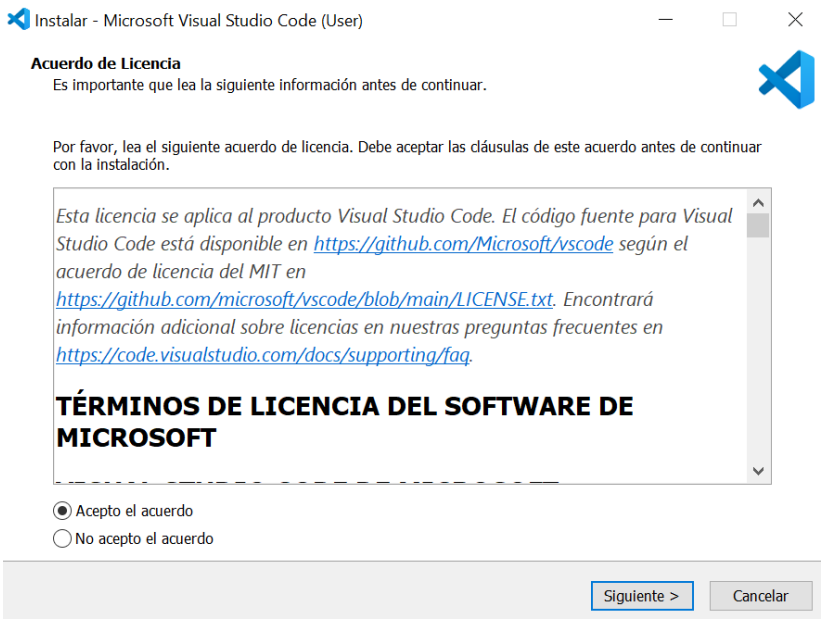

Figura 21 Instalar visual studio - Autoría propia

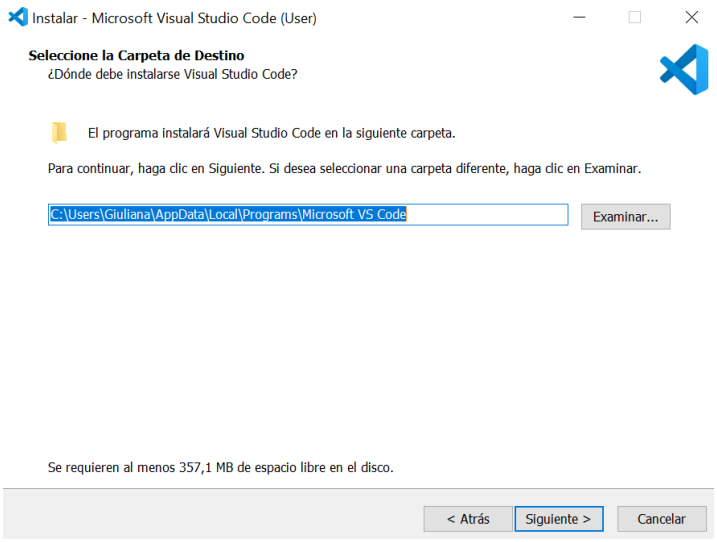

Figura 22 Instalar visual studio - Autoría propia

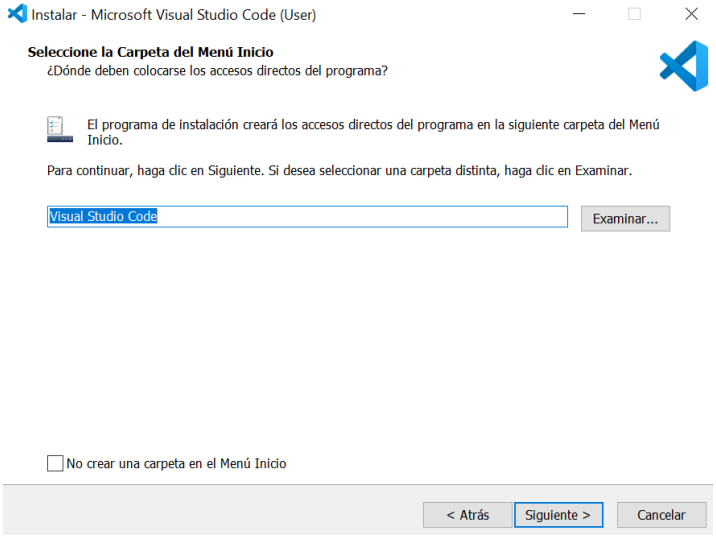

Figura 23 Instalar visual studio - Autoría propia

En esta parte de la instalación, podemos elegir las tareas adicionales que queremos que tenga el visual studio code. Una vez elegidas, damos click en siguiente.

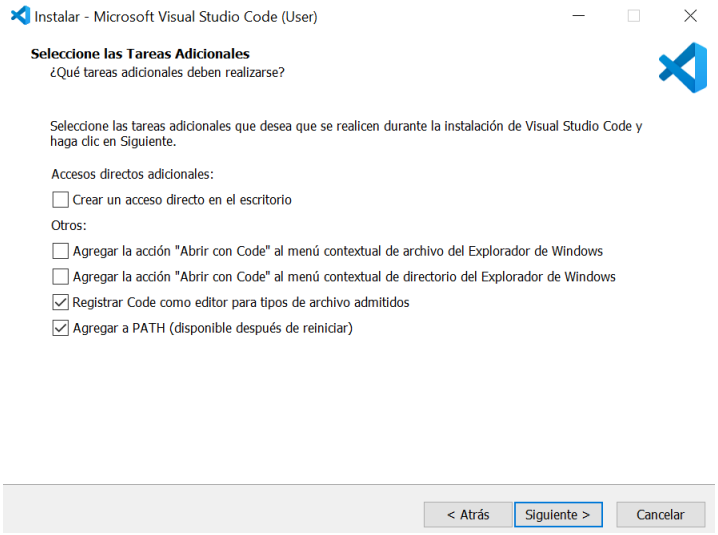

Figura 24 Instalar visual studio - Autoría propia

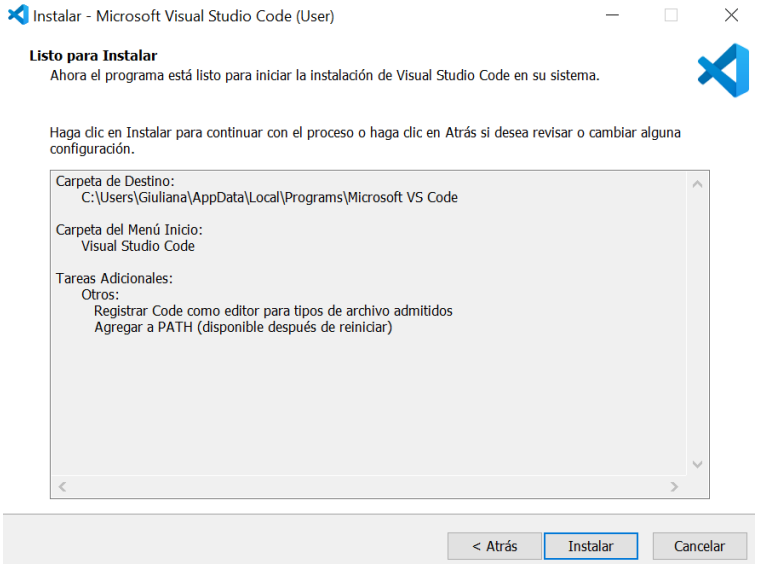

Figura 25 Instalar visual studio - Autoría propia

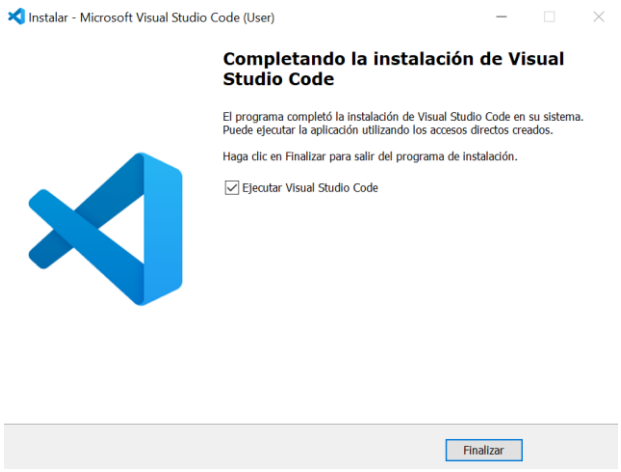

Figura 26 Instalar visual studio - Autoría propia

Una vez instalado el Visual studio code, debemos abrir el programa, nos dirigimos a file y open folder. Posterior a esto, debemos abrir una nueva terminal dentro del proyecto para la correcta instalación de los paquetes.

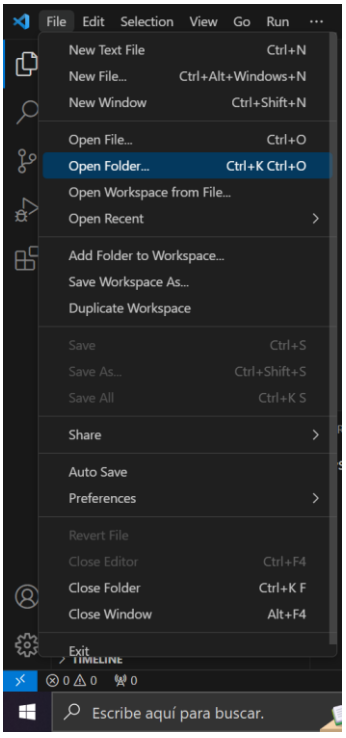

Figura 27 File visual studio - Autoría propia

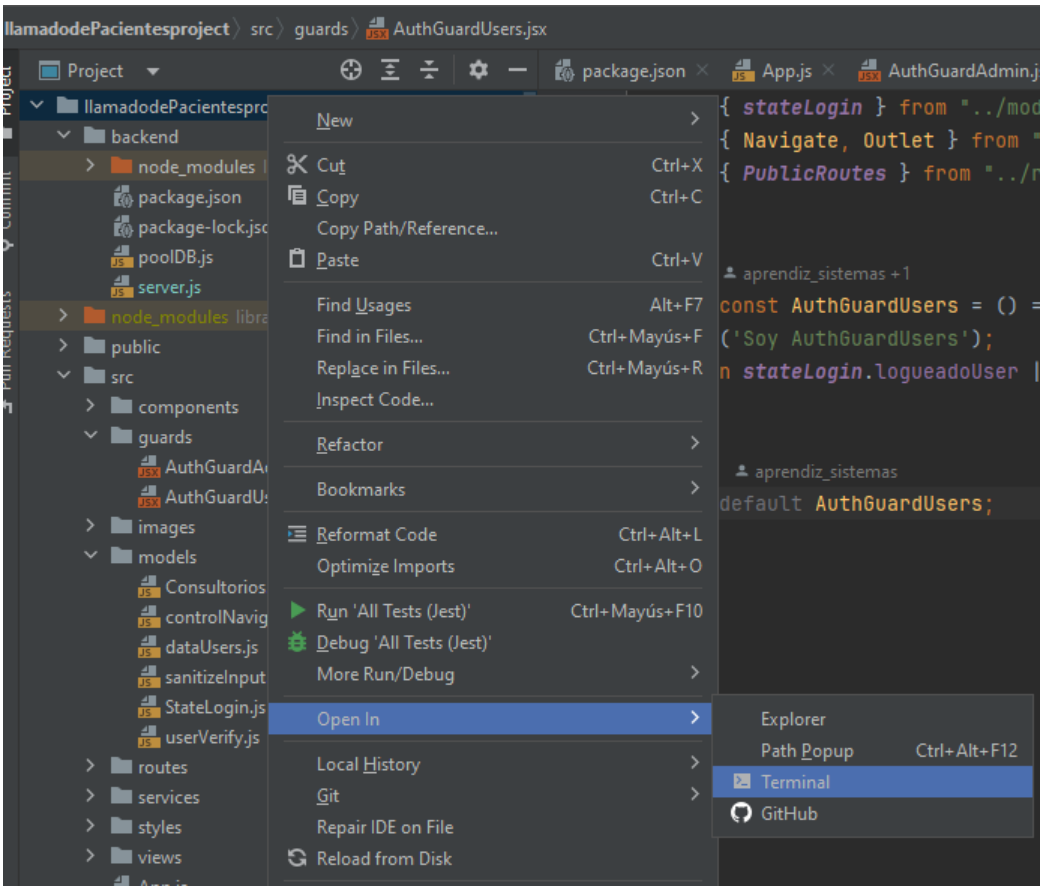

Figura 28 openIn terminal visual studio - Autoría propia

Debemos abrir dos terminales, la primera consola es en la carpeta raíz del proyecto, la segunda consola es en la carpeta backend. ejecutamos en ambas consolas el comando “npm install” y esperamos que las dependencias se instalen.

Esto instalará todas las dependencias que necesita tu proyecto, incluyendo React, Tailwind CSS y otras dependencias declaradas en tu archivo package.json (Comando primordial).

```
npm install
```

Figura 29 Comando npm install - Autoría propia

Esto instalará las bibliotecas fundamentales de React y ReactDOM dentro del proyecto (se debe ejecutar solo en la carpeta raíz).

```
npm install react react-dom
```

Figura 30 Comando instalar react - Autoría propia

Instala Tailwind CSS y sus dependencias utilizando npm (se debe ejecutar solo en la carpeta raíz).

```
npm install tailwindcss postcss autoprefixer
```

Figura 31 Comando instalar Tailwind - Autoría propia

Este entorno de servidor local no solo permite un desarrollo eficiente, sino que también facilita la prueba exhaustiva de proyectos antes de su despliegue en línea. Al configurar este entorno de manera adecuada, aseguramos un flujo de trabajo fluido y la capacidad de abordar cualquier problema potencial antes de que afecte a los usuarios finales.

4. Contraseñas de acceso

Este módulo del Manual Técnico aborda de manera integral la gestión de contraseñas destinadas a dos perfiles cruciales en el sistema: los usuarios asistenciales y los administradores. La seguridad de las contraseñas es esencial para garantizar la integridad, confidencialidad y disponibilidad de la información sensible manejada por la aplicación.

	User Name	**Password**
Medicos Urgencias	Medicos_urgencias	Urgenci4asMedCorp4s23
Medicos Ginecología	Ginecologia_urgencias	Urgenci4asGinecoCorp4s23
Enfermeria Procedimientos	Procedimientos_urgencias	Urgenci4asProCorp4s23
Enfermeria Triage Urgencias	Triage_urgencias	Urgenci4asTriaCorp4s23
Administradores Departamento de sistemas	Admin_sist3m4s	S1st3masCorp4s23

Tabla 1 Usuarios y contraseñas de acceso - Autoría propia

Estas contraseñas se encuentran almacenadas y establecidas dentro del proyecto, en la carpeta models, dataUsers.js. Por lo tanto, se pueden configurar directamente desde este archivo Javascript, se guardan los respectivos cambios en dado caso y el aplicativo tomara los cambios inmediatamente.

Consideraciones Adicionales:

Se resalta la importancia de fomentar la conciencia de seguridad entre los usuarios, proporcionando orientación sobre buenas prácticas de gestión de contraseñas y la necesidad de mantener la confidencialidad de la información de inicio de sesión.

5. Diccionario de datos

Tabla cache_pacientes

Columna	Tipo	Nulo	Predeterminado	Comentarios
id *(Primaria)*	int(11)	No		
nombre_paciente	varchar(255)	Sí	*NULL*	
cedula	varchar(255)	Sí	*NULL*	
consultorio_llamado	varchar(255)	Sí	*NULL*	
nivel_triage	varchar(255)	Sí	*NULL*	

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	0	A	No	

Figura 32 Tabla cache_pacientes - Autoría propia

En esta tabla llamada “cache_pacientes”, encontraremos los pacientes que han sido llamados a las áreas de urgencias como: Triage, Medicos Urgencias y Procedimientos. Esta tabla se creo a la necesidad de llevar un registro de los pacientes que han sido llamados a las distintas áreas de urgencias, y encontraremos datos básicos del paciente, como: Nombre, Cedula, Consultorio al que ha sido llamado y el nivel de triage actual.

En algunos casos, el paciente que ha sido llamado no tiene nivel de triage, se debe a que el paciente apenas fue llamado a triage por lo tanto, no tendrá el dato de nivel de triage. Es por esta razón también que es un campo el cual no es obligatorio.

Tabla consultorios

Columna	Tipo	Nulo	Predeterminado	Comentarios
id *(Primaria)*	int(11)	No		
nombre_consultorio	varchar(255)	Sí	*NULL*	
id_servicio	int(11)	Sí	*NULL*	

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	13	A	No	
id_servicio	BTREE	No	No	id_servicio	13	A	Sí	

Figura 33 Tabla consultorios - Autoría propia

En la tabla de “consultorios” encontraremos los distintos consultorios del aplicativo de llamados. Estos consultorios son los que se podrán visualizar en el aplicativo, en los componentes de Hub. Los consultorios de esta tabla tienen un id único el cual hace referencia al tipo de servicio que pertenece, esto con el objetivo de que cada consultorio pertenezca al correspondiente servicio. Por otra parte, cuando se modifica o se agrega un consultorio esta consulta o modificación a la tabla se dirige a la tabla de consultorios.

Tabla Servicio

Columna	Tipo	Nulo	Predeterminado	Comentarios
id *(Primaria)*	int(11)	No		
nombre_servicio	varchar(255)	Sí	*NULL*	

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	4	A	No	

Figura 34 Tabla Servicio - Autoría propia

En la tabla de servicio encontraremos los distintos servicios del área de urgencias, como lo son: procedimientos, triage, médicos urgencias y ginecología.

6. Diagramas

Figura 35 Diagrama base de datos - Autoría propia

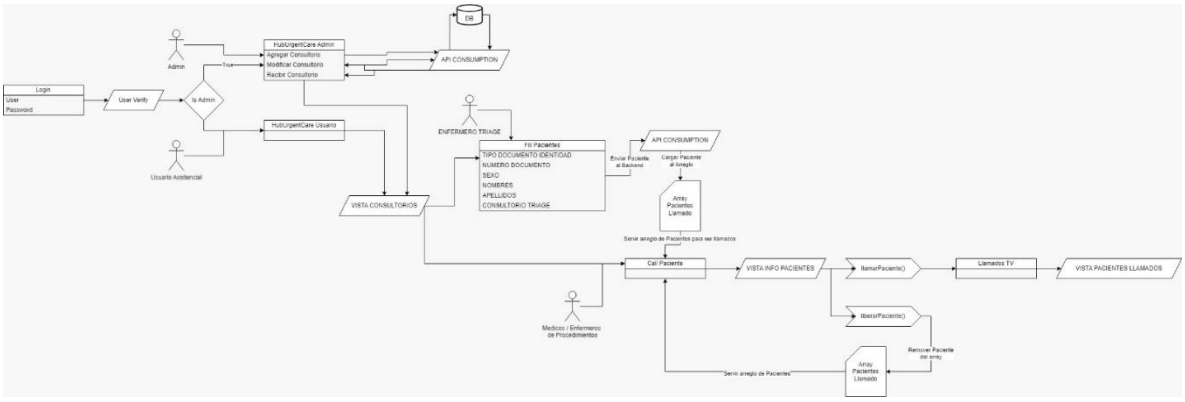

Figura 36 Diagrama uml y casos de uso - Autoría propia

7. Documentación aplicativo web

Estructura de directorios:

El proyecto "llamadodePacientes" tiene una estructura organizada en carpetas que representan diferentes partes del desarrollo, facilitando la modularidad y mantenimiento del código. Aquí está la representación de las carpetas y archivos:

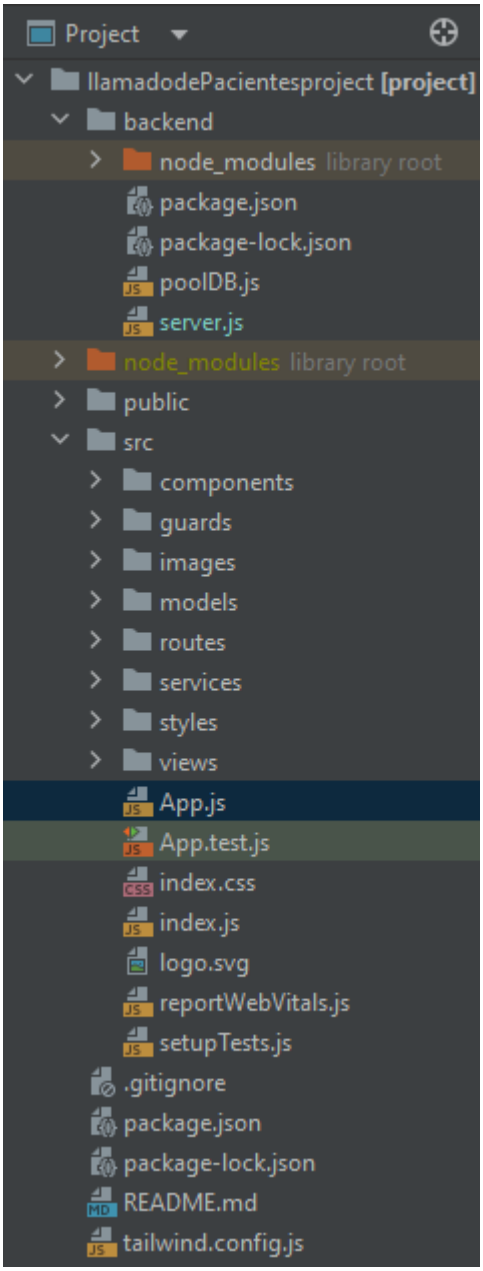

Figura 37 Estructura de directorios proyecto - Autoría propia

Carpeta Backend: Contiene archivos relacionados con el servidor y la lógica del lado del servidor. server.js contiene la lógica principal del servidor, poolDb.js esta relacionado con la conexión a la base de datos y package.json es el archivo de configuración de Node.js.

public:

Es la carpeta que contiene recursos estáticos accesibles públicamente, como imágenes, hojas de estilo y otros archivos netamente de react (Durante el desarrollo del proyecto no se almaceno nada dentro de esta carpeta, funcionando exclusivamente para librerías de react).

src:

Es la carpeta principal del código fuente de la aplicación.

Components:

Aquí se encuentran los componentes React utilizados en la aplicación. Cada componente tiene su propio archivo para facilitar el mantenimiento y la reutilización.

guards:

Contiene los componentes encargados de la autenticación y autorización de rutas.

images:

Almacena imágenes y recursos visuales utilizados en la interfaz de usuario.

models:

Contiene archivos relacionados con la estructura de datos y la lógica del modelo. Por ejemplo, Consultorios.js podría definir la estructura de los consultorios.

routes:

Contiene archivos relacionados con la configuración y definición de rutas de la aplicación.

services:

Puede contener archivos relacionados con la comunicación con servicios externos o API.

styles:

Contiene archivos de hojas de estilo, como styles.css, que definen el diseño y la apariencia de la aplicación.

views:

Contiene archivos que representan las vistas principales de la aplicación. Cada vista está asociada a un componente React.

index.js:

Punto de entrada principal de la aplicación React, donde se realiza la renderización del componente principal, como App.js.

Explicación detallada de componentes:

App Component(App.js).

Descripción:

El componente App es el componente principal que gestiona las rutas y la navegación de la aplicación. Utiliza react-router-dom para manejar las rutas y renderizar los componentes correspondientes.

Funcionalidades Principales:

Redirige automáticamente a la ruta principal al cargar la aplicación.

Define rutas públicas y privadas para administradores y usuarios.

AuthGuardAdmin Component

Descripción:

El componente AuthGuardAdmin es una guardia de autenticación que permite el acceso solo a usuarios autenticados como administradores.

Funcionalidades Principales:

- Redirige a la página de inicio de sesión si el usuario no está autenticado como administrador.
- Habilita rutas de admin, si las credenciales ingresadas de administrador son correctas.

AuthGuardUsers Component

Descripción:

El componente AuthGuardUsers es una guardia de autenticación que permite el acceso solo a usuarios autenticados como administradores o usuarios regulares.

Funcionalidades Principales:

- Redirige a la página de inicio de sesión si el usuario no está autenticado como administrador o usuario regular.
- Habilita rutas de Usuario asistencial, si las credenciales ingresadas del usuario son correctas.

LoginView Component

Descripción

El componente LoginView es la vista de inicio de sesión que permite a los usuarios ingresar sus credenciales.

Funcionalidades Principales

- Verifica las credenciales de administrador y usuario.
- Redirige a las páginas correspondientes después de iniciar sesión (HubUrgent Admin o HubUrgent).

Routes Configuration (Routes.jsx)

Descripción

El archivo Routes.js contiene la configuración de las rutas públicas y privadas para la aplicación.

Funcionalidades Principales

- Define rutas públicas como inicio de sesión y llamadas de TV.
- Define rutas privadas para administradores y usuarios.

Configuración de Base de Datos (PoolDb.js)

Conexión a la Base de Datos:

El archivo PoolDb.js contiene la configuración y funciones para interactuar con la base de datos MySQL. Asegúrese de haber configurado correctamente los siguientes parámetros en la conexión.

Funciones de Interacción con la Base de Datos:

- Obtener Todos los Consultorios

La función getAllConsultorios realiza una consulta SQL para obtener todos los consultorios y sus servicios asociados.

- Agregar Consultorio

La función addConsultorio permite agregar un nuevo consultorio, verificando la existencia del servicio asociado.

- Modificar Consultorio

La función modifyConsultorio realiza la modificación del nombre de un consultorio en la base de datos.

- Agregar Paciente a la Cache

Las funciones cachePaciente y cachePacienteTriage agregan nuevos pacientes a la tabla cache_pacientes considerando si son de triage o no.

Configuración del Servidor y API (Server.js)

- Configuración del Servidor y Socket.IO

Asegúrese de que el servidor esté configurado correctamente en el archivo Server.js. Se utiliza Socket.IO para gestionar eventos en tiempo real.

- Rutas de la API

Las rutas de la API se definen para obtener consultorios, agregar nuevos consultorios, modificar consultorios y agregar pacientes a la cache.

- Puerto de Escucha

El servidor está configurado para escuchar en el puerto 3001 por defecto. Puede cambiar el puerto según sus necesidades.

Consumo de la API desde el Cliente (apiConsumption.js)

Funciones de Consumo de API:

- fetchData: Obtiene datos de consultorios mediante una solicitud GET.
- addConsultorio: Agrega un nuevo consultorio mediante una solicitud POST.
- modifyConsultorio: Modifica un consultorio mediante una solicitud PUT.
- cachePacientes: Agrega pacientes a la cache mediante una solicitud POST.

Configuración de URL de la API

Asegúrese de que las funciones de consumo de la API tengan la URL correcta (http://localhost:3001 o en su defecto la dirección ip del servidor) en el archivo apiConsumption.js.

8. Despliegue

- Node.js

Node.js es un entorno de ejecución para JavaScript en el lado del servidor. Permite ejecutar código JavaScript fuera del navegador, lo que posibilita la creación de aplicaciones web completas, desde el frontend hasta el backend, utilizando el mismo lenguaje de programación. Node.js utiliza un modelo de operaciones no bloqueantes y orientadas a eventos, lo que lo hace eficiente y escalable.

- React-Scripts Start

react-scripts es un conjunto de scripts y configuraciones predeterminados para proyectos creados con Create React App. El comando npm run start ejecuta react-scripts start para iniciar el servidor de React.

Características:

Inicia un servidor para el frontend.

Ofrece recarga automática en el navegador al detectar cambios en el código.

Muestra mensajes de advertencia y errores en la consola.

- Despliegue del Backend

Para desplegar el backend, se utiliza el comando npm run start, que ejecuta el archivo server.js mediante Node.js. En modo desarrollo, se puede utilizar npm run dev para iniciar el servidor con nodemon, una herramienta que reinicia automáticamente la aplicación cuando se detectan cambios en los archivos.

- Despliegue del Frontend

Para desplegar el frontend, se utiliza el comando npm run start proporcionado por react-scripts. Este comando inicia la aplicación React y la hace accesible a través del navegador en el puerto especificado (generalmente el puerto 3000).

- Librería Concurrently

La librería concurrently permite ejecutar múltiples comandos de manera simultánea en un solo script. En este caso, se utiliza para iniciar ambos servidores (backend y frontend) al mismo tiempo.

- Despliegue con npm run deployment (Producción)

El script npm run deployment utiliza concurrently para iniciar tanto el servidor backend como el frontend al mismo tiempo. Esto es útil para simplificar el proceso de despliegue en **producción**, asegurando que ambas partes de la aplicación estén activas simultáneamente. Asegúrese de que las rutas y configuraciones sean correctas antes de ejecutar este comando, Por ejemplo: Debe asegurarse de que no haya ninguna dirección por “localhost” sino por la ip del servidor, es decir: http://10.27.1.23:3000 (puerto 3000 para dirigir las solicitudes al frontend) y http://10.27.1.23:3001 (puerto 3001 para dirigir las solicitudes al backend), esto en caso de que el aplicativo sea desplegado en el servidor 10.27.1.23.