UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

PIX.

Desarrollo de un sistema de monitoreo de usuarios y productos.

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniera en Computación

P R E S E N T A

Jennifer Estefanny Rivera Esquivel

ASESORA DE INFORME

Ing. Guadalupe Lizeth Parrales Romay

Ciudad Universitaria, Cd. Mx., 2023

Índice

Objetivo del trabajo

Mostrar las evidencias de mi participación en el proyecto y cómo los conocimientos adquiridos a lo largo de mi formación me permitieron contribuir en su desarrollo. Así como el proceso para crearlo, es decir, dificultades encontradas, metodología empleada y las soluciones que se dieron para llegar a los resultados esperados.

Objetivo de PIX

PIX busca ser un software que permita interactuar de manera amigable y eficiente con las grabaciones que llegan de la red de líneas telefónicas y de radios de una empresa, llamadas productos. Así, mediante estos archivos se pueda tener pleno conocimiento del personal y sus actividades.

El sistema contará con dos roles. La persona que ejecuta una función como usuario operador podrá revisar las grabaciones que provienen de las distintas fuentes; por otro lado, los que poseen el rol de administrador podrán visualizar todos los usuarios y sus accesos al sistema.

Por medio de la información recabada a través de las fuentes externas se permitirá tener conocimiento del motivo y el asunto de todas las llamadas realizadas por el personal.

Objetivos específicos de la aplicación

- Reproducción de productos en formato MP3.
- Visualización en tiempo real.
- Reproductor de audio con analizador de espectro.
- Ordenamiento de información por columnas.
- Reportes por correo electrónico.
- Notas a las grabaciones.
- Acceso a través de teléfonos y tabletas.
- Control de acceso.
- Búsqueda por fecha, hora, id de llamada, fuente de entrada, número de teléfono, duración, dirección o cualquier campo que el usuario desee agregar.
- Exportar a PDF la información visualizada en las tablas.
- Visualización en línea o tiempo real de estados y llamadas.

Marco teórico

Participé en un proyecto de una empresa nacional de innovación tecnológica en diversas ramas, principalmente en software, fundada en el 2011, especializada en telecomunicaciones y redes de misión crítica para la defensa, seguridad y protección del espacio aéreo, terrestre y marítimo.

A continuación, daré los conceptos básicos y la definición de algunas herramientas usadas a lo largo del desarrollo del proyecto; éstos son:

Algoritmos de ordenamiento

Un algoritmo de ordenamiento es aquel que reacomoda los elementos de una lista en una secuencia determinada por cierto orden, ya sea de menor a mayor o viceversa. Es decir, el resultado será una lista ordenada de la forma en que se lo solicitamos. Existen muchos tipos de algoritmos, su uso depende de las necesidades del consumidor, cada uno de ellos cuenta con ventajas y desventajas sobre los demás, características específicas y tiempos de ordenamiento diferentes.

API

Este término es una abreviatura de Application Programing Interfaces (interfaz de programación de aplicaciones). Es un conjunto de protocolos y definiciones que se utilizan para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre los diferentes componentes de software o aplicaciones que se usan, buscando automatizar procedimientos, desarrollar nuevas funcionalidades y reutilizar código. Para el intercambio de datos se suelen utilizar formatos predefinidos como YAML o JSON para el caso deaplicaciones web.

Arquitectura REST

La arquitectura REST *o* Representational State Transfer (Transferencia de Estado Representacional), fue definida en el año 2000, es una arquitectura de desarrollo web que puede ser utilizada por un cliente HTTP, e implementada con prácticamente cualquier lenguaje de programación. La arquitectura *REST* se basa en que el cliente envía peticiones para recuperar o modificar recursos y el servidor responde con el resultado, que puede ser el estado de la petición que se hizo o los datos solicitados.

Una petición está formada por:

- Un verbo HTTP que define la operación a realizar.
- Una cabecera o *header* que incluye información sobre la petición.
- Una ruta o *path* hacia un recurso.

- El cuerpo del mensaje o *body* con los datos de la petición.

La Ilustración 1 es la representación de la arquitectura *REST*.

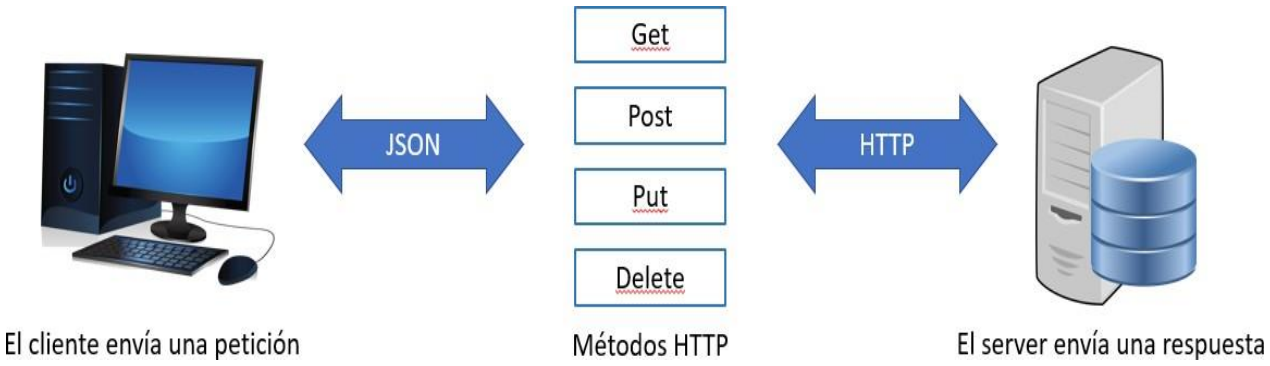


Ilustración 1. Peticiones de la arquitectura REST.

Arquitectura cliente servidor

Es un modelo usado en muchos protocolos y servicios de internet, consiste en un cliente o consumidor que solicita algún tipo de servicio al servidor o proveedor, el cual se encarga de darle respuesta a la petición. La principal característica de este modelo es que permite conectar a múltiples clientes simultáneos al conjunto de servicios y recursos que provee un servidor, sin importar que los consumidores se ubiquen en diferentes locaciones.

En la Ilustración 2 se puede ver una ejemplificación de dicho modelo.

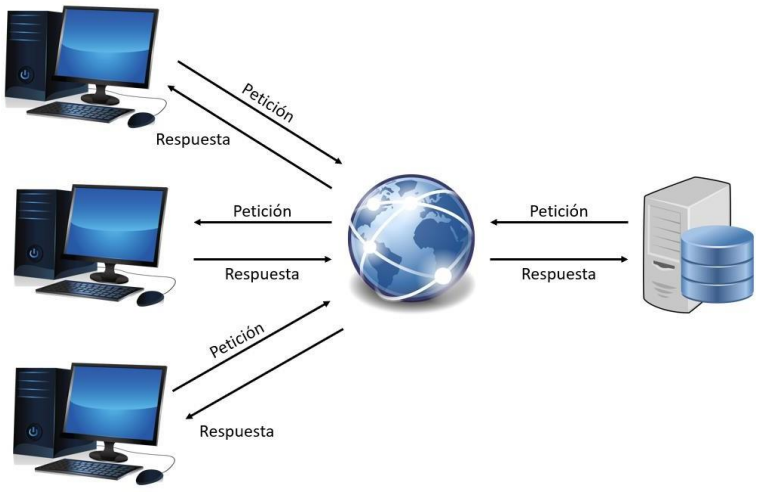


Ilustración 2. Representación de la arquitectura cliente servidor.

CRUD de bases de datos

CRUD es un acrónimo que hace referencia a las iniciales, por sus nombres en inglés, de las cuatro operaciones principales de los sistemas de almacenamiento y gestión de bases de datos, cuyo objetivo es el de almacenar, organizar y clasificar la información, a continuación, se describen las funciones:

- Create (Crear registros).
- Read (Leer registros).
- Update (Actualizar registros).
- Delete (Borrar registro).

Material Design

Es una guía en la que nos podemos basar para crear un diseño visual e interactivo para cualquier plataforma, fue desarrollado por Google en el 2014 con el fin de ser capaz de adaptarse a múltiples dispositivos. La idea de Google es la de incorporar este sistema de forma progresiva a todos sus productos, incluyendo las aplicaciones web y móviles, con la finalidad de crear una experiencia similar en todas sus plataformas.

Modelo de desarrollo de software

Son una colección de técnicas y formas para organizarse al momento de crear un software y llevarlo a cabo con grandes posibilidades de éxito. Esta sistematización describe el ciclo de vida de un proyecto, así como herramientas que ayudarán en el proceso, esto incluye dividir el proyecto en módulos más pequeños.

Las etapas de desarrollo de software son las siguientes:

- **Planificación**: Antes de empezar un proyecto de desarrollo de un sistema, es necesario hacer ciertas tareas que influirán decisivamente en el éxito de éste. Algunas de las tareas de esta fase incluyen actividades como la determinación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados, la estimación del costo del proyecto, su planificación temporal y la asignación de recursos a las diferentes etapas del proyecto.

- **Análisis**: La etapa de análisis en el ciclo de vida del software corresponde al proceso a través del cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema.

- **Diseño**: En esta fase se estudian posibles opciones de implementación para el software que hay

que construir, así como decidir la estructura general del mismo. El diseño es una etapa compleja y su proceso debe realizarse de manera iterativa.

- **Implementación**: En esta fase hay que elegir las herramientas adecuadas, un entorno de desarrollo que facilite el trabajo y un lenguaje de programación apropiado para el tipo de software a construir. Esta elección dependerá tanto de las decisiones de diseño tomadas como del entorno en el que el software deba funcionar.

- **Pruebas**: Busca detectar los fallos cometidos en alguna de las etapas anteriores con el fin de corregirlos. Se dice que una prueba es un éxito si se detecta algún error.

- **Instalación o despliegue**: Ésta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

  - Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).
  - Adaptarlo a nuevas necesidades (mantenimiento adaptativo).
  - Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Paginador

Es un componente que proporcionar navegación entre información dividida en diversas páginas, de ahí el nombre. Muestra el tamaño de la página actual, opciones seleccionables por el usuario para cambiar ese tamaño, qué elementos se muestran y botones de navegación para ir a la página anterior o siguiente.

Programación Orientada a Objetos

La Programación Orientada a Objetos (POO) es un modelo o un estilo de programación que se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas reutilizables llamadas clases, las cuales se usan como base para crear instancias individuales de objetos, que cuenten con las mismas características y métodos. La Programación Orientada a Objetos permite que el código sea reutilizable, organizado y fácil de mantener, además, busca la representación de elementos tangibles del mundo real, de ahí el nombre "objeto". Incluye 4 pilares principales:

- **Abstracción**: Permite separar los componentes principales de un objeto y a aquellos que no lo son, esto se hace para reducir la complejidad del código para hacerlo más fácil de entender y da la capacidad de reutilizar código.

- **Herencia**: Su propósito es reutilizar el código. Se refiere a que una clase pueda heredar los métodos y los atributos de otra y a su vez, pueda tener nuevos y redefinir los heredados. Este pilar permite que se puedan crear clases basadas en las que fueron creadas anteriormente, generando así una jerarquía de clases dentro de la aplicación.

- **Polimorfismo**: Es la capacidad que tienen los objetos o métodos de una clase para responder de forma distinta dependiendo de los parámetros que reciba al momento de ser invocada. Con esto se consigue que un objeto de la clase padre se comporte como un objeto de una clase hija.

- **Encapsulamiento**: Consiste en limitar el acceso a una propiedad, principalmente a clases,esto evita que se acceda a los datos por medios diferentes a los especificados.

Protocolo HTTP

Se refiere a la abreviación del protocolo de transferencia por hipertexto (Hypertext Transfer Protocol), éste nos permite realizar peticiones de datos y recursos a través de internet. Es la base de cualquier intercambio de información entre los clientes web y los servidores, por medio del protocolo cliente servidor, esto quiere decir que una petición de datos será iniciada por un cliente, normalmente un navegador web y el servidor se encargará de responder con los datos solicitados. Se usa no solo para transmitir documentos de hipertexto (HTML), sino que, además, se usa para transmitir imágenes, vídeos, o enviar datos o contenido a los servidores, como en el caso de los formularios de datos. HTTP puede incluso ser utilizado para transmitir partes de documentos y actualizar páginas web (ilustración 3).

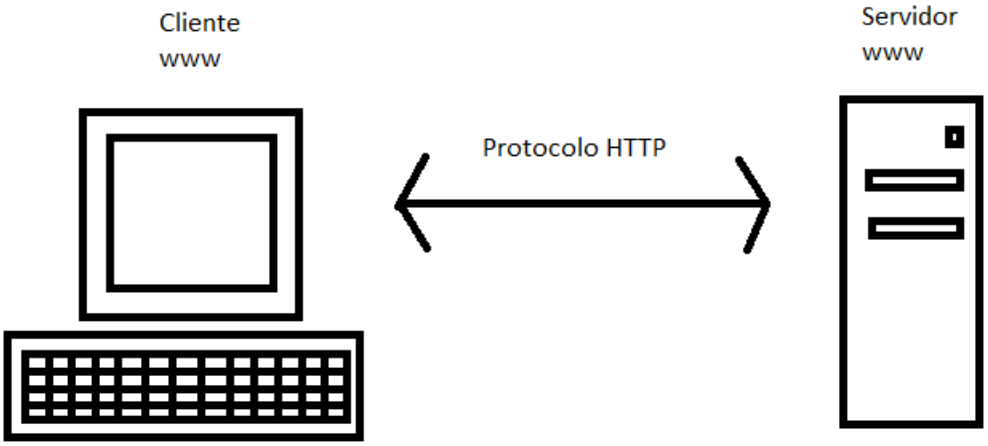


Ilustración 3. Funcionamiento del protocolo HTTP.

Protocolo TCP/IP

Su nombre completo es Protocolo de control detransmisión/Protocolo de Internet (Transmission Control Protocol/Internet Protocol), es un conjunto de reglas que permiten comunicación y transferencia de datos a través de redes, dispositivos e internet. Define como se mueve la información desde el remitente hasta el destinatario. El conjunto de protocolos TCP/IP se divide en capas, las cuales se describen a continuación:

- **Aplicación**: Se refiere al grupo de aplicaciones que permiten al usuario interactuar y acceder a la red, de esta forma puede ver, solicitar y recibir datos.

- **Transporte**: Es la encargada de proporcionar la conexión fiable entre los dispositivos que se comunican. Aquí se dividen los datos en partes más pequeñas llamadas paquetes, se verifica que la información haya llegado correctamente y se envía a las siguientes capas.

- **Internet**: También se le puede llamar capa de red, es la que se encarga de que los paquetes se envíen de forma rápida y correcta, ya que controla el flujo de los datos y el enrutamiento.

- **Acceso a la red**: También conocida como capa de enlace, es la que se encarga de gestionar la infraestructura física y técnica que se usa para el intercambio de información mediante redes inalámbricas, tarjetas de red, cables ethernet, etc., en la parte física y convertidores de señales en la técnica.

Pruebas de validación

Se utilizan para comprobar si un sistema o aplicación cumple con los requerimientos planteados por los clientes y así éstos den su visto bueno. Se llevan a cabo mediante un proceso de revisión en donde se verifica que el software cumpla con las especificaciones y logre su cometido. A continuación, se describen las que fueron utilizadas para el desarrollo de PIX.

- **Pruebas unitarias**: Las pruebas unitarias son una manera de comprobar que una parte del sistema funciona de manera correcta, se logra poniendo a prueba el funcionamiento únicamente del segmento que se quiere verificar.

- **Pruebas de sitio**: Consiste en la realización de pruebas por parte del cliente en un entorno controlado, con los desarrolladores como observadores. Se realizan con la intención de encontrar anomalías o características que no le agraden al cliente, para después corregirlas.

- **Pruebas de aceptación**: Estas pruebas las realiza el cliente de manera independiente sin ningún observador con el fin de garantizar que el funcionamiento del software sea el óptimo y cumpla con los requerimientos solicitados.

Responsividad Web

El diseño responsivo, es un término que se refiere a la capacidad de que una página web se adapte al tamaño de cualquier dispositivo (smarphone, tablet, laptop o computadora de escritorio) en donde se muestre.

Los elementos se van adaptando al ancho del dispositivo, reacomodando los elementos de tal forma que la información y los componentes no se empalmen o se deformen, buscando que se conserve un buen diseño para el usuario.

SASS

SASS (Syntactically Awesome Stylesheets) es una extensión del lenguaje de hojas de estilo en cascada (CSS). Ayuda a darle formato y buen diseño a las paginas HTML, ya que es posible agregar diferentes características (color, borde, altura, anchura, etc.) a cualquier elemento definido en el código HTML.

La principal ventaja de SASS es la posibilidad de convertir los CSS en algo dinámico. Permite trabajar más rápido en la creación de código porque cuenta con la posibilidad de crear variables para almacenar valores y con ellas llevar a cabo funciones que realicen operaciones matemáticas.

SCSS

Son hojas de estilo que contiene lenguaje Sass pero que mantienen la sintaxis de una hoja de estilo CSS, su nombre viene de *Sassy CSS*. Posee las capacidades adicionales de SASS, como variables, reglas anidadas, etc.

Scrum

Scrum es una metodología ágil que promueve la colaboración en los equipos de trabajo para lograr desarrollar productos complejos de forma rápida. Este marco de trabajo anima a los equipos a aprender a través de las experiencias, a autoorganizarse mientras aborda un problema y a reflexionar sobre sus aciertos y fallas para mejorar continuamente.

Scrum define un conjunto de prácticas y roles que se toman como base para definir el proceso de producción que usará un equipo de trabajo al desarrollar un proyecto.

Los roles principales en Scrum son:

- **Scrum Master**: Procura facilitar la aplicación de los procesos y las reglas de la metodología Scrum, gestionar cambios y reduce los impedimentos que se presenten a lo largo de la creación del proyecto.

- **Product Owner**: Es el representante los *stakeholders* (interesados en el proyecto), es el encargado de compartirle al equipo el objetivo que se busca para el proyecto.

- **Team**: Grupo de personas capacitadas para ejecutar el desarrollo de tareas en tiempo y forma, con la capacidad de autoorganizarse y no requieren seguimiento controlado.

Cuenta con cuatro etapas principales basadas en sprints, es decir, intervalos de tiempo establecidos que se plantea la empresa para hacer entregables parciales que sumados generan un producto terminado. Éstas son:

1. Planificación del sprint: El objetivo es definir el avance que se hará en ese intervalo y cómo se conseguirá. La planificación se hace en colaboración con todo el equipo Scrum.

2. Etapa de desarrollo: Cuando el trabajo del sprint está en curso, los encargados deben garantizar que no se generen cambios de último momento que puedan afectar los objetivos de éste. Además, se asegura el cumplimiento de los plazos establecidos para su término.

3. Revisión del sprint: Se lleva a cabo al final de cada Sprint para evaluar las tareas terminadas con éxito y las que no, para planificar las siguientes tareas. El equipo Scrum y las partes interesadas colaboran durante la revisión de lo que se hizo en el Sprint.

4. Retrospectiva: Es una oportunidad para que el equipo se inspeccione a sí mismo y cree un plan de mejoras que ejecutará durante el siguiente Sprint.

Tipos de datos

Los tipos de datos se refieren al contenido de una variable en el código y a la forma en la que ésta se manejará. Los tipos de datos pueden variar de un lenguaje de programación a otro, cada uno tiene propiedades y métodos diferentes. A continuación, se describirán los tipos de datos pertenecientes al lenguaje typescript, ya que es el que se usó para desarrollar la aplicación PIX:

- **Null**: Se refiere a que una variable no tiene un valor asociado, es decir, está vacía. Al establecer que una variable de este tipo null, ésta perderá su contenido.

- **Undefined**: Nos indica que una variable está definida, pero no se le ha especificado un valor.

- **Void**: El tipo de datos void se usa para indicar la falta de un tipo para una variable.

- **Boolean**: Este tipo de dato puede contener únicamente dos opciones; verdadero o falso. Estos

valores se utilizan mucho en las estructuras de control para determinar si una condición se cumple, dependiendo el caso se ejecutará un determinado bloque de código.

- **Number**: Se usa para representar tanto enteros como valores de punto flotante.

- **String**: El tipo de datos string se utiliza para almacenar cadenas de texto, puede ser desde un carácter hasta una oración o párrafo.

- **Array**: Es un conjunto de elementos del mismo tipo ordenados en fila.

- **Any**: Es una variable que puede cambiar de tipo durante la ejecución de un programa.

- **Never**: Se usa para representar valores que nunca se supone que ocurran. Por ejemplo, puede asignar never como el tipo de devolución de una función que nunca regresa. Esto puede suceder cuando una función siempre arroja un error o cuando está atrapada en un ciclo infinito.

Antecedentes del tema

Las empresas regularmente requieren aplicaciones donde sea posible analizar la información que reúnen, con ello determinar si existe algún tipo de riesgo o peligro, así mismo, requieren tener un estricto control sobre los individuos que tienen acceso a dicha información, por lo que contar con un sistema de monitoreo de usuarios es de gran ayuda, ya que muestra de forma gráfica y simplificada la información de cada usuario, su fecha de ingreso, la fecha en que su cuenta caduca, si se encuentra logueado o si es un usuario desactivado, entre otras funciones que varían de aplicación a aplicación.

Con dicho entorno se facilita la edición de la información de los usuarios, el cambio de contraseña o validez de la cuenta. Existen casos en donde es recomendable que se cree un software especializado de acuerdo con las necesidades específicas de cada cliente, puesto que los requerimientos varían dependiendo del tipo de uso que se busque.

Existen aplicaciones que realizan el control de usuarios, pero sin tener un objetivo en específico, por lo mismo pueden faltar o sobrar herramientas. PIX es un software creado bajo las necesidades específicas solicitadas por una empresa que desea monitorear a sus trabajadores, por lo mismo se encuentra optimizado para el uso que el cliente requiere, de manera que no hay desperdicio de funciones, de recursos ni de memoria. Así mismo, integra funciones que normalmente necesitarían de aplicaciones independientes; ya que engloba el control, la supervisión y la edición de un usuario, con el monitoreo de llamadas, reproductor y editor de audio y visualizador del espectro de frecuencias. Además, se diseñó para ser intuitivo, fácil de utilizar y visualmente atractivo utilizando con los colores solicitados, fomentando así la identidad de la empresa en las herramientas que usa.

La mayoría de los otros sistemas se enfocan en el tiempo que un usuario pasó dentro de la aplicación, así como de la visualización de su actividad. En cambio, el enfoque de PIX va más orientado a la escucha de los audios grabados de las líneas de teléfono y radio que se usan para comunicarse dentro de la empresa.

Contexto de la participación profesional

Mi aportación se ve reflejada en la parte visual y funcional del proyecto, mis actividades se apegaron a toda la parte *Frontend* de la aplicación. Es decir, me encargué de realizar la estructura del código fuente y de la aplicación web, enfocándome en la parte visual, a través de los diseños que se propusieron al inicio del proyecto, además de trabajar con todos los elementos y datos provenientes de la base de datos, por medio de servicios *REST* (POST: crear, GET: leer y consultar, PUT: editar y DELETE: eliminar), con el objetivo de que el usuario final pudiera visualizarlos, interactuar con ellos, editarlos, eliminarlos, según sea la función que necesite.

El proyecto se desarrolló con base en la metodología SCRUM, en la cual se cuenta con equipos de trabajo auto organizados y multifuncionales, en donde cada miembro del personal es responsable de terminar sus tareas en el tiempo establecido para ellas; esto sin la necesidad de supervisión constante. Se llevaba a cabo una reunión diaria de máximo 15 minutos de duración llamada *Daily Meeting*, en donde participaba todo el equipo de desarrollo y el Scrum Master; en esta reunión se tocaban y aclaraban puntos como: ¿Qué se logró el día anterior?, ¿Qué se quiere lograr hoy?, ¿Existe algún impedimento para avanzar? Preguntas que cada uno de los miembros del equipo debía responder.

Para organizar las tareas que se realizarían se llevaba a cabo una reunión llamada *Sprint Planning* (la cual se realiza de manera periódica antes de cada Sprint), en donde todo el equipo definía los puntos a abordar, el avance buscando, qué se iba a realizar y cómo se lograría. Cada Sprint (periodo de tiempo en el cual se deben entregar resultados) tuvo una duración aproximada de tres semanas; al finalizar cada uno se debe presentar al cliente los avances alcanzados mediante el *Sprint Review*, para que se validen los resultados y se obtenga una retroalimentación. Así mismo, cada vez que se concluía un Sprint se realizaba un Sprint Retrospective, en donde se evaluaba la forma en que se llevó a cabo la organización durante ese periodo y se planteaban posibles mejoras.

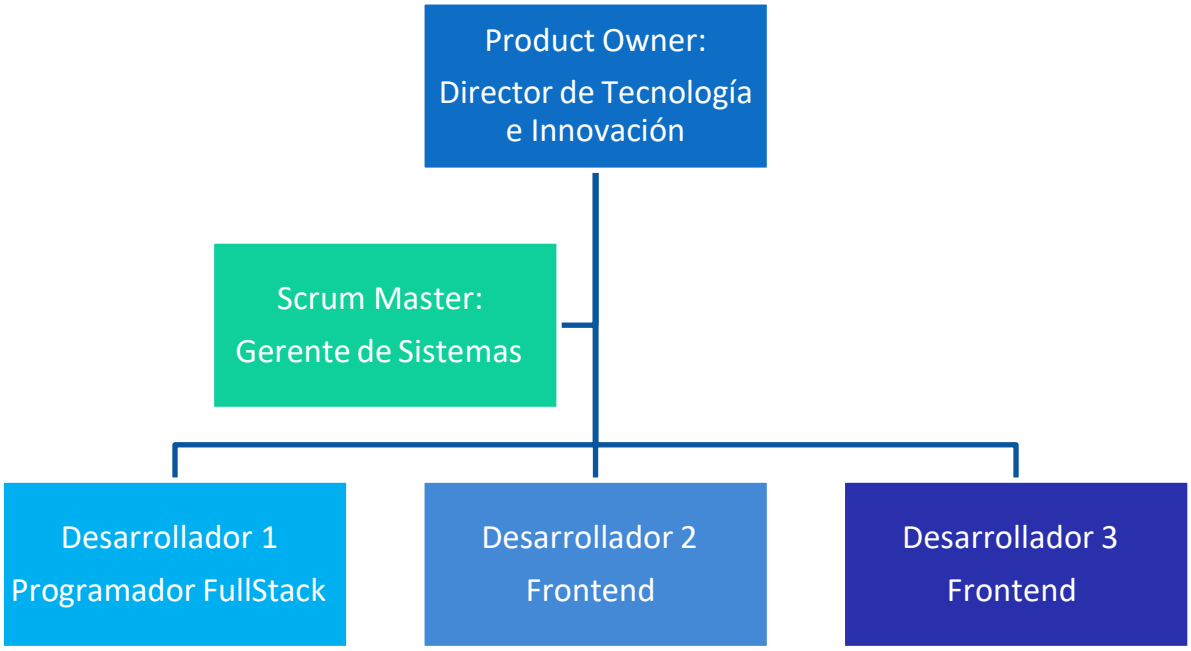


Ilustración 4. Diagrama de la jerarquía del equipo.

El equipo estuvo compuesto por cinco personas, como lo muestra en la Ilustración 4, en donde colaboramos tres programadores: el desarrollador que cuenta con más experiencia fue el encargado de crear todas las peticiones que realiza el *Backend*, de crear la base de datos y una pequeña parte del *Frontend* (su participación se puede ver en la sección de Resultados Obtenidos > Administrador > Control), además de brindar apoyo al resto del equipo. Los dos programadores restantes nos encargamos totalmente de la creación de la interfaz gráfica para el usuario final; mi función fue crear la estructura y muchas de las pantallas con las que cuenta la aplicación, además compartí ideas para mejorar la estructura de la base de datos.

Participación profesional

A mí me correspondió realizar la mayor parte del funcionamiento de la aplicación en cuanto a Frontend como se ha mencionado anteriormente.

A continuación, se muestra un esquema de los componentes (ubicados dentro de la carpeta components) realizados para el funcionamiento de la aplicación, ilustra la forma en la que se repartió el trabajo y el programador que se encontró a cargo de ellos. Cabe aclarar que cada uno de los elementos mostrados en la ilustración 5 representa una carpeta que contiene un archivo html, un archivo typescript y una hoja de estilos con extensión scss. Y para mayor organización estas mismas también pueden contener subcarpetas.

Carpeta Components

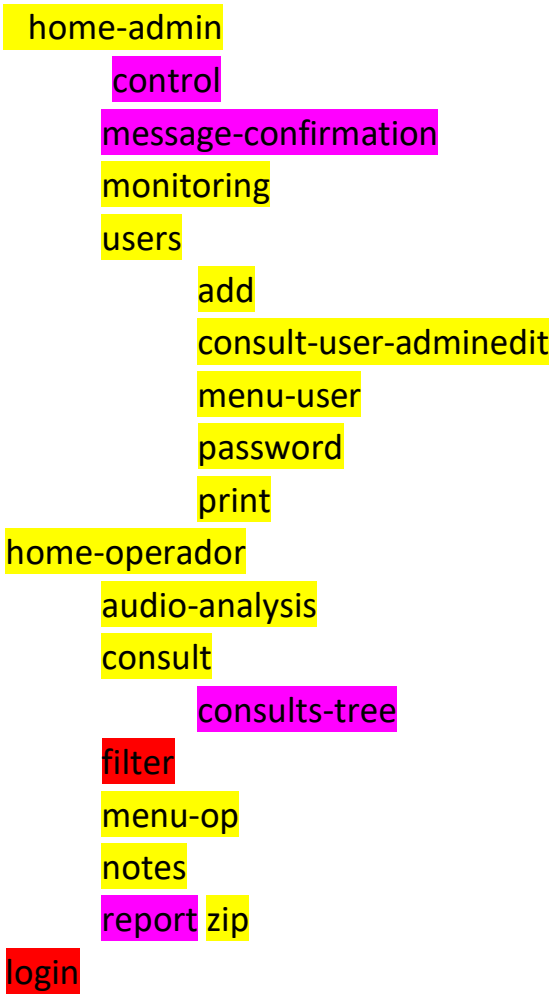


Ilustración 5. Encargados de la creación de las carpetas.

Los diferentes colores en las carpetas (Ilustración 5) representan qué programador estuvo encargado de su implementación (ver jerarquía en la Ilustración 4): el amarillo me representa a mí (desarrollador 2), el color rosa al desarrollador 1 y el rojo al desarrollador 3.

Para llevar a cabo la creación de la aplicación primero se planeó como se estructuraría y relacionaría la base de datos, después de la creación de ésta, se comenzó a implementar la *API* tipo *REST* y todas las peticiones que se necesitarían, a la mitad de dicho proceso inició la creación de la parte de *Frontend* en la cual tuve participación. Todo el equipo estuvo presente en las diferentes etapas que conformaron la creación del proyecto para poder contribuir con ideas y mejoras, sin embargo, cada miembro se desarrolló en su propio campo.

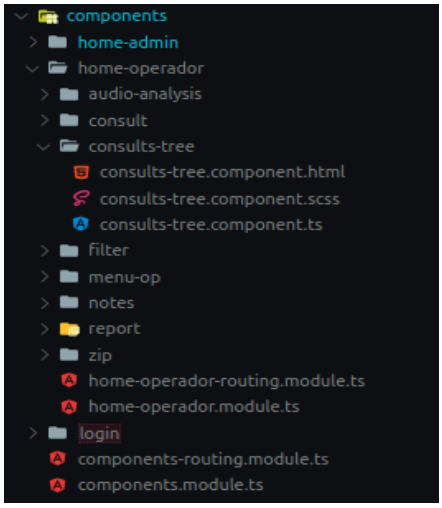


Ilustración 6. Organización del proyecto.

Como se puede ver en la Ilustración 5, yo me encargué de crear la plantilla de los dos tipos de usuarios solicitados por el cliente; operador y administrador, con el fin de que mis compañeros pudieran adecuarse a lo que yo hice e implementar sus funciones basándose en ello (Ilustración 6), facilitando la organización del proyecto y evitando crear un mismo archivo más de una vez. Todos los componentes los desarrollé por medio de comandos con angular CLI y para la funcionalidad de éstos, usé las tecnologías mencionadas en la sección de análisis y metodología empleada, la visualización de dichos componentes se puede visualizar mejor en la sección de resultados obtenidos.

En la sección dedicada al usuario administrador fue mi responsabilidad crear la pantalla principal, toda la sección de usuarios que es una tabla que cuenta con la información de todas las cuentas creadas en la aplicación, en esta misma pantalla es posible consultar la información referente a un usuario, crear nuevos, editar la información de alguno de ellos, borrar una cuenta, cambiar contraseña o imprimir la tabla completa. Y la sección de monitoreo en donde se muestra la información sobre los accesos que ha hecho cada uno de los usuarios existentes.

Para la parte de operador yo me encargué de crear la tabla de audios recabados, esta pantalla recibe el nombre de consultas, con los datos de esta tabla podemos ejecutar varias acciones, a ellas se acceden con el menú que aparece en la parte derecha (se puede visualizar el menú en la Ilustración 41), éstas son; agregar nota a uno o varios audios, crear zip con los audios seleccionados y borrar audio. También, realicé el desarrollo el reproductor de audio, en donde se puede dar pausa, reproducir, adelantar o atrasar el audio cinco segundos y subir o bajar volumen. Así mismo, desarrollé la pantalla de análisis de audio a la cual se accede dando doble click a un audio de la tabla consultas, en ésta es posible visualizar el espectro de frecuencia y un ecualizador, agregar algún comentario e interactuar con la barra reproductora de audio.

Análisis y metodología empleada

A nivel organizacional el equipo de trabajo se basó en la metodología Scrum. Este enfoque contempla que los requisitos sean cambiantes, permitiendo trabajar de manera competitiva, flexible y productiva, para lograr obtener resultados lo más pronto posible.

A nivel técnico fue necesario planear la estructura del proyecto (véase sección de anexos 1) el cual se estructuró con las siguientes carpetas:

- **common**: Contiene todos los archivos tipo ".ts" que se comparten en 2 ó más componentes del proyecto, son métodos reutilizables.

- **components**: En esta carpeta se encuentran todos los componentes que contiene el proyecto, desde la pantalla donde ingresan los usuarios hasta cada una de las tablas y sus funcionalidades.

- **core**: En esta carpeta se almacenan todos los servicios necesarios para el funcionamiento interno de la aplicación.

- **shared**: Todos los componentes que se usan más de una vez en el proyecto se colocaron en esta carpeta, para tener más orden entre los componentes compartidos y los únicos.

- **assets**: Aquí se encuentran todas las imágenes, logos y fuentes que se utilizan en el proyecto.

- **styles**: En esta carpeta se almacenan todas las hojas de estilos utilizadas en el proyecto.

Para la elaboración de este proyecto se utilizaron tecnologías como:

- Angular
- Bootstrap
- Angular Material
- HTML
- SCSS
- JSON
- Bases de datos
- TypeScript

El equipo de desarrollo optó por usar el *framework* Angular para el desarrollo de la aplicación porque es simple de usar, simplifica el proceso de desarrollo y la estructura del código, al trabajar por medio de módulos, además, cuenta con CLI o interfaz de línea de comandos, que reduce el tiempo que usualmente ocuparíamos al crear componentes, al hacerlo por nosotros. También es importante mencionar que está desarrollado en TypeScript, es de código abierto y recibe mantenimiento por parte de Google.

PIX requería ser una aplicación que se adecuara a tabletas y computadoras, por esta razón se decidió utilizar el *framework Bootstrap* para diseño *Frontend*, puesto que uno de sus principales objetivos es permitir que una página web se adapte automáticamente al tamaño del dispositivo utilizado, es decir la construcción de sitios web responsivos.

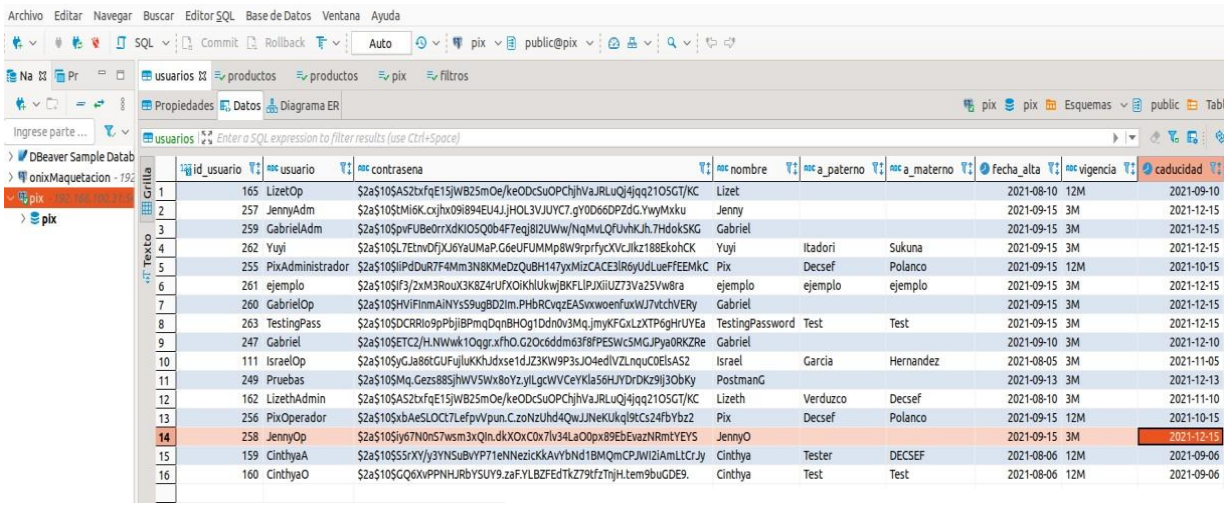
También se utilizó Angular Material, el cual es un módulo que fue construido por y para Angular, cuyo diseño está basado en Material Design porque facilita la implementación de muchos de los componentes Angular que se usan para la creación de una aplicación, cuenta con plantillas, estilos, herramientas y animaciones fáciles de modificar y adecuar a las necesidades de cada componente.

Una aplicación en Angular está compuesta por un conjunto de servicios y componentes que trabajando en conjunto logran la totalidad de la funcionalidad *Frontend*. Un componente es un elemento reutilizable conformado típicamente por un archivo HTML, una hoja de estilos y un archivo typescript (Ilustración 7). Sin embargo, de acuerdo con las necesidades del programador puede incluir más o menos archivos.

El HTML contiene las etiquetas necesarias para crear la vista que tendrá la página web. El archivo typescript se identifica por la extensión .ts y se refiere a la lógica de programación que requiera o se vaya a usar en la vista (HTML), éste permite mayor interactividad con los elementos, debe contener una clase que incluya todas las propiedades y métodos necesarios para la funcionalidad. El archivo CSS que puede también ser de tipo SASS o SCSS, posee todos los estilos y propiedades que se quieran incluir a la vista.

Ilustración 7. Contenido de un componente.

Se decidió utilizar la estructura JSON porque nos permite el intercambio de datos entre un servidor y aplicaciones web con un formato ordenado por medio de un nombre y valor, además es posible guardar varios tipos de datos, es fácil de leer y editar y no requiere código adicional para el análisis.



	id_usuario	usuario	contrasena	nombre	a_paterno	a_materno	fecha_alta	vigencia	caducidad
1	165	LizetOp	$2a$10$AS2bxfqE15jWB25mOe/keODcSuOPChjhVaJRLuQj4jqq21O5GT/KC	Lizet			2021-08-10	12M	2021-09-10
2	257	JennyAdm	$2a$10$tMi6K.cxjhx09i894EU4J.jHOL3VJUYC7.gY0D66DPZdG.YwyMxku	Jenny			2021-09-15	3M	2021-12-15
3	259	GabrielAdm	$2a$10$pvFUBe0rrXdKlO5Q0b4F7eqj8I2UWw/NqMvLQfUvhKJh.7HdokSKG	Gabriel			2021-09-15	3M	2021-12-15
4	262	Yuyi	$2a$10$L7EtnvDfjXJ6YaUMaP.G6eUFUMMp8W9rprfycXVcJIkz188EkohCK	Yuyi	Itadori	Sukuna	2021-09-15	3M	2021-12-15
5	255	PixAdministrador	$2a$10$liPdDuR7F4Mm3N8KMeDzQuBH147yxMizCACE3lR6yUdLueFfEEMkC	Pix	Decsef	Polanco	2021-09-15	12M	2021-10-15
6	261	ejemplo	$2a$10$lf3/2xM3RouX3K8Z4rUfXOiKhlUkwjBKFLlPJXiiUZ73Va25Vw8ra	ejemplo	ejemplo	ejemplo	2021-09-15	3M	2021-12-15
7	260	GabrielOp	$2a$10$HViFInmAiNYsS9ugBD2Im.PHbRCvqzEASvxwoenfuxWJ7vtchVERy	Gabriel			2021-09-15	3M	2021-12-15
8	263	TestingPass	$2a$10$DCRRIo9pPbjiBPmqDqnBHOg1Ddn0v3Mq.jmyKFGxLzXTP6gHrUYEa	TestingPassword	Test	Test	2021-09-15	3M	2021-12-15
9	247	Gabriel	$2a$10$ETC2/H.NWwk1Oqgr.xfhO.G2Oc6ddm63f8fPESWcSMGJPya0RKZRe	Gabriel			2021-09-10	3M	2021-12-10
10	111	IsraelOp	$2a$10$yGJa86tGUFujluKKhJdxse1dJZ3KW9P3sJO4edlVZLnquC0ElsAS2	Israel	Garcia	Hernandez	2021-08-05	3M	2021-11-05
11	249	Pruebas	$2a$10$Mq.Gezs88SjhWV5Wx8oYz.ylLgcWVCeYKla56HJYDrDkz9lj3ObKy	PostmanG			2021-09-13	3M	2021-12-13
12	162	LizethAdmin	$2a$10$AS2bxfqE15jWB25mOe/keODcSuOPChjhVaJRLuQj4jqq21O5GT/KC	Lizeth	Verduzco	Decsef	2021-08-10	3M	2021-11-10
13	256	PixOperador	$2a$10$xbAeSLOCt7LefpvVpun.C.zoNzUhd4QwJJNeKUkql9tCs24fbYbz2	Pix	Decsef	Polanco	2021-09-15	12M	2021-10-15
14	258	JennyOp	$2a$10$iy67N0nS7wsm3xQln.dkXOxC0x7lv34LaO0px89EbEvazNRmtYEYS	JennyO			2021-09-15	3M	2021-12-15
15	159	CinthyaA	$2a$10$SSrXY/jy3YNSuBvYP71eNNezicKkAvYbNd1BMQmCPJWI2iAmLtCrJy	Cinthya	Tester	DECSEF	2021-08-06	12M	2021-09-06
16	160	CinthyaO	$2a$10$GQ6XvPPNHJRbYSUY9.zaF.YLBZFEdTkZ79tfzTnjH.tem9buGDE9.	Cinthya	Test	Test	2021-08-06	12M	2021-09-06

Ilustración 8. Visualización de un usuario en DBeaver.

Para la visualización de la base de datos de forma gráfica se usó el programa DBeaver (Ilustración 8), tanto para la creación de funciones como de tablas y relaciones (anexo 4). Para ejemplificar esto se muestra la tabla de usuarios (Ilustración 9), la cual contiene todos los datos que el cliente pidió que se almacenaran, además de la información necesaria para el manejo de datos dentro de la aplicación, la contraseña encriptada, el tipo de usuario y si este se encuentra logueado, activo o si bloqueó la sesión.

nombre	a_paterno	a_materno	fecha_alta	vigencia	caducidad	id_rol	primero	activo	logueado	id_bl	bloqueado
Lizet			2021-08-10	12M	2021-09-10	2	[ ]	[v]	[ ]	[NULL]	[ ]
Jenny			2021-09-15	3M	2021-12-15	1	[ ]	[v]	[ ]	[NULL]	[ ]
Gabriel			2021-09-15	3M	2021-12-15	1	[ ]	[v]	[ ]	[NULL]	[ ]
Yuyi	Itadori	Sukuna	2021-09-15	3M	2021-12-15	1	[ ]	[v]	[ ]	[NULL]	[ ]
Pix	Decsef	Polanco	2021-09-15	12M	2021-10-15	1	[ ]	[v]	[ ]	[NULL]	[ ]
ejemplo	ejemplo	ejemplo	2021-09-15	3M	2021-12-15	2	[ ]	[ ]	[ ]	[NULL]	[ ]
Gabriel			2021-09-15	3M	2021-12-15	2	[ ]	[v]	[ ]	[NULL]	[ ]
TestingPassword	Test	Test	2021-09-15	3M	2021-12-15	2	[ ]	[v]	[ ]	[NULL]	[ ]
Gabriel			2021-09-10	3M	2021-12-10	2	[ ]	[v]	[v]	[NULL]	[ ]
Israel	Garcia	Hernandez	2021-08-05	3M	2021-11-05	1	[ ]	[v]	[v]	[NULL]	[v]
PostmanG			2021-09-13	3M	2021-12-13	2	[ ]	[v]	[v]	[NULL]	[ ]
Lizeth	Verduzco	Decsef	2021-08-10	3M	2021-11-10	1	[ ]	[v]	[ ]	[NULL]	[ ]
Pix	Decsef	Polanco	2021-09-15	12M	2021-10-15	2	[ ]	[v]	[ ]	[NULL]	[ ]
JennyO			2021-09-15	3M	2021-12-15	2	[ ]	[v]	[ ]	[NULL]	[ ]
Cinthya	Tester	DECSEF	2021-08-06	12M	2021-09-06	1	[ ]	[v]	[ ]	[NULL]	[ ]
Cinthya	Test	Test	2021-08-06	12M	2021-09-06	2	[ ]	[v]	[ ]	[NULL]	[ ]

Ilustración 9. Contenido de la tabla usuarios.

Para facilitar la transferencia de información a través de peticiones tipo GET, POST, DELETE y UPDATE,

es decir, cuando un cliente solicita algún tipo de información o realiza la edición de algún dato, se utiliza el protocolo HTTP, el cual es un protocolo ampliable y fácil de usar, la estructura Cliente-Servidor se adecua a las necesidades de PIX y posee la capacidad para usar cabeceras.

A continuación, se describen algunas de las partes de código fuente de la aplicación que considero más recalcables. Esto para poder ilustrar y explicar la forma en la que me organizo al programar y algunas de las herramientas que utilicé.

Para la implementación del espectro de frecuencia para la parte de análisis de audio para un usuario operador (se puede ver el resultado en la Ilustración 42) se usó una biblioteca llamada Wavesurfer (Ilustración 10).

```
var WaveSurfer = require('wavesurfer.js');
var TimelinePlugin = require('wavesurfer.js/dist/plugin/wavesurfer.timeline.min.js');
```

Ilustración 10. Permitir el uso de la biblioteca Wavesurfer.

La elección de esta biblioteca fue debido a que, comparando con otras alternativas, Wavesurfer cumple con la visualización del espectro de frecuencia que se buscaba mostrar al usuario, ya que la mayoría que llegué a encontrar únicamente mostraban el gráfico superior del análisis espectral es decir la amplitud de la onda. Y para la aplicación mi objetivo era mostrar las ondas completas, de cresta a valle (Ilustración 11).

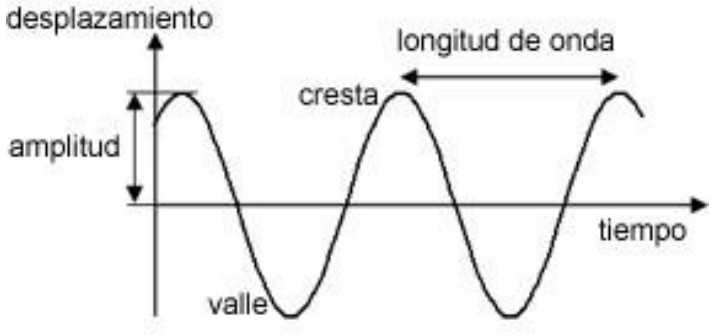


Ilustración 11. Partes de una onda.

Además, con esta librería es muy sencillo editar el estilo de la gráfica de frecuencia, por ejemplo: el color que tomarán las ondas al iniciar, el color de progreso, la altura, el intervalo de la regla, etc. Para ello solo es necesario indicarle el nombre de la propiedad y el color o el valor a editar, al momento de crear la visualización de la onda acústica de alguna de las llamadas que registra la aplicación.

A continuación, se presentan los métodos implementados para el funcionamiento de la frecuencia (Ilustración 13 e Ilustración 14), además de los estilos para su visualización (Ilustración 12).

```
/**
 * Se genera la frecuencia del audio y se le asignan las características.
 */
public generateWaveform(): void {
  this.isLoad = false
  Promise.resolve(null).then(() => {
    this.wave = WaveSurfer.create({
      container: '#waveform',
      waveColor: '#2a8f8d',
      progressColor: '#8d2888',
      barHeight: 1.3,
      autoCenter: true,
      height: 250,
      plugins: [
        TimelinePlugin.create({
          container: '#wave-timeline',
          //timeInterval: 0.5
          height: 15
        })
      ]
    });
    this.wave.on('ready', () => {
      this.isLoad = true;
    });
  });
}
```

Ilustración 12. Características para generar la frecuencia del audio.

```
/** Método para descargar el audio(base64) y crear el reproductor con sus propiedades de el audio,
 *  comienza a reproducirse desde donde se quedó la última vez*/
public async loadAudio() {
  try {
    const idProduct = this.data.idProduct;
    const audio: Audio = await this.productRequestService.getAudioByIdProduct(idProduct);
    this.audio = new Audio(`data:audio/wav;base64,${audio.file}`);
    fetch(this.audio.src)
      .then(res => res.blob())
      .then((blob) => {
        const auxAudio = new File([blob], 'testAudio.wav', { type: 'audio/wav' })
        this.wave.setVolume(0);
        this.wave.loadBlob(auxAudio);
      });
  } catch (error) {
    console.warn(error);
  }
}
```

Ilustración 13. Método para reproducir un audio.

```
/**
 * Verifica si la frecuencia ya esta creada, si no la crea y si sí detecta cambios.
 */
public async generateTimeWave() {
  if (!this.wave) { await this.generateWave() }
  this.cdr.detectChanges();
  Promise.resolve().then(() => { this.loadAudio(); });
}

/**
 * Genera la frecuencia de forma asíncrona.
 */
public async generateWave() {
  this.generateWaveform();
}
```

Ilustración 14. Métodos para la creación de la frecuencia.

Para tener un código más limpio, se buscó reutilizar lo más posible, es decir, cuando varios componentes usaban el mismo método, se implementaba este mismo en un servicio para que de esa forma sólo fuera necesario llamarlo, en lugar de tener las mismas instrucciones una y otra vez. Por ejemplo, tanto los componentes que se abren a través del menú al costado derecho de las tablas como el componente de Análisis de audio se manejan por medio de un servicio de Angular Material llamado MatDialog, el cual fue importado de la siguiente manera:

import {MatDialogModule} from '@angular/material/dialog'.

El siguiente método muestra el código necesario para abrir un modal, así como los datos que se deben pasar como parámetro:

```
public openModal(component: any, dataValue: any, widthP: string, heightP?: string) {
  const dialogRef = this.dialog.open(component, {
    data: dataValue,
    width: widthP,
    height: heightP,
    disableClose: true
  });
  return dialogRef.afterClosed();
}
```

Ilustración 15. Método utilizado para abrir las ventanas emergentes (modal).

La información que recibe de los parámetros es la siguiente (Ilustración 15):

- **component**: Componente que se quiere mostrar.
- **dataValue**: Cualquier valor que se quiera mandar al componente.

- **widthP**: Ancho del modal.
- **heightP**: Altura del modal

Así mismo, se creó un método en el servicio de modales que permite abrir cuadros de texto con diferentes estilos e información (
Ilustración 16).

```
public notificationInfo(sms: string, img: string, color: string, question: boolean, number?:any, ) {
  const dialogRef = this.dialog.open(NotificationComponent, {
    width: '450px',
    panelClass: 'styleDialog',
    data: {
      message: sms,
      icon: img,
      css: color,
      question: question,
      number: number
    },
    disableClose: true
  });

  return dialogRef.afterClosed();
}
```

Ilustración 16. Método para abrir cuadros de texto.

Datos que se requieren para usar este método (ver
Ilustración 16):

- **sms**: Texto que aparece en el cuadro de texto.
- **img**: Icono a mostrar.
- **color**: Nombre del estilo que tendrá el texto.
- **question**: Propiedad que indica si el cuadro contendrá una pregunta o no.
- **number**: Si se realizó la selección de elementos previamente, muestra la cantidad seleccionada, esto se puede ver al momento de borrar audios.

El resultado del código anterior se visualiza de la siguiente forma (Ilustración 17):

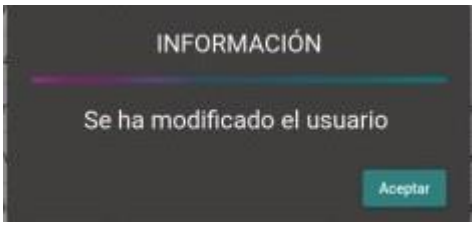


Ilustración 17. Visualización de un cuadro de texto.

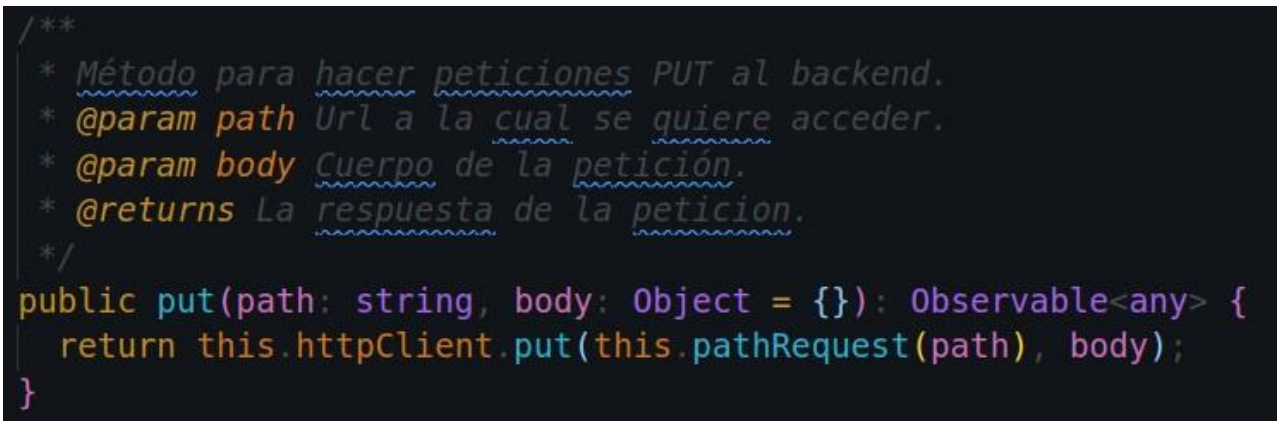
Para la parte de peticiones, es decir, cuando se solicita la información al *Backend* y a la base de datos, se creó un archivo tipo typescript, llamado RequestService.ts, en donde cada método tiene una función en específico. A continuación, se mostrará el código de las peticiones HTTP que se usaron: GET (Ilustración 18), POST (Ilustración 19), PUT (Ilustración 20) y DELETE (Ilustración 21).

```
/**
 * Método para hacer peticiones GET al backend.
 * @param path Url a la cual se quiere acceder.
 * @param params Parametros adicionales para realizar la petición.
 * @returns La respuesta de la peticion.
 */
public get(path: string, params: HttpParams = new HttpParams()): Observable<any> {
  return this.httpClient.get(this.pathRequest(path), { params });
}
```

Ilustración 18. Método GET.

```
/**
 * Método para hacer peticiones POST al backend.
 * @param path Url a la cual se quiere acceder.
 * @param body Cuerpo de la petición.
 * @returns La respuesta de la peticion.
 */
public post(path: string, body: Object = {}): Observable<any> {
  return this.httpClient.post(this.pathRequest(path), body);
}
```

Ilustración 19. Método POST.



```
/**
 * Método para hacer peticiones PUT al backend.
 * @param path Url a la cual se quiere acceder.
 * @param body Cuerpo de la petición.
 * @returns La respuesta de la peticion.
 */
public put(path: string, body: Object = {}): Observable<any> {
  return this.httpClient.put(this.pathRequest(path), body);
}
```

Ilustración 20. Método PUT.

```
/**
 * Método para hacer peticiones DELETE al backend.
 * @param path Url a la cual se quiere acceder.
 * @returns La respuesta de la peticion.
 */
public delete(path: string): Observable<any> {
  return this.httpClient.delete(this.pathRequest(path));
}
```

Ilustración 21. Método DELETE.

Para la creación de las tablas que muestran la información de los usuarios o las llamadas en la aplicación utilicé un componente de Angular Material llamado mat-table, el cual ya incluye plantillas, estilos y herramientas para facilitar la implementación de tablas, tales como un paginador y algoritmos de ordenamiento. Estos últimos se importaron y usaron en cada una de las tablas del proyecto y se implementaron de la siguiente forma (Ilustración 22):

```
import { MatPaginator } from '@angular/material/paginator';
import { MatSort } from '@angular/material/sort';
```

Ilustración 22. Importación de herramientas de Angular Material.

Para declarar las variables que se usan para cada uno de los elementos se implementaron las siguientes líneas de código (Ilustración 23) y luego se inicializaron (Ilustración 24). La etiqueta mat-paginator y las propiedades para visualizar el paginador se implementaron en cada uno de los archivos HTML de los componentes que contienen tablas (Ilustración 25).

```
@ViewChild(MatPaginator) paginator!: MatPaginator;
@ViewChild(MatSort) sort!: MatSort;
```

Ilustración 23. Implementación de herramientas de Angular Material.

```
this.dataSource.sort = this.sort;
this.dataSource.paginator = this.paginator;
```

Ilustración 24. Inicialización de herramientas de Angular Material.

```
<mat-paginator class="contentPaginator"[pageSizeOptions]="[pagination]"
showFirstLastButtons></mat-paginator>
```

Ilustración 25. Uso de sort y paginador en el código html.

Para implementar las tablas de manera más organizada y de forma dinámica, creé un arreglo de objetos en el archivo typescript que se itera en el archivo HTML y crea una columna de la tabla por cada objeto que encuentra, cada una de éstos contiene la propiedad **name**, en donde se especifica una cadena de texto que coincide en nombre con la propiedad del objeto que se obtiene de la base de datos con el objetivo de llenar la tabla, es de esta forma que los datos se posicionan en la columna correcta. La propiedad **value** es el nombre que tomará la columna y por último **show** determina si una columna es visible o no.

Esto se hizo para poder tener un mejor control sobre los datos y las columnas dentro de la aplicación, así como para hacer más fácil la modificación de campos y no llenar el archivo HTML con código repetitivo.

A continuación, se muestra la forma en la que se implementaron las columnas de la tabla para la sección de monitoreo de un usuario de tipo administrador (Ilustración 26), se puede observar el resultado en la Ilustración 39.

```
/** Columnas de la tabla */
columns = [
  // { name: 'idMonitoring', value: 'ID MONITOREO', show: true },
  { name: 'username', value: 'USUARIO', show: true },
  // { name: 'idUser', value: 'ID USUARIO', show: true },
  { name: 'entry', value: 'ENTRADA', show: true },
  { name: 'exit', value: 'SALIDA', show: true },
  { name: 'expiration', value: 'CADUCA', show: true },
  { name: 'time', value: 'TIEMPO', show: true },
  { name: 'device', value: 'EQUIPO', show: true },
];
```

Ilustración 26. Implementación de las columnas de una tabla.

Resultados obtenidos

A continuación, se muestra una parte del producto final al que se llegó, a pesar de que estuve presente en todas las etapas de desarrollo, mi participación se centró únicamente en la parte de *Frontend*, que es la parte de la que se hablará en esta sección. Todos los elementos visuales mostrados a continuación los hice apoyándome en las herramientas mencionadas en la sección de metodología empleada.

Comencé a crear los componentes a partir de las pantallas principales de cada tipo de usuario, es decir, si su rol al usar la aplicación es tipo **Operador** o **Administrador.**

Dependiendo del rol que desempeña cada uno de los usuarios, las funciones a realizar y las pantallas a mostrar son diferentes (únicamente se describirán los componentes que realicé).

En la parte superior de la pantalla siempre se encuentra un menú, en el cual se tiene la opción de cerrar o bloquear sesión (Ilustración 27), la simbología de las diversas pantallas que se cuentan dependiendo del perfil que tenga el usuario registrado, el rol y el nombre de usuario, así como la fecha con hora y un botón de ayuda en el cual se muestra la versión de la aplicación.

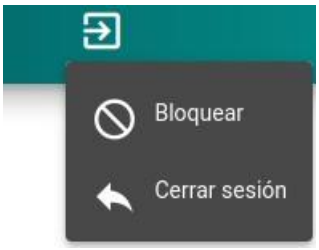


Ilustración 27. Menú para salir de la aplicación.

Administrador

A continuación, se muestra la pantalla principal del perfil **Administrador** (Ilustración 28), este tipo de usuarios llevan el control sobre todos los usuarios registrados en la aplicación, pueden visualizar la lista de los accesos, y cuentan con permiso de editar propiedades de otros usuarios y de crear nuevos. Para ver la estructura de las carpetas véase Anexo 2.

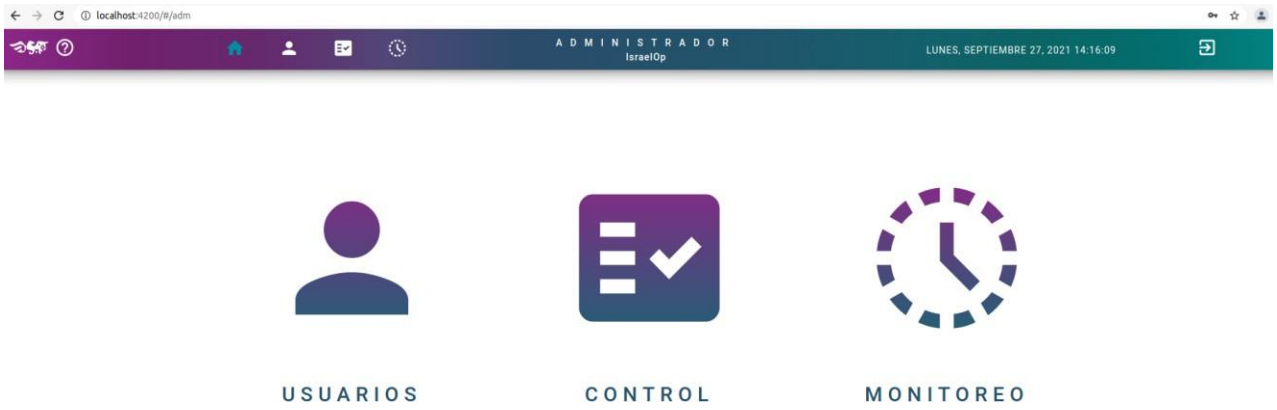


Ilustración 28. Pantalla principal del usuario administrador.

Para un usuario administrador, al ingresar a la aplicación con su nombre de usuario y contraseña, lo primero que se muestra es una pantalla con diferentes secciones: usuarios, control y monitoreo; se puede acceder a cada sección ya sea por el menú superior o por los íconos grandes al centro de la pantalla.

Usuarios

En la sección de usuarios se muestra una tabla con la información referente a cada una de las cuentas existentes dentro de la aplicación (Ilustración 30). Aquí se puede ver si un usuario está inactivo, logueado, deslogueado o desactivado; se visualiza por medio de los pequeños círculos ubicados en la primera columna al costado izquierdo de los recuadros de selección: el círculo negro significa que el usuario se encuentra inactivo en ese momento, el verde significa que está dentro del sistema en ese momento, es decir, está activo y el rojo se refiere a que el usuario se encuentra desactivado.

Al dar doble click sobre alguno de estos círculos se muestra un mensaje que espera la confirmación para activar/desactivar el usuario (Ilustración 29).

Ilustración 29. Cuadro de texto para activar/desactivar un usuario.

Al dar click en Aceptar se activa/desactiva al usuario sobre el que se dio doble click. Si se da click en Cancelar no se realiza ningún cambio.

También se puede ver el perfil al que pertenece cada usuario, es decir, operador o administrador, la fecha en que se dio de alta, la vigencia que se le asignó a un usuario al ser creado o modificado (referente al tiempo de vida de una cuenta, muestra la fecha en que caduca la cuenta), además del nombre completo de cada usuario (Ilustración 30).

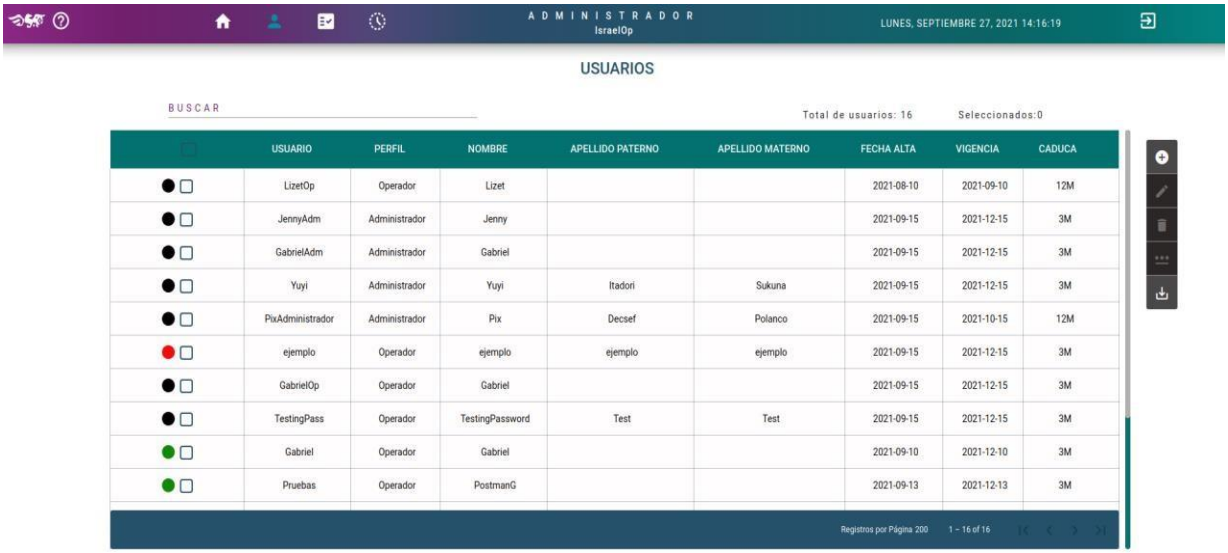


Ilustración 30. Tabla que muestra todos los usuarios de la aplicación.

Al dar doble click sobre un usuario de la tabla se despliega un cuadro de consulta en donde se visualizan los datos del usuario con la posibilidad de imprimir la información en un PDF (Ilustración 31). La información de este cuadro no puede ser editada.

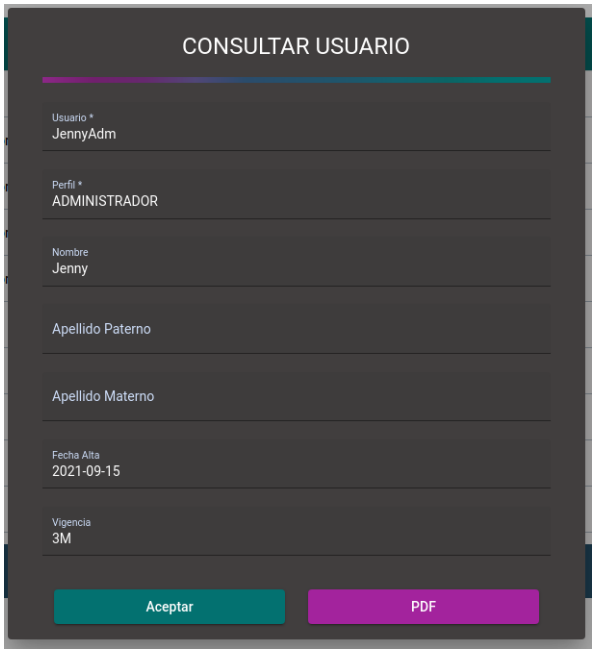


Ilustración 31. Modal para consultar la información de un usuario.

La pantalla Usuarios (Ilustración 30) cuenta con un menú, que se encuentra al costado derecho de la tabla, en el cual se puede acceder a las acciones que puede realizar un usuario de tipo administrador sobre los datos de la tabla.

Las acciones por realizar y su correspondiente visual en la aplicación son las siguientes (Ilustración 32);

- Agregar un nuevo usuario.
- Editar un usuario.
- Eliminar usuarios.
- Cambiar contraseña.
- Impresión de tabla.

Ilustración 32. Menú pantalla Usuarios.

Agregar un nuevo usuario

En este cuadro de diálogo se pide cierta información necesaria para crear un nuevo usuario (Ilustración 33), existen campos obligatorios como el nombre de usuario, el nombre de pila, el perfil o rol que tomará en la aplicación, la contraseña y la vigencia, con estos campos llenos se activa el botón de crear, aún si no se llenan los campos no obligatorios, el botón de crear podrá ser activado.

Ilustración 33. Creación de un nuevo usuario.

Editar un usuario

En este modal se encuentra la información referente al usuario que se seleccionó (Ilustración 34), es importante mencionar que únicamente se puede acceder a esta funcionalidad si está seleccionado únicamente un usuario en la tabla, si se selecciona más de uno la opción de editar se deshabilita. Los campos que pueden ser modificados son la vigencia y el nombre del usuario.

Ilustración 34. Editar un usuario.

Eliminar usuarios

Con esta función es posible eliminar desde uno hasta los usuarios que sean necesarios. El ícono en el menú se habilita cuando hay por lo menos un usuario seleccionado, para completar la acción es necesario confirmar que se desea eliminar la información seleccionada (Ilustración 35).

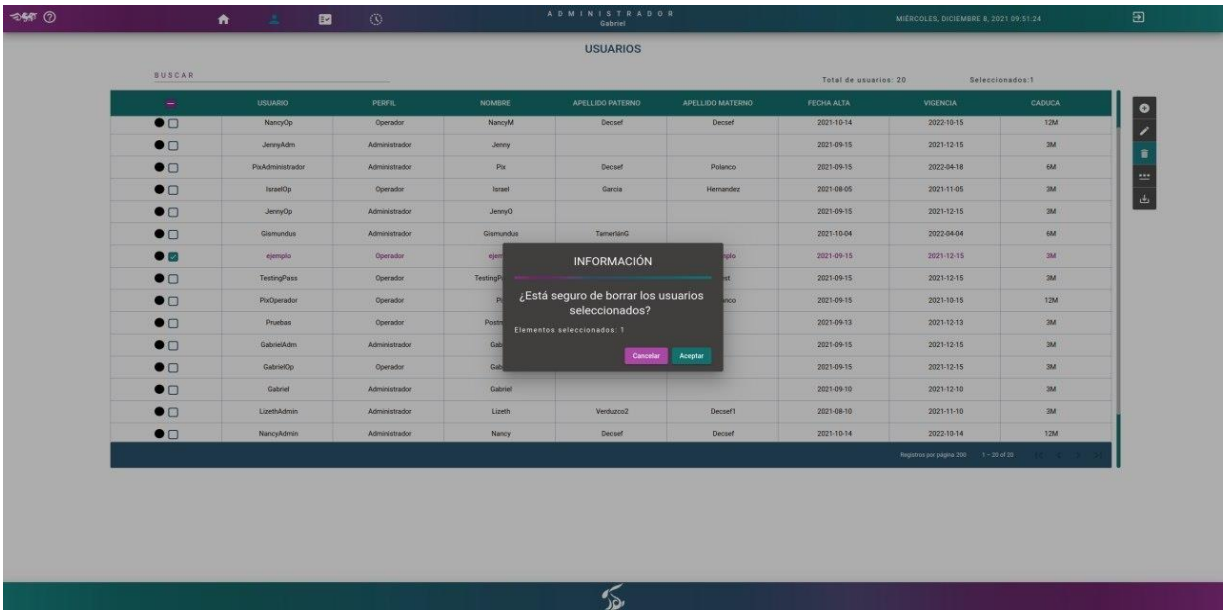


Ilustración 35. Función eliminar usuarios.

Cambiar contraseña

En este cuadro de diálogo es posible asignarle una nueva contraseña al usuario seleccionado, al igual que la función de editar usuario se habilita si se tiene seleccionado únicamente un usuario (Ilustración 36).

Para agregar una nueva contraseña es necesario repetir la contraseña para verificar que son iguales.

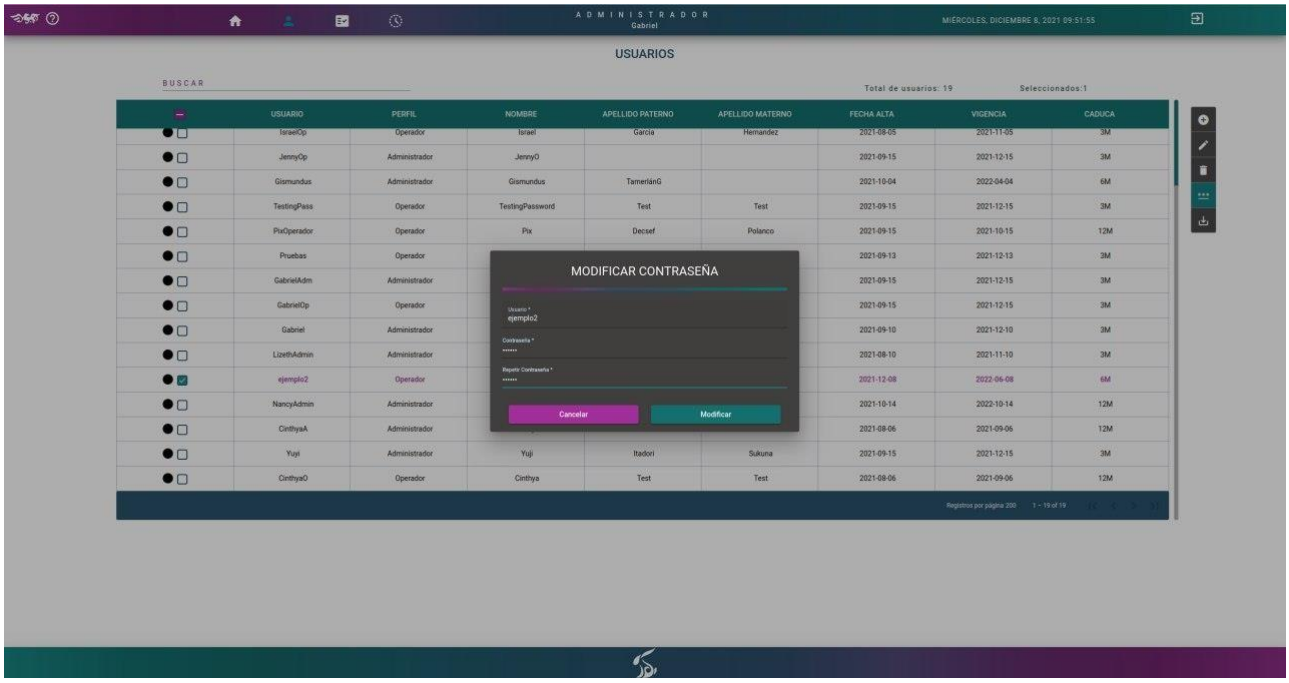


Ilustración 36. Función modificar contraseña de un usuario.

Impresión de tabla

Aquí se muestra una vista previa de la tabla que se va a imprimir (Ilustración 37), al dar click en el botón de imprimir se generará un PDF, se puede elegir qué columnas imprimir y cuales no, esto se hace dando click derecho sobre la cabecera de la tabla, esta acción despliega un cuadro de diálogo con una lista de todas las columnas, al deseleccionarlas se quitan de la tabla y de la vista previa.

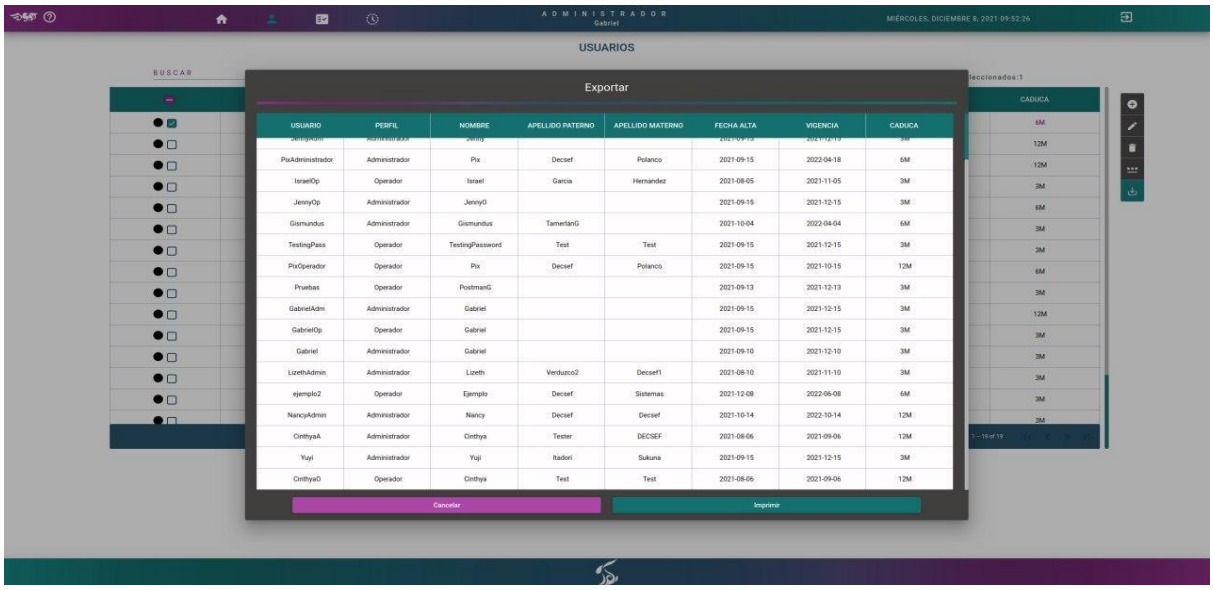


Ilustración 37. Previsualización de la impresión de una tabla.

Control

En la pantalla de control se muestra una lista de las líneas telefónicas y de radio usadas para la pantalla de consultas perteneciente al usuario operador (Ilustración 38).

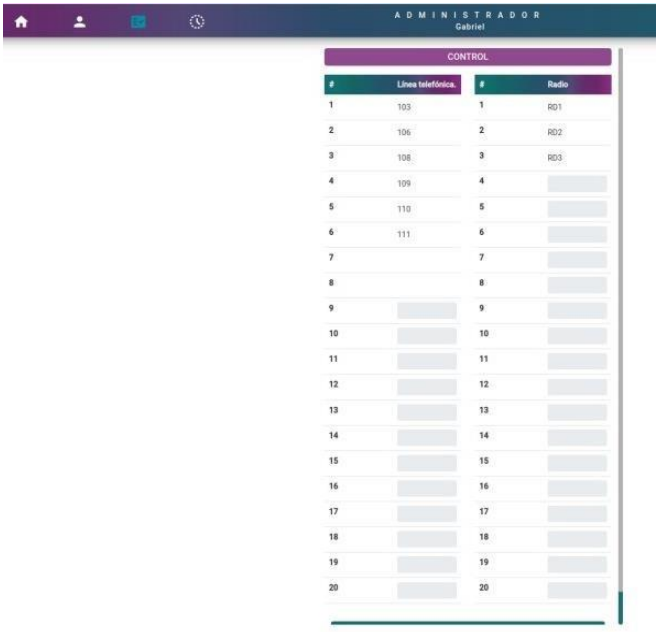


Ilustración 38. Pantalla Control.

Monitoreo

En la pantalla de monitoreo se muestra una tabla con la información sobre los accesos que ha hecho cada uno de los usuarios existentes, se muestra la hora de entrada, la hora de salida, la fecha en que la cuenta expira, el tiempo que pasó dentro de la aplicación y la dirección MAC de 48 bits que corresponde de forma única al dispositivo que se utilizó para acceder a la aplicación.

En la parte superior de la tabla (Ilustración 39) se encuentra un buscador y el contador total de ingresos a la aplicación.

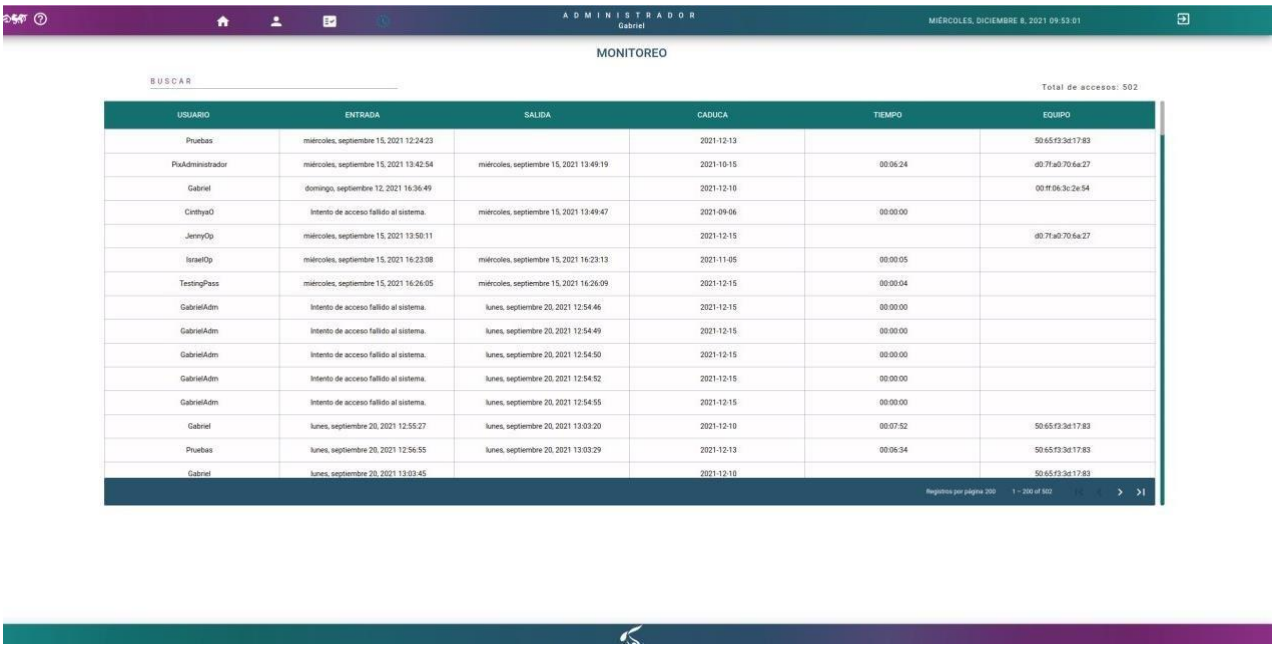


Ilustración 39. Pantalla monitoreo.

Operador

A esta sección tienen acceso todos los usuarios que no cuenten con el perfil de administrador, este tipo de perfil no tiene acceso a las actividades descritas en la sección anterior. Para ver la estructura de las carpetas véase el anexo 3.

Una vez que un usuario de tipo operador accede a la aplicación puede ver la siguiente pantalla (Ilustración 40):

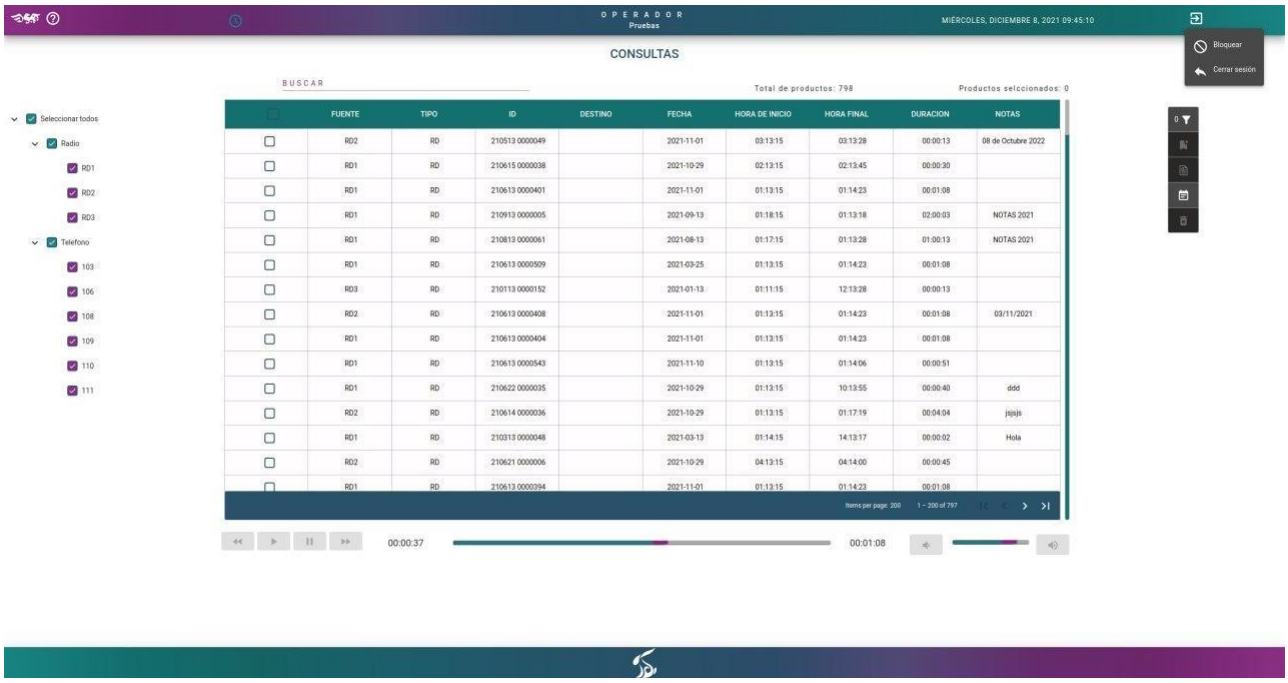


Ilustración 40. Pantalla principal de un usuario Operador.

Aquí se pueden llevar a cabo diversas actividades. El objetivo de esta sección es que se monitoreen los productos, que se refiere a la información recabada de diversas líneas telefónicas o radios (las cuales se pueden observar en la Ilustración 38) mismas que aparecen en un árbol al costado izquierdo de la tabla.

La información recabada de las líneas telefónicas y los radios se muestran en una lista, en donde cada registro se refiere a un audio. Es posible reproducir, pausar y adelantar el elemento seleccionado, así como modificar el volumen con el reproductor de la parte inferior. A continuación, se presenta el contenido del menú y su funcionamiento (Ilustración 41):

- Agregar filtro.
- Agregar nota a uno o varios audios.
- Crear un zip con los audios seleccionados.
- Crear reportes.
- Borrar audio.

Ilustración 41. Menú consultas.

Análisis de audio

Al dar doble click sobre un audio se abre un modal en donde se visualiza la información de la tabla, las notas obtenidas del análisis, el espectro de frecuencias (Ilustración 42), un panel de control para modificar el audio y el reproductor (Ilustración 43).

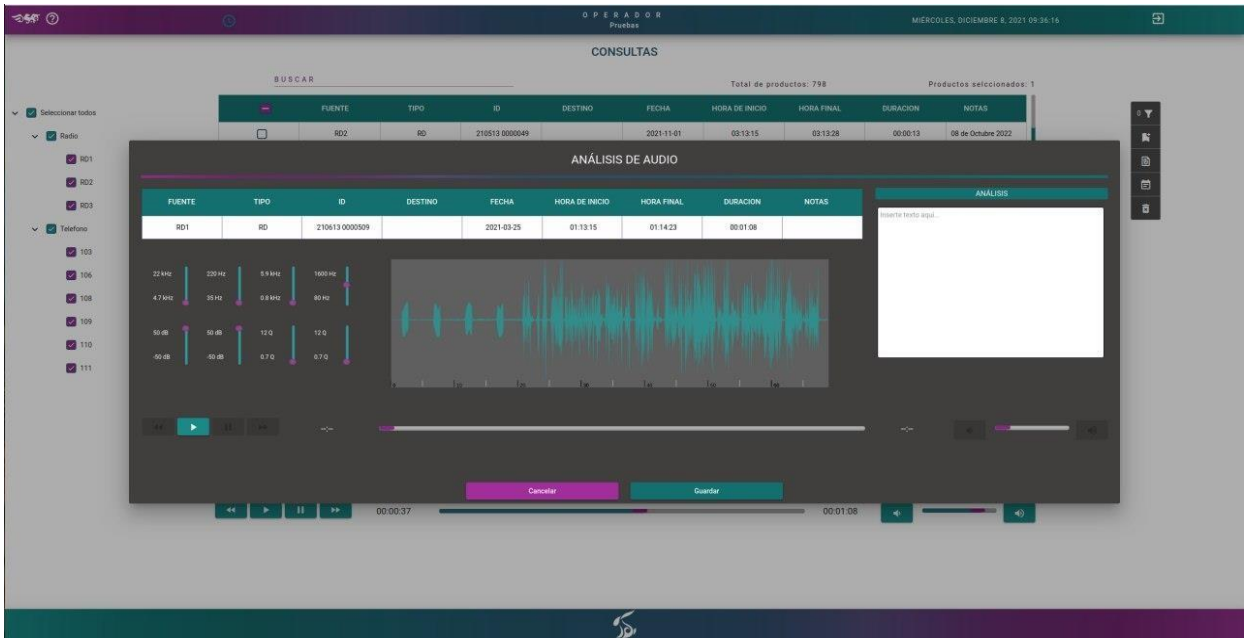


Ilustración 42. Espectro de frecuencias de un audio.

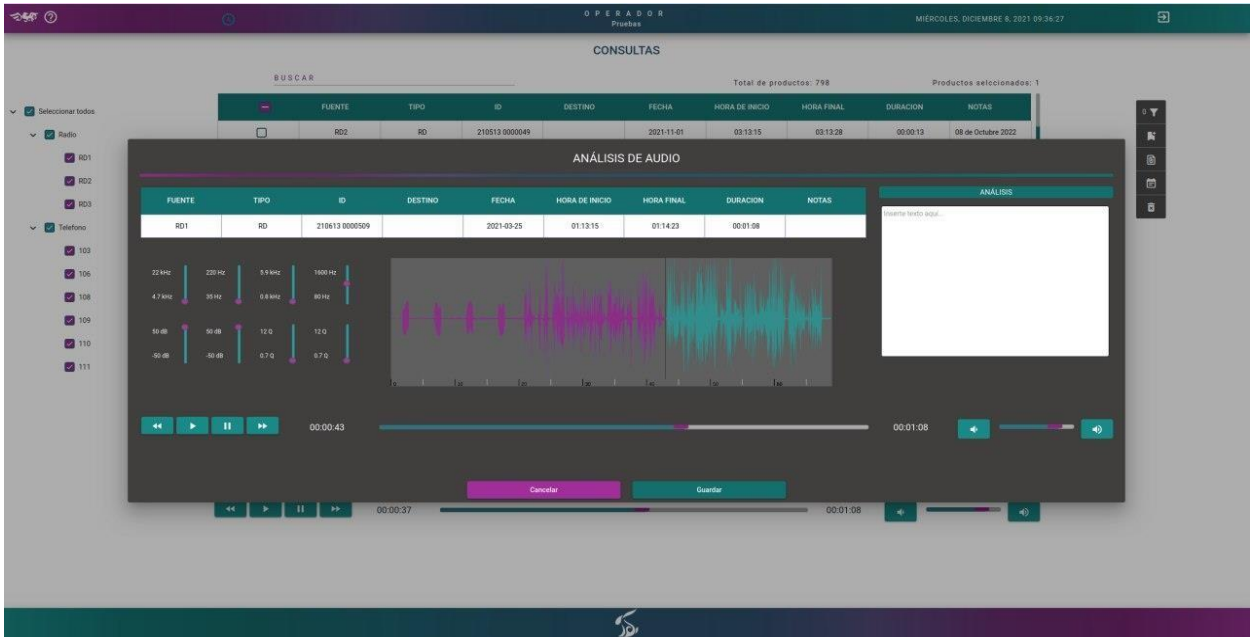


Ilustración 43. Visualización del espectro de frecuencia mientras se reproduce un audio.

Pruebas de validación

Para finalizar con la sección de resultados obtenidos es importante mencionar que esta aplicación aún no ha sido usada por la empresa que la solicitó, sin embargo, en cada Sprint Review se llevó a cabo una revisión de los nuevos puntos agregados y se recibió retroalimentación por parte del cliente. Al terminar de realizar todos los puntos requeridos al principio del proyecto, se entregó la aplicación y los accesos al cliente, además, se realizaron diversas pruebas de aceptación para verificar que se cumpliera con todos los elementos requeridos en el objetivo.

A continuación, se dará una descripción sobre ellas:

Pruebas unitarias

Este tipo de pruebas se realizaban en nuestras instalaciones con el equipo de trabajo, se verificaba que las nuevas implementaciones funcionaran correctamente, se evaluaban todos los módulos y componentes creados hasta ese momento tomando partes del código, validando que su comportamiento y funcionamiento fuera el esperado, regularmente quien llevaba a cabo las pruebas era un desarrollador diferente al creador de esa parte del software.

Pruebas de sitio

En estas pruebas el equipo de trabajo se reunía en las instalaciones del cliente, se instalaba la aplicación con la versión que se tuviera hasta ese momento y el cliente verificaba el correcto funcionamiento de todos los módulos con los desarrolladores como observadores.

Pruebas de aceptación

Consiste en verificar que el software cumple las expectativas desde el punto de vista del cliente y los usuarios finales, para ello, se usó una lista detallada de todas las funciones con las que contaba la aplicación para ese momento, en ésta misma se determinaba por parte del cliente si cierta función se ejecutaba correctamente o no y si era necesario se anexaban notas para mejorar o realizar algún cambio. Estas pruebas también se llevaron a cabo en las instalaciones del cliente.

Conclusiones

Al término del proyecto, la aplicación cuenta con todos los puntos que se plantearon en un inicio, ya que pasó todas las pruebas de validación hechas por parte del cliente, en cuanto a funcionalidad, requerimientos, necesidades y diseño.

Aunque a lo largo de mi carrera no conseguí profundizar en temas específicos relacionados al desarrollo web, tuve materias que me sirvieron mucho para entender el funcionamiento del sistema PIX, sobre todo la materia de bases de datos, porque así pude entender cómo se relacionó la información en las tablas.

Gracias a este trabajo pude profundizar en HTML, estructura y funcionamiento de hojas de estilo, Bootstrap y bases de datos, que eran temas que mis profesores mencionaban y siempre tuve curiosidad de aprender. A su vez, conocí muchas cosas nuevas como la sintaxis del formato JSON, el *framework* Angular, Angular Material y el lenguaje de programación typescript y como hacer peticiones a la base de datos por medio de una *API*. Todas son herramientas muy usadas en el campo al que me quiero dedicar y que me van a servir mucho para continuar con mi crecimiento profesional.

Tuve la oportunidad de corregir muchos fallos que tenía al momento de programar y de conocer muchas nuevas tecnologías que facilitan el desarrollo de una aplicación web. Logré utilizar muchas herramientas de *Frontend* que ya están implementadas y adaptarlas a mis necesidades para facilitarme y ahorrarme código repetitivo.

Fue un proyecto que se inició desde cero y eso me dio la oportunidad de estar presente en todos los procesos de su creación, desde la toma de requerimientos y el planteamiento hasta los resultados finales ya con todo el funcionamiento interno de la aplicación, pasando por todas las pruebas que se hacían al producto y resolviendo detalles que salían una y otra vez. A pesar de que ya había participado en otros proyectos de la empresa, siempre se aprende cosas nuevas.

Me enfrenté con muchas complicaciones desde el principio del proyecto hasta la aprobación por parte del cliente, tales como no saber hacer el *routing* en angular, que es necesario para poder comunicar componentes, o no saber implementar peticiones a la base de datos, pero gracias a que en este proyecto tuve que encargarme de esa parte pude aprender y practicar.

Algo que me gustó mucho de participar en el desarrollo de PIX es que gracias al panorama que me aportó me di cuenta de que el campo de la creación de aplicaciones web es muy amplio y que me gustaría dedicarme a alguna de sus ramas, por ejemplo, el desarrollo web o para dispositivos Android. Ya que descubrí que tengo mucho interés en eso y mi formación me orienta a ello.

Referencias

Angular Components Team. (s/f). *Angular material*. Angular Material. Recuperado el 18 de mayo de 2022, de https://material.angular.io/

DECSEF Sistemas S.A. DE C.V. (2022) DECSEF. Recuperado el 28 de noviembre del 2022, de https://www.decsef.com/#

Angular vs AngularJS: Principales Diferencias. Recuperado el 28 de noviembre del 2022, de CCWEB. https://www.comocrearunapaginaweb.com.mx/angular-vs-angularjs-diferencias/

API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. (2016, marzo 23). BBVA API_Market. Recuperado el 18 de julio de 2022, de https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el- desarrollo-de-proyectos/

Atlassian. (s/f). *Scrum*. Atlassian. Recuperado el 19 de mayo de 2022, de Recuperado el 30 de noviembre del 2022, de https://www.atlassian.com/es/agile/scrum

Blancarte, O. (s/f). *Arquitectura Cliente-Servidor*. Reactiveprogramming.io. Recuperado el 19 de mayo de2022, de https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

Blokdyk, G. (2018). *IBM docs: Complete self-assessment guide*. Createspace Independent PublishingPlatform.

Canelo, M. M. (2020, noviembre 2). *¿Qué es la Programación Orientada a Objetos?* Profile SoftwareServices. Recuperado el 18 de noviembre del 2022, de https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/

Cano, C. (2017). *La Arquitectura REST*. Recuperado el 12 de noviembre del 2022, de http://www.tsgroup.com.co/wps/portal/tsg/blog/detalle-blog/la-arquitectura-rest

Ciclo de vida del software: todo lo que necesitas saber. (s/f). Intelequia. Recuperado el 19 de mayo de 2022, de https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

Componentes. (s/f). Gitbook.io. Recuperado el 18 de octubre de 2022, de https://ngchallenges.gitbook.io/project/componentes

CRUD: la base de la gestión de datos. (s/f). IONOS Digitalguide. Recuperado el 19 de mayo de 2022, de https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/crud-las-principales-operaciones-de-bases-de-datos

Dios Miguel, A. (s/f). *Scrum: qué es y cómo funciona este marco de trabajo*. Wearemarketing.com. Recuperado el 18 de mayo de 2022, de https://www.wearemarketing.com/es/blog/metodologia-scrum- que-es-y-como-funciona.html

digite. (2021, mayo 18). *Pruebas De Aceptación: El Qué Y Porqué + Los Tipos Que Hay Que Conocer*. Digite. Recuperado el 22 de mayo de 2022, de https://www.digite.com/es/agile/pruebas-de-aceptacio

FACULTAD DE INGENIERÍA GUÍA PARA PRESENTAR UN INFORME DE TRABAJO PROFESIONAL COMO MODALIDAD DE TITULACIÓN. (s/f). Unam.mx. Recuperado el 16 de mayo de 2022, de https://www.ingenieria.unam.mx/pdf/Guia_InformeProfesional_9SEP.pdf

Fernández, Y. (2019, agosto 23). *API: qué es y para qué sirve*. Xataka.com; Xataka. Recuperado el 19 de mayo de 2022, de https://www.xataka.com/basics/api-que-sirve

Generalidades del protocolo HTTP. (s/f). Mozilla.org. Recuperado el 18 de mayo de 2022, de https://developer.mozilla.org/es/docs/Web/HTTP/Overview

Handa, U. (2021, octubre 27). *7 razones para usar angular para su aplicación web en 2022*. Cynoteck. Recuperado el 17 de mayo de 2022, de https://cynoteck.com/es/blog-post/reasons-to-use-angular-for-your-web-app/

Huambachano, J. F. (s/f). *¿qué es Scrum?* Scrum.Org. Recuperado el 19 de mayo de 2022, de https://www.scrum.org/resources/blog/que-es-scrum

Introducción a Angular Material. (2021, enero 19). Tribalyte Technologies. Recuperado el 16 de mayo de 2022, de https://tech.tribalyte.eu/blog- introduccion-angular-material

JSON. (s/f). Json.org. Recuperado el 18 de mayo de 2022, de https://www.json.org/json-es.html

Kickidler. (s/f). *Top 7 de los mejores sistemas gratuitos de monitoreo de empleados del 2022*. Kickidler. Recuperado el 18 de mayo de 2022, de https://www.kickidler.com/es/info/top-7-de-los-mejores-sistemas-gratuitos-de-monitoreo-de-tiempo-de-trabajo-y-control-de-empleados.html

Martínez, C. V. (2011). *Seguridad Informatica*. Recuperado el 18 de mayo de 2022, de http://www.ii.unam.mx/es-mx/AlmacenDigital/CapsulasTI/Paginas/seguridadinformatica.aspx

Métodos de petición HTTP. (s/f). Mozilla.org. Recuperado el 19 de mayo de 2022, de https://developer.mozilla.org/es/docs/Web/HTTP/Methods

Mousinho, A. (2019, noviembre 6). *HTML: ¿qué es y por qué es tan importante en el mundo digital?* Rock Content – ES. Recuperado el 19 de mayo de 2022, de https://rockcontent.com/es/blog/html/

Pàmpols, M. (s/f). *Arquitectura REST. Qué es y para qué sirve*. Marcpampols.net. Recuperado el 19 de mayo de 2022, de https://www.marcpampols.net/es/arquitectura-rest-que-es-para-que-sirve/#:~:text=La%20arquitectura%20REST%20se%20basa,define%20la%20operaci%C3%B3n%20a%20realizar.

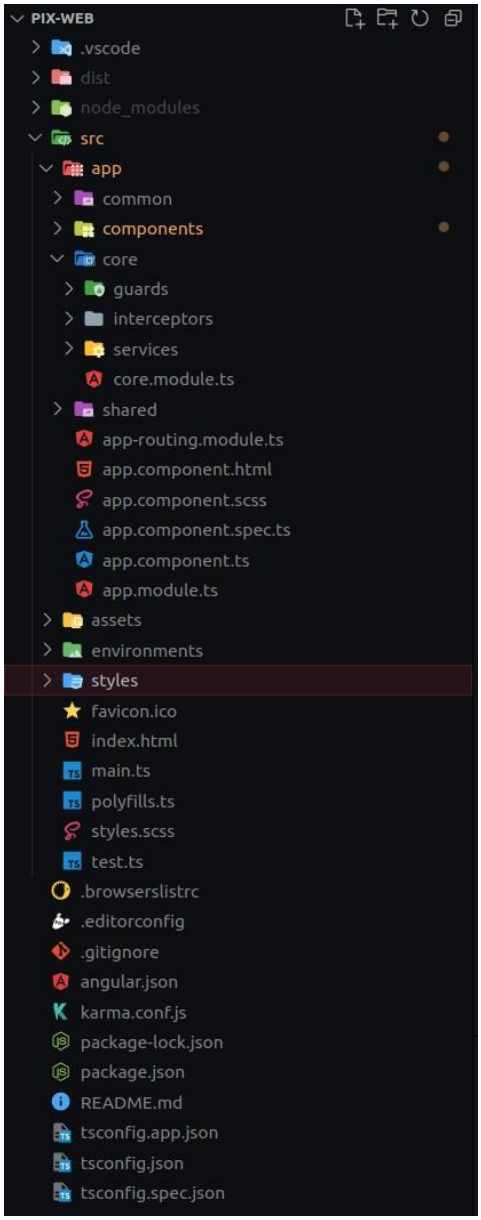
¿Qué es el JSON? (s/f). Oracle.com. Recuperado el 18 de mayo de 2022, de https://www.oracle.com/mx/database/what-is-json/

¿Qué es Material Design? (2019, febrero 26). Cleventy.com. Recuperado el 29 de mayo de 2022, de https://cleventy.com/que-es-material-design/

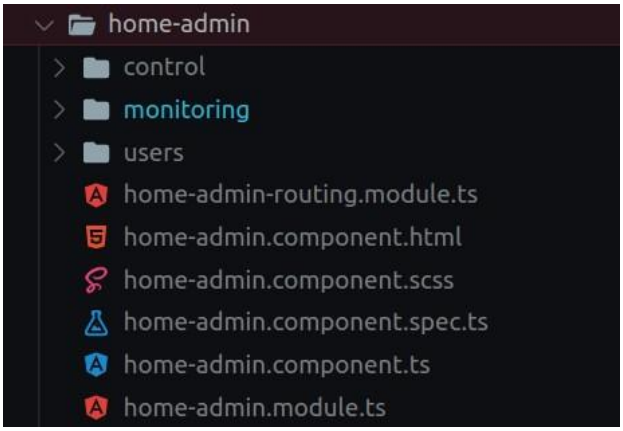
Villacampa, Ó. (2015, abril 16). *Qué es SASS y por qué los CSS pueden volver a divertirnos*. Ondho. Recuperado el 25 de mayo de 2022, de https://www.ondho.com/que-es-sass-y-por-que-los-css-pueden-volver-a-divertirnos/

Schiaffarino, A. (2019) Modelo cliente servidor: ¿Qué es? Características, Ventajas y Desventajas, Infranetworking. Recuperado el 25 de mayo de 2022, de https://blog.infranetworking.com/modelo-cliente-servidor/

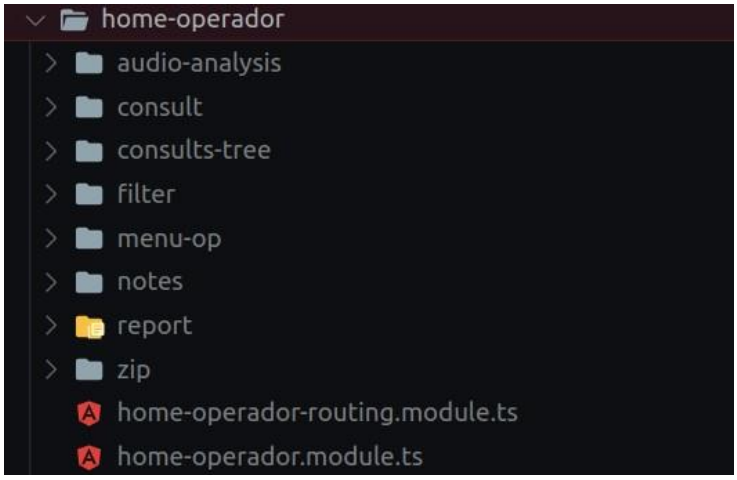
Anexos



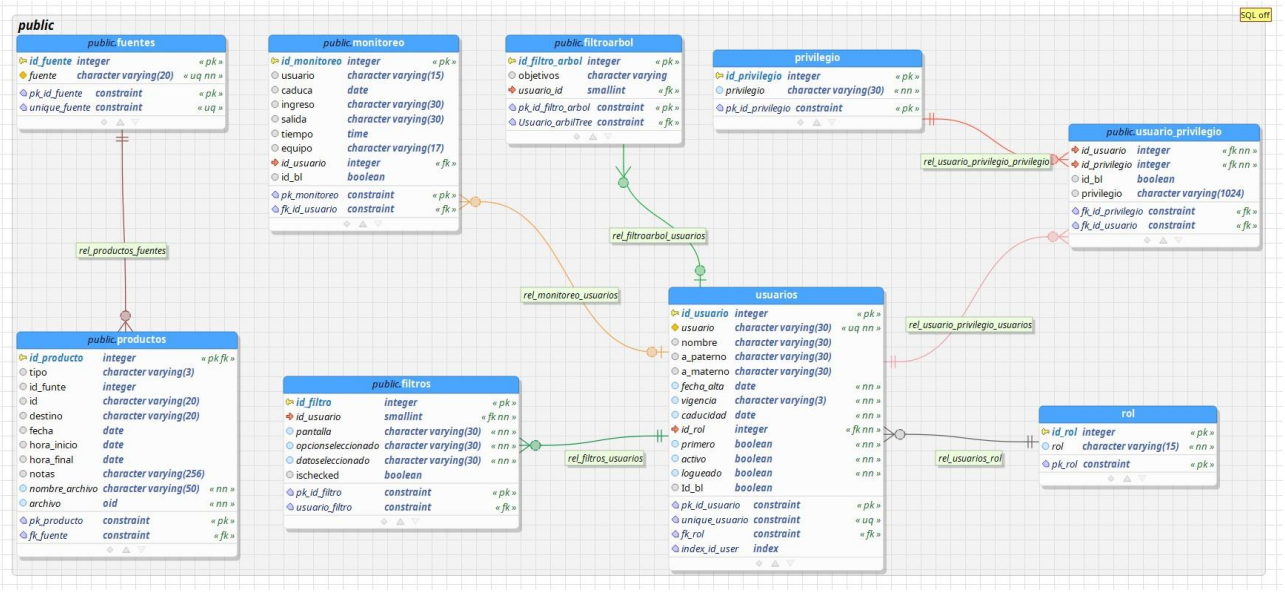
Anexo 1. Estructura de carpetas del proyecto PIX.



Anexo 2. Estructura de carpetas de sección Administrador.



Anexo 3. Estructura de carpetas de sección Operador.



Anexo 4. Arquitectura entidad relación de la base de datos de PIX