Federico Ezequiel Lozada

Aplicación web : conexión a brazo robótico guiado mediante visión artificial

2024

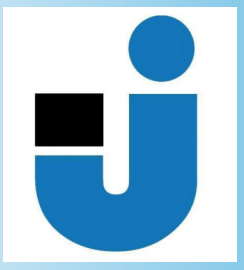
Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática



PRÁCTICA PROFESIONAL SUPERVISADA
Informe final

Aplicación web: conexión remota a brazo robótico guiado mediante visión artificial.

Federico Ezequiel Lozada

Florencio Varela, marzo y 2024

Estudiante

Federico Lozada
DNI: 38.325.610
Nº de Legajo: 3955
federico.lozada.94@gmail.com
Cantidad de materias aprobadas al comienzo de la PPS: 45
Práctica Profesional Supervisada (PPS) enmarcada en artículo 4 de la Resolución (CS) 103/16.

ORGANIZACIÓN DONDE SE REALIZA LA PPS

Universidad Nacional Arturo Jauretche
Av. Calchaquí 6200, Florencio Varela, (1888) Buenos Aires, Argentina
+54 11 4275 6100
Sector: Programa Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de interés social, Instituto de Ingeniería y Agronomía

Tutor por la organización

Prof. Mg. Osio, Jorge
josio@unaj.edu.ar

Docente supervisor por UNAJ

Ing. Salvatore, Juan
jsalvatore@unaj.edu.ar

Docente tutora por el Taller de Apoyo para la Producción de Textos Académicos

Lic. y Prof. Kelly, Carolina
kellygcarolina@gmail.com

Coordinador de la carrera de Ingeniería en Informática

Dr. Ing. Martín Morales
mmorales@unaj.edu.ar

Resumen

En el presente Trabajo Integrador Final se realiza una implementación de desarrollo web que conectará con un brazo robótico guiado mediante visión artificial.
Para llevar a cabo el mismo, en primer lugar se menciona el objetivo general y los específicos, seguido de las distintas tareas a ejecutar para poder crear la aplicación web.
En segundo lugar, se especifican las herramientas necesarias para dicha implementación, así como el marco teórico y la fundamentación del uso de las mismas.
En tercer lugar, se detalla la metodología que guía el desarrollo del presente trabajo, incluyendo más fundamentos de la implementación y arquitectura de la aplicación.
Luego, se hace mención de los diferentes diseños realizados y su implementación en los distintos ámbitos que fue realizada, comenzando con la arquitectura *back-end* y de base de datos, para luego poder proseguir con los diseños visuales del *front-end* estos divididos en dos ramas distintas, su aplicación web de escritorio y el diseño web para móviles.
Finalmente, se comentan los distintos inconvenientes a la hora de realizar el desarrollo, así como posibles mejoras tentativas que podría contener la arquitectura y ser un proyecto más robusto. Seguido de las conclusiones y reflexiones pertinentes.

- Palabras clave: Desarrollo web - arquitectura MVC - Sistema de conexión remota - Laravel - Vue

Abstract

In this Final Integrative Work, an implementation of web development is carried out that will connect with a robotic arm guided by artificial vision.
To carry it out, first the general objective and the specific ones are mentioned, followed by the different tasks to be executed in order to create the web application.
Secondly, the tools necessary for said implementation are specified, as well as the theoretical framework and rationale for their use.
Thirdly, the methodology that guides the development of this work is detailed, including more foundations of the application's implementation and architecture.
Then, mention is made of the different designs made and their implementation in the different areas that were carried out, starting with the back-end and database architecture, and then being able to continue with the visual designs of the front-end, these divided into two different branches, its desktop web application and mobile web design.
Finally, the different drawbacks when carrying out the development are discussed, as well as possible tentative improvements that the architecture could contain and be a more robust project. Followed by the relevant conclusions and reflections.

- Keywords: Web development - MVC architecture - Remote connection system - Laravel - Vue

Dedicatorias y agradecimientos

Quiero agradecer a Martin Morales, Carolina Kelly, Juan Salvatore y especialmente a Jorge Osio por el apoyo y guía durante la construcción de la PPS.
También quiero agradecer a Oscar Cortes Bracho por el apoyo brindado durante la carrera, así como la confianza que me hizo tener.
Gracias a Victoria Cáceres por todo el apoyo y aliento que fue brindando para poder continuar hasta el final en esta práctica.
Quiero agradecer a cualquiera que se haya cruzado en mi camino para ayudarme de manera desinteresada a completar esta última etapa de mis estudios: desde un consejo, una indicación, con un pequeño aporte o con una contribución vital.
Y, por supuesto, gracias a mi mamá Ana Finizio que siempre estuvo presente en cada momento y nunca faltó su apoyo y sabiduría para ayudarme a avanzar en la carrera hasta este punto.

Federico Lozada, Buenos Aires, 2024

Índice

Índice de figuras

1. Introducción

El presente trabajo forma parte de la Práctica Profesional Supervisada (PPS) dentro de la carrera de Ingeniería en Informática en la Universidad Nacional Arturo Jauretche (UNAJ) y tiene por objetivo general el desarrollo e implementación de una aplicación web para el sistema de control de un brazo robótico (*LeArm*) guiado mediante visión artificial (ESP 32). Esta tiene como objetivo principal vincular al usuario con la movilidad del brazo y su visión (cámara perteneciente al sistema embebido) dando al sistema la capacidad de realizar determinadas tareas y de ser controlado de forma remota desde una página web.

1.1. Objetivos

Generales:

- Desarrollar un *software* para la manipulación remota de un brazo robótico en un entorno determinado.
- Desarrollar una aplicación que permita el monitoreo y control remoto del brazo robótico.

Específicos:

- Poder iniciar sesión en la aplicación, y gestionar los usuarios desde la misma.
- Realizar peticiones al brazo robótico para que pueda moverse en todas las direcciones posibles.
- Recibir y visualizar la información de la cámara referente al brazo robótico.
- Evaluar las distintas medidas de seguridad a tomar para la conexión entre el brazo y la aplicación web.
- Analizar los máximos y mínimos de movimiento del brazo robótico para restringirlos.
- Analizar las diferentes funciones y librerías disponibles para el control de un brazo robótico.
- Diseñar un sistema de control, que interactúe con los sensores y manipule los mecanismos motores del robot.
- Desarrollar la aplicación web con el *framework* Vue que sea capaz de procesar las imágenes y enviar los correspondientes datos a través de los cuales recibirá instrucciones de dirección el brazo.

- Desarrollar una aplicación híbrida en un lenguaje de programación web que permita monitorear y controlar la dirección del brazo robótico de forma remota mediante el acceso a la cámara y a los comandos del robot para controlar sus cuatro movimientos básicos.
- Realizar distintas pruebas de funcionamiento al brazo y a la aplicación de control y monitoreo para validar la propuesta realizada.

1.2. Tareas a ejecutar

- Diseñar la arquitectura de la base de datos.
- Desarrollar el maquetado para la visualización de cada parte de la aplicación.
- Conectar los *frameworks* de *Back-end* con el *Front-end* y las funciones motoras del brazo robot.
- Desarrollar funciones para la creación y administración de acceso a la aplicación.
- Desarrollar funciones de control para evitar daños en el brazo robótico

2. Herramientas utilizadas

En el presente trabajo se busca la implementación de un desarrollo de aplicación web que se encuentre visible a través de una url en internet o sitio web. También incorpora las respectivas normas de seguridad para que el acceso sea restringido a las personas correspondientes que puedan usarla.

Un sitio web puede clasificarse de diferentes formas. Para cuestiones de desarrollo web principalmente se divide en dos partes:

- **Frontend**. Es la parte que interactúa con el usuario, tanto en imagen como en función. Por ello está íntimamente relacionada con la experiencia del usuario (UX) y la interfaz de usuario (IU).
- **Backend**. Se refiere a la parte que está en contacto directo con el servidor; es donde se aplica el código de programación para crear la estructura. Permanece en un segundo plano a cargo de la accesibilidad, actualización, bases de datos y cambios del sitio.

A menudo, para el desarrollo de estas aplicaciones se utilizan *Frameworks* y a estos se los puede denominar como un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de

plantilla que sirve como punto de partida para la organización y desarrollo de *software*. Utilizar *frameworks* puede simplificar una tarea o proceso y permite al desarrollador acelerar el trabajo y favorecer que este sea colaborativo, reducir errores y obtener un resultado de mejor calidad.

Por estas facilidades se optó en este proyecto por las siguientes herramientas:

- *Framework Front-end*: Vue versión 3.
- *Framework Back-end*: Laravel versión 9.
- Base de datos: Mysql.
- Placa ESP32 CAM
- Brazo robotico

Debido a que las mencionadas herramientas cuentan con estructuras de fácil aprendizaje, además de contar con una gran compatibilidad entre ellas y un sistema de seguridad robusto, esto condicionó a que fueran elegidas para este proyecto.

2.1. Vue

El *framework Front-end* Vue es utilizado debido a su accesibilidad, versatilidad y rendimiento.

- Accesibilidad: se basa en *HTML*, *CSS* y *JavaScript* estándar con *API* intuitiva y documentación de clase mundial.
- Versatilidad: cuenta con un ecosistema incremental adaptable que escala entre las librerías usadas y un marco completo de funciones del *framework*.
- Rendimiento: tiene reactividad real y un sistema de renderizado optimizado por compilador que rara vez requiere de una optimización manual.

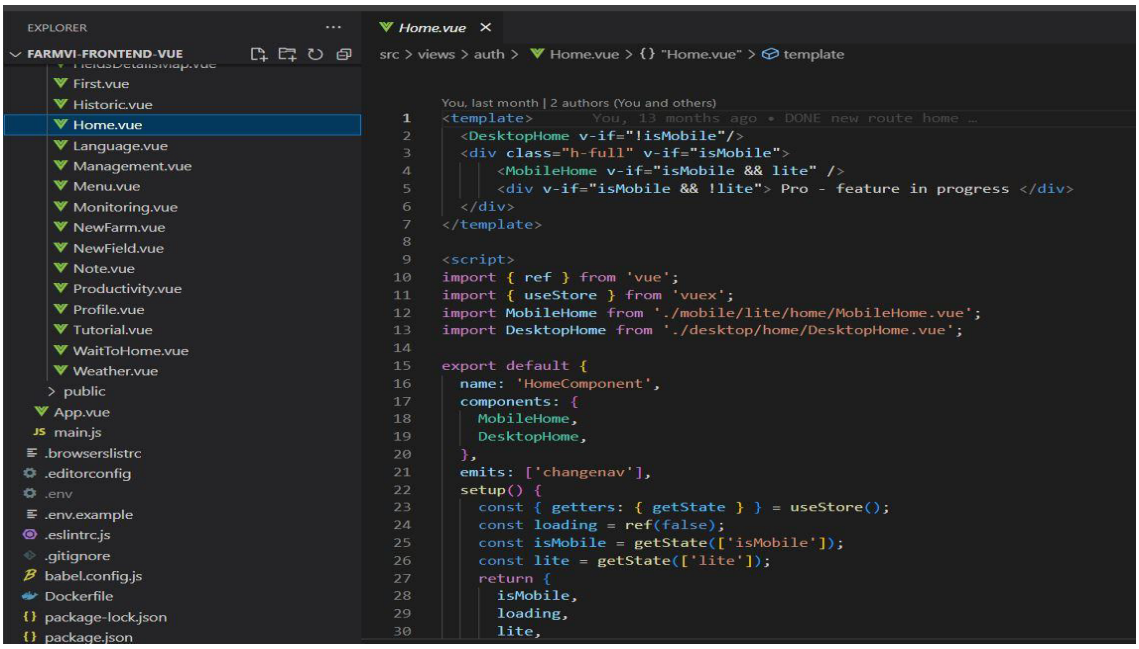


Figura 1. *Framework Vue* componente home.
Fuente: Elaboración propia, basada en la práctica.

2.2. Laravel

Laravel es un *framework PHP* para la creación de aplicaciones web con una sintaxis elegante y expresiva, que puede ser utilizada de forma monolítica o simplemente como una *API* (Interfaz de Programación de Aplicaciones o *Application Programming Interface*).

La *API* robusta que ofrece permite realizar una conexión inmediata y rápida con el *framework Front-end*, así como también incluye medidas de seguridad por sesión y una administración completa para la realización de consultas y actualizaciones a la base de datos.

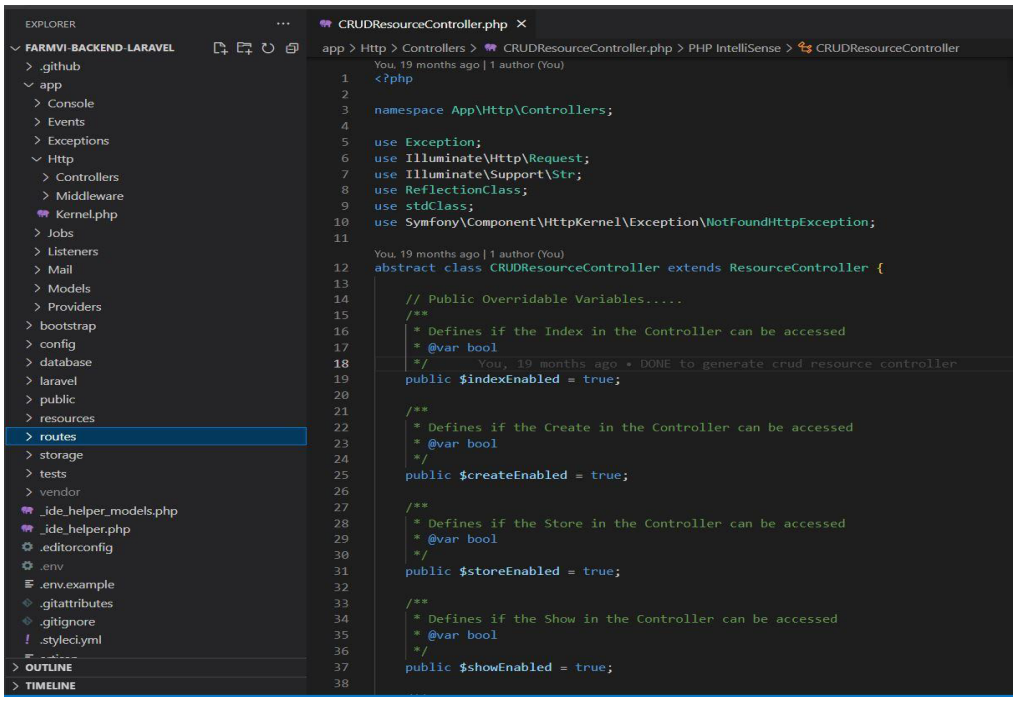


Figura 2. *Framework Laravel* controlador CRUD.
Fuente: Elaboración propia, basada en la práctica.

2.3. Mysql

Mysql es un potente sistema de base de datos relacional de objetos desarrollado bajo licencia dual y se encuentra considerado como la base de datos de código abierto más popular del mundo en desarrollo que le ha brindado una sólida reputación de confiabilidad, robustez en funciones y rendimiento.

Utiliza el lenguaje *SQL* combinado con muchas funciones que almacenan y escalan de forma segura hasta las cargas de trabajo de datos más complicadas. Todo esto debido a la arquitectura comprobada con la que cuenta: confiabilidad, integridad de datos, conjunto sólido de funciones, extensibilidad y la dedicación que la comunidad de código abierto del *software* ofrece en soluciones innovadoras y de alto rendimiento.

2.4. ESP32 CAM

Es un microcontrolador (MCU) con wifi integrado y conectividad *bluetooth* con altas prestaciones y de bajo costo. Al poseer un diseño robusto es capaz de funcionar fiablemente en entornos industriales o desfavorables.

Posee un diseño robusto capaz de funcionar en la industria de manera fiable. Expuesto a condiciones extremas cuenta con la capacidad de resolver de forma dinámica efectos indeseados sobre el *hardware* y soporta temperaturas entre los -40 °C y los +125 ° C. Debido a que fue diseñado para dispositivos móviles, electrónicos y portátiles con aplicaciones en Internet de las Cosas (IoT), posee un consumo de energía ultra bajo y cuenta con varios modos de funcionamiento en bajo consumo. Esp32 está integrado por antenas incorporadas, balun de RF (dispositivo conductor que convierte líneas de transmisión desbalanceadas en líneas balanceadas), amplificadores de potencia, amplificador de recepción de bajo ruido y módulo para el suministro de energía. Además de lo mencionado anteriormente, cuenta con gran versatilidad ya que posee mínimos requisitos para su Printed Circuit Board (PCB). También puede interactuar con otros sistemas y aplicaciones, proporcionando funcionalidades de wifi y *bluetooth* por medio de sus interfaces SPI/SDIO o I2C/UART.

Figura 3. Placa ESP32-CAM. Fuente: Recuperado de https://thinkrobotics.com/cdn/shop/products/Hb0a6a4bfe3b842bea9bec63be4eaf1c4d_720x.jpg?v=1640601323

2.5. Brazo Robótico

Hiwonder Co. Ltd, fundada en 2015 en Shenzhen, China, es un proveedor de soluciones integrales para *Science, Technology, Engineering and Mathematics* (*STEAM*) Education. Ofrece productos y servicios integrales para el desarrollo de *hardware*, *software* y diseño curricular para todas las edades. Es poseedor de más

de cuarenta patentes, *hardware* libre, robot biónico inteligente, brazos robóticos, kits IoT, robótica de inteligencia artificial, y muchos otros desarrollos más.

LeArm es un brazo robótico inteligente de escritorio de alto rendimiento, de aplicaciones con objetos livianos, destinado a la educación. Posee 6 servos que le permiten agarrar objetos en cualquier dirección. El brazo robótico también brinda soporte para programar aplicaciones en PC y para teléfonos móviles. Está constituido por una construcción totalmente metálica: garra de metal, soporte de aluminio y una base más grande de metal en la parte inferior. El diseño estructural 6DOF puede hacer que el brazo robótico se mueva de manera flexible, para que pueda tomar objetos en cualquier dirección. Fue implementado con servos digitales de alta precisión, que hace que el control sea más preciso. Utiliza un servo controlador de 6 canales con un módulo *bluetooth* 4.0 incorporado. Cuenta con varios métodos de control, proporcionados por un mando inalámbrico (joystick PS3), un *software* de escritorio para pc y una aplicación Android/iOS gratuita para dispositivos móviles inteligentes. También puede controlarse, lo que se hizo en este proyecto, a través de los pines Tx y Rx con aplicaciones ejemplo llamadas “*software* serial” (implementa el protocolo serie por *software*) y “*hardware* serial” (implementa el protocolo serie mediante el módulo de HW), además de los protocolos utilizados para realizar los movimientos simples del brazo.

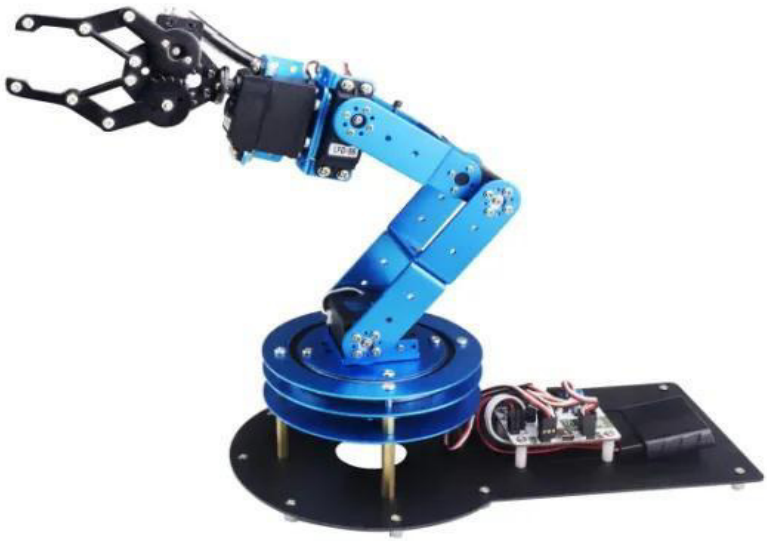


Figura 4. Brazo Robótico LeArm 6DOF. Fuente: recuperado de https://m.media-amazon.com/images/I/610tPYOvGHL._AC_SL1200_.jpg

3. Desarrollo de tareas

Para el desarrollo de las tareas definidas se utilizó una metodología ágil llamada *SCRUM* que segmenta la productividad en *sprints* o periodos de trabajo.

Una vez definida la metodología que organizó la manera de trabajar, se definieron las historias de usuario y las tareas que se realizarían en función a las historias.

Al finalizar las definiciones previas se prosiguió a realizar la arquitectura de la base de datos, eligiendo una base de datos relacional *Mysql*, para poder administrar el acceso de usuarios y guardar los registros que sean requeridos por el brazo robótico.

Al finalizar la arquitectura se procedió a implementarla desde *Laravel* donde se crearon las migraciones que modelizan físicamente la arquitectura teórica que se planteó.

En base a esto, se pudo comenzar a desarrollar las funciones principales para la petición y guardado de información en la base de datos. Es decir, la administración de usuarios (su creación e inicio de sesión a la aplicación).
Una vez se implementó el esqueleto principal del *framework* de *Back-end* se realizó el maquetado principal para que un usuario pueda ingresar, las secciones de ingreso y administración, tanto de usuario como del brazo robótico.

3.1. Historias de usuario

Para desarrollar las historias de usuario las mismas se dividieron en las secciones/módulos a realizar y dentro de estas se definieron qué tareas se realizarían:

- Módulo base de datos:
  Tarea 1: Arquitectura de la base de datos.
  Tarea 2: Creación de tablas y columnas.
  Tarea 3: Conexión con el *Back-end.*
- Módulo *Back-end*:
  Tarea 1: Creación del proyecto *Laravel*.
  Tarea 2: Migración para la creación física de la arquitectura de la base de datos.
  Tarea 3: Conexión con el *Front-end*.
  Tarea 4: Conexión con el brazo robótico.
  Tarea 5:Creación de funciones para el monitoreo y movimiento del brazo robótico.
- Módulo *Front-end*:
  Tarea 1: Plantilla visual para el inicio de sesión.
  Tarea 2: Generación del *layout* para visualizar la cámara del brazo robótico.
  Tarea 3: Botones con controles para el movimiento y agarre del *LeArm*.
  Tarea 4: Plantilla visual para la administración y creación de usuarios.

3.2. Arquitectura de base de datos

En la arquitectura se ha definido utilizar *MySql,* un gestor de base de datos relacional utilizado principalmente para la administración de usuarios, siendo esta la única tabla que se generó para el control de acceso a la aplicación. Esta no requiere

de ninguna otra tabla al no tener que realizar ningún tipo de guardado de información.

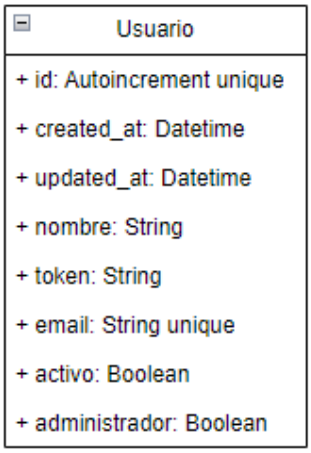


Figura 5. Modelo tabla de usuarios.
Fuente: Elaboración propia, basada en la práctica.

3.3. Arquitectura *MVC*

MVC es una propuesta de arquitectura de *software* utilizada para separar el código por sus distintas responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea muy concreta, lo que ofrece beneficios diversos.

MVC se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario, aunque en la práctica el mismo patrón de arquitectura se puede utilizar para distintos tipos de aplicaciones. Surge de la necesidad de crear un *software* más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, la reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, Model, Views & Controllers.

3.3.1. Modelo

Esta capa se encarga del manejo de los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los

tendremos habitualmente en una base de datos (Mysql en el caso de este proyecto), por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes *selects, updates, inserts*, etc.

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar distintos tipos de librerías como *PDO* o algún *ORM* (*Object relational mapping*) como *Doctrine*, que permite trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias *SQL*, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.

3.3.2. Vista

Esta sección del modelo se encarga de dar forma a la parte frontal de un sitio web o aplicación (*front-end)*, lo que incluye los fondos, colores, texto, animaciones o efectos. Es decir, las vistas contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario y que nos permitirá renderizar los estados de nuestra aplicación en *HTML*.

En las vistas generalmente se trabaja con los datos recibidos por los modelos generados, pero no se realiza un acceso directo a estos. Las vistas requerirán los datos a los modelos y de ellas se generará la salida, tal como nuestra aplicación requiera. Dado que se utilizará el *framework Vue* tenemos un dinamismo que, con la visual *HTML* renderizada, permite animaciones y el mantenimiento de los datos de una forma dinámica ante cada pantalla.

3.3.3. Controlador

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación (*Back-end*), como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace

entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

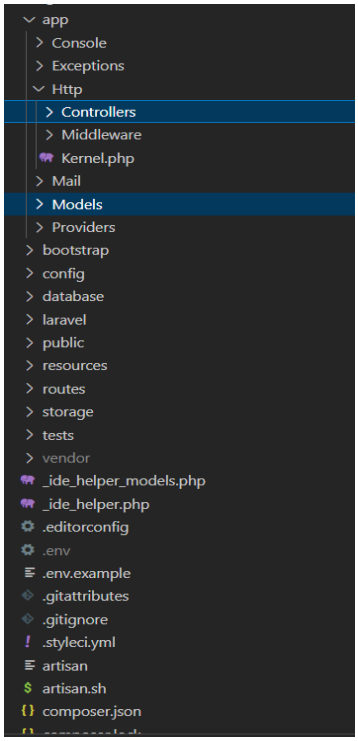


Figura 6. Estructura de carpetas en MVC.
Fuente: Elaboración propia, basada en la práctica.

4. Desarrollo

4.1. Back-end

La arquitectura de la aplicación *back-end* está basada en lo que se denomina “API REST”: una API es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el *software* de las aplicaciones. API significa interfaz de programación de aplicaciones. Es la forma por la que muchos sitios web o aplicaciones web obtienen la información. Estos realizan una petición a través de internet para obtener información de cierto recurso o *endpoint*, como los datos que se recibirán del brazo robótico. Ese “llamado” es dirigido hacia un servidor que está siempre activo esperando para ser consultado. El mismo devuelve la información y

se ve reflejada en el sitio. REST se compone de una lista de reglas que se deben cumplir en el diseño de la arquitectura de una API:

- Interfaz uniforme: los datos de los recursos que se solicitan, por ejemplo la administración de usuarios, serán suficientes para realizar todas las acciones necesarias, como pueden ser el alta de trámites, la edición y la eliminación. En otras palabras, cuando por la aplicación móvil se quiera realizar un trámite, se realizará una petición al servidor la cual le devolverá todos los datos necesarios para la creación del mismo.

- Peticiones sin estado: las peticiones que se realizan no almacenan ningún tipo de información entre ellas, esto proporciona un mayor rendimiento.

- Cacheable: esta característica permite guardar en memoria ciertas respuestas de datos que no cambian en un corto lapso de tiempo. Por ejemplo, los datos del usuario que sabemos que no cambiarán o hay muy pocas posibilidades que lo hagan en un periodo corto. Esto permite que la información llegue más rápido y además no se sature el servidor de peticiones.

- Separación de cliente y servidor: esta separación corresponde a la división de la página web que realiza consultas a los servidores para obtener información.

- Sistema de Capas: el cliente puede estar conectado mediante la interfaz al servidor o a un intermediario, para él es irrelevante y desconocido. El uso de capas o servidores intermedios puede servir para aumentar la escalabilidad (sistemas de balanceo de carga, cachés) o para implementar políticas de seguridad.

Además de API REST, la estructura del sistema manejará MVC, aunque solamente estará actuando con dos partes del manejo MVC: los modelos y controladores, que se comunicarán activamente con el *front-end* interactuando como el servidor al que apunta el cliente/usuario.

4.1.1. Modelo Usuario

Administrativamente se utilizarán las funciones predeterminadas de *Laravel* para la conexión a la base de datos relacional que son generadas directamente sobre el modelo para insertar nuevos registros de usuarios. Estos registros se podrán crear, editar e inhabilitar desde el *layout* de *front-end* que se creará.

La clase Usuario cuenta con distintos ítems denominados “variables” que se pueden ver en la figura 7:

- $table: indica la tabla de la base de datos a la que apunta el modelo.
- $timestamps: le indica a *Laravel* si el modelo cuenta o no con fechas de estado (*created_at* y *updated_at*), dos columnas que sirven para la realización de auditorías y control.
- $fillable: los campos/columnas de la tabla que pueden ser utilizados desde el *frontend* para crear un nuevo registro o editar. Esta variable denominada $fillable es propia del *framework* Laravel y es usada como un factor de seguridad, que permite designar qué columnas de la tabla pueden ser modificadas de forma masiva, inhabilitando la edición de campos que no se encuentren en esta variable.

Los tipos de usuario se basarán en la columna “administrador” siendo esta de tipo *Boolean*, es decir, de verdadero o falso. En base al valor que tome se mostrarán distintos perfiles:

- Usuario administrador: tiene acceso completo a ver la cámara del brazo robótico y sus funcionalidades, así como la creación, inhabilitación y edición de usuarios.
- Usuario general: cuenta solamente con acceso a la cámara del brazo robótico con las funcionalidades de movimiento.

Además de estos permisos particulares por usuario, ambos tipos contarán con accesos a sus perfiles y la edición tanto de sus datos personales como las contraseñas que han sido asignadas.

```php
User.php M
app > Models > User.php > PHP IntelliSense > User
You, 19 seconds ago | 1 author (You)
<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Carbon\Carbon;
use Hash;

You, 18 seconds ago | 1 author (You)
class User extends Authenticatable
{
    //INIT CONSTANTS
    const SUPERUSER = 1;
    //END CONSTANTS
    protected $table = 'usuario';        You, 1 second ago • Uncommitted changes
    public $timestamps = true;

    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'nombre',
        'token',
        'email',
        'password',
        'token',
        'activo',
        'administrador',
    ];
```

Figura 7. Modelo de usuario como clase *back-end* de *Laravel*.
Fuente: Elaboración propia, basada en la práctica.

4.1.2. CRUD controller

CRUD (*Create*, *Read*, *Update*, *Delete*) es un acrónimo de las maneras en las que se puede operar sobre información almacenada. Es un mnemónico para las cuatro funciones del almacenamiento persistente.

Este componente es una clase abstracta que permite la realización de un *CRUD* completo de cualquier modelo solo realizando la extensión del mismo en los

controladores requeridos. Esta abstracción permite la realización de funciones generales sin necesidad de crearlas:

- *Index*: función que brinda un listado paginado del modelo elegido, es decir, si usamos como ejemplo el modelo de usuario, nos brindará el listado completo de usuarios paginado/dividido de a 25 usuarios al mismo tiempo.
- *Show*: brinda la información completa del usuario que se elija, en base a su ID o identificador único, lo que puede dar una información más detallada del modelo elegido.
- *Store*: función que realiza la creación/inserción de un nuevo registro según el modelo definido.
- *Update*: edita/modifica una o más columnas de un registro existente a través de su ID.
- *Destroy*: permite borrar uno o varios registros del modelo elegido en el controlador.

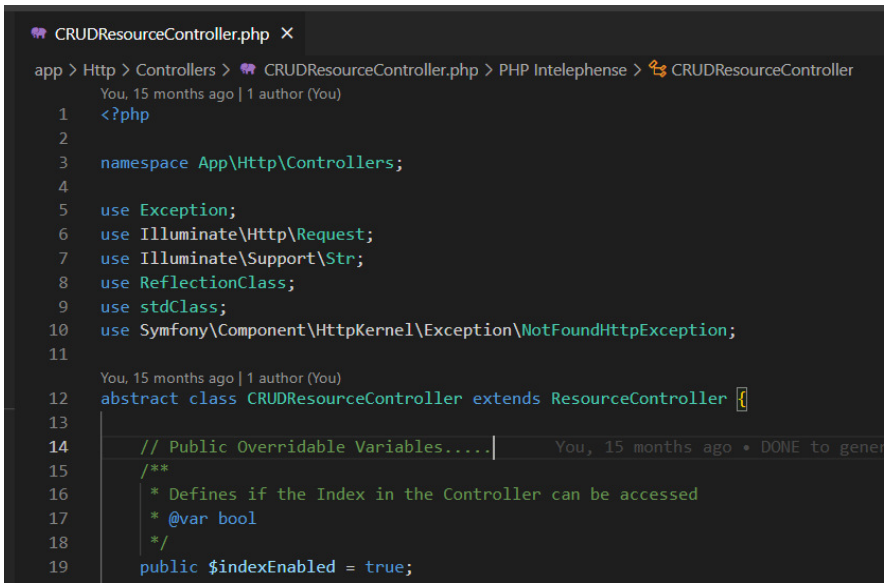


Figura 8. Clase abstracta CRUD para generación de funciones automáticas.
Fuente: Elaboración propia, basada en la práctica.

4.1.3. Middleware

El *middleware* es un *software* con el que las diferentes aplicaciones se comunican entre sí. Brinda funcionalidad para conectar las aplicaciones de manera inteligente y eficiente, de forma que se pueda innovar más rápido. Este actúa como un nexo entre tecnologías, herramientas y bases de datos diversas para que pueda integrarlas sin dificultad en un único sistema. Este sistema único provee un servicio unificado a sus usuarios. Por ejemplo, una aplicación web *frontend* de Windows envía y recibe datos desde un servidor *backend* de Linux, pero los usuarios de la aplicación no están al tanto de la diferencia.

Además de actuar como intermediario entre aplicaciones de *software*, los programas del *middleware* también llevan a cabo las siguientes funciones:

- Proveen un canal de comunicación seguro entre aplicaciones distribuidas para que los sitios web envíen información confidencial de forma segura a las aplicaciones *backend*.
- Administran la fluidez del tráfico y evitan abrumar a una aplicación o un servidor en particular.
- Automatizan y personalizan las respuestas a las solicitudes. Por ejemplo, el *middleware* clasifica y filtra los resultados antes de enviarlos a la aplicación frontend.

Por temas de seguridad para el inicio de sesión, se utiliza un *middleware* para definir las distintas rutas que serán públicas y privadas, evitando de esta forma que cualquier acceso no autorizado pueda ser realizado y un usuario ajeno a los administrados no pueda ingresar. En la figura 9 podemos visualizar el middleware *Onlylogged* el cual sirve como un intermediario en las rutas privadas y verifica en cada petición que se le haga a la página web al recibir el *header authorization* si quien está enviando esta petición es un usuario vigente en el sistema y pueda permitirle el ingreso solicitado.

```php
You, 11 months ago | 1 author (You)
class OnlyLogged
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle($request, Closure $next)
    {
        $token = explode(' ', $request->header('authorization'));
        if($request->header('authorization') !== null){
            $token = $token[1];
            try{
                $user = User::where(['api_token'=>$token])->firstOrFail();
                Auth::login($user);
            }catch(Exception $error){}
        }
        if(Auth::check()){
            return $next($request);
        };
        errorReturn(403,['message'=>'Your not logged in']);
    }
}
```

Figura 9. *Middleware* para rutas privadas (solo usuarios logueados).
Fuente: Elaboración propia, basada en la práctica.

4.2. *Front-end*

La arquitectura *front-end*, además de manejar la parte API REST del cliente, también usará la estructura MVC, aunque solamente estará actuando como la tercera parte faltante del *back-end*: las vistas, las cuales se comunicarán activamente interactuando con el servidor a medida que el cliente/usuario lo requiera.

Una gran parte del desarrollo *front-end* se basa en el diseño web, la actividad creativa enfocada en realizar la parte visual, encargada de *layouts* e interfaces para un sitio web o una app, y realiza una configuración visible y funcional para el usuario al distribuir y conceptualizar diversos elementos. Además se encarga de definir la apariencia estética del sitio web (*UI user interface*) y qué tan amigable es la aplicación con el usuario (*UX user experience*).

En el diseño web se encuentran distintos tipos de procedimientos:

- Fijo o estático: diseño en el que no hay mucha interacción y suele emplearse mayormente para transmitir información, por lo que en ningún caso debería

de usarse para vender u ofrecer algún tipo de servicio que se deba contratar por la web.

- Dinámico: se trata de un diseño más complejo en el que hay un mayor desarrollo y supone una mayor inversión para el cliente. A diferencia de los diseños fijos o estáticos, en los dinámicos el usuario puede interactuar con la información que se encuentre en el sitio.

- Responsivo o *responsive design*: es la implementación para que el *layout* pueda verse desde un ordenador, una tableta o un teléfono móvil sin importar cuál sea el tamaño de la pantalla. Con un *responsive web design*, el texto se ajusta y las imágenes se escalonan para que no haya inconvenientes al visualizarlas, sin importar el dispositivo en el que el usuario escoja ver la aplicación.

- Líquido o fluido: se comporta de manera similar a un diseño *responsive*, excepto que no manipula el diseño de la página según el tamaño de la pantalla. El diseño en sí reduce o estira toda la página web para que se ajuste al tamaño de la ventana elegida.

- Adaptativo: puede llegar a generar confusión con el diseño *responsive*, y es que el diseño responsivo se adapta literalmente mientras que el diseño adaptativo genera *layouts* distintos para cada dispositivo, sean teléfonos móviles, tabletas, aplicaciones de escritorio.

Este proyecto utilizará el diseño web adaptativo al ver que la aplicación web de escritorio tendría un mejor alcance a la hora de mostrar la información enviada por la cámara, mientras que se harán diseños adaptativos para toda pantalla móvil. Para explicar la diferenciación de estos *layouts*, en los siguientes puntos del desarrollo, se detallarán los diseños de cada uno de los mismos con su respectiva versión *mobile* y *desktop.*

4.2.1. *Desktop*

Las versiones de escritorio cuentan con una mayor capacidad informativa al poder contener en la pantalla más información, lo cual brinda una mejor oportunidad a la

hora de diseñar permitiendo generar más campos de los que podría obtener otro dispositivo.

Es una gran ventaja cuando se habla de funcionalidades para el usuario o en la muestra de información de reportes.

4.2.1.1. *Layout desktop login*

La primera pantalla que puede ver el usuario es el *login* o Inicio de sesión. El diseño minimalista del *layout* corresponde a una imagen del modelo de brazo robótico que se estará operando, así como también solicitará los datos de ingreso (email y contraseña)

Figura 10. *Layout login* de escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

El usuario puede, desde la opción “olvidaste tu contraseña”, recuperar la misma, desde una nueva pantalla que mantendrá el hemisferio izquierda del modelo de brazo robótico, y actualizará la información a solicitar del hemisferio derecho,

indicando dónde se encuentra e indicando que ingrese el email del usuario para que se genere el recupero de la contraseña.

Figura 11. *Layout* olvidar contraseña desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

Una vez el usuario haya ingresado el email y avanzado en la pantalla, se le enviará por *back-end* a su email los datos necesarios para recuperar su contraseña generando una nueva. Esta última instancia se debe a motivos de seguridad, ya que las contraseñas en base de datos deben de ser encriptadas, es decir, deben de estar ocultas y no ser de público conocimiento.

Enviando el email, se genera un control estricto, en el cual solo el receptor podrá ingresar a la página generada con el token de seguridad para poder realizar el cambio de contraseña desde donde el usuario pueda tener el control, ya sea en la parte *mobile* como en la de escritorio.

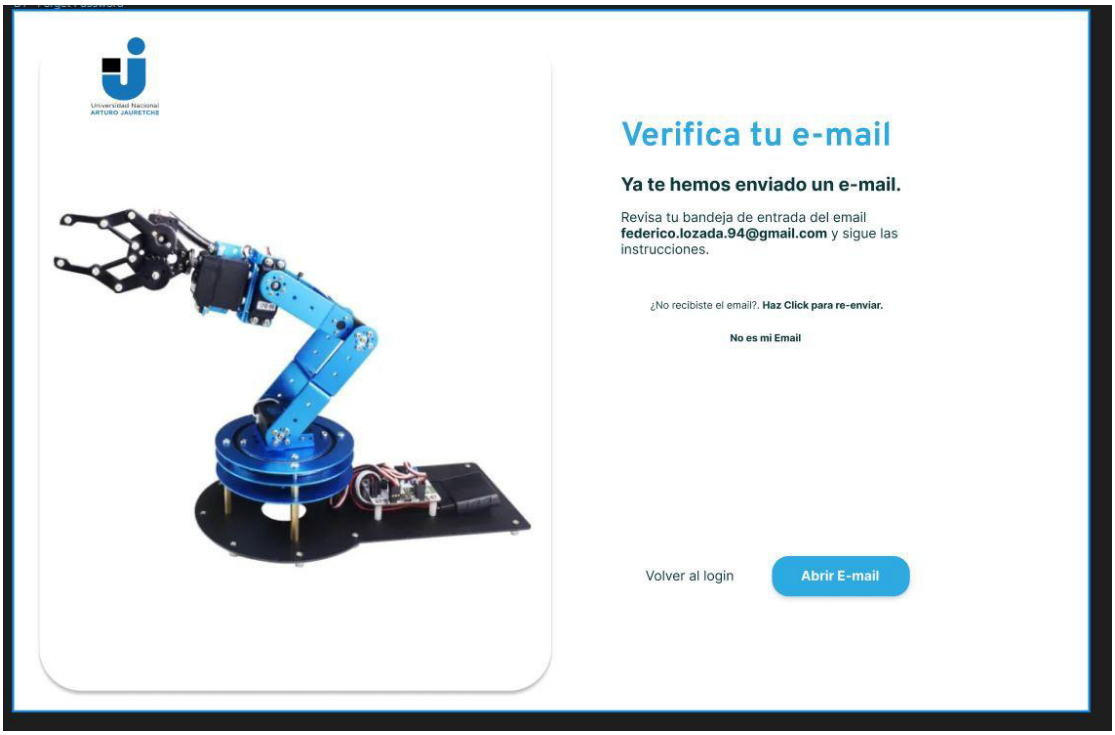


Figura 12. *Layout* email enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

El diseño de email enviado permite que si el usuario cometió algún error tipográfico pueda regresar a la pantalla anterior realizando click en “No es mi Email” así como también le permite reenviar el email en caso de haber fallado la primera vez o no haberlo recibido en su bandeja de entrada.

4.2.1.2. *Layout desktop Reset Password*

En esta ventana de la aplicación web el usuario podrá realizar el reseteo de su contraseña si, como se indicó en el punto anterior, indicó que ha olvidado el *password*; al recibir el email de confirmación recibirá en el contenido un *link* por el cual podrá ingresar en este *layout* y proseguir con el reseteo de contraseña

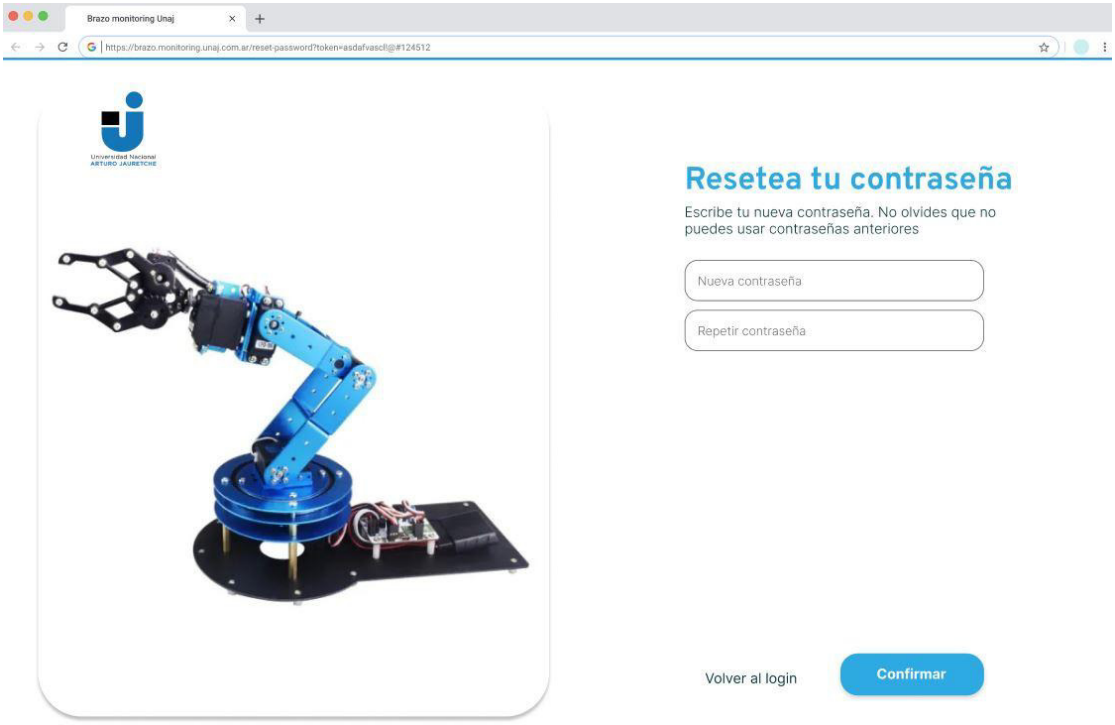


Figura 13. *Layout reset password* enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

Desde la figura 13 se puede visualizar que la ruta del explorador contiene el *query param* token recibido desde el link del email enviado: este identificador tiene que ser válido y es necesario para poder ingresar al *layout* debido a que sin este token la pantalla redireccionará a la persona que intente ingresar al *layout login* por no contar con una autorización apropiada.

En la pantalla podremos ver dos *inputs* y dos botones:

- Nueva contraseña: es el campo que solicita el ingreso del nuevo *password* que tendrá el usuario.
- Repetir contraseña: es un *input* de seguridad que sirve como verificación para evitar errores de tipeo del usuario, corroborar que la contraseña ingresada anteriormente no contenga fallos.

Una vez corroborados los datos ingresados en “Nueva contraseña” y “Repetir contraseña”, el usuario podrá clickear en la opción “Confirmar” para proceder con el cambio. Ante cualquier error que pudiera haber se le presentará una alerta a la

persona, solicitando los cambios necesarios para proceder.

Si todos los datos se encuentran en orden podrá avanzar a la siguiente pantalla.

Figura 14. *Layout* confirmación *reset password*.
Fuente: Elaboración propia, basada en la práctica.

Desde esta última visualización, el usuario puede corroborar con el mensaje desplegado en pantalla que su contraseña ha sido modificada con éxito y puede realizando click en “Volver al *login*” regresar al *layout* principal para iniciar sesión y comenzar a monitorear el brazo robótico o administrar los usuarios.

4.2.1.3. *Layout desktop Home*

Una vez el usuario ingresa su email y contraseña, e ingresa en su sesión, ya puede ingresar al panel principal denominado “*home*” o inicio. Este *layout* permite al usuario monitorizar la imagen que el brazo robótico está enviando. Así como también le brinda la posibilidad de poder moverlo en distintas direcciones, rotar, abrir y cerrar la garra.

Figura 15. *Layout* home enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

Como se puede ver en la figura 15, gran parte de la pantalla es utilizada para la visualización de la cámara del brazo robótico, que aún no tiene conexión por lo que despliega un texto para indicar lo que debería mostrar.

También podemos encontrar el botón "analizar imagen", el mismo es utilizado para enviar una petición al proyecto de inteligencia artificial de reconocimiento de elementos de reciclaje, el cual debería de generar toda las peticiones correspondientes para que el brazo robótico realice los movimientos necesarios, según la imagen enviada.

En este *layout* también se encuentra la barra de navegación a la izquierda de la pantalla, que acompaña el mismo diseño en cada sección. Desde este navegador se puede ir a los *layouts* de "*home*", "usuarios" y "perfil" respectivamente, si es que el usuario cuenta con los permisos.

4.2.1.4. *Layout desktop Usuarios*

En la barra de navegación, al seleccionar la ruta/ícono de usuarios, esta guiará al usuario administrador que cuenta con permisos de visualización al *layout* de

“Usuarios”, el cual contará con un listado completo de los usuarios vigentes en la aplicación y los distintos tipos de leyenda por columna:

- Nombre: indicado en el usuario que fue generado por un administrador.
- Email: método de inicio de sesión que le fue indicado al administrador que lo asignó.
- Administrador: recuadro verde o rojo que indica respectivamente si el usuario es administrador o no.
- Acciones: leyenda generada que le permite editar o borrar el usuario de la fila.

En base a estas columnas, el administrador que haya ingresado en este *layout* puede realizar las distintas acciones sobre cada usuario vigente de manera individual. Además de estas contemplaciones, se encuentra disponible un botón en la esquina inferior derecha denominado “crear usuario”, el cual permite, tal como lo indica su nombre, generar un nuevo usuario que podrá administrar o solamente tener una visual de la cámara del brazo robótico para poder moverla de manera remota.

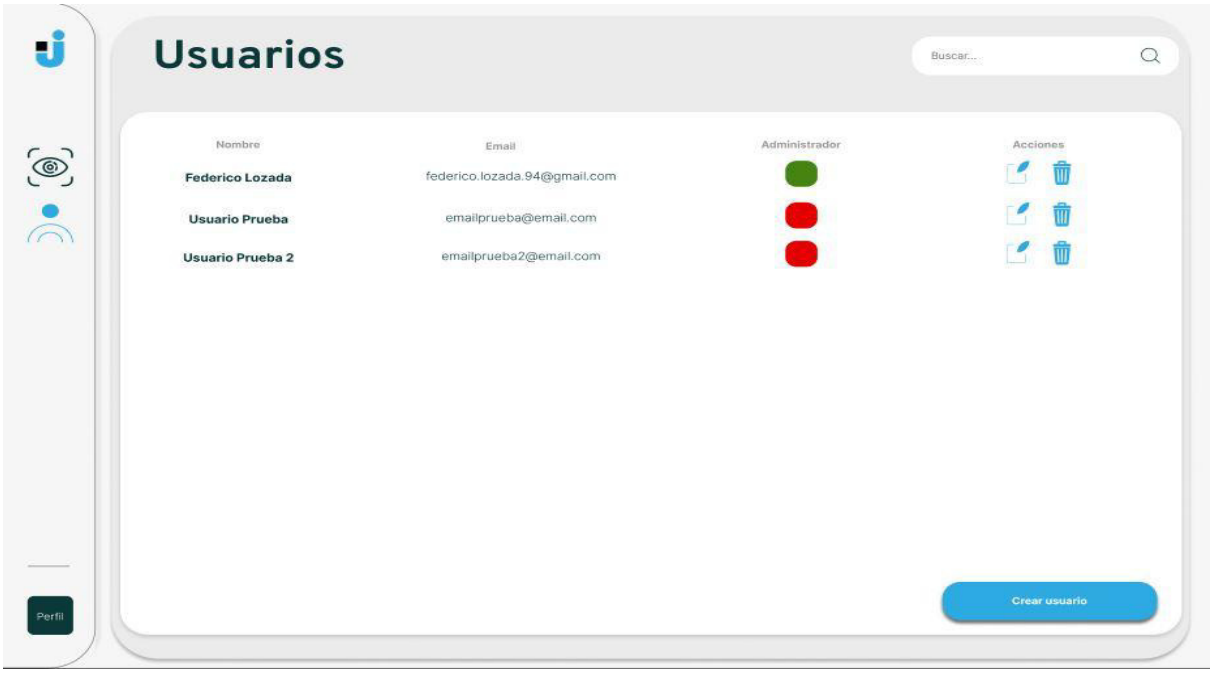


Figura 16. *Layout* Usuarios enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

Como se puede observar en la Figura 16, es un diseño minimalista, evocado para la fácil interpretación del mismo y no generar margen de errores a la hora de generar/editar/borrar usuarios. Además, como una funcionalidad extra, se puede observar en la esquina superior derecha un buscador, el cual permite al administrador filtrar el listado mostrado por nombre o email.

4.2.1.5. *Layout desktop Crear/Editar usuarios*

Al seleccionar la acción editar o el botón “crear usuario” desde el *layout* de “Usuarios”, se desplegará una nueva visualización la cual contará con distintos ítems que podrán ser completados o modificados dependiendo de si el *layout* fue ingresado a través de la acción o el botón de creación y luego tendrá en la esquina inferior derecha dos botones.

Los items serian:

- Nombre: que contará con un *inputbox* con el *placeholder* “Ingrese nombre del nuevo usuario”, en caso de ingresar desde la versión de crear usuario, o contará con el nombre del usuario a editar, como por ejemplo “Federico Lozada”, el cual podrá ser modificado si la situación lo requiere.
- E-mail: segundo *inputbox* con un nuevo texto por defecto que indica “Ingrese email del usuario” en la creación del usuario, o tendrá identificado el email que podrá ser modificado, como por ejemplo “federico.lozada.94@gmail.com”: si este tuviera errores de tipeo o se encontrara fuera de uso, el administrador podría editarlo fácilmente.
- Contraseña: esta sección se encuentra deshabilitada para el administrador mostrando, por seguridad, el texto “**********” si se ingresa en este *layout* para la edición de un usuario; en caso de realizar una creación, estará mostrando el *placeholder* “Genere la contraseña”.
- Generar Contraseña: es un botón que crea una nueva contraseña si se ingresa desde la sección de generar un nuevo usuario. Este *password* generado de forma automática será copiado al *clipboard* del sistema operativo usado del administrador, para que le sea más sencillo enviar esta contraseña generada al usuario que ha creado una vez se contacte con el. Hay que destacar que aunque el administrador pueda contactar al usuario, al mismo le estará llegando al email indicado su usuario y contraseña. Este

botón se encontrará deshabilitado en la sección de editar, debido a que una vez el usuario ingrese por primera vez su contraseña será modificada y solo podrá ser recuperada por ese mismo usuario.

- Administrador: botón de color rojo o verde que indicará respectivamente si el usuario es o no un administrador; al realizar click en el *checkbox* de color, este cambiará al opuesto, de esta forma se podrán editar los permisos de un usuario al editarlo, o designar en la creacion del usuario qué tipo de perfil este podrá visualizar.

Este *layout* cuenta con dos botones en la esquina inferior derecha:

- Botón rojo: este botón con la leyenda “Cancelar” funciona como un retorno para regresar al *layout “Usuarios”*, en caso de haber cometido un error al ingresar para generar un nuevo usuario, o seleccionado editar por error en un usuario diferente del cual se pensaba realizar una edición.
- Botón Azul: cuenta con dos leyendas distintas según desde donde se ha ingresado. Si el ingreso fue a través de crear usuario la leyenda será “confirmar”, como se puede ver en la figura 15. En el caso de editar un usuario vigente se podrá ver la leyenda “editar”, como indica la figura 16.

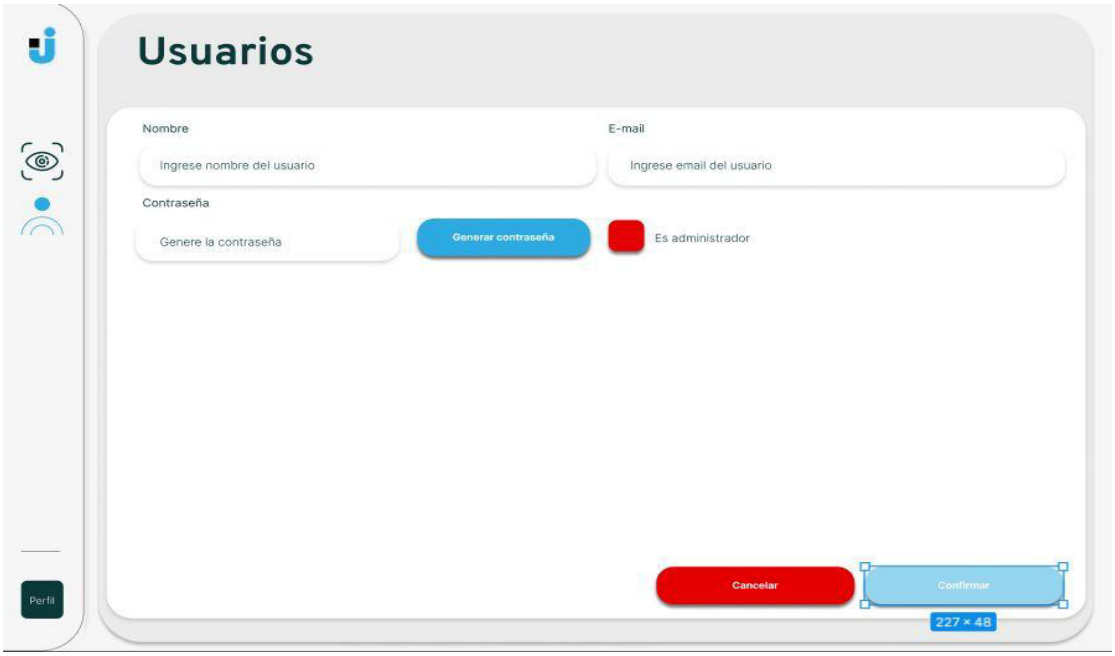


Figura 17. *Layout* crear usuario enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

En la figura 17 se puede ver el estado inicial del *layout* “Crear Usuario” cuando no hay ningún otro dato ingresado por el administrador, teniendo los *placeholders* indicados en los ítems y botones detallados en esta sección. Se puede observar como detalle adicional que el botón “confirmar” se encuentra deshabilitado, este solo se encontrará habilitado una vez el administrador ingrese los campos obligatorios (nombre, email y generar contraseña).

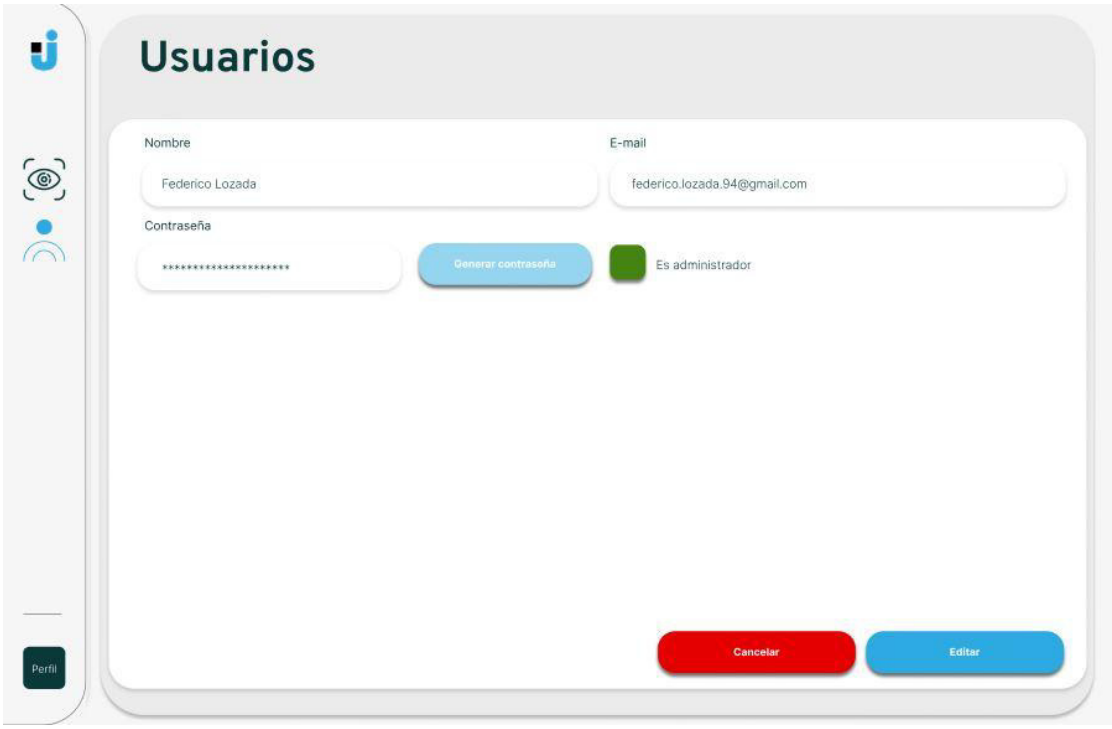


Figura 18. *Layout* editar usuario enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

Esta figura 18 nos muestra el estado inicial de este *layout*: una vez seleccionada la acción editar desde la visualización de usuarios, esta pantalla mostrará los campos del usuario indicado, que estarán a disposición para ser modificados y una vez clickeado el botón “Editar” se realizarán todos los cambios realizados.

4.2.1.6. *Layout desktop eliminar usuarios*

Al seleccionar la acción “eliminar” desde el *layout* de “Usuarios”, el mismo desplegará un *Pop-up*, que deshabilita cualquier otra acción con un fondo oscuro transparente y le indicará al administrador confirmar la decisión que ha realizado. Genera un *layout* en el centro de la pantalla con 3 micro secciones:

- Primera sección: título del *layout* que le indica al usuario si está seguro de que desea eliminar el usuario seleccionado
- Segunda sección: parte central de la visualización, mostrando los campos nombre y email, para poder verificar si no hubo algún error al intentar realizar la acción de eliminar, permitiendo al usuario administrador revisar la información de lo seleccionado y evitando así lo que se conoce como errores humanos a la hora de realizar esta acción de eliminar usuario.
- Tercera sección: contando con dos botones al pie del *layout*, uno de color rojo para cancelar la accion debido a errores de seleccion o cambios de opinión del administrador, y el segundo botón de color azul con la leyenda “confirmar”, que, una vez clickeado, se procederá a la eliminacion en la base de datos del registro seleccionado.

Una vez realizado el click en el botón confirmar de la tercera sección, el administrador podrá ver que se removerá del listado al usuario seleccionado y una notificación que le indicara que todo ha procedido de forma correcta y sin inconvenientes.

Figura 19. *Layout* eliminar usuario enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

El *Pop-up* que se puede visualizar en la figura 19 funciona como un paso preventivo que puede ayudar al administrador a evitar que se eliminen usuarios indiscriminadamente de manera masiva, así como solventar cualquier error de clickeo que el administrador pueda tener y no pueda realizar el borrado de un registro sin antes confirmarlo. De esta manera se generará un *log*/registro en la base de datos de que el administrador eliminó al usuario, para que pueda ser identificado a través de algún tipo de auditoría. Con esta funcionalidad se logra mantener un registro amplio en la administración de usuarios y se mantiene la integridad en la base de datos de los mismos.

4.2.1.7. *Layout desktop perfil*

Desde el navegador el usuario puede realizar click en el icono denominado “perfil”, que despliega un nuevo *layout,* donde el usuario puede ver sus datos principales, cerrar su sesión o modificar su password o información personal.

Dentro de esta sección se pueden ver dos botones y 3 *inputs*:

- Nombre: muestra el nombre del usuario y una vez se hace click en el botón “Editar Información” se convierte en un *input* editable, el cual permite la modificación y su guardado.
- E-mail: Es el *input* del email del usuario, es el mismo usado para poder ingresar al sistema. Una vez se realiza la acción de “Editar Información” se le permite realizar la modificación del mismo. Como un sistema de seguridad, este *input* cuenta con una validación para evitar que se puedan duplicar emails con otro usuario, ya que esta columna se encuentra generada con la restricción *unique* en la base de datos.
- *Password*: despliega en una pantalla de forma codificada el *input* de la contraseña del usuario, bloqueando su visualización por temas de seguridad. Este *input* no es editable: si se desea cambiar aquella, una vez se ingrese a “Editar información”, se generará un botón nuevo que permitirá realizar la edición.

- Cerrar sesión: botón ubicado en la esquina superior derecha que permite al usuario salir de la aplicación web una vez finalizada la actividad.
- Editar información: botón localizado en la esquina inferior izquierda que una vez clickeado habilita la visualización de tres nuevos botones y genera que los *inputs* nombre y email sean editables.

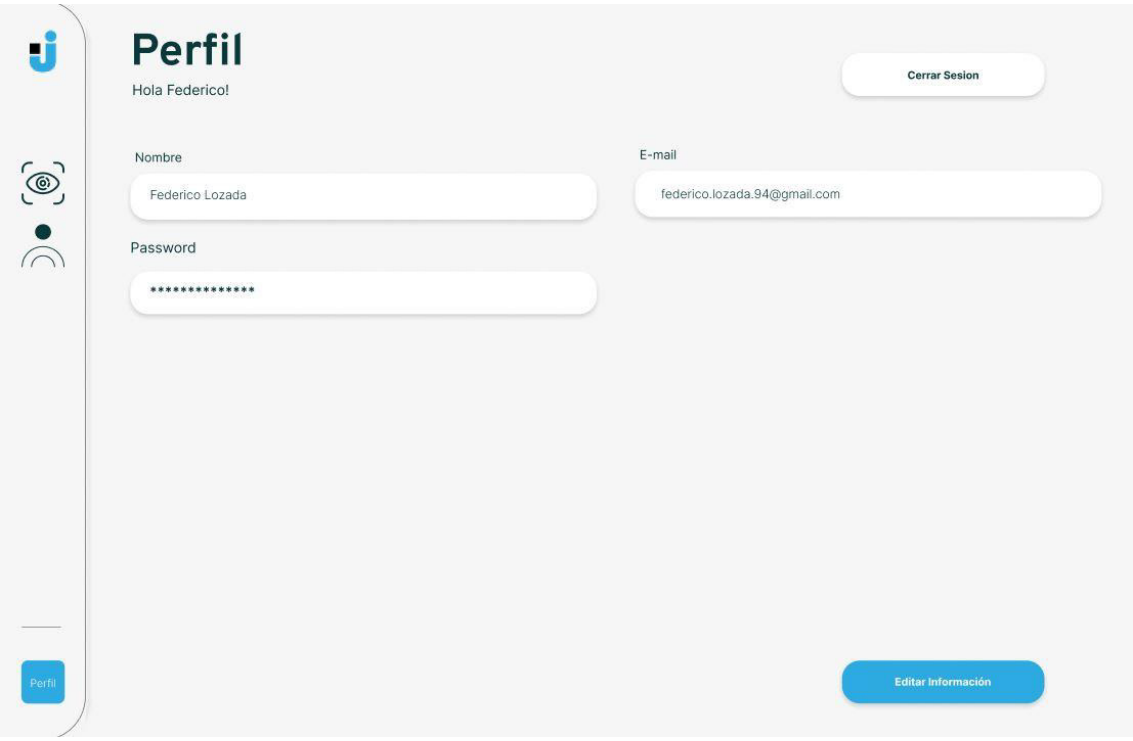


Figura 20. *Layout* perfil del usuario enviado desde el escritorio de la página web.
Fuente: Elaboración propia, basada en la práctica.

Según lo comentado en los ítems anteriores, se puede ver en la figura 20 los distintos *inputs*, el botón “Cerrar Sesión” que reenvía al usuario al *layout* de “*Login*” si se realiza la acción.

El usuario puede elegir desde la vista del perfil ir a cualquier punto de navegación con la barra izquierda o en realizar la acción del botón “Editar información” para que el *layout* cambie y se muestren las nuevas acciones para poder modificar los *inputs* y confirmar estos cambios; entre estas acciones se generan tres botones nuevos:

- Confirmar: procede a guardar la información modificada en los *inputs* asociados y regresa al *layout* original de perfil.

- Cancelar: retrocede todos los cambios realizados en los *inputs*, mostrando la ventana original de la aplicación.
- Cambiar contraseña: por temas de seguridad se implementó este botón adicional, que genera un *pop-up* o ventana emergente que requiere información adicional para poder modificar la contraseña, sin errores de tipeo o por equivocación del usuario.

Una vez el usuario realiza la acción del botón “Cambiar contraseña” se desplegará una ventana emergente o *pop-up*, generando un fondo negro difuminado que evitará al operario realizar alguna modificación por fuera de la ventana emergente.

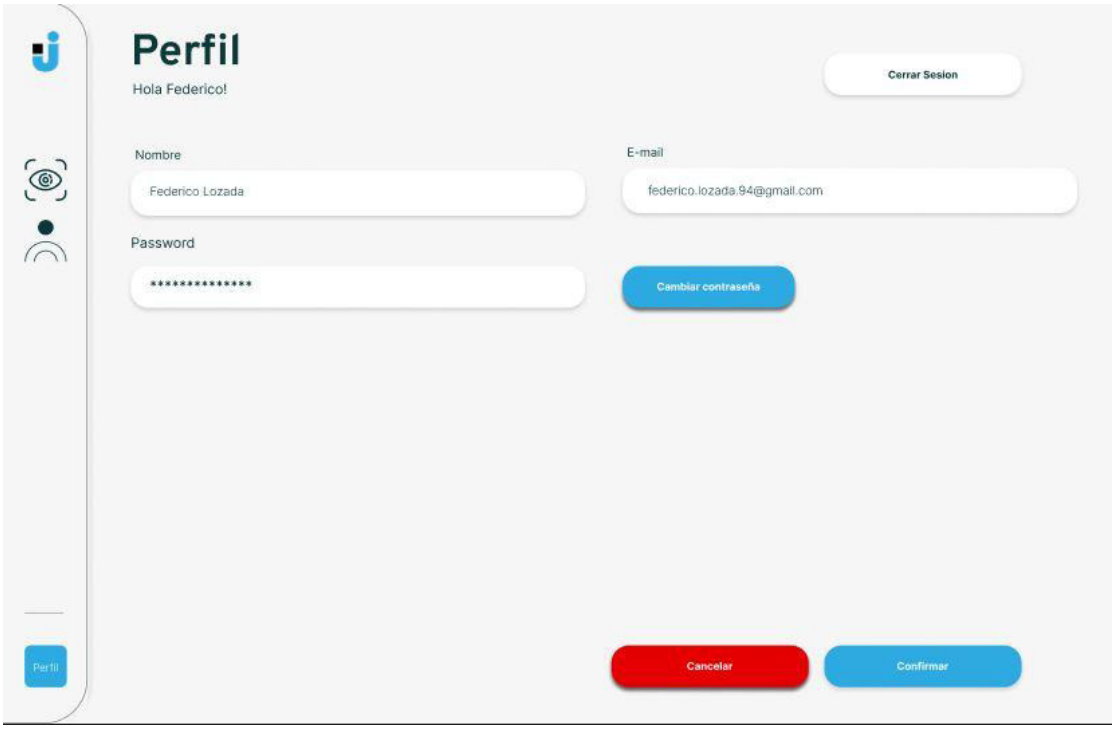


Figura 21. *Layout* editar perfil del usuario enviado desde la aplicación web.
Fuente: Elaboración propia, basada en la práctica.

En el centro aparecerán tres *inputs* nuevos con sus respectivos *labels*:

- Confirmar contraseña actual: pide el ingreso de la contraseña vigente del usuario, como seguridad, para verificar que no sea un operante ajeno a la aplicación intentando realizar una modificación no autorizada.

- Nueva contraseña: es el *input* que solicita el nuevo *password* que el usuario quiera para ingresar a la aplicación la próxima vez que inicie sesión.
- Repetir contraseña: requiere que repita lo ingresado en el campo anterior para corroborar que el usuario no cometa errores de tipeo y pueda confirmar que la contraseña que ha elegido escribir concuerde a la perfección.

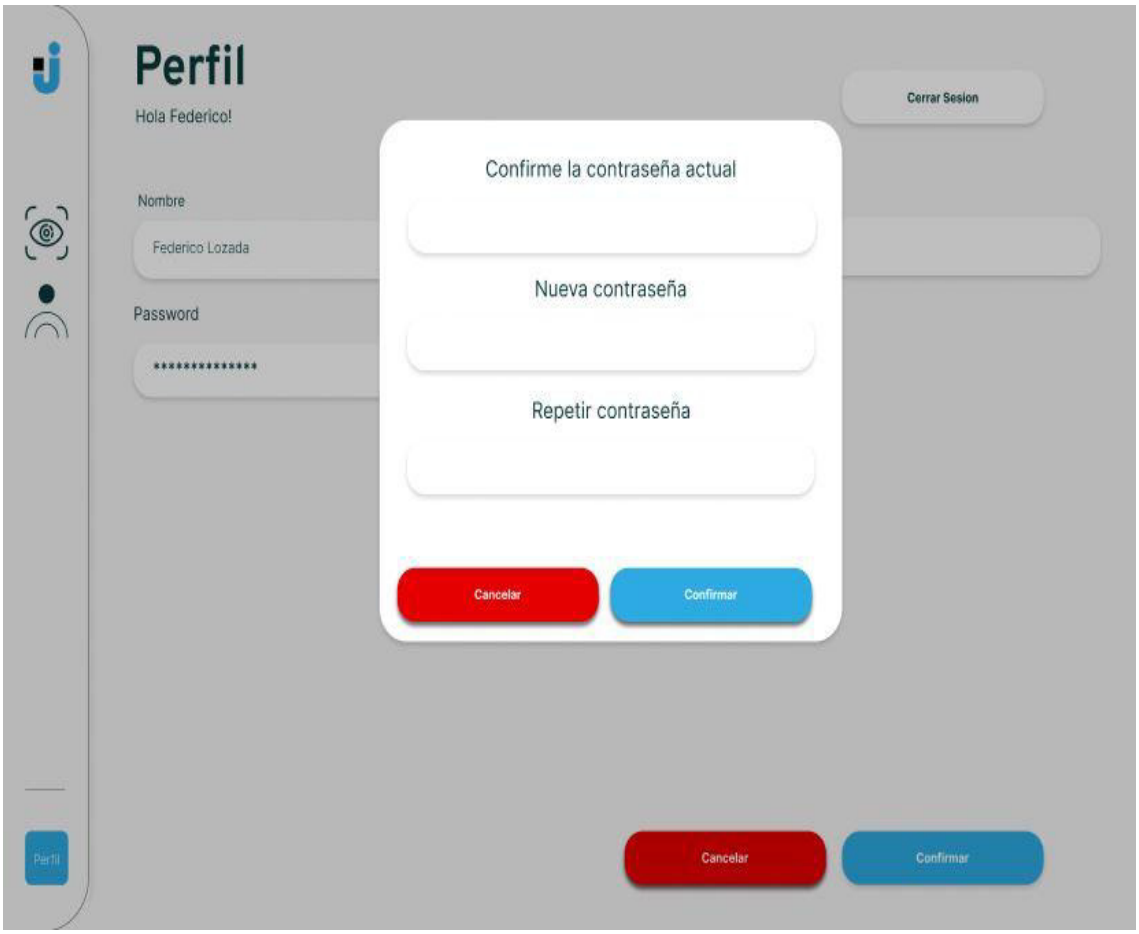


Figura 22. *Layout* editar *password* del usuario enviado desde la aplicación web.
Fuente: Elaboración propia, basada en la práctica.

Una vez el usuario realiza la acción del botón “Confirmar”, si se encuentran todos los campos completados correctamente, se generará una notificación indicando que el cambio de contraseña ha sido exitoso; en caso contrario, el usuario recibirá una alerta que le indicará qué campos cuentan con error para que pueda verificarlos.

4.2.2. *Mobile*

Las versiones *mobile* usualmente cuentan con una menor capacidad informativa al tener límites por los distintos tamaños de pantalla, lo cual brinda *layouts* minimalistas, lo que puede ocasionar que si la información a mostrar cuenta con un tamaño considerable no pueda mostrarse por completo en las pantallas generadas en la versión de escritorio. Esto implica dividir en sub-pantallas de acceso o *scrolls* con toda la información a mostrar .

Es una gran ventaja cuando se habla de practicidad para el usuario en la que se requiera el uso o manejo de la aplicación de una manera urgente y no se cuente con acceso a un equipo de escritorio.

4.2.2.1. *Layout mobile login*

La primer pantalla que puede ver el usuario desde el dispositivo es el *login* o Inicio de sesión; el diseño minimalista del *layout* corresponde al logo de la facultad con el nombre del modelo de brazo robótico que se estará operando, así como también solicitará los datos de ingreso (email y contraseña)

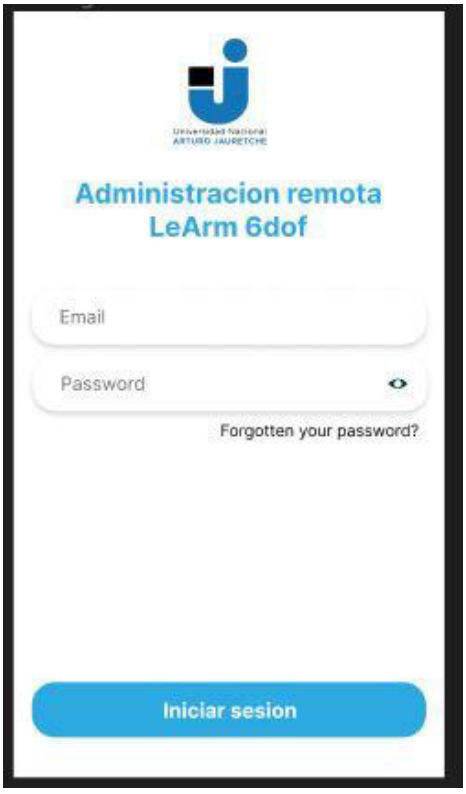


Figura 23. *Layout mobile login* del usuario enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Se pueden apreciar las diferencias sencillas con el *layout* de *desktop*, al tener menos espacio, ya que se elimina la previsualización de la imagen del brazo robótico que está siendo administrada. El usuario puede ingresar los campos de email y contraseña para avanzar hacia la pantalla denominada "*Home*" donde podrá monitorizar la cámara del brazo robótico y moverlo según sea requerido o, en caso de marcar la opción "olvidaste tu contraseña", proceder a la visualización de pantalla que le permitirá realizar las modificación de la contraseña para poder ingresar a la aplicación.

Figura 24. *Layout mobile* olvide contraseña enviada desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Este *layout* también cuenta con diferencias con su contraparte de escritorio y busca indicarle al usuario en dónde se encuentra, destacando el texto "Olvidaste tu

contraseña", con un *input* central que solicita el ingreso del email del usuario para proceder con el envío de un token para que pueda el usuario gestionar la modificación de la clave olvidada. Este campo central verifica que se esté escribiendo correctamente la sintaxis de un correo electrónico válido para proceder; una vez ingresado, se procede a indicar que el email ha sido enviado y el usuario puede corroborar los datos, para ver si ha tenido algún error de tipeo y volver al paso anterior en caso de que así sea.

Figura 25 *Layout mobile* olvide contraseña enviada desde la aplicación.
Fuente: Elaboración Propia, basada en la práctica.

Este diseño le permite al usuario que pueda regresar a la pantalla anterior realizando un click en "No es mi email" así como también le permite reenviar el correo electrónico en caso de haber fallado la primera vez o no haberlo recibido en su bandeja de entrada. El *layout* también cuenta con dos botones, uno para regresar a la pantalla de inicio de sesión y otro para poder abrir la aplicación de correo predeterminada del dispositivo.

4.2.2.2. *Layout mobile Reset Password*

En esta ventana de la aplicación *mobile* el usuario podrá ingresar el reseteo de su contraseña al haber recibido el link para ingresar desde el email enviado por la pantalla de “olvide contraseña” y así proseguir con el reseteo de contraseña.

En esta ventana de la aplicación *mobile* el usuario cuenta con dos campos para ingresar información y dos botones a pulsar. Dentro de los campos puede ingresar la contraseña nueva y luego repetir esa clave como verificación de seguridad para evitar errores de tipeo y corroborar que ambas *passwords* escritas son idénticas. Como adicional se implementó el ícono de un ojo a la derecha de los campos a ingresar, que permite al usuario tener la funcionalidad de revisar la contraseña escrita; esto debido a que todos los *inputs* de tipo password impiden la visualización del texto escrito reemplazando los caracteres por asteriscos.

Figura 26. *Layout mobile* resetear contraseña enviada desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Una vez ingresados los campos, el usuario puede marcar la opción “confirmar” para realizar el cambio de contraseña y avanzar a la siguiente ventana.

Figura 27. *Layout mobile* cambio exitoso enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Finalizado con éxito el cambio de contraseña se le notifica al usuario, en este *layout* de la figura 27, con un botón para regresar al inicio de sesión e ingresar a la aplicación para comenzar a monitorear el brazo robótico o administrar los usuarios.

4.2.2.3. *Layout mobile Home*

Una vez el operador ingresa en su sesión ya puede ingresar al panel principal denominado “*home*” o inicio. Este *layout* permite la monitorización de la imagen que la cámara del brazo robótico está enviando, así como también le brinda la posibilidad de poder moverlo en distintas direcciones, rotar, abrir y cerrar la garra. A diferencia de la contraparte de escritorio, al no contar con el suficiente espacio, las funcionalidades mencionadas se encuentran sobre la ventana de la cámara ocultando parcialmente la información recibida por video.

Continuando con las diferencias, se encuentra una barra de navegación en la parte inferior del dispositivo y se saluda al usuario que ingresó desde la parte superior de la pantalla señalada con un color azul para poder diferenciar las tres partes que componen este diseño y mantener las mismas características y funcionalidades que el explorador de escritorio tiene, pero para un dispositivo más chico.

Para poder contener la misma información hubo modificaciones en el botón “Analizar Imagen” que, a diferencia de la contraparte *desktop*, no cuenta con el espacio suficiente para desplegar información una vez es marcada la acción: esto tuvo su resolución al brindarle un *pop-up* donde mostrar los datos recibidos al realizar esa opción y así poder mantener la funcionalidad sin perder espacio o detalles.

En la barra de navegación inferior se pueden destacar tres iconos los cuales permiten al operador cambiar entre las pestañas disponibles y moverse entre los distintos *layouts* de forma dinámica. Hay que señalar que solamente un usuario administrador podrá ver los tres iconos ya que cuenta con los permisos necesarios para realizarlo, en caso de ser un operador regular solo tendrá acceso a dos *layouts* (*Home* y perfil).

Esta pantalla principal permite al usuario poder operar de forma eficiente ante cualquier emergencia de la misma forma en que lo haría desde una computadora, sin excluir ninguna funcionalidad debido a la abstracción de los diseños generados para *desktop* y de esta forma poder incorporarlos en una versión más simplificada.

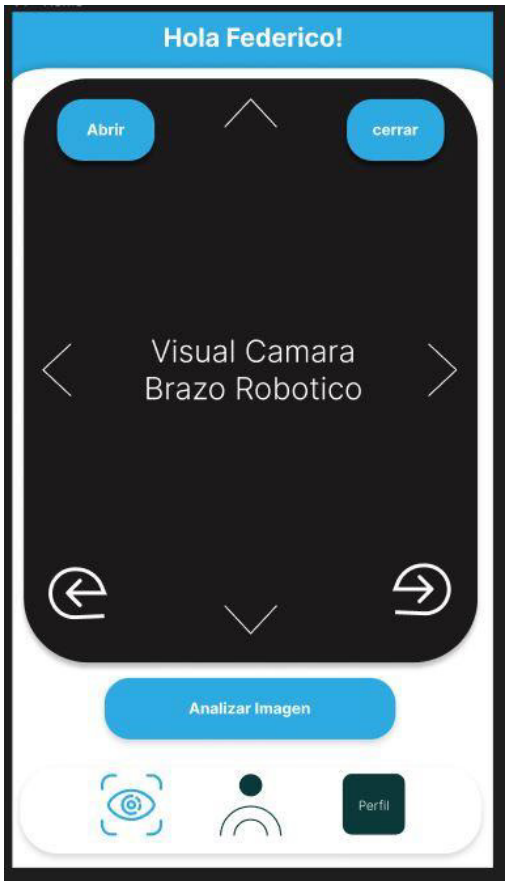


Figura 28. *Layout mobile home* enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

4.2.2.4. *Layout mobile* Usuarios

Una vez el administrador realiza *click* sobre el ícono de usuarios desde la pantalla en la que se encuentre, podrá ver un diseño completamente diferente a la versión de escritorio: dentro de esta pestaña se verán distintas tarjetas con la información de cada usuario de la aplicación.

Las tarjetas se encuentran divididas en tres secciones:

- Sección izquierda: con un fondo de color rojo o verde y un texto para indicar si es un administrador o un usuario regular.
- Sección central: cuenta con la información del usuario, su email y nombre.
- Sección derecha: en esta se encuentran las dos acciones que se pueden realizar sobre un usuario siendo administrador señaladas con dos íconos que

representan la edición o el borrado de estos (cualquiera de las dos acciones desplegará distintas pantallas).

Adicionalmente a esas secciones en la parte superior, separado de las tarjetas, se puede encontrar un buscador, manteniendo la funcionalidad de *desktop* de poder filtrar el listado de usuarios indicando un nombre o un email.y en la parte inferior de las tarjetas podemos encontrar el botón "+ Usuarios" con la funcionalidad de poder almacenar un nuevo registro por parte del administrador.

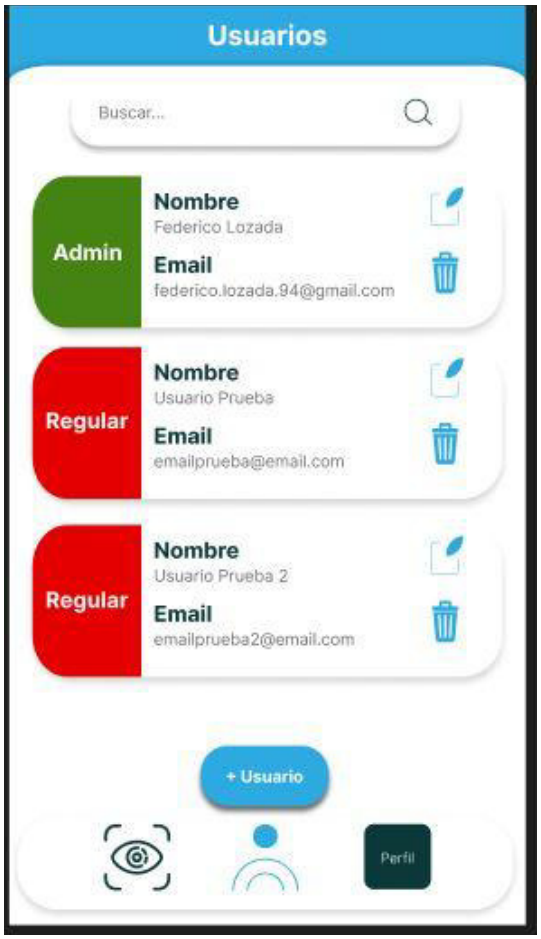


Figura 29. *Layout mobile* usuarios enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

4.2.2.5. *Layout mobile Crear/Editar usuarios*

El administrador tiene las opciones para seleccionar la acción editar o el botón "+ Usuario" como se indica en el punto anterior, lo cual desplegará dos nuevas visualizaciones que contarán con distintos ítems que podrán ser completados o

modificados dependiendo de si el *layout* fue ingresado a través de la acción ubicada en la sección derecha o el botón de creación.

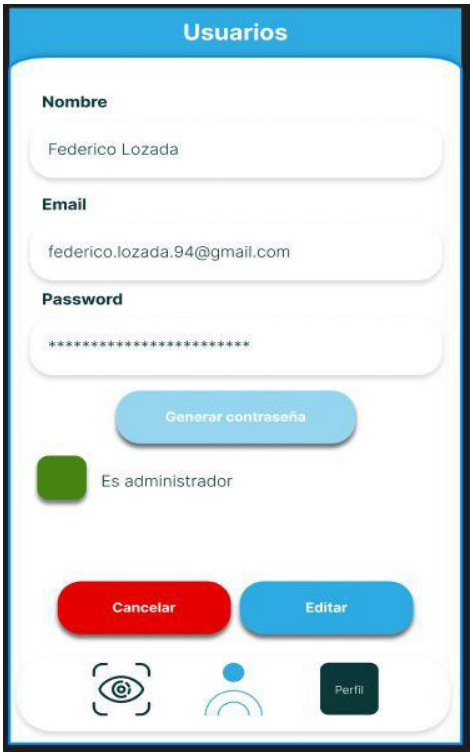


Figura 30. *Layout mobile edit* usuario enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Al igual que en el *layout* de *desktop*, se le permite al administrador realizar las modificaciones sobre los campos de nombre, email o rol del usuario seleccionado y poder realizar una edición de estos campos. También se puede observar en la figura 30 el *password* oculto por asteriscos y el botón “Generar contraseña” deshabilitado, debido que al ser un usuario activo solo el propio operador podría cambiar su clave: es un método de seguridad utilizado para evitar que un administrador pueda ingresar a un usuario distinto del suyo al realizar la modificación de la clave y utilizarlo para su beneficio.

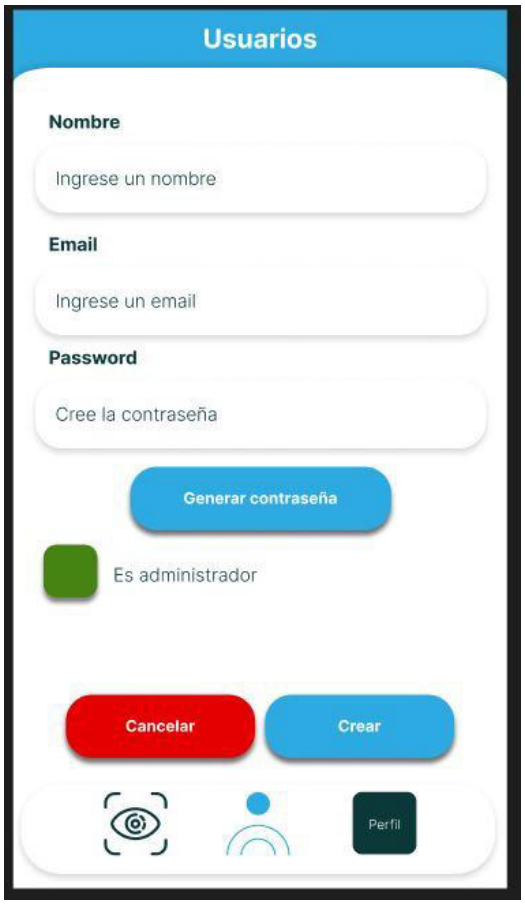


Figura 31. *Layout mobile* crear usuario enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Diferente a la acción “editar”, una vez que el usuario elige la opción “+ Usuario”, se le desplegará un formulario en blanco con toda la información disponible para ser ingresada. Se deberá de completar todos los datos que se puedan brindar sobre nombre y email. El administrador no puede escribir manualmente una contraseña para este nuevo usuario. Al hacer click en el botón “Generar contraseña” esto generará una clave de forma aleatoria con normativas de seguridad que luego podrán ser modificadas por el propio usuario creado. Una vez completado el formulario y marcada la opción “Crear”, le será notificado vía mail los datos de ingreso y se le confirmará al administrador por una alerta que se ha creado con éxito un nuevo registro.

4.2.2.6. *Layout Mobile* eliminar *usuarios*

En el caso de que el administrador eligiera la acción “eliminar” desde el *layout* de “Usuarios”, el mismo desplegará un *Pop-up*, que deshabilita cualquier otra acción con un fondo oscuro transparente, el cual mostrará en el centro de la pantalla un texto de advertencia para informarle que está por eliminar un usuario, el nombre y el email de este, para que así el administrador pueda controlar que no seleccionó la acción en una tarjeta que no corresponda.

Este paso previo a la eliminación también es utilizado en la versión *desktop*: ambos son casos preventivos que aseguran la integridad de los registros de la base de datos de usuarios y mantienen el margen de error humano al mínimo, contemplando que este último es el que genera mayores deficiencias o incongruencias a la hora de realizar auditorías. De esta manera se generará un *log*/registro en la base de datos de que el administrador eliminó al usuario, para que pueda ser identificado si necesita realizarse algún tipo de control.

Esta funcionalidad logra mantener un registro amplio en la administración de usuarios y los movimientos que estos puedan llegar a tener.

También se debe tener en cuenta que a pesar de que la función se denomine “eliminar”, esta en realidad oculta al usuario del listado y le prohíbe su ingreso al sistema. El registro quedará para los controles mencionados en este índice, tanto para *desktop* como *mobile*, y si el usuario es reincorporado se puede crear un nuevo registro o rehabilitar el anterior.

Figura 32. *Layout mobile* eliminar usuario enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

4.2.2.7. *Layout Mobile* perfil

Desde la barra de navegación inferior el usuario puede realizar click en el icono denominado “perfil”, donde se desplegará un nuevo *layout*, para que el operador pueda observar los datos asociados a su usuario, cerrar su sesión o modificar su password o información personal.

Este *layout* cuenta con una diferencia amplia con el generado para *desktop*, ya que para mayor eficiencia se ha simplificado generando que dentro de esta sección haya solo un botón para finalizar la sesión activa y 3 campos que permiten la edición individual de cada uno y no grupal.

Figura 33. *Layout mobile* perfil enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Se puede observar que, por motivos de diseño, el botón de “Cerrar sesión” se encuentra en la parte superior, denominado de esta forma porque corresponde a la experiencia del usuario. Al colocarlo en la parte inferior se detectó que los usuarios son propensos a marcar el botón accidentalmente, queriendo marcar uno de los íconos de la barra de navegación; para evitar estos errores y mejorar la experiencia se optó por modificar el diseño poniendo el botón donde se puede ver en la figura 33.

Además podemos observar que los distintos campos de nombre, email y contraseña tienen, en su lado derecho, un ícono en forma de lápiz, que indica que al tocarlos podrán ser editados.

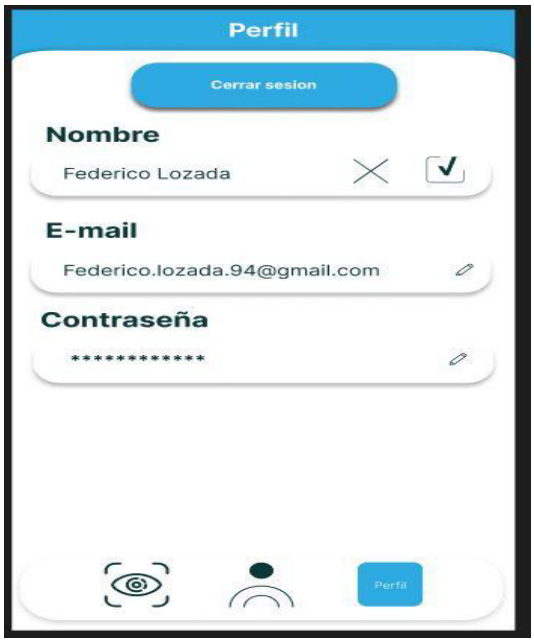


Figura 34. Layout mobile editar campo enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Una vez el usuario hace click en el ícono de editar, el campo permitirá la edición individual del *input* seleccionado, generando dos nuevos íconos: una X para cancelar y un *check* para confirmar la edición. Una vez se haya modificado lo necesario, puede marcar cualquiera de las acciones.

A diferencia de los campos nombre y email, el *input* de la figura 34 denominado contraseña cuenta con un paso de seguridad por lo que su *layout* se diferencia al tocar el botón de editar, desplegando un *pop-up* con nuevas indicaciones, evitando la edición sobre la misma pantalla que hacían los anteriores *inputs*.

Figura 35. *Layout mobile* cambiar *password* campo enviado desde la aplicación.
Fuente: Elaboración propia, basada en la práctica.

Una vez el usuario realiza la acción de editar contraseña, se desplegará una ventana emergente o *pop-up*, generando un fondo negro difuminado que evitará al operario realizar alguna modificacion por fuera de la ventana emergente. En el centro aparecerán tres *inputs* nuevos con sus respectivos *labels* para que el usuario ingrese su contraseña actual, corroborando que es el mismo que ingresó al sistema en un comienzo, para luego solicitarle la nueva contraseña que desea y que la repita. Los tres campos cuentan con un ícono en forma de ojo que permite al usuario, una vez realiza click sobre estos, ver los datos ingresados. Esta funcionalidad de *UX* permite al usuario *logueado* en el sistema corroborar la información que está tipeando y, de esta forma, evitar este tipo de errores o pérdidas de tiempo por reingreso de información.

5. Inconvenientes

A continuación, se mencionan los inconvenientes que surgieron a lo largo de cada actividad realizada junto con su resolución, si es que la tuvo, o con la alternativa que permitió subsanar el problema.

- ***Host* para el servicio web**

Para poder realizar la conexión entre el sitio web y el brazo robótico, se requería de una implementación que pudiera recibir y enviar la información. Normalmente se podría contratar un *host* como *amazon web services* o *google cloud*, pero el costo no valdría la pena para este sistema de monitorización.

Se ubicaron varios *hosts* gratuitos que pueden alojar el proyecto y levantar esta conexión de forma segura y sin inconvenientes, pero para uso práctico y de pruebas se levantó una *ip* pública con la aplicación *ngrok*, que permite conectar los servidores locales del sistema con la ruta generada y, de esta forma, luego poder conectar el *hardware* con la aplicación web.

- **Conexión con esp32-cam**

Debido a que era la primera vez que veía el *hardware*, y que las opciones son variadas por las herramientas utilizadas, fue una dificultad ver cuál era el mejor curso de acción a la hora conectarse a la cámara.

Después de mucha investigación, se decidió considerar utilizar la conexión desde la herramienta Vue a través de un *websocket* conectado a la *ip* fija por la cual el *esp32-cam* realiza el *streaming* de video.

En un comienzo se pensó realizar la conexión desde el *backend* asegurando desde *laravel* los paquetes de seguridad y manteniendo un puerto abierto en la ruta para el *streaming* constante, pero esto gestionó distintos problemas en los *middleware* lo que prohibía una conexión estable entre el *backend* y el *frontend*. Por lo que luego de la investigación, se pudo encontrar la solución de los *websockets* asociados directamente al *frontend* Vue y proceder a realizar la conexión.

- **Alertas Brazo robótico**

Para la conexión con los movimientos del brazo robótico se tuvieron que implementar una serie de controles, debido a que se le puede enviar la petición para que realice movimientos que podrían romper el *hardware* o generar inconvenientes a la hora de su utilización, por lo que para cada petición de movimiento o rotación, se investigó, por los márgenes máximos y mínimos, que el movimiento del brazo pudiera soportar sin romperse o bloquearse y desde el *backend*, antes de cada petición de movimiento, se crea una solicitud de informe de estado al brazo robótico para que indique la disponibilidad y si el movimiento que se está deseando realizar desde el *backend* tiene margen disponible para poder realizar dicho movimiento.

6. Posibles mejoras

- **Encriptado de peticiones**

Para salvaguardar la información enviada por el sistema hacia el *hardware* y los clientes, sería una posible mejora realizar el encriptado de todos los paquetes recibidos y enviados a través del *middleware* de *laravel* gestionando que, si ocurriera una situación de *man-in-the-middle*, el mismo no pueda visualizar la información de los paquetes

- **Https**

Esta posible mejora se pensó ya que con el protocolo de https se cifraba la información basándose en seguridad de capa de transporte (TLS). TLS usa claves públicas y privadas de largo plazo, a fin de generar una clave de sesión de corto plazo que se usa para encriptar el flujo de datos entre cliente y servidor. Se utilizan certificados para autenticar el servidor y, estos requieren que haya autoridades que los emitan, los validen y les den legitimidad. Estas acciones dan resistencia a ataques de intermediarios al encriptar las claves de las comunicaciones por lo que el sitio web debe estar alojado completamente en HTTPS, es decir, ninguna de sus partes debe estar alojada en http, o el usuario quedará vulnerable a esta clase de ataques. Las URL de HTTPS comienzan con “https://” y usan el puerto 443 por defecto, mientras que las URL de HTTP empiezan con “http://”y usan el puerto 80 por defecto.

- **Mejoras estéticas de la página web**

La estética de la página web no solo está relacionada con la visual, sino también con el contenido y la funcionalidad. Para que el sitio no se desactualice, es importante, cada cierto período de tiempo, realizarle alguna modificación de ser necesaria. Por una parte, las modificaciones pueden ser netamente visuales y solo cambiar el estilo a los componentes del sitio, pero también hay que considerar que en un futuro se pueden ampliar las funcionalidades y esto, a su vez, genera nuevos cambios que habría que implementar, tanto para el flujo y funcionalidad como estéticamente.

7. Conclusión

Según los objetivos específicos planteados al inicio de la práctica profesional supervisada, se llega a las siguientes conclusiones:

- Los operadores de esta aplicación podrán gozar de una administración remota bastante completa, pudiendo realizar análisis de imágenes y los movimientos manuales que sean necesarios, según la visualización de la cámara remota del brazo robótico.

- Se adicionó,a medida que se avanzaba en el proyecto a los objetivos, la implementación de un rol administrador, para la gestión y manejo de usuarios, generando así una nueva sección a implementar que no se tenía considerada pero resultó en una parte importante de la aplicación. Debido a que es un sistema con implementación privada, no se puede permitir la libre creación de usuarios desde las rutas públicas, ya que cualquiera podría tener acceso y realizar acciones sobre el brazo robótico sin tener autorización.

- Las capas de seguridad nunca parecen ser pocas: cada vez que se generaba un nuevo flujo o diseño había que implementar, en la lógica o funcionalidad, algún tipo de seguridad o control para evitar acciones no autorizadas o daños en el *hardware* por un mal manejo de la administración remota.

Algunos de los problemas encontrados para realizar este trabajo se ubicaron en la poca experiencia en el manejo de alguna de las herramientas mencionadas, tanto las de *hardware* como las de *software*.

El tiempo dedicado a entender los conceptos sirvió para comprender sus beneficios y las adiciones que se fueron realizando. Entre estas adiciones se pudo ver a medida que se avanzaba en el desarrollo que era necesario implementar nuevas capas de control de seguridad ya que fueron necesarias para evitar que distintos peligros ocurrieran, tanto en el software como en el hardware.

8. Reflexión sobre la PPS

Tanto en la investigación como en la práctica realmente pude incorporar conceptos que me ayudaron a conocer más de cerca lo que es un laboratorio remoto, cómo se trabaja con él y el por qué es tan útil para las nuevas modalidades académicas.

Durante el desarrollo de la PPS, pude incorporar nuevos conceptos y aplicarlos para poder resolver las problemáticas que se me fueron presentando. A pesar de haber manejado tecnologías conocidas, a la hora de poner en práctica mis conocimientos sobre estas tecnologías y nuevos conceptos incorporados, se fueron presentando dudas y trabas que solo surgen en el momento de la acción, de la implementación y creación de las distintas funcionalidades.Además incorporé más conocimientos sobre lo que implica un diseño de UX/UI, sus complejidades y cómo aplicarlo en el *frontend* y las funcionalidades con sus respectivas interacciones, por lo que me encuentro bastante conforme con el trabajo realizado y por todo el conocimiento adquirido a lo largo de la carrera que, a posteriori, me servirá tanto en mi formación laboral, como en la personal.

9. Bibliografía

Albaladejo, X. (2018). Una retrospectiva ágil de Scrum. Recuperado de https://proyectosagiles.org/2008/10/21/retrospectiva-agil-scrum/ (Fecha de consulta: 14/06/2023)

Benito, M. (2013). ¿Qué es la usabilidad web?. Recuperado de https://blog.admetricks.com/que-es-la-usabilidad (Fecha de consulta: 14/06/2023)

Darer, J. (2016). Desarrollo de aplicaciones web. Recuperado de https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html (Fecha de consulta: 14/06/2023)

RedHat (2023) . ¿Qué son las API y para qué sirven?. Recuperado de https://www.redhat.com/es/topics/api/ what-are-application-programming-interfaces (Fecha de consulta: 18/06/2023)

Amazon Web Services (2023). ¿Qué es el middleware?. Recuperado de https://aws.amazon.com/es/what-is/middleware/#:~:text=El%20middleware%20es%20un%20software,se%20pueda%20innovar%20m%C3%A1s%20r%C3%A1pido. (Fecha de consulta: 29/08/2023)

Hubspot (2023). ¿Qué es el desarrollo web?. Recuperado de https://blog.hubspot.es/website/que-es-desarrollo-web. (Fecha de consulta: 26/10/2023)

Unirfp (2022). ¿Qué es un framework?. Recuperado de https://www.edix.com/es/instituto/framework/ (Fecha de consulta: 08/11/2023)

Nieto, L. (2022). Tipos de diseño web. Recuperado de https://somospecesvoladores.com/blog/los-tipos-de-diseno-web-que-debes-conocer/ (Fecha de consulta: 10/12/2023)