MANUAL TÉCNICO

DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB MÓVIL PARA DENUNCIAS DE CONTROL URBANO DE OBRAS EN CONSTRUCCIÓN EN LA LOCALIDAD TRES DE LA CIUDAD DE CARTAGENA, APLICANDO BUENAS PRÁCTICAS DE INGENIERÍA DE SOFTWARE (CMMI)

PRESENTADO POR:

DEIMER AVILA ARGOTE

UNIVERSIDAD DEL SINÚ ELÍAS BECHARÁ ZAINÚM SECCIONAL CARTAGENA

ESCUELA DE INGENIERÍA DE SISTEMAS

CARTAGENA-COLOMBIA

Octubre 2019

Contenido

Tablas

INTRODUCCION

Este manual tiene como finalidad proporcionar al lector la lógica con la que se ha desarrollado el proyecto Gestor Urbano, aclarando que este manual pretende documentar la aplicación en el entorno que fue desarrollado. Describiendo las características principales que se involucran en el desarrollo de este, además de las herramientas utilizadas, instalación, alojamiento y accesos al sistema.

1. DETALLES DEL MANUAL

Nombre del sistema: Gestor Urbano

Versión Del sistema: 2.0

Tipo de manual: manual técnico

Fecha de elaboración: noviembre 6 del 2019

1.1 Especificaciones Técnicas

Tabla 1.Especificaciones técnicas

ITEM	CARACTERISTICA	DESCRIPCION
1	**Técnicas Sistema Operativo:**	Superiores. Windows 7 o versiones
2	**Manejador de Base de Datos:**	5.7.27 - MySQL Community Server
3	**Servidor de Aplicaciones:**	Servidor Web Apache 2.0, Servidor. Versión del protocolo: 10Usuario: gestorur@localhost, Conjunto de caracteres del servidor: UTF-8 Unicode (utf8)

4	**Servidor Web:**	cpsrvd 11.82.0.16, Versión del cliente de base de datos: libmysql 5.6.43, extensión PHP: mysqli curl mbstring, Versión de PHP: 7.2.7
5	**Lenguaje de Programación:**	PHP: 7.2.7, HTML5, CSS3, Framework (Bootstrap 2.8), JavaScript.
6	**Navegador Web:**	Cualquier versión de Google Chrome (para trabajar en la aplicación) y funciona en todos los exploradores Web.
7	**Dispositivos Móviles:**	Cualquier dispositivo móvil con sistema operativo Android o ios
8	**Desktop:**	Cualquier computador con navegador web Chrome
9	**Equipos portátiles:**	Cualquier equipo portátil con navegador web Chrome

2. DESARROLLO DE LA PLATAFORMA WEB

Gestor urbano está desarrollado en el lenguaje de programación php en el editor de código abierto Atom el cual funciona en Windows Linux Y Mac OS, se seleccionó esta herramienta debido a su versatilidad y facilidad de uso, además que cuenta con soporte para múltiples Plug-in , el control de versiones en Git . Además de que atom es una aplicación de escritorio que fue construida utilizando tecnologías web, ideal para nuestro proyecto. Otra de las ventajas de utilizar atom es que la mayor parte de los paquetes tienen licencia de software libre, desarrollados y mantenidos por la comunidad de usuarios.

Atom utiliza complementos predeterminados de múltiples lenguajes de programación, pero para este proyecto en específico nos enfocaremos en PHP y SQL los cuales fue nuestros lenguajes seleccionados. Además de JavaScript, HTML y para el manejo e integridad de los datos estarán en el motor de Base de datos SQL Server. Las librerías para utilizar son: Bootstrap para el diseño de las interfaces desarrolladas.

3. HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

3.1 PHP

(Acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. [9] En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!).El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga. **[9]**

3.2 MYSQL

MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto, basado en lenguaje de consulta estructurado (SQL).

Se ejecuta en prácticamente todas las plataformas, incluyendo Linux, UNIX y Windows. A pesar de que se puede utilizar en una amplia gama de aplicaciones, Se asocia más con las aplicaciones basadas en la web y la publicación en línea y es un componente importante de una pila empresarial de código abierto llamado LAMP. LAMP es una plataforma de desarrollo web que utiliza Linux como sistema operativo, Apache como servidor web, MySQL como sistema de gestión de base de datos relacional y PHP como lenguaje de programación orientado a objetos (a veces, Perl o Python se utiliza en lugar de PHP). [10]

.

3.3 HTML 5

Se utilizó la quinta versión de HTML5 (*HyperText Markup Language*, versión 5). Esta nueva versión (aún en desarrollo), y en conjunto con CSS3, define los nuevos estándares de desarrollo web, rediseñando el código para resolver problemas y actualizándolo así a nuevas necesidades. No se limita solo a crear nuevas etiquetas o atributos, sino que incorpora muchas características nuevas y proporciona una plataforma de desarrollo de complejas aplicaciones web (mediante los APIs).

3.4 CSS

Esta última evolución del lenguaje de las Hojas de Estilo en Cascada (Cascading Style Sheets), y pretende ampliar la versión CSS2.1. Trae consigo muchas novedades altamente esperadas, como las esquinas redondeadas, sombras, gradientes, transiciones o animaciones, y nuevos layouts como multi-columnas, cajas flexibles o maquetas de diseño en cuadrícula (grid layouts).

Las partes experimentales son particulares para cada navegador y deberían ser evitadas en entornos de producción, o usadas con extrema precaución, ya que tanto la sintaxis como la semántica pueden cambiar en el futuro

3.5 JAVASCRIPT

JavaScript es un lenguaje de programación que se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web.

3.6 BOOTSTRAP

Bootstrap es un framework desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es Open Source o código abierto, por lo que lo podemos usar de forma gratuita y sin restricciones.

4. DESCRIPCIÓN DE ALOJAMIENTO

Ingreso al cPanel

Gestor urbano se encuentra alojado bajo dominio y hosting privado, contratados con la empresa combia hosting. Para acceder al sistema done se encuentran alojados los archivos y base de datos se debe ingresar bajo la dirección www.gestorurbano.com:2083; nótese que el puerto utilizado es el 2083. Los accesos al sistema se encuentran descritos en el formato plan de integración del producto. La siguiente ilustración muestra el proceso de inicio de sesión:

Ilustración 1.inicio de sesión en cPanel

Una vez ingresamos al sistema encontraremos un menú con múltiples opciones; el cPanel se caracteriza por contar con varias herramientas de software para manejo de base de datos, alojamiento de los documentos, manejador de archivos, entre muchas otras opciones más. Para el caso de este manual técnico nos centraremos en las opciones:

- File manager (Sección Files)
- PHP MyAdmin (Sección Databases)
- MySQL Databases (Sección Databases)

En el primer caso tenemos a File manager; en este se encuentran los archivos de nuestro proyecto, tales como: librerías, clases, carpetas, imágenes, etc.

Ilustración 2.Dashboard del cPanel

Una vez ingresamos veremos la siguiente estructura de carpetas:

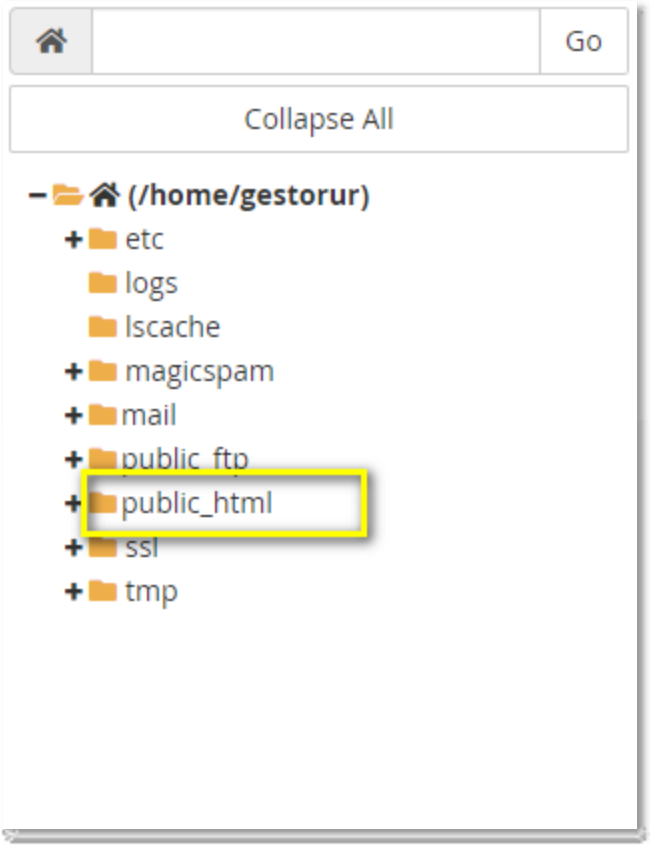

Ilustración 3.Estructura de carpetas, File Manager

En esta vista podemos apreciar la estructura de las carpetas contenidas en el manejador de archivos. La carpeta **public_html** es la que contiene todos los archivos de nuestro proyecto como lo podemos apreciar en la siguiente Ilustración:

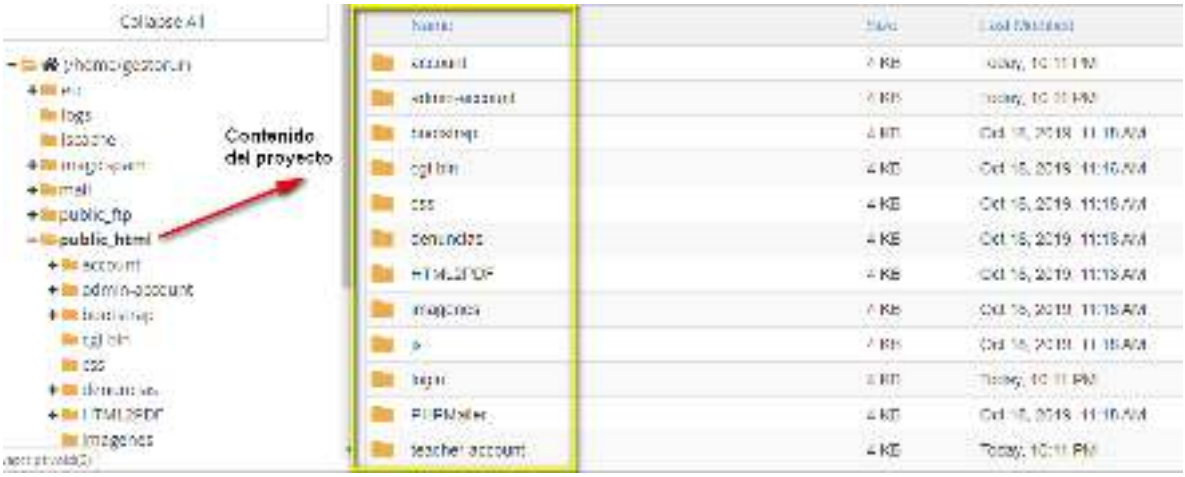

Ilustración 4.Contenido de la carpeta public_html

En esta carpeta es donde se encuentra todo el contenido del proyecto Gestor Urbano y a la cual por medio del protocolo HTTP podemos acceder a todas las características del sistema. Cada una de las carpetas que se muestran en la ilustración 4 tiene una función especial, la cual se detalla en la siguiente tabla:

Tabla 2.Estructura del contenido de la carpeta public_html, fuente autor

Nombre	Función	Localización	Tamaño	Número de líneas	Incluye
account	Es donde se encuentran los datos de la cuenta del ciudadano	/public_html/	4KB		
admin-account	Es donde se encuentran los datos relacionados al administrador	/public_html/	4KB		
bootstrap	Carpeta donde se encuentran las librerías de boostrap	/public_html/	4KB		
cgi-bin		/public_html/	4KB		
css	Estilos del sistema	/public_html/	4KB		
denuncias	Carpeta que contiene las imágenes de las denuncias	/public_html/	4KB		
TML2PDF	Librería para pasar las etiquetas html a PDF como reporte de denuncia	/public_html/	4KB		
imagenes	Contiene las imágenes del sistema como logo, iconos etc	/public_html/	4KB		
js	Contiene los js del Login recuperación de contraseña y registro	/public_html/	4KB		
login		/public_html/	4KB		
PHPMailer	Librería para enviar al usuario correo de actualización de contraseña	/public_html/	4KB		
teacher-account	Es donde se encuentran los datos del módulo docente	/public_html/	4KB		
unilab_clases	Aquí se encuentran las clases principales del sistema	/public_html/	4KB		
vendor		/public_html/	4KB		
composer.json		/public_html/	65 bytes		
composer.lock		/public_html/	3.2 KB		
favicon.ico	Es el favicon de la pagina	/public_html/	33.69 KB		
index.php	Aquí se encuentra la estructura principal de la pagina	/public_html/	40 bytes		
leerImg.php	Estructura para leer imágenes cargadas por el ciudadano	/public_html/	447 bytes		
login.php	Estructura del Login del sistema	/public_html/	2.14 KB		
ModalAdminLogin.html	Estructura modal del loging administrador	/public_html/	0 bytes		

Modaldenuncia.html	Estructura modal de la tabla de denuncia	/public_html/	21.92 KB		
modals-acerca-de.php	Estructura modal donde se encuentran los datos de información general del sistema	/public_html/	14.37 KB		
passRecovery.php		/public_html/	1.74 KB		
registro.php	Estructura para el registro de usuario	/public_html/	4.38 KB		
tablaEncuesta.html	Tabla html donde se encuentran las preguntas del formulario	/public_html/	11.05 KB		

Modulo Ciudadano:

Este módulo cuenta con una serie de carpetas clases, estilos y características las cuales permiten su funcionalidad e interacción con los módulos Administrador y Docente; los archivos que contiene la carpeta account bajo la ruta***:*** ***public_html/account,*** dichas características se encuentran descritas en la siguiente tabla:

Tabla 3.Estructura del contenido de la carpeta account, fuente autor

Nombre	Función	Localización	Tamaño	Número de líneas	Incluye
Ajax	Aquí se encuentran los js para validación de carga de denuncia y gestión de notas agregadas	/public_html/account/	4KB		
css	Carpeta que contiene los estilos del modulo	/public_html/account/	4KB		
img	Carpeta que contiene el icono de locación y el fondo de la pantalla principal	/public_html/account/	4KB		
js	Carpeta que contiene los js para actualización de datos, cambio de clave, validación de campos y ver notas en la denuncia.	/public_html/account/	4KB		
scss	Carpeta que contiene los estilos propios de la plantilla	/public_html/account/	4KB		
vendor	Carpeta que contiene estilos de boostrap, fuentes, datatables, y jquery propios de la plantilla	/public_html/account/	4KB		
404.html	Archivo que contiene la estructura de la página de error ante las eventualidades.	/public_html/account/	16.86 KB		
blank.html	Ejemplos propios de la plantilla	/public_html/account/	16.58 KB		

buttons.html	Ejemplos propios de la plantilla	/public_html/account/	24.37 KB		
cards.html	Ejemplos propios de la plantilla	/public_html/account/	23.82 KB		
charts.html	Ejemplos propios de la plantilla	/public_html/account/	19.1 KB		
favicon.ico	Favicon de la pagina	/public_html/account/	48.05 KB		
forgot-password.html	Ejemplos propios de la plantilla	/public_html/account/	2.75 KB		
gestor-notas.php	Archivo que contiene la estructura para mostrar las notas	/public_html/account/	718 bytes		complaint_class.php
gestorDenuncia.php	Archivo que contiene la estructura de la carga de imágenes y la gestión del formulario y donde se toma la ubicación de las coordenadas	/public_html/account/	1.41 KB		complaint_class.php
gestorUbicaciones.php	Archivo que toma los datos de la locación de la denuncia por método get	/public_html/account/	153 bytes		complaint_class.php
gulpfile.js	Ejemplos propios de la plantilla	/public_html/account/	3.71 KB		
index.php	Archivo que contiene la estructura de la página principal del modulo	/public_html/account/	13.17 KB		
leerImg.php	Estructura para la selección de imágenes	/public_html/account/	1.17 KB		
LICENSE	Licencia de la plantilla	/public_html/account/	1.07 KB		
login.html	Ejemplos propios de la plantilla	/public_html/account/	3.52 KB		
maps.php	Archivo que contiene los enlaces a la API de Google para el mapa	/public_html/account/	1.19 KB		
modalActualizarDatos.php	Archivo que contiene la estructura modal para actualizar datos	/public_html/account/	3.87 KB		
modalCambioClave.php	Archivo que contiene la estructura modal para el cambio de contraseña	/public_html/account/	1.65 KB		
modalCuestionario.php	Archivo que contiene la estructura modal para mostrar el cuestionario	/public_html/account/	18.89 KB		
modalNotas.php	Archivo que contiene la estructura modal para mostrar las notas agregadas por el docente	/public_html/account/	1.57 KB		

package-lock.json	Paquetes json propios de boostrap	/public_html/account/	256.53 KB		
package.json	Paquetes json propios de boostrap	/public_html/account/	1.46 KB		
		/public_html/account/			
README.md	Archivo que crea Git al subir los datos a GITHUB	/public_html/account/	4.16 KB		
register.html	Archivo que contiene la estructura HTML del registro de usuario	/public_html/account/	3.6 KB		
screen_denucnias.php	Archivo que contiene la estructura de la página donde se muestran las denuncias	/public_html/account/	2.72 KB		
screen_home.php	Archivo que contiene la estructura principal del home	/public_html/account/	2.66 KB		
tabla_denuncia.php	Archivo que contiene la estructura de la tabla de denuncia	/public_html/account/	4.04 KB		
tables.html	Ejemplos propios de la plantilla	/public_html/account/	35.17 KB		
utilities-animation.html	Ejemplos propios de la plantilla	/public_html/account/	21.22 KB		
utilities-border.html	Ejemplos propios de la plantilla	/public_html/account/	18.99 KB		
utilities-color.html	Ejemplos propios de la plantilla	/public_html/account/	20.45 KB		
utilities-other.html	Ejemplos propios de la plantilla	/public_html/account/	20.06 KB		

Módulo Administrador

A continuación se describen los archivos que contiene la carpeta Admin-account bajo la ruta***: public_html/admin-account***, estos archivos y subcarpetas son los encargados de interactuar con los demás módulos y clases del sistema:

Tabla 4.Estructura del contenido de la carpeta admin-account, fuente autor

Nombre	Función	Localización	Tamaño	Número de líneas	Incluye

ajax	**Carpeta que contiene las estructuras de los archivos para: Agregar, editar y eliminar preguntas; Editar, listar y eliminar docentes y el modal que contiene las imágenes cargadas por el ciudadano**	/public_html/admin-account/	4 KB		
clases	**Carpeta que contiene la clase teacher_data.php la cual tiene la función de registrar usuario, negociar los datos de los id de la de los docentes asignados y la actualización de los estados de las denuncias**		4 KB		
css	**Carpeta que contiene los estilos css de los módulos**		4 KB		
Js	**Carpeta que contiene los archivos js para actualización de docentes asignación de denuncias, registro de administrador, registro de docentes, validación de campos, validación de las imágenes adjuntas y eliminación de denuncias.**		4 KB		
scss	**Carpeta que contiene los estilos propios de boostrap**		4 KB		
vendor	**Carpeta que contiene estilos de boostrap, fuentes, datatables, y jquery propios de la plantilla**		4 KB		
404.html	**Archivo que contiene la estructura de la página de error ante las eventualidades.**		16.86 KB		
add_admin.php	**Archivo que contiene la estructura para registrar un administrador**		3.23 KB		
asignar-denuncia.php	**Archivo que contiene la estructura para la asignación de las denuncias**		669 bytes		
blank.html	**Ejemplos propios de la plantilla**		16.58 KB		
buttons.html	**Ejemplos propios de la plantilla**		24.37 KB		
cards.html	**Ejemplos propios de la plantilla**		23.82 KB		
charts.html	**Ejemplos propios de la plantilla**		19.1 KB		
complaint_Table.php	**Archivo que contiene la estructura PHP de la tabla de denuncias y su gestión**		5.6 KB		teacher_data.php
Cuestionario.php	**Archivo que contiene la estructura para editar y agregar preguntas al cuestionario**		2.25 KB		
favicon.ico	**Favicon de la pagina**		48.05 KB		
forgot-password.html	**Ejemplos propios de la plantilla**		2.75 KB		
gestorDenuncia.php	**Archivo que contiene los datos Por medio del método POST para los datos del usuario, cuestionario y ubicación**		391 bytes		complaint_class.php

gulpfile.js	**Ejemplos propios de la plantilla**		3.71 KB		
index.php	**Archivo que contiene la estructura de la página principal del modulo**		12.42 KB		
LICENSE	**Licencia de la plantilla**		1.07 KB		
Lista_docente.php	**Archivo que contiene la estructura de la página donde se visualizan el listado de docentes**		2.46 KB		
login.html	**Ejemplos propios de la plantilla**		3.52 KB		
modal_admin_login.php	**Archivo que contiene la estructura modal cuando se solicita la contraseña de administrador**		1.05 KB		
modal_preguntas.php	**Archivo que contiene la estructura para agregar preguntas al cuestionario**		1.97 KB		
modalActualizarDatosDocente.php	**Archivo que contiene la estructura modal para la actualización de los datos del docente**		4.09 KB		
modalEditarPregunta.php	**Archivo que contiene la estructura modal para editar las preguntas del cuestionario.**		3.57 KB		
modalEliminarPregunta.php	**Archivo que contiene la estructura modal para la eliminación de preguntas del cuestionario.**		643 bytes		
modals_asignacion_denuncia.php	**Archivo que contiene la estructura modal para la asignación y reasignación de las denuncias a docentes**		3.36 KB		teacher_data.php
package-lock.json	**Paquetes json propios de boostrap**		265.53 KB		
package.json	**Paquetes json propios de boostrap**		1.46 KB		
README.md	**Archivo que crea Git al subir los datos a GITHUB**		4.16 KB		
register.html	**Ejemplos propios de la plantilla**		3.6 KB		
screen_docente.php	**Archivo que contiene la estructura para la creación del docente**		4.66 KB		users_data_class.php
tables.html	**Ejemplos propios de la plnatilla**		35.17 KB		
utilities-animation.html	**Ejemplos propios de la plnatilla**		21.22 KB		
utilities-border.html	**Ejemplos propios de la plnatilla**		18.99 KB		
utilities-color.html	**Ejemplos propios de la plnatilla**		20.45 KB		

Módulo Docente

A continuación, se describen los archivos que contiene la carpeta teacher-account bajo la ruta***: public_html/teacher-account***, estos archivos y subcarpetas son los encargados de interactuar con los demás módulos y clases del sistema:

Tabla 5.Estructura del contenido de la carpeta teacher_account, fuente autor

Nombre	Función	Localización	Tamaño	Número de líneas	Incluye
Clases	**Contiene archivo con la clase teacher_data la cuel es la estructura para la creacion de docentes**	/public_html/teacher-ccount/	4 KB		
Css	**Carpeta que contiene los estilos del modulo**	/public_html/teacher-ccount/	4 KB		
Img	**Carpeta que contiene el icono de locación y el fondo de la pantalla principal**	/public_html/teacher-ccount/	4 KB		
js	**Carpeta que contiene los js para actualización de datos, cambio de clave, validación de campos y ver notas en la denuncia.**	/public_html/teacher-ccount/	4 KB		
Scss	**Carpeta que contiene los estilos propios de boostrap**	/public_html/teacher-ccount/	4 KB		
Vendor	**Carpeta que contiene estilos de boostrap, fuentes, datatables, y jquery propios de la plantilla**	/public_html/teacher-ccount/	4 KB		
404.html	**Archivo que contiene la estructura de la página de error ante las eventualidades.**	/public_html/teacher-ccount/	16.86 KB		
actualizar-estado-denuncia.php	**Contiene la estructura PHP para la actualización del estado de la denuncia**	/public_html/teacher-ccount/	303 bytes		`complaint_class.php`
blank.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	16.58 KB		
buttons.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	24.37 KB		
cards.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	23.82 KB		
charts.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	19.1 KB		
complaint_Table.php	**Archivo que contiene la estructura de las notas y estados de la denuncia**	/public_html/teacher-ccount/	6.57 KB		`teacher_data.php`
favicon.ico	**Favicon de la pagina**	/public_html/teacher-ccount/	48.05 KB		

forgot-password.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	2.75 KB		
gestor-notas.php	**Archivo que contiene la estructura backed con el método para agregar notas**	/public_html/teacher-ccount/	718 bytes		complaint_class.php
gestorDenuncia.php	**Archivo que contiene la estructura de cómo se suben los archivos desde la claser complait_cass.php**	/public_html/teacher-ccount/	1.37 KB		complaint_class.php
gestorUbicaciones.php	**Archivo que gestiona los datos de la ubicación donde se genera la denuncia**	/public_html/teacher-ccount/	153 bytes		complaint_class.php
gulpfile.js	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	3.71 KB		
index.php	**Archivo que contiene la Estructura principal del modulo**	/public_html/teacher-ccount/	11.62		complaint_class.php
leerImg.php	**Archivo que contiene la estructura backend para la lectura de las imágenes.**	/public_html/teacher-ccount/	302 bytes		
LICENSE	**Licencia de la plantilla**	/public_html/teacher-ccount/	1.07 KB		
login.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	3.52 KB		
modal-actualizar-estado-denuncia.php	**Archivo que contiene la estructura modal para la actualización de los estados de las denuncias**	/public_html/teacher-ccount/	2.08 KB		
modalCambioClave.php	**Archivo que contiene la estructura modal para realizar el cambio de contraseña**	/public_html/teacher-ccount/	1.65 KB		
package-lock.json	**Paquetes json propios de boostrap**	/public_html/teacher-ccount/	265.53 KB		
package.json	**Paquetes json propios de boostrap**	/public_html/teacher-ccount/	1.46 KB		
README.md	**Archivo que crea Git al subir los datos a GITHUB**	/public_html/teacher-ccount/	4.16 KB		
register.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	3.6 KB		
screen_home.php	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	2.66 KB		
tables.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	35.17 KB		
utilities-animation.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	21.22 KB		

utilities-border.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	18.99 KB		
utilities-color.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	20.45 KB		
utilities-other.html	**Ejemplos propios de la plantilla**	/public_html/teacher-ccount/	20.06 KB		

5. CLASES PRINCIPALES DEL SISTEMA

Gestor urbano cuenta con tres clases principales las cuales son importadas de acuerdo al archivo que la requiera; estas clases son: complaint_class.php la cual hace referencia a las denuncias generadas; login_registration_class.php la cual hace referencia al registro y conexión de usuarios la base de datos y la clase user_data_class.php la cual se encarga de generar los códigos aleatorios para las denuncias.

En la siguiente ilustración se puede apreciar la ruta de acceso a estas clases la cual se encuentra ingresando a la carpeta unilab_clases:

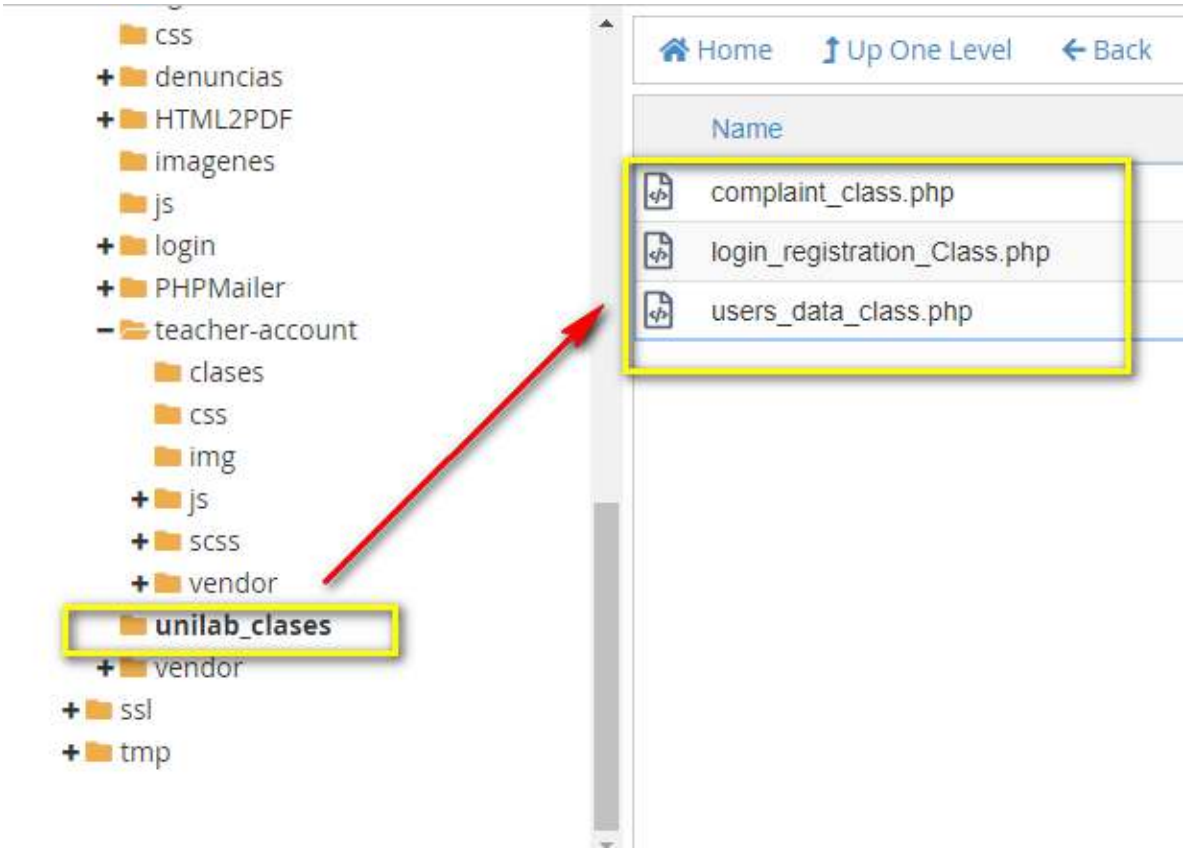

Ilustración 5.Clases principales del sistema, Fuente autor

A continuación, se describe la codificación de cada una de estas clases, cabe anotar que en el código se encuentra comentado la función que cada fragmento realiza para el funcionamiento de estas:

Clase complaint_class.php

Esta clase se encuentra bajo la siguiente ruta en el servidor: public_html/unilab_clases; en esta clase encontramos las funciones encargadas de creación de usuarios y guardado en la base de datos, Función para pasar los datos de “mis denuncias, Función para pasar los datos de las denuncias asignadas a docente, Función para pasar los datos de la dirección generada desde la denuncia entre muchas otras las cuales se encuentran descritas en la siguiente tabla:

Tabla 6.Codificación clase complaint_cass, fuente autor

complaint_class.php

```
<?php

/**
 * se inicia la clase Complaint la cual refiere a las denuncias
 */
class Complaint
{
        //Se crea la conexión a la base de datos
        private $conexion=false;
        function __construct($datos_conexion = false)
        {
                if(!$datos_conexion){
                        $server="54.39.133.88";
                        $username='gestorur_unilab';
                        $password='unilab2019.';
                        $db='gestorur_unilab_db_2';
                        /*$server="Localhost";
                        $username='root';
                        $password='';
                        $db='gestorur_unilab_db_2';*/

//Se validan los datos de la conexión
                        $this->conexion=                                new
mysqli($server,$username,$password,$db);
                        if($this->conexion->connect_errno){
                                echo($this->conexion->connect_error);
                        }else{
                                $this->conexion->set_charset("utf8");
                        }
                }
        }
//Se crea la función para envió de datos del usuario por medio del método
get y se realiza la consulta en la base de datos
        public function getInfoUser($idD){
                $query=$this->conexion->query("SELECT   idUsuario   FROM
`denuncia` WHERE idDenuncia= $idD");
                $result=$query->fetch_array();
                if(isset($result)){
                        $idU=$result['idUsuario'];
                        $query2=$this->conexion->query("SELECT   *   FROM
`usuario` WHERE `idUsuario` = '$idU'");
                        $result2=$query2->fetch_array();
                        $datos = array('Nombre' =>$result2['nombre'] ,
                                'Direccion'=>   $result2['direccion']   ,
'Cedula' => $result2['cedula'] ,
                                'Telefono'   =>   $result2['telefono']   ,
'Email' => $result2['email'] ,
                                'Id'    =>    $result2['idUsuario']    ,
'LugarExpedicion' => $result2['lugarExpedicion']);
```

```
                        return $datos;
                }
        }
//Función para la creación de usuarios en la base de datos
        public                                                  function
create($idUsuario,$location,$cuestionario,$imagenes){
                $idUser= $idUsuario;
                $idComplaint = $this -> consecutiveIdComplaint();
                $location = $location;
                if(strlen($idUsuario) == 5){
                        $consulta     =     "INSERT     INTO     denuncia
(idDenuncia,estado,geoReferencia,idUsuario,idDocente,imagenes)
                        values                          ('$idComplaint','EN
ESTUDIO','$location','$idUsuario','','$imagenes')";
                        // $query=$this->conexion
                        $query = $this->conexion->query($consulta);
                        if(!$query){
                                return "E01";
                        }else{
                                $consultaCuestionario   =   "INSERT   INTO
cuestionario (idCuestionario,descripcionCuestionario)
                                values ('$idComplaint','$cuestionario')";
                                $queryCustionario    =    $this->conexion-
>query($consultaCuestionario);
                                if(!$queryCustionario){
                                        return "E02";
                                }else{
                                        return true;
                                }
                        }
                }else{
                        return "E00";
                }
        }
//Función para pasar los datos de "mis denuncias " por método get
        public function getMyComplaints($idUsuario,$filtro = false){
                switch ($filtro) {
                        case 'All':
                                $consultaDenuncias=    "SELECT    *    from
denuncia";
                                break;

                        default:
                                $consultaDenuncias=                "SELECT
idDenuncia,fecha,estado,idDocente       from       denuncia       WHERE
idUsuario='$idUsuario'";
                                break;
                }

                $query=$this->conexion->query($consultaDenuncias);
                $counter=0;
                $idComplaintsArray = array();

                while ($d = $query -> fetch_assoc()) {
                        array_push($idComplaintsArray,$d );
                }
                return $idComplaintsArray;
```

```
        }
//Función para pasar los datos de la asignación del docente
        public function getDocAsigned($idDoc){
                $consulta = "SELECT nombre from docentes WHERE
codigoDocente='$idDoc'";
                $query=$this->conexion->query($consulta);

                $dato = $query -> fetch_assoc();

                return $dato['nombre'];
        }

//Función para pasar datos del usuario desde la denuncia
        public function getDataOfUserForComplaintId($idComplaint){

                $consulta = "SELECT idUsuario from denuncia WHERE
idDenuncia='".$idComplaint."'";

                $query=$this->conexion->query($consulta);
                $data_array = $query->fetch_assoc();

                $idUsuario = $data_array['idUsuario'];

                $consulta_datos_usuario = "SELECT
nombre,direccion,cedula,lugarExpedicion,telefono,email from usuario
WHERE idUsuario='".$idUsuario."'";

                $query=$this->conexion->query($consulta_datos_usuario);
                $datos_usuario = $query->fetch_assoc();

                return $datos_usuario;
        }
//Función para pasar los datos de las denuncias asignadas a docente
        public function getMyAsignedComplaints($idDocente,$filtro =
false){
                switch ($filtro) {
                        case 'All':
                                $consultaDenuncias= "SELECT * from
denuncia WHERE idDocente= '".$idDocente."'";
                                break;

                        default:
                                $consultaDenuncias= "SELECT
idDenuncia,fecha from denuncia WHERE idUsuario='$idUsuario'";
                                break;
                }

                $query=$this->conexion->query($consultaDenuncias);
                $counter=0;
                $idComplaintsArray = array();

                while ($d = $query -> fetch_assoc()) {
                        array_push($idComplaintsArray,$d );
                }
                return $idComplaintsArray;
        }
//Función para pasar los datos de la dirección generada desde la denuncia
```

```
        public function getAdressOfComplaint($idComplaint){
                $consultaGeoRef=  "SELECT  geoReferencia  from  denuncia
WHERE idDenuncia=$idComplaint";
                $query=$this->conexion->query($consultaGeoRef);
                $addr = "";

                while ($a = $query -> fetch_assoc()) {
                        $addr = json_decode($a['geoReferencia'],true);
                }
                return $addr;
        }
//Función para pasar los datos del cuestionario de denuncias

        public function getQuestionary($idComplaint){
         $consultaCuestionario=   "SELECT   descripcionCuestionario   from
cuestionario WHERE idCuestionario=$idComplaint";
                $query=$this->conexion->query($consultaCuestionario);
                $cuestionario = "";
                while ($cuest = $query -> fetch_assoc()) {
                        $cuestionario                                    =
$cuest['descripcionCuestionario'];
                }

                return $cuestionario;
        }

//Función para pasar los datos de la georeferenciacion de las coordenadas
donde se generó la denuncia
        public function getLocationDataOfAll(){
                $consulta = "SELECT * from denuncia";
                $query = $this->conexion->query($consulta);
                $datos = array();
                while ($dato_array = $query -> fetch_assoc()) {
                        $gr = $dato_array['geoReferencia'];
                        $gr = json_decode($gr,true);
                        $addr = $gr['addr'];
                        $coor = $gr['loc'];
                        $estado = $dato_array['estado'];
                        $id = $dato_array['idDenuncia'];
                        $fecha = $dato_array['fecha'];
                        $dato_array                                      =
array('addr'=>$addr,'coordenadas'=>$coor,'estado'=>$estado,'id'=>$id,
'fecha'=>$fecha);
                        array_push($datos, $dato_array);
                }
                return $datos;
        }
//Función para la actualización del estado de la denuncia.
        public function updateState($idComplaint,$state){
                $consultaActualizar=      "UPDATE      denuncia      SET
estado='".$state."' WHERE idDenuncia='$idComplaint'";
                $query = $this->conexion->query($consultaActualizar);

                $result = $this->conexion->affected_rows;

                return $result;
```

```
        }
//Función para el proceso de agregar las notas desde el docente
        public function addNote($id,$note,$first){
                $note = "{\"nota\":"."\"".$note."\"}";
                if(!$first){
                        $oldNotes = $this->getNotes($id);
                        $note = $oldNotes.",".$note;
                }
                $consulta = "UPDATE denuncia SET notas='".$note."' WHERE
idDenuncia='".$id."'";
                        $query = $this->conexion->query($consulta);

                        if($this->conexion->affected_rows < 1){
                                return "E01";
                        }else{
                                return "OK";
                        }
        }

//Función para papsar los datos de las notas
        public function getNotes($id){
                $consulta   =   "SELECT   notas   from   denuncia   WHERE
idDenuncia='$id'";
                $query = $this->conexion->query($consulta);
                $res = $query -> fetch_array();
                $res = $res['notas'];
                return $res;
        }
//Función para mostrar que la denuncia no tiene notas asignadas
        public function noteIsEmpy($id){ //verifica si el campo nota está
vacio
                $empty = false;
                $consultaCuestionario= "SELECT notas from denuncia WHERE
idDenuncia='$id'";
                $query = $this->conexion->query($consultaCuestionario);
                $res = $query -> fetch_array();
                $res = $res['notas'];
                if(strlen($res)<1){
                        $empty = true;
                }
                return $empty;
        }
//Función para pasar los datos en cero de la estructura del código del
docente
        private function getZeros($number){
                if($number < 10){
                        $number = "0000".$number;
                }else if($number > 9 && $number < 100){
                        $number = "000".$number;
                }else if($number > 99 && $number < 1000){
                        $number = "00".$number;
                }else if($number > 999 && $number < 10000){
                        $number = "0".$number;
                }else if($number > 9999 && $number < 100000){
                        $number = $number;
                }
```

```
                return $number;
        }
//Función que genera el consecutivo de las denuncias
        private function consecutiveIdComplaint() {
            $consultLastId= "SELECT idDenuncia from denuncia ORDER BY
idDenuncia DESC LIMIT 1";
                $query = $this->conexion->query($consultLastId);

                $result = $query->fetch_assoc();
                $lastId = $result['idDenuncia'];
                $newId = $lastId + 1;

                $newId = $this->getZeros($newId);
                return $newId;
        }
}

?>
```

Clase login_registration_Class.php

Esta clase permite el intercambio de datos entre el servidor y la base de datos; contiene funciones capaces de autenticar cualquier tipo de usuario; esta se encuentra bajo la ruta ***public_html/unilab_clases,*** la cual se describe a continuación:

Tabla 7.Estructura de la clase login_registration_Class.php, fuente autor

login_registration_Class.php

```
<?php
/**
*@nameProgram: login_registration_Class
*@version 0.1
*@Description: clase que permite el intercambio de datos entre el servidor
y la base de datos
*contiene funciones capaces de autenticar cualquier tipo de usuario
*@Author: unilab development team
*@date: 10/04/2019
**/
//Se inicia la clase Login_registration
class login_registration

//Se crea la conexión a la base de datos
{
        private $conexion=false;
        function __construct($datos_conexion = false)
        {
                if(!$datos_conexion){
                        $server="54.39.133.88";
                        $username='gestorur_unilab';
                        $password='unilab2019.';
                        $db='gestorur_unilab_db_2';
                        /*$server="Localhost";
                        $username='root';
                        $password='';
                        $db='gestorur_unilab_db_2';*/

                        $this->conexion=                                new
mysqli($server,$username,$password,$db);
//Se valida que la conexión a la base de datos
                        if($this->conexion->connect_errno){
                                echo($this->conexion->connect_error);
                        }else{
                                $this->conexion->set_charset("utf8");
                        }
                }
        }
//Función que para el registro de usuario e inserción en la base de datos
        public                                                  function
registerUser($nombre,$direccion,$cedula,$expedicionCedula,$telefono,$us
uario,$email,$clave_hash){
                $id= $this->randomId(5);
                if($this->userRegisterVerify($email,$cedula) < 1){
                        $consulta     =     "INSERT     INTO     usuario
(nombre,direccion,cedula,lugarExpedicion,telefono,nombreUsuario,email,p
assword,idUsuario)
                        values
('$nombre','$direccion','$cedula','$expedicionCedula','$telefono','$usu
ario','$email','$clave_hash','$id')";
                        // $query=$this->conexion
```

```
                        $query = $this->conexion->query($consulta);
                        if(!$query){
                                return "E01";
                        }else{
                                return true;
                        }
                }else{
                        return "E00";
                }
        }
//Función para la verificación de los datos en el registro de usuario
        private function userRegisterVerify($email,$cedula){ //debe ser
privado;
                $consulta= "SELECT * from usuario WHERE email='$email' or
cedula='$cedula'";
                $query=$this->conexion->query($consulta);
                $counter=0;
                while ($query->fetch_assoc()) {
                        $counter++;
                }
                return $counter;

        }
//Función que se encarga de verificar los datos al registrar un usuario
administrador
        private function adminRegisterVerify($email,$cedula){ //debe ser
privado;
                $consulta=   "SELECT   *   from   administrador   WHERE
email='$email' or cedula='$cedula'";
                $query=$this->conexion->query($consulta);
                $counter=0;
                while ($query->fetch_assoc()) {
                        $counter++;
                }
                return $counter;

        }
//Función que se encarga de generar un ID aleatorio para identificar al
usuario en la base de datos

        private function randomId($length) {
    $characters = '0123456789abcdefghijklmnopqrstuvwxyz';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
        }
        /*busca si el usuario existe en la base de datos. retorna true,
        false o credenciales invalidas. Crea una variable de sesion con
los datos del usuario*/
        public function getUserWithPass($email,$clave_hash){
                session_start();
                $query=$this->conexion->query("SELECT   *   from   usuario
WHERE
```

```
                        email='".$email."' AND password='".$clave_hash."'
");
                $result=$query->fetch_array();
                if(isset($result)){
                        $datos = array('Nombre' =>$result['nombre'] ,
                                'Direccion'=>   $result['direccion']    ,
'Cedula' => $result['cedula'] ,
                                'Telefono'   =>   $result['telefono']   ,
'Email' => $result['email'] ,
                                'Id'     =>     $result['idUsuario']     ,
'LugarExpedicion' => $result['lugarExpedicion']);
                        $_SESSION['usuario'] = $datos;
                        return true;
                }else{
                        unset($_SESSION['usuario']);
                        return false;
                }
        }
        public  function  getUserWithPassNoSession($email,$clave_hash){
//Busca si el usuario existe en la base de datos con la clave enviada,
identico a getUserWithPass pero no inicia sesion, solo envía true o false
segun sea el caso
                session_start();
                $query=$this->conexion->query("SELECT   *   from   usuario
WHERE
                        email='".$email."' AND password='".$clave_hash."'
");
                $result=$query->fetch_array();
                if(isset($result)){
                        return true;
                }else{
                        return false;
                }
        }

        //Función que se encarga de agregar  al usuario docente a la
tabla.
        public                                                  function
registerDoc($nombre,$direccion,$cedula,$expedicionCedula,$telefono,$usu
ario,$email,$codigo_docente,$clave_hash){
                $id= $this->randomId(5);
                if($this->userRegisterVerify($email,$cedula) < 1){
                        $consulta     =     "INSERT     INTO     docentes
(nombre,direccion,cedula,lugarExpedicion,telefono,email,codigoDocente,p
assword,idDocente)
                        values
('$nombre','$direccion','$cedula','$expedicionCedula','$telefono','$ema
il','$codigo_docente','$clave_hash','$id')";
                        // $query=$this->conexion
                        $query = $this->conexion->query($consulta);
                        if(!$query){
                                return "E01";
                        }else{
                                return true;
                        }
                }else{
                        return "E00";
```

```
                }
        }
Función que se encarga de pasar y validar los datos del docente

        public function getDataFromCode($codigo){
                $consulta    =    "SELECT    *    FROM    docentes    WHERE
codigoDocente='".$codigo."'";
                $query = $this->conexion->query($consulta);
                $datos = $query->fetch_assoc();
                if(sizeof($datos) > 0){
                        return json_encode($datos,true);
                }else{
                        return "codigo de docente inválido";
                }


        }
//Función que se encarga de de registrar los datos del administrador en
la base de datos
        public                                                  function
registerAdmin($nombre,$direccion,$cedula,$expedicionCedula,$telefono,$e
mail,$clave_hash){
                $id= $this->randomId(5);
                if($this->adminRegisterVerify($email,$cedula) < 1){
                        $consulta    =    "INSERT    INTO    administrador
(nombre,direccion,cedula,lugarExpedicion,telefono,email,password,idAdmi
nistrador)
                        values
('$nombre','$direccion','$cedula','$expedicionCedula','$telefono','$ema
il','$clave_hash','$id')";
                        // $query=$this->conexion
                        $query = $this->conexion->query($consulta);
                        if(!$query){
                                return "E01";
                        }else{
                                return true;
                        }
                }else{
                        return "E00";
                }
        }

        /*busca si el usuario administrador existe en la base de datos.
retorna true,
        false o credenciales invalidas. Crea una variable de sesion con
los datos del usuario*/
        public function getAdminWithPass($email,$clave_hash){
                session_start();
                $query=$this->conexion->query("SELECT       *       from
administrador WHERE
                        email='".$email."' AND password='".$clave_hash."'
");
                $result=$query->fetch_array();
                if(isset($result)){
                        $datosAdmin        =        array('Id'        =>
$result['idAdministrador'] , 'Email' => $result['email']);
                        $_SESSION['admin'] = $datosAdmin;
```

```
                        return true;
                }else{
                        unset($_SESSION['admin']);
                        return false;
                }
                /*$query=$this->conexion->query("SELECT * from admin
WHERE
                        email='".$email."' AND pass='".$clave_hash."' ");
                $result=$query->fetch_array();
                if(isset($result)){
                        $datosAdmin = array('Id' => $result['id'] ,
'Email' => $result['email']);
                        $_SESSION['admin'] = $datosAdmin;
                        return true;
                }else{
                        unset($_SESSION['admin']);
                        return false;
                }*/
        }

        /*busca si el usuario docente existe en la base de datos. retorna
true,
        false o credenciales invalidas. Crea una variable de sesion con
los datos del usuario administrador*/
        public function getDocWithPass($email,$clave_hash){
                session_start();
                $query=$this->conexion->query("SELECT * from docentes
WHERE
                        email='".$email."' AND password='".$clave_hash."'
");
                $result=$query->fetch_array();
                if(isset($result)){
                        $datosDoc = array('Id' =>
$result['codigoDocente'] , 'Email' => $result['email']);
                        $_SESSION['teacher'] = $datosDoc;
                        return true;
                }else{
                        unset($_SESSION['teacher']);
                        return false;
                }
        }

        public function getDocWithPassNoSession($email,$clave_hash){
//Busca si el usuario existe en la base de datos con la clave enviada,
identico a getUserWithPass pero no inicia sesion, solo envía true o false
segun sea el caso
                session_start();
                $query=$this->conexion->query("SELECT * from docentes
WHERE
                        email='".$email."' AND password='".$clave_hash."'
");
                $result=$query->fetch_array();
                if(isset($result)){
                        return true;
                }else{
                        return false;
                }
```

```
        }

        public function updatePass($email,$pass){
                $pass_hash = md5($pass);
                $query=$this->conexion->query("UPDATE    usuario    SET
password='".$pass_hash."' WHERE
                        email='".$email."'");
                $result = $this->conexion->affected_rows;
                if($result > 0){
                         return true;
                }else{
                        return false;
                }
                return $result;
        }

//Función para actualizar los datos del usuario
        public                                                  function
updateDataUser($Nombre,$Direccion,$Cedula,$ExpedicionCedula,$Telefono,$
Email,$id){
                $id = trim($id);
                $query=$this->conexion->query("UPDATE    usuario    SET
nombre='".$Nombre."', direccion='".$Direccion."', cedula='".$Cedula."',
lugarExpedicion='".$ExpedicionCedula."',      telefono='".$Telefono."',
email='".$Email."' WHERE idUsuario='".$id."'");
                $result = $this->conexion->affected_rows;
                if($result > 0){
                         return true;
                }else{
                        return false;
                }
                return $result;
        }
//Función que se encarga de actualizar la contraseña del docente
        public function updatePassDoc($email,$pass){
                $pass_hash = md5($pass);
                $query=$this->conexion->query("UPDATE    docentes    SET
password='".$pass_hash."' WHERE
                        email='".$email."'");
                $result = $this->conexion->affected_rows;
                if($result > 0){
                         return true;
                }else{
                        return false;
                }
                return $result;
        }
//Función para actualizar los datos del docente
        public                                                  function
updateDataDoc($Nombre,$Direccion,$Cedula,$ExpedicionCedula,$Telefono,$E
mail,$codigoDocente){
                $codigo = trim($codigoDocente);
                $query=$this->conexion->query("UPDATE    docentes    SET
nombre='".$Nombre."', direccion='".$Direccion."', cedula='".$Cedula."',
lugarExpedicion='".$ExpedicionCedula."',      telefono='".$Telefono."',
email='".$Email."' WHERE codigoDocente='".$codigo."'");
                $result = $this->conexion->affected_rows;
```

```
                if($result > 0){
                         return true;
                }else{
                        return "Los datos no tienen cambio";
                }
                return $result;
        }
// Verifica que exista un usuario registrado en la base de datos
ingresando su email    public function isUser($email){
                $query=$this->conexion->query("SELECT   *   from   usuario
WHERE
                        email='".$email."'");
                $result = $query->fetch_array();
                if(isset($result)){
                        return true;
                }else{
                        return false;
                }
        }

}


?>
```

Clase users_data_class.php

Esta clase, está dedicada a la gestión de los datos de un usuario ciudadano. Se encuentra bajo la ruta: ***public_html/unilab_clases,*** a continuación se detallan cada una de sus funciones:

Tabla 8.codifiacion de la clase user_data_cass, fuente autor

users_data_class.php

```
<?php

/**
 *
 */
// Clase dedicada a la gestión de los datos de un usuario ciudadano.
class userData
{

        function __construct()
        {
                # code...
        }

        function getAleatoryId($length){
            $characters = '0123456789';
            $charactersLength = strlen($characters);
            $randomString = '';
            for ($i = 0; $i < $length; $i++) {
                $randomString .= $characters[rand(0, $charactersLength -
1)];
            }
            return $randomString;
                }
}

?>
```

6. FUNCIONAMIENTO DE LA APLICACIÓN

En las tablas anteriores pudimos observar las clases principales del proyecto; acompañado de eso es necesario describir los archivos de cada uno de los módulos del sistema para tener claridad de su funcionamiento. A continuación, se procede a describir los archivos contenidos en las carpetas: **account, Admin-account** y **teacher account**

Descripción de los archivos en la carpeta *account*

El archivo imagenes-denuncia.php contiene la validación para las tres imágenes a la hora de que el ciudadano adjunta los archivos en el momento que genera la denuncia:

Tabla 9.Estructura del archivo imagenes-denuncia.php, fuente autor

Imagenes-denuncia.php

```
<?php
//Se crea la conexión a la base de datos
$idD=$_POST['idDenuncia'];
$rID= substr($idD, 1);
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
$conexion= new mysqli($server,$username,$password,$db);
$consulta= "SELECT * FROM denuncia WHERE  idDenuncia = $rID";
$query = mysqli_query($conexion, $consulta);
while ($imgD=mysqli_fetch_array($query)){
    $json = $imgD['imagenes'];
    if($json==null){
//Se validan los datos adjuntos de la denuncia
        echo "No contiene adjuntos";
    }else{
        $obj = json_decode($json);
        echo  "<div class='container'>
                <div class='row'>
                    <div class='col'>
//Se muestran las imágenes guardadas en la carpeta denuncias
                    N#1
                    <img  src='../denuncias/imagenes/".$obj->{'img1'}."'
width='100%' >
                    </div>
                    <div class='col'>
                    N#2
                    <img  src='../denuncias/imagenes/".$obj->{'img2'}."'
width='100%'>
                    </div>
                </div>
                <br />
                <div class='row'>
                    <div class='col'>
                    N#3
                    <img  src='../denuncias/imagenes/".$obj->{'img3'}."'
width='100%'>
                    </div>
                </div>
            </div>";
    }
```

```
  }
?>
```

En la siguiente tabla se muestra la codificación del proceso para la gestión de notas de la denuncia que genera el ciudadano:

Tabla 10.Estructura del archivo notas-denuncia.php, fuente autor

Notas-denuncia.php

```
<?php
//Se crea la conexión a la base de datos
$idD=$_POST['idDenun'];
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
$conexion= new mysqli($server,$username,$password,$db);
$consulta= "SELECT notas FROM denuncia WHERE  idDenuncia = $idD ";
$query = mysqli_query($conexion, $consulta);

$respuesta="";
//Se crea un array con las notas que el docente va generando desde el
módulo
while ($note=mysqli_fetch_array($query)){
    $r= str_replace("},{","",$note['notas']);
    $r2= str_replace('nota":',"",$r);
    $r3= str_replace('""',"",$r2);
    $r4= str_replace('"',"-",$r3);
    $r5= str_replace('{',"-",$r4);
    $r6 = trim($r5, '-}');
    Se valida si el campo contiene notas o no.
    $myString = str_replace('-',"<br />",$r6);
    if($myString==null){
        $respuesta="Esta denuncia no tiene notas";
    }else{

        $respuesta= $myString;
    }

 }

    echo $respuesta;
 ?>
```

Esta tabla muestra la estructura de inicio en la carpeta account en la cual se muestra el fondo de pantalla, se carga el mapa y se muestran las posiciones de las denuncias que han sido generadas

Tabla 11.Estructura del archivo secreen_home .php, fuente autor

screen_home.php

```
<div id="home">
//Se carga el fondo de la aplicación
  <img src="./img/fondo.png">
</div>
<div  id ="mapa" class="table-responsive" style="display: none;">
  <div id="mapas">Cargando mapa...</div>
</div>

<script type="text/javascript">
//Función para tomar las posiciones donde se generó la denuncia

  function getLocations(){

  $.post("./gestorUbicaciones.php",{},function(data,status){
    var pointers = JSON.parse(data);
    // $("#u").html(data+"<br><br>"+point.coordenadas);
    showPosition_all(pointers);
  });

}
//Funciónes para pasar los datos de las coordenadas(Latitud y longitud)

function getLat(coor){
  var string = coor;
  var i = string.indexOf(",");
  var lat = string.slice(0,i);
  return lat;
}

function getLon(coor){
  var string = coor;
  var i = string.indexOf(",");
  var lon = string.slice(i+1,string.length);
  return lon;
}

function showPosition_all(json_data){
  var uluru2 = {lat: 10.4032694, lng: -75.511243};
//Ubica las la posición en el mapa de acuerdo a las variables

    var mapas = new google.maps.Map(
      document.getElementById('mapas'), {zoom: 13, center: uluru2});
```

```
    var infoWindowContent = [];
    for(i=0; i<json_data.length; i++){
      point = json_data[i];
      infoWindowContent.push(['<div class="info_content">' +
        '<h3>'+point.estado+'</h3>' +
        '<p>Denuncia: '+point.id+'</p>'+
        '<p>'+point.addr+'</p>'+
        '<p>'+point.fecha+'</p></div>']);
    }
//Condicional para colocar el punto en el mapa
    for(i=0; i<json_data.length; i++){
      point = json_data[i];
      var lati = getLat(point.coordenadas);
      var long = getLon(point.coordenadas);
      lati = parseFloat(lati);
      long = parseFloat(long);
      var pos = {lat:lati, lng:long};
      var marker2 = new google.maps.Marker({position: pos, map: mapas,
        title: 'Uluru (Ayers Rock)',
        label:{
          text:i,
          color:'white'
        },//Se carga el icono Locate.png
        icon:{
          url:'img/Locate.png',
          scaledSize: new google.maps.Size(50, 50),
          origin: new google.maps.Point(0,0), // origin
          anchor: new google.maps.Point(0, 0)
        }
      });

      var contentString = '<div id="content">'+
      "hola <br> probando <br> sonido "+ point.estado+
      '</div>';



      var infoWindow = new google.maps.InfoWindow(), marker2, i;

      google.maps.event.addListener(marker2, 'click', (function(marker2,
i) {
            return function() {
                infoWindow.setContent(infoWindowContent[i][0]);
                infoWindow.open(mapas, marker2);
            }
        })(marker2, i));

        // Se centra el contenido ene l mapa


    }

  }

</script>

<script>
```

```
  // Initialize and add the map
  function initMap() {
    // The location of Uluru

  }
</script>

<style>
Se colocan los elementos en el mapa de acuerdo a el tamaño del div
#mapas {
  height: 500px;  /* The height is 400 pixels */
  width: 100%  ;  /* The width is the width of the web page */
}
</style>
```

En la siguiente tabla se muestra la estructura del código relacionado a la vista modal de las notas que genera el docente y posteriormente son visualizadas por el usuario ciudadano:

Tabla 12.Estructura del archivo Modalnotas .php, fuente autor

modalNotas.php

```
//Inicio del div para el modal Ver notas
<div class="modal fade" id="modalVerNotas" tabindex="-1" role="dialog"
aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
//Titulo de la vista
        <h5 class="modal-title" id="exampleModalCenterTitle">Notas de la
denuncia</h5>
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
//Se muestra el id el titulo y la tabla de las notas
      <div class="modal-body">
        <label id="lblIdNoteShow"></label>
        <table class="table">
          <thead>
            <tr>
              <th scope="col" id="tituloNotas">Nota</th>
            </tr>
          </thead>
          <tbody id="tablaNotas">
          </tbody>
        </table>
      </div>
      <div class="modal-footer">
        <button    type="button"    class="btn    btn-primary"    data-
dismiss="modal">OK</button>
      </div>
    </div>
  </div>
</div>
//Estas etiquetas permiten el desenfoque del fondo mientras se activa el
modal
<div    class="modal    fade"    id="modal-info-result"    tabindex="-1"
role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="titulo-info-result"></h5>
```

```
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body" id="cuerpo-info-result">

      </div>
//Estilo del botón cerrar
      <!-- <div class="modal-footer">
        <button    type="button"    class="btn    btn-secondary"    data-
dismiss="modal">Close</button>
        <button      type="button"      class="btn      btn-primary">Save
changes</button>
      </div> -->
    </div>
  </div>
</div>
```

La siguiente tabla muestra la estructura de la vista modal del cuestionario, el cual es diligenciado por el usuario ciudadano, al igual que las funciones que validan las preguntas marcadas como “NO”, la información de cada una de las preguntas, y la función que envía los resultados al formato PDF:

Tabla 13. Estructura del modalcuestionario .php, fuente autor

modalCuestionario.php

```
// Se crea la conexión a la base de datos
<?php
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
$conexion= new mysqli($server,$username,$password,$db);

?>
//Div que contiene los datos del enlace a nueva denuncia
<div class="modal fade NuevaDenuncia" id="cuestionario" tabindex="-1"
role="dialog" aria-labelledby="myExtraLargeModalLabel" aria-
hidden="true">
<div class="modal-dialog modal-xl">
    <div class="modal-content">
      <div class="modal-header">
//Titulo de la vista
        <h5 class="modal-title" >DILIGENCIAR CUESTIONARIO</h5>
        <br>
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <table class="table table-hover">
            <thead>
//Se inicia el formato con cada una de las preguntas
                              <th colspan="3" style="text-
aling:center;color:white;" class="bg-danger">Formato</th>
        </thead>
        <tbody>
              <tr>
                  <td>¿Cuántos pisos tiene la obra?
                    <i class="fas fa-fw fa-info" onclick="info(0)"
style="cursor:pointer;"></i>
                  </td>
                  <td colspan="2">
                    <input type="number" id="inputAnswer1" class="form-
control" value="0">
```

```
                    </td>
            </tr>
                        <?php
                        $i=1;
            $consulta= "SELECT * from preguntas_cuetionario";
                        $query = mysqli_query($conexion, $consulta);
                        while ($nt=mysqli_fetch_array($query)){

            ?>
              <tr>
                <td>¿<input id="pc<?php echo $i+1;?>" style='display:
none'     value="<?php     echo     $nt['pregunta'];?>"/><?php     echo
$nt['pregunta'];?>?
                  <i class="fas fa-fw fa-info" onclick="info(<?php echo
$i;?>)" style="cursor:pointer;"></i>
                </td>
                <td>
                  <input   type="radio"   id="radioAnswer_<?php   echo
$i+1;?>_s" name="Answer_<?php echo $i+1;?>" class="custom-control-input"
value="SI">
                  <label               class="custom-control-label"
for="radioAnswer_<?php echo $i+1;?>_s">SI</label>
                  </td>
                  <td>
                  <input  type="radio"    id="radioAnswer_<?php  echo
$i+1;?>_n" name="Answer_<?php echo $i+1;?>" class="custom-control-input"
value="NO">
                  <label               class="custom-control-label"
for="radioAnswer_<?php echo $i+1;?>_n">NO</label>
                  </td>
              </tr>
                        <?php
                        $i=$i+1;
                       }
                       echo    "<div    id='numeroP'    style='display:
none'>$i</div>";
                        ?>
          </tbody>
                 </table>
                 <form enctype="multipart/form-data">
                       <h6><b>Agregue 3 imagenes</b></h6>
                       <div class="row">
                       <div class="col">
                                       <div class="custom-file">
                                              <input       type="hidden"
name="MAX_FILE_SIZE" value="10000000"/>
                                              <input         type="file"
class="custom-file-input" name="archivo">
                                              <label class="custom-file-
label" for="customFile">Seleccione imagen</label>
                                       </div>
                                       <div id="imageLoadResult">

                                       </div>
                                </div>
                                <div class="col">
                                       <div class="custom-file">
```

```
                                                <input       type="hidden"
name="MAX_FILE_SIZE" value="10000000"/>
                                                <input         type="file"
class="custom-file-input" name="archivo1">
                                                <label class="custom-file-
label" for="customFile">Seleccione imagen</label>
                                        </div>
                                        <div id="imageLoadResult1">

                                        </div>
                                </div>
                                <div class="col">
                                        <div class="custom-file">
                                                <input       type="hidden"
name="MAX_FILE_SIZE" value="10000000"/>
                                                <input         type="file"
class="custom-file-input" name="archivo2">
                                                <label class="custom-file-
label" for="customFile">Seleccione imagen</label>
                                        </div>
                                        <div id="imageLoadResult2">

                                        </div>
                                </div>
                        </div>
                </form>
//Se inicia el div donde se encuentra el campo de texto para digitar el
barrio
          <div class="form-group">
            <label for="txtDirecion">Escriba El Barrio</label>
            <input  type="text"  class="form-control"  id="txtDirecion"
placeholder="Av. pedro de heredia No 12">
          <label for="btnLocation" id="lblLocation">...</label><br>
          <button     class="btn     btn-warning"     id="btnLocation"
onclick="getMyLocation()">Agregar Ubicacion</button><br><br>
        </div>
        <div id="map"></div>
          <label id="consola-denuncias" style="color:red"></label>
          <div class="modal-footer">
          <button class="btn btn-success" onclick="getAnswerResult(<?php
echo $i; ?>)">Enviar resultados</button><br><br>
          <button class="btn btn-info" onclick="refresh()">Volver al menu
principal</button>
          </div>
      </div>
    </div>
  </div>
</div>

<!-- MODAL INFO-->
<div class="modal fade" id="modal-info" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">info</h5>
```

```
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body" id="info">

      </div>
    </div>
  </div>
</div>

<!—Se muestra la vista modal con la información de que la denuncia se ha
generado correctamente -->
<div   class="modal   fade"   id="modal-denuncia-success"   tabindex="-1"
role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5     class="modal-title"     id="exampleModalLabel">Denuncia
generada</h5>
        <button type="button" class="close" onclick="refresh()" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        Se generó la denuncia satisfactoria mente.
      </div>
    </div>
  </div>
</div>
<div    class="modal    fade"    tabindex="-1"    role="dialog"    aria-
labelledby="mySmallModalLabel"     aria-hidden="true"     id="mi-modal-
denunciaC">
  <div class="modal-dialog modal-sm">
    <div class="modal-content">
      <div class="modal-header">
        <h5       class="modal-title"       id="myModalLabel">Confirmar
Denuncia</h5>
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" id="modal-btn-si-
denuncia">Si</button>
        <button type="button" class="btn btn-primary" id="modal-btn-no-
denuncia">No</button>
      </div>
    </div>
  </div>
</div>
</div>

<script type="text/javascript">
//Función que muestra la información correspondiente a cada una de las
preguntas
```

```
function info(id){
        switch(id){
                case 0:
                        $("#modal-info").modal('show');
                        $("#info").html("sta información corresponde a
que las licencias de construcción expedidas contienen el número de pisos
permitidos, ya sea por el suelo donde se ubica la construcción, el tipo
de sector o localidad. Si una obra contiene más de los pisos estipulados
en la licencia, infringe una norma.");
                        break;
                case 1:
                        $("#modal-info").modal('show');
                        $("#info").html("Sin información asociada");
                        break;
                case 2:
                        $("#modal-info").modal('show');
                        $("#info").html("Esta información responde a que
la obra debe tener en su estructura (ya sea por el número de pisos o
apartamentos que esta tenga), la capacidad de albergar el posible número
de vehículos relacionados a la misma. Si esta no cuenta con lo mencionado
ocasiona estacionamientos en vías, congestionando el flujo vehicular e
infringiendo normas urbanísticas y de convivencia.");
                        break;
                case 3:
                        $("#modal-info").modal('show');
                        $("#info").html("Cualquier inmueble que se
construya en la ciudad debe tener una valla donde se informe la clase de
obra que se está realizando: para vivienda, comercio, industria o de tipo
institucional. También debe tener el número de la licencia de construcción
o de urbanización, o los dos. Además, áreas, número de soluciones, si son
varias, e identificación de constructor y del arquitecto proyectista,
entre otros datos. Por lo tanto, cuando usted vea una obra que no tiene
valla de identificación, puede estar seguro de que está infringiendo las
normas, en particular el artículo 520 del Acuerdo 6 de 1990 y 16 del
Decreto Nacional 958 de 1992, sobre nuevos procedimientos para la
obtención de las licencias de construcción y urbanización.");
                        break;
                case 4:
                        $("#modal-info").modal('show');
                        $("#info").html("Esta información responde a si el
espacio donde se desarrolla la obra corresponde a un bien inmueble
público, por ende infringe un derecho al ciudadano sobre el bien
colectivo.");
                        break;
                case 5:
                        $("#modal-info").modal('show');
                        $("#info").html("Se refiere a la protección con
que cuenta las obras ubicadas cerca andenes o transito continuo de
personas cerca de la misma, para evitar accidentes. (Ya sea señalizaciones
o letreros de precaución, etc.) También responde a si está en el proceso
de construcción no deja a fácil acceso materiales o herramientas que
puedan causar daño al transeúnte.");
                        break;
                case 6:
                        $("#modal-info").modal('show');
```

```
                        $("#info").html("Se refiere este punto a las
mallas o barreras de protección.");
                        break;
                case 7:
                        $("#modal-info").modal('show');
                        $("#info").html("Se refiere a si el proceso de
construcción se desarrolla en el espacio público donde puede generar
problemas de convivencia, o si bien daños a bienes colectivos.");
                        break;
                case 8:
                        $("#modal-info").modal('show');
                        $("#info").html("Sin información asociada");
                        break;
                case 9:
                        $("#modal-info").modal('show');
                        $("#info").html("Sin información asociada");
                        break;
        }
}

$('#modal-info').on('hidden.bs.modal', function (e) {
  $('#md').modal('show');
})
//Función que toma la georeferencia donde se genera la denuncia
function getMyLocation(){
        $("#lblLocation").html("espere...");
        var coor;
        if(navigator.geolocation){
                navigator.geolocation.getCurrentPosition(showPosition);
        }else{
                $("#lblLocation").html("su navegador no soporta esta
funcion");
        }
}
//Función para mostrar la posición en el mapa de donde se generó la
denuncia
function showPosition(position){
        var latitude = position.coords.latitude;
        var longitude = position.coords.longitude;
        $("#lblLocation").html(latitude+","+longitude);

        var uluru = {lat: latitude, lng: longitude};
          // The map, centered at Uluru
          var map = new google.maps.Map(
              document.getElementById('map'), {zoom: 13, center: uluru});
          // The marker, positioned at Uluru
          //  var label = new google.maps.MarkerLabel({color:'red',
text:'hola'})
          var marker = new google.maps.Marker({position: uluru, map:
map});
          // marker2.setLabel("hola");
}
//Función que valida que no estén vacios los campos de las preguntas del
cuestionario
function getAnswerResult(num){

        var idUsuario = document.getElementById("lblId").textContent;
```

```
        var                                location                                =
document.getElementById("lblLocation").textContent;
        var direccion = new String($("#txtDirecion").val());
        var ubicacion = "";
        strl = new String(location);
        var data = new FormData();
        data.append('v',"cvr");
        if(!($("#inputAnswer1").val())  ||  ($("#inputAnswer1").val()  <
1)){
                        alert("Ingrese un valor válido en la pregunta 1");
        }
        for (var i = 1; i < num; i++) {
                var cadena1= "#radioAnswer_"+(i+1)+"_s:checked";
                var cadena2= "#radioAnswer_"+(i+1)+"_n:checked";
                if(!($(cadena1).val()) && !($(cadena2).val())){
                        alert("Responda la pregunta "+(i+1));
                        }
        }
//Estructura de control que valida que las imágenes se suban en el formato
establecidos

        if($("#img1").val() == "ErrorImg1" ){
                alert("Formato  Incorrecto  en  ajusto  de  archivo  N#1.
Verifique su archivo");
        }else if($("#img2").val() == "ErrorImg2"){
                alert("Formato  Incorrecto  en  ajusto  de  archivo  N#2.
Verifique su archivo");
        }else if($("#img3").val() == "ErrorImg3"){
                alert("Formato  Incorrecto  en  ajusto  de  archivo  N#3.
Verifique su archivo")
        }else if(strl.length <= 9){
                alert("Agregue su ubicacion");
        }else if(direccion.length <= 5){
                alert("Ingrese una direccion más detallada");
        }else{

        $("#mi-modal-denunciaC").modal('show');
    var modalConfirm = function(callback){
      $("#modal-btn-si-denuncia").on("click", function(){
        callback(true);
        $("#mi-modal-denunciaC").modal('hide');
      });

      $("#modal-btn-no-denuncia").on("click", function(){
        callback(false);
        $("#mi-modal-denunciaC").modal('hide');
      });
    };
//Al confirmarse la acción se validan las preguntas que fueron marcadas
como “NO” para enviarse al PDF

    modalConfirm(function(confirm){
      if(confirm){
                jQuery.each($('input[type=file]')[0].files,   function(i,
imgF) { data.append('file-'+0, imgF); });
                jQuery.each($('input[type=file]')[1].files,   function(i,
imgF) { data.append('file-'+1, imgF); });
```

```
                jQuery.each($('input[type=file]')[2].files,  function(i,
imgF) { data.append('file-'+2, imgF); });
        $("#consola-denuncias").html("Generando denuncia, espere...");

       ubicacion="{\"addr\":\""+direccion+"\",\"loc\":\""+location+"\""
+"}";
                var      AnswerAll      =      "La      obra      tiene:
"+$("#inputAnswer1").val()+" pisos"; // respuesta 1 si
                var Answer_2_s = $("#radioAnswer_2_s:checked").val(); //
respuesta 2 si
                // var Answer_2_n = $("#radioAnswer_2_n:checked").val();
// respuesta 2 no
                var Answer_3_s = $("#radioAnswer_3_s:checked").val(); //
respuesta 3 si
                // var Answer_3_n = $("#radioAnswer_3_n:checked").val();
// respuesta 3 no
                // var Answer_4_s = $("#radioAnswer_4_s:checked").val();
// respuesta 4 si
                var Answer_4_n = $("#radioAnswer_4_n:checked").val(); //
respuesta 4 no
                var Answer_5_s = $("#radioAnswer_5_s:checked").val(); //
respuesta 5 si
                // var Answer_5_n = $("#radioAnswer_5_n:checked").val();
// respuesta 5 no
                var Answer_6_s = $("#radioAnswer_6_s:checked").val(); //
respuesta 6 si
                // var Answer_6_n = $("#radioAnswer_6_n:checked").val();
// respuesta 6 no
                var Answer_7_s = $("#radioAnswer_7_s:checked").val(); //
respuesta 7 si
                // var Answer_7_n = $("#radioAnswer_7_n:checked").val();
// respuesta 7 no
                var Answer_8_s = $("#radioAnswer_8_s:checked").val(); //
respuesta 7 si
                // var Answer_8_n = $("#radioAnswer_8_n:checked").val();
// respuesta 7 no
                // var Answer_9_s = $("#radioAnswer_9_s:checked").val();
// respuesta 7 si
                var Answer_9_n = $("#radioAnswer_9_n:checked").val(); //
respuesta 7 no
                var  Answer_10_s  =  $("#radioAnswer_10_s:checked").val();
// respuesta 7 si
                //              var             Answer_10_n             =
$("#radioAnswer_10_n:checked").val(); // respuesta 7 no

                if(Answer_2_s){ AnswerAll += ", "+$("#pc2").val();}
                // if(Answer_2_n){var Answer_2 = Answer_2_n;}
                if(Answer_3_s){AnswerAll += ", "+$("#pc3").val();}
                // if(Answer_3_n){var Answer_3 = Answer_3_n;}
                if(Answer_4_n){AnswerAll += ", "+$("#pc4").val();}
                // if(Answer_4_n){var Answer_4 = Answer_4_n;}
                if(Answer_5_s){AnswerAll += ", "+$("#pc5").val();}
                // if(Answer_5_n){var Answer_5 = Answer_5_n;}
                if(Answer_6_s){AnswerAll += ", "+$("#pc6").val();}
                // if(Answer_6_n){var Answer_6 = Answer_6_n;}
                if(Answer_7_s){AnswerAll += ", "+$("#pc7").val();}
                // if(Answer_7_n){var Answer_7 = Answer_7_n;}
```

```
                if(Answer_8_s){AnswerAll += ", "+$("#pc8").val();}
                // if(Answer_8_n){var Answer_8 = Answer_8_n;}
                if(Answer_9_n){AnswerAll += ", "+$("#pc9").val();}
                // if(Answer_9_n){var Answer_9 = Answer_9_n;}
                if(Answer_10_s){AnswerAll += ", "+$("#pc10").val();}
                // if(Answer_10_n){var Answer_10 = Answer_10_n;}

                var cuestionario = AnswerAll;
                //ubicacion =
                //
$.post("./gestorDenuncia.php",{idUsuario:idUsuario,ubicacion:ubicacion,
cuestionario:cuestionario},function(data,status){

                // });
                data.append('idUsuario',idUsuario);
                data.append('ubicacion',ubicacion);
                data.append('cuestionario',cuestionario);
                $.ajax({
            type: 'POST',
            url: "./gestorDenuncia.php",
            data: data,
            cache: false,
                   contentType: false,
                   processData: false,
                       method: 'POST',
// Mostramos un mensaje con la respuesta de PHP
            success: function(data) {
                if(data == true){
                       $("#md").modal('hide');
                                        $("#modal-denuncia-
success").modal('show');
                                        alert("Su      denuncia      sera
visualizada cuando el Administrador la asigne a un docente");
                                }else{
                                        $("#consola-denuncias").html("");
                                        alert(data);
                                        $("#consola-denuncias").html("");
                                }
            }
         });

      }else{
 //Acciones si el usuario no confirma
        //$("#result").html("NO CONFIRMADO");
      }
      });

        }
}

function imageVerify(){
        var data = new FormData();
        jQuery.each($('input[type=file]')[0].files, function(i, file) {
                data.append('file-'+0, file);
        });
        jQuery.each($('input[type=file]')[1].files, function(i, file) {
                data.append('file-'+1, file);
```

```
        });
        jQuery.each($('input[type=file]')[2].files, function(i, file) {
                    data.append('file-'+2, file);
        });
        $.ajax({
            type: 'POST',
            url: "leerImg.php",
            data: data,
            cache: false,
                   contentType: false,
                   processData: false,
                   method: 'POST',
 // Mostramos un mensaje con la respuesta de PHP
            success: function(data) {
                               var datos = JSON.parse(data);
                                                          numeroArchivos    =
datos.numeroArchivos;
                                                          nombreImagen1     =
datos.nombreImagen1;
                                                          nombreImagen2     =
datos.nombreImagen2;
                                                          nombreImagen3     =
datos.nombreImagen3;
                                                          var      img1     =
nombreImagen1.substring(nombreImagen1.length - 4, nombreImagen1.length);
                                                          var      img2     =
nombreImagen2.substring(nombreImagen2.length - 4, nombreImagen2.length);
                                                          var      img3     =
nombreImagen3.substring(nombreImagen3.length - 4, nombreImagen3.length);
                                                          //
$("#imageLoadResult1").html("datos: "+datos.numeroArchivos);

        if(nombreImagen1.length > 0){

        if(img1==".jpg" || img1==".png" || img1=="jpeg" || img1==".JPG"
|| img1==".PNG" || img1=="JEPG"){

        $("#imageLoadResult").html("<i class='fas fa-fw fa-check-circle'
style='cursor:pointer;color: green'>  </i>");
                                                                   }else{

        $("#imageLoadResult").html("<input  id='img1'  value='ErrorImg1'
style='display:none'/><p    style='cursor:pointer;color:    red'>Formato
Incorrecto. Debe ser .JPG , .PNG o .JPEG</p>");
                                                                    }
                                                          }else{

        $("#imageLoadResult").html("<i  class='fas  fa-fw  fa-exclamation-
triangle' style='cursor:pointer;color: red'></i>");
                                                          }

        if(nombreImagen2.length > 0){

        if(img2==".jpg"||img2==".png"||img2=="jpeg"||img2==".JPG"||img2=
=".PNG"||img2=="JEPG"){
```

```
        $("#imageLoadResult1").html("<i   class='fas   fa-fw   fa-check-
circle' style='cursor:pointer;color: green'>  </i>");
                                                                }else{

        $("#imageLoadResult1").html("<input  id='img2'  value='ErrorImg2'
style='display:none'/><p    style='cursor:pointer;color:    red'>Formato
Incorrecto. Debe ser .JPG , .PNG o .JPEG</p>");
                                                                }
                                                        }else{

        $("#imageLoadResult1").html("<i class='fas fa-fw fa-exclamation-
triangle'  style='cursor:pointer;color: red'></i>");
                                                        }

        if(nombreImagen3.length > 0){
                                                                if(img3  ==
".jpg"||img3==".png"||img3=="jpeg"||img3==".JPG"||img3==".PNG"||img3==""
JEPG"){

        $("#imageLoadResult2").html("<i   class='fas   fa-fw   fa-check-
circle' style='cursor:pointer;color: green'>  </i>");
                                                                }else{

        $("#imageLoadResult2").html("<input  id='img3'  value='ErrorImg3'
style='display:none'/><p    style='cursor:pointer;color:    red'>Formato
Incorrecto. Debe ser .JPG , .PNG o .JPEG</p>");
                                                                }
                                                        }else{

        $("#imageLoadResult2").html("<i class='fas fa-fw fa-exclamation-
triangle' style='cursor:pointer;color: red'></i>");
                                                        }

            }
        })
}

function autoplay_imageVerify() {
        interval = setInterval(function(){
                imageVerify();
        }, 3000);
}
autoplay_imageVerify();

function refresh(){
        $('#md').modal('hide');
        location.reload();
}
</script>
```

La siguiente tabla muestra la estructura del código para la vista modal del cambio de contraseña:

Tabla 14. Estructura del modalCambioClave .php, fuente autor

modalCambioClave.php

```
//Se inicia el código con los div de contiene la estructura modal para el cambio de contraseña

<div class="modal fade" id="modalCambioClave" tabindex="-1" role="dialog" aria-labelledby="Cambiar Contraseña" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
//Se muestra el titulo de la vista
        <h5   class="modal-title"   id="TituloModalCambioClave">Cambiar contraseña ciudadano</h5>
        <br>

//Se crean y estilizan los botones de la vista
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <div   id="contenedorFormularioCambioClave"   class="text-center shadow p-3 mb-5 bg-white rounded">
//Se crean los campos para el ingreso de contrseña
                      <form class="form-signin">
                          <label            for="inputClave">Contraseña Antigua</label>
                          <input  type="password"  class="form-control" id="inputClave-antigua"    placeholder="********"    pattern="[A-Za-z0-9]{4,15}">
                          <label                 for="inputClave">Nueva contraseña</label>
                          <input  type="password"  class="form-control" minlength="6" id="inputClave-nueva" placeholder="********" pattern="[A-Za-z0-9]{4,15}">

                          <label     for="inputClave">Repetir     nueva contraseña</label>
                          <input  type="password"  class="form-control" minlength="6"       id="inputClave-nueva-r"       placeholder="********" pattern="[A-Za-z0-9]{4,15}">

                          <label id="consola"></label><br>

                              <button   type="submit"   id="btn-cambiar-clave"   class="btn   btn-danger">ACTUALIZAR<i   class="fas   fa-fw   fa-sync"></i></button>
                              <br>
                              <label id="label-correo"><?php
```

```
//Se validala sesión del usuario por medio del correo
echo($_SESSION['usuario']['Email']); ?></label>
                                <br>
                        </form>
                </div>
      </div>
    </div>
  </div>
</div>


<style type="text/css">
        .modal-header{
                border: none;
        }
</style>
```

La siguiente tabla describe la estructura del código para la actualización de datos del ciudadano

Tabla 15. Estructura del modalActualizarDatos .php, fuente autor

modalActualizarDatos.php

```
<?php
//Se crea la instancia de sesión con los datos del usuario
$nombre = $_SESSION['usuario']['Nombre'];
$direccion = $_SESSION['usuario']['Direccion'];
$cedula = $_SESSION['usuario']['Cedula'];
$correo = $_SESSION['usuario']['Email'];
$telefono = $_SESSION['usuario']['Telefono'];
$expedicion = $_SESSION['usuario']['LugarExpedicion'];
$id = $_SESSION['usuario']['Id'];
?>
//Se crean los div para la vista modal donde se actualizaran los datos
<div   class="modal   fade"   id="modalActualizarDatos"   tabindex="-1"
role="dialog" aria-labelledby="Actualizar datos" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5  class="modal-title"  id="TituloModalCambioClave">Actualizar
datos ciudadano</h5>
        <br>
        <button  type="button"  class="close"  data-dismiss="modal"  aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <h5  id="idUsuario"  style="display:  none;">  <?php  echo($id);  ?>
</h5>

//Se crea el contenedror que contiene el formulario para la clave del
usuario
        <div   id="contenedorFormularioCambioClave"   class="text-center
shadow p-3 mb-5 bg-white rounded">
                      <form class="form-signin">
                          <div class="form-row">
                              <div class="col">
                                              <label
for="inputNombre"><b>Nombre completo</b></label><br>
                                              <input         type="text"
class="form-control"      onkeypress="return      validarLetras(event)"
id="inputNombre"    placeholder="Pablo    Perez    Paso"    value="<?php
echo($nombre); ?>">
                                      </div>
                                      <div class="col">
                                      <label
for="inputDireccion"><b>Direccion</b></label>
                                      <input  type="text"  class="form-
control" id="inputDireccion" placeholder="Cra 1 No 4 CASA 2" pattern="[A-
Za-z0-9]{4,45}" value="<?php echo($direccion); ?>" >
                                  </div>
```

```
                            </div>
                            <div class="form-row">
                                <div class="col">
//Recibe los datos de la cedula del ciudadano
                        <label
for="inputCedula"><b>Cedula</b></label><br>
                                                    <input         type="text"
class="form-control"     onkeypress="return     validaNumericos(event)"
id="inputCedula" placeholder="1.222.333.444" value="<?php echo($cedula);
?>" >
                                            </div>
                                            <div class="col">
//Recibe los datos del lugar de  expedición
                <label      for="inputExpedicionCedula"><b>Lugar      de
Expedicion</b></label><br>
                                                    <input         type="text"
class="form-control"      onkeypress="return      validarLetras(event)"
id="inputExpedicionCedula"  placeholder="CARTAGENA-BOLIVAR"  value="<?php
echo($expedicion); ?>" >
                                            </div>
                            </div>
                            <div class="form-row">
                                            <div class="col">
//Recibe los datos de la dirección
        <label for="inputDireccion"><b>Correo electrónico</b></label>
                                             <input type="email" class="form-
control"         data-validation="email"                  id="inputEmail"
placeholder="usuario@usuario.com"        pattern="[A-Za-z0-9@._-]{4,45}"
value="<?php echo($correo); ?>" >
                                </div>
                                <div class="col">
                                            <label
for="inputDireccion"><b>Telefono</b></label>
                                            <input  type="text"  class="form-
control"  onkeypress="return  validaNumericos(event)"  id="inputTelefono"
placeholder="300 000 0000" value="<?php echo($telefono); ?>">
                                </div>
                            </div>
//Se crea el botón de actualizar                                <br>
                            <button        type="submit"        id="btn-
actualizarDatos" class="btn btn-danger">ACTUALIZAR<i class="fas fa-fw fa-
sync"></i></button>
                                <br>
                                <label id="consolaActualizar">...</label>

                        </form>
                </div>
      </div>
    </div>
  </div>
</div>

<div   class="modal"   tabindex="-1"   role="dialog"   id="modal-success-
actualizar-datos">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
```

```
        <h5 class="modal-title">Datos actualizados con exito!</h5>
        <a href="../login/cerrarSesion.php">
                <button type="button" class="close"  aria-label="Close">
                  <span aria-hidden="true">&times;</span>
                </button>
            </a>
      </div>
//Se muestra la información de la actualización de los datos
      <div class="modal-body">
        <p>Se actualizó su informacion personal, vuelva a iniciar sesion
para cargar los cambios</p>
      </div>
      <div class="modal-footer">
        <a href="../login/cerrarSesion.php">
                <button class="btn btn-success">OK</button>
        </a>
      </div>
    </div>
  </div>
</div>
```

La siguiente tabla describe la estructura del código para la lectura de las imágenes que debe adjuntar el ciudadano a la hora de generar la denuncia:

Tabla 16. Estructura del Leerimg .php, fuente autor

leerImg.php

```
<?php
$imageData ="";
$imageData1 ="";
$imageData2 ="";
if(isset($_FILES['file-0'])){
        $imageData = $_FILES['file-0']['name'];
}
if(isset($_FILES['file-1'])){
        $imageData1 = $_FILES['file-1']['name'];
}
if(isset($_FILES['file-2'])){
        $imageData2 = $_FILES['file-2']['name'];
}

echo("{\"numeroArchivos\":".sizeof($_FILES).",\"nombreImagen1\":\"".$im
ageData."\",\"nombreImagen2\":\"".$imageData1."\",\"nombreImagen3\":\""
.$imageData2."\"}");

/*
* Este script se encarga de leer imagenes agregadas al formulario con el
fin de
* mostrar los datos que corresponden al nombre de del archivo agregado
al formulario
*/
$mTipo1 = exif_imagetype($imageData);
$mTipo2 = exif_imagetype($imageData1);
$mTipo3 = exif_imagetype($imageData2);

// Se vefirica que el archivo esté agregado al formulario

if (($mTipo1 != IMAGETYPE_JPEG) &amp;amp;&amp;amp; ($mTipo1 != IMAGETYPE_PNG)){
        echo("IMG1FALSE");
// Se asignan los datos a la variable
}else   if   (($mTipo2   !=   IMAGETYPE_JPEG)   &amp;amp;&amp;amp;   ($mTipo2   !=
IMAGETYPE_PNG)){
        echo("IMG2FALSE");
}else   if   (($mTipo3   !=   IMAGETYPE_JPEG)   &amp;amp;&amp;amp;   ($mTipo3   !=
IMAGETYPE_PNG)){
        echo("IMG3FALSE");
}else {
        echo("{\"numeroArchivos\":".sizeof($_FILES).",\"nombreImagen1\":
\"".$imageData."\",\"nombreImagen2\":\"".$imageData1."\",\"nombreImagen
3\":\"".$imageData2."\"}");
}

//      echo    sprintf('<img    src="%s"    />',
base64_encode($imageData));
// }else{
//      echo "Seleccione una imagen";
```

```
// }
// var_dump($_FILES);*/

?>
```

La siguiente tabla describe la estructura del código del índex o página principal del módulo:

Tabla 17 Estructura de Index .php, fuente autor

index.php

```
<?php
//Se inicia la sesión y se importa la case complaint_class.php
session_start();
if(isset($_SESSION['usuario'])){
  require("../unilab_clases/complaint_class.php");
  $myComplaints = new Complaint ();
  $datos_usuario = $_SESSION['usuario'];
?>
<!DOCTYPE html>
<html lang="es">

<head>

  <meta http-equiv="Content-Type" content="text/html; charset=utf8"/>
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta  name="viewport"  content="width=device-width,  initial-scale=1,
shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

//Se crea el título del proyecto
  <title>GESTOR URBANO - CIUDADANO</title>

  <!-- Custom fonts for this template-->
  <link  href="vendor/fontawesome-free/css/all.min.css"  rel="stylesheet"
type="text/css">
  <link
href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,
400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!—Se referencias los sitlos personalizados de la plantilla -->
  <link href="css/sb-admin-2.min.css" rel="stylesheet">

  <script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBJZZ0bfYm8I3Y9tU
W4mCp-q9nXIEOFvnY&callback=initMap">
  </script>

  <!—Se carga el SDK de facebook para el chat bot  -->
      <div id="fb-root"></div>
      <script>
        window.fbAsyncInit = function() {
          FB.init({
            xfbml            : true,
            version          : 'v4.0'
          });
        };
```

```
        (function(d, s, id) {
        var js, fjs = d.getElementsByTagName(s)[0];
        if (d.getElementById(id)) return;
        js = d.createElement(s); js.id = id;
        js.src                                                    =
'https://connect.facebook.net/es_LA/sdk/xfbml.customerchat.js';
        fjs.parentNode.insertBefore(js, fjs);
      }(document, 'script', 'facebook-jssdk'));</script>

      <!—Aquí se presenta el código del chat bot personalizado  -->
      <div class="fb-customerchat"
        attribution=setup_tool
        page_id="110135763721247"
  theme_color="#fa3c4c"
  logged_in_greeting="Hola!  Somos  Gestores  Urbanos.  ¿En  que  puedo
servirte?"
  logged_out_greeting="Hola!  Somos  Gestores  Urbanos.  ¿En  que  puedo
servirte?">
      </div>

</head>

<body id="page-top">

  <!-- Page Wrapper -->
  <div id="wrapper">

    <!—Sidebar de la pagina principal -->
    <ul   class="navbar-nav   bg-gradient-danger   sidebar   sidebar-dark
accordion" id="accordionSidebar">

      <!-- Sidebar - Brand -->
      <a class="sidebar-brand d-flex align-items-center justify-content-
center" href="index.html">
        <div class="sidebar-brand-text mx-3">CUENTA<br>
        <label          id="lblId"          style="color:white;"><?php
echo($datos_usuario['Id']);?></label>

        </div>
      </a>

//Desde esta línea se inician los enlaces del sidebar para los sub menus
      <hr class="sidebar-divider">

      <!-- Heading -->
      <div class="sidebar-heading">
        Denuncias
      </div>
      <!-- Nav Item - Tables -->
      <li class="nav-item">
        <a class="nav-link" href="#" onclick="getScreen('home')">
          <i class="fas fa-fw fa-home"></i>
          <span>Home</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#" onclick="getScreen('mapa')">
          <i class="fas fa-fw fa-map-marked-alt"></i>
```

```
          <span>Mapa de casos</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#" onclick="getScreen('tabla')">
          <i class="fas fa-fw fa-file"></i>
          <span>Mis denuncias</span></a>
      </li>
      <li class="nav-item">
        <a                    class="nav-link"                    href="#"
onclick="getScreen('todas_las_denuncias')">
          <i class="fas fa-fw fa-file"></i>
          <span>Todas las denuncias</span></a>
      </li>
      <li class="nav-item">
        <a    class="nav-link"    href="#"    data-toggle="modal"    data-
target="#cuestionario">
          <i class="fas fa-fw fa-plus-circle"></i>
          <span>Nueva denuncia</span></a>
      </li>
      <!-- Divider -->
      <hr class="sidebar-divider">

      <!-- Heading -->
      <div class="sidebar-heading">
        Mis datos
      </div>

      <!-- Nav Item - Tables -->
      <li class="nav-item">
        <a    class="nav-link"    href="#"    data-toggle="modal"    data-
target="#modalActualizarDatos">
          <i class="fas fa-fw fa-sync"></i>
          <span>Actualizar mis datos</span></a>
      </li>
      <li class="nav-item">
        <a   class="nav-link"   href="#"     data-toggle="modal"   data-
target="#modalCambioClave">
          <i class="fas fa-fw fa-lock"></i>
          <span>Cambiar contraseña</span></a>
      </li>

      <!-- Divider -->
      <hr class="sidebar-divider d-none d-md-block">

      <div class="sidebar-heading">
        Acerca de
      </div>
      <li class="nav-item">
        <a    class="nav-link"    href="#"    data-toggle="modal"    data-
target="#modal-quienes-somos">
          <i class="fas fa-fw fa-lightbulb"></i>
          <span>¿Quienes somos?</span></a>
      </li>
      <li class="nav-item">
        <a    class="nav-link"    href="#"    data-toggle="modal"    data-
target="#modal-redes">
          <i class="fas fa-fw fa-share-alt"></i>
```

```
          <span>Siguenos en redes sociales</span></a>
      </li>
      <li class="nav-item">
        <a   class="nav-link"   href="#"   data-toggle="modal"   data-
target="#modal-aspectos-legales">
          <i class="fas fa-fw fa-balance-scale"></i>
          <span>Aspectos legales</span></a>
      </li>

      <!-- <li class="nav-item">
        <a class="nav-link" href="../login/cerrarSesion.php">
          <i class="fas fa-fw fa-male"></i>
          <span>Sistema de participación ciudadana</span></a>
      </li> -->
      <li class="nav-item">
        <a   class="nav-link"     href="#"   data-toggle="modal"   data-
target="#modal-preguntas-frecuentes">
          <i class="fas fa-fw fa-comments"></i>
          <span>Preguntas frecuentes</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="../login/cerrarSesion.php">
          <i class="fas fa-fw fa-sign-out-alt"></i>
          <span>Cerrar sesion</span></a>
      </li>
      <hr class="sidebar-divider d-none d-md-block">
      <!-- Sidebar Toggler (Sidebar) -->
      <div class="text-center d-none d-md-inline">
        <button             class="rounded-circle             border-0"
id="sidebarToggle"></button>
      </div>

    </ul>
    <!—Aquí termina el sidebar  -->

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!—Contenido principal  -->
      <div id="content">

        <!-- barra superior -->
        <nav class="navbar navbar-expand navbar-light bg-white topbar mb-
4 static-top shadow">

          <!-- Sidebar Toggle (Topbar) -->
          <button id="sidebarToggleTop" class="btn btn-danger d-md-none
rounded-circle mr-3">
            <i class="fa fa-bars"></i>
          </button>

          <!—campo de búsqueda  -->
          <form class="d-none d-sm-inline-block form-inline mr-auto ml-
md-3 my-2 my-md-0 mw-100 navbar-search">
            <div class="input-group">
```

```
              <input type="text" class="form-control bg-light border-0
small"  placeholder="Buscar  denuncia..."  aria-label="Search"  aria-
describedby="basic-addon2">
              <div class="input-group-append">
                <button class="btn btn-danger" type="button">
                  <i class="fas fa-search fa-sm"></i>
                </button>
              </div>
            </div>
          </form>

          <!—Barra de herramientas de navegación  -->
          <ul class="navbar-nav ml-auto">

            <!-- Nav Item - Search Dropdown (Visible Only XS) -->
            <li class="nav-item dropdown no-arrow d-sm-none">
              <a     class="nav-link     dropdown-toggle"     href="#"
id="searchDropdown"    role="button"    data-toggle="dropdown"    aria-
haspopup="true" aria-expanded="false">
                <i class="fas fa-search fa-fw"></i>
              </a>
              <!-- Dropdown - Messages -->
              <div  class="dropdown-menu dropdown-menu-right p-3 shadow
animated--grow-in" aria-labelledby="searchDropdown">
                <form class="form-inline mr-auto w-100 navbar-search">
                  <div class="input-group">
                    <input  type="text"  class="form-control  bg-light
border-0  small"  placeholder="Buscar  denuncia..."  aria-label="Search"
aria-describedby="basic-addon2">
                    <div class="input-group-append">
                      <button class="btn btn-primary" type="button">
                        <i class="fas fa-search fa-sm"></i>
                      </button>
                    </div>
                  </div>
                </form>
              </div>
            </li>

            <div class="topbar-divider d-none d-sm-block"></div>

            <!-- Nav Item - User Informacion del usuario -->
            <li class="nav-item dropdown no-arrow">
              <a     class="nav-link     dropdown-toggle"     href="#"
id="userDropdown"     role="button"     data-toggle="dropdown"     aria-
haspopup="true" aria-expanded="false">
                <span  class="mr-2  d-none  d-lg-inline  text-gray-600
small"><?php echo($datos_usuario['Nombre']);?></span>
                <img         class="img-profile         rounded-circle"
src="https://gestorurbano.com/imagenes/favicon.ico">
              </a>
              <!—Información del usuario  -->
              <div   class="dropdown-menu   dropdown-menu-right   shadow
animated--grow-in" aria-labelledby="userDropdown">
                <!-- <a class="dropdown-item" href="#">
                  <i  class="fas  fa-user  fa-sm  fa-fw  mr-2  text-gray-
400"></i>
```

```
                  Profile
                </a> -->
                <!-- <a class="dropdown-item" href="#">
                  <i  class="fas  fa-cogs  fa-sm  fa-fw  mr-2  text-gray-
400"></i>
                  Settings
                </a> -->
                <!-- <a class="dropdown-item" href="#">
                  <i  class="fas  fa-list  fa-sm  fa-fw  mr-2  text-gray-
400"></i>
                  Activity Log
                </a> -->
                <div class="dropdown-divider"></div>
                <a  class="dropdown-item"  href="#"  data-toggle="modal"
data-target="#logoutModal">
                  Cerrar sesion
                </a>
              </div>
            </li>

          </ul>

        </nav>
        <!-- fin bara superior -->

        <!—Aquí inicia el contenido de la pagina  -->
        <div class="container-fluid">

          <!—Cabecera de la pagina  -->
          <div   class="d-sm-flex   align-items-center   justify-content-
between mb-4">
            <h1 class="h3 mb-0 text-gray-800">Gestor urbano</h1>
          </div>

          <!-- Content Row -->
            <div class="card shadow mb-4">
            <div class="card-header py-3">
              <h6 class="m-0 font-weight-bold text-primary" id="titulo-
screen">Home</h6>
            </div>
            <div class="card-body">
               <?php
                  include("./screen_home.php");
                  include("./screen_denucnias.php");
                  include("./tabla_denuncia.php");

               ?>



            </div>
          </div>


           <!-- Content Row -->
          <div class="row">
```

```
          </div>

        </div>
        <!-- /.container-fluid -->

      </div>
      <!-- End of Main Content -->

      <!-- Footer -->
      <footer class="sticky-footer bg-white">
        <div class="container my-auto">
          <div class="copyright text-center my-auto">
            <span>Copyright &copy; gestor urbano 2019</span>
          </div>
        </div>
      </footer>
      <!—Fin del footer  -->

    </div>
    <!-- End of Content Wrapper -->

  </div>
  <!-- End of Page Wrapper -->

  <!-- Scroll to Top Button-->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <!—Modal para cerrar sesión -->
  <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog"
aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Seguro desea
salir?</h5>
          <button class="close" type="button" data-dismiss="modal" aria-
label="Close">
            <span aria-hidden="true">×</span>
          </button>
        </div>
        <div class="modal-body">Al aceptar su sesion finalizará</div>
        <div class="modal-footer">
          <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
          <a class="btn btn-primary"
href="../login/cerrarSesion.php">Aceptar</a>
        </div>
      </div>
    </div>
  </div>

  <!-- Bootstrap core JavaScript-->
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
```

```
  <!-- Core plugin JavaScript-->
  <script src="vendor/jquery-easing/jquery.easing.min.js"></script>

  <!-- Custom scripts for all pages-->
  <script src="js/sb-admin-2.min.js"></script>
  <script type="text/javascript" src="./js/cambiarClave.js"></script>
  <script type="text/javascript" src="./js/actualizarDatos.js"></script>
  <script type="text/javascript" src="./js/admin-screen.js"></script>
  <script type="text/javascript" src="./js/notas-denuncias.js"></script>
  <script          type="text/javascript"          src="./js/validacion-
campos.js"></script>
  <script type="text/javascript" src="./js/denuncia-img.js"></script>
  <script           type="text/javascript"           src="./js/ver-notas-
denuncia.js"></script>

</body>

</html>
<?php
  require("./modalCuestionario.php");
  require("./modalCambioClave.php");
  require("./modalActualizarDatos.php");
  require("./modalNotas.php");
  require("../modals-acerca-de.php");
}else{
  header("location:../login.php");
}

?>
```

La siguiente tabla muestra la estructura del archivo gestorUbicaciones el cual se encarga de devolver los datos de la geolocalización de todas las denuncias registradas:

Tabla 18. Estructura del gestorUbicaciones .php, fuente autor

gestorUbicaciones.php

```
<?php

require("../unilab_clases/complaint_class.php");  //  Se  requiere  la  clase
complaint_class


//Instancia de la clase
$a = new Complaint();


// Se usa la funcion getLocationDataOfAll() que devuelve los datos de geolocalizacion de
todas las denuncias registradas en la base de datos, los datos son almacenados en la
variable $datos

$datos = $a->getLocationDataOfAll();
echo(json_encode($datos,true));


?>
```

La siguiente tabla muestra la estructura del archivo gestorDenuncia.php donde se describe la estructura del código para el guardado de las imágenes adjuntas desde el módulo ciudadano:

Tabla 19. GestorDenuncia.php

gestorDenuncia.php

```
<?php
//Se importa la clase complaint_class.php
require("../unilab_clases/complaint_class.php");
if(isset($_POST['idUsuario'])    &&    isset($_POST['cuestionario'])    &&
isset($_POST['ubicacion'])){

//Se establecen la cantidad de archivos a subir y la ruta de
almacenamiento
        if(sizeof($_FILES)==3     &&     isset($_FILES['file-0'])     &&
isset($_FILES['file-1']) && isset($_FILES['file-2'])){
                $dir_subida = '../denuncias/imagenes/';
                $fichero_subido1  =  $dir_subida.basename($_FILES['file-
0']['name']);
                $fichero_subido2  =  $dir_subida.basename($_FILES['file-
1']['name']);
                $fichero_subido3  =  $dir_subida.basename($_FILES['file-
2']['name']);
                $nombre_archivo1= basename($_FILES['file-0']['name']);
                $nombre_archivo2= basename($_FILES['file-1']['name']);
                $nombre_archivo3= basename($_FILES['file-2']['name']);
//A través de este json se mueven los archivos a la base de datos
                $json_imagenes                                          =
"{\"img1\":\"$nombre_archivo1\",\"img2\":\"$nombre_archivo2\",\"img3\":
\"$nombre_archivo3\"}";
                if    (move_uploaded_file($_FILES['file-0']['tmp_name'],
$fichero_subido1)  &&  move_uploaded_file($_FILES['file-1']['tmp_name'],
$fichero_subido2)  &&  move_uploaded_file($_FILES['file-2']['tmp_name'],
$fichero_subido3)) {
                        $idUsuario = $_POST['idUsuario'];
                        $cuestionario = $_POST['cuestionario'];
                        $ubicacion = $_POST['ubicacion'];

                        $complaint = new Complaint ();
                        $complaint                                      ->
create($idUsuario,$ubicacion,$cuestionario,$json_imagenes);
                        echo true;
                }else{
                         echo "¡Posible ataque de subida de ficheros!\n";
                }//Se muestra mensaje al usuario para que cargue las tres
imagens
        }else{
                        echo "Agregue 3 imagenes";
                        echo"<br>";
                        echo sizeof($_FILES);
        }
```

```
}else{
        echo("error");
}?>
```

En la siguiente tabla se presenta la estructura del código en el archivo gestor-notas.php el cual tiene la función de gestionar las notas agregadas a las denuncias:

Tabla 20. Estructura del Gestor-notas .php, fuente autor

Gestor-notas.php

```
<?php
require("../unilab_clases/complaint_class.php");
$denuncia = new Complaint();
/*
*Este scrip se encarga de gestoionar las notas agregadas en las denuncias.
*Dependiendo los datos POST recibidos, el Script desarrolla una accion u
otra.
*
*/
require("../unilab_clases/complaint_class.php"); //Se requiere la clase
compaint_class encargada de manejar las denuncias.
$denuncia = new Complaint(); //Instancia de la clase
if(isset($_POST['id']) && isset($_POST['nota'])){
        $nota = $_POST['nota'];
        $id = $_POST['id'];
if(isset($_POST['id'])  &&  isset($_POST['nota'])){  //Verifica  que  se
reciban los datos necesarios para continuar la ejecucion. este proceso
agrega notas a las denuncias.
        $nota = $_POST['nota']; //Asignacion de datos a variables.
        $id = $_POST['id']; //Asignacion de datos a variables.
        if($denuncia->noteIsEmpy($id)){
                echo($denuncia        ->        addNote($id,$nota,true));
//addNote(nota,first)
        if($denuncia->noteIsEmpy($id)){ //Verifica de el campo notas de
la denuncia se encuentra o no vacio
                echo($denuncia -> addNote($id,$nota,true)); //Agrega una
nota a la denucia y establece el valor first en true para indicar que es
la primera nota agregada a la denucnia.
        }else{
                echo($denuncia -> addnote($id,$nota,false));
                echo($denuncia -> addnote($id,$nota,false)); //Agrega una
nota a la denucia y establece el valor first en false para indicar que
no es la primera nota agregada a la denucnia.
        }
        // echo($id.",".$nota);
}
if(isset($_POST['id']) && isset($_POST['Get'])){
if(isset($_POST['id'])  &&  isset($_POST['Get'])){  //  Verifica  que  se
reciban los datos necesarios para continuar la ejecucion. este proceso
retorna datos de tabla (<tr><td></td></tr>) con las notas agrgadas a una
denucnia.
        $id = $_POST['id'];
        $notas = $denuncia->getNotes($id);
        $notasJson = "[".$notas."]";
        $arrayNotas = json_decode($notasJson,true);
        $id = $_POST['id']; //Asignacion de variables
        $notas  =  $denuncia->getNotes($id);  //Uso  del  metodo  getNotes()
para recuperar las notas agregadas a la denuncia
```

```
        $notasJson = "[".$notas."]"; //Agrega caracteres necesarios para
cumplir con el estandar JSON
        $arrayNotas = json_decode($notasJson,true); // Se convierte la
variable a Array

        $tablaNotas = "";
        $tablaNotas = ""; //Variable vacia

        for($i=0;$i<sizeof($arrayNotas);$i++){
        for($i=0;$i<sizeof($arrayNotas);$i++){ //Se generan los datos de
tabla recorriendo el array de notas
                $tablaNotas                                                =
$tablaNotas."<tr><td>".$arrayNotas[$i]['nota']."</td></tr>";
        }
      echo($tablaNotas);
```

Descripción de los archivos en la carpeta admin_account

A continuación se describen los archivos que contiene la carpeta, bajo la ruta: public_html/admin-account/

La siguiente tabla muestra la información correspondiente a la estructura del archivo: ***agregarPreguntasC.php***, esta tiene la función de gestionar las preguntas del cuestionario y guardarlas en la base de datos.

Tabla 21. Estructura de agragarPreguntas .php, fuente autor

agregarPreguntasC.php

```
<?php
//Se establece la conexión con la base de datos
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
$pregunta= $_POST['pregunta'];
$tipo= $_POST['tipo'];

//Se valida que los campos para seleccionar los archivos no estén vacíos
y muestra alerta

if($pregunta=="" && $tipo=="Seleccione..."){
    echo "<div class='alert alert-danger' role='alert'>Debe ingresar
valores en los campo</div>";
} else if($pregunta=="" || $tipo=="Seleccione..."){
    echo "<div class='alert alert-danger' role='alert'>Es obligacion
rellenar todo </div>";
}else{

//Se guardan las preguntas en lavase de datos

    $conexion= new mysqli($server,$username,$password,$db);
    $consulta = "INSERT INTO preguntas_cuetionario (pregunta,tipo) VALUES
('$pregunta','$tipo')";
    $query = $conexion->query($consulta);
    if(!$query){
//Se muestra mensaje al usuario de acuerdo  a la validación
    echo "<div class='alert alert-danger' role='alert'>No se puedo
agregar la pregunta a la base de datos</div>";
    }else{
    echo "<div class='alert alert-success' role='alert'>Operacion
exitoza</div>";
    }
}

?>
```

La siguiente tabla muestra la estructura del archivo ***listarPreguntaId.php*** el cual cumple la función de listar las preguntas en el modal de edición de las mismas:

Tabla 22. ListarPreguntaId .php, fuente autor

listarPreguntaId.php

```
<?php
// Se inicia la conexión a la base de datos
$id_pregunta=$_POST['idp'];
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
$conexion= new mysqli($server,$username,$password,$db);
$consulta = "SELECT * FROM preguntas_cuetionario WHERE id= $id_pregunta
";
$query = $conexion->query($consulta);
$pregunta=mysqli_fetch_array($query);
$p=$pregunta['pregunta'];
$tipop=$pregunta['tipo'];
$respuesta="";

        $respuesta="<input     id     ='etiID'     value='$id_pregunta'
style='display:          none;'/><label          for='select-docentes'
style='color:black'>Pregunta</label>
                    <input    class='form-control'  id='txtPreguntaEdit'
value='$p'><br>
                    <label  style='color:black'>Tipo</label>
                    <select    class='browser-default    custom-select'
id='selectTipoEdit' >";

        if( $tipop == 'cerrada'){
                $respuesta=$respuesta."<option
value='cerrada'>CERRADA</option>
                                        <option
value='abierta'>ABIERTA</option></select>";
         }else{
                $respuesta=$respuesta."<option
value='abierta'>ABIERTA</option>
                                        <option
value='cerrada'>CERRADA</option></select>";
         }

         echo $respuesta;

?>
```

La siguiente tabla muestra la estructura del archivo ***listarDocenteCod.php*** el cual tiene como función mostrar los datos en pantalla de la lista de docente existentes en la base de datos, en el módulo administrador:

Tabla 23. ListarDocenteCod .php, fuente autor

listarDocenteCod.php

```
<?php

//Se inicia la conexión  a la base de datos
$codigoDocente=$_POST['codD'];
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
//Se realiza la consulta en la base de datos para pasar los datos del
docente
$conexion= new mysqli($server,$username,$password,$db);
$consulta   =   "SELECT   *   FROM   docentes   WHERE   codigoDocente=
'$codigoDocente'";
$query = $conexion->query($consulta);
$docente=mysqli_fetch_array($query);
$nombreD=$docente['nombre'];
$direccionD=$docente['direccion'];
$cedulaD=$docente['cedula'];
$lugarExpedicionD=$docente['lugarExpedicion'];
$emailD=$docente['email'];
$telefonoD=$docente['telefono'];

//Se imprimen los datos del docente en pantalla

      echo '
        <form class="form-signin">
        <input  id  ="etiCod"  value="'.$codigoDocente.'"  style="display:
none;"/>
        <div class="form-row">
            <div class="col">
                <label                    for="inputNombreMD"><b>Nombre
completo</b></label><br>
                <input  type="text"  value="'.$nombreD.'"  class="form-
control" onkeypress="return validarLetras(event)" id="inputNombreMD" >
            </div>
            <div class="col">
                <label for="inputDireccionMD"><b>Direccion</b></label>
                <input type="text" value="'.$direccionD.'" class="form-
control" id="inputDireccionMD" pattern="[A-Za-z0-9]{4,45}">
            </div>
        </div>
        <div class="form-row">
            <div class="col">
                <label for="inputCedulaMD"><b>Cedula</b></label><br>
                <input  type="text"  value="'.$cedulaD.'"  class="form-
control" onkeypress="return validaNumericos(event)" id="inputCedulaMD"  >
            </div>
```

```
            <div class="col">
                <label     for="inputExpedicionCedulaMD"><b>Lugar     de
Expedicion</b></label><br>
                <input     type="text"     value="'.$lugarExpedicionD.'"
class="form-control"      onkeypress="return      validarLetras(event)"
id="inputExpedicionCedulaMD" >
            </div>
        </div>
        <div class="form-row">
            <div class="col">
                <label><b>Correo electrónico</b></label>
                 <input  type="email"  value="'.$emailD.'"  class="form-
control"  data-validation="email"  id="inputEmailMD"   pattern="[A-Za-z0-
9@._-]{4,45}">
            </div>
            <div class="col">
                <label><b>Telefono</b></label>
                <input  type="text"  value="'.$telefonoD.'"  class="form-
control" onkeypress="return validaNumericos(event)" id="inputTelefonoMD"
>
            </div>
        </div>
    </form>';
```

La siguiente tabla muestra la estructura del archivo imagenes-denuncia.php

El cual tiene como función:

Tabla 24. Imágenes-denuncia .php, fuente autor

Imagenes-denuncia.php

```
<?php
//Ser realiza la conexión a la base de datos
$idD=$_POST['idDenuncia'];
$rID= substr($idD, 1);
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
//Se realiza la consulta sobre la base de datos y se virifica si la
denuncia contiene adjuntos
$conexion= new mysqli($server,$username,$password,$db);
$consulta= "SELECT * FROM denuncia WHERE  idDenuncia = $rID";
$query = mysqli_query($conexion, $consulta);
while ($imgD=mysqli_fetch_array($query)){
    $json = $imgD['imagenes'];
    if($json==null){
        echo "No contiene adjuntos";
    }else{

//Se crea container el cual mostrará las tres imágenes
        $obj = json_decode($json);
        echo  "<div class='container'>
                <div class='row'>
                    <div class='col'>
                    N#1
                    <img  src='../denuncias/imagenes/".$obj->{'img1'}."'
width='100%' >
                    </div>
                    <div class='col'>
                    N#2
                    <img  src='../denuncias/imagenes/".$obj->{'img2'}."'
width='100%'>
                    </div>
                </div>
                <br />
                <div class='row'>
                    <div class='col'>
                    N#3
                    <img  src='../denuncias/imagenes/".$obj->{'img3'}."'
width='100%'>
                    </div>
                </div>
            </div>";    }


 }
?>
```

La siguiente tabla muestra la estructura del código para el archivo *eliminarPreguntaC.php*, este archivo se encarga de gestionar la eliminación de las preguntas del cuestionario con inferencia en la base de datos:

Tabla 25. Eliminar Pregunta .php, fuente autor

eliminarPreguntaC.php

```
<?php
//Se crea la conexión con la base de datos
$id_pregunta =$_POST['idp'];
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
//Se realiza el CRUD en la base de datos con la pregunta a eliminar
$conexion= new mysqli($server,$username,$password,$db);
$consulta = "DELETE FROM preguntas_cuetionario  WHERE id=$id_pregunta";
$query = $conexion->query($consulta);
if(!$query){
    echo false;
}else{
    echo true;
}
?>
```

La siguiente tabla muestra la estructura del código para el archivo *eliminarDocente.php,* este archivo se encarga de gestionar la eliminación de los docentes:

Tabla 26. Estructura eliminarDocente.php, fuente autor

eliminarDocente.php

```
<?php
//Se crea la conexión a la base de datos
$cod=$_POST['codDE'];
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
//Se realiza el CRUD en la base de datos mediante el código del docente
$conexion= new mysqli($server,$username,$password,$db);
$consulta= "DELETE FROM docentes  WHERE codigoDocente = '$cod'";
$query = mysqli_query($conexion, $consulta);

if(!$query){
    echo false;
}else{
    echo true;
}
?>
```

La siguiente tabla muestra la estructura del código para el archivo eliminarDenuncia.php, este archivo se encarga de gestionar la eliminación de las denuncias:

Tabla 27. Estructura eliminarDenuncia .php, fuente autor

eliminarDenuncia.php

```
<?php
//Se inicia la conexión a la base de datos
$idD=$_POST['idDe'];
$rID= substr($idD, 1);
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
// Se realiza el CRUD mediante el id de la denuncia
$conexion= new mysqli($server,$username,$password,$db);
$consulta= "DELETE FROM denuncia  WHERE idDenuncia = $rID";
$query = mysqli_query($conexion, $consulta);

if(!$query){
    echo false;
}else{
    echo true;
}
?>
```

La siguiente tabla muestra la estructura del código para el archivo editarPreguntaC.php, este archivo se encarga de gestionar la edición de las preguntas del cuestionario:

Tabla 28. Estructura editarPreguntaC .php, fuente autor

editarPreguntaC.php

```
<?php
$id_pregunta =$_POST['idp'];
$p=$_POST['pregunta'];
$t=$_POST['tipo'];
$server="54.39.133.88";
$username='gestorur_unilab';
$password='unilab2019.';
$db='gestorur_unilab_db_2';
$conexion= new mysqli($server,$username,$password,$db);
$consulta = "UPDATE preguntas_cuetionario SET pregunta='$p',tipo='$t'
WHERE id= $id_pregunta ";
$query = $conexion->query($consulta);
if($p==""||$t==""){
    echo "vacio";
}else{
    if(!$query){
        echo false;
    }else{
        echo true;
    }

}
?>
```

Descripción de los archivos en la carpeta teacher-account

A continuación, se describen los archivos que contiene la carpeta, bajo la ruta: public_html/ teacher-account /

La siguiente tabla muestra la estructura del código para el archivo teacher_data.php, este archivo se encarga de gestionar las funciones encargadas de devolver el código del docente, y los datos relacionados al mismo:

Tabla 29. Estructura teacher_data.php, fuente autor

teacher_data.php

```
<?php
/**
 *
 */
class teachers{
//Se crea la conexión a la base de datos

        private $conexion=false;
        function __construct($datos_conexion = false)
        {
                if(!$datos_conexion){
                        $server="54.39.133.88";
                        $username='gestorur_unilab';
                        $password='unilab2019.';
                        $db='gestorur_unilab_db_2';


                        $this->conexion=                                new
mysqli($server,$username,$password,$db);
                        if($this->conexion->connect_errno){
                                echo($this->conexion->connect_error);
                        }else{
                                $this->conexion->set_charset("utf8");
                        }
                }
        }
//Función que se encarga de registrar al usuario
        public                                                          function
registerUser($nombre,$direccion,$cedula,$expedicionCedula,$telefono,$us
uario,$email,$clave_hash){
                $id= $this->randomId(5);
                if($this->userRegisterVerify($email,$cedula) < 1){
                        $consulta     =     "INSERT     INTO     usuario
(nombre,direccion,cedula,lugarExpedicion,telefono,nombreUsuario,email,p
assword,idUsuario)
```

```
                        values
('$nombre','$direccion','$cedula','$expedicionCedula','$telefono','$usu
ario','$email','$clave_hash','$id')";
                        // $query=$this->conexion
                        $query = $this->conexion->query($consulta);
                        if(!$query){
                                return "E01";
                        }else{
                                return true;
                        }
                }else{
                        return "E00";
                }
        }

        public function getAllDocList(){ //funcion que devuelve una lista
para un select [OPTIONS] con los nombres de los docentes registrados.
                $consulta= "SELECT nombre from docentes";
                $query=$this->conexion->query($consulta);
                $lista = "<option>SELECCIONE</option>";

                while($datos = $query->fetch_assoc()){
                        $lista                                          =
$lista."<option>".$datos['nombre']."</option>";
                }
                return $lista;
        }

        public function getIdOfDoc($nombre){ //funcion que devuelve el
codigo del docente
                $consulta= "SELECT codigoDocente from docentes WHERE
nombre='".$nombre."'";
                $query=$this->conexion->query($consulta);

                $codigo = $query->fetch_assoc();

                return $codigo;
        }

        public function getNameOfIdDoc($id){ //funcion que devuelve el
codigo del docente
                $consulta= "SELECT nombre from docentes WHERE
codigoDocente='".$id."'";
                $query=$this->conexion->query($consulta);

                $res = $query->fetch_assoc();
                $nombre = $res['nombre'];
                return $nombre;
        }

        public function updateComplaintAsignedDoc($idComplaint,$idDoc){
                $query=$this->conexion->query("UPDATE denuncia SET
idDocente='".$idDoc."' WHERE
                        idDenuncia='".$idComplaint."'");
                $result = $this->conexion->affected_rows;
                if($result > 0){
                         return true;
```

```
                }else{
                        return false;
                }
                return $result;
        }
}

?>
```

La siguiente tabla muestra la estructura del código para el archivo *actualizar-estado-denuncia.php,* este archivo se encarga de gestionar los estados de las denuncias:

Tabla 30. Estructura Actualizae- estado-denuncia.php, fuente autor

Actualizar-estado-denuncia.php

```
<?php
//Se utiliza la estructura de control para colocar el nuevo estado de la
denuncia
if(isset($_POST['nuevo_estado']) && isset($_POST['id'])){

        $idDenuncia =$_POST['id'];
        $nuevo_estado= $_POST['nuevo_estado'];
//Se implementa la clase complaint_class.php

        require("../unilab_clases/complaint_class.php");

        $Complaint = new Complaint ();
//Se pasan los datos resultantes a la actualización de la denuncia
        $result = $Complaint->updateState($idDenuncia,$nuevo_estado);

        echo($result);
}

?>
```

La siguiente tabla muestra la estructura del código para el archivo *gestorDenuncia.php,* este archivo se encarga de almacenar los archivos de la denuncia (Imágenes adjuntas a la denuncia) y los almacena en la carpeta "Denuncias" en el servidor:

Tabla 31. Estructura gestorDenuncia.php, fuente autor

gestorDenuncia.php

```
<?php
//Se importa la clase complaint_class.php
require("../unilab_clases/complaint_class.php");
if(isset($_POST['idUsuario'])    &&    isset($_POST['cuestionario'])    &&
isset($_POST['ubicacion'])){
//Se realiza la validación de los 3 archivos
        if(sizeof($_FILES)==3     &&     isset($_FILES['file-0'])     &&
isset($_FILES['file-1']) && isset($_FILES['file-2'])){
                $dir_subida = '../denuncias/imagenes/';
                $fichero_subido1   =   $dir_subida.basename($_FILES['file-
0']['name']);
                $fichero_subido2   =   $dir_subida.basename($_FILES['file-
1']['name']);
                $fichero_subido3   =   $dir_subida.basename($_FILES['file-
2']['name']);
                $nombre_archivo1= basename($_FILES['file-0']['name']);
                $nombre_archivo2= basename($_FILES['file-1']['name']);
                $nombre_archivo3= basename($_FILES['file-2']['name']);
//El json toma las imagnes y las almacena
                $json_imagenes                                          =
"{\"img1\":\"$nombre_archivo1\",\"img2\":\"$nombre_archivo2\",\"img3\":
\"$nombre_archivo3\"}";

                if     (move_uploaded_file($_FILES['file-0']['tmp_name'],
$fichero_subido1)  &&  move_uploaded_file($_FILES['file-1']['tmp_name'],
$fichero_subido2)  &&  move_uploaded_file($_FILES['file-2']['tmp_name'],
$fichero_subido3)) {

                        $idUsuario = $_POST['idUsuario'];
                        $cuestionario = $_POST['cuestionario'];
                        $ubicacion = $_POST['ubicacion'];

                        $complaint = new Complaint ();
                        echo($complaint                                 ->
create($idUsuario,$ubicacion,$cuestionario,$json_imagenes));
                }else{
                         echo "¡Posible ataque de subida de ficheros!\n";
                }
        }else{
                        echo "Agregue 3 imagenes";
        }

}else{
        echo("error");
}
```

```
?>
```

7. GESTION DE LA BASE DE DATOS EN CPANNEL

En esta sección se describe la ruta donde se encuentra alojada la base de datos al igual que el usuario para la gestión de esta y el módulo de PHP, se describe el diccionario de datos y algunas recomendaciones a tener en cuenta a la hora de restaurar, importar o resguardar la base de datos.

En la ilustración 6 podemos apreciar los iconos con los que estaremos trabajando el primero refiere al modulo de PHP MyAdmin y el segundo a las características de la base de datos MySQL:

Ilustración 6.Seccion DATABASES del cPanel, fuente autor

Al seleccionar la opción de php MyAdmin nos aparecerá la ventana con nuestra base de datos a la izquierda y el menú de modulo en la parte central de la pantalla

Ilustración 7.Sección PHP MyAdmin, fuente autor

La base de datos la podemos manipular tal y como lo hacemos desde el servidor local, funciona exactamente igual, con la única diferencia que en este proyecto ya se encuentra alojada en un servidor remoto que se encarga de brindarnos el servicio gracias a nuestro proveedor de hosting

MySQL Data bases

En esta sección del cPannel podemos encontrar los siguientes datos para otorgar permisos para la manipulación de la base de datos y demás opciones como lo describe la ilustración:

1. Hace referencia al nombre de la base de datos
2. Hace referencia a los usuarios asignados a la base
3. Opción para renombrar la base de datos
4. Opción para eliminar la base de datos

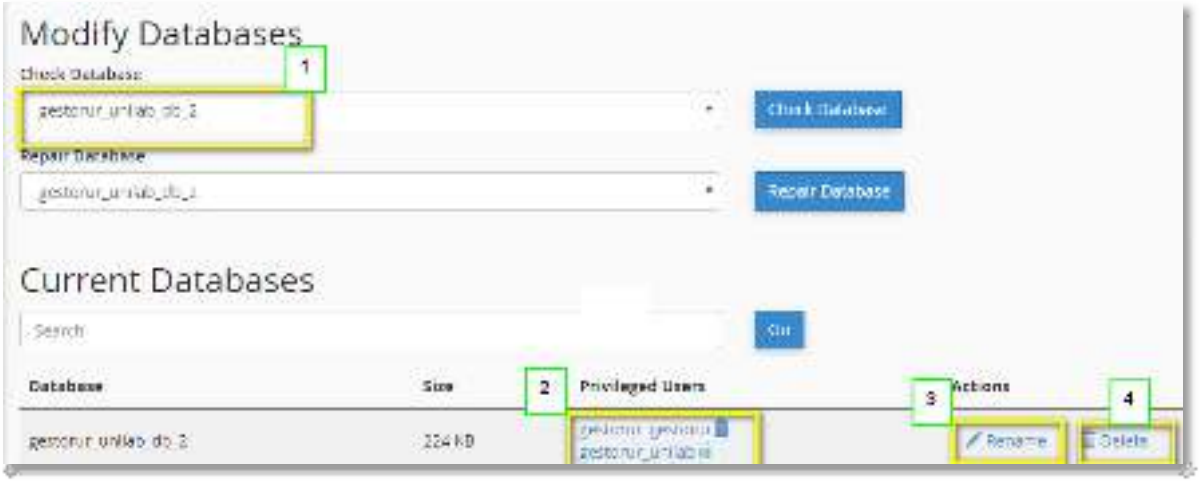

Ilustración 8.MySQL Databases, fuente autor

Nota importante:

El hosting actual solo cuenta con la opción de tener una sola base de datos al tiempo; si desea hacer uso de este hosting instalando más bases de datos, contacte al proveedor.

8. DICCIONARIO DE DATOS

A continuación de detallan cada una de las tablas del proyecto teniendo en cuenta el campo, el tipo y la cantidad de caracteres

Tabla administrador

Campo	Tipo	Caracteres
nombre	text	45
direccion	text	60
cedula	int	15
lugraExpedicion	varchar	40
telefono	int	15
email	varchar	30
password	varchar	255
FK_idAdministrador	varchar	10

Tabla preguntas_cuestionario

Campo	Tipo	Caracteres
PK_id	int	11
pregunta	varchar	255
tipo	text	15

Tabla usuario

Campo	Tipo	Caracteres
nombre	text	45
dirección	varchar	60
cedula	int	15
lugarExpedicion	varchar	40
teléfono	int	15
nombreUsuario	text	12
email	varchar	30
password	varchar	255
PK_idUsuario	int	10

Tabla docente

Campo	Tipo	Caracteres
nombre	varchar	45
dirección	varchar	60
cedula	int	15
lugarExpedicion	varchar	40
teléfono	int	15
codigoDocente	int	10
email	varchar	30
password	varchar	255
PK_idDocente	int	10

Ilustración 9.Diccionario de datos, tablas docentes, fuente autor

Tabla denuncia

Campo	Tipo	Caracteres
PK_idDenuncia	int	20
estado	varchar	20
geoReferencia	varchar	300
idUsuario	int	20
IdDocente	int	20
imágenes	text	
fecha	timestamp	
notas	text	

Ilustración 10...Diccionario de datos, tabla denuncia, fuente autor

Tabla docente

Campo	Tipo	Caracteres
PK_idDocente	int	20
codigoDocente	int	20
especialidadDocente	varchar	30

Ilustración 11...Diccionario de datos, docente, fuente autor

Tabla respuesta

Campo	Tipo	Caracteres
PK_idResp	int	20
descripcionRespuesta	text	
idDenuncia	int	20

Ilustración 12...Diccionario de datos, tabla respuesta, fuente autor