



CICLO III – DESARROLLO DE SOFTWARE

SPRINT 4 – Diseño e Implementación de portal de acceso usando método de autenticación basado en usuario y contraseña

Adrián García

José Luis Viveros

Juan Carlos Salazar Mesa

Trabajo presentado al experto facilitador

OSCAR VANEGAS SUAREZ
Ingeniero de Sistemas

**MINISTERIO DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS
COMUNICACIONES
UNIVERSIDAD DEL NORTE**

Barranquilla, 16 de diciembre de 2020



TABLA DE CONTENIDO

SPRINT 1

1. Enunciado del proyecto
2. Identificación de los requerimientos funcionales
3. Diagramas de Casos de Uso
4. Diagramas de Clases
5. Diagrama de Actividades

SPRINT 2

6. Mapa del Sitio
7. HTML
8. CSS
9. JavaScript

SPRINT 3

10. Tabla de funciones
11. Mockup funciones

SPRINT 4

12. Diseño e implementación de bases de datos
13. Diseño e implementación de portal de acceso usando método de autenticación basado en usuario y contraseña
14. Implementación de certificado SSL

1. Enunciado del proyecto

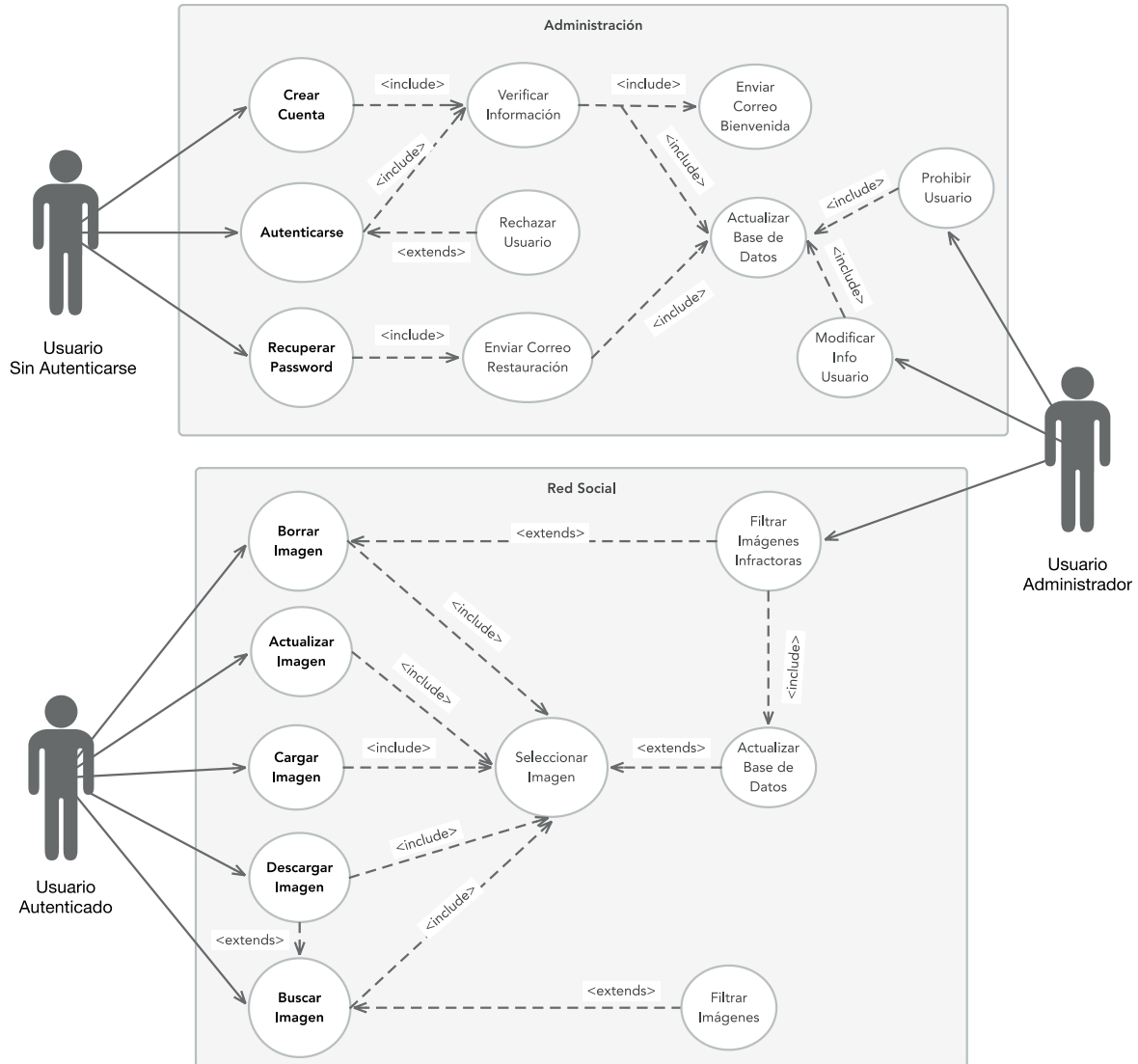
Se desea crear una red social para los diseñadores y programadores la cual facilite la obtención de imágenes para nuestro portafolio y/o nuestras paginas web.

2. Identificación de los requerimientos funcionales

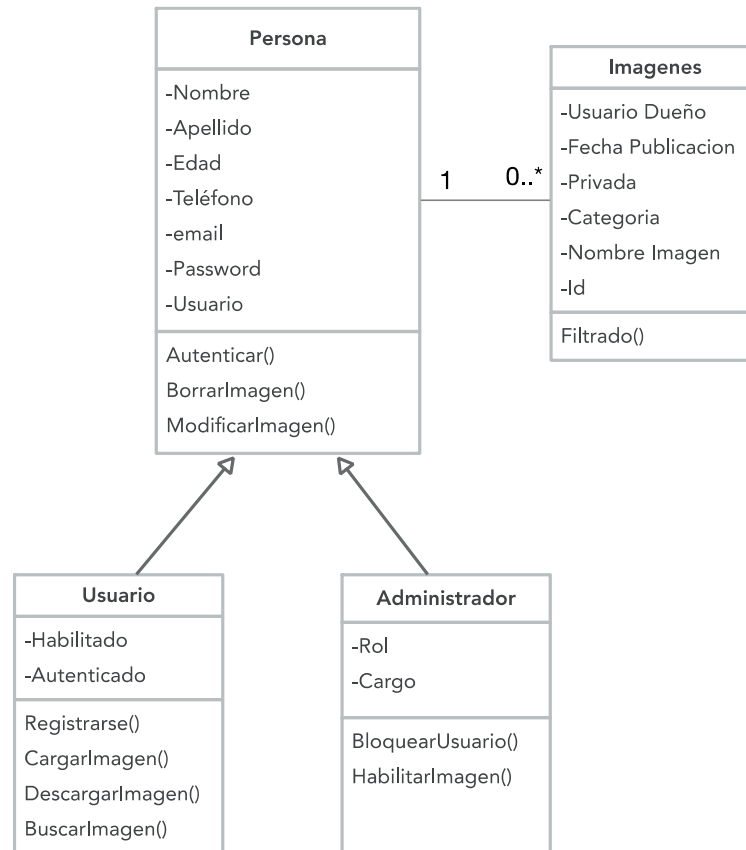
El sistema permitirá el registro de usuarios, con un nombre de usuario único, contraseña y correo electrónico, evaluando la sintaxis del correo, el tipo de datos ingresados, y validando que el usuario sea único en el sistema.

- El sistema enviara un correo electrónico cuando se registre un usuario para validar el correo y activar el registro.
- El sistema 3assword de un portal de acceso donde el usuario ingrese el correo registrado y su contraseña validando el ingreso autorizado al sistema con los mínimos requerimientos de seguridad.
- En el portal de acceso el sistema proveerá de la opción de recuperar contraseña enviando la contraseña a través del correo electrónico.
- El sistema debe brindar a los usuarios autorizados la opción de crear, actualizar y eliminar imágenes de la plataforma, y a su vez debe tener la opción de publicar o privatizar la imagen. Toda imagen debe guardarse con un nombre específico.
- El sistema debe proveer de un portal de búsqueda donde los usuarios autenticados puedan buscar las imágenes publicas por medio de la palabra clave nombre. Cuando el usuario ingrese a la imagen debe tener un botón para descargar la misma.

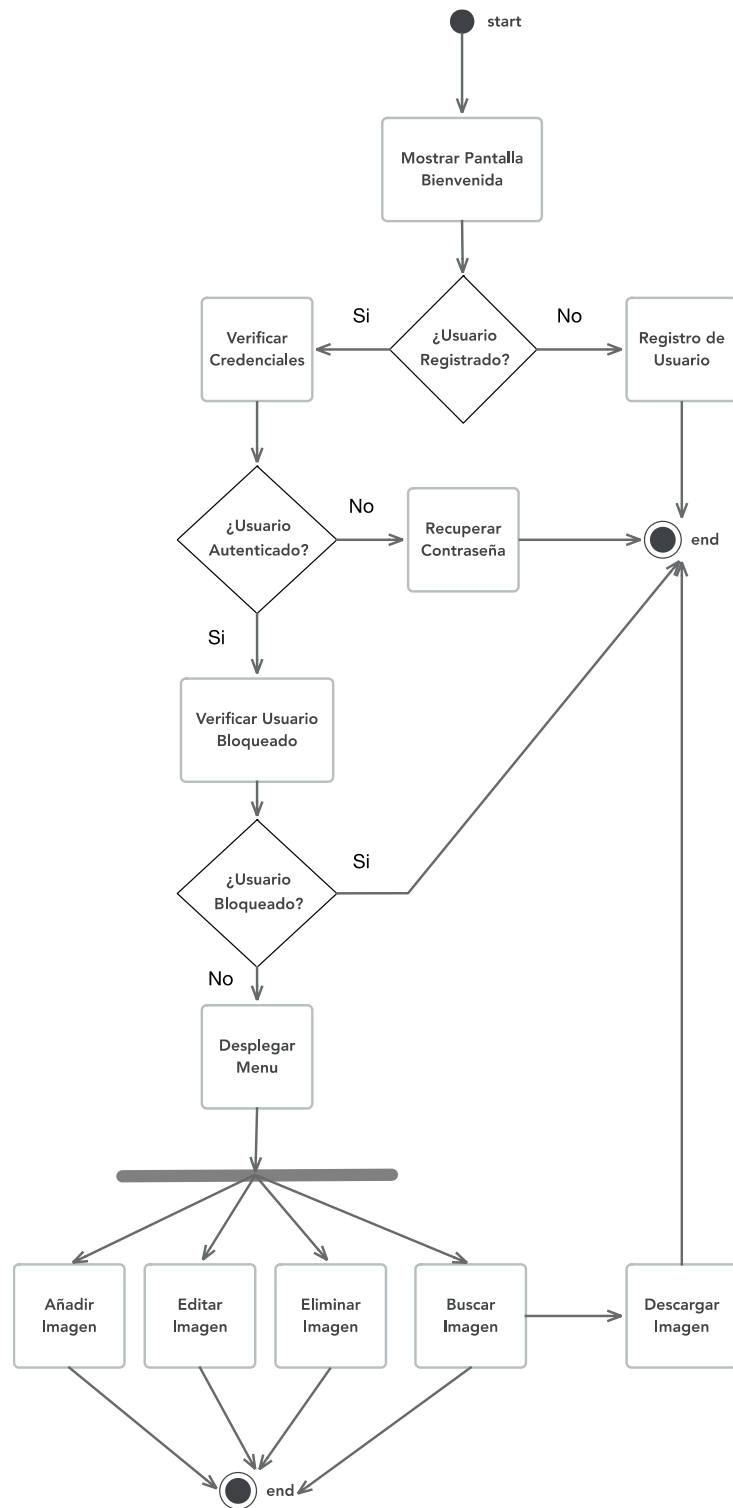
3. Diagramas de Casos de Uso



4. Diagrama de Clases



5. Diagrama de Actividades



6. Mapa del Sitio

El mapa del sitio se muestra en la Figura 1 y describe el funcionamiento de la página web. Inicia en una vista correspondiente al panel de control con las funciones de inicio de sesión (*login*), registro (*sign up*) y contacto.

Inicio de sesión y registro están conectadas o llevan al usuario a la vista principal (*main*) que contiene las funciones de configuración, cerrar sesión y buscar, agregar y editar foto. Por su parte contacto guarda la información del usuario en la base de datos.

SITEMAP:

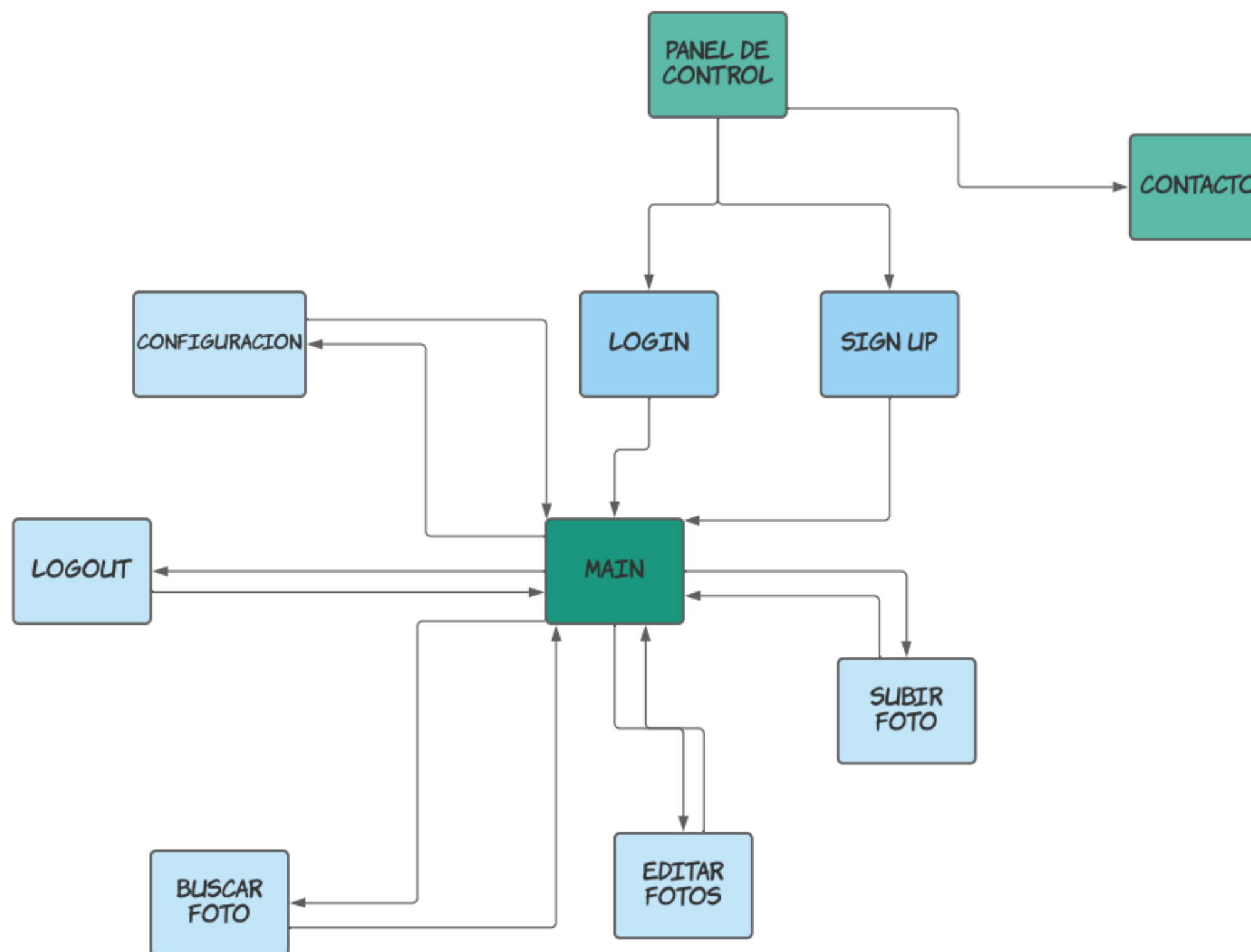


Figura 1. Ilustración del mapa del sitio

7. HTML

Como se mencionó en el mapa del sitio, existen varias vistas de interacción con el usuario, a continuación, se presentan en el orden de acceso del usuario.

a. Index.html

Corresponde a la pestaña de inicio de sesión, registro o ingreso como invitado para usuario de la página, está compuesta por el código mostrado en la Figura 2 que crea la imagen del logo, los campos de texto de usuario y contraseña y adicionalmente, los botones de crear cuenta, ingresar e invitado. Dichos elementos se pueden observar en la Figura 3.

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>SocialNetwork</title>
<link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
<link href="template.css" rel="stylesheet" type="text/css">
</head>

<body>
<div class="center">
  <table width="900" border="0" align="center">
    <tbody>
      <tr>
        <td></td>
        <td><p align="center">Usuario <br><input type="text"></p>
          <p align="center">Contraseña <br><input type="password"></p>
          <p align="center"><button type="button" class="Button1">Crear Cuenta</button></p>
          <p align="center"><button type="button" class="Button2">Ingresar</button></p>
          <p align="center"><button type="button" class="Button2">Invitado</button></p></td>
        </tr>
      </tbody>
    </table>
  </div>
  <footer id="footer" align="center">Photogram, Derechos reservados © 2020 </footer>
</body>
</html>
```

Figura 2. Código HTML del index o inicio



USUARIO

CONTRASEÑA

CREAR CUENTA

INGRESAR

INVITADO

PHOTOGRAM, DERECHOS RESERVADOS ® 2020

Figura 3. Vista de usuario generada por index.html

b. Signup.html

Corresponde a la pestaña de registro, está compuesta por el código mostrado en la Figura 4 que crea la imagen del logo, los campos de texto de usuario, nombre, correo electrónico, teléfono y fecha de nacimiento. Adicionalmente, el botón de ingreso de imagen de perfil y el botón de siguiente. Dichos elementos se pueden observar en la Figura 5.

```

<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Signup</title>
<link href="template.css" rel="stylesheet" type="text/css">
<link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
</head>

<div class="center">
<table width="900" border="0">
  <tbody>
    <tr>
      <td>
        
      </td>
      <td>
        <form>
          <p>Usuario<br><input type="text"></p>
          <p>Nombre<br><input type="text"></p>
          <p>Correo Electronico<br><input type="email"></p>
          <p>Telefono<br><input type="tel"></p>
          <p>Fecha de Nacimiento<br><input type="date"></p>
        </form>
      </td>
      <td>
        <div class="container">
        <div class="centered">+ Añadir Imagen</div>
        </div>
        <div class="container" style="margin-top: 185px">
          <button type="button" class="Button2" >Siguiete</button>
        </div>
      </td>
    </tr>
  </tbody>
</table>
</div>
<body>
<footer id="footer" align="center">Photogram, Derechos reservados © 2020</footer>
</body>
</html>

```

Figura 4. Código HTML del registro




USUARIO

NOMBRE

CORREO ELECTRONICO

TELEFONO

FECHA DE NACIMIENTO

SIGUIENTE

PHOTOGRAM, DERECHOS RESERVADOS ® 2020

Figura 5. Vista de usuario generada por signup.html

c. Main.html

Corresponde a la pestaña principal donde el usuario puede modificar su foto de perfil, buscar, agregar (subir) y editar fotos, así como entrar a configuración de cuenta y cerrar sesión. Está compuesta por el código mostrado en las Figura 6 y Figura 7. Adicionalmente, en la pestaña principal el usuario puede revisar el contenido ingresado previamente y el contenido de otros usuarios. Dichos elementos se pueden observar en la Figura 8.

```

<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>main</title>
<link href="template.css" rel="stylesheet" type="text/css">
<link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,300&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>

<body>
  <div class="main-container">

    <div class="user-container">
      <div>
        <div class="container">
        <h3 style="color: #27BBAA">Paula Aguilera</h3>
      </div>
    </div>
    <div class="user--menu">
      <a href="MainScreen.html"><i class="material-icons">home</i>Inicio</a><br>
      <br>
      <a href="SearchScreen.html"><i class="material-icons">image_search</i> Buscar Foto</a><br>
      <br>
      <a href="#"><i class="material-icons">add_circle</i> Agregar Foto</a><br><br>
      <a href="#"><i class="material-icons">brush</i> Editar Foto</a><br><br>
      <a href="#"><i class="material-icons" style="padding-top: 130px">settings</i> Configuración</a><br><br>
      <a href="#"><i class="material-icons">exit_to_app</i> Cerrar Sesión</a><br>
    </div>
  </div>

```

Figura 6. Código HTML de la pestaña Principal (1/2)

```
<div class="main-feed">
  <h1 style="font-size: 30px; text-align: left; color: ■ rgb(112, 112, 112); margin: 10px;"> Bienvenida,</h1>
  |   <h1 style="font-size: 20px; text-align: left; color: ■ rgb(39, 187, 165); margin: 10px;"> Paula</h1>
  <ul class="galeria">
    <li>
      <a href=""></a>
       Mariana Gutierrez
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>

    </li>
    <li>
      <a href=""></a>
       Emilia Perez
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>
    </li>
    <li>
      <a href=""></a>
       Jacobo Meza
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>
    </li>
    <li>
      <a href=""></a>
       Rasmus Rodriguez
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>
    </li>
    <li>
      <a href=""></a>
       Emilia Perez
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>
    </li>
  </ul>
</div>
</div>
</body>
</html>
```

Figura 7. Código HTML de la pestaña Principal (2/2)



PAULA AGUILERA

INICIO

BUSCAR FOTO

AGREGAR FOTO

EDITAR FOTO

CONFIGURACIÓN

CERRAR SESIÓN

BIENVENIDA,

PAULA



MARIANA GUTIERREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



EMILIA PEREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



JACOBO MEZA

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



RASMUS RODRIGUEZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



EMILIA PEREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

Figura 8. Vista de usuario generada por main.html

d. Search.html

Corresponde a la pestaña de búsqueda de fotos en la aplicación, está compuesta por el código mostrado en las Figura 9 y Figura 10 que permite el ingreso del nombre o código de la imagen para acceder al contenido. Esta vista se puede observar en la Figura 11.

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>search</title>
<link href="template.css" rel="stylesheet" type="text/css">
<link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,300&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>

<body>
  <div class="main-container">

    <div class="user-container">
      <div>
        <div class="container">
        <h3 style="color: #27BBA5">Paula Aguilera</h3>
      </div>
    </div>
    <div class="user--menu">
      <a href="MainScreen.html"><i class="material-icons">home</i> Inicio</a><br>
      <br>
      <a href="SearchScreen.html"><i class="material-icons">image_search</i> Buscar Foto</a><br>
      <br>
      <a href="#"><i class="material-icons">add_circle</i> Agregar Foto</a><br><br>
      <a href="#"><i class="material-icons">brush</i> Editar Foto</a><br><br>
      <a href="#"><i class="material-icons" style="padding-top: 130px">settings</i> Configuración</a><br><br>
      <a href="#"><i class="material-icons">exit_to_app</i> Cerrar Sesión</a><br>
    </div>
  </div>
```

Figura 9. Código HTML de la pestaña de búsqueda (1/2)


```
<div class="main-feed">
  <h1 style="font-size: 30px; text-align: left; color: rgb(112, 112, 112); margin: 30px;"> Buscar Imagen</h1>
  <p><input type="text" style="width: 200px;margin-left: 30px;"></p>
  <ul class="galeria">
    <li>
      <a href=""></a>
       Mariana Gutierrez
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>
    </li>
    <li>
      <a href=""></a>
       Emilia Perez
      <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!</h2>
      <h3>Categoría: Lorem ipsum.</h3>
    </li>
  </ul>
</div>
</div>
</body>
</html>
```

Figura 10. Código HTML de la pestaña de búsqueda (2/2)



BUSCAR IMAGEN



MARIANA GUTIERREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



EMILIA PEREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

Figura 11. Vista de usuario generada por search.html



e. **Add.html**

Corresponde a la pestaña de adición de fotos en la aplicación, está compuesta por el código mostrado en la Figura 12 que crea un botón de subir archivo y otro de postear la foto, una caja para la adición de comentario y un *check box* para ajustar el estado de la foto a público o privado. Esta vista se puede observar en la Figura 13.

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>add</title>
<link href="template.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,300&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
<script src="add.js"></script>
</head>
<script src="https://code.jquery.com/jquery-3.4.0.min.js"></script>
<body>


  <div class="main-container">

    <div class="user-container">
      <div>
        <div class="container">
        <h3 style="color: #278BA5">Paula Aguilera</h3>
        </div>
      </div>
      <div class="user--menu">
        <a href="MainScreen.html"><i class="material-icons">home</i> Inicio</a><br>
        <br>
        <a href="SearchScreen.html"><i class="material-icons">image_search</i> Buscar Foto</a><br>
        <br>
        <a href="#"><i class="material-icons">add_circle</i> Agregar Foto</a><br><br>
        <a href="#"><i class="material-icons">brush</i> Editar Foto</a><br><br>
        <a href="#"><i class="material-icons" style="padding-top: 130px">settings</i> Configuración</a><br><br>
        <a href="#"><i class="material-icons">exit_to_app</i> Cerrar Sesión</a><br>
      </div>
    </div>


    <div class="main-feed">
      <h1 style="font-size: 30px; text-align: left; color: rgb(112, 112, 112); margin: 30px;"> Agregar Foto</h1>


      <form action="">
        <div class="add-container">
          <img id="preview"></img><br>
          <input type="file" name="file" id="file" accept="image/*" onchange="previewImage();">
          <input class="Button2" style="font-size: 20px; margin-bottom: 20px; color: white" type="button" value="Subir Archivo" onclick="document.getElementById('file').click()>
          <br>
          <label style="color: rgb(112,112,112); margin: 5px; font-family: 'Bebas neue'; text-align: left;" for="comentario"> Agrega Comentario</label><br>
          <input name="comentario" class="comentario" type="text" id="comentario"><br>
          <input name="privado" class="privado" type="checkbox" id="privado" value="esPrivado">
          <label style="color: rgb(112,112,112); margin: 5px; font-family: 'Bebas neue'; text-align: left;" for="privado"> Privado</label><br>
          <input name="postear" type="submit" class="Button2" style="font-size: 20px; margin-top: 20px; color: white" value="Postear Foto">
        </div>
      </form>
    </div>
  </div>
</body>
</html>
```


Figura 12. Código HTML de la pestaña de adición de foto





PAULA AGUILERA


 INICIO

 BUSCAR FOTO

 AGREGAR FOTO

 EDITAR FOTO

 CONFIGURACIÓN

 CERRAR SESIÓN

AGREGAR FOTO

SUBIR ARCHIVO

AGREGA COMENTARIO

☐ PRIVADO

POSTEAR FOTO

Figura 13. Vista de usuario generada por add.html

Corresponde a la pestaña de edición de fotos en la aplicación, está compuesta por el código mostrado en la Figura 14 que permite mostrar los archivos subidos por el usuario para edición de comentarios y cambio de estado de privacidad. Así como para eliminar un *post*. Esta vista se puede observar en la Figura 15.

```

<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>edit</title>
<link href="template.css" rel="stylesheet" type="text/css">
<link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,300&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>

<body>

<div class="main-container">

<div class="user-container">
<div>
<div class="container">
<h3 style="color: #27B8A5">Paula Aguilera</h3>
</div>
</div>
<div class="user--menu">
<a href="MainScreen.html"><i class="material-icons">home</i> Inicio</a><br>
<br>
<a href="SearchScreen.html"><i class="material-icons">image_search</i> Buscar Foto</a><br>
<br>
<a href="#"><i class="material-icons">add_circle</i> Agregar Foto</a><br><br>

<a href="#"><i class="material-icons">brush</i> Editar Foto</a><br><br>
<a href="#"><i class="material-icons" style="padding-top: 130px">settings</i> Configuración</a><br><br>
<a href="#"><i class="material-icons">exit_to_app</i> Cerrar Sesión</a><br>
</div>
</div>

<div class="main-feed">

<ul class="galeria">
<li>
<a href=""></a>
<h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!<i class="material-icons" style="font-size: 15px; top: 3px; margin-1">
<h3>Categoría: Lorem ipsum. <i class="material-icons" style="font-size: 20px; top: 3px; margin-left: 5px">edit</i></h3>
<input type="checkbox"> Privado <i class="material-icons" style="font-size: 22px;padding-left: 20px; top: 3px; margin-left: 5px;color: red">delete</i>

```

Figura 14. Código HTML de la pestaña de la edición de fotos

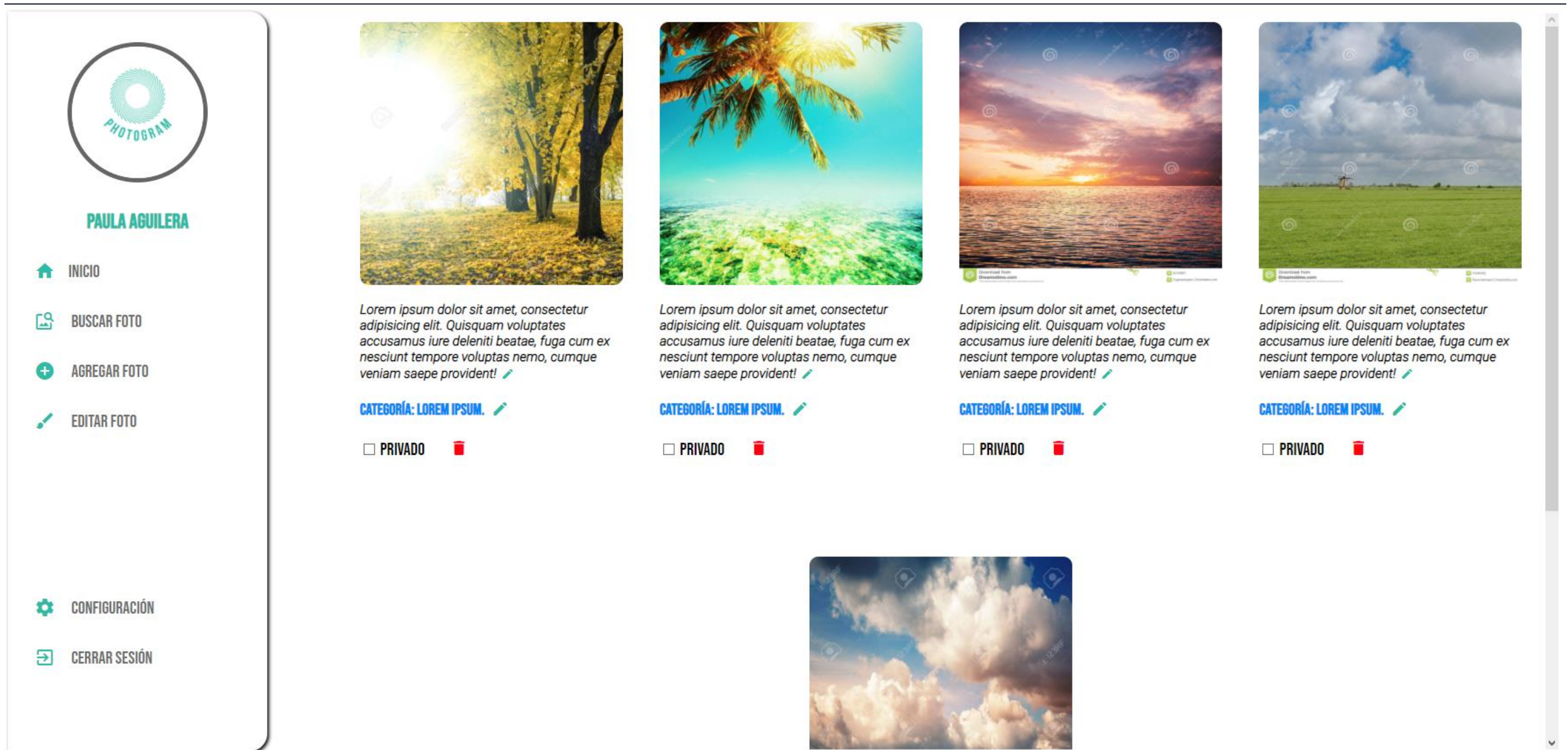


Figura 15. Vista de usuario generada por edit.html

8. CSS

Para toda la parte visual de las vistas se utilizó el código template.css que se muestra parcialmente a continuación:

```
@charset "UTF-8";
/* CSS Document */
body {
    font-family: 'Bebas Neue', cursive;
    font-size: 20px;
}

a{
    text-decoration: none;
    color: #6f6f6f;
}

input{
    border-radius: 20px;
    padding: 9px 28px;
}

button {
    font-family : inherit;
    font-size: 0.8em;
    border-radius: 20px;
    color: white;
    padding: 9px 28px;
}

.main-container {
    display: flex;
}

.user-container {
    left: 0;
    top: 0;

    flex-basis: 300px;
    box-shadow: 2px 2px 4px #000000;
    width: 400px;
    height: auto;
    border-radius: 0px 20px 20px 0px;
}

.imgRedonda {
    width:150px;
    height:150px;
    border-radius:150px;
    border:5px solid #666;
}

.Button1{
    background-color: #6F6F6F;
    font-family: 'Bebas Neue', cursive;
}
```

Figura 16. Código CSS utilizado para los ajustes visuales a los archivos html mostrados anteriormente

9. JavaScript

Finalmente, se realizó un archivo en lenguaje JavaScript con nombre add.js, encargado de previsualizar la imagen cargada. Extrae los archivos subidos en el input de tipo file, compara si el formato es correcto y después añade un objeto de tipo imagen a un div.

El código se muestra en la figura a continuación:

```
1  function previewImage() {  
2      var file = document.getElementById("file").files;  
3      if (file.length > 0) {  
4          var fileReader = new FileReader();  
5          fileReader.onload = function(event) {  
6              document.getElementById("preview").setAttribute("src", event.target.result);  
7              document.getElementById("preview").setAttribute("height", 500);  
8          };  
9          fileReader.readAsDataURL(file[0]);  
10     }  
11 }  
12 }// JavaScript Document
```

Figura 17. Código JS utilizado para interactuar con la página

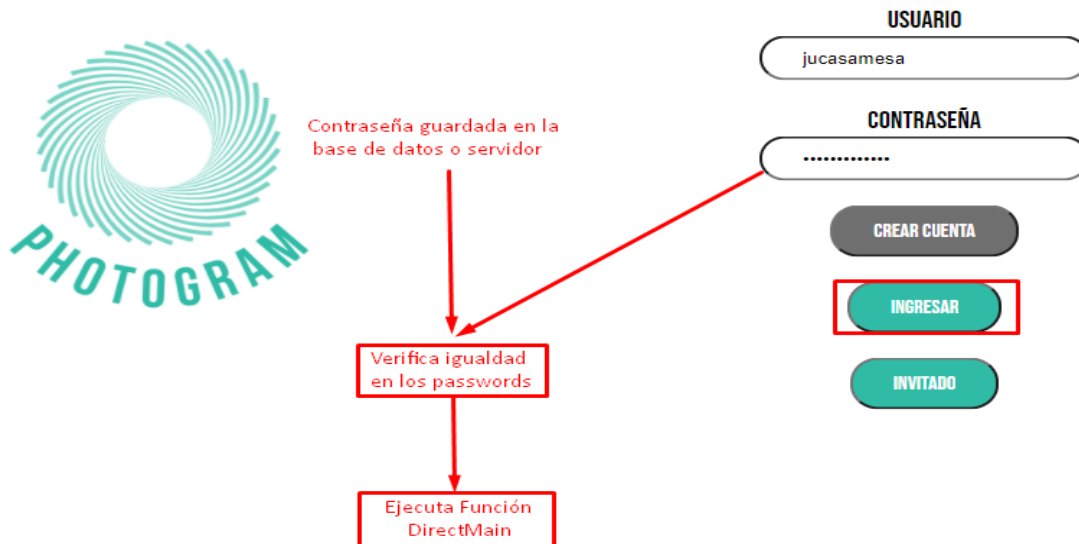
Vista	Nombre de la función	Rutas Asociadas	Parámetros	Métodos	Descripción	Mockups
Inicio o Login/index	CheckPass	index.html	27assword, password2	Post (Envío de datos de forma oculta para el usuario): Envío de las contraseñas para su comparación.	Se activa cuando el usuario presiona el botón Ingresar, verifica que la contraseña ingresada corresponda al usuario ingresado. En caso de hacer juego con el usuario se activa DirectMain() y en caso contrario envía un alert.	1
	DirectMain	main.html	N/A	N/A	Se activa cuando el usuario ingresa correctamente su contraseña redirigiéndolo a la vista principal.	2
	HidePass	index.html	N/A	N/A	Oculto la contraseña.	3
	ShowPass	index.html	N/A	N/A	Muestra la contraseña.	4
Registro o Sign up	CheckFields	signup.html	usuario, nombre, 27assword, password2, correo, tel, date	Post: Envío de los campos para realizar las verificaciones.	Se activa cuando el usuario presiona el botón siguiente, verifica que: a) El campo usuario no sea usuario repetido. B) El campo correo incluya el símbolo @. C) El campo teléfono no incluya letras. D) Que todos los campos estén llenos.	5
	LoadDB	signup.html	usuario, nombre, 27assword, password2, correo, tel, date	Post: Envío de variables a la base de datos.	Se activa cuando el usuario presiona el botón siguiente y se verifican correctamente los platos, incorpora el usuario a la Base de Datos.	6
	DirectMain	main.html	N/A	N/A	Luego de cargarse los datos a la base, se redirige al usuario a la vista principal.	2
Main	LoadScreen	main.html	imagen_n (representando las imágenes de la base de datos)	Post: Envío de variables desde la base de datos a la vista.	Se activa al iniciarse la ventana principal, se encarga de cargar las fotos.	7
	DirectSearch	index.html	N/A	N/A	Se activa cuando el usuario presiona el botón buscar foto, direcciona a la vista buscar foto.	8
	DirectAdd	add.html	N/A	N/A	Se activa cuando el usuario presiona el botón agregar foto, direcciona a la vista adicionar foto.	9
	DirectEdit	edit.html	N/A	N/A	Se activa cuando el usuario presiona el botón editar foto, direcciona a la vista de edición.	10
	Logout	index.html	N/A	N/A	Se activa cuando el usuario presiona el botón de cerrar sesión, direcciona a la vista de inicio.	11
Add	UpLoad	add.html	N/A	N/A	Se activa cuando el usuario presiona el botón de subir archivo, activa en el computador del usuario la ventana de búsqueda de archivo para subir la imagen.	12
	Status	add.html	status	Guarda la variable de nombre status	Se activa cuando el usuario presiona el check box. Guarda el estado de la variable booleana que indica si la imagen es pública o privada.	13
	LoadPost	add.html	imagen_n (representando las imágenes de la base de datos), category, comment, status.	Post: Envío de variables a la base de datos. Comentario, Category, Status e Imagen.	Se activa cuando el usuario presiona el botón Postear Foto. Incorporando a la ventana principal la imagen, el comentario y el status (privada/pública).	14

Search	Search	search.html	category	Get: Envío de las imágenes usando la URL.	Se activa cuando el usuario presiona la casilla de búsqueda, lo que permite al programa a través de los caracteres ingresados buscar las imágenes cargadas.	15
	Show	search.html	imagen_n (representando las imágenes de la categoría llamada)	Post: Envío de variables desde la base de datos a la vista.	Esta función se utiliza luego de la búsqueda sea exitosa y muestra la foto que hace juego con los caracteres ingresados. En caso de no encontrar se envía una alerta al usuario.	16
	Download	search.html	imagen escogida	Post: El usuario elige la imagen. Get: Recibe la imagen usando la URL.	Luego de que la función mostrar foto se haya llevado a cabo se habilita el botón de descargar foto que le permite al usuario descargar la foto encontrada.	17
Edit	LoadScreen	edit.html	imagen_n (representando las imágenes de la base de datos)	Post: Envío de variables desde la base de datos a la vista.	Se activa al iniciarse la ventana de edición, se encarga de cargar las fotos, comentarios y status.	7
	EditCategory	edit.html	category	Get: Guarda la nueva categoría en la variable de la vista.	Se activa cuando el usuario presiona el botón de editar categoría, permitiéndole al usuario cambiar la categoría ingresada previamente.	18
	EditComment	edit.html	comment	Get: Guarda el nuevo comentario en la variable de la vista.	Se activa cuando el usuario presiona el botón de editar comentario, permitiéndole al usuario cambiar los caracteres ingresados previamente.	19
	ChangeStatus	edit.html	status	Get: Guarda el nuevo status en la variable de la vista.	Se activa cuando el usuario presiona el botón de status, permitiéndole al usuario cambiar de público a privado o viceversa.	20
	Delete	edit.html	category, comment, status	Post: Elimina el id de la imagen seleccionada en la base de datos.	Se activa cuando el usuario presiona el botón de eliminar foto, borrando la imagen del servidor y/o base de datos.	21
	Save	edit.html	category, comment, status	Post: Realiza el update de la imagen seleccionada en la base de datos.	Se activa cuando el usuario presiona el botón de actualizar habilitado luego de algún cambio en el post, permitiendo guardar los cambios realizados.	22

Tabla 1. Tabla de las funciones utilizadas en la app Photogram

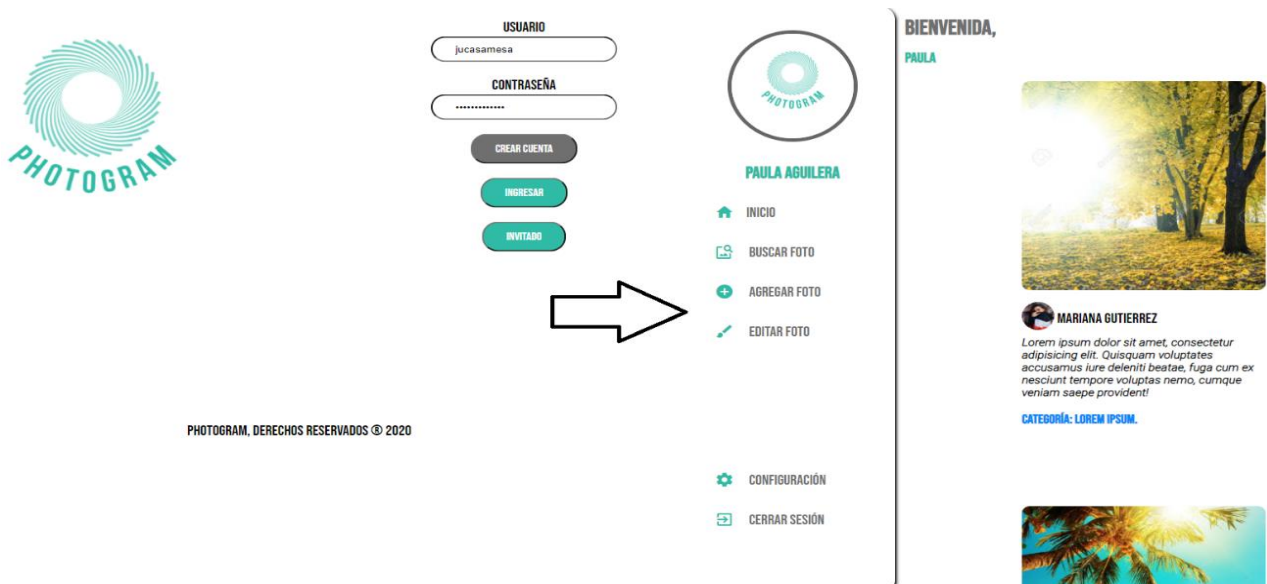
11. Mockups Funciones

1. CheckPass



PHOTOGRAM, DERECHOS RESERVADOS © 2020

2. DirectMain



PHOTOGRAM, DERECHOS RESERVADOS © 2020

3. HidePass



USUARIO

jucasamesa

CONTRASEÑA

.....



CREAR CUENTA

INGRESAR

INVITADO

PHOTOGRAM, DERECHOS RESERVADOS ® 2020

4. ShowPass



USUARIO

jucasamesa

CONTRASEÑA

monkey123



CREAR CUENTA

INGRESAR

INVITADO

PHOTOGRAM, DERECHOS RESERVADOS ® 2020

5. CheckFields

Verifica que el usuario no sea repetido

Verifica que incluya el simbolo: @

Verifica que no hayan letas en el telefono

Verifica que todos los campos esten llenados

USUARIO
jucasamesa

CONTRASEÑA
monkey123

REPETIR CONTRASEÑA
monkey123

NOMBRE
Juan Salazar

CORREO ELECTRONICO
jsalazar@uninorte.edu.co

TELEFONO
3004654659

FECHA DE NACIMIENTO
27 / 01 / 1993

+ AÑADIR IMAGEN
PHOTOGRAM

SIGUIENTE

PHOTOGRAM, DERECHOS RESERVADOS © 2020

6. LoadDB

USUARIO
jucasamesa

CONTRASEÑA
monkey123

REPETIR CONTRASEÑA
monkey123

NOMBRE
Juan Salazar

CORREO ELECTRONICO
jsalazar@uninorte.edu.co

TELEFONO
3004654659

FECHA DE NACIMIENTO
27 / 01 / 1993


+ AÑADIR IMAGEN
PHOTOGRAM

SIGUIENTE

Luego de ejecutarse la función CheckFields incorpora los datos del nuevo usuario a la base de datos.

PHOTOGRAM, DERECHOS RESERVADOS © 2020

7. LoadScreen




PAULA AGUILERA


- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN

BIENVENIDA,
PAULA




MARIANA GUTIERREZ
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!
CATEGORÍA: LOREM IPSUM.




EMILIA PEREZ
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!
CATEGORÍA: LOREM IPSUM.

Carga los Post antiguos.



8. DirectSearch




PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN


BUSCAR IMAGEN



MARIANA GUTIERREZ
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!
CATEGORÍA: LOREM IPSUM.

Envía al usuario de la vista principal a la vista de búsqueda.

9. DirectAdd



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN

AGREGAR FOTO

SUBIR ARCHIVO

AGREGA COMENTARIO

☐ PRIVADO

POSTEAR FOTO

Direcciona al usuario a la vista de agregar foto

10. DirectEdit



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

☐ PRIVADO

Direcciona al usuario a esta vista de edición

11. Logout



USUARIO

jucasamesa

CONTRASEÑA

.....

CREAR CUENTA


INGRESAR

INVITADO

Direcciona al usuario a la vista de inicio o de login

PHOTOGRAM, DERECHOS RESERVADOS © 2020

12. UpLoad



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO
- CONFIGURACIÓN
- CERRAR SESIÓN

AGREGAR FOTO

Subir Archivo

AGREGA COMENTARIO

☐ PRIVADO

POSTEAR FOTO

Abrir


Nombre: IMG_20171016_105928

Ficheros de imagen

Abrir Cancelar


Sube la imagen a la vista

13. Status



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO
- CONFIGURACIÓN
- CERRAR SESIÓN



SUBIR ARCHIVO


AGREGA COMENTARIO

☒ PRIVADO

POSTEAR FOTO


El usuario elige si la imagen y comentarios a subir serán públicos o privados.

14. LoadPost



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO
- CONFIGURACIÓN
- CERRAR SESIÓN



SUBIR ARCHIVO

AGREGA COMENTARIO

Comentario

☒ PRIVADO

POSTEAR FOTO

Modifica la vista principal del usuario, adicionando el post mostrado

15. Search



PAULA AGUILERA


- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

Si una o varias imagenes hacen juego con los caracteres ingresados se activa la función Show.

- CONFIGURACIÓN
- CERRAR SESIÓN

BUSCAR IMAGEN

Toma los caracteres adicionados en el campo y busca coincidencia con las imagenes guardadas.




MARIANA GUTIERREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

Si ninguna imagen hace juego con los caracteres se envía una alerta al usuario.

16. Show




PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN

BUSCAR IMAGEN

Según los caracteres adicionados en el campo, filtra las imagenes de los usuarios.



MARIANA GUTIERREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

17. Download



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN

BUSCAR IMAGEN

arboles



MARIANA GUTIERREZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



Se activa el botón de descargar, permitiéndole al usuario obtener el archivo.

18. EditCategory



PAULA AGUILERA

- INICIO
- BUSCAR FOTO
- AGREGAR FOTO
- EDITAR FOTO

- CONFIGURACIÓN
- CERRAR SESIÓN



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.



☐ PRIVADO



Permite al usuario cambiar la categoría para cambiar la agrupación de sus fotos.

19.EditComment



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

☐ PRIVADO

Permite al usuario cambiar el comentario ingresado previamente

20.ChangeStatus



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

☒ PRIVADO

Permite cambiar el estado del post.

21.Delete



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

☒ PRIVADO



Permite al usuario eliminar el post completo.

22.Save



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisquam voluptates accusamus iure deleniti beatae, fuga cum ex nesciunt tempore voluptas nemo, cumque veniam saepe provident!

CATEGORÍA: LOREM IPSUM.

☒ PRIVADO



Una vez realizado alguna modificación se activa el botón de actualizar, que permite guardar los cambios realizados.

12. Diseño e implementación de bases de datos.

En esta sección se diseñó e implementó la base de datos para la aplicación Web, teniendo en cuenta las funcionalidades deseadas para la aplicación y sus formularios de registro e inicio de sesión.

Inicialmente se realizó el diagrama relacional:

12.1. Diagrama Relacional

Un diagrama relacional o de entidad-relación, se basa en modelo entidad relación o ERD y es un tipo de diagrama de flujo que ilustra cómo las "entidades", se relacionan entre sí dentro de un sistema. Estos diagramas son un reflejo de la estructura gramatical y emplean entidades como sustantivos y relaciones como verbos.

El diagrama realizado para Photogram consta de dos tablas, una tabla de **usuarios** que almacena los usuarios y administradores de la aplicación con nueve variables de identificación.

Las variables son **id**, que corresponde a la llave primaria de la tabla y es única para cada usuario y auto incrementable, lo que facilita la adición de información a la base. Posteriormente, se encuentra la variable **usuario** que funciona como identificador para inicio de sesión (variable que el usuario debe recordar), luego está la variable **nombre** seguida del **password** o contraseña. Las contraseñas son almacenadas de manera cifrada, como se explicará posteriormente en la sección 13. A continuación, se encuentran las variables de **correo**, **teléfono**, **date** (fecha de nacimiento), **link_user** que almacena el URL de la imagen de perfil registrada por el usuario y finalmente **rol** que guarda el tipo de usuario (usuario o admin).

La segunda tabla corresponde a **fotos**, esta se encarga de almacenar las fotos utilizadas y mostradas en la aplicación. Cuenta con seis variables, que corresponden a: **id_foto**, **id_usuario**, **esPrivada**, **category**, **comment** y **link_foto**.

Id_foto, al igual que **id** en la tabla usuarios, es la llave primaria, lo que la hace única para cada foto y también es auto incrementable. Por su parte, **id_usuario** es la llave foránea que conecta los usuarios con las fotos, teniendo en cuenta que la relación es de uno a muchos (usuarios a fotos), ya que un solo usuario puede publicar o descargar muchas fotos. Además, **esPrivada** guarda el estado de la foto, pública o privada, **category**, guarda la categoría en la que está clasificada la foto y **comment** guarda el comentario que adicionó el usuario al subir el Post a la aplicación. Finalmente, **link_foto** guarda el URL de la foto. A continuación, se muestra la Figura 18 que muestra el diagrama relacional de la aplicación con su correspondiente tipo de variable.

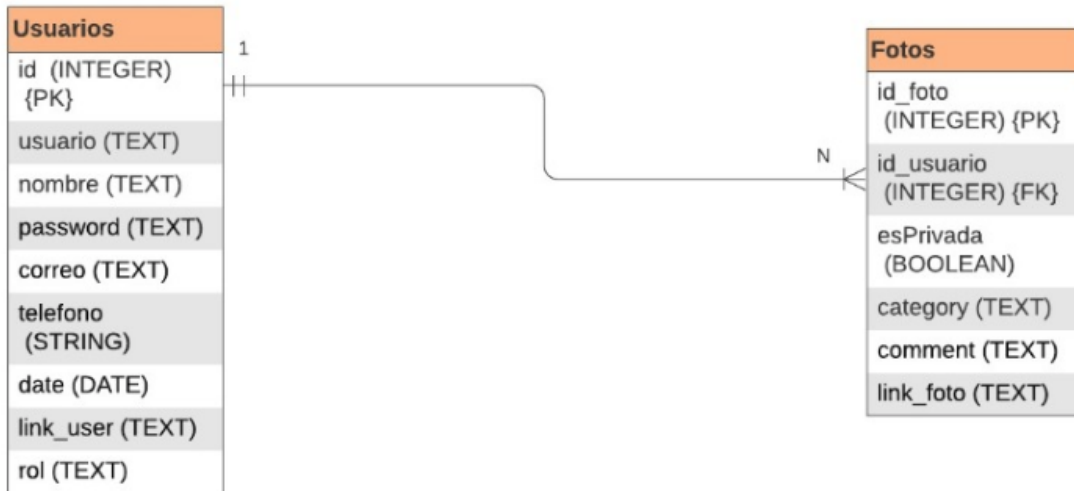


Figura 18. Diagrama Relacional Base de Datos Photogram

12.2. Gestor de base de datos

Se utilizó y utiliza SQLite3 para la creación y manipulación de las bases de datos por lo cual en Flask y Python se debieron importar las siguientes librerías y funciones: `import sqlite3` y `from sqlite3 import Error`.

De esta forma se podían ejecutar las queries o búsquedas a SQLite3 desde Python3, a continuación, se muestra un ejemplo de la ejecución de las búsquedas:

```

con = sqlite3.connect("BaseDatos.db") #Permite conectarse a la BaseDatos.db.
con.row_factory = sqlite3.Row #Permite capturer las filas.
cur = con.cursor() #Conecta al cursor con la base de datos.
cur.execute("SELECT * FROM usuarios") # Ejecuta la búsqueda.
rows = cur.fetchall() # Carga a la variable rows las filas encontradas que cumplen los
parámetros de la búsqueda.
con.close() # Cierra la conexión con la base de datos.
  
```

cursor.fetchall() recupera todas las filas del resultado de una consulta. Devuelve todas las filas como una lista de tuplas. Se devuelve una lista vacía si no hay ningún registro que recuperar. De esta forma según las columnas seleccionadas en la búsqueda se tomaba el dato requerido, ejemplo:

```

cur.execute("SELECT DISTINCT usuario FROM usuarios WHERE usuario = ?",
[usuario])
records = cur.fetchall()
print("Total rows are: ", len(records))
print("Printing each row")
  
```



```
temp = ""
for row in records:
    print("Usuario: ", row[0])
    temp = row[0]
if temp == usuario:
    flash("El usuario ya existe")
    cur.close()
```

El código mostrado anteriormente, se utiliza para verificar si el usuario ingresado en el registro ya existe o no. Informando en el momento de ejecución a través del método **flash**.

12.3. Validación de los datos de entrada

Para realizar la validación de datos en los formularios login.html y signup.html, se utilizaron los métodos del **wtfirms.validators**, DataRequired, InputRequired y Length. Importados a page.py de la siguiente forma:

```
from wtforms.validators import DataRequired, InputRequired, Length
```

De esta forma, al usuario llenar los formularios de registro y/o inicio de sesión se verifica que la información este completa en primero lugar. En segundo lugar, según el tipo de información que se requiera en el campo hay restricciones específicas.

Para el nombre de usuario y el nombre existe la restricción de mínimo 4 caracteres y máximo 20. Por su parte, la contraseña, mínimo 6 y máximo 20. El correo ya que es tipo de variable tipo email pide el símbolo especial "@" y adicionalmente no puede haber repetido un correo dentro de la base de datos.

El teléfono debe tener diez dígitos lo que permitiría en un futuro hacer verificación de seguridad por mensaje de texto. Además, la fecha de nacimiento tiene la restricción de variable tipo fecha y debe ser del formato día, mes, año (la interfaz gráfica ayuda al llenado de los campos).

Además, de las restricciones adicionadas a los campos por tipo de variable, por wtf.validators y por restricciones de SQL, se adicionó el comando "|safe" a los html de los formularios. Lo que evita que se llenen los campos con valores especiales que no concuerden con el tipo de dato.

A continuación, se muestra el código encargado de las validaciones desde el wtf:

```
class Login_Form(FlaskForm):
    username=StringField('usuario',validators=[InputRequired(message="Campo Requerido")])
    password=PasswordField('password',validators=[InputRequired(message="Campo Requerido")])
    loginbtn=SubmitField('Ingresar')

class Register_Form(FlaskForm):
    signupbtn=SubmitField('Registrarse')
    forgotbtn=SubmitField('Recuperar Password')

class Signup_Form(FlaskForm):
    username=StringField('usuario',validators=[validators.Length(min=4, max=20,message="Debe tener más de 4 caracteres"),validators.DataRequired(message="Campo Requerido")])
    name=StringField('nombre',validators=[validators.Length(min=4, max=20,message="Debe tener más de 4 caracteres"),validators.DataRequired(message="Campo Requerido")])
    password=PasswordField('password',validators=[validators.Length(min=6, max=20,message="Debe tener más de 6 caracteres"),validators.DataRequired(message="Campo Requerido"), validators.EqualTo('repassword', message=
    repassword=PasswordField('repassword')
    email=EmailField('email',validators=[validators.DataRequired(message="Campo Requerido")])
    telefono=TelField('telefono',validators=[validators.DataRequired(message="Campo Requerido"),validators.Length(min=10, max=10,message="Debe tener diez dígitos")])
    born= DateField('Nacimiento',validators=[validators.DataRequired(message="Campo Requerido")],format='%Y-%m-%d')
    nextbtn= SubmitField('Siguiete')

class Delete_Form(FlaskForm):
    userid= StringField('userid',validators=[validators.DataRequired(message="Id de usuario Requerido")])
    deletebtn= SubmitField('Borrar Usuario')
```

Figura 19. Código Python3 de la validación de campos utilizando WTF

12.4. Uso de **prepared staments** para consultar y actualizar bases de datos

Para realizar la modificación y las consultas necesarias para realizar operaciones de CRUD (Create, Read, Update and Delete), se utilizaron las sentencias preparadas que permitían ejecutar de manera segura las búsquedas para evitar inyección de código que afectara el funcionamiento de la aplicación.

Las sentencias fueron utilizadas según los requerimientos funcionales de los formularios, por ello para el login se utilizó:

“SELECT password, rol FROM usuarios WHERE usuario = ?”, encargada de leer los datos guardados del usuario en las variables **password** y **rol**. Para posteriormente, comprobar que la contraseña ingresada sea igual a la guardada en la base de datos y según el rol dirección al usuario a su respectiva vista de la aplicación. Usuario tipo usuario, hacia el main y el tipo admin hacia admin.

```
login
if login_form.is_submitted():
    if login_form.loginbtn.data and login_form.validate():
        usuario= login_form.username.data
        password= login_form.password.data

        if usuario != "" and password != "":

            try:
                con = sqlite3.connect('BaseDatos.db')
                cur = con.cursor()
                print("Connected to SQLite")
                cur.execute("SELECT password,rol FROM usuarios WHERE usuario = ? ",[usuario])
                records = cur.fetchall()
                size = len(records)
                print("Total rows are: ", size)
                print("Printing each row")

                if size == 0:
                    flash("Usuario o contraseña incorrectos")
                else:
                    for row in records:
                        print("password: ", row[0])
                        print("rol: ", row[1])
                        temp = row[0]
                        rol = row[1]

                        print(check_password_hash(temp,password))

                    if check_password_hash(temp,password) is True:

                        if rol == "admin":
                            session["admin"] = usuario
                            return redirect('admin')
                        else:
                            session["usuario"] = usuario
                            return redirect('main')
                        cur.close()

            except sqlite3.Error as error:
                print("Failed to read data from table", error)

            finally:
                if (con):
                    con.close()
                    print("The Sqlite connection is closed")

        else:
            flash("Usuario o contraseña invalidos")
```

Figura 20. Código Python3 de la validación de contraseña en el login



Por su parte en el registro, se utilizaron dos sentencias, la primera para verificar si el usuario ingresado en el registro ya existía o no, y posteriormente una sentencia para adicionar el nuevo registro y sus campos a la base de datos.

A continuación se muestra la primera sentencia: `cur.execute("SELECT DISTINCT usuario FROM usuarios WHERE usuario = ?", [usuario])` y la segunda sentencia dividida en dos partes por la cantidad de variables

```
sql = "INSERT into usuarios (usuario, nombre, password, correo, telefono, date,  
link_user, rol) values (?, ?, ?, ?, ?, ?, ?, ?)"
```

```
cur.execute(sql,[usuario,nombre,clavehash,correo,telefono,date,link_user,rol])
```

El código del registro se ve a continuación:

```

page.py > signup
def signup():

    signup_form = Signup_Form()
    try:
        if request.method=="POST":
            if signup_form.is_submitted():
                if signup_form.nextbtn.data and signup_form.validate():
                    try:
                        usuario = signup_form.username.data
                        nombre = signup_form.name.data
                        password = signup_form.password.data
                        clavehash = generate_password_hash(password)
                        correo = signup_form.email.data
                        telefono = signup_form.telefono.data
                        date = signup_form.born.data
                        link_user = "link"
                        rol = "usuario"
                        con = sqlite3.connect('BaseDatos.db')
                        cur = con.cursor()
                        print("Connected to SQLite")
                        cur.execute("SELECT DISTINCT usuario FROM usuarios WHERE usuario = ?", [usuario])
                        records = cur.fetchall()
                        print("Total rows are: ", len(records))
                        print("Printing each row")
                        temp = ""
                        for row in records:
                            print("Usuario: ", row[0])
                            temp = row[0]

                        if temp == usuario:
                            flash("El usuario ya existe")
                            cur.close()
                        else:
                            print("Entro al else")
                            sql = "INSERT into usuarios (usuario,nombre,password,correo,telefono,date,link_user,rol) values (?, ?, ?, ?, ?, ?, ?, ?)"
                            cur.execute(sql,[usuario,nombre,clavehash,correo,telefono,date,link_user,rol])
                            con.commit()
                            session["usuario"] = usuario
                            return redirect('main')
                    except sqlite3.Error as error:
                        print("Failed to read data from table", error)

                finally:
                    if (con):
                        con.close()
                        print("The Sqlite connection is closed")

    return render_template('signup.html',signup_form=signup_form)
except:render template('signup.html',signup form=signup form)

```

Figura 21. Código Python3 de la validación de usuario y adición a la base de datos

Finalmente, se utilizaron las sentencias de lectura para recuperar los valores de la base de datos y organizarlos en filas. Y la sentencia de eliminación mediante la llave primaria de **id** de la tabla usuarios. Se muestran a continuación, en la vista view se ejecuta la función lectura y en delete la función de eliminación del registro.

```
@app.route('/view')
def view():
    if "admin" in session:
        con = sqlite3.connect("BaseDatos.db")
        con.row_factory = sqlite3.Row
        cur = con.cursor()
        cur.execute("SELECT * FROM usuarios")
        rows = cur.fetchall()
        con.close()
        return render_template("view.html", rows=rows)
    else:
        return redirect('login')

@app.route("/delete", methods=["GET","POST"])
def delete():
    if "admin" in session:
        delete_form= Delete_Form()
        if request.method=="POST":
            if delete_form.is_submitted():
                print("submitted")
                if delete_form.deletebtn.data and delete_form.validate():
                    userid = delete_form.userid.data
                    print(userid)
                    con = sqlite3.connect("BaseDatos.db")
                    try:
                        cur=con.cursor()
                        cur.execute("DELETE FROM usuarios where id = ?", [userid])
                        con.commit()
                        msg = "Registro eliminado exitosamente"
                    except:
                        con.rollback()
                        msg = "No se pudo eliminar el registro"
                    finally:
                        con.close()
                        return redirect('view')

                return render_template('delete.html',delete_form=delete_form)

    else:
        return redirect('login')
```

Figura 22. Código Python3 de lectura y de eliminación de registros

13. Diseño e implementación de portal de acceso usando método de autenticación basado en usuario y contraseña.

13.1. Creación de Sesiones Usuario y Admin:

Para la creación de las sesiones en Photograph, toma un papel importante el tema de los privilegios/permisos de usuario en la plataforma, es decir, que puede y que no puede hacer un usuario dentro de ella.

Para ello, se hizo la creación de dos tipos de usuario: El admin quien es el posee los permisos de administración totales de la aplicación; y el usuario quien es la persona que va a realizar uso de ella.

En la Figura 20, se muestra el código que redirecciona al Admin y al usuario en sus respectivas páginas. Esto se hace para que haya una jerarquía o control de la App y que se evite cualquier intrusión o malversación de la información que se trate.

Por ello, cada ruta fue modificada para que cada usuario tenga su sesión específica y no pueda invadir la de otro. Ni entre usuarios con roles de usuario ni usuarios con roles de administradores. En el siguiente código se muestra un ejemplo de la adición de las sesiones a las rutas:

```
@app.route('/search')
def search():
    if "usuario" in session:
        return render_template('search.html')
    else:
        return redirect('login')

@app.route('/edit')
def edit():
    if "usuario" in session:
        return render_template('edit.html')
    else:
        return redirect('login')

@app.route('/admin')
def admin():
    if "admin" in session:
        return render_template('admin.html')
    else:
        return redirect('login')
```

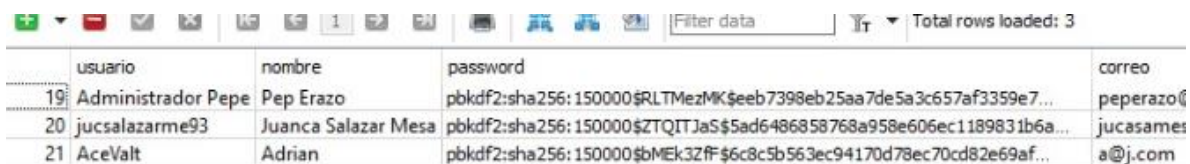
Figura 23. Código en Python3 de la creación de sesiones

13.2. Uso de funciones hash criptográficas para almacenar contraseñas:

Utilizando la librería **wekzeug.security** se importan las funciones para generar y verificar las contraseñas. **generate_password_hash** recibe la contraseña a encriptar y la encripta utilizando el modelo SHA256 (encripta según la hora y caracteres añadidos) y **check_password_hash** recibe dos parámetros, la contraseña encriptada y la contraseña escrita por el usuario en la ventana de inicio de sesión. Esta última devuelve un booleano que indica si las contraseñas coinciden o no. En la Figura 20, se muestra como se recupera mediante la sentencia la contraseña encriptada y luego se verifica si corresponde a la ingresada para así otorgar el acceso al usuario.

Se encripta de tal manera de que se reduzcan al mínimo los intentos de hackeo o robo de identidad y como consecuencia genere estragos e inconvenientes en la aplicación.

Posteriormente se modificó la función signup o registro, mostrada en la Figura 21 en donde se añade a la base de datos la contraseña digitada por el usuario de forma encriptada, a continuación, se muestra un ejemplo:



	usuario	nombre	password	correo
19	Administrador Pepe	Pep Erazo	pbkdf2:sha256:150000\$RLTMezMK\$eeb7398eb25aa7de5a3c657af3359e7...	peperazo@
20	jucsalazarme93	Juanca Salazar Mesa	pbkdf2:sha256:150000\$ZTQITJaS\$5ad6486858768a958e606ec1189831b6a...	jucasame
21	AceValt	Adrian	pbkdf2:sha256:150000\$bMEk3ZfF\$6c8c5b563ec94170d78ec70cd82e69af...	a@j.com

La contraseña se encripta mediante la sintaxis -->
method/sha256/salt/secuencia alfanumérica.

Se utilizó la función de derivación PBKDF2 debido a que incorpora los salts y el estándar sha256 que a diferencia de otros estándares, es una de las más seguras y utilizadas a nivel global.

Figura 24. Encriptación SHA256

14. Implementación de certificado SSL

Para la aplicación web se utilizó un criptosistema de llave única o método simétrico en el cual los procesos de cifrado y descifrado se realizan con base en una única

llave. En nuestro caso se utilizó la llave llaveprivada.pem y el micertificado.pem.

Estas llaves fueron creadas utilizando Open SSL y ejecutando el siguiente código en la carpeta del proyecto:

```
C:\FlaskCiclo3\DesarrolloWeb-master (master -> origin)
λ openssl req -x509 -newkey rsa:4096 -out micertificado.pem -keyout llaveprivada.pem -days 365
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'llaveprivada.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CO
State or Province Name (full name) [Some-State]:Cundinamarca
Locality Name (eg, city) []:Bogota
Organization Name (eg, company) [Internet Widgits Pty Ltd]:uninorte
Organizational Unit Name (eg, section) []:UN
Common Name (e.g. server FQDN or YOUR name) []:UniNorte
Email Address []:jucasamesa2@gmail.com
```

Figura 25. Creación de llave y certificado

De esta forma, y utilizando el código de ejecución a continuación:

```
if __name__=="__main__":
```

```
app.run(host='127.0.0.1', port=443, ssl_context = ('micertificado.pem',
'llaveprivada.pem'))
```

Se obtiene certifica la conexión segura:

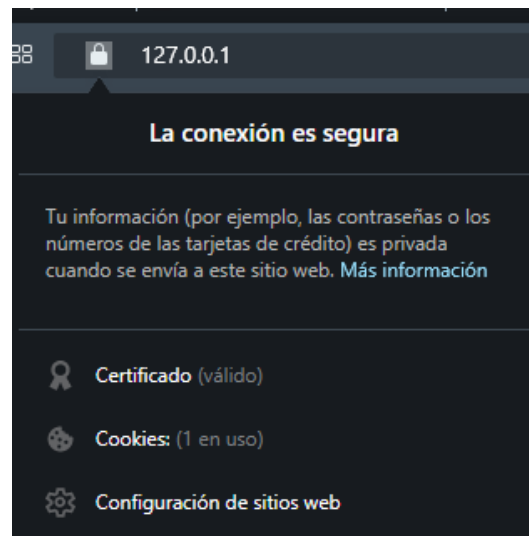


Figura 26. Conexión segura