

## LEVANTAMIENTO DE REQUERIMIENTOS

Roles del equipo:

- Mario Andres Gomez      Product owner
- María Camila Muñoz      **Developer**
- Edwar Sierra Sáenz      **Developer**
- Nicolas Hernandez      **Developer**
- Jose David cantor      Scrum Master

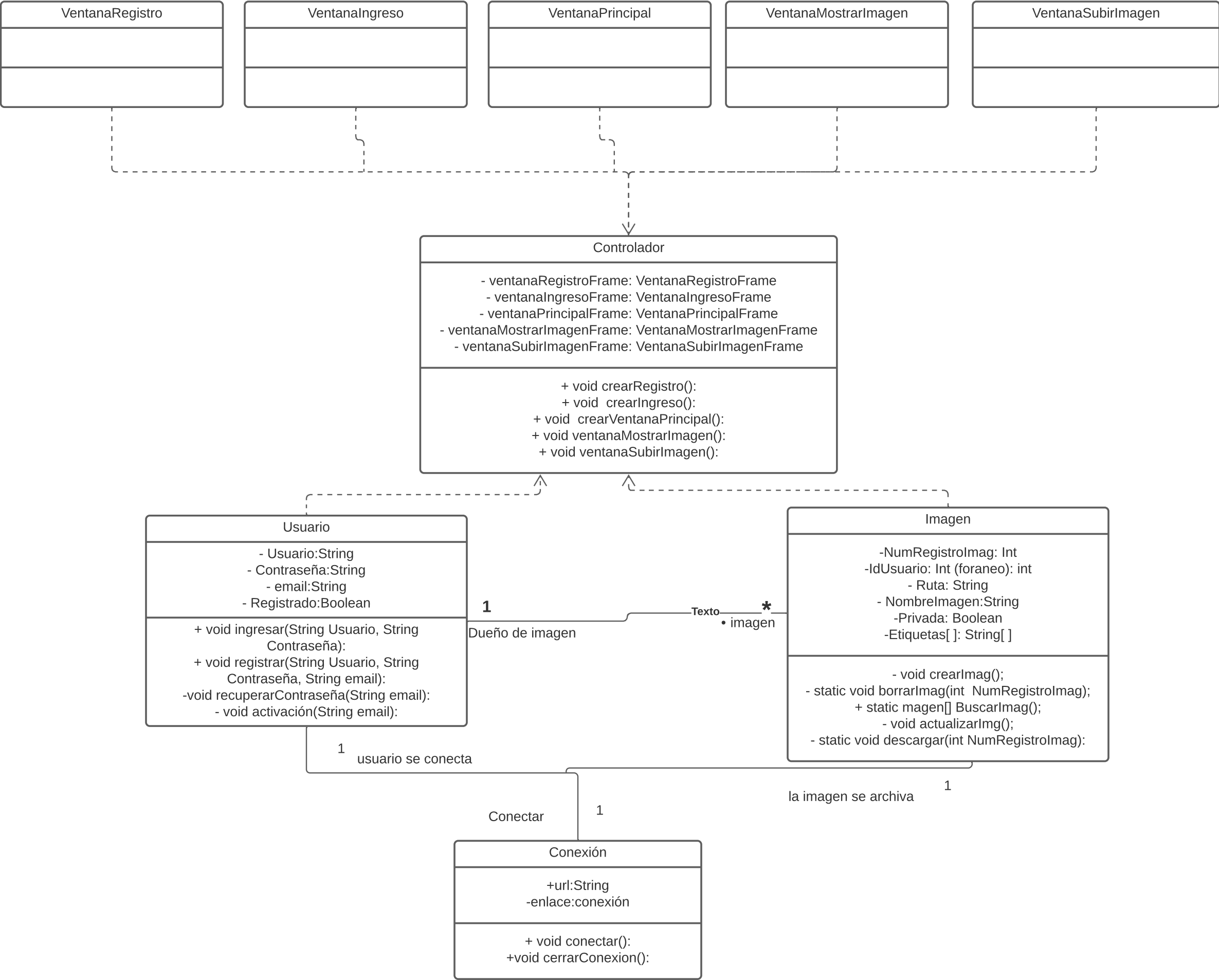
Requerimientos Iniciales (Backlog del producto):

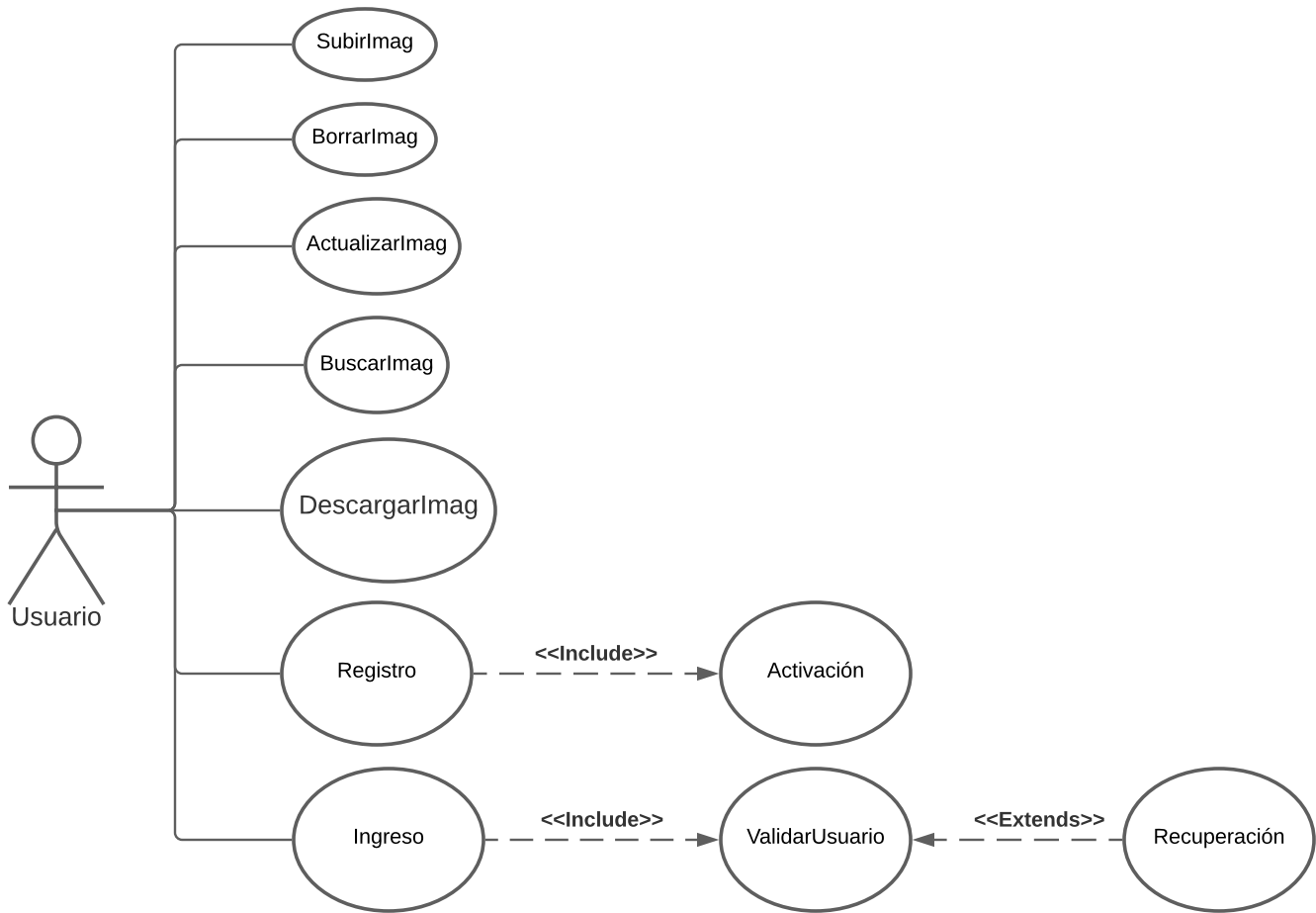
- **Aplicación de gestión de imágenes online (Función Principal).**
- **Roles: Usuario (1: importancia baja -- 5: importancia alta)**
  - **(5)Registro de Usuario:** El usuario podrá registrarse con un usuario, contraseña y email. La página web enviará un link de activación al correo electrónico, si el usuario se registra.
  - **(5)Ingresar:** El usuario podrá ingresar con el usuario y contraseña registrados. El servidor validará estos datos (Se debe cumplir con requerimientos de seguridad). Si el usuario olvida la contraseña tendrá la opción **Recuperar contraseña**.
  - **(4)Subir Imágenes:** Solo el usuario autenticado podrán subir imágenes a la plataforma. Deberá definir si es pública o privada la imagen, asignar un nombre y asignar mínimo una etiqueta o palabra clave que caracteriza la imagen.
  - **(3)Actualizar Imágenes:** Solo el usuario autenticado podrá actualizar sus imágenes a la plataforma, es decir, subir o cambiar otra imagen con el mismo nombre. Deberá definir si es pública o privada la imagen, y actualizar la etiqueta de ser necesario.
  - **(2)Eliminar Imágenes:** Solo el usuario autenticado podrán borrar sus imágenes en la plataforma. Se eliminará la imagen de su perfil.
  - **(3)Buscar imágenes:** El usuario normal y el usuario autenticado podrán buscar imágenes en la plataforma si ingresa una etiqueta. De lo cual, se desplegará una galería de imágenes según la etiqueta dada (palabra clave).
  - **(1)Descargar imágenes:** El usuario autenticado podrá descargar imágenes de la plataforma, si la imagen es pública.
- **(4)Clase Usuario** □ Atributos: Usuario, contraseña, email, boolean registrado. Métodos: RegistroUsuario, Registrar y Activación, Ingresar, Auténtica y RecuperarContraseña.
- **(5)Clase Imágenes** □ NumeroRegistroImagen , IdUsuario, Ruta, NombrelImagen, BOOL Privada, Etiquetas. Métodos: Crear, Borrar, Actualizar, Descargar.

Backlog del Sprint Inicial (Tareas Iniciales)

- Levantamiento de requerimientos (este documento)

- Diagrama Casos de uso y Diagrama de Clases.
- Estudio de Mercado





## ESTUDIO DE MERCADEO

Roles del equipo:

- Mario Andres Gomez      Product owner
- María Camila Muñoz      **Developer**
- Edwar Sierra Sáenz      **Developer**
- Nicolas Hernandez      **Developer**
- Jose David cantor      Scrum Master

1. <https://co.pinterest.com/>

- Sistemas de recomendaciones
- Categorías favoritas o de interés
- Etiquetas de imágenes (Checklist)
- Perfil de creador de la imagen

2. <https://unsplash.com/>

- Menú de categorías
- Navegación intuitiva
- Botón de descarga en cada imagen
- Compartir imagen en redes sociales

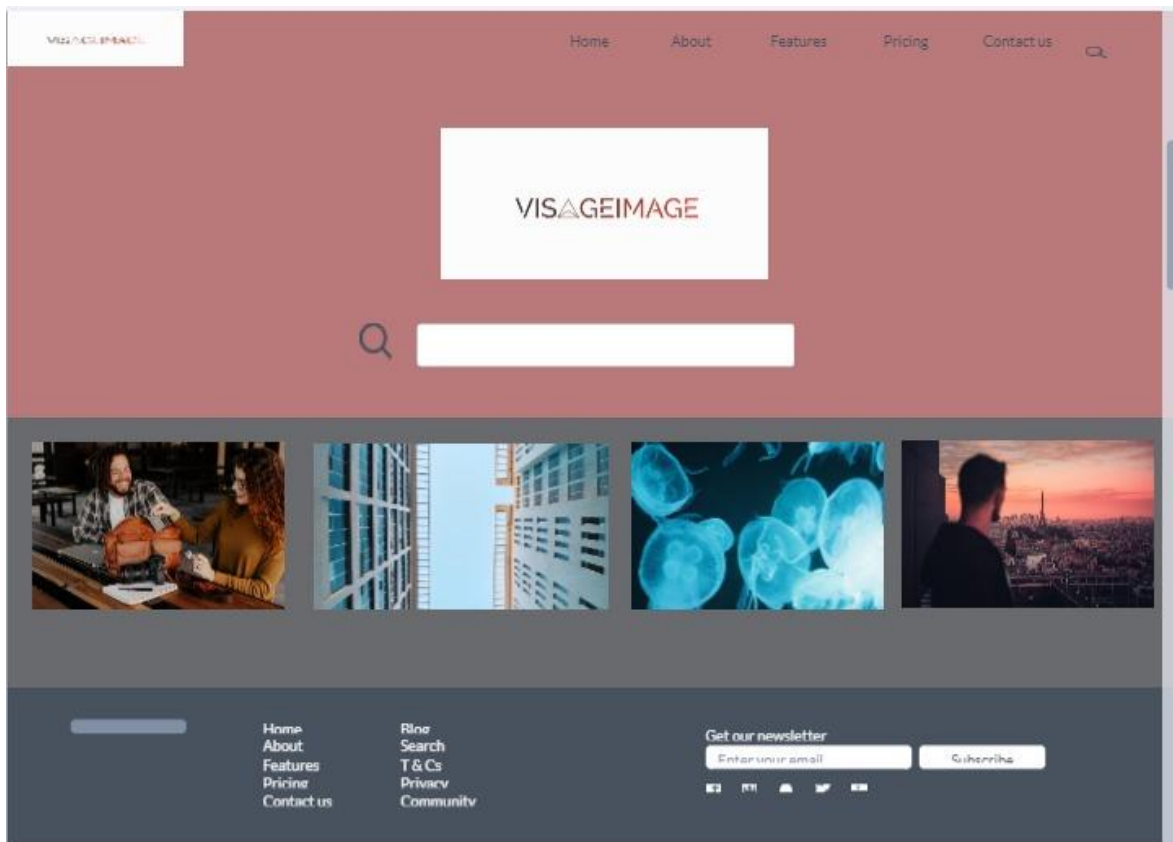
3. <https://www.pexels.com>

- Manejo de extensiones para editar la imagen
- Inicio con redes sociales
- Dar “gracias” al creador de la imagen (donar, dar gracias por redes sociales)

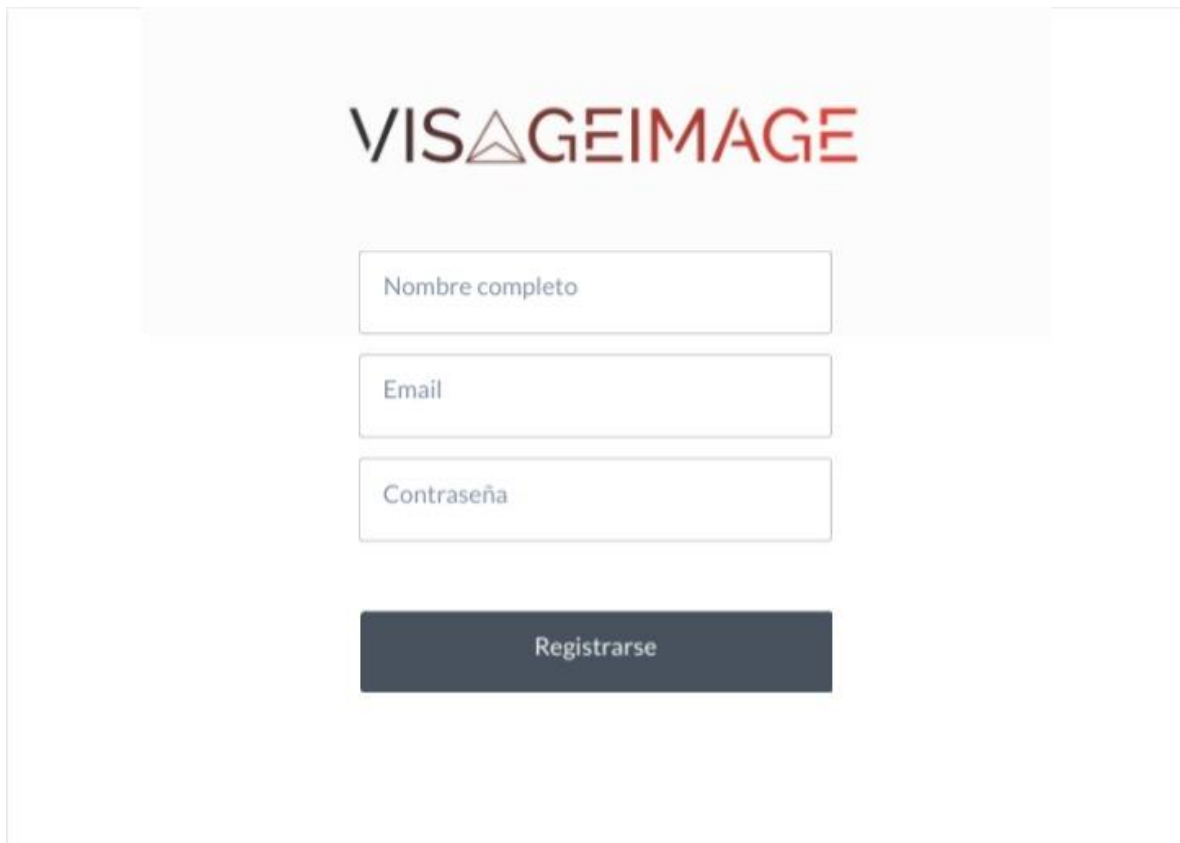
## URL DE LAS VENTANAS:

- **Mario Gómez**. URL ventana principal: src\vista\Html\index.html
- **Nicolas Hernandez**:  
URL ventana registro: src\vista\Html\registro.html  
URL ventana ingreso: src\vista\Html\ingreso.html  
URL ventana cambiar contraseña: src\vista\Html\cambiaPass.html  
URL ventana recuperar contraseña: src\vista\Html\olvidaPass.html
- **Edwar Sierra**. URL ventana subir imagen: src\vista\Html\SubirImagen.html
- **Maria Camila Muñoz**. URL ventana ver imagen: src\vista\Html\verImagen.html
- **Jose Cantor**. URL ventana modificar imagen: src\vista\Html\vistaModificar.html
- **Benjamin de Castro**. URL ventana perfil usuario: src\vista\Html\perfil.html

# DISEÑO DE LAS VENTANAS



La “ventana principal” o “home” está compuesta por un header donde estarán localizados los diferentes botones de navegación (barra de navegación) y un botón de login para acceder. Siguiendo tenemos un contenedor o div donde tendremos el buscador de nuestra página web, el usuario pondrá una palabra clave y todas las imágenes relacionadas se mostrarán. En la siguiente sección se mostrarán las últimas imágenes subidas a la plataforma por los creadores. Y finalmente tendremos un footer con las cuentas relacionadas al proyecto, los derechos de la página y la opción de suscribirse para información relevante.



VISAGEIMAGE

Nombre completo

Email

Contraseña

Registrarse

**Ventana “registro”** en la cual el usuario a través de datos básicos tales como nombre completo, correo electrónico y una contraseña podrá crear su cuenta en nuestra plataforma, de esta manera accederá a las funciones de buscar, subir, modificar y descargar imágenes

Además, se crearon los mockup de las ventanas **“ingreso”**, **“recuperar contraseña”**, **“cambiar contraseña”**, **“términos y condiciones”**.



## Nueva imagen

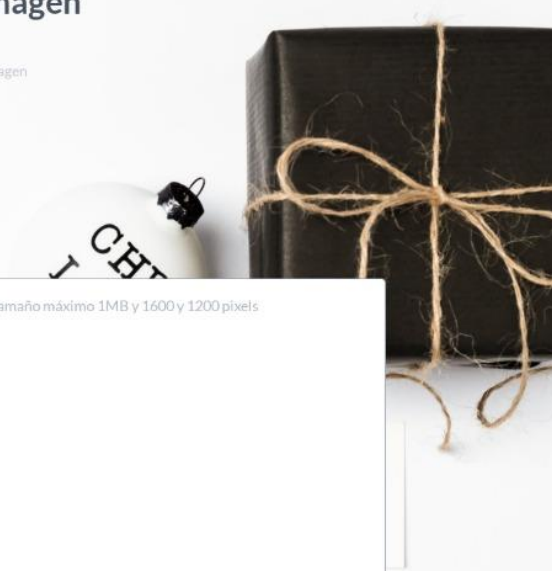
Sube la nueva imagen

**Asigna un nombre\***

**Asigna una etiqueta\***

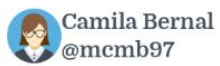
Elegir un archivo o arrastralo hasta aquí. Puede subir archivo JPG, gif, png. Tamaño máximo 1MB y 1600 y 1200 pixels

CancelarGuardar



**La ventana “subir imagen”** tendrá dos "campo de entrada" donde se podrá asignar un título a la imagen a subir y las etiquetas que categorizaran a dicha imagen. Además, un campo de entrada de imágenes donde el usuario podrá dar clic y buscar el recurso dentro de su dispositivo. Además de un botón que permite guardar la imagen y otro botón que permite cancelar la imagen seleccionada en caso de que el usuario quiera subir otra.

En el mockup se adicionó dos radioButton donde el usuario podrá definir si la imagen es pública o privada.



(Título de la imagen)

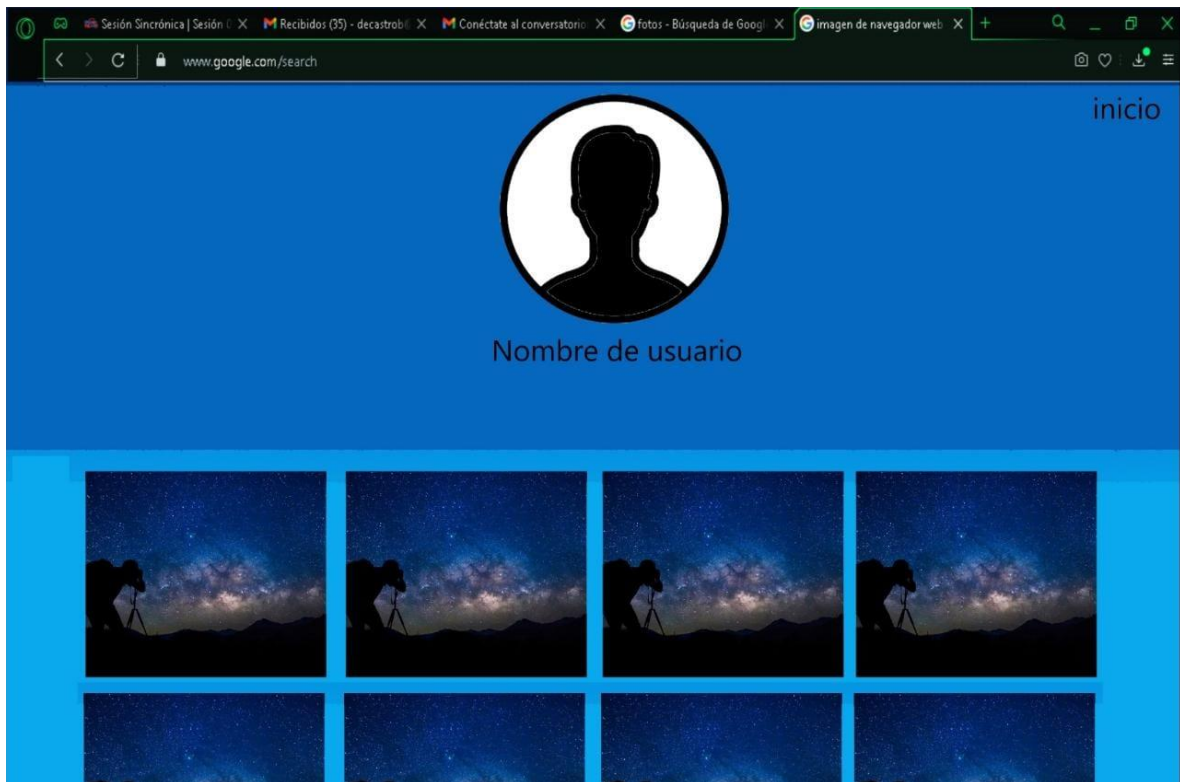
**Sunset in Cappadocia**

(Etiquetas)

#holidays #dreams #travel



**En la ventana “ver imagen”** se podrá ver con más detalle la fotografía que seleccione el usuario de la galería principal. En la parte superior izquierda se podrá ver el nombre del dueño de la imagen y una arroba que lo identifica como usuario. También se podrá visualizar el nombre de la imagen, y las etiquetas que el creador de la imagen definió. Igualmente se podrá descargar en el equipo.



**En la ventana “perfil usuario”** el usuario registrado podrá ver las imágenes que haya creado. Además, podrá acceder a cada imagen para modificarla o eliminarla.



Seleccionar archivo

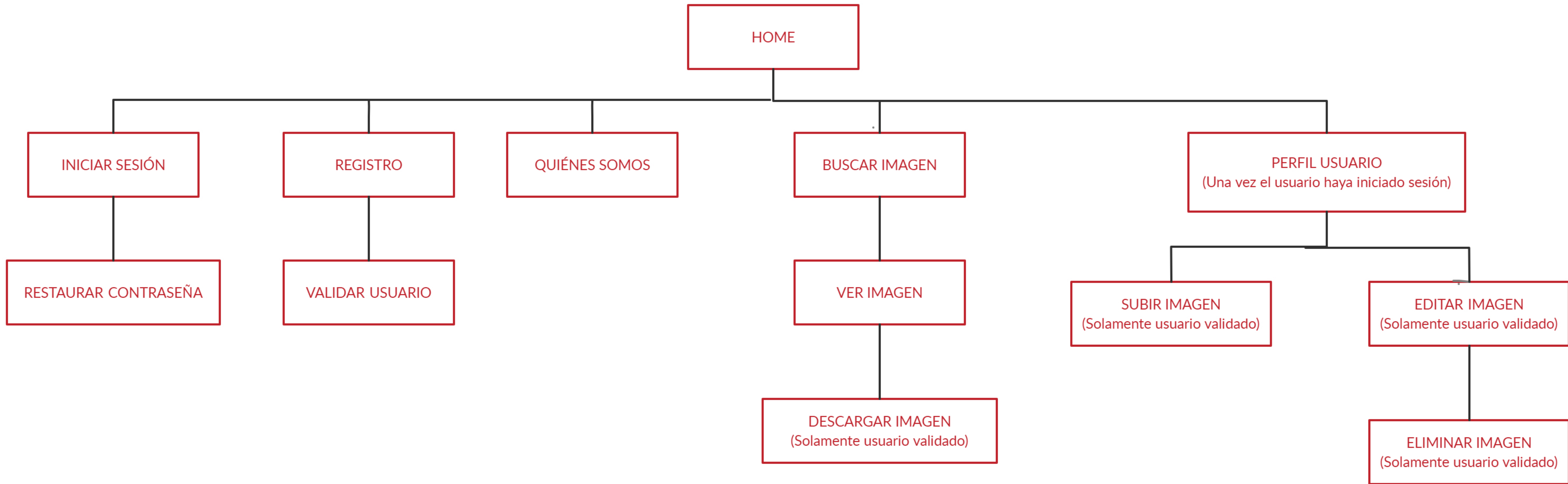
Ningún archivo seleccionado

ACTUALIZAR

ELIMINAR

El usuario podrá realizar modificaciones a la imagen creada, nombre y etiquetas de las diferentes imágenes que haya creado.

La vista contiene un cuadro de texto para el nombre, un cuadro de texto para la etiqueta, dos radioButton para indicar si la imagen es pública o privada (se adicionó en el mockup), un botón de seleccionar archivo que permite cargar el archivo de la imagen, un botón para actualizar que permite guardar los cambios hechos a la imagen, un botón de eliminar, y por último un espacio donde se visualiza la imagen cargada.



Nombre de la funcion	Ruta	metodo (GET / POST)
login()	ruta: "/login/"   html: ingreso.html	POST
perfil()	ruta: "/perfil"   html: perfil.html	POST
index()	ruta: "/"   html: index.html	POST
devolverImgRecientes()	NA	POST
terminos()	ruta: "/terminos"   html: termsCond.html	POST
nosotros()	ruta: "/nosotros"   html: Nosotros.html	POST
buscarImgs()	NA	POST
verImagen()	ruta: "/verImagen"   html: verImagen.html	POST
registro()	ruta: "/registro"   html: registro.html	
perfil()	ruta: "/perfil"   html: perfil.html	
DESDE AQUI TU LO HACES		
namePerfil()	NA	
llenarContainerPerfil()	NA	
iniciarSesion()	NA	POST
registro()	ruta: "/registro"   html: registro.html	GET
crearCuenta()	NA	POST
olvidaPass()	ruta: "/olvidaPass"   html: olvidaPass.html	GET
olvidaPass_procesar()	NA	POST
cambiaPass()	NA	POST
actualizarImagen()	NA	POST
eliminarImagen()	NA	POST
mostrarImagen()	NA	GET
datosverImagen()	NA	GET

---

## Visagelimage funciones de bac

parametros
------------

null

String usuario

null

null

null

null

tag

long id\_img

String img\_perfil

String name\_us

String nombre\_usuario, String contrasena

null

String nombre, String usuario, String email, String contrasena,

Bool terminos

null

String email

string nuevo\_pass

ombrelmg String - Etiquetalmg Booleano - Publica Booleano - Priv

String Nombrelmg String - Etiquetalmg

long idlmg

long idlmg

---

---

## kend 3 Spring

breve descripcion de la funcion
---------------------------------

Funcion para redireccionar a la pagina login, donde el usuario ingresara sus datos de ingreso nar a la pagina de perfil por el medio del usuario (String unico), donde se podra vizualizar las imagenes que ha moi
---

Funcion para regresar a la pagina principal haciendo click en el icon/image VisageImage

Funcion que retorna las 9 imagenes mas recientes agregadas al sitio para desplegarlas en la pagina principal.

Funcion para redireccionar a la pagina de tratamiento de datos y confidencialidad

Funcion para redireccionar a la pagina de about del equipo de desarrollo

Funcion para recuperar las imagenes que contengan la etiqueta (tag) y desplegarlas en la pagina principal.

redireccionar a la pagina de ver imagen, es importante enviarle el parametro img\_id para poder desplegar la imagen

funcion para tomar la imagen que elija el usuario y ponerla como su foto de perfil

funcion que toma el nombre de usuario y lo plasma bajo la imagen de perfil

lee las imagenes agregadas y las muestra en el container de la parte inferior de la vista de perfil

Se valida si el usuario existe en la base de datos a través de su usuario y contraseña para así poder ingresar a su  
perfil y a todas las funciones de la plataforma

Función que redirecciona a la vista registro.html para que el usuario pueda crear su cuenta

El usuario crea una nueva cuenta en la plataforma para así tener todas las funcionalidades disponibles

Función que redirecciona a la vista olvidaPass.html para que el usuario pueda recuperar su contraseña  
El usuario provee su correo electrónico al cual será enviado un link en el que se le darán instrucciones para  
recuperar su contraseña

Se reestablece la contraseña del usuario después de éste haberla perdido u olvidado

Actualiza datos de la imagen y copia la imagen en la ruta por defecto

Elimina Imagen seleccionada

Carga la imagen seleccionada en vista preliminar

La función retorna en la ventana "ver imagen" el nombre, la(s) etiqueta(s) de la imagen, el nombre y perfil del usuario

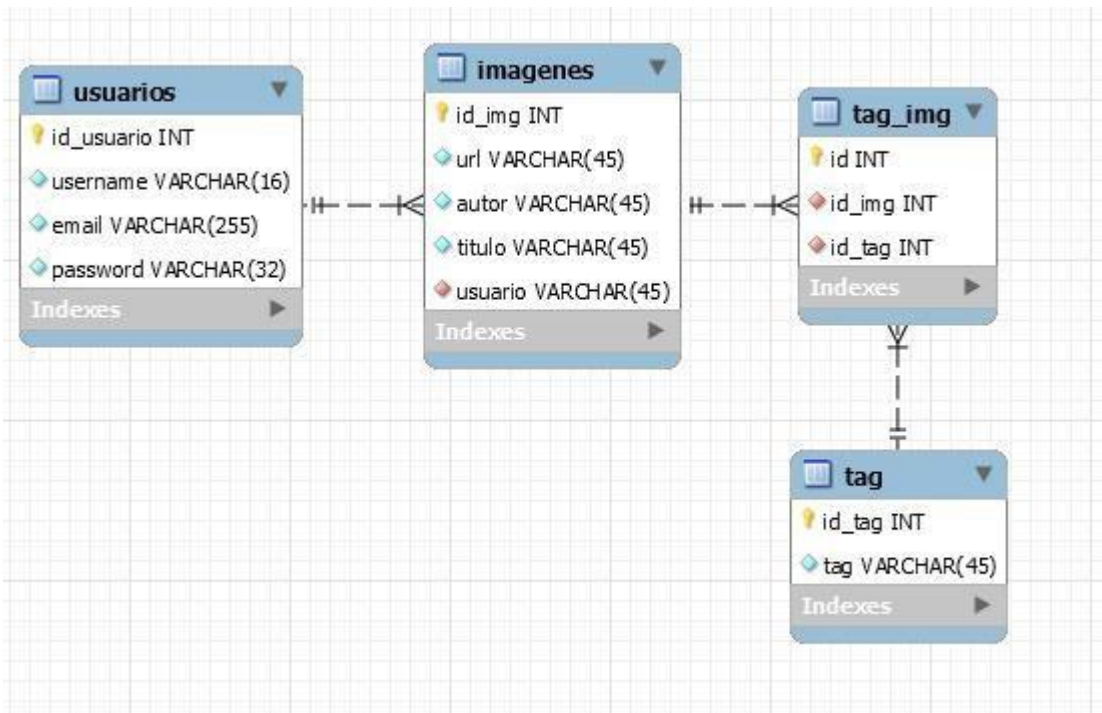
---



Vista o mock up que retonar	Responsable
mock up Login	Mario G
mock up Perfil	Mario G
mock up Index/Principal	Mario G
null	Mario G
mock up Tratamiento de datos	Mario G
mock up About	Mario G
null	Mario G
mock up Ver Imagen	Mario G
mock up perfil	benjamin DC
mock up perfil	benjamin DC
mock up perfil	benjamin DC
Vista perfil	Nicolás H
Vista registro	Nicolás H
Vista perfil	Nicolás H
Vista olvidaPass	Nicolás H
ventana emergente	Nicolás H
ventana emergente + vista	Nicolás H
ingreso	
Null	José Cantor
Null	José Cantor
Null	José Cantor y Camila M.
Null	Camila M.

## Documento sprint N. 4:

### Modelo relacional:



### Tablas que hacen parte de la base de datos:

Las tablas que hacen parte de la base de datos, al igual que se observa en el modelo relacional, son las siguientes:

- **usuarios:** Cuenta con cinco atributos, todos de tipo “not null”. Cuatro de ellos son de tipo “varchar”, el atributo “id\_usuario” es el único de tipo “int” además de ser de tipo “primary key”. Los atributos “id\_usuario”, “username” y “email” son de tipo “unique”.

Esta tabla nos permitirá almacenar la información del usuario al momento de registrarse en la página, para poder comprobar después si es un usuario ya registrado o no, además de poderle suministrar la contraseña al usuario que así lo solicite por medio del email.

- **imágenes:** Cuenta con cinco atributos, todos de tipo “not null”. Cuatro de ellos son de tipo “varchar”, el atributo “pk\_id\_img” es el único de tipo “int” además de ser de tipo “primary key”, el atributo “fk\_usuario” es de tipo “foreign key”. Los atributos “pk\_id\_img” y “url” son de tipo “unique”.

Esta tabla nos permitirá guardar las imágenes y la información de la misma en el momento en que el usuario desee subirla a la página.

- **tag\_img:** Cuenta con tres atributos, todos de tipo “int” y de tipo “not null”. El atributo “pk\_id\_tag\_img” es de tipo “primary key” y de tipo “unique”. Los atributos “fk\_id\_img” y “fk\_id\_tag” son de tipo “foreign key”.

Esta tabla nos permitirá guardar las etiquetas que el usuario desee ponerle a su imagen en el momento que la suba a la página, mínimo debe ser una.

- **tag:** Cuenta con dos atributos, todos de tipo “not null”. El atributo “id\_tag” es de tipo “int”, de tipo “primary key” y también de tipo “unique”. El atributo “tag” es de tipo “varchar” y también de tipo “unique”.

Esta tabla nos permitirá relacionar las etiquetas guardadas en la tabla “tag” con el id único de la imagen relacionada.

### Conceptos de programación segura:

- Los queries en sql se realizaron usando “prepared statement”.
- Las contraseñas se han encriptado en la base de datos.
- Credenciales para probar login: mariotest - test12345 o mariogomez96 - test4321.
- Se utilizó la función “secure\_filename()” para evitar que el usuario suba archivos basura.

```
app.py X
visageimage > app.py > ...
70 @app.route('/login/', methods=['GET', 'POST'])
71 def login():
72     form = LoginForm()
73     if request.method == 'POST':
74         db = get_db()
75         form = LoginForm() # Se instancia el objeto form --> WIForm
76         if form.validate_on_submit(): # Se valida si el usuario envia un metodo POST con los datos validos
77             usuario = form.usuario.data # Con estos datos se valida en la BD si esta registrado
78             contrasena = form.contrasena.data
79             user = db.execute('SELECT * FROM usuarios WHERE username = ?', (usuario,))
80             user.fetchone()
81             if user is None: #Si el no se encuentra en la base de datos
82                 error = 'Usuario o contrasena invalidos'
83                 flash(error)
84                 return render_template("ingreso.html", form=form)
85                 #Si se encuentra, se crea la session y se devuelve a la pagina principal
86
87             if check_password_hash(user['password'], contrasena):
88                 session.clear()
89                 session['user_id'] = user[0]
90                 return redirect(url_for('index'))
91             else:
92                 error = 'Usuario o contrasena invalidos'
93                 flash(error)
94                 return render_template("ingreso.html", form=form)
95         else:
96             return render_template("ingreso.html", form=form)
97
98     if request.method == 'GET':
99         return render_template("ingreso.html", form=form) # Solo es ejecuta en el metodo GET
100
```

```

app.py x
visageimage > app.py > ...
107 @app.route('/registro', methods=['GET', 'POST'])
108 def registro():
109     form = RegistroForm()
110     if request.method == 'POST':
111         form = RegistroForm()
112         if form.validate_on_submit():
113             db = get_db()
114             nombre = form.nombre.data
115             usuario = form.usuario.data
116             email = form.email.data
117             contrasena = generate_password_hash(form.contrasena.data)
118
119             #Si el usuario ya existe
120             if db.execute( 'SELECT id_usuario FROM usuarios WHERE username = ?', (usuario,) ).fetchone() is not None:
121                 error = 'El usuario {} ya existe'.format( usuario )
122                 flash( error )
123                 return render_template( 'registro.html', form=form)
124
125             #Si el correo ya existe
126             if db.execute( 'SELECT id_usuario FROM usuarios WHERE email = ?', (email,) ).fetchone() is not None:
127                 error = 'El correo {} ya existe'.format( email )
128                 flash( error )
129                 return render_template( 'registro.html', form=form)
130
131             db.execute( #Ejecucion del query
132                 'INSERT INTO usuarios (username, email, password, name) VALUES (?, ?, ?, ?)',
133                 (usuario, email, contrasena, nombre)
134             )
135             db.commit() #Chequear informacion en la BD
136             flash( 'Revisa tu correo para activar tu cuenta' )
137             enviar_mensaje_activacion(email=email)
138             return redirect(url_for('index'))

```

```

142
143 @app.route('/olvidaPass', methods=['GET', 'POST'])
144 def olvidaPass():
145     form = OlvidaPassForm()
146     if form.validate_on_submit():
147         email = form.email.data
148
149         next = request.args.get('next', None)
150         if next:
151             return redirect(next)
152         return redirect(url_for('enviar_mensaje', email=email))
153     return render_template('olvidaPass.html', form=form)
154

```

```

168
169 @app.route('/cambiaPass', methods=['GET', 'POST'])
170 def cambiaPass():
171     form = CambiaPassForm()
172     if form.validate_on_submit():
173         db = get_db()
174         email = form.email.data
175         contrasena = form.contrasena.data
176         confirmar = form.confirmar.data
177
178         db.execute(
179             'UPDATE usuarios SET password = ? WHERE email = ?', (generate_password_hash(contrasena), email)
180         )
181         db.commit()
182
183         next = request.args.get('next', None)
184         if next:
185             return redirect(next)
186         return redirect(url_for('index'))
187     return render_template('cambiaPass.html', form=form)
188

```

```

209 def subirimagen():
210     form = subirimagenForm()
211     if request.method == 'POST':
212         if form.validate_on_submit():
213             usuario = g.user[1] #Nombre del usuario en la session
214             titulo_img = form.titulo_img.data
215             etiq_img = form.etiq_img.data
216             archivo = request.files['file']
217             db = get_db()
218             print('Ingreso al metodo subirimagen POST')
219
220             if archivo is None or archivo == '':
221                 flash('Ingresa un archivo con extension .jpg o .png')
222                 return render_template('subirimagen.html', form = form)
223
224             if archivo and allowed_file(archivo.filename): #Si existe el archivo y tiene extension permitida
225                 archivo_seguro = secure_filename(archivo.filename)
226                 path_archivo = os.path.join(app.config['UPLOAD_FOLDER'], archivo_seguro)
227
228                 #Si la imagen ya existe en la BD
229                 if db.execute( 'SELECT url FROM imagenes WHERE url = ?', (path_archivo,) ).fetchone() is not None:
230                     error = 'La imagen ya existe en la base de datos, pruebe con un nombre diferente'
231                     flash( error )
232                     return render_template( 'subirimagen.html', form=form)
233                 else:
234                     archivo.save(os.path.join(app.config['UPLOAD_FOLDER'], archivo.filename)) #Se graba en la carpeta
235                     db.execute( #Ejecucion del query
236                         'INSERT INTO imagenes (url, autor, titulo, fk_usuario) VALUES (?, ?, ?, ?)',
237                         (path_archivo, usuario, titulo_img, usuario) )
238                     db.commit() #Chequear informacion en la BD
239                     return redirect(url_for('perfil'))
240
241     return render_template('SubirImagen.html', form = form)

```

## **Documento sprint N. 5:**

Url : <https://54.91.130.114/>

Usuario: maca97

Contraseña: visajeimage97