

# **Documentación Proyecto**

## **Plataforma de Mensajería Electrónica**

**NRC: 2104**

**Equipo No. 5**

**Integrantes:**

*Anderson Yarit Marimon C.*

*Carlos Contreras Turca*

*Alba Arteaga Nuñez*

*Juan Bernal Clavijo*

*Dayana Dávila M.*

**UNIVERSIDAD DEL NORTE**  
*Ciclo 3 – Desarrollo de Software*  
*Octubre - 2022*

## Descripción de roles del equipo

Rol	Integrante	Descripción	Tareas
SCRUM Master	Leider Enrique Pineda	Persona encargada de llevar el seguimiento del proyecto.	Seguimiento del correcto desarrollo de las tareas del proyecto.
Desarrollador Front-End	Alba Arteaga / Carlos Contreras	Persona encargada de realizar la parte visual de la aplicación.	Desarrollo de las vistas y estilos de la aplicación.
Desarrollador Back-End	Dayana Dávila	Persona encargada de realizar la lógica de negocios de la aplicación.	Desarrollo de los controladores de la aplicación y diseño e integración de la base de datos.
Desarrollador Back-End	Juan Bernal	Persona encargada de realizar la lógica de negocios de la aplicación.	Desarrollo de los controladores de la aplicación y diseño e integración de la base de datos.
Desarrollador Full-Stack	Anderson Marimón	Persona encargada de realizar la parte visual y la lógica de negocios de la aplicación.	Desarrollo de las vistas y estilos de la aplicación, de los controladores de la aplicación y diseño e integración de la base de datos.

# Definición de artefactos

## Backlog Sprint 1

User story	Descripción	Estimación	Responsable
Definición de roles	Definición de los roles de los integrantes del equipo de trabajo del proyecto.	2 horas	Anderson Marimon C. Carlos Contreras Turca Alba Arteaga Nuñez Juan Bernal Clavijo Dayana Dávila M.
Definición de artefactos	Definición de los artefactos de la metodología SCRUM para el desarrollo del proyecto.	4 horas	Anderson Marimon C. Carlos Contreras Turca Alba Arteaga Nuñez Juan Bernal Clavijo Dayana Dávila M
Diseño del diagrama de clases	Diseño del diagrama de clases de la aplicación a desarrollar.	2 horas	Anderson Marimon C. Carlos Contreras Turca Alba Arteaga Nuñez Juan Bernal Clavijo Dayana Dávila M
Definición del cronograma	Definición del cronograma de actividades del proyecto.	2 horas	Anderson Marimon C. Carlos Contreras Turca Alba Arteaga Nuñez Juan Bernal Clavijo Dayana Dávila M

## Backlog Sprint 2

User story	Descripción	Estimación	Responsable
Definición del mapa de navegabilidad	Definición del mapa de navegabilidad que mostrará la distribución de las vistas de la aplicación	3 horas	Alba Arteaga / Carlos Contreras / Anderson Marimón
Selección de la plantilla de estilos	Selección de la librería CSS a usar para los estilos de las vistas de la aplicación.	1 hora	Alba Arteaga / Carlos Contreras / Anderson Marimón
Diseño e implementación	Diseño e implementación de las	15 horas	Alba Arteaga / Carlos Contreras / Anderson

de las vistas	vistas en HTML y CSS.		Marimón
Creación del proyecto en GIT	Creación del proyecto en GIT y posterior cargue a la nube en la plataforma Github.	1 hora	Alba Arteaga / Carlos Contreras

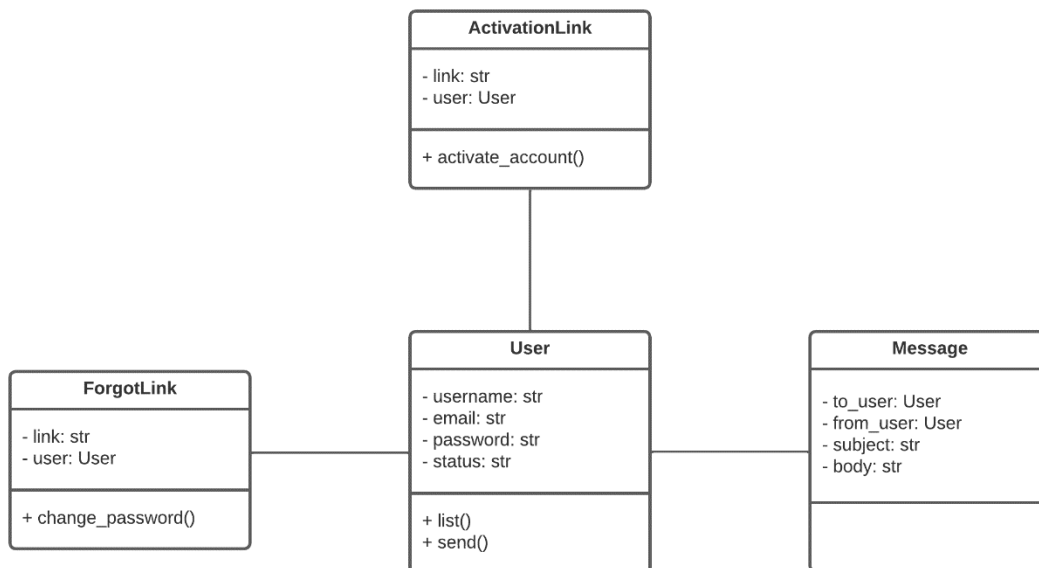
### Backlog Sprint 3

User story	Descripción	Estimación	Responsable
Diseño e implementación de los controladores para formularios y otras funcionalidades	Diseño e implementación de los controladores que representan la lógica de negocios de la aplicación.	40 horas	Dayana Dávila / Juan Bernal
Diseño e implementación de base de datos	Diseño de la base de datos relacional que alojará los datos de la aplicación e implementación de la misma usando el motor SQLite.	10 horas	Anderson Marimón
Desarrollo de integración de controladores y bases de datos	Integración de la base de datos con los controladores para la búsqueda y almacenamiento de información persistente de manera segura.	30 horas	Dayana Dávila / Juan Bernal
Diseño e implementac. de portal de acceso usando método de autenticación basado en usuario y contraseña	Diseño e implementación del método de autenticación de los usuarios y manejo de sesiones.	20 horas	Anderson Marimón

## Backlog Sprint 4

User story	Descripción	Estimación	Responsable
Definición de requerimientos para el despliegue de la aplicación	Definición de los requerimientos necesarios para realizar el despliegue de la aplicación en la plataforma PythonAnywhere	3 horas	Anderson Marimon C. Carlos Contreras T. Alba Arteaga Nuñez Juan Bernal Clavijo Dayana Dávila M
Despliegue de la aplicación	Configuración, despliegue y verificación del funcionamiento de la aplicación en la plataforma PythonAnywhere	12 horas	Anderson Marimon C. Carlos Contreras T. Alba Arteaga Nuñez Juan Bernal Clavijo Dayana Dávila M

## Diagrama de clases

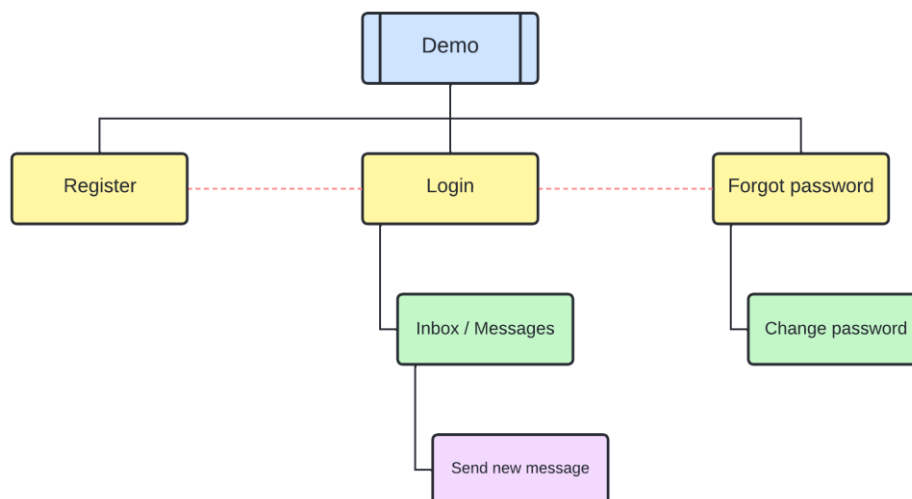


La clase usuario tendrá la información referente al usuario, es decir, su nombre de usuario, su correo, su contraseña y el estado de su cuenta. La clase mensaje tendrá la información de los mensajes de la plataforma, es decir, el usuario que envía el mensaje, el usuario que lo recibe, el asunto y el cuerpo del mensaje.

## Cronograma de tareas

[illegible]

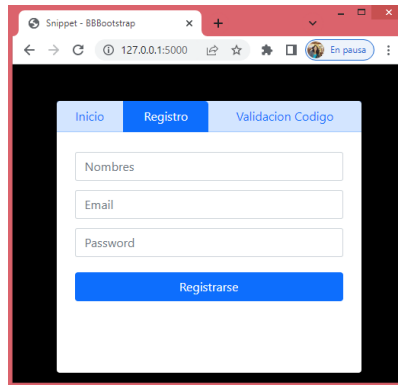
## Mapa de navegabilidad



La aplicación en su pantalla principal tiene 3 opciones, estas son el registro, el inicio de sesión y la recuperación de contraseña, y se puede desplazar entre ellas. La pantalla de recuperación de contraseña, dará paso a la pantalla de cambio de contraseña al acceder a ella mediante el enlace enviado. La pantalla de inicio de sesión da paso a la de visualización de mensajes y esta a su vez da paso a la pantalla de envío de nuevo mensaje.

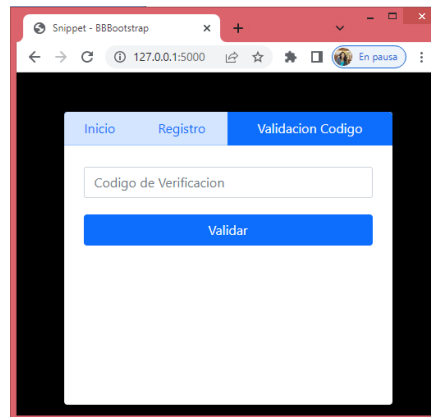
# Vistas de la aplicación

- Vista de Registro de Usuario:



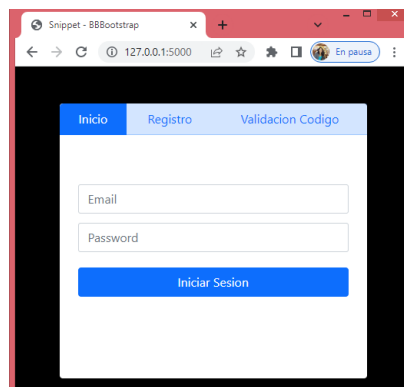
A screenshot of a web browser displaying a registration form. The browser's address bar shows '127.0.0.1:5000'. The page has a dark background with a white form area. At the top, there are three tabs: 'Inicio', 'Registro' (which is active), and 'ValidacionCodigo'. The form contains three input fields labeled 'Nombres', 'Email', and 'Password'. Below these fields is a blue button labeled 'Registrarse'.

- Vista de Validación de código:



A screenshot of a web browser displaying a code validation form. The browser's address bar shows '127.0.0.1:5000'. The page has a dark background with a white form area. At the top, there are three tabs: 'Inicio', 'Registro', and 'ValidacionCodigo' (which is active). The form contains a single input field labeled 'Codigo de Verificacion'. Below this field is a blue button labeled 'Validar'.

- Vista de inicio de sesión:



A screenshot of a web browser displaying a login form. The browser's address bar shows '127.0.0.1:5000'. The page has a dark background with a white form area. At the top, there are three tabs: 'Inicio' (which is active), 'Registro', and 'ValidacionCodigo'. The form contains two input fields labeled 'Email' and 'Password'. Below these fields is a blue button labeled 'Iniciar Sesion'.

- Vista una vez se ingresa a la plataforma:

A screenshot of a web browser showing the 'Center MAIL -' registration form. The browser's address bar displays '127.0.0.1:5000/verificarUsuario'. The page has a teal header with the title 'Center MAIL -'. Below the header, there is a navigation bar with four buttons: 'Regresar' (red), 'Gestion de Envio' (blue), 'Historial' (grey), and 'Perfil' (grey). The main form area contains three input fields: 'Asunto:' (text), 'Seleccione un destinatario...' (dropdown), and 'Mensaje:' (text area). At the bottom of the form is a green 'Enviar' button.

- Vista de la pestaña Historial:

A screenshot of the 'Center MAIL -' application with the 'Historial' tab selected. The browser's address bar shows '127.0.0.1:5000/verificarUsuario'. The navigation bar now highlights 'Historial' in blue, with a red arrow pointing to it. The main content area is divided into two sections: 'Correos Enviados' and 'Correos Recibidos'. Each section contains a table with four columns: 'Asunto', 'Mensaje', 'Fecha / Hora', and 'Destinatario'. The 'Correos Enviados' table is currently empty. The 'Correos Recibidos' table has one row with a grey background, also appearing empty.

- Vista de solicitud de recuperación de contraseña a través de la pestaña Perfil:

A screenshot of the 'Center MAIL -' application with the 'Perfil' tab selected. The browser's address bar shows '127.0.0.1:5000/verif...'. The navigation bar highlights 'Perfil' in blue, with a red arrow pointing to it. The main form area contains two input fields: 'Nueva contraseña' and 'Confirmacion Nueva contraseña'. At the bottom of the form is a yellow 'Actualizar' button.

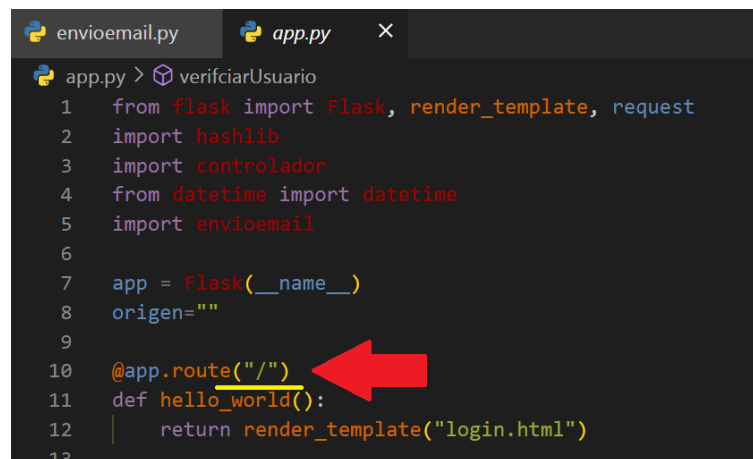


## DESCRIPCIÓN DEL SPRINT # 3 DE ACUERDO A REQUERIMIENTOS

### 1. Diseño e implementación de los controladores para formularios y otras funcionalidades

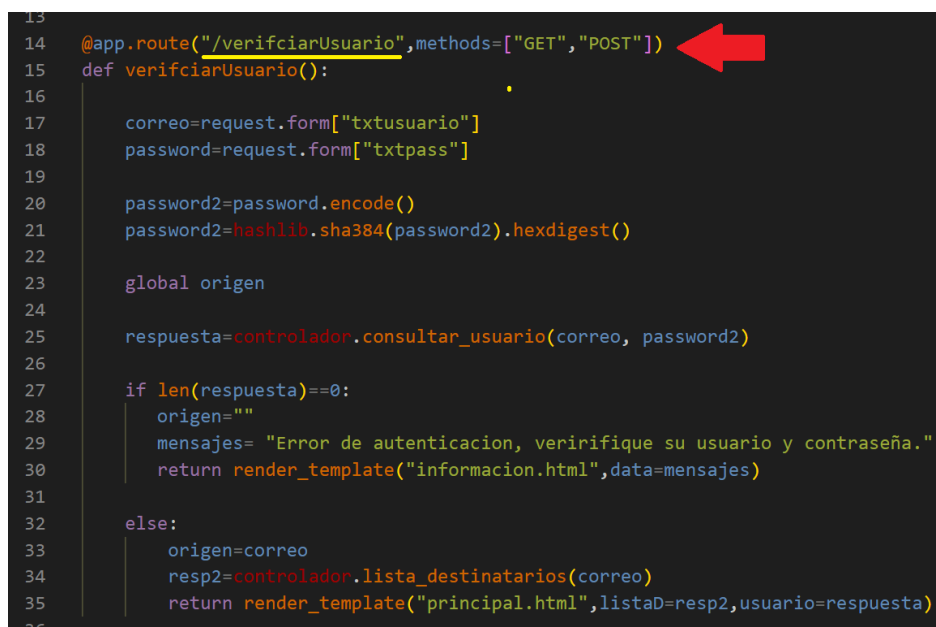
#### 1.1 Especificación de ruta

- Ruta: raíz. Carga una plantilla llamada **login.html**



```
envioemail.py  app.py  X
app.py > verficarUsuario
1  from flask import Flask, render_template, request
2  import hashlib
3  import controlador
4  from datetime import datetime
5  import envioemail
6
7  app = Flask(__name__)
8  origen=""
9
10 @app.route("/") ←
11 def hello_world():
12     return render_template("login.html")
13
```

- Ruta: verificarUsuario. Consulta si el usuario existe y de ser así direcciona hacia una plantilla llamada **principal.html**



```
13
14 @app.route("/verificarUsuario",methods=["GET","POST"]) ←
15 def verficarUsuario():
16     .
17     correo=request.form["txtusuario"]
18     password=request.form["txtpass"]
19
20     password2=password.encode()
21     password2=hashlib.sha384(password2).hexdigest()
22
23     global origen
24
25     respuesta=controlador.consultar_usuario(correo, password2)
26
27     if len(respuesta)==0:
28         origen=""
29         mensajes= "Error de autentificacion, verifique su usuario y contraseña."
30         return render_template("informacion.html",data=mensajes)
31
32     else:
33         origen=correo
34         resp2=controlador.lista_destinatarios(correo)
35         return render_template("principal.html",listaD=resp2,usuario=respuesta)
36
```

- **Ruta:** registrarUsuario. Se toman los datos de los nuevos registros de usuario, la contraseña es encriptada, se registra el usuario y por último se le envía un correo electrónico.

```
@app.route("/registrarUsuario",methods=["GET","POST"])
def registrarUsuario():

    nombre=request.form["txtnombre"]
    correo=request.form["txtusuarioregistro"]
    password=request.form["txtpassregistro"]
    password2=password.encode()
    password2=hashlib.sha384(password2).hexdigest()

    codigo=datetime.now()
    codigo2=str(codigo)
    codigo2=codigo2.replace("-","")
    codigo2=codigo2.replace(":","")
    codigo2=codigo2.replace(".", "")
    codigo2=codigo2.replace(" ","")

    controlador.regisUsuario(nombre,correo,password2,codigo2)

    asunto="Codigo de activacion"
    mensaje="su codigo de activacion es "+codigo2;
    envioemail.enviar(correo,asunto,mensaje)

    mensajes= "Usuario registrado satisfactoriamente..."
    return render_template("informacion.html",data=mensajes)
```

- **Ruta:** ActivarUsuario. Verifica si el usuario existe confirmando con el código de activación suministrado. Si el usuario existe, arroja un mensaje: "Usuario Activado con éxito" pero si no existe, aparecerá el mensaje "El código es incorrecto".

```
@app.route("/ActivarUsuario",methods=["GET","POST"])
def ActivarUsuario():

    codigo=request.form["txtcodigo"]

    respuesta=controlador.activarU(codigo)
    if len(respuesta)==0:
        mensajes= "El codigo es incorrecto"
        return render_template("informacion.html",data=mensajes)
    else:
        mensajes= "Usuario Activado con EXITO"
        return render_template("informacion.html",data=mensajes)
```

- **Ruta:** enviarEE. Permite enviar mensajes entre los usuarios que se encuentren registrados en el sistema.

```
@app.route("/enviarEE",methods=["GET","POST"])
def enviarEE():

    asunto=request.form["asunto"]
    mensaje=request.form["mensaje"]
    destino=request.form["destino"]

    controlador.registroEmail(asunto,mensaje,origen,destino)

    asunto2="Nuevo Mensaje"
    mensaje2="Usted recibio un nuevo mensaje por favor ingrese a
    la plataforma para observarlo."

    envioemail.enviar(destino,asunto2,mensaje2)

    return "Email Enviado Satisfactoriamente"
```

- **Ruta:** correosEnviados. Permite revisar los mensajes enviados a otros usuarios que se encuentren registrados en el sistema y direcciona hacia la plantilla [historial.html](#)

```
@app.route("/correosEnviados",methods=["GET","POST"])
def correosEnviados():

    respuesta=controlador.enviados(origen)
    return render_template("historial.html",listaCorreos=respuesta)
```

- **Ruta:** correosRecibidos. Permite revisar los mensajes recibidos de otros usuarios que se encuentren registrados en el sistema y direcciona hacia la plantilla [historial.html](#)

```
@app.route("/correosRecibidos",methods=["GET","POST"])
def correosRecibidos():

    respuesta=controlador.recibidos(origen)
    return render_template("historial.html",listaCorreos=respuesta)
```


- **Ruta:** actualizarPa. Permite que una vez logueado el usuario pueda actualizar su password desde la aplicación con un sistema de contraseña encriptada.

```
@app.route("/actualizarPa",methods=["GET","POST"])
def actualizarPa():
    password=request.form["password"]

    password2=password.encode()
    password2=hashlib.sha384(password2).hexdigest()

    controlador.actualziarPassW(password2,origen)

    return "La contraseña se ha actualizado correctamente"
```



## 1.2 Definición de métodos HTTP permitidos

Para **Consultar** el método usado es: **GET**

Para **Insertar** el método usado es: **POST**

Para **Actualizar** (el código de activación) el método usado es: **PUT**

## 1.3 Lógica Algorítmica

Se encuentran fragmentos de código como el siguiente:

```
@app.route("/verificarUsuario",methods=["GET","POST"])
def verificarUsuario():

    correo=request.form["txtusuario"]
    password=request.form["txtpass"]
    password2=password.encode()
    password2=hashlib.sha384(password2).hexdigest()

    global origen

    respuesta=controlador.consultar_usuario(correo, password2)

    if len(respuesta)==0:
        origen=""
        mensajes= "Error de autenticacion, verifique su usuario y contraseña."
        return render_template("informacion.html",data=mensajes)

    else:
        origen=correo
        resp2=controlador.lista_destinatarios(correo)
        return render_template("principal.html",listaD=resp2,usuario=respuesta)
```

Al consultar el usuario comparamos la cantidad de registros que tiene. Si éste viene vacío, es decir, viene "=="0", se envía un error de autenticación. De lo contrario se envía la información del usuario.

También podemos encontrar procesos lógicos como el correspondiente a la contraseña:

```
@app.route("/actualizarPa",methods=["GET","POST"])
def actualizarPa():
    password=request.form["password"]

    password2=password.encode()
    password2=hashlib.sha384(password2).hexdigest()

    controlador.actualziarPassW(password2,origen)

    return "La contraseña se ha actualizado correctamente"
```

Tomamos el password del formulario donde lo digita el usuario

Lo codificamos

Lo encriptamos

Por último destacamos otro de los procesos lógicos más relevantes que hace referencia al Código de activación:

```
@app.route("/registrarUsuario",methods=["GET","POST"])
def registrarUsuario():

    nombre=request.form["txtnombre"]
    correo=request.form["txtusuarioregistro"]
    password=request.form["txtpassregistro"]
    password2=password.encode()
    password2=hashlib.sha384(password2).hexdigest()

    codigo=datetime.now()
    codigo2=str(codigo)
    codigo2=codigo2.replace("-","")
    codigo2=codigo2.replace(":","")
    codigo2=codigo2.replace(".", "")
    codigo2=codigo2.replace(" ","")

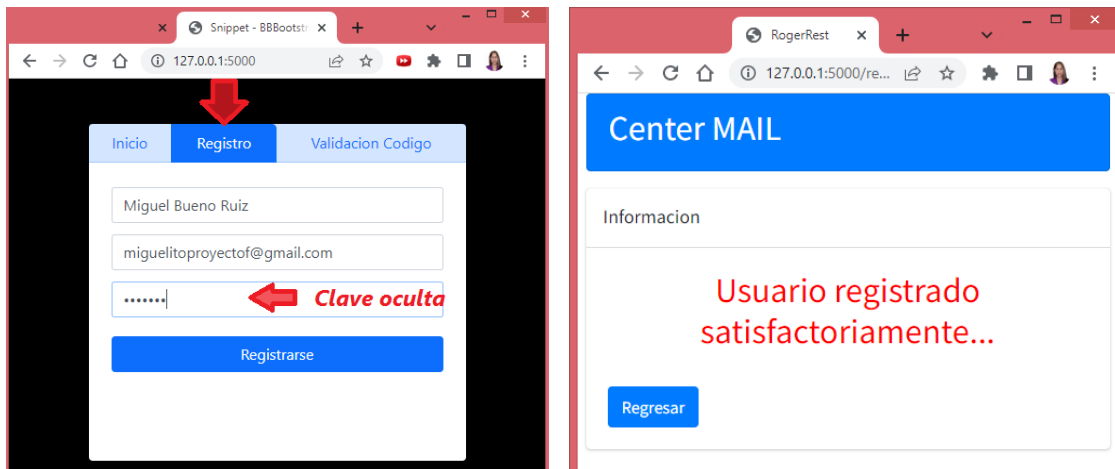
    controlador.regisUsuario(nombre,correo,password2,codigo2)

    asunto="Codigo de activacion"
    mensaje="su codigo de activacion es "+codigo2;
    envioemail.enviar(correo,asunto,mensaje)
```

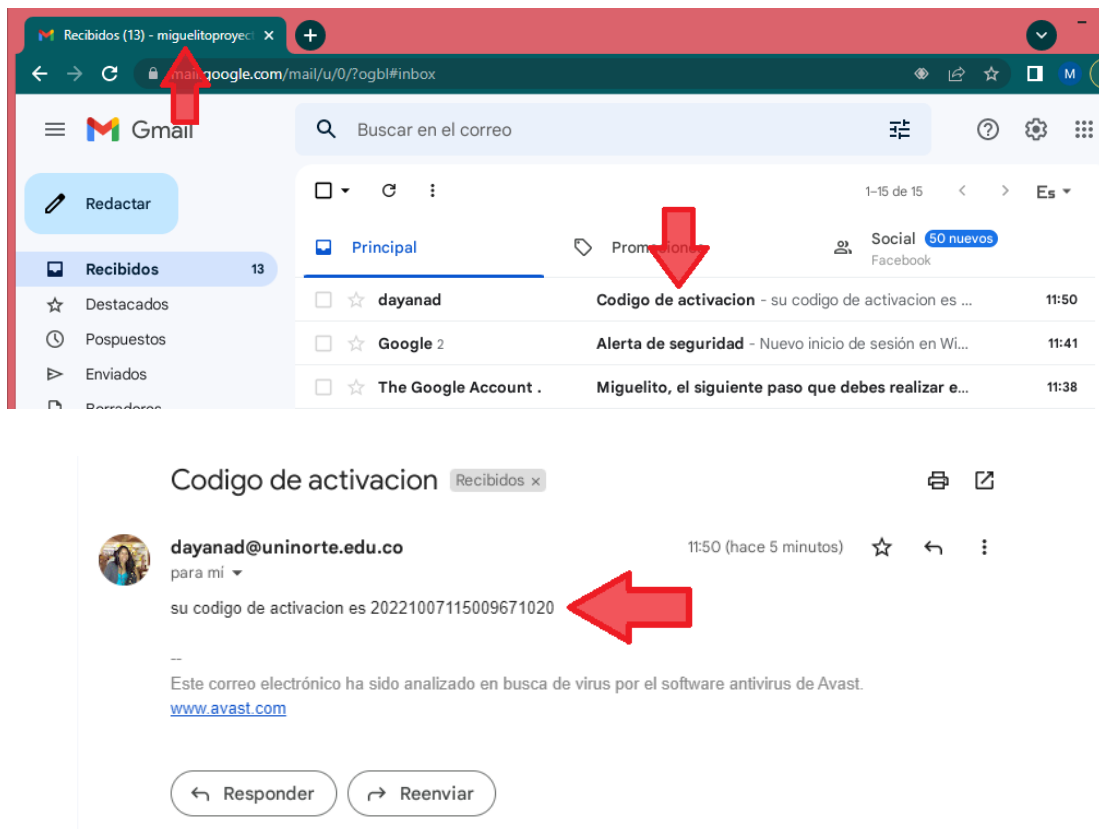
Código de activación único para cada usuario el cual evita la duplicidad de información. Aquí se incluye la función `datetime` y `now` que trae el día, mes, año, hora, minutos, segundos y milisegundos. Se limpian los signos de puntuación que trae el objeto para obtener un código único y almacenarlo en la Base de Datos.

## 1.4 Documento descriptivo del diseño y la especificación de los controladores definidos

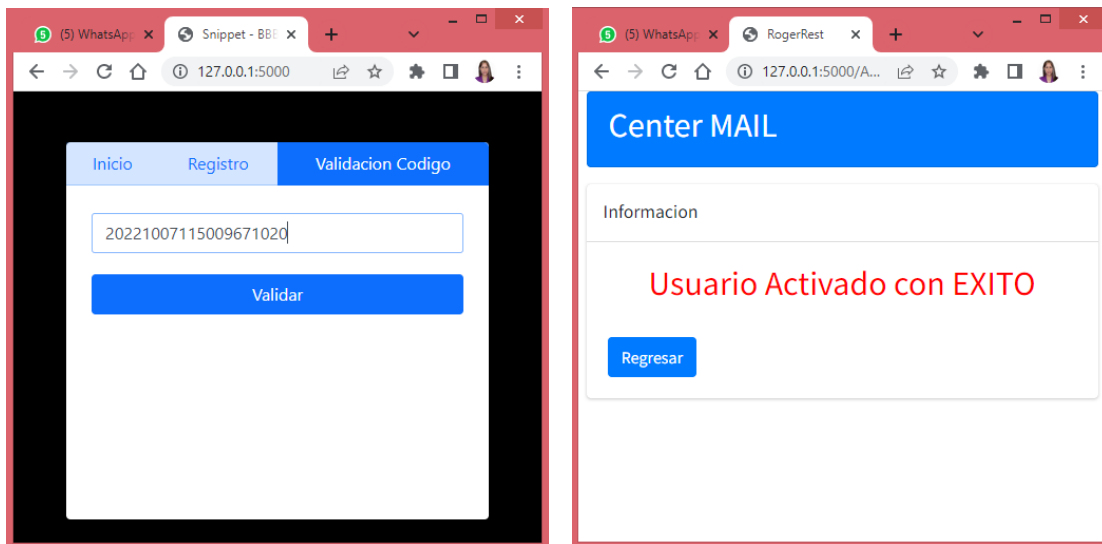
- **Vista de registro:**



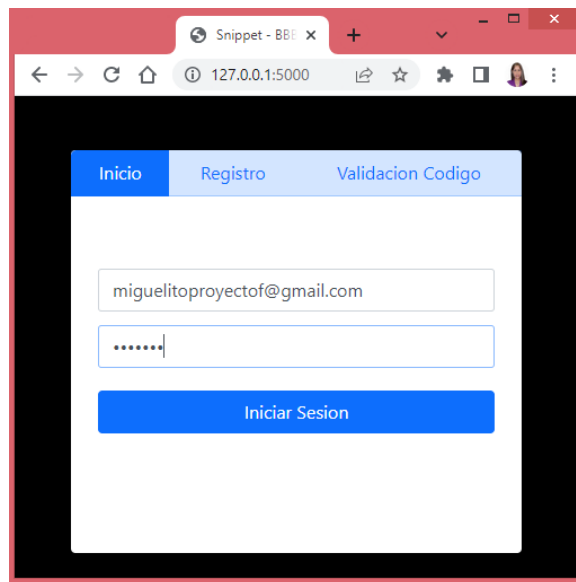
- **Vista de Validación de código entregado en el registro de usuario:**



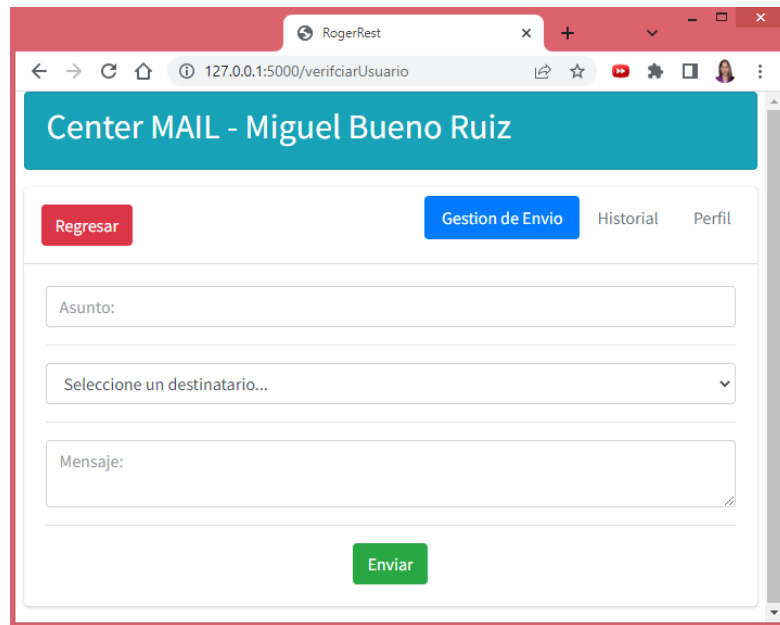
- **Ingreso y validación del código en la plataforma:**



- **Vista de inicio de sesión:**

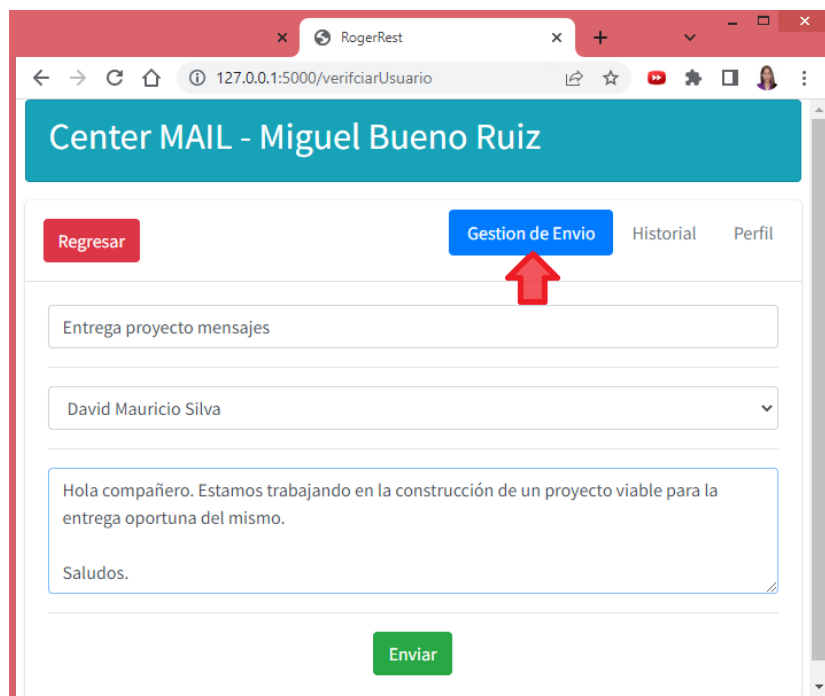


- **Vista una vez se ingresa a la plataforma:**



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/verificarUsuario". The page title is "Center MAIL - Miguel Bueno Ruiz". The interface includes a navigation bar with buttons: "Regresar" (red), "Gestion de Envio" (blue), "Historial", and "Perfil". Below the navigation bar, there is a form with the following fields: "Asunto:" (Subject), "Seleccione un destinatario..." (Select a recipient), and "Mensaje:" (Message). A green "Enviar" (Send) button is located at the bottom of the form.

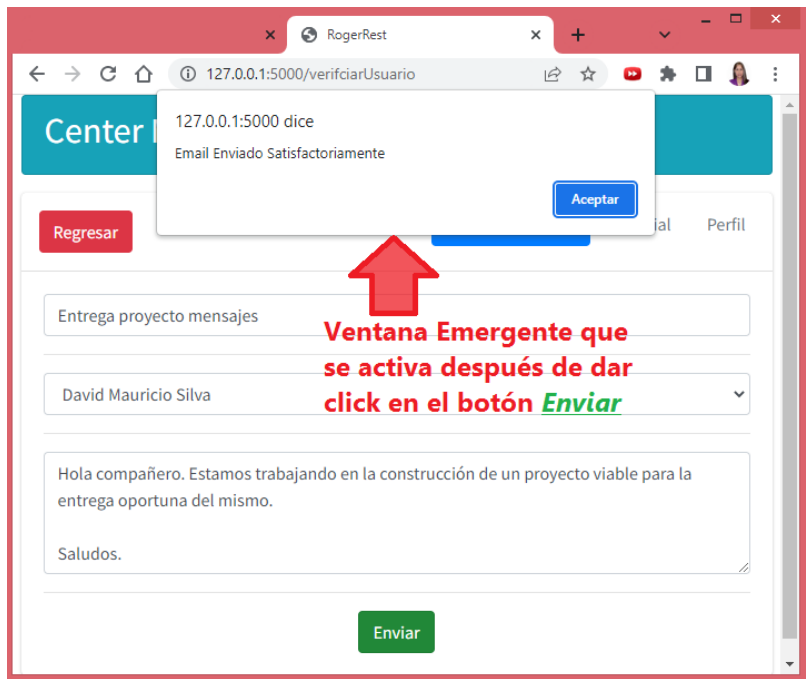
- **Vista de la pestaña Gestión de Envío de Mensajes:**



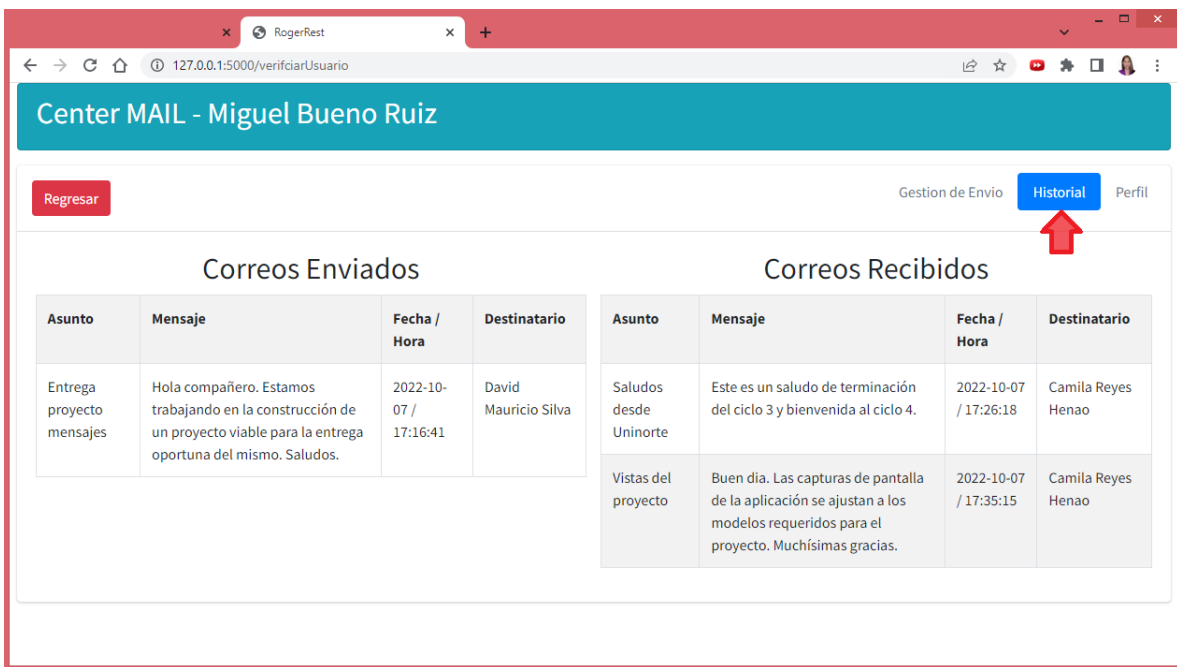
The screenshot shows the same web browser window, but the "Gestion de Envio" tab is selected and highlighted in blue. A red arrow points to this tab. The form content has changed: the "Asunto:" field now contains "Entrega proyecto mensajes", the "Seleccione un destinatario..." dropdown now shows "David Mauricio Silva", and the "Mensaje:" field contains the text "Hola compañero. Estamos trabajando en la construcción de un proyecto viable para la entrega oportuna del mismo. Saludos." The green "Enviar" button remains at the bottom.



- **Vista de envío de un nuevo mensaje:**



- **Vista de la pestaña Historial (muestra al detalle los correos enviados y recibidos):**



- **Vista de solicitud de recuperación de contraseña:**

Center MAIL - Miguel Bueno Ruiz

Regresar Gestion de Envio Historial Perfil

Nueva contraseña

Confirmacion Nueva contraseña

Actualizar

- **Vista de cambio/recuperación de contraseña:**

Center MAIL - Miguel Bueno Ruiz

Regresar Gestion de Envio Historial Perfil

\*\*\*\*\*

\*\*\*\*\*

Actualizar

127.0.0.1:5000 dice  
La contraseña se ha actualizado correctamente

Aceptar

*Mensaje de confirmación*

Actualizar

- **Botón Regresar:** Una vez logueado, este botón se encuentra en todas las vistas y permite volver a la página de inicio de la aplicación.

Center MAIL - Miguel Bueno Ruiz

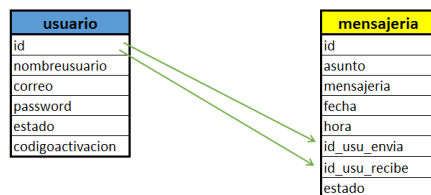
Regresar Gestion de Envio Historial Perfil

Asunto:

## 2. DISEÑO E IMPLEMENTACIÓN DE BASE DE DATOS

### 2.1 Diseño de Diagrama Relacional

MODELO RELACIONAL DEL PROYECTO



### 2.2 Diseño e implementación de las tablas de las bases de datos con SQLite

Tabla **mensajería** con el contenido de los campos, tipos de campos y atributos asignados a los mismos.

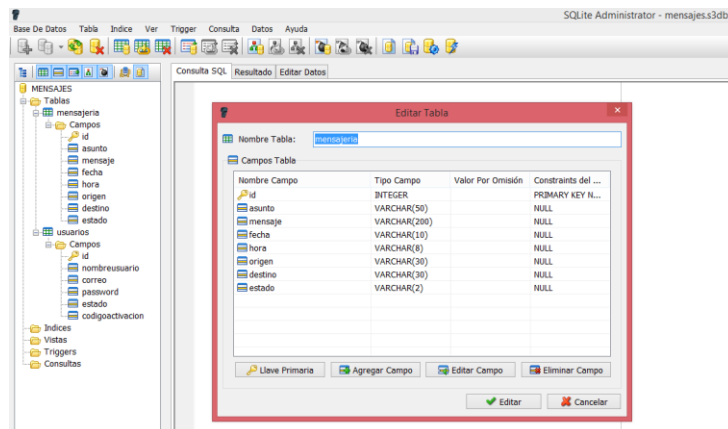
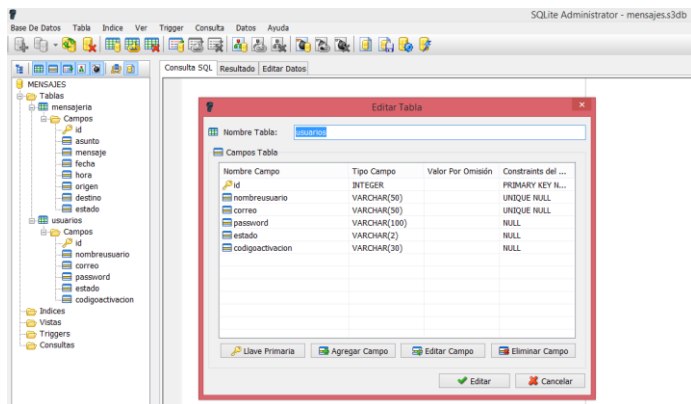


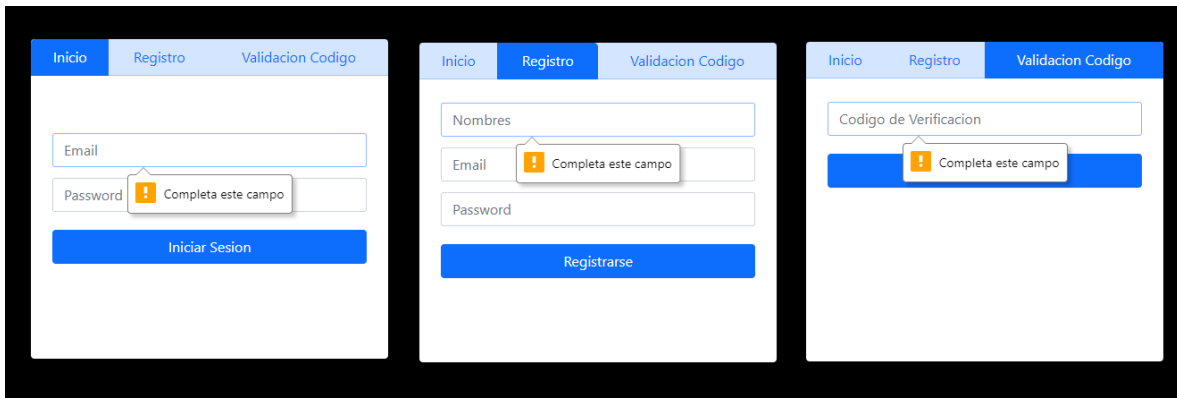
Tabla **usuarios** con el contenido de los campos, tipos de campos y atributos asignados a los mismos.



El nombre de usuario y correo tienen características únicas para ser verificados a la hora de validar el usuario en el registro a la plataforma.

### 3. DESARROLLO E INTEGRACIÓN DE CONTROLADORES Y BASES DE DATOS

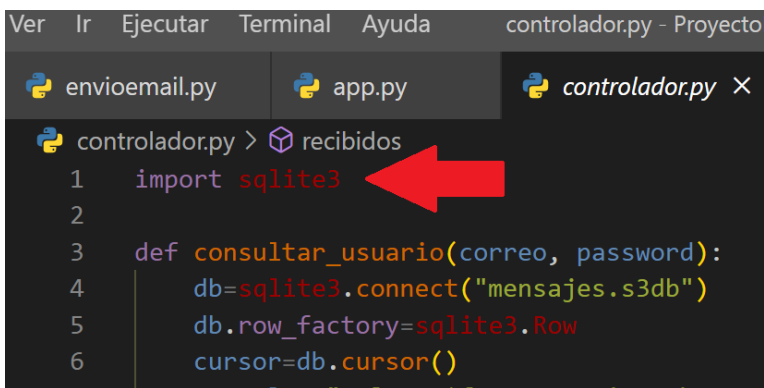
#### 3.1 Validación de los datos de entrada



The image displays three side-by-side screenshots of a web application interface, each showing a validation error. The interface has a top navigation bar with three tabs: 'Inicio', 'Registro', and 'ValidacionCodigo'.  
1. The first screenshot shows the 'Inicio' (Login) page. It has input fields for 'Email' and 'Password'. A red error message 'Completa este campo' (Complete this field) is shown next to the 'Password' field.  
2. The second screenshot shows the 'Registro' (Registration) page. It has input fields for 'Nombres', 'Email', and 'Password'. A red error message 'Completa este campo' is shown next to the 'Email' field.  
3. The third screenshot shows the 'ValidacionCodigo' (Verify Code) page. It has a 'Codigo de Verificacion' (Verification Code) input field. A red error message 'Completa este campo' is shown next to the input field.

El sistema valida los datos de entrada ya que al dar click en Iniciar Sesión, Registrarse o Validar Código sin haber llenado la información solicitada, enseguida arroja un mensaje de error y solicita el ingreso de datos.

#### 3.2 Uso de librerías seguras (o prepared statements) para consultar/actualizar las bases de datos



The image shows a code editor window with the title 'controlador.py - Proyecto'. The editor has three tabs: 'envioemail.py', 'app.py', and 'controlador.py'. The 'controlador.py' tab is active, showing the following code:

```
controlador.py > recibidos
1  import sqlite3
2
3  def consultar_usuario(correo, password):
4      db=sqlite3.connect("mensajes.s3db")
5      db.row_factory=sqlite3.Row
6      cursor=db.cursor()
```

A red arrow points to the 'import sqlite3' line on line 1.

Uso de la librería **sqlite3**

### 3.3 Diseño de queries para consultar/actualizar las bases de datos

```
def consultar_usuario(correo, password):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="select *from usuarios where correo='"+correo+"' and password='"+password+"' and estado='1'"
    cursor.execute(consulta)
    return cursor.fetchall()

def lista_destinatarios(correo):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="select *from usuarios where estado='1' and correo>='"+correo+"'"
    cursor.execute(consulta)
    return cursor.fetchall()

def enviados(correo):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="select m.asunto,m.mensaje,m.fecha,m.hora, u.nombreusuario from mensajeria m, usuarios u where m.origen='"+correo+"' and m.destino=u.correo"
    cursor.execute(consulta)
    return cursor.fetchall()

def recibidos(correo):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="select m.asunto,m.mensaje,m.fecha,m.hora, u.nombreusuario from mensajeria m, usuarios u where m.destino='"+correo+"' and m.origen=u.correo"
    cursor.execute(consulta)
    return cursor.fetchall()

def regisUsuario(nombre,correo, password,codigo):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="insert into usuarios (nombreusuario,correo,password,estado,codigoactivacion) values ("
    cursor.execute(consulta)
    db.commit()
    return "1"

def actualizarPassW(pwd, correo):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="update usuarios set password='"+pwd+"' where correo='"+correo+"' "
    cursor.execute(consulta)
    db.commit()
    return "1"

def registroEmail(asunto,mensaje,origen,destino):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="insert into mensajeria (asunto,mensaje,fecha,hora,origen,destino,estado) values ("
    cursor.execute(consulta)
    db.commit()
    return "1"

def activarU(codigo):
    db=sqlite3.connect("mensajes.s3db")
    db.row_factory=sqlite3.Row
    cursor=db.cursor()
    consulta="update usuarios set estado='1' where codigoactivacion='"+codigo+"'"
    cursor.execute(consulta)
    db.commit()

    consulta="select *from usuarios where codigoactivacion='"+codigo+"' and estado='1'"
    cursor.execute(consulta)
    return cursor.fetchall()
```

### 3.4 Documento descriptivo de las buenas prácticas de programación segura para fortalecer su aplicación.

- ✓ La tabla usuario comienza su seguridad creando usuarios con nombres únicos y correos únicos para evitar que se repitan nombres de usuarios o correos electrónicos en la plataforma de mensajería creada.
- ✓ En el momento almacenar, consultar o actualizar el password se usa una técnica de codificación y encriptación con la finalidad de almacenar la contraseña de forma segura.

```
@app.route("/actualizarPa", methods=["GET", "POST"])
def actualizarPa():
    password=request.form["password"]

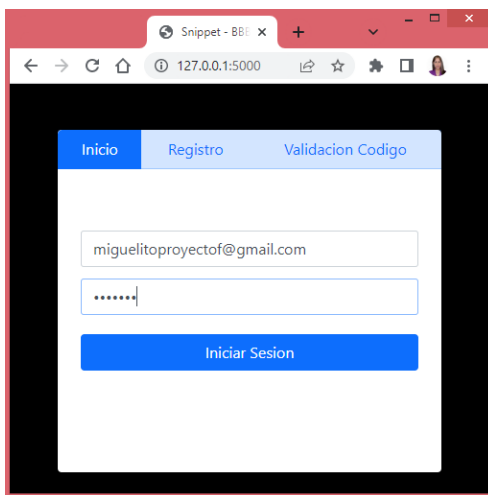
    password2=password.encode()
    password2=hashlib.sha384(password2).hexdigest()

    controlador.actualizarPassw(password2,origen)

    return "La contraseña se ha actualizado correctamente"
```

## 4. DISEÑO E IMPLEMENTACIÓN DE PORTAL DE ACCESO USANDO MÉTODO DE AUTENTICACIÓN BASADO EN USUARIO Y CONTRASEÑA

### 4.1 Creación de sesiones



Al iniciar se crean variables globales con el fin de mantener el correo electrónico de manera única en el sistema para así coordinar todo el proceso de validación y control.

Se anexa Vínculo de acceso a la carpeta del proyecto para su respectiva revisión:

[https://drive.google.com/drive/folders/1cwCSsqfyW8f2\\_mY8jsyNxuis2kW6V4Dr?usp=sharing](https://drive.google.com/drive/folders/1cwCSsqfyW8f2_mY8jsyNxuis2kW6V4Dr?usp=sharing)