**Documentación Proyecto**

**PLATAFORMA DE MENSAJERIA ELECTRONICA**

**NRC:** 2131

**Equipo No.** 14

**Integrantes:**

*YOLIANA ROSADO CABRERA*

*MAURICIO SEPULVEDA G.*

*JUAN CAMILO GIRALDO VALENCIA*

*NIDIA JULIETH VELA MORENO*

**Descripción de roles del equipo** (*Sprint 1*)

|  |  |  |  |
| --- | --- | --- | --- |
| Rol | Integrante | Descripción | Tareas |
| LIDER EQUIPO | **JUAN CAMILO** | **Es la persona encarda de velar por las entregas y que se estén cumpliendo todos los entregables.** | 1. **Entregar los entregables.** 2. **Organizar los documentos.** 3. **Revisiones de ortografía.** 4. **Apoyar a los desarrolladores backend.** |
| FRONTEND | **NIDIA J. VELA M.** | **Se encarga de la capa frontal y la compatibilidad de los dispositivos, incluye el diseño, la distribución del contenido, las funcionalidades y todos los elementos con los que un usuario interactúa.** | 1. **Implementación del html.** 2. **Implementación del CSS.** 3. **Implementación del javascript.** |
| BACKEND | **YOLIANA ROSADO** | **Su función es acceder a la información que se solicita, a través de la app, para luego combinarla y devolverla al usuario final.** | 1. **Coordina y realiza paginas y formularios.** 2. **Revisa la funcionalidad de las bases de datos.** 3. **Revisa la funcionalidad del sitio web.** |
| FULLSTACK | **MAURICIO SEPULVEDA** | **Su función es estar en desarrollo tanto de la parte de usuarios y clientes como el servidor.** | 1. **Desarrollo de la estructura y arquitectura web.** 2. **Diseño de la interfaz y parte visual.** 3. **Facilitar la navegación y mejorar la experiencia del usuario.** |
|  |  |  |  |

**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 | Todos los integrantes |
| Definición de artefactos | Definición de los artefactos de la metodología SCRUM para el desarrollo del proyecto. | 4 horas | Todos los integrantes |
| Diseño del diagrama de clases | Diseño del diagrama de clases de la aplicación a desarrollar. | 2 horas | Todos los integrantes |
| Definición del cronograma | Definición del cronograma de actividades del proyecto. | 2 horas | Todos los integrantes |

**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 | Integrante 1 y 4 |
| Selección de la plantilla de estilos | Selección de la librería CSS a usar para los estilos de las vistas de la aplicación. | 1 hora | Integrante 1 y 4 |
| Diseño e implementación de las vistas | Diseño e implementación de las vistas en HTML y CSS. | 15 horas | Integrante 1 y 4 |
| Creación del proyecto en GIT | Creación del proyecto en GIT y posterior cargue a la nube en la plataforma Github. | 1 hora | Integrante 1 |

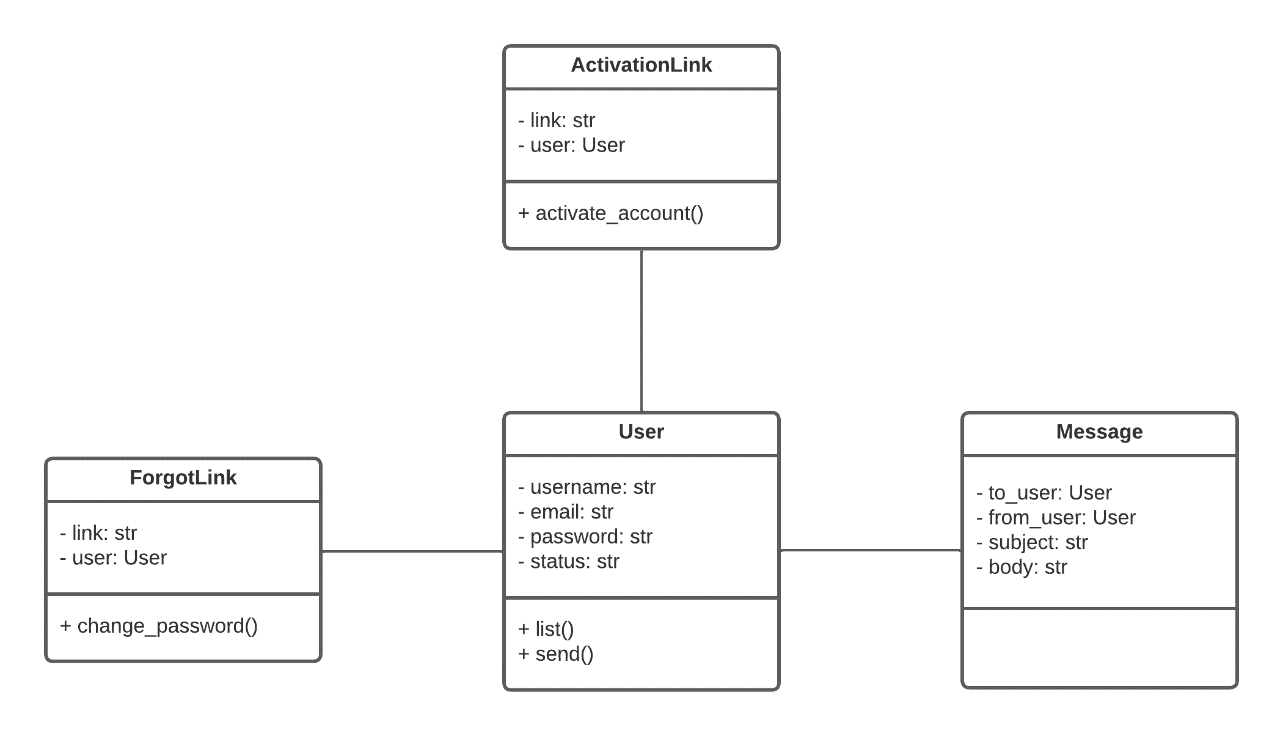
**Backlog Sprint 3**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Integrante 2 y 3 |
| 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 | Integrante 4 |
| 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 | Integrante 2 y 3 |
| Diseño e implementación 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 | Integrante 4 |

**Backlog Sprint 4**

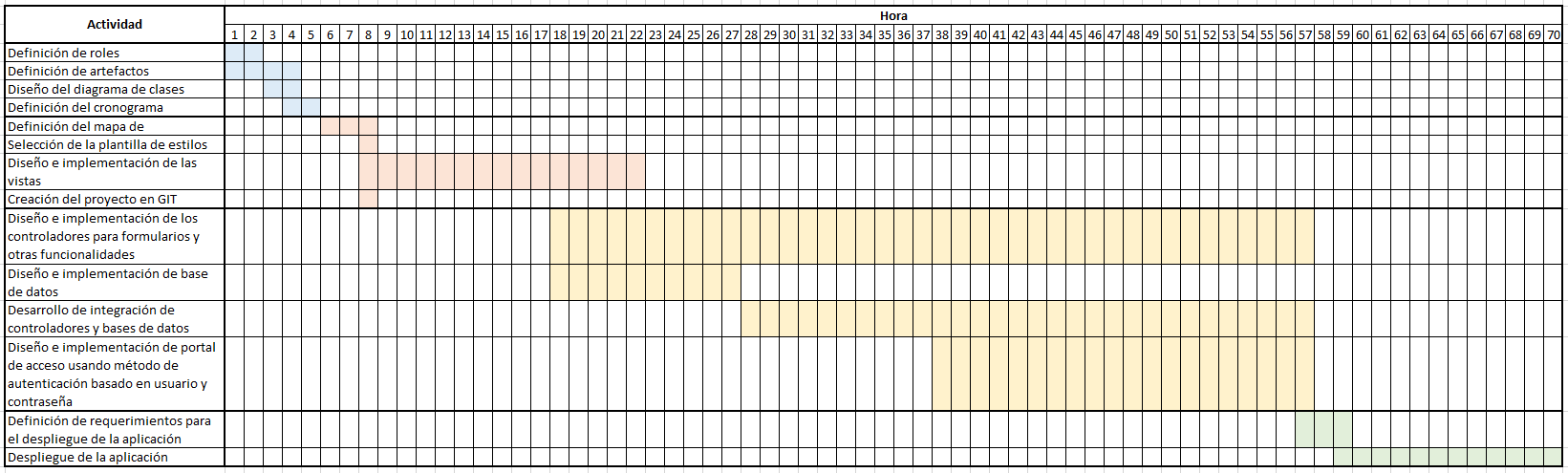
|  |  |  |  |
| --- | --- | --- | --- |
| User story | Descripción | Estimación | Responsable |
| Definición de requerimientos para el despliegue de la aplicación | Definición de los requerimientos necesarios para realizar el despliegue de la aplicación en la plataforma PythonAnywhere | 3 horas | Todos los integrantes |
| Despliegue de la aplicación | Configuración, despliegue y verificación del funcionamiento de la aplicación en la plataforma PythonAnywhere | 12 horas | Todos los integrantes |

**Diagrama de clases**

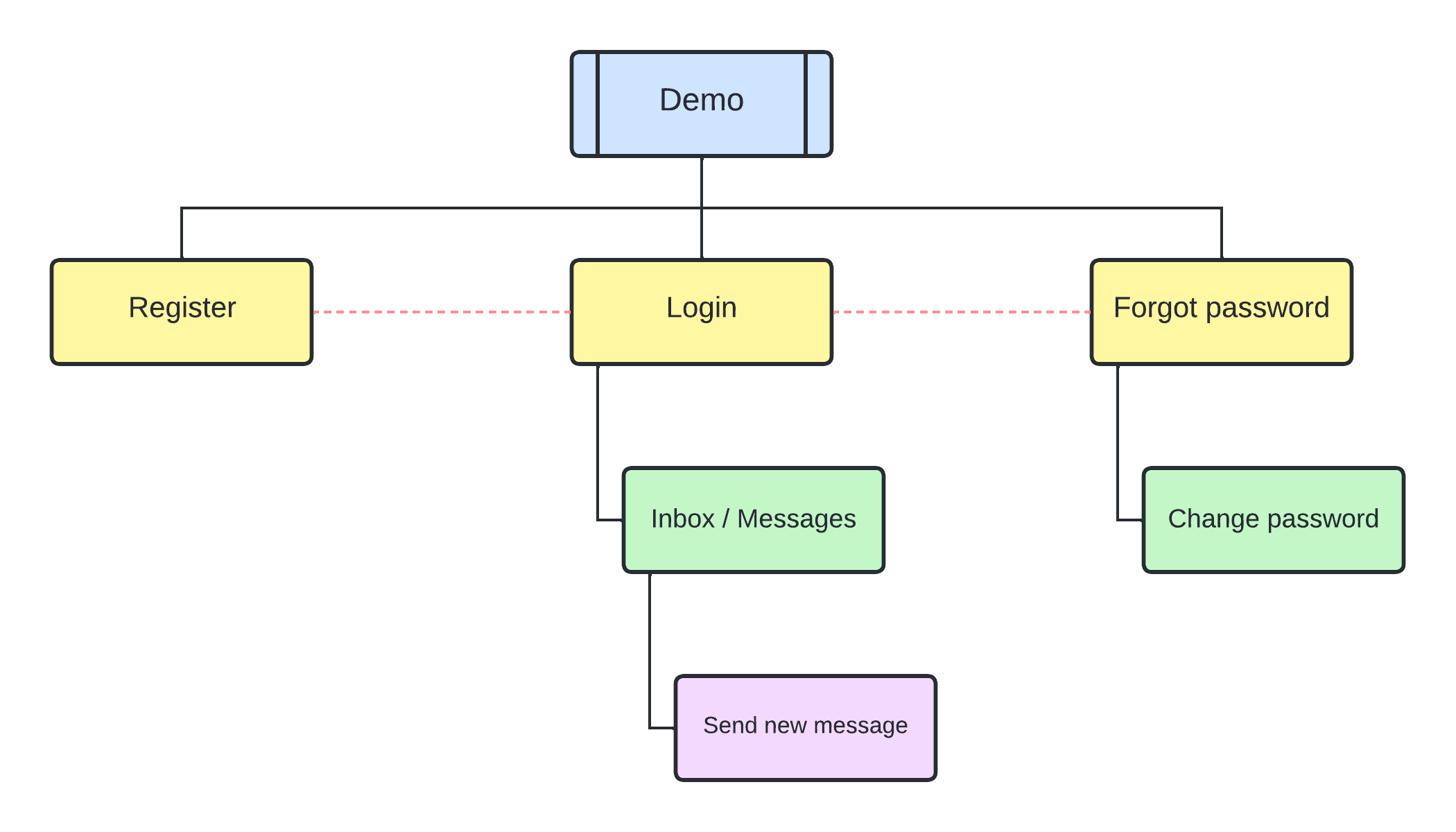


La clase usuario tendrá la información referente al usuario, es decir, su nombre de usuario, su correo, su contraseña y el estado de su cuenta. La clase mensaje tendrá la información de los mensajes de la plataforma, es decir, el usuario que envía el mensaje, el usuario que lo recibe, el asunto y el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

**Cronograma de tareas**



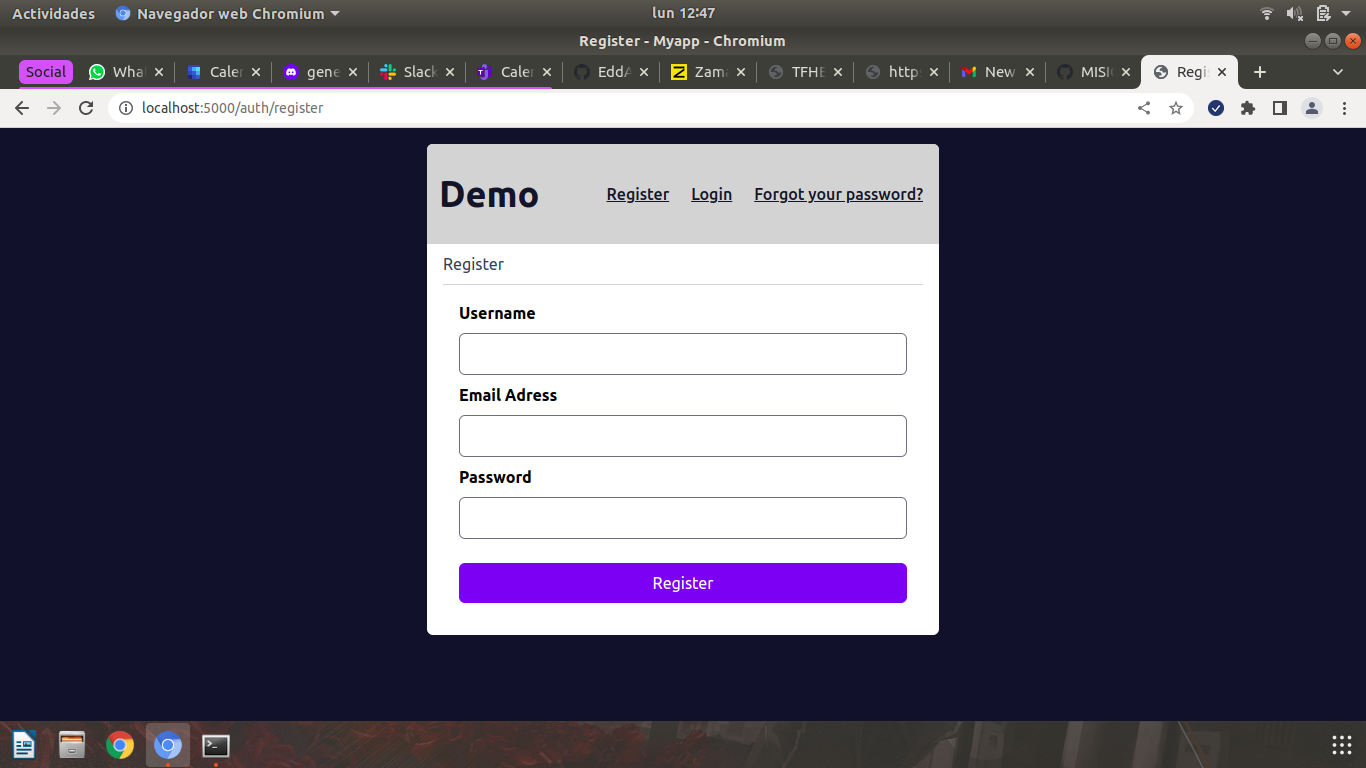
**Mapa de navegabilidad**



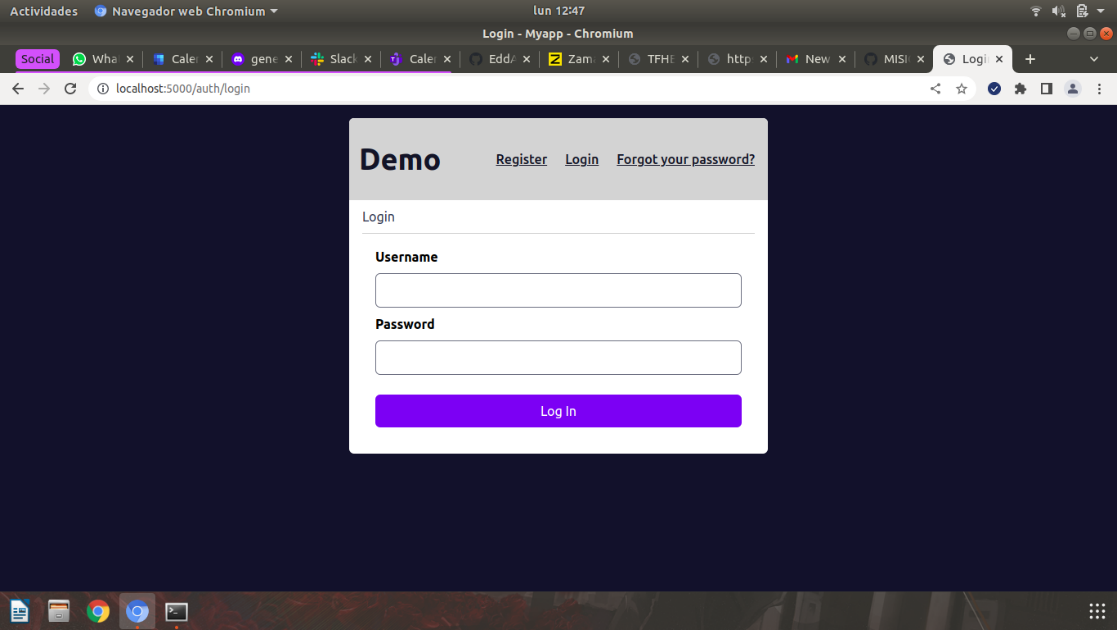
La aplicación en su pantalla principal tiene 3 opciones, estas son el registro, el inicio de sesión y la recuperación de contraseña, y se puede desplazar entre ellas. La pantalla de recuperación de contraseña, dará paso a la pantalla de cambio de contraseña al acceder a ella mediante el enlace enviado.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

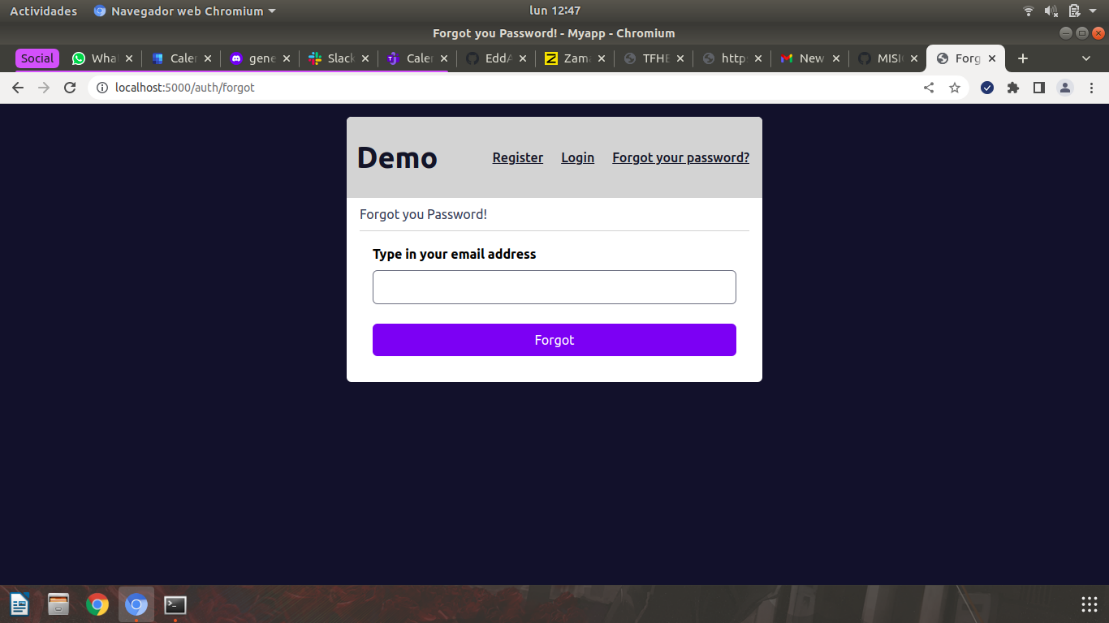
* **Vista de registro:**



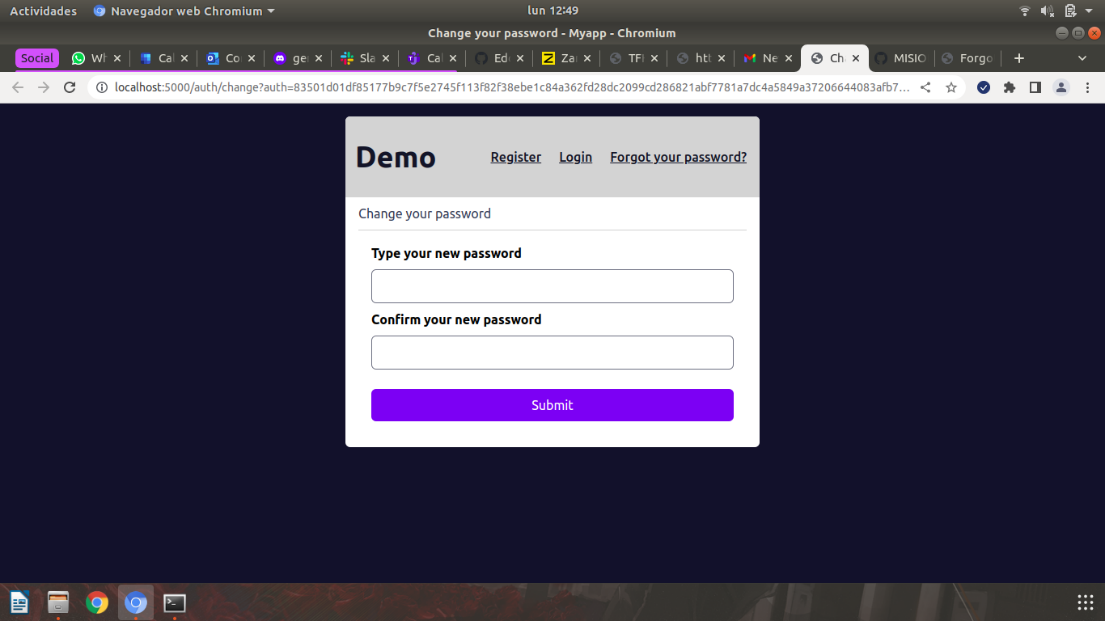
* **Vista de inicio de sesión:**



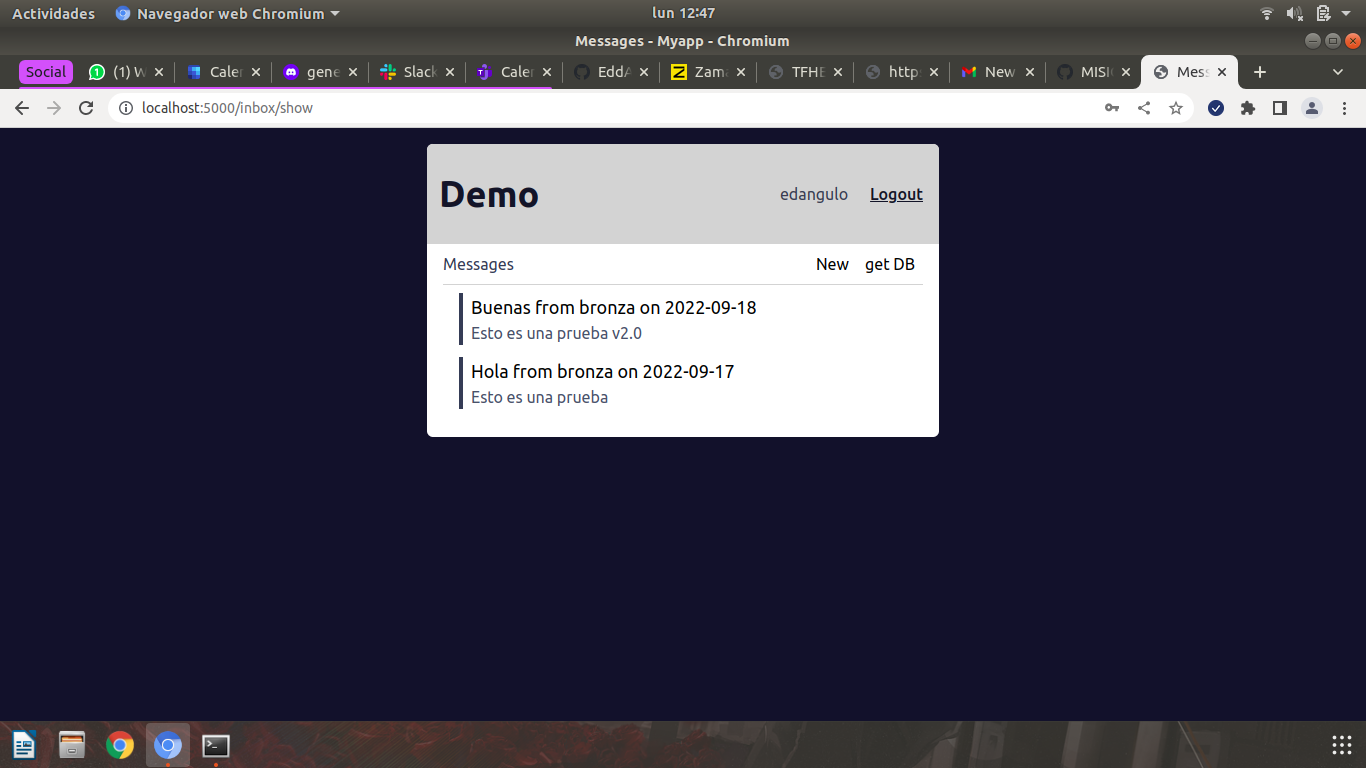
**Vista de solicitud de recuperación de contraseña:**



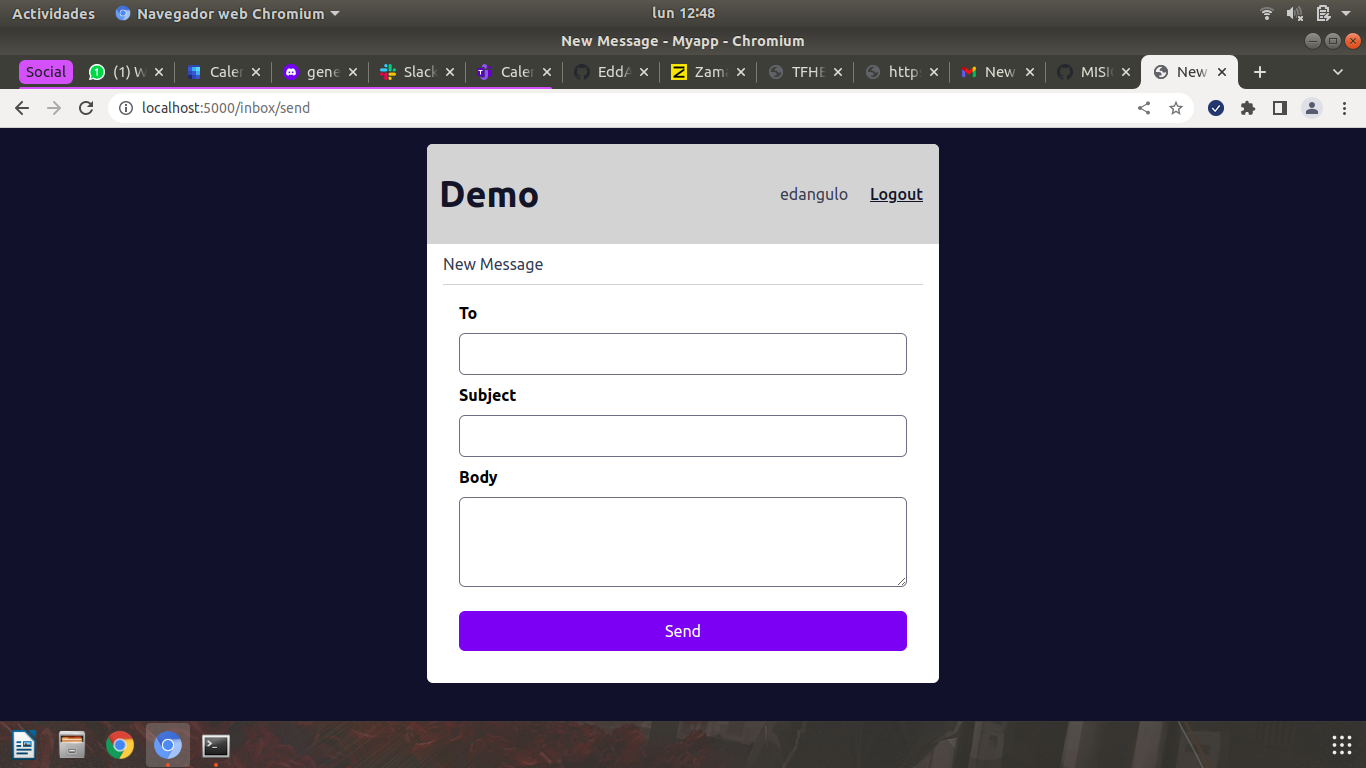
* **Vista de cambio/recuperación de contraseña:**



* **Vista de visualización de mensajes:**



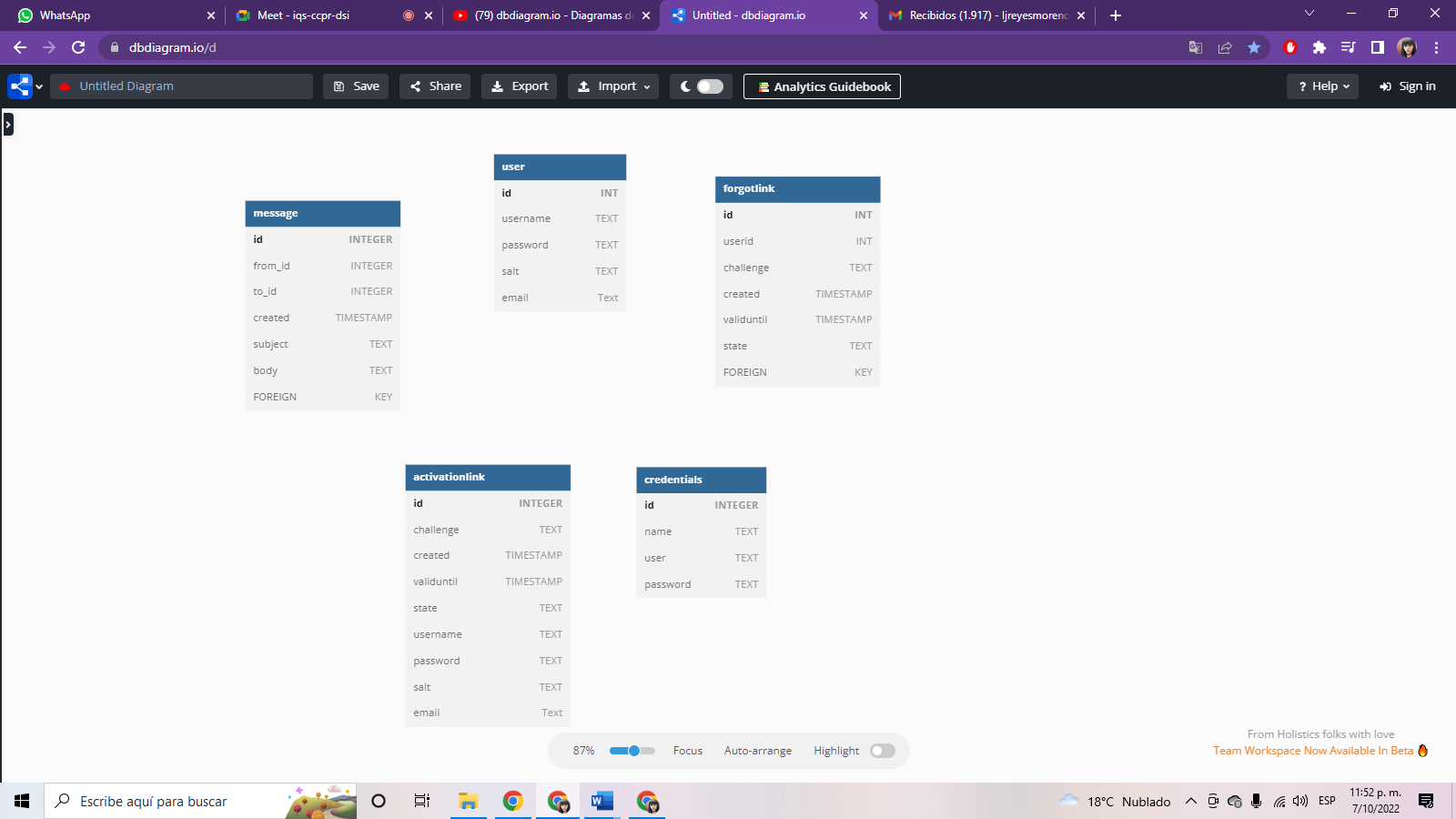
* **Vista de envío de un nuevo mensaje:**



**Métodos controladores del Back-End** (*Sprint 3*)

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Ruta | Métodos HTTP permitidos | Lógica algorítmica |
| Register() | **/register** | **'POST'** | Para poder acceder a la plataforma, el usuario final deberá registrarse en la misma suministrando los siguientes datos:   * Nombre de usuario. * Correo electrónico.   Contraseña. |
| Activate() | **/activate** | **'GET'** | El Controlador Activa su cuenta mediante el enlace enviado al correo electrónico ingresado durante el registro, el cual debe estar vigente y coincidir con el correo enviado para actualizar el campo state de la tabla “activationlink” e inserta los datos en la tabla “user” y redirige al template login.html |
| Login() | **/login** | **'POST'** | Ingresar a la plataforma realizando las siguientes validaciones:   * Sesión activa redirige al inbox.show. * Que los campos de login estén diligenciados y coincidan con los registrados en la base de datos. * Redirige al template show.html, si se cumplen todas las validaciones, caso contrario se queda en el template login.html |
| Forgot() | **/forgot** | **'POST'** | Este controlador, si hay sesión activa redirige al inbox.show, valida el campo del correo que no esté vacío y este registrado y actualiza el campo state de la tabla forgotlink.  Envía enlace de cambio de contraseña al correo registrado. |
| Change() | **/change** | **'GET'** | Este controlador, si hay sesión activa redirige al inbox.show, permitiendo el cambio mediante el ingreso del enlace de recuperación de contraseña enviado al correo, previa verificación de vigencia del mismo. El enlace redirige al template change.html |
| Confirm() | **/confirm** | **'POST'** | Esta función realiza la siguiente rutina.   * Si hay sesión activa redirige al inbox.show * Valida que los campos de cambio de contraseñas no estén vacíos y que además ambos coincidan. Sino renderiza el template change.html. * Si se cumplen las validaciones, actualiza el campo state de la tabla “forgotlink” * Actualiza el campo password de la tabla “user”. * Y Redirige al template de login.html. |
| Logout() | **/logout** |  | Este método cierra la sesión y redirige al template login.html |
| Show() | **/show** |  | Este controlador renderiza el template show.html donde aparecen los mensajes recibidos por parte de otros usuarios. Además, se visualiza el asunto, el usuario que lo ha enviado, la fecha en que lo envió y el cuerpo del mensaje. |
| Send() | **/send** | **'POST'** | Este controlador validad que los campos to\_id, subject, body no estén vacíos, que el destinatario exista en la base de datos y almacena en la tabla “message” la información del mensaje enviado.  Si cumple con todas las validaciones redirige al template show.html, caso contrario renderiza el template send.html |

**Base de datos** (*Sprint 3*)



*La tabla user, almacenará la información de los usuarios y tendrá una relación de uno a muchos con las tablas “message” y “forgotlink”. Por otra parte, tenemos a la tabla “activationlink” la cual almacenara la información referente a la generación de enlaces de activación y la tabla “credentials” la cual almacenara los datos de la cuenta encargada de generar los enlaces de activación y recuperación de contraseña, ambas sin relaciones con las demás tablas.*

**Prácticas de programación segura** (*Sprint 3*)

*Las siguientes son buenas prácticas de programación aplicadas en el desarrollo de este proyectó:*

*Librerías seguras para generar y validar passwords encriptados (from werkzeug.security import check\_password\_hash, generate\_password\_hash)*

*Librería segura para Manejo de sesión.*

*No se ejecuta sentencias QUERY concatenando las variables para prevenir el ataque de inyección SQL a la base de datos. Evitando vulnerabilidades de acceso y modificación de datos.*

*Se establece y controla la entrada de datos desde el frontend validando los datos antes de ser procesados.*

**Link Repositorio GitHub** (*Sprint 3*)