UNIVERSIDAD DEL VALLE DE GUATEMALA

1CC20082020232- PROGRAMACIÓN ORIENTADA A OBJETOS

Sección 20

Proyecto parte 2

Felipe Aguilar, 23195
Hugo Barillas, 23306
Gabriel Quan, 23479

GUATEMALA, 08 de octubre de 2023

Identificación de requisitos funcionales

- Crear cuentas de correo electrónico: las aplicaciones deben permitir a los usuarios crear cuentas de correo electrónico proporcionando su nombre y apellido.

- Generación automática de contraseñas: la aplicación debería generar automáticamente contraseñas seguras para las cuentas de correo electrónico creadas por los usuarios.

- Selección de departamento: las aplicaciones deben permitir a los usuarios seleccionar su departamento (por ejemplo, estudiante, profesor, etc.) al crear una cuenta de correo electrónico.

- Cambio de contraseña: los usuarios deberían poder cambiar su contraseña en caso de que quieran establecer una contraseña personalizada.

- Capacidad del buzón: los usuarios deben poder configurar la capacidad de su buzón (por ejemplo, el tamaño en megabytes).

- Correo electrónico alternativo: debería haber una opción para que los usuarios establezcan una dirección de correo electrónico alternativa.

En base a lo que hace el programa con todo lo anterior dicho, este primero preguntará por el nombre, luego por el apellido. Finalmente este pedirá el departamento el cual este pertenece. Este será pedido en forma númerica; 1, 2 y 3. 1) estudiante; 2) profesor; 3) Ninguno de los anteriores, el cual quedará vacío y no se añadirá al correo creado

Priorización de los requisitos funcionales encontrados

En orden:

<u>Prioridad 1 (Los más importantes):</u>

1. Registro de usuario: Sin la capacidad de registrar nuevos usuarios, el sistema no puede tener usuarios que utilicen el correo electrónico. Esta es una funcionalidad crítica.

2. Generación de contraseña aleatoria: La generación de una contraseña segura es crucial para garantizar la seguridad de las cuentas de usuario. Sin esta función, las cuentas estarían en riesgo.

3. Selección de departamento: Si bien es necesario para la creación de la dirección de correo electrónico, no es una funcionalidad que los usuarios deban realizar constantemente. Puede ser implementado después de las funcionalidades críticas.

4. Creación de dirección de correo electrónico: La creación de la dirección de correo electrónico es esencial para que los usuarios tengan identificadores únicos y puedan enviar y recibir correos electrónicos.

Prioridad 2 (Importantes, pero no críticos):

1. Mostrar información de la cuenta: Mostrar la información de la cuenta es importante para que los usuarios puedan ver y verificar su información. Aunque no es crítico para el funcionamiento del sistema, es útil.
2. Mostrar el resultado de la contraseñas
3. Mostrar el dominio del correo electrónico

Identificación y Descripción de Clases necesarias:

Clase `Email`:

>Propiedades:
1. `firstName` (String): Almacena el primer nombre del usuario.
2. `lastName` (String): Almacena el apellido del usuario.
3. `password` (String): Almacena la contraseña del correo electrónico.
4. `department` (String): Almacena el departamento (estudiante, profesor) del usuario.
5. `email` (String): Almacena la dirección de correo electrónico completa del usuario.
6. `mailboxCapacity` (int): Almacena la capacidad del buzón de correo (valor predeterminado: 500).
7. `defaultPasswordLength` (int): Almacena la longitud predeterminada de la contraseña (valor predeterminado: 10).
8. `alternateEmail` (String): Almacena la dirección de correo electrónico alternativa del usuario.
9. `companySuffix` (String): Almacena el sufijo de la empresa para el dominio del correo electrónico (valor predeterminado: "UNI.com")

>Métodos:

1. `Email(String firstName, String lastName)`: Constructor que recibe el primer nombre y el apellido del usuario. Este constructor inicializa las propiedades `firstName`, `lastName`, `department`, `password`, y `email` utilizando otros métodos dentro de la clase.
2. `private String setDepartment()`: Método privado que pregunta al usuario por el departamento y devuelve el departamento seleccionado como una cadena.

3. `private String randomPassword(int length)`: Método privado que genera una contraseña aleatoria de la longitud especificada y la devuelve como una cadena.
4. `public void setMailboxCapacity(int capacity)`: Método público para establecer la capacidad del buzón de correo.
5. `public void setAlternateEmail(String altEmail)`: Método público para establecer la dirección de correo electrónico alternativa.
6. `public void changePassword(String password)`: Método público para cambiar la contraseña.
7. `public int getMailboxCapacity()`: Método público para obtener la capacidad del buzón de correo.
8. `public String getAlternateEmail()`: Método público para obtener la dirección de correo electrónico alternativa.
9. `public String getPassword()`: Método público para obtener la contraseña.
10. `public String showInfo()`: Método público que muestra la información del usuario, incluyendo el nombre, correo electrónico y la capacidad del buzón.