

**UNIVERSIDAD NACIONAL DE INGENIERIA**

**FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA**

Proyecto:

**CyberPass, app orientada a la gestión y validación de contraseñas seguras**

Icono

Descripción generada automáticamente

**Fecha**:7 de diciembre 2024

**Integrantes**:

Espinoza Alvarado Franco

Soto Lulo Arisky

Tapia Taype Bryan

**Curso**: Programación orientada a objetos (BMA15)

Ciclo: Tercero

**Profesor**: Tello Canchapoma Yuri Oscar

***Documentación del proyecto Python***

1.Antecedentes del proyecto.

2.Analisis de contraseñas y su vulnerabilidad ante ataques de fuerza bruta.

3.Definiciones y especificaciones de requerimientos

3.1. Definición general del proyecto

3.1.1. Objetivos de la aplicación

3.1.2. Diagrama de Gantt del proyecto

3.2 Especificación de requerimientos del proyecto

3.3. Procedimientos de instalación y prueba

4.Arquitectura del sistema

a. Descripción jerárquica

b. Diagrama de módulos

c. Descripción individual de los módulos

d. Dependencias externas

5. Diseño del modelo de datos

6. Descripción de procesos y servicios ofrecidos por el sistema

7.Documentacion técnica -Especificación API

8.Aspectos relevantes

# 1.Antecedentes del proyecto.

Lesscop(2023): En su repositorio de Git Hub de nombre: ”Passord-genrator” presenta una una aplicación de generador de contraseñas descargable en los sistemas operativos Windows, Linux y macOS. Este repositorio tiene la licencia MIT y algunas instrucciones a seguir para descargar y entender el código.

Hazan y Ramesh(2023) :En su artículo titulado “An Application-Based Tool That Contains Both an Enhanced Password Generator and a Password Strength Checker” con un algoritmo escrito en Python, mostrado en el mismo artículo ,realizan el análisis de las métricas de seguridad de las contraseñas ingresadas por el usuario, haciendo uso ataques de diccionario, reglas de transformación, fuerza bruta y búsqueda de tablas a gran escala. Indicando al final el nivel de seguridad de dicha contraseña

Sursayrs007(2021): En su repositorio en Git Hub de nombre “WEB and API App for Random password Generator” nos proporciona muchos archivos tanto del generador de contraseñas en Java scrip, css y su posterior compactación en Pyhton de una página web generadora de contraseñas, este es un repositorio con licencia MIT.

# 2.Analisis de contraseñas y su vulnerabilidad ante ataques de fuerza bruta.

Fuerza bruta

Al respecto Corini (2022) lo define de la siguiente manera:

Un ataque de fuerza bruta ocurre cuando el atacante emplea determinadas técnicas para probar combinaciones de contraseñas con el objetivo de descubrir las credenciales de una potencial víctima y así lograr acceso a una cuenta o sistema. Existen diferentes tipos de ataque de fuerza bruta, como el “credential stuffing”, el ataque de diccionario, el ataque de fuerza bruta inverso o el ataque de password spraying. Generalmente, los ataques de fuerza bruta tienen mayor éxito en los casos en los que se utilizan contraseñas débiles o relativamente fáciles de predecir, En cuanto al hardware que se utiliza, cuanta más potencia se tenga más combinaciones por segundo se podrán evaluar, mientras que en lo que respecta el software, existen programas que son utilizados desde hace tiempo para aplicar la fuerza bruta en el descifrado de contraseña

Explosión combinacional

Al respecto Lopez(2023):

La implementación al parecer sencilla del ataque de fuerza bruta en realidad posee un gran problema matemático, ya que cada vez que se incrementa el abecedario y/o la longitud de la cadena, lo que sería respectivamente el número de valores de entrada “n” y/o la longitud de la cadena “l”, la cantidad de combinaciones que puede llegar surgir crece estrepitosamente; a tal punto que en algunos casos tardaría años en encontrarla, a este fenómeno se le denomina Explosión Combinacional

Entonces la expresión para el tiempo en que un porcesador se demore en obtener una contraseña es el siguiente:

**Tiempo VS Longitud**

**Longitud**

**tiempo**

**Hash y salt**

Un Hash BBVA (2023) lo define de la siguiente manera:

Un 'hash' es el resultado de aplicar una función matemática que toma una entrada y la transforma en una cadena de caracteres, generalmente una representación alfanumérica de longitud fija de los datos de entrada. Es decir, es un código formado por letras y números que resume o representa a un conjunto de datos determinados.

Por otro lado se puede decir que el salt “es un valor de un tamaño fijo que se introduce en el proceso de cifrado de tal forma que ante una misma contraseña existen tantos hashes como posibles valores pueda valer la semilla” Lopez(2013).

# 1.-Definiciones y especificaciones de requerimientos:

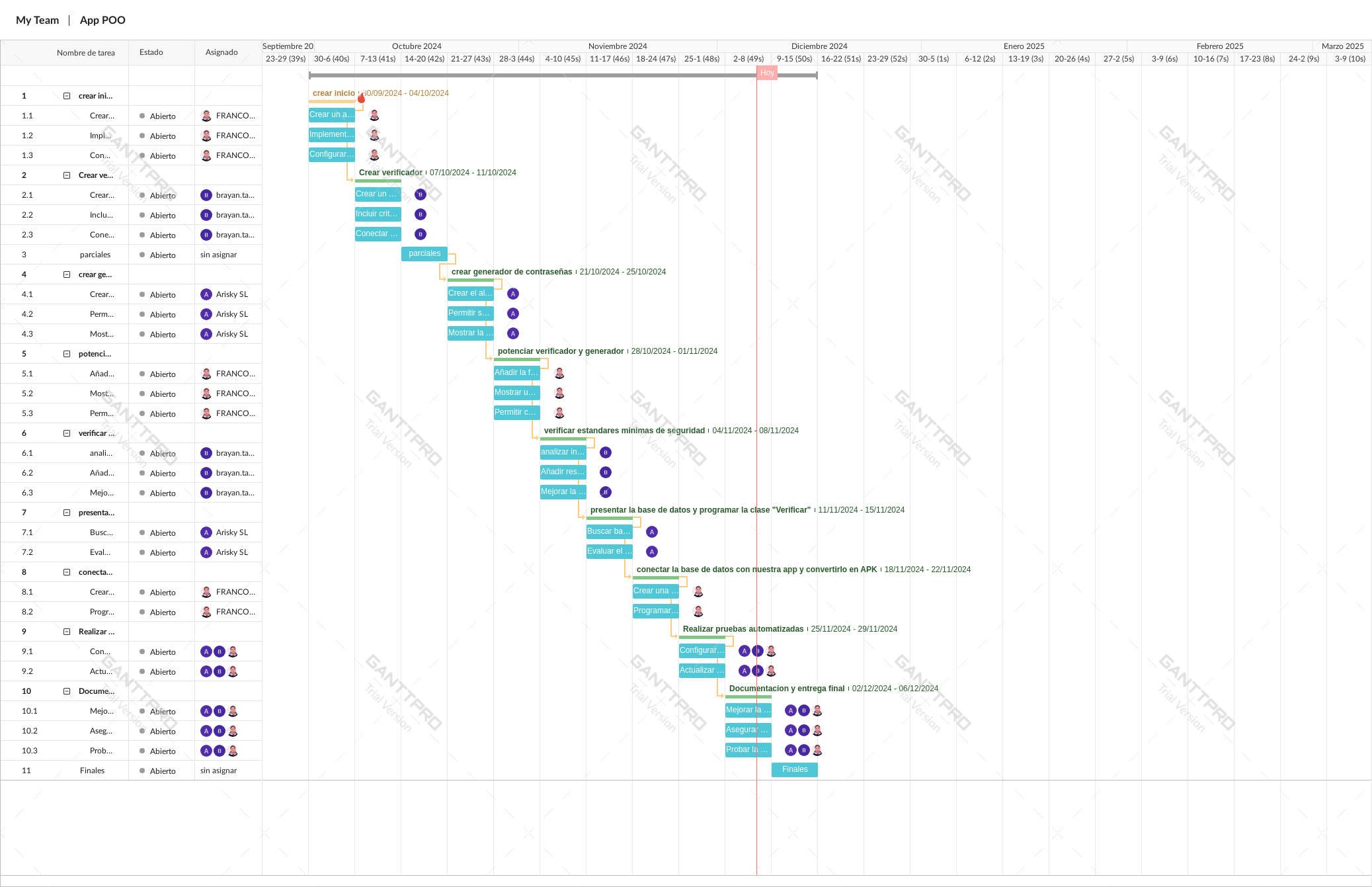
1.1) Definición general del proyecto**:**

Este proyecto fue creado con la intención de proporcionar la seguridad al usuario a la hora de usar sus contraseñas, usando un generador de contraseñas con las características necesarias para que sea segura así mismo también cuenta con una lista de contraseñas con las que nuestro algoritmo compara la contraseña que le proporciona el usuario y este detecta alguna similitud con esta lista y procederá a notificar la coincidencia con la lista.

Este proyecto también cuenta con un ataque de fuerza bruta que notifica al usuario si su contraseña puede tomar poco o bastante tiempo en ser descubierto ya habiendo cumplido con los estándares requeridos.

Objetivos

Esta aplicación fue creada con dos objetivos principales el primero de ellos es el poder establecer una contraseña segura para el usuario en este en este caso existiendo dos formas de poder hacerlo la primera es que el usuario le brinde a la aplicación la contraseña y esta verifica si es seguro o no una vez cumpliendo los requisitos mínimos para poder ser segura la contraseña le hace un ataque de fuerza bruta el cual indica que si la contraseña es fácilmente vulnerable dentro de un período largo o corto de tiempo, luego de esto el segundo como segundo objetivo también tenemos el poder crear contraseñas esta aplicación cuenta con la opción de crear una contraseña segura para el usuario que puede ser usada en cualquier momento esta contraseña se puede copiar y pegar en la en la primera opción qué es la de contraseña segura en la cual también verificará que cumpla con los estándares y le realizará un ataque de fuerza bruta la aplicación para ver si es completamente seguro o no.

* + 1. Diagrama de Gantt

1.2) Especificación de requerimientos del proyecto:

**a. Requerimientos Funcionales**

* **Almacenamiento de contraseñas:**
  + Almacenar contraseñas encriptadas utilizando un algoritmo de cifrado robusto.
  + Permitir al usuario agregar, modificar y eliminar contraseñas.
  + Ofrecer opciones de búsqueda y organización de las contraseñas.
* **Generación de contraseñas:**
  + Generar contraseñas aleatorias de alta complejidad, incluyendo mayúsculas, minúsculas, números y símbolos.
  + Permitir al usuario personalizar la longitud y la composición de las contraseñas generadas.
* **Evaluación de la seguridad de las contraseñas:**
  + Realizar ataques de fuerza bruta simulados para estimar el tiempo necesario para romper una contraseña.
* **Interfaz de usuario:**
  + Diseñar una interfaz intuitiva y fácil de usar.
  + Proporcionar una experiencia de usuario agradable y segura.

**b. Requerimientos No Funcionales**

* **Seguridad:**
  + Implementar medidas de seguridad robustas para proteger los datos de los usuarios (por ejemplo, cifrado, autenticación de dos factores).
  + Cumplir con las regulaciones de privacidad de datos aplicables.
* **Desempeño:**
  + La aplicación debe responder rápidamente a las acciones del usuario.
  + La evaluación de la seguridad de las contraseñas debe realizarse en un tiempo razonable.
* **Compatibilidad:**
  + La aplicación debe ser compatible con los sistemas operativos más comunes (iOS, Android, Windows, macOS).
* **Disponibilidad:**
  + Definir el tiempo de actividad esperado de la aplicación.

1.3.- Procedimientos de instalación y prueba:

. Procedimientos de Instalación de bibliotecas:

* **Pasos de instalación de la interfaz gráfica:** para proceder a la instalación de la interfaz gráfica que tiene el código que en este caso sería kivy. Para esto tenemos que usar los siguientes comandos:

python -m pip install --upgrade pip wheel setuptools

python -m pip install docutils pygments pypiwin32 kivy. deps. sdl2 kivy.deps.glew

python -m pip install kivy.deps.gstreamer

python –m pip install kivy

pip install kivymd

pip install kivymd==0.104.2

(documentación de Python)

* **Dependencias:** para poder usar este código es necesario descargar algunas librerías para poder correr el código en este caso usamos la librería de Levenshtein, esta librería le permite al código hacer una comparación midiendo la diferencia entre dos palabras, lo cual hace a nuestro código aún más preciso a la hora de hallar coincidencias con nuestra base de datos.
* **Configuraciones iniciales:** para que el usuario pueda hacer uso de esta aplicación primero deberá crear un usuario con su respectiva contraseña, habiéndose registrado el usuario podrá hacer usa de la aplicación que este caso nos ofrece la opción de generar una contraseña o verificar la contraseña que desee el usuario.

**Pruebas:**

**. pruebas automatizadas:** Para poder realizar esta prueba creamos un código aparte que corrobore si el código es totalmente funcional en este caso la función del código es hacerse pasar como un usuario y verificar cada opción que proporciona la aplicación. Esto se refiere al guardado del usuario en la base de datos

# 2. Arquitectura del sistema:

Descripción jerárquica:

1. **Interfaz de usuario (UI)**
   * **ScreenManager**: Gestiona las pantallas y la navegación de la aplicación, implementando transiciones deslizantes ( SlideTransition).
   * Principales características:
     + **LoginScreen**: Pantalla de inicio de sesión.
     + **RegisterScreen**: Pantalla de registro de nuevos usuarios.
     + **Pantalla de bienvenida**: Pantalla de bienvenida tras autenticación exitosa.
     + **VerifyPasswordScreen**: Pantalla para verificar la seguridad de contraseñas.
     + **GeneratePasswordScreen** : Pantalla para generar contraseñas aleatorias seguras.
   * Componentes de diseño:
     + Botones personalizados ( CyberButton).
     + Estilo visual basado en una paleta de colores oscuros con acentos cian.
2. **Backend**
   * **Base de datos** (Database):
     + Se utiliza SQLite para gestionar usuarios.
     + Funciones principales:
       - **create\_table()** : Crea la tabla userssi no existe.
       - **hash\_password(contraseña)**: Hashea contraseñas con SHA-256.
       - **add\_user (nombre de usuario, contraseña)** : Agrega un usuario nuevo a la base de datos.
       - **verificar\_usuario (nombre de usuario, contraseña)** : Verifica si las credenciales coinciden con un usuario existente.
   * **Lógica de seguridad**:
     + **is\_password\_secure ()**: Verifica que la contraseña no sea común ni similar a contraseñas públicas.
     + **is\_password\_secure2()**: Asegura que la contraseña cumpla con requisitos mínimos (longitud, caracteres especiales, etc.).
     + **is\_password\_secure3()** : Simula un ataque de fuerza bruta para medir la vulnerabilidad de una contraseña.
   * **Generación de contraseñas**:
     + **generate\_random\_password ()**: Genera contraseñas seguras con una combinación aleatoria de caracteres (mayúsculas, minúsculas, números y símbolos).
3. **Funcionalidades principales**
   * **Inicio de sesión**:
     + Auténtica al usuario mediante la base de datos.
     + Cambia a la pantalla de bienvenida al inicio de sesión exitosa.
   * **Registro**:
     + Permite a los usuarios crear una cuenta nueva.
     + Valida que los campos estén completos y que el usuario no existe ya.
   * **Generación de contraseñas seguras**:
     + Genera y muestra una contraseña aleatoria.
     + Ofrece opción para copiar al portapapeles.
   * **Verificación de contraseñas**:
     + Evalúa contraseñas según reglas de seguridad y bases de datos de contraseñas comunes.
     + Simula ataques de fuerza bruta para determinar vulnerabilidades.
   * **Notificaciones**:
     + Úselo toastpara mensajes breves.
     + Utilice MDDialogpara alertas más detalladas.
4. **Archivos externos**
   * **Comunes.txt**: Lista de contraseñas comunes utilizadas para evaluar similitudes.
   * **Ataque.txt**: Lista de contraseñas utilizadas en ataques de fuerza bruta.
5. **Dependencias**
   * **Bibliotecas principales**:
     + Kivyy KivyMDpara la interfaz gráfica.
     + sqlite3para la gestión de base de datos.
     + hashlibpara hashing de contraseñas.
     + Levenshteinpara medir la similitud entre cadenas.
   * **Módulos estándar** :
     + re, random, string, threading, time.

Descripción individual de los módulos:

**. Aplicación de inicio de sesión**

* **Propósito** : Controlador principal de la aplicación, gestiona la navegación entre las diferentes pantallas y la inicialización de la interfaz gráfica.
* **Funciones clave** :
  + build(): Configura la aplicación y define el gestor de pantallas ( ScreenManager).
  + on\_generate\_password(): Llama a la pantalla de generación de contraseñas.
  + on\_verify\_password(): Navega a la pantalla de verificación de contraseñas.
  + on\_save\_password(): Implementa lógica para guardar contraseñas generadas.

**. Base de datos**

* **Propósito**: Proporciona funcionalidades para la gestión de usuarios en la base de datos.
* **Funciones clave**:
  + create\_table(): Crea la tabla de usuarios si no existe.
  + hash\_password(password): Genera un hash seguro para las contraseñas.
  + add\_user(username, password): Inserta nuevos usuarios en la base de datos.
  + verify\_user(username, password): Verifica credenciales en la base de datos.

**. Pantalla de inicio de sesión**

* **Propósito**: Permite a los usuarios iniciar sesión en la aplicación.
* **Características**:
  + Interactúa con el módulo **Database** para autenticar usuarios.
  + Usa un formulario para ingresar credenciales.

**. Pantalla de registro**

* **Propósito**: Gestiona el registro de nuevos usuarios.
* **Características**:
  + Valida que los campos sean correctos y que el usuario no existe ya.
  + Guarda las credenciales mediante el módulo **Database**.

. Pantalla de bienvenida

* **Propósito**: Muestra un mensaje de bienvenida tras un inicio de sesión exitosa.
* **Características**:
  + Actúa como una pantalla intermedia tras la autenticación.
  + Puede incluir accesos directos a otras funciones, como generación de contraseñas.

**. Verificar contraseña Pantalla**

* **Propósito**: Permite verificar la fortaleza de contraseñas.
* **Funciones clave**:
  + is\_password\_secure(password): Compara contraseñas con una lista de contraseñas comunes.
  + is\_password\_secure2(password): Valida reglas de complejidad (longitud mínima, símbolos, etc.).
  + is\_password\_secure3(password): Realiza simulaciones de ataques de fuerza bruta para evaluar vulnerabilidades.

**Generar contraseña de pantalla**

* **Propósito**: Facilita la generación de contraseñas seguras.
* **Funciones clave**:
  + generate\_random\_password (): Crea contraseñas seguras con caracteres aleatorios.
  + Copia las contraseñas al portapapeles para su uso inmediato.

**. Seguridad y Gestión de Contraseñas**

* **Propósito**: Proporciona funcionalidades avanzadas para la verificación y creación de contraseñas seguras.
* **Submódulos**:
  + **Administrador de contraseñas**:
    - Centraliza la lógica de seguridad y generación de contraseñas.
    - Métodos:
      * verify\_password\_strength(password): Evalúa la fortaleza de una contraseña.
      * save\_password\_to\_database(password): Guarda contraseñas generadas en la base de datos.
  + **Base de datos de contraseñas**:
    - Gestiona el almacenamiento persistente de contraseñas.
    - Métodos:
      * save\_password\_to\_db(password):Guarda contraseñas.
      * load\_password\_from\_db():Recuperar contraseñas.
  + **Administrador del portapapeles**:
    - Función principal:
      * copy\_to\_clipboard(password): Copia una contraseña generada al portapapeles.
  + **Administrador de archivos de contraseñas**:
    - Proporciona funciones para interactuar con archivos de texto.
    - Métodos:
      * load\_common\_passwords (): Carga una lista de contraseñas comunes para evaluaciones.
      * save\_common\_passwords(passwords): Guarda listas de contraseñas en archivos.

Dependencias externas

**Kivy**:

* + **Propósito**: Framework principal para crear interfaces gráficas.
  + **Uso en tu aplicación**:
    - Gestión de pantallas con ScreenManager.
    - Creación de widgets como botones, etiquetas y layouts.
  + Instalación:

bash

Copiar código

pip install kivy

**KivyMD**:

* + **Propósito**: Extensión de Kivy para implementar Material Design en las aplicaciones.
  + **Uso en tu aplicación**:
    - Widgets como MDDialog, toast y estilos visuales modernos.
  + Instalación:

bash

Copiar código

pip install kivymd

**SQLite (sqlite3)**:

* + **Propósito**: Sistema de base de datos liviano incorporado en Python.
  + **Uso en tu aplicación**:
    - Gestión de usuarios y contraseñas (almacenamiento y consulta).
  + **Nota**: Es una biblioteca estándar de Python, por lo que no requiere instalación adicional.

**hashlib**:

* + **Propósito**: Biblioteca estándar para hashing seguro de contraseñas.
  + **Uso en tu aplicación**:
    - Hashear contraseñas con el algoritmo SHA-256.
  + **Nota**: Es parte del módulo estándar de Python.

**Levenshtein**:

* + **Propósito**: Calcular la distancia de Levenshtein (similitud entre cadenas).
  + **Uso en tu aplicación**:
    - Verificar similitudes entre contraseñas y contraseñas comunes.
  + Instalación:

bash

Copiar código

pip install python-Levenshtein

**Módulos estándar de Python**:

* + **random**: Generar contraseñas aleatorias.
  + **string**: Crear caracteres para la generación de contraseñas.
  + **re**: Validaciones mediante expresiones regulares.
  + **threading**: Manejar procesos concurrentes (por ejemplo, ataques de fuerza bruta simulados).
  + **time**: Medir tiempos en simulaciones o procesos.

(documentación de Python)

# Diseño del modelo de datos

**clase: users**

* **Propósito**: Almacena la información de los usuarios registrados en la aplicación.
* **Estructura:**

Tabla

Descripción generada automáticamente

**passwords**

* **Propósito**: Gestiona las contraseñas generadas por los usuarios para su uso o almacenamiento.
* Estructura:

Tabla

Descripción generada automáticamente

**common\_passwords**

* **Propósito**: Almacena una lista de contraseñas comunes para la verificación de seguridad.
* **Estructura:**

Tabla

Descripción generada automáticamente

# Descripción de procesos y servicios ofrecidos por el sistema.

Servicios o tareas que el sistema ofrece

* **Gestión de usuarios:** Los usuarios tienen dos opciones al estar en la primera pantalla de la interfaz: Inicio de sesión o registrase como nuevo usuario, para el primer paso se debe brindar un usuario y una contraseña, el cual será comparada con la base de datos existente, el sistema almacena la información en una base de datos **SQLite**. Para registrarse como usuario nuevo se debe escribir primero el nombre de un usuario y luego una contraseña, antes de guardar las credenciales la app verificara si el nombre de usuario de este es único en la base de datos, una vez confirmado esto se pasa a encriptar la contraseña con el algoritmo **SHA-256** para luego guardarlo en la base de datos.
* **Generador de contraseñas seguras:** Una vez que el usuario se haya registrado o iniciado sección tiene la opción de generar contraseñas, estas contraseñas cumplen con los más altos **estándares de seguridad**, con una longitud que va de 12-16 caracteres, estas están compuestas de: de **mayúsculas, minúsculas, números y caracteres especiales**. Posterior a ello se tiene la posibilidad de **copiar la contraseña en el portapapeles**, esto sirve para que el usuario pueda utilizarlo donde él lo necesite.
* **Verificación de contraseñas seguras:** Esta opción le permite al usuario comprobar que tan segura es la contraseña que él tiene en mente, primero se le hace una **revisión** si la posible contraseña está en una base de datos de contraseñas usuales: comunes.txt, contemplando un distanciamiento mínimo de tres ediciones, para ello se usa la distancia de levenshetein.Después se le hace una **verificación** en cuanto a la longitud y contención de los cuatro tipos de caracteres antes mencionados. Para finalmente hacerle un **ataque de diccionario** usando el documento ataque.txt. Una vez pasada estas pruebas se dice que la contraseña es segura.
* **Interacción con el usuario:** Para este apartado se usan dos librerías: **Kivy** y **kyvyMD** para crear una experiencia atractiva y fácil de usar, con ello contemplamos las siguientes pantallas:
* **LoginScreen**: El usuario puede iniciar sesión con su nombre de usuario y contraseña.
* **RegisterScreen.** Permite el registro de nuevos usuarios mediante la creación de un nombre de usuario y una contraseña.
* **WelcomeScreen**: Después de iniciar sesión, se muestra un mensaje de bienvenida y ofrece opciones adicionales como generar o verificar una contraseña.
* **VerifyPasswordScreen**: Permite a los usuarios verificar la seguridad de una contraseña ingresada.
* **GeneratePasswordScreen**: Facilita la generación de una nueva contraseña aleatoria con requisitos de seguridad específicos.

Para interactuar el usuario cuenta con botones (CyberButton), campos de texto para ingresar datos y una retroalimentación visual con notificaciones mediante (MDToast y MDDialog)

* **Funcionalidades de dialogo y notificaciones:** Para comunicar la app con el usuario se usan mensajes de retroalimentación como diálogos y notificaciones, para ello se usan los siguientes métodos:
* **MDDialog**: Se utiliza para mostrar mensajes emergentes que informan al usuario sobre el estado de la aplicación (errores, éxitos, advertencias, etc.).
* **MDToast:** Utilizado para mostrar mensajes breves y no intrusivos. Por ejemplo, cuando se genera una contraseña o se copia al portapapeles.
* **Funcionalidad especial de "Ataque de Fuerza Bruta":** Cuando un usuario verifica la seguridad de una contraseña, si esta es vulnerable a un ataque de fuerza bruta, se muestra un diálogo que indica que el ataque está siendo procesado y el tiempo estimado de vulnerabilidad.
* **Gestión de base de datos:** Para la base de datos se usa SQLite, esta se usa para guardar los usuarios y contraseñas. Primero la app crea la tabla users en la cual están los nombres de los usuarios y las contraseñas. Cuando un usuario se registra se verifica la existencia única de usuario primero y después se guarda la contraseña encriptada usando el algoritmo de hash Sha-256. Para iniciar sección lo que se hace es comparar el usuario y el hash de la contraseña guardada en la base de datos. Y finalmente se cierra la conexión de la base de datos cuando la aplicación se cierra (on stop).
* **Rendimiento:** Para la fluidez de la interfaz se usan kyvy y kyvyMD. Además de ello se hace uso de hilos ayuda a que la app evite congelarse.
* **Funcionalidades especiales:** Aquí contemplamos a las siguientes funciones:
* **Generación de contraseñas aleatorias**. Aquí el Sistema genera contraseñas aleatorias cumpliendo los requisitos mínimos de seguridad, estas son mostradas en la interfaz y pueden ser copiados para su uso inmediato.
* **Copia al portapapeles:** Ofrece la funcionalidad de copiar las contraseñas generadas directamente al portapapeles del dispositivo, simplificando su uso en otras aplicaciones o servicios
* **Interacción con archivos de contraseñas comunes:** El sistema utiliza archivos externos como comunes.txt y Ataque.txt. Las cuales son usadas para revisar si las contraseñas ingresadas son fácilmente atacables.

Diagrama de UML:

**Diagrama

Descripción generada automáticamente**

# Documentación técnica - especificación API

**1.Clase LoginApp**

**Propósito:**

Controla la lógica principal de la aplicación, incluyendo la gestión de pantallas, la autenticación de usuarios y la generación de contraseñas.

**Métodos:**

**\_\_init\_\_(self, \*\*kwargs)**

* **Propósito**: Inicializa la aplicación y la conexión a la base de datos.
* **Argumentos**:
  + kwargs (dict): Parámetros adicionales para la inicialización de la clase base.
* **Respuesta**: Ninguna.

**build(self)**

* **Propósito**: Configura la apariencia de la aplicación utilizando KivyMD.
* **Argumentos**: Ninguno.
* **Respuesta**:
  + (Widget): Retorna el widget raíz que se usará para la interfaz.

**show\_dialog(self, title, text)**

* **Propósito**: Muestra un cuadro de diálogo con un mensaje de texto.
* **Argumentos**:
  + title (str): Título del cuadro de diálogo.
  + text (str): Texto a mostrar en el cuadro de diálogo.
* **Respuesta**: Ninguna.

**show\_result\_dialog(self, title, text)**

* **Propósito**: Muestra un cuadro de diálogo con un resultado, como la seguridad de la contraseña.
* **Argumentos**:
  + title (str): Título del cuadro de diálogo.
  + text (str): Texto a mostrar en el cuadro de diálogo.
* **Respuesta**: Ninguna.

**copy\_to\_clipboard(self)**

* **Propósito**: Copia la contraseña generada al portapapeles.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**login(self)**

* **Propósito**: Verifica las credenciales del usuario e inicia sesión si son correctas.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**register\_user(self)**

* **Propósito**: Registra un nuevo usuario en la base de datos.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**generate\_random\_password(self)**

* **Propósito**: Genera una contraseña aleatoria que cumpla con los criterios de seguridad.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**verify\_password(self)**

* **Propósito**: Verifica si la contraseña ingresada cumple con los requisitos de seguridad.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**is\_password\_secure(self, password)**

* **Propósito**: Verifica si una contraseña no está en una lista de contraseñas comunes y si es suficientemente segura según su distancia con contraseñas comunes.
* **Argumentos**:
  + password (str): Contraseña a verificar.
* **Respuesta**:
  + (bool): Retorna True si la contraseña es segura, False en caso contrario.

**is\_password\_secure2(self, password)**

* **Propósito**: Verifica si la contraseña cumple con los requisitos mínimos de seguridad (longitud, mayúsculas, minúsculas, números y caracteres especiales).
* **Argumentos**:
  + password (str): Contraseña a verificar.
* **Respuesta**:
  + (bool): Retorna True si la contraseña cumple con los requisitos, False en caso contrario.

**is\_password\_secure3(self, password)**

* **Propósito**: Realiza un ataque de fuerza bruta contra la contraseña utilizando un archivo de contraseñas comunes y mide el tiempo que tarda en vulnerarla.
* **Argumentos**:
  + password (str): Contraseña a verificar.
* **Respuesta**:
  + (float o None): Retorna el tiempo en segundos que tarda en vulnerar la contraseña, o None si no es vulnerable rápidamente.

**password\_in\_thread(self, password)**

* **Propósito**: Ejecuta el proceso de verificación de la contraseña en un hilo separado y muestra el resultado al usuario.
* **Argumentos**: password (str): Contraseña a verificar.
* **Respuesta**: Ninguna.

**switch\_to\_verify\_password(self)**

* **Propósito**: Cambia a la pantalla de verificación de contraseñas.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**switch\_to\_generate\_password(self)**

* **Propósito**: Cambia a la pantalla de generación de contraseñas.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**on\_stop(self)**

* **Propósito**: Cierra la conexión con la base de datos cuando la aplicación se cierra.
* **Argumentos**: Ninguno.
* **Respuesta**: Ninguna.

**2. Funciones Auxiliares**

**Levenshtein.distance(str1, str2)**

* **Propósito**: Calcula la distancia de Levenshtein entre dos cadenas, útil para determinar qué tan similar es una contraseña a una común.
* **Argumentos**:
  + str1 (str): Primer texto para comparar.
  + str2 (str): Segundo texto para comparar.
* **Respuesta**:
  + (int): Distancia de Levenshtein entre las dos cadenas.

**Clipboard.copy(text)**

* **Propósito**: Copia el texto al portapapeles.
* **Argumentos**:
  + text (str): Texto a copiar.
* **Respuesta**: Ninguna.

# Aspectos relevantes

**Invocación del Programa**

La ejecución del programa se realiza de la siguiente manera:

Bash: python password\_generator.py

**Parámetros:**

* **longitud (opcional)**: Un número entero que define la longitud de la contraseña generada. El valor por defecto es **12**.
  + **Ejemplo**: python password\_generator.py --longitud 16
* **mayusculas (opcional)**: Un valor booleano (True o False) que determina si la contraseña generada debe contener letras mayúsculas. El valor por defecto es **True**.
  + **Ejemplo**: python password\_generator.py --mayusculas False
* **minusculas (opcional)**: Un valor booleano (True o False) que indica si se deben incluir letras minúsculas en la contraseña generada. El valor por defecto es **True**.
  + **Ejemplo**: python password\_generator.py --minusculas False
* **numeros (opcional)**: Un valor booleano (True o False) que especifica si la contraseña generada debe contener números. El valor por defecto es **True**.
  + **Ejemplo**: python password\_generator.py --numeros False
* **especiales (opcional)**: Un valor booleano (True o False) que controla si se deben incluir caracteres especiales (como @, #, etc.). El valor por defecto es **True**.
  + **Ejemplo**: python password\_generator.py --especiales False

**Comportamiento por Defecto:** Si no se especifican los parámetros opcionales, la aplicación generará una contraseña con las siguientes configuraciones por defecto:

* Longitud de 12 caracteres.
* Incluirá mayúsculas, minúsculas, números y caracteres especiales.

**Estrategia General:**

1. **Recogida de parámetros:** El primer paso en la ejecución del programa es la captura de los parámetros opcionales desde la línea de comandos. Los parámetros controlan las características de la contraseña generada (longitud, tipos de caracteres).
2. **Generación de caracteres disponibles:** Basado en los parámetros proporcionados por el usuario, se genera una lista con los caracteres disponibles para la contraseña. Si se activa la opción de incluir mayúsculas, la lista incluirá letras mayúsculas, lo mismo para las minúsculas, números y caracteres especiales.
3. **Generación de la contraseña:** Usando el módulo random, la contraseña es generada seleccionando aleatoriamente caracteres de la lista preparada previamente.
4. **Visualización y copia:** Finalmente, se muestra la contraseña generada al usuario en la interfaz y se le ofrece la opción de copiarla al portapapeles.

**3. Tipos de Datos Abstractos (TDAs)**

**TDA GeneradorContraseñas:**

Este TDA maneja el proceso de generación de contraseñas y encapsula toda la lógica relacionada.

**Representación:** El TDA utiliza listas para almacenar los diferentes tipos de caracteres disponibles, y un string para representar la contraseña generada.

* **Métodos principales:**
  + \_\_init\_\_(self, longitud=12, mayusculas=True, minusculas=True, numeros=True, especiales=True):
    - Inicializa el generador con los parámetros dados.
    - Si no se proporciona un parámetro, se toma el valor por defecto.
  + generar\_contrasena(self):
    - Genera una contraseña aleatoria usando los caracteres disponibles.
  + copiar\_al\_portapapeles(self):
    - Copia la contraseña generada al portapapeles del sistema.
  + validar\_contrasena(self):
    - Verifica si la contraseña generada cumple con los requisitos de los parámetros (longitud, caracteres).

**Limitaciones de la Representación:**

* La longitud de la contraseña está limitada por el valor entero proporcionado como parámetro.
* Los caracteres especiales que se pueden incluir están limitados a los definidos en el código.
* El proceso de validación asegura que la contraseña cumpla con los criterios de tipo de caracteres seleccionados, pero no verifica la robustez de la contraseña de manera profunda (por ejemplo, no evalúa si la contraseña es fácilmente adivinable).

**4. Conclusiones**

**Complicaciones Encontradas:**

* **Manejo de caracteres especiales:** La inclusión de caracteres especiales requirió una implementación cuidadosa, ya que algunos caracteres pueden ser interpretados como símbolos especiales por el sistema operativo.
* **Validación de parámetros:** El manejo adecuado de los parámetros opcionales fue clave para asegurar que la aplicación tuviera un comportamiento predecible, incluso si se omitían algunos parámetros.

**Políticas Adoptadas para Resolución:**

* Se implementaron valores por defecto razonables para los parámetros opcionales, de modo que el programa pueda ejecutarse de manera funcional sin necesidad de intervención del usuario.
* Para la validación de los parámetros, se implementó un control de excepciones para garantizar que los valores proporcionados sean correctos y evitar errores en la ejecución.

**Restricciones al Problema Original:**

* El sistema no realiza una validación avanzada de seguridad para las contraseñas generadas, como verificar si la contraseña es de difícil adivinación (esto podría hacerse agregando un análisis de entropía, por ejemplo).
* Solo se soporta la generación de contraseñas simples (sin complejidades adicionales como combinar contraseñas preexistentes).

**Casos Particulares:**

* Si el usuario no especifica ningún parámetro, se genera una contraseña de 12 caracteres con todos los tipos de caracteres activados.
* Si se seleccionan todos los parámetros como False, el generador generará contraseñas sin caracteres, lo que podría no ser válido, pero es un comportamiento esperado en esta configuración.

**Experiencia Obtenida:**

* El proyecto ayudó a comprender cómo manejar la entrada del usuario y la validación de parámetros, lo cual es crucial en el desarrollo de aplicaciones interactivas. Además, se reforzó el entendimiento sobre la generación de contraseñas seguras y cómo integrar funcionalidades de copia al portapapeles y validación de parámetros en una interfaz de usuario.

**REFERENCIA:**

Universidad Nacional del Sur(2017): Guía para documentación de proyectos de software, recuperado de:

<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj_07mF5YSKAxUNOrkGHXx2ELUQFnoECCYQAQ&url=https%3A%2F%2Fcs.uns.edu.ar%2F~ldm%2Fmypage%2Fdata%2Foc%2Finfo%2Fguia_para_la_documentacion_de_proyectos_de_software.pdf&usg=AOvVaw1W9SD7FcwrN8TRHS9B1vNJ&opi=89978449>

BBVA(2023):De la verificación de contraseña a la firma electrónica: el secreto está en el 'hash, recuperado el 11 de noviembre de :

<https://www.bbva.com/es/innovacion/de-la-verificacion-de-contrasena-a-la-firma-electronica-el-secreto-esta-en-el-hash/>

Corini.A(2022):Validación del modelo criptográfico de punto a punto basado en laobsolescencia programada usando ataque de fuerza bruta, recuperado de: <https://ciencialatina.org/index.php/cienciala/article/view/2269>

Lopez.V(2023):Papel de la explosión combinacional en ataques de fuerza bruta, recuperado de: <https://revistas.unisimon.edu.co/index.php/innovacioning/article/view/2069>

Lesscorp(2023): ”password generator” recuperado de: <https://github.com/lesskop/password-generator?tab=readme-ov-file>

Hazan y Ramesh(2023):An Application-Based Tool That Contains Both an Enhanced Password Generator and a Password Strength Checker” recuperado de <https://www.proquest.com/docview/2921105766?pq-origsite=gscholar&fromopenview=true&sourcetype=Scholarly%20Journals>

Sursays007(2021): ”WEB and API App for Random password Generator” recuperado de: <https://github.com/suryasr007/rpg-web/blob/main/app.py>