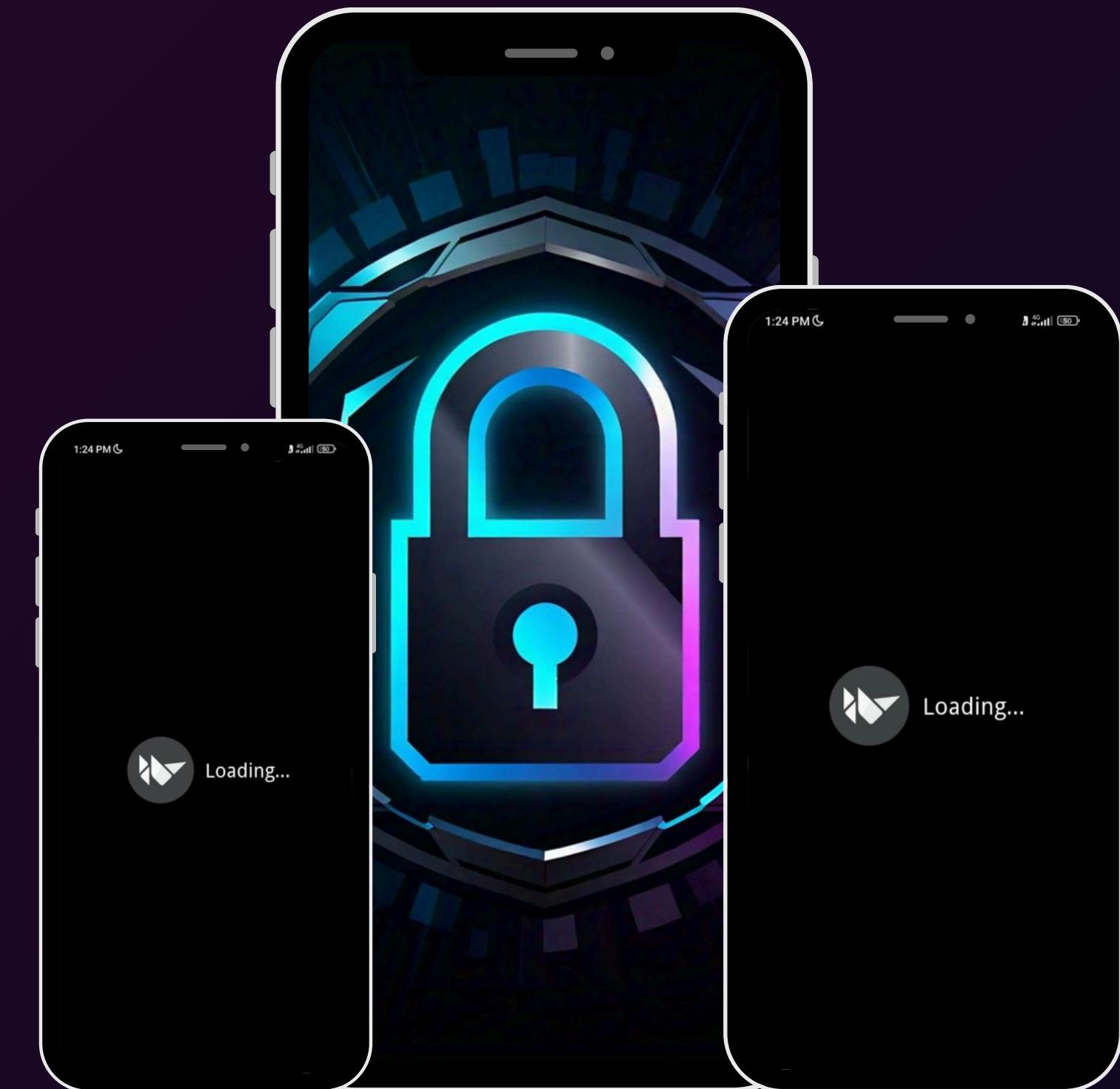
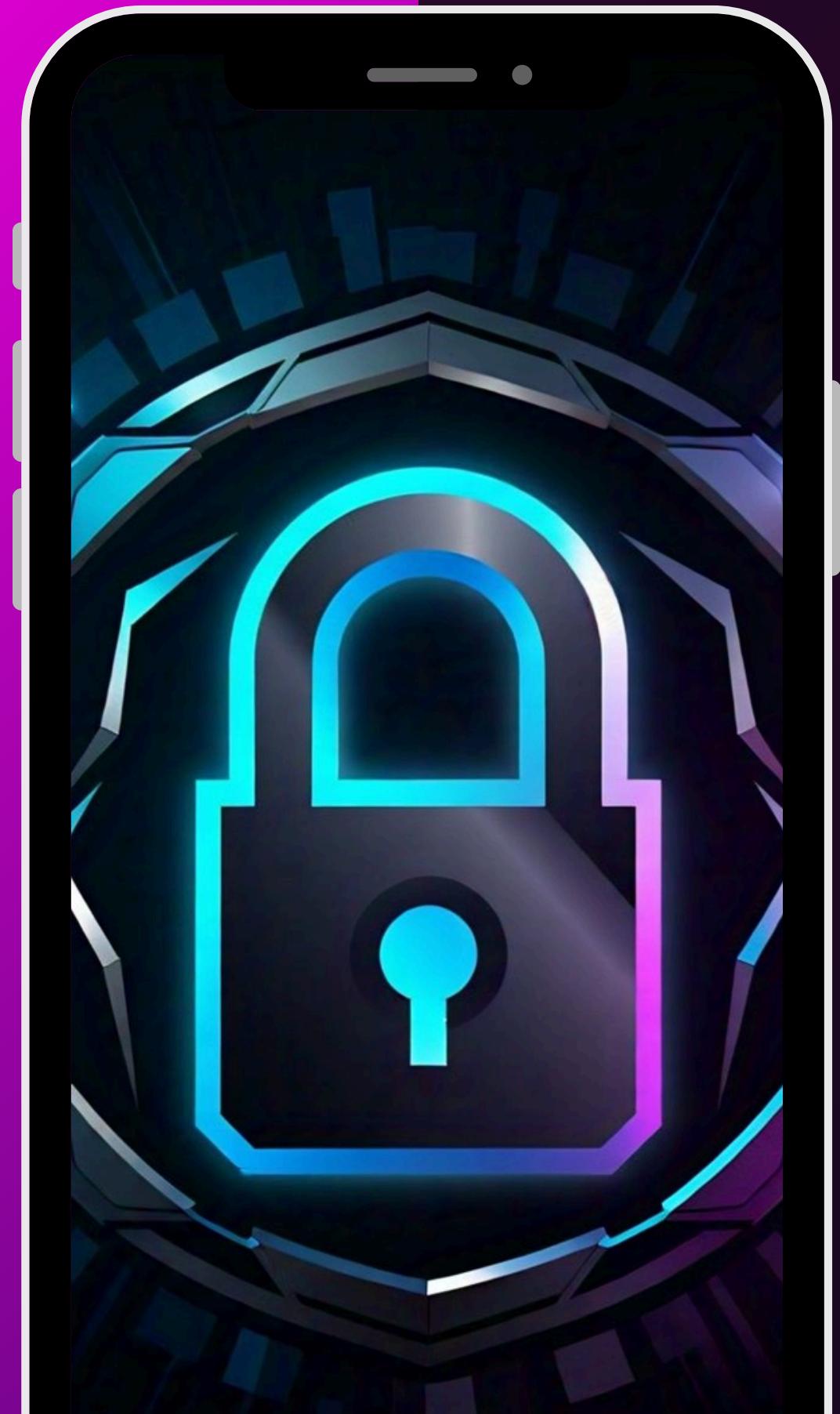




# cyber pass



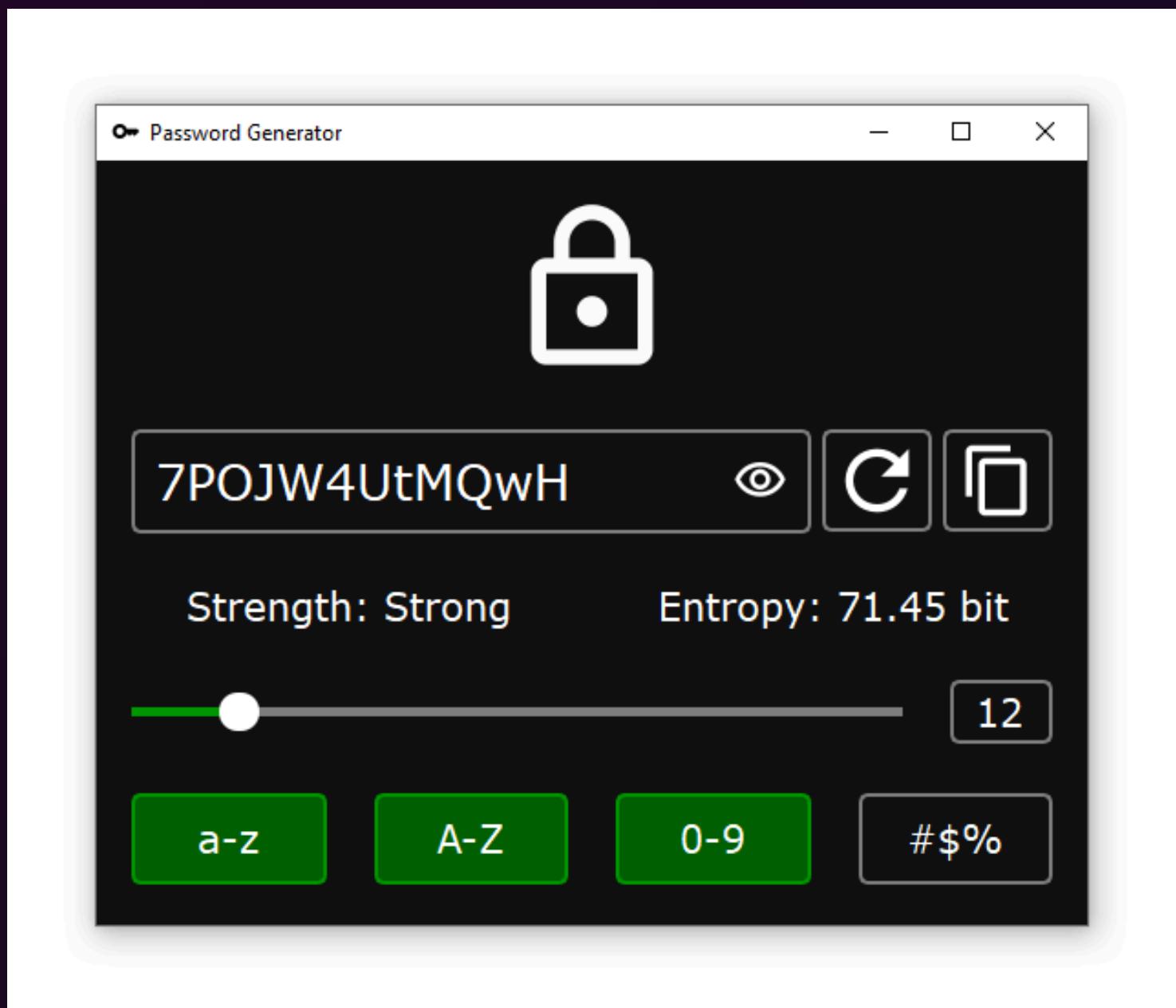


# ¿que es cyber pass?

cyber pass es una aplicacion que le proporciona al usuario una mayor seguridad en cuanto a contraseñas se trata , esta aplicacion le permite al usuario verificar si la contrasela cumple con los estandares minimos de seguridad realizando así tambien un ataque de fuerza bruta a la contraseña para saber si es totalmente segura y si es posible ser hackeada en un corto periodo de tiempo .

## Antecedentes

Lesscop(2023): En su repositorio de Git Hub de nombre "Passord-generator" presenta 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.





# objetivos

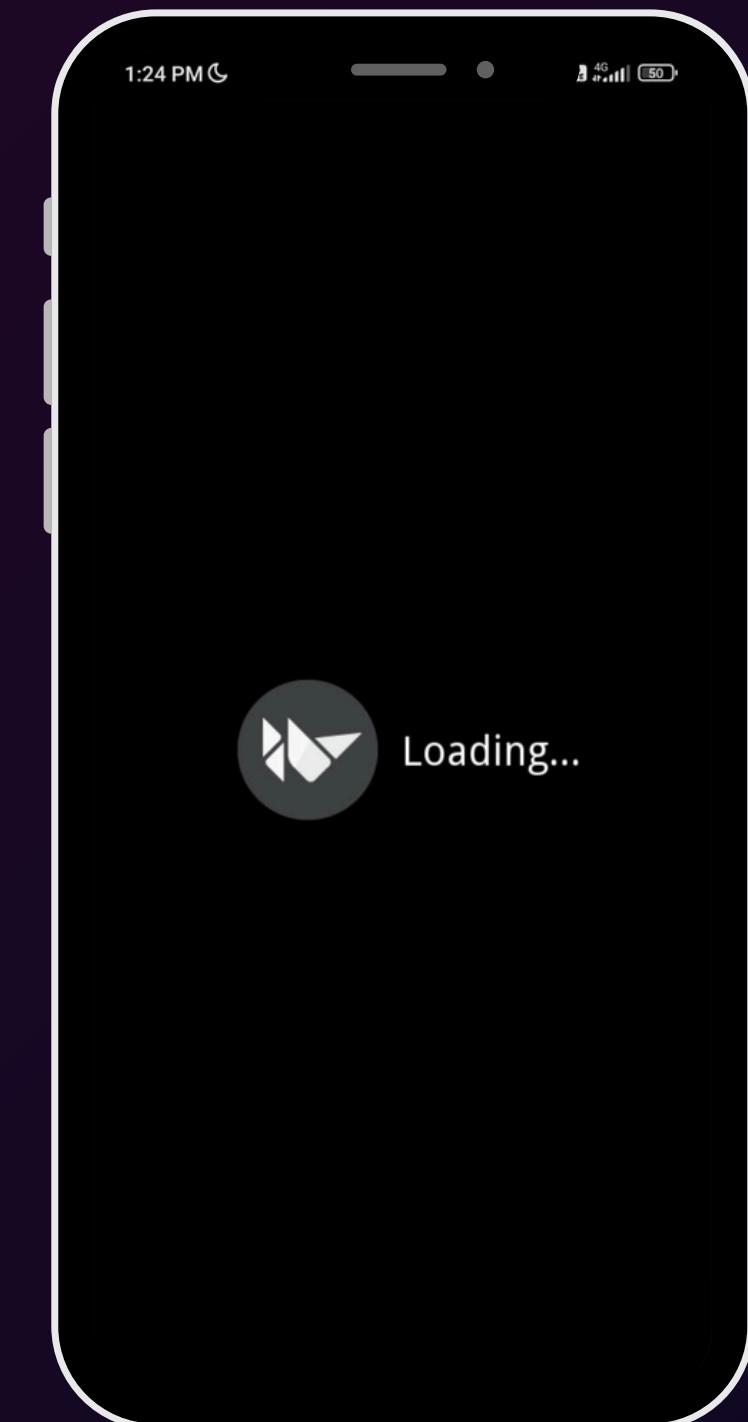
---

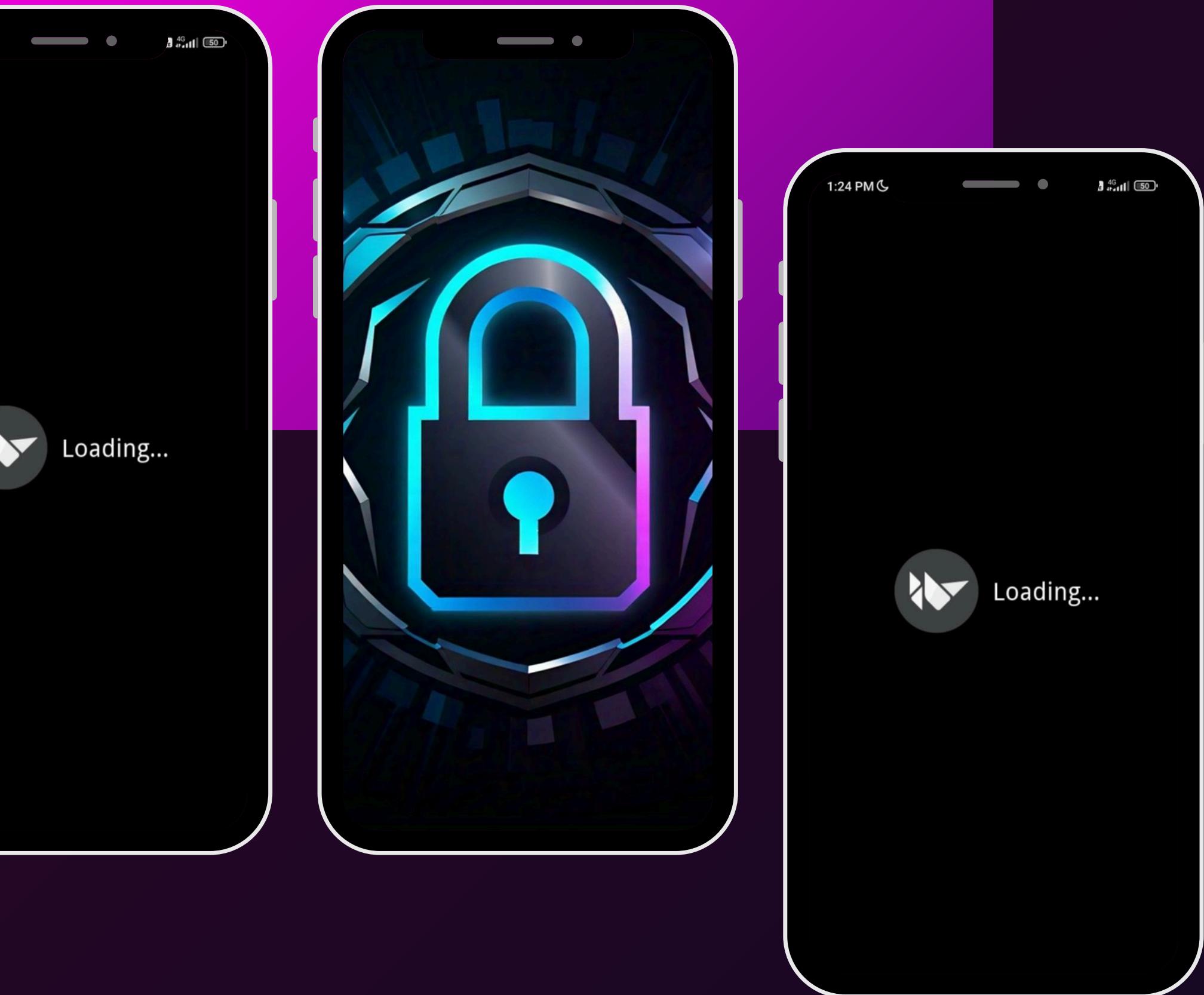
01

avisar al usuario si la contraseña que proporciona  
es completamente segura .

02

proporcionar al usuario una contraseña generada  
por la misma aplicación .





## características

01

para nuestra interfaz grafica usamos kivymd esta interfaz le permite al usuario un menu mas accesible y comprensible .

02

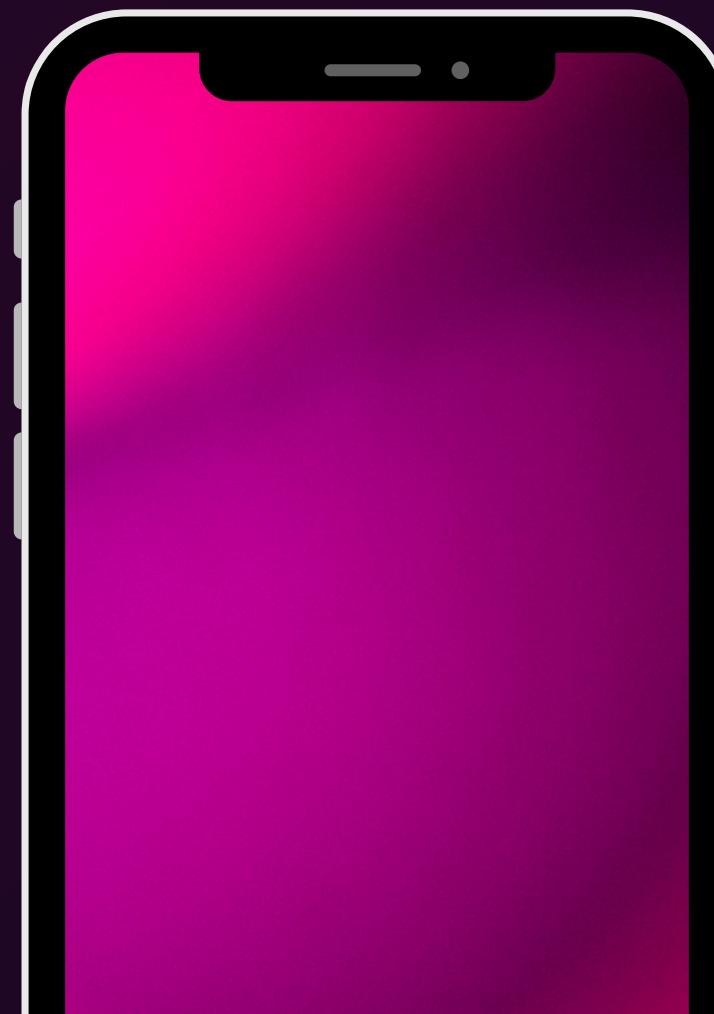
esta aplicacion cuenta con opcion muy particular la cual es un ataque de fuerza bruta la cual le indica al usuario si tomaria bastante o poco tiempo en ser hackeada o hallada la contraseña .



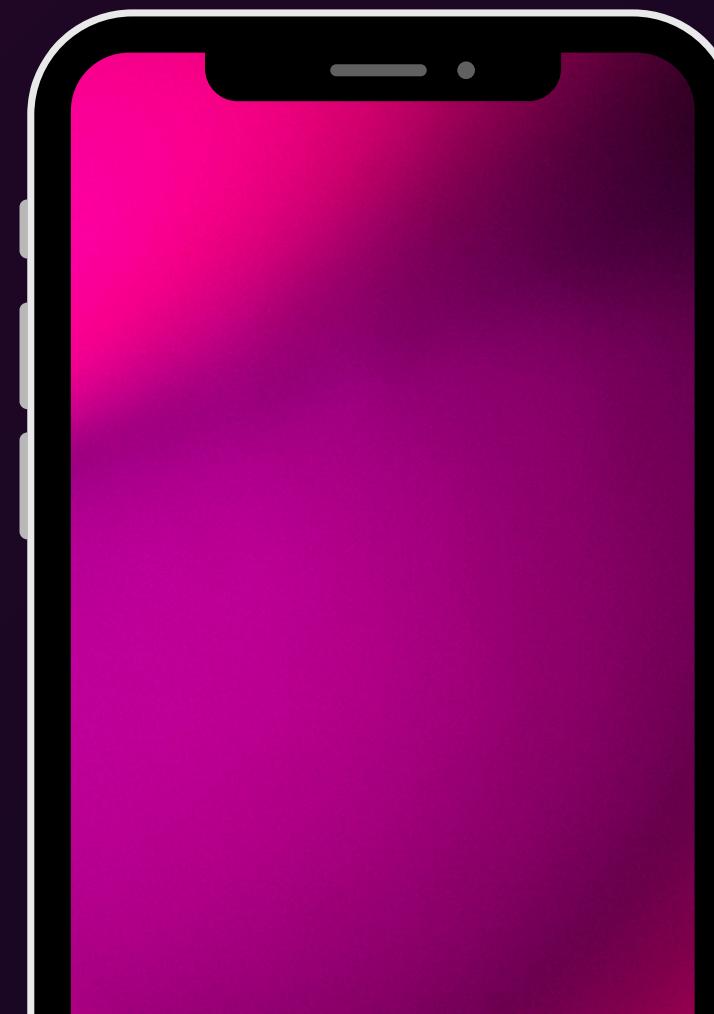
# librerias importantes

---

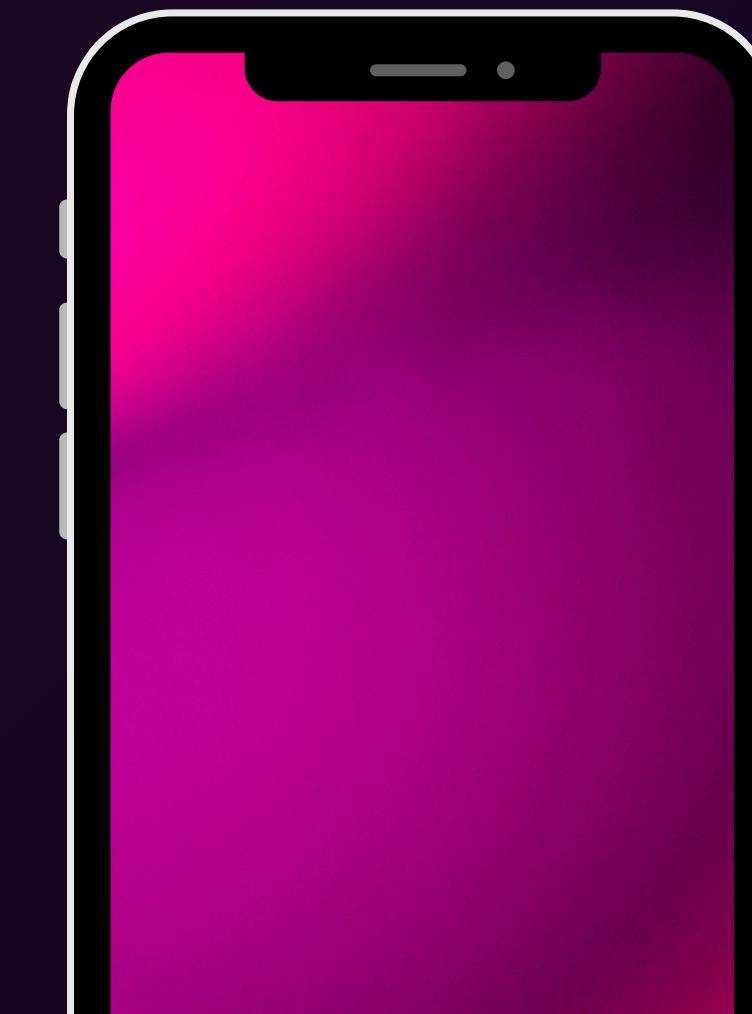
01

Levenshtein

02

sqlite3

03

kivy\_

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

## Fuerza bruta

```
Hilo-7 probando: 398400
Hilo-16 probando: 954868
Hilo-16 probando: 954869
Hilo-16 probando: 954870
Hilo-12 probando: 712800
Hilo-12 probando: 712801
Hilo-2 probando: 078199
Hilo-9 probando: 522010
Hilo-9 probando: 522011
Hilo-4 probando: 207477
Hilo-4 probando: 207478
Hilo-4 probando: 207479
Hilo-15 probando: 895301
Hilo-3 probando: 143787
Hilo-3 probando: 143788
Hilo-13 probando: 773295
Hilo-13 probando: 773296
Hilo-11 probando: 644158
Hilo-11 probando: 644159
Hilo-11 probando: 644160
Hilo-11 probando: 644161
Hilo-12 probando: 712802
```

Contraseña encontrada: 712802 por Hilo-12Hilo-2 probando: 078200

```
Hilo-6 probando: 334007
Hilo-9 probando: 522012
Hilo-5 probando: 266599
Hilo-10 probando: 586065
Hilo-4 probando: 207480
Hilo-15 probando: 895302
Hilo-1 probando: 018992
Hilo-3 probando: 143789
Hilo-7 probando: 398409
Hilo-13 probando: 773297
Hilo-14 probando: 831771
Hilo-16 probando: 954871
Hilo-8 probando: 456903
Hilo-11 probando: 644162
Búsqueda completada en 29.38 segundos.
```

## Explosion combinacional

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

$$\text{tiempo} = \frac{(\# \text{caracteres})^{\text{longitud}}}{\text{procesamiento/s}}$$

Para analizar la longitud y tiempo en nuestra app y su resistencia a un ataque de fuerza bruta para nuestra aplicación:

Letras mayúsculas [A – Z] = 27

Letras minúsculas [a – z] = 27

Números: [0 – 9] = 10

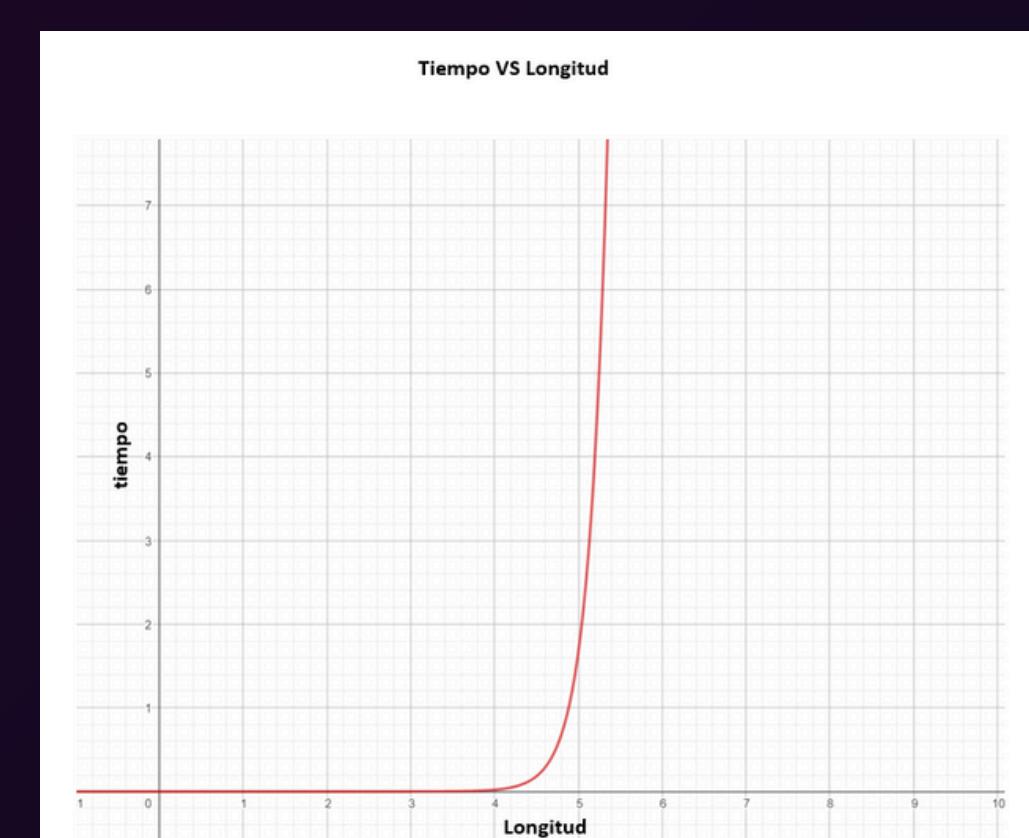
Caracteres especiales [! @#\$%^& \* (), . ? \": {}| <>] = 20

Total = #caracteres = 84

Considerando un procesador estándar → Procesamiento/s = 2.5GHz

$$\text{tiempo} = \frac{(84)^l}{2.5 * 10^9} \text{ s}$$

Tiempo VS Longitud

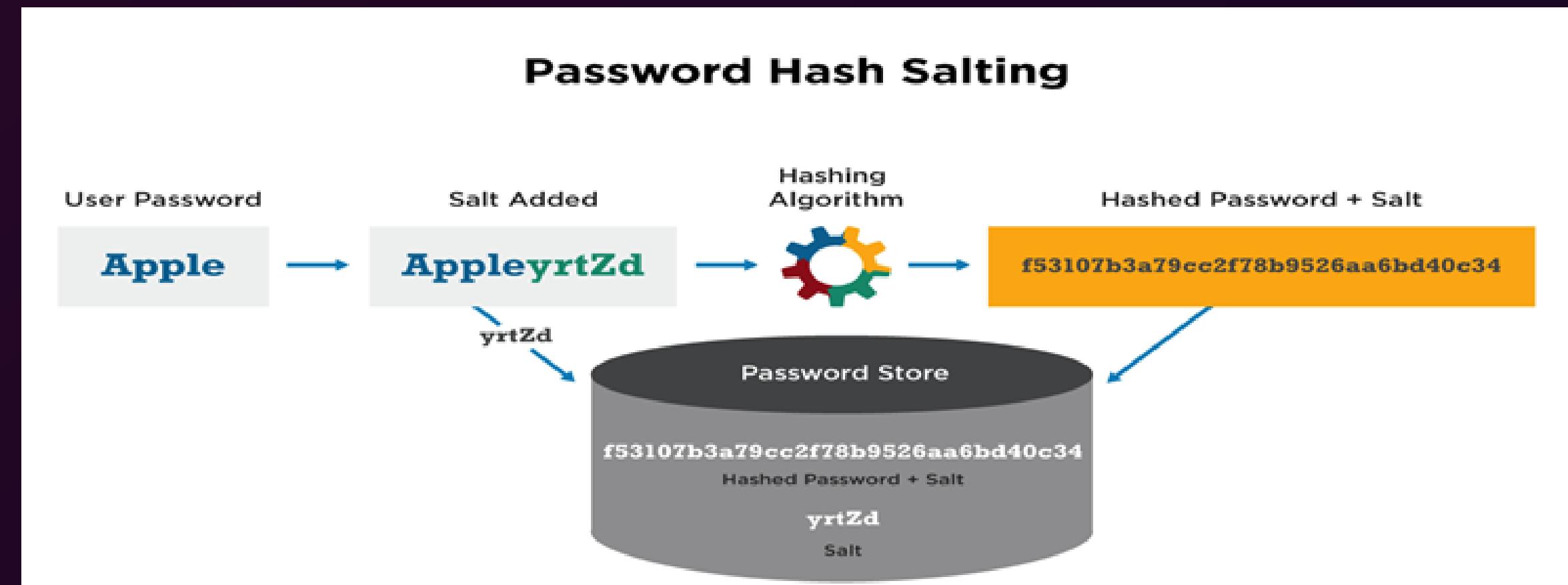


Ahora considerando a l como nuestra longitud mínima l = 12

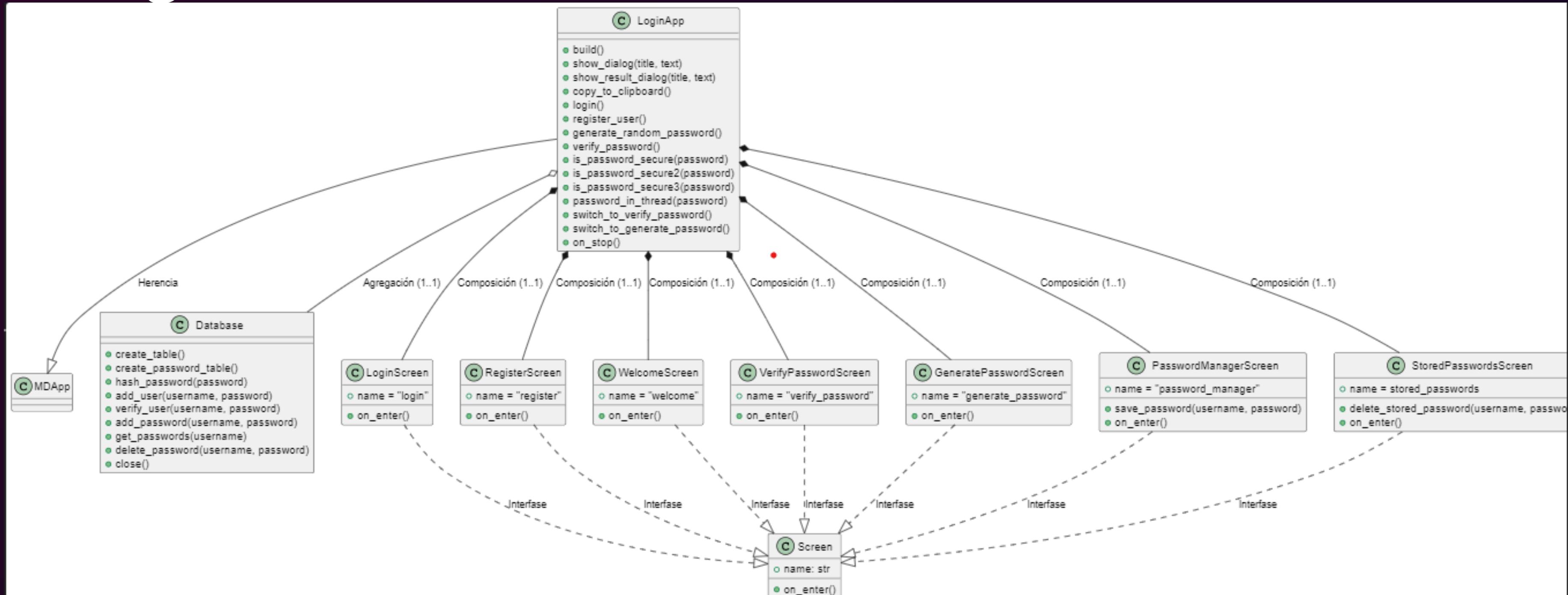
$$t = 49,364,122,806,910.45 \text{ s} = 1,565,326.07 \text{ años}$$



# Hash y Salt



# Diagrama UML



**CONTINUAMOS CON LA PRUEBA**