
**Universidad de Valladolid**

E.T.S Ingeniería Informática

**Trabajo Fin de Grado**

Grado en Ingeniería Informática

# Entrophy

Autor:

**D. David Marciel Pariente**


# Universidad de Valladolid

## E.T.S Ingeniería Informática

**Trabajo Fin de Grado**

Grado en Ingeniería Informática

# Entrophy

Autor:
**D. David Marciel Pariente**
Tutor:
**D. Carlos Enrique Vivaracho Pascual**

# Resumen

El presente trabajo intenta solucionar el problema que acarrea el uso de contraseñas en lugares en los que la persona que introduce sus credenciales pueda ser observada, grabada o la conexión interceptada.

Para ello, se utiliza una técnica de ofuscación visual novedosa, combinada con la introducción de varias posibilidades para cada posición de la contraseña.

De esta forma conseguimos autentificarnos ofuscando nuestra contraseña a aquellos que nos estén viendo, y revelando un conjunto de posibilidades para cada dígito de la contraseña, de los cuales solo uno (el solución) no es aleatorio, a aquellos que intercepten la comunicación.

El proyecto no tiene aspiraciones de prototipo comercial, sino de prueba de concepto para probar la viabilidad de las ideas presentadas.

Tanto la idea como su diseño son originales del autor.

# Índice

# Capítulo 1. Introducción

Desde el inicio de los computadores la forma de conseguir niveles de seguridad y/o autentificación han sido las contraseñas.

Con una contraseña podemos demostrar que somos quienes decimos ser, proteger información, acceder a lugares reservados.

Las contraseñas se han convertido en la forma de protección informática por excelencia.

Las contraseñas tienen ciertos problemas con los que ya estamos acostumbrados a vivir.

Problemas, por otra parte, que pueden llegar a ser graves, como la falta de encriptación, y es que, es muy difícil pedirle a un humano que encripte "de cabeza" algo. Por otra parte, en el caso de los ordenadores la autenticación sí que ha sufrido mejoras sustanciales (el cifrado con clave pública es un buen ejemplo).

Sin embargo, la forma de introducir las contraseñas apenas ha evolucionado desde su creación: seguimos usando un teclado.

Esto implica un riesgo de seguridad conocido, que es la captura de la clave por visualización del tecleo.

Hoy en día muchas empresas están preocupadas por su seguridad informática, sobre todo por el robo de credenciales.

Éstas son objetivo de ladrones y hackers a diario, lo que obliga a las empresas a invertir grandes cantidades de dinero en el desarrollo de mejoras para aumentar su seguridad informática.

Los bancos llevan años buscando la forma de mejorar su seguridad, sin embargo, siguen siendo vulnerables a ataques físicos tales como cámaras grabando el teclado en el momento del tecleo de la clave.

Los ataques más conocidos son: "Shoulder surfing" (personas espiando mientras introduces la contraseña), grabación directa al introducir la contraseña y ataques "Man in the middle" (servidores/nodos intermedios infectados espiando la comunicación).

La finalidad de este proyecto es plantear una solución novedosa a este problema planteando una alternativa original a la hora de teclear la clave. Esta solución se ha centrado en el presente trabajo al ámbito de uso a los teléfonos móviles, por considerarlos de uso más extendido y poseer unas características que se adaptan perfectamente a la solución planteada.

El proyecto no pretende construir una aplicación totalmente funcional, sino que pretende ser una prueba de concepto, que demuestre la viabilidad o no de esta solución. Por eso, el presente trabajo es un proyecto de investigación más que de desarrollo.

## 1.1 Breve introducción Histórica

A lo largo de la historia, se han dado casos de **protocolos en los que no se cifraba la contraseña** y esta viajaba en texto plano, lista para ser leída por cualquier nodo intermedio (el sonoro caso de telnet), actualmente este problema tiene varias soluciones.

Una de ellas, la más sencilla, es enviar al servidor el hash de la contraseña en vez de la contraseña plana. De esta forma el atacante no conocerá la contraseña, pero no evitará que se autentifique **utilizando el hash** (dado que lo conoce y es lo que se le requerirá) en vez de la contraseña. Ni tampoco nos asegura seguridad frente a ataques de fuerza bruta, estos ataques son tan altamente conocidos y utilizados que tienen herramientas específicas optimizadas para llevarse a cabo (por ejemplo John the ripper[1]).

La solución que se da hoy a este problema es mucho más elaborada. Se utiliza cifrado de clave pública para iniciar una sesión privada, una vez se ha iniciado se puede utilizar cifrado simétrico (que es mucho más eficiente) para mantener la sesión encriptada.

Se puede decir que el **cifrado de clave pública** es seguro, pero no es irrompible. Con una capacidad de computo suficientemente alta, super-ordenadores, o computación cuántica[2][3] (aún se está investigando como conseguirla, aunque últimamente ha habido grandes avances D-Wave), el cifrado puede ser roto.

Existe otra forma con mala reputación para conseguirla, controlando de una forma o de otra las entidades certificadoras. Éste último caso, considerado paranoia durante mucho tiempo, es hoy aceptado por la comunidad gracias a la información revelada por Edward Snowden[4][5][6]. Por suerte para los ciudadanos estadounidenses, fue el gobierno de los EEUU el encargado de realizarlo, no un grupo terrorista, pero demuestra la posibilidad de atacar el considerado buque insignia de la seguridad informática actual.

Desde el punto de vista de la introducción de contraseñas poco se ha avanzado mas allá de un ligero movimiento en teclados virtuales (Ilustración 1 Ilustración 2, Ilustración 3).


*Ilustración 1*

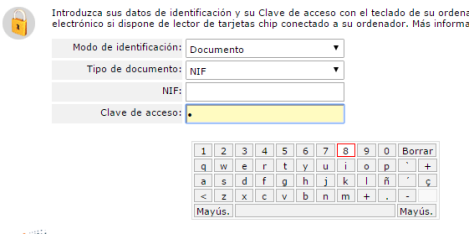


*Ilustración 2*

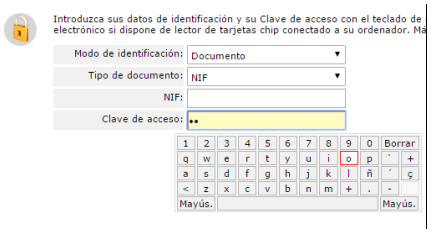


*Ilustración 3*

## 1.2 Nuestra propuesta

Conociendo los problemas actuales y el estado del arte daremos una solución, intentando mantener o mejorar el nivel de seguridad global, y solucionando a su vez los problemas enumerados.

Para ello, plantearemos una aplicación capaz de ofuscar visualmente la introducción de contraseñas, e intentaremos extender estos beneficios al total de la comunicación. De esta forma conseguiremos beneficiarnos de ellos también a nivel de conexión.

La ofuscación visual se conseguirá por medio de caos, a su vez generado por gran cantidad de letras en movimiento indeterminado.

Además, utilizaremos conjuntos como solución, al utilizar diferentes letras/números como candidatas a una misma posición conseguimos demostrar conocer nuestra contraseña sin revelarla.

## 1.3 Objetivos

### *Objetivo general*

Realizar un análisis experimental de una nueva propuesta de introducción de clave, para evaluar su viabilidad.

### *Objetivos específicos*

- Construir distintos prototipos como prueba de concepto.
- Realizar una evaluación cuantitativa de la interacción del usuario con cada prototipo.
- Realizar una evaluación cualitativa de la interacción del usuario con cada prototipo.
- Realizar diferentes pruebas modificando alfabetos posiciones, colores, movimientos y velocidades para encontrar la mejor forma de combinarlos.
- Probar con usuarios las versiones de desarrollo.
- Obtener gran cantidad de datos para poder cotejar los resultados.
- Llegar a una conclusión.

## 1.4 Contenido de la memoria

### *Capítulo 1. Introducción*

En este capítulo realizamos una pequeña introducción histórica, repasamos los problemas de las contraseñas actuales y estado del arte e introducimos la propuesta.

### *Capítulo 2. Interacción hombre-máquina*

Hablamos sobre los conocimientos actuales, el concepto de usabilidad y las actuales técnicas conocidas para mejorar el software.

### *Capítulo 3. Evaluación*

Comentamos cómo realizaremos la captura de información, evaluación de la interfaz, y análisis.

### *Capítulo 4. Entrophy*

Explicación de la propuesta y diseño de la solución.

### *Capítulo 5. Aplicación inicial*

Realización del primer prototipo, análisis y conclusiones.

### *Capítulo 6. Empezando de Cero*

Segundo prototipo, empezando desde cero para construir de forma más robusta.

### *Capítulo 7. Predice el movimiento*

Tercer prototipo, mejoramos su usabilidad añadiendo nuevos movimientos predecibles para hacer más fácil encontrar lo que buscamos.

### *Capítulo 8. Conclusiones*

Análisis del trabajo realizado así como la posible viabilidad de la propuesta.

# PARTE I Fundamentos Teóricos

## Capítulo 2. Interacción hombre-máquina

### 2.1 Introducción

Los humanos somos seres visuales [7].

La interacción de los humanos con su entorno es sobretodo visual, la proximidad entre el ojo y el cerebro no es casual, se podría decir que el cerebro crece alrededor del nervio óptico, y eso nos convierte en seres visuales.

Para entender conceptos podemos utilizar abstracciones, como por ejemplo letras o conceptos, sin embargo, se ha demostrado que un buen diseño hace uso de las habilidades nativas a los humanos más que de las artificiales creadas por nosotros.

La **usabilidad** es una palabra que hace referencia a la facilidad de uso de algo, de esta forma, si decimos que algo es usable estamos diciendo que su uso es afable con el usuario, intuitivo, fiable, fácil de usar.

Cada vez más se exige una buena usabilidad para el software. Para asegurarse de ello, se recomienda seguir ciertas reglas o principios. En nuestro caso, es importante tener en cuenta este punto, ya que tan importante como evitar la captura del tecleo de la clave, es que el usuario esté lo más cómodo posible con la nueva forma planteada para su introducción.

### 2.2 Principios Gestalt

Guestalt (figura en alemán) hace referencia a los principios de percepción estudiados psicólogos alemanes a principios de siglo XX, estos estudios intentan explicar cómo percibimos los humanos el mundo y se agrupan en "principios".

Conociendo estos principios somos capaces de hacer aplicaciones más intuitivas, consiguiendo así un desempeño mucho más eficiente y un tiempo de "aprendizaje" del sistema y la interfaz mucho más bajo o nulo. Su uso está estrechamente relacionado con la usabilidad, por este motivo es importante utilizarlos debidamente.

Principios Gestalt:

**Proximidad**: los elementos cercanos tienden a considerarse como parte de un conjunto.

**Similitud**: los elementos semejantes tienden a considerarse parte de un mismo conjunto.

**Continuidad**: el cerebro intenta encontrar patrones conocidos basándose en la posición de los objetos.

**Cierre**: el cerebro intenta "llenar" los vacíos de formas conocidas viéndoles como un todo al que le falta algo en vez de un algo que se parece a un patrón.

**Simetría**: el cerebro intenta encontrar simetría en los elementos mostrados.

**Figura/fondo**: el cerebro intenta ver los objetos como parte de un plano o del fondo.

**Destino común**: aquellos elementos que tienen algo en común (se mueven igual) se tienden a asociar como parte de un mismo conjunto.

**Combinación**: si varios principios se combinan el resultado es aún más intuitivo.

Por ejemplo: teniendo estos principios en cuenta, en un reproductor de música:

- Pondremos los botones relacionados con el mismo fin juntos. Los botones reproducir, parar, siguiente canción y canción anterior estarán juntos (principio de proximidad)

- La forma de los botones siguiente canción y canción anterior será muy similar entre sí (similitud). Y probablemente inversos respecto a un eje imaginario. Podrían tener forma de flecha o triángulo apuntando hacia la derecha e izquierda (simetría) lo que recordaría a ir hacia atrás y adelante (similitud). En conjunto sería una imagen más fuerte de su funcionalidad (combinación).
- Además, todos ellos destacarán frente al fondo (figura/fondo) para poder diferenciarlos bien.

## 2.3 Limite de la visión humana

La visión humana es limitada, por ello, el número de colores diferentes que somos capaces de percibir, así como diferenciar varía de una persona a otra, pero no suele ser muy elevado.

Por ello, utilizar colores que destaquen entre sí, es una buena idea cuando intentamos diferenciar elementos.

Se recomienda utilizar los colores primarios y secundarios acompañados por el blanco y el negro. En concreto amarillo, rojo, azul, verde, negro y blanco dan un buen resultado cuando se busca un contraste notable, estos colores destacan muy bien unos sobre otros y permiten diferenciarlos bien.

También hemos de tener en cuenta hechos bien conocidos como que la palidez, los objetos pequeños y la separación entre objetos nos impiden apreciar el color debidamente. Así que si algo es muy pálido intentaremos no hacerlo muy pequeño y a la inversa, de esta forma estaremos seguros de que el color se diferencia suficientemente bien.

Además, es bien sabido que la visión humana está optimizada para detectar contrastes pero no brillos y que existen grandes diferencias respecto del mismo color presentado en los diferentes dispositivos. Por ello, no podemos fiarnos simplemente de lo que estamos viendo en nuestro dispositivo, estamos invitados a utilizar las guías anteriormente mencionadas aun cuando en nuestro dispositivo todo se vea bien.

## 2.4 Uso de la memoria humana

Los estudios de la memoria humana ayudan a mejorar mucho la usabilidad. Los humanos tenemos una pequeña memoria a corto plazo que debemos explotar al máximo. La memoria a largo plazo es más grande, pero requiere de más esfuerzo, por lo que se recomienda no forzar al usuario a utilizarla, a menos que sea estrictamente necesario.

Si seguimos los principios Gestalt nos aseguraremos de que **reconozca** y entienda el significado de partes de la aplicación. Aprovechar recuerdos que ya están presentes en el usuario es como **utilizar su memoria a largo plazo sin coste**, este reconocimiento es parte de la ya mencionada "**usabilidad**" que queremos explotar, de esta forma conseguiremos que el usuario se sienta a gusto con la aplicación.

Priorizando el reconocimiento por encima del recuerdo a largo plazo no tendremos que forzar la limitada memoria humana. Igualmente, utilizar **feedback** al interactuar con el programa, permitir **recuperarse de errores** y añadir **consejos** son considerados buenas prácticas, ya que aumentan considerablemente la usabilidad del programa.

## 2.5 Coordinación mano-ojo

En 1954 Paul Fitts, un psicólogo y militar americano estudió la dificultad de alcanzar un objetivo para un humano, para ello utilizó diferentes dispositivos, entre los que destaca el ratón del ordenador.

Fitts llegó a la conclusión de que el tiempo en alcanzar un objetivo se podía medir basándose en: el tamaño del objetivo(W), la distancia al objetivo desde el punto de partida (D) y otras dos variables dependientes de cada persona a y b.

$$T = a + b * \log2( 1 + D / W )$$

Esta ley, popularizada hoy en día no es suficiente cuando se habla de dispositivos móviles puesto que el dispositivo de entrada es muy diferente a los que Fitts utilizó en sus estudios, sin embargo, existen aproximaciones no formales a una posible ley de Fitts para dispositivos móviles (Ilustración 4) [8][9].

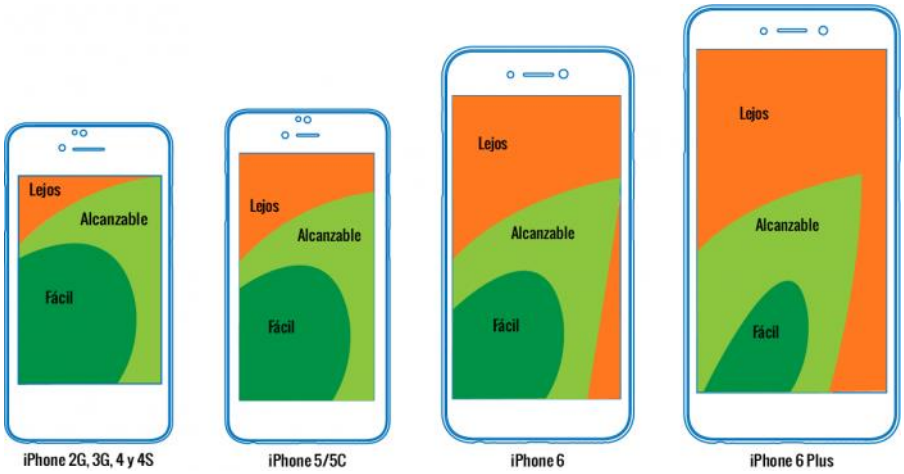


*Ilustración 4*

## 2.6 Requisitos temporales

Los requisitos temporales son una parte fundamental de la usabilidad. Es totalmente necesario hacer saber al usuario de inmediato que su orden ha sido recibida. Igualmente ha de saber el tiempo que requerirá una petición en ser procesada. De otra forma el usuario no será consciente de que su orden ha sido recibida o está siendo procesada, esto crea una sensación de malestar y de indecisión sobre si repetir o no la acción.

Para evitarlo basta con añadir cierto feedback que muestre sencillamente que la orden ha sido recibida, de esta forma el usuario será consciente de que ha de esperar y no volverá a solicitar lo mismo cargando el sistema.

Para el caso en el que la acción lleve asociado un alto tiempo de espera es recomendable añadir un contador o una barra porcentual, de esta forma el usuario será consciente en todo momento del estado en el que el dispositivo se encuentra. Otra buena solución para mejorar la respuesta del sistema es anticiparse en la realización de tareas costosas comunes, pudiendo de esta forma devolver el resultado de forma "aparentemente inmediata" al ser transparente para el usuario.

Respecto al problema que vamos a tratar es importante conocer los tiempos de respuesta del ojo humano, se estima que el ojo humano es capaz de reconocer elementos a una velocidad de 0.15 segundos/palabra y requiere de 0.25 segundos para identificar un objeto así que probablemente debamos encontrar la forma de facilitar el encontrar la letra que el usuario está buscando.

## 2.7 Usuario

Es necesario conocer el usuario al que va dirigida la aplicación, de esta forma podremos centrarnos en él y especializar nuestra aplicación para resolver sus problemas de la forma más eficiente posible.

En este caso, el usuario al que se enfoca la aplicación es un usuario el cual no tiene por qué tener grandes conocimientos informáticos pero sí que está habituado con el uso de tecnología del siglo XXI de forma que, probablemente haya comprado por internet en varias ocasiones, está preocupado por su seguridad informática y utilice dispositivos móviles de forma habitual.

# Capítulo 3. Evaluación

## 3.1 Evaluación de la interfaz

Para evaluar el interfaz hemos de tener en cuenta tres puntos:

- Captura de información.
- Análisis de los datos.
- Critica para mejorar el sistema.

## 3.2 Captura de información

Para capturar información de forma eficiente utilizaremos SUS (System Usability Scale) [Figura 1] [10]. SUS consta de 10 afirmaciones las cuales se puntúan bajo una metodología de escala Likert de cinco niveles.

La escala de Likert [11] es una escala balanceada (tiene el mismo número de posiciones positivas que negativas). Cada nivel está asociado a un baremo de conformidad que va de "muy en desacuerdo" a "muy de acuerdo", siendo las escalas de elección no forzosa (las impares con un nivel neutro) las más utilizadas, y dentro de estas la escala de 5 niveles ("muy en desacuerdo", "en desacuerdo", "ni de acuerdo ni en desacuerdo", "de acuerdo", "muy de acuerdo") la más habitual debido a sus buenos resultados empíricos.

Además, a fin de conseguir información provechosa para nuestro caso, lo complementaremos con alguna pregunta específica para el dominio, estas preguntas serán tanto preguntas objetivas como subjetivas de uso y una pregunta abierta de opinión.

Para la medida de datos objetivos utilizaremos una plantilla [Plantilla 1] [Platilla_evaluación_prototipo_color.docx adjunta] basada en escala Likert y que se amolde a nuestras necesidades. Mediremos: tiempo requerido, número de errores cometido y las letras por tic. Además, se tomarán medidas más subjetivas como la facilidad para encontrar las letras, si se diferencian bien los colores o si la velocidad resulta adecuada.

**Platilla evaluación de prototipo**

**1. Clave:** DavID

**2. Evaluación Objetiva**

| | 1 | 2 | 3 |
|---|---|---|---|
| **Tiempo empleado** | | | |
| **Errores al pulsar** | | | |
| **Letras por tic** | , , , , | , , , , | , , , , |

**3. Evaluación Subjetiva**

*1 (Muy en desacuerdo) 2 (En desacuerdo) 3 (Normal) 4 (De acuerdo) 5 (Muy de acuerdo)*

Se diferencian bien los colores:

Encuentro fácilmente las letras de mi clave:

La velocidad al moverse de las letras es adecuada:

En caso de respuesta negativa (1 o 2), indicar si hay que acelerar o ralentizar:

**4. Comentarios/Observaciones/Mejoras**

*Plantilla 1*

"SUS" (System Usability Scale).

1. Specify usability evaluation goals.
2. Determine UI aspects to evaluate.
3. Identify target users.
4. Select usability metrics.
5. Select evaluation method(s).
6. Select tasks.
7. Design experiments.
8. Capture usability data.
9. Analyze and interpret usability data.
10. Critique UI to suggest improvements.
11. Iterate the process if necessary.
12. Present results.

*Figura 1*

## 3.3 Análisis de datos

Tras cada versión recogeremos datos de uso. Para ello cada usuario de prueba rellenará la plantilla descrita en el punto anterior, donde recogeremos datos objetivos y subjetivos.

Utilizando estos datos haremos un análisis para evaluar la versión, extrayendo los puntos fuertes y débiles de cada prototipo para ir mejorando.

## 3.4 Critica para mejorar el sistema

Basándonos en el análisis de datos intentaremos sacar conclusiones para guiar las futuras versiones. Así plantearemos los pasos a seguir para futuras versiones.

Tomaremos las decisiones oportunas para llevar a cabo la siguiente versión de la forma más provechosa posible.

Intentamos, de esta forma, averiguar cómo mejorar la aplicación por medio de la información obtenida en las versiones previas.

# Parte II Nuestra propuesta

# Capítulo 4. Entrophy

## 4.1 Nuestra propuesta

Como ya hemos dicho la intención de este trabajo es dar solución al problema generado cuando una persona es observada al introducir su contraseña.

Para abordar este problema utilizaremos un enfoque diferente al habitual. Este enfoque intenta resolver el problema de una forma novedosa y es idea original del autor.

La propuesta consta de una base teórica, aunque esta base es abstracta en algunos puntos, y por ello, sobre esta base se da una solución concreta sobre la que trabajar. Las bases teóricas son las siguientes:

1. **Generar caos** La persona debe ver lo que está buscando pero el resto no.

   Para ello hemos de generar un "Caos" suficientemente grande como para ser absorbido por completo. Es una solución controvertida, no es fácil generar un caos selectivo.

   No podemos olvidarnos de que ese caos no puede interferir con el usuario. Por eso, debemos a su vez facilitar encontrar la contraseña al usuario que la está buscando.

2. **Utilizar conjuntos**. Para evitar el problema del revelado de contraseña, permitiremos que el usuario pueda pulsar diferentes "teclas" o "números" para una misma posición. De esta forma, un atacante externo (ya sea persona o máquina) no podrá conocer cuál de ellas es la solución.

   Por ejemplo: si alguien escribe su contraseña en un teclado tradicional "1234" y es visto, grabado o la conexión es espiada, será trivial averiguar cuál es. Pero si en vez de eso dice que el primer dígito es uno del conjunto {1,a,56,g}, el segundo {6,o,2,T} y así sucesivamente para cada dígito de la contraseña será imposible determinar de forma eficaz la contraseña, ya que será una de las siguientes combinaciones:

$$\prod_{i=1}^{i=(dígitos\ de\ la\ contraseña)} (nº\ de\ dígitos\ posibles\ para\ la\ pulsación\ i)$$

   Es decir, dado que no se nos dice la contraseña no sabemos cuál es, lo máximo que podemos conseguir es con un conjunto de posibles claves entre las cuales estará la real.

$$\left(\frac{1}{\text{Dígitos candidatos por pulsación}}\right)^{Número\ de\ \ pulsaciones} = \varrho$$

   Actualmente la probabilidad de adivinar la clave viendo escribirla es 100%, el dígito seleccionado es el único candidato, por lo que **si vemos cual se introduce conocemos el dígito solución.**

$$\left(\frac{1}{1}\right)^4 = \frac{1}{1} = \mathbf{100}\%$$

   Utilizando esta técnica podemos evitar que eso pase aun cuando hayamos sido espiados, ya que **sabremos que una de las letras del conjunto es la válida, pero no habrá forma de saber cuál, ya que el conjunto constará**

**de varias letras.** En el caso propuesto había 4 dígitos por conjunto, por lo que, aun conociendo los conjuntos introducidos, el número de posibles soluciones es 4^4 y la probabilidad de adivinar la clave es:

$$\left(\frac{1}{4}\right)^4 = \frac{1}{256} \approx \mathbf{0.4}\%$$

Esta solución, eficaz en la teoría no lo es tanto en la práctica puesto que nada impide al atacante elegir los mismos conjuntos de entrada. Por ejemplo, podría elegir literalmente los mismos conjuntos que nosotros hemos elegido; si nos ve escribir como solución al primer dígito el conjunto {1,a,56,g},

y tiene libertad para elegir su conjunto respuesta, podría repetir éste conjunto como solución él y estar seguro al 100% de su validez, dejando inservible nuestro razonamiento.

Por ello hemos de encontrar una mejora que no permita seleccionar los mismos conjuntos de entrada permitiendo a su vez elegir siempre el dígito solución.

3. **Ser impredecibles.** Casi tenemos solución a nuestro problema: generar conjuntos de forma dinámica, si conseguimos conjuntos aleatorios podremos evitar el problema generado en el punto anterior.

   Para ello ¿por qué no utilizar el caos del que hablamos en el primer punto? Generar un modelo impredecible y generar caos son dos conceptos bastante cercanos, lo suficiente como para permitir que uno implique el otro sin dificultad.

   Utilizaremos un patrón de selección aleatorio para generar los conjuntos.

4. **Echar cuentas.** Nuestras conclusiones parecen estables. Sin embargo, al echar cuentas las cosas cambian.

   Necesitamos una forma de medir la viabilidad de uno u otro sistema. Sabemos que las variables que definen el sistema son número de dígitos candidatos por pulsación, número de dígitos total y número de dígitos de la clave. También sabemos que se relacionan de la siguiente forma:

$$\left(\frac{\text{nº dígitos candidatos por pulsacion}}{\text{nº dígitos total}}\right)^{nº\ de\ digitos\ clave} = \mathbb{I}$$

Donde $\mathbb{I}$ es la probabilidad de acertar nuestra clave al azar. Por este motivo, para conseguir aumentar la seguridad tendremos que será **minimizar $\mathbb{I}$ tanto como podamos**.

Utilizando 10 dígitos (0-9) con cuatro pulsaciones podemos llegar a hacer 10000 contraseñas diferentes (0000-9999) de las cuales solo una es solución. Podemos por lo tanto decir que hay 1/10000 probabilidades de que acierten nuestra contraseña (aunque en realidad hay bastantes menos [12]).

$$\left(\frac{1}{10}\right)^4 = \frac{1}{10000}$$

Como ya hemos dicho, **nuestra intención es aumentar el número de letras posibles (candidatas) por pulsación** para evitar que los atacantes externos conozcan nuestra contraseña.

$$\left(\frac{\mathrm{X}}{\mathrm{Y}}\right)^{nº\ de\ digitos\ clave} = \mathbb{I} \qquad ; \qquad \frac{X}{Y} = Z$$

Matemáticamente esto se traduce **en aumentar el numerador** de la fracción **X/Y = $\boldsymbol{Z}$**.

Pero aumentar el numerador implica aumentar las probabilidades de acertar nuestra contraseña. En el caso de utilizar 4 dígitos esta probabilidad X/Y = Z es 1/10 = Z.

Si aumentamos X entonces Z aumentará también, y no podemos permitirnos que Z aumente porque cuanto más alta sea Z más alta será 𝕀𝕀 y por lo tanto más probabilidades de acertar nuestra contraseña en cada intento. **Nuestra seguridad informática se basa en lo pequeña que sea** 𝕀𝕀, para evitar que se vea comprometida tenemos que mantener Z con un valor tan pequeño como sea posible (al menos similar al 1/10 actual).

En el caso propuesto habría cuatro posibilidades por dígito, por lo que, como ya se veía venir 10 dígitos se quedan pequeños pasando de 10000 posibilidades a unas 40.

Las posibilidades de acertar la contraseña aumentan mucho al permitir selección de múltiples letras candidatas, de esta forma, sería más vulnerable a ataques de fuerza bruta. Por ello, necesitamos encontrar una forma de equilibrar la balanza.

$$\left(\frac{1}{10}\right)^4 = \frac{\mathbf{1}}{\mathbf{10000}} \quad << \quad \left(\frac{4}{10}\right)^4 = \frac{256}{10000} \approx \frac{\mathbf{1}}{\mathbf{40}}$$

Volviendo otra vez al campo matemático al aumentar X crece Z y ahora es demasiado grande como para ser viable. Para mantener Z en un nivel aceptable hemos de disminuirla tanto como podamos. Esto se traduce en **aumentar Y** (denominador) para estabilizar Z.

La forma de aumentar Y es añadir nuevas posibilidades, **hemos de utilizar más "dígitos" diferentes**.

La opción inmediata es utilizar el alfabeto (27 letras alfabeto inglés), probablemente también se queden cortas así que podríamos **utilizar otros alfabetos** diferentes (cirílico, devanagari, árabe, hebreo o chino).

Aun así, es posible que agotemos las letras viables (reconocibles por los usuarios) antes de tener suficientes como para que nos salgan los números, así que tenemos que pensar en otras posibilidades. Poder **"reutilizar" una misma letra** arreglaría este problema, para eso darles color o fuente diferente para poder diferenciarlas sería una buena opción.

Con estas transformaciones conseguimos dar un nuevo significado a la fórmula X/Y = Z de diez dígitos (1/10 = Z) y transformarla en (nº dígitos candidatos por pulsación / nº dígitos total = Z), así es mucho más maleable, dado que **ahora controlamos ambas variables.**

$$\left(\frac{\text{nº dígitos candidatos por pulsacion}}{\text{nº dígitos total}}\right)^{nº\ de\ digitos\ clave} = \mathbb{I}\mathbb{I}$$

5. **Encontrar lo que buscamos:** Ahora que podemos controlar tanto el número de dígitos como el número de dígitos candidatos hemos de centrarnos en la viabilidad de estas mejoras. De nada sirve una buena idea si no cumple con su cometido.

Por ello, debemos tener en cuenta que, al aumentar drásticamente el número de dígitos aumentamos también la complejidad al buscarlos, lo que implica una necesidad de facilitar esta tarea.

## 4.2 Diseño de la solución

Tras mostrar el concepto teórico intentaremos plasmarlo de la forma más pura posible.

Se implementará como **prueba de concepto** cuya finalidad no es en sí dar un producto final sino una aproximación al concepto teórico descrito para averiguar su factibilidad.

Podríamos centraremos en botones que contengan los conjuntos de elementos (sería lo más inmediato), pero estos botones ocupan mucho espacio, permiten introducir poca información y la mejor optimización que podemos utilizar para facilitar encontrar el dígito a los usuarios es ordenar los elementos que contiene.

De hecho, dado que queremos una gran cantidad de caos evitaremos todo aquello que no sea los dígitos. Partiremos de un fondo sobre el que destaquen bien las letras y **permitiremos pulsaciones sobre cualquier punto del espacio**, permitiendo de esta forma una **selección dinámica.**

Tomar estas decisiones tiene implicaciones muy importantes, por ejemplo, un movimiento aleatorio sobre un tablero suficientemente grande y con un conjunto de elementos suficientemente grande puede ser una buena forma de "mezclar" las letras y conseguir conjuntos dinámicos suficientemente fiables. De esta forma evitamos a los posibles atacantes repetir los mismos conjuntos vistos/grabados.

Pero el mismo caos que mezcla las letras genera a su vez dificultad para encontrar las letras buscadas, el usuario principal no tendrá fácil encontrar sus dígitos y generará un problema con ello.

Por ello, considerar utilizar **movimientos pseudo aleatorios** en vez de totalmente aleatorios empieza a verse como algo totalmente necesario **para bajar los tiempos de búsqueda.**

Los movimientos pseudo aleatorios de los que estamos hablando son movimientos aleatorios dentro de un rango, es decir, aleatorio pero dentro de la mitad superior de la pantalla, aleatorio en una vertical u horizontal, aleatorio grupal, si el grupo se mueve en conjunto será más fácil encontrar una letra concreta, basta con encontrar el grupo y posicionarla relativamente respecto a este.

La mejora temporal con estos movimientos es de esperar crucial, ya que aunque tengamos que buscar el dígito o letra correspondiente, el espacio de búsqueda será mucho menor y por lo tanto, mantendremos el tiempo en límites aceptables sin renunciar a la ofuscación que nos aporta la aleatoriedad.

Existen diferentes alternativas a explorar en cuanto a movimientos aleatorios, intentaremos probar varias por separado y en conjunto para decidiremos después por las más efectivas.

## 4.3 Alcance de la solución

Las soluciones informáticas actuales son capaces de cifrar conexiones punto a punto, es decir, consiguen que los dispositivos intermedios no conozcan la información que transmiten. La seguridad existe entre el primer dispositivo (dispositivo cliente) y el último (servidor).

Actualmente no existe forma de cifrar una conexión humano-servidor, dejando todos los nodos (cliente incluido) ajenos a esa comunicación.

Tampoco existe una forma eficiente de evitar el robo de credenciales al ser introducidos por un humano, se suele decir que "el factor de seguridad más débil es el factor humano", pero no se habla de la debilidad de la interacción hombre-máquina, ni de los protocolos existentes entre ellos.

Es pretensión de Entrophy solucionar estos problemas, siguiendo los pasos ya mencionados.

Conseguiremos ese extra de seguridad en el que únicamente el usuario y el servidor conocen la contraseña, el resto de nodos, en el mejor de los casos, solo pueden llegar a obtener un conjunto de posibilidades.

## 4.4 Requisitos

### *Requisitos funcionales*

- Generar letras dinámicamente.
- Mover las letras.
- Permitir cambiar el color de las letras.
- Permitir cambiar la fuente de las letras.
- Permitir movimientos de letras de forma grupal.
- Permitir marcar/desmarcar las letras de la candidatas.
- Tener una pantalla en la que se muestre la información del último intento, incluyendo nº de letras candidatas por pulsación, las letras pulsadas y si contiene a la letra solución.
- Mostrar si la contraseña ha sido aceptada o no en la pantalla de información del intento.
- Permitir su funcionamiento con respuesta positiva/negativa sin necesidad de servidor.

### *Requisitos no funcionales*

- Desarrollar la aplicación sobre la que trabajar para Android.
- Las letras se deben mover en tiempo real.
- Debe haber diferentes tipos de movimiento.

# Parte III Investigación y Desarrollo

## Capítulo 5. Versión 0, Aplicación Inicial

Como primera aproximación al concepto Entrophy vamos a realizar una implementación "libre" para ir familiarizándonos con el concepto y poder hacernos a la idea de los posibles caminos a tomar.

### 5.1 Versión 0.1

Para empezar se ha partido de un prototipo (v0.1) caracterizado por (Figura 2):

- Cinco colores de letras latinas (elegidos conforme a los principios de diseño expuestos en el apartado I)
- Letras indias árabes y rusas (explorando la posibilidad de diferentes alfabetos).
- Bucle inicial (existe un bucle inicial que reinicia el sistema 3 veces, de esta forma podemos ver hacia dónde se dirigen las letras de nuestra contraseña para prever dónde estarán en el futuro)
- Señala las letras candidatas pulsadas (Tras hacer una pulsación todas las letras seleccionadas se marcarán en rojo, de esta forma sabemos cuáles han sido seleccionadas y cuales no)

En esta implementación, cada letra latina tiene un movimiento aleatorio independiente, las letras no occidentales se mueven por bloques (cada bloque con un movimiento diferente), de esta forma, al moverse al unísono, su movimiento es más predecible. Basta por ejemplo, con encontrar una letra rusa para poder localizar aquella letra rusa que estamos buscando, puesto que su posición relativa se mantendrá entre sí.

En la pantalla principal (Figura 2. Pantalla principal v0.1 Figura 3. Letras mezcladas en la v0.1 Figura 4. Resultado v0.1 se puede ver la posición de partida del sistema con todas sus letras ordenadas. En la Figura 2 se aprecia el sistema una vez ha sido desordenado, aquí vemos como los bloques siguen manteniendo la posición relativa entre sus letras, sin embargo, las letras latinas no siguen ningún orden. Por último, en la Figura 2 podemos observar la respuesta del sistema, esta pantalla es la que muestra el sistema tras ejecutar el programa principal.

La respuesta muestra: **la contraseña** en la parte superior con letras grandes, cada carácter de la contraseña tiene como superíndice el **número de letras candidatas** y en caso de haber sido **acertada** una A roja como subíndice, además, bajo cada letra de la contraseña están las **letras candidatas** en esta pulsación y una copia algo más grande junto a ellas en caso de haber sido solución.


*Figura 2. Pantalla principal v0.1*

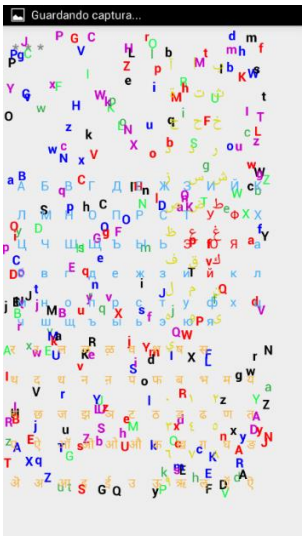


*Figura 3. Letras mezcladas en la v0.1*

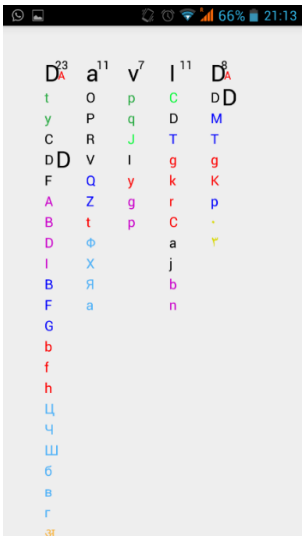

*Figura 4. Resultado v0.1*

**Resultados**

La Tabla 1 recoge los resultados cuantitativos, las conclusiones y comentarios de la evaluación son los siguientes:

- Se encuentran mejor las mayúsculas
- Muchos alfabetos distintos (es mejor empezar por letras conocidas)
- Quitar el bucle inicial (más que ayudar lía)
- Quitar el pulsado rojo (cuando las letras al ser candidatas se ponen rojas)
- Ralentizar el movimiento
- El movimiento matrix es fácil de seguir (el de las letras verdes, recuerda a matrix)
- Dejar un solo tono de verde (en vez de uno mas claro y otro mas oscuro)
- Pone nervioso no encontrar una letra
- Se confunde la l con la I (L e i)

| Version | 0.1 (primera iteración) | | | | | | 0.1 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | | Carlos | | | David | | |
| Tiempo | 51 | 35 | 75 | 25 | 22 | 28 | 60 | 60 | 12 | 53 | 40 | 50 |
| errores | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 1 | 0 | 1 |
| letras 1 tic | 13 | 21 | 16 | 26 | 25 | 16 | 15 | 19 | 22 | 17 | 13 | 20 |
| letras 2 tic | 16 | 11 | 11 | 3 | 19 | 17 | 14 | 19 | 13 | 18 | 12 | 16 |
| letras 3 tic | 17 | 25 | 9 | 4 | 26 | 11 | 8 | 9 | 17 | 5 | 7 | 8 |
| letras 4 tic | 9 | 8 | 6 | 9 | 20 | 7 | 9 | 7 | 9 | 5 | 7 | 16 |
| letras 5 tic | 11 | 17 | 7 | 8 | 9 | 10 | 9 | 10 | 7 | 11 | 9 | 7 |
| velocidad | 5 | | | 5 | | | 5 | | | 5 | | |
| se diferencian bien los colores | 4 | | | 5 | | | 4 | | | 4 | | |
| encuentro facilemnte mi clave | 1 | | | 3 | | | 1 | | | 2 | | |
| velocidad de las letras adecuada | 2 | | | 2 | | | 2 | | | 2 | | |
| acelerar/ralentizar | ralentizar poco | | un | ralentizar poco | | un | ralentizar poco | | un | ralentizar algo | | |

*Tabla 1*

**Análisis**

Parece que desconcierta un poco tantas letras juntas, no siempre es fácil encontrar la letra que buscas y en ocasiones puedes llegar a confundirlas.

El bucle inicial y el no encontrar la letra buscada tienen un efecto negativo en la experiencia de usuario.

El movimiento es algo rápido.

**Conclusiones**

Empezaremos desde cero, con pocos alfabetos e iremos aumentando el número de alfabetos hasta ver el punto en el que el usuario es capaz de manejar la aplicación.

Vamos a eliminar el bucle inicial y a reducir la velocidad para mejorar la experiencia de usuario.

Descartaremos los alfabetos no latinos para limitar la complejidad del sistema.

# Capítulo 6. Versión 1, Empezando de Cero

Tras la primera versión empezaremos desde cero e iremos haciendo crecer la aplicación de forma controlada.

## 6.1 Versión 1.2

**Introducción**

Teniendo en cuenta la primera iteración, empezaremos desde cero, para ello, se hemos realizado cambios.

**Cambios**

- Suprimido bucle inicial (no dio el resultado esperado)
- Añadida primera pulsación de inicio (ahora el programa no empezará hasta que nosotros pulsemos)
- Fondo de pantalla blanco
- Velocidad más lenta
- Añade un contador para conocer el tiempo que se tarda (al mostrar el resultado)
- Se puede seleccionar si quieres que señale las letras candidatas pulsadas o no (colorear en rojo las seleccionadas)
- Añadido marcador de contraseña correcta (al mostrar el resultado)
- Solo letras matrix y negras (una cantidad manejable de caracteres en pantalla)
- Bajo el tiempo de recuperación entre pulsaciones (ahora el tiempo de espera entre pulsaciones es menor)

Tras los cambios efectuados la pantalla principal es la de la Figura 5.

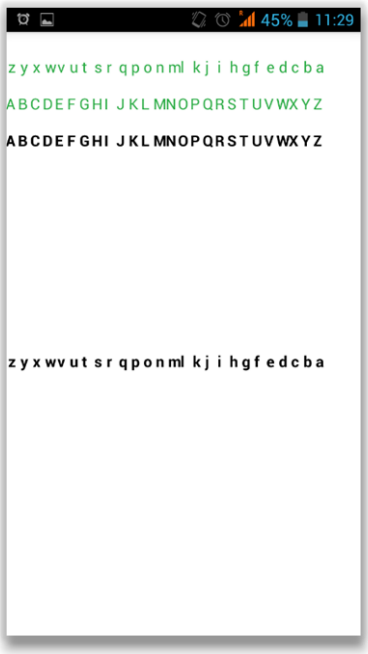


*Figura 5. Pantalla principal v1.2*

**Resultados**

Tras probar esta versión nos dimos cuenta de que las letras negras están demasiado lejos, esta distancia se ve innecesaria y desconcertante, se vio que los alfabetos podrían empezar por el mismo lado para intentar encontrar las letras con más facilidad.

Los datos objetivos de la evaluación se encuentran en la Tabla 2 se recogen los resultados cuantitativos de la evaluación.

| Version | 1.2 (segunda iteración) | | | | | |
|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | |
| Tiempo | 12 | 16 | 6.6 | 6.6 | 6 | 5 |
| errores | 1 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 15 | 13 | 13 | 8 | 16 | 12 |
| letras 2 tic | 8 | 4 | 5 | 5 | 5 | 5 |
| letras 3 tic | 2 | 2 | 4 | 3 | 5 | 5 |
| letras 4 tic | 4 | 5 | 3 | 3 | 4 | 4 |
| letras 5 tic | 5 | 1 | 3 | 1 | 1 | 3 |
| velocidad | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | |
| encuentro facilemnte mi clave | 3 | | | | | |
| velocidad de las letras adecuada | 2 | | | | | |
| acelerar/ralentizar | ralentizar | | | | | |

*Tabla 2*

**Conclusiones**

Pondremos los alfabetos en la misma dirección y juntaremos más las letras para ver si conseguimos una experiencia de usuario mejor.

## 6.2 Versión 1.3

**Cambios**

- Ambos alfabetos aleatorios miran en la misma dirección (alfabeto negro minúsculas de a-z en vez de ir z-a)
- Letras negras más juntas

La Figura 6 6 muestra la pantalla principal de la v1.3

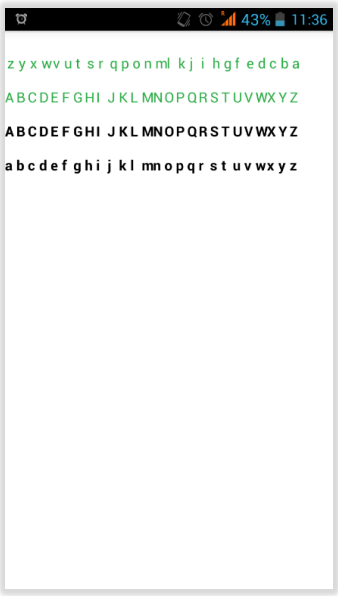


*Figura 6. Pantalla principal v1.3*

**Resultados**

En la Tabla 3 se recogen los resultados cuantitativos.

Los usuarios opinan que el resultado es favorable, aunque en este caso, las letras están demasiado juntas, también nos hemos dado cuenta de que en ocasiones cuesta encontrar la letra v.

| Version | 1.3 | | | | | |
|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | |
| Tiempo | 6 | 13 | 3.6 | 8.5 | 7.5 | 7.4 |
| errores | 0 | 1 | 0 | 1 | 0 | 0 |
| letras 1 tic | 17 | 17 | 13 | 17 | 16 | 16 |
| letras 2 tic | 8 | 12 | 3 | 9 | 8 | 9 |
| letras 3 tic | 6 | 5 | 5 | 7 | 9 | 13 |
| letras 4 tic | 6 | 5 | 6 | 4 | 4 | 8 |
| letras 5 tic | 4 | 5 | 8 | 5 | 8 | 4 |
| velocidad | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | |
| encuentro facilemnte mi clave | 3 | | | | | |
| velocidad de las letras adecuada | 2 | | | | | |
| acelerar/ralentizar | | | | | | |

*Tabla 3*

**Conclusiones**

Intentaremos separar los alfabetos negros en la próxima iteración manteniendo el orden de los alfabetos actual.

## 6.3 Versión 1.4

**Cambios**

- Separar un poco más letras negras (intentamos encontrar la distancia ideal)

La Figura 7 muestra la pantalla principal de la versión 1.4.


*Figura 7. Pantalla principal v1.4*

**Resultados**

La distancia entre alfabetos es correcta.

Hemos detectado problemas con el tiempo de espera entre pulsaciones, el estado no se ve lo suficientemente bien. Estos cambios serán arreglados en la próxima iteración.

En la Tabla 4 se recogen los resultados cuantitativos de esta evaluación.

| Version | 1.4 | | | | | |
|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | |
| Tiempo | 3.8 | 3.8 | 5.5 | 7.4 | 12 | 8.1 |
| errores | 0 | 0 | 0 | 0 | 1 | 0 |
| letras 1 tic | 13 | 14 | 12 | 14 | 13 | 13 |
| letras 2 tic | 3 | 4 | 4 | 7 | 9 | 3 |
| letras 3 tic | 11 | 7 | 9 | 10 | 7 | 1 |
| letras 4 tic | 6 | 5 | 1 | 9 | 7 | 11 |
| letras 5 tic | 5 | 5 | 5 | 7 | 6 | 3 |
| velocidad | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | |
| encuentro facilemnte mi clave | 4 | | | | | |
| velocidad de las letras adecuada | 3 | | | | | |
| acelerar/ralentizar | | | | | | |

*Tabla 4*

**Conclusiones**

Bajaremos el tiempo mínimo entre pulsaciones para permitir una velocidad de introducción mayor.

## 6.4 Versión 1.5

**Cambios**

- Reducido el tiempo entre pulsaciones (se necesita un tiempo mínimo para evitar pulsaciones múltiples, pero aún es demasiado alto como para detectar diferentes pulsaciones)

La Figura 8 muestra la pantalla principal de la versión 1.5.


*Figura 8. Pantalla principal 1.5*

**Resultados**

El problema de tiempo entre pulsaciones se ha solucionado, ahora es más cómodo.

Los resultados cuantitativos de la evaluación están recogidos en la Tabla 5 .

| Version | 1.5 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | | Carlos | | | David | | |
| Tiempo | 2.7 | 3.5 | 3.4 | 4.3 | 2.7 | 2.6 | 6.3 | 4.2 | 4.6 | 6.1 | 3.9 | 5.2 |
| errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 9 | 9 | 13 | 5 | 6 | 8 | 8 | 14 | 10 | 12 | 6 | 12 |
| letras 2 tic | 5 | 7 | 5 | 4 | 3 | 6 | 5 | 2 | 8 | 6 | 8 | 10 |
| letras 3 tic | 16 | 6 | 11 | 13 | 12 | 6 | 5 | 8 | 9 | 9 | 7 | 15 |
| letras 4 tic | 9 | 6 | 3 | 5 | 6 | 12 | 2 | 6 | 3 | 12 | 12 | 5 |
| letras 5 tic | 10 | 7 | 2 | 5 | 9 | 5 | 1 | 4 | 1 | 4 | 11 | 5 |
| velocidad | 5 | | | | | | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | | 5 | | | | | |
| encuentro facilemnte mi clave | 5 | | | | | | 4 | | | | | |
| velocidad de las letras adecuada | 3 | | | | | | 3 | | | | | |
| acelerar/ralentizar | Bajar un poco | | | | | | | | | | | |

*Tabla 5*

**Conclusiones**

En la próxima versión pondremos todos los alfabetos en el mismo orden, así comprobaremos si se encuentran mejor las letras cuando están ordenadas.

## 6.5 Versión 1.6

**Introducción**

- Alfabetos en el mismo orden (queremos comprobar si se encuentran mejor las letras)

La Figura 9 muestra la pantalla principal de la versión 1.6.


*Figura 9. Pantalla principal v1.6*

**Resultados**

No existe mejora.

Desestimar cambios.

En la Tabla 6 se recogen los resultados cuantitativos de la evaluación.

| Version | 1.6 | | | | | |
|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | |
| Tiempo | 5.4 | 3.9 | 6.8 | 9.8 | 6.4 | 27 |
| errores | 1 | 0 | 1 | 0 | 0 | 1 |
| letras 1 tic | 11 | 10 | 11 | 12 | 12 | 11 |
| letras 2 tic | 8 | 9 | 4 | 7 | 12 | 7 |
| letras 3 tic | 5 | 10 | 9 | 9 | 7 | 9 |
| letras 4 tic | 4 | 8 | 1 | 2 | 3 | 5 |
| letras 5 tic | 5 | 3 | 7 | 3 | 4 | 5 |
| velocidad | | | | | | |
| se diferencian bien los colores | | | | | | |
| encuentro facilemnte mi clave | | | | | | |
| velocidad de las letras adecuada | | | | | | |
| acelerar/ralentizar | | | | | | |

*Tabla 6*

**Conclusiones**

Se desestima el orden, en la próxima versión introduciremos más colores manteniendo el antiguo orden.

## 6.6 Versión 1.7

**Introducción**

Introduciremos dos nuevos alfabetos de color rojo, sin embargo, seguiremos utilizando la clave en negro, intentamos así averiguar si la introducción de nuevos alfabetos interfiere con lo ya construido.

**Cambios**

- Alfabeto verde uno en cada dirección de nuevo
- Añadido alfabeto rojo
- Probaremos con la Clave en negro

La Figura 10 muestra la pantalla principal de la versión 1.7.

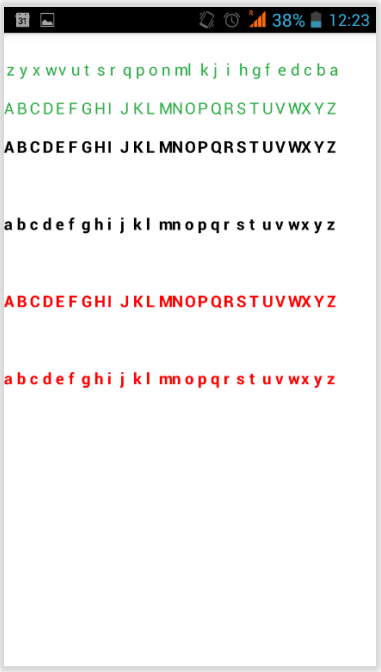


*Figura 10. Pantalla principal v1.7*

**Resultados**

No aparecen letras rojas mezcladas con la solución, tampoco parece haber un efecto negativo al añadir nuevos alfabetos.

No existe la mejora por ofuscación que estamos buscando.

Los resultados cuantitativos se encuentran en la Tabla 7.

| Version | 1.7 | | | | | |
|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | |
| Tiempo | 3.5 | 3.3 | 3 | 5.1 | 6 | 3.9 |
| errores | 0 | 0 | 0 | 1 | 0 | 0 |
| letras 1 tic | 9 | 12 | 11 | 12 | 12 | 13 |
| letras 2 tic | 5 | 5 | 4 | 2 | 6 | 8 |
| letras 3 tic | 14 | 6 | 6 | 6 | 10 | 10 |
| letras 4 tic | 11 | 11 | 6 | 8 | 4 | 5 |
| letras 5 tic | 9 | 3 | 8 | 4 | 4 | 8 |
| velocidad | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | |
| encuentro facilemnte mi clave | 4 | | | | | |
| velocidad de las letras adecuada | 4 | | | | | |
| acelerar/ralentizar | | | | | | |

*Tabla 7*

**Conclusiones**

Los nuevos alfabetos no interfieren con lo que teníamos, tampoco hemos notado una mejoría en la ofuscación, para mejorar la ofuscación mezclaremos los alfabetos en la próxima iteración.

## 6.7 Versión 1.8

**Introducción**

Mezclaremos los alfabetos negro y rojo, intentamos comprobar si existe una mezcla en la solución sin perjuicio de lo que ya teníamos.

**Cambios**

- Mezcla de alfabetos negro y rojo

La Figura 11 muestra la pantalla principal de la versión 1.8.

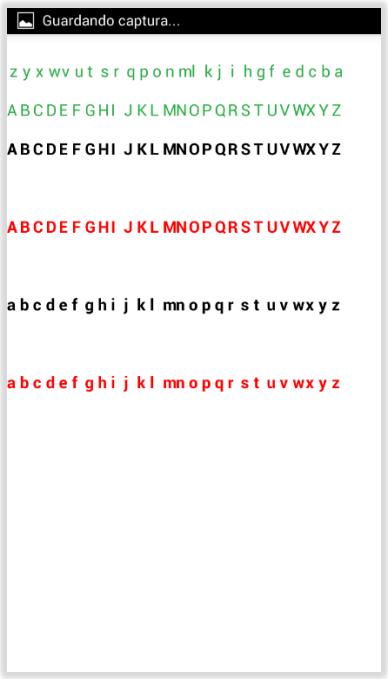


*Figura 11. Pantalla principal v1.8*

**Resultados**

Hemos conseguido nuestro objetivo conseguimos aumentar la ofuscación haciendo que aparezcan letras rojas en las pulsaciones, parece cómodo.

En laTabla 8 se recogen los resultados cuantitativos de la evaluación.

| Version | 1.8 | | | | | |
|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | |
| Tiempo | 3.9 | 3.4 | 3.2 | 3.8 | 5.6 | 3.3 |
| errores | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 10 | 9 | 8 | 12 | 9 | 10 |
| letras 2 tic | 4 | 5 | 6 | 4 | 5 | 4 |
| letras 3 tic | 6 | 5 | 6 | 3 | 6 | 4 |
| letras 4 tic | 5 | 7 | 8 | 6 | 6 | 9 |
| letras 5 tic | 8 | 9 | 6 | 2 | 5 | 7 |
| velocidad | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | |
| encuentro facilemnte mi clave | 5 | | | | | |
| velocidad de las letras adecuada | 4 | | | | | |
| acelerar/ralentizar | | | | | | |

*Tabla 8*

**Conclusiones**

Intentaremos acercar algo más los alfabetos para aumentar más la ofuscación.

## 6.8 Versión 1.9

**Introducción**

- Juntamos los alfabetos rojo y negro

La Figura 12 muestra la pantalla principal de la versión 1.9.

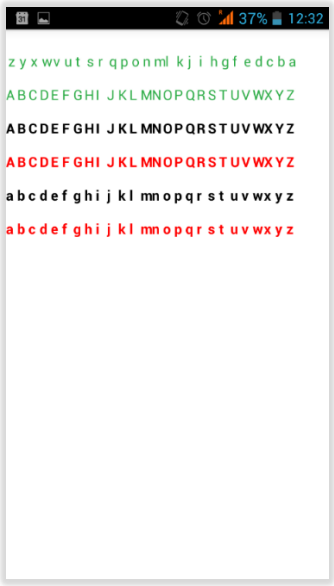


*Figura 12. Pantalla principal v1.9*

**Resultados**

Se puede manejar con entrenamiento, pero no es cómodo (desestimado).

Desestimado.

En la Tabla 9 se recogen los resultados cuantitativos de la evaluación.

| Version | 1.9 | | | | | | 1.9 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | | DavID | | |
| Usuario | Carlos | | | David | | | Carlos | | | David | | |
| Tiempo | 13 | 3.7 | 4.9 | 8 | 5.2 | 2.7 | 3.7 | 3.8 | 8.3 | 2.9 | 3.6 | 3.1 |
| errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 17 | 15 | 15 | 13 | 12 | 13 | 14 | 8 | 14 | 11 | 13 | 11 |
| letras 2 tic | 8 | 9 | 4 | 7 | 10 | 10 | 6 | 5 | 6 | 6 | 4 | 7 |
| letras 3 tic | 8 | 12 | 12 | 11 | 8 | 13 | 14 | 12 | 9 | 7 | 8 | 10 |
| letras 4 tic | 5 | 14 | 6 | 15 | 7 | 10 | 11 | 8 | 10 | 13 | 9 | 8 |
| letras 5 tic | 5 | 8 | 11 | 2 | 5 | 6 | 10 | 13 | 12 | 16 | 5 | 14 |
| velocidad | 5 | | | | | | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | | 5 | | | | | |
| encuentro facilemnte mi clave | 3 | | | | | | 3.5 | | | | | |
| velocidad de las letras adecuada | 3 | | | | | | 4 | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | |

*Tabla 9*

**Conclusiones**

Dado que no mejoramos acercando las letras intentaremos ahora alejándolas.

Las separaremos un poco más que en la versión 1.8.

## 6.9 Versión 1.10

**Introducción**

Como ya dijimos en la versión anterior, en esta exploraremos el comportamiento de sistema cuando separamos los alfabetos.

**Cambios**

- Separamos alfabetos

La Figura 13 muestra la pantalla principal de la versión 1.10.


*Figura 13. Pantalla principal v1.10*

**Resultados**

Cómodo, pero no aparecen letras rojas en las pulsaciones, por lo que no ofusca correctamente y lo descartaremos.

Descartada.

**Conclusiones**

Aunque esta versión es más cómoda que la 1.8 tiene más problemas, por lo que utilizaremos la 1.11 (similar a la 1.8) para probar con otros usuarios.

## 6.10 Versión 1.11

**Introducción**

Probaremos esta versión con otros usuarios, intentamos validar los resultados obtenidos, para ello, hemos retomado la posición de los alfabetos con la que más a gusto nos encontrábamos (Figura 14).

**Cambios**

- Separación de la versión 1.8
- Probando con usuarios externos

La Figura 14 muestra la pantalla principal de la versión 1.11.

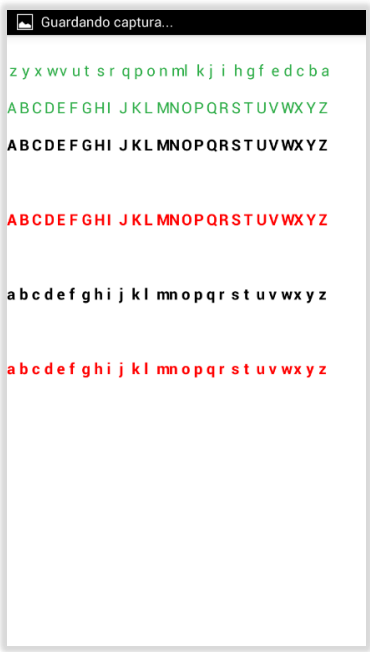


*Figura 14. Pantalla principal v1.11 (misma que v1.8)*

**Resultados**

Los usuarios parecen coincidir en que

- Más cómodo que antes
- Hay que entrenar la clave
- Los movimientos predecibles son preferidos
- Se siguen confundiendo las letras parecidas
- Se podrían añadir números
- Añadir diferentes niveles de dificultad

En las tablas (Tabla 10, *Tabla 11* y Tabla 12) se recogen los resultados cuantitativos de esta evaluación.

| Version | 1.11. | | | | | | 1.11. | | | | | | 1.11. | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | | DavID | | | AJmxt | | | AJmxt | | |
| Usuario | Carlos | | | David | | | Carlos | | | David | | | Conchi | | | Conchi | | |
| Tiempo | 4.4 | 3.5 | 5.7 | 4.1 | 4.4 | 4.3 | 4.1 | 4.3 | 4.2 | 6.6 | 8.2 | 4.6 | 22 | 12 | 26 | 17 | 9.3 | 11 |
| Errores | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| letras 1 tic | 12 | 14 | 13 | 12 | 14 | 11 | 10 | 12 | 10 | 7 | 11 | 10 | 6 | 8 | 8 | 4 | 9 | 11 |
| letras 2 tic | 5 | 4 | 4 | 5 | 3 | 5 | 3 | 6 | 5 | 7 | 5 | 5 | 5 | 2 | 8 | 2 | 14 | 10 |
| letras 3 tic | 9 | 8 | 12 | 7 | 6 | 6 | 12 | 12 | 7 | 7 | 10 | 13 | 7 | 9 | 8 | 3 | 7 | 8 |
| letras 4 tic | 6 | 10 | 3 | 9 | 9 | 6 | 8 | 8 | 8 | 4 | 6 | 8 | 13 | 5 | 5 | 3 | 4 | 4 |
| letras 5 tic | 4 | 5 | 2 | 6 | 7 | 6 | 8 | 5 | 7 | 4 | 6 | 8 | 6 | 6 | 3 | 2 | 5 | 7 |
| Velocidad | 5 | | | | | | 5 | | | | | | 5 | | | | | |
| se diferencian bien los colores | 5 | | | | | | 5 | | | | | | 4 | | | | | |
| encuentro facilemnte mi clave | 4 | | | | | | 3.5 | | | | | | 3 | | | | | |
| velocidad de las letras adecuada | 3 | | | | | | 3 | | | | | | 4.5 | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | | | | | | | |

Tabla 10

| Version | 1.11. | | | | | | 1.11. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | Marci | | | MaRcI | | | Coche | | | Coche | | | Coche | | |
| Usuario | Amando | | | Amando | | | Tejo | | | Tejo | | | Tejo | | |
| Tiempo | 16 | 28 | 5 | 13 | 7 | 10 | 6 | 4 | 5 | 9 | 13 | 15 | 10 | 24 | 7 |
| Errores | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 13 | 13 | 13 | 7 | 6 | 5 | 5 | 6 | 6 | 12 | 11 | 14 | 5 | 5 | 5 |
| letras 2 tic | 4 | 6 | 6 | 1 | 3 | 1 | 5 | 8 | 6 | 10 | 11 | 17 | 14 | 15 | 12 |
| letras 3 tic | 13 | 9 | 11 | 7 | 10 | 7 | 12 | 7 | 11 | 8 | 13 | 5 | 6 | 7 | 16 |
| letras 4 tic | 4 | 2 | 5 | 1 | 0 | 9 | 11 | 8 | 11 | 9 | 4 | 2 | 5 | 6 | 8 |
| letras 5 tic | 6 | 5 | 11 | 4 | 9 | 4 | 11 | 12 | 7 | 9 | 4 | 8 | 9 | 4 | 3 |
| Velocidad | 5 | | | | | | 5 | | | | | | | | |
| se diferencian bien los colores | 5 | | | | | | 4 | | | | | | | | |
| encuentro facilemnte mi clave | 2 | | | | | | 3 | | | | | | | | |
| velocidad de las letras adecuada | 4 | | | | | | 3 | | | | | | | | |
| acelerar/ralentizar | | | | | | | ralentizar un poco | | | | | | | | |

*Tabla 11*

| Version | 1.11. | | | | | | 1.11. | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | POTAT | | | POTaT | | | JORGE | | | JoRge | | |
| Usuario | Diego | | | Diego | | | Jorge | | | Jorge | | |
| Tiempo | 11 | 12 | 9 | 24 | 9 | 14 | 13 | 10 | 7 | 16 | 16 | 17 |
| Errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 5 | 8 | 3 | 3 | 9 | 7 | 11 | 14 | 10 | 13 | 13 | 10 |
| letras 2 tic | 4 | 1 | 5 | 3 | 5 | 2 | 3 | 8 | 11 | 8 | 2 | 9 |
| letras 3 tic | 7 | 12 | 1 | 9 | 4 | 5 | 9 | 11 | 6 | 7 | 2 | 5 |
| letras 4 tic | 7 | 1 | 2 | 2 | 1 | 1 | 8 | 5 | 12 | 4 | 10 | 6 |
| letras 5 tic | 4 | 7 | 5 | 8 | 2 | 3 | 2 | 4 | 1 | 2 | 7 | 1 |
| Velocidad | 5 | | | | | | 5 | | | | | |
| se diferencian bien los colores | 4 | | | | | | 5 | | | | | |
| encuentro facilemnte mi clave | 3 | | | | | | 2 | | | | | |
| velocidad de las letras adecuada | 4 | | | | | | 3 | | | | | |
| acelerar/ralentizar | ralentizar un poco | | | | | | ralentizar un poco | | | | | |

*Tabla 12*

**Análisis**

Los usuarios se sienten más a gusto que antes, pero parecen seguir teniendo problemas con las letras parecidas.

Existe cierta predilección por las letras fáciles de encontrar (movimientos predecibles).

Practicar la clave se hace más necesario cuanto más lejos están las letras.

Sugieren mejorar usando números y diferentes niveles de dificultad.

**Conclusiones**

Utilizaremos diferentes tipos de textura para los alfabetos más difíciles de diferenciar, intentaremos introducir nuevos alfabetos predecibles.

También introduciremos números y un selector de velocidades, de forma que cada usuario pueda elegir su velocidad de movimiento.

# Capítulo 7. Versión 2, Predice el movimiento

En esta versión hemos hecho grandes modificaciones del programa, entre ellas destaca la inclusión de manejadores (objetos capaces de mover un conjunto de letras) de forma grupal, la opción de mover las letras en el selector de contraseña y un selector de velocidades.

**Cambios**

- Añadido checkbox para mover las letras al seleccionar contraseña
- Velocidad modificable
- Añadidos manejadores de letras para movimientos más complejos (permiten movimientos grupales)
- Movimiento gusano (movimiento grupal)
- Movimiento laser (movimiento grupal)
- Botón atrás saca directamente de la pantalla en vez de borrar una posición (los usuarios pidieron poder cancelar por completo y volver a empezar)
- Añadida fuente para las minúsculas (así evitaremos problemas al confundir letras como por ejemplo i-I/l-L entre sí)
- Añadidos números (sin activar)
- Añadido movimiento viento lateral (similar al movimiento matrix pero de forma lateral)
- Tamaño de las letras aumentado de forma porcentual
- Aumentado el número de movimientos por segundo (parece que se mueve más fluido)
- Añadidos manejadores a selección de contraseña (para poder elegir las letras como parte de la contraseña)
- Añadido manejador de bloque
- Añadido número de pasos variable para manejadores (para aumentar la fluidez del movimiento)

## 7.1 Versión 2.10

**Introducción**

Continuaremos desde el desarrollo tal y como lo dejamos, al cual, le añadiremos unas letras en movimiento lateral predecible en color azul.

También añadiremos textura en las letras minúsculas, de esta forma intentamos eliminar los problemas que existen al confundir las letras que se escriben igual en mayúscula que en minúscula.

**Cambios**

- Mismas letras de la versión 1.12
- Se añaden las letras de movimiento predecible “viento lateral”
- Textura para las letras minúsculas

La Figura 14 Figura 15 muestra la pantalla principal de la versión 2.10.

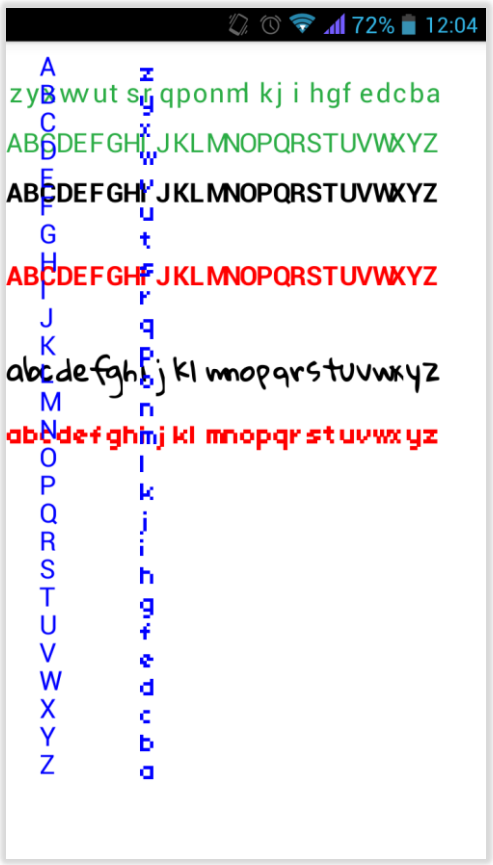


*Figura 15. Pantalla principal v2.10*

**Resultados**

Los nuevos alfabetos son fáciles de encontrar, además, con los colores elegidos es muy fácil discriminar entre los diferentes alfabetos.

El poder modificar la velocidad es algo muy positivo, y nos damos cuenta de que por debajo de 5 no hay problema, sin embargo, al subir a 7 se vuelve demasiado difícil seguir las letras.

Igualmente el cambio de textura se percibe como algo beneficioso, aunque en una ocasión se confundieron la v y la u del mismo alfabeto.

Los datos obtenidos durante la evaluación se encuentran en las tablas (Tabla 13, Tabla 14).

| Version | 2.10. | | | | | | 2.10. | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | | DavID | | |
| Usuario | Carlos | | | Carlos | | | Carlos | | | | | |
| Tiempo | 5.1 | 3 | 2.7 | 3.8 | 3.5 | 2.7 | 4 | 4.4 | 3.5 | 3.3 | 3.3 | 3.7 |
| Errores | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 9 | 8 | 6 | 10 | 10 | 8 | 7 | 11 | 9 | 9 | 10 | 11 |
| letras 2 tic | 6 | 5 | 6 | 4 | 5 | 6 | 4 | 6 | 5 | 5 | 5 | 4 |
| letras 3 tic | 8 | 8 | 6 | 4 | 8 | 7 | 15 | 10 | 13 | 7 | 15 | 12 |
| letras 4 tic | 9 | 14 | 17 | 9 | 7 | 14 | 8 | 13 | 12 | 8 | 5 | 5 |
| letras 5 tic | 9 | 8 | 13 | 7 | 6 | 15 | 7 | 10 | 4 | 11 | 9 | 9 |
| Velocidad | 5 | | | 7 | | | 5 | | | 7 | | |
| se diferencian bien los colores | 5 | | | | | | 5 | | | | | |
| encuentro facilemnte mi clave | 4 | | | | | | 4 | | | | | |
| velocidad de las letras adecuada | 5 | | | | | | 5 | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | |

*Tabla 13*

| Version | 2.10. | | | | | | 2.10. | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | DavID | | |
| Usuario | Carlos | | | Carlos | | | Carlos | | Carlos | | |
| Tiempo | 6.5 | 3.9 | 4.6 | 4 | 3.7 | 3.5 | 3.4 | 9 | 4.3 | 6.4 | 4.1 |
| Errores | 0 | 0 | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 |
| letras 1 tic | 10 | 11 | 8 | 13 | 11 | 14 | 10 | | 16 | 17 | 14 |
| letras 2 tic | 16 | 14 | 17 | 15 | 14 | 17 | 15 | | 3 | 3 | 3 |
| letras 3 tic | 9 | 6 | 5 | 3 | 2 | 3 | 7 | | 15 | 18 | 18 |
| letras 4 tic | 7 | 4 | 5 | 6 | 6 | 8 | 6 | | 13 | 12 | 15 |
| letras 5 tic | 3 | 5 | 3 | 10 | 10 | 8 | 5 | | 9 | 7 | 10 |
| Velocidad | 5 | | | 3 | | | 7 | | 5 | | |
| se diferencian bien los colores | 5 | | | | | | | | 5 | | |
| encuentro facilemnte mi clave | 3 | | | | | | | | 4 | | |
| velocidad de las letras adecuada | 3 | | | | | | | | 5 | | |
| acelerar/ralentizar | ralentiza | | | | | | | | | | |

*Tabla 14*

**Conclusiones**

Dado que la versión ha sido un éxito continuaremos introduciendo los nuevos manejadores.

## 7.2 Versión 2.11

**Introducción**

Tal y como se dijo en la versión anterior, se introducirán los manejadores de movimiento disponibles (movimiento laser y gusano). Además, para realzarlos colores claros, cambiaremos el fondo de blanco a gris claro.

**Cambios**

- Se ha puesto el fondo a gris claro (se ve mucho mejor)
- Se ha añadido movimiento gusano (amarillo)
- Se ha añadido movimiento laser magenta

La Figura 16 muestra la pantalla principal de la versión 2.11.


*Figura 16. Pantalla principal v2.11*

**Resultados**

A los usuarios les resulta mas cómodo que versiones anteriores, además, valoran bien los movimientos predecibles (resultan fáciles de localizar) y el fondo gris.

Algunos usuarios consideran las letras mayúsculas más fáciles de encontrar, los cambios de textura ayudan, aunque aún hay letras difíciles de diferenciar.

El poder modificar la velocidad ha sido bien recibido.

Por otra parte, existe una concentración desigual de letras, en ocasiones al empezar se crean “pegotes” de letras del mismo color, y no se diferencian bien las letras que lo forman (en la parte superior) y/o en la parte inferior hay pocas letras, lo que provoca falta de ofuscación.

Otra de las cosas a mejorar es el estado, no se ve suficientemente bien.

Los datos tomados se encuentran en las tablas 15-18 (Tabla 15, Tabla 16, Tabla 17 y Tabla 18)

| Version | 2.11 | | | | | | 2.11 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | | DavID | | |
| Usuario | David | | | David | | | David | | | | | |
| Tiempo | 3.5 | 3.7 | 3.5 | 3.7 | 4.7 | 2.7 | 4.9 | 5.8 | 5.3 | 6.4 | 6.3 | 9.2 |
| Errores | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 13 | 17 | 19 | 16 | 16 | 15 | 14 | 13 | 12 | 15 | 17 | 17 |
| letras 2 tic | 5 | 7 | 7 | 4 | 3 | 4 | 4 | 4 | 3 | 3 | 4 | 5 |
| letras 3 tic | 6 | 6 | 8 | 6 | 6 | 6 | 16 | 17 | 12 | 8 | 9 | 11 |
| letras 4 tic | 17 | 15 | 16 | 3 | 12 | 12 | 13 | 8 | 16 | 7 | 4 | 8 |
| letras 5 tic | 15 | 11 | 18 | 12 | 9 | 7 | 13 | 11 | 12 | 11 | 9 | 3 |
| Velocidad | 5 | | | 7 | | | 5 | | | 7 | | |
| se diferencian bien los colores | 5 | | | | | | | | | | | |
| encuentro facilemnte mi clave | 5 | | | | | | | | | | | |
| velocidad de las letras adecuada | 5 | | | | | | | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | |

*Tabla 15*

| Version | 2.11 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | DavID | | | DavID | | |
| Usuario | David | | | David | | | David | | |
| Tiempo | 3.1 | 4.5 | 3.7 | 6.6 | 6.7 | 5 | 11 | 6 | 5.1 |
| Errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 22 | 17 | 17 | 20 | 19 | 16 | 20 | 21 | 20 |
| letras 2 tic | 2 | 4 | 4 | 4 | 1 | 1 | 2 | 2 | 1 |
| letras 3 tic | 16 | 18 | 18 | 18 | 17 | 20 | 12 | 9 | 9 |
| letras 4 tic | 10 | 13 | 9 | 14 | 18 | 15 | 10 | 10 | 7 |
| letras 5 tic | 5 | 10 | 10 | 11 | 11 | 10 | 6 | 3 | 7 |
| Velocidad | 5 | | | 3 | | | 7 | | |
| se diferencian bien los colores | 5 | | | 5 | | | | | |
| encuentro facilemnte mi clave | 5 | | | 5 | | | | | |
| velocidad de las letras adecuada | 5 | | | 5 | | | | | |
| acelerar/ralentizar | | | | | | | | | |

*Tabla 16*

| Version | 2.11 | | | | | | | | | 2.11 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DavID | | | rMCfA | | | rMCfA | | | Davld | | |
| Usuario | Diego | | | | | | | | | David | | |
| Tiempo | 10 | 9.8 | 7.9 | 8.2 | 16 | 11 | 6.7 | 5.2 | 5.3 | 5.6 | 4.3 | 5.2 |
| Errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 14 | 15 | 16 | 8 | 9 | 7 | 9 | 8 | 7 | 12 | 12 | 14 |
| letras 2 tic | 6 | 6 | 6 | 20 | 21 | 24 | 19 | 21 | 21 | 4 | 4 | 8 |
| letras 3 tic | 13 | 11 | 7 | 11 | 11 | 15 | 14 | 7 | 12 | 19 | 18 | 11 |
| letras 4 tic | 4 | 3 | 9 | 3 | 7 | 8 | 7 | 10 | 4 | 11 | 5 | 10 |
| letras 5 tic | 7 | 8 | 16 | 12 | 9 | 12 | 10 | 9 | 19 | 9 | 15 | 13 |
| Velocidad | 5 | | | 5 | | | 7 | | | 5 | | |
| se diferencian bien los colores | 5 | | | | | | | | | 5 | | |
| encuentro facilemnte mi clave | 5 | | | | | | | | | 5 | | |
| velocidad de las letras adecuada | 5 | | | | | | | | | 5 | | |
| acelerar/ralentizar | | | | | | | | | | | | |

*Tabla 17*

| Version | 2.11 | | | | | | | | | 2.11 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | ArTUr | | | ArTur | | | ARtUR | | | ABcDe | | | aSJmj | | |
| Usuario | Arturo | | | | | | | | | Antonio | | | | | |
| Tiempo | 4.3 | 5.4 | 7.5 | 7 | 7.5 | 7.5 | 7.2 | 7.5 | 11 | 3 | 6.1 | 1.9 | 12 | 9.6 | 5.1 |
| Errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 9 | 9 | 11 | 5 | 9 | 14 | 8 | 13 | 13 | 9 | 13 | 14 | 9 | 8 | 8 |
| letras 2 tic | 9 | 10 | 13 | 9 | 13 | 11 | 4 | 4 | 7 | 20 | 16 | 16 | 8 | 6 | 6 |
| letras 3 tic | 19 | 16 | 11 | 11 | 12 | 9 | 9 | 9 | 14 | 15 | 7 | 10 | 9 | 13 | 13 |
| letras 4 tic | 11 | 16 | 12 | 12 | 11 | 12 | 2 | 8 | 6 | 19 | 14 | 19 | 14 | 9 | 13 |
| letras 5 tic | 13 | 18 | 12 | 16 | 13 | 12 | 2 | 1 | 5 | 17 | 14 | 17 | 8 | 8 | 1 |
| Velocidad | 5 | | | 5 | | | 5 | | | 5 | | | 5 | | |
| se diferencian bien los colores | 4 | | | | | | | | | 5 | | | | | |
| encuentro facilemnte mi clave | 3 | | | | | | | | | 4 | | | | | |
| velocidad de las letras adecuada | 5 | | | | | | | | | 5 | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | | | | |

Tabla 18

**Análisis**

La iteración parece ir por buen camino, el cambio de fondo ha dado mejores resultados de los esperados y los movimientos grupales no parecen molestar.

Dada la buena acogida de los movimientos predecibles se puede considerar la introducción de algún otro alfabeto predecible.

El tiempo requerido es algo positivo, no suele pasar de 10 segundos incluso con usuarios nuevos.

Hemos detectado un efecto negativo en usuarios primerizos, parece dar miedo enfrentarse a tanto caos. Este efecto se elimina una vez se usa la aplicación, volviéndose bastante positivo.

**Conclusiones**

Intentaremos mezclar las letras negras y rojas para evitar los grumos.

Mantendremos el fondo gris.

Mantendremos los movimientos predecibles laser y gusano, así como el movimiento lateral.

## 7.3 Versión 2.12

**Introducción**

En este caso, y siendo la última iteración haremos una exploración sobre cuál es la impresión de los usuarios con cada tipo de movimiento.

Además, para eliminar los huecos demasiado grandes hemos duplicado la sección superior en la mitad inferior, de esta forma evitaremos la falta de ofuscación que caracterizaba la parte inferior a la vez que duplicamos letras (el repetir alfabetos fue una petición expresa de algunos usuarios).

**Cambios**

- Tanto el movimiento gusano como el movimiento laser son ahora amarillos
- Se ha duplicado la sección superior en la parte inferior (de esta forma evitamos el vacío de la parte inferior)
- Se ha separado las letras verdes para seguir el mismo patrón en la mitad superior e inferior

La Figura 17 muestra la pantalla principal de la versión 2.12.

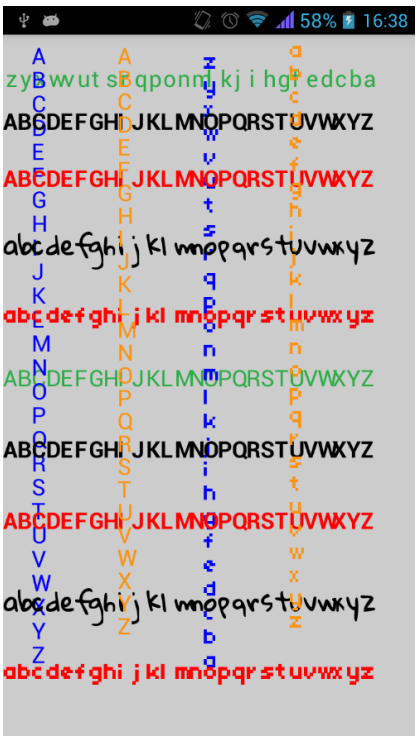


*Figura 17. Pantalla principal v2.12*

**Resultados**

En este caso, y siendo la última iteración haremos una exploración sobre cuál es la impresión de los usuarios con cada tipo de movimiento.

- Los usuarios coinciden en que los movimietos matrix son los más difíciles.
- También coinciden en que hay demasiadas letras.
- Hay tantas letras que se ha tenido que bajar la velocidad de 5 a 4 porque si no se saturaba el dispositivo.

Los datos de esta evaluación se encuentran en Tabla 19, Tabla 20 y **¡Error! No se encuentra el origen de la referencia.**.

| Version | 2.1 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | DiEgO | DiEgO | | DiEgO | | DiEgO | | DiEgO | | DiEgO | | | | | |
| Usuario | Diego | | | | | | | | | | | | | | |
| Tiempo | 6.2 | 4.4 | 4 | 4.5 | 5 | 6.4 | 4.6 | 4.2 | 3.6 | 5.8 | 3.6 | 4.3 | 5.5 | 4 | 4.2 |
| Errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 14 | 14 | 12 | 12 | 10 | 9 | 14 | 15 | 11 | 16 | 13 | 13 | 13 | 11 | 10 |
| letras 2 tic | 11 | 10 | 11 | 16 | 18 | 17 | 10 | 9 | 10 | 14 | 12 | 11 | 10 | 8 | 8 |
| letras 3 tic | 8 | 11 | 14 | 9 | 13 | 13 | 12 | 12 | 19 | 18 | 19 | 18 | 11 | 12 | 11 |
| letras 4 tic | 5 | 12 | 10 | 14 | 14 | 16 | 10 | 10 | 10 | 9 | 12 | 9 | 9 | 9 | 13 |
| letras 5 tic | 12 | 10 | 7 | 7 | 9 | 11 | 18 | 19 | 17 | 2 | 6 | 3 | 11 | 12 | 12 |
| Velocidad | 4 | | | | | | | | | | | | | | |
| los colores | 4 | | | | | | | | | | | | | | |
| encuentro facilemnte mi clave | 4 | | | | | | | | | | | | | | |
| velocidad de las letras adecuada | 4 | | | | | | | | | | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | | | | |

*Tabla 19*

| Version | 2.1 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | COCHE | CocHe | | CocHe | | CocHe | | COchE | | | | | | | |
| Usuario | Tejo | | | | | | | | | | | | | | |
| Tiempo | 3 | 3.4 | 3.7 | 7.3 | 5.8 | 4.5 | 4.5 | 3.6 | 3.1 | 5.1 | 4.5 | 4.9 | 6.5 | 3.4 | 3.3 |
| errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| letras 1 tic | 14 | 8 | 9 | 12 | 12 | 11 | 11 | 11 | 14 | 13 | 15 | 15 | 10 | 9 | 9 |
| letras 2 tic | 16 | 8 | 10 | 15 | 15 | 11 | 14 | 16 | 12 | 8 | 7 | 11 | 14 | 14 | 14 |
| letras 3 tic | 10 | 13 | 12 | 16 | 13 | 11 | 20 | 18 | 15 | 10 | 13 | 13 | 16 | 17 | 16 |
| letras 4 tic | 14 | 20 | 17 | 18 | 14 | 17 | 12 | 9 | 10 | 12 | 11 | 14 | 8 | 11 | 10 |
| letras 5 tic | 10 | 10 | 11 | 13 | 13 | 15 | 6 | 13 | 11 | 16 | 10 | 12 | 14 | 13 | 11 |
| velocidad | 4 | | | | | | | | | | | | | | |
| colores | 4 | | | | | | | | | | | | | | |
| encuentro | 4 | | | | | | | | | | | | | | |
| velocidad | 5 | | | | | | | | | | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | | | | |

Tabla 20

| Version | 2.1 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clave | JorgE | | | JorgE | | | JorgE | | | JorgE | | | JORge | | |
| Usuario | Jorge | | | | | | | | | | | | | | |
| Tiempo | 5.6 | 8.2 | 4.9 | 9.3 | 14 | 6.7 | 4.9 | 12 | 7.1 | 8.9 | 8.2 | 6.7 | 9.2 | 7.9 | 8.3 |
| errores | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| letras 1 tic | 10 | 10 | 11 | 10 | 11 | 8 | 8 | 7 | 6 | 8 | 8 | 7 | 9 | 10 | 10 |
| letras 2 tic | 8 | 10 | 10 | 1 | 15 | 13 | 14 | 16 | 16 | 12 | 8 | 11 | 15 | 12 | 15 |
| letras 3 tic | 10 | 10 | 13 | 14 | 14 | 21 | 13 | 19 | 14 | 6 | 12 | 13 | 10 | 9 | 9 |
| letras 4 tic | 16 | 14 | 11 | 10 | 9 | 8 | 9 | 8 | 11 | 15 | 5 | 12 | 10 | 13 | 15 |
| letras 5 tic | 16 | 2 | 4 | 13 | 11 | 9 | 10 | 15 | 14 | 15 | 9 | 11 | 10 | 13 | 8 |
| velocidad | 4 | | | | | | | | | | | | | | |
| se diferencian bien los colores | 5 | | | | | | | | | | | | | | |
| encuentro facilemnte mi clave | 2 | | | | | | | | | | | | | | |
| velocidad de las letras adecuada | 4 | | | | | | | | | | | | | | |
| acelerar/ralentizar | | | | | | | | | | | | | | | |

Tabla 21

**Análisis**

Los usuarios han hecho notar que añadir tantas letras de golpe ha sido algo precipitado.

Dada la alta cantidad de letras que existen hemos encontrado algo inesperado y es que las letras matrix, que hasta el momento no habían tenido problemas al ser identificadas empiezan a tenerlos y debería ser tenido en cuenta en el futuro.

Otra de las cosas que hemos notado al aumentar tanto el número de letras es que hemos llegado al punto en el que el dispositivo empieza a encontrar problemas con la carga.

**Conclusiones**

De existir tiempo para una futura iteración las medidas que tomar serían: dar más visibilidad a las letras matrix, y bajar el número de letras, quizá bajando el número de alfabetos duplicados mientras mantenemos una densidad de letras uniforme.

# Parte IV Conclusiones

## Capítulo 8. Conclusiones del proyecto

Tras la realización del trabajo se puede concluir de que, se han conseguido los objetivos planteados en el presente proyecto.

Cuando empezamos con este trabajo intentábamos demostrar la viabilidad de Entrophy. Una vez llevado a cabo el prototipo, probado diferentes implementaciones, evaluado los prototipos con usuarios y analizado los datos podemos determinar que sí, el concepto puede ser viable.

Aún quedan muchas cosas por hacer, el prototipo no es todo lo agradable al usuario como debería, de hecho se ve áspero, lo que produce un rechazo en primera instancia.

En un primer momento los usuarios se asustan al ver tantas letras en movimiento, intentan absorber todo lo que se muestra en la pantalla en vez de centrarse en su contraseña, lo que produce un efecto de agobio y un rechazo asociado.

Este efecto desaparece rápidamente cuando utilizan la aplicación. Tras su uso la mayoría de usuarios cambia su punto de vista, empiezan a verlo como un juego, y al "aprender" a centrarse solo en su contraseña y olvidarse del resto el efecto de agobio existente en un primer momento desaparece.

### Prueba de seguridad

Para probar la seguridad del proyecto se buscó el caso más desfavorable posible, grabación directa y se realizó una prueba lo más realista posible.

La prueba consiste en la grabación de un vídeo de introducción de contraseña en Entrophy. Después, se pide a los usuarios (supuestos atacantes) que intentaran adivinar la contraseña utilizando la grabación (podían verla tantas veces como quisieran).

Tras visualizarlo gran cantidad de veces ninguno acertó ni un solo dígito de la contraseña introducida.

### Seguridad

Desde el punto de vista de la seguridad el proyecto ha demostrado conseguir lo que se buscaba. Para ello utilizaremos las bases teóricas formuladas en el capítulo 1 junto con los valores obtenidos en la última implementación (2.12) [Datos evaluaciones.xlsx anexo].

$$\left(\frac{\text{nº dígitos candidatos por pulsacion}}{\text{nº dígitos total}}\right)^{nº\ de\ digitos\ clave} = \mathbb{III}$$

Gran parte de nuestra seguridad reside en el número de letras candidatas para una pulsación, por ello y por la naturaleza dinámica del sistema es importante estudiar cómo se distribuye esta variable en casos reales. En nuestro caso la distribución de letras candidatas por pulsación está representada en la Figura 18 (sobre 225 muestras). En la Tabla 22 se muestra la distribución de letras candidatas por pulsación, podemos ver como sigue una distribución aproximadamente gaussiana cuya media es 11.77.

| | | |
|---|---|---|
| Entre 10 y 14 letras | 57 | % |
| Entre 8 y 14 letras | 73 | % |
| Entre 8 y 16 letras | 85 | % |
| Entre 6 y 18 letras | 94 | % |
| Entre 1 y 21 letras | 100 | % |

*Tabla 22*


*Figura 18*

A la vista de lo datos tomaremos la media aritmética como el número medio de **letras candidatas por pulsación** en la última implementación: **11.77**.

En la última implementación existen 14 alfabetos, de los cuales 4 están repetidos, es decir, existen 10 alfabetos diferentes. Dado que cada alfabeto tiene 27 letras el número es de **270 letras diferentes**, además existen otras 108 letras repetidas por lo que, con ánimo de dar una visión pesimista a nuestro resultado, no contaremos con ellas.

Por si fuera poco, además hemos añadido **un dígito más a la clave**.

Comparando la seguridad $\mathbb{III}$ anterior con la actual:

$$\left(\frac{1}{10}\right)^4 = 10^{-4} \quad << \quad \left(\frac{11.77}{270}\right)^5 = 1.574213 * 10^{-7}$$

En el capítulo 1 describimos $\mathbb{III}$ como el inverso de la seguridad que somos capaces de ofrecer, al disminuir en 3 órdenes de magnitud desde la seguridad estándar a la actual vemos en esta una ganancia de seguridad igual (tres órdenes superior).

Por otra parte, aún en caso de haber sido espiados mantenemos un grado de seguridad considerable. Comparando nuestra seguridad tras haber sido espiados (y autentificados) sin usar Entrophy solo hay 1 posibilidad,

por lo que la probabilidad sigue siendo del 100%, sin embargo, usándolo conseguimos que existan 225882 posibilidades, por lo que el número de probabilidades disminuye hasta 0.044%.

$$\left(\frac{1}{\text{Dígitos candidatos por pulsación}}\right)^{Número\ de\ \ pulsaciones} = \varrho$$

$$\left(\frac{\mathbf{1}}{\mathbf{1}}\right)^{\mathbf{5}} = \mathbf{100}\% \quad >> \quad \left(\frac{\mathbf{1}}{\mathbf{11.77}}\right)^{\mathbf{5}} = \mathbf{0.044}\%$$

## Usabilidad

La eficacia del sistema en la introducción de la clave ha quedado probada, ya que el tiempo requerido está dentro de lo aceptable, en algún caso sube a 13 segundos, pero en general se sitúa en torno a los 5 segundos, siendo el tiempo medio 5,75 segundos.

Consideramos un logro conseguir estos tiempos, ya que en las primeras versiones se situaban en torno a los 50 segundos. Es todo un reto y una demostración de lo que el sistema puede mejorar utilizando los principios Guestalt y las bases teóricas del capítulo 3.

Otro aspecto importante para el usuario es que el número de errores es muy bajo, no llega a ser 0 (dado que es un teclado táctil es más propenso a errores que es un teclado tradicional) pero en el conjunto de datos sobre el que estamos realizando la evaluación solo ha habido dos errores, lo que supone un 4.4%, cifra muy por debajo de lo esperado para considerar válido este prototipo.

## Viabilidad

Aún queda mucho por hacer, seguir mejorando la interfaz, seguir buscando la forma de añadir información y la mejor forma de permitir al usuario encontrar lo que busca ofuscándolo al mismo tiempo para el resto, pero se ha conseguido la finalidad del proyecto, demostrar la viabilidad del concepto.

En el punto en el que nos encontramos, a la vista de los datos obtenidos y tras haber realizado un estudio del sistema concluimos:

- El sistema proporciona la seguridad por ofuscación deseada.
- El sistema proporciona la seguridad por conjuntos deseada.
- El sistema proporciona seguridad y permite introducir contraseñas de forma segura frente a atacantes “man in the middle”, “shoulder sourfing” y grabación directa tal y como se pretendía demostrar.
- Existe un efecto rechazo en primera instancia, este efecto rechazo es causado por la gran cantidad de información que se muestra de golpe, pero se elimina rápidamente con el uso de la aplicación.
- El sistema ha sido un éxito como prueba de concepto, pero debe ser mejorado para poder ser considerado como un producto comercial.
- El sistema es viable.

# Bibliografía

[1] http://www.openwall.com/john/

[2] https://plus.google.com/+QuantumAILab/posts

[3] https://www.youtube.com/watch?v=CMdHDHEuOUE

[4] http://www.elladodelmal.com/2013/08/podria-http-s-acabar-con-el-esiponaje.html

[5]http://www.genbeta.com/seguridad/la-nsa-habria-estado-suplantando-a-google-y-otros-servicios-para-espiar-trafico-de-sus-objetivos

[6] https://windypundit.com/2013/08/does-the-nsa-have-a-ca/

[7] Jeff Johnson , “Design with the mind in mind”, editorial: Morgan Kaufmann, 2 edición, 07 Feb 2014

[8] http://www.ubicuostudio.com/es/diseno-grafico/ley-fitts-aplicada-diseno-apps/

[9] http://www.ubicuostudio.com/es/diseno-grafico/ley-fitts-iphone-6/

[10] http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (SUS)

[11] http://en.wikipedia.org/wiki/Likert_scale (escala de Likert)

[12] http://www.datagenetics.com/blog/september32012/

La validez de todos los enlaces ha sido comprobada el día 28/08/2015

# Anexo 1. Contenido del CD

## Memoria

Contiene toda la información lo relativo al proyecto y su realización. Estudio, problemas a tratar, propuesta, proceso, solución y conclusiones, así como los fundamentos teóricos usados y el proceso de desarrollo seguido.

## Código

Código desarrollado para el proyecto, incluye a su vez una guía en la que se explica su estructura y organización.

## Ejecutable

Software desarrollado.

## Manual

Manual de uso, contiene información sobre el software desarrollado, información para su correcto despliegue y una explicación de éste, incluye capturas de pantalla para guiar al usuario. Únicamente incluye información para su uso, no explica los fundamentos en los que se basa.

## Adjuntos

“Datos Evaluaciones.xlsx”: Incluye los datos recolectados en cada iteración, así como el estudio de letras candidatas por pulsación.

“Platilla_evaluación_prototipo.docx”: plantilla utilizada para la recolección de información.