Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRAO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍA DE LA INFORMACIÓN

Sistema de autenticación biométrica basado en el análisis del comportamiento mediante interacción por pantalla táctil y sensores de movimiento

Estudiante: Arturo Silvelo Pallín
Dirección: José Carlos Dafonte Vázquez
Daniel Garabato Míguez

A Coruña, septiembre de 2019.

A mi familia

Agradecimientos

A mi familia, por su apoyo incondicional, en los buenos momentos, pero sobre todo en los malos.

A los directores del proyecto, Carlos Dafonte y Daniel Garabato, por su continuo apoyo para sacar esto adelante.

Resumen

Los usuarios de aplicaciones móviles con temática social, financiera o personal quieren que sus datos estén siempre protegidos sin necesidad de realizar procesos o configuraciones complicadas. Por ello, un buen sistema de seguridad es crucial para evitar que los datos sean modificados o robados. Los sistemas de autenticación son una de las principales barreras para evitar este tipo de situaciones.

Con este objetivo, se han implementado a lo largo de los años métodos de autenticación que permitan al sistema verificar si un usuario es quién dice ser y, por lo tanto, permitir al usuario realizar ciertas acciones dentro del sistema. Los métodos de autenticación biométricos son los que más destacan entre los existentes, estos pretenden identificar a un usuario basándose en sus propias características, tanto físicas (huella dactilar, retina…) como de comportamiento (escritura, voz…).

En este proyecto se plantea un estudio para comprobar la viabilidad de un sistema biométrico extrayendo los patrones de comportamiento de un usuario a la ahora de usar un dispositivo móvil, mediante el análisis de la iteración de este individuo con la pantalla táctil y los sensores de movimiento.

Para este propósito, se ha planteado un sistema de autenticación basado en el uso de perfiles personales construidos mediante técnicas de Inteligencia Artificial, a partir de eventos capturados por una aplicación implementada para este propósito.

Abstract

Users of mobile applications with social, financial or personal themes want their data be protected without a complicated process or configuration. Therefore, a good security system is crucial to prevent data from being modified, corrupted or stolen. Authentication systems are one of the main barriers to avoid this situation.

With this aim, authentication methods have been implemented over the years so that they allow the system to know if a user is who claims to be and, therefore, grant access permissions to legitimate users. The biometric authentication methods are the ones that stand out among the existing ones, these are intended to identify a user based on their own characteristics, both physical (fingerprint, retina…) and behavioral (writing, voice…).

This project proposes a study to check the viability of a biometric system by extracting behavioral patterns of the user when using a mobile device by analyzing the iteration of this individual with the touch screen and motion sensors.

For this purpose, we propose an authentication system based on the use of personal profiles constructed using Artificial Intelligence techniques, based on events captured by an application implemented for this purpose.

Palabras clave:

- Inteligencia Artificial
- Seguridad Informática
- Sistemas biométricos
- Sistema Autenticación
- Aplicación Híbrida
- NodeJS
- Red Neuronal Artificial

Keywords:

- Artificial Intelligence
- Computer Security
- Biometric Systems
- Authentication System
- Hybrid Application
- NodeJS
- Artificial Neural Networks

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

En este capítulo se expondrá una visión generalizada del proyecto, abordando la idea original del mismo, cómo surge y qué objetivos se pretenden conseguir con la realización de este.

1.1 Motivación

Las preocupaciones sobre la seguridad informática han estado presentes desde el origen de la informática. Hoy en día el uso de ordenadores y portátiles ha perdido parte de su protagonismo, debido al creciente uso de los teléfonos inteligentes.

Según un estudio de Eurostat[1] [1] en 2018, un 75% de la población de la Unión Europea emplea el *smartphone* para uso personal. Además, un 28% indicó que aceptan los permisos de acceso a datos personales requeridos por las aplicaciones pero, sin embargo, menos de la mitad disponen de una aplicación relacionada con la seguridad.

Por otra parte, la seguridad que ofrecen dichos dispositivos frente a robos o pérdidas es únicamente una pequeña barrera como un patrón o un *PIN*[2]. En la que la mayoría de los casos, por el factor humano, no es una medida de seguridad robusta, ya sea por el uso de patrones simples o por el uso de información personal para la creación de dicho *PIN*.

Por estos motivos, surge la necesidad de implementar sistemas que ofrezcan una mayor seguridad. En los *smartphones* los sensores biométricos como el reconocimiento facial o el de huella dactilar, son algunos de ellos. Estos sistemas de seguridad pueden llegar a ser más seguros que una contraseña, porque la verificación de la identidad se realiza mediante un rasgo o característica propia que no necesita ser recordada.

Todos estos sistemas funcionan correctamente cuando se trata de iniciar una sesión, pero son totalmente vulnerables ante un ataque cuando la sesión está abierta. Esto es debido a que

[1] Oficina Europea de Estadística
[2] Personal Identification Number

los mecanismos mencionados solo autentican al usuario en un momento concreto, normalmente al inicio de la sesión, y mientras esta permanezca abierta no se le solicitará una nueva autenticación.

Debido a estos factores, surge la motivación de plantear un sistema de monitorización continua basado en el comportamiento del usuario frente al dispositivo. De este modo, si el sistema detecta un comportamiento anómalo, podría solicitar al usuario que se verificara, utilizando para ello un nuevo método de autenticación distinto al original (por ejemplo, un código enviado al móvil) o advertir al administrador del sistema para que comprobase el incidente.

El trabajo que se ha realizado en este proyecto no pretende ser un mecanismo de autenticación que sustituya a los ya existentes, sino el de ofrecer un segundo factor de autenticación continuo y transparente para el usuario.

La realización de este proyecto incluye un estudio para comprobar la viabilidad de la autenticación de un conjunto de usuarios mediante su comportamiento frente al uso de una aplicación móvil. Por ello, el proyecto se llevará a cabo en un entorno controlado, empleando una aplicación creada específicamente para el proyecto, con el objetivo de obtener los eventos generados por los usuarios. Aplicando diferentes técnicas de Inteligencia Artificial sobre los eventos capturados, podremos encontrar patrones que permitan autenticar al usuario.

1.2 Objetivos

El proyecto debe cumplir los siguientes objetivos, para garantizar la calidad y funcionalidad del mismo:

1. Implementación de una aplicación móvil que permita la captura de eventos por parte del usuario para su posterior envío.
    - Seleccionar un *framework* de desarrollo orientado a dispositivos móviles que disponga de soporte multiplataforma para múltples sistemas operativos.
    - Analizar y seleccionar los eventos que puedan ser más característicos a la hora de identificar a un usuario.
2. Implementación de un servidor que permita la conexión con la aplicación creada para guardar los eventos generados y su posterior acceso.
    - Seleccionar una herramienta de desarrollo versátil y con un bajo consumo de recursos.
    - Seleccionar una base de datos que permita trabajar con grandes volúmenes de datos cuya estructura pueda cambiar a lo largo del tiempo.

3. Realizar una fase de recogida de la información con los sistemas creados anteriormente, con el fin de obtener un conjunto de datos que contenga una muestra representativa de la población, es decir, personas de diferente edad y género.

4. Analizar y seleccionar técnicas de Inteligencia Artificial/Estadística a los datos recogidos para obtener una serie de características que sean identificativas de los usuarios y entrenar los diferentes algoritmos de aprendizaje con estas.

5. Implementar un servicio de autenticación en línea que permita recibir eventos generados por un usuario y obtenga una predicción de la legitimidad de dicho usuario.

1.3 Organización de la memoria

La memoria se estructura en base a una serie de capítulos:

- **Fundamentos teóricos**: En este apartado se abordan los diferentes términos para comprender mejor el dominio del proyecto.
- **Estudios de alternativas**: En este capítulo se muestran algunas de las alternativas disponibles en el mercado y artículos encontrados relacionados con el tema que se trata en este proyecto.
- **Planificación y evaluación de costes**: En esta parte se exponen las fases en las que se dividió el proyecto, su realización, costes y riesgos.
- **Tecnologías**: En este capítulo se analizan las herramientas utilizadas para el desarrollo de la aplicación.
- **Metodología**: En esta parte se muestra la información relativa al desarrollo del proyecto.
- **Análisis y tratamiento de datos**: En este apartado se muestra el núcleo del proyecto, donde se comentarán los procedimientos utilizados para la búsqueda de un modelo que se adecue a los objetivos del proyecto.
- **Resultados**: En este capítulo se mostrarán los resultados y conclusiones obtenidas a partir del procesamiento de los datos.
- **Trabajo futuro**: En este parte se postularán diversas opciones para continuar con el desarrollo del proyecto.
- **Conclusiones**: En el último apartado se comprobará el grado de cumplimiento de los objetivos y su impacto.

Capítulo 2

Fundamentos teóricos

En este capítulo se presentarán las principales temáticas relacionadas con el domino del proyecto, con el objetivo de comprenderlo mejor.

2.1 Seguridad Informática

La seguridad informática forma parte de un término más genérico como es la seguridad de la información, y tiene como objetivo prevenir y detectar el uso no autorizado de un sistema informático.

2.1.1 Conceptos previos

- **Atacante**: Sujeto o entidad que pone en riesgo un sistema
- **Ataque**: Consiste en cualquier acción hecha por individuos u organizaciones que roban, alteran o destruyen a un blanco específico.
  - **Ataque pasivo**: Son aquellos que buscan en el sistema sin llegar a modificar el mismo.
  - **Ataque activo**: Son aquellos que dañan el objetivo atacado o lo modifican a su favor.
- **Intrusión**: Conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso. Es decir, la intrusión no solo consiste en el acceso no autorizado sino también en la denegación del acceso a otros usuarios o la manipulación de la información.
- **Vulnerabilidad [2]**: Debilidad o fallo en un sistema de información que pone en riesgo la seguridad de la información permitiendo que un atacante pueda comprometer la integridad, disponibilidad o confidencialidad de la misma.

- **Amenaza [2]**:Acción que aprovecha una vulnerabilidad para atentar contra la seguridad de un sistema de información. Es decir, que podría tener un potencial efecto negativo sobre algún elemento de nuestro sistema.
- **Riesgo [2]** : El riesgo es la probabilidad de que se produzca un incidente de seguridad, materializándose una amenaza y causando pérdidas o daños.
- **Política de Seguridad [3]**: Conjunto de requisitos definidos por los responsables de un sistema que indican en términos generales qué está y qué no está permitido en el área de la seguridad durante el uso del sistema.

2.1.2 Objetivos de la seguridad informática

Los sistemas de información guardan, distribuyen o generan información para usuarios, empresas, procesos, aplicaciones... y esta información debe garantizar ciertas características para evitar fraudes. La seguridad informática intenta asegurar estas garantías sobre la información [4]:

- **Confidencialidad**: Es la capacidad de que solo los usuarios autorizados puedan acceder a nuestros recursos, datos e información. Este es uno de los principales problemas a los que se enfrentan las empresas.
- **Integridad**: Es la capacidad de asegurar que los datos sean legítimos, es decir, asegurar que los datos recibidos sean los generados inicialmente y que nada, ni nadie ajeno pueda modificar dichos datos.
- **Disponibilidad**: Esta característica es de las más importantes y asegura que los datos estén accesibles siempre que el usuario, proceso o sistema lo necesite.

2.1.3 Autenticación de usuario

En el ámbito de la seguridad informática es muy importante demostrar que un usuario o una aplicación es realmente quien dicha persona o aplicación asegura ser. Para esta verificación se pueden usar varias técnicas [5]:

1. **Sistemas basados en algo conocido**: Es el modelo de autenticación más básico y consiste en decidir si un usuario dice ser quien es simplemente basándonos en un prueba que a priori solo ese usuario puede saber. Esta aproximación es la más barata y también las más vulnerable a todo tipo de ataques.

   Las entidades que participan en la autenticación acuerdan una clave, que mantendrán en secreto. Cuando una de las partes necesita autenticarse solo tiene que mostrar la clave secreta que han acordado para poder acceder a los recursos.

2. **Sistemas basados en algo poseído**: Este modelo de autenticación es más complejo y se puede complementar con otros sistemas como el anterior de una manera fácil, pero también implica un mayor coste. Por lo general, este modelo se caracteriza por el uso de una tarjeta con un chip integrado el cual incorpora la información necesaria para la autenticación del usuario. Este dispositivo fue patentado por **Roland Moreno** en 1970, y consistía en una tarjeta de plástico con un chip integrado. A día de hoy este tipo de tarjeta está presente en multitud de aplicaciones como tarjeta bancarias, tarjetas de acceso...

3. **Sistemas de autenticación biométrica**: Hoy en día existen otros mecanismos que se basan en las cualidades del usuario. Este tipo de sistemas basados en el reconocimiento de cualidades físicas del usuario, utilizan características únicas del individuo para su identificación. El proceso general para este tipo de sistemas es el siguiente:

    1. Se captura una muestra de los datos del usuario.
    2. Se extraen las características que sean únicas de ese usuario.
    3. Se comparan dichas características con las extraídas en un primer momento.
    4. En base a esa comparación se decide si el usuario es quien dice ser.

    Los sistemas de autenticación biométricos más comunes son:

    - **Verificación de huella dactilar**: La huella de un ser humano es un rasgo de identificación único (salvo alguna excepción), pero los sistemas de autenticación miden ciertas características que se han demostrado únicas en los todos los individuos [6].
    - **Verificación de patrones oculares**: Estos modelos de reconocimiento mediante patrones oculares, por lo general miden rasgos en el iris o en la retina. Han demostrado ser los más fiables con una probabilidad de coincidencia cercana a cero.

    Otros sistemas de autenticación biométricos basados en el comportamiento humano, como el que se trata en este proyecto, son:

    - **Reconocimiento de voz**: El reconocimiento de voz intenta detectar características típicas de la voz del usuario para identificarlo. La técnica aplicada para lograr este fin es parecida a los sistemas de reconocimiento de canciones como *shazam* [7].
    - **Verificación de escritura**: Estos sistemas intentan verificar la firma manuscrita de una persona. En ella se verifican los patrones y características del trazo realizado.

- **Sistemas de reconocimiento de escritura por teclado** [8]: Este tipo de sistemas intenta caracterizar al usuario a partir de ciertos patrones que se producen cuando escribe con un teclado.
- **Sistemas de reconocimiento mediante el uso del ratón** [9]: Intentan caracterizar a los usuarios mediante el estudio de las curvas que se trazan cuando se realizan movimientos con el ratón

2.2 Características del uso de los sensores

Los dispositivos móviles actuales vienen equipados con un conjunto de sensores [10] que permiten conocer la aceleración, posición, luminosidad... del dispositivo. En este proyecto se han utilizado dos de estos sensores para extraer los datos.

2.2.1 Acelerómetro

Este sensor [11] es capaz de medir las aceleraciones en tres ejes espaciales [Figura 2.1]. Es decir, es capaz de medir que el dispositivo se está moviendo y con cuanta intensidad lo está haciendo.

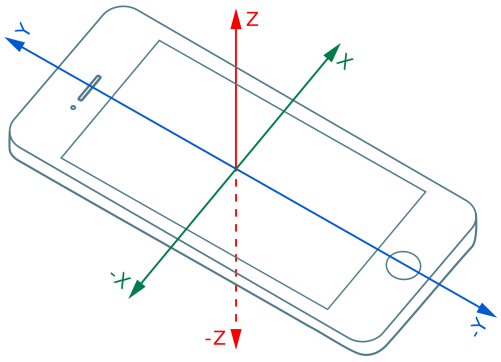

Figura 2.1: Sistema de coordenadas del acelerómetro

2.2.2 Giroscopio

Este sensor [12] es capaz de medir la rotación del dispositivo en los tres ejes espaciales [Figura 2.2]. Es decir, nos ofrece la posición exacta en la que se encuentra el dispositivo.

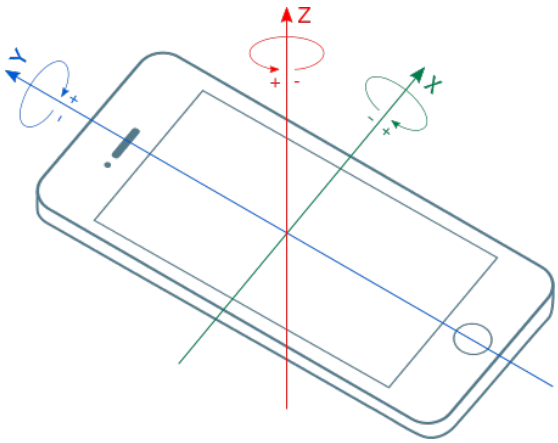

Figura 2.2: Sistema de coordenadas del giroscopio

2.2.3 Extracción de características

Los sensores mencionados anteriormente ofrecen los datos de un espacio tridimensional. Es decir, nos ofrecen tres valores correspondientes a cada uno de los ejes del espacio. Debido a que ambos son dependientes de la posición, se ha decidido calcular una cuarta dimensión que no le afecte dicha posición, este nuevo valor es la magnitud que se calcula de la siguiente manera:

$$\texttt{magnitude} = \sqrt{x^2 + y^2 + z^2} \tag{2.1}$$

Para cada uno de estos cuatro valores obtenidos se han calculado nuevos valores [Tabla 2.1] en ventanas de tiempo de cinco segundos. Estos valores, con ventanas de cinco segundos, han sido utilizados en otros estudios [13] con buenos resultados.

Nombre	Descripción
Media	Es un medida con tendencia al valor central.
Desviación estándar	Mide cuánto se separan los datos con respecto a la media aritmetica.
Varianza	Es el valor de la desviación estándar al cuadrado
Mínimo	Es el valor mínimo del conjunto.
Máximo	Es el valor máximo del conjunto.
Skewness	Es una variable que indica la asimetría de la distribución frente a la media.
Kurtosis	Es una variable que muestra la frecuencia de los datos.

Tabla 2.1: Características de los sensores

2.3 Características del uso de la pantalla táctil

La mayoría de los dispositivos de hoy en día son táctiles y, por lo tanto cuentan con una pantalla que es capaz de generar información sobre la interacción que el usuario realiza. Esta información [14] extraída se calcula en base a la posición y tiempo entre el primer y último contacto con la pantalla.

A partir de la concatenación de varios de estos eventos en el tiempo, se pueden llegar a extraer otros más complejos como arrastrar, rotar... Para este proyecto se han capturado los mostrados en la Tabla 2.2.

Número de puntos	Nombre	Descripción
Un sólo punto	**Swipe**	Es el movimiento de deslizamiento.
	Tap	Tocar la pantalla durante un corto periodo de tiempo
	Press	Tocar la pantalla durante un largo periodo de tiempo.
	Pan	Es el movimiento de arrastrar.
Dos puntos	**Pinch**	Es el movimiento de juntar o separar los dedos para hacer zoom.
	Rotate	Es el movimiento de girar los dedos para hacer rotar un objeto.

Tabla 2.2: Tipos de eventos

Como los eventos *press* son poco frecuentas en cualquier tipo de aplicación se ha decidido descartarlos para el análisis. Los eventos de *pinch* y *rotate* se han unido en un solo evento, puesto que ambos generan las mismas características. Este nuevo evento generado se ha llamado *multitouch*. La Tabla 2.3 muestra las características extraídas para cada tipo de evento, que serán utilizadas para el análisis.

Ev.	Nombre	Descripción
Swipe	**duration**	Duración del evento en milisegundos
	distance	Distancia recorrida del evento en píxeles
	velocity	Velocidad media del evento en píxeles/milisegundos
	angle	Ángulo creado desde el punto inicial al punto final
	width	Ancho del área de pulsación en píxeles
	height	Alto del área de pulsación en píxeles
	frame_index	Posición en la pantalla donde ha ocurrido el evento
	direction	Dirección del evento
Tap	**duration**	Duración del evento en milisegundos

	distance	Distancia recorrida del evento en píxeles
	width	Ancho del área de pulsación en píxeles
	height	Alto del área de pulsación en píxeles
	frame_index	Posición en la pantalla donde ha ocurrido el evento
Pan	**duration**	Duración del evento en milisegundos
	distance	Distancia recorrida del evento en píxeles
	velocity	Velocidad media del evento en píxeles/milisegundos
	angle	Ángulo creado desde el punto inicial al punto final
	width	Ancho del área de pulsación en píxeles
	height	Alto del área de pulsación en píxeles
	angle_total	Suma de los ángulos realizados a lo largo del recorrido.
	velocity_y	Suma total de las velocidades a lo largo del recorrido, sobre el eje Y.
	velocity_x	Suma total de las velocidades a lo largo del recorrido, sobre el eje X.
	avg_velocity_x	Velocidad media en el eje X
	avg_velocity_y	Velocidad media en el eje Y
Multitouch	**area**	Área cubierta del evento, tomando como vértices los puntos mínimos y máximos de los dedos
	abfs	Ángulo entre los dos primeros puntos
	abfe	Ángulo entre los dos últimos puntos
	rbfs	Distancia entre los dos primeros puntos eje y
	rbfe	Distancia entre los dos primeros puntos eje x
	df_up	Distancia entre del dedo superior eje x
	df_dow	Distancia entre del dedo inferior eje x
	avg_width_0	Ancho del área de pulsación en píxeles
	avg_width_1	Ancho del área de pulsación en píxeles
	avg_height_0	Alto del área de pulsación en píxeles
	avg_height_1	Alto del área de pulsación en píxeles
	velocity_dow	Velocidad media del dedo inferior
	velocity_up	Velocidad media del dedo superior
	duration	Duración del evento en milisegundos
	direction	Dirección del evento

Tabla 2.3: Características por tipo de evento

2.4 Evaluación del rendimiento

Para analizar los resultados obtenidos y comprobar la fiabilidad de los algoritmos usados, necesitamos emplear unas métricas que evalúen su rendimiento y permitan comparar diferentes aproximaciones.

Al ser este un problema de clasificación binaria (usuarios legítimos, usuarios no legítimos) se ha utilizado la matriz de confusión [Figura 2.3], la cual categoriza los datos en cuatro grupos:

- **Verdaderos Negativos (TN)**: Es la cantidad de negativos que fueron clasificados correctamente como negativos.
- **Falsos Positivos (FP)**: Es la cantidad de negativos que fueron clasificados incorrectamente como positivos.
- **Falsos Negativos (FN)**: Es la cantidad de negativos que fueron clasificados incorrectamente como negativos.
- **Verdaderos Positivos (TP)**: Es la cantidad de positivos que fueron clasificados correctamente como positivos.

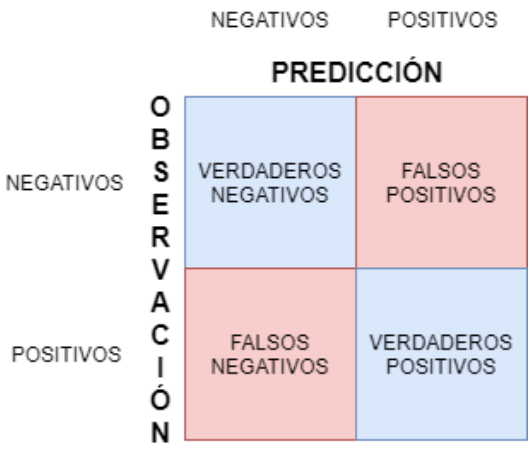

Figura 2.3: Esquema matriz de confusión

A partir de los datos obtenidos de una matriz de confusión, se pueden calcular medidas que permiten obtener una representación analítica de los resultados. Por otra parte, durante el entrenamiento de los algoritmos también se han calculado los tiempos de cómputo.

- **Recall (RC)**: Es la relación entre las predicciones positivas correctas y el total de observaciones positivas.

$$\frac{TP}{TP + FN} \tag{2.2}$$

- **Precision (PS)**: Es la relación entre las predicciones positivas correctas y el total de predicciones positivas.

$$\frac{TP}{TP + FP} \tag{2.3}$$

- **F1**: Es la media armónica de los valores anteriores.

$$\frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{2.4}$$

- **Accuracy**: Es la relación entre las predicciones positivas y el total de casos.

$$\frac{TP}{TP + FP + FN + TN} \tag{2.5}$$

- **Fit Time**: Tiempo de cómputo que ha tardado en entrenar el algoritmo.

- **Score Time**: Tiempo de cómputo que ha tardado en ejecutar la predicción.

Para analizar los resultados, se ha favorecido el uso de la métrica *precision* para tratar de minimizar posibles intrusiones.

2.5 Inteligencia Artificial

En 1956, **Jhon McCarthy** [15], conocido como el padre de la Inteligencia Artificial, acuñó el término Inteligencia Artificial, en adelante *IA*, como la ciencia y la ingeniería de hacer máquinas inteligentes, especialmente programas informáticos inteligentes.

En la *IA* existen dos grupos definidos, fuerte y débil. La *IA* fuerte es aquella que tiene las mismas características que un humano inteligente. La *IA* débil es aquella que muestra inteligencia en un área en concreto pero carece de la misma en otras.

Para conseguir dotar de inteligencia a una máquina se suele utilizar el aprendizaje máquina o *machine learning*. El aprendizaje máquina, en adelante *ML*, es una de las áreas más extendidas de la *IA*. En 1959, **Arthur Samuel** [16] definió *ML* como el campo de estudio que brinda a las computadoras la capacidad de aprender sin estar programado explícitamente.

2.5.1 Tipo de aprendizaje

Dentro del *ML* se pueden distinguir dos tipos de aprendizaje:

- **Aprendizaje supervisado**: Los algoritmos son entrenados en base a un conjunto de datos de los que conocemos su respuesta correcta. De esta manera lo que se intenta es que el algoritmo obtenga la respuesta correcta a partir de las características disponibles.

  - **Problemas de regresión**: Son usados para evaluar las relaciones que existen entre las variables y obtener un valor de esa estimación.
  - **Problemas de clasificación**: Son utilizados para dividir un conjunto de datos de entrada en distintas clases según sus características.

- **Aprendizaje no supervisado**: En este modelo, los datos no contienen una respuesta correcta. Este tipo de aprendizaje intenta buscar ciertos patrones o respuestas, pero no una respuesta concreta.

La finalidad de todos estos tipos de aprendizajes es la de generalizar, es decir, que puedan resolver problemas que no han visto. Cuando entrenamos modelos computacionales con un conjunto de datos de entrada estamos haciendo que el algoritmo sea capaz de generalizar un concepto para que al consultarle por un nuevo conjunto de datos desconocido este sea capaz de comprenderlo y proporcionarnos un resultado fiable [Figura 2.4].

Si nuestros datos de entrenamiento son muy pocos o poco representativos nuestra máquina no será capaz de generalizar el conocimiento y estará incurriendo en *underfitting* [Figura 2.4].

Si sobrentrenamos nuestro modelo lo que ocurrirá es que nuestra máquina sólo se limitará a memorizar los casos particulares que le enseñamos y será incapaz de reconocer nuevos datos de entrada, perdiendo toda capacidad de generalización.

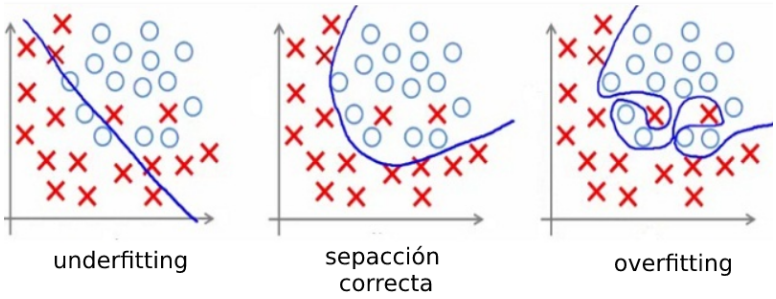

Figura 2.4: Problemas de entrenamiento

2.5.2 Redes Neuronales Artificiales

Las redes neuronales artificiales [17], en adelante *RNA*, son un modelo computacional inspirado en la estructura del sistema nervioso de los seres humanos. La unidad elemental de una *RNA* es la neurona artificial [Figura 2.5] y generalmente están organizadas en capas. Poseen varias entradas y una salida, que se calcula normalmente realizando una suma ponderada de

las entradas con sus pesos [2.6].

$$\sum_{i=1}^{n} w_i x_i + w_0 = \begin{cases} \geq 0 & y = 1 \\ < 0 & y = 0 \end{cases} \tag{2.6}$$

Este resultado es modificado por una función de activación y el valor obtenido se transmite directamente al siguiente elemento. Normalmente para conseguir esta transformación se emplean funciones como la sigmoidal, gaussiana o la tangente hiperbólica:

- Función sigmoidal:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

- Función tangente hiperbólica:

$$f(x) = tanh(x) \tag{2.8}$$

- Función gaussiana:

$$f(x) = e^{\frac{-x^2}{2}} \tag{2.9}$$

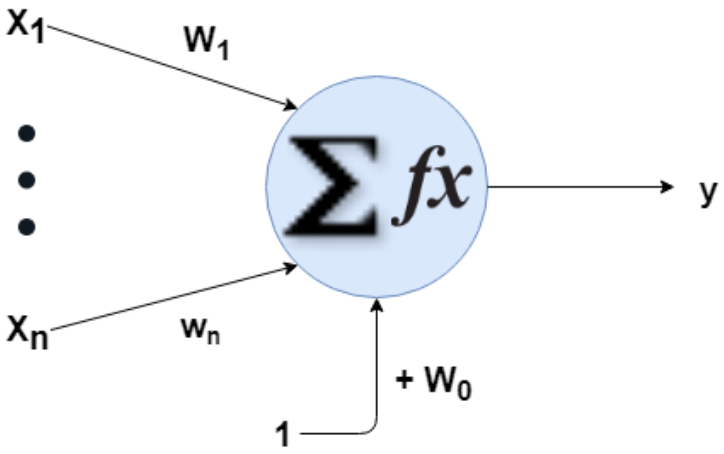

Figura 2.5: Esquema de un neurona artificial

2.5.3 Perceptrón Multicapa

En 1958, **Rosenblatt** [18] diseñó y desarrollo el perceptrón. Este modelo implementa el funcionamiento de una sola neurona que es capaz de resolver problemas lineales. En 1969, **Minsky y Papert** escribieron un libro [19] donde demostraron que un solo perceptrón era incapaz de aprender la función exclusiva (XOR), es decir, problemas cuya resolución no es

lineal. En este mismo libro se expone un nuevo paradigma de *RNA* llamado perceptrón multicapa también conocido como *MLP* por sus siglas en inglés (*Multi-Layer Perceptron*). Este modelo es una combinación de varios perceptrones, que permiten aproximar cualquier problema, aunque no sea lineal. Éste se caracteriza por tener sus neuronas agrupadas en capas de diferentes niveles, por lo general tres [Figura 2.6].

- **Capa de entrada** : Esta capa conecta la red con el exterior, cada neurona se corresponde con cada una de las variables de entrada a la red.
- **Capas ocultas** : Es un conjunto de capas que cuyas entradas son las salidas de la capa anterior y cuya salida pasan a la capa sucesora.
- **Capa de salida** : Conecta las capas ocultas con la salida de la red que proporciona los resultados.

Además, sus conexiones están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa y generalmente todas las neuronas de una capa se encuentran enlazadas con las de la siguiente capa.

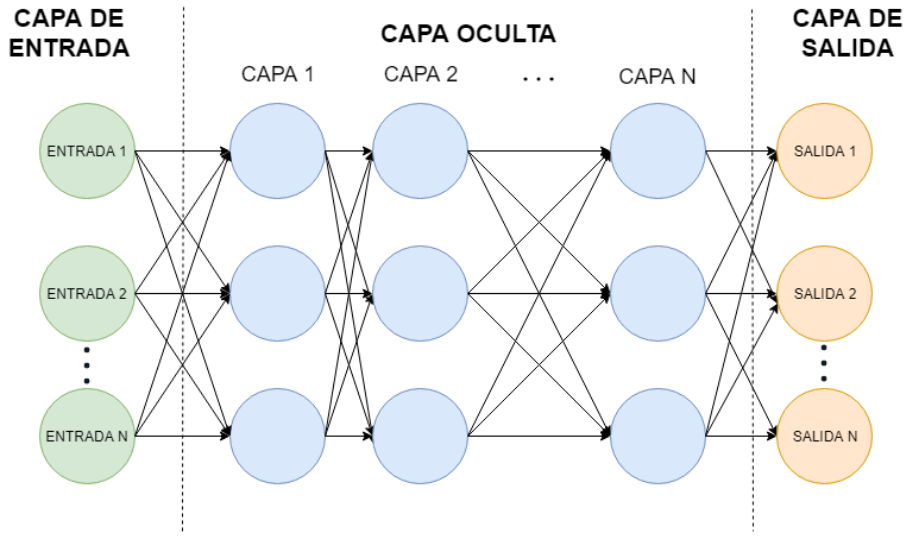

Figura 2.6: Esquema del MLP

Inicialmente, este planteamiento no se pudo materializar porque no existía un mecanismo que ajustase automáticamente los pesos de la capa oculta. En 1986, **Rummelhart, Hinton y Wiliams** desarrollan la *Regla Delta Generalizada* [20] para adaptar los pesos propagando los errores hacia atrás. De esta manera se demuestra que el *MLP* es capaz de resolver problemas lineales y no lineales.

En la Figura 2.7 se puede ver el funcionamiento del *MLP*. En ella se muestra el esquema de un *MLP* que contiene una capa de entrada, dos capas ocultas y la salida. Cada neurona está representada visualmente con una gráfica que muestra la función aplicada a los datos, por ejemplo, en la capa de entrada, las variables de entrada son dos funciones que dividen los datos en vertical (X_1) y horizontal (X_2). También se pueden observar la distintas conexiones que existen entre las neuronas e incluso el peso de las conexiones representadas por un color y grosor diferentes.

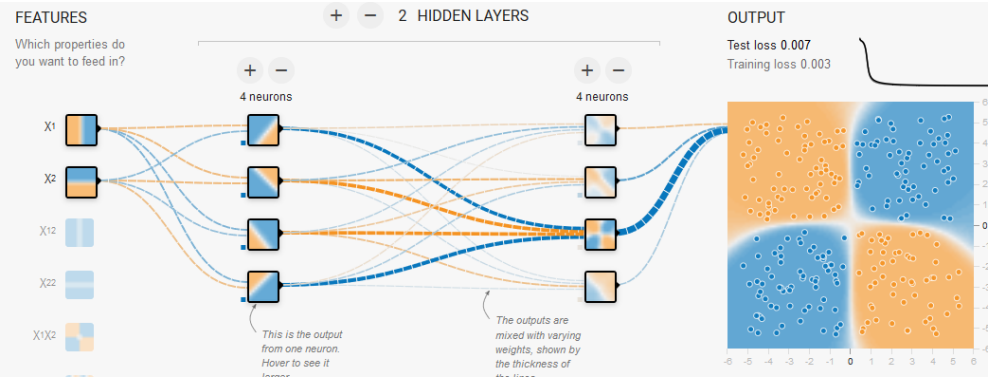

Figura 2.7: Ejemplo de una MLP [21]

Este tipo de arquitectura es muy utilizada debido a su capacidad de aproximación universal. No obstante, requieren un largo proceso de aprendizaje para problemas complejos que requieren de un gran número de variables y una arquitectura compleja.

2.5.4 Maquinas de soporte vectorial

Las máquinas de soporte vectorial [22] también conocidas como *SVMs* de sus siglas en inglés (*Support Vector Machines*), es un conjunto de algoritmos desarrollados por **Vladimir Vapnik**, capaces de realizar clasificaciones, regresiones e incluso detectar valores atípicos.

Las *SVMs* generan un hiperplano[1] para intentar clasificar los datos [Figura 2.8]. En dicho hiperplano se forma una *calle* para separar los datos, donde la línea continua separa el conjunto y las líneas discontinuas indican el margen de error. Las *SVMs* intentan maximizar el margen de error, lo que ocasiona una *calle* más grande y por lo tanto generalizará mejor el problema.

Tipos de problemas

El conjunto de datos utilizados por estos algoritmos suelen ser multidimensionales, por lo tanto, en ciertas ocasiones puede ser complicado separar los datos. En base a esto podemos

[1]Es un plano de una dimensión inferior al origen, que divide el espacio en dos mitades

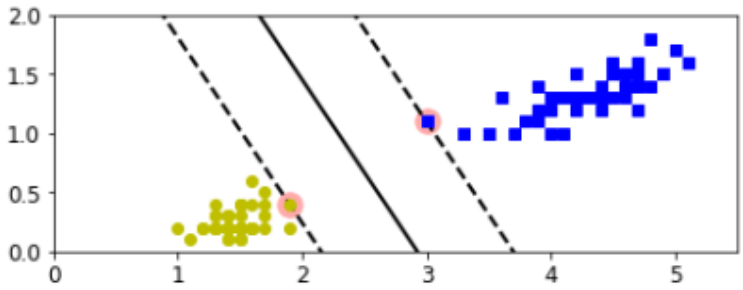

Figura 2.8: Ejemplo de Clasificación SVM

encontrarnos con dos tipos de escenarios:

- **Lineales [Figura 2.8]** : La solución de este tipo de problemas genera un hiperplano que es capaz de separar el conjunto de datos perfectamente.
- **No lineales**: Son aquellos en los que el conjunto de entrada no es posible separarlos con un hiperplano, pero el hecho de que no sean separables en el espacio original, no significa que no lo sean en un espacio de dimensiones distinto. Para transformar el espacio de entradas a una dimensión diferente se emplean las funciones kernel:
  - **Lineal** : Kernel utilizado cuando pretendemos aproximar nuestra función con una función lineal.
    $$K(x, y) = xy + c \tag{2.10}$$
    donde c es una constante.
  - **Polinómica**: Representa la similitud de los vectores en un espacio polinómico distinto al original.
    $$K(x, y) = (a + xy)^d \tag{2.11}$$
    donde d es el orden del polinomio y a es una constante.
  - **Función de base radial Gaussiana (RBF) [Figura 2.9]**: Genera un nuevo espacio calculando las distancias entre un puntos concreto y el resto de puntos.
    $$K(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2}) \tag{2.12}$$
    donde σ es la anchura del kernel y $||x - y||$ es la distancia euclídea[2] entre x e y
  - **Función sigmoide**: Aplica la función sigmoide para generar un nuevo espacio de valores.
    $$K(x, y) = tanh(\alpha xy + c) \tag{2.13}$$

[2]es la distancia entre dos puntos de un espacio euclídeo, la cual se deduce a partir del teorema de Pitágoras.

donde α es la pendiente y c es una constante.

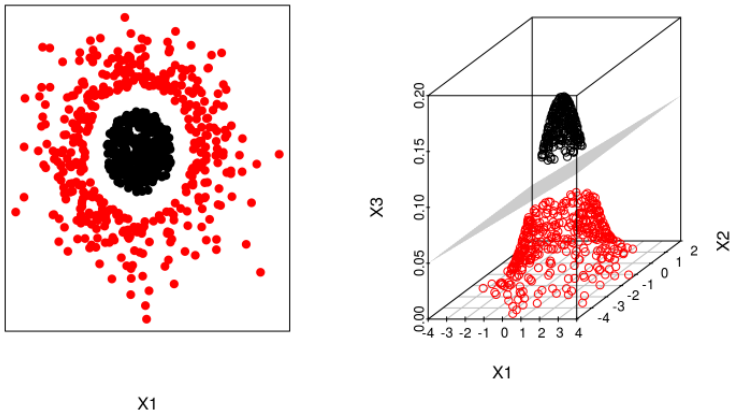

Figura 2.9: Ejemplo de Clasificación SVM no lineal, usando función gaussiana [23]

2.5.5 Árboles de Decisión

Generan modelos de clasificación o regresión usando árboles como estructuras internas [Figura 2.10]. En dicha estructura cada nodo representa una característica del problema, cada rama representa una decisión de esa característica y los nodos hoja contienen el valor de la predicción o clase.

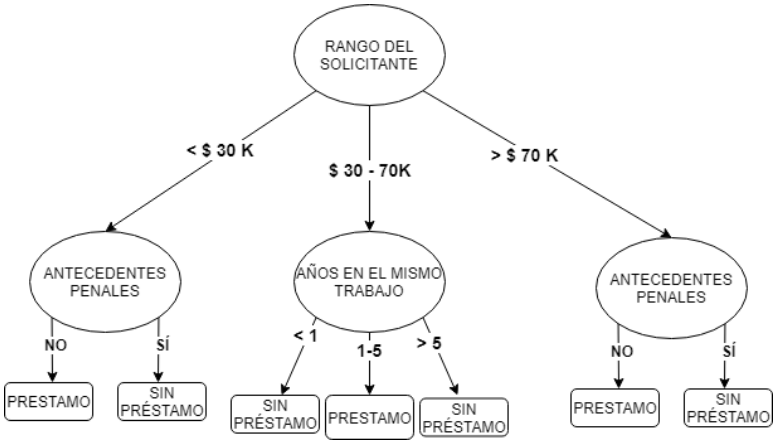

Figura 2.10: Árbol de decisión para conceder un préstamo

2.5.6 Regresión Logística

Es un algoritmo de clasificación usado para asignar un conjunto de valores a dos tipos de clases. Para obtener el valor de la predicción utiliza la función sigmoide.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

2.5.7 *k-Nearest Neighbours*

Es un método que busca en las observaciones más cercanas a la que se está tratando de predecir y clasifica en base a la mayoría de datos que lo rodean [Figura 2.11]. Las técnicas utilizadas para calcular las distancias suelen ser distancia euclidiana, distancia Manhattan...

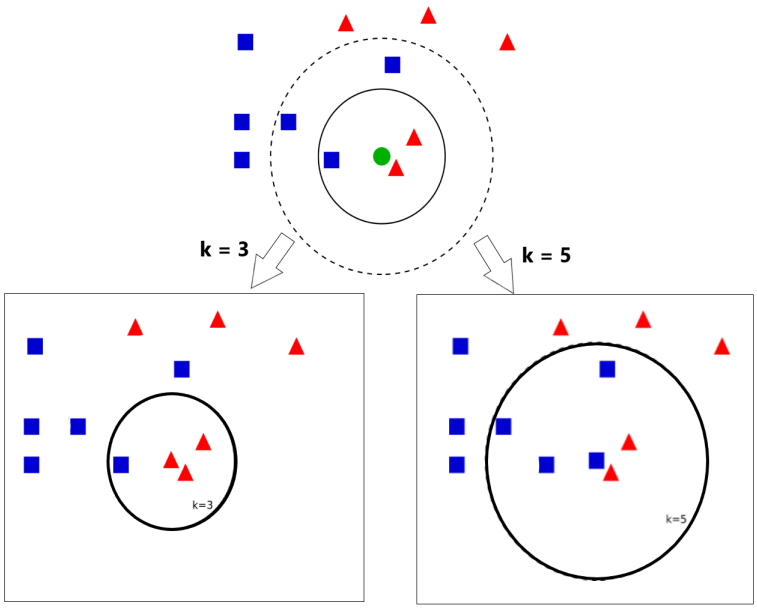

Figura 2.11: Ejemplo de k-Nearest Neighbours

2.5.8 Métodos Ensambladores

Los métodos de tipo ensamblador están formados por un grupo de modelos predictivos que permiten alcanzar una mejor precisión y estabilidad del modelo. Estos utilizan diferentes técnicas para mejorar los resultados de un algoritmo, ya sea combinándolo con otros o utilizando varias instancias del mismo.

Algunas de estas técnicas son *Stacking*, *Bagging* y *Boosting*. Estas dos últimas utilizan *Bootstrapping* como método de muestreo de datos.

Bootstrapping

Es un técnica de muestreo. De las n muestras disponibles, se escogen k con reemplazo. Luego ejecutamos nuestros algoritmos utilizando esas muestras. Se utiliza el remplazo para asegurar que las muestras sean aleatorias. Si se realizase sin remplazo, las muestras extraídas dependerán de las anteriores.

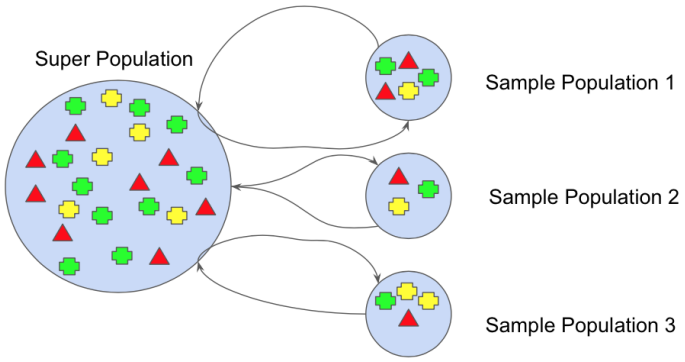

Figura 2.12: Ejemplo de *Bootstrapping* [24]

Bagging [25]

Este método genera múltiples instancias de un mismo modelo predictivo para conseguir una mejora en la precisión de la predicción. Generalmente esta técnica puede ser usada para reducir en algoritmos que tienen una alta varianza para reducirla.

Boosting

Esta técnica emplea un conjunto de algoritmos que utilizan promedios ponderados para convertir aprendizajes débiles en fuertes. Cada modelo ejecutado, dicta en que características se centrará el siguiente modelo.

Stacking

Esta técnica combina múltiples modelos de clasificación o regresión. Los modelos son entrenados individualmente utilizando un conjunto de entrenamiento y sus resultados son combinados para obtener una predicción final.

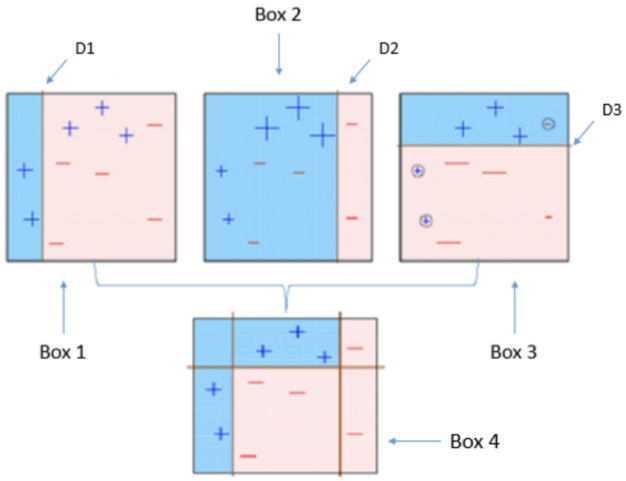

Figura 2.13: Ejemplo de *Boosting* [26]

2.5.9 Bosques Aleatorios

Los bosques aleatorios [27] también conocidos como *Random Forest*, fueron desarrollados por **Leo Breiman** y **Adele Cutler**. Este algoritmo utiliza la técnica de *Bagging* para realizar las predicciones.

Son un conjunto de árboles de decisión en el que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de los árboles del bosque [Figura 2.14].

Ventajas

Las ventajas de los *Random Forest* son:

- Es uno de los algoritmos de aprendizaje más fiables.
- Funciona bien con conjunto de datos muy grandes.
- Puede manejar cientos de variables de entrada.
- Guarda la información sobre las variables más importantes.

Desventajas

- No funcionan bien cuando hay variables categóricas.
- Puede sobre ajustar en ciertos grupos de datos con mucho ruido.

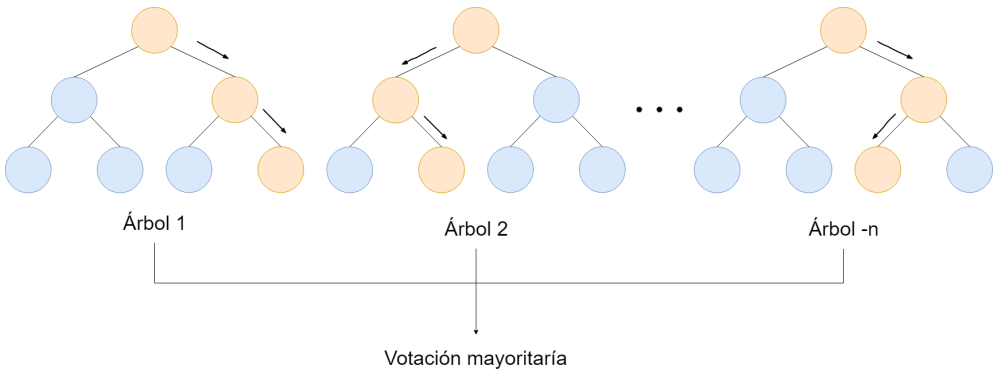

Figura 2.14: Esquema *Random Forest*

2.5.10 Clasificador por Votación

La clasificación por votación [28] es un meta-clasificador, es decir, no implementa un algoritmo de clasificación sino que evalúa las predicciones de otros algoritmos para obtener una nueva [Figura 2.15]. Este algoritmo se basa en la técnica de *Stacking* para obtener las predicciones.

Tipos de votación

- **Votación Dura/Mayoritaria** : Es un caso de selección por mayoría simple. La predicción se hace en base al mayor número de votos por parte de los algoritmos utilizados.
- **Votación Blanda** : Esta técnica calcula el mejor resultado obteniendo la media de las probabilidades calculadas por los algoritmos individualmente.

Ventajas

- Por norma general suelen proporcionar mejores resultados, si se rigen por ciertas condiciones, como que los clasificadores sean totalmente independientes [29].

Desventajas

- No todos los algoritmos son válidos para esta técnica, especialmente cuando se usa el método de votación blanda, ya que no todos los algoritmos estiman las probabilidades de las salidas.

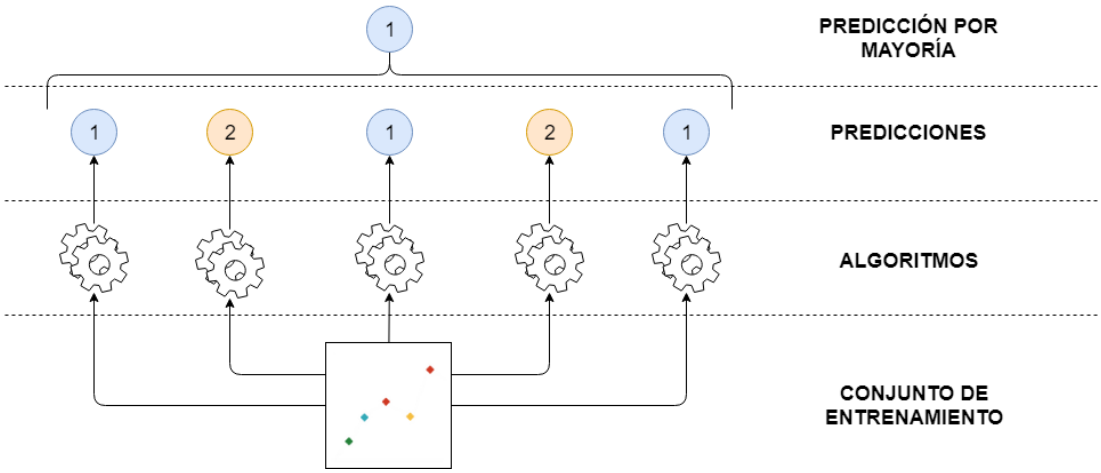

Figura 2.15: Esquema clasificación por votación

2.6 Estudio de alternativas

Debido al interés de este tipo de sistemas de autenticación, se pueden encontrar múltiples estudios relacionados. Existen un número importante de estudios/proyectos sobre la autenticación basada en el uso de un dispositivo móvil y los sensores que este contiene. Pero la mayoría de estos estudios/proyecto no consiguen obtener un producto final [30] [31] [32] [33] [34].

Las alternativas que tienen o muestran un producto final usable se mencionan en las siguientes secciones del capitulo.

2.6.1 SealSign

En el *Mobile World Congress* del año 2015 se dio a conocer *SealSign* [35] por **Eleven Paths**, una aplicación que permite la firma manuscrita en los dispositivos móviles, mediante el análisis de parámetros biométricos para validar al usuario. Estos parámetros, según los desarrolladores, son la velocidad y la presión en los ejes X e Y, de los cuales obtienen la huella biométrica de esa firma. Este sistema cuenta con la posibilidad de realizar firmas masivas de documentos y, además, permite una integración con cualquier otro lenguaje de manera sencilla.

2.6.2 Biosig-id

Biosig-id [36] es una herramienta de seguridad que no solo valida la contraseña del usuario, sino que además analiza el comportamiento de escritura tanto con ratón como por pantalla táctil. Esta herramienta cuenta con aplicación de demostración [3] en la que retan al usuario a falsificar una firma, dando una recompensa al que consiga entrar.

2.6.3 Touch me once and I know it's you

En el año 2012 un grupo de la Universidad de Múnich, desarrolló una aplicación [37] que capturaba los movimientos del usuario cuando estos realizaban el patrón para desbloquear el móvil. La aplicación consistía en repetir ochenta veces seguidas el patrón de desbloqueo, para evitar la monotonía del proceso, se realizaban pausas cada veinte patrones, en las cuales el usuario debía de realizar otro tipo de acciones durante un periodo largo de tiempo.

[3] https://biosig-id.com/go-verify-yourself/

Capítulo 3

Planificación y evaluación de costes

Todo proyecto debería de disponer de una planificación inicial para establecer un guión a seguir, asegurando de esta manera el cumplimiento de los plazos, costes y calidad del producto final. A lo largo de este capítulo se detallará la planificación realizada para llevar a cabo el proyecto.

En un primer punto se mostrarán los recursos que han participado en el proyecto y las tareas en las que se ha divido. Con estos datos definidos podemos establecer la planificación determinando de esta manera su línea base[1].

Al finalizar el proyecto se compararán los resultados estimados con los reales, con la finalidad de ver las desviaciones obtenidas.

En la última sección de este capítulo se realizará un análisis de riesgos, para intentar controlar algunos puntos que creemos que son críticos.

3.1 Recursos y actividades

Para la elaboración de cualquier proyecto primero deberemos conocer los recursos que tenemos y su disponibilidad, de la misma manera deberemos dividir nuestro proyecto en tareas más pequeñas con el fin de poder gestionarlas mejor.

Para este proyecto se contaron con los recursos que se mostrarán en la Tabla 3.1 y las siguientes tareas[2]:

1. **Iteración 0: Búsqueda de información sobre el dominio**

    - **Recopilación de información (80 horas)**: Búsqueda de información relacionada con el dominio de autenticación usando dispositivos móviles y técnicas de *IA*.

[1] Es una foto fija de la planificación a efectos de comparación.
[2] Las tareas con horas son condicionados en esfuerzo y en días son condicionadas en duración.

- **Elaborar los requisitos (5 días)** : Se decidió elaborar una lista de requisitos que debería cumplir la aplicación para su correcto funcionamiento.
- **Peer Review Requisitos (2 días)** : La lista de requisitos elaborada se ha sometido a la revisión por parte de los directores.
- **Elaborar arquitectura - diseño de alto nivel (40 horas)** : Se describen los componentes principales del sistema y el modo en que interactúan entre sí. Elección de tecnologías, modelo arquitectónico junto con sus componentes.
- **Detalle de componentes (80 horas)**: Último paso en la descomposición del diseño, en el que se llega a las unidades de programación (las clases de implementación) detallando los componentes identificados en el diseño de alto nivel. Diagrama final de clases, diagramas de secuencia, etc.

2. **Iteración 1: Sistema de captura de información**

   - **Elaborar pruebas (40 horas)**: Se elaboraran las pruebas para validar el correcto funcionamiento de la aplicación.
   - **Implementación (240 horas)**: Se realizará el código de la aplicación siguiendo los parámetros establecidos.
   - **Ejecutar pruebas (2 días)**: Se ejecutarán las pruebas preparadas en la tarea anterior.

3. **Iteración 2: Sistema para el almacenamiento de la información**

   - **Elaborar pruebas (40 horas)**: Se elaboraran las pruebas para validar el correcto funcionamiento del servidor.
   - **Implementación (120 horas)**: Se realizará el código del servidor siguiendo los parámetros establecidos.
   - **Ejecutar pruebas (2 días)**: Se ejecutarán las pruebas preparadas en la tarea anterior.

4. **Iteración 3: Comunicar la aplicación con servidor**

   - **Desplegar Servidor (1 día)**: Es esta tarea se usará un servidor para que nuestro sistema sea accesible en cualquier momento.
   - **Comunicar servidor y aplicación (5 días)**: Se establecerán las conexiones y configuraciones necesarias, para que ambos sistemas se comuniquen.
   - **Securizar conexiones (3 días)** : Se creará un certificado digital y se configurará el servidor y la aplicación para que cifren las conexiones.

5. **Iteración 4: Implementación de la Inteligencia Artificial**

   - **Obtener datos (30 días)**: Se buscarán voluntarias para probar el sistema creado y conseguir los eventos necesarios para su posterior análisis.
   - **Analizar características (240 horas)** : Analizar los datos recogidos y extraer sus características para el entrenamiento.
   - **Implementar técnicas de Inteligencia Artificial (112 horas)**: Estudiar los distintos algoritmos para implementar la inteligencia artificial y programarla.

6. **Iteración 5: Implementación online de los algoritmos**

   - **Implementación online (32 horas)**: Crear un servidor que habilite el acceso al algoritmo implementado y responda con la predicción.

7. **Documentación del proyecto**: Elaborar la memoria del proyecto y otra documentación sobre instalación y configuración del proyecto. Esta tarea de documentación no tiene una duración establecida, ya que se desarrollará a lo largo de todo el proyecto.

Recurso	Coste [38]
Programador	10 €/ hora
Analista	12 €/ hora
Diseñador	10 €/ hora
Director	17 €/ hora
Servidor	0.013 €/hora
Servidores para entrenamiento IA	0.70 €/hora
Samsung Tab S4	600 €
Ordenador	650 €

Tabla 3.1: Recursos del proyecto [3]

3.2 Planificación Inicial

La planificación inicial estima como deberá de ejecutarse todo el proyecto, las relaciones entre las tareas y los recursos asignados a estas. En la figura 3.1 se pueden ver las tareas planificadas con el coste, tiempo y dependencias de nuestro proyecto y la figura 3.2 muestra el *Diagrama de Gantt*[4].

[3]Los roles de programador, analista y diseñador han sido realizados por la misma persona.
[4]Mapa donde se muestran las tareas relacionadas a lo largo del tiempo

Id	Modo de tarea	Nombre de tarea	Duración	Trabajo	Comienzo	Fin	Predecesoras	Nombres de los recursos
1		Proyecto iniciado	0 días?		0 horas lun 01/10/18	lun 01/10/18		
2		**Documentación del proyecto**	**232,63 días?**	**1.534,9 horas**	**lun 01/10/18**	**mié 21/08/19**		**Programador[8%];Analista[2%]**
3		Hito: Inicio Iteración 0	1 día?	0 horas	lun 01/10/18	lun 01/10/18	1	
4		**Iteración 0**	**34,19 días?**	**252 horas**	**mar 02/10/18**	**lun 19/11/18**		
5		Recopilación de información	11,11 días?	80 horas	mar 02/10/18	mié 17/10/18	3	Programador[90%]
6		Elaborar los requisitos	5 días	36 horas	mié 17/10/18	mié 24/10/18	5	Analista[90%]
7		Peer Review de los requisitos	2 días	16 horas	mié 24/10/18	vie 26/10/18	6	Director 1[50%];Director 2[50%]
8		Elaborar arquitectura (Diseño de alto nivel)	5,02 días?	40 horas	vie 26/10/18	vie 02/11/18	7	Programador[70%];Director 1[10%];Director 2[10%];Analista[20%]
9		Detalle de componentes (Diseño de bajo nivel)	11,05 días?	80 horas	vie 02/11/18	lun 19/11/18	8	Programador[60%];Director 1[10%];Director 2[10%];Diseñador[30%]
10		Hito: Fin Iteración 0	1 día?	0 horas	lun 19/11/18	mar 20/11/18	5;6;7;8;9	
11		Review Iteración 0	2 días	19,2 horas	mar 20/11/18	jue 22/11/18	10	Analista[60%];Director 1[30%];Director 2[30%]
12		Hito: Inicio Iteración 1	1 día?	0 horas	jue 22/11/18	vie 23/11/18	11	
13		**Iteración 1**	**59,27 días?**	**294,4 horas**	**vie 23/11/18**	**jue 14/03/19**		
14		Elaborar pruebas (Aplicación)	20 días?	40 horas	mar 15/01/19	mar 12/02/19	15CC+25%;16FF	Programador[25%]
15		Implementar Aplicación	47,27 días?	240 horas	vie 23/11/18	mar 12/02/19	12	Programador[55%];Diseñador[10%]
16		Hito: Fin Implementación	0 días	0 horas	mar 12/02/19	mar 12/02/19	15	
17		Ejecutar pruebas (Aplicación)	2 días	14,4 horas	mar 12/02/19	jue 14/02/19	15	Programador[90%]
18		Hito: Fin Iteración 1	1 día?	0 horas	jue 14/02/19	vie 15/02/19	14;15;16;17	
19		Review Iteración 1	2 días	19,2 horas	vie 15/02/19	mar 19/02/19	18	Analista[60%];Director 1[30%];Director 2[30%]
20		Hito: Inicio Iteración 2	1 día?	0 horas	mar 19/02/19	mié 20/02/19	19	
21		**Iteración 2**	**27 días?**	**174,4 horas**	**mié 20/02/19**	**vie 29/03/19**		
22		Elaborar pruebas (Servidor)	16,67 días?	40 horas	jue 28/02/19	lun 25/03/19	23CC+25%	Programador[30%]
23		Implementar Servidor	25 días?	120 horas	mié 20/02/19	mié 27/03/19	20	Programador[60%]
24		Ejecutar pruebas (Servidor	2 días	14,4 horas	mié 27/03/19	vie 29/03/19	23	Programador[90%]
25		Hito: Fin Iteración 2	1 día?	0 horas	vie 29/03/19	lun 01/04/19	22;23;24	
26		Review Iteración 2	2 días	19,2 horas	lun 01/04/19	mié 03/04/19	25	Analista[60%];Director 1[30%];Director 2[30%]
27		Hito: Inicio Iteración 3	1 día?	0 horas	mié 03/04/19	jue 04/04/19	26	
28		**Iteración 3**	**9 días**	**64,8 horas**	**jue 04/04/19**	**mié 17/04/19**		**Servidor[64,8 h.]**
29		Desplegar servidor	1 día	7,2 horas	jue 04/04/19	vie 05/04/19	27	Programador[90%]
30		Comunicar servidor y aplicación	5 días	36 horas	vie 05/04/19	vie 12/04/19	29	Programador[90%]
31		Segurizar conexiones	3 días	21,6 horas	vie 12/04/19	mié 17/04/19	30	Programador[90%]
32		Hito: Fin Iteración 3	1 día?	0 horas	mié 17/04/19	jue 18/04/19	29;30;31	
33		Review Iteración 3	2 días	19,2 horas	jue 18/04/19	lun 22/04/19	32	Analista[60%];Director 1[30%];Director 2[30%]
34		Hito: Inicio Iteración 4	1 día?	0 horas	lun 22/04/19	mar 23/04/19	33	
35		**Iteración 4**	**74,72 días?**	**424 horas**	**mar 23/04/19**	**mar 06/08/19**		**Servidor[424 h.]**
36		Obtener datos	30 días	24 horas	mar 23/04/19	mar 04/06/19	34	Programador[10%]
37		Analizar características	37,5 días?	240 horas	mar 14/05/19	jue 04/07/19	36CC+50%	Programador[80%];Servidores Alto Rendimiento[240 h.]
38		Implementar Inteligencia Artificial	22,22 días?	160 horas	jue 04/07/19	mar 06/08/19	37	Programador[90%];Servidores Alto Rendimiento[160 h.]
39		Hito: Fin de iteración 4	1 día?	0 horas	mar 06/08/19	mié 07/08/19	36;37;38	
40		Review Iteración 4	2 días	19,2 horas	mié 07/08/19	vie 09/08/19	39	Analista[60%];Director 1[30%];Director 2[30%]
41		Hito: Inicio Iteración 5	1 día?	0 horas	vie 09/08/19	lun 12/08/19	40	
42		**Iteración 5**	**4,44 días?**	**32 horas**	**lun 12/08/19**	**vie 16/08/19**		
43		Implementación online	4,44 días?	32 horas	lun 12/08/19	vie 16/08/19	41	Programador[90%]
44		Hito: Fin Iteración 5	1 día?	0 horas	vie 16/08/19	lun 19/08/19	43	
45		Review Iteración 5	2 días	19,2 horas	lun 19/08/19	mié 21/08/19	44	Analista[60%];Director 1[30%];Director 2[30%]
46		Finalización de documentación	7,14 días	68 horas	lun 19/08/19	mié 28/08/19	44	Programador[70%];Analista[30%];Director 1[20%];Director 2[20%]
47		Poyecto Finalizado	0 días	0 horas	mié 28/08/19	mié 28/08/19	46	

Figura 3.1: Lista de tareas planificadas

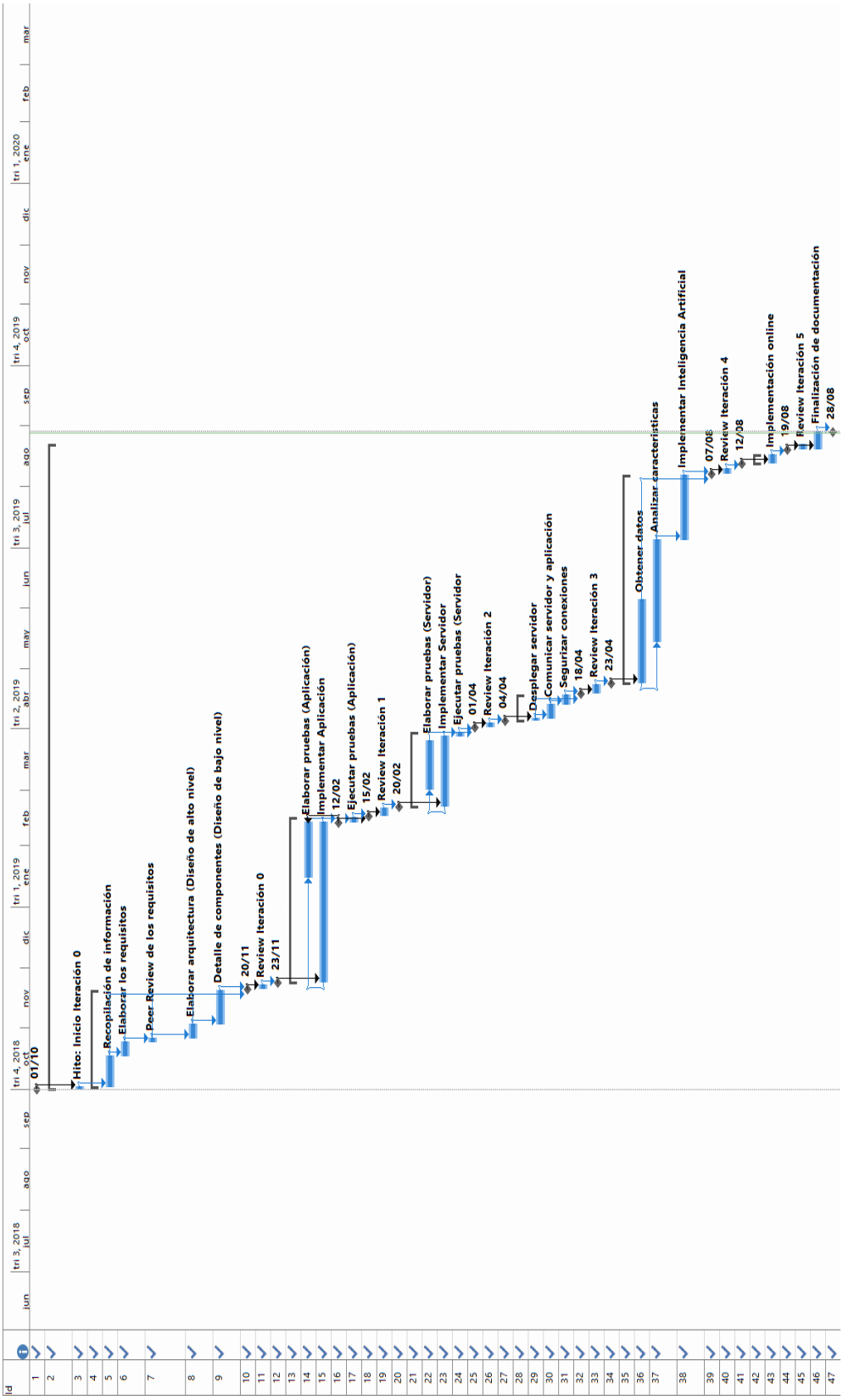

Figura 3.2: Diagrama de Gantt

3.3 Seguimiento del proyecto

En esta sección se detalla el seguimiento realizado sobre el proyecto a su finalización, ya que, durante el transcurso del proyecto se produjeron algunos retrasos.

Durante las fechas comprendidas entre el 23/12/2018 y el 6/1/2019, el *Programador* se ausentó por motivos personales, y el proyecto tuvo que paralizarse durante esos días. Una vez finalizado el proyecto se analizaron las desviaciones con respecto a la planificación estimada [Tabla 3.2]. La comparación de los datos muestran que debido a la ausencia *Programador* durante esos días el proyecto sufriera ese mismo retraso en días.

	Estimación	Real
Fecha Inicio	1/10/2018	1/10/2019
Fecha Fin	14/08/2019	28/8/2019
Trabajo	1.378,17 horas	1378.17 horas
Duración	227,77 días	237,77 días
Costo	18.607,93 €	18.607,93 €
Variación		10 días

Tabla 3.2: Costes y tiempos del proyecto [5]

3.4 Análisis de riesgos

A la hora de planificar un proyecto hay que tener en cuenta los riesgos de este. Por muy bien planificado que se encuentre sino se tienen en cuenta los riesgos, la planificación puede verse totalmente desajustada.

A la hora de preparar un plan de riesgos se siguen ciertos pasos ya establecidos:

- **Identificación**: Elaborar un lista de posibles riesgos.
- **Valoración**: Cuantificar los riesgos para conocer el impacto que tendrían.
- **Análisis**: Estudiar alternativas y crear planes de prevención y contención.

Para este proyecto se han identificado algunos riesgos [Tabla 3.3] que en caso de suceder podrían retrasar el proyecto. Para algunos de ellos se ha elaborado un plan de contingencia para para minimizar su impacto en caso de que ocurran [Tabla 3.4].

[5] Mirando la fila de fecha de fin se observa que hay 14 días de diferencia, pero estos son naturales y no laborables

Nombre	Descripción	Prob.	Impacto
Conocimiento del dominio	El dominio del proyecto es prácticamente desconocido para el autor.	Alto	Medio
Análisis de los datos	Buscar características que sean identificativas del usuarios.	Alto	Alto
Técnicas de IA	Buscar técnicas o métodos que obtengan resultados concluyentes.	Alto	Alto
Configuración del sistema	La configuración de cada una de las partes puede llegar a ser bastante complicada en ciertos puntos y generar conflictos.	Media	Bajo
Caída de servidores	Los servidores pueden sufrir caídas y dejar sin servicio.	Media	Bajo

Tabla 3.3: Tabla de riesgos

Riesgos	Plan de Gestión
Configuración del sistema	Tener una copia de seguridad actualizada con toda la configuración y servicios necesarios, para poder volver a una configuración estable
Caída de servidores	Se ha decidido utilizar un servicio en la nube con alta disponibilidad para evitar que el riesgo ocurra.

Tabla 3.4: Plan de gestión de riesgos

Capítulo 4
Tecnologías

A lo largo del capítulo se explicarán las tecnologías utilizadas para la realización del proyecto, así como otros aspectos relacionados con su desarrollo.

4.1 Arquitectura

Para comenzar a desarrollar un proyecto software hay que tener en cuenta qué herramientas vamos a usar y qué requisitos debemos cumplir. Por ello se necesita conocer cómo va a ser la estructura de alto nivel de nuestro proyecto para desarrollarla y de esta manera conseguir un sistema robusto y escalable.

En este proyecto se ha utilizado una arquitectura de tres capas [Figura 4.1] que se distribuye de la siguiente manera:

- **Capa de Presentación:** Presenta el sistema al usuario, le comunica la información y captura la información. Esta capa se comunica únicamente con la capa de negocio.
- **Capa de Negocio:** Se reciben las peticiones del usuario y se envían las respuestas tras la ejecución. Se denomina capa de negocio porque es donde se establecen las reglas de procesado de los datos. Esta capa se comunica con la capa de presentación, para recibir solicitudes y enviar los resultados, y con la capa de datos, para solicitar el almacenamiento y recuperación de los datos.
- **Capa de Datos:** Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o varios gestores de bases de datos que se encargan de procesar las solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Cada una de estas capas suele distribuirse en distintos servidores, aunque podría darse que todas las capas se encontrasen en uno solo. Este tipo de arquitectura permite replicar la capa que más lo necesite. Lo que otorga la capacidad de escalar la aplicación sobre las partes

que se consideren más críticas. Es decir, si tenemos un sistema que necesita procesar muchas operaciones de guardado o lectura de datos, podríamos replicar la capa de datos en otros ordenadores balanceando las peticiones sobre los nuevos servicios creados.

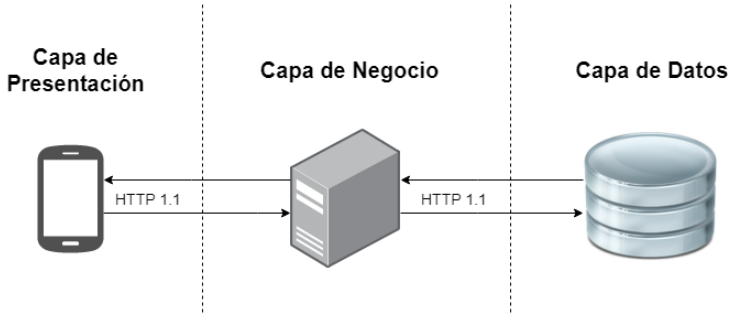

Figura 4.1: Arquitectura de tres capas

4.1.1 Capa de Presentación

Para esta parte de la arquitectura se ha utilizado *Angular* como *framework* de desarrollo principal e *Ionic* y *Capacitor* como para la interfaz visual y creación de una aplicación *Android* respectivamente.

Al desarrollar una aplicación en *Angular* nos vemos forzados a usar el patrón *MVVM* (Modelo-Vista-Vista-Modelo) [Figura 4.2], ya que es el utilizado por este *framework*. Este patrón consiste en la modificación del modelo por parte de la vista y la modificación de la vista por parte del modelo, utilizando la técnica del doble enlace (*two way binding*) implementada por este *framework*.

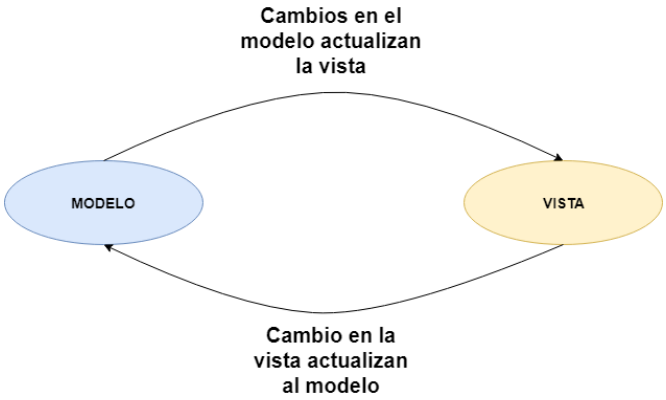

Figura 4.2: Patrón Modelo-Vista-Vista-Modelo

Angular [39]

Es un *framework* de desarrollo *Javascript* , creado por Google. Dicho *framework* nos ayuda al desarrollo de aplicaciones web SPA (*Single Page Aplication*).

Se ha decidido utilizar *Angular* por las siguientes razones:

- Permite crear aplicaciones modulares y con componentes que pueden ser reutilizables a lo largo de la aplicación, lo que genera un código más limpio y fácil de mantener.
- Se integra con bien con *Ionic* para crear aplicaciones híbridas.
- Utiliza inyección de dependencias, un patrón que permite instanciar los servicios directamente sin necesidad de crearlos localmente.
- Proporciona un programa de línea de comandos que permite generar plantillas para ciertos componentes del código.
- Proporciona un conjunto de librerías que facilitan aspectos como la realización de peticiones y envío de datos a través de *HTTP* o navegación entre las páginas de la aplicación.

Ionic [40]

Es una librería *Javscript* que nos permite construir de manera sencilla y rápida la interfaz de nuestra aplicación.

Se ha decidido usar *Ionic* por las siguientes razones:

- Abstrae al diseñador de dibujar los componentes, centrándose en cómo se verán la pantalla.
- Permite utilizar la misma interfaz en distintas plataformas, adaptándose automáticamente a estas plataformas (Android, escritorio, web e IOS)
- Se integra perfectamente con los *frameworks* más usados del mercado como *Angular*.
- Está implementado en su totalidad con componentes web[1], lo que proporciona un rendimiento y compatibilidad mayores.

Capacitor [41]

Es una librería *Javascript*, creada por *Ionic*, que contiene una conjunto de herramientas, las cuales nos permiten exportar una aplicación web a una aplicación móvil de manera nativa.

Se ha decidido usar *Capacitor* por las siguientes razones:

[1]Componentes reutilizables que siguen las especificaciones *HTML* y *DOM* establecidas por el *W3C*

- Proporciona un desarrollo único para las diferentes plataformas móviles.
- Proporciona acceso a los componentes del dispositivo de forma nativa.
- Soporte para *plugins* que permiten acceder a nuevas funcionalidades del sistema.
- Permite la ejecución de código nativo dentro de la plataforma.

Cypress [42]

Es un librería *Javascript* que nos permite ejecutar pruebas en aplicaciones web de manera automática, pero pensada para aplicaciones web, ya que dichas pruebas se ejecutan en el navegador y comprueban tanto el correcto funcionamiento de la aplicación como su flujo de ventanas.

Las ventajas que proporciona *cypress* frente a otras herramientas de pruebas son las siguientes:

- Contiene un entorno gráfico que permite una programación más ágil.
- Soporta la ejecución de pruebas en cualquier navegador, independientemente del sistema operativo utilizado.
- Permite acceder a los elementos que se necesitan probar de una manera sencilla.

4.1.2 Capa de Negocio

La capa de lógica de negocio es ejecuta en *NodeJS* y programada en *Typescript* usando para ello el *framework NestJS* y otras librerías como: *JWT*, *bcrypt*, *mongoose*... para autenticicación y conexiones con los servicios.

Adicionalmente, para el desarrollo numérico/computacional se ha usado *Python* y la librería *Sklearn*.

Se han creado dos servidores [Figura 4.3] para comunicar ambos lenguajes de programación, donde el servidor de *Python* solo se comunica con el de *Node*, el cual realiza las conexiones al exterior.

NestJS [43]

Es un *framework* de desarrollo para aplicaciones servidor en *NodeJS*. Utiliza Typescript como lenguaje principal de programación y contiene una extensa documentación que abarca la mayor parte de las necesidades requeridas de un servidor. Las ventajas que proporciona este *framework* son las siguientes:

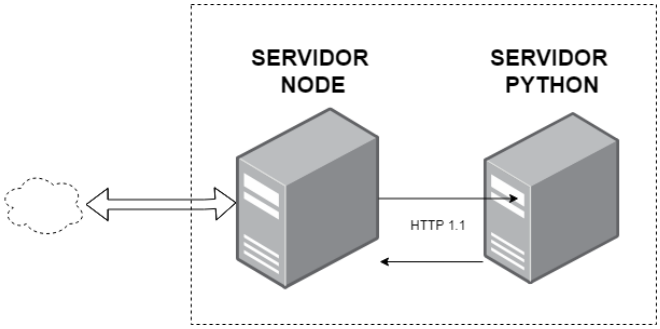

Figura 4.3: Capa de lógica de negocio

- Hace uso de decoradores[2], lo que supone una programación más ágil, legible y menos propensa a errores.
- Se integra perfectamente con la mayoría de las herramientas existentes, haciendo que la programación con servicios de terceros funcione correctamente (Base de datos, OpenAPI, Jest ...)
- Utiliza una estructura similar a la usada por *Angular*. Por lo tanto, al usar estos dos *frameworks* en el lado del cliente y el lado del servidor se reduce la complejidad de lectura del código.

Jest [44]

Es un librería *Javascript*, desarrollado por Facebook, que contiene un conjunto de herramientas para ejecutar pruebas de forma automática.

El *framework* de *NestJs* proporciona documentación sobre las pruebas utilizando esta librería, además de ser una de las más usadas en entornos *Javascript*. También proporciona soporte para *Typescript*.

Sklearn

Sklearn [45] es una librería de aprendizaje máquina para el lenguaje de programación *Python*. Se ha usado *Sklearn* por las siguientes razones:

- Se encuentra bien documentada, incluyendo ejemplos para cada función.
- Mantiene un interfaz consiste ante los distintos modelos de aprendizaje automático.
- Proporciona funcionalidades adicionales que permiten un desarrollo más ágil.

[2]Envolturas que se utilizan sobre funciones, clases... para añadir más funcionalidad

4.1.3 Capa de Datos

Esta capa se encarga de almacenar los datos, para ello se ha usado *MongoDB*, una base de datos NoSQL. Se ha utilizado este tipo de base de datos frente a una relacional por los siguientes motivos:

- **Velocidad:** Debido a que el sistema necesita realizar muchas operaciones de lectura y escritura.

- **Volumen:** La aplicación generará una gran cantidad de datos, puesto que estará almacenando principalmente los eventos de un usuario. El volumen de eventos generados de un uso continuado será elevado, por lo que se necesita una base de datos que soporte tal volumen de información.

- **Variabilidad:** La aplicación manejara datos, que pueden no ser consistentes e incluso algunos de ellos pueden modificarse con el tiempo. Este tipo de base de datos no necesitan un esquema para trabajar por lo que las hace ideales para este tipo de casos.

MongoDB [46]

Es una base de datos orientada a documentos. Los datos están almacenados en documentos JSON y agrupados en colecciones. El formato interno que maneja es BSON, el cual extiende las características de JSON.

Dado que estamos usando una base de datos basada en documentos, el esquema definido para guardar los datos no estará restringido, es decir, se podrán insertar datos no definidos en el esquema. Se ha hecho de esta manera para permitir cambiar el esquema en un futuro, debido a que los datos que se van a guardar pueden ser modificados en el tiempo dado la naturaleza del proyecto.

La base de datos contiene un esquema [Tabla 4.1] para almacenar los datos de los eventos y por cada tipo de evento se crea un colección.

EVENTS	
email	String
events	Object

Tabla 4.1: Esquema de la base de datos

4.2 Herramientas de Gestión y Desarrollo

Para desarrollar todo este sistema se han utilizado las siguientes herramientas tanto para gestión del proyecto como para el desarrollo del mismo.

- **Git [47]**: Herramienta para la gestión y control de versiones, desarrollado por Linus Torvalds.
- **VS Code [48]**: *IDE* creado por *Microsoft*, para el desarrollo de software, centrado en lenguaje *Javscript*.
- **Android Studio [49]**: *IDE* creado por *Google*, para el desarrollo de aplicaciones Android.
- **Javascript [50]**: Lenguaje de programación creado en 1995 por *Netscape* y *Mozilla*, que sigue el estándar definido por *ECMA*.
- **Typescript [51]**: Lenguaje de programación creado por *Microsoft* que es un *superset* de *Javascript*, que esencialmente añade tipos, clases y decoradores.
- **Python [52]**: Lenguaje de programación orientado a objetos, creado por Guido van Rossum en 1991.
- **LaTeX [53]**: es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica.

Capítulo 5

Metodología

En este capítulo se detallarán el conjunto de procedimientos utilizados para alcanzar los objetivos del proyecto.

5.1 Desarrollo incremental

Este método consiste en una serie de iteraciones, en pequeñas ventanas de tiempo, en las que al final de las mismas tendremos un producto que el cliente podrá revisar y posteriormente mejorar y/o corregir [Figura 5.1].

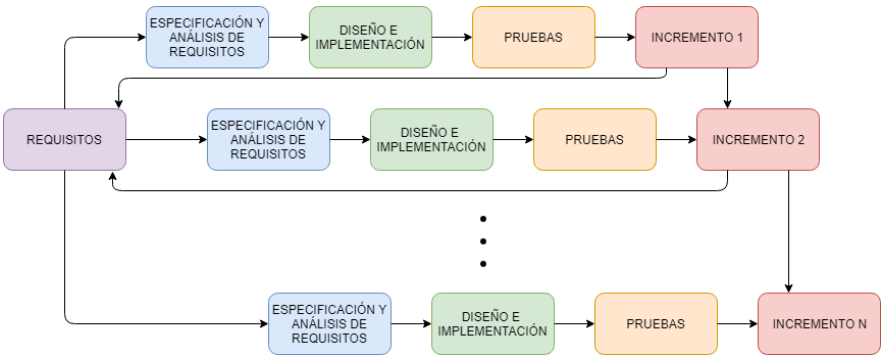

Figura 5.1: Fases del modelo incremental

Esta metodología exige tener dos grupos de usuarios:

- **Cliente**: Se encarga de revisar el producto al final de las iteraciones.
- **Desarrollador**: Se encarga de desarrollar el producto.

Al ser un proyecto de fin de carrera, no existe un cliente como tal, por lo que se ha decidido que los directores del proyecto tomen el rol de clientes.

Las principales razones por la se ha elegido esta metodología frente a otras existentes son las siguientes:

1. **El cliente no sabe exactamente lo que necesita**: Al inicio del proyecto no se podía prever cómo sería la aplicación, pues debido a la complejidad del proyecto, no se sabía cuál podría ser la mejor manera de hacer que la aplicación para recoger los datos necesarios.

2. **Obtener un producto usable rápidamente**: Este proyecto tiene dos etapas claramente diferenciadas, construir una aplicación para recoger datos y analizar esos datos. Claramente la segunda es totalmente dependiente de la primera, por lo que era necesario disponer de un producto usable que permita recopilar información a medida que se amplia su funcionalidad.

5.2 Iteración 0: Búsqueda de información sobre el dominio

El objetivo de esta iteración es la búsqueda y recopilación de información sobre el dominio del proyecto. También se definirán los requisitos que el sistema final deberá cumplir, la estructura que se usará para el desarrollo [Figura 4.1] y se detallarán los componentes de más bajo nivel [Figura 5.2] [Figura 5.4]. De esta forma el desarrollo del proyecto estará bien definido y su implementación debería ser más fácil.

5.3 Iteración 1: Sistema de captura de información

El objetivo de esta fase es la de crear una aplicación que permita capturar la información proveniente del dispositivo. El producto creado al final de esta iteración será una aplicación funcional. El manual de usuario asociado a la aplicación puede consultarse en el Apéndice B.

5.3.1 Especificación y análisis de requisitos

A continuación, se describen los requisitos que la aplicación ha de cumplir:

- **Gestión de los recursos:** Dado que la aplicación recolectará toda la información de los eventos generados por la interacción del usuario con el dispositivo, hay que tener en cuenta cómo manejar dicho volumen de información para evitar el colapso del dispositivo, ya que al tratarse de un *smartphone* o *Tablet*, cuentan con unos recursos limitados tanto en hardware como batería.

- **Formato de los eventos:** Para facilitar el análisis a posteriori la aplicación debe mantener un formato similar en todos los eventos que capture.

5.3.2 Diseño e implementación

Para realizar la implementación y el diseño de la aplicación se ha usado *Angular* e *Ionic* respectivamente, usando la estructura que se muestra en la Figura 5.2.

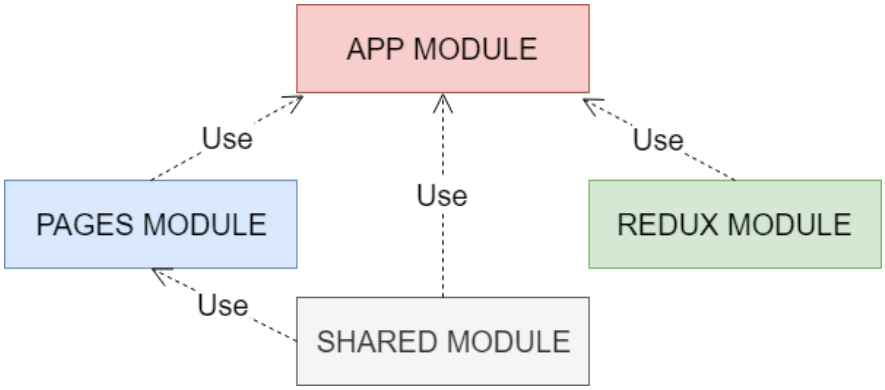

Figura 5.2: Estructura de la aplicación

App Module

Es el módulo principal de la aplicación, en él se encuentra la configuración de los sistemas, y se encarga de hacer de nodo de enlace entre los demás módulos.

Redux Module

Este módulo implementa toda la lógica del patrón *Redux*. Este patrón permite manejar el estado de la toda aplicación haciendo uso de un objeto central inmutable. Esto permite generar un flujo de datos de una sola dirección y de esta manera que el código sea menos propenso a errores.

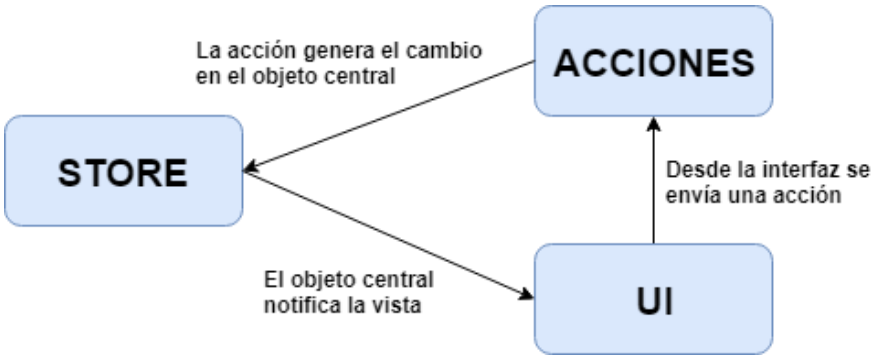

Figura 5.3: Ciclo del patrón redux

Pages Module

Este módulo agrupa al conjunto de ventanas de la aplicación. Cada una de ellas esta implementada en un submódulo. Dicha estructura es implementada por defecto por *Ionic* para poder aplicar la estrategia de carga perezosa (*lazy loading*), la cual solo carga lo que se ve en el momento. Esto permite que la primera carga sea mucho más rápida.

Shared Module

Este módulo contiene los distintos servicios, componentes... que son reutilizados por varios módulos de la aplicación.

5.4 Iteración 2: Sistema para el almacenamiento de la información

En esta fase se pretende crear un servidor que permita el almacenamiento de los eventos recogidos en la primera fase para su posterior análisis. El producto creado al final de esta iteración expone una *API* que se puede consultar en el Apéndice A.

5.4.1 Especificación y análisis de requisitos

A continuación, se describen los requisitos que el servidor ha de cumplir:

- **Gestión de los recursos:** Dado que el servidor ha de permitir el envío y recibo de muchas peticiones por segundo, se ha de tener en cuenta, que el servidor soporte dicha funcionalidad a un coste de recursos bajo.
- **Almacenamiento de los eventos:** Dado que la cantidad de eventos que se manejarán será muy alta y con un esquema semi-definido, ya que podría cambiar en un futuro, se necesita una base de datos que sea compatible con estos puntos.

5.4.2 Diseño e implementación

Debido a los requisitos comentados se ha optado por usar Node.js como entorno, por su bajo consumo de recursos. La estructura generada se puede ver en la Figura 5.4.

App Module

Es el módulo principal desde el cual se llaman a los otros módulos e inicia servicios de terceros, como la conexión con la base datos.

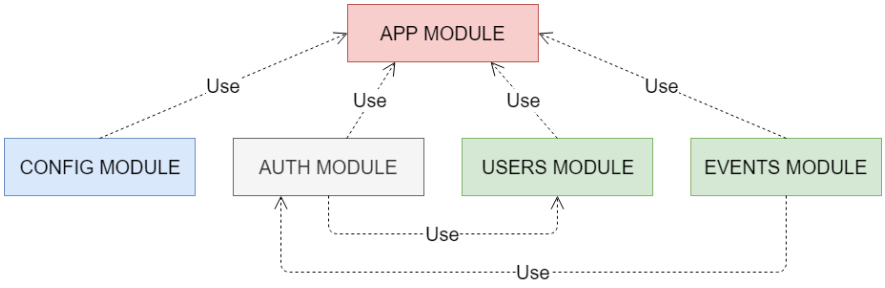

Figura 5.4: Estructura de la aplicación

Config Module

Este módulo contiene todas las variables externas del sistema, como el puerto, dirección de la base de datos ...

Auth Module

En este módulo se realiza la autenticación del usuario, la cual permitirá al usuario realizar ciertas acciones y restringirá otras en función a su rol.

Users Module

En este módulo se encuentra la lógica encargada de procesar la información de los usuarios, como la creación y actualización de estos.

Events Module

En este módulo se encuentra la lógica necesaria para procesar la información relacionada con los eventos que se generan en la aplicación, almacenándolos de forma persistente en la base de datos.

5.5 Iteración 3: Comunicar aplicación con servidor

El objetivo de esta fase es la de comunicar los dos sistemas creados hasta el momento, con el fin de guardar los eventos generados en la primera iteración

5.5.1 Especificación y análisis de requisitos

A continuación, se describen los requisitos establecidos para esta iteración:

- **Securizar conexiones**: Dado que la comunicación se pretende realizar a través de internet usando el protocolo *HTTP 1.1*, es necesario que esta comunicación permanezca cifrado de punto a punto, por lo que usarán los protocolos criptográficos *SSL/TLS*.
- **Restringir peticiones**: Algunas peticiones hacia nuestro servidor deberán estar solo disponibles si el usuario cumple ciertas restricciones. Estás peticiones deberán ser realizadas por un usuario administrador.

5.5.2 Diseño e implementación

Para segurizar toda nuestra aplicación y hacer que esta sea menos propensa a ataques, se ha creado un certificado digital para poder realizar conexiones *HTTPS*.

Para restringir el acceso a nuestro servidor hemos utilizado el estándar *JSON Web Token* (*JWT*), el cual es una cadena de texto codificada que se envía en cada petición. Esta cadena de texto contiene tres partes que contienen la información necesaria para comprobar el usuario y la validez del token.

5.6 Iteración 4: Implementación de la Inteligencia Artificial

El objetivo de esta iteración es la de tratar los datos obtenidos. Analizando que características obtienen un mejor resultado a la hora de identificar al usuario. Esta iteración se detallará más a fondo en el capítulo 6.

5.6.1 Especificación y análisis de requisitos

A continuación, se describen los requisitos establecidos para esta iteración:

- **Obtener un conjunto de datos grande**: Para este tipo de análisis se necesitan una gran cantidad de datos y representativos de la población. Por lo tanto, es necesario obtener datos de bastantes usuarios de diferentes edades y sexos.
- **Obtener características no relacionadas**: Para analizar los eventos, lo ideal es obtener características poco correlacionadas, ya que si estas están correlacionadas estos datos serán redundantes. Es decir, no aportarán ninguna información adicional, pero podrán empeorar los resultados.

5.7 Iteración 5: Implementación online de los algoritmos

El objetivo de esta iteración es conseguir que el algoritmo entrenado sea accesible en cualquier momento y que su respuesta sea la predicción.

5.7.1 Especificación y análisis de requisitos

A continuación, se describen los requisitos establecidos para esta iteración:

- **Integrar los servidores**: Para que el algoritmo no este expuesto directamente en internet, sino que solo sea accesible mediante el servidor principal. De esta manera, la configuración relacionada con la seguridad solo será necesario implementarla en un servidor.
- **Gestión de recursos**: El servidor se encontrará en un maquina física por lo que es necesario controlar que los procedimientos que se lleven a cabo no sean computacionalmente costosos.

5.7.2 Diseño e implementación

Para conseguir el objetivo de esta iteración se ha creado un servidor en *Python* que permita la ejecución de un conjunto de librerías las cuales usarán el algoritmo entrenado y la respuesta de la petición generada será la predicción hecha por el algoritmo.

5.8 Pruebas

Las pruebas son las actividades dirigidas a evaluar la capacidad de un programa o sistema y determinar que alcanza los resultados requeridos [54].

Existen distintos niveles de pruebas, según se puede observar en la figura 5.5, el nivel más bajo pertenecen a las pruebas unitarias, que son las que mayor atención deberemos prestar, seguidas de las pruebas de integración y por último las pruebas *e2e*.

Figura 5.5: Pirámide de Cohn [55]

5.8.1 Prueba unitarias

Estas pruebas son de muy bajo nivel, en ellas se comprueba el correcto funcionamiento de los métodos, clases, componentes...

Este tipo de pruebas han sido ejecutadas tanto en el servidor como en la aplicación, para validar el correcto funcionamiento de algunas de las partes más críticas del sistema.

5.8.2 Pruebas de integración

Comprueban el correcto funcionamiento de los componentes una vez integrados. Se ha revisado la integración completa del sistema, validando que todos los módulos funcionasen correctamente.

Este tipo de pruebas han sido ejecutadas en el servidor para validar las rutas creadas. Se han implementado tres bancos de pruebas, cada uno de ellos perteneciente a un módulo de nuestro sistema. En total se han ejecutado doce casos diferentes, y todas han sido satisfactorias [Figura 5.6]

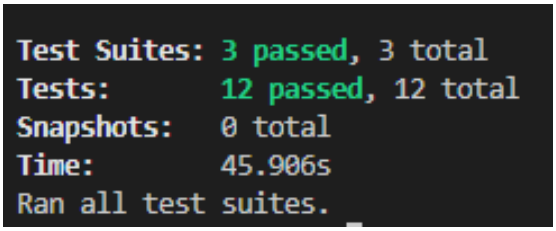

Figura 5.6: Resultado de las pruebas en el servidor

5.8.3 Pruebas end to end (e2e)

Este tipo de pruebas simulan el comportamiento de un usuario real. Prueban toda la aplicación de principio a fin, verificando el correcto funcionamiento de todo el sistema y la navegación entre las distintas ventanas de la aplicación.

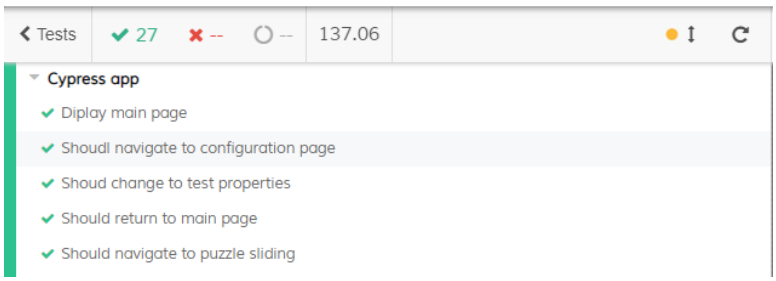

Figura 5.7: Resultado de las pruebas en la aplicación [56]

5.8.4 Cobertura

La cobertura es una medida que nos ayuda a comprobar que partes de nuestro código han sido probadas. Además, sirve para determinar la calidad de la prueba, pues gracias a ella podremos saber que partes del código no hemos comprobado y cuales sí.

97.93% Statements 378/386 **83.33%** Branches 20/24 **94.44%** Functions 51/54 **97.63%** Lines 329/337

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲	Statements		Branches		Functions		Lines	
src	100%	31/31	100%	0/0	100%	3/3	100%	28/28
src/auth	98.04%	50/51	75%	3/4	100%	5/5	97.67%	42/43
src/auth/models	100%	6/6	100%	0/0	100%	0/0	100%	6/6
src/config	81.82%	18/22	50%	1/2	62.5%	5/8	80%	16/20
src/decorators	100%	3/3	100%	0/0	100%	1/1	100%	2/2
src/events	99.37%	158/159	90%	9/10	100%	23/23	99.26%	135/136
src/events/models	100%	15/15	100%	2/2	100%	1/1	100%	15/15
src/guards	95%	19/20	50%	1/2	100%	3/3	93.75%	15/16
src/pipes	100%	7/7	100%	2/2	100%	1/1	100%	5/5
src/users	98.25%	56/57	100%	0/0	100%	8/8	98.04%	50/51
src/users/models	100%	15/15	100%	2/2	100%	1/1	100%	15/15

Figura 5.8: Cobertura de las pruebas del servidor

Capítulo 6

Análisis y tratamiento de datos

Para implementar un sistema de Inteligencia Artificial, en adelante *IA*, que clasifique un conjunto de datos, necesitaremos obtener mucha información para *alimentarla*. Estos datos normalmente en bruto contienen información redundante, valores atípicos, campos nulos… todo esto haría que nuestra *IA* se viese afectada negativamente, para ello necesitaremos tratar correctamente los datos.

En la última parte del capítulo se seguirá el proceso de selección de los algoritmos más plausibles, así como su ajuste para refinar progresivamente los modelos elaborados.

En el capítulo 7, se mostrarán y analizarán los resultados obtenidos después de realizar el procedimiento mencionado en este capítulo.

6.1 Tratamiento de los datos

La mayoría de los de algoritmos de aprendizaje presentan varias necesidades previas a la hora de trabajar con un conjunto de datos. Por ello, se necesita procesar la información y formatearla correctamente para los entrenamientos. En esta sección se abordarán las técnicas empleadas para solventar dicho problema.

6.1.1 Variables de texto

La mayoría de los algoritmos de aprendizaje mejor con números que con etiquetas. Para que nuestros datos se adecuen a estas circunstancias debemos manipularlos correctamente.

Nuestro conjunto de datos solo presenta una variable de texto que es la de usuario, la cual hace referencia al dueño del evento. Para formatear estos valores a números, se ha usado una función de la librería de *Sklearn*, la cual asigna un único valor a cada etiqueta.

6.1.2 Variables categóricas

Estas variables representa un conjunto de datos, es decir, cada valor representa una categoría.

En nuestro caso el dato dirección es una variable categórica, en la cual cada valor representa una dirección única. Sino formateamos este valor, el algoritmo entenderá que los valores más próximos están más relacionados. Para solucionar este problema hemos usado una función de la librería de *Sklearn*, la cual genera una columna por cada valor posible y le asigna el valor cero o uno en función del valor de la columna original 6.1.

Columna Original	**Columna Transformada**		
direction	direction_4	direction_8	direction_16
8	0	1	0
4	1	0	0
16	0	0	1

Tabla 6.1: Corrección de la variable categórica

6.1.3 Valores Nulos

La mayoría de los algoritmos de aprendizaje no pueden trabajar con valores nulos, por lo tanto, necesitamos tratar nuestros datos para detectarlos y procesarlos correctamente. Para realizar este paso se contemplaron las siguientes opciones:

1. Eliminar la fila que los contiene.
2. Eliminar la columna que los contiene.
3. Rellenar a cero dichos valores.
4. Rellenar los valores con la mediana.

De las opciones mencionas se eligió la última ya que es menos agresiva a la hora de modificar los datos. La librería de *Sklearn* proporciona un método que nos ayuda con este proceso, y asigna a los valores nulos el valor que hemos escogido, en nuestro caso la mediana.

6.1.4 Transformación de las características

Para encontrar nuevas relaciones entre las características y las clases se ha utilizado el método de *polynomial feature* de la librería *Sklearn*.

En el ejemplo de la Figura 6.1 la imagen de la izquierda representa un problema linealmente no separable. Pero si elevamos al cuadrado la características x_1 los datos pasan a ser linealmente separables.

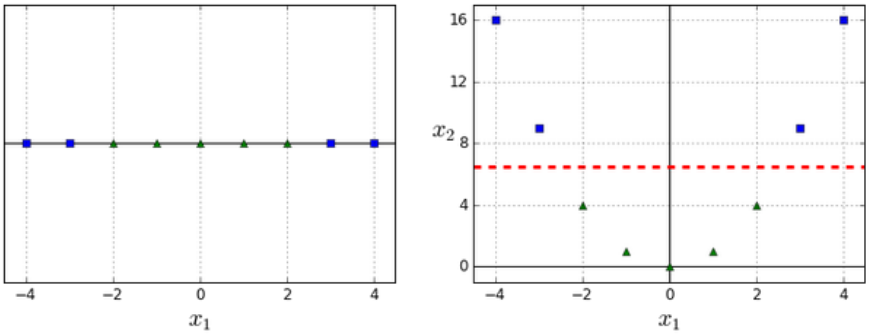

Figura 6.1: Ejemplo de Polynomial Feature

En este proyecto se utilizo esta técnica para generar características de segundo grado, es decir, si tenemos las características $[a, b]$, el segundo grado sería $[1, a, b, a^2, ab, b^2]$.

6.1.5 Estandarizar los datos

La mayor parte de los algoritmos de aprendizaje disminuyen su rendimiento cuando los valores de las variables se mueven en un rango muy amplio. Dentro de las características de los eventos existen varias variables con esta problemática, como es el caso de *duración*. Esta variable tiene un valor medio de 122 *ms* mientras que su rango de valores es [13, 1654] [Tabla 6.2].

	count	mean	std	min	25%	50%	75%	max
duration	3436.0	122.40	72.48	13.0	83.0	104.0	146.0	1654.0

Tabla 6.2: Estadísticas de la característica duración

Para realizar esta tarea se hace uso de la librería *Sklearn*, la función utilizada realiza el siguiente proceso:

- Resta el valor medio, lo cual hace que los valores cerca de la media tiendan a cero.
- Divide el valor por el rango intercuartílico [1]. Se usan los cuartiles como valor de división porque hace que sea menos propenso a los valores atípicos

[1] Diferencia entre el tercer y el primer cuartil

6.2 Selección de algoritmos

Para realizar las pruebas iniciales se han seleccionado los siguientes algoritmos:

- **LogisticRegression (LR)** (Véase Sección [2.5.6)
- **RandomForest (RF)** (Véase Sección 2.5.9)
- **DecisionTree (DT)** (Véase Sección 2.5.5)
- **Perceptrón Multicapa (MLP)** (Véase Sección 2.5.3)
- **K-Nearest Neighbors (kNN)** (Véase Sección 2.5.7)
- **Máquinas de soporte vectorial (SVM)** (Véase Sección 2.5.4)

El primer paso consiste en realizar una prueba para tener un punto de inicio sobre el cuál comparar los resultados, y de esta manera comprobar analíticamente si las técnicas aplicadas a lo largo del capítulo han servido para mejorar la clasificación.

Analizando los resultados de la Tabla 6.3 en general todos los algoritmos obtienen unos valores similares en las columnas de *Recall*, *Precision* y *F1 Score*, con una desviación del ±5%. Si comparamos los resultados de los algoritmos individualmente podemos obtener las siguientes conclusiones:

- ***RF***: Los resultados obtenidos por este algoritmo en la columna de *precision* son los más altos exceptuando el evento de *tap*. Los tiempos de entrenamiento (*Fit Time*) están muy por debajo del resto de algoritmos.
- ***DT***: Los valores de las columnas de *Recall*, *Precision* y *F1 Score* se mantienen en un rango de valores pequeño.
- ***LR***: Los valores obtenidos en la columna *precision* son los segundos más altos excepto en *tap* donde es el más alto y, al igual que *RF* el tiempo de entrenamiento suele ser bajo.
- ***MLP***: Los tiempos de predicción (*Score Time*) obtenidos son los más bajos en todos los eventos.
- ***SVM***: Su desviación es la más alta y, por lo tanto, optimizando sus parámetros podría reducirse esta desviación y mejorando los resultados.
- ***kNN***: Este algoritmo obtuvo el rendimiento más bajo tanto en *precision* como en *Fit Time*.

Para reducir el tiempo de procesamiento y optimizar la búsqueda se ha decidido los siguientes algoritmos:

- (*DT*) debido a que *RF* obtiene mejores resultados y al ser este un conjunto de *DT* es redundante entrenar dos algoritmos de características similares.
- *KN* porque sus valores son lo que han obtenido el rendimiento más bajo.

Evento	Alg.	Recall	Precision	F1 Score	Fit Time (s)	Score Time (s)
motion	*DT*	71.01% (±0.122)	71.96% (±0.10 8)	71.11% (±0.110)	0.072	0.121
	kNN	70.85% (±0.122)	70.48% (±0.108)	70.29% (±0.110)	0.225	0.125
	LR	71.22% (±0.130)	73.14% (±0.112)	71.76% (±0.115)	0.088	0.182
	MLP	71.66% (±0.120)	71.48% (±0.107)	71.22% (±0.107)	0.269	0.094
	RF	71.09% (±0.135)	76.33% (±0.104)	73.33% (±0.118)	0.054	0.324
	SVM	69.01% (±0.139)	70.82% (±0.123)	69.08% (±0.118)	0.193	0.118
multitouch	*DT*	71.45% (±0.126)	72.55% (±0.110)	71.58% (±0.112)	0.073	0.145
	kNN	70.98% (±0.120)	70.82% (±0.107)	70.51% (±0.108)	0.251	0.103
	LR	71.70% (±0.131)	74.40% (±0.108)	72.55% (±0.113)	0.044	0.246
	MLP	71.61% (±0.121)	71.75% (±0.108)	71.30% (±0.108)	0.147	0.105
	RF	75.95% (±0.120)	77.86% (±0.097)	76.52% (±0.104)	0.048	0.326
	SVM	69.74% (±0.138)	71.00% (±0.119)	69.53% (±0.115)	0.206	0.122
orientation	*DT*	71.86% (±0.121)	72.82% (±0.106)	71.95% (±0.108)	0.071	0.128
	kNN	71.37% (±0.120)	71.01% (±0.106)	70.81% (±0.107)	0.233	0.118
	LR	72.25% (±0.130)	74.53% (±0.107)	72.95% (±0.112)	0.072	0.199
	MLP	72.13% (±0.119)	72.02% (±0.105)	71.72% (±0.106)	0.220	0.097
	RF	75.41% (±0.110)	79.54% (±0.086)	77.14% (±0.096)	0.053	0.325
	SVM	69.51% (±0.138)	71.38% (±0.119)	69.61% (±0.115)	0.196	0.119
pan	*DT*	70.96% (±0.122)	71.71% (±0.110)	70.97% (±0.110)	0.069	0.115
	kNN	70.65% (±0.129)	70.06% (±0.111)	69.93% (±0.113)	0.218	0.128
	LR	70.59% (±0.129)	72.38% (±0.113)	71.05% (±0.115)	0.084	0.168
	MLP	71.19% (±0.122)	71.11% (±0.108)	70.75% (±0.110)	0.271	0.090
	RF	70.29% (±0.133)	75.78% (±0.110)	72.60% (±0.121)	0.053	0.324
	SVM	67.57% (±0.162)	71.14% (±0.125)	67.91% (±0.137)	0.188	0.115
swipe	*DT*	72.07% (±0.125)	73.09% (±0.109)	72.17% (±0.111)	0.070	0.136
	kNN	71.42% (±0.120)	71.18% (±0.107)	70.92% (±0.108)	0.242	0.111
	LR	73.25% (±0.132)	75.39% (±0.106)	73.87% (±0.113)	0.044	0.220
	MLP	72.20% (±0.121)	72.13% (±0.106)	71.80% (±0.108)	0.158	0.101
	RF	77.98% (±0.104)	80.90% (±0.091)	79.13% (±0.094)	0.049	0.326
	SVM	69.97% (±0.137)	71.44% (±0.120)	69.88% (±0.115)	0.201	0.120
tap	*DT*	70.55% (±0.125)	72.10% (±0.111)	70.92% (±0.112)	0.078	0.156
	kNN	71.17% (±0.120)	71.11% (±0.106)	70.76% (±0.108)	0.261	0.098
	LR	70.15% (±0.125)	74.98% (±0.108)	72.15% (±0.114)	0.046	0.280
	MLP	71.03% (±0.119)	71.74% (±0.109)	71.03% (±0.108)	0.140	0.110
	RF	68.49% (±0.072)	73.82% (±0.082)	70.98% (±0.075)	0.048	0.327
	SVM	70.72% (±0.128)	70.15% (±0.111)	70.00% (±0.112)	0.212	0.125

Tabla 6.3: Resultados de la primera ejecución

6.3 Selección de hiperparámetros

En esta sección se verá cómo se han configurado los algoritmos. Dicha configuración se materializa a través de los hiperparámetros. Estos son valores que se establecen antes del entrenamiento del algoritmo y que deciden en parte como van a funcionar dichos algoritmos, permitiendo que se ajusten mejor al modelo.

Para buscar estos hiperparámetros Se suelen utilizar dos técnicas:

- ***Grid Seach***: Es un busca exhaustiva de los mejores parámetros sobre una combinación dada.
- ***Random Search***: Es un búsqueda aleatoria sobre una combinación de parámetros dada.

Para este proyecto se ha usado la técnica de *Grid Search* como la idónea debido a publicaciones [57] sobre el rendimiento de ambas técnicas.

Para analizar los resultados y obtener un valor los más aproximado posible se han seguido los siguientes pasos:

1. Se han ordenado los resultados en orden descendente y se han seleccionado los cinco mejores valores de cada algoritmo.
2. Se han calculado las puntuaciones medias de esos valores y su desviación, con el fin de obtener unos tiempos y puntuaciones medios [Tabla 6.4] para los mejores hiperparámetros.
3. Se han calculado también los valores más comunes para obtener los mejores hiperparámetros [Tabla 6.5].

Random Forest			
Criterion	Max Depth	Max Features	Number Of Estimators
gini	5.0	log2	21.0
Support Vector Machines			
Gamma	C	Kernel	
0.1	1.0	rbf	
Multilayer Perceptron			
Solver	Alpha	Hidden Layer Sizes	
lbfgs	10.0	60.0	

Tabla 6.5: Mejores hiperparámetros para cada algoritmo

Evento	Alg.	Precision	Fit Time (s)	Score Time (s)
motion	*LR*	63.23% (± 0.047)	1.1629 (± 0.047)	0.0268 (± 0.047)
	MLP	65.79% (± 0.080)	10.4182 (± 0.080)	0.0923 (± 0.080)
	RF	68.42% (± 0.060)	0.3167 (± 0.060)	0.0160 (± 0.060)
	SVM	65.64% (± 0.132)	0.1740 (± 0.132)	0.0762 (± 0.132)
multitouch	*LR*	76.98% (± 0.128)	0.0346 (± 0.128)	0.0164 (± 0.128)
	MLP	80.99% (± 0.030)	0.6049 (± 0.030)	0.0218 (± 0.030)
	RF	86.93% (± 0.089)	0.0811 (± 0.089)	0.0091 (± 0.089)
	SVM	100.00% (± 0.000)	0.0059 (± 0.000)	0.0043 (± 0.000)
orienntation	*LR*	70.09% (± 0.069)	12.5253 (± 0.069)	0.0248 (± 0.069)
	MLP	66.72% (± 0.033)	28.8783 (± 0.033)	0.1177 (± 0.033)
	RF	77.28% (± 0.048)	0.1783 (± 0.048)	0.0133 (± 0.048)
	SVM	90.43% (± 0.152)	0.1458 (± 0.152)	0.0746 (± 0.152)
pan	*LR*	67.48% (± 0.148)	0.1268 (± 0.148)	0.0103 (± 0.148)
	MLP	77.65% (± 0.089)	1.1181 (± 0.089)	0.0334 (± 0.089)
	RF	80.99% (± 0.094)	0.1343 (± 0.094)	0.0122 (± 0.094)
	SVM	78.96% (± 0.099)	0.0067 (± 0.099)	0.0045 (± 0.099)
swipe	*LR*	84.67% (± 0.043)	0.5641 (± 0.043)	0.0241 (± 0.043)
	MLP	81.14% (± 0.044)	0.7857 (± 0.044)	0.0822 (± 0.044)
	RF	85.56% (± 0.045)	0.1575 (± 0.045)	0.0130 (± 0.045)
	SVM	94.07% (± 0.066)	0.0847 (± 0.066)	0.0383 (± 0.066)
tap	*LR*	70.54% (± 0.080)	0.1695 (± 0.080)	0.0149 (± 0.080)
	MLP	75.42% (± 0.090)	7.6387 (± 0.090)	0.1006 (± 0.090)
	RF	75.52% (± 0.092)	0.1926 (± 0.092)	0.0149 (± 0.092)
	SVM	85.90% (± 0.100)	0.0768 (± 0.100)	0.0278 (± 0.100)

Tabla 6.4: Resultados medios con los mejores hiperparámetros

6.4 Selección de características

En esta sección se detallarán las técnicas utilizadas para la selección de características y los resultados obtenidos. La selección de características es una técnica que permite distinguir qué características son las más importantes a la hora de clasificar nuestros datos, esta técnica aporta las siguientes mejoras:

1. A menudo aumenta la precisión de la clasificación a través de la eliminación de características irrelevantes, redundantes o altamente correlacionadas.

2. Eliminar características redundantes implica que exista un menor riesgo de *overfitting*.

3. Disminuye la cantidad de características, lo que hace que el proceso de entrenamiento

del modelo sea más eficiente

Existen diferentes técnicas para la selección de características, algunas de ellas son:

- **Selección univariante**: Muestra la relación de la salida con las variables de entrada basándose en métodos estadísticos, como chi cuadrado.
- ***Recursive feature elimination* (RFE)**: Esta técnica consiste en ir eliminar progresivamente las características menos relevantes.
- **Basada en árboles**: Esta técnica consiste en entrenar un algoritmo basado en árboles, como el *Random Forest*, y después examinar que características son más relevantes para el algoritmo.

Para seleccionar las características hemos empleado el último método [Figura 6.2]. Se ha descartado aplicar la primera técnica porque solo es confiable si las características son completamente independientes. La segunda técnica fue descartada porque los algoritmos *SVM*, con un kernel no lineal, y *MLP* no exponen los coeficientes con los pesos internos de las características necesarios para aplicar *RFE*.

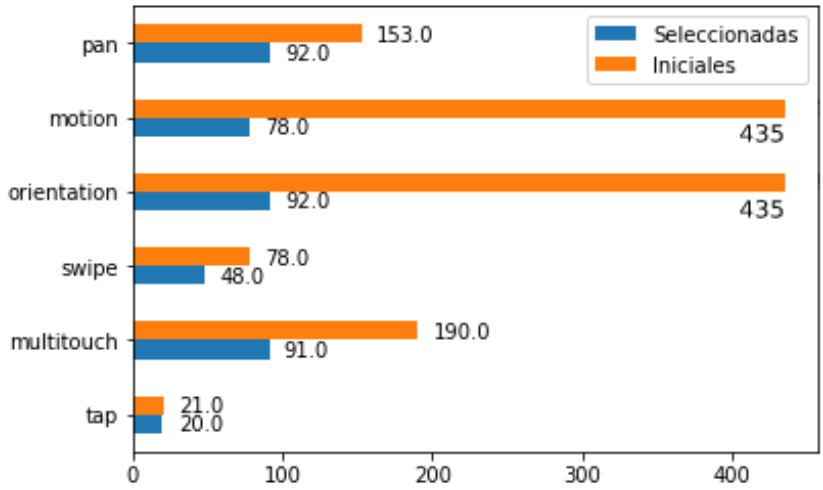

Figura 6.2: Características Iniciales VS Características Seleccionadas

Capítulo 7

Resultados

En este capítulo se presentarán los resultados obtenidos para cada una de las técnicas exploradas. Para ello se utilizarán las métricas de *precision*, *acuraccy*, *F1 Score*, *Fit Time* y *Score Time* (véase la Sección 2.4) y los algoritmos *Random Forest*, *SVM*, *MLP* y votación.

Dada la gran cantidad de conjuntos de entrenamiento que se han establecido, primero se analizarán los resultados globales agrupados por eventos y, posteriormente, se expandirán para cada uno de los usuarios. Por último, se mostrarán los resultados obtenidos de la prueba online.

7.1 Datos agrupados por eventos

En esta sección se presenta el rendimiento que han ofrecido los diferentes algoritmos. En la Tabla 7.1 se muestran los resultados obtenidos utilizando todas las características de los eventos.

Como se puede apreciar, el valor medio de *precision* obtenido se encuentra en 72%, un porcentaje aceptable para esta primera aproximación del problema. El resultado medio del resto de métricas es bajo. Esto es debido a que ciertos tipos de eventos no obtienen el rendimiento esperado y, por lo tanto la media decae. Si analizamos los algoritmos observamos que *MLP* no alcanza los resultados deseados, mientras que los otros tres algoritmos obtienen los valores esperados.

La siguiente prueba se realizó bajo las mismas configuraciones que la primera, a excepción del número de características. En esta segunda prueba los algoritmos se entrenaron con un conjunto de datos cuyas características fueron seleccionadas (veáse Seccción 6.4). Los datos de esta ejecución se muestran en la Tabla 7.2.

Los resultados obtenidos por esta prueba, muestran un incremento promedio en las métricas de *precision*, *acuraccy*, *F1 Score* y una disminución de los tiempos de cómputo. Observando los resultados individualmente se puede comprobar una mejora en la mayoría de casos, es-

Ev.	Alg.	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Fit Time(s)	Score Time(s)
tap	Vot.	74.66 (±0.05)	77.08 (±0.07)	73.52 (±0.09)	74.82 (±0.06)	0.197	0.013
	SVM	73.27 (±0.05)	75.43 (±0.07)	72.35 (±0.07)	73.59 (±0.05)	0.017	0.008
	RF	76.48 (±0.05)	78.31 (±0.07)	76.24 (±0.08)	76.91 (±0.05)	0.020	0.002
	MLP	70.55 (±0.10)	69.57 (±0.19)	67.85 (±0.20)	68.31 (±0.19)	0.156	0.001
swipe	Vot.	74.33 (±0.07)	84.63 (±0.07)	64.38 (±0.14)	71.69 (±0.11)	0.143	0.015
	SVM	72.51 (±0.08)	83.29 (±0.09)	62.05 (±0.15)	69.59 (±0.11)	0.020	0.010
	RF	79.58 (±0.07)	82.12 (±0.06)	78.99 (±0.09)	80.22 (±0.07)	0.019	0.001
	MLP	54.27 (±0.09)	60.60 (±0.13)	38.27 (±0.24)	43.00 (±0.20)	0.111	0.001
pan	Vot.	59.66 (±0.11)	72.57 (±0.17)	50.42 (±0.26)	54.66 (±0.16)	0.076	0.004
	SVM	49.43 (±0.06)	70.50 (±0.38)	28.64 (±0.38)	26.49 (±0.25)	0.002	0.001
	RF	70.02 (±0.13)	74.28 (±0.14)	68.76 (±0.14)	71.05 (±0.13)	0.011	0.001
	MLP	53.97 (±0.08)	59.98 (±0.10)	53.98 (±0.21)	54.17 (±0.11)	0.053	0.001
orientation	Vot.	73.13 (±0.06)	85.68 (±0.09)	59.35 (±0.11)	69.34 (±0.08)	0.082	0.012
	SVM	61.29 (±0.06)	89.34 (±0.10)	29.20 (±0.11)	43.12 (±0.13)	0.013	0.007
	RF	80.45 (±0.06)	82.65 (±0.08)	80.28 (±0.08)	81.07 (±0.06)	0.017	0.001
	MLP	49.66 (±0.05)	51.69 (±0.09)	41.59 (±0.26)	42.33 (±0.17)	0.040	0.001
multitouch	Vot.	68.99 (±0.13)	81.30 (±0.24)	53.26 (±0.26)	61.20 (±0.24)	0.097	0.003
	SVM	59.04 (±0.14)	63.98 (±0.48)	29.10 (±0.31)	35.42 (±0.33)	0.001	0.001
	RF	75.94 (±0.14)	77.50 (±0.23)	72.16 (±0.26)	73.38 (±0.23)	0.011	0.001
	MLP	51.01 (±0.11)	49.57 (±0.20)	45.37 (±0.27)	45.44 (±0.22)	0.072	0.001
motion	Vot.	61.41 (±0.08)	68.67 (±0.08)	51.44 (±0.23)	55.63 (±0.15)	0.151	0.009
	SVM	54.06 (±0.05)	69.34 (±0.16)	33.96 (±0.35)	35.50 (±0.22)	0.011	0.007
	RF	69.40 (±0.06)	71.19 (±0.07)	70.72 (±0.10)	70.52 (±0.07)	0.017	0.001
	MLP	52.19 (±0.07)	54.42 (±0.07)	56.88 (±0.20)	54.05 (±0.11)	0.119	0.001
Medias		65.22	72.24	56.62	59.65	0.061	0.005

Tabla 7.1: Resultados medios con todas las características

pecialmente en los casos del algoritmo *MLP* y los eventos *motion* y *orientation*. Comparando los resultados de ambas tablas podemos afirmar que la técnica de selección de características aplicada en esta parte del problema ha sido satisfactoria.

En resumen, con respecto a los eventos de *swipe*, *multitouch* y *orientation* han demostrado un rendimiento sensiblemente superior al resto de eventos. Con respecto a los algoritmos todos han obtenido unos porcentajes elevados, pero el más destacado es *RF*. El algoritmo de votación ha obtenido un buen rendimiento, especialmente si consideramos que este tipo de algoritmos suelen trabajar mejor cuando tienen muchos estimadores, en nuestro caso el número de estimadores es tres, uno por cada algoritmo.

Ev.	Alg.	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Fit Time(s)	Score Time(s)
tap	Vot.	76.16 (±0.05)	77.70 (±0.06)	76.19 (±0.07)	76.69 (±0.05)	0.174	0.013
	SVM	74.11 (±0.06)	75.90 (±0.06)	73.60 (±0.08)	74.50 (±0.06)	0.016	0.008
	RF	77.23 (±0.05)	78.42 (±0.06)	77.97 (±0.06)	77.95 (±0.04)	0.017	0.002
	MLP	73.43 (±0.08)	74.54 (±0.08)	73.67 (±0.11)	73.90 (±0.09)	0.152	0.001
swipe	Vot.	78.71 (±0.06)	82.88 (±0.05)	75.74 (±0.09)	78.86 (±0.05)	0.085	0.022
	SVM	78.23 (±0.05)	83.18 (±0.04)	73.86 (±0.07)	78.06 (±0.05)	0.028	0.018
	RF	79.93 (±0.05)	82.45 (±0.04)	78.79 (±0.08)	80.41 (±0.05)	0.018	0.002
	MLP	69.09 (±0.14)	72.75 (±0.15)	61.27 (±0.28)	63.54 (±0.26)	0.047	0.001
pan	Vot.	68.26 (±0.13)	74.71 (±0.14)	62.97 (±0.22)	66.58 (±0.17)	0.051	0.004
	SVM	55.34 (±0.14)	68.10 (±0.31)	31.34 (±0.27)	38.44 (±0.24)	0.002	0.001
	RF	71.68 (±0.13)	75.67 (±0.13)	70.92 (±0.17)	72.40 (±0.13)	0.011	0.001
	MLP	58.69 (±0.13)	63.16 (±0.13)	60.17 (±0.27)	58.69 (±0.17)	0.035	0.001
orientation	Vot.	78.72 (±0.06)	83.49 (±0.09)	75.39 (±0.11)	78.48 (±0.07)	0.098	0.037
	SVM	73.19 (±0.07)	83.67 (±0.10)	62.87 (±0.15)	70.36 (±0.09)	0.053	0.033
	RF	82.43 (±0.06)	84.56 (±0.08)	82.14 (±0.08)	82.96 (±0.06)	0.019	0.002
	MLP	56.47 (±0.12)	58.81 (±0.11)	57.10 (±0.17)	57.08 (±0.13)	0.031	0.001
multitouch	Vot.	71.96 (±0.12)	82.25 (±0.23)	58.20 (±0.25)	65.59 (±0.23)	0.057	0.003
	SVM	59.65 (±0.14)	70.00 (±0.46)	26.34 (±0.25)	35.67 (±0.31)	0.001	0.001
	RF	78.35 (±0.10)	81.67 (±0.11)	80.20 (±0.17)	79.32 (±0.11)	0.011	0.001
	MLP	62.05 (±0.12)	67.46 (±0.13)	66.14 (±0.23)	63.55 (±0.16)	0.039	0.001
motion	Vot.	67.20 (±0.06)	69.20 (±0.07)	67.79 (±0.10)	68.10 (±0.06)	0.160	0.039
	SVM	62.51 (±0.06)	67.12 (±0.08)	59.33 (±0.17)	61.30 (±0.09)	0.053	0.034
	RF	68.15 (±0.08)	69.65 (±0.08)	70.64 (±0.10)	69.78 (±0.08)	0.020	0.002
	MLP	62.98 (±0.08)	65.29 (±0.09)	61.77 (±0.14)	62.90 (±0.11)	0.117	0.001
Medias		70.19	74.69	66.02	68.13	0.054	0.010

Tabla 7.2: Resultados medios con las características seleccionadas

7.2 Datos agrupados por usuarios

Después de realizar las pruebas con los eventos agrupados ya sabemos que algoritmos y eventos funcionan mejor. Conociendo estos datos podemos agrupar los eventos por usuario y comprobar su rendimiento [Tabla 7.3].

Los resultados obtenidos son aceptables para una primera aproximación, obteniendo unos valores medios del 70% en cada una de las columnas. Para estas pruebas se utilizó el algoritmo de votación, donde sus estimadores fueron los algoritmos entrenados en la sección anterior.

Para intentar optimizar el rendimiento de la votación, se decidió seleccionar los mejores algoritmos (*SVM* y *RF*). Esta combinación se realizó con todos los eventos pero a cada uno de

User	Accuracy	Precision	Recall	F1 Score
Usuario 1	68.20%	64.83%	81.54%	72.23%
Usuario 2	68.02%	65.08%	81.69%	72.44%
Usuario 3	70.93%	73.08%	65.60%	69.14%
Usuario 4	73.15%	81.20%	61.70%	70.12%
Usuario 5	65.54%	63.08%	78.84%	70.08%
Usuario 6	67.02%	68.50%	68.03%	68.27%
Usuario 7	72.76%	85.42%	58.16%	69.20%
Usuario 8	60.82%	56.72%	75.90%	64.92%
Usuario 9	74.32%	71.47%	78.25%	74.71%
Usuario 10	70.30%	73.91%	64.59%	68.94%
Usuario 11	73.00%	69.19%	81.53%	74.85%
Usuario 12	68.88%	63.28%	80.87%	71.00%
Usuario 13	69.48%	65.76%	81.50%	72.79%
Usuario 14	82.30%	79.55%	86.99%	83.11%
Usuario 15	71.86%	68.81%	83.11%	75.29%
Usuario 16	66.17%	79.09%	44.84%	57.23%
Usuario 17	70.05%	68.81%	75.04%	71.79%
Usuario 18	68.41%	63.38%	89.37%	74.16%
Medias	70,07%	70,06%	74,31%	71,13%

Tabla 7.3: Resultados de los eventos agrupados por usuario

ellos se le asignó un peso. Este peso indica al algoritmo de votación que estimadores internos debe tener más en cuenta. Para calcular estos pesos, se utilizaron los valores obtenidos en la Tabla 7.2. La Tabla 7.4 muestra los resultados obtenidos de esta combinación.

Los datos muestran un mejora significativa en *precision* a cambio de reducir en *recall*, pero en el global (*accuracy*) también se ha mejorado. Esto se debe a que algunos eventos obtienen un porcentaje de acierto más bajo y esto afecta negativamente al conjunto, incluso usando pesos para minimizar dicha impacto. Por este motivo, se ha planteado el uso de aquellos eventos para los que se obtiene un mejor rendimiento (*orientation*, *multitouch* y *swipe*) y, nuevamente, se aplicará un sistema de votación que ponderará su influencia en el resultado final, tal y como se puede observar en la Tabla 7.5

El rendimiento obtenido en última instancia mejora considerablemente a los anteriores, como fruto de un proceso de refinamiento tanto de las características y los eventos empleados, como de los modelos de aprendizaje máquina construidos.

User	Accuracy	Precision	Recall	F1 Score
Usuario 1	81.72%	85.26%	76.70%	80.75%
Usuario 2	77.40%	83.44%	69.61%	75.90%
Usuario 3	79.30%	90.27%	68.44%	77.86%
Usuario 4	81.05%	91.15%	70.33%	79.40%
Usuario 5	79.62%	83.53%	73.45%	78.17%
Usuario 6	72.97%	85.19%	58.97%	69.70%
Usuario 7	70.00%	98.45%	47.83%	64.71%
Usuario 8	66.25%	73.77%	54.22%	62.50%
Usuario 9	69.70%	81.48%	59.46%	68.75%
Usuario 10	82.84%	87.37%	78.30%	82.59%
Usuario 11	79.38%	84.29%	72.84%	78.15%
Usuario 12	71.23%	89.74%	47.95%	62.50%
Usuario 13	78.27%	84.67%	69.05%	76.07%
Usuario 14	84.94%	92.38%	77.60%	84.35%
Usuario 15	78.29%	84.67%	70.56%	76.97%
Usuario 16	77.15%	82.46%	70.36%	75.93%
Usuario 17	76.99%	81.41%	70.95%	75.82%
Usuario 18	76.85%	83.20%	65.82%	73.50%
Medias	71,54%	77,22%	61,98%	68,53%

Tabla 7.4: Resultados de los mejores algoritmos agrupados por usuario

7.3 Servicio de autenticación en línea

A lo largo de este capítulo hemos refinado progresivamente los algoritmos tratando de alcanzar unos resultados que fuesen satisfactorios. En esta última sección, se analiza la implementación de un servicio web que permite verificar la identidad de los usuarios a través de su perfil específico, elaborado mediante los modelos detallados anteriormente.

Para ello se selecciona aleatoriamente una muestra correspondiente a un usuario (denominado *Usuario 1*) y se realizan diferentes peticiones al servidor para simular su autenticación contra su propio perfil y tres perfiles diferentes seleccionados también de forma aleatoria (denominados *Usuario 2*, *Usuario 3* y *Usuario 4*). Tal y como se puede observar en la Tabla 7.6, el servicio de autenticación proporciona un mayor índice de coincidencia con el perfil de usuario legítimo que para perfiles de otros usuarios, cumpliéndose el propósito del sistema. Así mismo, los tiempos de respuesta obtenidos por el servidor son aceptables para su implantación en un entorno real, ya que se trata de un proceso en segundo plano que no afectaría a la interacción del usuario con el sistema.

User	Accuracy	Precision	Recall	F1 Score
Usuario 1	80.29%	81.18%	78.85%	80.00%
Usuario 2	78.11%	85.57%	68.78%	76.26%
Usuario 3	80.17%	90.91%	69.67%	78.89%
Usuario 4	81.82%	92.28%	70.92%	80.20%
Usuario 5	79.62%	81.09%	76.90%	78.94%
Usuario 6	72.97%	80.65%	64.10%	71.43%
Usuario 7	52.50%	62.50%	43.48%	51.28%
Usuario 8	63.12%	69.35%	51.81%	59.31%
Usuario 9	68.94%	80.00%	59.46%	68.22%
Usuario 10	84.07%	89.73%	78.30%	83.63%
Usuario 11	83.75%	89.86%	76.54%	82.67%
Usuario 12	71.92%	82.00%	56.16%	66.67%
Usuario 13	76.19%	80.99%	68.45%	74.19%
Usuario 14	82.85%	90.38%	75.20%	82.10%
Usuario 15	81.43%	86.16%	76.11%	80.83%
Usuario 16	76.07%	80.90%	69.76%	74.92%
Usuario 17	77.84%	84.83%	68.72%	75.93%
Usuario 18	76.85%	80.29%	69.62%	74.58%
Medias	76,03%	82,70%	67,94%	74,45%

Tabla 7.5: Resultados de los mejores eventos y algoritmos agrupados por usuario

Usuario	Predicción	Tiempo de ejecución (s)
Usuario 1	86.37%	5
Usuario 2	36.89%	6
Usuario 3	35.65%	5
Usuario 4	40.43%	6

Tabla 7.6: Resultados de la prueba online

Capítulo 8

Trabajo Futuro

En este capítulo se comentarán algunos de los posibles desarrollos que se podrían llevar a cabo en un futuro.

- **Refinar el proceso de extracción y selección de características**: Partiendo de la implementación actual, se propone explorar nuevas características en busca de una optimización de los resultados.
- **Integración con aplicaciones de terceros**: Parte del producto final del proyecto es una aplicación propia que recoge los eventos, abstraer este concepto en forma de librería permitiría integrarla en aplicaciones de terceros.
- **Actualización de los modelos**: Este tipo de sistemas suelen estar entrenados con un conjunto de datos iniciales que es persistente en el tiempo, pero los seres humanos pueden cambiar su comportamiento a lo largo de su vida. Por lo tanto, un aprendizaje que evolucionase con el usuario permitiría obtener una mayor tolerancia a este problema.
- **Explorar nuevas técnicas**: Las técnicas utilizadas a lo largo del proyecto son algunas de las múltiples posibilidades existentes dentro del marco de la *IA*, por lo que realizar pruebas con otros métodos para comparar resultados e intentar mejorar el rendimiento, sería una buena forma de darle continuidad al proyecto.

De todas las opciones de trabajo futuro comentadas, conseguir que el algoritmo pueda adaptarse a lo largo del tiempo al usuario, es la más interesante. Pues se conseguiría que el algoritmo pueda seguir reconociendo al usuario, en cualquier momento de su vida y ante cualquier circunstancia externa (enfermedad, estrés ...)

Capítulo 9

Conclusiones

En este último capítulo de la memoria se comentarán las conclusiones extraídas de la realización del proyecto, haciendo especial énfasis en los resultados obtenidos con las técnicas comentadas para intentar encontrar una solución al problema planteado al inicio del proyecto.

Para obtener los datos se creó una aplicación móvil que, en su desarrollo, planteaba varios desafíos asociados a la captura de lo eventos para su análisis. El estudio fue realizado en un entorno controlado, por lo tanto, la aplicación fue diseñada e implementada para el dispositivo adquirido, **Samsung Tab**, aunque su funcionamiento en otros dispositivos también fue comprobado.

Para guardar de manera persistente estos datos, se implementó un servidor que, mediante una **API REST** [Apéndice A] permite realizar operaciones de consulta y almacenamiento de los datos. Para securizar estas peticiones, se ha utilizado el protocolo *HTTPS* que garantiza el cifrado de la información.

Los datos obtenidos pertenecen a 18 usuarios con edades comprendidas entre los 24 y 70 años de los cuales un 30% son mujeres y el 70% restante hombres. De esta manera, las muestras utilizadas son suficientemente representativas para esta primera aproximación al problema.

Las técnicas de *IA* empleadas para el reconocimiento de patrones fueron el perceptrón multicapa, las máquinas de soporte vectorial y los bosques aleatorios (*Random Forest*). En una primera instancia y de manera individual los resultados obtenidos por estos algoritmos fueron aceptables. En una segunda fase se utilizó el algoritmo de votación para obtener una combinación de los métodos mencionados. Los resultados obtenidos con estas técnicas mejoraron notablemente los resultados sin una pérdida importante de rendimiento. Por último, combinando los algoritmos y seleccionando únicamente los eventos más significativos se consiguió una mejora de un 12% en los resultados.

Se implementó un servidor en línea que utilizó este método combinado y optimizado. En las pruebas realizadas, se enviaron peticiones de manera continua, las cuales contenían los eventos de un usuario y el nombre al que, en teoría, debían de comprobar. El comportamiento

de este mecanismo también demostró ser eficiente manteniendo la eficacia anteriormente descrita.

En conclusión, el estudio planteado en este proyecto demuestra que es factible el uso de un sistema de autenticación continuo comprobando la legitimidad del usuario.

Apéndices

Apéndice A

REST API

A.1 Ruta Autenticación

- **/auth/token**
  - **POST**
    - **Descripción**: Obtiene el token JWT
    - **Parámetros**

Posición	Nombre	Descripción	Requerido	Tipo A.1
body	JwtPayload	El par usuario contraseña	true	JwtPayload

    - **Respuestas**

Código	Descripción	Esquema A.1
201	Devuelve el token de autenticación	JwtPayloadResponse
401	Credenciales incorrectas	HttpException

A.2 Ruta Usuarios

- **/users/admin**
  - **PUT**
    - **Descripción**: Actualiza la información del usuario administrador.
    - **Parámetros**

Posición	Nombre	Descripción	Requerido	Tipo A.1
body	CreateUserDto	Datos del usuario nuevo	true	CreateUserDto

    - **Respuestas**

Código	Descripción	Esquema A.1
200	Devuelve el token de autenticación	JwtPayloadResponse
403	Excepción, no tiene permisos para realizar esta operación	HttpException

A.3 Ruta Eventos

- **/events**
  - **POST**
    - **Descripción**: Crea un nuevo evento
    - **Parámetros**

Posición	Nombre	Descripción	Requerido	Tipo A.1
query	type	Tipo de la coleción (detect | profile), por defecto: detect	false	string
body	email	Email del usuario	true	string
body	events	Objeto con los datos de los eventos	false	EventResponse

    - **Respuestas**

Código	Descripción	Esquema A.1
201	Petición realizada correctamente	-
400	No se ha encontrado el email	HttpException

- **/events/{email}**
  - **GET**
    - **Descripción** : Obtiene los eventos de un usuario concreto
    - **Parámetros**

Posición	Nombre	Descripción	Requerido	Tipo
path	email	Email para buscar los eventos	true	string
query	type	Tipo de datos a buscar (detect | profile), por defecto: detect	false	string

    - **Respuestas**

Código	Descripción	Esquema A.1
200	Devuelve los datos de los eventos del usuario	EventResponse

- **/events/{email}/groups**
  - **GET**
    - **Descripción**: Obtiene los eventos de un usuario concreto, agrupados por tipo.
    - **Parámetros**

Posición	Nombre	Descripción	Requerido	Tipo
path	email	Email para buscar los eventos	true	string
query	type	Tipo de datos a buscar (detect | profile), por defecto: detect	false	string

    - **Respuestas**

Código	Descripción	Esquema A.1
200	Devuelve los datos de los eventos del usuario agrupados por tipo	EventResponse

A.4 Definición de los esquemas

- **Definiciones**

Objeto	Nombre	Tipo	Requerido
JwtPayload	email	string	true
	password	string	true
JwtPayloadResponse	accessToken	string	true
HttpException	message	string	true
	statusCode	number	true
CreateUserDto	email	string	true
	password	string	true
EventResponse	pan	array	false
	motion	array	false
	orientation	array	false
	pinch	array	false
	press	array	false
	rotate	array	false
	swipe	array	false
	tap	array	false

Tabla A.1: Tabla de Definiciones

Apéndice B

Manual de usuario de la aplicación móvil

B.1 Acceso a la aplicación

Al iniciar la aplicación se muestra la ventana principal [Figura B.1] la cual contiene una breve descripción sobre su uso y configuración. Desde esta ventana se pueden realizar dos acciones:

- Acceder a la ventana de configuración
- Iniciar la prueba

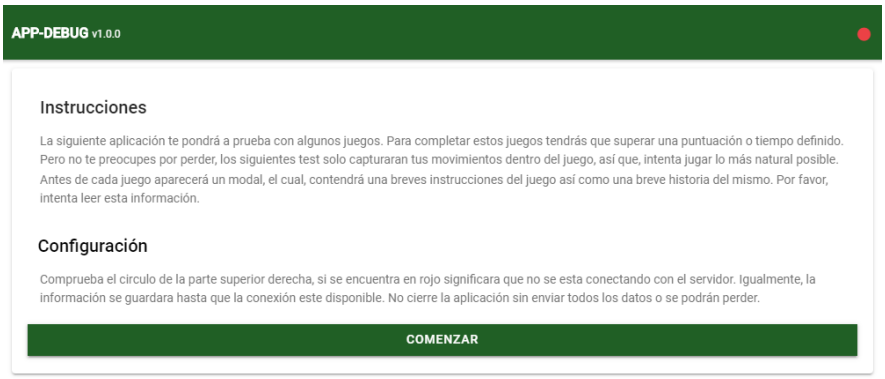

Figura B.1: Ventana principal

B.2 Configuración de la aplicación

Para acceder a la pantalla de configuración es necesario realizar un patrón, que consiste en tocar las cuatro esquinas de la pantalla. Al realizar el patrón se mostrará la configuración de la

aplicación [Figura B.2]. En esta ventana se puede configurar algunos de los aspectos de la aplicación como tiempo de la aplicación y la dirección del servidor. Además de la configuración también se muestran los datos actuales de la aplicación y un registro de eventos.

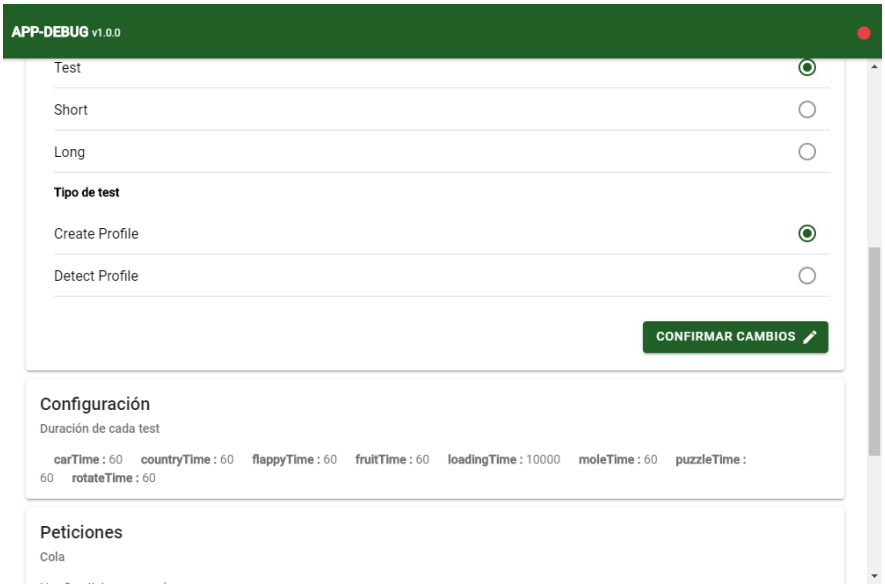

Figura B.2: Menú de configuración

B.3 Inicio de la prueba

Para iniciar la prueba solo es necesario presionar el botón **COMENZAR** y a continuación se abrirá un dialogo donde tendremos que introducir el nombre de usuario. [Figura B.3].

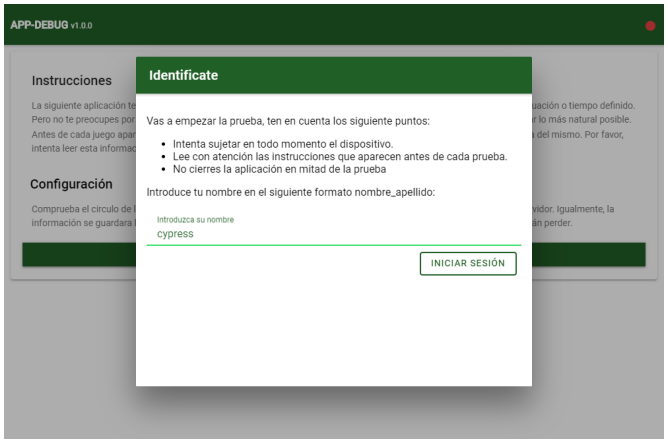

Figura B.3: Ventana de login

B.3.1 Desafíos

Puzzle Sliding

Esta primera prueba [Figura B.4] consiste en resolver el puzzle que se muestra en pantalla, realizando movimientos de deslizamiento, en un tiempo límite establecido.

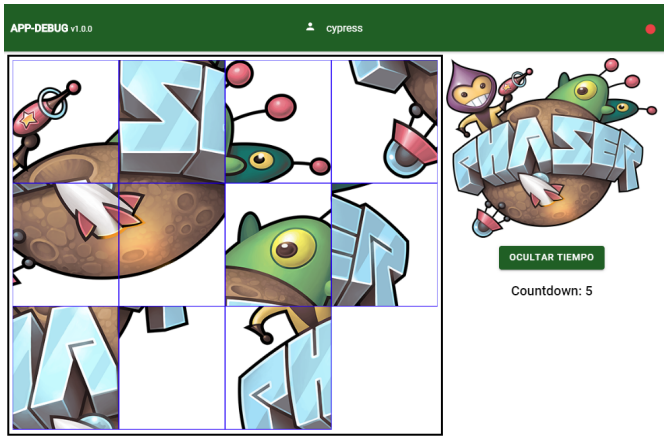

Figura B.4: Ventana del juego de *Puzzle Sliding*

Whack a mole

En esta prueba [Figura B.5] tendremos que tocar la pantalla sobre los animales, los cuales aparecerán cada cierto tiempo.

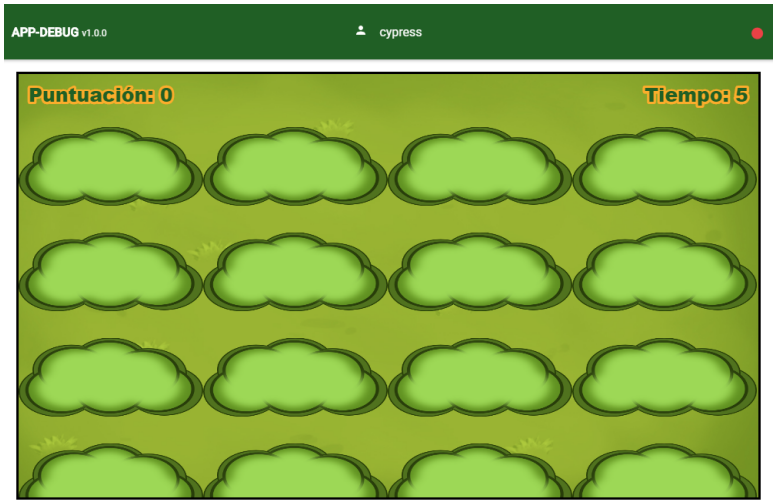

Figura B.5: Ventana del juego de *Whack a Mole*

Flappy Bird

En esta prueba [Figura B.6] tendremos tocar la pantalla repetidamente, para intentar que el pájaro no choque contra las tuberías que irán apareciendo.

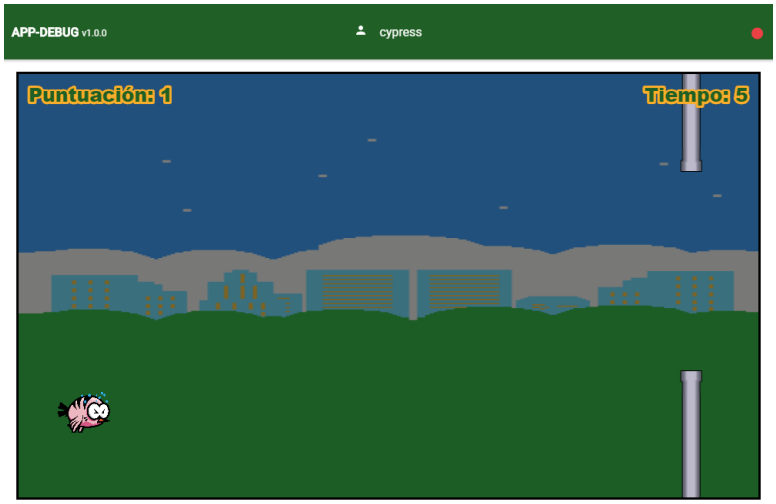

Figura B.6: Ventana del juego de *Flappy Bird*

Fruit Ninja

En esta prueba [Figura B.6] tendremos que delizar el dedo sobre la pantalla para cortar la fruta que vaya apareciendo.

Figura B.7: Ventana del juego de *Fruit Ninja*

Outrun

En esta prueba [Figura B.6] tendremos que coger el dispositivo como si fuera el volante e intentar esquivar los obstaculos que irán apareciendo.

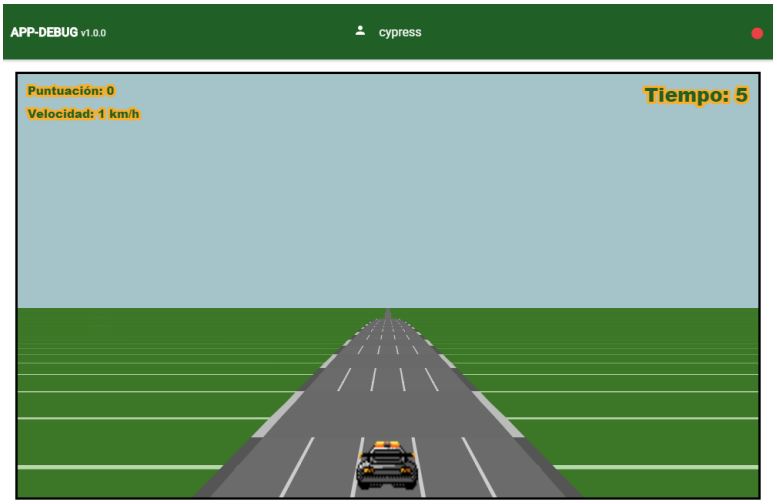

Figura B.8: Ventana del juego de *Outrun*

Buscar Países

En esta prueba [Figura B.9] se nos muestra una lista de países ordenados alfabéticamente. El objetivo es buscar los países que se muestran en la parte superior izquierda.

Figura B.9: Ventana del juego de *Buscar Países*

Rotación

En esta prueba [Figura B.10] tendremos que hacer encajar las piezas y para ello tendremos que mover, rotar y escalar la pieza negra.

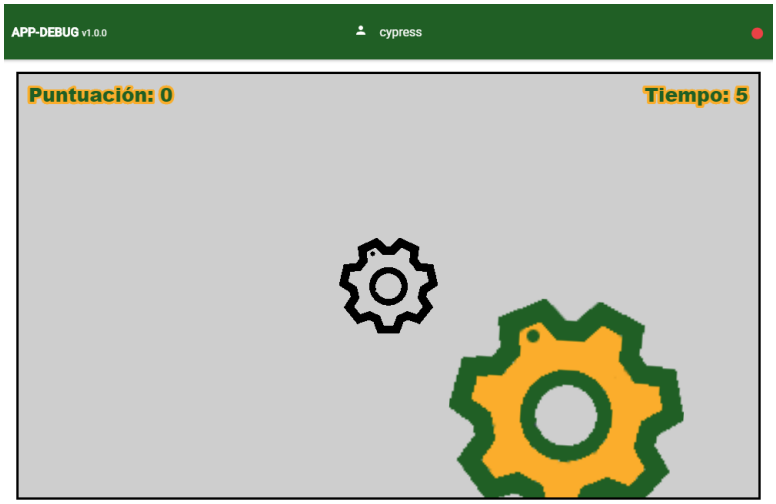

Figura B.10: Ventana del juego de *Rotación*

B.3.2 Fin de la prueba

Una vez finalizadas todas la pruebas la aplicación nos mostrara una tabla de puntuaciones [Figura B.11] de todos los desafíos hechos. En esta ventana podremos volver al inicio pulsando el botón *IR A INICIO*.

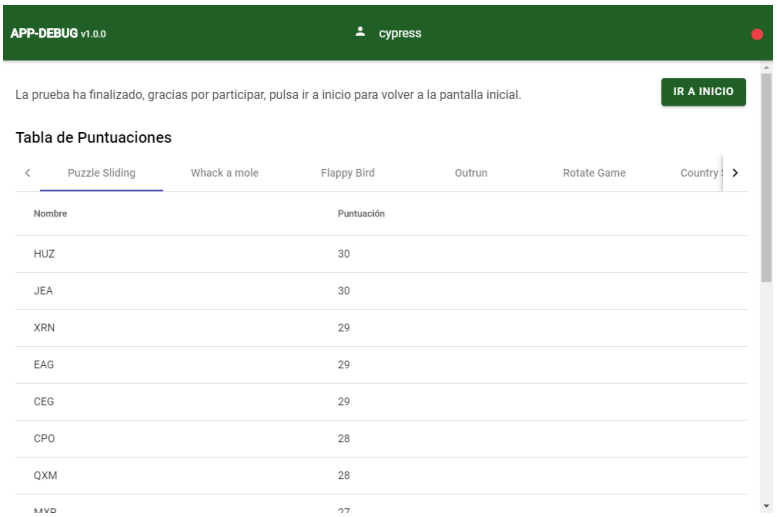

Figura B.11: Ventana de puntuaciones

Bibliografía

[1] eurostat, "Do you protect the personal data on your smartphone?" [En linea]. Disponible en: https://ec.europa.eu/eurostat/en/web/products-eurostat-news/-/EDN-20190128-1

[2] INCIBE, "Amenaza vs vulnerabilidad, ¿sabes en qué se diferencian?" [En linea]. Disponible en: https://www.incibe.es/protege-tu-empresa/blog/amenaza-vs-vulnerabilidad-sabes-se-diferencian

[3] Rediris, "Gestión de la seguridad." [En linea]. Disponible en: https://www.rediris.es/cert/doc/unixsec/node31.html

[4] ——, "Introducción y conceptos previos." [En linea]. Disponible en: https://www.rediris.es/cert/doc/unixsec/node5.html

[5] "Autenticación de usuarios." [En linea]. Disponible en: https://www.rediris.es/cert/doc/unixsec/node14.html

[6] "Juan vucetich." [En linea]. Disponible en: https://www.ecured.cu/Juan_Vucetich

[7] "An industrial-strength audio search algorithm."

[8] J. C. D. V. Daniel Garabato Míguez, Francisco Javier Nóvoa de Manuel, "Sistema de autenticación basado en patrones de escritura por teclado 2014," 2014.

[9] J. C. D. V. Jorge Rodríguez García, Daniel Garabato Miguez, "Sistema de autenticación biométrico basado en el análisis del comportamiento utilizando mecanismos de entrada convencionales." 2019.

[10] A. S. Kenneth Rohde Christiansen, "Motion sensors explainer." [En linea]. Disponible en: https://www.w3.org/TR/motion-sensors

[11] A. Shalamov, "Accelerometer." [En linea]. Disponible en: https://www.w3.org/TR/accelerometer/

[12] M. Pozdnyakov, "Gyroscope." [En linea]. Disponible en: https://www.w3.org/TR/gyroscope/

[13] M. Ehatisham-ul Haq, M. A. Azam, U. Naeem, Y. Amin, and J. Loo, "Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing," *Journal of Network and Computer Applications*, vol. 109, pp. 24–35, 2018.

[14] M. B. A. B. Doug Schepers, Sangwhan Moon, "Touch events." [En linea]. Disponible en: https://www.w3.org/TR/touch-events/

[15] J. McCarthy, "What is artificial intelligence?" [En linea]. Disponible en: http://jmc.stanford.edu/articles/whatisai/whatisai.pdf

[16] J. F. Puget, "What is machine learning?" [En linea]. Disponible en: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Is_Machine_Learning?lang=en

[17] R. Salas, "Redes neuronales artificiales," *Universidad de Valparaıso. Departamento de Computación*, vol. 1, 2004.

[18] F. Rosenblatt, "Perceptron simulation experiments," *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, 1960.

[19] M. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[20] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[21] "Neural network." [En linea]. Disponible en: https://playground.tensorflow.org/

[22] R. Berwick, "An idiot's guide to support vector machines (svms)."

[23] J. A. Rodrigo, "Máquinas de Vector Soporte (Support Vector Machines, SVMs)," https://rpubs.com/Joaquin_AR/267926.

[24] "Boosting and bagging: How to develop a robust machine learning algorithm." [En linea]. Disponible en: https://hackernoon.com/how-to-develop-a-robust-algorithm-c38e08f32201

[25] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[26] "Ensemble learning — bagging and boosting." [En linea]. Disponible en: https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e

[27] L. Breiman, "Random forests," *Machine Learning*, 2001. [En linea]. Disponible en: https://doi.org/10.1023/A:1010933404324

[28] S. Raschka, "EnsembleVoteClassifier," http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/.

[29] D. Ruta and B. Gabrys, "Classifier selection for majority voting," 2005. [En linea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S1566253504000417

[30] M. E. ul Haq, M. A. Azam, U. Naeem, Y. Amin, and J. Loo, "Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing," *Journal of Network and Computer Applications*, vol. 109, pp. 24 – 35, 2018. [En linea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S1084804518300717

[31] A. Jain and V. Kanhangad, "Gender recognition in smartphones using touchscreen gestures," *Pattern Recognition Letters*, vol. 125, pp. 604 – 611, 2019. [En linea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0167865519301758

[32] D. Damopoulos and G. Kambourakis, "Hands-free one-time and continuous authentication using glass wearable devices," *Journal of Information Security and Applications*, vol. 46, pp. 138 – 150, 2019. [En linea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S2214212618305714

[33] "Continuous face authentication scheme for mobile devices with tracking and liveness detection," *Microprocessors and Microsystems*, vol. 63, pp. 147 – 157, 2018. [En linea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0141933117305239

[34] Y. Yang, B. Guo, Z. Wang, M. Li, Z. Yu, and X. Zhou, "Behavesense: Continuous authentication for security-sensitive mobile apps using behavioral biometrics," *Ad Hoc Networks*, vol. 84, pp. 9 – 18, 2019. [En linea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S1570870518306899

[35] SealSign, sitio web oficial. [En linea]. Disponible en: https://www.elevenpaths.com/es/tecnologia/sealsign/index.html

[36] Biosig-id, sitio web oficial. [En linea]. Disponible en: https://biosig-id.com

[37] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," in *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 987–996.

[38] M. de empleo y seguridad social, "Boe." [En linea]. Disponible en: https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf

[39] Angular, sitio web oficial. [En linea]. Disponible en: https://angular.io/

[40] Ionic, sitio web oficial. [En linea]. Disponible en: https://ionicframework.com

[41] Capacitor, sitio web oficial. [En linea]. Disponible en: https://capacitor.ionicframework.com/

[42] Cypress, sitio web oficial. [En linea]. Disponible en: https://www.cypress.io/

[43] Nest JS, sitio web oficial. [En linea]. Disponible en: https://nestjs.com/

[44] Jest, sitio web oficial. [En linea]. Disponible en: https://jestjs.io/

[45] Scikit-Learn, sitio web oficial. [En linea]. Disponible en: https://scikit-learn.org/stable/

[46] MongoDB, sitio web oficial. [En linea]. Disponible en: https://www.mongodb.com/

[47] Git, sitio web oficial. [En linea]. Disponible en: http://git-scm.com

[48] VS Code, sitio web oficial. [En linea]. Disponible en: https://code.visualstudio.com/

[49] Android Studio, sitio web oficial. [En linea]. Disponible en: https://developer.android.com/

[50] ECMAScript, sitio web oficial. [En linea]. Disponible en: https://www.ecma-international.org/publications/standards/Ecma-262.htm

[51] Typescript, sitio web oficial. [En linea]. Disponible en: https://www.typescriptlang.org/

[52] Python, sitio web oficial. [En linea]. Disponible en: https://www.python.org/

[53] LaTeX, sitio web oficial. [En linea]. Disponible en: https://www.latex-project.org/

[54] W. C. Hetzel and B. Hetzel, *The complete guide to software testing.* QED Information Sciences Wellesley, MA, 1988.

[55] M. Cohn, *Succeeding with agile: software development using Scrum.* Pearson Education, 2010.

[56] "Vídeo pruebas e2e." [En linea]. Disponible en: https://youtu.be/8Np7hrq6kpQ

[57] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.