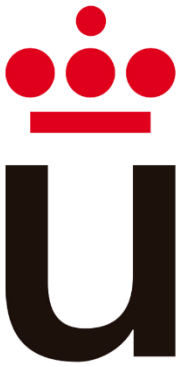

Universidad
Rey Juan Carlos

Escuela Técnica Superior
Ingeniería Informática

GRADO EN INGENIERÍA DEL SOFTWARE

Curso Académico 2022/2023

Trabajo Fin de Grado

¿QUIÉN ES QUIÉN?

Aplicación web para la identificación de usuarios mediante técnicas de análisis de comportamientos

Autor: Adrián López Couso.
Director: Alejandro García de Marina
Isaac Martín de Diego

Resumen

En los últimos años, la sociedad ha avanzado considerablemente hacia un mundo más tecnológico e interconectado. La dependencia de la tecnología es cada vez mayor y la mayoría de las tareas que realiza el ser humano en su día a día requieren el uso de algún aparato tecnológico. Por esta razón, los dispositivos deben garantizar seguridad a los usuarios. No solo debemos centrarnos en la seguridad de los dispositivos, sino también en asegurar las páginas web, aplicaciones móviles, aplicaciones web, etc.

Este proyecto consiste en crear una aplicación web que utilice un algoritmo de aprendizaje automático (más conocido por su término en inglés como *machine learning*) capaz de poder identificar al usuario que está escribiendo por teclado mediante un análisis de su comportamiento. El objetivo de la aplicación web es implementar este nuevo método de autenticación utilizando *machine learning*.

La aplicación web permite al usuario registrado efectuar dos tipos de pruebas. En ambas pruebas, se mostrará un texto que el usuario deberá escribir. En la primera prueba, el algoritmo de *machine learning* creará un modelo utilizando la cadencia de tecleo del usuario. Cuantas más muestras existan, más preciso será el modelo. En la segunda prueba, los datos obtenidos se compararán con el modelo creado por el algoritmo. Si la mayoría de los datos de la cadencia de tecleo de la segunda prueba se encuentran dentro del modelo del usuario autenticado (cada modelo está relacionado con un usuario), la prueba concluye con un mensaje indicando que el usuario que ha realizado la segunda prueba corresponde al usuario autenticado. En caso de que la mayoría de los datos no se encuentren dentro del modelo, se muestra un mensaje indicando que el usuario que ha realizado la segunda prueba no es el usuario con la sesión iniciada.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

I. Introducción

a. Información general

"La seguridad humana es una condición o estado caracterizado por la libertad ante amenazas dominantes sobre los derechos de las personas, sobre su tranquilidad e, incluso, sobre sus vidas" [1]. Esta sensación siempre será buscada por el ser humano, especialmente en la actualidad cuando navegamos por internet.

Gracias a que la tecnología está presente en nuestra vida cotidiana, ha crecido el interés de las personas por aprender más sobre internet y las nuevas tecnologías. Esto ha derivado en que la mayoría de la población acabe poseyendo al menos un dispositivo tecnológico. De la misma manera que internet nos abre un abanico de nuevas posibilidades, también nos expone a numerosos peligros, como páginas falsas, *malware*[1], robos de identidad, entre otros. Si bien estas amenazas siempre existirán, a lo largo de los años se han implementado dispositivos más seguros, aplicaciones y páginas web con métodos y herramientas que nos ayudan a proteger nuestros datos, entre otros aspectos.

Llamamos web (*World Wide Web*) a la colección de páginas que se asientan en internet. En la actualidad, la web que utilizamos toma el nombre de web 2.0 y está compuesta por páginas web donde los usuarios podemos interactuar con ellas. Se basa en una arquitectura centralizada donde los usuarios dependemos de una compañía o punto central.

Para entender este concepto, podemos tomar como ejemplo el uso de un servicio como WhatsApp[2]. A veces, nos encontramos con la incapacidad de utilizarlo debido a que la aplicación ha experimentado una interrupción. Desde un punto de vista tecnológico, lo que realmente sucede es que los servidores de la compañía han presentado algún error y han dejado de estar operativos, lo que impide que nos devuelvan la respuesta.

El auge de la web 2.0 ha popularizado el término "identidad digital". La identidad digital pertenece al área de la cibercultura[3] y se va formando cada vez que tenemos algún tipo de actividad en la red. Desde los últimos años, la cantidad de datos personales que existen en internet va incrementándose, lo que expone nuestra identidad humana. Además, el principal problema reside en que los propietarios de estos datos personales no somos nosotros, sino las grandes empresas. Por ello, es importante evolucionar hacia la web 3.0, donde los propietarios de esos datos volveremos a ser nosotros, brindándonos el control total de nuestra identidad.

Una persona puede tener varias identidades y cada una de ellas puede estar ligada a una parte de nuestra identidad humana, aun así, estas identidades pertenecen a una

[1] Programa malicioso que realiza acciones dañinas en un sistema informático de forma intencionada y sin el conocimiento del usuario.

[2] Aplicación de mensajería instantánea

[3] Cultura asociada al mundo de las redes informáticas y a la realidad virtual.

persona y esto se puede comprobar en la red. “Como puede apreciarse, la construcción de esta identidad digital distingue entre la información que se revela expresamente por la persona, la identidad que es revelada por las acciones que esta realiza y la que es calculada o inferida según el análisis de las acciones que la persona lleva a cabo” [5].

Lo ideal para enriquecer nuestra vida en la red y en el mundo real es gestionar u homogenizar correctamente nuestra identidad digital con nuestra identidad humana. Sin embargo, no todo el mundo es capaz de hacerlo, por eso es fundamental promover cursos y formaciones que nos ayuden a gestionar correctamente nuestra identidad digital y desenvolvernos en un mundo digital cada vez más interconectado.

Pero ¿cómo se gestiona eficazmente la identidad digital? Lo primero que debemos entender para gestionar correctamente nuestra identidad digital es conocer las oportunidades y peligros que nos ofrece internet. La red nos proporciona un gran abanico de soluciones y oportunidades, como solucionar un problema cotidiano, encontrar trabajo, conocer gente, facilidades para estudiar, entre otras cosas. Por otro lado, puede acarrear algunos problemas, ya que toda interacción que tengamos en la red queda registrada (comentarios en un foro, búsquedas, fotos, videos, ...) y podemos sufrir una suplantación de identidad. Esto último es posiblemente lo más peligroso que te puede ocurrir, ya que cualquier interacción que tenga esa identidad falsa se ligará a tu identidad humana.

Mucha gente decide no tener una identidad digital, aunque esto sea una opción respetable y personal, también conlleva ciertos riesgos. Es más fácil suplantar esta identidad en la red debido a la ausencia de otra identidad digital con la cual contrastarla (la verdadera) [6].

Por otro lado, la web 3.0 [7] propone una arquitectura descentralizada suprimiendo el punto central que controla toda la red, permitiendo a los nodos participantes (usuarios) gestionar los datos. De esta manera, si uno de estos nodos fallara, la red seguiría operando. Esto significa que la red sería capaz de autogestionarse sin necesidad de una unidad que lo controle todo, de esta forma los participantes operarían de igual a igual o también conocido como *peer-to-peer* (P2P).

En esta nueva web, la inteligencia artificial (IA) tomará un papel muy importante, a diferencia de en la web 2.0 donde se usaba exclusivamente para la publicidad. En la web 3.0, la inteligencia artificial funcionará para que el usuario pueda personalizar al máximo su experiencia de navegación. Por la misma razón por la que la web evoluciona, también debemos evolucionar la forma de proteger nuestra identidad digital y los métodos de autenticación disponibles, pasando de simples contraseñas, a métodos más seguros y complejos como FIDO (*Fast Identity Online*).

Recientemente, las empresas más grandes de tecnología del mundo [2] (Apple, Google y Microsoft) están buscando eliminar el uso de contraseñas. De esta forma, se pretende evitar que el usuario deba recordar cada una de ellas o mantenerlas guardadas

en sus dispositivos o navegadores web. De cualquier manera, ambos métodos presentan riesgos para el usuario [3]. Muchas personas creen que mantener sus contraseñas guardadas en su navegador es seguro, pero se equivocan, ya que cualquier persona que pueda acceder a nuestro dispositivo también podrá acceder a nuestras contraseñas.

La propuesta de estas empresas es implementar el sistema anteriormente nombrado FIDO, el cual trabajaría como un gestor de nuestras contraseñas. Su funcionamiento se basa en la creación de una pareja de claves: una clave privada, inmodificable y almacenada en el dispositivo, y otra clave pública almacenada en los servicios *online* [4]. La decisión de que las empresas más grandes en el ámbito tecnológico estén apostando por una tecnología que suprima por completo las contraseñas nos hace reflexionar en lo vulnerables que son en la actualidad y la necesidad de buscar otros métodos de autenticación más seguros y fáciles de usar.

Las contraseñas ya no son tan seguras como antes, al menos así lo ven las grandes empresas tecnológicas. En el presente trabajo, se propone un nuevo método que busca una forma de mejorar la seguridad de nuestra identidad digital. Este método está basado en algoritmos de aprendizaje automático. Además, se ha desarrollado una aplicación web para poner en práctica este método y analizar su eficacia.

b. Objetivos

Se han definido una serie de objetivos que nos han ayudado a marcar unas pautas en el presente trabajo. Los objetivos principales del proyecto son:

- Desarrollo de un método de autenticación basado en *machine learning* como alternativa a las contraseñas. Además, se pretende observar el impacto que puede llegar a tener este método poco convencional en usuarios que están acostumbrados a usar contraseñas para autenticarse.

- Desarrollo de una aplicación web en Django. La finalidad del proyecto se centra en la creación de una aplicación web que utilice un sistema de autenticación basado en la manera de escribir del usuario.

- Integración del método de autenticación en la aplicación web.

- Validar la eficacia del método para comprobar como de seguro es.

Aparte de los objetivos expuestos anteriormente, se han establecido objetivos secundarios:

- Desarrollo de un *dashboard* con una interfaz intuitiva y fácil de navegar para usuarios convencionales.

- Incrementar la visibilidad de un método de autenticación basado en *machine learning*.

c. Metodología de trabajo

En cuanto a la metodología de trabajo, para el desarrollo y diseño del software se han contemplado cinco etapas [9]:

- Génesis de la idea: Siempre se parte de una idea principal, la cual irá evolucionando, añadiendo o suprimiendo elementos. La idea inicial de un software contiene la semilla del **QUÉ** (materia y nivel) y **CÓMO** (estrategia que se va a llevar a cabo).

- Prediseño o diseño funcional: A partir de la idea principal se crea un diseño donde se especificarán los objetivos, contenidos, etc. Se han establecido tres fases. En la primera fase, se desarrolló una aplicación con diversas plantillas donde se podría navegar entre ellas, pero no disponía de ningún elemento funcional. En la segunda fase, se añadieron las funcionalidades que permiten al usuario iniciar sesión, realizar las pruebas, entre otras. Por último, en la tercera fase, se "dockerizó" la aplicación para que pueda ser ejecutada por cualquier usuario.

- Programación de un prototipo: A medida que se iba avanzando en el proyecto, se fueron creando prototipos funcionales que, aunque estuviesen lejos de su resultado final, nos proporcionaba una visión de este.

- Redacción de la documentación: Además de contar con esta memoria, el proyecto cuenta con una documentación en la plataforma de GitHub junto a instrucciones, imágenes y videos.

- Publicación y mantenimiento del producto: Aunque se haya publicado una versión funcional y *a priori* final del producto, la vida del software no termina ahí. Hasta el día de su "muerte", estará en constante desarrollo y mantenimiento, ofreciendo así a los usuarios una versión más optimizada y mejor que la anterior.

Sin embargo, hay que aclarar que el proceso del software no es lineal, sino iterativo, es decir, el software está en constantes pruebas, comprobando así su funcionalidad, su resultado e incluso evaluando el producto en general.

Por último, se han incorporado herramientas y metodologías *agile* que han ayudado al desarrollo de esta metodología de trabajo. Se ha decidido usar Scrum para estructurar y gestionar la carga de trabajo. Una vez definido la metodología ágil que se iba a utilizar, el proyecto ha sido desglosado en historias de usuario y usando la herramienta

Trello, se ha creado un *Product Backlog*, que ha servido de ayuda a la hora de organizar el proyecto y realizar los *Sprints*.

Se ha desarrollado un mando interactivo o *dashboard* en Django que nos ayudará a mostrar al usuario las distintas opciones de nuestra aplicación, así como sus datos. En el proceso de desarrollo de la aplicación se han ido aprendiendo y mejorando aspectos relacionados con la carrera de ingeniería de software y con el mundo de la informática en general.

- Metodologías y herramientas de trabajo para el desarrollo de un proyecto software.

- Desarrollo de una aplicación web usando por primera vez el *framework* Django[4] y ampliando conocimientos en lenguajes de programación ya utilizados (Python[5], JavaScript[6], HTML[7], CSS[8] y Bootstrap[9])

[4] Framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador.

[5] Lenguaje de alto nivel de programación interpretado

[6] Lenguaje de programación orientado a objetos.

[7] Lenguaje de marcado para la elaboración de páginas web

[8] Lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

[9] Conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.

II. Estado del arte

En este apartado se presentarán los pilares fundamentales para el desarrollo de la aplicación: poner en contexto sobre la disciplina escogida (*machine learning*) y las tecnologías utilizadas para desarrollarla.

a. ¿Qué es el machine learning?

"El *machine learning* es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuya finalidad es desarrollar técnicas que permitan a las computadoras aprender, convirtiéndose en un pilar fundamental para el trato de datos a gran escala." [10]. Como bien se comenta, el *machine learning* es una rama de la inteligencia artificial (IA), por lo que deberemos remontarnos al inicio para entender mejor esta disciplina.

En 1943, el neurólogo Warren McCulloch y el lógico Walter Pitts publicaron un trabajo donde se proponía estudiar un cerebro como un modelo computacional, este fue el inicio de lo que se conoce como red neuronal artificial. Hoy se considera un pilar fundamental de la inteligencia artificial.

El interés por explorar como de inteligente podría llegar a ser una máquina desembocó en otro hito esencial de la inteligencia artificial. En 1950, Alan Mathison Turing publicó un artículo titulado "*Computing Machenery and Intellegence*", donde plantea el famoso "Test de Turing". Esta prueba se basa en la habilidad de una máquina para mostrar un comportamiento humano, donde un interrogador deberá entablar una conversación mediante texto con otro humano y la máquina. La premisa principal de esta prueba es que el evaluador sabe que uno de los miembros de la conversación es una máquina. En caso de no poder diferenciar entre la máquina y el humano, la máquina habrá pasado el "Test de Turing" [11].

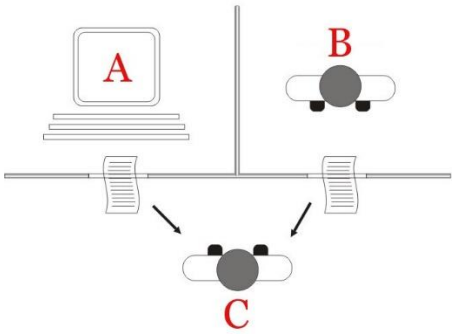

Ilustración 1: Test de Turing

En 1979, los estudiantes de la Universidad de Stanford diseñaron un robot capaz de moverse por una habitación sin colisionar con ningún obstáculo. Todo esto se logró gracias a un algoritmo llamado "*Nearest Neighbor*" que conseguía reconocer patrones y aprenderlos para formular una solución y anticiparse.

En la actualidad, se ha desarrollado nuevas técnicas de *machine learning* y se han mejorado las ya existentes, esto se debe a la gran cantidad de datos que se manejan y al avance tecnológico a lo largo de los años.

Las técnicas nombradas anteriormente se pueden clasificar en tres tipos de algoritmos según el paradigma que se esté utilizando [10]:

- Aprendizaje supervisado: Se basa en clasificar una serie de datos para poder detectar patrones, es decir, encontrar una relación entre unos datos de entrada y unos datos de salida, con el fin de poder encontrar esa correlación en nuevos datos.

- Aprendizaje no supervisado: Los datos no se clasificarán previamente, por lo que se deberán analizar semejanzas y disconformidades para permitir diferenciar los datos.

- Aprendizaje de refuerzo: Al igual que en el aprendizaje no supervisado, los datos no estarán clasificados. La principal diferencia reside en que el sistema realizará varias acciones y se irá retroalimentando mediante actualizaciones.

Para poder clasificar los datos en el presente trabajo, se ha optado por la clasificación "*One Class*". En la clasificación "*One Class*", se define una clase *target* u objetivo y la clase *outlier*. La clase *target* se entrena con muestras del modelo, mientras que la clase *outlier* tiene pocas o ninguna muestra [18]. Una vez el algoritmo haya creado el modelo en base a las muestras, clasificara los datos en base a si pertenecen a la clase *target* o no. Es decir, no distingue entre dos clases, sino si la instancia pertenece a la clase *target* o no. Como se puede observar en la ilustración 2, en nuestro caso la clase *target* estará representada por los usuarios autenticados (positivo o 1), mientras que todo aquello que se encuentre fuera de esta clase (en la clase *outlier*), se clasificará como usuario "intruso" (negativo o -1).

Ilustración 2: Clasificación One Class

En base a la clasificación "*One Class*", el método utilizado en el trabajo se basa en el algoritmo de máquinas de vectores (SVM, por sus siglas en inglés *Support Vector Machine*).

Aunque las SVMs abarcan diversos campos, en este caso nos vamos a centrar en la clasificación y predicción de la cadencia de tecleo del usuario. Se llevarán a cabo tres fases:

- Clasificación de los datos recogidos.

- Aprendizaje de los datos y creación de un modelo.

- Predicción del usuario usando como referencia el modelo anteriormente creado

El algoritmo *Support Vector Machine* o máquina de vector soporte que utilizamos en el presente proyecto, pertenece al grupo de aprendizaje supervisado, ya que nos permitirá clasificar entre dos clases distintas según los datos de entrada. SVM mapea los datos de entrada a un espacio de características mayor (si los datos se encuentran en una dimensión d^2, serán mapeados por la SVM a d^3) con el objetivo de encontrar un hiperplano que los clasifique [12].

Existen infinitos hiperplanos posibles, pero nos interesa encontrar el más optimo. El hiperplano optimo será aquel que maximiza el margen entre clases usando vectores de soporte (datos de entrada más cercanos o alejados al hiperplano). Para logarlo, nos apoyamos en las funciones *kernels*[10].

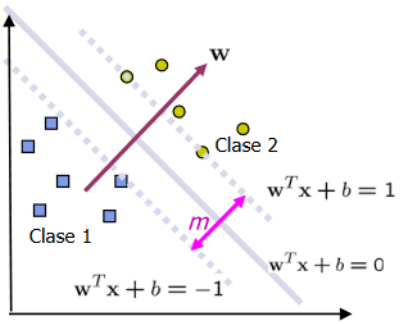

Ilustración 3: Calculo del hiperplano óptimo [12]

Existen casos donde encontraremos datos de clases diferentes entremezclados que nos impedirán encontrar fácilmente un hiperplano. El principal problema reside en que corremos el riesgo de clasificar erróneamente los datos y nunca se va a encontrar un hiperplano optimo. Estos casos se denominan "casos no linealmente separables".

Para poder clasificar correctamente los datos, es necesario utilizar la constante *C*. "El parámetro C puede ser definido como un parámetro de regularización. Este es el único parámetro libre de ser ajustado en la formulación de la SVM. El ajuste de este parámetro puede hacer un balance entre la maximización del margen y la violación a la clasificación" [12].

[10] "El kernel es un administrador de procesos que crea el entorno necesario para que los procesos puedan ejecutarse." [17]

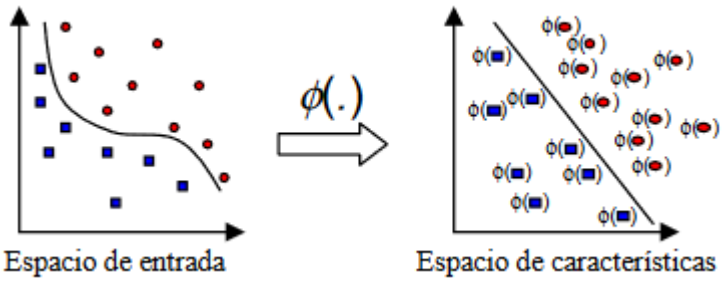

Ilustración 4: "Idea del uso de un kernel para transformación del espacio de los datos" [12]

También, podemos aplicar un concepto más complejo llamado "Truco del *Kernel*". En esta propiedad no es necesario conocer φ, simplemente necesitaremos una función *K* o función *Kernel*. "Son *kernels* todas aquellas funciones *K* (u, v) que verifican el teorema de Mercer" [13], es decir, para las cuales:

$$\text{“}\int_{u,v} K(u, v)\, g(u)\, g(v)\, du dv > 0\text{”}.$$

Ecuación 1: Teorema de Mercer [13]

Las funciones *Kernel* más comunes son:

$$\text{Lineal: } K(x_i,\ x_j) = x_i^T x_j$$

Ecuación 2: Función Kernel Lineal [13]

$$\text{Gaussiana: } K(x_i,\ x_j) = \exp\left(\frac{\left|\left|x_i - x_j\right|\right|^2}{2\sigma^2}\right)$$

Ecuación 3: Función Kernel Gaussiana [13]

$$\text{Polinómica: } K(x_i,\ x_j) = (x_i^T x_j\ +\ 1)^n$$

Ecuación 4: Función Kernel Polinómica [13]

b. Tecnologías utilizadas

La aplicación ha sido desarrollada en el *framework* Django, juntando herramientas y lenguajes de web como HTML, JavaScript, CSS y Bootstrap con Python para el desarrollo de aplicaciones web. Django es un entorno de trabajo que usa un *Backend*[11] compuesto por Python, y un *Frontend*[12] compuesto por HTML, JavaScript y CSS. Además, nos proporciona herramientas para configurar aspectos de nuestra aplicación web, ya sean sus urls, sus aplicaciones, los documentos estáticos que utilizará la aplicación web y su base de datos. En cuanto a las bases de datos, se ha utilizado PostgreSQL[13] como sistema de gestión de bases de datos y PgAdmin4 para su futura gestión y administración. En este último caso, se quiere destacar la importancia de ser capaces de implementar una base de

[11] Área lógica del software, en este caso de la página web
[12] Área que ve el usuario y con la que interactúa.
[13] Sistema de base de datos relacional de código abierto

datos relacional[14] para nuestra aplicación. Muchos programadores prematuros conocemos la finalidad de las bases de datos, además de trabajar con consultas, pero pocos conocen como implementarlas en una aplicación. Otra ventaja que nos ofrece Django es su seguridad [16]. Django ofrece varias medidas de seguridad que nos puede proteger de distintos ataques como:

- Protección contra ataques *Cross-site scripting* (XSS[15])
- Protección contra ataques *Cross-site request forgery* (CSRF[16])
- Protección contra ataques de inyecciones de SQL[17]

A pesar de que Django es un *framework* seguro, es importante aplicar buenas prácticas de seguridad y complementarlo con la protección de nuestro sistema operativo, servidor web y otros mecanismos.

En cuanto al diseño de la arquitectura, Django respeta el diseño conocido como MVC o modelo-vista-controlador, aunque no lo implementa de manera estricta. En su lugar, emplea un enfoque del modelo llamado MVT o modelo-vista-*template*, en el que la vista actúa como el controlador y la plantilla o *template* como la vista del patrón MVC [14].

El uso de Python para la programación lógica nos ha ayudado a controlar las solicitudes (conocidas en inglés como *requests*) que se generan en la aplicación web, a controlar qué métodos de petición (*POST* o *GET*) se utilizan y a conectarnos a la base de datos para realizar consultas como insertar y obtener los datos.

Se ha desarrollado código en JavaScript donde se analizan los eventos de teclado, trabajando así con el tipo de software llamado *Keylogger*. Además, utilizamos AJAX para enviar la información obtenida del usuario desde el archivo JavaScript al archivo Python.

Para finalizar, se quiere destacar el uso de la herramienta Docker[18], que ha supuesto un gran avance a la hora de desarrollar software. Esta herramienta permite al software ser portátil, ligero y autosuficiente. Se caracteriza por utilizar contenedores en vez de máquinas virtuales, haciéndolo independiente del sistema operativo. A diferencia de

[14] Los datos están relacionados entre sí.

[15] "Los ataques XSS permiten a un usuario inyectar secuencias de comandos del lado del cliente en los navegadores de otros usuarios." [16]

[16] "Los ataques CSRF permiten que un usuario malintencionado ejecute acciones usando las credenciales de otro usuario sin el conocimiento o consentimiento de ese usuario." [16]

[17] "La inyección SQL es un tipo de ataque en el que un usuario malintencionado puede ejecutar código SQL arbitrario en una base de datos. Esto puede provocar la eliminación de registros o la fuga de datos." [16]

[18] Proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos [15].

las máquinas virtuales, que necesitan recursos de disco, CPU y RAM, Docker solo necesitará el sistema operativo de la máquina donde se esté ejecutando el contenedor. Todo esto hace que sea una herramienta imprescindible para los programadores [15].

III. Diseño e Implementación de la propuesta

En este apartado se explicará cómo se ha llevado a cabo la implementación de la propuesta a partir de los pilares expuestos en el estado del arte.

a. Método de autenticación

Como se ha comentado con anterioridad, las contraseñas se están quedando obsoletas en cuanto a seguridad se refiere, por lo que debemos apostar por métodos de autenticación menos ortodoxos y más difíciles de ser robados. En el presente trabajo, se quiere implementar un método de autenticación mediante la cadencia de tecleo, una técnica poco utilizada. Usando esta nueva forma de autenticación, conseguiremos una seguridad más sólida al estudiar la manera de escribir del usuario. De la misma manera que en aplicaciones como Amazon, Facebook o Twitter, donde se almacenan numerosos usuarios con sus respectivas contraseñas, se podría reemplazar esto por usuarios y un modelo creado a partir de *machine learning*. Independientemente del ámbito donde se utilice este nuevo método, proporcionaría al usuario una forma mucho más segura de acceso.

Por ello, se ha desarrollado este proyecto para crear un prototipo de una aplicación web que emplee un nuevo método de autenticación. Además, la aplicación será accesible desde el ordenador de cualquier persona, ya que gracias a la herramienta Docker se descargarán los contenedores necesarios para ejecutar la aplicación en local.

Una vez ejecutada la aplicación, el usuario será capaz de navegar por ella sin necesidad de registrarse, pero solo será accesible el apartado “introducción”, que expondrá al usuario la finalidad del proyecto, una breve explicación de que consiste y quienes son los autores. Una vez el usuario se haya registrado, podrá acceder a los apartados de recogida de datos y de predicción, pudiendo así realizar las pruebas. Además, se permitirá la creación de un superusuario con acceso al panel de control de administración, donde se podrán monitorizar todos los eventos que ocurran en la aplicación web.

El flujo del modelo que se representa en la ilustración 5, consta de cinco etapas: recogida de datos, aprendizaje del usuario, predicción, comprobación y respuesta. En la primera etapa, se recogerán los datos del usuario una vez haya terminado de escribir un texto de prueba, y estos datos se almacenarán en la base de datos. En la etapa de aprendizaje del usuario, el algoritmo de *machine learning* deberá estudiar la cadencia de tecleo del usuario mediante los datos obtenidos y creará un modelo (cuantas más muestras estudie el modelo, más eficaz será). Luego, en la etapa de predicción, se recogerán y almacenarán los datos de la misma manera que en la primera etapa. Durante la etapa de comprobación, el algoritmo comparará los datos de la etapa de predicción con los modelos creados. Finalmente, una vez se hayan comparado los datos, se llevará a cabo la etapa de respuesta, en la cual se mostrará al usuario si es quien ha realizado ambas pruebas.

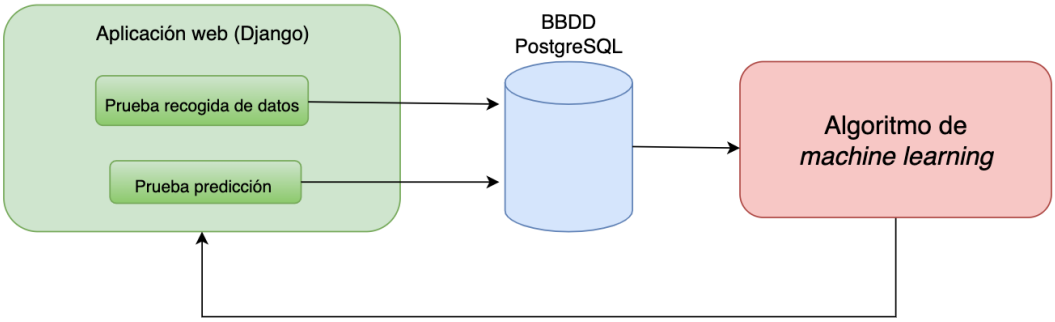

Ilustración 5: Flujo del modelo

b. Planificación

Como en todo proyecto software, se ha realizado una captura de requisitos previo al desarrollo de este. Su planificación se complementa con metodologías *agile* que ayudan a crear versiones o iteraciones, formando así fases incrementales. En la primera versión, se han incorporado las características necesarias para que el producto funcione. A lo largo de esta primera versión, se puede definir un orden a la hora de implementar las características de nuestra aplicación. De esta forma, en cada iteración se generará un prototipo que validará que el producto cumple con los requisitos, produciendo así un ciclo mucho más ágil. Además, si el resultado en cada iteración no es el esperado, se puede replanificar y rediseñar el producto, añadiendo, modificando o eliminando requisitos.

En cuanto a los requisitos funcionales que debía proporcionar la aplicación web:

- Registrarse en la aplicación web para ser capaces de realizar las pruebas.
- Iniciar sesión.
- Cerrar sesión.
- Acceder a la aplicación sin necesidad de estar registrado, pudiendo así navegar en ella.
- Recoger los datos del usuario en el apartado de “Recogida de datos” para la etapa de estudio.
- Recoger los datos del usuario en el apartado de “Predicción” para su posterior análisis con los datos estudiados.
- Ofrecer al usuario una predicción diciendo si es el usuario que dice ser o no.

Por otro lado, la aplicación debía cumplir una serie de requisitos no funcionales:

- Facilidad a la hora de navegar: Como se mencionó anteriormente, la aplicación está destinada a usuarios convencionales, por lo tanto, deben poder navegar por la aplicación sin ningún tipo de dificultad o esfuerzo.

- Tiempo de respuesta: La aplicación deberá de ser capaz de estudiar y comparar los datos recogidos de manera inmediata, lo que proporciona fluidez a la hora de realizar las pruebas de manera sucesiva.

- Portabilidad: Se deberá ejecutar la aplicación el cualquier entorno que posea las herramientas instaladas con la versión mínima indicada.

El código de la aplicación, junto a su documentación, se encuentra en la plataforma de GitHub[19] [8], y el software de la aplicación es de código abierto. Gracias a esta decisión, además de recibir feedback, la comunidad podrá trabajar de manera conjunta aportando ideas o corrigiendo errores.

c. Arquitectura de la aplicación

Para evitar cualquier problema de compatibilidad, se utilizó Django para el desarrollo de la aplicación. Como se explicó anteriormente, Django es una herramienta de software que nos permite trabajar tanto con el *Backend* (Python) como con el *Frontend* (HTML, JavaScript y CSS) de la aplicación web. Se basa en el patrón de diseño MVC (Modelo-Vista-Controlador) para crear el patrón MVT (Modelo-Vista-*Template*). Cada una de estas capas forma parte de cada aplicación o "app" en nuestro proyecto Django.

En este patrón, la capa de "*template*" está representada por los archivos HTML o plantillas que contiene la aplicación, la capa de "modelo" está representada por el archivo Python "models.py", y la capa de "vista" está formada por los archivos Python "urls.py" y "views.py" (véase la ilustración 6).

[19] Plataforma para alojar proyectos utilizando el sistema de control de versiones Git.

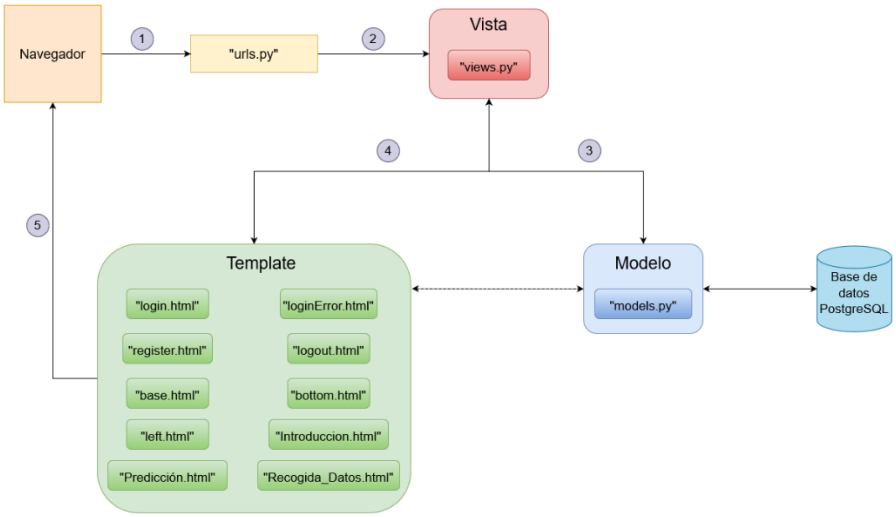

Ilustración 6: Arquitectura modelo-vista-controlador Django

Modelo

El modelo es la capa de la arquitectura que accede a la base de datos, dentro de nuestra aplicación encontraremos un archivo llamado "models.py" donde definiremos los modelos que utilizará nuestra aplicación. Estos modelos pueden ser propios o importados de las librerías de Django. En nuestro caso, utilizamos el modelo usuario de Django, así como su formulario para el registro.

Cada vez que se añada, modifique o elimine algún modelo, será necesario realizar las migraciones correspondientes, ya que la estructura de la base de datos se verá alterada. Todas las migraciones que se realicen se encontrarán en una carpeta llamada "*migrations*" dentro de nuestra aplicación de Django. Para realizar las migraciones deberemos escribir en nuestro directorio raíz los siguientes comandos:

- *python manage.py makemigrations*
- *python manage.py migrate*

Vista

Esta capa contiene toda la parte lógica de nuestra aplicación web y se programa en Python. Sirve como "puente" entre la capa modelo y la capa *template*. Su función es acceder al modelo y mostrar la plantilla correspondiente (aunque no todas las vistas devuelven un archivo HTML). Para ello, se declararán funciones que actuarán como vistas para la aplicación web. No todas las funciones en esta capa deben ser necesariamente vistas (pueden ser simplemente funciones auxiliares). Es decir, todas las vistas deben ser funciones, pero no todas las funciones tienen que ser vistas.

La programación lógica se encuentra en el archivo llamado "views.py", pero requiere del archivo llamado "urls.py" para definir sus URLs. En este archivo, se importarán las vistas y se les asignará una URL específica para de esta forma navegar y utilizar

las distintas vistas. También, se podrá asignar un nombre específico a cada URL para simplificar su uso.

Además, al importar la librería del adaptador psycopg2[20] para las bases de datos PostgreSQL, podemos establecer conexión y realizar consultas en nuestra base de datos, como insertar y devolver datos. Debido a su vínculo con el *Backend*, podemos enviar datos almacenados en el archivo "views.py" a nuestras plantillas utilizando diccionarios y consultar datos de nuestros archivos JavaScript empleando Ajax.

Esta capa es la más importante, ya que se encarga de asegurar el correcto funcionamiento de la aplicación. Aquí se encuentran funciones claves, como el registro del usuario, el inicio de sesión, cerrar sesión, calcular los datos necesarios que utilizará el algoritmo, insertar datos a la base de datos, consultar datos para generar un modelo y proceder la predicción.

Template

La capa *template* está formada por todos nuestros archivos HTML y sirve para separar la parte lógica de la parte visual. Estos archivos formarán la "presentación" de nuestra aplicación web y se ubicarán en la carpeta "templates" de nuestra aplicación.

Django nos permite trabajar con plantillas incrustadas y herencia de plantillas. En nuestro caso, contamos con una plantilla principal llamada "base.html", que incluye o incrusta otros archivos HTML que corresponden a partes concretas de nuestra aplicación, para así, no tener código duplicado. Para incluir estos archivos, debemos importarlos en la parte correspondiente de nuestra plantilla. Además, gracias a la herencia de plantillas, también evitaremos código duplicado, ya que tendremos la estructura general definida en el archivo "padre" (que en nuestro caso también corresponde con la plantilla "base.html"). Dentro de esta plantilla, podemos declarar bloques cambiantes que se definirán en lugares específicos. Estos bloques cambiantes afectarán a las plantillas "hijas" y serán únicos para cada una.

Los datos analizados se recogerán desde la propia aplicación. Con el fin de comprender mejor la estructura de la base de datos, se ha creado un Modelo Entidad-Relación (MER).

[20] Adaptador de la base de datos PostgreSQL para Python

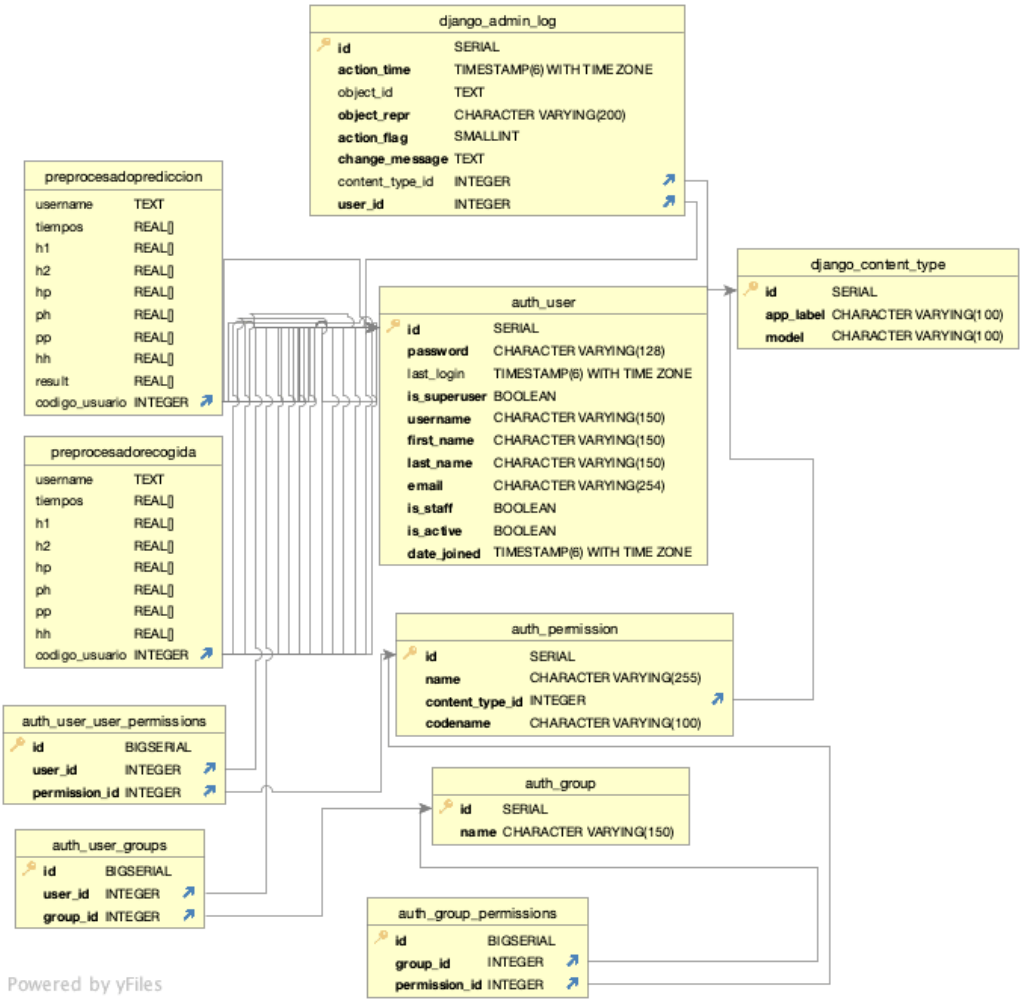

Ilustración 7: Modelo Entidad-Relación

IV. Desarrollo software

En este apartado se explica el proceso de implementación y desarrollo del software programado. Desde un punto de vista informático, se desarrollará la arquitectura del *framework* utilizado. Las pantallas que se desarrollarán a lo largo de esta sección se pueden observar en el Anexo B.

a. Registro e inicio de sesión

Lo primero que nos encontraremos al iniciar la aplicación web será la pantalla de inicio de sesión. Podremos acceder como usuario o invitado. En caso de acceder como invitado, solo podremos acceder a la pestaña de "introducción". Para poder acceder a las demás funcionalidades de la aplicación web, deberemos tener un usuario registrado y acceder con él. Todo esto se define en tres *templates* (*Frontend*) y tres funciones (*Backend*).

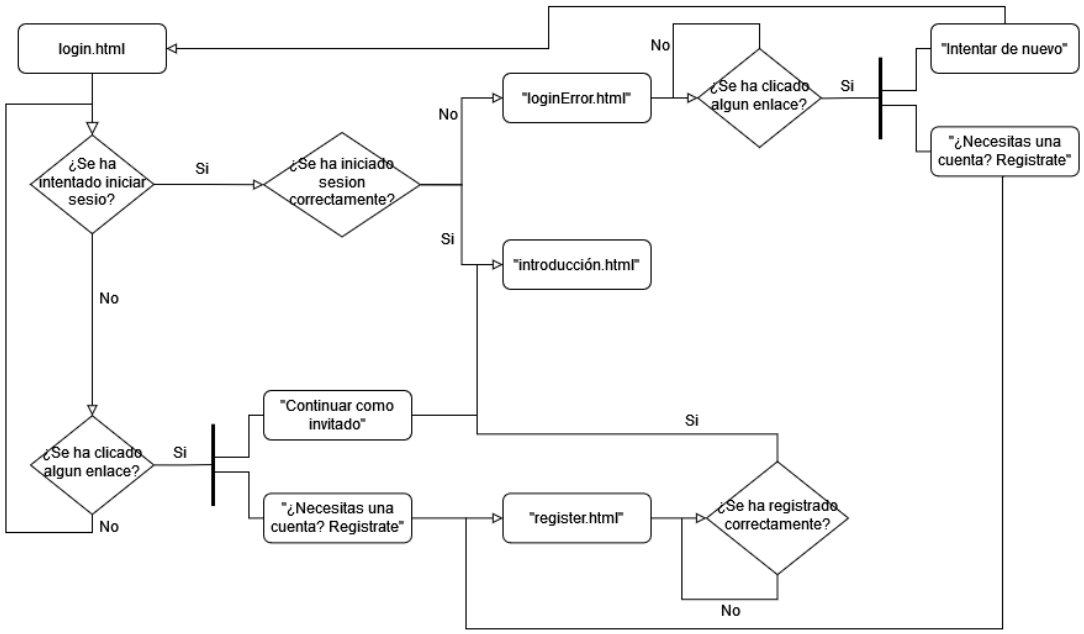

Ilustración 8: Diagrama de flujo registro e inicio de sesión

Frontend

Para la parte visual, con la que interactuará el usuario, se ha implementado tres *templates*.

- "login.html": Se encarga del inicio de sesión y mostrará:
  - Dos campos que rellenar (usuario y contraseña).
  - Un botón de acción "Iniciar sesión" para validar si ambos campos existen en nuestra base de datos y acceder con la cuenta a la página web.
  - Dos enlaces: el enlace "¿Necesitas una cuenta? Regístrate" nos redirigirá al *template* "register.html", y "Continuar como invitado" al *template* "introducción.html" (accediendo como invitado).

- “loginError.html”: Se redirigirá a este *template* en caso de que los campos rellenados no correspondan a ninguno de los usuarios que se encuentren en la base de datos y mostrará:
    - Un mensaje de error.
    - Un botón accionable “Intentar de nuevo” que nos redirigirá a “login.html”.
    - Un enlace “¿Necesitas una cuenta? Regístrate” que nos redirigirá al *template* “register.html”.

- “register.html”: *Template* cuya finalidad es el registro de una nueva cuenta. Para ello, deberemos rellenar una serie de campos mediante esta interfaz. Los campos requeridos son:
    - “Nombre de usuario”.
    - “Email”.
    - “Contraseña”.
    - “Confirme Contraseña”.

  Una vez hayamos rellenados todos estos campos, debemos pulsar el botón “Registrarse”. Si todos los campos son válidos, se creará un nuevo usuario y se almacenará en la base de datos. Si hemos accedido a este template pero no queremos crear una nueva cuenta, podremos hacer clic en el enlace "¿Tienes una cuenta? Inicia Sesión" para redirigirnos a "login.html".

Backend

En cuanto a la parte lógica, Django nos ofrece herramientas predefinidas que nos ayudan en su implementación. Tanto en los archivos “login.html” y “register.html”, añadiremos un formulario que creará automática los campos que se deben rellenar. Luego, en el archivo “forms.py”, deberemos importar las librerías “forms”, “UserCreationForm” y “User” de Django:

Una vez importadas, se ha definido una clase “UserRegisterForm” que utiliza la librería “UserCreationForm” para definir los campos que se deberán rellenar en el registro de una nueva cuenta. Deberemos crear tres funciones en el archivo “views.py”:

- “login_user”: Lo primero que se hará será comprobar que se está realizando un método *POST*. En caso afirmativo, almacenaremos el nombre del usuario y la contraseña escritos en dos variables. Usando el método *authenticate* de Python, podremos comprobar si el usuario existe o no en nuestra base de datos. En caso de que exista, se redirigirá a la página principal de nuestra aplicación con el usuario autenticado.

- “logout_user”: Usando la función *logout* de Python, podremos cerrar la sesión del usuario, para ello se deberá pulsar el botón de “Cerrar sesión” que se encuentra en el desplegable de la página principal.

- “register”: Lo primero que se hará será comprobar que se está realizando un método *POST*. En caso afirmativo, almacenaremos en una variable el formulario que se ha creado y se comprobará si es válido. En caso de que sea válido, se almacenará en la base de datos y se realizará el *login* del usuario.

b. Recogida de datos y tratamiento

En nuestra base de datos, encontraremos un total de 10 tablas en las cuales se recogen diversos datos, como usuarios registrados, permisos que pueden tener los usuarios (dependiendo de los permisos, podrán acceder al panel de control, realizar búsquedas en el panel de control y, agregar, modificar o eliminar usuarios), migraciones de la aplicación de Django, datos de la etapa de estudio, datos de la etapa de predicción, entre otros. Como se puede observar en la ilustración 8, todas las tablas están relacionadas entre sí debido al uso de un sistema de gestión de base de datos relacional (PostgreSQL). El uso de un sistema relacional proporciona simplicidad en el manejo de los datos y garantiza su seguridad. Por ejemplo, nuestras tablas principales son: “auth_user”, “preprocesadorecogida” y “preprocesadoprediccion” (se ha evitado el uso de mayúsculas y tildes para evitar conflictos entre el *framework* y la base de datos).

En la primera tabla se almacenarán los usuarios registrados en nuestra aplicación, donde encontraremos información relacionada con cada usuario, como su nombre de usuario, si es un superusuario, su última conexión, la hora a la que se registró, etc. Cada usuario contará con un ID que lo identificará, el cual será su clave primaria o *primary key*.

La tabla “preprocesadorecogida” almacenará el nombre del usuario que ha realizado la prueba, el identificador o ID del usuario, y una serie de listas o arrays que guardarán diferentes tiempos que usaremos más a delante.

Por último, la tabla “preprocesadoprediccion” contará con los mismos atributos que la tabla “preprocesadorecogida”, añadiéndole una última columna llamada “*result*”. Esta columna almacenará una lista que tomará los valores 1 o -1 El valor 1 representa "verdadero" o "positivo", mientras que el valor -1 representa "falso" o "negativo". Cada valor corresponderá al resultado de la comparación de la prueba de predicción con cada una de las muestras del modelo. El último valor de la lista indicará la decisión del algoritmo. Si el último valor es 1, significa que el usuario que haya ejecutado la prueba de predicción es el usuario que ha iniciado sesión. En caso contrario, si el último valor es -1, indica lo contrario.

Las listas de tiempos que se guardarán en las tablas “preprocesadorecogida” y “preprocesadoprediccion” corresponden con los siguientes atributos:

- “tiempos”: Un array que contiene otros arrays. Cada uno de estos subarrays almacenará los arrays: "tiempoKeyDown" (que guarda el valor de *timestamp* de cada evento *KeyDown* del teclado), "tiempoKeyPress" (que guarda el valor de *timestamp* de cada evento *KeyPress* del teclado) y "tiempoKeyUp" (que guarda el valor de *timestamp* de cada evento *KeyUp* del teclado).

- “h1”: Diferencia de tiempo entre el *timestamp* del evento *KeyDown* de la tecla A y el *timestamp* del evento *KeyUp* de la tecla A.

- “h2”: Diferencia de tiempo entre el *timestamp* del evento *KeyDown* de la tecla B y el *timestamp* del evento *KeyUp* de la tecla B.

- “hp”: Diferencia de tiempo entre el *timestamp* del evento *KeyUp* de la tecla A y el *timestamp* del evento *KeyDown* de la tecla B.

- “ph”: Diferencia de tiempo entre el *timestamp* del evento *KeyDown* de la tecla A y el *timestamp* del evento *KeyUp* de la tecla B.

- “hh”: Diferencia de tiempo entre el *timestamp* del evento *KeyUp* de la tecla A y el *timestamp* del evento *KeyUp* de la tecla B.

- “pp”: Diferencia de tiempo entre el *timestamp* del evento *KeyDown* de la tecla A y el *timestamp* del evento *KeyDown* de la tecla B.

En la ilustración 9 se puede observar el cálculo de los tiempos: H1, H2, HP, PH, HH y PP.

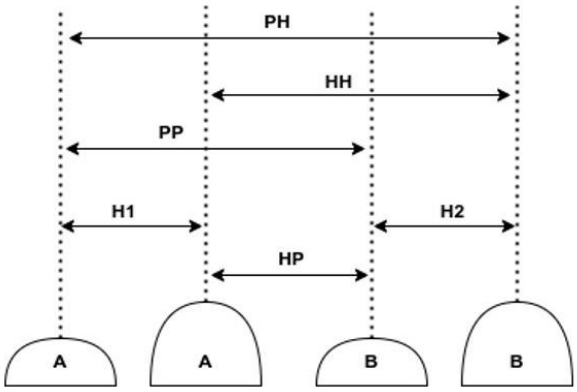

Ilustración 9: Relación tiempos H1, H2, HP, PH, HH y PP

c. Software de recogida

El software utilizado para la recopilación de datos del usuario, la generación de modelos y su posterior análisis en la sección de "Predicción" es de tipo *keyloggers*. Este tipo de software registra los eventos del teclado y lo utilizamos para capturar tres tipos de eventos:

- *KeyDown*: Evento que captura la tecla que se pulsa.
- *KeyPress*: Evento que captura la tecla que se presiona.
- *KeyUp*: Evento que captura la tecla que se levanta.

Estos eventos empezarán a capturarse una vez pulsemos el botón de “Iniciar prueba”. Desde ese instante, los eventos capturados se comparan con la cadena de texto que se muestra al usuario. En caso de que el carácter del evento corresponda con el carácter de la cadena de texto, se escribirá por pantalla el carácter capturado (esto sirve como guía para que el usuario sepa cuál es el próximo carácter que debe escribir). Además, se almacenará el *timestamp* del evento a su correspondiente array. Tenemos tres arrays, uno para cada evento. Estos *timestamps* se utilizan para calcular una serie de tiempos que ayudarán al algoritmo a crear el modelo y predecir al usuario.

Una vez escrita la cadena de texto correctamente, se habilitará un botón para enviar la información. Esta información se almacenará en nuestra base de datos y posteriormente se consultarán para generar el modelo y analizar al usuario.

Para desarrollar este *software*, se ha empleado el lenguaje de programación JavaScript. JavaScript nos proporciona tres tipos de eventos por teclado: *keydown*, *keypress* y *keyup*. En las ilustraciones 10, 11, 12 y 13 se puede observar cómo inicializamos las variables que vamos a utilizar, capturamos dichos eventos, y vamos almacenando sus *timestamp*. También realizamos una serie de comprobaciones para asegurar su correcto funcionamiento, como verificar que el carácter escrito sea el correcto o que se deba pulsar la tecla "Dead" o "´" para escribir una vocal con tilde.

Contaremos con dos códigos, uno para cada prueba. A continuación, se mostrará el código de la prueba de “Recogida de datos”. Este código es idéntico al de la prueba de “Predicción” debido a que es más sencillo para el algoritmo comparar datos extraídos de textos iguales. Solo varían el nombre de algunas varíales, pero no su comportamiento. Estas son: “texto_Recogida”, “text_in_Recogida”, “botonEnviarRecogidaDatos”, “idNombre_Recogida“ y “segundo_Recogida”.

En la ilustración 10, podemos observar la declaración de las variables que vamos a utilizar en nuestro código:

- “texto_Recogida”: Almacena los datos que se encuentren en la etiqueta del archivo “Recogida_Datos.html” con el id: “textoRecogida”. Con el atributo “outerText”, podremos acceder a la cadena de texto.

- “contadorKeyUp”, “contadorKeyDown” y “contadorKeyPress”: Estos contadores nos ayudarán a la hora de realizar un seguimiento del carácter que se debe escribir de la cadena.

- “errores”: Se almacenarán en un contador los errores que se cometan a la hora de escribir, en este caso, cada carácter mal escrito corresponderá a un error. Por ahora no se utilizará esta variable en el modelo del algoritmo, pero en futuras versiones se espera utilizarlo.

- “text_in_Recogida”: En esta variable se almacenará los datos que se encuentre en la etiqueta del archivo “Recogida_Datos.html” con el id: “text_in_Recogida”. Con el atributo “textContent” podremos acceder a la cadena de texto y modificarla.

- “tildeKeyUp”, “tildeKeyDown” y “tildeKeyPress”: Variable *booleana* que, en caso de almacenar *true,* indicarán que el siguiente carácter es una vocal con tilde. Si almacena *false*, nos indicará que el siguiente carácter no lleva tilde.

- “primeraLetra”: Variable *booleana*. Al desarrollar el código, se encontró con un problema a la hora de escribir por pantalla la primera letra. Al ser la primera letra se debe escribir con mayúscula, pero el evento de teclado *keyup* no detectaba que la tecla pulsada fuese mayúscula, por lo que, a la hora de comparar el carácter del evento con el carácter de la cadena se comparaba una letra minúscula con una mayúscula. Para resolver este problema, se agregó esta variable *booleana*, la cual cuando en el evento de teclado *keydown* se detecte que se ha escrito correctamente la primera letra, se le asigna el valor de *true*. Después, cómo ya hemos comprobado que el carácter es el que corresponde, se indicará en el evento *keyup* que el carácter en minúscula está escrito correctamente.

- “idNombre_Recogida”: En esta variable se almacenará los datos que se encuentre en la etiqueta del archivo “Recogida_Datos.html” con el id: “nombreUsuario”.

- “username”: Una vez se haya almacenado el nombre de usuario nos aseguraremos de que se suprimen los saltos de línea o espacios en la variable idNombre_Recogida.

- “segundo_Recogida”: Como el valor de los atributos *timestamp* se guarda en milisegundos, se ha declarado esta constante con el valor de 1000. Esto servirá para convertir los tiempos de milisegundos a segundos.

- “tiempos”: Este array almacenará otros tres arrays. Cada uno de estos arrays representará a un evento distinto: *keydown*, *keypress* y *keyup* (en ese orden).

```
BEGIN
        initialize: texto_Recogida = "", contadorKeyUp = 0, contadorKeyDown = 0, contadorKeyPress = 0, errores = 0, texto_in_Recogida = "",
        botorEnviarRecogidaDatos = null, tildeKeyUp = false, tildeKeyDown = false, tildeKeyPress = false, primeraLetra = false,
        idNombre_Recogida = null, username = "", segundo_Recogida = 1000, tiempos = [[], [], []]


        texto_Recogida = getElementById("textoRecogida")
        texto_in_Recogida = getElementById("texto_in_Recogida")
        botorEnviarRecogidaDatos = botorEnviarRecogidaDatos ("botorEnviarRecogidaDatos")
        idNombre_Recogida = getElementById("nombreUsuario")
        username = idNombre_Recogida.textContent.replace(/\s+g, '')

        PROCEDURE keyUp(type 'keyup', event KeyboardEvent)

        PROCEDURE keyPress(type 'keypress', event KeyboardEvent)

        PROCEDURE keyDown(type 'keydown', event KeyboardEvent)

END
```

Ilustración 10: Pseudocódigo declaración variables para los keyloggers

En la ilustración 11 se puede observar el evento *keydown*. Este evento se encargará de almacenar los *timestamp* correspondientes al *keydown* y de escribir por pantalla los caracteres tecleados de manera correcta. Primero se comprueba si la tecla del evento corresponde con el carácter de la cadena de texto. En caso de que nos encontremos en la primera letra, nuestro “contadorKeyDown” tendrá el valor de 0 y deberemos asignarle el valor *true* a nuestra variable “primeraLetra”. Después, se verifica si el carácter no lleva tilde, en caso afirmativo, se actualizará el contenido de nuestra variable “text_in_Recogida” añadiéndole el carácter actual, se incrementará en uno el valor de nuestro “contadorKeyDown” y añadiremos a la primera posición del array “tiempos” el *timestamp* del evento calculado en segundos.

En caso de que el carácter tenga tilde, debemos realizar un filtro en la posición específica de la cadena de texto. Primero, deberemos capturar la tecla “*Dead*” o “´”, una vez hecho esto, se actualizarán los valores de “tildeKeyDown” y “tildeKeyPress” a *true* (este último valor se actualiza en el evento *keydown* ya que el evento *keypress* no detecta los carácteres “*Dead*” o “´”). Luego comprobamos que nos encontramos en la posición del carácter con tilde y que se captura esa tecla del evento, ya sea con tilde o sin ella (esto debido a que el sistema operativo MacOS captura el carácter sin tilde y Windows lo captura con tilde).

Por último, verificamos que la variable “tildeKeyDown” almacena el valor true, en caso afirmativo, actualizamos el contenido de nuestra variable “text_in_Recogida” añadiéndole el carácter con tilde, añadimos en la primera posición del array “tiempos” el *timestamp* del evento calculado en segundos, le asignaremos el valor *false* a la variable “tildeKeyDown” y se incrementará en uno el valor de nuestro “contadorKeyDown”.

Para finalizar con este evento, se imprimirá por pantalla la tecla que el usuario haya pulsado, de esta forma podrá hacer un seguimiento de todas las teclas pulsadas.

```
PROCEDURE keyDown(type 'keydown', event KeyboardEvent)
BEGIN
        if event.key = texto_Recogida[contadorKeyUp] then
                if contadorKeyDown = 0 then
                        primeraLetra = true
                end if

                if tildeKeyDown = false then
                        text_in_Recogida = text_in_Recogida + texto_Recogida[contadorKeyDown]
                        tiempos[0].add(event.timeStamp/ segundo_Recogida)
                        contadorKeyDown = contadorKeyDown + 1
                end if
        end if

        if (event.key = "i" or event.key = "í") and contadorKeyPress = 36 then
                if tildeKeyUp = true then
                        text_in_Recogida = text_in_Recogida + "\uBED"
                        tildeKeyDown = false
                        tiempos[0].add(event.timeStamp/ segundo_Recogida)
                        contadorKeyDown = contadorKeyDown + 1
                end if
        end if

        if (event.key = "o" or event.key = "ó") and contadorKeyPress = 105 then
                if tildeKeyUp = true then
                        text_in_Recogida = text_in_Recogida + "\uBF3"
                        tildeKeyDown = false
                        tiempos[0].add(event.timeStamp/ segundo_Recogida)
                        contadorKeyDown = contadorKeyDown + 1
                end if
        end if
END
```

Ilustración 11: Pseudocódigo evento por teclado keydown

En cuando al evento *keypress*, el procedimiento es idéntico al evento *keydown*, con la única diferencia de que en este caso se agregan los *timestamp* a la segunda posición del array “tiempos”. Además, el control de las tildes se realizará en el evento *keydown*, como hemos explicado anteriormente.

```
PROCEDURE keyPress(type 'keypress', event KeyboardEvent)
BEGIN
        if event.key = texto_Recogida[contadorKeyUp] then
                if tildeKeyPress = false then
                        tiempos[1].add(event.timeStamp/ segundo_Recogida)
                        contadorKeyPress = contadorKeyPress + 1
                end if
        end if

        if (event.key = "i" or event.key = "í") and contadorKeyPress = 36 then
                if tildeKeyUp = true then
                        tiempos[1].add(event.timeStamp/ segundo_Recogida)
                        tildeKeyPress = false
                        contadorKeyPress = contadorKeyPress + 1
                end if
        end if

        if (event.key = "o" or event.key = "ó") and contadorKeyPress = 105 then
                if tildeKeyUp = true then
                        tiempos[1].add(event.timeStamp/ segundo_Recogida)
                        tildeKeyPress = false
                        contadorKeyPress = contadorKeyPress + 1
                end if
        end if
END
```

Ilustración 12: Pseudocódigo evento por teclado keypress

El procedimiento del evento *keyup* es similar al evento *keydown*, pero agrega nuevas funcionalidades. Estas funcionalidades se basan en:

- Captura de errores: Cada vez que el usuario escriba incorrectamente un carácter, se incrementará en uno el contador “errores”. Sin embargo, en algunos casos, se deberá corregir este incremento, como al utilizar las teclas “Meta”, “Shift” o “CapsLock” para escribir letras mayúsculas, o las teclas “Dead” o “´” para escribir un carácter con tilde.

- Fin de la prueba: Cuando el usuario haya completado la prueba, es decir, haya escrito correctamente todos los caracteres, la variable “contadorKeyUp” tendrá el mismo valor que la longitud de nuestra cadena de texto. Si esto ocurre, mostraremos en la consola del navegador cuantos errores ha cometido el usuario, el array de tiempos y eliminaremos la clase “disable” de nuestro botón “botonEnviarRecogidaDatos” para poder enviar los datos recopilados a la base de datos. Antes de completar la prueba, el botón no podía ser presionado.

```
PROCEDURE keyUp(type 'keyup', event KeyboardEvent)
BEGIN
	if event.key <> texto_Recogida[contadorKeyUp] then
		errores = errores + 1
		if event.key = 'Meta' or event.key = 'Shift' or event.key = 'CapsLock' then
			errores = errores - 1
		end if
	end if

	if event.key = texto_Recogida[contadorKeyUp] or primeraLetra then
		if contadorKeyUp = 0 then
			primeraLetra = false
		end if
		tiempos[2].add(event.timeStamp/ segundo_Recogida)
		contadorKeyUp = contadorKeyUp + 1
	end if

	if (event.key = "i" or event.key = "í") and contadorKeyUp = 36 then
		if tildeKeyUp = true then
			tiempos[2].add(event.timeStamp/ segundo_Recogida)
			tildeKeyUp = false
			contadorKeyUp = contadorKeyUp + 1
		end if
	end if

	if (event.key = "o" or event.key = "ó") and contadorKeyUp = 105 then
		if tildeKeyUp = true then
			tiempos[2].add(event.timeStamp/ segundo_Recogida)
			tildeKeyUp = false
			contadorKeyUp = contadorKeyUp + 1
		end if
	end if

	if (event.key = "Dead" or event.key = "'") and (contadorKeyUp = 36 or contadorKeyUp = 105) then
		tildeKeyUp = true
		errores = errores - 1
	end if

	if contadorKeyUp = texto_Recogida.size then
		writeln("Has cometido " +errores+ " errores")
		writeln(tiempos)
		botonEnviarRecogidaDatos.removeClass("disable")
	end if
END
```

Ilustración 13: Pseudocódigo evento por teclado keyup

Una vez se hayan almacenado todos los *timestamp* en nuestros diversos arrays, se añadirán a un array conjunto llamado “tiempos”. Luego, haremos uso de Ajax para enviar la información a nuestro archivo “views.py”. Los datos que se enviarán a nuestra función incluyen el *token* del usuario, el nombre del usuario y el array “tiempos”. Después, utilizando nuestro array “tiempos”, calcularemos el preprocesado de datos, que implica

calcular una serie de tiempos (H1, H2, HP, PH, HH y PP) que utilizará nuestro algoritmo de *machine learning* para crear un modelo. Se observa el cálculo de estas variables en el pseudocódigo de la ilustración 14.

```
PROCEDURE convertData( JSON result)
BEGIN
    initialize : cont=0, keyDownTime = [], keyPressTime = [], keyUpTime = []

    for each times ∈ result do
        for each time ∈ times do
            if cont = 0 then
                insert time into keyDownTime
            end if
            if cont = 1 then
                insert time into keyPressTime
            end if
            if cont = 2 then
                insert time into keyUpTime
            end if
        end for

        cont = cont + 1
    end for

    times = [keyDownTime, keyPressTime, keyUpTime]

    H1 = [], H2 = [], HP = [], PH = [], PP = [], HH = []

    for i to keyDownTime length do
        if i =< keyDownTime length - 1 then
            insert (keyUpTime[i] - keyDownTime[I]) into H1

            if i < keyDownTime length - 1
                insert (keyUpTime[i+1] - keyDownTime[i+1]) into H2
                insert (keyDownTime[i + 1] - keyUpTime[i]) into HP
                insert (keyUpTime[i+1] - keyDownTime[i]) into PH
                insert (keyDownTime[i+1] - keyDownTime[i]) into PP
                insert (keyUpTime[i+1] - keyUpTime[i]) into HH
            end if

        i = i + 1
        end if
    end for

    return times, H1, H2, HP, PH, PP, HH
END
```

Ilustración 14: Pseudocódigo preprocesado de los tiempos para calcular H1, H2, HP, PH, HH y PP

Por último, una vez calculados los tiempos necesarios y almacenarlos en los arrays “H1”, “H2”, “HP”, “PH”, “HH” y “PP”, se insertarán en la base de datos. A continuación, estos datos se consultarán para crear el modelo y para el posterior análisis del usuario. Para visualizar el resultado del análisis realizado por el algoritmo, se debe hacer clic en el botón "Ver Resultado" en la pestaña de "Predicción". Esto mostrará un cuadro emergente que indicará el resultado final.

V. Experimentos

En este apartado se presentarán una serie de experimentos que se han llevado a cabo para poner a prueba el algoritmo. Además, para evaluar la eficacia y el correcto funcionamiento del algoritmo, se han desarrollado matrices de confusión.

Una vez programado y definido el algoritmo, se pretende validar la propuesta a través de una serie de experimentos (4 experimentos), cada uno con veinte predicciones. Cada experimento utilizará un modelo diferente alimentado con muestras distintas. Esto significa que cada modelo se comportará de manera distinta, por lo tanto, será necesario analizar sus diferentes capacidades a la hora de determinar si la cadencia de tecleo pertenece a un usuario autenticado o a un usuario "intruso". Como se ha comentado a lo largo del trabajo, cuantas más muestras estudie SVM, más eficaz será. En base a estas muestras, generará un modelo que utilizaremos para realizar predicciones sobre un conjunto de pruebas.

Para cada uno de los experimentos se ha elaborado un *train* o conjunto de entrenamiento con 1, 5, 10 y 20 muestras respectivamente. Estas muestras son heterogéneas, lo que significa que encontraremos diferentes cadencias de tecleo, como rápidas, lentas o neutras.

En cuanto al conjunto de pruebas, se realizarán 20 predicciones: 10 de ellas corresponden al usuario genuino y las 10 restantes corresponden al usuario impostor.

Para evitar que el algoritmo identifique de manera sencilla al usuario "intruso" en caso de escribir más rápido o más lento, deberemos de entrenar el modelo con algunas muestras que se adapten a estas velocidades de tecleo. Identificamos los siguientes experimentos y sus respectivas muestras:

- Experimento 1 (1 muestra y 20 predicciones), muestra perteneciente al usuario genuino:
    - Para entrenar el modelo que utilizará el algoritmo en este experimento, se ha optado por escribir con una cadencia de tecleo "normal", es decir, la velocidad de escritura que usaremos habitualmente.

- Experimento 2 (5 muestras y 20 predicciones), muestras pertenecientes al usuario genuino:
    - 3 muestras con velocidad de tecleo habitual.
    - 1 muestra con velocidad de tecleo más rápida.
    - 1 muestra con velocidad de tecleo más lenta.

- Experimento 3 (10 muestras y 20 predicciones), muestras pertenecientes al usuario genuino:
    - 4 muestras con velocidad de tecleo habitual.
    - 3 muestras con velocidad de tecleo más rápida.
    - 3 muestras con velocidad de tecleo más lenta.

- Experimento 4 (20 muestras y 20 predicciones), muestras pertenecientes al usuario genuino:

- 16 muestras con velocidad de tecleo habitual.
- 2 muestras con velocidad de tecleo más rápida.
- 2 muestras con velocidad de tecleo más lenta.

Matriz de confusión

La matriz de confusión es una herramienta estadística que nos permite visualizar el desempeño de un modelo a través de la matriz, donde su tamaño dependerá de la cantidad de clases que deseemos predecir [20].

Métodos de validación

En la tabla 1 se muestra un ejemplo de una matriz de confusión que se divide en dos clases: Positivos y Negativos. En estos experimentos, se asume que la clase positiva o 1 está representada por el usuario autenticado, mientras que la clase negativa o -1 está representada por el usuario "intruso". Las instancias serán clasificadas en 4 posibles resultados: VP (Verdadero Positivo), FN (Falso Negativo), FP (Falso Positivo) y VN (Verdadero Negativo). En la casilla VP se representarán las instancias correctamente clasificadas por el modelo como positivas, mientras que la casilla FN representará las instancias positivas que han sido clasificadas incorrectamente como negativas. Por otro lado, la casilla FP representará las instancias negativas que han sido clasificadas incorrectamente como positivas, y la casilla VN representa el número de instancias clasificadas correctamente como negativas [19].

		Condiciones de Predicción	
	Total = P + N	Positivo (PP) Usuarios autenticados	Negativo (PN) Usuarios “intrusos”
Condición Actual	Positivo (P) Usuarios autenticados	Verdadero Positivo (VP) Aquí se clasificarán las predicciones correctamente predichas sobre los usuarios autenticados	Falso negativo (FN) Aquí se clasificarán las predicciones incorrectamente predichas sobre los usuarios autenticados
	Negativo (N) Usuarios “intrusos”	Falso positivo (FP) Aquí se clasificarán las predicciones incorrectamente predichas sobre los usuarios “intrusos”	Verdadero negativo (VN) Aquí se clasificarán las predicciones correctamente predichas sobre los usuarios “intrusos”

Tabla 1: Clasificación de los datos en la matriz de confusión [19]

Métricas

Se han definido diferentes métricas de rendimiento en nuestra matriz que utilizarán los valores mencionados en el anterior punto [20].

- *Accuracy*: Nos indica la proporción de aciertos en mi modelo.

$$Accuracy = \frac{VP + VN}{VP + FP + FN + VN}$$

Ecuación 5: Cálculo de Accuracy [19]

- Precisión: Nos indica la proporción de instancias positivas clasificadas correctamente frente al total de instancias clasificadas como positivas.

$$\text{Precisión} = \frac{VP}{VP + FP}$$

Ecuación 6: Cálculo de Precisión [19]

- Sensibilidad: Nos indica el porcentaje de instancias positivas clasificadas correctamente.

$$\text{Sensibilidad} = \frac{VP}{VP + FN}$$

Ecuación 7: Cálculo de Sensibilidad [19]

- Especificidad: Nos indica el porcentaje de instancias negativas clasificadas correctamente.

$$\text{Especificidad} = \frac{VN}{VN + FP}$$

Ecuación 8: Cálculo de Especificidad [19]

Experimento 1

Se genera un modelo con 1 muestra en el conjunto de entrenamiento. Posteriormente, se realizan 20 predicciones de las cuales 10 pertenecen al usuario autenticado y 10 son simulaciones que pertenecen al usuario impostor.

		Condiciones de Predicción Experimento 1	
	Total 20	Usuario autenticado	Usuario “intruso”
Condición Actual	Usuario autenticado	0	10
	Usuario “intruso”	0	10

Tabla 2: Matriz de confusión experimento 1

El modelo del experimento 1 muestra síntomas de ser muy estricto. Ha desautenticado al usuario autenticado en 10 ocasiones, prediciendo erróneamente que era un usuario "intruso". Por otro lado, ha acertado en 10 ocasiones al predecir que el usuario "intruso" está suplantando al usuario autenticado (este es el error más "costoso", ya que implica un acceso no deseado a la aplicación).

Experimento 2

Se genera un modelo con 5 muestras en el conjunto de entrenamiento. Posteriormente, se realizan 20 predicciones de las cuales 10 pertenecen al usuario autenticado y 10 son simulaciones que pertenecen al usuario impostor.

		Condiciones de Predicción Experimento 2	
	Total 20	Usuario autenticado	Usuario “intruso”
Condición Actual	Usuario autenticado	8	2
	Usuario “intruso”	5	5

Tabla 3: Matriz de confusión experimento 2

El modelo del experimento 2 se podría clasificar como efectivo a la hora de predecir a los usuarios autenticados y equilibrado a la hora de predecir a los usuarios “intrusos”. Ha acertado en 8 ocasiones al identificar al usuario autenticado correctamente. Sin embargo, también ha desautenticado al usuario autenticado en 2 ocasiones, es decir, ha realizado predicciones erróneas al considerarlo como un usuario "intruso". Además, ha acertado en 5 ocasiones al detectar que un usuario impostor estaba suplantando al usuario autenticado. Por último, ha realizado predicciones erróneas en 5 ocasiones al identificar a un usuario impostor como el usuario autenticado.

Experimento 3

Se genera un modelo con 10 muestras en el conjunto de entrenamiento. Posteriormente, se realizan 20 predicciones de las cuales 10 pertenecen al usuario autenticado y 10 son simulaciones que pertenecen al usuario impostor.

		Condiciones de Predicción Experimento 3	
	Total 20	Usuario autenticado	Usuario “intruso”
Condición Actual	Usuario autenticado	6	4
	Usuario “intruso”	6	4

Tabla 4: Matriz de confusión experimento 3

El modelo del experimento 3 se podría clasificar como equilibrado a la hora de predecir tanto usuarios autenticados como usuarios “intrusos”. Ha acertado en 6 ocasiones al identificar al usuario autenticado correctamente. Sin embargo, también ha desautenticado al usuario autenticado en 4 ocasiones, es decir, ha realizado predicciones erróneas al considerarlo como un usuario "intruso". Además, ha acertado en 4 ocasiones al detectar que un usuario impostor estaba suplantando al usuario autenticado. Por último, ha realizado predicciones erróneas en 6 ocasiones al identificar a un usuario impostor como el usuario autenticado.

Experimento 4

Se genera un modelo con 20 muestras en el conjunto de entrenamiento. Posteriormente, se realizan 20 predicciones de las cuales 10 pertenecen al usuario autenticado y 10 son simulaciones que pertenecen al usuario impostor.

		Condiciones de Predicción Experimento 4	
	Total 20	Usuario autenticado	Usuario “intruso”
Condición Actual	Usuario autenticado	9	1
	Usuario “intruso”	2	8

Tabla 5: Matriz de confusión experimento 4

El modelo del experimento 4 muestra síntomas de ser más completo frente a los otros tres experimentos. Ha predicho correctamente en 9 ocasiones al usuario autenticado. Sin embargo, ha desautenticado al usuario autenticado en 1 ocasión, prediciendo erróneamente que era un usuario "intruso". Como podemos observar, este modelo es el que mejor clasifica a los usuarios autenticados. Por otro lado, ha acertado en 8 ocasiones al predecir que el usuario "intruso" está suplantando al usuario autenticado. Por último, ha realizado predicciones erróneas en 2 ocasiones al identificar a un usuario impostor como el usuario autenticado. En cuanto a la clasificación de usuarios “intrusos”, no es tan eficaz como el primer modelo, pero se acerca bastante a los resultados. Gracias a los resultados podemos concluir que este modelo es permisivo con los usuarios autenticados y estricto con los usuarios “intrusos”.

En nuestro caso, vamos a profundizar en los valores obtenidos tanto en sensibilidad como en especificidad.

La sensibilidad nos indica qué tan eficaz es el modelo al clasificar correctamente las predicciones que corresponden al usuario autenticado. Cuanto mayor sea el número obtenido, mayor será la cantidad de usuarios autenticados correctamente clasificados, en comparación con los usuarios autenticados clasificados incorrectamente.

En cuanto a la especificidad, este valor nos indica qué tan estricto es nuestro modelo al clasificar a los usuarios "intrusos". Cuanto mayor sea el número obtenido, mayor será la cantidad de usuarios "intrusos" que están siendo clasificados correctamente como usuarios “intrusos”.

Si un modelo es más estricto o más laxo dependerá del contexto en el que se utilice. Por ejemplo, en una aplicación bancaria, es preferible que la especificidad del modelo sea mayor para evitar accesos de usuarios "intrusos" aumentando así la seguridad de la aplicación, aunque esto signifique clasificar incorrectamente a algunos usuarios autenticados (experimento 1). Por otro lado, si el lugar al que queremos acceder no contiene datos sensibles, es conveniente obtener un mayor número de usuarios autenticados clasificados correctamente (experimento 4). En la aplicación web desarrollada, se le brinda al usuario la opción de elegir entre un modelo más estricto o más laxo, según cómo entrene el modelo.

A priori, puede parecer que, para obtener un modelo más estricto, este debe ser menos sensible. Sin embargo, esto no es del todo correcto. A continuación, clasificaremos los valores obtenidos en las métricas explicadas anteriormente. Estos valores están comprendidos en el intervalo [0-1].

Experimentos	Accuracy	Precisión	Sensibilidad	Especificidad
Experimento 1	0,5	Indeterminación	0	1
Experimento 2	0,65	0,6154	0,8	0,5
Experimento 3	0,5	0,5	0,6	0,4
Experimento 4	0,85	0,81	0,9	0,8

Tabla 6: Recopilación valores obtenidos en las matrices de confusión

Si analizados los valores obtenidos en las métricas, podemos observar que se relaciona con lo que se ha comentado anteriormente.

- Experimento 1: Cuenta con un modelo muy estricto que ha predicho correctamente a todos los usuarios "intrusos", lo que resulta en un valor máximo de especificidad. Sin embargo, el modelo no ha predicho correctamente ningún usuario autenticado, lo que implica que el valor obtenido en la sensibilidad será el mínimo posible.

- Experimento 2: Utiliza un modelo permisivo a la hora predecir a los usuarios autenticados, lo que acerca el valor de sensibilidad al máximo. Por otro lado, a la hora de predecir usuarios "intrusos", el modelo se muestra equilibrado, por lo que el valor obtenido en especificidad deberá encontrarse dentro del valor medio del intervalo.

- Experimento 3: Cuenta con un modelo equilibrado para predecir tanto a los usuarios autenticados como a los usuarios "intrusos", lo que significa que los valores de sensibilidad y especificidad se encontraran dentro del valor medio del intervalo.

- Experimento 4: El modelo utilizado en este experimento es permisivo a la hora de clasificar a los usuarios autenticados y estricto a al clasificar los usuarios "intrusos", por lo que los valores obtenidos tanto en la sensibilidad como especificidad deberán acercarse al máximo posible.

VI. Conclusiones y trabajo futuro

El auge de la web 3.0 y la creciente automatización en los distintos sectores nos obliga a reinventarnos, en este caso, buscando nuevos métodos o formas de implantar seguridad.

En cuanto a la aplicación web diseñada, el objetivo era crear interfaces sencillas para que los usuarios pudieran navegar con la menor cantidad de problemas. Aunque se ha logrado este objetivo, la interfaz no ha resultado ser “llamativa” o “atractiva” como se esperaba en un primer momento.

Como se ha explicado anteriormente, a través del método de autenticación propuesto, es posible detectar de manera precisa si el usuario que realiza la prueba de predicción es el mismo que está autenticado en la aplicación web, lo cual cumple con su propósito. Aunque se ha cumplido con las expectativas iniciales, aún nos encontramos en la etapa inicial de este proyecto, ya que no ha sido probado por múltiples usuarios. Para avanzar en el desarrollo, se consideran los siguientes trabajos futuros:

- Implementar el método de autenticación propuesto al registro e inicio de sesión de los usuarios en la aplicación. En lugar de utilizar contraseñas, se utilizará este nuevo método.

- Implementar otro método de autenticación basado en el uso del ratón. En este caso se mostrará un punto rojo que deberá ser clicado por el usuario y se deberá realizar un número determinado de veces. El punto aparecerá en una posición aleatoria y estará visible por un tiempo limitado. Esto permitirá evaluar el rendimiento del usuario en términos de precisión y velocidad al utilizar el ratón.

- Poder realizar encuestas a varios usuarios para obtener *feedback* sobre la aplicación.

- Llevar a cabo un seguimiento por parte de la comunidad para realizar un estudio que permita evaluar el desempeño de la aplicación web, así como su mejora y mantenimiento.

- Implementar el algoritmo de aprendizaje automático en una página o aplicación web reconocida con el fin de demostrar su funcionalidad y utilidad.

A continuación, se han recopilado las tecnologías y métodos empleados a lo largo del proyecto, así como sus ventajas y desventajas:

- Django: El uso de este *framework* ha supuesto un acierto en términos de compatibilidad, debido a que el algoritmo de *machine learning* estaba escrito en Python. También, proporciona una capa de seguridad adicional en nuestra aplicación web.
    - Ventajas: Compatibilidad con el algoritmo de *machine learning* y facilidad de implementación de algoritmos propios. También brinda una capa adicional de seguridad.
    - Desventajas: No se había utilizado previamente esta herramienta.

- *Keyloggers*: Para la recopilación de datos destinados al análisis posterior, se utilizó la tecnología de *keyloggers*.
    - Ventajas: Al ser una tecnología que lleva utilizándose muchos años, cuenta con diversas funciones predefinidas que facilitan la extracción de datos.
    - Desventajas: Durante el desarrollo del código se han presentado varios conflictos en la compatibilidad con los sistemas operativos Windows y MacOS.

- Docker: Nos ha ayudado a la hora de desplegar el software de la aplicación en contenedores.
    - Ventajas: Gracias a la tecnología de contenedores nuestro software puede ser portable.
    - Desventajas: Puede resultar difícil de instalar y configurar el software. Además, se requieren conocimientos específicos para manejar los archivos Docker de la aplicación.

- PostgreSQL: Desde el principio, se decidió trabajar con una base de datos relacional para que, al eliminar un usuario de la base de datos, también se eliminaran las pruebas relacionadas con él. Afortunadamente, Django admite oficialmente varios tipos de bases de datos relacionales, y en nuestro caso hemos elegido PostgreSQL.
    - Ventajas: Se está familiarizado con esta base de datos.
    - Desventajas: A pesar de ser una de las bases de datos relacionales más utilizadas, dispone de menos documentación en comparación con Oracle o MySQL.

- *Machine learning*: El núcleo principal de este proyecto se basa es la creación de modelos mediante *machine learning* utilizando SVM. Aunque anteriormente no se ha trabajado con este algoritmo de aprendizaje supervisado, nos ha ayudado a clasificar correctamente los datos recogidos para la posterior detección de usuarios.
    - Ventajas: Actualmente, es una tecnología muy importante y cotizada.

    - Desventajas: Es una tecnología con la que nunca se ha tenido la oportunidad de trabajar.

- Scrum: Se ha decidido usar Scrum como metodología de trabajo para agilizar el desarrollo del proyecto.
    - Ventajas: El uso de metodologías agiles ayuda a ser más productivos a la hora de realizar un trabajo.

Durante los meses de trabajo, el principal objetivo de este proyecto fue desarrollar una aplicación web con nuevas herramientas y promover de alguna forma el uso de este método de autenticación innovador. A pesar de no estar familiarizados con varias de estas tecnologías, han ayudado a que el desarrollo del proyecto haya podido realizarse cumpliendo las expectativas y objetivos nombrados anteriormente.

Durante el desarrollo del Trabajo de Fin de Grado, se ha podido poner en práctica diversas herramientas y metodologías aprendidas en la carrera de Ingeniería de Software, especialmente en asignaturas cómo Desarrollo de Aplicaciones Web (DAW), Diseño y Análisis de Algoritmos (DAA), Procesos del Software (PS) y Calidad del Software (CS).

En resumen, la elaboración de este proyecto ha significado una mejora considerable en los conocimientos y experiencia como ingeniero. El desarrollo de una aplicación web desde cero utilizando nuevas herramientas y metodologías ha brindado confianza y tranquilidad para enfrentar cualquier tipo de problema en el ámbito laboral e incluso personal. Por último, en el presente TFG o Trabajo de Fin de Grado, además de poder desarrollar una aplicación web que nos permite utilizar un posible nuevo método de autenticación para esta futura web 3.0, se ha desarrollado un interés particular por tecnologías cómo: *keyloggers*, *machine learning*, *blockchain*, FIDO y la web 3.0. Si bien todas estas tecnologías y términos son especulativos en la actualidad, al igual que la historia nos ha enseñado, debemos estar en constante aprendizaje y evolución. Aunque a menudo resistimos al cambio debido a que nos obliga a salir de nuestra zona de *confort*, estas tecnologías son necesarias tanto para la evolución de la web como para los usuarios que navegamos en ella.

VII. Bibliografia

[1] Mack, A. (2005). El concepto de seguridad humana. Papeles de cuestiones internacionales, 90, 11-18.

[2] Castelló, C. Salces, L. (2 de enero de 2022). ¿Cuál es la empresa más grande del mundo? Cinco Días. https://cincodias.elpais.com/cincodias/2021/12/30/companias/1640886339_354215.html

[3] Holgado, R. (6 de mayo de 2022). Así acabarán con las contraseñas Google, Microsoft y Apple para simplificar el inicio de sesión. 20bits. https://www.20minutos.es/tecnologia/ciberseguridad/asi-acabaran-con-las-contrasenas-google-microsoft-y-apple-para-simplificar-el-inicio-de-sesion-4995999/

[4] González Arrieta, M. A. (2018). Llaves FIDO (Fast IDentify Online) como segundo factor de autenticación en la gestión on-line de los procesos de enseñanza y aprendizaje.

[5] Telefónica, F. (2013). Identidad Digital: El nuevo usuario en el mundo digital. *Barcelona: Editorial Ariel. Recuperado de http://www. educando. edu. do/files/9513/9281/6433/identidad_digital. pdf*.

[6] Giones-Valls, A., & Serrat-Brustenga, M. (2010). La gestión de la identidad digital: una nueva habilidad informacional y digital.

[7] Chen, C., Zhang, L., Li, Y., Liao, T., Zhao, S., Zheng, Z., ... & Wu, J. (2022). When digital economy meets web 3.0: Applications and challenges. IEEE Open Journal of the Computer Society.

[8] Lopez Couso, A. (2023). quienEsQuien (Version 1.6). https://github.com/Adriton1/quienEsQuien

[9] Marquès, P. (1995). Metodología para la elaboración de software educativo. Barcelona (España). Editor. Estel.

[10] Hinestroza Ramírez, D. (2018). El Machine Learning a través de los tiempos, y los aportes a la humanidad.

[11] Pamela, Á. C. Inteligencia Artificial (Test de Turing). *La Paz, Bolivia*.

[12] Betancourt, G. A. (2005). Las máquinas de soporte vectorial (SVMs). *Scientia et technica*, *1*(27).

[13] Ramón, M. M. (2008). Introducción a los métodos Kernel. *Universidad Autónoma de Madrid*, *29*.

[14] Vidal-Silva, C. L., Sánchez-Ortiz, A., Serrano, J., & Rubio, J. M. (2021). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Formación universitaria*, *14*(5), 85-94.

[15] Ponsico Martin, P. (2017). Tecnología de Contenedores Docker (Bachelor's thesis, Universitat Politècnica de Catalunya).

[16] Django Software Foundation. (2019). *Django*. Obtenido de https://djangoproject.com

[17] Zabaljáuregui, M. Introducción al kernel Linux. *Argentina. Recuperado el*, *25*.

[18] Khan, S. S., & Madden, M. G. (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, *29*(3), 345-374.

[19] Wikipedia (2023). Confusion matrix. *Wikipedia*. https://en.wikipedia.org/wiki/Confusion_matrix

[20] Posadas Ruiz, G. C. M. (2022). *Modelo estadístico de predicción de desarrollo de diabetes mellitus tipo 2 en pacientes de alto riesgo aplicando Random Forest y SMOTE* (Doctoral dissertation, Industriales).

VIII. Anexo

A. Visualización de resultados en la base de datos

A.1 Tiempos del usuario en la prueba de recogida de datos para generar el modelo

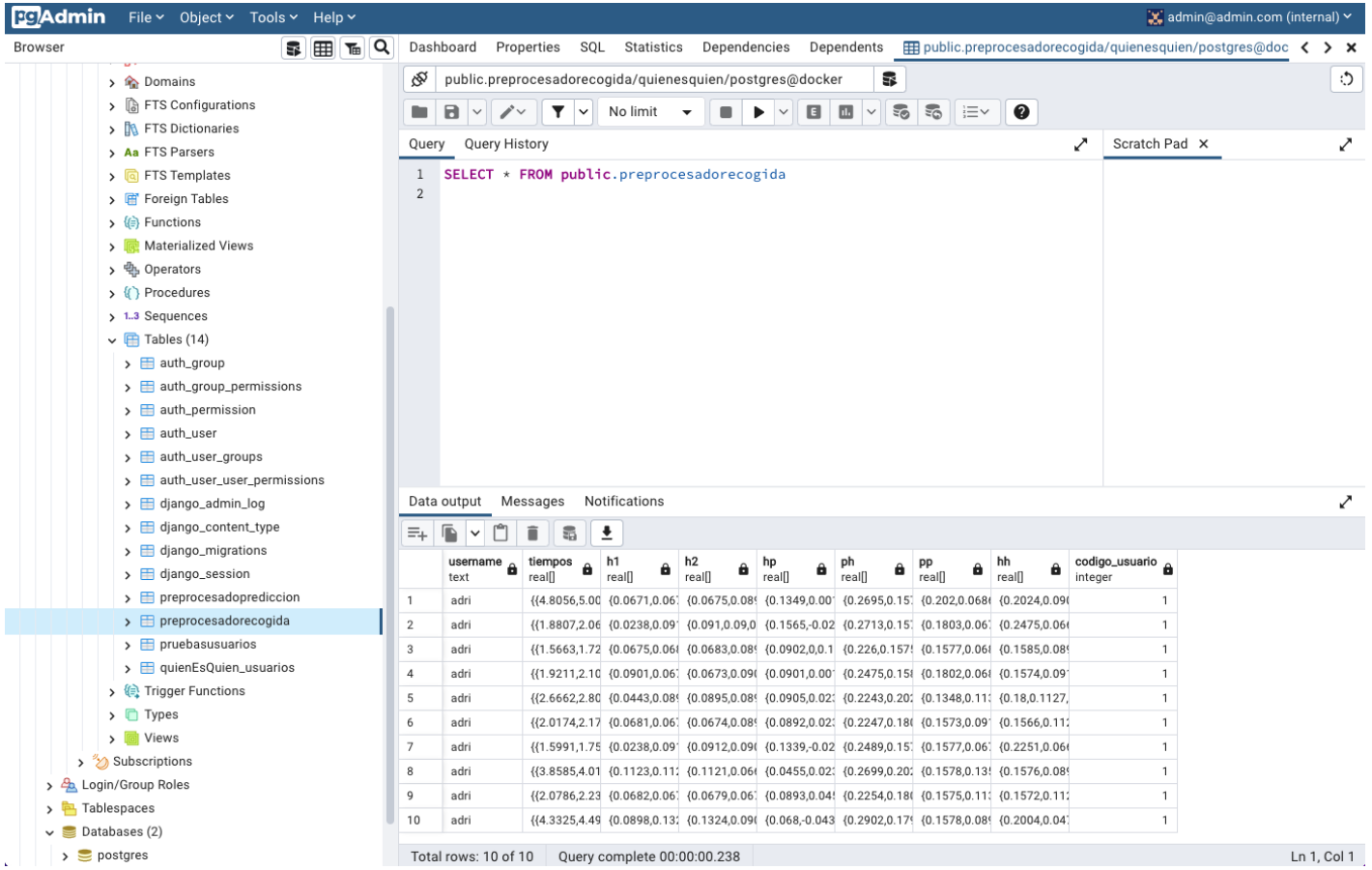

Ilustración 15: Tiempos del usuario en la prueba de recogida de datos para generar el modelo

A.2 Tiempos del usuario en la prueba de predicción junto al resultado de esta.

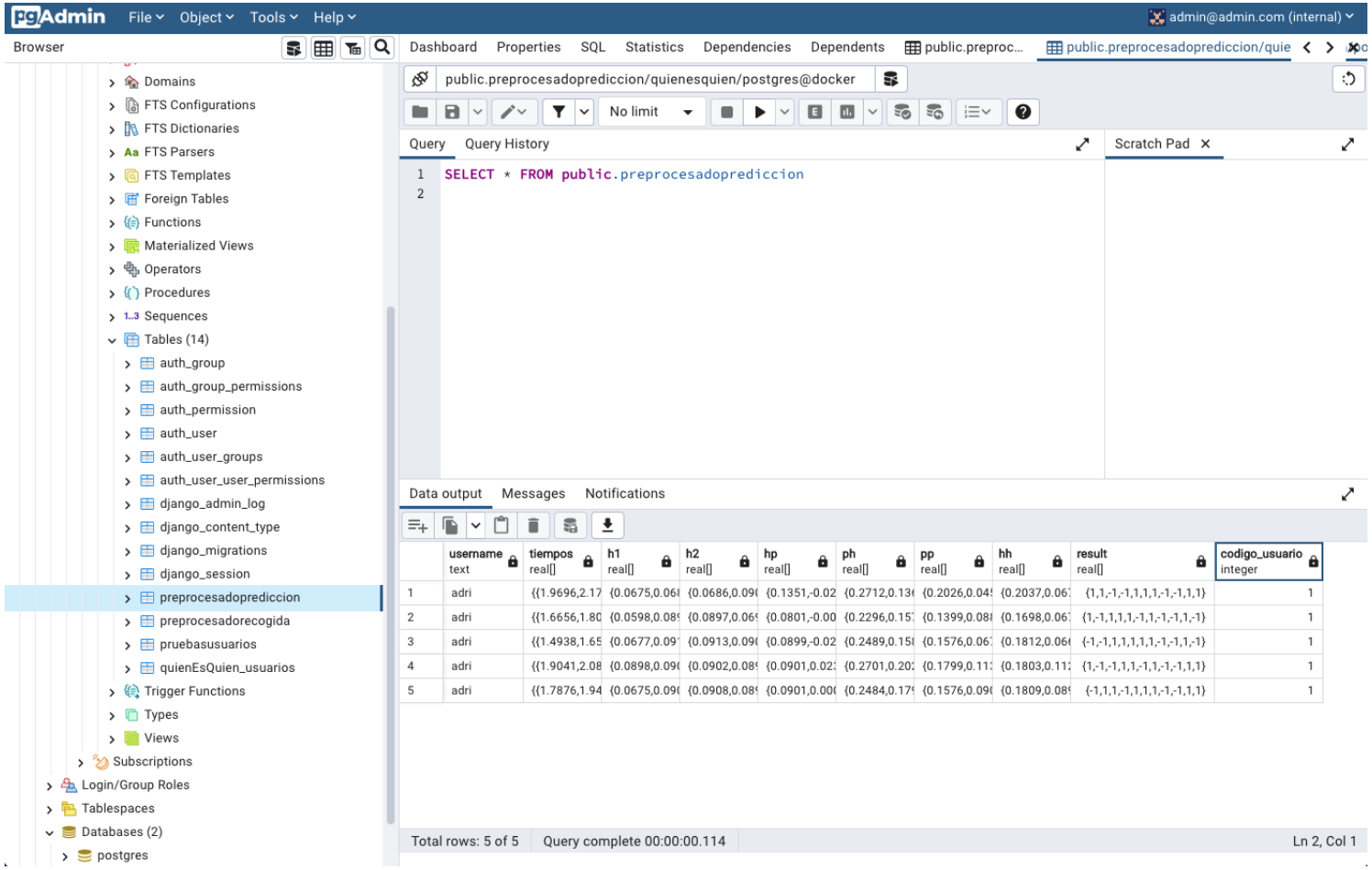

Ilustración 16: Tiempos del usuario en la prueba de predicción

B. Interfaz gráfica

B.1 Pestaña *Registro.*

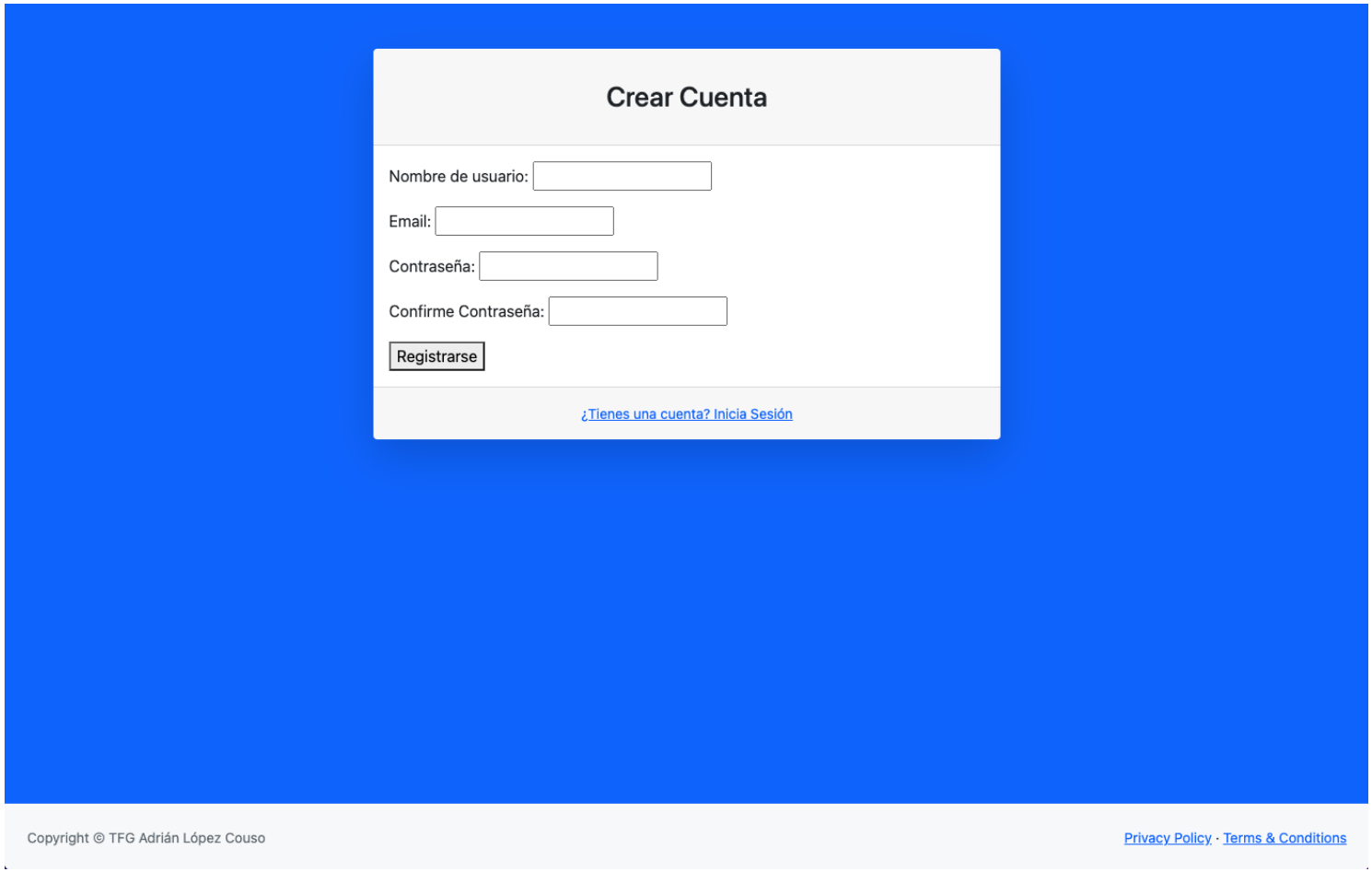

Ilustración 17: Pestaña Registro

B.2 Pestaña *Inicio de sesión*

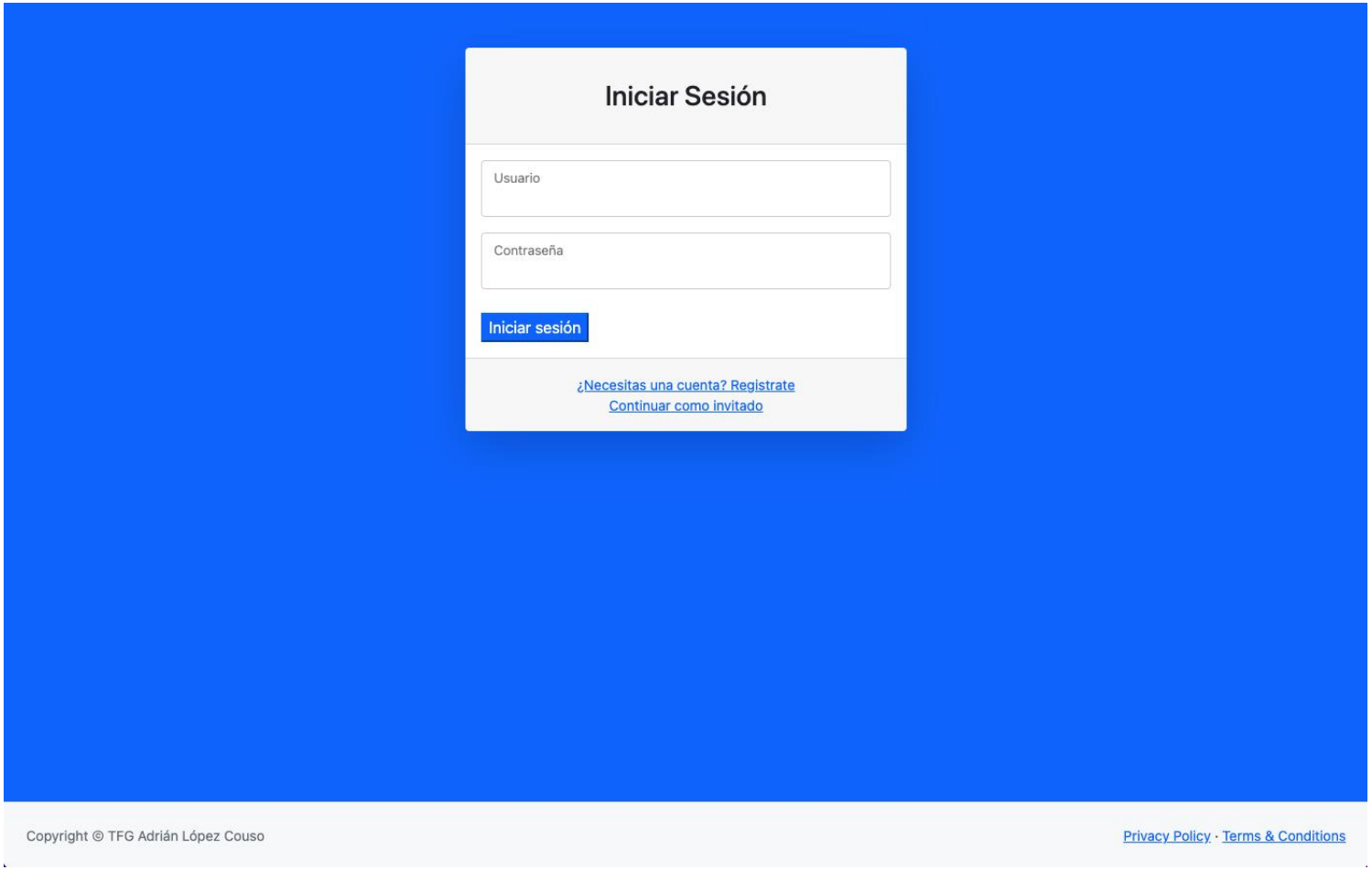

Ilustración 18: Pestaña Inicio de sesión

B.3 Pestaña *Error inicio de sesión*

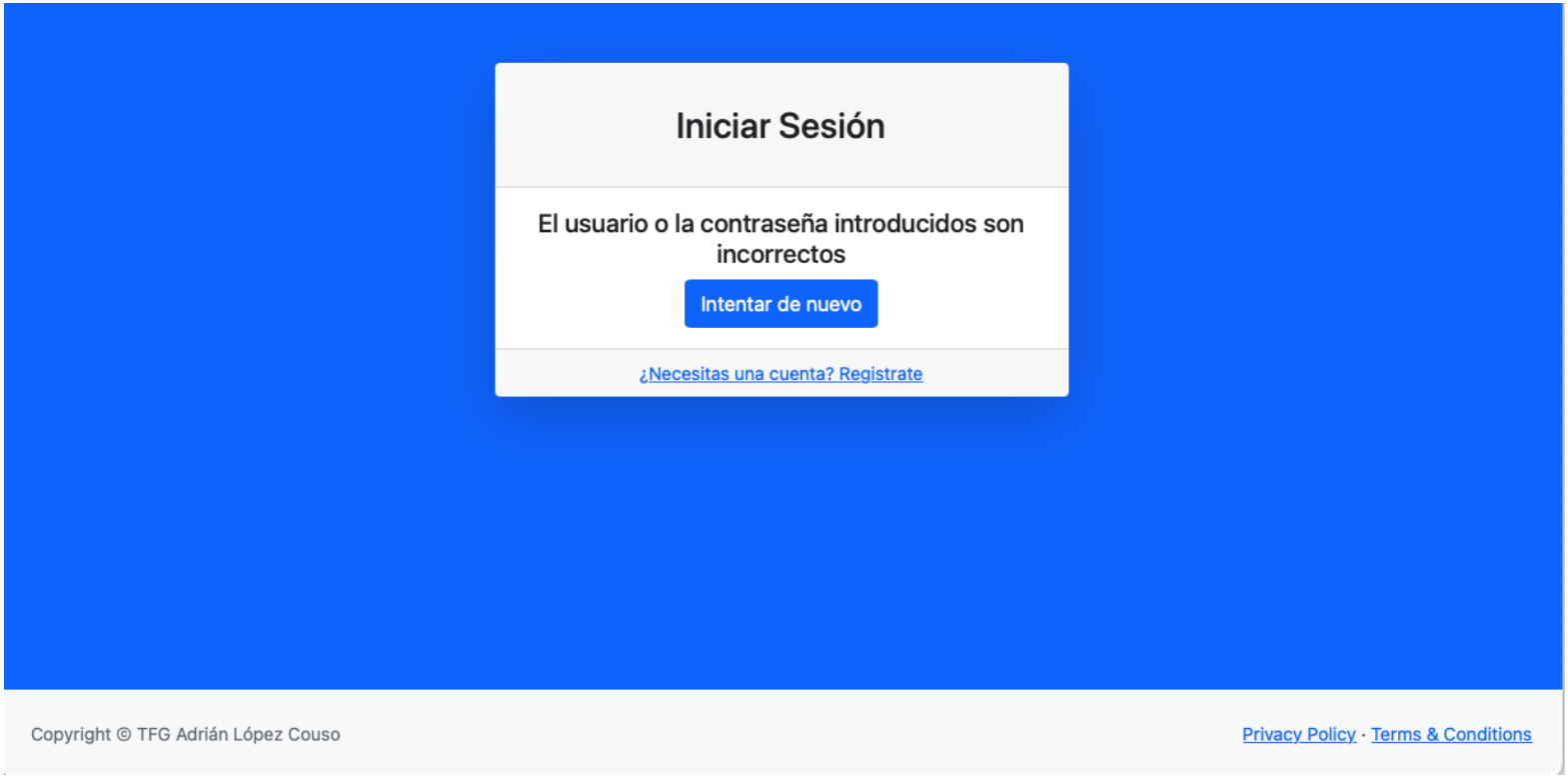

Ilustración 19: Pestaña de error al iniciar sesión

B.4 Pestaña *Introducción.*

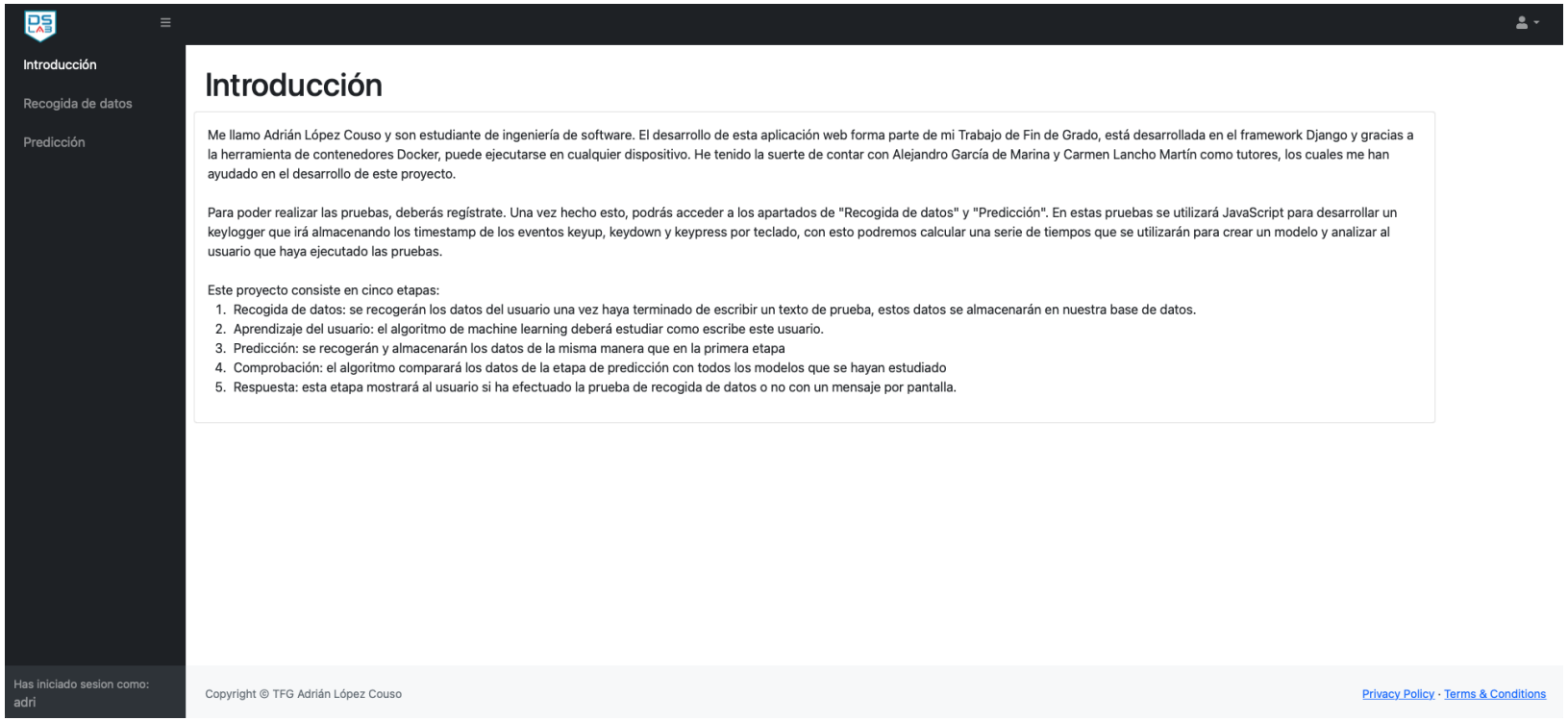

Ilustración 20: Pestaña Introducción

B.5 Pestaña *Introducción vista desde el usuario Invitado.*

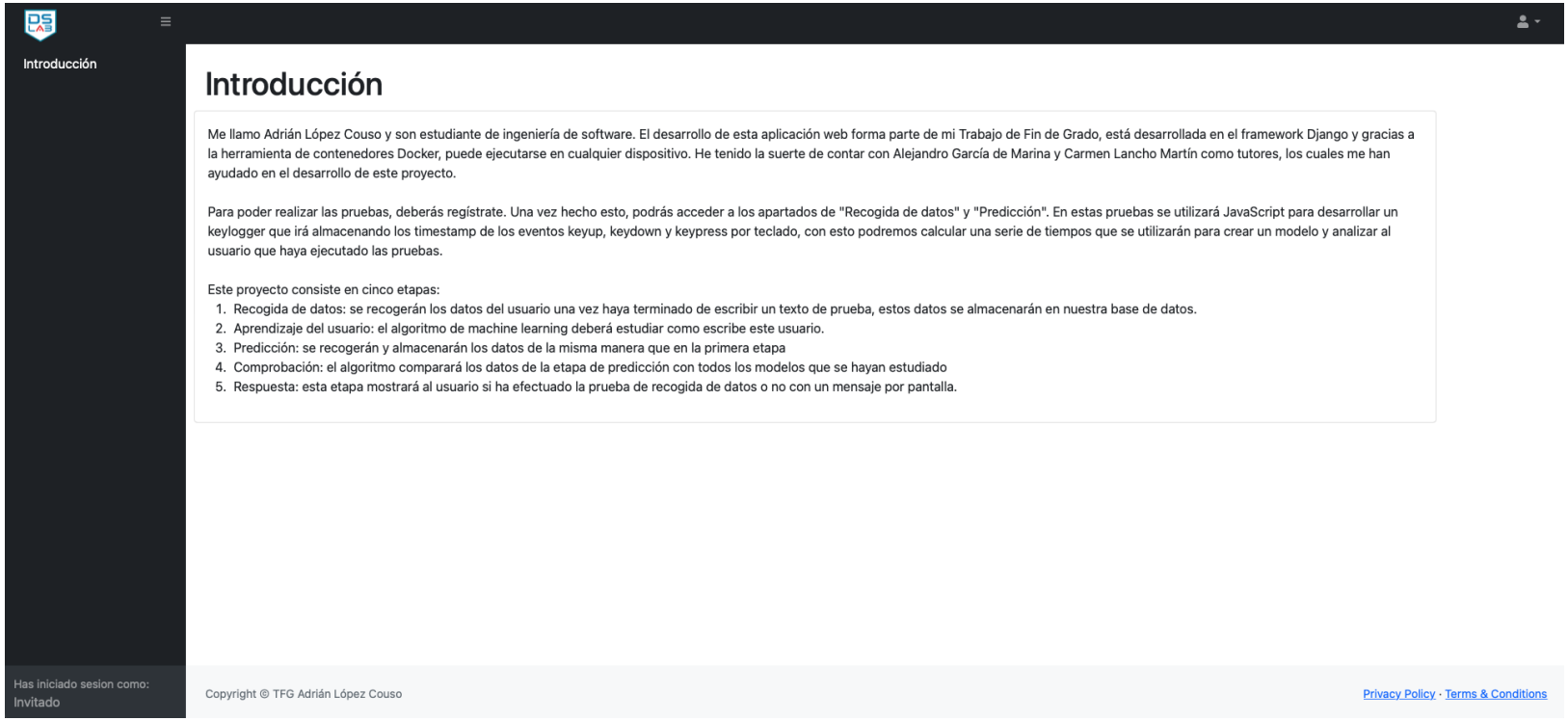

Ilustración 21: Pestaña Introducción vista desde el usuario Invitado

B.6 Pestaña *Recogida de datos.*

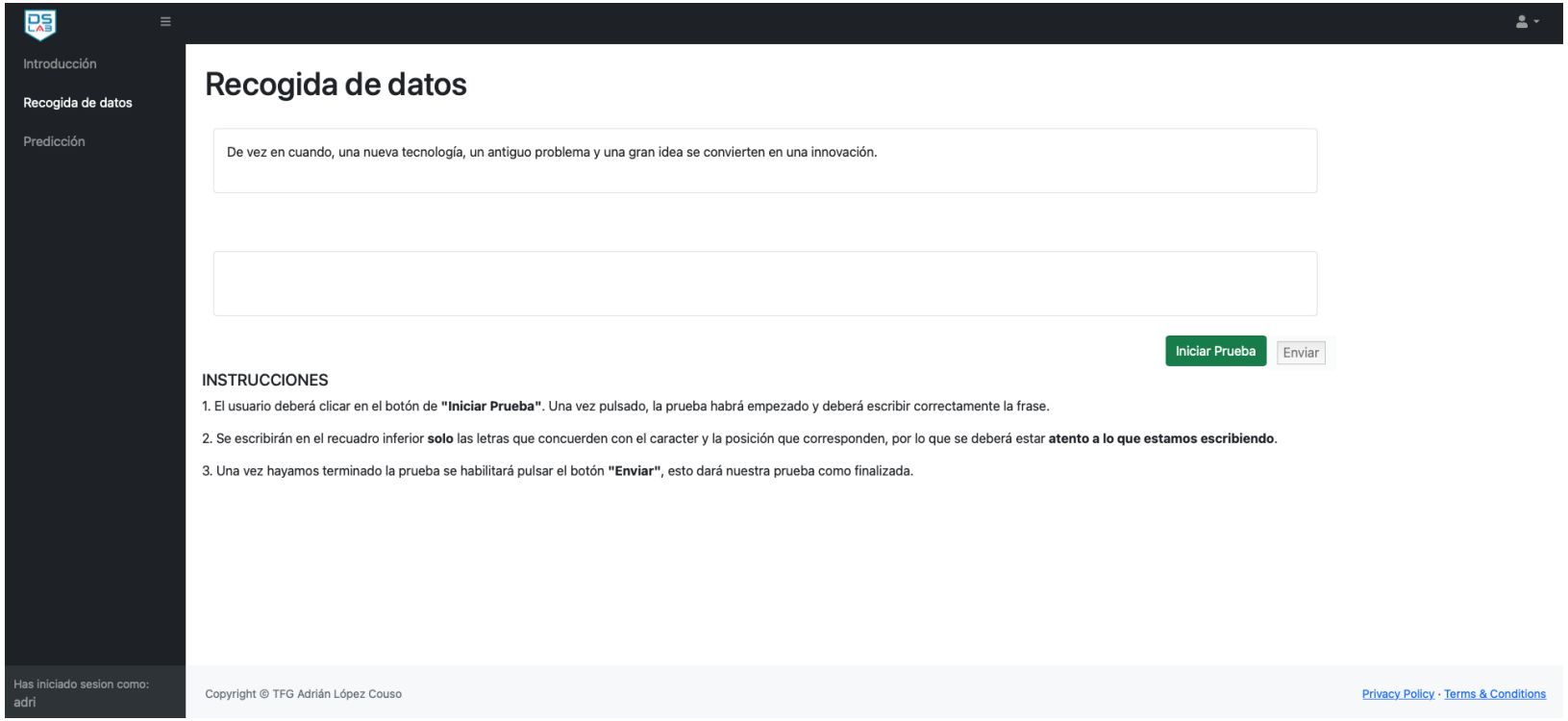

Ilustración 22: Pestaña Recogida de datos

B.7 Pestaña *Predicción.*

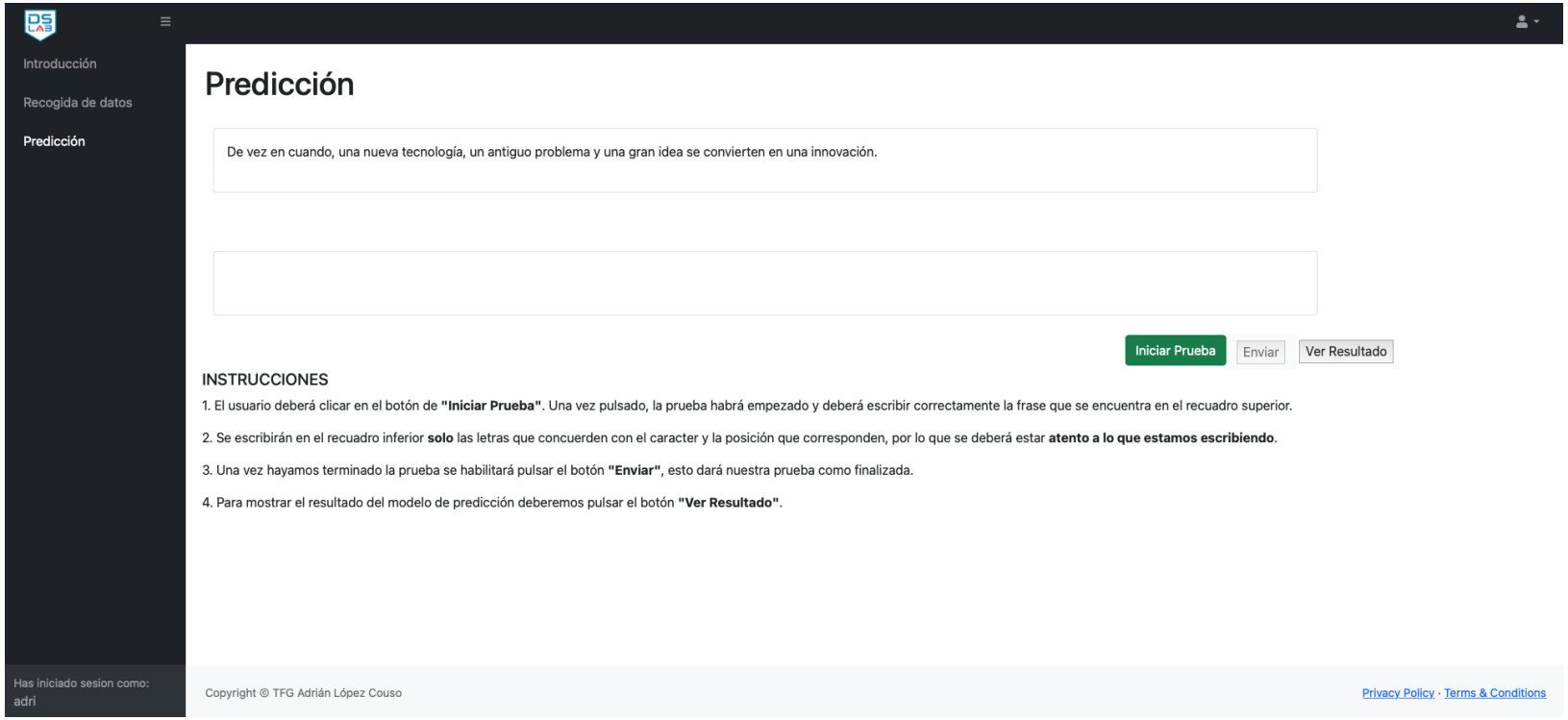

Ilustración 23: Pestaña Predicción

B.8 Pestaña *Predicción* tras realizar exitosamente la prueba.

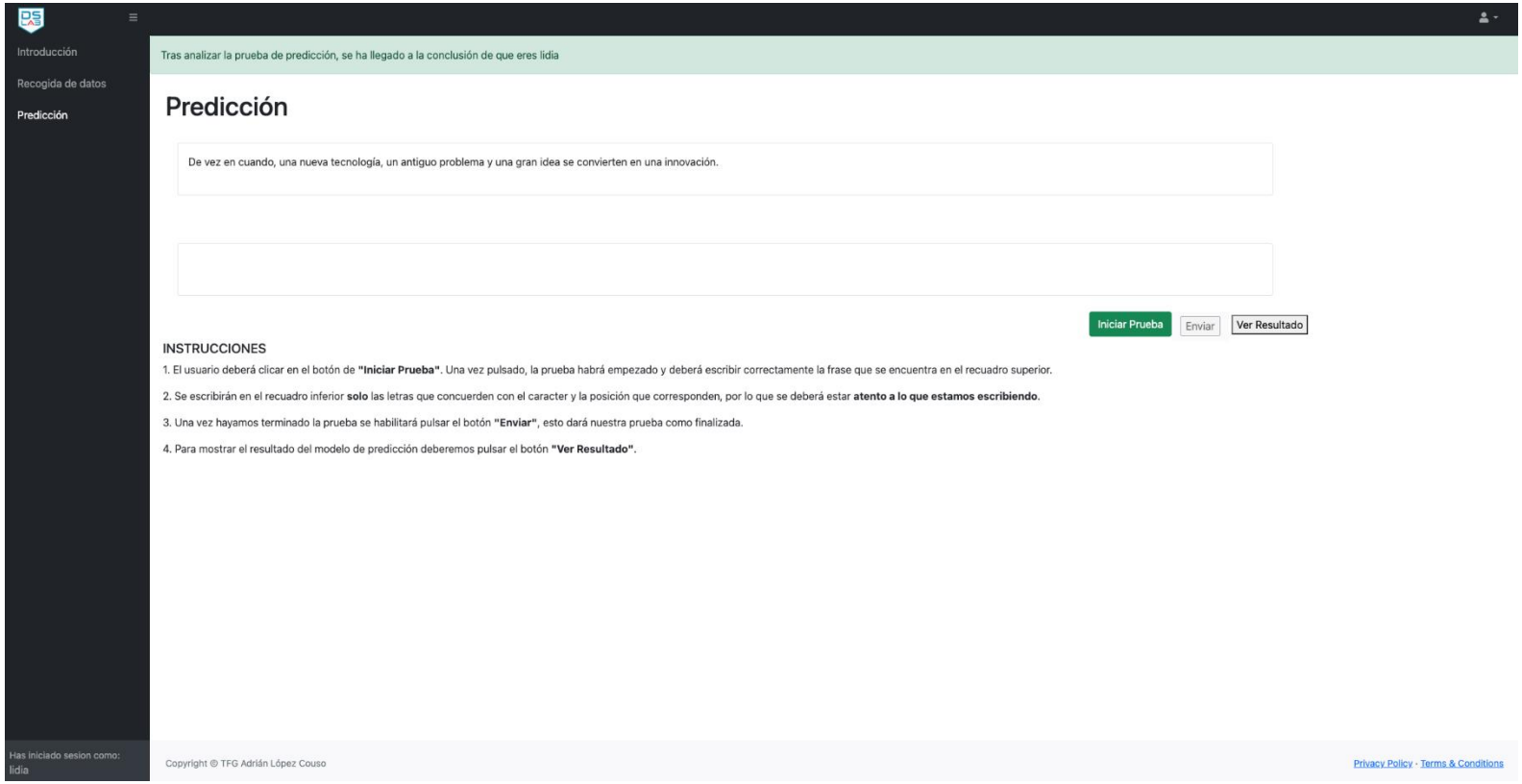

Ilustración 24: Pestaña Predicción tras realizar exitosamente la prueba

B.9 Pestaña *Predicción* tras realizar de manera fallida la prueba.

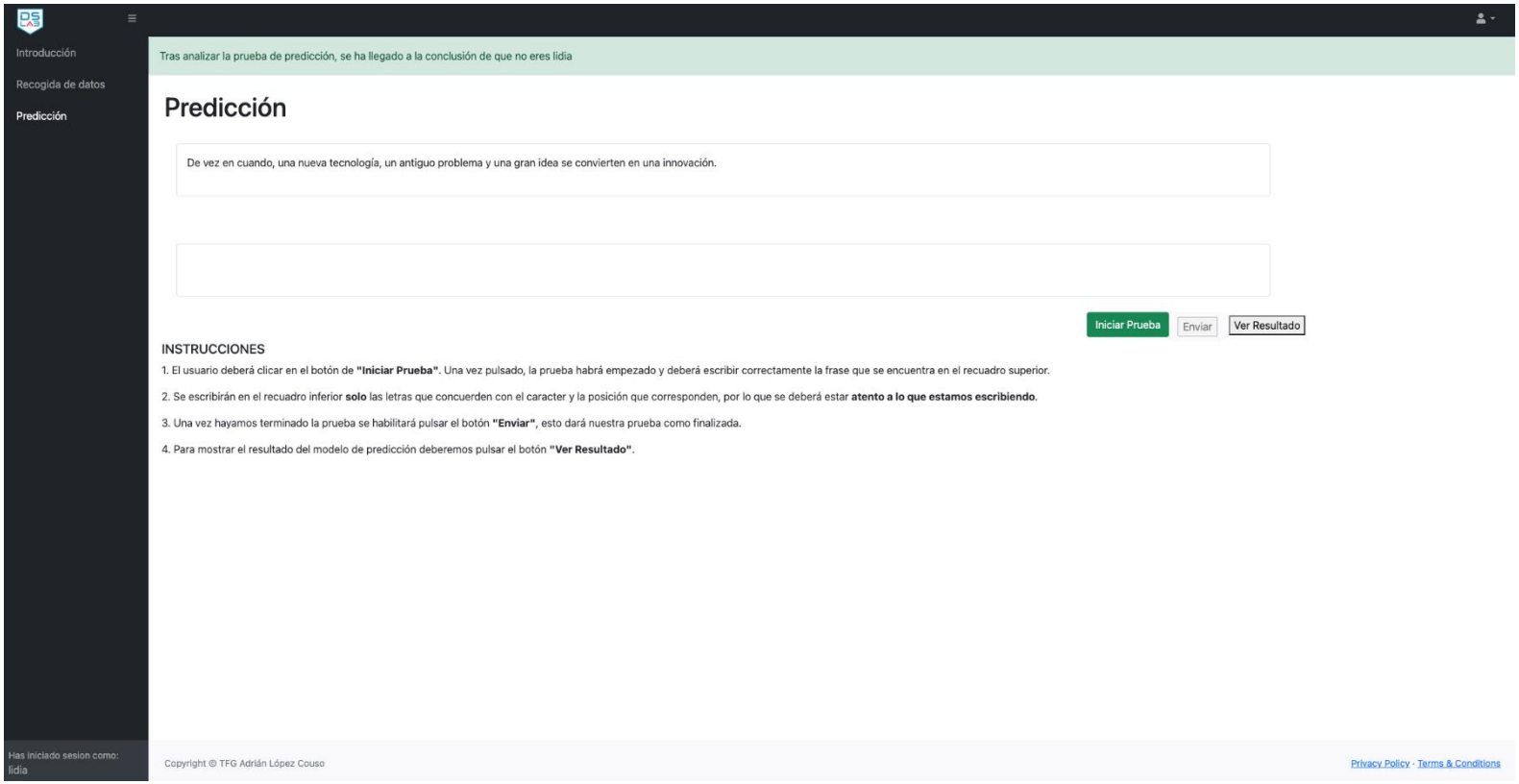

Ilustración 25: Pestaña Predicción tras realizar de manera fallida la prueba

B.8 Pestaña *login pgAdmin4.*

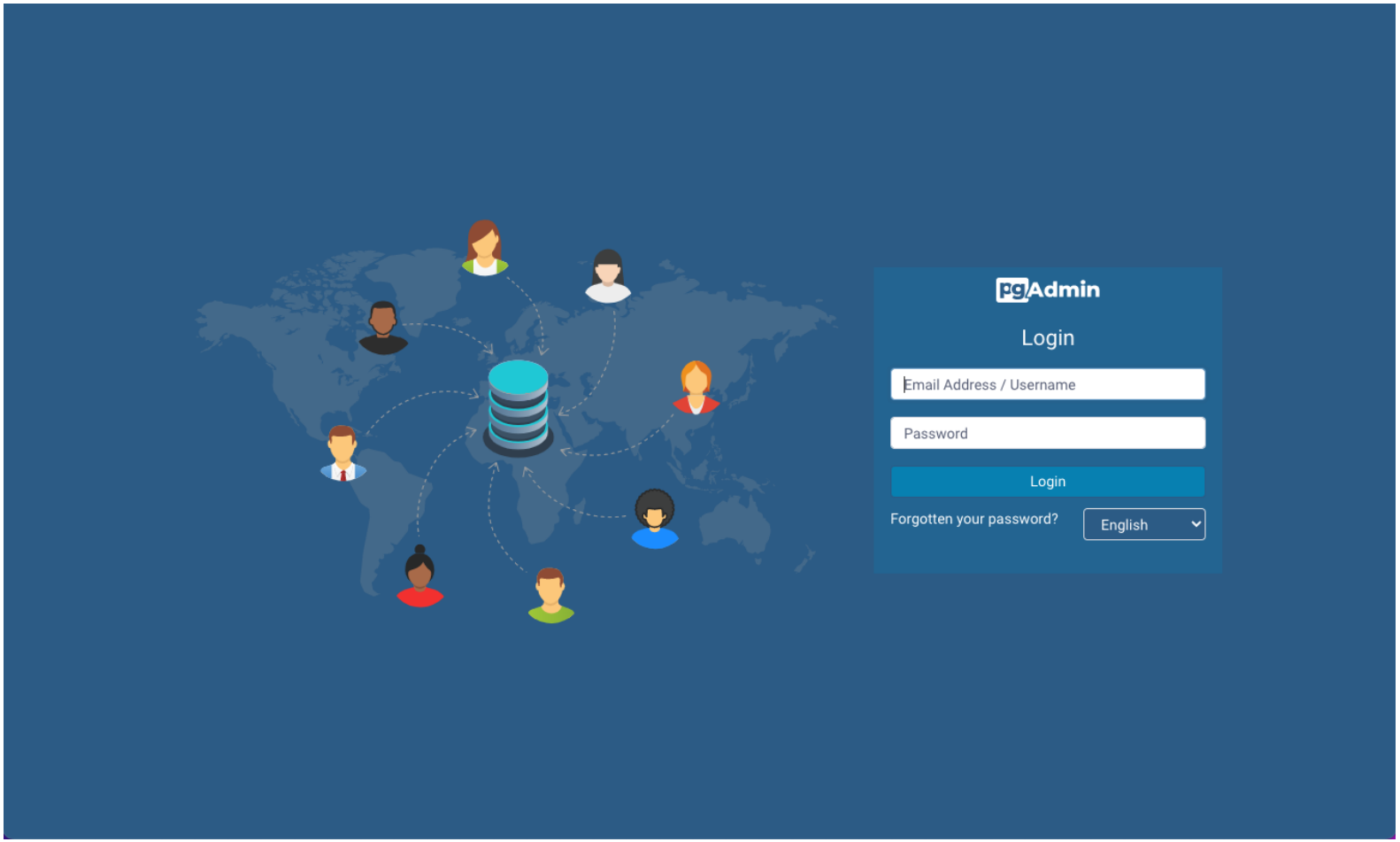

Ilustración 26: Pestaña login pgAdmin4

B.9 Interfaz *pgAdmin4*.

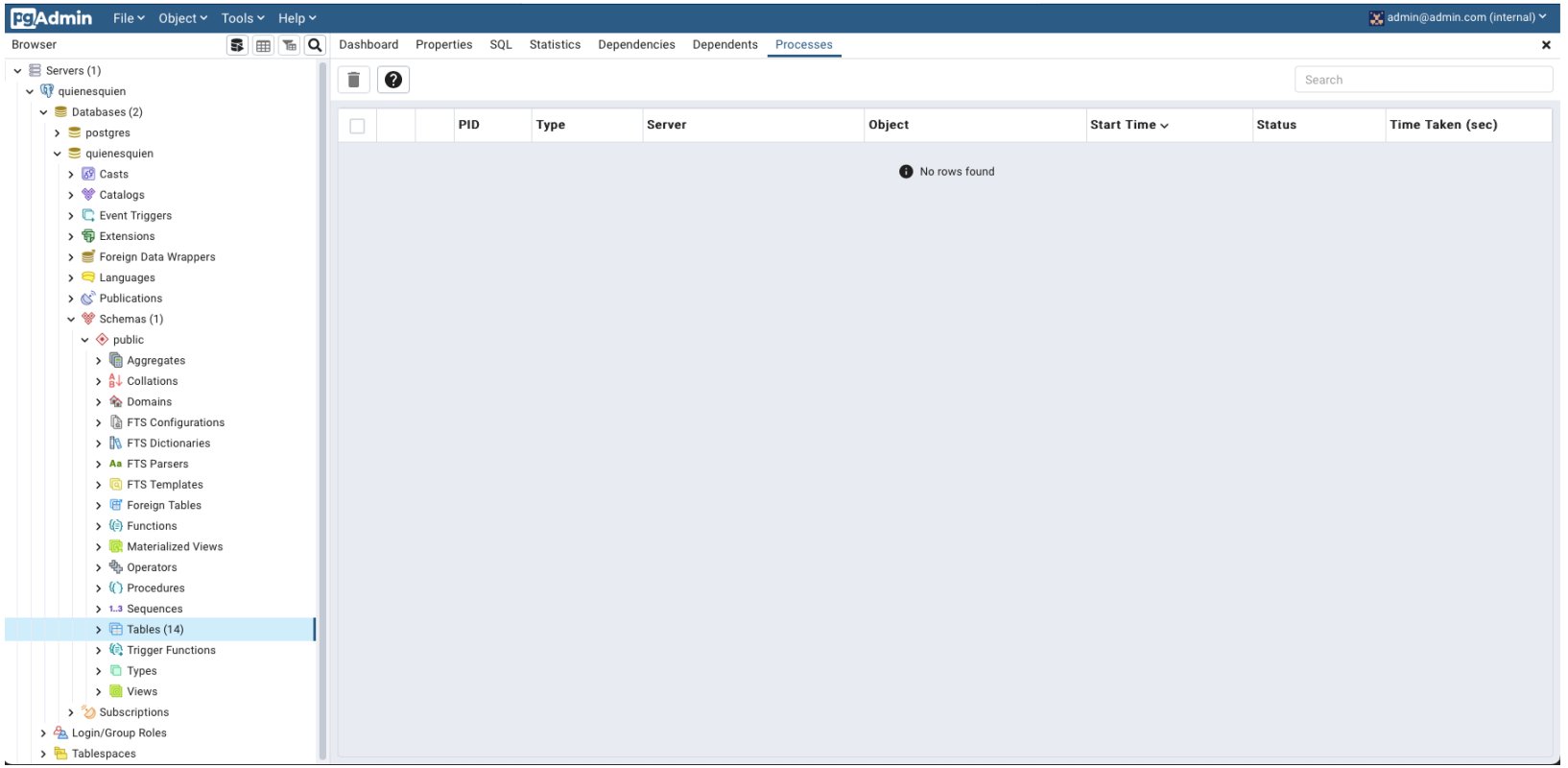

Ilustración 27: Interfaz pgAdmin4

B.10 Pestaña *login panel de administración.*

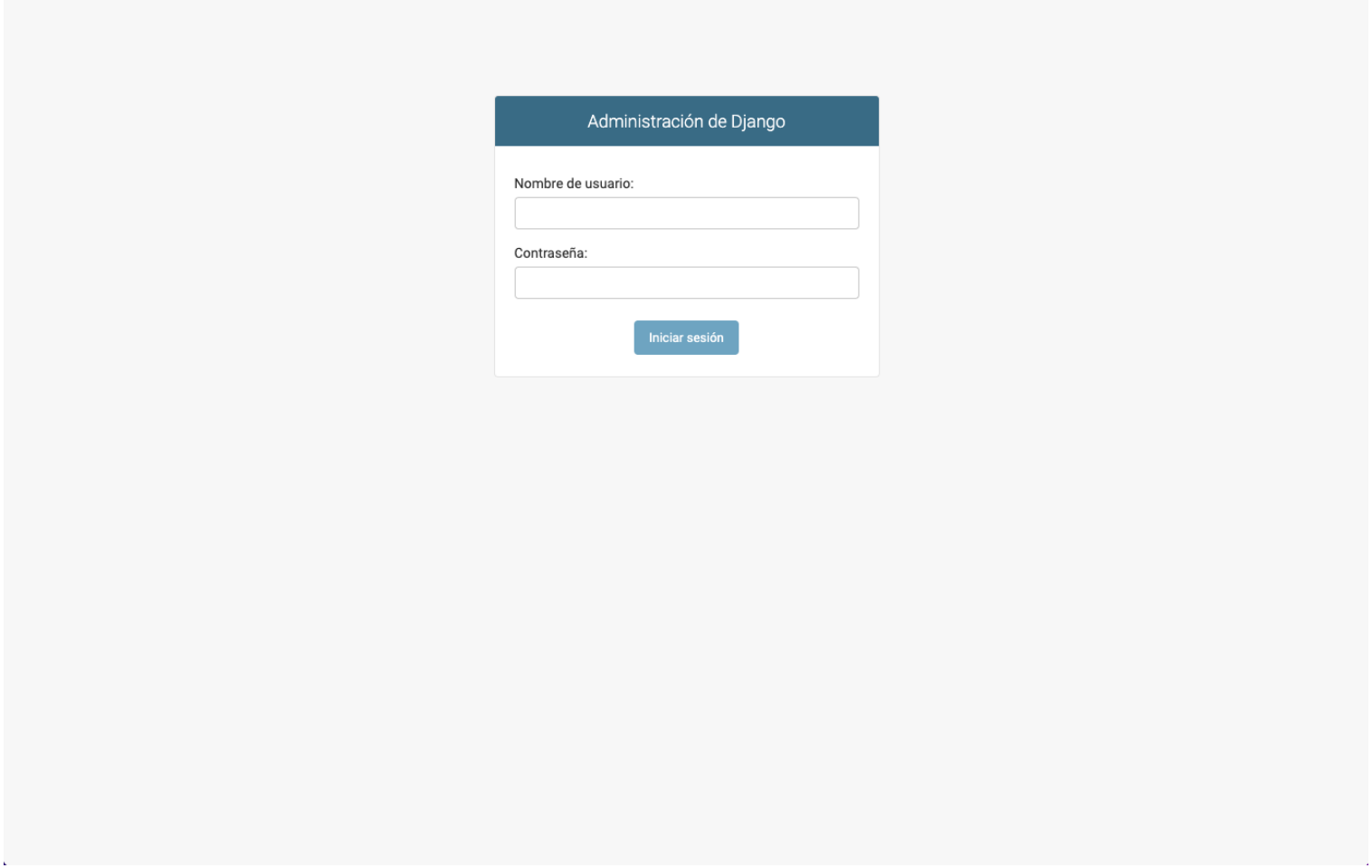

Ilustración 28: Pestaña login panel de administración

B.11 Pestaña *panel de administración.*

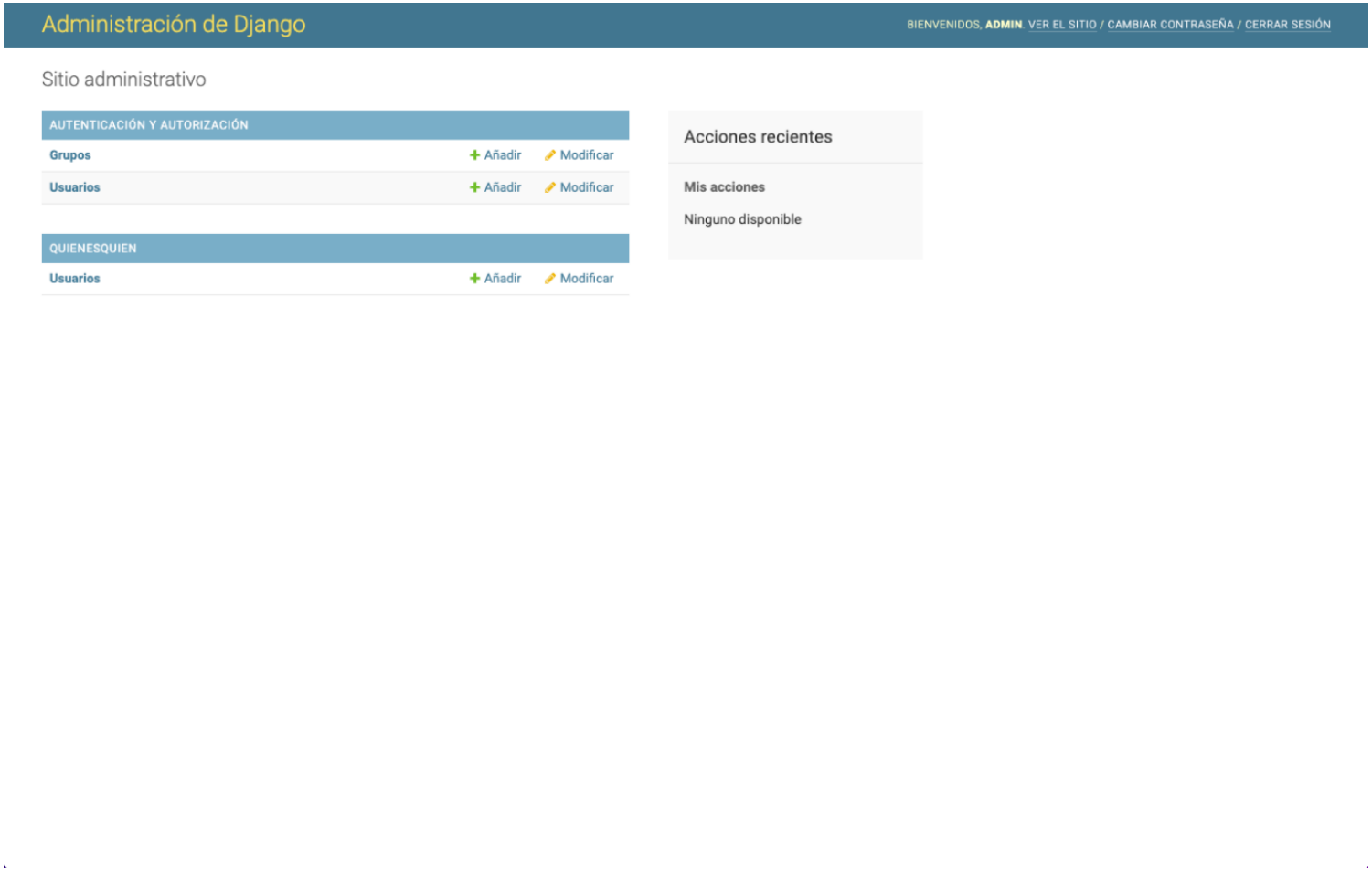

Ilustración 29: Pestaña panel de administración

B.12 Pestaña *usuarios panel de administración.*

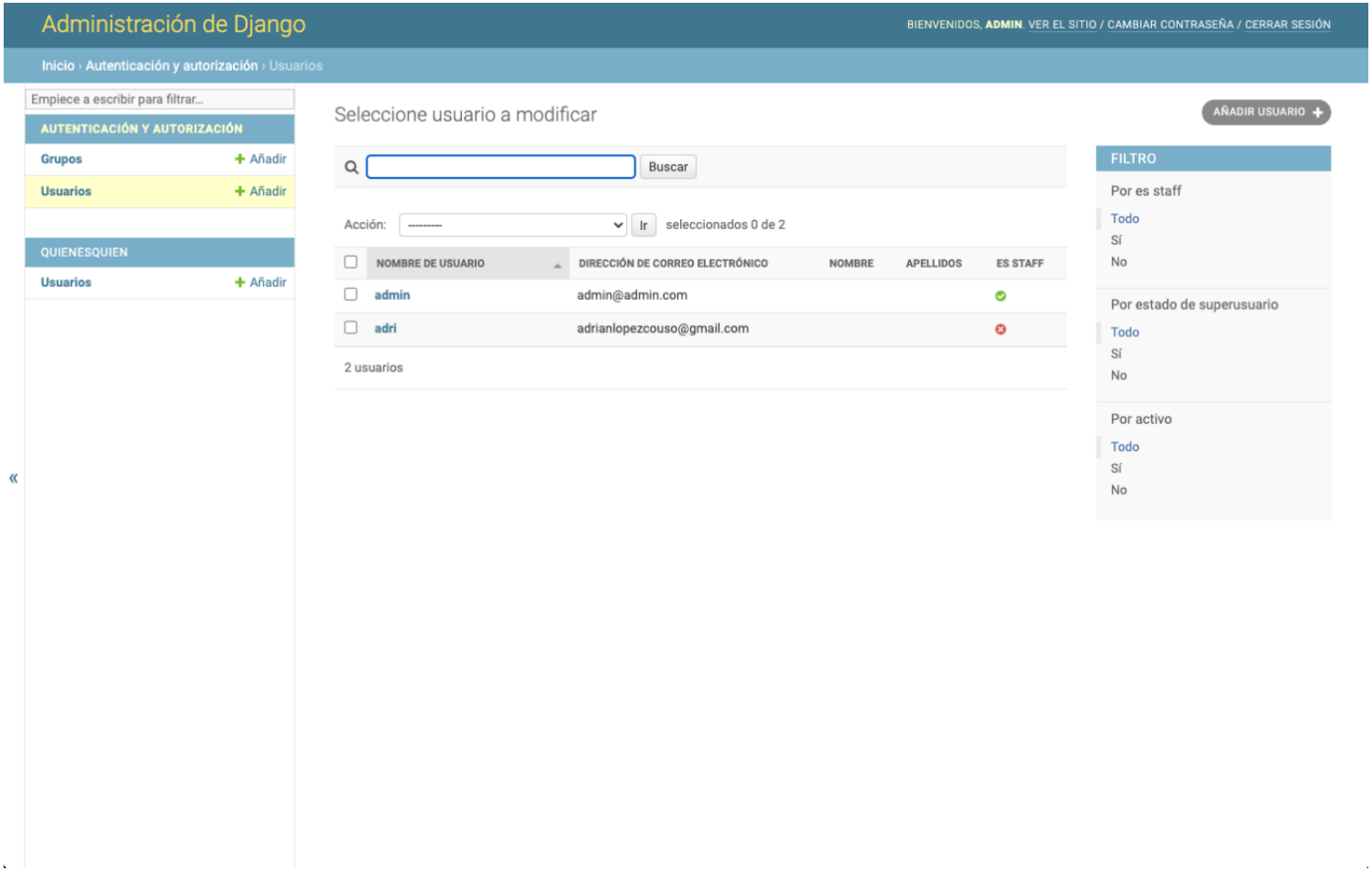

Ilustración 30: Pestaña usuarios panel de administración