TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

BankingFraud: backend NoSQL para detección de fraude en pagos

Autor: Adrián Arboledas Fernández

Grado: Ingeniería informática

Director: D. Miguel Ángel García Cumbreras
Departamento del director: Informática

Fecha: 29/05/2024

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Miguel Ángel Garcia Cumbreras , tutor del Proyecto Fin de Carrera titulado: "Banking Fraud: backend NoSQL para detección de fraude en pagos", que presenta Adrián Arboledas Fernández, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Mayo de 2024

El alumno:

ARBOLEDAS FERNANDEZ ADRIAN - 26523518X

Firmado digitalmente por ARBOLEDAS FERNANDEZ ADRIAN - 26523518X
Fecha: 2024.06.07 15:06:34 +02'00'

Adrián Arboledas Fernández

Los tutores:

GARCIA CUMBRERAS MIGUEL ANGEL - 52871272E

Firmado digitalmente por GARCIA CUMBRERAS MIGUEL ANGEL - 52871272E
Fecha: 2024.06.07 00:08:02 +02'00'

Miguel Ángel García Cumbreras

Agradecimientos

Agradecer a mi familia por apoyarme en este largo camino, por mantener siempre su confianza en mí y ayudarme a encontrarme a mí mismo en mis peores momentos, incluso cuando todo parecía no tener sentido.

Agradecer también a mis amigos, por hacer más ameno este gran recorrido por la universidad y por todos sus consejos, siendo un pilar fundamental en mi desarrollo como persona y estudiante.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1 Contexto y justificación

Cuando se habla de fraude bancario se estima que el 93% corresponde a la banca online. Kaspersky realizó un estudio en 2019, en el que estimó que el 2% de todas las transacciones bancarias online fueron realizadas por estafadores y el 16% eran sospechosas[1]. Estimando el FBI una pérdida anual de 2700 millones de dólares para Estados Unidos.

En este TFG se propone desarrollar un backend impulsado por aprendizaje automático de manera que se puedan clasificar las transacciones bancarias como válidas o fraudulentas, de forma automatizada antes de que se produzcan. De esta manera los clientes de las entidades bancarias estarán protegidos en tiempo real de acciones fraudulentas llevadas a cabo dentro de sus cuentas por ciberdelincuentes.

La justificación de este proyecto fue la necesidad de mejorar la forma en la que se detecta el fraude bancario. Ya que durante décadas se han utilizado métodos tradicionales basados en el análisis manual de datos, algo insostenible para el gran volumen de datos que se manejan diariamente.
En este sentido, una detección de fraude haciendo uso del aprendizaje automático nos permite manejar una gran cantidad de conjunto de datos con precisión.

Para entender la importancia de la detección de fraude a través del aprendizaje automático primero debemos conocer qué es la IA y el aprendizaje automático y en qué se diferencian.

La inteligencia artificial es un campo amplio que se refiere al uso de la tecnología para imitar funciones cognitivas asociadas con la inteligencia

[1] Recuperado de la página: https://kfp.kaspersky.com/es/news/una-de-cada-50-transacciones-en-linea-en-los-sectores-de-la-banca-y-el-comercio-electronico-fue-fraudulenta-en-2019/ el dia 14/08/2023

humana, como la capacidad de ver, entender el lenguaje hablado o escrito y responder a él, analizar datos, para así actuar para resolver un problema complejo.

Mientras que el aprendizaje automático es considerado como un subconjunto de la inteligencia artificial que le permite a una máquina o sistema aprender y mejorar de forma automática a partir de la experiencia. Es decir, en vez de una programación explícita, hace uso de algoritmos para analizar grandes cantidades de datos, aprender de las estadísticas y tomar decisiones fundamentadas.

1.2 Objetivos y alcance del proyecto

El objetivo principal de este TFG es detectar las posibles transferencias fraudulentas que pueden producirse hoy en día en el sector de la banca digital (Phishing, Pharming, Skimming…). La idea es generar un backend impulsado por el machine learning que nos permita detectar en tiempo real cuando se produce una transacción fraudulenta**.**

Y para cubrir este objetivo principal , desde el punto de vista funcional se establecen los siguientes objetivos específicos:

1. **Gestión de usuario:** Permite a los usuarios, poder registrarse en la API, para poder acceder a la funcionalidad de la misma. Asimismo, brinda la opción de cerrar sesión en sus cuentas.

2. **Entrenamiento de modelos:** Ofrece a los usuarios la posibilidad de entrenar un modelo de datos específico a sus necesidades, de manera que pueda adaptarse de la mejor manera a su problema.

3. **Predicción de transacciones fraudulentas:** Permite a los usuarios el poder realizar predicciones sobre transacciones de forma personalizada, ya que pueden elegir entre cualquiera de los modelos entrenados anteriormente.

4. **Listado de modelos entrenados:** Ofrece a los usuarios la posibilidad de poder listar los modelos entrenados, para poder así elegir cual de ellos poder utilizar para predecir transacciones, o si desean eliminar un modelo que ya no le sea de utilidad.

5. **Gestión de modelos:** Permite a los usuarios, eliminar un modelo en concreto, que ya no le resulte de utilidad, ó si desea reentrenar un modelo con una nueva fuente de datos.

1.3 Análisis fraude bancario

Fraude bancario hace referencia al conjunto de prácticas ilegales para la obtención de dinero o activos de un banco u otra institución financiera. Este tipo de fraude ha existido desde el origen de la banca, pero cuya dimensión se ha visto exponencialmente aumentada con la aparición de la banca online.

1.3.1 Fraude bancario online

En esta sección se van a comentar los procedimientos llevados a cabo por los ciberdelincuentes, para realizar los distintos tipos de fraude bancario online.

Phishing

Es uno de los métodos más usados, esto se debe a que no requiere saltarse la capa de protección de los sistemas informáticos. El atacante solo se encargará de engañar a las personas haciendo uso de la ingeniería social, para que compartan información confidencial como contraseñas y números de tarjetas de crédito.

Según Adam Kujawa, Director de Malwarebytes Labs[2], "El phishing es la forma más sencilla de ciberataque, y al mismo tiempo las más peligrosa y efectiva. Eso se debe a que ataca el ordenador más vulnerable y potente del planeta: la mente humana."

Figura 1: Ilustración phishing

Tipos de ataques

- **Spear phishing:** Es un tipo de ataque dirigido a una persona u organización en específico. De manera que se requiere un reconocimiento previo, con el fin de descubrir nombres, cargos y poder realizar un ataque personalizado.
  Según un informe realizado por la revista "InfoSecurity Magazine"[3], es responsable del 38% de los ciberataques a empresas. Teniendo un coste medio de 1.8 millones de dólares por incidente.

- **Phishing de clonación:** En este tipo de ataque, los delincuentes hacen una copia de correos electrónicos legítimos enviados anteriormente que contenían enlaces o archivos adjuntos, sustituyendo

[2] Recuperado de la página: https://es.malwarebytes.com/phishing/ el día 14/08/2023
[3] Recuperado de la página : https://www.infosecurity-magazine.com/news/average-cost-of-a-spear-phishing/ el dia 18/08/2023

estos por contenido malicioso. De modo que cuando un usuario hace click, permite al atacante tomar control sobre sus sistemas.

- **Phishing telefónico:** El atacante se hace pasar por personal de su banco. Tras ganarse su confianza, le relata que ha ocurrido un problema y que deben de solucionarlo inmediatamente, asustandolos. De esta manera el usuario aporta sus datos bancarios al supuesto agente, para zanjar el problema.

Identificación

No siempre es sencillo reconocer un intento de phishing, pero estas son algunas de las señales que ayudan a su identificación.

- **Reconocer al remitente:** Sospechar si es alguien con quien normalmente no trata, o cuyo contenido del correo electrónico no coincide con sus responsabilidades laborales habituales.

- **Mensaje aterrador:** Se debe de sospechar cuando un correo utiliza mensajes alarmistas, para crear una sensación de urgencia, incitando a hacer click.

- **Enlaces extraños:** No se debe de asumir que los hiperenlaces llevan a donde parece, ya que pueden redirigir a una página web creada por el atacante, que imite la original, para obtener los datos de los usuarios que intenten iniciar sesión en la misma.

Pharming

Es un tipo de fraude donde el atacante, redirige el tráfico de una página web legítima a sitios falsos, que asemejan a los sitios web de los bancos. Aprovechándose de cómo los navegadores convierten las URL a direcciones IP. Con el fin de recopilar datos personales, como su DNI y contraseña de la cuenta bancaria.

Al igual que ocurre con el phishing, este método evita tener que eludir las protecciones de los sistemas informáticos al hacer hacer uso de la ingeniería social consiguiendo engañar a los usuarios.

Identificación

- **Fijarse en la URL de la web:** Es necesario identificar si esta, empieza por http o https, ya que aquellas páginas web que gestionan información personal, protegen tus datos con conexión https, siendo las conexiones http inseguras.

  Debe de tener en cuenta el dominio de la misma, ya que muchas páginas falsas, pueden añadir guiones o alterar algún carácter a la url original, para ser redirigidos a la página del atacante.

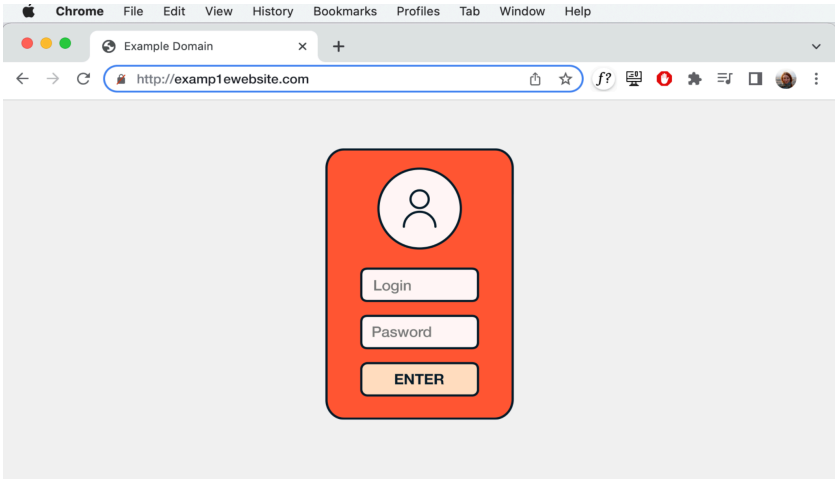


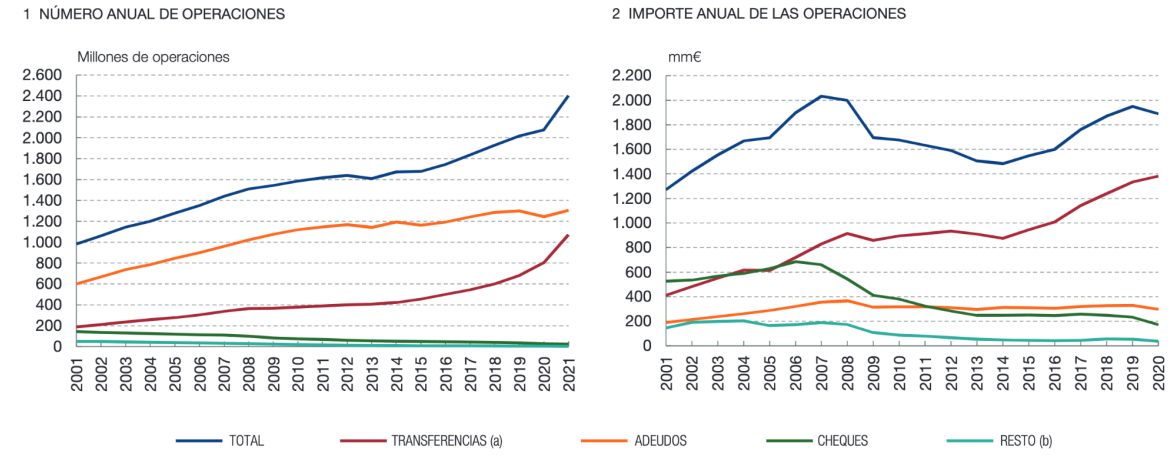
Figura 2: Ilustración pharming

- **Examinar la página web:** Observar si está mal formada o si la posición de los botones resulta extraña. Si hay alguna sospecha podría tratarse de una página falsa.

Fraude de transferencia bancaria

Las transferencias bancarias son uno de los métodos más usados en la actualidad para transferir dinero de una persona a otra llegando a una media de 1000 millones de operaciones anuales solo en España. Con una tasa de fraude en 2021 de tan solo 0.023%.

Según el Banco nacional de España[4], se estima que en términos de importe monetario las transferencias abarcan un 75% del total.



FUENTE: Banco de España, a partir de los datos de Iberpay.

Figura 3: Gráficas evolución transacciones. Fuente: Banco de España

Tipos transferencias fraudulentas

En esta sección se comentarán los tipos de transferencias bancarias fraudulentas más usuales.

- **Te piden que realices un pago a un tercero:** Destinada a profesionales que trabajan por cuenta propia. Donde un supuesto cliente les contacta para que realicen un trabajo. El problema reside en

[4] Recuperado de la página : https://www.bde.es/f/webbde/Secciones/Publicaciones/PublicacionesAnuales/MemoriaSupervisionBancaria/21/MemoriaSupervision2021_Cap5.pdf el dia 20/08/2023

que el supuesto cliente le insiste al profesional para que pague a un tercero para la obtención de los materiales.

Aportándole el dinero al profesional mediante transferencia bancaria. Pero cuando el profesional realiza dicha transferencia el dinero del cliente no aparece reflejado en su cuenta bancaria, ya que este pudo haber cancelado la transferencia o haber denunciado la tarjeta de crédito como robada.

- **Transferencias ilegítimas:** En este el atacante hace uso de un malware que suplanta la identidad de tu banco habitual. Una vez dentro aparece un mensaje de error, donde asegura que has recibido una transferencia errónea y que debes devolver el dinero.

  En el caso de picar, se realizará una transferencia ilegítima con cargo a tu cuenta.

1.3.2 Fraude bancario offline

En esta sección se van a comentar los procedimientos llevados a cabo por los delincuentes, para realizar los distintos tipos de fraude bancario offline.
Este tipo de fraude es mucho menos frecuente que el online, por el riesgo de exposición del atacante.

Skimming

El skimming es un método que consiste en extraer los datos de la tarjeta bancaria en un punto de venta o cajero, con el fin de poder replicar dichas tarjetas.

Tipos Skimming

En esta sección se comentarán los tipos de skimming más usuales.

- **Captura de datos NFC:** Los datos NFC se envían a través de ondas. Estas pueden ser interceptadas, si un dispositivo móvil u otro lector, se encuentra cerca del aparato emisor.

- **Captura de datos de superposiciones:** Es uno de los métodos más antiguos utilizados en los cajeros automáticos. Se superpone un lector de tarjetas adicional y un teclado, todo esto de manera imperceptible para el usuario del cajero. De esta manera el atacante consigue obtener los datos de la tarjeta y el PIN de la misma.

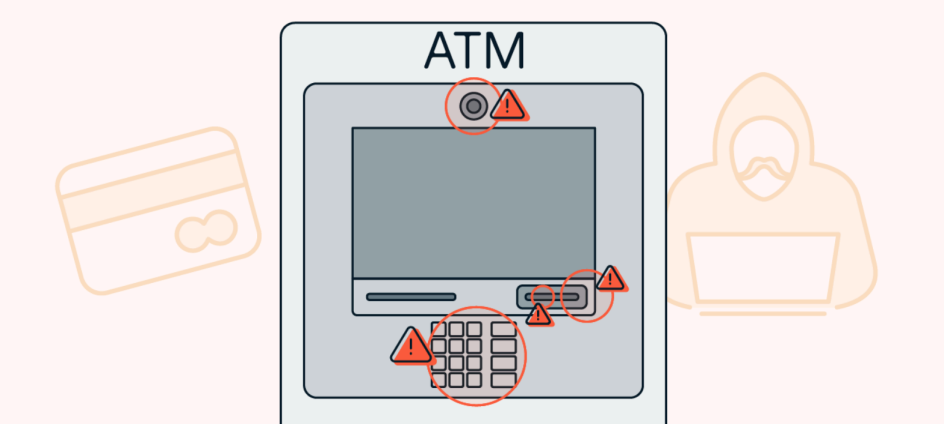


Figura 4: Ilustración skimming

Redireccionamiento de correspondencia

El redireccionamiento de correspondencia, consiste en que el atacante pueda modificar la dirección de correspondencia para recibir los estadios de la cuenta y tarjetas de crédito a una dirección controlada por ellos.

Los métodos más frecuentes utilizados por los atacantes, para modificar la dirección del usuario son los siguientes.

- **Suplantación de identidad:** Usando la información personal que poseen del usuario, estos pueden visitar una entidad bancaria, para hacerse pasar por la víctima y solicitar un cambio de dirección.

- **Colaboración interna:** En algunos casos, los atacantes, pueden contar con la colaboración de empleados internos de la entidad bancaria, que les ayuden a realizar el cambio en la dirección.

1.3.3 Métodos para detectar el fraude bancario

Debido al gran aumento en el número de fraudes bancarios, las entidades se han visto obligadas a adoptar medidas para la detección de fraudes.

Estos se encargan de prevenir, detectar y mitigar las actividades fraudulentas con el objetivo de proteger los activos de los clientes y guardar la integridad de las transacciones fraudulentas.

Dentro de los métodos de detección de fraude, podemos encontrar dos alternativas, los tradicionales, basados en métodos estadísticos y los de detección automática, basada en la inteligencia artificial y machine learning.

Características Recomendadas

- **Puntuación de fraude:** La puntuación de fraude, se encarga de examinar cada transacción en función de docenas de indicadores para luego asignar una puntuación numérica simple que representa el nivel de riesgo de la transacción.

- **Aprendizaje automático:** El aprendizaje automático, compara la información entrante con los datos históricos antes de aprobar una transacción. Haciendo uso de algoritmos sofisticados, para analizar

los resultados y "aprender" de las nuevas entradas, lo que le permite tomar decisiones más precisas con el tiempo.

- **Monitoreo en tiempo real:** Herramienta encargada de monitorear y responder activamente a los pagos. Implementada normalmente junto a la verificación de identidad automatizada para decidir si rechazar o aprobar transacciones en función de las puntuaciones de riesgo.

- **Informes detallados:** La generación de informes con indicadores de rendimiento, son fundamentales para comprender que funciona, que no funciona y donde poder optimizar los procesos y eliminar errores.

- **Conjunto de reglas personalizadas:** Un buen sistema de detección de fraude, permitirá a los negocios crear conjuntos de reglas para aprobar o rechazar pedidos en función de parámetros que se adapten a las necesidades únicas de cada empresa.

Métodos tradicionales de detección

Los métodos tradicionales, basados en el análisis manual de datos, se han utilizado durante décadas desde tiempos anteriores a los ordenadores, siendo estos aún relevantes en la actualidad, aunque siendo ampliamente reemplazados por los métodos de machine learning.

Tipos de métodos tradicionales

- **Muestreo:** El muestreo, implica elegir un conjunto de datos representativo de las transacciones de forma aleatoria. Este método permite al banco obtener una visión general de la posible presencia de fraude dentro de una población más grande sin tener que revisar cada transacción o cuenta de forma individual.

- **Análisis Ad hoc:** El análisis Ad hoc es un método de detección de fraude bancario en respuesta a eventos particulares, es decir, se centra en casos individuales o en áreas específicas que se consideran sospechosas o de alto riesgo.
  Los analistas, se encargan de estudiar las transacciones en profundidad y determinar si existe un posible fraude.

- **Ley de Benford:** La ley de Benford se establece como un parámetro o indicador de datos fraudulentos, siendo los dígitos de menor tamaño los que más posibilidad tienen de ser catalogados como fraude.
  Es muy útil para detectar operaciones que pretendan pasar desapercibidas como por ejemplo múltiples depósitos.

Métodos de detección automática de fraude (IA)

La inteligencia artificial y el aprendizaje automático son útiles para encontrar soluciones rápidas y eficientes a la hora de detectar fraudes y malas prácticas.

Estas nos permiten que se procese una gran cantidad de conjunto de datos con precisión, algo en lo que los humanos pueden fallar.

Gracias a los métodos de machine learning, los bancos pueden implementar soluciones que pueden mejorar ampliamente sus capacidades de detección de fraude y fortalecer su reputación hacia el cliente, evitando pérdidas antes de que ocurran.

Beneficios frente a métodos tradicionales

- **Detección rápida:** El cómputo rápido es un beneficio de la inteligencia artificial y el machine learning, dado a que desarrolla una comprensión de los patrones de uso de la aplicación de un usuario, como los métodos de pago, transacción, etc. Pudiendo detectar anomalías en

tiempo real, con mayor eficiencia que los métodos manuales y evitando en mayor medida los falsos positivos.

- **Mayor precisión:** La inteligencia artificial, nos permite calcular cantidades masivas de datos a mayor velocidad y mayor precisión, permitiendo así a los equipos analistas trabajar más rápido.

1.4 Estudio de alternativas de software de detección de fraude bancario

En este apartado vamos a explorar las diferentes propuestas software para la detección de fraude bancario ya existentes, basándonos en la información del ranking generado por la empresa Seon[5].

Seon

Seon aporta herramientas de detección de fraude que proporcionan análisis que nos permite extraer datos del correo electrónico, número de teléfono y dirección IP en concreto, de esta manera se puede aprender mucho sobre con quién se está tratando basándose en su huella digital.

La clave de Seon es que proporciona a los gestores de riesgos bancarios una capa adicional de datos para ganar plena confianza a la hora de aceptar o rechazar clientes de riesgo, influyendo esto sobre los índices de fraude.

Actualmente utilizados por empresas como Revolut, Albo o Nubank, con un precio de partida de 599 dólares.

Seon presenta una serie de pros y contras que debemos de tener en cuenta:

- **Pros:**

[5] Recuperado de la página : https://seon.io/resources/comparisons/banking-fraud-detection-software-tools/ el dia 05/09/2023

- **Búsqueda inversa en redes sociales:** Seon escanea alrededor de 50 redes sociales en busca de datos de clientes, ofreciendo una imágen precisa de con quién estás tratando.
- **Integración modular y flexible:** Las entidades bancarias modernas pueden elegir los módulos que necesitan para complementar su estrategia de gestión de riesgos o crear una pila de riesgos completamente nueva.
- **Precios transparentes:** Dado a que es un software como servicio (SaaS), Seon le permite pagar por llamada API y cancelar el contrato en cualquier momento.
- **Combina KYC y AML en un solo panel:** admite todas sus comprobaciones AML en el mismo panel de control ó a través de llamadas api.

- **Contras:**
  - **Sin instalación in situ:** Aunque tengas muchas formas de integrar Seon, solo está disponible como solución de terceros.

Figura 5: Logo Seon

ThreatMetrix[6]

ThreatMetrix es una plataforma de seguridad digital que se utiliza principalmente en el ámbito de la prevención de fraude y la autentificación de los usuarios en línea, ayudando a más del 78% de las empresas y operando en más de 170 países.

Es una plataforma especialmente útil si buscas reglas de aprendizaje automático e instalación local, así como la visualización de gráficos para la gestión de los riesgos.

ThreatMetrix se caracteriza por lo siguiente:

- **Autentificación de usuarios:** Permite la identificación de los usuarios en línea mediante la evaluación de una amplia gama de factores.

- **Detección de Fraude:** Utiliza el análisis de comportamiento y la detección de patrones sospechosos para identificar posibles actividades fraudulentas.

- **Análisis de dispositivos:** Evalúa los dispositivos utilizados por los usuarios para detectar dispositivos comprometidos o sospechosos.

- **Evaluación de la Geolocalización:** Se encarga de verificar la localización del usuario en tiempo real para detectar si realmente se encuentra en la ubicación declarada.

ThreatMetrix consta por tanto de una serie de pros y contras:

- **Pros:**

[6] Recuperado de la página: https://risk.lexisnexis.es/products/threatmetrix el dia 14/01/2024

- **Gran base de datos IP:** ThreatMetrix, tiene una de las bases de datos más grandes de direcciones IP que se encuentran en la lista negra.
- **Visualización de gráficos:** Los administradores pueden explorar los detalles de los titulares de cuentas con herramientas de visualización de datos.
- **Implementación en local:** Permite su integración de forma local para aquellas instituciones financieras que lo necesiten.

- **Contras:**
  - **Enriquecimiento de datos como extra:** Si deseas agregar datos de usuarios externos, deberás comprar productos adicionales como Emailage.

Figura 6: Logo ThreatMetrix

Experian Hunter[7]

Experian Hunter es un servicio proporcionado por Experian, una empresa global de servicios de información y análisis de datos, relacionada con la gestión de riesgos financieros y de crédito.
Este, se usa para ayudar a las instituciones financieras a evaluar su riesgo crediticio, cuya función principal es detectar y prevenir el fraude y el sobreendeudamiento.

Experian Hunter consta por tanto de una serie de pros y contras:

[7] Recuperado de la página: https://www.experian.es/soluciones-empresas/capacidades/datos/fraude-hunter el dia 14/01/2024

- **Pros:**
    - **Nombre confiable en BFSI:** Experian es conocido por sus informes crediticios y es una marca con la que cualquiera en la industria encontrará familiar.
    - **Integración con otras herramientas de Experian:** Puedes vincular Hunter con CrossCore como herramienta de biometría conductual o con sus herramientas de calificación crediticia.

- **Contras:**
    - **Cantidad abrumadora de productos:** Hunter es solo uno de los productos y servicios que ofrece experiencia, la cantidad de productos a veces puede resultar confusa y es posible una investigación previa para estar 100% seguro de estar seleccionando una herramienta adecuada.

Figura 7: Logo Experian

1.5 Estudio de alternativas y viabilidad backend

Las alternativas que hemos tenido en cuenta para el desarrollo del módulo backend son las siguientes:

- **Backend:** El backend es la parte invisible pero esencial de una aplicación. Esta es la encargada de manejar la lógica y el

procesamiento de todos los datos necesarios para que todo funcione de manera correcta y segura. Es decir, se encarga de tareas tales como almacenar y recuperar datos de una base de datos, formularios, accesos al servidor, gestionar la seguridad del sitio entre otras.

  - **Flask:** Enlace al sitio web oficial: https://flask.palletsprojects.com
  - **Ruby on rails:** Enlace al sitio web oficial: https://rubyonrails.org/
  - **Node.js:** Enlace al sitio web oficial: https://nodejs.org/es

- **Base de datos noSQL:** Una base de datos noSQL, es un tipo de base de datos no relacional, es decir, almacenan sus datos de forma no tabular.

  A diferencia de las bases de datos relacionales que utilizan una construcción de columnas y filas, este tipo de base de datos, emplean un modelo de almacenamiento optimizado que cumpla con los objetivos preestablecidos para dichos datos, construyendo a su vez una mejor forma en la que los datos puedan consultarse.

  - **MongoDB:** Enlace al sitio web oficial https://www.mongodb.com/es
  - **DynamoDB:** Enlace al sitio web oficial https://aws.amazon.com/es/dynamodb/
  - **ElasticSearch:** Enlace al sitio web oficial https://www.elastic.co/es/

- **Arquitectura:** Una arquitectura de aplicaciones describe los patrones y las técnicas que se utilizan para diseñar y desarrollar aplicaciones, proporcionando planes y prácticas recomendadas para que la aplicación quede bien estructurada.

  - **Arquitectura N-tier**
  - **Arquitectura monolítica**
  - **Arquitectura de microservicios**

1.5.1 Alternativas Backend

Flask

Flask es un framework minimalista open source escrito en python que permite crear aplicaciones rápidamente y con un mínimo número de líneas de código dado a su filosofía de desarrollo ágil. Es un componente de aplicaciones muy conocidas como LinkedIn, Uber, Airbnb y Spotify entre muchas otras.

A pesar de ser un framework minimalista nos ofrece la posibilidad de crear aplicaciones profesionales, robustas y escalables gracias a la capacidad de poder integrar las dependencias que necesitemos para nuestro proyecto. Además las revisiones se realizan con una media de uno a dos meses con el fin de corregir errores o añadir nuevas funcionalidades.

Figura 8: Logo flask

Ruby on Rails

Ruby on Rails es un marco de creación de aplicaciones de código abierto construido sobre Ruby. Permite escribir menos código, haciendo más que muchos otros lenguajes y framework ya que asume que siempre hay una forma de hacer mejor las cosas, su forma.

Debido a que está basado en dos principios: "Don't repeat yourself", que advierte que es contraproducente escribir el mismo código reiteradamente y "Convention over configuration", que indica que Rails supone lo que se quiere hacer y como quiere hacerse.

Figura 9: Logo ruby on rails

Node.js

Node.js es un entorno de ejecución multiplataforma basado en JavaScript, especialmente diseñado para crear aplicaciones escalables.

A comparación con otras plataformas utiliza arquitectura asíncrona, funcionando como un único proceso y haciendo uso de operaciones de E/S no bloqueantes, es decir, en vez de bloquear un hilo y desperdiciar recursos de la CPU esperando una respuesta, este lo mandara al bucle de eventos. Gracias a esto es capaz de manejar una cantidad masiva de peticiones simultáneas.

Figura 10: Logo node.js

1.5.2 Alternativas de bases de datos no relacionales

MongoDB

MongoDB es una base de datos noSQL orientada a documentos utilizada para almacenar volúmenes masivos de datos. Los documentos son pares de clave/valor que sirven como unidad básica de datos y su estructura depende de como decidamos construir los objetos, debido a que no presentan ningún esquema predefinido.

Un documento es el equivalente a un registro en una base de datos tradicional, identificado por un "_id" cuya información es almacenada en el formato de texto plano JSON, haciéndolo compatible con muchos lenguajes de programación.

Figura 11: Logo MongoDB

DynamoDB

DynamoDB es un servicio de base de datos noSQL sin servidor que admite modelos de datos de clave-valor y de documentos. Desarrollado por Amazon para ejecutar aplicaciones de alto rendimiento a gran escala que sobrecargarían las bases de datos tradicionales.

Al ser una base de datos sin servidor nos ofrece la ventaja de que no hay servidores que parchear o administrar ni software que instalar o mantener, por tanto no presenta ventanas de mantenimiento, por lo que no presentará inactividad.

Además por defecto DynamoDB protegerá los datos mediante cifrados y copias de seguridad continuas.

Figura 12: Logo DynamoDB

ElasticSearch

ElasticSearch es un motor de búsqueda y análisis altamente escalable y distribuido, de código abierto diseñado para almacenar, buscar y analizar grandes volúmenes de datos de manera rápida y eficiente.

Algunas de las características principales son:

- **Búsqueda de texto completo:** Elasticsearch, permite indexar y buscar información en diferentes idiomas y realizar búsquedas con coincidencia parcial, corrección ortográfica y con resultados resaltados.

- **Motor de análisis de datos:**Elasticsearch, junto a herramientas como Logstash y Kibana, permite realizar análisis de datos en tiempo real y generar visualizaciones en tiempo real.

- **Monitorización y observabilidad:** Elasticsearch, ayuda a recopilar y analizar datos de métricas, registros y trazas para proporcionar una visión completa del entorno backend.

Figura 13: Logo elasticsearch

1.5.3 Alternativas Arquitectura

Arquitectura N-tier

La arquitectura N-tier, es un enfoque de diseño de software en el que la aplicación se divide en varias capas o niveles, normalmente 3, donde cada una cumple una función particular.

Las capas nos ayudan a poder gestionar las dependencias y ejecutar funciones lógicas, con la condición de que una capa solo puede acceder a los recursos de la que está debajo de ella, o de cualquiera de las inferiores.

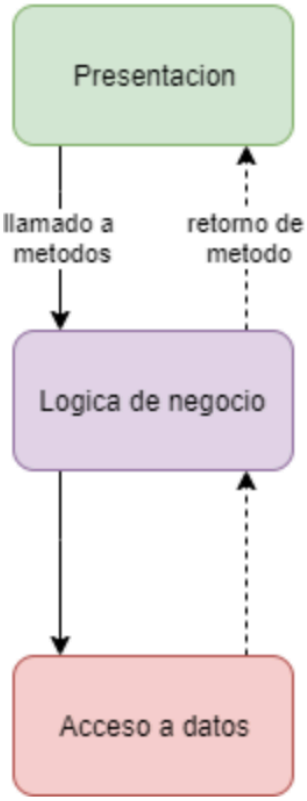


Figura 14: Ilustración arquitectura N-tier

Arquitectura Monolítica

La arquitectura monolítica son pilas de aplicaciones únicas que contienen todas las funciones dentro de cada aplicación, con conexión directa, tanto en la interacción entre los servicios como en la manera en que se desarrollan y distribuyen.

Esto significa que actualizar un solo aspecto de la aplicación, repercutirá en toda la infraestructura, teniendo que volver a lanzarla por completo, por lo que cada vez se hace menos uso de este tipo de arquitectura.

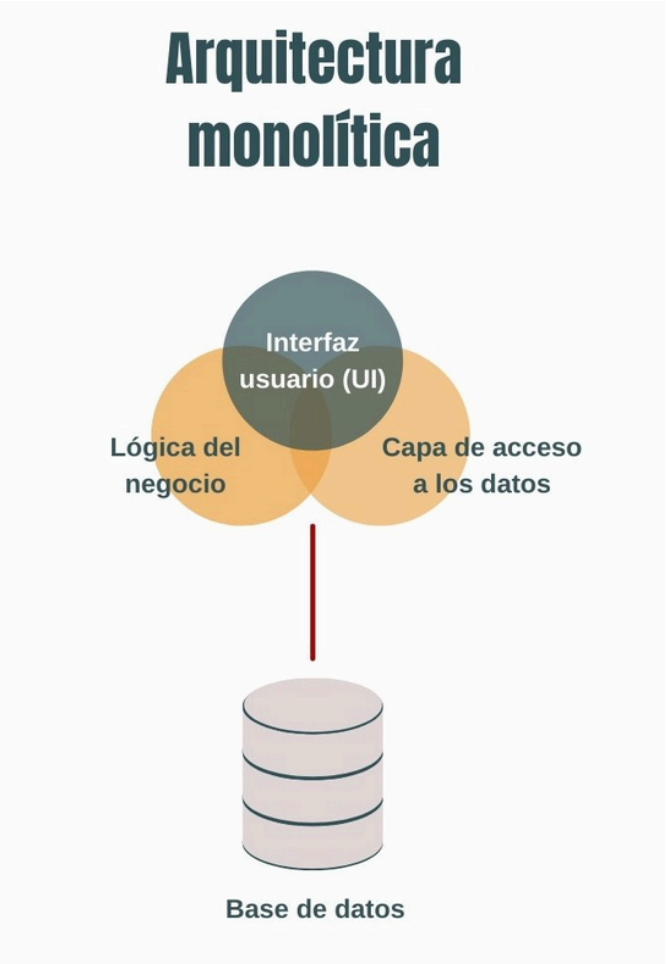


Figura 15: Ilustración arquitectura monolítica

Arquitectura de microservicios

Los microservicios no son solo un tipo de arquitectura, sino que también un modo de desarrollar software, ya que la aplicación se divide en elementos más pequeños que son independientes entre sí.

El objetivo de esta arquitectura es distribuir software de calidad con una mayor rapidez, ya que nos permite poder desarrollar múltiples microservicios independientes de forma simultánea, aportando así mayor tolerancia a fallos, ya que un fallo en un microservicio, no afecta a otros servicios.

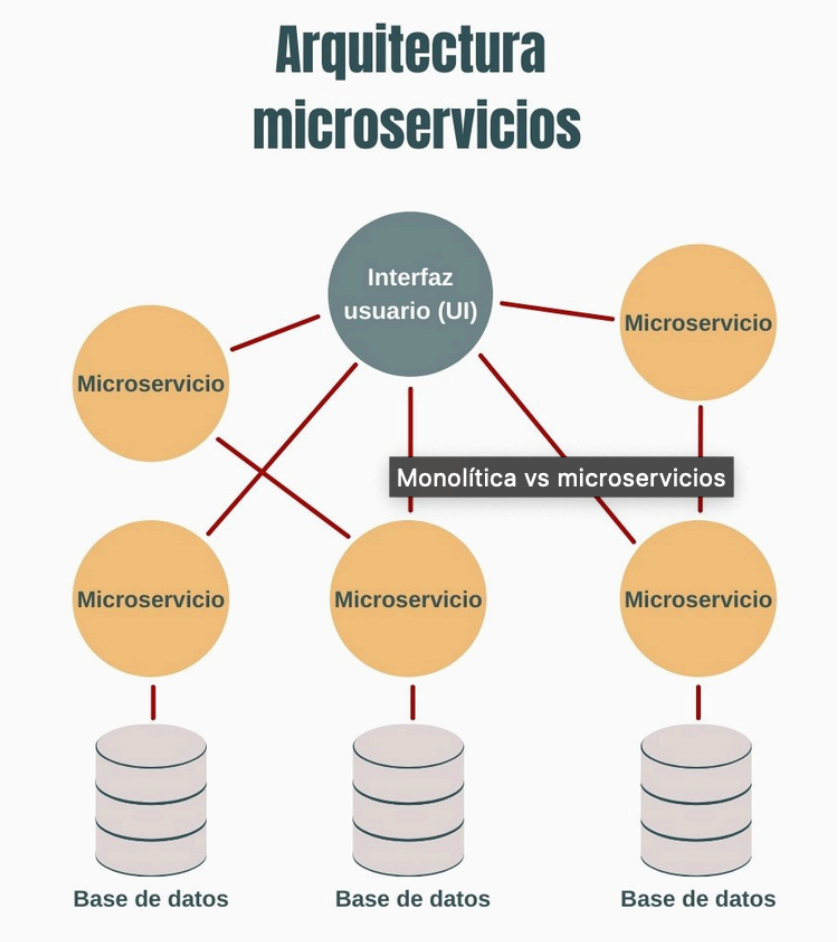


Figura 16: Ilustración arquitectura microservicios

1.6 Descripción de la solución propuesta

Para la tecnología backend, python flask ha sido elegido como marco de creación de backend, si bien Node.Js y Ruby on Rails son excelentes opciones en muchos casos, Flask destaca por su simplicidad y flexibilidad, dado que para este proyecto disponemos de un tiempo limitado de 300 horas, se ha priorizado la agilidad de desarrollo que nos ofrece flask junto al lenguaje python.

Además a diferencia de las otras dos alternativas, Python flask nos ofrece una escalabilidad gradual, lo que nos va a permitir adaptarnos a las necesidades cambiantes que encontremos durante el desarrollo de nuestro backend.

La decisión de una base de datos adecuada para nuestra aplicación es crítica, ya que puede presentar un gran impacto en el rendimiento y escalabilidad para nuestra aplicación, por tanto se ha elegido ElasticSearch como nuestra base de datos no relacional, debido a que para detectar transacciones fraudulentas en tiempo real en un entorno financiero, se va a necesitar procesar un gran volumen de datos, elasticsearch es la mejor opción para búsquedas y consultas en tiempo real.

En términos de arquitectura servidor, se ha elegido, la arquitectura híbrida basada en microservicios y n-tier, ya que al implementar una IA para la detección de transacciones fraudulentas, se necesita de escalabilidad, flexibilidad, tolerancia a los fallos y un mantenimiento eficiente.
Manteniendo los datos en una capa independiente para asegurar la privacidad y robustez de los datos de cada usuario.

Podemos decir por tanto que nuestra aplicación está diseñada con las últimas tecnologías en auge, de manera que podamos conseguir que esta sea segura, escalable y consiga adaptarse a las nuevas necesidades que se presenten.

1.7 Tecnologías utilizadas

Las tecnologías que se han utilizado para el desarrollo de nuestro sistema backend han sido:

- **Herramientas:**
  - **Postman:** Herramienta que nos permite realizar peticiones HTTP a cualquier API, pudiendo comprobar el correcto funcionamiento de nuestro backend.

    - **Docker:** Plataforma para automatizar la construcción y ejecución de contenedores de nuestro proyecto.
    - **Github:** Plataforma donde alojaremos el proyecto, bajo el control de versiones que nos ofrece Git.
    - **Google Docs:** Editor de texto con el fin de compartir el documento con el tutor, de manera que se puedan aportar comentarios y sugerencias, para mejorar la calidad de la memoria.
    - **Pip:** Es un sistema de gestión de paquetes, utilizado para instalar y administrar paquetes de software escritos en Python.

- **Entorno de desarrollo:**
    - **Visual Studio Code:** Entorno de desarrollo gratuito que nos ofrece una gran eficiencia y versatilidad, además de una amplia gama de extensiones que facilitan el desarrollo de aplicaciones Flask de manera rápida y sencilla.

2. Metodología de trabajo

En el proyecto a desarrollar, hemos decidido implementar una metodología ágil debido a su capacidad para abordar grandes proyectos mediante la división en iteraciones/incrementos. Dentro del ámbito ágil podemos encontrar diversos marcos de trabajo como Scrum, Kanban,Crystal, etc

2.1 Análisis metodologías

Nos centraremos en el estudio de los 2 primeros marcos mencionados anteriormente, ya que actualmente lideran el ranking de los marcos más utilizados.

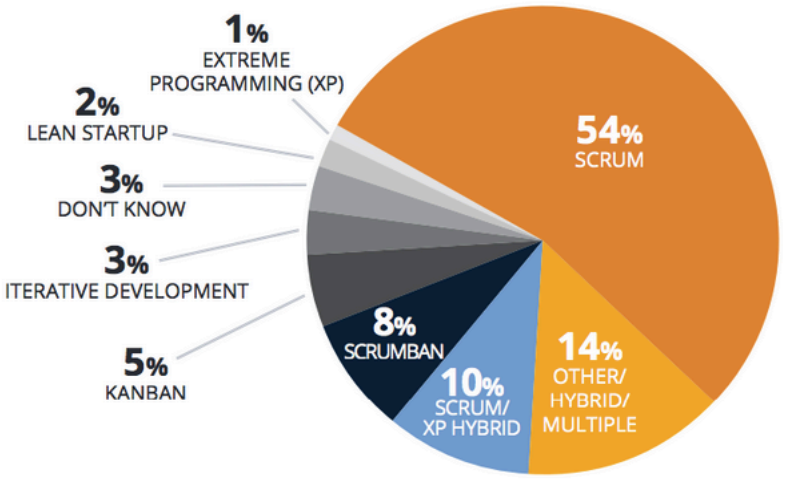


Figura 17[8]: Clasificación de metodologías ágiles

Scrum[9]

Scrum es un marco de trabajo ágil que permite abordar problemas complejos a la vez que se entregan productos de forma eficiente al cliente.

[8] Recuperado de la página: https://www.javiergarzas.com/2019/05/cual-es-el-estado-de-la-agilidad-a-nivel-mundial-13th-state-of-agile.html el dia 07/04/2024
[9] Recuperado de la página: *https://es.indeed.com/orientacion-laboral/desarrollo-profesional/ventajas-desventajas-metodologia-scrum* el día 04/02/2024

Por tanto ayuda a los equipos a colaborar y realizar un trabajo de alto impacto, gracias a proporcionar un plan de valores, roles que ayudan al equipo a concentrarse en la iteración y mejora continua en proyectos complejos.

Scrum presenta una serie de pros y contras a tener en cuenta:

- **Pros:**
    - **Favorece el trabajo en equipo:** Al basarse en dividir el trabajo en fases con objetivos y tareas concretas, todos los miembros del equipo pueden participar en todas las etapas del proyecto.

    - **Obtiene resultados anticipados:** Se obtienen resultados parciales al final de cada fase de trabajo, no siendo necesario esperar a que el proyecto llegue a su fin.

    - **Se trata de una metodología flexible:** Permite actuar sobre la marcha, pudiendo plantear acciones para solucionar las dificultades que aparezcan durante el proyecto.

- **Contras:**
    - **Las tareas y plazos deben de estar siempre definidos:** Si no se encuentran definidas en cada fase del proyecto, la metodología no sirve de nada.

    - **No hay lugar para actividades sin terminar:** Las tareas deben de estar terminadas en la fecha establecida, ya que en caso contrario el resto de tareas se van posponiendo automáticamente.

Kanban[10]

Kanban es una metodología que permite gestionar el desarrollo de tareas gracias a su visualización de trabajo por fases permitiendo evitar la sobrecarga a la vez que se mide el tiempo estimado para cada tarea.

Kanban presenta una serie de pros y contras a tener en cuenta:

- **Pros:**
    - **Evita la acumulación de trabajo:** Al poder previsualizar todas las tareas y el tiempo estimado, el equipo es capaz de organizarse.

    - **Distribución de tareas:** El poder ver el trabajo que se ha hecho, se está haciendo o queda por hacer, ayuda al equipo a saber cual es el siguiente paso.

    - **Medición del rendimiento:** Con este método podemos medir el rendimiento de los trabajadores o equipos, para poder detectar cualquier problema que se detecte durante el transcurso de la tarea.

- **Contras:**
    - **Costes:** Si se usa este método para proyectos muy grandes, el almacenamiento del sistema será muy costoso.

    - **No permite anticiparse a grandes aumento de la demanda:** Con este método resulta difícil de manejar cambios de gestión provocados por la acumulación de nuevas tareas.

[10] Recuperado de la página: ***https://www.fhios.es/metodologia-kanban-pros-y-contras/*** el día 04/02/2024

	SCRUM	KANBAN
Cadencia	Sprints regulares de duración limitada (2-4 semanas)	Flujo continuo
Entrega de valor	Al final de cada Sprint	Entrega continua o a discreción del equipo.
Roles	Product Owner, Scrum Master, DEV. team	No hay roles
Métricas	Velocidad	Tiempo de ciclado
Filosofía de cambio	No hay cambio durante el Sprint	El cambio puede ocurrir en cualquier momento

Tabla 1: Diferencias Scrum VS kanban

En el mundo de la metodología ágil, no cabe duda que la preferida entre las empresas es la metodología scrum[11], ocupando un 56% del panorama actual sin contar sus variantes. Aunque Kanban sólo un 5%, su utilidad no debe de subestimarse en este contexto.

Ambas metodologías ofrecen enfoques valiosos para la gestión de proyectos, pero en nuestro caso elegimos la metodología Kanban.

[11] Recuperado de la página : *https://www.javiergarzas.com/2019/05/cual-es-el-estado-de-la-agilidad-a-nivel-mundial-13th-state-of-agile.html* el dia 04/02/2024

2.2 ¿Por qué Kanban?

La elección de la metodología ágil Kanban en lugar de Scrum para este proyecto se basa en la flexibilidad que ofrece Kanban. En comparación con Scrum, Kanban ofrece un enfoque más continuo y adaptable, lo que resulta fundamental dada la naturaleza dinámica de nuestro proyecto.

Además la ausencia de roles fijos y la capacidad de ajustar el proceso sin la rigidez de los sprints, encajan mejor a nuestra metodología de desarrollo, siendo posible visualizar todo el flujo de trabajo de un solo vistazo al basarse en una distribución en tarjetas.

2.3 Planificación

La planificación de un proyecto es el pilar fundamental que establece las bases para su éxito, donde además combinada con la metodología Kanban lograremos una ejecución más flexible y centrada en los resultados permitiendo que las tareas se desplacen de manera fluida a lo largo del tablero.

Para una correcta planificación del proyecto se ha utilizado Trello, donde inicialmente se ha dividido en 2 incrementos, donde al final de cada uno se entregará una versión usable del proyecto.

Trello

Hemos elegido la herramienta Trello[12], dado que proporciona un entorno visual y altamente intuitivo, basado en tarjetas, que se adapta de manera efectiva a nuestra metodología ágil basada en Kanban.

Las tarjetas en Trello, son como post-it de oficina, que puedes ordenar en el tablero. Es decir son post-it digitales que se pueden buscar, compartir y te sirven a modo de recordatorio.

[12] Enlace a la página web: https://trello.com

De esta manera nos permite trabajar de una manera mucho más visual, pudiendo dividir el tablero en columnas, para posicionar cada tarjeta según el estado en el que se encuentre la tarea, en nuestro caso, se divide en las siguientes listas, TO DO, DOING, REVIEW, DONE.

- **TO DO:** Se encuentran todas aquellas tareas que no han sido iniciadas.
- **DOING:** Se encuentran aquellas tareas que se encuentran actualmente en desarrollo.
- **REVIEW:** Columna donde se encuentran las tareas que actualmente están en revisión
- **DONE:** Se encuentran las tareas que han pasado la revisión y han finalizado correctamente.

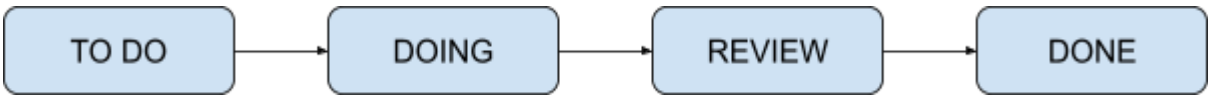


Figura 18: Esquema columnas Trello

Quedando nuestro tablero del proyecto de la siguiente manera:

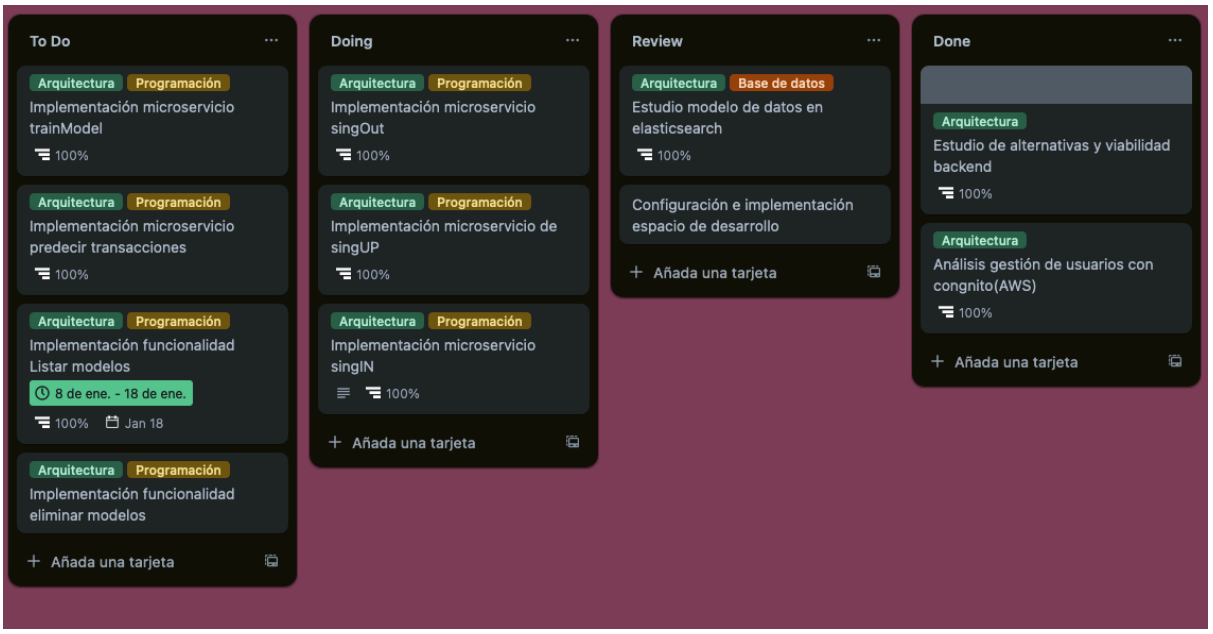


Figura 19: Tablero Bankingfraud

2.3.1 Estimación del tamaño y esfuerzo

Dado que este proyecto se trata de un trabajo de fin de grado, es importante señalar que las restricciones no se relacionan con aspectos económicos, sino que están principalmente vinculadas a las limitaciones de tiempo.

Por tanto la evaluación del tamaño del proyecto se realizará en función al tiempo dedicado.

Respecto al esfuerzo se asume que solo se contará con la participación de un único desarrollador, que en este caso será el autor del proyecto.

2.3.2 Presupuesto

Para el cálculo del presupuesto de nuestro proyecto, no se van a tener en cuenta los costes indirectos como la electricidad y se va a considerar una jornada laboral de 3.5 - 4 horas diarias de Lunes a Viernes durante 4 meses (80 días laborables), rigiendonos así al límite de 300 horas de nuestro proyecto.

Para el sueldo del trabajador[13], vamos a tener en cuenta el salario medio del puesto de desarrollador backend, donde se hará el cómputo de horas para ajustarla a la jornada laboral anteriormente descrita.

Figura 20: Sueldo medio mensual backend España

Recurso	Unidad	Tiempo	Costo Global	Costo Parcial
Visual studio code	Licencia	-	0 €	0 €
Postman	Licencia	-	0 €	0 €
Docker	Licencia	-	0 €	0 €
GitHub	Licencia	-	0 €	0 €
Macbook pro 14"	Unidad	300 horas	2500 €	833.3 €

[13] Recuperado de la página: https://es.talent.com/salary?job=back+end el dia 04/02/2024

Magic Mouse	Unidad	300 horas	110 €	36.6 €
Magic keyboard	Unidad	300 horas	135 €	45 €
Monitor Philips ultrawide 34"	Unidad	300 horas	320 €	106.6 €
Monitor dell 4k 27"	Unidad	300 horas	413 €	137.6 €
Programador Backend	Mes	300 horas	33600 €	5600 €
Coste Total				6759.1 €

Tabla 2: Presupuesto

Adicional al desarrollo del backend, es esencial el realizar una estimación económica detallada para el despliegue cloud de la misma, en nuestro caso en AWS, al ser la plataforma con mayor relación calidad precio.

Para ello vamos a evaluar los componentes y servicios de AWS[14] necesarios para soportar la carga de trabajo esperada, teniendo en cuenta tanto el número de usuarios concurrentes como el rendimiento de la infraestructura para mantener un rendimiento óptimo.

A continuación, se representará una estimación de costos basada en un escenario con 250 consultas por segundo en horas pico.

EC2 (Elastic compute cloud): Servicio encargado de ejecutar la aplicación y realizar todo el cómputo necesario.
Ajustandonos a nuestros requerimientos, se necesitarán dos instancias "m5.large(2 vCPUs, 8GB ram)" para un óptimo tiempo de respuesta,

[14] Recuperado de la página: https://aws.amazon.com/es/pricing/?aws-products-pricing.sort-by=item.additionalFields.productNameLowercase&aws-products-pricing.sort-order=asc&awsf.Free%20Tier%20Type=*all&awsf.tech-category=tech-category%23compute el dia 29/04/2024

suponiendo un costo de 0.096 euros/hora, siendo una facturación mensual de 69 euros.

API Gateway: Servicio encargado de manejar las solicitudes de la aplicación. Este servicio, presenta un costo fijo, siendo este de 3.50 euros por millón de solicitudes.

Dónde 250 consultas por segundo tendría un coste mensual de 2268 euros

CloudWatch: Servicio encargado del monitoreo y logs de la aplicación, algo altamente necesario para detectar posibles comportamientos anómalos.
Se estima en nuestra aplicación un total de 10GB de datos mensuales, lo que supondría un costo de 0.50 euros/GB.

Lo que supondría un coste mensual de 5 euros.

Por tanto el costo mensual estimado del despliegue de la aplicación sería de 2342 euros.

2.3.3 Diagrama de Gantt

Para la representación temporal del desarrollo del proyecto, se ha optado por representar las líneas temporales de las tareas mediante un diagrama de Gantt. De esta manera se puede visualizar de forma sencilla, la fecha de inicio y fin de cada tarea y sobre qué incremento han sido desarrolladas. Este a su vez nos otorga una mejora en la comunicación y colaboración de los miembros de un equipo, ya que podemos conocer qué tarea tiene asignada cada miembro del equipo.

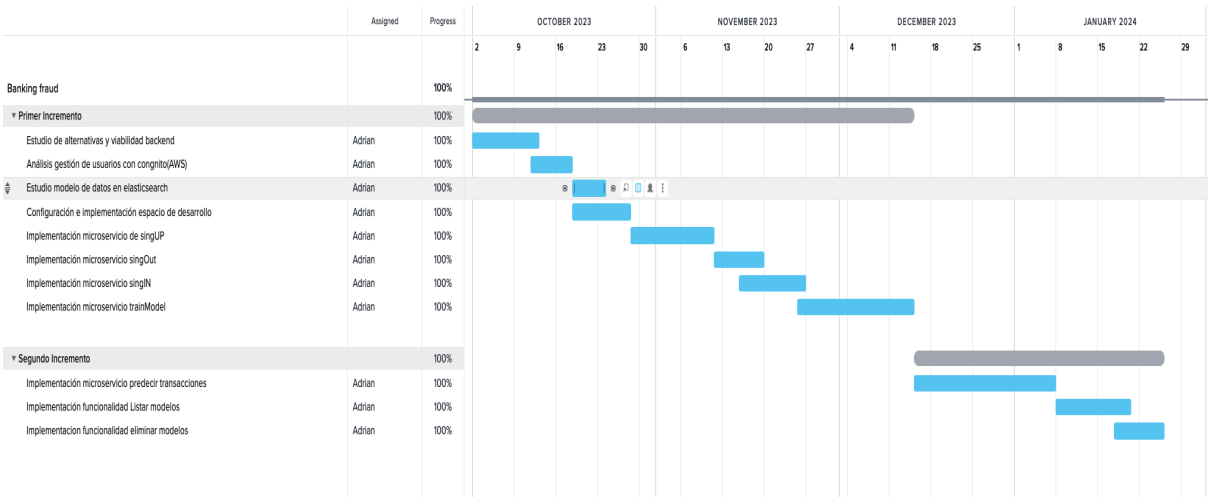


Figura 21: Diagrama de Gantt

3. Diseño Inicial

El objetivo principal de esta sección es establecer las estructura sobre la que se cimentará el desarrollo del proyecto. Siendo primordial el alcanzar un desarrollo eficiente mediante la implementación de la metodología por incrementos, lo que nos permitirá un desarrollo óptimo gracias a los elementos definidos en esta sección.

3.1 Requisitos iniciales

Los requisitos iniciales son la base sobre la que se construirá nuestro proyecto. Es decir, una descripción general de la funcionalidad, necesidades y características que debe de cubrir nuestro proyecto.

Encontramos dos tipos de requisitos iniciales:

- **Requisitos funcionales:** Definen las funcionalidades que debe de tener el proyecto desde la perspectiva del usuario final.

- **Requisitos no funcionales:** Definen las funcionalidades del proyecto que no son directamente visibles por el usuario.

3.1.1 Requisitos funcionales

A continuación se presentarán los requisitos funcionales que conformarán nuestro proyecto. Cabe resaltar que los requisitos pueden variar según las necesidades particulares y objetivos del proyecto.

- **Gestión de usuario:** Permite a los usuarios, poder registrarse en la API, para poder acceder a la funcionalidad de la misma. Asimismo, brinda la opción de cerrar sesión en sus cuentas.

- **Entrenamiento de modelos:** Ofrece a los usuarios la posibilidad de entrenar un modelo de datos específico a sus necesidades, de manera que pueda adaptarse de la mejor manera a su problema.

- **Predicción de transacciones fraudulentas:** Permite a los usuarios el poder realizar predicciones sobre transacciones de forma personalizada, ya que pueden elegir entre cualquiera de los modelos entrenados anteriormente.

- **Listado de modelos entrenados:** Ofrece a los usuarios la posibilidad de poder listar los modelos entrenados, para poder así elegir cual de ellos poder utilizar para predecir transacciones, ó si desean eliminar un modelo que ya no le sea de utilidad.

- **Gestión de modelos:** Permite a los usuarios, eliminar un modelo en concreto, que ya no le resulte de utilidad, ó si desea reentrenar un modelo con una nueva fuente de datos.

3.1.2 Requisitos no funcionales

A continuación se presentarán los requisitos no funcionales que conformarán nuestro proyecto. Estos se centran en las características generales y su rendimiento.

- **Rendimiento:** La API debe de tener un tiempo de respuesta rápido, así como la capacidad de poder manejar un alto volumen de solicitudes, de manera que se tenga una experiencia de usuario fluida.

- **Seguridad:** La API debe de incluir autenticación y autorización, para poder garantizar la integridad y la confidencialidad de la información.

- **Confiabilidad:** La API debe de funcionar de manera consistente y predecible bajo diversas condiciones, minimizando los casos de error.

- **Disponibilidad:** La API debe de tener un alto nivel de disponibilidad, incorporando manejo de errores.

- **Usabilidad:** La API aunque no disponga de una interfaz, debe de facilitar a los desarrolladores su entendimiento e implementación.

- **Documentación:** La API debe de ofrecer una documentación exhaustiva y clara facilitando a los desarrolladores utilizarla de manera efectiva.

3.2 Casos de uso

En este apartado se va a explicar los casos de uso de nuestra API, describiendo las diferentes situaciones y acciones a través de las cuales puede ser utilizada para cumplir objetivos específicos.

Para ello vamosa seguir el Marco de Desarrollo *MADEJA* de la junta de Andalucía[15]

CU-01	**Registrar Usuario**	
Versión	1.0(24/02/2024)	
Dependencias	-	
Precondiciones	El usuario realiza la integración con la API	
Descripción	Permite al usuario crear una cuenta, para acceder a todas las funcionalidades del API	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario realiza la integración de la API y como no está registrado, llama al endpoint del registro.
	2	El usuario realiza la petición con los datos de registro dentro de un body.
	3	El sistema valida los datos de entrada y si este se encuentra ya registrado en el sistema.
	4	Si la validación es exitosa, y su correo electrónico no se encuentra previamente registrado, el sistema crea

[15] Recuperado de la página: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416 el dia 24/02/2024

		el usuario en un estado "sin confirmar" y devuelve un mensaje que indica que el usuario se ha registrado correctamente.	
	5	El sistema envía un mensaje OTP al correo electrónico del usuario, de manera que valida si el correo pertenece realmente al usuario.	
	6	El usuario realiza la llamada al endpoint de confirmación de cuenta, enviando en el body de la petición, el correo electrónico y el código OTP	
	7	La APi devuelve un mensaje de confirmación indicando que el usuario se ha confirmado correctamente	
Postcondición	El usuario ha creado una cuenta de forma satisfactoria y puede proceder a iniciar sesión utilizando sus credenciales.		
Excepciones	**Paso**	**Acción**	
	2	El usuario no incluye todo los campos requeridos en el formulario	
		E.1	El sistema muestra un mensaje de error indicando los campos faltantes en el registro
	3	Si el sistema detecta que el correo electrónico se encuentra ya en la base de datos, muestra un mensaje indicando que el usuario ya se encuentra registrado	
		E.1	El sistema verifica la existencia del correo electrónico en la base de datos.
		E.2	El sistema muestra un error indicando que el correo electrónico ya se encuentra registrado.

	6	El usuario introduce un código OTP no válido.	
		E.1	El sistema verifica con cognito que el código OTP sea válido.
		E.2	En caso de no ser válido, el sistema devuelve un mensaje indicando que el código es incorrecto.

Tabla 3: Caso de uso -- Registrar usuario

CU-02	**Iniciar sesión**	
Versión	1.0(24/02/2024)	
Dependencias	El usuario debe de tener una cuenta registrada.	
Precondiciones	El usuario realiza la llamada al endpoint para iniciar sesión.	
Descripción	Permite al usuario iniciar sesión en el API, con los datos utilizados en su registro.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario realiza la llamada al endpoint de inicio de sesión con las credenciales utilizadas en el registro.
	2	El sistema realiza una validación de los datos de entrada.
	3	El sistema verifica si el correo está registrado.
	4	El sistema verifica si el usuario está confirmado.
	5	El sistema devuelve un mensaje de éxito, junto al token de cognito asociado a su cuenta.

Postcondición	El usuario ha iniciado sesión y puede acceder a todas las funcionalidades de la API.		
Excepciones	**Paso**	**Acción**	
	2	Si la validación de los datos de entrada no son correctos.	
		E.1	El sistema devuelve un mensaje de error indicando cuales son los campos incorrectos.
	3	Si el usuario no se encuentra registrado en el sistema.	
		E.1	El sistema devuelve un mensaje indicando que ese usuario no se encuentra registrado.
	4	Si el usuario no se encuentra confirmado.	
		E.1	El sistema devuelve un mensaje indicando que se encuentra registrado pero no confirmado.

Tabla 4: Caso de uso -- Iniciar sesión

CU-03	**Cerrar sesión**	
Versión	1.0(24/02/2024)	
Dependencias	El usuario debe de tener una sesión iniciada.	
Precondiciones	El usuario realiza la llamada al endpoint para cerrar sesión	
Descripción	Permite al usuario cerrar sesión en el API, haciendo uso de su token personal de cognito.	
Secuencia	**Paso**	**Acción**

Normal	1	El usuario realiza la llamada al endpoint de cierre de sesión con el bearer token de su inicio de sesión anterior.	
	2	El sistema verifica que el token de sesión se encuentra activo.	
	3	El sistema cierra el token de sesión administrado por el usuario.	
	4	El sistema devuelve un mensaje, indicando que se ha cerrado sesión de manera satisfactoria.	
Postcondición	El usuario ha podido cerrar sesión de manera satisfactoria.		
Excepciones	**Paso**	**Acción**	
	2	Si el bearer token no se encuentra en activo.	
		E.1	El sistema devuelve que el usuario no se encuentra logueado.

Tabla 5: Caso de uso -- Cerrar sesión

CU-04	**Entrenar modelos**	
Versión	1.0(24/02/2024)	
Dependencias	El usuario debe de tener una sesión iniciada.	
Precondiciones	El usuario realiza la llamada al endpoint para entrenar el modelo.	
Descripción	Permite al usuario entrenar un modelo personalizado.	
Secuencia Normal	**Paso**	**Acción**

	1	El usuario realiza la llamada al endpoint para entrenar modelos con el bearer token de su inicio de sesión anterior.	
	2	El sistema verifica que el token de sesión se encuentra activo.	
	3	El sistema comprueba si los datos de entrada para entrenar el modelo son correctos.	
	4	El sistema verifica si ya se encuentra un modelo, con ese mismo nombre y algoritmo para el usuario actual.	
	5	El sistema procede al entrenamiento del modelo de predicción.	
	6	El sistema devuelve un mensaje indicando, que el entreno del modelo ha sido satisfactorio, junto a su ratio de acierto del mismo.	
Postcondición	El usuario ha entrenado correctamente un modelo de predicción, el cual puede utilizar para futuras predicciones.		
Excepciones	**Paso**	**Acción**	
	2	Si el bearer token no se encuentra en activo.	
		E.1	El sistema devuelve que el usuario no se encuentra logueado.
	3	Si la validación de los datos de entrada no son correctos.	
		E.1	El sistema devuelve un mensaje de error indicando cuales son los campos incorrectos.

	4	Si el modelo ya se encuentra entrenado en el sistema.	
		E.1	El sistema devuelve un mensaje indicando que ese modelo ya ha sido entrenado para ese usuario.

Tabla 6: Caso de uso -- Entrenar modelos

CU-05	**Predecir transacciones**	
Versión	1.0(24/02/2024)	
Dependencias	El usuario debe de tener una sesión iniciada.	
Precondiciones	El usuario realiza la llamada al endpoint para predecir transacciones. El usuario debe de tener al menos un modelo entrenado.	
Descripción	Permite al usuario predecir si una transacción es fraudulenta o legítima.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario realiza la llamada al endpoint para predecir transacciones con el bearer token de su inicio de sesión anterior.
	2	El sistema verifica que el token de sesión se encuentra activo.
	3	El sistema comprueba si los datos de entrada para predecir la transacción son correctos.
	4	El sistema verifica que existe el modelo indicado por

		el usuario en la base de datos.	
	5	El sistema realiza la predicción de la transacción proporcionada.	
	6	El sistema responde con el modelo de la transacción y el resultado de la predicción.	
Postcondición	El usuario ha podido predecir si la transacción es fraudulenta o legítima.		
Excepciones	**Paso**	**Acción**	
	2	Si el bearer token no se encuentra en activo.	
		E.1	El sistema devuelve que el usuario no se encuentra logueado.
	3	Si la validación de los datos de entrada no son correctos.	
		E.1	El sistema devuelve un mensaje de error indicando cuales son los campos incorrectos.
	4	Si el modelo no se encuentra entrenado.	
		E.1	El sistema devuelve un mensaje indicando que ese modelo no se encuentra entre los modelos de los usuarios.

Tabla 7: Caso de uso -- Predecir transacciones.

CU-06	Listar modelos	
Versión	1.0(24/02/2024)	
Dependencias	El usuario debe de tener una sesión iniciada.	
Precondiciones	El usuario realiza la llamada al endpoint para listar modelos entrenados.	
Descripción	Permite al usuario conocer cuales son los modelos que actualmente tiene entrenados.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario realiza la llamada al endpoint para listar modelos con el bearer token de su inicio de sesión anterior.
	2	El sistema verifica que el token de sesión se encuentra activo.
	3	El sistema comprueba si los datos de entrada para listar modelos son correctos.
	4	El sistema comprueba los modelos entrenados por el usuario.
	5	El sistema devuelve un mensaje en formato JSON con todos los modelos del usuario, ordenados por fecha de entrenamiento.
Postcondición	El usuario puede ver los modelos entrenados.	
Excepciones	**Paso**	**Acción**
	2	Si el bearer token no se encuentra en activo.

		E.1	El sistema devuelve que el usuario no se encuentra logueado.
	3	Si la validación de los datos de entrada no son correctos.	
		E.1	El sistema devuelve un mensaje de error indicando cuales son los campos incorrectos.
	4	Si el usuario no tiene ningún modelo entrenado.	
		E.1	El sistema devuelve un mensaje indicando que no se ha encontrado ningún modelo.

Tabla 8: Caso de uso -- Listar modelos.

CU-06	**Eliminar modelos**	
Versión	1.0(24/02/2024)	
Dependencias	El usuario debe de tener una sesión iniciada.	
Precondiciones	El usuario realiza la llamada al endpoint para eliminar modelos.	
Descripción	Permite al usuario eliminar modelos que ya no le sean útiles.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario realiza la llamada al endpoint para listar modelos con el bearer token de su inicio de sesión anterior.
	2	El sistema verifica que el token de sesión se encuentra activo.
	3	El sistema comprueba los datos de entrada para eliminar el modelo.

	4	El sistema comprueba los modelos entrenados por el usuario.	
	5	El sistema devuelve un mensaje satisfactorio indicando que el modelo ha sido eliminado correctamente.	
Postcondición	El usuario ha eliminado un modelo de su lista de modelos entrenados.		
Excepciones	**Paso**	**Acción**	
	2	Si el bearer token no se encuentra en activo.	
		E.1	El sistema devuelve que el usuario no se encuentra logueado.
	3	Si la validación de los datos de entrada no son correctos.	
		E.1	El sistema devuelve un mensaje de error indicando cuales son los campos incorrectos.
	4	Si el usuario no tiene ningún modelo entrenado.	
		E.1	El sistema devuelve un mensaje indicando que no se ha encontrado ningún modelo.

Tabla 9: Caso de uso -- Eliminar modelos.

3.3 Análisis y diseño del sistema de datos

Para el desarrollo de nuestro proyecto vamos a emplear una base de datos no relacional que contendrá diversas colecciones.

Una colección en el mundo de las bases de datos no relacionales son agrupaciones de documentos equivalentes a las tablas de las bases de datos relacionales, con la diferencia de que las colecciones no siguen una estructura rígida para guardar la información, siendo estas más flexibles a la hora de almacenar datos.

Las colecciones usadas en nuestro modelo de datos, son las siguientes:

- **Individual:**
  Esta colección representa a los usuarios del sistema, permitiéndonos gestionar y mantener un registro detallado de los usuarios registrados en el sistema.
  Incluyendo información relevante como el estado de confirmación del correo electrónico, el estado global del registro y la fecha del mismo, entregándonos un control sobre los usuarios registrados.

  En esta colección se almacena la siguiente información:
  - user (Objeto)
    - email (Correo electrónico)
    - name (Nombre del usuario)
    - last_name (Apellido del usuario)
  - user_credentials
  - email_confirmed (Email confirmado)
  - register (Estado del registro)
  - timestamp (Fecha de registro)

Individual			
CAMPOS		**TIPO**	**DESCRIPCIÓN**
	email	keyword	Correo electrónico del

user			usuario.
	name	keyword	Nombre del usuario.
	last_name	keyword	Apellido del usuario
user_credentials		Keyword	Credencial del usuario.
email_confirmed		boolean	Booleano que indica si el email ha sido confirmado.
register		boolean	Booleano que indica si el usuario está registrado.
timestamp		date	Fecha de registro del usuario.

Tabla 10: Colecciones BBDD -- Individual.

- **Trining_model:**

  Esta colección asume una funcionalidad crucial al almacenar y organizar los modelos entrenados por cada usuario a través de la API. Es decir, almacena información relevante de cada modelo, permitiéndonos un acceso eficiente a los modelos específicamente entrenados por cada usuario, fortaleciendo la capacidad del sistema de ofrecer experiencias eficientes a los clientes.

  En esta colección se almacena la siguiente información:

  - model (Lista de objetos)
    - model_data (Modelo base64)
    - model_name (Nombre del modelo)
    - file_name (Archivo fuente del modelo)
    - algorithm (Algoritmo del modelo)
  - user (Usuario)

- timestamp (Fecha de entrenamiento del modelo)

Traininig_model			
CAMPOS		**TIPO**	**DESCRIPCIÓN**
Model	model_data	keyword	Datos del modelo en base64
	model_name	keyword	Nombre del modelo.
	file_name	keyword	Nombre del archivo fuente, a través del que se ha entrenado el modelo.
	algorithm	keyword	Tipo de algoritmo utilizado para entrenar el modelo.
user		Keyword	Usuario al que pertenece el modelo.
timestamp		date	Fecha de registro del usuario.

Tabla 11: Colecciones BBDD -- Training_model.

- **Transaction_history**

  Esta colección representa el histórico de predicción de transacciones, esto garantiza que cada petición realizada por el usuario queda debidamente registrada y disponible para su posterior consulta y análisis si el usuario lo desea.

Permitiendo por tanto a los usuarios mejorar y optimizar los modelos predictivos en caso de ser necesario.

En esta colección se almacena la siguiente información:

- user (Usuario)
- model_name (nombre del modelo)
- algorithm (Algoritmo del modelo)
- prediction (Predicción del a transacción)
- transaction
    - type (Tipo de transacción)
    - amount (Importe de la transacción)
    - nameOrig (Nombre del originador)
    - oldbalanceOrig (Balance antiguo del originador)
    - newBalanceOrig (Balance actualizado originador)
    - nameDest (Nombre del destinatario)
    - oldBalanceDest (Balance antiguo del destinatario)
    - newBalanceDest (Balance actualizado destinatario)
- timestamp (Fecha de la predicción)

Transaction_history		
CAMPOS	**TIPO**	**DESCRIPCIÓN**
user	keyword	Usuario al que pertenece el modelo.
model_name	keyword	Nombre del modelo.
algorithm	keyword	Tipo de algoritmo utilizado para entrenar el modelo.

prediction		keyword	Campo que indica si la predicción ha sido fraudulenta o legítima.
transaction	type	keyword	Tipo de transferencia.
	amount	float	Importe de la transacción.
	nameOrig	keyword	Nombre del originador.
	oldBalanceOrg	float	Balance antiguo del originador.
	newBalanceOrig	float	Balance actualizado originador.
	nameDest	keyword	Nombre del destinatario.
	oldBalanceDest	float	Balance antiguo del destinatario.
	newBalanceDest	float	Balance actualizado del destinatario.
timestamp		date	Fecha de registro del usuario.

Tabla 12: Colecciones BBDD -- Transaction_history.

3.4 Diseño arquitectónico.

En esta sección nos centraremos en establecer los principios arquitectónicos sobre los que se cimentará nuestra aplicación backend.

3.4.1 Arquitectura del sistema

La arquitectura de nuestra aplicación backend, solo contará con la parte del servidor, cuya funcionalidad se encarga del procesamiento y almacenamiento de datos, permitiendo gestionar la lógica de la aplicación, desde la gestión de modelos de datos, usuarios, predicciones…., como establecer la comunicación con la base de datos e interactuar con los mismos.

La arquitectura va a estar formada por un híbrido de microservicios y de capas, en nuestro caso siendo necesarias 2.

Algunos de los beneficios que tiene la arquitectura híbrida de microservicios y capas son los siguientes:

- **Escalabilidad:** La arquitectura en microservicios permite escalar individualmente según la demanda, además al estar dividido por capas, estas permiten una escalabilidad horizontal y vertical, según las necesidades específicas del momento.

- **Mantenimiento:** El desarrollo aislado de cada microservicio facilita la implementación continua de nuevas funcionalidades, sin afectar a otras partes del sistema, además las capas nos permiten realizar cambios específicos sin afectación global.

- **Flexibilidad:** La arquitectura híbrida nos facilita la flexibilidad tecnológica al permitir que tanto los microservicios como las capas utilicen tecnologías específicas según sus requisitos. En nuestro caso permitiendo tener las siguientes capas:

    - **Capa de aplicación:** Esta capa es la encargada de implementar la lógica de negocio. Incluyendo, la creación/gestión de usuarios, entrenamiento/gestión de modelo

y predicción de transacciones bancarias. Para ello, se utilizarán las tecnologías y herramientas anteriormente expuestas en la sección 1.6, permitiendo alcanzar un rendimiento y desarrollo eficiente.

- **Capa de datos:** Esta capa es la encargada de gestionar el acceso a los datos almacenados, proporcionando una interfaz para la manipulación de los mismos. Se encargará de gestionar la información de los usuarios, modelos de datos entrenados y un historial de predicción de transacciones. En nuestro caso, tal y como desarrollamos anteriormente en la sección 1.6, utilizaremos elasticsearch, en el cual se implementarán los modelos de datos necesarios para su correcto funcionamiento.

- **Resiliencia y Tolerancia a fallos:** Se minimiza el impacto de posibles errores gracias a que los microservicios y las capas al pueden gestionarse de manera independiente.

- **Fácil colaboración:** El desarrollo se puede realizar de manera independiente en áreas específicas de la aplicación, permitiendo agilizar el desarrollo.

4. Seguridad en APIs

En la actualidad, las APIs, juegan un papel fundamental en la comunicación entre sistemas, permitiendo intercambiar información sin importar el tipo de aplicación ni sistema se utilice.
Sin embargo este tipo de comunicación está expuesta a una serie de amenazas que hay que tener en cuenta a la hora de su desarrollo.

En este apartado expondremos los diversos ataques y amenazas a las que están expuestas y las mejores estrategias para defendernos de las mismas.

4.1 Vulnerabilidades principales y mitigación de amenazas

La fundación *Open web Application Security Project (OWASP)*[16] gracias a su proyecto *Api security*[17], nos permite conocer el ranking de las 5 amenazas más comunes y la forma más óptima de mitigarlas.

Autorización incorrecta a nivel de objeto

Este tipo de amenaza se basa en aprovecharse de los objetos del API, que no estén protegidos mediante autorización. Siendo por tanto dichos objetos vulnerables a pérdida de datos y a la manipulación de los mismos sin autorización, tal y como podemos apreciar en la siguiente imágen, donde Bob, realiza peticiones objetos que pertenecen a Alice y la Api le responde con los datos solicitados.

Figura 22[18]: Ilustración autorización incorrecta a nivel de objeto

Para prevenir esta vulnerabilidad se recomienda:

- Aplicar un mecanismo de autorización adecuado que depende de las políticas de los usuarios y su jerarquía .

[16] Web oficial: https://owasp.org/about/
[17] Recuperado de la página: https://owasp.org/www-project-api-security/ el día 20/04/2024
[18] Recuperado de la página: https://oasishack.medium.com/api-rest-desde-0-api1-2023-660480525ac5 el dia 20/04/2024

- Hacer uso de valores aleatorios e impredecibles como GUIDs para los documentos de registro.

Autorización incorrecta a nivel de propiedad de objeto

En este tipo de vulnerabilidad, aunque la Api, tenga implementado un control de acceso adecuado a nivel de objeto, puede permitir a un usuario acceder a propiedades específicas de un objeto al cual no debería de ser accesible dado su rol.
Por ejemplo, si un usuario solicita información sobre su perfil, este debería devolver propiedades como nombre, correo electrónico o teléfono. Pero en este caso la APi no filtra bien los datos, respondiendo por tanto con datos sensibles del usuario, produciendose una violación en la privacidad de un usuario.

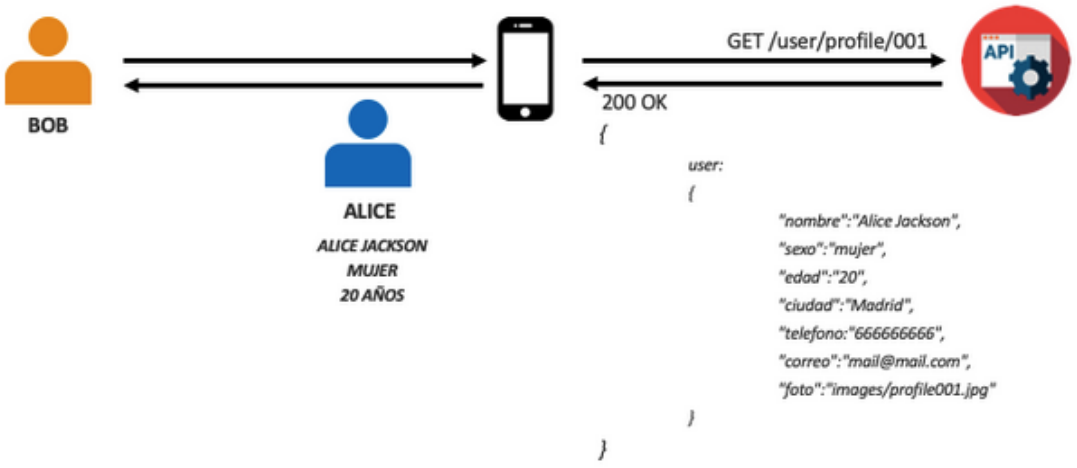


Figura 23[19]: Ilustración Autorización incorrecta a nivel de propiedad de objeto

Para prevenir esta vulnerabilidad se recomienda:

- Mantener las estructuras de datos devueltas al mínimo, de acuerdo con las funcionalidades del endpoint.

[19] Recuperado de la página: https://oasishack.medium.com/api-rest-desde-0-api1-2023-660480525ac5 el dia 20/04/2024

- Implementar un mecanismo de validación de respuestas basado en esquemas como una capa extra de seguridad.

Falsificación de solicitudes del lado del servidor

En este tipo de vulnerabilidades el atacante explota una vulnerabilidad en la APi, que le permite controlar la entrada a una solicitud del lado del servidor, es decir, el atacante es capaz de manipular la aplicación de manera que realice solicitudes a otros sistemas o recursos en el servidor donde esté alojada la aplicación.
Permitiendo al atacante acceder a recursos sensibles a los cuales normalmente no tendría acceso.

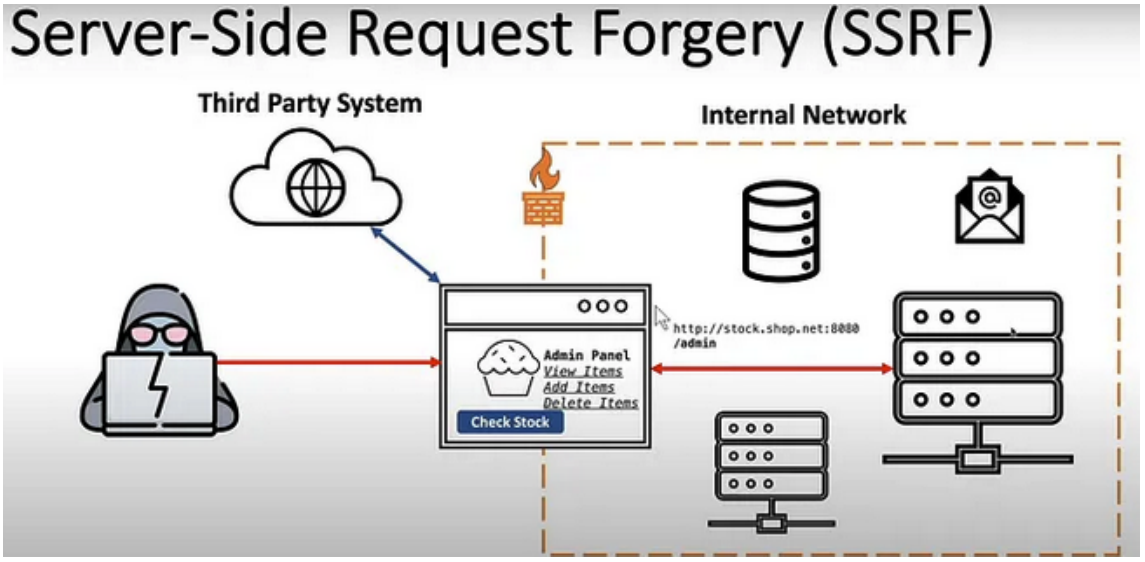


Figura 24[20]: Ilustración Falsificación de solicitudes del lado del servidor

Para prevenir esta vulnerabilidad se recomienda:

- Desactivar la redirección HTTP/HTTPS
- Evitar la respuestas "crudas" a los clientes
- Diseño y filtrado de URLs y puertos.

[20] Recuperado de la página: https://medium.com/@adithyakrishnav001/server-side-request-forgery-ssrf-bf23802cfb12 el día 20/04/2024

Acceso sin restricciones a flujo comerciales sensibles

En este tipo de vulnerabilidad, le permite al usuario saltarse el flujo normal de la aplicación e ir a flujos alternativos que no corresponden al flujo natural de la misma.
Esto ocurre debido a que la API, no implementa medidas suficientes para proteger la utilización de recursos más restringidos, es decir no se requiere autenticación para hacer uso de los mismos.

Para prevenir esta vulnerabilidad se recomienda:

- Autorización basada en roles.
- Cifrado de datos.
- Auditorías regulares de seguridad

Consumo inseguro de APIs

Este tipo de vulnerabilidad se refiere a la forma en la que son utilizadas las APIs de terceros, que acaban poniendo en riesgo la integridad de la aplicación.

La vulnerabilidad puede manifestarse de varias formas:

1. **Exposición de información sensible:** Si una API, devuelve más información de la que realmente es necesaria, puede exponer datos sensibles, que comprometan la seguridad de la aplicación.

2. **Falta de validación de entrada/salida:** Si una API, no valida los datos de entrada que recibe de los usuarios, esta queda expuesta a ataques de inyección de código o a la manipulación de datos.

3. **Falta de autenticación:** Si la API, no requiere autenticación o no implementa controles de de autorización puede permitir que personas nos autorizadas accedan a datos sensibles.

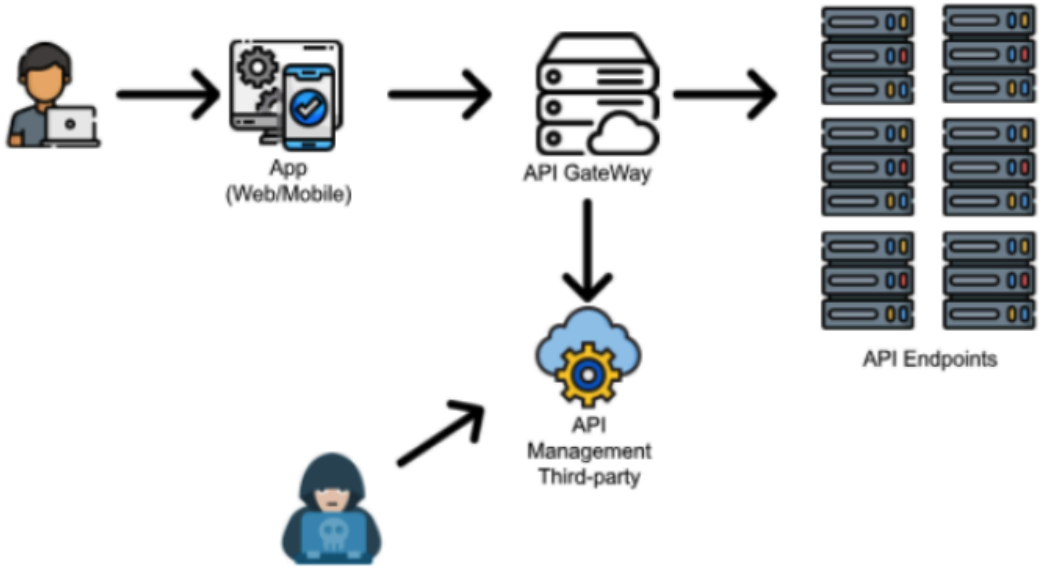


Figura 25[21]: Ilustración consumo inseguro de APIs

Para prevenir esta vulnerabilidad se recomienda:

- Asegurarse que todas las comunicaciones se realicen a través de un canal seguro de comunicación TLS.

- Validar y desinfectar adecuadamente los datos recibidos de las API integradas antes de usarlos.

- Controlar y validar las redirecciones que la API pueda realizar.

5. Desarrollo de la aplicación backend

Tal y como se ha expuesto en las secciones anteriores, se va a utilizar una metodología de desarrollo ágil, basado en incrementos, más concretamente Kanban, llevando a cabo un desarrollo más flexible y centrado en los resultados.

[21] Recuperado de la página: https://www.linkedin.com/pulse/gu%C3%ADa-de-owasp-para-apis-2023-base4-security/ el dia 20/04/2024

5.1 Primer incremento

En el primer incremento, es fundamental realizar un proceso de estudio y análisis sobre cómo llevar a cabo la implementación de la estructura del proyecto. Ya que previamente a iniciar el desarrollo de un producto software, debemos realizar estimaciones rigurosas del esfuerzo y tiempo necesario para ello, tal y como hemos analizado en secciones previas.

Además de unas tareas iniciales, relacionadas con la preparación del entorno de desarrollo para el equipo de trabajo.

De esta manera aseguramos un inicio eficiente y minimizamos los posibles obstáculos durante el desarrollo del proyecto.

5.1.1 Historias de usuario.

En este apartado, se van a presentar las historias de usuario que se han llevado a cabo durante el primer incremento. Las historias de usuario son las siguientes:

- HU-01: Registrar Usuario
- HU-02: Iniciar Sesión
- HU-03: Cerrar sesión
- HU-04: Entrenamiento modelo de predicción

Historia de Usuario - Registrar usuario	
Versión	1.0(24/02/2024)
Autor/es	Adrián Arboledas Fernández
ID	HU-01

Fuentes	Usuario base
Propósito	Registrar a un usuario en el sistema.
Descripción	Permite al usuario crear una cuenta, para acceder a todas las funcionalidades del API
Prioridad	Alta
Validación	• Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. • Al realizar la petición si el email, ya se encuentra registrado, se muestra un mensaje indicando que el usuario ya está en el sistema • Al realizar la petición con datos válidos, se registra el usuario en estado "no confirmado". • El usuario recibe un correo electrónico de confirmación del usuario. • Si al realizar la petición para confirmar el usuario, el código OTP es erróneo, el sistema devolverá un mensaje indicando que el código no es válido. • El usuario tras enviar el código OTP correctamente su cuenta debe de activarse correctamente. • Tras activarse la cuenta del usuario este puede proceder a iniciar sesión.

Tabla 13: Historia de usuario HU-01

Historia de Usuario - Iniciar sesión	
Versión	1.0(24/02/2024)
Autor/es	Adrián Arboledas Fernández
ID	HU-02
Fuentes	Usuario base
Propósito	Iniciar sesión en el sistema.
Descripción	Permite al usuario iniciar sesión en el API, con los datos utilizados en su registro.
Prioridad	Alta
Validación	• Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. • Al realizar la petición si el email no se encuentra registrado en el sistema, se devuelve un mensaje de error. • El usuario es capaz de iniciar sesión, con los datos utilizados en el registro, obteniendo el token de cognito, necesario para posteriores llamadas a los endpoint de la API.

Tabla 14: Historia de usuario HU-02

Historia de Usuario - Cerrar sesión	
Versión	1.0(24/02/2024)
Autor/es	Adrián Arboledas Fernández
ID	HU-03
Fuentes	Usuario base
Propósito	Cerrar sesión en el sistema.
Descripción	Permite al usuario cerrar sesión en el API, haciendo uso de su token personal de cognito.
Prioridad	Alta
Validación	• Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. • Si el bearer token no se encuentra en activo, el usuario no podrá hacer uso del recurso. • El usuario es capaz de cerrar sesión y su token de cognito es inhabilitado.

Tabla 15: Historia de usuario HU-03

Historia de Usuario - Entrenar modelos	
Versión	1.0(24/02/2024)

Autor/es	Adrián Arboledas Fernández
ID	HU-04
Fuentes	Usuario base
Propósito	Entrenar modelo de predicción de transacciones.
Descripción	Permite al usuario entrenar un modelo de predicción de transacciones personalizado.
Prioridad	Alta
Validación	• Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. • Si el bearer token no se encuentra en activo, el usuario no podrá hacer uso del recurso. • Si el modelo de predicción a entrenar, ya se encuentra en la base de datos, el sistema devuelve un mensaje indicándolo. • Al realizar la petición con los datos válidos, el usuario es capaz de entrenar un modelo de predicción correctamente. • El usuario ya tiene el modelo de datos asociado a su cuenta.

Tabla 16: Historia de usuario HU-04

5.1.2 Análisis

En este punto, vamos a mostrar los diagramas de secuencia asociados a cada historia de usuario que hemos expuesto anteriormente.

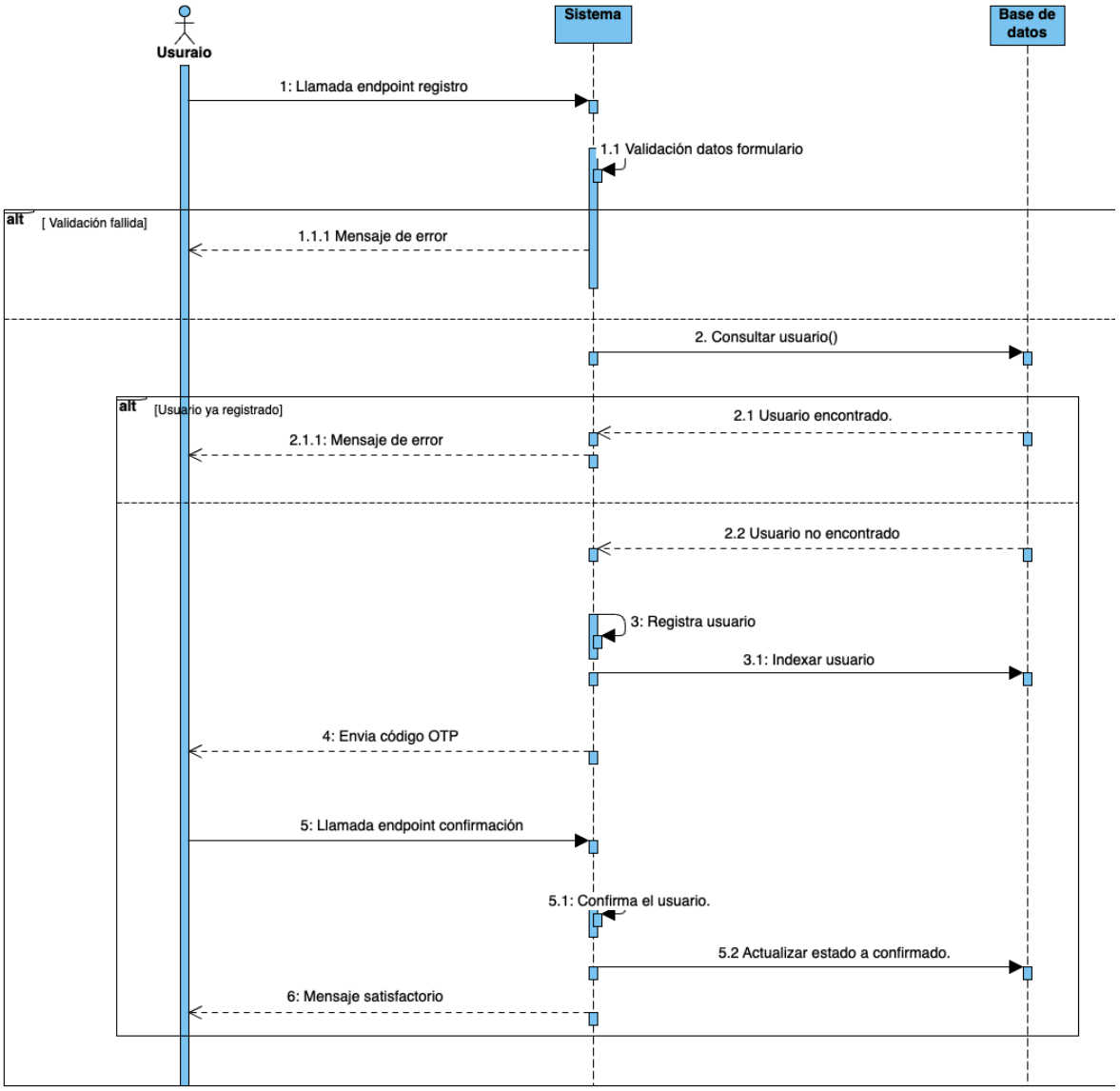


Figura 26: Diagrama de secuencia - Historia de usuario HU-01

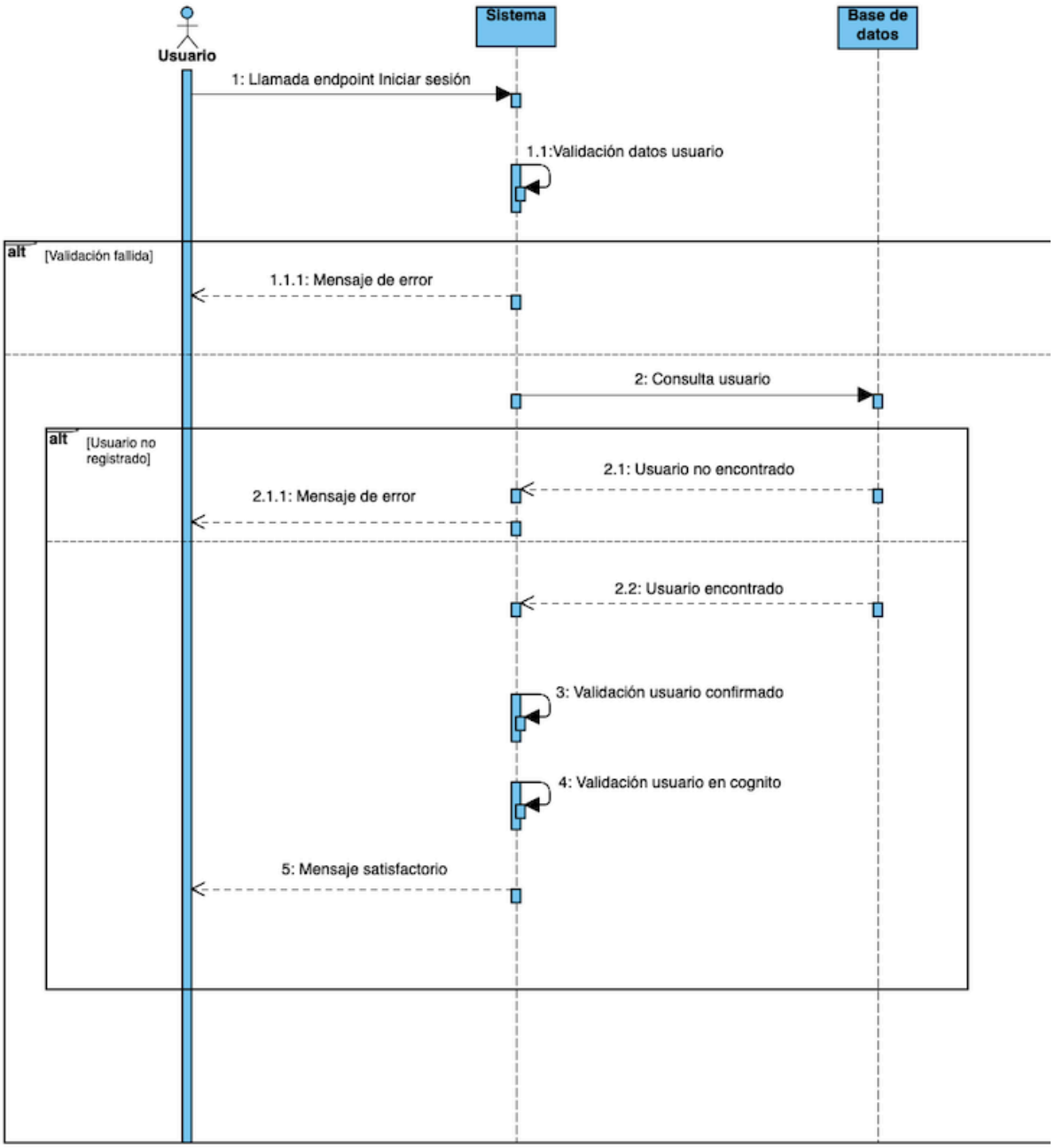


Figura 27: Diagrama de secuencia - Historia de usuario HU-02

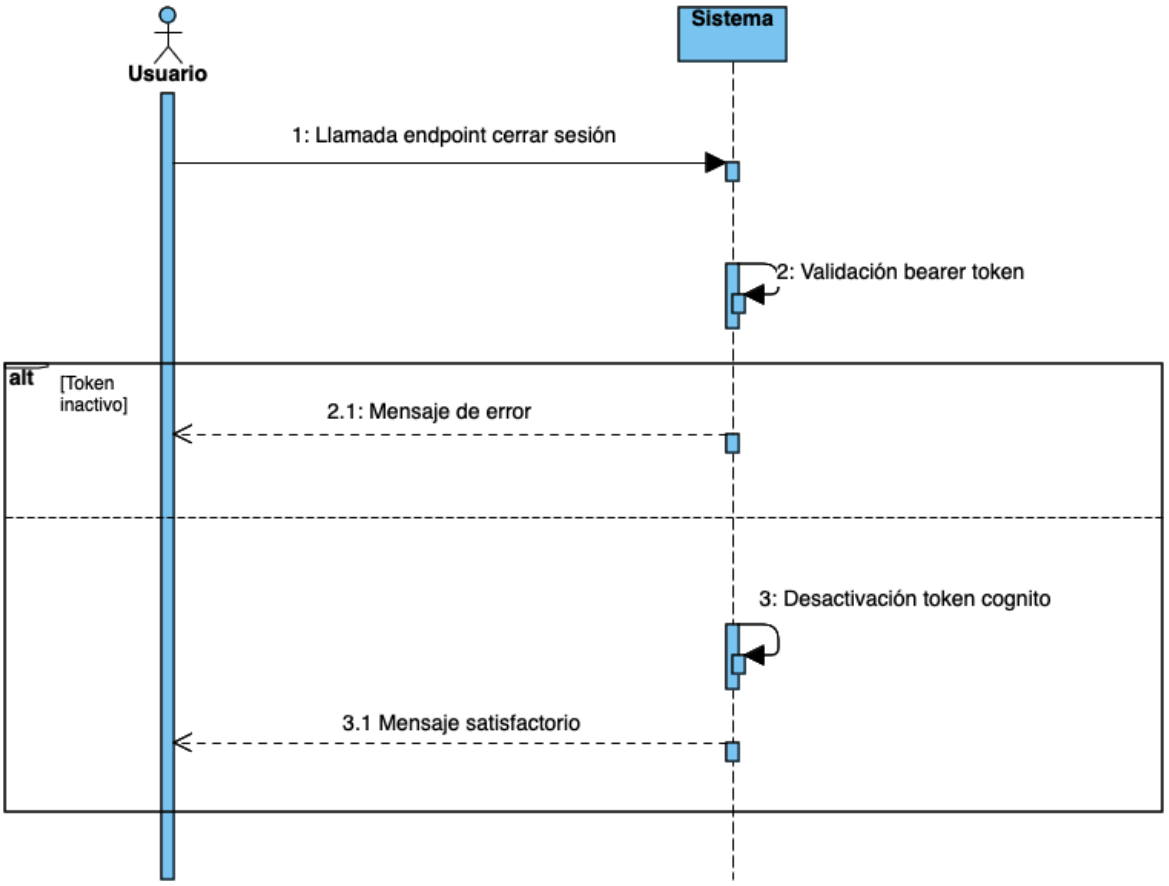


Figura 28: Diagrama de secuencia - Historia de usuario HU-03

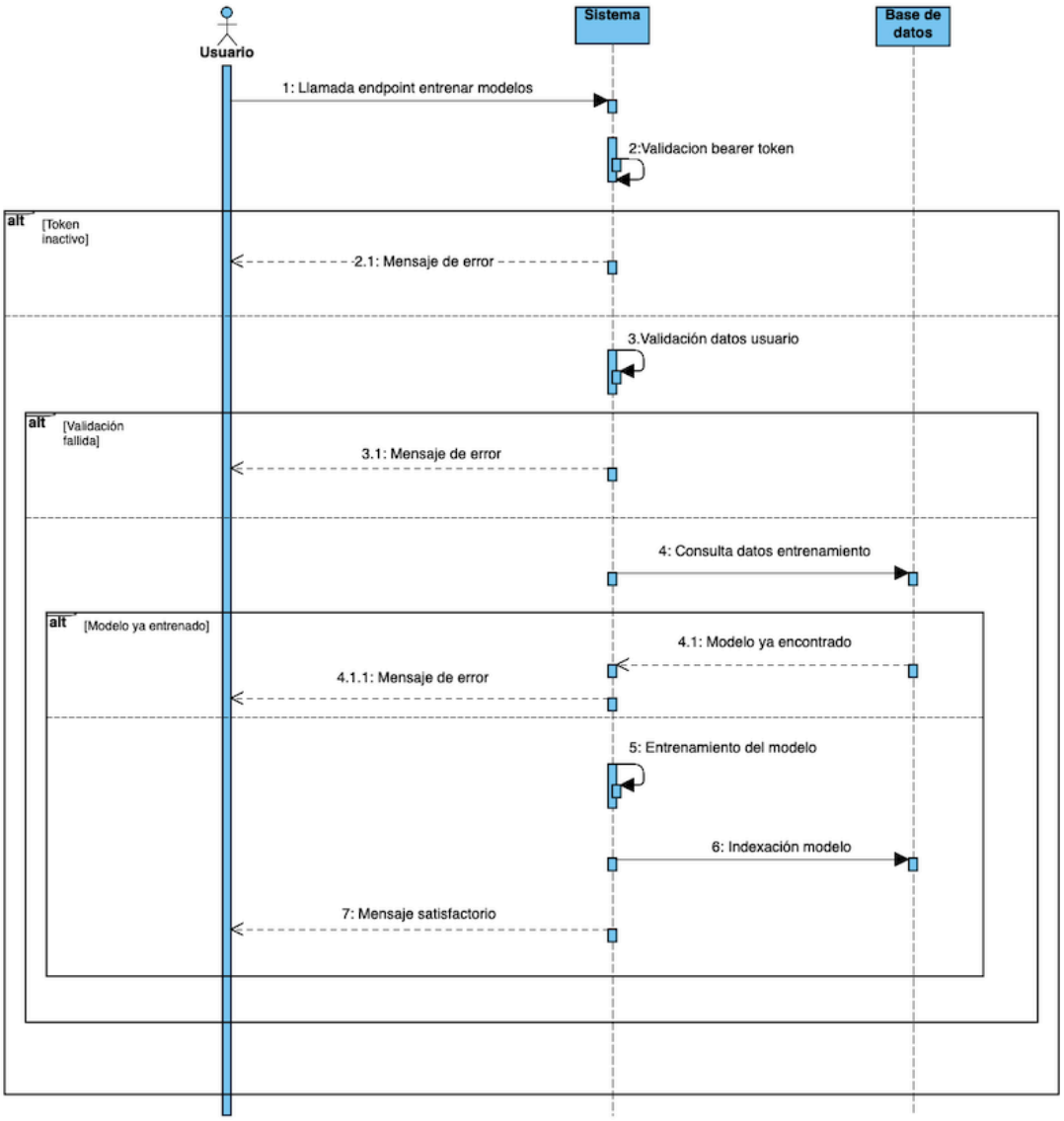


Figura 29: Diagrama de secuencia - Historia de usuario HU-04

5.1.3 Entrega y reunión final del incremento

Durante el primer incremento de nuestro desarrollo, se ha llevado a cabo la implementación de la creación y validación de usuarios, junto con el módulo que permite el entrenamiento de modelos de predicción personalizados.

A lo largo del proceso de desarrollo, nos hemos encontrado con diversos desafíos, siendo el más destacable el relacionado con el módulo de entrenamiento de modelos, dado que al permitir el uso de cualquier conjunto

de datos para el entrenamiento, se detectó que al emplear el conjunto completo, el tiempo necesario para el entrenamiento del modelo se incrementan considerablemente. En respuesta se ha implementado una validación que impide el entrenamiento si no se especifican las columnas a omitir.

En términos generales este incremento ha resultado una buena muestra del alcance que puede tener el proyecto, por lo que se ha acordado seguir trabajando en el resto de los incrementos.

5.2 Segundo incremento

En este segundo incremento, continuaremos con el desarrollo de nuestro proyecto, implementando los módulos de predicción de transacciones y el de gestión de modelos entrenados, permitiendo conocer cuáles son los modelos que actualmente tenemos implementados y teniendo la posibilidad de eliminar cualquiera de ellos.

5.2.1 Historia de usuario

En este apartado, se van a presentar las historias de usuario que se han llevado a cabo durante el segundo incremento. Las historias de usuario son las siguientes:

- HU-05 Predecir transacciones.
- HU-06 Listar modelos.
- HU-07 Borrar modelos.

Historia de Usuario - Predecir transacciones	
Versión	1.0(24/02/2024)
Autor/es	Adrián Arboledas Fernández
ID	HU-05
Fuentes	Usuario base
Propósito	Predecir legitimidad de una transacción
Descripción	Permite al usuario predecir si una transacción de tarjeta es fraudulenta o legítima.
Prioridad	Alta
Validación	• Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. • Si el bearer token no se encuentra en activo, el usuario no podrá hacer uso del recurso. • El usuario debe de ser capaz de utilizar el modelo de datos ya entrenado. • Una vez el usuario utiliza el modelo de predicción con una transacción en concreto, recibe un mensaje indicando si la transacción es fraudulenta o legítima.

Tabla 17: Historia de usuario HU-05

Historia de Usuario - Listar modelos	
Versión	1.0(24/02/2024)
Autor/es	Adrián Arboledas Fernández
ID	HU-06
Fuentes	Usuario base
Propósito	Listar los modelos entrenados
Descripción	Permite al usuario listar los modelos entrenados que se encuentran asociados a su cuenta.
Prioridad	Alta
Validación	• Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. • Si el bearer token no se encuentra en activo, el usuario no podrá hacer uso del recurso. • El usuario debe de ser capaz de obtener la información de sus modelos entrenados.

Tabla 18: Historia de usuario HU-06

Historia de Usuario - Eliminar modelos	
Versión	1.0(24/02/2024)
Autor/es	Adrián Arboledas Fernández
ID	HU-07

Fuentes	Usuario base
Propósito	Eliminar los modelos entrenados
Descripción	Permite al usuario eliminar los modelos entrenados que se encuentran asociados a su cuenta.
Prioridad	Alta
Validación	● Al realizar la petición con el body vacío, se muestra un mensaje de error indicando los campos vacíos. ● Si el bearer token no se encuentra en activo, el usuario no podrá hacer uso del recurso. ● El usuario debe de ser capaz de eliminar la información de sus modelos entrenados.

Tabla 19: Historia de usuario HU-07

5.2.2 Análisis

En este punto, vamos a mostrar los diagramas de secuencia asociados a cada historia de usuario que hemos expuesto anteriormente.

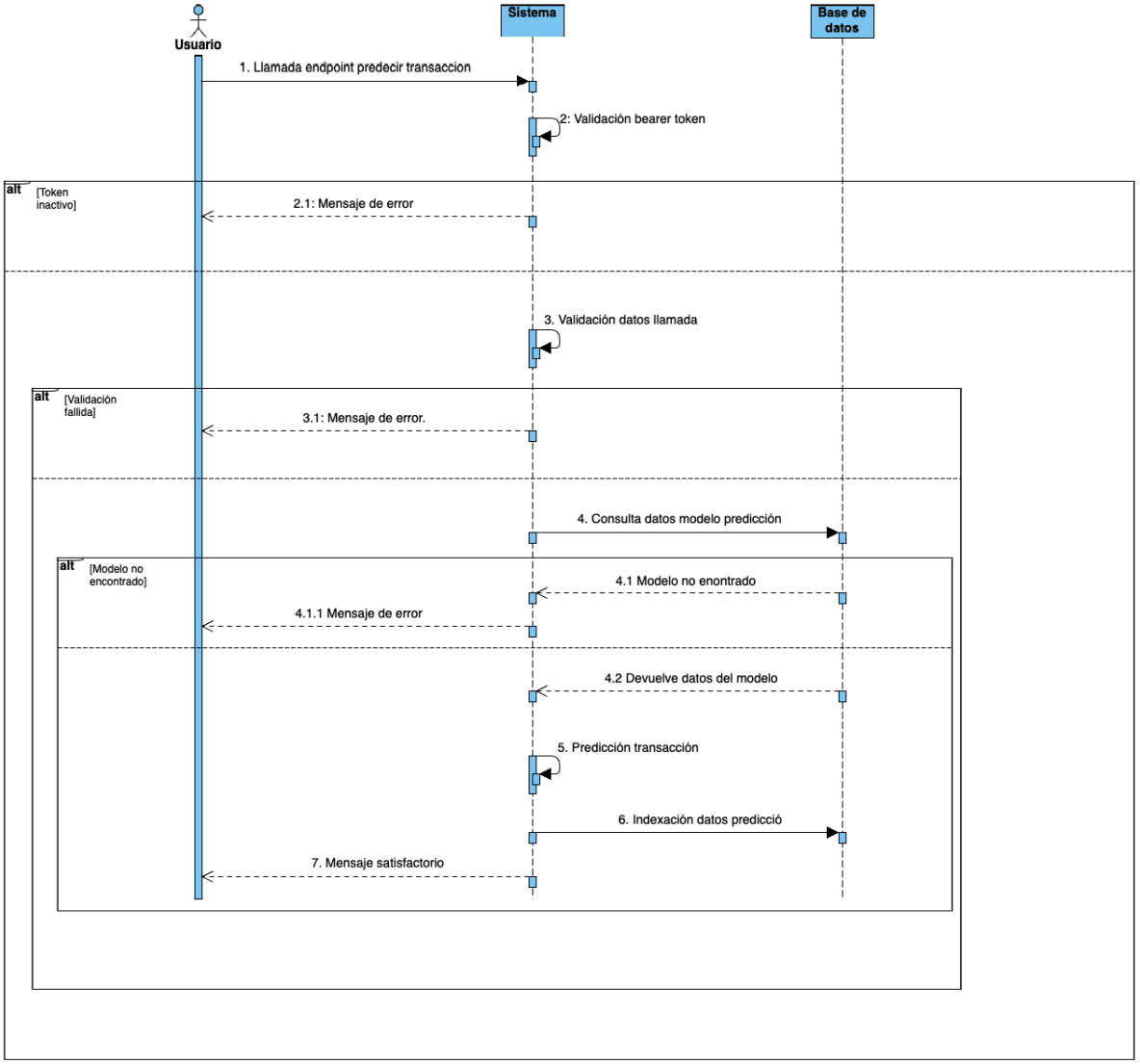


Figura 30: Diagrama de secuencia - Historia de usuario HU-05

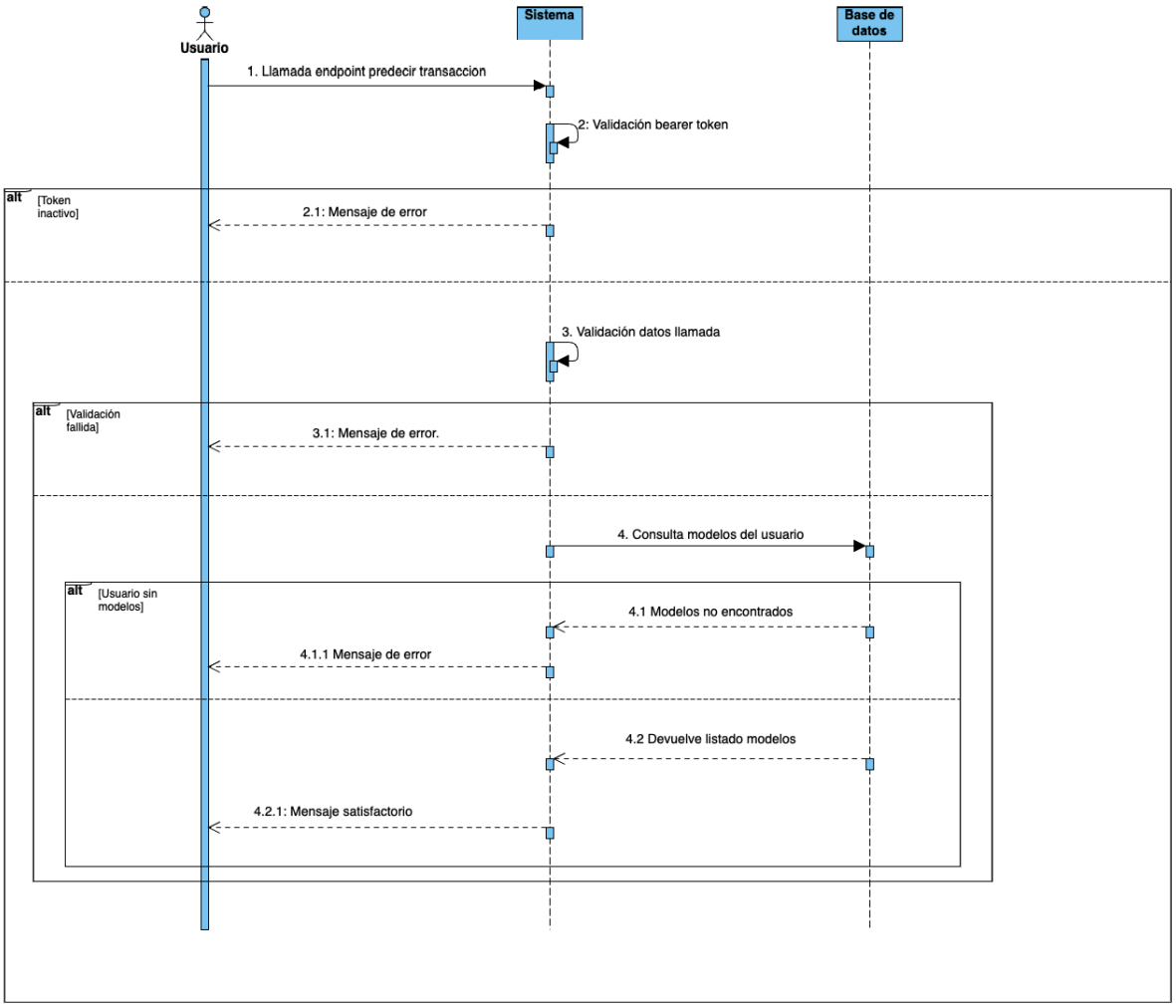


Figura 31: Diagrama de secuencia - Historia de usuario HU-06

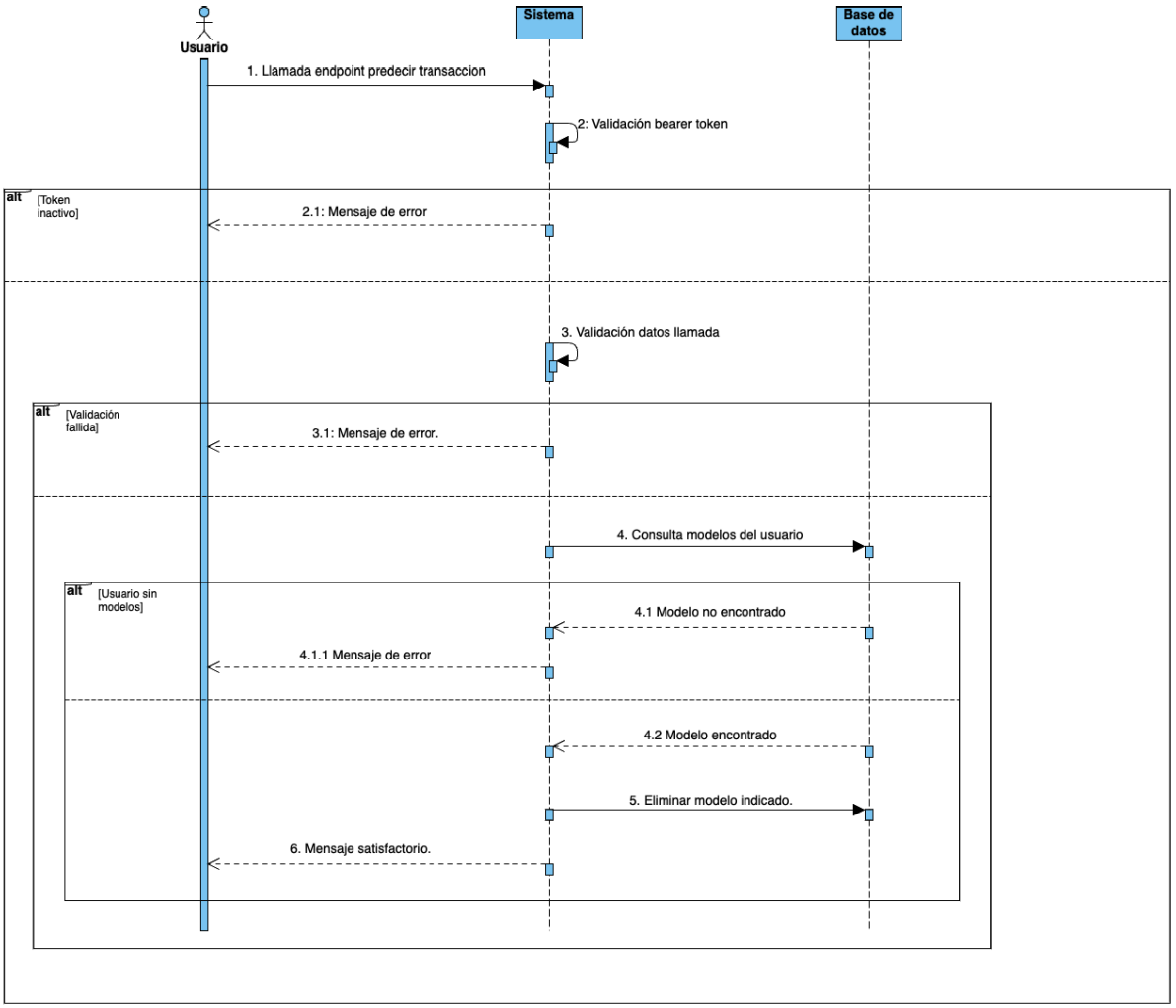


Figura 32: Diagrama de secuencia - Historia de usuario HU-07

5.2.3 Entrega y reunión final del incremento

Dado que nos encontramos en el incremento final, cabe resaltar que todas las historias de usuarios previamente planificadas han sido implementadas con éxito.

De cara al cliente es importante destacar que se encuentran enormemente satisfechos y agradecidos por el esfuerzo realizado en el proyecto. Proponiendo futuras mejoras, como la creación de un rol de administrador.

Por tanto podemos decir que el proyecto ha sido todo un éxito, ya que se ha conseguido satisfacer todas las necesidades del cliente, superando incluso

sus propias expectativas, lo que se traducirá en nuevos proyectos o mejoras del mismo por parte del cliente.

5.3 Encuesta de satisfacción

Hemos recopilado las respuestas de una encuesta realizada anónimamente por usuarios que han hecho uso del API, de manera que podemos obtener una idea de cómo se ve nuestro proyecto de cara al público y a su vez recoger datos, para saber en qué parte es necesaria realizar mejoras.

La población de respuestas a la encuesta, está conformada en su totalidad, por compañeros de clase, es decir, una población con altos conocimientos en el mundo de la informática y uso de APIs, cuya edad oscila entre los 20 y 30 años.
Gracias a las respuestas de esta población tan específica, nos ofrece una idea más cercana del acogimiento de las API en el mundo profesional de la informática.

A continuación, podemos ver las respuestas de los encuestados.

Figura 33: Resultado encuesta - Pregunta 1

Figura 34: Resultado encuesta - Pregunta 2

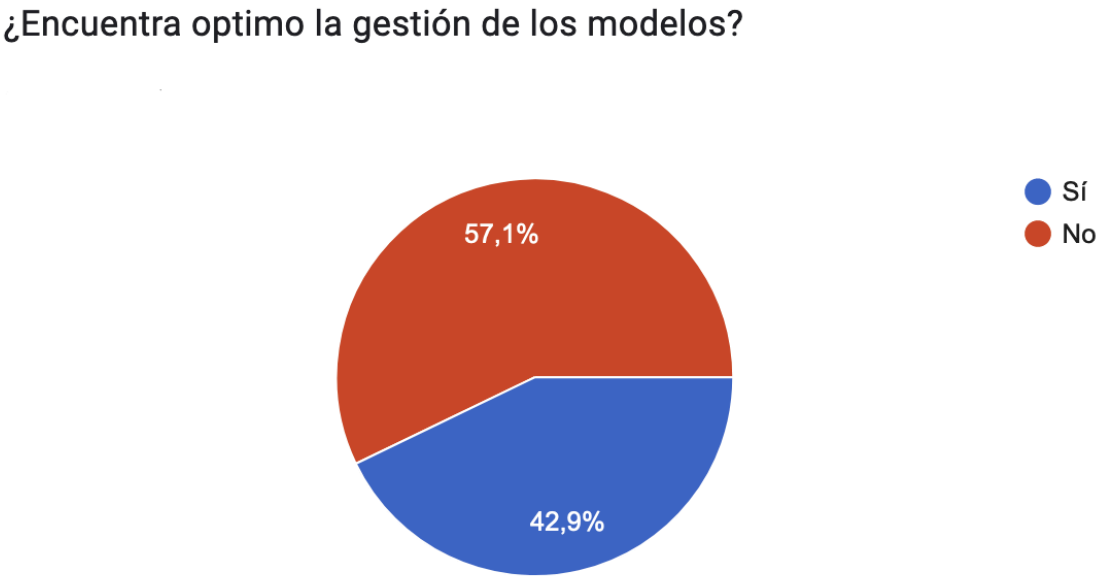


Figura 35: Resultado encuesta - Pregunta 3

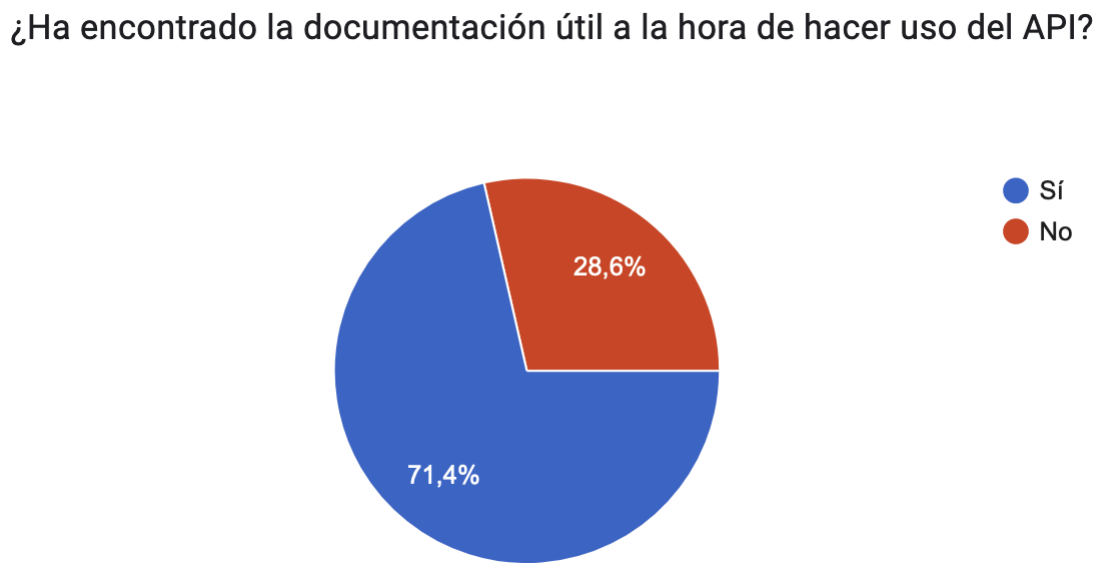


Figura 36: Resultado encuesta - Pregunta 4

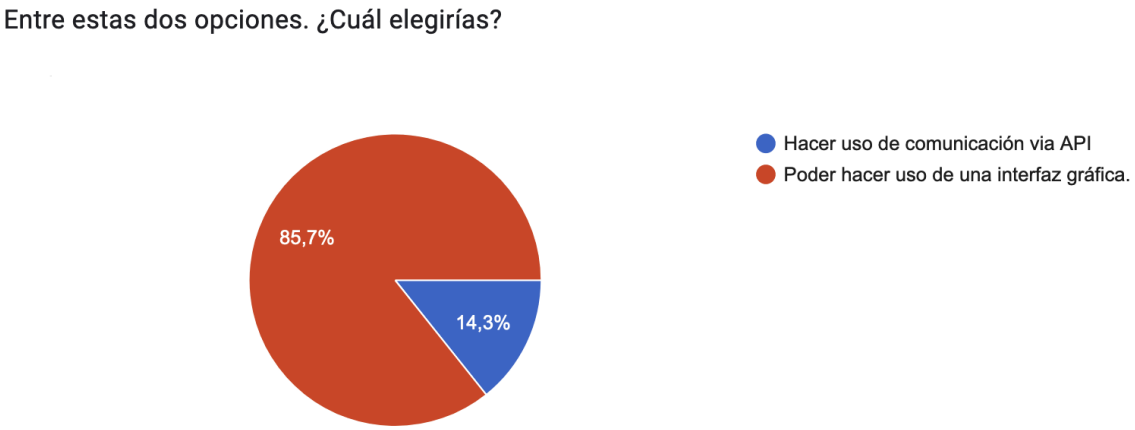


Figura 37: Resultado encuesta - Pregunta 5

En cuanto a los resultados de las encuestas, podemos apreciar como los usuarios están bastante satisfechos con el rendimiento que proporciona el API, así como su facilidad para la instalación y posterior uso de la misma.

Cabe resaltar que también hay dos aspectos enormemente a mejorar como son la gestión de los usuarios, tal y como el cliente nos recomendó al final del segundo incremento y el disponer de una interfaz gráfica para poder manejar

de una forma más amigable e intuitiva la realización de predicciones de transacciones.

No obstante, en aspectos generales, podemos asegurar que ha sido todo un éxito la acogida por parte de los usuarios.

6. Conclusiones

En el apartado 1.2 se definió el objetivo principal y los específicos de este TFG. A continuación se repasan y se confirma si se han alcanzado a la finalización de este proyecto.

1. **Objetivo principal:**
   Crear backend impulsado por el machine learning que nos permita detectar en tiempo real cuando se produce una transacción fraudulenta
   a. **Estado:** Alcanzado.
   b. **Justificación:** Tras las exhaustivas pruebas realizadas, el backend ha sido capaz de detectar una transacción fraudulenta con un índice de acierto del 98%.

2. **Objetivo específico 1:** Gestión de usuario
   a. **Estado:** Alcanzado.
   b. **Justificación:** Los usuarios han podido registrarse, iniciar/cerrar sesión en el sistema de manera satisfactoria, permitiendo acceder a sus propios recursos almacenados.

3. **Objetivo específico 2:** Entrenamiento de modelos
   a. **Estado:** Alcanzado.
   b. **Justificación:** La aplicación backend es capaz de entrenar un modelo de machine learning específico para cada usuario, ya que permite utilizar datasets propios.

4. **Objetivo específico 3:** Predicción de transacciones fraudulentas
   a. **Estado:** Alcanzado.
   b. **Justificación:** El usuario es capaz de elegir entre todos los modelos que tiene entrenados, a la hora de realizar la predicción, pudiendo utilizar siempre el modelo que más se adapte al tipo de transacciones en cuestión.

5. **Objetivo específico 4:** Listado de modelos entrenados

   a. **Estado:** Alcanzado.
   b. **Justificación:** El usuario tiene la posibilidad de listar todos los modelos entrenados previamente, junto a su información asociada.

6. **Objetivo específico 5:** Gestión de modelos
   a. **Estado:** Alcanzado.
   b. **Justificación:** El usuario tiene la posibilidad de que una vez listados lo modelos entrenados, poder borrar aquellos que ya no le resulten de utilidad

Como hemos podido comprobar a lo largo de la realización del proyecto, el mundo de las transacciones bancarias online es cada vez más frecuente, creciendo de forma abrumadora cada año que pasa. Por lo que es altamente necesario la incorporación de herramientas que nos permitan poder detectar en la medida de lo posible, aquellas transacciones realizadas de forma fraudulenta o no autorizadas, con el fin de establecer un esquema seguro y confiable a la hora de utilizar la banca online.

El desarrollo del proyecto ha supuesto a nivel personal una serie de desafíos significativos, los cuales he tenido que abordar con eficiencia y determinación. Esto me ha permitido ampliar y desarrollar mis conocimientos en áreas que no terminaba de dominar, como ha podido ser en la planificación y gestión de tareas y su correspondiente administración de tiempos.

En aquellas áreas que ya tenía ciertos conocimientos asentados me ha permitido consolidar las bases y explorar nuevos enfoques y soluciones, lo que me ha permitido afrontar nuevos desafíos con una perspectiva renovada y un mayor abanico de opciones.

Anexo

Manual de instalación y ejecución API

Este manual proporciona las instrucciones paso a paso para configurar y lanzar la aplicación backend. La aplicación al encontrarse dockerizada, es compatible con cualquiera de los siguientes sistema operativos: Linux, Windows y MacOs.

Para hacer uso de la aplicación asegúrate de hacer uso de los requisitos previos y los pasos proporcionados a continuación.

Requisitos previos:

1. **Docker Desktop:** La aplicación banking Fraud, utiliza contenedores docker para facilitar la implementación y ejecución del entorno de desarrollo. Puedes descargar e instalar Docker Desktop desde el siguiente enlace: https://www.docker.com/products/docker-desktop/

2. **Visual studio Code:** Se recomienda utilizar Visual studio Code, como editor de código para trabajar en la aplicación Banking Fraud. Puede descargar e instalar Visual Studio code desde el siguiente enlace: https://code.visualstudio.com/

3. **Amazon web Service:** La aplicación Banking Fraud, utiliza como gestor de usuarios la aplicación de cognito de AWS. Puede crear una cuenta en el siguiente enlace: https://aws.amazon.com/es/console/

Pasos de instalación

1. **Clonar repositorios:** Abre una terminal o línea de comandos y ejecuta el siguiente comando para clonar el backend de la aplicación

```
git clone https://github.com/Adry2317/BankingFraudTFG.git
```

2. **Descargar y levantar la base de datos:** Dado que la base de datos al encontrarse dockerizada es muy pesada, no puede clonarse desde git, por lo que será necesario descargarla desde el siguiente enlace.

   Enlace:https://drive.google.com/drive/folders/1uzjRG-z2bzBUluxECqoBtw9IuNSmY6ry?usp=sharing

   Una vez descargado, descomprimimos el archivo .zip, que genera google drive y encontraremos dos archivos bajo el nombre: *kibana.tar y es01.tar.*

   Ejecutamos la aplicación docker y abrimos una terminal o línea de comandos de nuestro sistema operativo y ejecutamos en orden los siguientes comandos.

   2.1 Creación de red privada:

   ```
   docker network create elastic
   ```

   2.2 Carga de archivos .tar en docker:

   ```
   docker load -i kibana.tar
   docker load -i es01.tar
   ```

   Al ejecutar estos comandos, obtendremos el nombre de la imágen de cada archivo.tar, que debemos usar para lanzar los contenedores.

   2.3 Lanzamiento de contenedores:

   ```
   docker run -d --name es01 --network elastic -p
   9200:9200
   docker.elastic.co/elasticsearch/elasticsearch:8.10.3
   ```

```
docker run -d --name kibana --network elastic -p
5601:5601  docker.elastic.co/kibana/kibana:8.10.3
```

Tras la ejecución de los comandos anteriores, en nuestra aplicación docker debemos de ver algo parecido a lo siguiente:

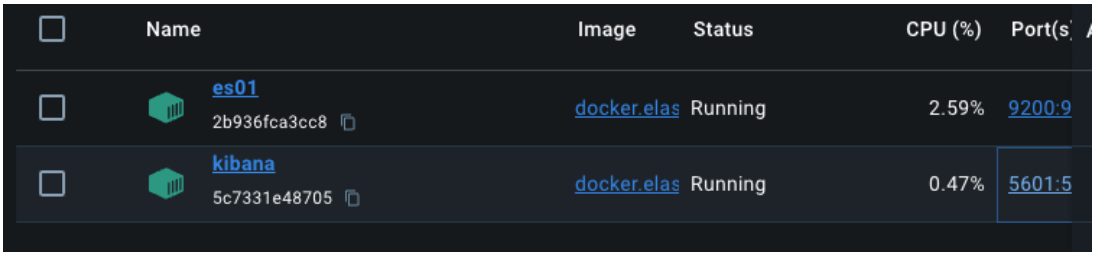


Figura 38: Anexo - Docker Base de datos

3. **Configuración AWS:** En nuestro proyecto como gestor de usuarios hemos utilizado Cognito, a continuación se muestran los pasos a seguir para configurarlo.

    3.1 **Configuración usuario IAM:** Para poder acceder a los servicios de aws desde nuestros microservicios es necesario disponer de unas credenciales de usuario, para ello nos dirigimos a la siguiente funcionalidad:

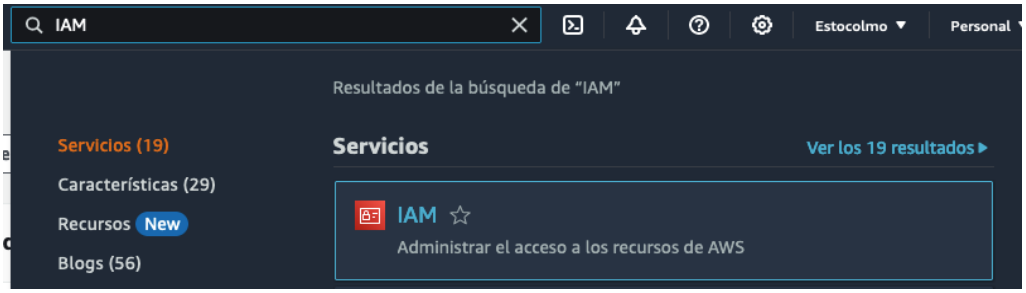


Figura 39: Anexo - AWS IAM

Una vez nos encontremos en la página principal, accedemos a la opción que se encuentra en la columna izquierda: *administración de acceso -> usuarios.*

Seleccionamos la opción Crear usuario:

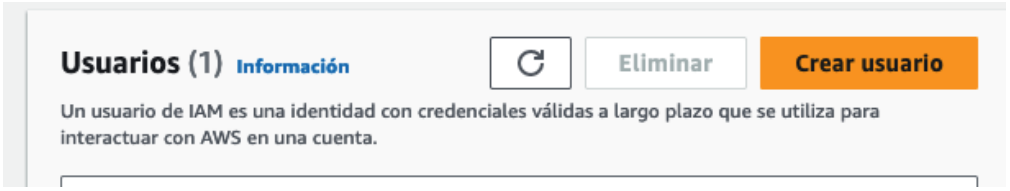


Figura 40: Anexo - Opción crear usuario IAM

Asignamos un nombre de usuario, a través del cual lo vamos a identificar.

Posteriormente, de las tres opciones de permisos, escogemos: *Adjuntar políticas directamente*

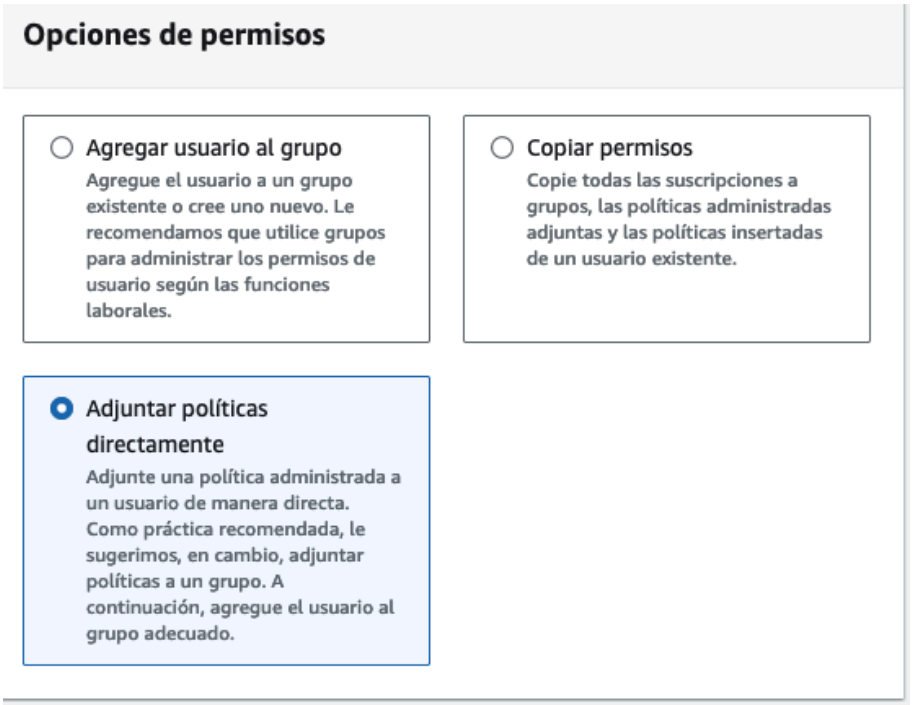


Figura 41: Anexo - IAM opciones de permiso

Una vez seleccionada la opción, añadimos las siguientes políticas:

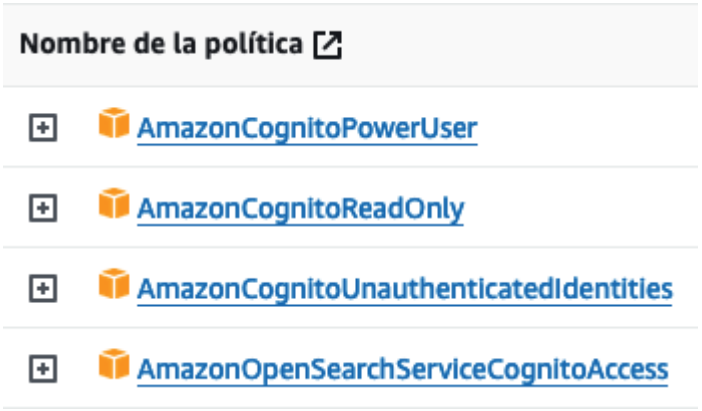


Figura 42: Anexo - Políticas IAM

Tras esto, nuestro usuario, ya aparecerá como creado en la pestaña "Usuarios", hacemos click sobre nuestro usuario y nos dirigimos a la opción *Credenciales de seguridad -> Claves de acceso* y hacemos click sobre *Crear clave de acceso.*

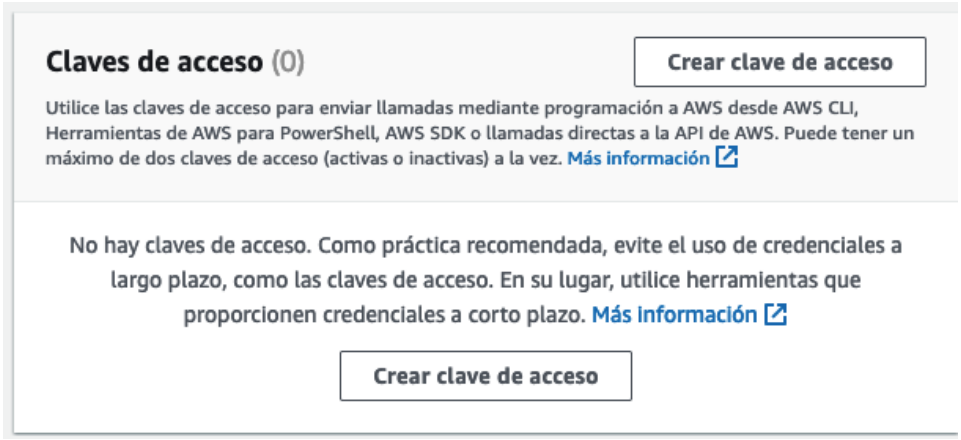


Figura 43: Anexo - Claves de acceso IAM

De todo el conjunto de opciones, seleccionamos la que se encuentra en primera posición:

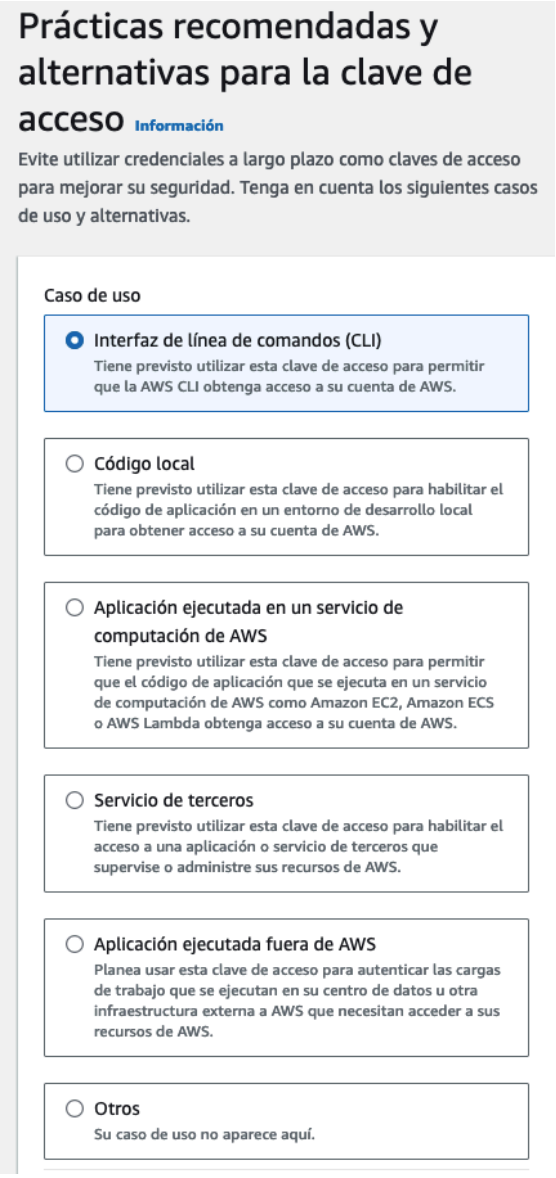


Figura 44: Anexo - casos de uso políticas IAM

El resto de los pasos, solo es necesario seleccionar la opción "siguiente". Una vez llegado al final, almacena los valores de las claves en un lugar seguro, ya que posteriormente haremos uso de las mismas

3.2 Configuración grupo de usuarios Cognito: Para poder almacenar y gestionar los usuarios que se registran dentro de nuestra aplicación backend, es necesario configurar Cognito.

Para ello nos dirigimos a la siguiente opción:

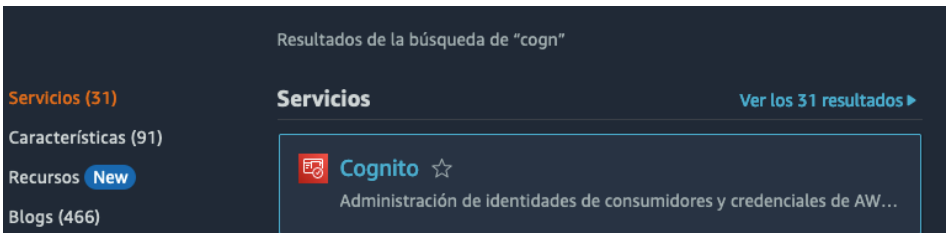


Figura 45: Anexo - AWS Cognito

Una vez nos encontremos en la página principal de cognito, pulsaremos "Crear grupo de usuarios":

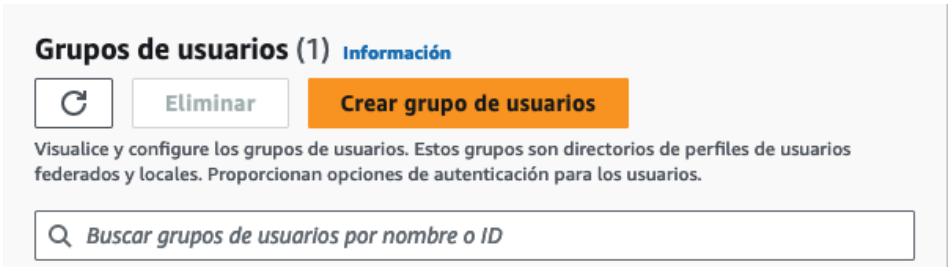


Figura 46: Anexo - Opción crear grupo usuarios Cognito

En el paso 1, seleccionamos como proveedor de autenticación la opción "grupo de usuarios de Cognito" y como opción de inicio de sesión, seleccionamos "Correo electrónico".

Figura 47: Anexo - Proveedores de autenticación Cognito

En el paso número dos, seleccionamos las siguientes opciones:

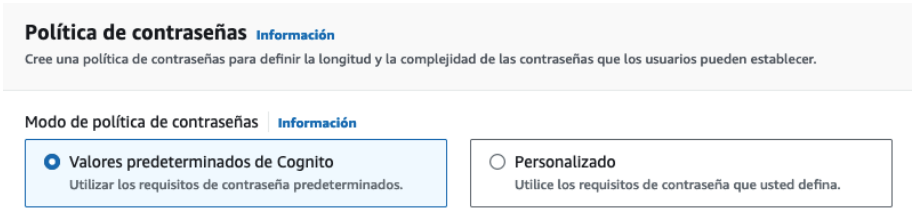


Figura 48: Anexo - Política de contraseñas Cognito

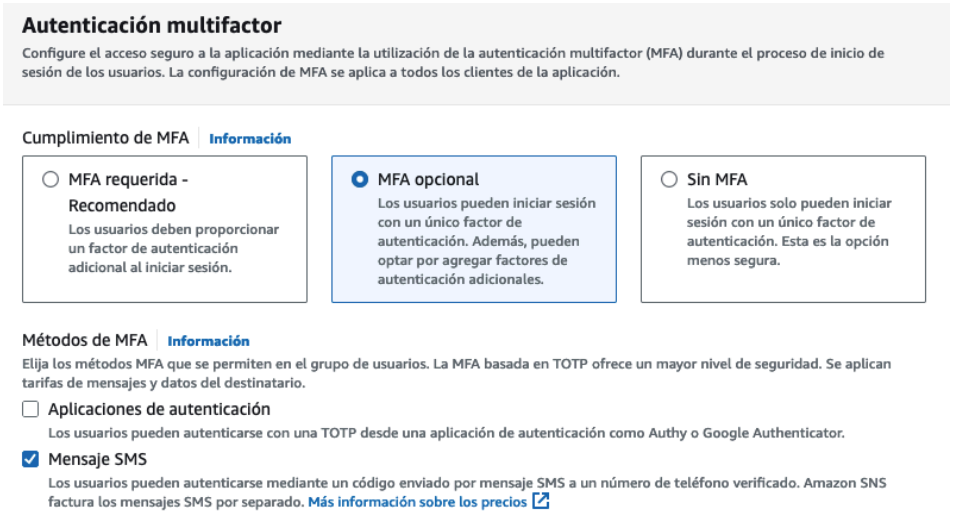


Figura 49: Anexo - Autenticación multifactor Cognito

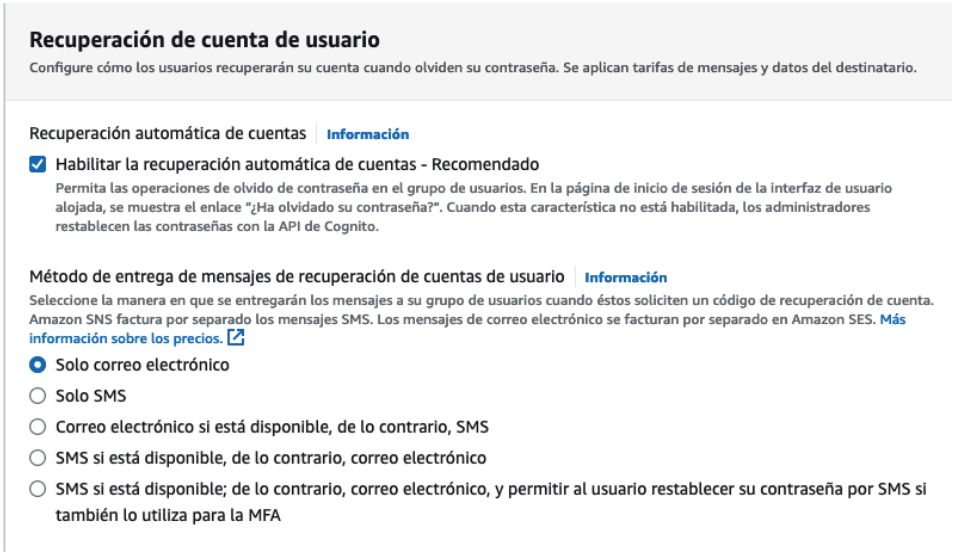


Figura 50: Anexo - Recuperación cuentas Cognito

En el paso número 3, dejamos las opciones por defecto y seleccionamos continuar.

En el paso número 4, seleccionamos la opción de enviar correo electrónico con Cognito:

Figura 51: Anexo - proveedor de correo electrónico Cognito

En el resto de los pasos, dejar las opciones por defecto y pulsar "continuar".

Si has seguido todos los pasos correctamente, en la página principal de Cognito debe de aparece el grupo de usuarios creado:

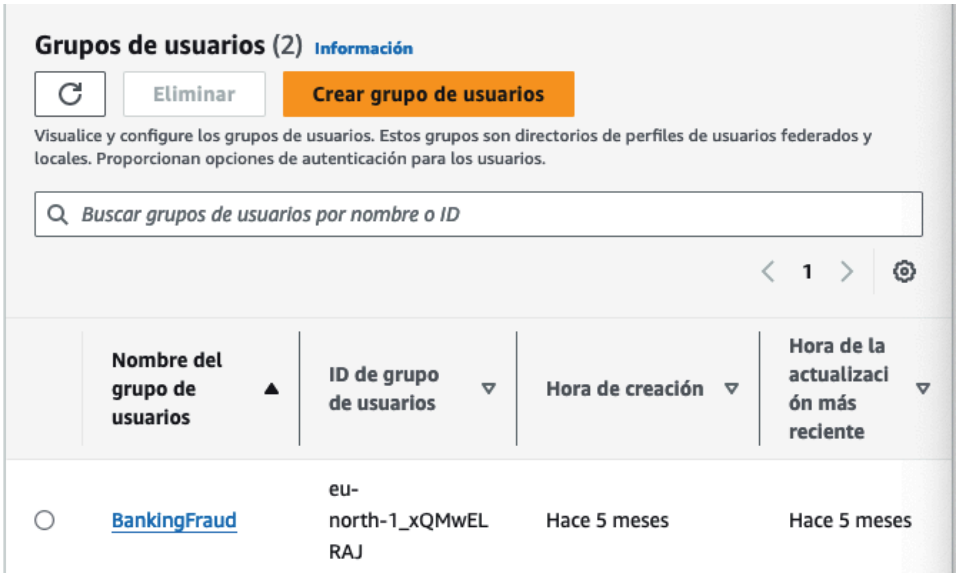


Figura 52: Anexo - Página principal Cognito

4. **Configuración y ejecución backend:** Una vez seguidos los pasos anteriores, tenemos todo lo necesario para que nuestra aplicación backend pueda ser ejecutada. Antes de nada, debemos de levantar los contenedores correspondientes a la base de datos, tal y como hemos mencionado anteriormente.

   Una vez levantados los contenedores de la base de datos, abrimos el proyecto copiado desde nuestro github con Visual Studio y nos dirigimos a la ruta *banking_fraud_lib -> banking_fraud_lib -> config.ini* y veremos un archivo de configuración con los siguientes campos:

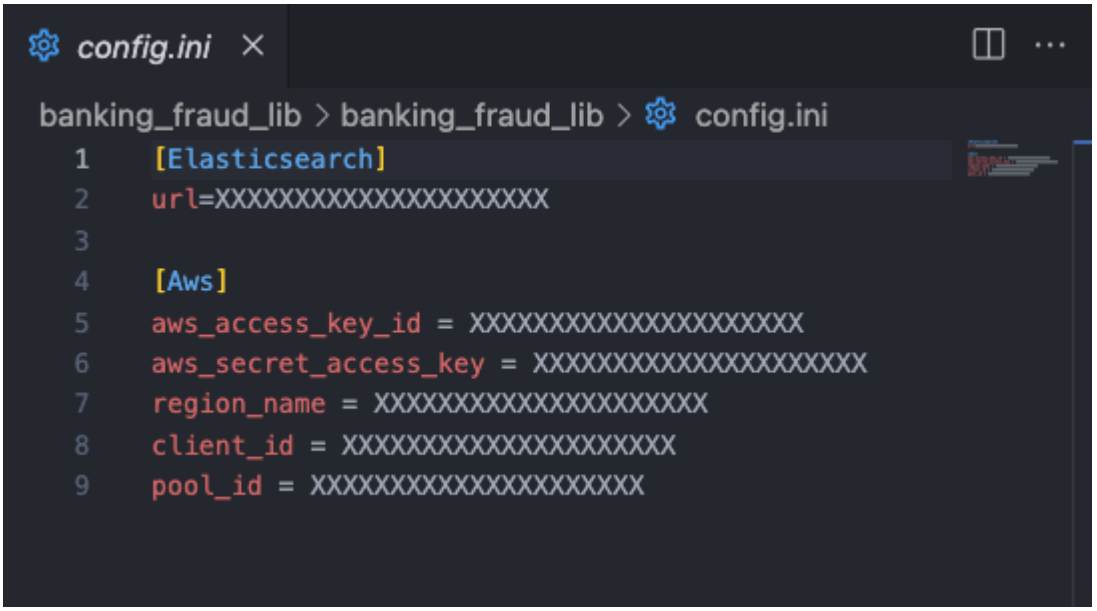


Figura 53: Anexo - Archivo configuración proyecto

En la etiqueta Aws, debemos de cambiar las "X", por los valores correspondientes de lo que hemos configurado en el paso anterior, mientras que la url de elasticsearch debe de se sustituida por la siguiente: http://elastic:87JmtoQsZ0OksRHgEj0f@172.20.0.2:9200

Una vez hemos rellenado el archivo de configuración, sólo será necesario ejecutar dos comandos, para ejecutar nuestra aplicación backend.

Para ello abrimos la consola de visual studio y ejecutamos lo siguiente:

```
docker build -t bankingfraud .
docker-compose up --build
```

Si el proceso ha terminado de forma satisfactoria, deberíamos de ser capaces de visualizar los siguiente en nuestra aplicación docker:

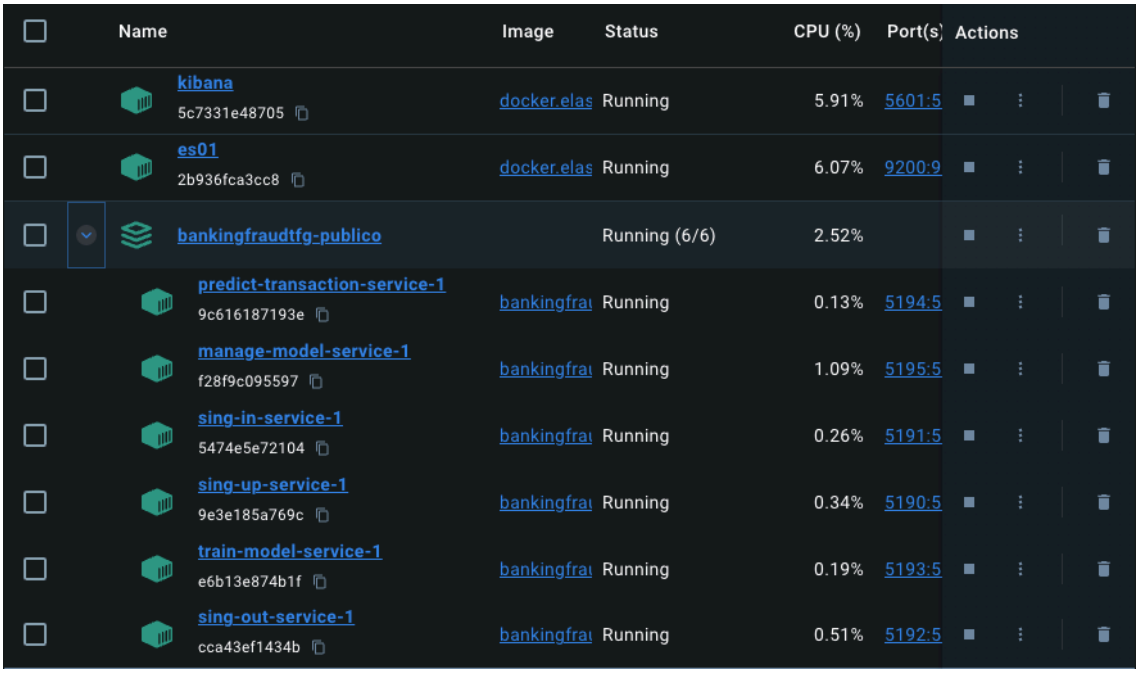


Figura 54: Anexo - Contenedores docker