

Entregable 3 de Gestión de Proyectos

Rol: Arquitecto-Diseñador

**Autores:**

Adrian Marqués Gaute

Carlos Raúl Robinson Thompson

**Tutores:**

Dra.C. Nayma Cepero Pérez

Ing. Daniel Pardo Echevarría

**La Habana, 2024**

**Resumen**

El auge de los *bots* en el comercio electrónico ha convertido este sector en un campo de batalla constante entre los actores legítimos y los malintencionados. Los *bots*, utilizados tanto para tareas legítimas como para actividades maliciosas, representan un desafío significativo para los sitios de comercio electrónico, ya que pueden realizar acciones de diversa índole, llegando a ser perjudiciales, sobre todo partiendo del hecho de que no son los usuarios de interés de los sitios de comercio que buscan trabajar con personas. Este problema ha llevado a la necesidad de desarrollar soluciones efectivas para detectar y mitigar la acción de los *bots*, utilizando técnicas de inteligencia artificial (IA) que puedan adaptarse y aprender de patrones de comportamiento malicioso.

La solución propuesta es un módulo de detección de bots basado en IA, cuya arquitectura es crucial para garantizar que la detección de bots no interfiera con la experiencia del usuario ni con las operaciones comerciales normales. La arquitectura debe ser diseñada para ser transparente y eficiente, permitiendo que el módulo opere en segundo plano sin afectar la interacción normal de los usuarios con el sitio. Además, debe ser flexible y escalable, permitiendo que el módulo se adapte a las cambiantes tácticas de los bots y a los crecientes volúmenes de tráfico en línea.

**Palabras clave**: Inteligencia Artificial, *bots*, comercio electrónico, detección de *bots* , arquitectura

**Abstract**

The rise of bots in e-commerce has turned this sector into a constant battleground between legitimate and malicious actors. Bots, used for both, legitimate tasks and malicious activities, represent a significant challenge for e-commerce sites, as they can perform a variety of actions and can become disruptive, especially given the fact that they are not the intended users of commerce sites seeking to work with people. This problem has led to the need to develop effective solutions to detect and mitigate the action of bots, using artificial intelligence (AI) techniques that can adapt and learn from malicious behaviour patterns.

The proposed solution is an AI-based bot detection module, whose architecture is crucial to ensure that bot detection does not interfere with user experience and normal business operations. The architecture must be designed to be transparent and efficient, allowing the module to operate in the background without affecting users' normal interaction with the site. It should also be flexible and scalable, allowing the module to adapt to changing bot tactics and increasing volumes of online traffic.

**Keywords:** Artificial Intelligence, bots, e-commerce, bot detection, architecture

**Índice de contenido**

[Introducción 1](#_Toc183242150)

[Desarrollo 4](#_Toc183242151)

[Posibles lenguajes: 4](#_Toc183242152)

[Selección: 8](#_Toc183242153)

[Tratamiento de los problemas frecuentes mediante la tecnología seleccionada 10](#_Toc183242154)

[Control de versiones 12](#_Toc183242155)

[IDEs 13](#_Toc183242156)

[Selección de IDE 14](#_Toc183242157)

[Framework y bibliotecas 14](#_Toc183242158)

[Otros elementos arquitectónicos 15](#_Toc183242159)

[Microservicios 15](#_Toc183242160)

[Event-Driven 16](#_Toc183242161)

[Selección final 16](#_Toc183242162)

[Estilos 17](#_Toc183242163)

[Llamada y retorno 17](#_Toc183242164)

[Centrado en datos 21](#_Toc183242165)

[Flujo de datos 22](#_Toc183242166)

[Principios de diseño 23](#_Toc183242167)

[Patrones de diseño 26](#_Toc183242168)

[Template method 26](#_Toc183242169)

[Strategy 27](#_Toc183242170)

[Fachada 28](#_Toc183242171)

[Mecanismos de Diseño 29](#_Toc183242172)

[Preprocesamiento de datos 30](#_Toc183242173)

[Caracterización de instancias por Meta-Componente 32](#_Toc183242174)

[Conclusiones 35](#_Toc183242175)

[Referencias bibliográficas 36](#_Toc183242176)

**Índice de figuras**

[Figura 1 Diagrama de procesos de Eriksson Penker[1] 3](#_Toc162356384)

[Figura 2 Enfoque de reutilización 18](#_Toc162356385)

[Figura 3 Enfoque de responsabilidad 20](#_Toc162356386)

[Figura 4 Representación del patrón Blackboard 22](#_Toc162356387)

[Figura 5 Representación de Filtros y Tuberías 23](#_Toc162356388)

[Figura 6 Patrón Template method 27](#_Toc162356389)

[Figura 7 Patrón estrategia 28](file:///C:\Users\Adrian%20y%20Karina\Desktop\4to\Entregable%20Final%20de%20Diseño%20de%20Software.docx#_Toc162356390)

[Figura 8 Patrón Facade 29](#_Toc162356391)

[Figura 9 Diagrama de despliegue 30](#_Toc162356392)

[Figura 10 Diagrama de clases de la solución 31](#_Toc162356393)

[Figura 11 Diagrama de clases para el preprocesamiento 32](#_Toc162356394)

[Figura 12 Diagrama de secuencia del preprocesamiento 33](#_Toc162356395)

[Figura 13 Diagrama de clases de la caracterización de instancias 34](#_Toc162356396)

[Figura 14 Diagrama de secuencia de la caracterización de instancias 35](#_Toc162356397)

# Introducción

El comercio electrónico ha experimentado un crecimiento exponencial en los últimos años, lo que ha llevado a un aumento en el uso de *bots* para realizar actividades maliciosas. Esto ha generado una necesidad imperante de desarrollar soluciones efectivas para detectar y mitigar el uso de *bots*, utilizando técnicas de IA que puedan adaptarse y aprender de patrones de comportamiento malicioso.

La IA, con sus capacidades de aprendizaje automático y procesamiento de lenguaje natural, ofrece una herramienta poderosa para la detección de anomalías. Estas tecnologías no solo son capaces de identificar patrones de comportamiento sospechosos sino que también pueden ser entrenadas para reconocer y adaptarse a nuevas formas de ataque de *bots*, lo que la convierte en una solución dinámica y efectiva para este problema.

Basado en este principio en la CUJAE se desarrollo un módulo de detección de *bots* malignos basado en aprendizaje automático, específicamente a través de la implementación del algoritmo de aprendizaje supervisado Proactive Forest. Además, se diseñó un componente de meta-aprendizaje, que sirve de apoyo al proceso con la caracterización de datos de *bots* y humanos. Este módulo se pretende desplegarlo en futuros trabajos que no incluyen a los integrantes actuales. El módulo de detección de bots malignos comprende varias etapas, dentro de las cuales existen varios procesos. Cada uno se centra en funcionalidades particulares de su desarrollo. La Figura 1 muestra el diagrama de Eriksson Penker, que describe las etapas de la solución propuesta y los procesos asociados.

Etapa 1- Detección de potenciales usuarios *bots* malignos: En esta etapa se extraen las características de aquellos usuarios que acceden a un sitio web. Para detectar si en un conjunto determinado usuarios hay potenciales amenazas o no, a partir de un proceso de meta-aprendizaje. Los usuarios considerados como potenciales amenazas son analizados en la próxima etapa. Comprende los procesos de captura de características de usuario y clasificación por meta-componente.

Etapa 2- Detección de instancias de usuarios *bots* malignos: Esta etapa se centra en clasificar individualmente cada instancia de usuario en *bot* maligno o humano. A partir de un conjunto de datos de usuarios potenciales *bots* malignos recibido de la etapa 1. Empleando para ello un modelo de clasificación, entrenado sobre una base de datos de ataques de *bots* malignos. Esta etapa comprende los procesos de entrenar algoritmo *Proactive Forest* y clasificar datos de potenciales *bots*.

Etapa 3 - Caracterización de datos de usuarios *bots* malignos y humanos: En esta etapa se caracterizan las instancias de usuarios que interactúan con la plataforma web, estando dentro de la misma. A partir de todos los datos ya clasificados en la anterior etapa, empleando además diversas medidas de complejidad. Almacenando los resultados de esta caracterización en una base de hechos. Esta etapa incluye el proceso de caracterizar usuarios por meta-componente.

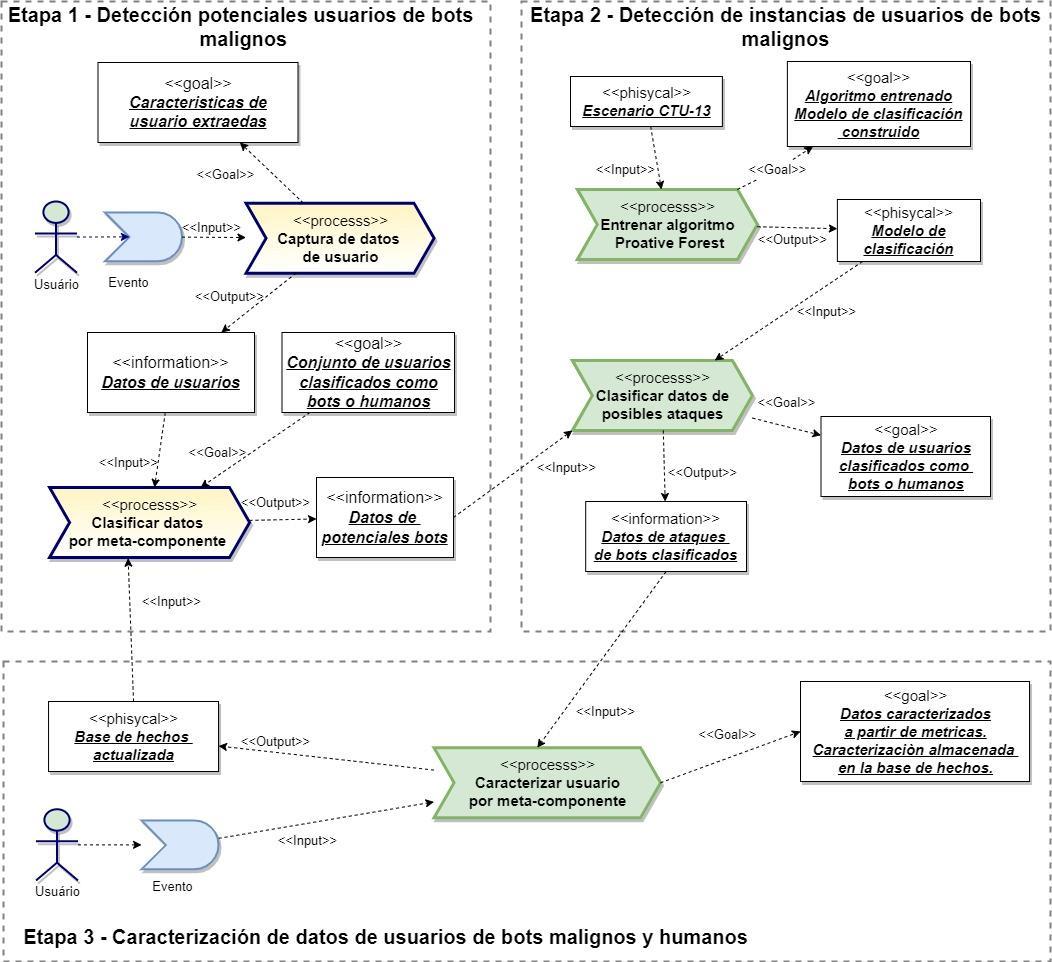


Figura 1 Diagrama de procesos de Eriksson Penker[1]

# Desarrollo

El proyecto en cuestión se centra en la aplicación de técnicas de inteligencia artificial (IA) y cálculo de métricas para la detección de anomalías en un entorno de correo electrónico. La detección de anomalías es vital para identificar comportamientos anómalos o actividades sospechosas que podrían indicar la presencia de *bots* o acciones maliciosas. Para lograr esto, es esencial seleccionar un lenguaje de programación y herramientas adecuados que puedan soportar los complejos algoritmos de aprendizaje automático y aprendizaje profundo, así como facilitar el manejo de grandes volúmenes de datos y la construcción de modelos de IA eficientes.

La elección del lenguaje de programación debe tener en cuenta factores como la facilidad de uso, la disponibilidad de bibliotecas y *frameworks* especializados, la comunidad de desarrollo y el soporte para la integración con otras tecnologías y plataformas utilizadas en el proyecto. Además, es importante considerar las capacidades de procesamiento y almacenamiento de datos, así como la escalabilidad y la capacidad de implementar y mantener el sistema en un entorno dinámico y en constante evolución.

## Posibles lenguajes:

1. Python

Python emerge como una opción potente para el desarrollo de proyectos de detección de *bots* gracias a su simplicidad y legibilidad, así como a su rica ecosistema de bibliotecas y comunidad activa. Su sintaxis es clara y comprensible, lo que facilita el aprendizaje y la depuración del código. Además, cuenta con una amplia gama de bibliotecas de IA y ML como TensorFlow, PyTorch y Scikit-learn, que son fundamentales para la creación de modelos de detección avanzados.

* **Amplia variedad de bibliotecas de IA/ML**: Python ofrece librerías especializadas que aceleran el desarrollo y reducen la necesidad de reinventar la rueda. Herramientas como NumPy y Pandas facilitan el manejo de datos, mientras que Scikit-Learn proporciona algoritmos de aprendizaje automático y TensorFlow y PyTorch son excelentes para tareas de aprendizaje profundo[2-4].
* **Sintaxis clara y legible**: La facilidad de uso de Python se traduce en un código limpio y comprensible, lo cual es beneficioso tanto para principiantes como para programadores experimentados. Esto permite transformar rápidamente algoritmos complejos en scripts comprensibles[2-4].
* **Comunidad activa**: La comunidad Python es un recurso valioso, ya que es probable que alguien haya enfrentado desafíos similares y compartido sus soluciones. Además, la documentación completa de Python es una herramienta útil para entender funciones, clases y métodos[2-4].

Estas ventajas hacen de Python una elección sólida para el desarrollo de un módulo de detección de bots, donde la capacidad de prototipado rápido y la integración con otros sistemas son particularmente valiosas. Además, su capacidad para trabajar con otras tecnologías y su compatibilidad con varios sistemas operativos garantizan una implementación eficiente y flexible.

1. Java

Java se presenta como una alternativa robusta y escalable para el desarrollo de un módulo de detección de bots, con ventajas significativas que lo hacen atractivo para aplicaciones empresariales y proyectos de gran envergadura.

* **Robustez y escalabilidad**: Java es conocido por su estabilidad y capacidad para manejar grandes cantidades de datos y usuarios simultáneos, lo que lo hace ideal para sistemas críticos y con alta demanda, como un módulo de detección de *bots* para un sitio de correo electrónico[5, 6].
* **Bibliotecas para IA/ML**: Existen bibliotecas como Weka y Deeplearning4j que pueden ser muy útiles para la detección de anomalías. Weka, por ejemplo, es una colección de software gratuito para el descubrimiento de conocimientos y minería de datos, mientras que Deeplearning4j proporciona una implementación de aprendizaje profundo en Java [5, 6].
* **Despliegue en la nube**: Java puede ser desplegado fácilmente en la nube, como en Azure, lo que brinda acceso desde cualquier lugar con conexión a Internet y permite manejar un alto volumen de usuarios sin comprometer el rendimiento [5, 6].
* **Entorno de desarrollo y herramientas**: Java cuenta con un ambiente de desarrollo maduro y herramientas como Maven o Gradle para gestionar dependencias y construir proyectos, facilitando la configuración y el mantenimiento de un proyecto de detección de *bots*[5, 6].
* **Integración con servicios de inteligencia artificial**: Al igual que con Python, Java puede integrarse con servicios de inteligencia artificial y procesamiento de lenguaje natural para mejorar la interacción con los usuarios y la eficacia de la detección de *bots* [5, 6].

Estas características hacen de Java una opción sólida para el desarrollo de sistemas de detección de *bots*, especialmente en entornos empresariales donde la fiabilidad, escalabilidad y soporte para IA son prioritarios.

1. R

R es una opción fuerte para proyectos que requieren análisis estadístico y visualización de datos, como la detección de anomalías en un módulo de correo electrónico. Sus paquetes especializados, como caret y *RandomForest*, son altamente valorados por su eficiencia y versatilidad en la identificación de patrones en los datos[7, 8].

* **Análisis estadístico y visualización**: R es bien conocido por sus capacidades de análisis estadístico y su rica biblioteca de gráficos, lo que lo convierte en una herramienta poderosa para la interpretación de datos y la presentación de resultados [7, 8].
* **Paquetes para detección de anomalías**: Los paquetes como anomalize y AnomalyDetection están diseñados específicamente para la detección de anomalías y proporcionan funcionalidades robustas y fáciles de usar para este propósito. Estos paquetes pueden integrarse con el flujo de trabajo del tidyverse, lo que facilita el manejo de datos y la generación de informes [7, 8].
* **Ejemplos prácticos**: La capacidad de R para realizar análisis de series temporales y detección de anomalías está respaldada por ejemplos prácticos y tutoriales, como el uso de anomalize para ajustar parámetros y visualizar anomalías en conjuntos de datos de tiempo[7, 8].

Aunque R es excelente para el análisis estadístico y la visualización, su enfoque puede ser menos adecuado para la implementación de modelos de IA complejos, especialmente si se requiere una integración estrecha con otros sistemas o un desempeño en tiempo real. Sin embargo, para el análisis de datos y la detección de anomalías, R es una elección sólida debido a su eficiencia y a la calidad de sus paquetes dedicados[7, 8].

1. Scala

Scala emerge como una elección viable para el proyecto de detección de bots, especialmente por su combinación de características de concisión y eficiencia, y por las bibliotecas especializadas para IA y ML que posee.

* **Concisión y eficiencia**: Scala es conocido por su sintaxis concisa y su capacidad para escribir código legible y expresivo. Combina la funcionalidad de programación orientada a objetos con la programación funcional, lo que permite un desarrollo más eficiente y un mejor manejo de la concurrencia y el paralelismo, lo cual es crucial para el procesamiento de grandes volúmenes de datos, como es común en la detección de anomalías[9].
* **Bibliotecas para IA/ML**: Scala cuenta con bibliotecas como Breeze y Deeplearning.scala que facilitan la implementación de algoritmos de IA y ML. Breeze es una biblioteca matemática de alto nivel que proporciona una interfaz rica para álgebra lineal, optimización y estadística, mientras que Deeplearning.scala es un marco de trabajo para el aprendizaje profundo en Scala[9].
* **Interoperabilidad con Java**: Dado que Scala se ejecuta en la JVM, puede integrarse fácilmente con bases de código existentes en Java. Esto significa que se puede aprovechar el vasto ecosistema de Java para el acceso a bibliotecas y herramientas adicionales que pueden ser útiles en el proyecto[9].
* **Escalabilidad y distribución**: Scala fue diseñado con escalabilidad en mente y es ideal para sistemas distribuidos. El marco Akka, construido en Scala, es una herramienta de alto rendimiento para la construcción de aplicaciones orientadas a mensajes, resistentes y concurrentes, que simplifica considerablemente el diseño de sistemas distribuidos[9].

Por estas razones, Scala puede ser una opción interesante si se busca un equilibrio entre la concisión y el rendimiento, con la capacidad de manejar proyectos a gran escala y proporcionar un código más robusto debido a su fuerte sistema de tipos estáticos.

### Selección:

A partir de la información anterior se estableció una comparación plasmada en la Tabla 1 donde se ha seleccionado Python como lenguaje de programación para la implementación del módulo de detección de *bots* debido a su simplicidad, flexibilidad y amplia gama de bibliotecas y marcos de trabajo especialmente diseñados para el aprendizaje automático y la inteligencia artificial. Python es conocido por su sintaxis clara y fácil de entender, lo que facilita tanto el desarrollo inicial como la colaboración entre desarrolladores. Además, Python es una herramienta altamente versátil y puede ejecutarse en múltiples plataformas sin necesidad de modificaciones significativas, lo que reduce la barrera de entrada para los desarrolladores.

Tabla 1 Comparación de los lenguages

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lenguage | Python | Java | R | Scala |
| Interoperabilidad | Alta | Media | Baja | Alta |
| Bibliotecas para IA y análisis estadístico | Abundantes | Limitadas | Especializadas | Variedad limitada |
| Sintaxis | Sencilla | Detallada | Específica para análisis estadísticos | Concisa y funcional |
| Comunidad | Extensa | Grande | Activa en análisis de datos | En crecimiento |
| Escalabilidad | Media | Alta | Media | Alta |

Sin embargo, Python presenta algunas desventajas cuando se trata de operaciones de cálculo intensivo. La velocidad de ejecución de Python puede ser más lenta en comparación con otros lenguajes como C++ o Java, especialmente en tareas que requieren procesamiento paralelo o gráficos computacionales avanzados. Esto puede ser problemático en el trabajo con grandes volúmenes de datos en tiempo real.

Alternativas y soluciones:

1. Uso de lenguajes de bajo nivel: En el caso de un rendimiento crítico, se podría considerar la implementación de módulos específicos en lenguajes de bajo nivel como C++ o Rust para realizar operaciones intensivas de cálculo.

2. Uso de extensiones de Python: La integración de módulos escritos en lenguajes de bajo nivel a través de extensiones de Python (como ctypes o Cython) podría ofrecer un equilibrio entre rendimiento y facilidad de desarrollo.

## Tratamiento de los problemas frecuentes mediante la tecnología seleccionada

En el actual entorno digital, la detección y mitigación de problemas frecuentes son desafíos cruciales para garantizar el buen funcionamiento de sistemas basados en inteligencia artificial. A continuación se expone el tratamiento de los mismos a través de los medios brindados por Python:

* Calidad de los Datos: Se pueden utilizar bibliotecas como Pandas y NumPy para realizar tareas de limpieza de datos, manejo de valores faltantes, transformación de datos y más. Estas herramientas ayudan a garantizar la calidad de los datos antes de entrenar modelos de IA, lo que es crucial para la precisión del sistema de detección de *bots[10]*.
* Desbalance de Clases: Python ofrece bibliotecas como Scikit-learn, que incluyen métodos de remuestreo como SMOTE (Synthetic Minority Over-sampling Technique). Estos métodos pueden equilibrar las clases de datos desbalanceadas, permitiendo que el modelo de detección de *bots* aprenda de manera más efectiva sin verse afectado por la falta de representación de ciertas clases[11].
* Sobreajuste o Subajuste: Con Python, es posible implementar técnicas de validación cruzada que dividen los datos en conjuntos de entrenamiento y prueba múltiples, lo que ayuda a evaluar la capacidad de generalización del modelo. Esto previene problemas de sobreajuste o subajuste al ajustar el modelo a los datos de forma adecuada[12].
* Selección de Características Inadecuadas: Python ofrece herramientas como Scikit-learn que permiten la selección automática de características mediante algoritmos especializados. Estos algoritmos identifican las características más relevantes para la detección de *bots*, mejorando la eficacia del modelo al utilizar solo las variables significativas[4].
* Problemas de Escalabilidad: Python es compatible con bibliotecas como TensorFlow y PyTorch, que son ideales para escalar modelos de IA a grandes volúmenes de datos y operaciones intensivas de manera eficiente. Estos frameworks facilitan el manejo de grandes cantidades de datos, asegurando que el módulo de detección de *bots* sea escalable y efectivo[13].
* Interpretabilidad del Modelo: Con herramientas de visualización como Matplotlib y Seaborn en Python, se pueden crear representaciones visuales de los modelos de detección de *bots*. Estas visualizaciones ayudan a comprender cómo opera el modelo y a interpretar sus decisiones, lo que es fundamental para la confianza en el sistema[14].
* Reentrenamiento y Actualización: Python permite automatizar el proceso de reentrenamiento y actualización del modelo en función de nuevos datos. Esta automatización asegura que el sistema de detección de *bots* se mantenga actualizado y pueda adaptarse a cambios en el comportamiento de los *bots* de manera rápida y eficiente.
* Rendimiento y Tiempo de Respuesta: Python ofrece opciones para optimizar el código y paralelizar tareas, lo que mejora el rendimiento del sistema de detección de *bots* y reduce significativamente el tiempo de respuesta del mismo[13].
* Integración con el Sistema Existente: Python es compatible con una amplia gama de tecnologías y frameworks, lo que simplifica la integración del módulo de detección de *bots* con sistemas de comercio electrónico existentes. La compatibilidad con diferentes plataformas hace que la integración sea más fluida y menos propensa a errores[15].
* Seguridad y Privacidad: Python ofrece un conjunto de herramientas especializadas en manejo de datos sensibles, seguridad y cifrado que pueden implementarse para garantizar que los datos utilizados en el proceso de detección de *bots* estén protegidos y se respete la privacidad de los usuarios[16].

## Control de versiones

Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo. A medida que los entornos de desarrollo se aceleran, los sistemas de control de versiones ayudan a los equipos de software a trabajar de forma más rápida e inteligente. Para el control de versiones del proyecto se seleccionó Git con GitHub (o GitLab)[17].

Ventajas:

* Git: Esta tecnología ofrece un control de versiones distribuido que permite un seguimiento preciso de todos los cambios en el código, facilitando la colaboración y la integración continua.
* GitHub (o GitLab): Estas plataformas ofrecen servicios sólidos para alojar repositorios, gestionar problemas, realizar solicitudes de extracción, y facilitar la integración continua
* Control de versiones eficiente: Git proporciona un historial completo de cambios, lo que es crucial para la trazabilidad y la gestión de versiones en un entorno de desarrollo activo.
* Colaboración y gestión de proyectos: Tanto GitHub como GitLab ofrecen características sólidas para la colaboración y la gestión de proyectos que son esenciales para equipos de desarrollo distribuidos y métodos ágiles.

Aunque esta selección aporta ventajas a medida que el proyecto crece, la gestión de múltiples ramas, problemas y pruebas puede requerir una configuración y gestión cuidadosa para garantizar la eficiencia y la organización adecuada

Alternativas:

* GitFlow: Adoptar un flujo de trabajo basado en GitFlow podría ayudar a manejar las complejidades de un sistema de control de versiones en constante evolución[18].
* Bitbucket: Si se identifican limitaciones en GitHub o GitLab, Bitbucket ofrece una alternativa que también admite control de versiones distribuido y una gama de características de colaboración[19].
* Implementación cuidadosa de ramas: El uso cuidadoso y deliberado de ramas en Git puede ayudar a mitigar posibles problemas de escalabilidad y complejidad en la gestión de versiones[17].

## IDEs

Para el desarrollo de un módulo de detección de *bots* basado en técnicas de inteligencia artificial utilizando Python, hay varias opciones de IDEs disponibles que podrían ser útiles, como, por ejemplo: PyCharm es una opción muy robusta y completa, gracias a su integración con bibliotecas de ML/AI y a sus funcionalidades de desarrollo avanzadas. Permite el uso de la *shell* de Python durante la depuración, puede recordar los paquetes de Python faltantes y permite la instalación de Python, sus paquetes y Git todo dentro del IDE. Además, ofrece un soporte de refactorización sobresaliente, permitiendo cambiar rápidamente el nombre de variables, clases y métodos en todo un archivo o proyecto[20].

Visual Studio Code (VSCode) es otro IDE popular que es altamente personalizable y cuenta con una gran cantidad de extensiones que pueden mejorar la experiencia de desarrollo para Python y proyectos de ML/AI. VSCode es conocido por su buen rendimiento y velocidad, lo cual puede ser beneficioso en proyectos de ML/AI que pueden requerir un manejo intensivo de recursos[21].

Otras opciones de IDEs incluyen[22]:

* IDLE: Es el editor de código predeterminado que acompaña a Python. Es adecuado para desarrolladores de nivel principiante y se puede utilizar en Mac OS, Windows y Linux. Es gratuito y tiene características básicas.
* Spyder: Es un IDE ligero y está diseñado para la ciencia de datos. Ofrece una interfaz de usuario intuitiva y una integración con Jupyter Notebooks, lo que puede ser útil para la experimentación y visualización de datos.
* Jupyter Notebook: Es una aplicación web que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Es ideal para la ciencia de datos y el aprendizaje automático.
* Rodeo: Es un IDE desarrollado para tareas relacionadas con la ciencia de datos, como la toma de datos e información de diferentes recursos y la visualización de datos. Es compatible con funcionalidades multiplataforma.
* PyDev: Es un potente intérprete de Python distribuido como un plugin de terceros para el IDE Eclipse. Es flexible y es uno de los IDEs de código abierto preferidos por los desarrolladores.
* Thonny: Es un IDE ideal para enseñar y aprender programación en Python. Es fácil de usar y es adecuado para principiantes.
* Atom: Es un IDE de código abierto desarrollado por Github. Atom es altamente personalizable y ofrece múltiples plugins, lo que puede ser beneficioso para personalizar el IDE y mejorar el flujo de trabajo

### Selección de IDE

Aunque PyCharm es una elección idónea debido a su potencia y características avanzadas, debido a la disponibilidad de hardware de los desarrolladores, se opta por elegir Visual Studio Code en busca de una experiencia de desarrollo más ligera manteniendo los estándares de calidad necesarios.

## Framework y bibliotecas

Para desarrollar un proyecto es importante considerar varios factores, como la facilidad de aprendizaje y uso, la amplia variedad de bibliotecas y *frameworks* disponibles, la comunidad activa, la integración con otras tecnologías y la escalabilidad y rendimiento.

Algunos de los *frameworks* y bibliotecas más destacados en Python para la inteligencia artificial incluyen[2, 15, 23, 24]:

* TensorFlow: Desarrollado por Google, es una biblioteca de código abierto para el aprendizaje automático y las redes neuronales profundas. TensorFlow ofrece un entorno integral para la investigación y la producción, permitiendo la creación y despliegue eficiente de modelos de IA.
* PyTorch: Respaldado por Facebook, PyTorch es conocido por su naturaleza dinámica y amigable, lo que facilita la experimentación interactiva con modelos. Es una opción popular entre investigadores y desarrolladores que valoran la flexibilidad y la agilidad en el desarrollo de modelos de IA.
* Scikit-learn: Es una biblioteca de aprendizaje automático en Python que proporciona una selección de algoritmos de clasificación, regresión y clustering. Es fácil de usar y tiene una documentación extensa, lo que lo hace ideal para tareas de machine learning tradicionales.
* Keras: Es una API de alto nivel para construir y entrenar modelos de redes neuronales. Keras se puede ejecutar sobre TensorFlow, Microsoft Cognitive Toolkit o Theano, lo que permite una gran flexibilidad.

Para la detección de *bots*, se puede considerar el uso de TensorFlow o PyTorch, especialmente si implica el aprendizaje profundo; pero como el módulo está basado en un enfoque más tradicional de aprendizaje automático dirigido a la clasificación basada en características Scikit-learn sería el ideal.

## Otros elementos arquitectónicos

Teniendo en cuenta la funcionalidad del módulo se realizó un análisis y una selección de arquitectura basada en las necesidades de rendimiento, escalabilidad y flexibilidad para la detección en tiempo real de comportamientos automatizados no deseados en un sitio de comercio electrónico. Inicialmente, se consideró una arquitectura de microservicios, que permite la descomposición de la aplicación en servicios independientes que se comunican entre sí. Sin embargo, esta visión evolucionó hacia una arquitectura event-driven, que se centra en la generación y manejo de eventos para coordinar acciones entre los componentes del sistema.

### Microservicios

Los microservicios son una arquitectura que descompone una aplicación en servicios independientes, cada uno con su propia lógica de negocio y responsabilidades claras. Esta descomposición permite un desarrollo, implementación y escalado independiente de cada componente del sistema. Cada microservicio puede ser desarrollado por diferentes equipos, utilizando diferentes tecnologías y lenguajes de programación, lo que favorece la innovación y la adaptabilidad. Además, al ser independientes, los microservicios pueden ser escalados horizontalmente según las necesidades, lo que resulta en un sistema más eficiente y menos propenso a fallos en su totalidad debido a la falla de un único servicio[25].

### Event-Driven

La arquitectura event-driven se basa en la generación y manejo de eventos para coordinar las acciones entre los componentes del sistema. En lugar de que los componentes del sistema soliciten activamente información a otros componentes (como en un enfoque sincrónico), la arquitectura event-driven opera de manera reactiva, respondiendo a eventos que ocurren en el sistema. Esto permite una alta cohesión y un bajo acoplamiento entre los componentes, ya que cada uno solo necesita saber cómo responder a los eventos que le interesan, en lugar de conocer los detalles de cómo los otros componentes realizan sus tareas. La arquitectura event-driven es especialmente adecuada para sistemas con requisitos de escalabilidad y flexibilidad, ya que permite agregar, modificar o eliminar componentes sin afectar el funcionamiento del sistema en su totalidad[26].

### Selección final

A pesar de que el módulo actual presenta una arquitectura monolítica enfocada en la experimentación, el análisis realizado demuestra que hay opciones más favorables para su rendimiento una vez desplegado. La combinación de microservicios y arquitectura event-driven en un módulo de detección de *bots* puede ofrecer un sistema robusto y flexible. Cada etapa del proceso de detección de *bots* (como la recopilación de datos, el análisis de patrones y la toma de decisiones) puede ser implementada como un microservicio independiente. Estos microservicios pueden interactuar entre sí mediante la generación y manejo de eventos, lo que permite una coordinación eficiente y adaptable de las acciones entre las etapas del proceso. Esta arquitectura permite una fácil escalabilidad y adaptabilidad, ya que se pueden añadir nuevos servicios o modificar los existentes sin interrumpir el sistema en su totalidad[27, 28].

## Estilos

Un estilo arquitectónico de diseño de software es una forma de organizar y estructurar un sistema de software que define cómo se deben dividir las responsabilidades y cómo deben interactuar las diferentes partes del sistema. Estos estilos proporcionan un marco conceptual que ayuda a los equipos de desarrollo a tomar decisiones sobre cómo se construirán y se mantendrán los sistemas de software. En el proyecto se evidencian los estilos llamada y retorno, centrado en datos y flujo de datos mediante el uso de patrones arquitectónicos asociados[29].

### Llamada y retorno

Este estilo arquitectónico se centra en la organización de un sistema de software en torno a la comunicación entre subrutinas o funciones. Las subrutinas se llaman (o invocan) desde otras partes del programa para realizar ciertas operaciones y luego regresan un resultado o control a la subrutina llamadora[29].

#### Patrón N-capas

El patrón n-capas consiste en estructurar el sistema en grupos de componentes que formen capas unas encima de otras.

Enfoque por reutilización

Es una manera de representar la aplicación cuando nuestro interés es ver la arquitectura como un todo y el lente por el que estamos mirando es la reutilización**.** Como se puede ver en la Figura 2 la aplicación se muestra en 4 capas, en las cuales se logra ver la aplicación desplegada desde sus componentes más específicos hasta los más reutilizables.

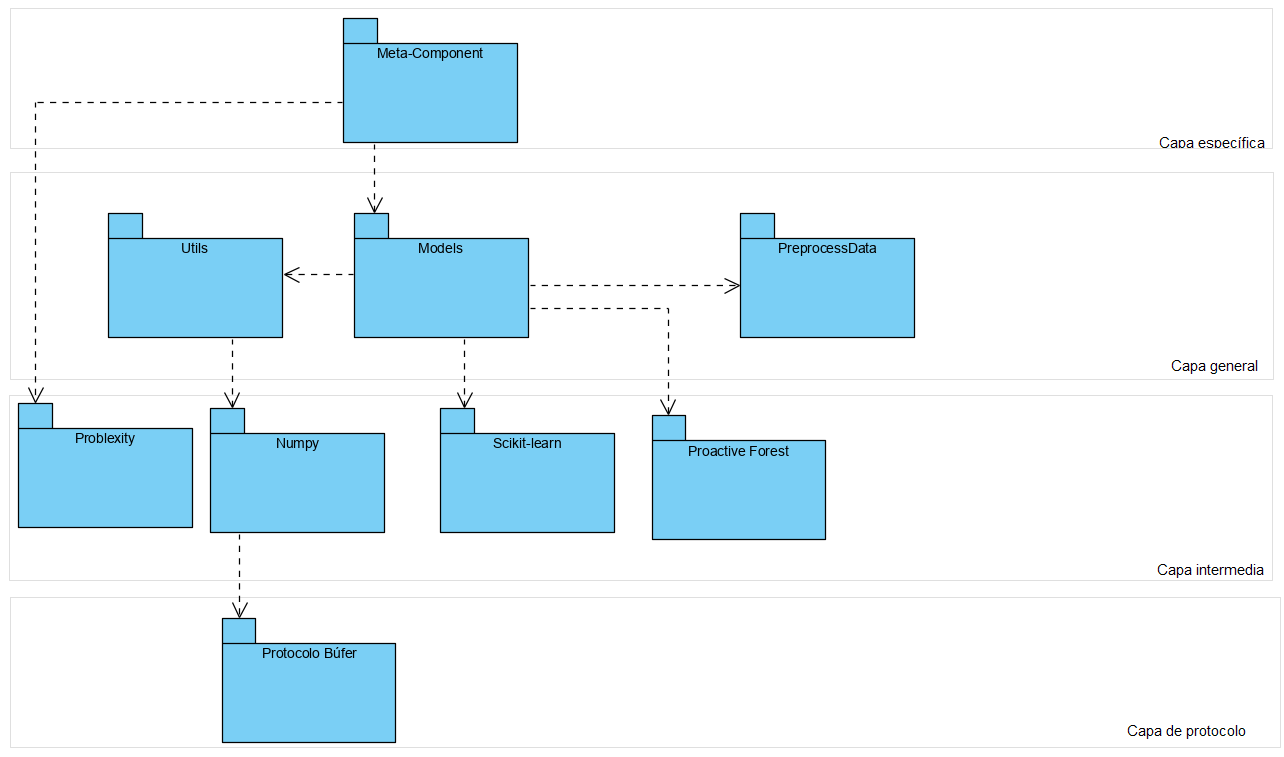


Figura 2 Enfoque de reutilización

1- Capa específica:

• Meta-component: Elemento específico del sistema. Contiene todos los métodos asociados a la implementación del meta-componente de caracterización de usuarios. Sus métodos no son reutilizados por ningún otro módulo.

2- Capa general:

• Utils: Elemento reutilizable del sistema. Contiene todos los métodos para el trabajo con datos y otras operaciones que se requieran.

• Models: Elemento reutilizable del sistema, que se encuentra en desarrollo. Contiene todos los métodos y clases asociados a la implementación de algoritmos de aprendizaje automático. Paquete donde se implementa el algoritmo Proactive Forest, como propuesta de solución del presente trabajo.

• Preprocess-Data: Elemento reutilizable del sistema, ya implementado y desarrollado en[30]. Contiene clases y métodos para cargar y preprocesar la base de datos de ataque de bots.

3- Capa intermedia:

• NumPy: Elemento reutilizable, obtenido como biblioteca del lenguaje de programación. Contiene métodos para el trabajo con arreglos multidimensionales y la ejecución de cálculos complejos[10] .

• Scikit-learn: Elemento reutilizable, obtenido como biblioteca del lenguaje de programación. Contiene métodos y clases para el trabajo con algoritmos de aprendizaje automático y para obtener métricas de precisión, error, entre otras, de los modelos generados[4].

• Problexity: Elemento reutilizables, obtenido como biblioteca del lenguaje de programación. Desarrollado en [31], contiene las medidas de complejidad de datos.

• Proactive Forest: Elemento reutilizable, obtenido de un tercero. Contiene métodos y clases que comprenden la ejecución del algoritmo Proactive Forest. Tanto para la construcción del modelo clasificador, como para predecir el valor de nuevas instancias y evaluar su rendimiento.

4- Capa de software:

• Protocolo buffer: Protocolo del lenguaje de programación. Permite un mayor control en el manejo de estructuras array, utilizadas en bibliotecas de análisis numérico.

Enfoque por responsabilidad

El enfoque por responsabilidad en la arquitectura de software n-capas, se refiere a la organización del código y sus funcionalidades en diferentes capas, cada una con una responsabilidad específica. Cada capa se encarga de una parte particular del sistema y se comunica con las capas adyacentes a través de interfaces bien definidas, lo que permite una separación clara de las responsabilidades y una modularidad adecuada como se ve en la Figura 3.

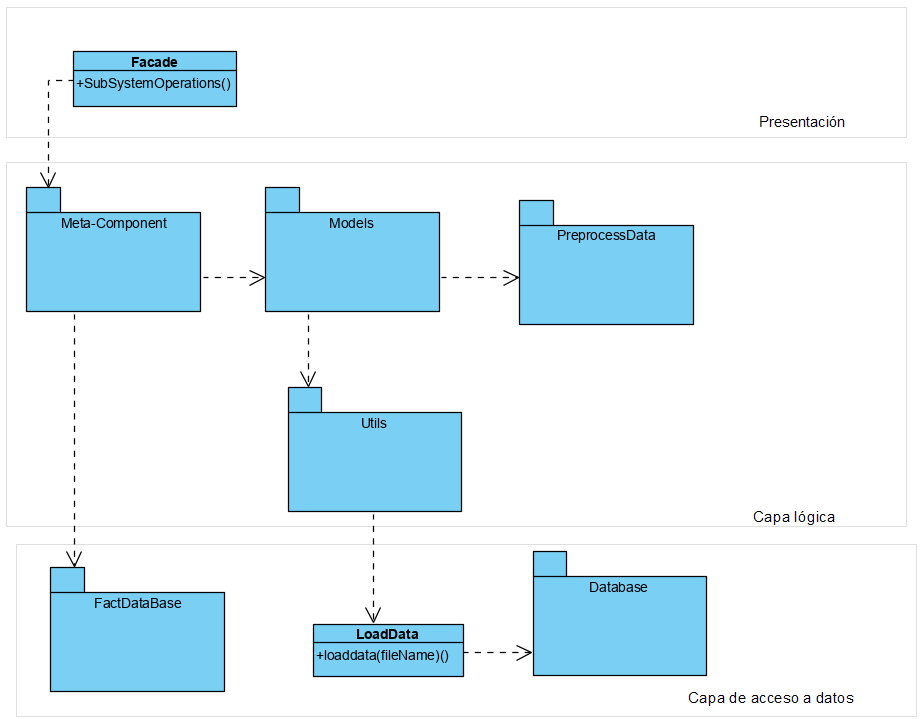


Figura 3 Enfoque de responsabilidad

El módulo no presenta una interface gráfica puesto que está diseñado para funcionar en segundo plano de forma automatizada, en otras palabras, no requiere interacción humana para su funcionamiento y la comunicación con el sitio se establece a partir de una interface fachada. El **componente de meta aprendizaje (Meta-Component)** es responsable de clasificar un grupo de usuarios como sospechosos o no, utilizando una serie de métricas calculadas que se comparan con los valores existentes en la **base de hechos (FactDatabase)** y en función de dicha clasificación se utilizan los modelos de inteligencia artificial disponibles en el **paquete Models para una clasificación más profunda{Géron, 2022 #32}**. Estos modelos son producto de un proceso de minería de datos realizado sobre datos preprocesados cargados por la clase **l**oadData de la biblioteca Utils y tienen la capacidad de clasificar si un usuario es o no un bot en función de los datos con los que fue entrenado. El preprocesamiento de estos datos se lleva a cabo mediante clases específicas ubicadas en el **paquete PreprocessData**.

### Centrado en datos

Este estilo arquitectónico se enfoca en la estructuración y manipulación de datos dentro de un sistema de software.Prioriza el manejo de los datos y la interacción con ellos como el núcleo del sistema[29].

#### Patrón BlackBoard

El patrón Blackboard permite que múltiples procesos trabajen de manera más estrecha en hilos separados, sondeando y reaccionando cuando sea necesario. Esto es beneficioso cuando la base de datos no es centralizada y hay varias fuentes de conocimiento que necesitan compartir información y tomar decisiones basadas en esa información.Como se puede ver en la Figura 4 el módulo trabaja con diferentes fuentes de conocimiento a las que se accede mediante el componente de meta aprendizaje(Base de hecho para las métricas) y la clase loadData del paquete Util(base de datos de los ataques para el entrenamiento de los modelos de inteligencia artificial).

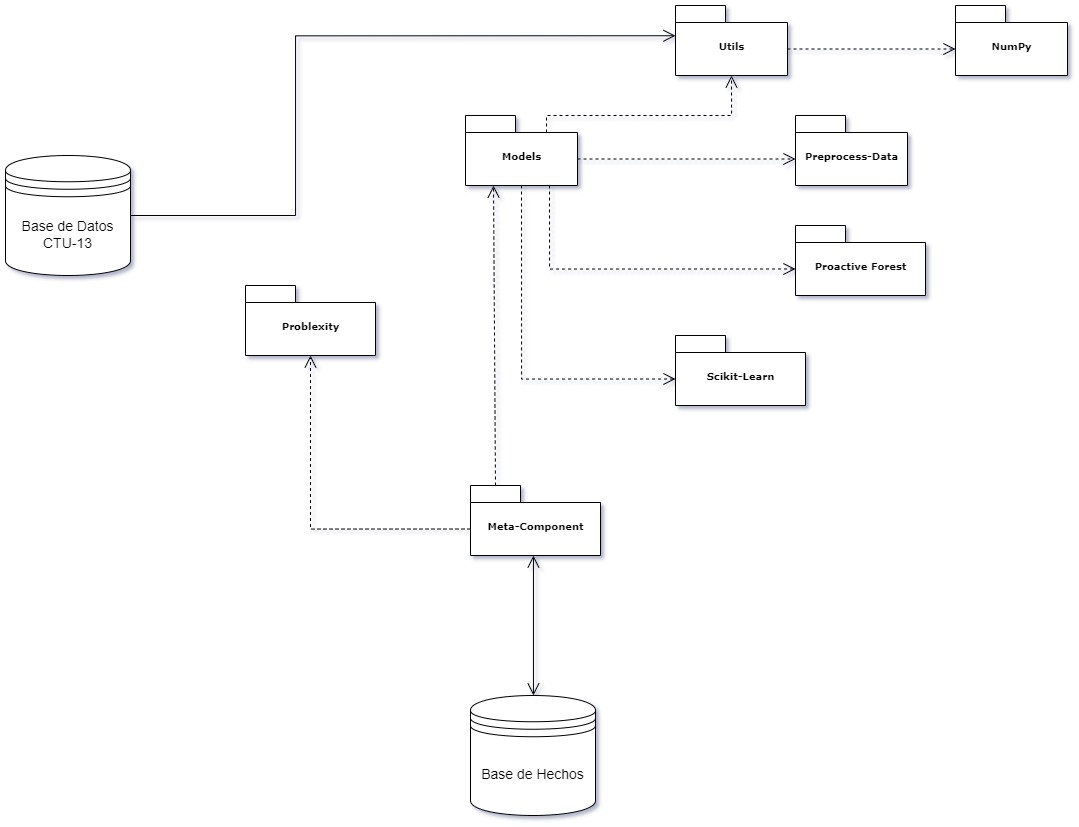


Figura 4 Representación del patrón Blackboard

### Flujo de datos

Este estilo arquitectónico se centra en cómo los datos fluyen a través de los componentes del sistema para ser procesados, enriquecidos y utilizados en diferentes etapas del proceso[29].

#### Pipes and Filters

Provee una estructura para los sistemas que procesan un flujo de datos como se aprecia en la Figura 5. Cada paso de procesamiento está encapsulado en un componente filtro. El dato pasa a través de conexiones entre filtros adyacentes.

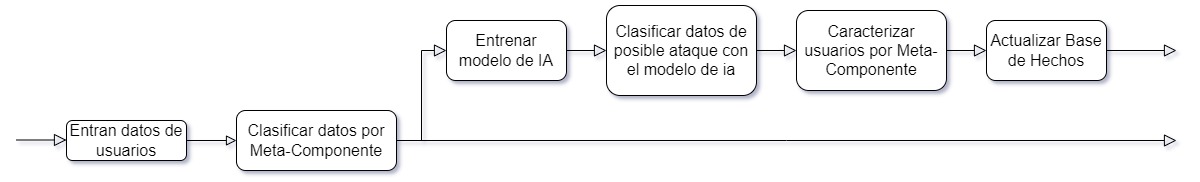


Figura 5 Representación de Filtros y Tuberías

## Principios de diseño

Los principios de diseño de software son reglas generales y recomendaciones que se aplican durante el proceso de diseño de software para resolver problemas de diseño comunes. Estos principios proporcionan orientación sobre cómo diseñar componentes de software y cómo relacionarlos entre sí para crear sistemas robustos y fáciles de mantener.

1. **Principio de responsabilidad única (SRP)**

Cada clase o componente debe tener una única responsabilidad. Esto no solo facilita el mantenimiento y la escalabilidad del sistema, sino que también mejora la comprensión y la colaboración entre los desarrolladores.

* **Paquete de Preprocesamiento**: Este componente se encargará exclusivamente de cargar y preprocesar los datos necesarios para el análisis de bots. Su responsabilidad es garantizar que los datos estén limpios, normalizados y en el formato correcto antes de ser utilizados por los modelos de IA. Al tener esta responsabilidad única, este componente puede ser reutilizado en diferentes partes del sistema o en otros proyectos que requieran preprocesamiento de datos similar.
* **Metacomponente de Cálculo de Métricas**: Este componente se centrará en calcular métricas específicas relacionadas con la detección de *bots*. Su responsabilidad es aplicar métodos estadísticos para extraer información relevante sobre los *bots,* que posteriormente determinará el flujo de acción.
* **Modelos de IA**: Cada modelo de IA que se utilice para la detección de *bots* tendrá su propia responsabilidad en función del motivo por el que fue entrenado. Por ejemplo, un modelo podría ser responsable de identificar patrones de comportamiento sospechosos, mientras que otro podría centrarse en analizar la estructura de los datos para detectar anomalías. Al asignar responsabilidades específicas a cada modelo, se asegura que cada uno pueda ser desarrollado, entrenado y actualizado de manera independiente, lo que facilita la experimentación y la optimización del rendimiento del sistema.
* **Base de Hechos**: La base de hechos servirá como un repositorio centralizado de datos y métricas. Su responsabilidad es almacenar y gestionar la información recopilada por los componentes del sistema, facilitando el acceso y la consulta de los datos por parte de los componentes que necesiten ellos. Al ser un componente de almacenamiento y acceso a datos, su responsabilidad se centra en la persistencia y recuperación de información, lo que lo mantiene simple y eficiente.

1. **Principio de inversión de dependencias (DIP)**

Al seguir el DIP, se promueve la reutilización de componentes de bajo nivel como el paquete de preprocesamiento en diferentes partes del sistema, lo que ayuda a reducir la duplicación de código y a mantener una base de código más limpia y mantenible. Se reduce el acoplamiento entre los diferentes módulos del sistema, lo que facilita la evolución independiente de cada componente y la introducción de cambios con menor impacto en el sistema en su conjunto. En otras palabras:

* + El módulo de detección de *bots* (alto nivel) depende del paquete de preprocesamiento (bajo nivel), en lugar de que sea al revés. Es decir, las dependencias deben apuntar hacia abajo.
  + En este sentido, el módulo de detección de *bots* utilizará y se apoyará en el paquete de preprocesamiento para realizar las tareas de preparación de datos necesarias para la detección efectiva de *bots*.
  + Esto permite que los componentes de bajo nivel, como el paquete de preprocesamiento, puedan ser reutilizables en otros contextos y se mantengan de manera independiente, lo que facilita la escalabilidad y mantenibilidad del sistema en general.

1. **Principio de abierto/cerrado (OCP)**

El Principio de Abierto/Cerrado (OCP) se enfoca en el diseño de sistemas que sean extensibles sin necesidad de modificar su código fuente existente. Este principio es fundamental en el desarrollo de software orientado a objetos, permitiendo que las clases, módulos y funciones sean diseñadas de tal manera que puedan ser ampliadas para soportar nuevas funcionalidades o cambios en los requisitos, manteniendo al mismo tiempo su integridad y funcionalidad original.

En el caso concreto del proyecto, cada modelo de IA puede tener un proceso de entrenamiento y evaluación distinto, pero todos deben ser capaces de interactuar con un paquete de preprocesamiento común que se encarga de cargar y preparar los datos para el análisis. En otras palabras, se tiene:

* Una clase abstracta ModeloAbstracto que define una interfaz común para todos los modelos de IA, como entrenarModelo, evaluarModelo, etc.
* Cada modelo de IA específico (ModeloIA1, ModeloIA2, etc.) hereda de ModeloAbstracto e implementa sus métodos específicos.
* El paquete de preprocesamiento interactúa con los modelos de IA a través de la interfaz definida en ModeloIAAbstracto, sin necesidad de conocer los detalles de implementación de cada modelo.
* Para agregar un nuevo modelo de IA, simplemente se crea una nueva clase que herede de ModeloIAAbstracto y se implementan sus métodos, sin necesidad de modificar el código existente o la interfaz de ModeloIAAbstracto.

1. **Principio de sustitución de Liskov (LSP)**

De acuerdo con el principio de Liskov, cualquier modelo de IA derivado de una clase base debe poder reemplazar a esa clase base sin cambiar el comportamiento esperado del sistema.

En el caso de los modelos de IA en el proyecto, cada modelo implementado puede ser utilizado de manera intercambiable en el sistema sin alterar su funcionamiento general. Por ejemplo, si se introduce un nuevo modelo de IA, este debe integrarse sin causar efectos no deseados en el comportamiento del software. Este principio promueve una arquitectura más flexible, consistente y fácil de mantener a medida que se introduzcan y evolucionen los modelos de inteligencia artificial utilizados para la detección de *bots*.

## Patrones de diseño

Los patrones de diseño son soluciones típicas a problemas recurrentes que se encuentran al diseñar software y sistemas de información. Estos patrones representan las mejores prácticas y experiencias probadas que han sido formalizadas para facilitar su reutilización y aplicación en diversas situaciones.

### Template method

En el módulo se han implementado varios modelos de clasificación con el objetivo de compararlos y saber cuál o cuáles obtienen una mayor precisión. Sin embargo,como se aprecia en la Figura 6 se ha establecido un comportamiento común entre todos, variando solamente la parte donde se inicializa y entrena el propio modelo pues este procedimiento varía en función del mismo.

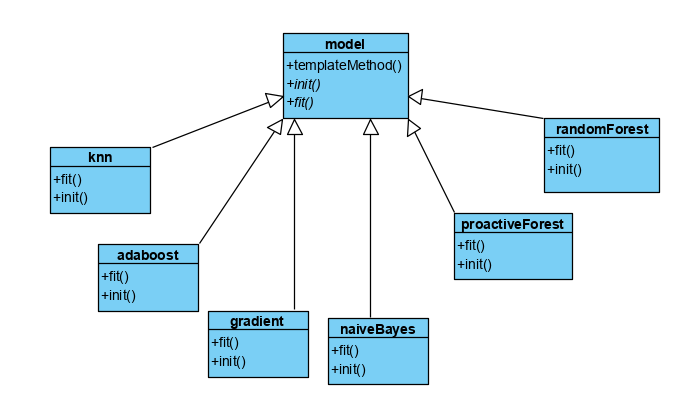


Figura 6 Patrón Template method

### Strategy

Como ya se dijo se implementaron varios modelos, de forma que se puedan analizar varios algoritmos en función de las necesidades del investigador, lo cual se pone de manifiesto en el patrón estrategia, o sea el contexto determina qué tipo de estrategia(modelo) aplicar para el análisis como se puede ver en la   
Figura 7.

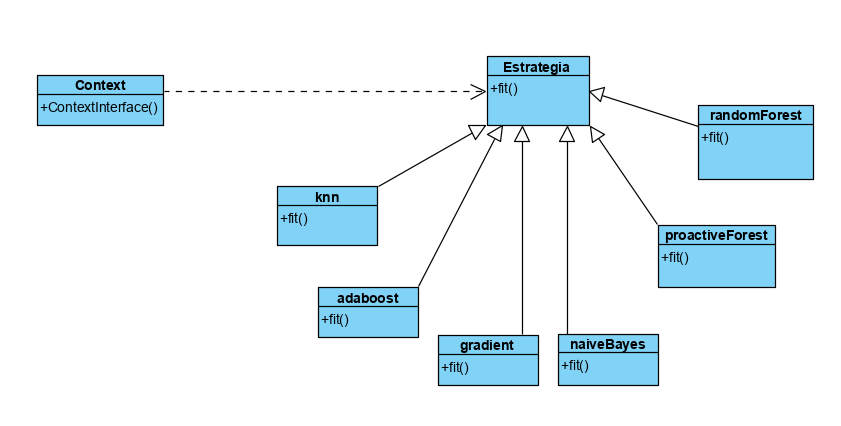


Figura 7 Patrón estrategia

### Fachada

El módulo no tiene interacción directa con los seres humanos, su comunicación con el cliente (Sitio de comercio electrónico) funciona a través de una interface facade como se puede apreciar en la Figura 8, que ayuda a simplificar la interacción y mejorar la usabilidad del sistema en general.

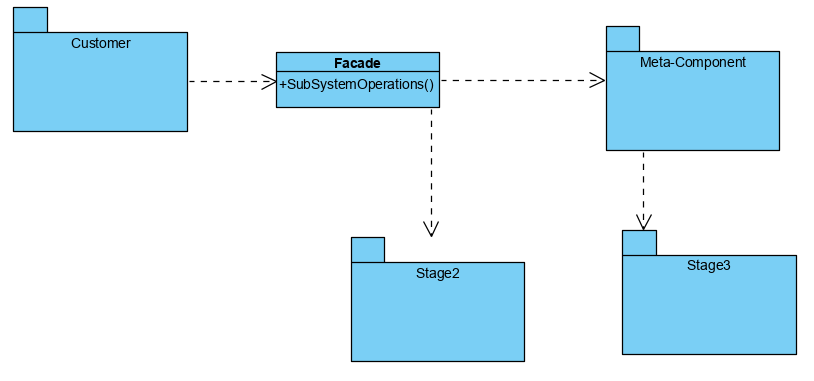


Figura 8 Patrón Facade

## Mecanismos de Diseño

Los mecanismos de diseño son conjuntos de clases y/o elementos de mayor complejidad que agrupados colaboran para solucionar requisitos que son comunes a varios casos de uso o incluso a varios proyectos. En este proyecto a partir del diagrama de clases de la Figura 10, se identificaron dos mecanismos principales, los cuales son: el preprocesamiento de los datos para que puedan ser trabajados correctamente por el modelo de aprendizaje supervisado y la caracterización de un conjunto de instancias de usuarios como amenazas o no mediante el uso del Meta-Componente.

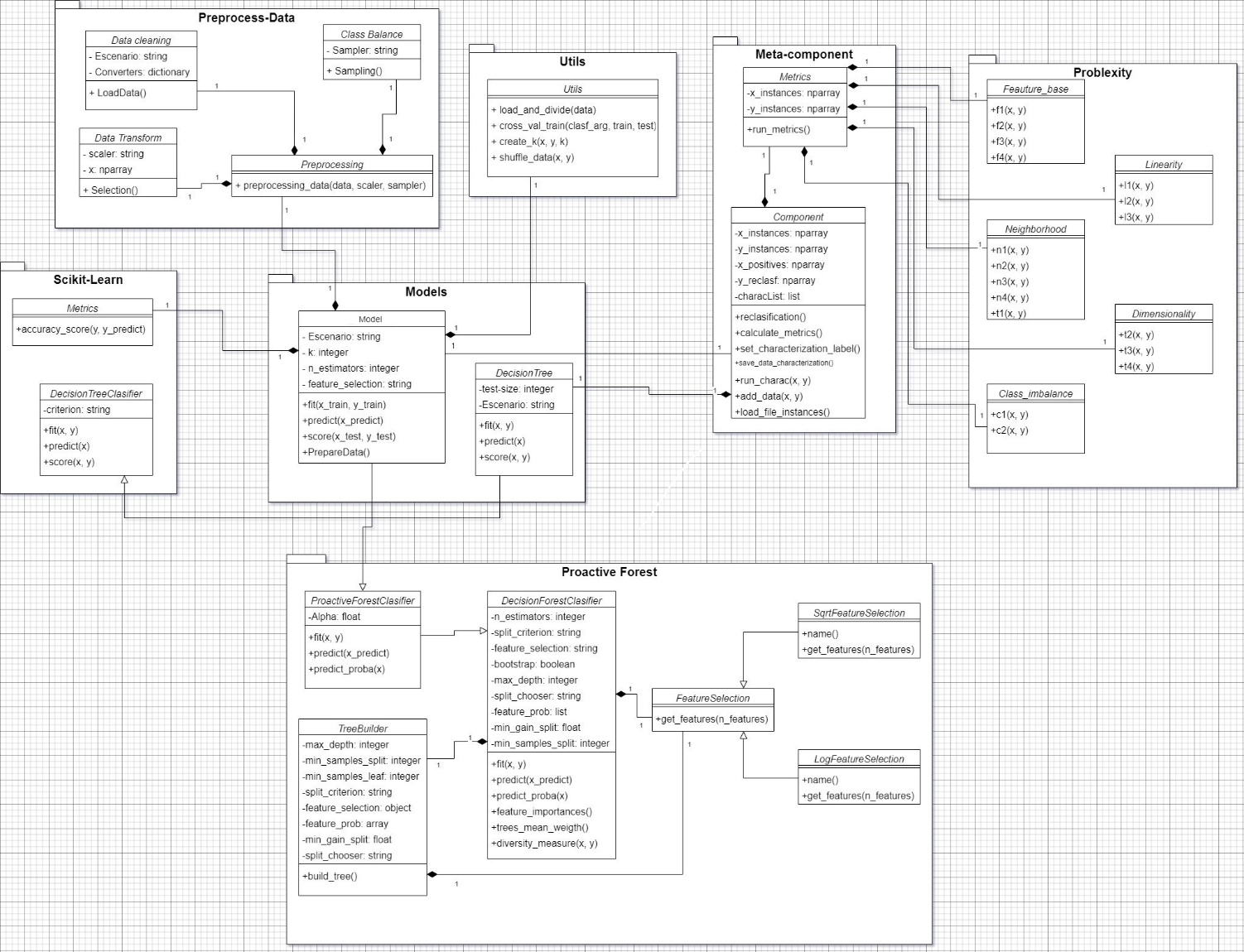


Figura 10 Diagrama de clases de la solución

### Preprocesamiento de datos

A partir del fragmento de diagrama de la Figura 11 se identifica el paquete Preprocess-Data como componente encargado del preprocesamiento de los datos a través de las clases Data-Cleaning, Class Balance,Data Transform y Preprocessing.

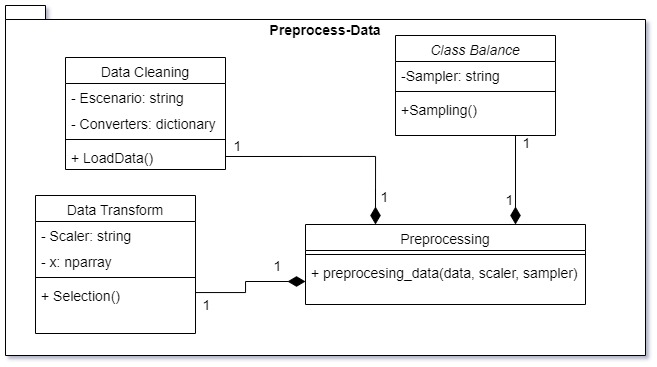


Figura 11 Diagrama de clases para el preprocesamiento

Como se muestra en la Figura 12 para adecuar los datos al proceso de minería, se aplicaron un conjunto de procedimientos para eliminar irregularidades, inconsistencias y ruidos en los datos, convertir, escalar y reducir de dimensionalidad. Utilizando el método loadData() todos los datos a tipo numérico, incluyendo las etiquetas de clase, donde las instancias de clase son background y normal, se trataron como usuarios humanos, asignando valor 0. Mientras que las de clase botnet, fueron tratadas como usuarios bots malignos, asignando valor 1. Además, se eliminaron los datos vacíos.

Después, en el método selection(), se normaliza la muestra de datos, estableciendo los valores en una escala de 0 a 1 y efectúa una reducción de la dimensionalidad, aplicando la técnica de PCA, que resultó en 7 componentes principales. Finalmente, en el caso de que el valor del sampler indique que procedimiento de balanceo se ejecutara, se efectúa un equilibrio de clases con el método Sampling(), aumentando el número de instancias de clase usuarios bots malignos en la base de datos ctu-13, al ser la clase minoritaria. En el caso de que se especifique que no se realizara un balanceo de clases se finaliza el preprocesamiento y se devuelven los datos.

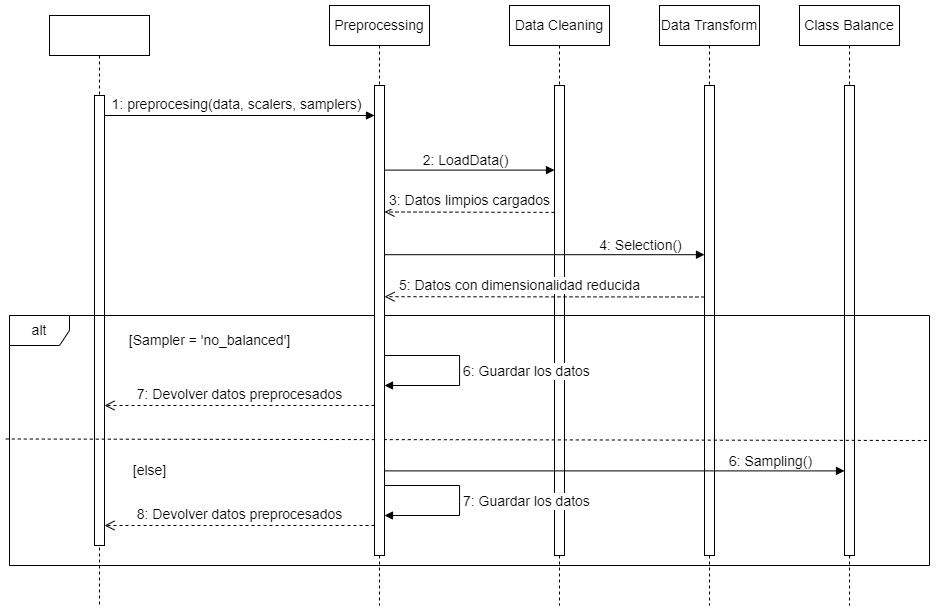


Figura 12 Diagrama de secuencia del preprocesamiento

### Caracterización de instancias por Meta-Componente

En este caso se evidencia en la Figura 13 como participan una mayor cantidad de paquetes para el correcto funcionamiento de este mecanismo:Meta-component, Models, Problexity

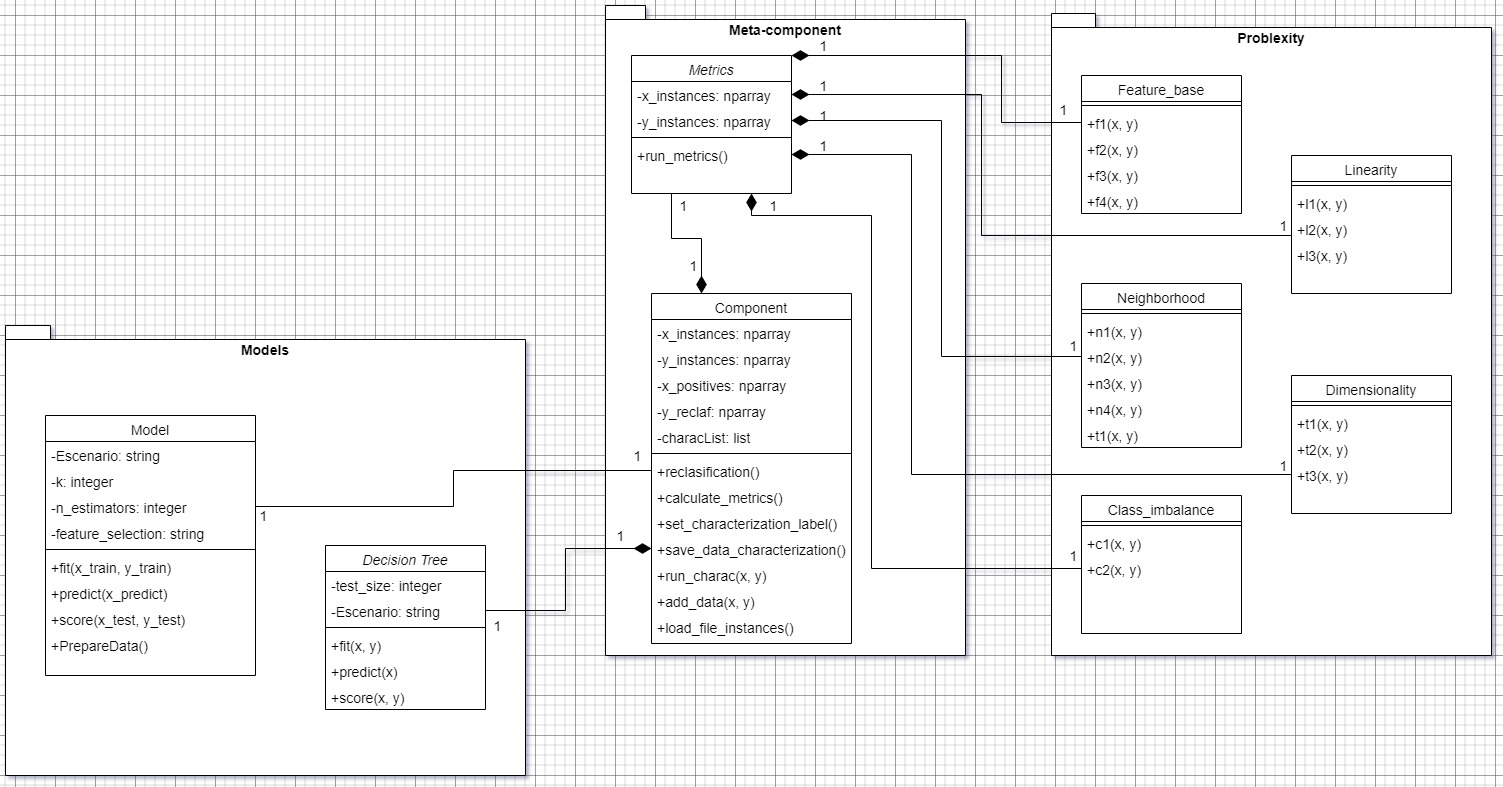


Figura 13 Diagrama de clases de la caracterización de instancias

En la Figura 14 se aprecia que el proceso comienza con la ejecución del método run\_charact (x,y), y los datos de ataque de bots clasificados. Se aplica el método set\_positives (x,y), para obtener los datos clasificados como humanos. Posteriormente estos pasan por la fase de reclasificación con la ejecución de reclasificaction(). Durante esta fase, se crea un árbol de decisión, se entrena con el método fit(x,y) con las instancias del fichero cargadas y se reclasifican los datos con predict(x\_positiv). A partir de obtener los datos reclasificados se calculan las métricas para caracterizar datos con calculate\_metrics(). Ejecutándose el método run\_metrics (xr,yr), para devolver la caracterización del conjunto de datos con las medidas de complejidad calculadas. Posteriormente se asigna el valor de clase a la caracterización obtenida con set\_characterization\_label(). Empleando el modelo para revisar el valor de clase de los datos clasificados como humanos, a través del método predict (x\_positive). Obteniéndose la caracterización con la clase correspondiente. Como último paso con save\_data\_characterization (), se guarda la caracterización de los datos en una base de hechos. Actualizando el valor de la misma en caso de tratarse de un conocimiento ya aprendido.

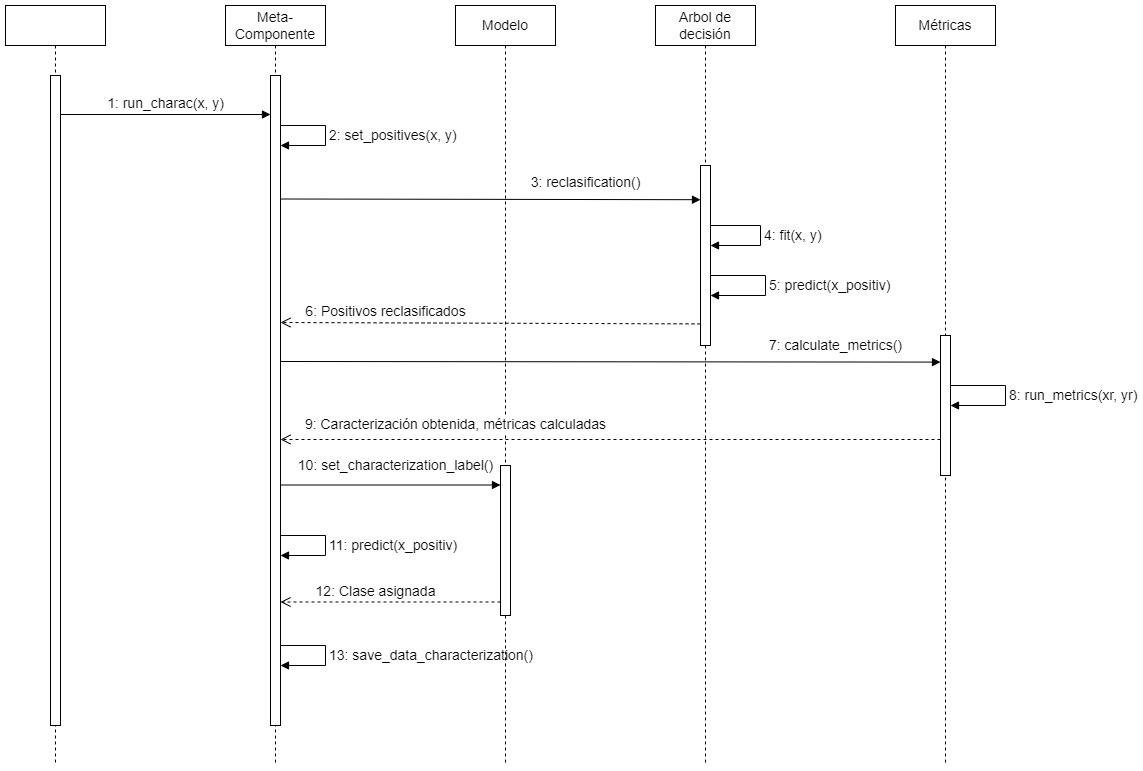


Figura 14 Diagrama de secuencia de la caracterización de instancias

# Conclusiones

En un mundo impulsado por la innovación tecnológica, la selección de elementos arquitectónicos para un módulo de detección de bots basado en IA en el comercio electrónico se convierte en un pilar fundamental para garantizar la seguridad y la integridad en las transacciones online.En este sentido la combinación de tecnologías avanzadas, como Python, y enfoques arquitectónicos innovadores, como la arquitectura por eventos, es la clave para desarrollar una solución eficaz y adaptable a las demandas cambiantes del entorno digital actual.

# Referencias bibliográficas

[1] D. P. Echevarría, "Módulo de detección de bots malignos basado en aprendizaje automático," Grado, Universidad Tecnológica de La Habana José Antonio Echeverría CUJAE, 2022.

[2] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[3] M. Abadi *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265-283.

[4] F. Pedregosa *et al.*, "Scikit-learn: Machine Learn-ing in Python, Journal of Machine Learning Re-search, 12," 2011.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter,* vol. 11, no. 1, pp. 10-18, 2009.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arXiv preprint arXiv:1602.07360,* 2016.

[7] M. Kuhn, "Building predictive models in R using the caret package," *Journal of statistical software,* vol. 28, pp. 1-26, 2008.

[8] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R news,* vol. 2, no. 3, pp. 18-22, 2002.

[9] M. Odersky, L. Spoon, and B. Venners, "Programming in Scala. Walnut Creek," ed: USA: Artima Press, 2010.

[10] C. R. Harris *et al.*, "Array programming with NumPy," *Nature,* vol. 585, no. 7825, pp. 357-362, 2020.

[11] O. V. Suárez and H. R. G. Diez, "Problemas desbalanceados en el aprendizaje profundo," *Serie Científica de la Universidad de las Ciencias Informáticas,* vol. 15, no. 6, pp. 18-34, 2022.

[12] I. K. Nti, O. Nyarko-Boateng, and J. Aning, "Performance of machine learning algorithms with different K values in K-fold cross-validation," *International Journal of Information Technology and Computer Science,* vol. 13, no. 6, pp. 61-71, 2021.

[13] C. A. Galan Guerra, "Computación Paralela en Python sobre un clúster de alto rendimiento," 2017.

[14] E. Dementei, "Desarrollo de un modelo interpretable utilizando redes neuronales y SVM," 2023.

[15] S. Ajel, F. Ribeiro, R. Ejbali, and J. Saraiva, "Energy Efficiency of Python Machine Learning Frameworks," in *International Conference on Intelligent Systems Design and Applications*, 2022: Springer, pp. 586-595.

[16] C. Torres Gómez and J. S. Torres Sandoval, "Seguridad en Redes con Python," 2022.

[17] V. Cosentino, J. Luis, and J. Cabot, "Findings from GitHub: methods, datasets and limitations," in *Proceedings of the 13th International Conference on Mining Software Repositories*, 2016, pp. 137-141.

[18] A. Dwaraki, S. Seetharaman, S. Natarajan, and T. Wolf, "GitFlow: Flow revision management for software-defined networks," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, 2015, pp. 1-6.

[19] S. Chakraborty and P. Aithal, "A Practical Approach to GIT Using Bitbucket, GitHub and SourceTree," *International Journal of Applied Engineering and Management Letters (IJAEML),* vol. 6, no. 2, pp. 254-263, 2022.

[20] Q. N. Islam, *Mastering PyCharm*. Packt Publishing Ltd, 2015.

[21] M. Kurniawan and B. Kurniawan, "Implementasi Pemrograman Python Menggunakan Visual Studio Code," *Jurnal Informatika dan Komputer (JIK),* vol. 11, no. 2, pp. 1-9, 2020.

[22] S. Saabith, T. Vinothraj, and M. Fareez, "A review on Python libraries and Ides for Data Science," *Int. J. Res. Eng. Sci,* vol. 9, no. 11, pp. 36-53, 2021.

[23] Y. H. Liu, *Python Machine Learning By Example: Build intelligent systems using Python, TensorFlow 2, PyTorch, and scikit-learn*. Packt Publishing Ltd, 2020.

[24] M. N. Gevorkyan, A. V. Demidova, T. S. Demidova, and A. A. Sobolev, "Review and comparative analysis of machine learning libraries for machine learning," *Discrete and Continuous Models and Applied Computational Science,* vol. 27, no. 4, pp. 305-315, 2019.

[25] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2016: IEEE, pp. 44-51.

[26] R. B. Gallinas, M. M. Sánchez, M. E. B. Gutiérrez, and A. M. F. García, "An event mesh for event driven IoT applications," *IJIMAI,* vol. 7, no. 6, pp. 54-59, 2022.

[27] H. F. O. Rocha, *Practical Event-Driven Microservices Architecture: Building Sustainable and Highly Scalable Event-Driven Microservices*. Springer, 2021.

[28] S. Zhelev and A. Rozeva, "Using microservices and event driven architecture for big data stream processing," in *AIP Conference Proceedings*, 2019, vol. 2172, no. 1: AIP Publishing.

[29] M. Cristiá, "Catálogo Incompleto de Estilos Arquitectónicos," *Cátedra Ing. Software Universidad Nacional de Rosario,* 2006.

[30] H. J. a. Chissingui, "Mecanismo de detección de bots basado en caracterización de datos," 2020.

[31] J. Komorniczak and P. Ksieniewicz, "problexity--an open-source Python library for binary classification problem complexity assessment," *arXiv preprint arXiv:2207.06709,* 2022.