

# **ESCUELA POLITÉCNICA NACIONAL**

**FACULTAD DE INGENIERÍA ELÉCTRICA Y  
ELECTRÓNICA**

**EVALUACIÓN DE PRÁCTICAS DE PRIVACIDAD EN  
APLICACIONES MÓVILES**

**DESARROLLO DE UN PROTOTIPO DE INTERFAZ GRÁFICA  
PARA EL DESPLIEGUE DE EXPERIMENTOS DE EVALUACIÓN  
AUTOMÁTICA DE ASPECTOS DE PRIVACIDAD EN  
APLICACIONES MÓVILES**

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO  
REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN  
TECNOLOGÍAS DE LA INFORMACIÓN**

**ANTONIO GREGORIO PEÑA TULCÁN**

**antonio.pena@epn.edu.ec**

**DIRECTOR: DANNY SANTIAGO GUAMÁN LOACHAMÍN**

**danny.guaman@epn.edu.ec**

**DMQ, AGOSTO 2023**

## **CERTIFICACIONES**

Yo, Antonio Gregorio Peña Tulcán declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

---

**ANTONIO GREGORIO PEÑA TULCÁN**

Certifico que el presente trabajo de integración curricular fue desarrollado por Antonio Gregorio Peña Tulcán, bajo mi supervisión.

---

**DANNY SANTIAGO GUAMÁN LOACHAMÍN**  
**DIRECTOR**

## **DECLARACIÓN DE AUTORÍA**

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Antonio Gregorio Peña Tulcán

Danny Santiago Guamán Loachamín

## DEDICATORIA

La bendición de Dios me ha permitido tener el amor y apoyo de mis padres, agradezco a mi madre Mercedes Tulcán quien ha dado todo por proteger y velar por sus hijos, a mi padre José Peña quien se ha esforzado toda la vida por seguir adelante y no dejar de lado ninguno de sus hijos y nietos y procurar que nunca les falte la educación ni la salud y por ser un ejemplo a seguir. Y agradezco por tener uno hermanos y hermanas que siempre me han apoyado y han confiado en mí.

Doy gracias por esos buenos amigos, que en este periodo universitario han sido un gran apoyo en lo personal y profesional, aquellos que han valorado mi esfuerzo y dedicación, y que aprecio mucho. A mi gran amigo Daniel Villavicencio que ha estado presente apoyándome desde el principio hasta hoy día. Y de manera especial a Sebastián Corrales, Andrés Sandoval y Jonathan Villareal, por su apoyo en esta travesía, en estos últimos semestres. Esperando que todos tengan éxito en su vida profesional.

Finalmente, agradezco a los docentes de la Escuela Politécnica Nacional por su dedicación y formación que dan a sus estudiantes, cada uno ha permitido que hoy este aquí culminado este periodo de formación profesional. Y mi más sincero agradecimiento al Dr. Danny Guamán, por permitirme ser parte de este trabajo, por su enseñanza y apoyo constante.

# ÍNDICE DE CONTENIDO

CERTIFICACIONES.....	I
DECLARACIÓN DE AUTORÍA.....	II
DEDICATORIA .....	III
ÍNDICE DE CONTENIDO.....	IV
RESUMEN .....	VII
ABSTRACT.....	VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO.....	1
1.1 Objetivo general.....	1
1.2 Objetivos específicos.....	1
1.3 Alcance .....	2
1.4 Marco teórico.....	3
1.4.1 Privacidad y protección de datos .....	3
1.4.1.1 Privacidad.....	3
1.4.1.2 Datos personales .....	4
1.4.1.3 Tecnologías para protección de datos personales.....	5
1.4.2 Evaluación de cumplimiento de privacidad y protección de datos.....	6
1.4.2.1 La plataforma CLIP .....	7
1.4.3 Tecnologías y herramientas usadas en este trabajo.....	8
1.4.3.1 Google Cloud Platform.....	8
1.4.3.2 Docker .....	8
1.4.3.3 React.....	9
1.4.3.4 Node JS.....	9
1.4.3.5 GraphQL.....	9
1.4.3.6 Prisma (ORM) .....	9
1.4.3.7 Material UI .....	9
1.4.3.8 Metodología ágil SCRUM .....	10
2 METODOLOGÍA .....	10
2.1 Metodología de desarrollo .....	11
2.2 Análisis de requerimientos .....	11
2.2.1 Requerimientos funcionales .....	11
2.2.2 Requerimientos no funcionales .....	24

2.2.3	Historias de usuario.....	24
2.2.4	Diagrama de casos de uso .....	27
2.2.5	Casos de uso.....	28
2.2.6	Product backlog.....	34
2.2.7	Planificación de sprints.....	34
2.3	Diseño .....	36
2.3.1	Modelo - Diseño de la base de datos .....	37
2.3.2	Vista - Diseño de Interfaces .....	38
2.3.3	Controlador .....	44
2.4	Implementación .....	49
2.4.1	Implementación primer y segundo sprint – Frontend del prototipo .....	50
2.4.2	Implementación tercer y cuarto sprint – Backend del prototipo de interfaz .....	54
2.4.3	Implementación quinto sprint – Integración del backend con el frontend - Implementación de ejecución local de un pipeline de evaluación con Docker.....	59
2.5	Validación y verificación .....	69
2.5.1	Pruebas – Módulos de análisis .....	70
2.5.1.1	Criterios de aceptación - Creación de un módulo de análisis - HU-01 .....	70
2.5.1.2	Pruebas funcionales - Creación de un módulo de análisis - HU-01.....	71
2.5.1.3	Análisis de aceptación - Creación de un módulo de análisis - HU-01 .....	74
2.5.1.4	Criterios de aceptación - Actualización de un módulo de análisis - HU-02..	76
2.5.1.5	Pruebas funcionales – Actualización de un módulo de análisis - HU-02 .....	76
2.5.1.6	Análisis de aceptación - Actualización de un módulo de análisis - HU-02...	79
2.5.1.7	Criterios de aceptación - Eliminación de un módulo de análisis - HU-03 ....	80
2.5.1.8	Pruebas funcionales - Eliminación de un módulo de análisis - HU-03.....	81
2.5.1.9	Análisis de aceptación - Eliminación de un módulo de análisis - HU-03 .....	82
2.5.1.10	Criterios de aceptación - Visualización de módulos de análisis-HU-04.....	83
2.5.1.11	Pruebas funcionales - Visualización de módulos de análisis - HU-04 .....	83
2.5.1.12	Análisis de aceptación - Visualización de módulos de análisis - HU-04 .....	86
2.5.2	Pruebas – Pipelines de evaluación .....	87
2.5.2.1	Criterios de aceptación - Creación de pipelines de evaluación - HU-05.....	87
2.5.2.2	Pruebas funcionales- Creación de pipeline de evaluación - HU-05.....	88
2.5.2.3	Análisis de aceptación - Creación de un pipeline de evaluación - HU-05....	90
2.5.2.4	Criterios de aceptación - Eliminación de un pipeline de evaluación - HU-06 91	
2.5.2.5	Pruebas funcionales- Eliminación de un pipeline de evaluación - HU-06....	91
2.5.2.6	Análisis de aceptación - Eliminación de un pipeline de evaluación - HU-06	93
2.5.2.7	Criterios de aceptación - Configuración de un pipeline de evaluación - HU- 07 y HU-08.....	93

2.5.2.8	Pruebas funcionales - Configuración de un pipeline de evaluación - HU-07	94
2.5.2.9	Análisis de aceptación - Configuración de un pipeline de evaluación - HU-07	98
2.5.2.10	Pruebas funcionales- Visualización de pipelines de evaluación - HU-09 ....	99
2.5.2.11	Pruebas funcionales- Visualización de pipelines de evaluación - HU-09 ....	99
2.5.2.12	Análisis de aceptación - Visualización de pipelines de evaluación - HU-09	103
2.5.2.13	Criterios de aceptación - Ejecución de un pipeline de evaluación - HU-10	103
2.5.2.14	Pruebas funcionales - Ejecución de un pipeline de evaluación - HU-10....	104
2.5.2.15	Análisis de aceptación - Ejecución de un pipeline de evaluación - HU-10	107
2.5.3	Pruebas – Pipeline de evaluación usando la imagen del módulo de tráfico “cliip/platform:traffic1.12” de CLIP.....	108
2.5.3.1	Criterios de aceptación - Pipeline de evaluación - módulo “traffic1.12”.....	108
2.5.3.2	Pruebas funcionales - Pipeline de evaluación - módulo “traffic1.12” .....	109
2.5.3.3	Análisis de aceptación - Pipeline de evaluación - módulo “traffic1.12” .....	115
2.5.4	Aclaración.....	116
3	CONCLUSIONES Y RECOMENDACIONES.....	116
3.1	Conclusiones .....	116
3.2	Recomendaciones .....	117
4	REFERENCIAS BIBLIOGRÁFICAS.....	119
5	ANEXOS .....	121

## RESUMEN

Este trabajo presenta el desarrollo de un prototipo de aplicación que soporte el despliegue de experimentos de evaluación de requisitos de privacidad en aplicaciones móviles.

Para ello, en el Capítulo 1, se realiza una breve descripción del marco teórico requerido en este trabajo. Se incluye una breve explicación de los conceptos relacionados con la privacidad y la protección de datos personales en el contexto de las aplicaciones móviles. Además, se explica brevemente la plataforma CLIP, que ha sido desarrollada en proyectos previos. Esta plataforma consiste en un conjunto de microservicios desplegados en contenedores Docker para la evaluación de requisitos de privacidad. Finalmente, en este capítulo se exponen las tecnologías y lenguajes de programación empleados para el desarrollo del prototipo.

En el Capítulo 2 se expone el proceso seguido para el desarrollo del prototipo. Basado en el proceso de ingeniería de software, primero, se realiza el análisis de requisitos funcionales y no funcionales, usando para ello historias de usuario y casos de uso. Posteriormente, se realiza el diseño del prototipo, que incluye la definición de su arquitectura, modelo de datos mediante un diagrama relacional, procesos mediante diagramas de actividades, e interfaces gráficas de usuario mediante mockups.

Luego, se proveen los detalles más relevantes de la implementación de prototipo de la aplicación. Basado en las mejores prácticas propuestas en SCRUM, se dividió la implementación en un conjunto de sprints.

Para el desarrollo del Frontend, se utilizó el lenguaje de programación *typescript*, y la biblioteca React. El Backend, también fue implementado con el lenguaje *typescript*; además, se usó GraphQL con el servidor Apollo y el ORM Prisma para la interacción con el modelo de datos. Finalmente, las pruebas de validación y verificación se usaron para evidenciar que se cumplen los requisitos inicialmente definidos.

Finalmente, el capítulo 3 muestra las conclusiones obtenidas y las recomendaciones que aportan experiencia para futuros trabajos relacionados.

**PALABRAS CLAVE:** privacidad, protección de datos, software, aplicaciones, Docker



## ABSTRACT

This work presents the development of a prototype application that supports the deployment of privacy requirements evaluation experiments in mobile applications.

For this purpose, in Chapter 1, we present a brief description of the theoretical framework required in this work. It includes a brief explanation of the concepts related to privacy and personal data protection in the context of mobile applications. In addition, the CLIP platform, which has been developed in a previous research project, is briefly explained. This platform consists of a set of microservices deployed in Docker containers for assessing privacy requirements. Finally, this chapter presents the technologies and programming languages used for developing the aforementioned prototype.

Chapter 2 presents the process followed for developing the prototype. Based on the software engineering process, firstly, we performed the analysis of functional and non-functional requirements by using user stories and use cases. Subsequently, the prototype design is performed, which includes the definition of its architecture, data model through a relational diagram, processes through activity diagrams, and graphical user interfaces through mockups.

Then, the most relevant details of the prototype implementation are provided. Based on the best practices proposed in SCRUM, the implementation was divided into a set of sprints. For the Frontend, we used the typescript programming language and the React library. The Backend was also implemented with the typescript language. Furthermore, GraphQL was used with the Apollo server and the ORM Prisma for the interaction with the data model. Finally, validation and verification tests were used to prove that the initially defined requirements are met.

Finally, Chapter 3 shows the conclusions obtained and recommendations that provide experience for future related work.

**KEYWORDS:** privacy, data protection, software, applications, Docker.

# **1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO**

Las aplicaciones móviles actualmente se desarrollan en todas partes del mundo y para múltiples propósitos. La evolución que han tenido estas aplicaciones ha ido facilitando al ser humano muchas de las labores cotidianas. Pero esto presenta una problemática a la hora de suministrar información personal, confiando en que los proveedores de aplicaciones no usen ni expongan estos datos personales. Con el objetivo de regular las prácticas llevadas a cabo por esos proveedores, se han definido leyes y reglamentos para la protección de los datos personales, que incluyen ciertas penalidades en caso de infringirlas.

En el Ecuador está presente la Ley Orgánica de Protección de Datos Personales (LOPD), que pretende garantizar la protección de datos personales. Esta ley regula, prevé y desarrolla un conjunto de principios, derechos, obligaciones que tienen que cumplir las organizaciones que procesan datos personales. En este contexto, es necesario contar con herramientas que permitan evaluar que los sistemas de software, incluyendo las aplicaciones móviles, cumplan con los requisitos establecidos en la LOPD.

En un proyecto de investigación previo se ha implementado la plataforma CLIIP, que permite analizar aplicaciones móviles, principalmente con el fin de evaluar aspectos de privacidad y protección de datos. CLIIP se basa en microservicios, que realizan tareas específicas. Al conectar apropiadamente estos microservicios se puede evaluar diferentes aspectos de privacidad y protección de datos. No obstante, la configuración y despliegue de estos microservicios, actualmente, se llevan a cabo de manera manual.

En el presente trabajo de integración curricular se desarrollará un prototipo de aplicación que provea una interfaz gráfica para el despliegue de experimentos de evaluación de aspectos de privacidad en aplicaciones móviles, mediante la plataforma CLIIP.

## **1.1 Objetivo general**

Desarrollar un prototipo de interfaz gráfica que permita desplegar experimentos de evaluación de aspectos de privacidad en aplicaciones móviles, mediante la plataforma CLIIP.

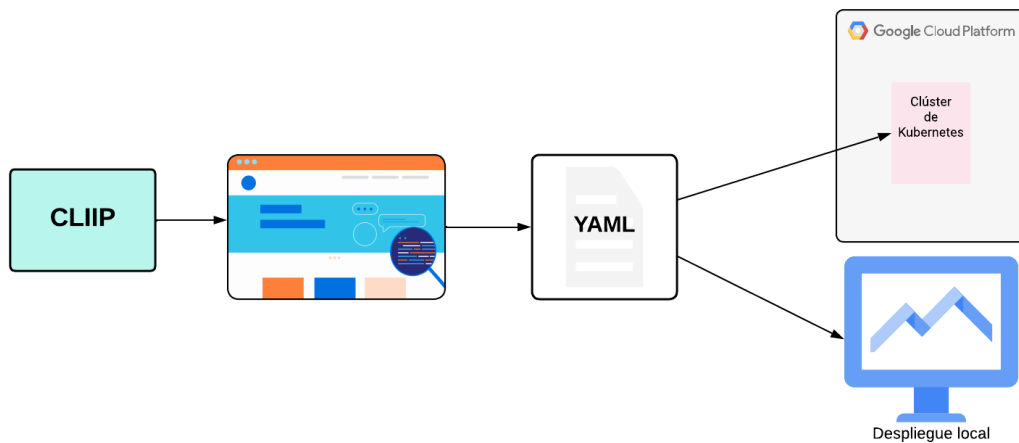
## **1.2 Objetivos específicos**

1. Estudiar el fundamento teórico relativo a la privacidad y protección de datos y las tecnologías requeridas para el desarrollo de este trabajo.

2. Estudiar la plataforma CLIP y el funcionamiento básico de sus componentes.
3. Analizar los requisitos para la implementación de una interfaz gráfica que permita ejecutar experimentos con los componentes de CLIP.
4. Diseñar la interfaz gráfica y artefactos de software requeridos que satisfagan los requisitos definidos.
5. Implementar la interfaz gráfica y artefactos de software previamente diseñados.
6. Validar y verificar la interfaz gráfica y artefactos implementados.

### 1.3 Alcance

Se utilizó la metodología ágil SCRUM para el desarrollo del prototipo de interfaz gráfica. Además, se llevó a cabo las actividades estándar de un proceso de desarrollo de software, esto es, la especificación de requisitos, diseño, implementación, y validación y verificación del prototipo. Como resultado, se obtuvo una aplicación web que permite el despliegue automático de los módulos de la plataforma CLIP. En la Figura 1.1 se muestran los elementos básicos que conforman el prototipo.



**Figura 1.1** Componentes fundamentales del prototipo a desarrollar

- Plataforma CLIP: tiene módulos de análisis basados en microservicios capaces de llevar a cabo tareas específicas con fin de evaluar ciertos aspectos de privacidad y protección de datos. Estos módulos han sido desarrollados en un proyecto de investigación previo y, en el presente trabajo, solamente serán usados para desplegarlos con el prototipo implementado. Un módulo de análisis para su despliegue requiere de un conjunto de parámetros. Por ejemplo, un módulo de

análisis de tráfico requiere la IP y Puerto del terminal móvil al que se redirigirá el tráfico de red.

- Frontend: permite registrar los módulos de análisis con sus respectivos parámetros. Permite, además, conformar un pipeline con microservicios convenientes para evaluar de aspectos de privacidad en aplicaciones móviles, mediante el uso de los módulos de análisis registrados previamente.
- Documento YAML: contiene la configuración del pipeline de evaluación a desplegar.
- Clúster de Kubernetes: está encargado de desplegar los microservicios contenidos en el documento YAML en la nube (p.ej., sobre Google Cloud Platform).
- Despliegue local: el computador local en el que se desplegará con Docker Compose los múltiples servicios contenidos en el documento YAML.

## **1.4 Marco teórico**

En esta sección se introduce a algunos conceptos importantes para comprender el contexto del presente trabajo y están relacionados con los datos personales y su protección. Posteriormente, se explica brevemente las tecnologías de protección de datos que estipula Troncoso et al [5] sobre la Privacidad y Derechos Online, así como los principios de protección que rigen en el Ecuador según la Ley Orgánica de Protección de Datos Personales (LOPDP). Finalmente se presenta brevemente las tecnologías y herramientas que se usaron en este trabajo.

### **1.4.1 Privacidad y protección de datos**

#### **1.4.1.1 Privacidad**

En el contexto de este trabajo, se usará la definición de privacidad propuesta por Helen Nissenbaum [1] dentro de un marco conceptual que conecta la protección de la información personal a las normas específicas que rigen un contexto particular.

En cualquier ámbito, la vida de las personas es regida por normas. Son normas de flujo de información basadas en los detalles de contextos, entornos o situaciones estereotipadas, que dan una perspectiva para una normativa de la privacidad en términos de integridad contextual. Nissenbaum [1] propone dos tipos de normas: normas de adecuación y normas de flujo o distribución. Así, la integridad contextual se mantiene o vulnera, con base en el respeto de ambas o la transgresión de al menos una de ellas.

Las normas de adecuación definen en un contexto concreto que información de una persona es adecuada o apropiada para ser revelada. Este tipo de normas generalmente

determinan la naturaleza de la información, y dentro de ese contexto se espera o exige que se revele la información. Por ejemplo, en el contexto médico, es apropiado que el paciente revele o comparta detalles de su salud física con el médico, pero no que el médico comparta detalles de su salud física con el paciente. Pero también lo importante es la información que no es apropiada compartir, y tener en cuenta que, en todo lado, en todo lugar que se rigen de al menos algunas normas informativas. Las personas en general tienen diferentes relaciones con diferentes personas, y eso es un punto importante. Apropiarse de información de una situación en particular para tomar ventaja sobre otra situación, constituye en una violación de los derechos personales, y violaciones de este tipo captan el concepto de adecuación.

A demás de las normas de adecuación, hay otro conjunto de normas contextuales que regulan el flujo o distribución de la información. Esta idea fue influenciada por la teoría pluralista de la justicia de Michael Walzer, que proporciona ideas útiles para la construcción de la privacidad como integridad contextual. Walzer concibe a las sociedades como una composición de esferas sociales distributivas, cada una caracterizada por un bien social internos. Bienes sociales como, la riqueza, los cargos políticos, el honor, las mercancías, la educación, la seguridad, el bienestar y el empleo. En una sociedad justa los resultados de distribuir los bienes son diferentes para diferentes personas y diferentes esferas. Estos bienes sociales están sujetos a criterios o principios que difieren según la esfera en las que operan. Lo importante además de, si la información es apropiada o inapropiada, es la distribución o flujo de la información y si este flujo respeta las normas contextuales de flujo de información. Entre las normas de flujo están, la libre elección, la discreción, la confidencialidad, la necesidad, el derecho y la obligación. Por lo que hay distintas formas posibles de normas de información en cuanto a su flujo.

La privacidad en base a la integridad contextual tiene que ver con un subconjunto de estas normas ya mencionadas, enlazado al riesgo de amenaza a estas normas informativas.

#### **1.4.1.2 Datos personales**

Según la Ley Orgánica de Protección de Datos Personales (LOPDP) [3], establece que; dato personal es, “el dato que identifica o hace identificable a una persona natural, directa o indirectamente”. Considerando que a una persona la hace identificable el hecho de cuya identidad se pueda determinar particularmente mediante un identificador ya sea directa o indirectamente.

También define que, datos personales crediticios son, “datos que integran el comportamiento económico de personas naturales, para analizar su capacidad financiera”. Los datos crediticios de una persona según la superintendencia de bancos del Ecuador,

refleja, por individuo, las obligaciones contraídas por ese individuo con terceros, ya sea con bancos públicos, bancos privados, casas comerciales o cooperativas, por lo que esta información es muy sensible y las personas tienen la responsabilidad de mantener esta información privada o asegurarse de que lo esté.

#### **1.4.1.3 Tecnologías para protección de datos personales**

La recopilación, procesamiento y difusión de datos implica claramente problemas graves de privacidad, pudiendo causar a gran escala, daños irreversibles a la sociedad respecto a la democracia. Por ello es importante, más allá de la privacidad, garantizar la libertad de expresión, de decisión y de autodeterminación de las personas en los sistemas que construimos. La privacidad es: “el derecho a ser dejado en paz”, “el derecho a la autodeterminación informativa”, “la libertad de no sufrir restricciones irrazonables en la construcción de la propia identidad”. Ejemplos de datos privados que deben ser protegidos, son: el contenido de las comunicaciones de los usuarios, sus patrones de uso de servicio, incluso la propia existencia de los usuarios y sus acciones. Y los ejemplos de medios de protección son: la configuración del control de acceso, o la encriptación.

Según Troncoso [5], se consideran tres paradigmas de privacidad: El primero es la privacidad como confidencialidad, el segundo es la privacidad como control informativo, y tercero, la privacidad como transparencia.

##### **Privacidad como confidencialidad**

Una interpretación común de la privacidad es evitar que la información personal sea accesible para cualquier entidad. Por lo que el objetivo de las tecnologías de la privacidad es permitir el uso de los servicios minimizando la cantidad de información expuesta tanto explícita como implícita.

Para la confidencialidad de los datos se presentan dos enfoques para minimizar la cantidad de información expuesta. El primero, métodos que impiden el acceso no autorizado de forma comprobable, normalmente basados en primitivas criptográficas avanzadas para garantizar que no se pueda inferir ningún dato. El segundo, métodos de control de divulgación, para garantizar que la información divulgada se limita a una cierta cantidad o no la vincula con una persona en particular.

##### **Privacidad como control**

En muchos casos la revelación de datos puede ser inevitable, así que se aconseja tecnologías que respondan a dos aspectos importantes. El primero, permitir que los usuarios expresen cómo esperan que sus datos sean utilizados y revelados al proveedor de servicios, con el fin de evitar el tratamiento indeseado de estos datos. El segundo,

permitir a las organizaciones definir y aplicar políticas que impidan el uso indebido de la información.

### **Privacidad como transparencia**

Los mecanismos de transparencia analizan las actividades en línea de los usuarios con el fin de proporcionarles información sobre las implicaciones de sus acciones, o realizar auditorías para comprobar que no se ha violado la privacidad.

La transparencia basada en la retroalimentación, con mecanismos que hacen transparente el modo en que se recoge, agrega, analiza y utiliza la información para la toma de decisiones. Uno es el concepto de espejos de privacidad, que muestran a los usuarios como ven los demás sus datos en línea. Otro tipo de retroalimentación son los nudges de privacidad, que ayudan a los usuarios a tomar decisiones sobre su configuración de privacidad y seguridad, proporciona respuesta inmediata al permitir cancelar o modificar una acción en línea realizada. Los nudges pueden ser más sofisticados al usar aprendizaje automático y percibir que acciones en línea realizadas son adecuadas. Aunque esa respuesta inmediata con cada acción puede ser molesta para algunos usuarios.

#### **1.4.2 Evaluación de cumplimiento de privacidad y protección de datos**

La actual Ley Orgánica de Protección de Datos Personales del Ecuador, proveen las políticas de tratamiento de datos que sirven como principal fundamento para los servicios informen al usuario sobre el uso de sus datos personales. El tratamiento de estos datos como lo menciona uno de los principios que rigen la ley, la confidencialidad, deben concebirse en secreto, no comunicarse o tratarse para un fin distinto para el cual fueron recogidos, por tanto, el responsable del tratamiento de estos datos debe aplicar las medidas técnicas organizativas necesarias para cumplir con el principio de confidencialidad.

Por lo que en nuestro contexto la normativa vigente es la mencionada Ley Orgánica de Protección de Datos Personales que establece un conjunto de requisitos que tiene ser cumplido, y se rige, en caso de incumplimiento a estas normas, según el reglamento que se aplica a esta ley, de la cual se presenta principios importantes que rigen esta ley.

Se hace uso de plataformas como CLIP, que faciliten la evaluación de los aspectos de privacidad y protección de datos en una determinada aplicación móvil y que permite generar informes que proporciona resultados detallados de estas evaluaciones con el fin de establecer el cumplimiento de las normas de privacidad y protección de datos que cada aplicación estipula en sus reglamentos.

## 1.4.2

### 1.4.2.1 La plataforma CLIIP

Esta plataforma se basa en microservicios, que analiza aplicaciones móviles con el fin de evaluar los aspectos de privacidad y protección de datos, tiene varios microservicios que realizan tareas concretas. Al evaluar una aplicación se conectan varios módulos de análisis que tienen sus propias tareas específicas, y se ejecuta finalmente en conjunto todos estos módulos ya declarados en esta plataforma. Al conjunto de módulos de análisis se lo denomina pipeline de evaluación.

El orquestador de evaluación de la privacidad es el encargado de desplegar el pipeline de evaluación y ejecutarlo. Para ello, se crea manualmente un archivo YAML que refleja el pipeline de evaluación.

El núcleo de CLIIP se basa en los componentes de búsqueda aleatoria de aplicaciones, alimentación de aplicaciones, cola de descarga, descargador de APK, descargador de la política de privacidad y descargador de metadatos. Todos ellos implementados dentro en contenedores Docker.

Entre los Módulos de análisis de CLIIP, también implementados dentro de contenedores Docker, se encuentran los siguientes: módulo de **Análisis de tráfico**, que es el que captura, almacena y analiza el tráfico de red generado por una aplicación, capaz de detectar rastros de red que filtran ciertos datos personales importantes. El módulo de **Metadatos** esencialmente extrae los metadatos generados por las aplicaciones de Android, como el certificado de firmado de la aplicación o los permisos solicitados, entre otros. El módulo **Bibliotecas** que permite la extracción de bibliotecas de terceros utilizadas por determinada aplicación. El módulo **Resource Usage**, el uso de recursos, que determina que parte es la que accede a los recursos sensibles a la privacidad. El módulo **TICCA** que detecta fuentes y sumideros de APIs confidenciales. Y módulos de **Anotación**, de políticas de privacidad, que extraen las que son más relevantes.

En fin, CLIIP consta de módulos que pueden: rastrear aplicaciones móviles, políticas de privacidad y otros metadatos relevantes de Google Play Store; analizar las políticas de privacidad del lenguaje natural para extraer algunas prácticas de privacidad comprometidas; analizar el comportamiento de las aplicaciones móviles en base a técnicas de análisis estático, dinámico y de tráfico para extraer las prácticas de privacidad llevadas a cabo y, por último; analizar posibles problemas de cumplimiento comparando las prácticas de privacidad de la aplicación con las prácticas de privacidad comprometidas de la política.



Por ejemplo, la plataforma, para evaluar el cumplimiento de una aplicación con las transferencias transfronterizas requiere básicamente dos entradas: los flujos de datos personales de la aplicación y las prácticas comprometidas en su política de privacidad, así que es necesario interpretar todas las prácticas pertinentes de la política de privacidad para determinar si las transferencias transfronterizas detectadas se han divulgado adecuadamente.

### **1.4.3 Tecnologías y herramientas usadas en este trabajo**

En esta subsección se explican brevemente las tecnologías empleadas durante el desarrollo y despliegue del prototipo desarrollado en este trabajo. Primero, se explican las tecnologías empleadas para el desarrollo de la aplicación, incluyendo React, JavaScript, Node JS, GraphQL y Prisma. Posteriormente, se explica acerca de Google Cloud Platform y Docker, que son tecnologías de soporte para el despliegue de los microservicios.

#### **1.4.3**

##### **1.4.3.1 Google Cloud Platform**

Google Cloud Platform (GCP) es una plataforma integral de servicios en la nube ofrecida por Google que brinda recursos informáticos, almacenamiento y herramientas para desarrollar, desplegar y gestionar aplicaciones y servicios en línea de manera escalable y flexible.

Google Cloud permite ejecutar aplicaciones donde se las necesite, usando los datos necesarios de la aplicación se puede ejecutar en cualquier tipo de nube o entorno. Esta plataforma ofrece funciones de aprendizaje automático y de analíticas avanzadas que permite automatizar procesos, realizar predicciones inteligentes además de optimizar la gestión y operaciones. Además, se encarga de proteger los datos y las aplicaciones de actividades fraudulentas proporcionando encriptado de datos en reposo, en tránsito y en uso, manteniendo la confidencialidad de los datos.

##### **1.4.3.2 Docker**

Docker es una herramienta de software que posibilita la ágil creación, evaluación y despliegue de aplicaciones. Esta tecnología empaqueta el software en componentes uniformes conocidos como "contenedores", que engloban todos los elementos esenciales para la ejecución del software, abarcando bibliotecas, utilidades del sistema, código y entorno de ejecución.

### **1.4.3.3 React**

React es una biblioteca de JavaScript para construir interfaces de usuario del tipo SPA (*Single Page Applications*). De este modo, esta biblioteca ayuda a la creación de interfaces de usuario que son interactivas y con vistas que se aproximan a una aplicación de escritorio. Además, esta biblioteca permite construir aplicaciones a partir de componentes (similares a una clase o una función) que favorecen el desarrollo modular y la reutilización de código. Otra característica relevante de React consiste en la ejecución eficiente de las interfaces, gracias al algoritmo diferencial. Este algoritmo permite actualizar solo los elementos HTML que han cambiado y no todo el DOM del navegador.

### **1.4.3.4 Node JS**

Originalmente, JavaScript fue concebido para ejecutarse en un navegador (p.ej., Chrome, Edge, Opera, etc.); es decir, se lo utilizaba para construir el Frontend. No obstante, NodeJS provee un entorno de ejecución de JavaScript sobre un sistema operativo como Windows y Linux. De este modo, se puede usar JavaScript para implementar también el backend de una aplicación.

### **1.4.3.5 GraphQL**

Esta tecnología posibilita la consulta y manipulación de datos de una manera que es más eficiente y flexible en comparación con métodos tradicionales como REST. En lugar de requerir múltiples puntos finales (*endpoints*) para diferentes tipos de datos, GraphQL permite solicitar precisamente los datos necesarios mediante una única consulta. Esto facilita una comunicación más eficiente con los servidores.

### **1.4.3.6 Prisma (ORM)**

ORM es un modelo de programación para el mapeo objeto-relacional que facilita la interacción con una Base de Datos y agiliza las tareas en relación a la persistencia de datos. Utiliza su propio lenguaje de consulta PQL *Prisma Query Language* para interactuar con la base de datos, más flexible y expresivo que SQL, facilitando la construcción de consultas y operaciones en la Base de Datos.

### **1.4.3.7 Material UI**

Es una biblioteca de componentes de interfaz de usuario que facilita el desarrollo de aplicaciones web y móviles estilizadas utilizando para ello los principios de Material Design. Material UI proporciona una amplia variedad de componentes predefinidos y estilizados, como botones, tarjetas, barras de navegación, etc. Estos componentes han sido utilizados en este proyecto para desarrollar las interfaces gráficas de usuario requeridas.

#### **1.4.3.8 Metodología ágil SCRUM**

Scrum es un enfoque de gestión de proyectos dentro de la metodología ágil que asiste a los equipos en la organización y administración de tareas proporcionando un conjunto de valores, principios y métodos estructurados. El proceso de construcción de un proyecto se compone de diferentes interacciones llamadas *Sprints*, cada una con una duración fija de 1 a 4 semanas. Los *Sprints* facilitan la construcción del producto de manera incremental. Es necesario recopilar todas las ideas, funcionalidades y demás elementos que van a ser parte del producto. Este conjunto de elementos se llama Pila de Producto o *Product Backlog*.

Para poder gestionar el *Product Backlog* y toda comunicación acerca del producto, existe un rol en SCRUM llamado Dueño de Producto, o *Product Owner*, quien es responsable de mantener la visión del producto que se estará construyendo. También tiene la función de mantener actualizada la pila de producto y las funcionalidades, con prioridad para cada uno.

El equipo de construcción es el encargado de entregar en cada Sprint una parte del incremento del producto. El equipo de construcción de software está conformado por desarrolladores, arquitectos, testers y personas que estén estrechamente ligadas a la construcción del producto.

La selección de los items del product backlog (PBI) y sus respectivas tareas técnicas se determinan Reunión de Planificación de Sprint o *Sprint Planing*. La reunión se dará entre el *Product Owner*, el Equipo de Construcción y el *Scrum Master*, la reunión debe resultar en determinar una Pila de *Sprint* y con objetivos claros. El *Scrum Master* es el vigía del proceso ayudando a resolver problemáticas y asegurándose de que todo se lleve a cabo, además de estar pendiente de las personas que conforman el equipo.

Uno de los atributos que tiene el *Product Backlog* es la estimación, que ayuda en la planificación del *sprint*, para este trabajo la estimación se basa en la serie de Fibonacci es decir toma los valores siguientes: 1, 2, 3, 5, 8, 13, etc.

## **2 METODOLOGÍA**

En este apartado se describe la metodología de desarrollo usada en este trabajo, previo a las fases de análisis, diseño, implementación y la validación y verificación del prototipo desarrollado.

## 2.1 Metodología de desarrollo

Basándose en la metodología de desarrollo ágil, SCRUM, el Equipo Scrum estuvo conformado por los miembros mostrados en la Tabla 2.1.

**Tabla 2.1.** Miembros del equipo *Scrum*

<b>Rol</b>	<b>Nombre</b>
<i>Scrum master</i>	Director del proyecto
<i>Product owner</i>	Director del proyecto
Equipo de desarrollo	Antonio Peña

A continuación, se describe el proceso de desarrollo seguido en el desarrollo del prototipo.

## 2.2 Análisis de requerimientos

En esta sección se realiza el análisis de requisitos del prototipo mediante las historias de usuario (HU) y casos de uso. Posteriormente, se define el *Product Backlog*, *Sprints* y la planificación de los Sprints.

### 2.2.1 Requerimientos funcionales

Los requerimientos funcionales obtenidos principalmente como resultado de las reuniones con el *product owner* son los siguientes:

- El prototipo de interfaz en principio no tiene un control de acceso o inicio de sesión.
- El prototipo de interfaz debe ser capaz de mostrar una tabla con los módulos de análisis existentes, así como crear, editar y eliminar módulos. Para la creación y edición de módulos se requiere el ingreso del nombre, la versión y los nombres de los parámetros necesarios.
- El prototipo de interfaz debe ser capaz de mostrar una tabla con los pipelines de evaluación existentes, así como crear, editar y eliminar pipelines. Para la creación y edición de pipelines se requiere el ingreso del nombre y la versión.
- El prototipo de interfaz, en la tabla de pipelines de evaluación, debe permitir seleccionar uno de los pipelines para su configuración siendo capaz de mostrar una tabla con los módulos configurados en el pipeline, así como la capacidad de agregar y configurar más módulos, además poder editar y eliminar los módulos ya configurados en el pipeline.

- El prototipo de interfaz, en la configuración de un módulo en un pipeline de evaluación, debe ser capaz de mostrar una tabla con todos los módulos disponibles, así como la opción de selección que permite configurar el módulo seleccionado, insertando los valores de los parámetros que contiene este módulo.
- El prototipo de interfaz debe ser capaz de mostrar un pipeline de evaluación configurado, indicando en una tabla los módulos configurados en el pipeline, con la capacidad de ejecutar pipeline de evaluación con un botón, así como detener la evaluación con otro botón.
- El prototipo de interfaz al presionar el botón de ejecución del pipeline de evaluación debe ser capaz de generar un archivo YAML en el servidor, este debe contener los microservicios (módulos de análisis configurados en el pipeline ejecutado) de CLIP que se deben ejecutarse inmediatamente después de la creación del archivo YAML.

### 2.2.2 Requerimientos no funcionales

Los requerimientos no funcionales son los siguientes:

- El prototipo debe ser capaz de desplegar microservicios que se encuentran en contenedores Docker. Docker podrá estar instalado de manera local o en la nube.

### 2.2.3 Historias de usuario

La plantilla de historia de usuario que se ha usado en este trabajo tiene la estructura que se describe a continuación:

- “Como [perfil]”: ¿a quién está destinado este desarrollo? ¿para quién estamos creando esta función de software? ¿qué tipo de características del producto quiere el usuario final? ¿cuáles son los datos demográficos y psicográficos del usuario final?
- “Quiere”: aquí detallamos su propósito, no las acciones específicas que emplean. ¿Cuál es su verdadera meta? Esta exposición debe ser independiente de las ejecuciones correctas; si se menciona algún componente de la interfaz y no el propósito del usuario, se incurre en un error.
- “Para”: ¿cómo se relaciona su impulso inmediato de llevar a cabo una acción con una visión global? ¿Qué beneficio busca alcanzar? ¿Cuál es el desafío fundamental que requiere solución?

Por ejemplo:

Como Experimentador.

Quiero gestionar los módulos de análisis.

Para crear, actualizar, borrar un módulo de análisis, y también poder listar los módulos de análisis creados.

A continuación, se presentan las historias de usuario obtenidas:

**Tabla 2.2.** Historia de usuario HU-01: Creación de módulos de análisis

<b>Historia de usuario</b>	
<b>HU-01</b>	CRUD de módulo de análisis – Creación de módulo de análisis
<b>Como</b>	Experimentador
<b>Quiero</b>	crear módulos de análisis
<b>Para</b>	establecer el nombre de un módulo de análisis y su versión; ingresar y agregar todos los parámetros requeridos por un módulo de análisis (cada uno puede tener un número de parámetros distinto).

**Tabla 2.3.** Historia de usuario HU-02: Actualización de módulos de análisis

<b>Historia de usuario</b>	
<b>HU-02</b>	CRUD de módulo de análisis – Actualización de módulo
<b>Como</b>	Experimentador
<b>Quiero</b>	actualizar los módulos de análisis
<b>Para</b>	cambiar el nombre del módulo de análisis, si es el caso; cambiar los parámetros existentes o agregar más parámetros; listar el nombre de cada parámetro agregado, y; actualizar la lista de módulos de análisis.

**Tabla 2.4.** Historia de usuario HU-03: Eliminación de módulos de análisis

<b>Historia de usuario</b>	
<b>HU-03</b>	CRUD de módulo de análisis – Supresión de módulo
<b>Como</b>	Experimentador
<b>Quiero</b>	eliminar módulos de análisis.
<b>Para</b>	actualizar la lista de módulos de análisis.

**Tabla 2.5.** Historia de usuario HU-04: Visualización de módulos de análisis

<b>Historia de usuario</b>	
<b>HU-04</b>	CRUD de módulo de análisis – Visualización de módulos
<b>Como</b>	Experimentador
<b>Quiero</b>	visualizar una lista de los módulos de análisis

<b>Para</b>	conocer y analizar los módulos creados; poder filtrarlos por su nombre y/o por su versión; además de tener acceso para crear, actualizar o borrar los módulos.
-------------	--

**Tabla 2.6.** Historia de usuario HU-05: Creación de pipelines de evaluación

<b>Historia de usuario</b>	
<b>HU-05</b>	Definición de pipelines de evaluación en Docker – Creación de pipelines de evaluación
<b>Como</b>	Experimentador
<b>Quiero</b>	crear pipelines de evaluación
<b>Para</b>	establecer su nombre y versión.

**Tabla 2.7.** Historia de usuario HU-06: Eliminación de pipelines de evaluación

<b>Historia de usuario</b>	
<b>HU-06</b>	Definición de pipelines de evaluación en Docker – Eliminación de pipelines de evaluación
<b>Como</b>	Experimentador
<b>Quiero</b>	eliminar un pipeline de evaluación
<b>Para</b>	actualizar la lista existente.

**Tabla 2.8.** Historia de usuario HU-07: Configuración de módulos de análisis para un pipeline de evaluación particular

<b>Historia de usuario</b>	
<b>HU-07</b>	Definición de pipelines de evaluación en Docker – Configuración de módulos de análisis para un pipeline de evaluación particular
<b>Como</b>	Experimentador
<b>Quiero</b>	configurar un módulo de análisis
<b>Para</b>	instanciarlo mediante la asignación de valores concretos a los parámetros de un módulo creado con anterioridad; estos valores concretos dependen de la configuración específica requerida en un pipeline de evaluación.

**Tabla 2.9.** Historia de usuario HU-08: Configuración de pipelines de evaluación

<b>Historia de usuario</b>	
<b>HU-08</b>	Definición de pipelines de evaluación en Docker – Configuración de pipelines de evaluación
<b>Como</b>	Experimentador
<b>Quiero</b>	agregar los módulos de análisis configurados a una lista

<b>Para</b>	especificar un pipeline de evaluación.
-------------	--

**Tabla 2.10.** Historia de usuario HU-09: Visualización de pipelines de evaluación

<b>Historia de usuario</b>	
<b>HU-09</b>	Definición de pipelines de evaluación en Docker– Visualización de pipelines de evaluación
<b>Como</b>	Experimentador
<b>Quiero</b>	visualizar una lista de pipelines de evaluación.
<b>Para</b>	observar los pipelines que ya han sido creados; filtrar por su nombre y/o por su versión; además de tener acceso para crear, actualizar o borrar los pipelines

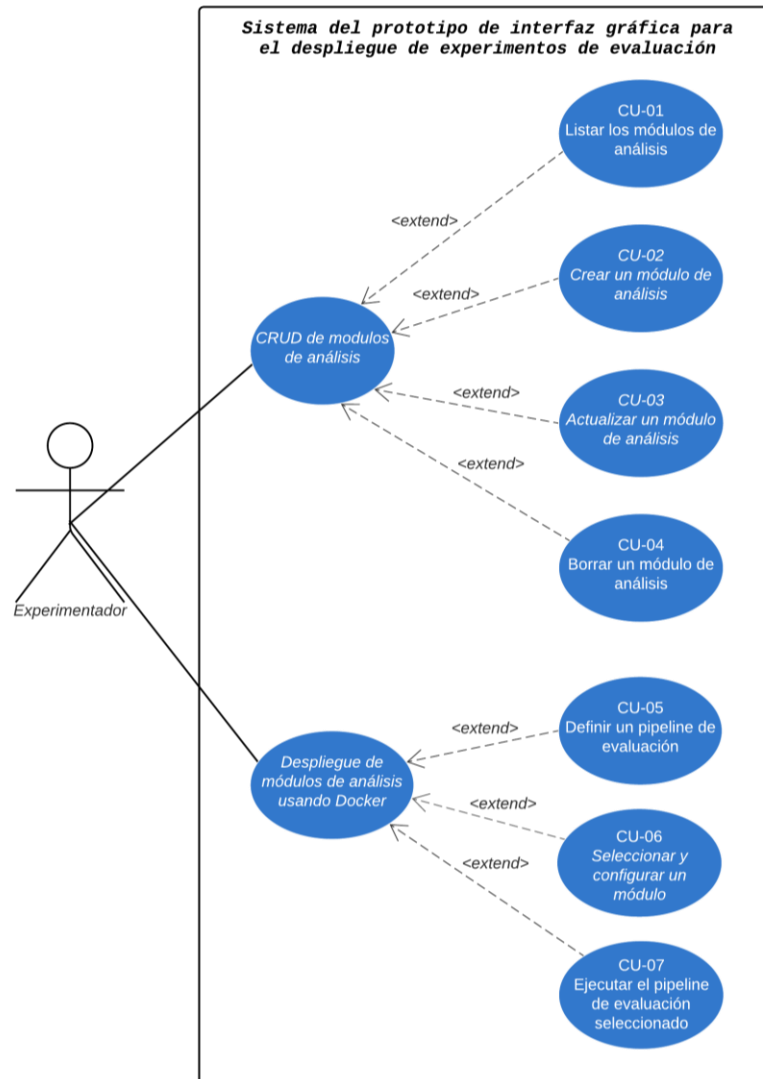
**Tabla 2.11.** Historia de usuario HU-10: Ejecución de pipelines de evaluación

<b>Historia de usuario</b>	
<b>HU-10</b>	Despliegue de pipelines de evaluación en Docker – Ejecución de pipelines de evaluación
<b>Como</b>	Experimentador
<b>Quiero</b>	seleccionar un pipeline de evaluación
<b>Para</b>	desplegar con Docker la configuración establecida en los módulos del pipeline de evaluación y detener la ejecución cuando se requiera.

## 2.2.4 Diagrama de casos de uso

En la Figura 2.1 se muestra el diagrama de casos de uso del prototipo desarrollado. El rol del Experimentador es el único definido para los propósitos de este trabajo. A futuro se puede implementar un control de inicio de sesión de tal manera que, además del rol de Experimentador exista un rol Administrador o Super usuario que cuente con permisos extendidos y de esa manera acceder a funcionalidades especiales para la administración del sistema.





**Figura 2.1** Diagrama de casos de uso del prototipo de interfaz

### 2.2.5 Casos de uso

A continuación, en las Tablas 2.12 a la 2.18, se desarrolla los casos de uso que permite describirlos y analizar su flujo.

**Tabla 2.12.** Caso de uso CU-01: Listar módulos de análisis

<b>Caso de uso</b>	<b>CU-01</b>
<b>Historias de usuario relacionadas</b>	HU-04
<b>Nombre</b>	Listar los módulos de análisis
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	16/08/2022
<b>Fecha de actualización</b>	23/08/2023

<b>Descripción</b>	Permite al experimentador visualizar la lista de módulos de análisis existentes.
<b>Actor</b>	Experimentador
<b>Precondiciones</b>	No requiere de control de acceso
<b>Flujo normal</b>	<p>1. El prototipo de interfaz muestra una pantalla con los módulos de análisis creados. En una tabla se muestra el nombre, la versión y las operaciones de actualización y eliminación que se pueden realizar sobre un módulo.</p> <p>2. El experimentador puede escoger alguna de las operaciones en alguno de los módulos ya sea filtrándolos por nombre o versión o simplemente buscándolo de manera visual.</p> <p>2.1. Si el experimentador presiona el botón para crear nuevo módulo de análisis, el sistema cambia a la pantalla de creación de nuevo módulo de análisis. Y finaliza caso de uso.</p> <p>2.2. Si el experimentador selecciona la operación de actualizar módulo de análisis, el sistema cambia a la ventana de actualización de módulo de análisis. Y finaliza caso de uso.</p> <p>2.3. Si el experimentador selecciona la operación de eliminar módulo de análisis, el sistema cambia a la ventana de eliminación de módulo de análisis. Y finaliza caso de uso.</p>
<b>Excepciones</b>	-
<b>Postcondiciones</b>	La lista de módulos de análisis debe actualizarse correctamente
<b>Extensiones</b>	<p>Crear un módulo de análisis</p> <p>Actualizar un módulo de análisis</p> <p>Eliminar un módulo de análisis</p>
<b>Inclusiones</b>	-
<b>Prioridad</b>	Alta

**Tabla 2.13.** Caso de uso CU-02: Crear un módulo de análisis

<b>Caso de uso</b>	<b>CU-02</b>
<b>Historias de usuario relacionadas</b>	HU-01
<b>Nombre</b>	Crear un módulo de análisis
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	16/08/2022
<b>Fecha de actualización</b>	23/08/2023
<b>Descripción</b>	Muestra al experimentador un formulario para crear un nuevo módulo de análisis.
<b>Actor</b>	Experimentador
<b>Flujo normal</b>	<p>1. El prototipo de interfaz muestra una ventana con un formulario donde puede ingresar los datos del nuevo módulo de análisis, incluyendo su nombre, versión, y parámetros requeridos. Para los parámetros se muestra un botón para agregar nuevo parámetro y poder ingresar el nombre del parámetro.</p>

	<p>2. El experimentador ingresa nombre del nuevo módulo.</p> <p>3. El experimentador ingresa la versión del nuevo módulo.</p> <p>4. El experimentador ingresa un nombre de parámetro para el nuevo módulo.</p> <p>4.1. El experimentador agrega más parámetros si se requiere, presionando el botón para agregar un nuevo parámetro.</p> <p>5. El experimentador presiona el botón <i>create</i> para crear el nuevo módulo de análisis.</p> <p>6. La interfaz confirma la creación del nuevo módulo y retorna a la interfaz de visualización de módulos de análisis.</p> <p>8. Finaliza caso de uso</p>
<b>Excepciones</b>	-
<b>Postcondiciones</b>	La lista de módulos de análisis debe actualizarse correctamente
<b>Extensiones</b>	-
<b>Inclusiones</b>	-
<b>Prioridad</b>	Alta

**Tabla 2.14.** Caso de uso CU-03: Actualizar módulo de análisis

<b>Caso de uso</b>	<b>CU-03</b>
<b>Historias de usuario relacionadas</b>	HU-02
<b>Nombre</b>	Actualizar módulo de análisis
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	20/08/2022
<b>Fecha de actualización</b>	23/08/2023
<b>Descripción</b>	Muestra al experimentador un formulario con los campos llenos para actualizar la información de un módulo de análisis seleccionado.
<b>Actor</b>	Experimentador
<b>Precondiciones</b>	No requiere de control de acceso
<b>Flujo normal</b>	<p>1. El prototipo de interfaz muestra una ventana con un formulario donde puede visualizar los datos del módulo de análisis seleccionado, nombre del módulo, versión del módulo y parámetros del módulo. Para los parámetros se muestra un botón para agregar nuevo parámetro y poder ingresar el nombre del parámetro.</p> <p>2. El experimentador ingresa el nuevo nombre del nuevo módulo.</p> <p>3. El experimentador ingresa la nueva versión del nuevo módulo.</p> <p>4. El experimentador ingresa un nuevo nombre de parámetro.</p> <p>4.1. El experimentador agrega más parámetros si se requiere, presionando el botón correspondiente a agregar parámetro.</p> <p>5. El experimentador presiona el botón <i>update</i> para actualizar el módulo de análisis.</p> <p>6. La interfaz confirma la actualización del módulo de análisis y retorna a la interfaz de visualización de módulos de análisis.</p> <p>7. Finaliza caso de uso</p>
<b>Excepciones</b>	-

<b>Postcondiciones</b>	La lista de módulos de análisis debe actualizarse correctamente
<b>Extensiones</b>	
<b>Inclusiones</b>	-
<b>Prioridad</b>	Alta

**Tabla 2.15.** Caso de uso CU-04: Eliminar módulo de análisis

<b>Caso de uso</b>	<b>CU-04</b>
<b>Historias de usuario relacionadas</b>	HU-03
<b>Nombre</b>	Eliminar módulo de análisis
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	20/08/2022
<b>Fecha de actualización</b>	23/08/2023
<b>Descripción</b>	Muestra al experimentador un modal con una advertencia de eliminación de un módulo de análisis y cuestionando al usuario si acepta o rechaza la eliminación.
<b>Actor</b>	Experimentador
<b>Precondiciones</b>	No requiere de control de acceso
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El sistema muestra un modal con una advertencia indicando que se va a eliminar un módulo.</li> <li>2. El experimentador presiona el botón <i>accept</i> para borrar el módulo de análisis.</li> <li>3. Si el experimentador presiona el botón <i>cancel</i>, se cancela la eliminación del módulo. Finaliza caso de uso.</li> <li>4. El sistema confirma la eliminación del módulo de análisis y retorna a la interfaz de visualización de módulos de análisis.</li> <li>5. Finaliza caso de uso.</li> </ol>
<b>Excepciones</b>	-
<b>Postcondiciones</b>	La lista de módulos de análisis debe actualizarse correctamente
<b>Extensiones</b>	
<b>Inclusiones</b>	-
<b>Prioridad</b>	Alta

**Tabla 2.16.** Caso de uso CU-05: Ejecutar módulos de análisis

<b>Caso de uso</b>	<b>CU-05</b>
<b>Historias de usuario relacionadas</b>	HU-05, HU-07, HU-08, HU-09
<b>Nombre</b>	Definir pipeline de evaluación
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	20/08/2022

<b>Fecha de actualización</b>	23/08/2023
<b>Descripción</b>	Muestra al experimentador una interfaz para la configuración de un pipeline de evaluación
<b>Actor</b>	Experimentador
<b>Precondiciones</b>	No requiere de control de acceso
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El prototipo de interfaz muestra una ventana con una tabla de los módulos de análisis definidos en el pipeline. Un botón superior <i>Add new</i> para agregar nuevo módulo.</li> <li>2. El experimentador presiona el botón <i>Add new</i>, para agregar un módulo de análisis a la lista de módulos para su ejecución.</li> <li>3. El sistema muestra la lista de módulos que puede seleccionar para configurar.</li> <li>4. El experimentador selecciona un módulo de análisis de la lista.</li> <li>5. El sistema muestra un formulario para configurar el módulo de análisis seleccionado.</li> <li>6. El experimentador configura el módulo y presiona el botón <i>Add</i>.</li> <li>7. El sistema retorna a la ventana del sistema de evaluación de módulos de análisis y visualiza la tabla de módulos.</li> <li>8. El experimentador presiona el botón <i>Execute</i> para ejecutar el análisis. Presiona el botón <i>Stop</i> para parar el análisis según convenga.</li> <li>8. Finaliza caso de uso</li> </ol>
<b>Excepciones</b>	-
<b>Postcondiciones</b>	La lista de módulos de análisis debe actualizarse correctamente
<b>Extensiones</b>	-
<b>Inclusiones</b>	Seleccionar y configurar un módulo de análisis
<b>Prioridad</b>	Alta

**Tabla 2.17.** Caso de uso CU-06: Seleccionar y configurar módulos de análisis

<b>Caso de uso</b>	<b>CU-06</b>
<b>Historias de usuario relacionadas</b>	HU-07
<b>Nombre</b>	Seleccionar y configurar un módulo de análisis
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	20/08/2022
<b>Fecha de actualización</b>	23/08/2023
<b>Descripción</b>	Muestra al experimentador la tabla de módulos de análisis para seleccionarlos y configurarlos para al pipeline de evaluación.
<b>Actor</b>	Experimentador
<b>Precondiciones</b>	No requiere de control de acceso
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El prototipo de interfaz muestra una ventana con una tabla de los módulos de análisis que puede seleccionar.</li> </ol>

	<p>2. El experimentador selecciona un módulo presionando el icono <i>Select</i> situado en la última columna de la tabla.</p> <p>3. El prototipo muestra un formulario, con el nombre y versión del módulo, y con los parámetros del módulo que debe configurar.</p> <p>4. El experimentador escribe los valores para todos los parámetros y presiona el botón Add para añadir el módulo configurado al pipeline de evaluación.</p> <p>5. El sistema confirma la configuración establecida en el módulo y retorna a la ventana de visualización de pipelines de evaluación de módulos de análisis.</p> <p>6. Finaliza caso de uso</p>
<b>Excepciones</b>	-
<b>Postcondiciones</b>	La lista de módulos de análisis para su evaluación debe actualizarse correctamente
<b>Extensiones</b>	-
<b>Inclusiones</b>	-
<b>Prioridad</b>	Alta

**Tabla 2.18.** Caso de uso CU-07: Ejecutar pipeline de evaluación

<b>Caso de uso</b>	<b>CU-07</b>
<b>Historias de usuario relacionadas</b>	HU-09, HU-10
<b>Nombre</b>	Ejecutar pipeline de evaluación usando Docker
<b>Autor</b>	Antonio Peña
<b>Fecha de creación</b>	20/08/2022
<b>Fecha de actualización</b>	25/08/2023
<b>Descripción</b>	Muestra al experimentador la tabla de pipelines de evaluación para seleccionarlos y ejecutarlos.
<b>Actor</b>	Experimentador
<b>Precondiciones</b>	No requiere de control de acceso, Docker Desktop debe estar activo
<b>Flujo normal</b>	<p>1. El prototipo de interfaz muestra una ventana con una tabla de los pipelines de evaluación que puede seleccionar.</p> <p>2. El experimentador selecciona un pipeline de evaluación haciendo clic sobre su nombre.</p> <p>3. El prototipo de interfaz muestra una ventana con una tabla de los módulos de análisis definidos en el pipeline.</p> <p>4. El experimentador presiona el botón <i>execute</i> ubicado en la parte inferior, y se ejecuta el pipeline con Docker</p> <p>5. En la interfaz de Docker Desktop, el experimentador comprueba la ejecución del contenedor correspondiente al pipeline ejecutado.</p> <p>7. El experimentador retorna al prototipo de interfaz, presiona el botón stop y la ejecución se detiene.</p> <p>6. Finaliza caso de uso</p>

<b>Excepciones</b>	-
<b>Postcondiciones</b>	-
<b>Extensiones</b>	-
<b>Inclusiones</b>	-
<b>Prioridad</b>	Alta

## 2.2.6 Product backlog

Con las historias de usuario definidas anteriormente se estableció el Product Backlog para priorizar el orden de implementación y tiempo de desarrollo aproximado de cada uno. El Product Backlog se muestra en la Tabla 2.19.

**Tabla 2.19. Product Backlog**

<b>Código HU</b>	<b>Nombre historia de usuario</b>	<b>Prioridad</b>	<b>Estimación</b>
HU-01	Creación de módulo	Media	3
HU-02	Actualización de módulo	Alta	8
HU-03	Supresión de módulo	Alta	8
HU-04	Visualización de módulos	Media	3
HU-05	Creación de pipelines de evaluación	Media	5
HU-06	Eliminación de un pipeline de evaluación		
HU-07	Configuración de módulos de análisis	Alta	5
HU-08	Configuración de un pipeline de evaluación	Alta	8
HU-09	Visualización de pipelines de evaluación	Media	3
HU-10	Ejecución de un pipeline de evaluación	Alta	21

## 2.2.7 Planificación de sprints

En esta sección, se resume las funcionalidades que se deben implementar en cada sprint.

### 1. Primer Sprint – Frontend del prototipo de interfaz – parte 1

En esta primera fase se implementaron las interfaces que corresponden a las acciones principales detalladas en las primeras cuatro HU (HU-01, HU-02, HU-03 Y HU-04). Es decir, se implementó el frontend relacionado con la creación, actualización, eliminación y visualización de módulos de análisis.

### 2. Segundo Sprint – Frontend del prototipo de interfaz – parte 2

En esta segunda fase se continuó con la implementación del frontend del prototipo. Es decir, se desarrolló el frontend correspondiente con las historias de usuario HU-

05, HU-06, HU-07, HU-08 y HU-09 para la creación, eliminación, configuración de pipelines de evaluación y visualización, respectivamente.

### 3. Tercer Sprint – Backend del sistema – parte 1

En la tercera fase se inicia con la implementación del backend del sistema. Para ello, se implementó el backend relacionado con las historias de usuario HU-01, HU-02, HU-03 y HU-04, para poder gestionar la creación, actualización, eliminación y visualización de módulos de análisis.

### 4. Cuarto Sprint – Backend del sistema – parte 2

En la cuarta fase, se continuó la implementación del backend del sistema. Concretamente, se implementó el backend correspondiente a las historias de usuario HU-05, HU-06, HU-07, HU-08 y HU-09, para poder gestionar la creación, eliminación, configuración y visualización de pipelines de evaluación.

### 5. Quinto Sprint – Integración del backend con el frontend y ejecución del pipeline usando Docker

En esta última fase, se implementó la funcionalidad definida en la historia de usuario HU-10, que consiste en la ejecución de un pipeline de evaluación. Para ello, se trabajó tanto en el Frontend como en el Backend, de tal manera que al llamar al evento de ejecución del pipeline, se realice la ejecución del pipeline configurado usando contenedores Docker.

La Tabla 2.20 muestra un resumen de la planificación de *sprints*. **Tabla 2.20.**

Planificación de *sprints*

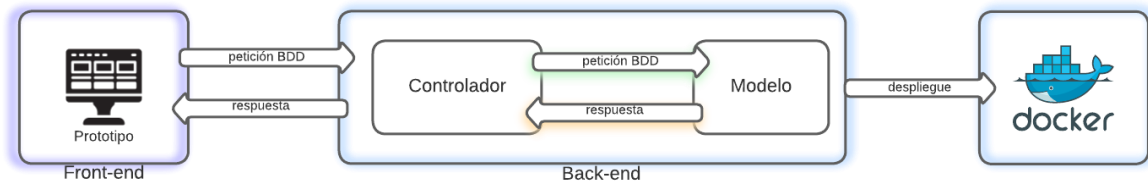
#	Sprint	Historia de usuario			
		Código	Nombre	Prioridad	Estimación
1	Frontend del prototipo de interfaz – parte 1	HU-01	Creación de módulo	Media	3
		HU-02	Actualización de módulo	Alta	8
		HU-03	Supresión de módulo	Alta	8
		HU-04	Visualización de módulos	Media	3
Total, de puntos del sprint					22
2	Frontend del prototipo de interfaz – parte 2	HU-05 HU-06	Creación de pipelines de evaluación Eliminación de un pipeline de evaluación	Media	5



		HU-07	Configuración de módulos de análisis	Alta	5
		HU-08	Configuración del pipeline de evaluación	Alta	8
		HU-09	Visualización de pipelines	Media	3
Total, de puntos del sprint					21
3	Backend del sistema – parte 1	HU-01	Obtención de módulos	Media	3
		HU-02	Creación de módulo	Alta	8
		HU-03	Actualización de módulo	Alta	8
		HU-04	Supresión de módulo	Media	3
Total, de puntos del sprint					22
4	Backend del sistema – parte 2	HU-05	Creación de pipelines de evaluación	Media	5
		HU-06	Eliminación de un pipeline de evaluación		
		HU-07	Configuración de módulos de análisis	Alta	5
		HU-08	Configuración del pipeline de evaluación	Alta	8
		HU-09	Obtención de pipelines	Media	3
Total, de puntos del sprint					21
5	Integración del backend con el frontend	HU-10	Ejecución de pipeline de evaluación	Alta	21

## 2.3 Diseño

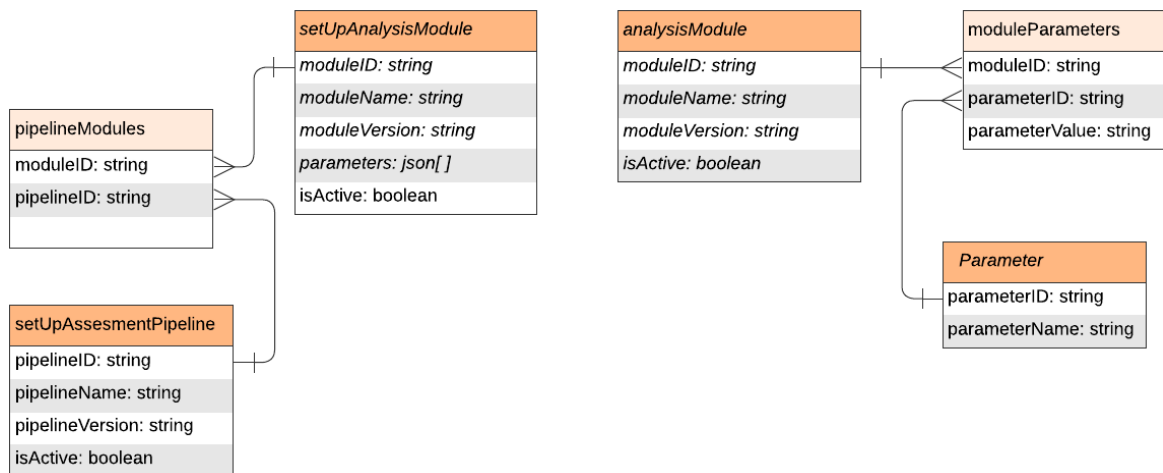
En esta sección se muestra el diseño del prototipo con base en los requisitos establecidos en la sección anterior. La Figura 2.2 muestra la arquitectura; el diseño de los componentes se describe en las siguientes secciones. Fundamentalmente, el frontend provee al experimentador las interfaces gráficas para poder configurar y desplegar un pipeline de evaluación. El back-end, por su parte, brinda persistencia de los datos requeridos, incluyendo aquellos requeridos por los módulos de análisis y pipelines de evaluación; este, además, interactúa con los contenedores Docker para ejecutarlos o detenerlos.



**Figura 2.2** Esquema general de la herramienta web

### 2.3.1 Modelo - Diseño de la base de datos

En esta sección se muestra el diagrama del modelo de datos usado en el prototipo. La Figura 2.3 muestra el diagrama relacional del modelo de datos empleado.



**Figura 2.3** Diagrama relacional del modelo de datos

A continuación, se describe brevemente las entidades incluidas en el modelo.

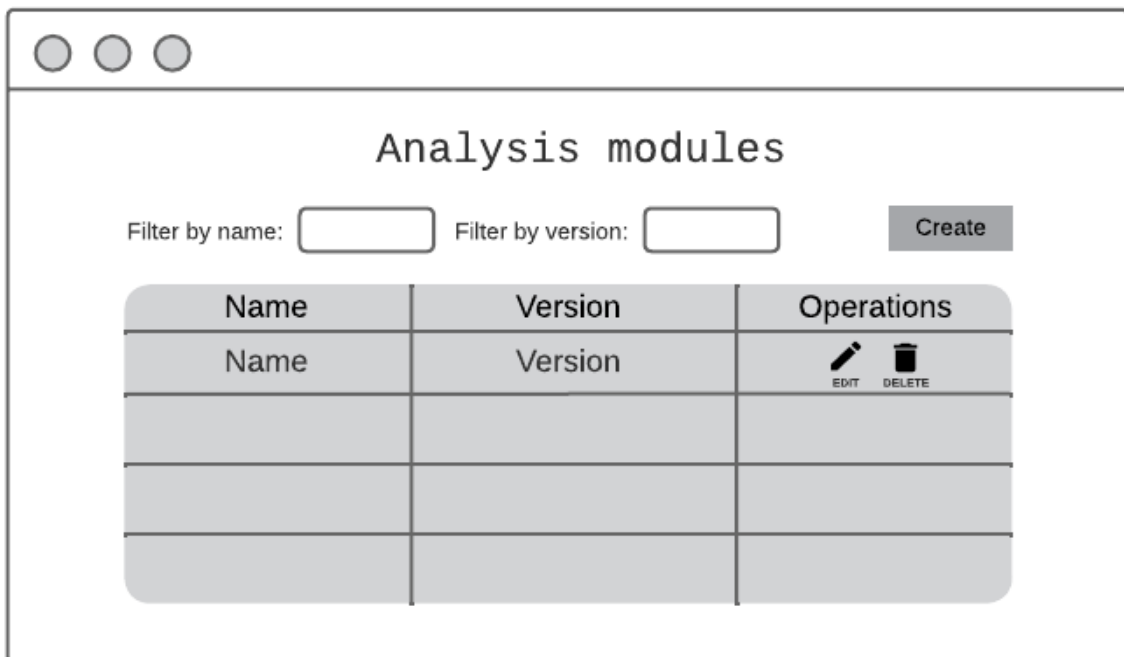
- **analysisModule:** esta tabla almacena los módulos de análisis que se van creando. Por cada módulo de análisis se almacena un identificador (`moduleID`), el nombre del módulo (`moduleName`), la versión del módulo (`moduleVersion`) y un indicador lógico que establece si el módulo está activo o inactivo para ser usado en un pipeline de evaluación (`isActive`).
- **parameters:** esta tabla almacena los parámetros que se van añadiendo en los módulos de análisis; los datos que guarda son, un identificador (`parameterID`) y el nombre del parámetro (`parameterName`).
- **moduleParameters:** esta tabla intermedia almacena la relación entre los parámetros y los módulos de análisis. De esta manera, se puede identificar que parámetros están asignados a cada módulo de análisis.
- **setUpAssesmentPipeline:** esta tabla almacena los pipelines de evaluación que se van creando. Los datos que guarda son: un identificador (`pipelineID`), el nombre del

pipeline de evaluación (pipelineName), la versión del pipeline (pipelineVersion), y un indicador lógico que establece si el pipeline está activo o inactivo (isActive).

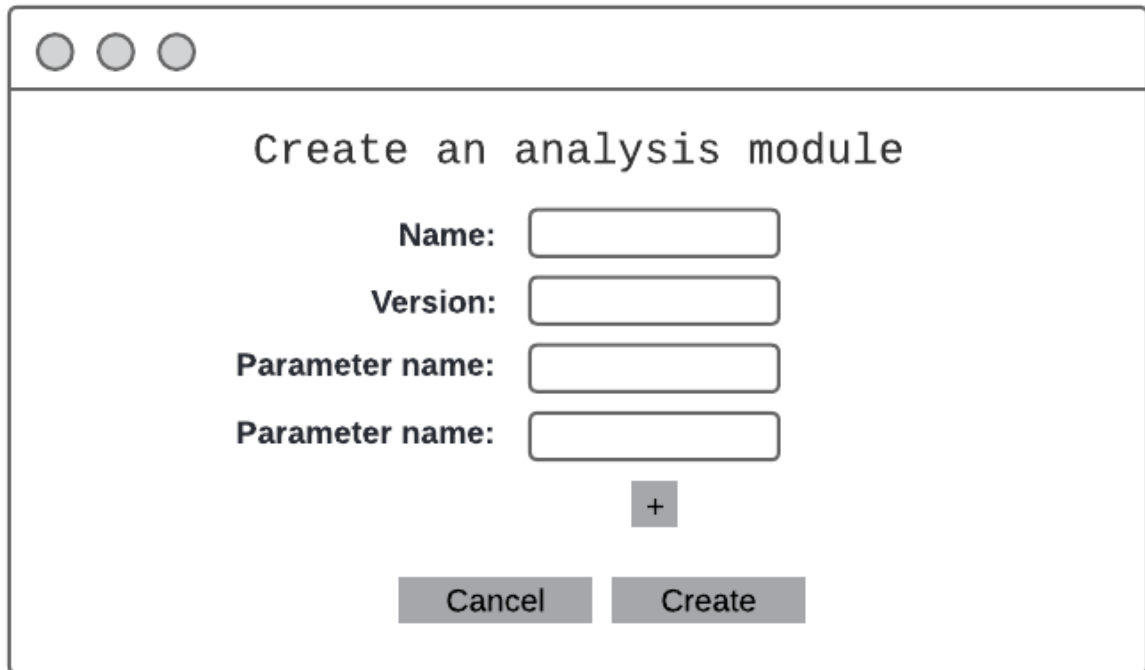
- setUpAnalysisModule: esta tabla almacena los módulos de análisis que conforman los pipelines de evaluación. Los datos que guarda son: un identificador (moduleId), el nombre del módulo (moduleName), la versión del módulo (moduleVersion), un array de json que almacena los parámetros configurados en el módulo (parameters) y un indicador lógico que establece si el módulo está activo o inactivo (isActive).

### 2.3.2 Vista - Diseño de Interfaces

Tomando como base los casos de uso definidos en la sección 2.2.5, en las Figuras 2.4 a las 2.14 se presentan las interfaces diseñadas, para lo cual se ha empleado Lucidchart.

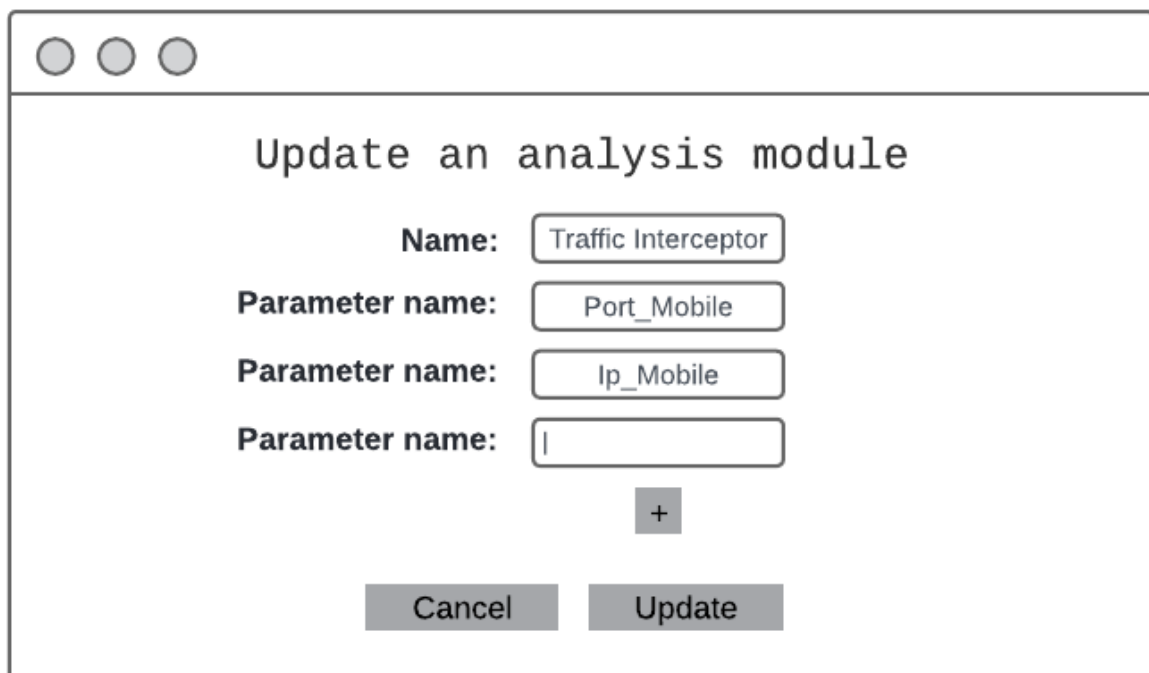


**Figura 2.4** Maquetado de la interfaz de visualización de módulos de análisis



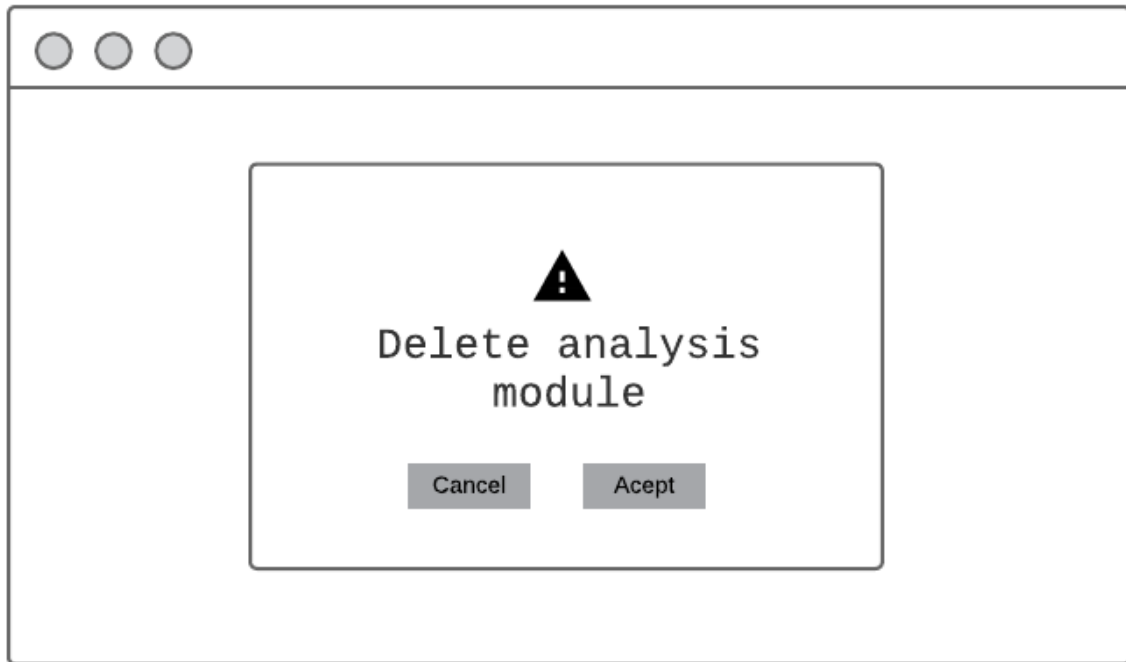
Maquetado de la interfaz de creación de un módulo de análisis. El formulario tiene un título "Create an analysis module" y cuatro campos de entrada etiquetados como "Name:", "Version:", "Parameter name:" y "Parameter name:". Debajo de los campos hay un botón "+". En la parte inferior hay dos botones: "Cancel" y "Create".

**Figura 2.5** Maquetado de la interfaz de creación de un módulo de análisis

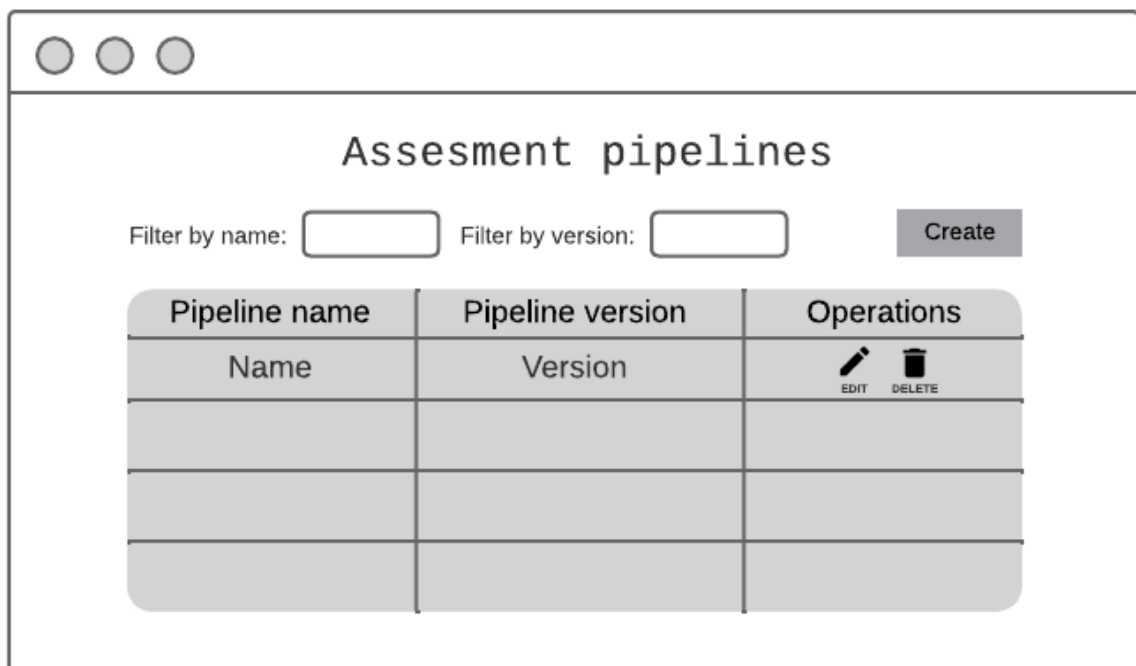


Maquetado de la interfaz de actualización de un módulo de análisis. El formulario tiene un título "Update an analysis module" y cuatro campos de entrada etiquetados como "Name:", "Parameter name:", "Parameter name:" y "Parameter name:". Los campos "Name:" y "Parameter name:" (primero) contienen "Traffic Interceptor", "Port\_Mobile" e "Ip\_Mobile" respectivamente. Debajo de los campos hay un botón "+". En la parte inferior hay dos botones: "Cancel" y "Update".

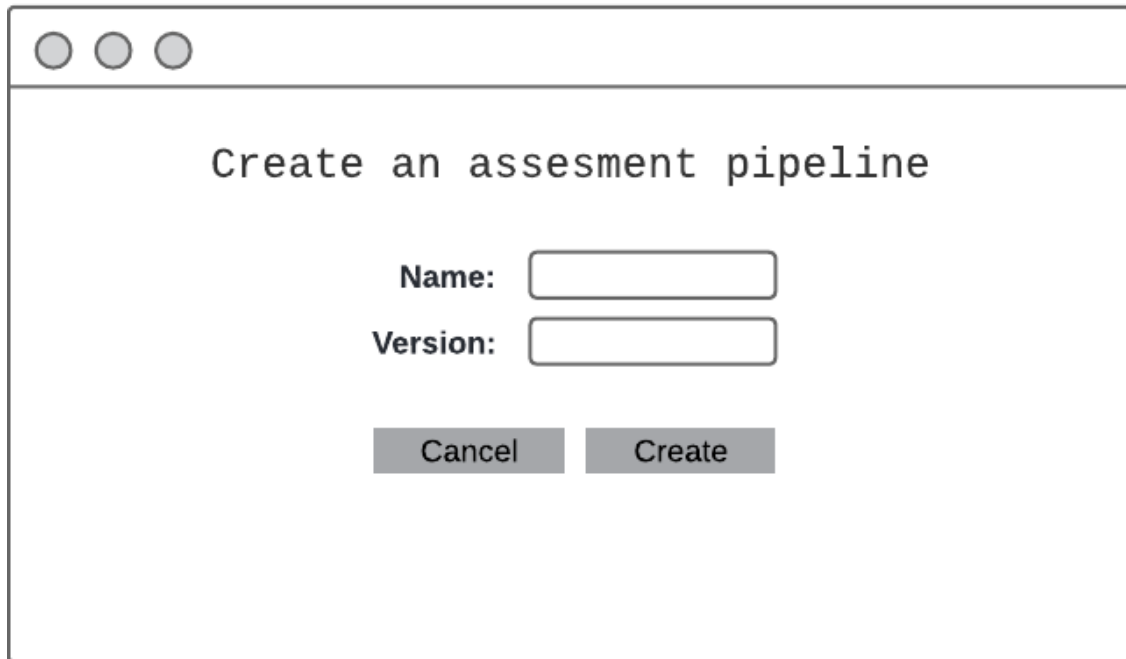
**Figura 2.6** Maquetado de la interfaz de actualización de un módulo de análisis



**Figura 2.7** Maquetado del modal de borrado de un módulo de análisis

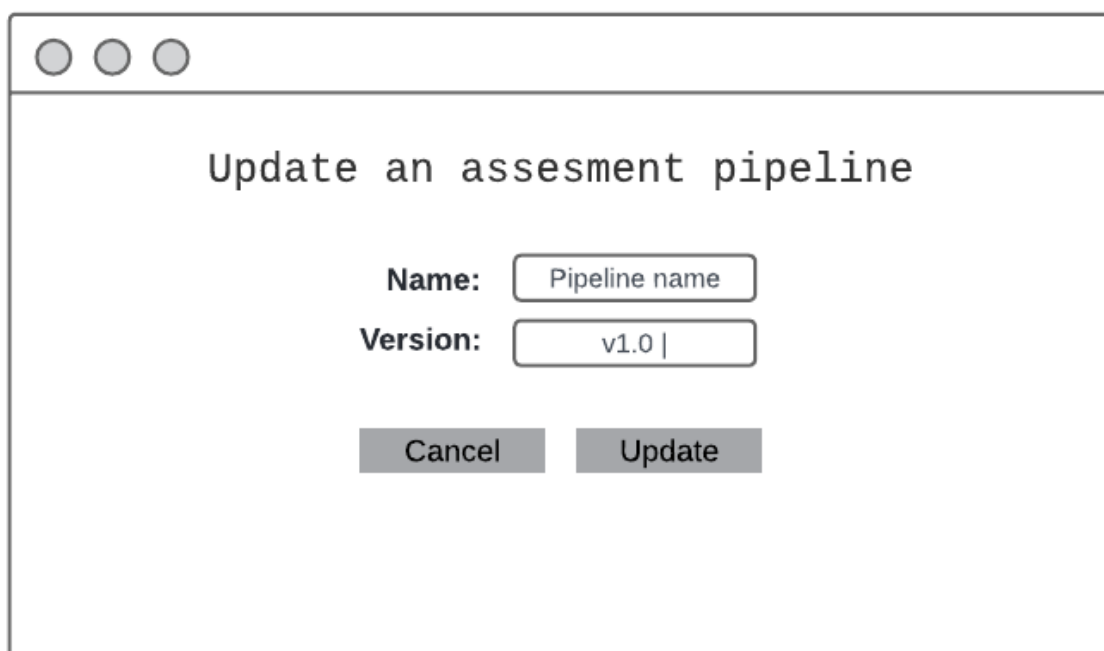


**Figura 2.8** Maquetado de la interfaz de visualización de los pipelines de evaluación



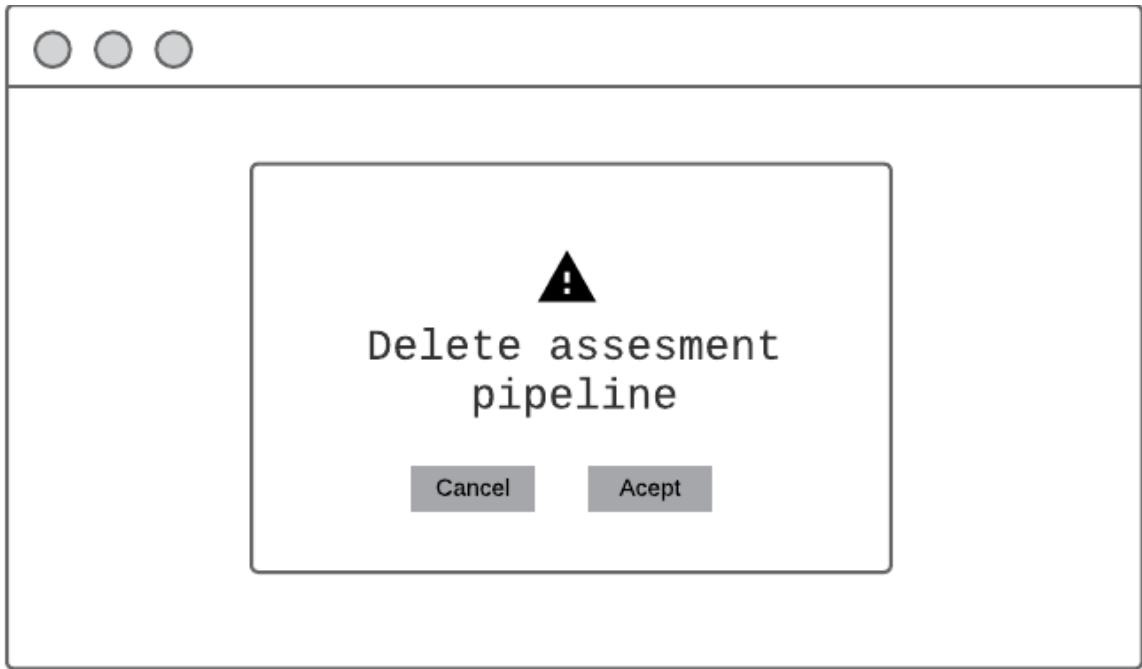
Maquetado de la interfaz de creación de un pipeline de evaluación. El formulario tiene un título "Create an assesment pipeline" y dos campos de entrada: "Name:" y "Version:". Debajo de los campos hay dos botones: "Cancel" y "Create".

**Figura 2.9** Maquetado de la interfaz de creación de un pipeline de evaluación

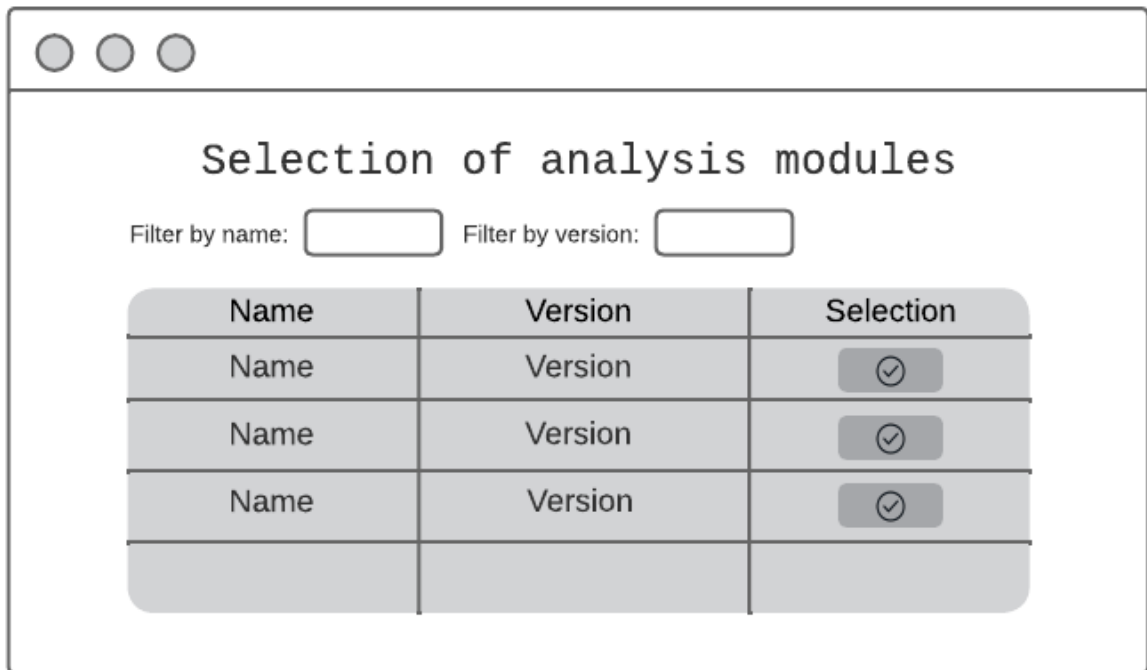


Maquetado de la interfaz de actualización de un pipeline de evaluación. El formulario tiene un título "Update an assesment pipeline" y dos campos de entrada: "Name:" con el valor "Pipeline name" y "Version:" con el valor "v1.0 |". Debajo de los campos hay dos botones: "Cancel" y "Update".

**Figura 2.10** Maquetado de la interfaz de actualización de un pipeline de evaluación



**Figura 2.11** Maquetado del modal de eliminación de un pipeline de evaluación



**Figura 2.12** Maquetado de la interfaz de visualización de módulos de análisis para su selección y posterior configuración

Setting up an analysis module

**Name:** Traffic Interceptor  
**Version:** 1.0

**Memory:**   
**Number of instances:**   
 .....   
**Port\_Mobile:**   
**Ip\_Mobile:**

**Figura 2.13** Maquetado de la interfaz de configuración de un módulo de análisis para ser agregado al pipeline de evaluación

Setting up an assesment pipeline

Filter by name:  Filter by version:

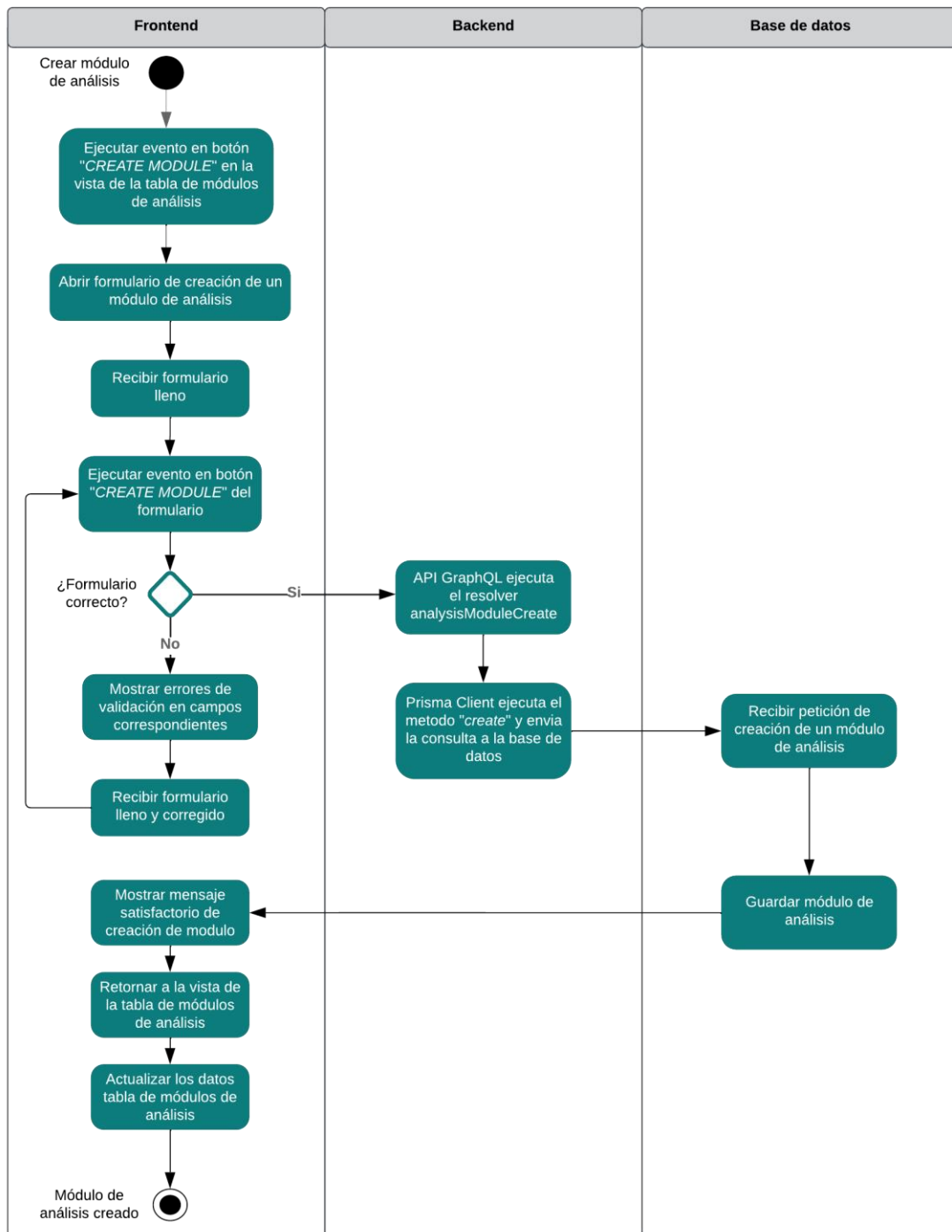
Name	Version	Operations
Name	Version	 EDIT DELETE
.....	.....	 EDIT DELETE

**Figura 2.14** Maquetado de la interfaz de configuración y ejecución de un pipeline de evaluación

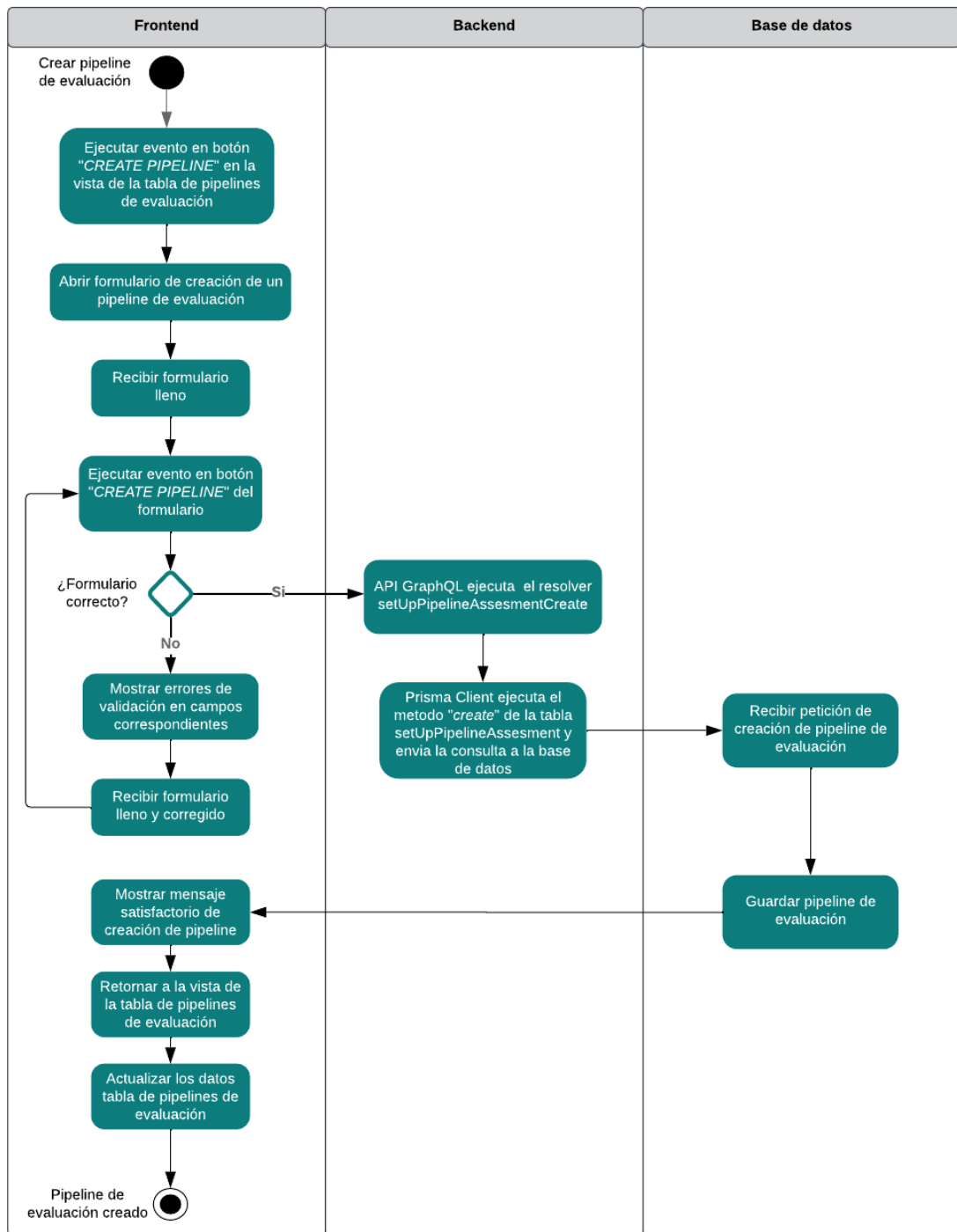


### **2.3.3 Controlador**

En esta sección se presenta los procesos llevados a cabo por el backend ante las peticiones realizadas por los clientes mediante las interfaces descritas en la sección anterior. Para modelar estos procesos se ha empleado los diagramas de actividades de UML. La Figura 2.15 muestra el diagrama de actividades para la creación de un módulo de análisis mientras que la Figura 2.16 muestra el diagrama de actividades para la creación de un pipeline de evaluación. Asimismo, la Figura 2.17, Figura 2.18 y la Figura 2.19 modelan los procesos para la configuración, ejecución y detención de un pipeline de evaluación, respectivamente. En todos estos diagramas se puede ver la interacción entre el backend, frontend y base de datos.



**Figura 2.15** Diagrama de actividades para la creación de un módulo de análisis



**Figura 2.16** Diagrama de actividades para la creación de un pipeline de evaluación

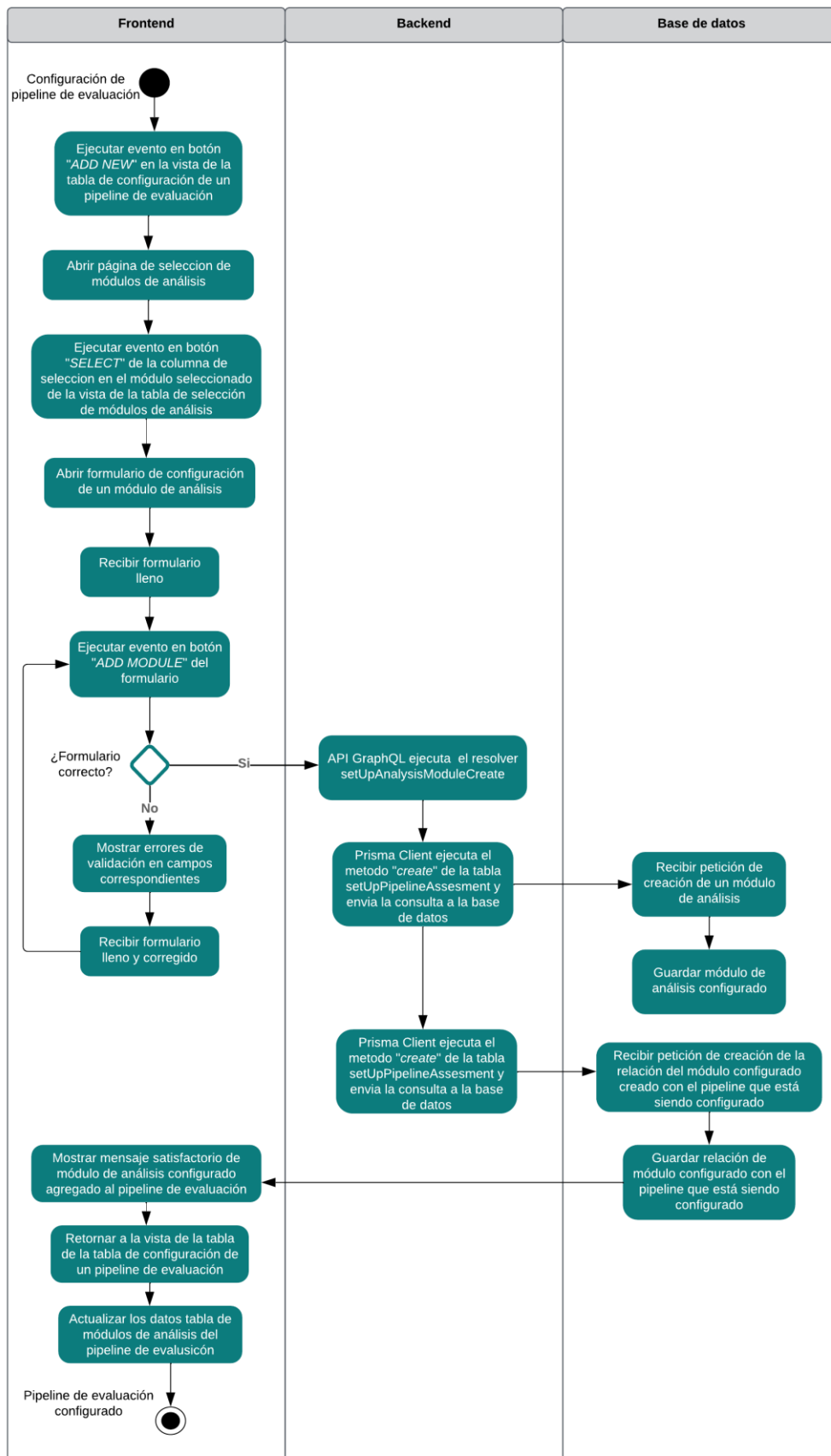


Figura 2.17 Diagrama de actividades para la configuración de un pipeline de evaluación

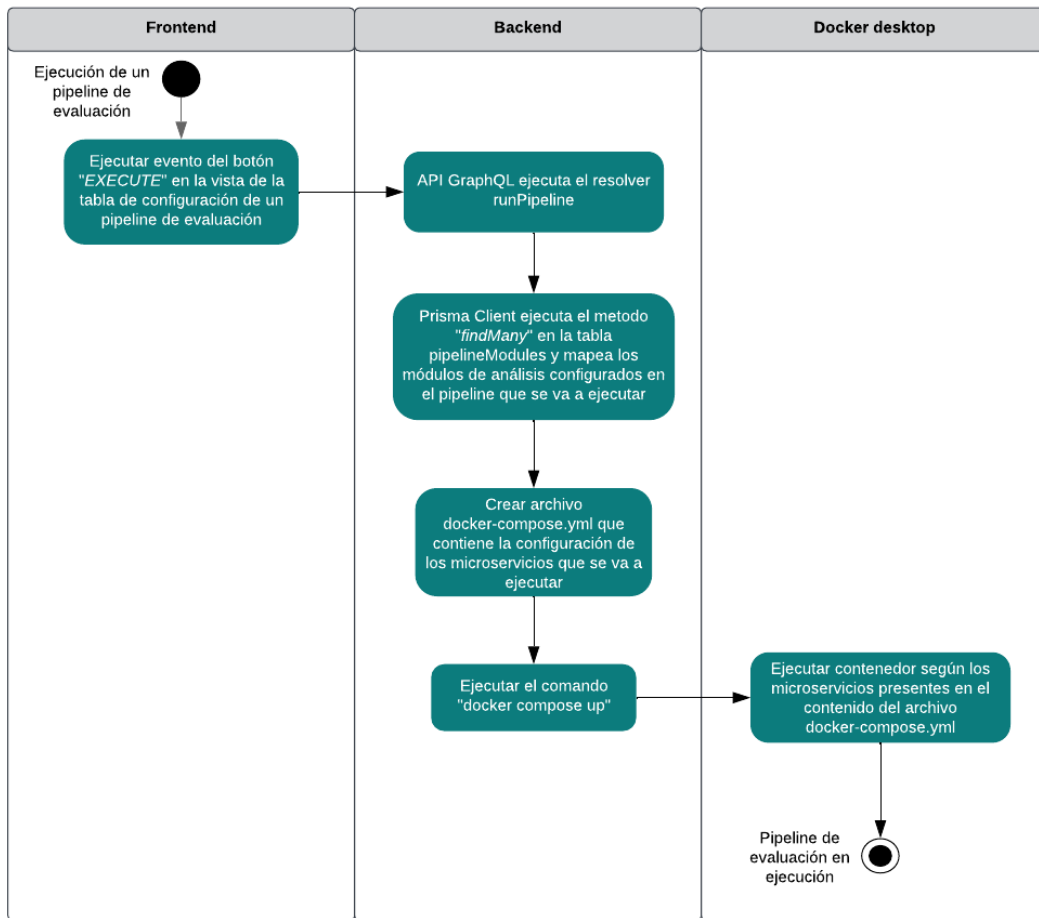


Figura 2.18 Diagrama de actividades para la ejecución de un pipeline de evaluación

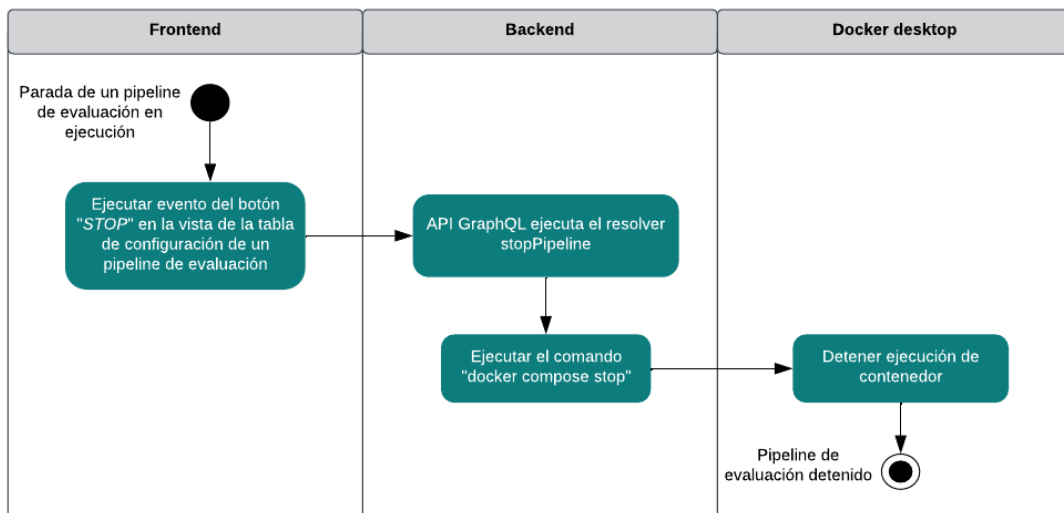


Figura 2.19 Diagrama de actividades que detiene el pipeline de evaluación en ejecución

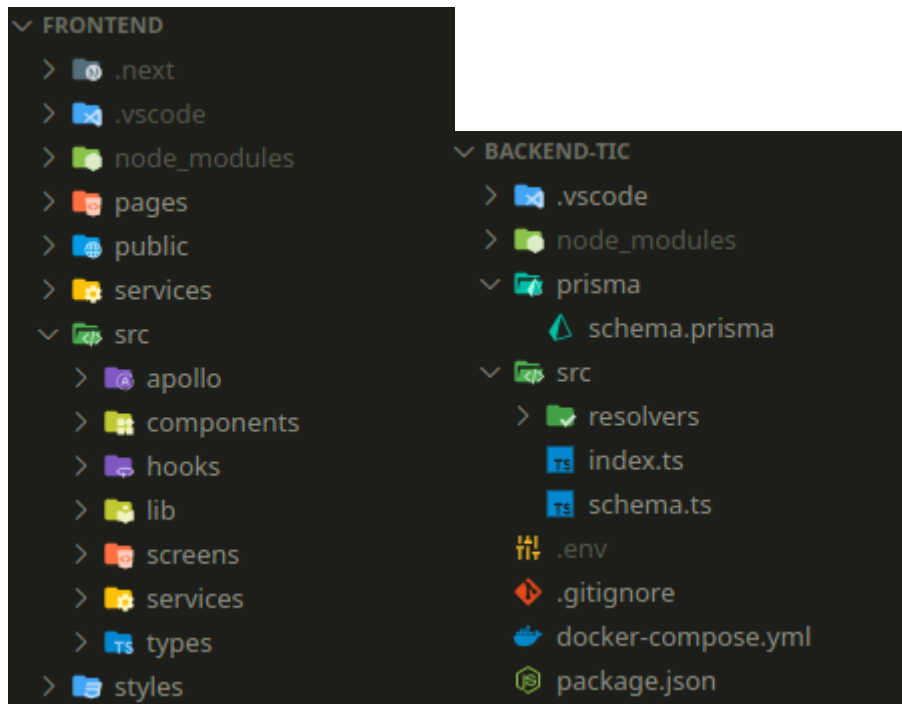
## 2.4 Implementación

En esta sección se resume el proceso efectuado para la implementación de los diseños presentados en la sección anterior. Se usaron los lenguajes de programación y tecnologías brevemente explicadas en la sección 2.4.3. La Tabla 2.21 detalla los lenguajes y tecnologías utilizadas.

**Tabla 2.21.** Tecnologías utilizadas en el desarrollo del prototipo

Componente	Lenguaje de programación	Versión	Framework/Biblioteca	Versión	IDE
Frontend	Typescript	4.9.3	React	18.2.0	VS Code
Backend (GraphQL)	Typescript	5.0.4	Apollo Server	3.12.0	VS Code
Backend (Prisma ORM)	Prisma Schema Language	4.14.1	Prisma client	4.14.1	VS Code

La estructura del código para el frontend (Vista) y para el backend (Modelo y Controlador) se presentan en la Figura 2.20. En el frontend el directorio `src` contiene el código fuente. El directorio `screens` contiene las pantallas necesarias para la Vista del prototipo, basadas en los diseños presentados en la sección 2.3.2. En el backend, el directorio `prisma` contiene la implementación del modelo de datos diseñado en la sección 2.3.1, mientras que `src` contiene los *resolvers* o controladores que implementan los procesos diseñados en la sección 2.3.2.



**Figura 2.20** Estructura del código del frontend y backend del prototipo implementado

## 2.4

### 2.4.1 Implementación primer y segundo sprint – Frontend del prototipo

Según lo indicado en la Tabla 2.17, durante el primer y segundo sprint se llevó a cabo el desarrollo de las interfaces de visualización, creación, actualización y eliminación de módulos de análisis, así como la de pipelines de evaluación. Para dar inicio, se procedió a crear la carpeta "screens" dentro del directorio "src". Dentro de esta carpeta, se organizaron los componentes destinados a la construcción de las interfaces tanto para los módulos de análisis como para los pipelines de evaluación. Cada uno de estos elementos se ubicó en su propia carpeta para una organización eficiente.

De esta manera, se comenzaron a construir los componentes esenciales que permitirían mostrar los módulos de análisis en formato tabular. Para lograr esto, se empleó el componente de interfaz proporcionado por MaterialUI, conocido como DataGrid. Este componente facilita la presentación de datos en un formato estructurado de filas y columnas. Como se muestra en la Figura 2.21, se detalla cómo se configuraron las columnas en el componente DataGrid. Este proceso es esencial para la construcción del componente de React encargado de mostrar la tabla de módulos de análisis.

```
1   ...
2
3   const columns: GridColumns = [
4     {
5       field: "id",
6       headerName: "ID",
7       hide: true,
8     },
9     {
10      field: "name",
11      headerName: "Module name",
12    },
13    {
14      field: "version",
15      headerName: "Module version",
16    },
17    {
18      field: "actions",
19      type: "actions",
20      headerName: "Operations",
21      ...
22    },
23  ];
24
25  ...
26
27  <Box
28  >
29    <DataGrid
30      rows={analysisModulesToShow}
31      columns={columns}
32      pageSize={5}
33      rowsPerPageOptions={[5]}
34      onClick={handleRowClick}
35    />
36  </Box>;
37
38  ...
```

**Figura 2.21** Fragmento de código que hace uso del componente Datagrid de MaterialUI



Fundamentalmente, contamos con una carpeta llamada "components" ubicada en el directorio "src". En esta carpeta, se encuentran componentes generales que resultan de gran utilidad en la creación de las interfaces que residen en la carpeta "screens". Uno de estos ejemplos es el componente `TextInputFormik`, el cual desempeña un papel esencial en los formularios utilizados en el prototipo para la introducción de datos, como se muestra en la Figura 2.22

```
1 import React, { memo } from "react";
2 import TextField from "@mui/material/TextField";
3 import { useField } from "formik";
4
5 interface Props {
6   name: string;
7   label: string;
8   [x: string]: any;
9 }
10
11 const TextInputFormik = ({ label, ...props }: Props) => {
12   const [field, meta] = useField(props);
13   return (
14     <TextField
15       label={label}
16       fullWidth
17       sx={{ mt: "0.5rem", mb: "0.5rem" }}
18       variant="outlined"
19       error={meta.touched && Boolean(meta.error)}
20       helperText={meta.touched && meta.error}
21       InputLabelProps={{ shrink: field.value }}
22       {...field}
23       {...props}
24     />
25   );
26 };
27 export default memo(TextInputFormik);
28
```

**Figura 2.22** Fragmento de código que conforma el componente *TextInputFormik*

Dentro de cada una de las carpetas en el directorio "screens", encontramos una subcarpeta denominada "components". Estas subcarpetas contienen componentes esenciales que, en su mayoría, son de utilidad específica dentro de su propio contexto, aunque también pueden resultar útiles en otros contextos. Por ejemplo, tenemos el componente encargado de mostrar el formulario para la creación de un módulo de análisis, tal como se ilustra en la Figura 2.23. Es importante destacar que este formulario de creación es considerablemente similar al utilizado para crear un pipeline de evaluación.

```
1  ...
2
3  <Formik
4    initialValues={initialValuesAux}
5    validationSchema={analysisModulesValidationSchema}
6    onSubmit={handleSubmit}
7  >
8    ({ values, setFieldValue }: FormikProps<IAnalysisModule>) => {
9      return (
10       <Form>
11         <TextInputFormik name="name" label="Module name" />
12         <TextInputFormik name="version" label="Module version" />
13
14         <Parameters values={values} setFieldValue={setFieldValue} />
15
16         ...
17
18       <Box>
19         >
20           <Button
21             color="warning"
22             sx={{ mt: 1, mr: 1 }}
23             variant="contained"
24             onClick={() => {
25               router.push("/");
26             }}
27           >
28             CANCEL
29           </Button>
30           <Button
31             sx={{ mt: 1, mr: 1 }}
32             type="submit"
33             variant="contained"
34           >
35             {labelSubmitButton}
36           </Button>
37         </Box>
38       </Form>
39     );
40   }}
41 </Formik>
42
43  ...
```


**Figura 2.23** Fragmento de código con el uso de Formik para el formulario de creación un módulo de análisis

Cada carpeta dentro del directorio "screens" se encuentra organizada de forma meticulosa, con subcarpetas que poseen nombres descriptivos. Cada una de estas subcarpetas alberga componentes que, en última instancia, conforman todas las interfaces necesarias para el prototipo. En el Anexo A se adjunta el código fuente del frontend de la interfaz de este proyecto, mientras que el Anexo B proporciona las instrucciones para ejecutar localmente el frontend.

## 2.4.2 Implementación tercer y cuarto sprint – Backend del prototipo de interfaz

Como se indicó en la Tabla 2.17, durante esta fase de desarrollo se implementó la API en GraphQL, la cual permite la obtención, creación, actualización y eliminación de módulos de análisis, así como de pipelines de evaluación, incluyendo su configuración y ejecución.

Para comenzar con el modelo de datos, se creó un nuevo proyecto de Prisma dentro del directorio del backend. El comando "init" de Prisma genera el directorio "prisma", que contiene el archivo básico "schema.prisma". Este archivo principalmente contiene la definición del generador y el proveedor de la fuente de datos. El generador corresponde a "generator", cuyo proveedor es "prisma-client-js", el cual básicamente genera el Prisma Client. Por otro lado, en "datasource" se indica el proveedor de la fuente de datos, que en este proyecto es "PostgreSQL". También se especifica la URL de conexión con la base de datos, la cual se almacena como variable de entorno en el archivo ".env" del proyecto. La Figura 2.25 muestra este fragmento de código del archivo "schema.prisma"



```
1 generator client {
2   provider = "prisma-client-js"
3 }
4
5 datasource db {
6   provider = "postgresql"
7   url      = env("DATABASE_URL")
8 }
```

**Figura 2.24** Fragmento de código que define el *datasource* y *generator* del modelo de datos

Con base en el diseño del modelo de datos diseñado en la sección 2.3.1, se implementó la BDD mediante ORM (*Object Relational Mapping*) Prisma, cuyo esquema proporciona una manera intuitiva de modelar datos, se crea un esquema de base de datos el SGBD PostgreSQL basado en los modelos definidos. La Figura 2.25 se muestra un fragmento de código con uno de los modelos que creó la tabla en la base de datos.

```

1  model analysisModules {
2    id          String          @id
3    name        String
4    version     String
5    isActive    Boolean         @default(true)
6    moduleParameters moduleParameters[]
7  }

```

**Figura 2.25** Modelo *analysisModules*

Este modelo se encuentra en el archivo "schema.prisma", junto con los demás modelos que conformaron las distintas tablas de la base de datos, mostradas en el diagrama relacional de la Figura 2.3. Este modelo, "analysisModules", contiene las columnas necesarias según los datos que se almacenan en él: id, name, version e isActive, como se detallaron anteriormente. Además, hay un dato adicional: "moduleParameters". Esto indica que tiene una relación de uno a varios con la tabla "moduleParameters". En el modelo "parameters" ocurre lo mismo y también hay un modelo correspondiente a "moduleParameters", donde se efectuará la relación con los modelos "analysisModules" y "parameters".

El resto del código en el esquema de Prisma define cada uno de los modelos, como el de "analysisModules" que se mostró previamente en la Figura 2.25. Por ejemplo, también se tienen en la Figura 2.26 los modelos para los módulos de análisis configurados y los pipelines de evaluación, junto con el modelo para su relación. Esto se refleja en el diagrama relacional del modelo de datos que se presentó en la Figura 2.3.

```

1  model setUpAnalysisModules {
2    id          String          @id
3    name        String
4    version     String
5    isActive    Boolean         @default(true)
6    parameters  Json[]
7    pipelineModules pipelineModules[]
8  }
9
10 model setUpPipelineAssesment {
11  id          String          @id
12  name        String
13  version     String
14  isActive    Boolean         @default(true)
15  pipelineModules pipelineModules[]
16 }
17
18 model pipelineModules {
19  setUpPipeline      setUpPipelineAssesment @relation(fields: [setUpPipelineAssesmentId], references: [id])
20  setUpPipelineAssesmentId String
21  setUpModule        setUpAnalysisModules  @relation(fields: [setUpAnalysisModuleId], references: [id])
22  setUpAnalysisModuleId String
23
24  @@id([setUpPipelineAssesmentId, setUpAnalysisModuleId])
25 }

```

**Figura 2.26** Fragmento de código que define los modelos *setUpAnalysisModules*, *setUpPipelineAssesment* y *pipelineModules*

Para el API en GraphQL, se ha definido un documento (schema.ts) de lenguaje de esquema que describe la estructura de datos. Para lograr esto, se importó "gql" de "apollo-server", lo que permitió establecer el tipado conforme al modelo de datos. Además, en este documento se han definido los dos tipos principales de operaciones: queries y mutations, que posibilitan la interacción con el servidor para solicitar y modificar datos.

Las queries habilitan al cliente a especificar qué datos necesita consultar y en qué forma los desea, evitando así la sobrecarga de datos. Por otro lado, las mutaciones modifican los datos al especificar la información a enviar. Esto se utiliza para crear, actualizar o eliminar datos. Además, las mutaciones también pueden retornar datos específicos según lo que se requiera.

Las queries implementadas tienen la función de consultar datos de diversos elementos: un módulo de análisis, varios módulos de análisis, los parámetros existentes, un módulo de análisis configurado, varios módulos de análisis configurados, un pipeline de evaluación y varios pipelines de evaluación. En el caso de las queries que solicitan datos de un elemento específico, es necesario proporcionar un parámetro concreto, en este caso, un identificador.

A continuación, en las Figuras 2.27 y 2.28 se presentan los fragmentos de código que establecen el esquema para las queries y mutaciones. Las mutaciones que se muestran abordan la creación, actualización y eliminación de distintos elementos, tales como un módulo de análisis, un módulo de análisis configurado y un pipeline de evaluación.

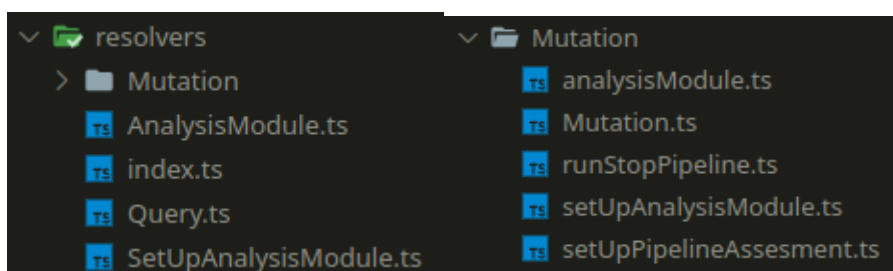
```
1 type Query {
2   analysisModules: [AnalysisModule]
3   analysisModule(analysisModuleId: ID!): AnalysisModule
4   parameters: [Parameter]
5   setUpAnalysisModules: [SetUpAnalysisModule]
6   setUpAnalysisModule(analysisModuleId: ID!): SetUpAnalysisModule
7   setUpPipelinesAssesment: [SetUpPipelineAssesment]
8   setUpPipelineAssesment(pipelineAssesmentId: ID!): SetUpPipelineAssesment
9 }
```

**Figura 2.27** Fragmento de código que define las *queries*

```
1  type Mutation {
2    analysisModuleCreate(input: CreateAnalysisModuleInput!): Boolean
3    analysisModuleUpdate(input: UpdateAnalysisModuleInput!): Boolean
4    analysisModuleDelete(id: ID!): Boolean
5    setUpAnalysisModuleCreate(input: CreateSetUpAnalysisModuleInput!): Boolean
6    setUpAnalysisModuleUpdate(input: UpdateSetUpAnalysisModuleInput!): Boolean
7    setUpAnalysisModuleDelete(id: ID!): Boolean
8    setUpPipelineAssesmentCreate(
9      input: CreateSetUpPipelineAssesmentInput!
10   ): Boolean
11   setUpPipelineAssesmentUpdate(
12     input: UpdateSetUpPipelineAssesmentInput!
13   ): Boolean
14   setUpPipelineAssesmentDelete(id: ID!): Boolean
15   runPipeline(id: ID!): Boolean
16   stopPipeline: Boolean
17 }
```

**Figura 2.28** Fragmento de código que define las *mutations*

Para controlar cómo se ejecutan las queries y mutations, se han implementado resolvers, los cuales efectivamente gestionan cada una de las solicitudes que estas queries y mutations requieren. La estructura de la carpeta de resolvers, que se muestra en la Figura 2.29, contiene el archivo "Query.ts". Este archivo contiene los resolvers para las queries. De manera similar, las mutaciones están ubicadas en la carpeta "Mutation", donde se dividen en distintos archivos. Esto permite mantener organizados los resolvers para los módulos de análisis, los módulos de análisis configurados y los pipelines de evaluación.



**Figura 2.29** Estructura de la carpeta *resolvers*

Los resolvers para las queries tienen una naturaleza relativamente sencilla. En términos generales, un resolver es una función que acepta tres parámetros: parent, args y context. "Parent" (padre) hace referencia al resultado que el resolver del campo superior devuelve, y contribuye a resolver un campo dentro de un objeto anidado basándose en dicho resultado. "Args" hace referencia a los argumentos de entrada en la consulta o mutación. "Context" es un espacio compartido por todos los resolvers que almacena información relevante para ellos. En este caso particular, el contexto consiste en el acceso a la base de datos a través del cliente de Prisma.

A continuación, en la Figura 2.30, se presentan los resolvers para las queries. En esta representación, se observa cómo todos comparten el mismo contexto, el de "prisma". Cada uno de estos resolvers accede a las tablas de la base de datos utilizando este contexto, lo que les permite realizar consultas mediante los métodos "findMany" y "findUnique".

```
1  {
2    analysisModules: (_, __: any, { prisma }: Context) => {
3      return prisma.analysisModules.findMany({ orderBy: [{ name: "asc" }] });
4    },
5    analysisModule: (
6      _: any,
7      { analysisModuleId }: { analysisModuleId: string },
8      { prisma }: Context
9    ) => {
10     return prisma.analysisModules.findUnique({
11       where: { id: analysisModuleId },
12     });
13   },
14   parameters: (_, __: any, { prisma }: Context) => {
15     return prisma.parameters.findMany({ orderBy: [{ name: "asc" }] });
16   },
17   setUpAnalysisModules: (_, __: any, { prisma }: Context) => {
18     return prisma.setUpAnalysisModules.findMany({ orderBy: [{ name: "asc" }] });
19   },
20   setUpAnalysisModule: (
21     _: any,
22     { analysisModuleId }: { analysisModuleId: string },
23     { prisma }: Context
24   ) => {
25     return prisma.setUpAnalysisModules.findUnique({
26       where: { id: analysisModuleId },
27     });
28   },
29   setUpPipelineAssesment: (_, __: any, { prisma }: Context) => {
30     return prisma.setUpPipelineAssesment.findMany({
31       orderBy: [{ name: "asc" }],
32     });
33   },
34   setUpPipelineAssesment: (
35     _: any,
36     { pipelineAssesmentId }: { pipelineAssesmentId: string },
37     { prisma }: Context
38   ) => {
39     return prisma.setUpPipelineAssesment.findUnique({
40       where: { id: pipelineAssesmentId },
41     });
42   },
43 }
```

**Figura 2.30** Fragmento de código que contiene la lógica de los *resolvers* de las *queries*

Las mutaciones implican una lógica más compleja, ya que conllevan el manejo de los datos de entrada. Por ejemplo, en el caso de la actualización de módulos de análisis, es necesario

gestionar los parámetros que han sufrido cambios. Esto implica comparar los parámetros existentes en el módulo, identificar los nuevos parámetros a agregar y eliminar los antiguos para sustituirlos por otros. Además, esto implica analizar la relación entre las tablas de módulos de análisis y la de parámetros. Es necesario invalidar la relación que se está eliminando, y todo esto requiere una evaluación precisa de la información almacenada en la base de datos.

En estas mutaciones, se hacen uso de una variedad más amplia de métodos de Prisma, tales como "create", "update", "delete", "findFirst", y los ya mencionados "findMany" y "findUnique". Cada uno de estos métodos trabaja en conjunto dentro de los resolvers para finalmente lograr el propósito específico de cada mutación.

Por ejemplo, en la Figura 2.31 se representa la lógica que implica la actualización de un módulo de análisis. En este esquema, es posible observar cómo se lleva a cabo una serie de validaciones y pasos. En primer lugar, se verifica inicialmente si los argumentos de entrada contienen el valor del ID que identifica al módulo que se desea actualizar. A continuación, se comprueba la existencia del módulo de análisis que se pretende actualizar. Posteriormente, se procede a actualizar el módulo en caso de que haya cambios en el nombre o la versión.

El proceso continúa con un análisis de los parámetros ingresados en los argumentos. Si es necesario actualizar los parámetros, se buscan en la tabla de relación que conecta los módulos con los parámetros. Aquí, se añaden los nuevos parámetros y se elimina la relación con los parámetros que ya no están vinculados al módulo. Finalmente, se retorna el valor booleano "true", indicando que todo el proceso se ha ejecutado correctamente.

De manera similar, los otros resolvers para las mutaciones también presentan su propio nivel de complejidad en sus algoritmos. En el Anexo A se incluye el código fuente del backend correspondiente al prototipo de la interfaz desarrollada en este trabajo. Además, en el Anexo C se proporcionan las instrucciones para llevar a cabo la ejecución local del backend.

### **2.4.3 Implementación quinto sprint – Integración del backend con el frontend - Implementación de ejecución local de un pipeline de evaluación con Docker.**

Esta etapa representa la fase final del desarrollo, conforme a lo indicado en la Tabla 2.17. En esta fase, se implementó la capacidad de ejecutar y detener un pipeline de evaluación. Esto incluyó la integración entre el backend y el frontend, dado que el frontend se desarrolló



inicialmente utilizando pruebas con un servidor JSON temporal denominado "db.json", que facilitó la realización de pruebas CRUD con una base de datos ideal.

Para comenzar, es importante destacar que en el sprint previo se creó el archivo principal que forma la base. En este archivo, se instaura la instancia del servidor a través de Apollo Server (ver Figura 2.32). Esta instancia engloba todo lo definido en los "typeDefs", los "resolvers" y el contexto de Prisma. Todos estos elementos son esenciales para garantizar el funcionamiento adecuado del servidor.

```
1  analysisModuleUpdate: async (
2    _: any,
3    { input }: analysisModuleArgs,
4    { prisma }: Context
5  ): Promise<Boolean> => {
6    const { id, name, version, parameters } = input;
7    if (!id) {
8      return false;
9    }
10
11   const existingAnalysisModule = await prisma.analysisModules.findUnique({
12     where: { id },
13   });
14   if (!existingAnalysisModule) {
15     return false;
16   }
17
18   let analysisModuleToUpdate = { id, name, version };
19
20   await prisma.analysisModules.update({
21     data: {
22       ...analysisModuleToUpdate,
23     },
24     where: { id },
25   });
26
27   const parametersInModule = await prisma.moduleParameters.findMany({
28     where: { analysisModuleId: id },
29   });
30
31   parameters?.forEach(async (parameter) => {
32     const parameterFound = await prisma.parameters.findFirst({
33       where: {
34         name: parameter,
35       },
36     });
37     if (!parameterFound) {
38       const newParameterId = uuid();
39       await prisma.parameters.create({
40         data: { id: newParameterId, name: parameter },
41       });
42       await prisma.moduleParameters.create({
43         data: { analysisModuleId: id, parameterId: newParameterId },
44       });
45     }
46     if (
47       parameterFound &&
48       !parametersInModule.find(
49         (item) => item.parameterId === parameterFound.id
50       )
51     ) {
52       await prisma.moduleParameters.create({
53         data: { analysisModuleId: id, parameterId: parameterFound.id },
54       });
55     }
56   });
57
58   parametersInModule.forEach(async (item) => {
59     const parameter = await prisma.parameters.findFirst({
60       where: {
61         id: item.parameterId,
62       },
63     });
64     if (!parameters?.find((p) => p === parameter?.name)) {
65       await prisma.moduleParameters.delete({
66         where: {
67           analysisModuleId_parameterId: {
68             analysisModuleId: id,
69             parameterId: parameter!.id,
70           },
71         },
72       });
73     }
74   });
75
76   return true;
77 }
```

**Figura 2.31** Fragmento de código que contiene la lógica del resolver de la mutación para actualizar un módulo de análisis

```
1 import { ApolloServer } from "apollo-server";
2 import { typeDefs } from "./schema";
3 import { Prisma, PrismaClient } from "@prisma/client";
4 import { Mutation, Query, AnalysisModule } from "./resolvers";
5 import { SetUpAnalysisModule } from "./resolvers/SetUpAnalysisModule";
6
7 const prisma = new PrismaClient();
8
9 export interface Context {
10   prisma: PrismaClient<
11     Prisma.PrismaClientOptions,
12     never,
13     Prisma.RejectOnNotFound | Prisma.RejectPerOperation | undefined
14   >;
15 }
16
17 const server = new ApolloServer({
18   typeDefs,
19   resolvers: {
20     Query,
21     Mutation,
22     AnalysisModule,
23     SetUpAnalysisModule,
24   },
25   context: {
26     prisma,
27   },
28 });
29
30 server.listen().then(({ url }) => {
31   console.log(`Server ready on ${url}`);
32 });
```

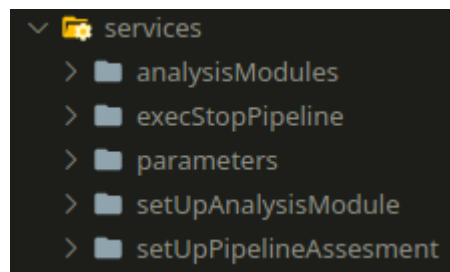
**Figura 2.32** Fragmento de código en el backend que genera la instancia del servidor en *Apollo*

Para llevar a cabo la integración entre el backend y el frontend, en el frontend se estableció una carpeta denominada "apollo". En esta carpeta, se configuró el cliente Apollo, creando una nueva instancia de Apollo Client tal como se ilustra en el fragmento de código presentado en la Figura 2.33. Este cliente Apollo Client se pasa a través de las props del ApolloProvider en el componente principal de la aplicación. Con este enfoque, todo el conjunto de la aplicación emplea el cliente de Apollo para realizar consultas y mutaciones al backend.

Posteriormente, se establecieron los servicios que consumen las queries y mutaciones del API del backend. La estructura de la carpeta "services" se expone en la Figura 2.34. Esta estructura organiza las carpetas de manera que separa las funcionalidades requeridas por el prototipo de interfaz en unidades individuales.

```
1 import { ApolloClient, InMemoryCache } from "@apollo/client";
2
3 const client = new ApolloClient({
4   uri: "http://localhost:4000/",
5   cache: new InMemoryCache(),
6 });
7
8 export default client;
```

**Figura 2.33** Fragmento de código en el frontend con la instancia que contiene al cliente de *Apollo*



**Figura 2.34** Estructura de la carpeta *services*

En las Figuras 2.35 y 2.36, se muestra cómo se consume las queries y mutaciones de la sección de módulos de análisis que ofrece la API, como se puede observar, se importa *gql* de *apollo/client*, y con este se definen cada uno de los servicios. Según la funcionalidad, las queries tiene la característica de que se puede establecer que parámetros se requieran, evitando la sobrecarga de datos consultados, en esta sección de módulos de análisis se tienen:

- GET\_ANALYSIS\_MODULES
- GET\_ANALYSIS\_MODULE
- UPDATE\_ANALYSIS\_MODULE
- CREATE\_ANALYSIS\_MODULE
- DELETE\_ANALYSIS\_MODULE

```
1 import { gql } from "@apollo/client";
2
3 export const GET_ANALYSIS_MODULES = gql`
4   query AnalysisModules {
5     analysisModules {
6       id
7       isActive
8       name
9       version
10      parameters {
11        id
12        name
13      }
14    }
15  }
16 `;
17
18 export const GET_ANALYSIS_MODULE = gql`
19   query AnalysisModule($analysisModuleId: ID!) {
20     analysisModule(analysisModuleId: $analysisModuleId) {
21       id
22       name
23       version
24       parameters {
25         id
26         name
27       }
28       isActive
29     }
30   }
31 `;
```

**Figura 2.35** Fragmento de código con los servicios para las queries de módulos de análisis

```
1 import { gql } from "@apollo/client";
2
3 export const UPDATE_ANALYSIS_MODULE = gql`
4   mutation AnalysisModuleUpdate($input: UpdateAnalysisModuleInput!) {
5     analysisModuleUpdate(input: $input)
6   }
7 `;
8
9 export const CREATE_ANALYSIS_MODULE = gql`
10  mutation AnalysisModuleCreate($input: CreateAnalysisModuleInput!) {
11    analysisModuleCreate(input: $input)
12  }
13 `;
14
15 export const DELETE_ANALYSIS_MODULE = gql`
16  mutation AnalysisModuleDelete($analysisModuleDeleteId: ID!) {
17    analysisModuleDelete(id: $analysisModuleDeleteId)
18  }
19 `;
```

**Figura 2.36** Fragmento de código con los servicios para las mutaciones de módulos de análisis

Para hacer uso de los servicios en los distintos componentes del frontend, se usaron los *hooks useQuery* y *useMutation* que son parte de la librería de *Apollo Client*, respectivamente estos efectúan las consultas y mutaciones de los datos. En la Figura 2.37 se muestra un ejemplo del uso de estos hooks en el ámbito de módulos de análisis.

```
1
2 const { data, error, loading } = useQuery(GET_ANALYSIS_MODULES, {
3   pollInterval: 500,
4 });
5
6 const [
7   AnalysisModuleCreate,
8   { data: dataCreated, loading: loadingCreation, error: errorCreating },
9 ] = useMutation(CREATE_ANALYSIS_MODULE);
```

**Figura 2.37** Fragmento de código con el ejemplo de uso de los *hooks useQuery* y *useMutation* de apollo

En este ejemplo está la mutación para crear un módulo, donde requiere datos de entrada para su creación, ingresando como parámetro un objeto con las variables que requieran, en este caso es un *input* con todos los datos para la creación del módulo, esto se muestra en el fragmento de código de la Figura 2.38.

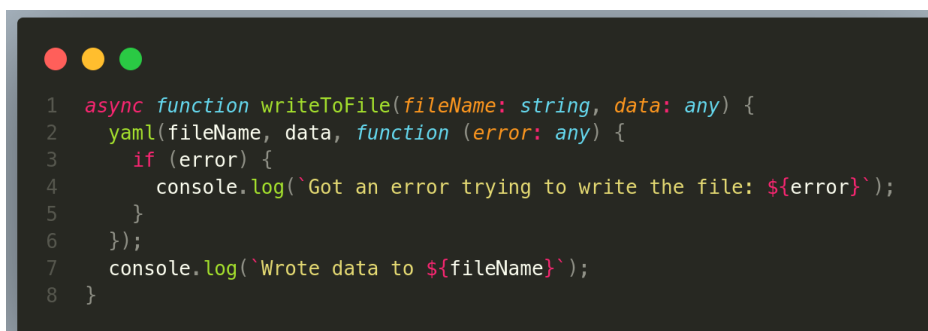
```
1
2 const analisisModuleToCreate: IUpdateCreateAnalysisModule = {
3   id: uuid(),
4   name,
5   version,
6   parameters: parametersToUpdateCreate,
7 };
8
9 AnalysisModuleCreate({
10   variables: { input: analisisModuleToCreate },
11 });
```

**Figura 2.38** Fragmento de código con el ejemplo de creación de un módulo de análisis

Este proceso se realizó en cada uno de los componentes en los que se requiere la consulta y mutación de datos.

En la etapa final, se procedió a implementar en el backend las mutaciones que posibilitan la ejecución y detención de pipelines de evaluación. Para llevar esto a cabo, se hizo uso de la biblioteca "write-yaml-file". Esta biblioteca permitió crear un archivo YAML, mediante la implementación de una función específica que fue empleada para generar el archivo definitivo, denominado "docker-compose.yml". Dicho archivo alberga la configuración de

los microservicios que se ejecutarán utilizando Docker. El esquema de esta función se expone en la Figura 2.39.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in TypeScript and shows an asynchronous function named `writeToFile`. The function takes `fileName: string` and `data: any` as arguments. It uses the `yaml` library to write data to a file. The code includes an error handling block that logs a message if an error occurs, and a success log message after the data is written. The lines are numbered from 1 to 8.

```
1  async function writeToFile(fileName: string, data: any) {
2    yaml(fileName, data, function (error: any) {
3      if (error) {
4        console.log(`Got an error trying to write the file: ${error}`);
5      }
6    });
7    console.log(`Wrote data to ${fileName}`);
8  }
```

**Figura 2.39** Fragmento de código que muestra la función *writeToFile*

El resolver destinado a la mutación de ejecución de un pipeline de evaluación (ver Figuras 2.40 y 2.41) inicia con la búsqueda del pipeline de evaluación utilizando el contexto de Prisma y su método "findUnique". Esta búsqueda se basa en el ID proporcionado en los argumentos del resolver. A continuación, se procede a buscar los módulos de análisis configurados dentro de ese pipeline. Para lograr esto, se obtienen los IDs correspondientes buscando en la tabla de relación "pipelineModules", que conecta los módulos configurados con los pipelines. Esto se realiza empleando el método "findMany" de Prisma.

Utilizando los IDs obtenidos, se mapea la tabla de "módulos de análisis configurados" y se filtran los módulos que coincidan con cada uno de los IDs. De esta manera, se logra obtener toda la información relativa a cada módulo de análisis configurado. Para llevar a cabo esta obtención, se recurre al método "findFirst" de Prisma, que permite recuperar los datos de un elemento específico.

```

1  runPipeline: async (_, any, { id }: { id: string }, { prisma }: Context) => {
2    const setUpPipelineAux = await prisma.setUpPipelineAssesment.findUnique({
3      where: { id },
4    });
5    const setUpAnalysisModulesInPipeline =
6      await prisma.pipelineModules.findMany({
7        where: { setUpPipelineAssesmentId: id },
8      });
9
10   const modules = async (item: pipelineModules) => {
11     return await prisma.setUpAnalysisModules.findFirst({
12       where: { id: item.setUpAnalysisModuleId },
13     });
14   };
15
16   const setUpAnalysisModules = await Promise.all(
17     setUpAnalysisModulesInPipeline.map(modules)
18   );
19
20   const modulesAux = setUpAnalysisModules
21     .map((item) => {
22       const parametersAux = item?.parameters!;
23
24       return {
25         name: item?.name!,
26         version: item?.version!,
27         parameters: parametersAux!,
28         isActive: item?.isActive,
29       };
30     })
31     .filter((item) => item.isActive);

```

**Figura 2.40** Fragmento de código con el resolver para ejecutar un pipeline de evaluación



```
1
2  const services: any = {};
3  modulesAux?.forEach((mod) => {
4    const parameters: any = {};
5    mod.parameters?.forEach((param) => {
6      const paramAuxJson = JSON.stringify(param);
7      const paramAux = JSON.parse(paramAuxJson);
8      const paramValue = paramAux.value as string;
9      const value = paramValue.split(" ");
10     switch (paramAux.name) {
11       case "ports":
12       case "volumes":
13       case "devices":
14       case "networks":
15         parameters[paramAux.name] = [...value];
16         break;
17
18       case "network_mode":
19         const valueAux = '"' + paramValue.toString() + '"';
20         console.log(valueAux);
21         parameters[paramAux.name] = valueAux;
22         break;
23
24       default:
25         parameters[paramAux.name] = paramAux.value;
26         break;
27     }
28   });
29
30   services[mod.name!] = parameters;
31 });
32
33 const data = {
34   version: setUpPipelineAux?.version || "1",
35   services: services,
36 };
37
38 await writeFile("docker-compose.yml", data);
39
40 await exec("docker compose up", (error, stdout, stderr) => {
41   if (error) {
42     console.error(`exec error: ${error}`);
43     return;
44   }
45   console.log(`stdout: ${stdout}`);
46   console.error(`stderr: ${stderr}`);
47 });
48 return true;
49 },
```

**Figura 2.41** Fragmento de código con el resolver para ejecutar un pipeline continuación

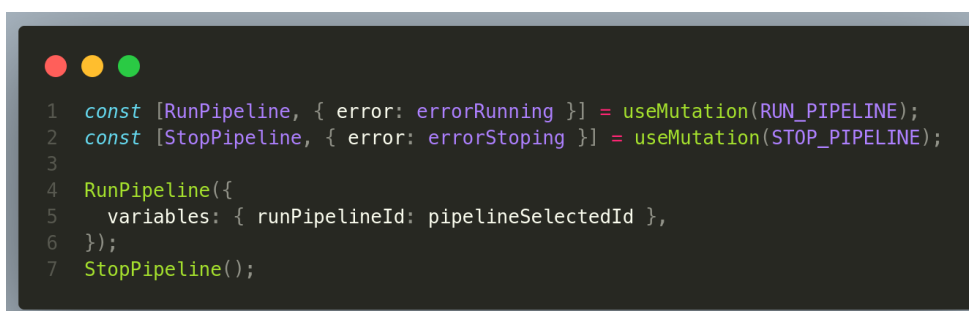
De entre los módulos obtenidos, se lleva a cabo un filtrado para seleccionar únicamente aquellos que están activos. El resolver sigue formateando los datos de manera que construye un objeto que contiene los servicios correspondientes a cada uno de los módulos

de análisis. A continuación, se establece un objeto con los datos necesarios para la creación del archivo YAML, "docker-compose.yml", utilizando la función "writeToFile" presentada en la Figura 2.39.

Para dar continuidad, el resolver ejecuta el comando "docker compose up", que inicia la ejecución del contenedor con los microservicios que componen el pipeline de evaluación, utilizando el archivo previamente creado, "docker-compose.yml". Finalmente, el resolver concluye al devolver el valor booleano "true", indicando que todos los pasos anteriores han sido ejecutados de manera exitosa.

En el caso del resolver que detiene la ejecución del contenedor de Docker, se emplea el comando "docker compose stop", que detiene inmediatamente todos los microservicios en ejecución dentro del contenedor.

En el frontend, en el componente correspondiente a la ejecución del pipeline de evaluación, se implementaron las mutaciones tanto para la ejecución como para la detención del pipeline. Para lograr esto, como ya se ha mencionado previamente, se empleó el hook "useMutation", en conjunto con los servicios correspondientes. Este proceso se ilustra en la Figura 2.42.



```
1  const [RunPipeline, { error: errorRunning }] = useMutation(RUN_PIPELINE);
2  const [StopPipeline, { error: errorStopping }] = useMutation(STOP_PIPELINE);
3
4  RunPipeline({
5    variables: { runPipelineId: pipelineSelectedId },
6  });
7  StopPipeline();
```

**Figura 2.42** Fragmento de código usado en el frontend para llamar a las mutaciones de ejecución y parada de un pipeline de evaluación

El código fuente, tanto de frontend como de backend se encuentran en el ANEXO A.

## 2.5 Validación y verificación

Con el propósito de ofrecer evidencia que respalde el funcionamiento del prototipo de interfaz, esta sección presenta un conjunto de pruebas funcionales que validan cada una de las historias de usuario mencionadas en la sección 2.2.3, de acuerdo con los casos de uso detallados en la sección 2.2.5.

En estas pruebas preliminares, se emplea la imagen del contenedor "nginx" para evaluar la operatividad de la ejecución de contenedores utilizando Docker. Posteriormente, se procede a llevar a cabo pruebas de ejecución utilizando el módulo de tráfico de CLIIP. Estas pruebas tienen como objetivo demostrar la funcionalidad y efectividad del prototipo, respaldando así su desempeño y utilidad.

## **2.5**

### **2.5.1 Pruebas – Módulos de análisis**

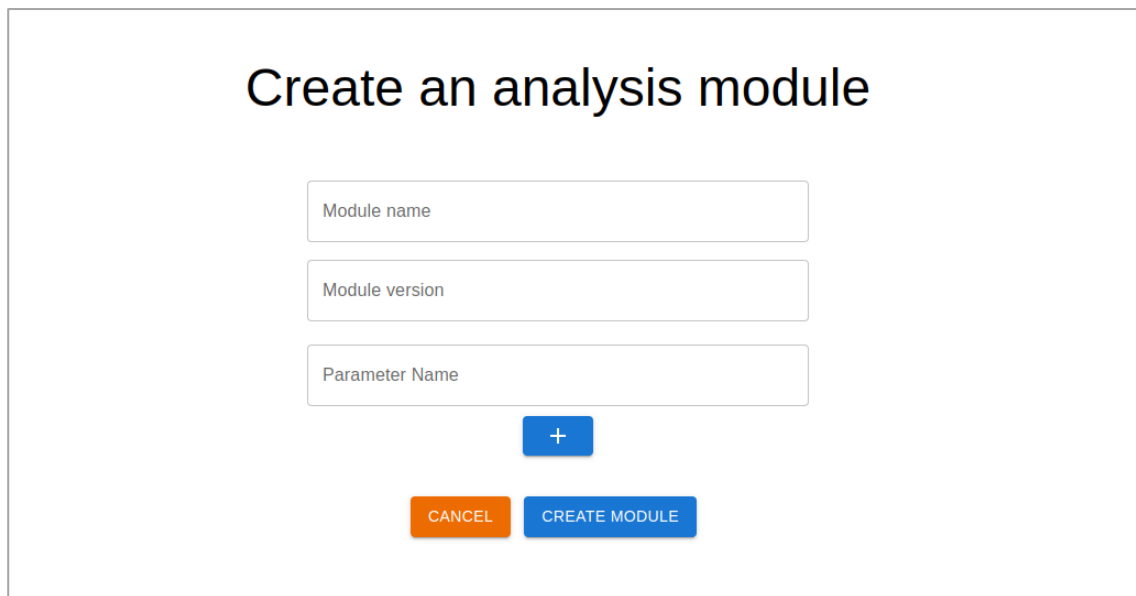
Se realizaron pruebas de las historias respecto a los módulos de análisis, estas corresponden a las historias HU-01, HU-02, HU-03 y HU-04, los criterios de aceptación para estas historias se derivan de los casos de uso CU-01, CU-02, CU-03 y CU-04.

#### **2.5.1.1 Criterios de aceptación - Creación de un módulo de análisis - HU-01**

1. En el formulario de creación de un módulo de análisis, debe permitir ingresar los datos del nuevo módulo de análisis, como el nombre del módulo, la versión del módulo y los parámetros del módulo. Para los parámetros debe visualizarse un botón para agregar nuevo parámetro y poder ingresar el nombre del parámetro. Y debe visualizarse un botón para crear el módulo y un botón para cancelar la creación del módulo.
2. El experimentador debe ser capaz de ingresar el nombre, la versión, y parámetros del módulo de análisis.
3. El experimentador puede agregar más parámetros al presionar el botón correspondiente a agregar parámetro.
4. Si el experimentador presiona el botón de crear, y los campos llenados no se han validado, campos nombre y versión, no puede crear el módulo.
5. Si el experimentador presiona el botón de crear, y el formulario está validado correctamente, se crea el módulo, si el nombre de algún parámetro no se ingresa el formulario no toma en cuenta ese campo vacío al crear el módulo y se muestra un mensaje de creación de módulo de análisis satisfactorio momentáneamente para después redirigir a la página de visualización de módulos de análisis, y la tabla se actualiza mostrando el nuevo módulo.
6. Si el experimentador presiona el botón cancelar, no se crea el módulo y se redirige a la página de visualización de módulos de análisis.

### 2.5.1.2 Pruebas funcionales - Creación de un módulo de análisis - HU-01

La interfaz que muestra el formulario de creación de un módulo de análisis, Figura 2.43, contiene cajas de texto para ingresar el nombre del módulo, la versión del módulo y los parámetros del módulo. Para los parámetros se visualiza un botón para agregar nuevo parámetro que añade un nuevo cuadro de texto para ingresar el nombre del nuevo parámetro. Y un botón para crear el módulo y un botón para cancelar la creación del módulo.



The screenshot shows a web form titled "Create an analysis module". It features three text input fields stacked vertically, labeled "Module name", "Module version", and "Parameter Name". Below the "Parameter Name" field is a blue button with a white plus sign (+). At the bottom of the form are two buttons: an orange "CANCEL" button and a blue "CREATE MODULE" button.

**Figura 2.43** Formulario de creación de un módulo de análisis

Se ingresó del nombre, la version, y parámetro para el nuevo módulo.

## Create an analysis module

Module name

Module version

Parameter Name

**Figura 2.44** Ingreso de datos en el formulario de creación de un módulo de análisis

Se agregó dos nuevos parámetros presionando el botón para agregar parámetros. (Figura 2.45 y Figura 2.46)

Module name

Module version

Parameter Name

Parameter Name

Parameter Name

**Figura 2.45** Nuevos cuadros de texto para agregar nuevos parámetros

Module name —  
traffic3

Module version —  
2.0

Parameter Name —  
container\_name

Parameter Name —  
image

Parameter Name —  
ports

+

**Figura 2.46** Nuevos parámetros ingresados

No se pudo crear el nuevo módulo si los campos de nombre y versión no son válidos.

## Update an analysis module

Module name  
Module name is required

Module version  
Module version is required

Parameter Name —  
container\_name

Parameter Name —  
image

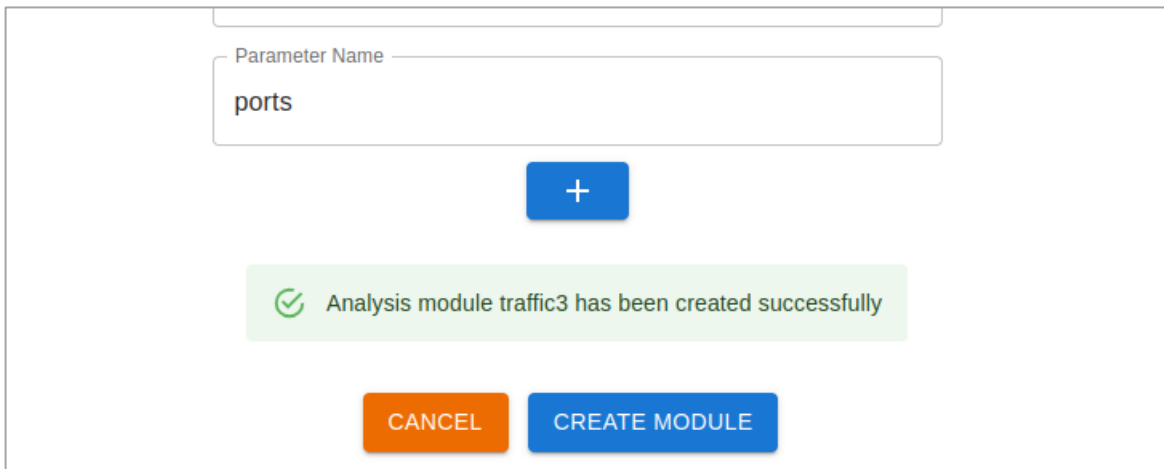
Parameter Name —  
ports

+

CANCEL UPDATE MODULE

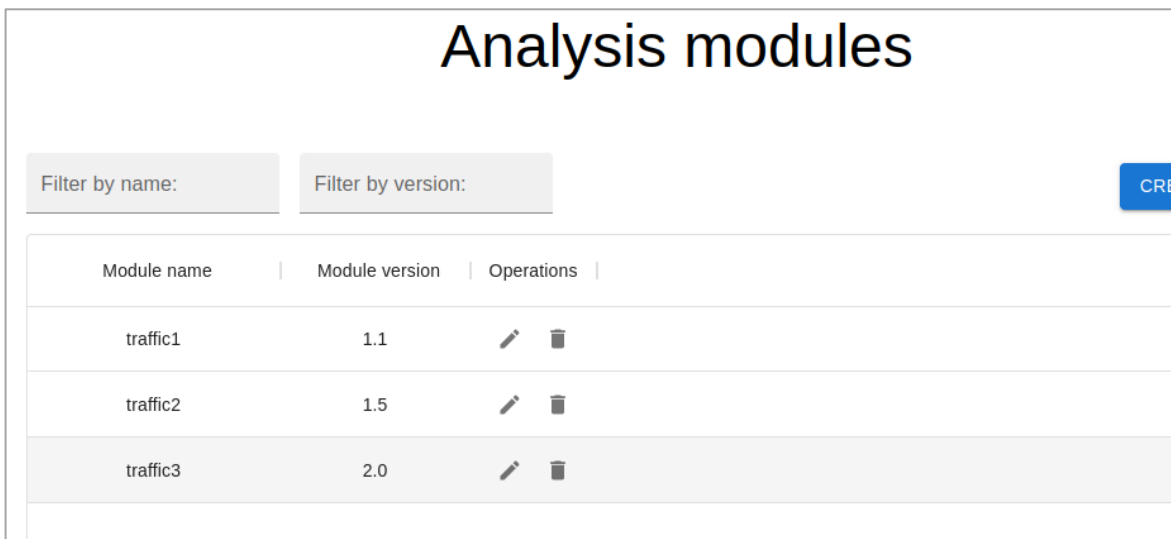
**Figura 2.47** Formulario de creación de módulo no válido

Cuando el formulario está correcto, después de presionar el botón de crear, momentáneamente apareció un mensaje de creación satisfactoria.



**Figura 2.48** Mensaje de creación de modulo satisfactorio

Después de crear el nuevo módulo de análisis, la página se redirigió a la visualización de módulos de análisis y el nuevo módulo apareció en la tabla.



**Figura 2.49** Interfaz de visualización de módulos de análisis que incluye el nuevo módulo creado

### 2.5.1.3 Análisis de aceptación - Creación de un módulo de análisis - HU-01

**Tabla 2.22.** Análisis de aceptación - HU-01 - Creación de un módulo de análisis

<b>Terminada</b>	<b>Criterio de aceptación</b>	<b>Aceptación</b>
------------------	-------------------------------	-------------------

Sí	En el formulario de creación de un módulo de análisis, se puede ingresar los datos del nuevo módulo de análisis, como el nombre del módulo, la versión del módulo y los parámetros del módulo. Para los parámetros se muestra un botón para agregar nuevo parámetro y poder ingresar el nombre del parámetro. Y se muestra un botón para crear el módulo y un botón para cancelar la creación del módulo	Sí
	El experimentador debe ser capaz de ingresar el nombre, la versión, y parámetros del módulo de análisis.	Sí
	El experimentador puede agregar más parámetros al presionar el botón correspondiente a agregar parámetro.	Sí
	Si el experimentador presiona el botón de crear, y los campos llenados no se han validado, campos nombre y versión, no puede crear el módulo.	Sí
	Si el experimentador presiona el botón de crear, y el formulario está validado correctamente, se debe crear el módulo, si el nombre de algún parámetro no se ingresa el formulario no toma en cuenta ese campo vacío al crear el módulo y se muestra un mensaje de creación de módulo de análisis satisfactorio momentáneamente para después redirigir a la página de visualización de módulos de análisis, y la tabla se actualiza mostrando el nuevo módulo.	Sí
	Si el experimentador presiona el botón cancelar, no se crea el módulo y se redirigirá a la página de visualización de módulos de análisis.	Sí



#### **2.5.1.4 Criterios de aceptación - Actualización de un módulo de análisis - HU-02**

1. En el formulario de actualización de un módulo de análisis, con los datos actuales del módulo, se puede actualizar estos datos si es el caso, como el nombre del módulo, la versión del módulo y los parámetros del módulo. Para los parámetros debe visualizarse un botón para agregar nuevo parámetro y poder ingresar el nombre del nuevo parámetro. Y un botón para actualizar el módulo y un botón para cancelar la actualización del módulo.
2. El experimentador debe ser capaz de ingresar un nuevo nombre, nueva versión, y cambiar o agregar parámetros.
3. Si el experimentador presiona el botón de actualizar, y los campos llenados no se han validado, campos nombre y versión, no se puede actualizar el módulo.
4. Si el experimentador presiona el botón de actualizar, y el formulario está validado correctamente, se debe actualizar el módulo, si el nombre de algún parámetro no se ingresa el formulario no toma en cuenta ese campo vacío al crear el módulo y se muestra un mensaje de actualización de módulo de análisis satisfactorio momentáneamente para después redirigir a la página de visualización de módulos de análisis.
5. Si el experimentador selecciona el módulo actualizado en la tabla de módulos, debe mostrarse la información de ese módulo con sus datos actualizados.
6. Si el experimentador presiona el botón cancelar, no se actualiza el módulo y se redirige a la página de visualización de módulos de análisis.

#### **2.5.1.5 Pruebas funcionales – Actualización de un módulo de análisis - HU-02**

La interfaz que muestra el formulario de actualización del módulo de análisis “traffic3”, Figura 2.50, contiene cajas de texto para actualizar el nombre del módulo, la versión del módulo y los parámetros del módulo. Para los parámetros se visualiza un botón para agregar nuevo parámetro que añade un nuevo cuadro de texto para ingresar el nombre del nuevo parámetro. Y un botón para actualizar el módulo y uno para cancelar la actualización del módulo.

## Update an analysis module

Module name

Module version

Parameter Name

Parameter Name

Parameter Name

**Figura 2.50** Formulario de actualización de un módulo de análisis

Se cambió el nombre y un par de parámetros en este módulo.

## Update an analysis module

Module name

Module version

Parameter Name

Parameter Name

Parameter Name

Parameter Name

**Figura 2.51** Formulario de actualización de un módulo de análisis modificado

Al dejar vacío el campo de version en este caso, no se pudo actualizar el módulo.

Update an analysis module

Module name  
traffic3

Module version  
Module version is required

Parameter Name  
image

Parameter Name  
container\_name

Parameter Name  
ports

Parameter Name  
tty

+

CANCEL UPDATE MODULE

**Figura 2.52** Formulario de actualización de un módulo de análisis invalido

tty

+

✓ Analysis module traffic3 has been updated successfully

CANCEL UPDATE MODULE

**Figura 2.53** Mensaje de actualización de modulo satisfactorio

Después de actualizar, la página se redirigió a la visualización de módulos, Figura 2.54 y en esta se selecciona el módulo actualizado “*traffic3*”, se visualiza una interfaz con la información del módulo de análisis actualizada, Figura 2.55.

Module name	Module version	Operations
traffic1	1.1	
traffic2	1.5	
traffic3	3.0	

**Figura 2.54** Visualización de la tabla de módulos con el módulo “*traffic3*” actualizado

**Analysis module: traffic3**

Module name —  
traffic3

Module version —  
3.0

Parameter Name —  
ports

Parameter Name —  
container\_name

Parameter Name —  
image

Parameter Name —  
tty

**BACK**

**Figura 2.55** Información actual del módulo actualizado “*traffic3*”

### 2.5.1.6 Análisis de aceptación - Actualización de un módulo de análisis - HU-02

**Tabla 2.23.** Análisis de aceptación - HU-02 - Actualización de un módulo de análisis

Terminada	Criterio de aceptación	Aceptación
Sí	En el formulario de actualización de un módulo de análisis, con los datos actuales del módulo, se puede actualizar estos datos si es el caso, como el nombre del módulo, la versión del módulo y los parámetros del módulo. Para los parámetros se muestra un botón para agregar nuevo parámetro y poder ingresar el nombre del nuevo parámetro. Y se muestra un botón	Sí

	para actualizar el módulo y un botón para cancelar la actualización del módulo	
	El experimentador debe ser capaz de ingresar un nuevo nombre, nueva version, y cambiar o agregar parámetros.	Sí
	Si el experimentador presiona el botón de actualizar, y los campos llenados no se han validado, campos nombre y version, no se puede actualizar el módulo.	Sí
	Si el experimentador presiona el botón de actualizar, y el formulario esta validado correctamente, se actualiza el módulo, si el nombre de algún parámetro no se ingresa, el formulario no toma en cuenta ese campo vacío al crear el módulo y se muestra un mensaje de actualización de módulo de análisis satisfactorio momentáneamente para después redirigir a la página de visualización de módulos de análisis.	Sí
	Si el experimentador selecciona el módulo actualizado en la tabla de módulos, debe mostrarse la información de ese módulo con sus datos actualizados.	Sí
Sí	Si el experimentador presiona el botón cancelar, no se actualiza el módulo y se redirige a la página de visualización de módulos de análisis.	Sí

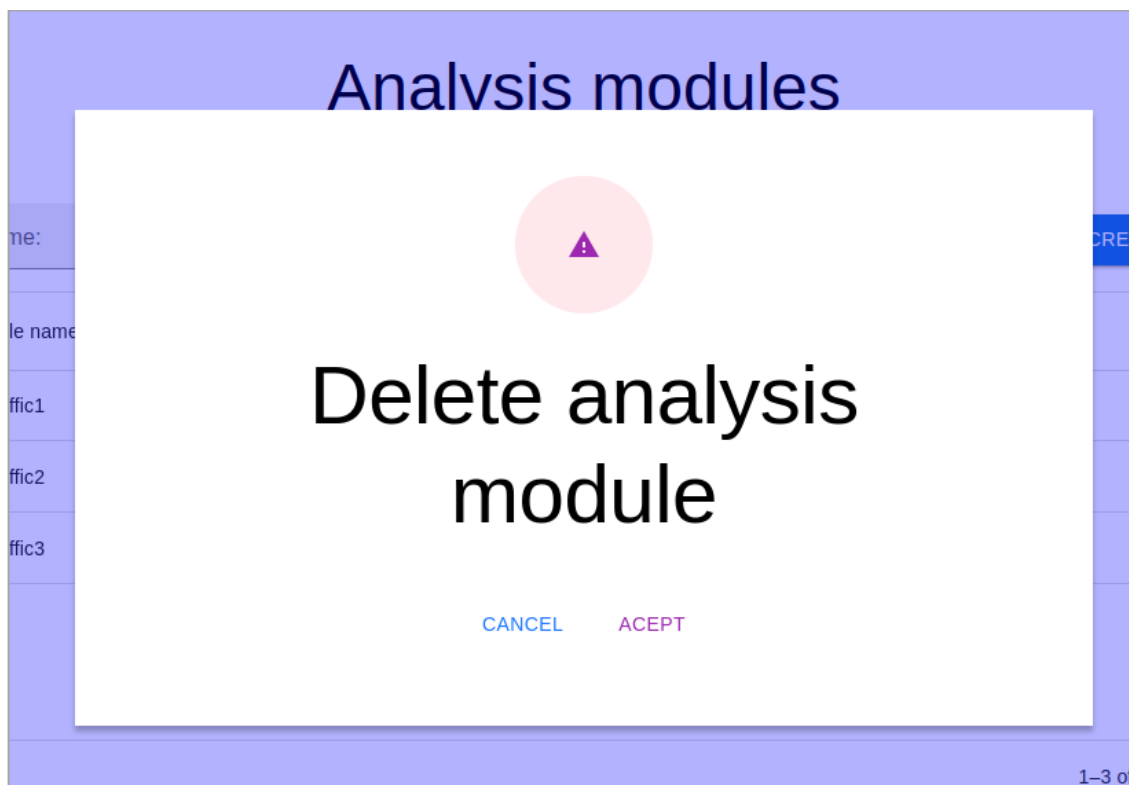
### 2.5.1.7 Criterios de aceptación - Eliminación de un módulo de análisis - HU-03

1. Al seleccionar la operación de eliminar módulo de análisis, el sistema debe mostrar un modal con una advertencia indicando que se va a eliminar el módulo, y unos botones para aceptar o cancelar la eliminación del módulo.
2. Si el experimentador presiona el botón *accept* para borrar el módulo de análisis mostrado, se elimina el módulo y en la visualización de módulos de análisis se actualiza la tabla desapareciendo el módulo eliminado.

3. Si el experimentador presiona el botón *cancel* para cancelar la eliminación del módulo. El modal se cierra y no se elimina el módulo.







### 2.5.1.8 Pruebas funcionales - Eliminación de un módulo de análisis - HU-03

Al seleccionar la operación de eliminar en el módulo de análisis “*traffic3*” de la tabla de módulos, apareció una ventana modal advirtiendo de la eliminación del módulo, con botones en la parte inferior para seleccionar si se acepta o cancela la eliminación del módulo de análisis seleccionado.



**Figura 2.56** Modal de advertencia al intentar eliminar el módulo de análisis “*traffic3*”

Al selecciona cancelar en el modal de eliminación, este se cerró y se visualizó nuevamente la tabla de módulos sin cambios.

Module name	Module version	Operations
traffic1	1.1	 
traffic2	1.5	 
traffic3	3.0	 

**Figura 2.57** Tabla de módulos de análisis sin cambios

Al seleccionar aceptar en el modal de eliminación del módulo de análisis “*traffic3*”, este se cerró y la tabla de módulos se actualizó quitando de la lista el módulo eliminado.

Module name	Module version	Operations
traffic1	1.1	
traffic2	1.5	

**Figura 2.58** Tabla de módulos de análisis actualizada

### 2.5.1.9 Análisis de aceptación - Eliminación de un módulo de análisis - HU-03

**Tabla 2.24.** Análisis de aceptación - HU-03 - Eliminación de un módulo de análisis

Terminada	Criterio de aceptación	Aceptación
Sí	Al seleccionar la operación de eliminar módulo de análisis, el sistema debe mostrar un modal con una advertencia indicando que se va a eliminar el módulo, y unos botones para aceptar o cancelar la eliminación del módulo	Sí

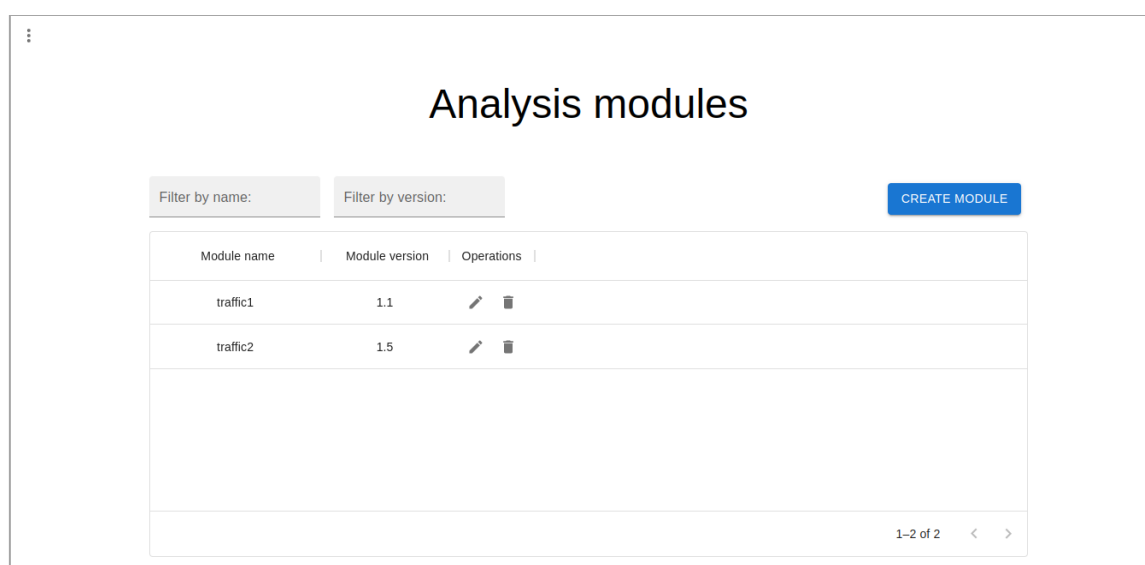
Sí	Si el experimentador presiona el botón <i>accept</i> para borrar el módulo de análisis mostrado, se elimina el módulo y en la visualización de módulos de análisis se actualiza la tabla desapareciendo el módulo eliminado	Sí
	Si el experimentador presiona el botón <i>cancel</i> para cancelar la eliminación del módulo. El modal se cierra y no se elimina el módulo	Sí

### 2.5.1.10 Criterios de aceptación - Visualización de módulos de análisis - HU-04

1. El prototipo de interfaz debe mostrar una interfaz con módulos de análisis que existen en una tabla donde muestra el nombre, la versión y las operaciones que se puede realizar.
2. En cada módulo las operaciones deben ser las de editar o eliminar módulos.
3. Sobre la tabla de módulos debe visualizarse cuadros de texto que corresponden a filtros por nombre o por versión y un botón para crear un nuevo módulo.
4. El experimentador debe poder hacer uso de los filtros ya sea por nombre o versión.
5. Al presionar el botón de crear módulo, debe mostrarse en pantalla el formulario de creación de un nuevo módulo.
6. Al seleccionar la operación de editar módulo, debe mostrarse en pantalla el formulario de actualización de un nuevo módulo.
7. Al seleccionar la operación de eliminar módulo, debe mostrarse en pantalla el modal de eliminación del módulo de análisis como advertencia.

### 2.5.1.11 Pruebas funcionales - Visualización de módulos de análisis - HU-04

La interfaz con la tabla de módulos de análisis muestra el nombre, la version y operaciones de los módulos listados.



**Figura 2.59** Pantalla de visualización de módulos de análisis

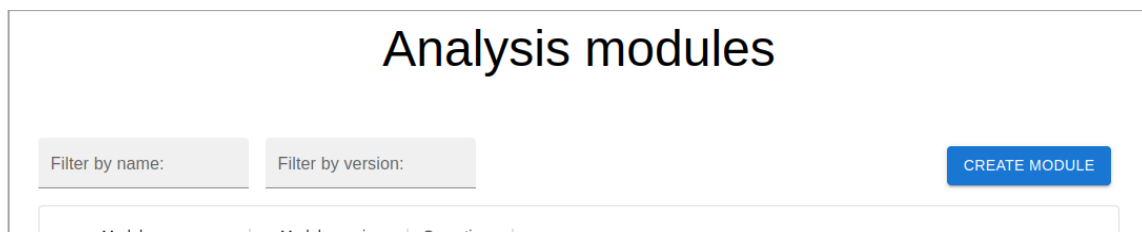


En la columna de operaciones, se visualiza las opciones de editar y eliminar.



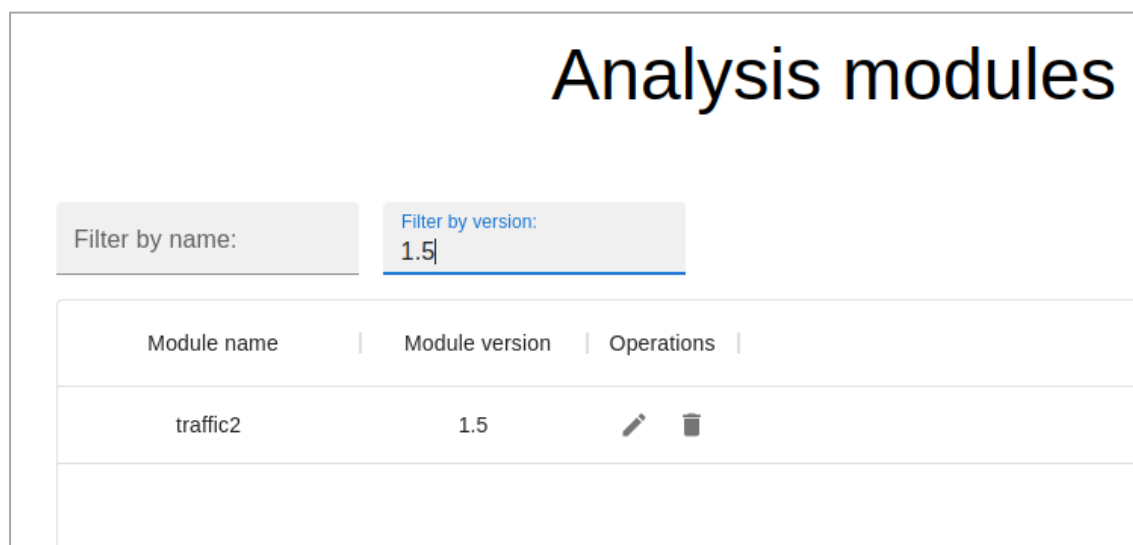
**Figura 2.60** Columna de operaciones en la tabla de módulos de análisis

Sobre la tabla de módulos de análisis se visualiza los filtros y en el extremo derecho un botón para agregar nuevos módulos.



**Figura 2.61** Filtros y botón para agregar módulos

Al usar los filtros la tabla mostró correctamente los módulos que coinciden con los filtros como se observa en las Figuras 2.62 y 2.63.



**Figura 2.62** Tabla de módulos de análisis filtrada por versión

# Analysis modules

Filter by name:  
traffic1

Filter by version:

Module name	Module version	Operations
traffic1	1.1	<span style="font-size: 1.2em;">✎</span> <span style="font-size: 1.2em;">🗑</span>

**Figura 2.63** Tabla de módulos de análisis filtrada por nombre

Al presionar el botón para crear nuevo módulo, la interfaz redirigió y mostró el formulario de creación de un nuevo módulo.

## Create an analysis module

+

CANCEL
CREATE MODULE

**Figura 2.64** Formulario de creación de un módulo de análisis

Al seleccionar la opción de editar en el módulo “*traffic 1*” para actualizar el módulo, la interfaz redirigió y mostró el formulario de actualización del módulo.

**Update an analysis module**

Module name  
traffic1

Module version  
1.1

Parameter Name  
image

Parameter Name  
container\_name

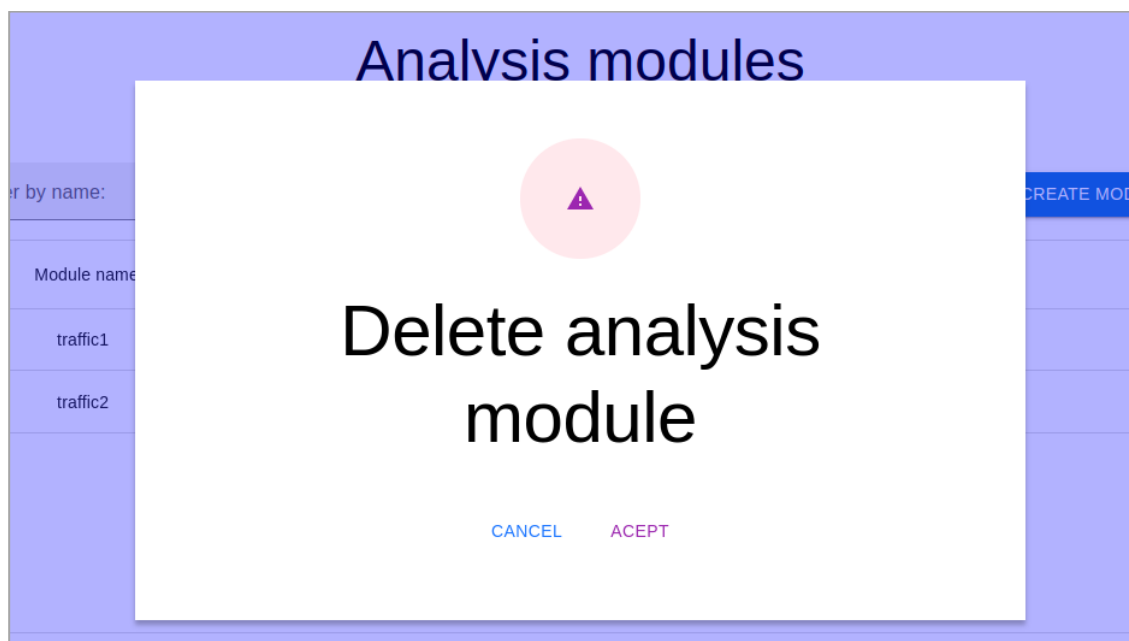
Parameter Name  
ports

+

CANCEL UPDATE MODULE

**Figura 2.65** Formulario de actualización del módulo de análisis “*traffic1*”

Al seleccionar la opción de eliminar en el módulo “*traffic1*” para eliminar el módulo de la lista, la interfaz mostró un modal de advertencia de eliminación del módulo.



**Figura 2.66** Modal de advertencia de eliminación de un módulo de análisis

#### 2.5.1.12 Análisis de aceptación - Visualización de módulos de análisis - HU-04

**Tabla 2.25.** Análisis de aceptación - HU-04 - Visualización de módulos de análisis

Terminada	Criterio de aceptación	Aceptación
Sí	La interfaz muestra los módulos de análisis que existen, en una tabla donde muestra el nombre, la versión y las operaciones que se puede realizar en cada modulo	Sí
	En cada módulo las operaciones deben ser las de editar o eliminar módulos	Sí
	Sobre la tabla de módulos debe visualizarse cuadros de texto que corresponden a filtros por nombre o por versión y un botón para crear un nuevo módulo	Sí
	El experimentador debe poder hacer uso de los filtros ya sea por nombre o versión	Sí
	Al presionar el botón de crear módulo, debe mostrarse en pantalla el formulario de creación de un nuevo módulo	Sí
Sí	Al seleccionar la operación de editar módulo, debe mostrarse en pantalla el formulario de actualización de un módulo	Sí
	Al seleccionar la operación de eliminar módulo, debe mostrarse en pantalla el modal de eliminación del módulo de análisis como advertencia	Sí

## 2.5.2 Pruebas – Pipelines de evaluación

Se realizaron pruebas de las historias respecto a los pipelines de evaluación, estas corresponden a las historias HU-05, HU-06, HU-07, HU-08, HU-09 y HU-10, los criterios de aceptación para estas historias se derivan de los casos de uso CU-05, CU-06 y CU-07.

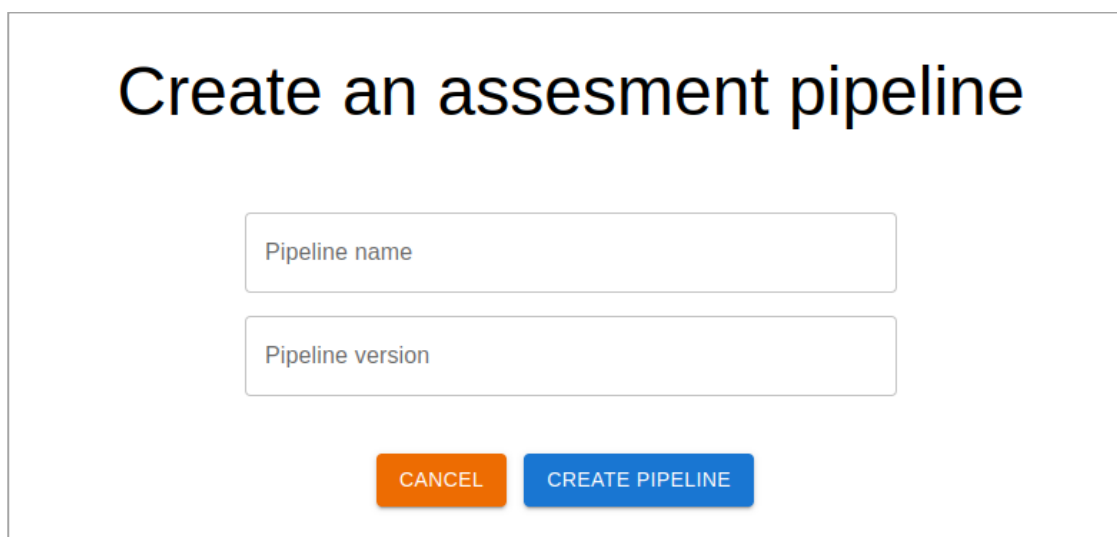
### 2.5.2.1 Criterios de aceptación - Creación de pipelines de evaluación - HU-05

1. En el formulario de creación de un pipeline de evaluación, se puede ingresar los datos del nuevo pipeline, como el nombre y la versión. Y se muestra un botón para crear el pipeline de evaluación y un botón para cancelar la creación.
2. El experimentador debe ser capaz de ingresar el nombre, la version del pipeline de evaluación.

3. Si el experimentador presiona el botón de crear, y los campos llenados no se han validado, campos nombre y version, no puede crear el pipeline.
4. Si el experimentador presiona el botón de crear, y el formulario esta validado correctamente, se crea el pipeline de evaluación, y se muestra un mensaje de creación del pipeline satisfactorio momentáneamente para después redirigir a la página de visualización de pipelines de evaluación, y la tabla se actualiza mostrando el nuevo pipeline en la lista.
5. Si el experimentador presiona el botón cancelar, no se crea el pipeline y se redirige a la página de visualización de pipelines de evaluación.

### 2.5.2.2 Pruebas funcionales- Creación de pipeline de evaluación - HU-05

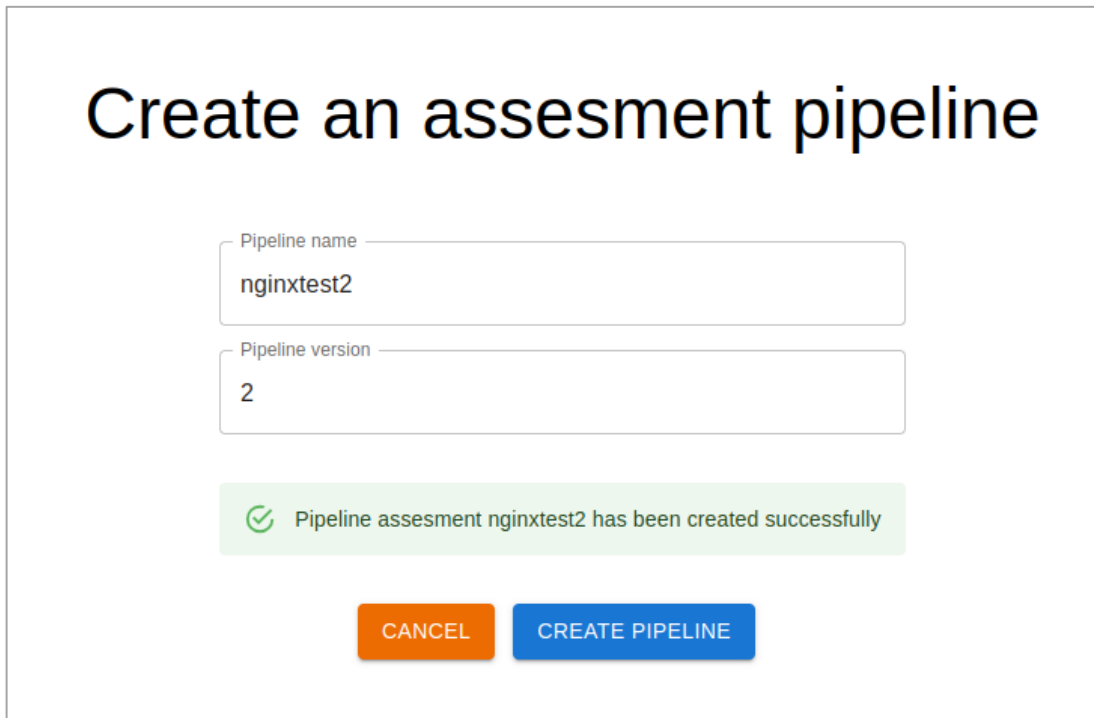
La interfaz que muestra el formulario de creación de un pipeline de evaluación, Figura 2.67, contiene cajas de texto para ingresar el nombre del módulo, la versión del módulo. Y se visualiza un botón para crear el módulo y un botón para cancelar la creación del módulo.



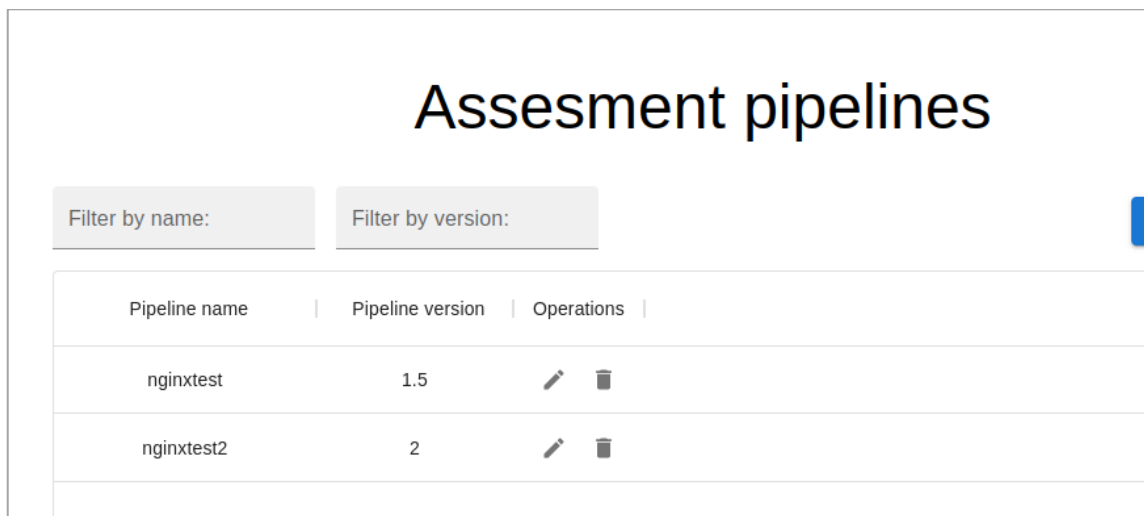
The image shows a web form titled "Create an assesment pipeline". It contains two text input fields: "Pipeline name" and "Pipeline version". Below the input fields are two buttons: "CANCEL" (orange) and "CREATE PIPELINE" (blue).

**Figura 2.67** Formulario de creación de un pipeline de evaluación

En este se ingresaron los valores en los campos de nombre y version. Si ambos campos están llenos, al dar clic en el botón de crear, la interfaz mostró un mensaje de creación satisfactorio momentáneamente Figura 2.68 y luego se redirigió a la interfaz con la lista de pipelines, Figura 2.69.



**Figura 2.68** Mensaje de creación de un pipeline de evaluación satisfactorio



**Figura 2.69** Interfaz de visualización de pipelines de evaluación actualizado

Para el caso de que los campos del formulario no se llenaron o no se validaron, no se pudo crear el pipeline de evaluación. Al presionar cancelar la interfaz retornó a la visualización de pipelines de evaluación.

# Create an assesment pipeline

Pipeline name

Name must be at least 5 characters long

Pipeline version

Version is required

CANCEL
CREATE PIPELINE

**Figura 2.70** Interfaz de visualización de pipelines de evaluación actualizado

### 2.5.2.3 Análisis de aceptación - Creación de un pipeline de evaluación - HU-05

**Tabla 2.26.** Análisis de aceptación - HU-05 - Creación de pipelines de evaluación

Terminada	Criterio de aceptación	Aceptación
Sí	En el formulario de creación de un pipeline de evaluación, se puede ingresar los datos del nuevo pipeline, como el nombre y la versión. Y se muestra un botón para crear el pipeline de evaluación y un botón para cancelar la creación	Sí
	El experimentador debe ser capaz de ingresar el nombre, la version del pipeline de evaluación	Sí
	Si el experimentador presiona el botón de crear, y los campos llenados no se han validado, campos nombre y version, no puede crear el pipeline	Sí
	Si el experimentador presiona el botón de crear, y el formulario esta validado correctamente, se crea el	Sí

	pipeline de evaluación, y se muestra un mensaje de creación del pipeline satisfactorio momentáneamente para después redirigir a la página de visualización de pipelines de evaluación, y la tabla se actualiza mostrando el nuevo pipeline en la lista.	
	Si el experimentador presiona el botón cancelar, no se crea el pipeline y se redirige a la página de visualización de pipelines de evaluación	Sí

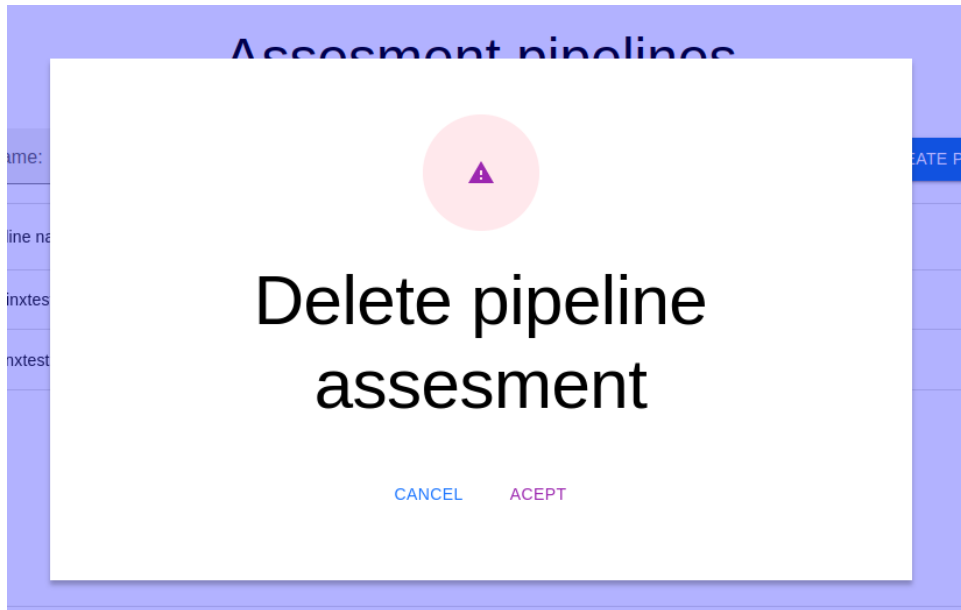
#### 2.5.2.4 Criterios de aceptación - Eliminación de un pipeline de evaluación - HU-06

1. Al seleccionar la operación de eliminar pipeline, el sistema debe mostrar un modal con una advertencia indicando que se va a eliminar el pipeline, y unos botones para aceptar o cancelar la eliminación de este.
2. Si el experimentador presiona el botón *accept* para borrar el pipeline de evaluación seleccionado, se elimina el pipeline y en la visualización de pipelines de evaluación se actualiza la tabla desapareciendo el pipeline eliminado.
3. Si el experimentador presiona el botón *cancel* para cancelar la eliminación del pipeline de evaluación. El modal se cierra y no se elimina el pipeline.

#### 2.5.2.5 Pruebas funcionales- Eliminación de un pipeline de evaluación - HU-06

Seleccionando la operación de eliminar, en pipeline de evaluación “*nginxtest2*”, el modal de advertencia de la eliminación del pipeline seleccionado apareció, también los botones para aceptar o cancelar la eliminación del pipeline.








**Figura 2.71** Modal de advertencia de eliminación de un pipeline de evaluación

Al seleccionar cancelar en el modal de eliminación, este se cerró y se visualizó nuevamente la tabla de pipelines sin cambios.

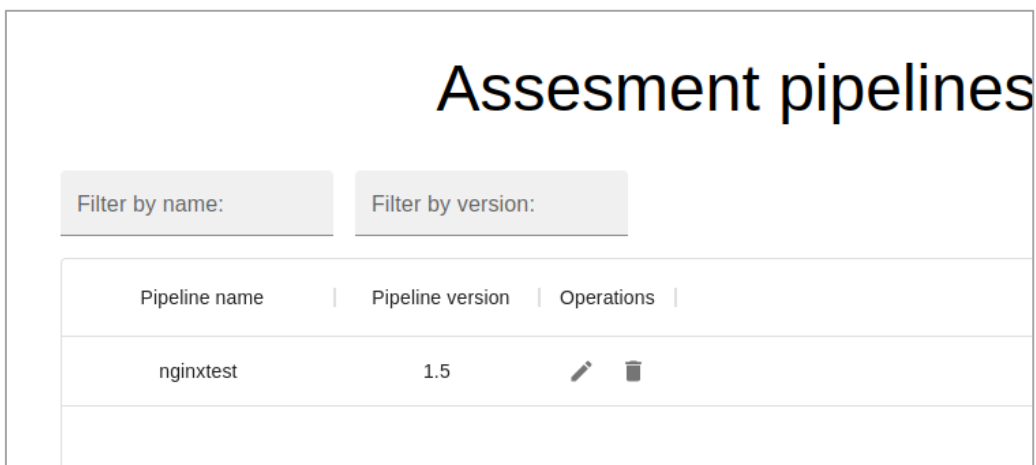
Assesment pipelines

Filter by name: Filter by version:

Pipeline name	Pipeline version	Operations
nginxtest	1.5	 
nginxtest2	2	 

**Figura 2.72** Visualización de la tabla de pipelines de evaluación sin cambios

Al seleccionar aceptar en el modal de eliminación del pipeline de análisis “*nginxtest2*”, este se cerró y la tabla de pipelines de evaluación se actualizó quitando de la lista el pipeline eliminado.



**Figura 2.73** Visualización de la tabla de pipelines de evaluación actualizada

### 2.5.2.6 Análisis de aceptación - Eliminación de un pipeline de evaluación - HU-06

**Tabla 2.2.** Análisis de aceptación - HU-06 - Eliminación de un pipeline de evaluación

Terminada	Criterio de aceptación	Aceptación
Sí	Al seleccionar la operación de eliminar pipeline, el sistema debe mostrar un modal con una advertencia indicando que se va a eliminar el pipeline, y unos botones para aceptar o cancelar la eliminación de este	Sí
	Si el experimentador presiona el botón <i>accept</i> para borrar el pipeline de evaluación seleccionado, se elimina el pipeline y en la visualización de pipelines de evaluación se actualiza la tabla desapareciendo el pipeline eliminado	Sí
	Si el experimentador presiona el botón <i>cancel</i> para cancelar la eliminación del pipeline de evaluación. El modal se cierra y no se elimina el pipeline	Sí

### 2.5.2.7 Criterios de aceptación - Configuración de un pipeline de evaluación - HU-07 y HU-08

1. El experimentador selecciona el pipeline de evaluación que desea configurar, el prototipo debe mostrar una ventana con una tabla de los módulos de análisis que se van a ejecutar. En la parte superior debe visualizarse campos de texto para

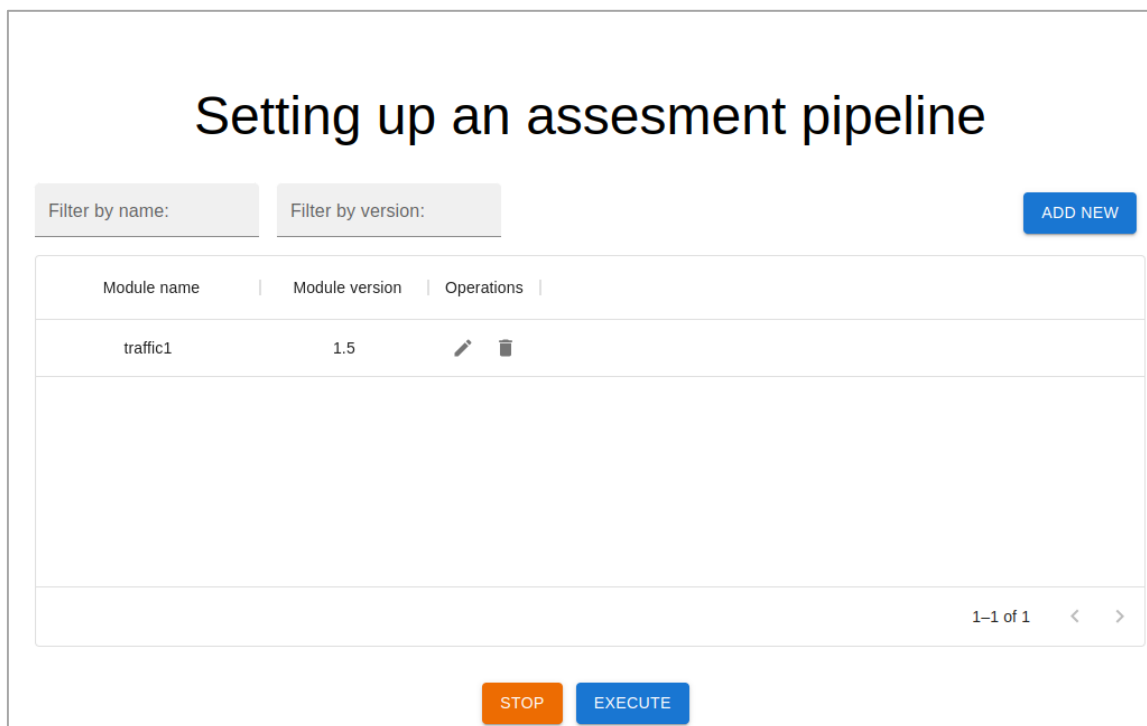
efectuar filtros por nombre o version además de un botón en el extremo derecho para "Add new" para agregar nuevo módulo al pipeline. Y dos botones inferiores Execute y Stop para ejecutar o detener el despliegue de los módulos en local con Docker compose.

2. Si el experimentador presiona el botón "Add new", para agregar un módulo de análisis a la lista de módulos para su ejecución. Se debe mostrar la lista de módulos que puede seleccionar para configurar.
3. Cuando el experimentador selecciona un módulo de análisis, debe visualizarse un formulario para configurar el módulo de análisis seleccionado este formulario debe tener el botón "Add module" para agregarlo al pipeline de evaluación. Y un botón para cancelar.
4. El formulario debe tener información del módulo a configurar como nombre y versión, y debe tener campos de texto para configurar los parámetros
5. Después de que el experimentador configura el módulo llenando todos los parámetros correctamente y presiona el botón de agregar modulo al pipeline. La interfaz redirige a la ventana de configuración del pipeline de evaluación, la tabla de módulos debe actualizarse añadiendo a la lista el módulo configurado.
6. En cambio, si en el formulario de configuración del módulo el experimentador presiona el botón de cancelar, la interfaz se redirecciona a la vista de pipelines de evaluación.

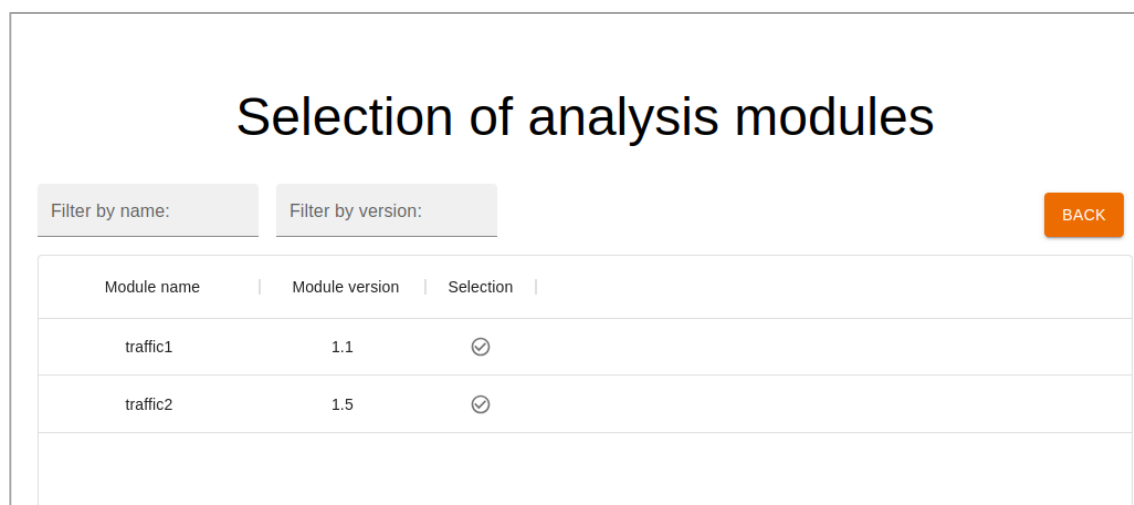
### **2.5.2.8 Pruebas funcionales - Configuración de un pipeline de evaluación - HU-07**

Seleccionando el pipeline de evaluación "*nginxtest*", se visualizó la interfaz con una tabla de los módulos de análisis configurados en el pipeline. En la parte superior están campos de texto para filtrar los módulos por nombre o version además de un botón en el extremo derecho para agregar un nuevo módulo al pipeline. También se visualizó dos botones inferiores Execute y Stop para ejecutar o detener el despliegue del pipeline de evaluación en local con Docker compose. Esto se muestra en la Figura 2.74.

Presionando el botón para agregar un módulo de análisis a la lista de módulos del pipeline la interfaz mostró la lista de módulos que se puede seleccionar para configurar Figura 2.75.



**Figura 2.74** Visualización de la tabla de módulos configurados en un pipeline de evaluación



**Figura 2.75** Visualización de la interfaz para la selección de un módulo para su configuración en el pipeline de evaluación

Al seleccionar un módulo de análisis, se abrió el formulario para configurar el módulo de análisis seleccionado, este formulario en la parte inferior contiene un botón para agregar el módulo al pipeline de evaluación, y un botón para cancelar. Este formulario indica la información del módulo nombre y versión, y tiene campos de texto para ingresar los valores de los parámetros.

## Setting Up an analysis module

Module name	traffic2
Module version	1.5
ports *	8181:81
container_name *	traffic2
image *	nginx

**Figura 2.76** Formulario de configuración un módulo de análisis

Al presionar el botón de cancelar, la interfaz redireccionó a la vista de pipelines de evaluación. En cambio, después de configurar el módulo llenando todos los valores de todos los parámetros correctamente y al presionar el botón de agregar modulo, apareció un mensaje de configuración satisfactorio, Figura 2.77. Luego la interfaz redireccionó a la ventana de configuración del pipeline de evaluación, la tabla de módulos se actualizó incluyendo el módulo que se configuró, Figura 2.78.

## Setting Up an analysis module

Module name

Module version

ports\*

container\_name\*

image\*

✓ Analysis module traffic2 has been set up successfully

CANCEL ADD MODULE

**Figura 2.77** Módulo de análisis configurado satisfactoriamente

## Setting up an assesment pipeline

Filter by name:  Filter by version:  ADD NEW

Module name	Module version	Operations
traffic1	1.5	
traffic2	1.5	

1-2 of 2 < >

STOP EXECUTE

**Figura 2.78** Lista de módulos de análisis configurados en el pipeline actualizada

### 2.5.2.9 Análisis de aceptación - Configuración de un pipeline de evaluación - HU-07

**Tabla 2.28.** Análisis de aceptación - HU-07-08 - Configuración de un pipeline de evaluación

Terminada	Criterio de aceptación	Aceptación
Sí	El experimentador selecciona el pipeline de evaluación que desea configurar, el prototipo muestra una ventana con una tabla de los módulos de análisis que se van a ejecutar. En la parte superior debe visualizarse campos de texto para efectuar filtros por nombre o version además de un botón en el extremo derecho para "Add new" para agregar nuevo módulo al pipeline. Y dos botones inferiores Execute y Stop para ejecutar o detener el despliegue de los módulos en local con Docker compose.	Sí
	Si el experimentador presiona el botón "Add new", para agregar un módulo de análisis a la lista de módulos para su ejecución. Se debe mostrar la lista de módulos que puede seleccionar para configurar	Sí
	Cuando el experimentador selecciona un módulo de análisis, debe visualizarse un formulario para configurar el módulo de análisis seleccionado este formulario debe tener el botón "Add module" para agregarlo al pipeline de evaluación. Y un botón para cancelar	Sí
	El formulario debe tener información del módulo a configurar como nombre y versión, y debe tener campos de texto para configurar los parámetros	Sí
	Después de que el experimentador configura el módulo llenando todos los parámetros correctamente y presiona el botón de agregar modulo al pipeline. La interfaz redirige a la ventana de configuración del pipeline de evaluación, la tabla de módulos debe actualizarse añadiendo a la lista el módulo configurado	Sí

Sí	En cambio, si en el formulario de configuración del módulo el experimentador presiona el botón de cancelar, la interfaz se redirecciona a la vista de pipelines de evaluación	Sí
----	---	----

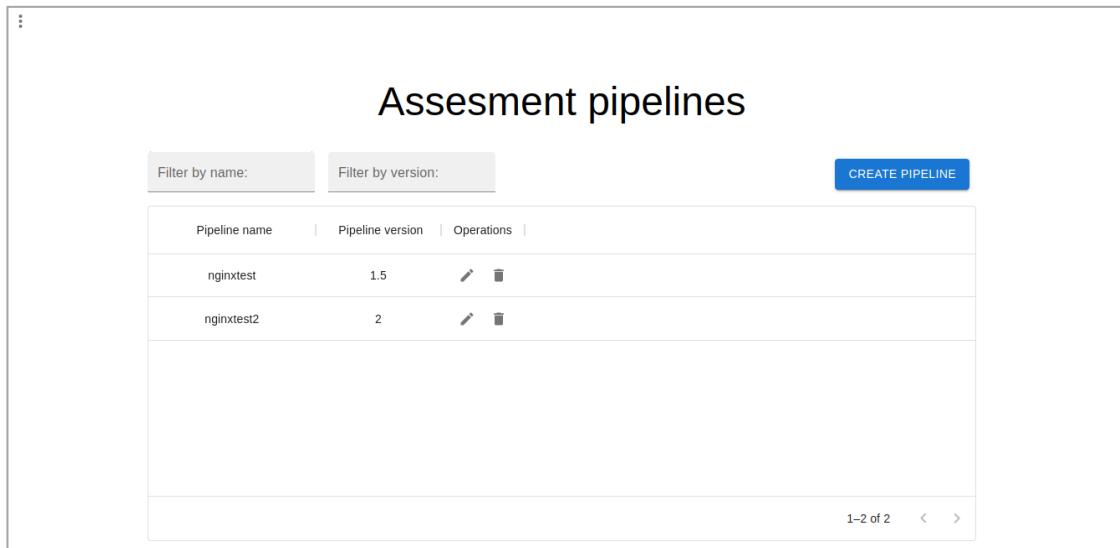
#### **2.5.2.10 Pruebas funcionales- Visualización de pipelines de evaluación - HU-09**

1. El prototipo de interfaz debe mostrar una pantalla con los pipelines de evaluación que existen en una tabla donde muestra el nombre, la versión y las operaciones que se puede realizar.
2. En cada registro de la tabla las operaciones deben ser las de editar o eliminar pipelines de evaluación.
3. Sobre la tabla de pipelines debe visualizarse cuadros de texto que corresponden a filtros por nombre o por versión y un botón para crear un nuevo pipeline.
4. El experimentador debe poder hacer uso de los filtros ya sea por nombre o versión.
5. Al presionar el botón de crear pipeline, debe mostrarse en pantalla el formulario de creación de un nuevo pipeline.
6. Al seleccionar la operación de editar pipeline, debe mostrarse en pantalla el formulario de actualización de un pipeline.
8. Al seleccionar la operación de eliminar pipeline, debe mostrarse en pantalla el modal de eliminación del pipeline de evaluación como advertencia.

#### **2.5.2.11 Pruebas funcionales- Visualización de pipelines de evaluación - HU-09**

Accediendo al menú en la parte superior izquierda, y seleccionando la opción “*Assesment Pipelines*”, se visualizó una pantalla con los pipelines de evaluación en una tabla donde muestra el nombre, la versión y las operaciones que se puede realizar, Figura 2.79. En cada módulo las operaciones que se mostraron son las de editar o eliminar pipelines de evaluación. Sobre la tabla de pipelines hay cuadros de texto que corresponden a filtros por nombre o por versión y un botón al extremo izquierdo para crear un nuevo pipeline.





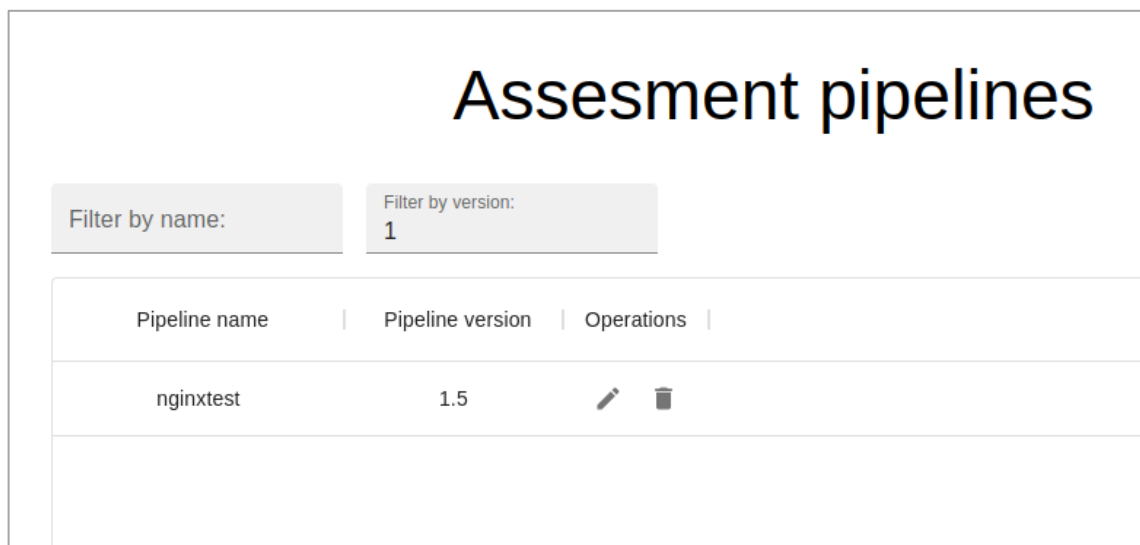
**Figura 2.79** Interfaz de visualización de pipelines de evaluación

Presionando el botón para crear un nuevo pipeline, el prototipo de interfaz mostró el formulario de creación de un pipeline de evaluación.

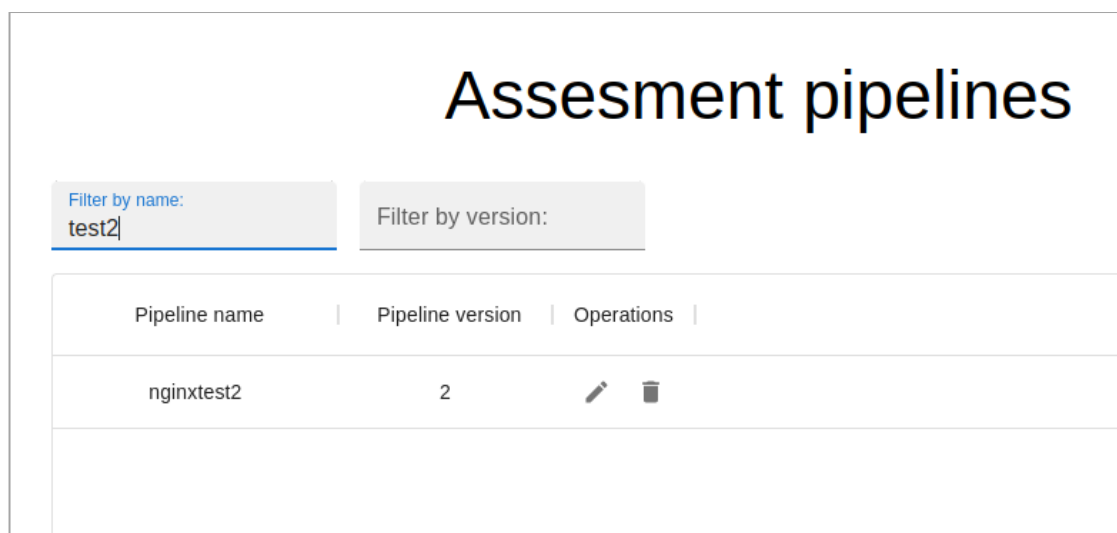
The screenshot shows a form titled "Create an assesment pipeline". It contains two input fields: "Pipeline name" and "Pipeline version". Below the input fields are two buttons: an orange "CANCEL" button and a blue "CREATE PIPELINE" button.

**Figura 2.80** Formulario de creación de un pipeline de evaluación

Haciendo uso de los filtros, se pudo filtrar los pipelines de evaluación, ya sea por nombre, así como por version. Figuras 2.81 y 2.82.

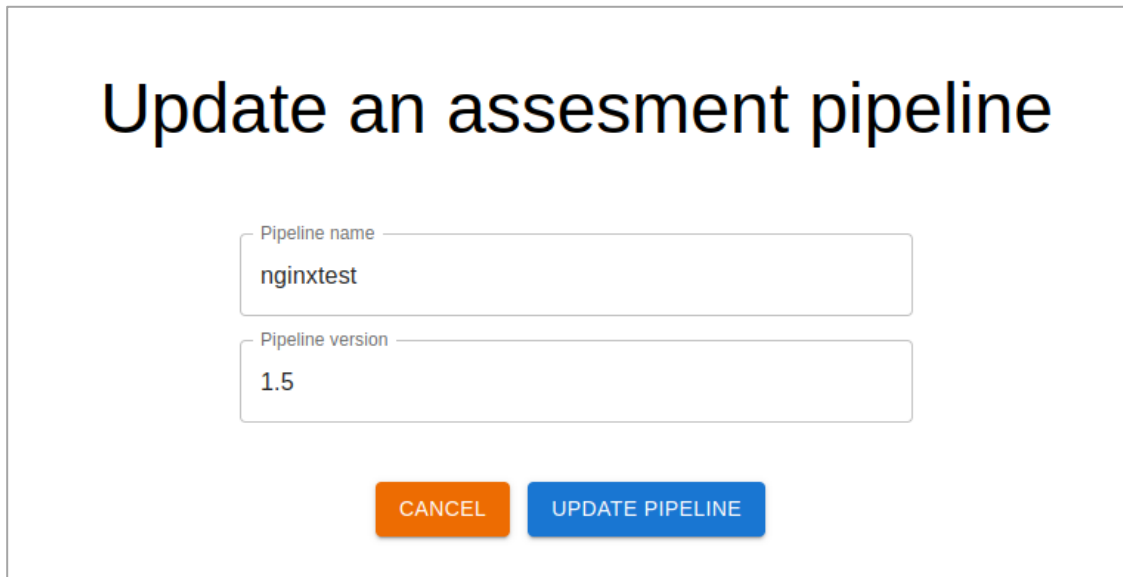


**Figura 2.81** Pipelines de evaluación filtrados por versión



**Figura 2.82** Pipelines de evaluación filtrados por nombre

Seleccionando la operación de editar en uno de los pipelines, se abrió un formulario de actualización en este caso del pipeline de evaluación "nginxtest", Figura 2.83.



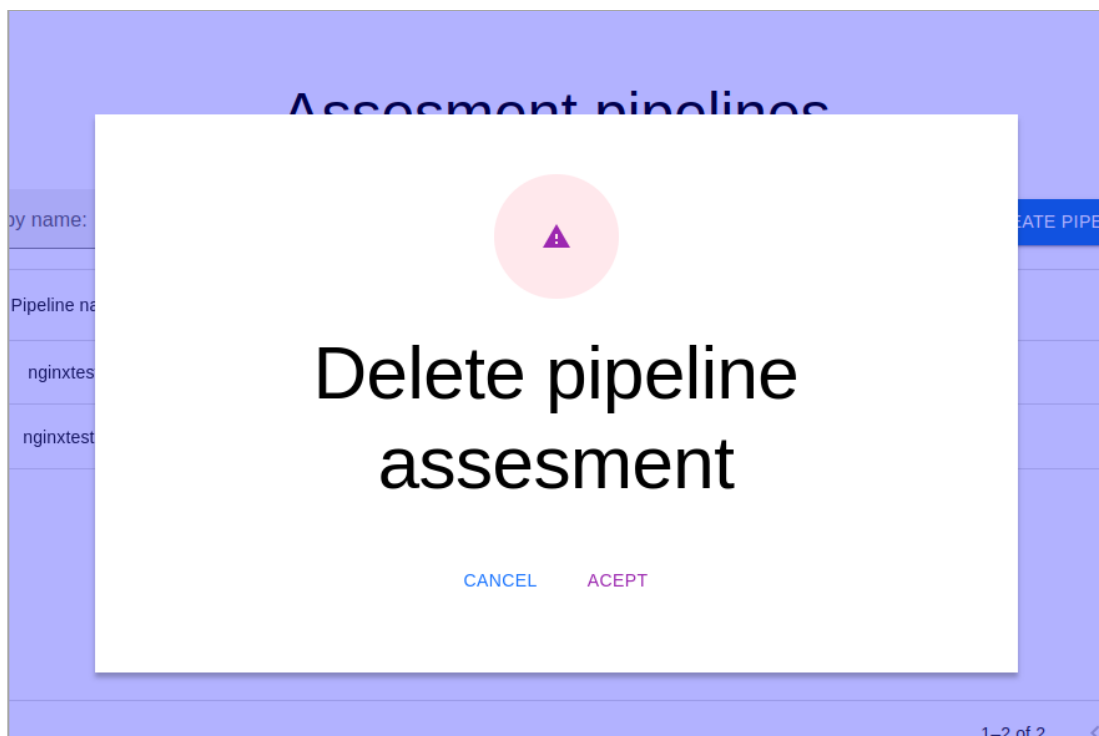
## Update an assesment pipeline

Pipeline name

Pipeline version

**Figura 2.83** Formulario de actualización de un pipeline de evaluación

Al seleccionar la operación de eliminar en uno de los pipelines de evaluación, se abrió una ventana modal, advirtiendo la eliminación de un pipeline de evaluación con las opciones de aceptar o cancelar la eliminación.



**Figura 2.84** Modal de advertencia de la eliminación de un pipeline de evaluación

### 2.5.2.12 Análisis de aceptación - Visualización de pipelines de evaluación - HU-09

**Tabla 2.29.** Análisis de aceptación - HU-09 - Visualización de pipelines de evaluación

Terminada	Criterio de aceptación	Aceptación
Sí	El prototipo de interfaz debe mostrar una pantalla con los pipelines de evaluación que existen en una tabla donde muestra el nombre, la versión y las operaciones que se puede realizar	Sí
	En cada módulo las operaciones deben ser las de editar o eliminar pipelines de evaluación.	Sí
	Sobre la tabla de pipelines debe visualizarse cuadros de texto que corresponden a filtros por nombre o por versión y un botón para crear un nuevo pipeline.	Sí
	El experimentador debe poder hacer uso de los filtros ya sea por nombre o versión.	Sí
	Al presionar el botón de crear pipeline, debe mostrarse en pantalla el formulario de creación de un nuevo pipeline.	Sí
	Al seleccionar la operación de editar pipeline, debe mostrarse en pantalla el formulario de actualización de un pipeline	Sí
	Al seleccionar la operación de eliminar pipeline, debe mostrarse en pantalla el modal de eliminación del pipeline de evaluación como advertencia	Sí

### 2.5.2.13 Criterios de aceptación - Ejecución de un pipeline de evaluación - HU-10

1. El experimentador selecciona el pipeline de evaluación que desea ejecutar, el prototipo debe mostrar una ventana con una tabla de los módulos de análisis que componen el pipeline de evaluación.

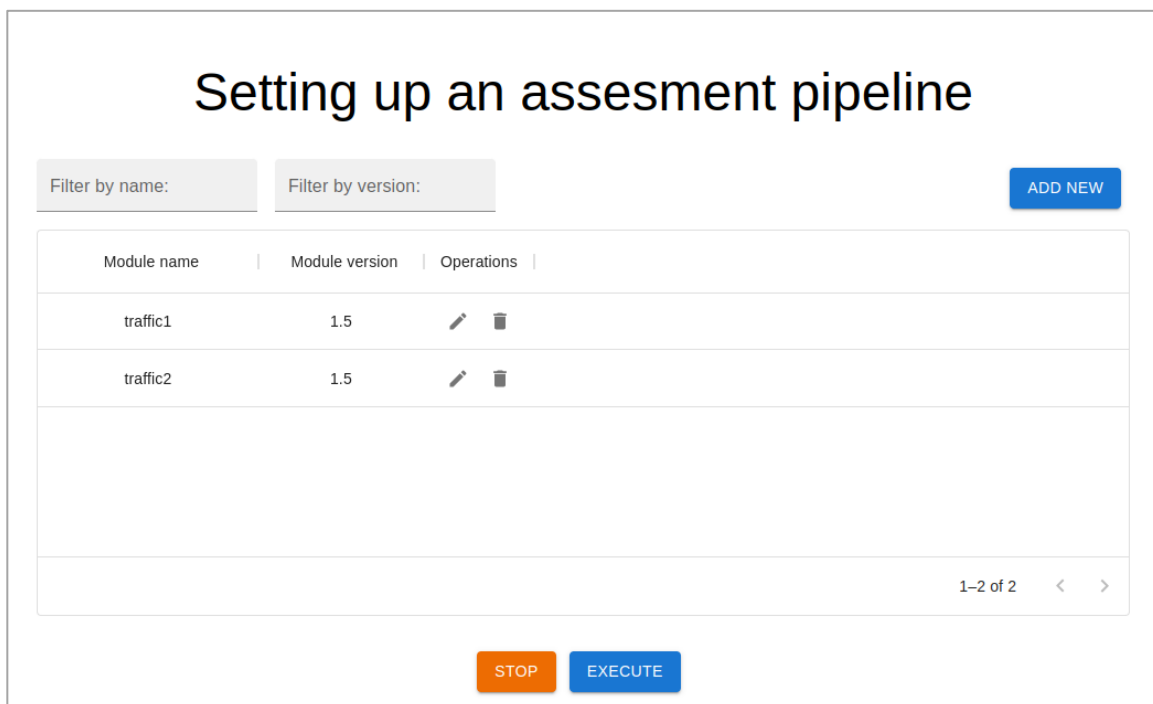
2. Al presionar el botón para ejecutar el pipeline de evaluación, el servidor (backend) crea el archivo "*docker-compose.yml*" con la configuración de la composición de contenedores a ejecutar. El servidor debe ejecutar el Docker compose en base al archivo "*docker-compose.yml*", si este se ha configurado correctamente.
3. En la interfaz se debe visualizar junto al botón de ejecución, el mensaje que indica que el pipeline está en ejecución.
4. En el equipo local, la interfaz de Docker desktop debe mostrar los contenedores que están ejecutándose en tiempo real.
5. Mientras el pipeline está en ejecución, al presionar el botón "*STOP*", debe desaparecer el mensaje informativo, y en la interfaz de Docker *desktop*, los contenedores que estaban en ejecución deben estar detenidos.

#### **2.5.2.14 Pruebas funcionales - Ejecución de un pipeline de evaluación - HU-10**

Seleccionando el pipeline de evaluación "*nginxtest*" para ejecutarlo, se visualizó una ventana con una tabla de los módulos de análisis que componen el pipeline de evaluación, Figura 2.85.

Presionando el botón para ejecutar el pipeline de evaluación, el servidor creó el archivo "*docker-compose.yml*" con la configuración de la composición de contenedores a ejecutar. El servidor enseguida ejecutó el Docker compose basándose en el archivo "*docker-compose.yml*", Figura 2.86, si este se ha configurado correctamente.

La interfaz mostró el mensaje que indica que el pipeline está en ejecución. Figura 2.87.



**Figura 2.85** Lista de módulos de análisis configurados en el pipeline "nginxtest"

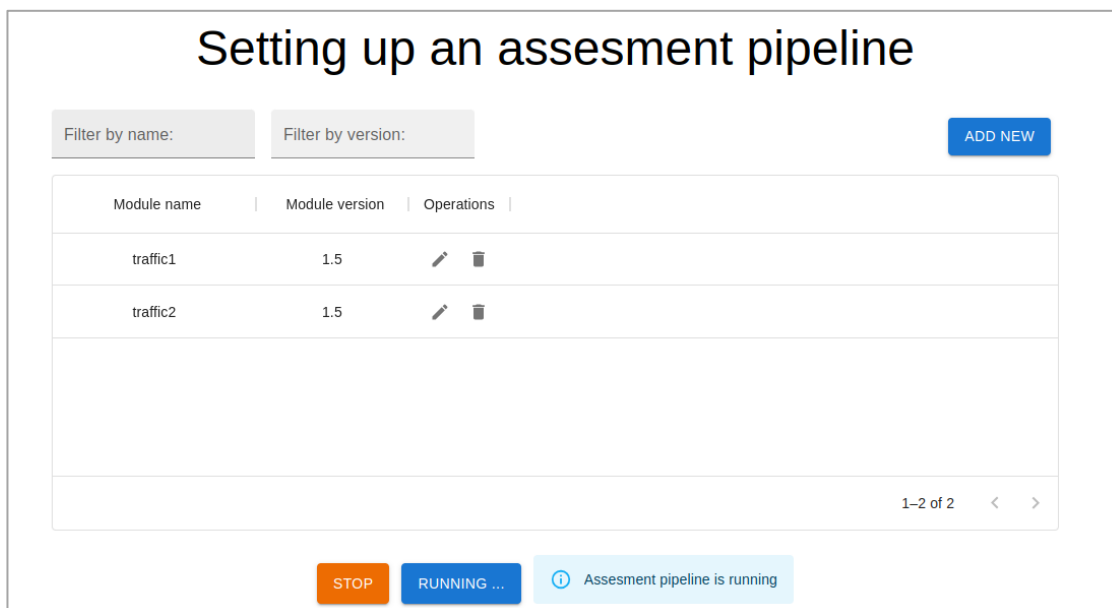
```

1  version: '1.5'
2  services:
3    traffic1:
4      image: nginx
5      container_name: traffic1
6      ports:
7        - '8080:80'
8    traffic2:
9      ports:
10       - '8181:81'
11     container_name: traffic2
12     image: nginx

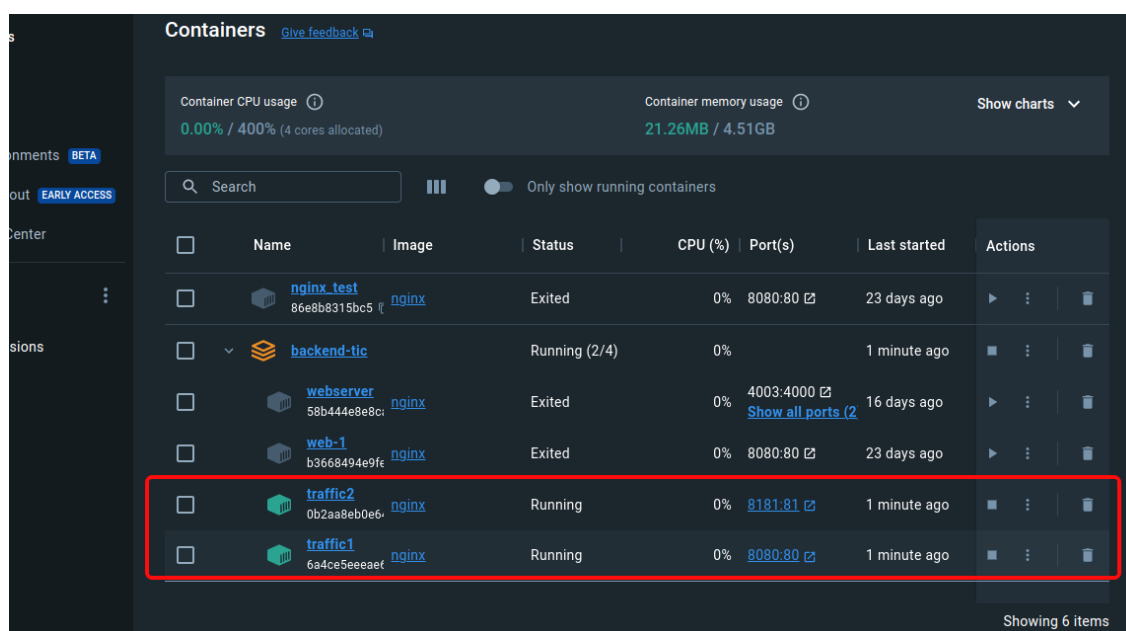
```

**Figura 2.86** Contenido del archivo generado en el servidor "docker-compose.yml"

En el equipo local, la interfaz de Docker Desktop mostró los contenedores en ejecución. Figura 2.88.

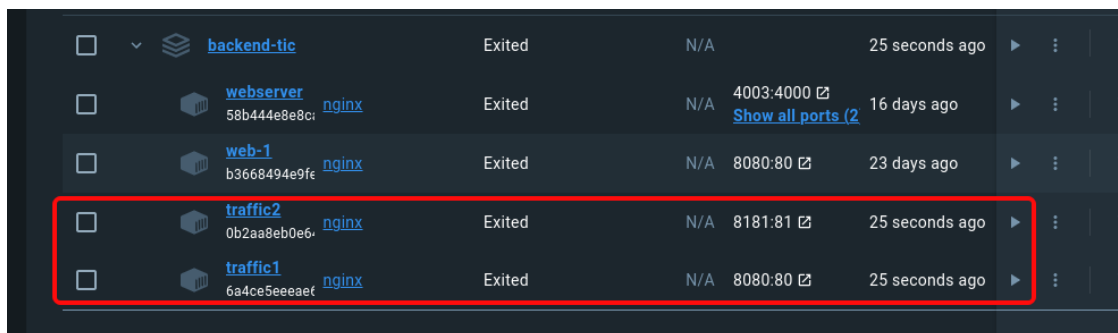


**Figura 2.87** Interfaz muestra mensaje de aviso de ejecución del pipeline de evaluación



**Figura 2.88** Interfaz en *Docker Desktop* muestra los contenedores *traffic1* y *traffic2* ejecutándose en los puertos 8080 y 8181 respectivamente

Al detener la ejecución del pipeline de evaluación, en la interfaz de Docker Desktop mostró los contenedores *traffic1* y *traffic2* detenidos, Figura 2.89.



**Figura 2.89** Interfaz en *Docker Desktop* muestra los contenedores *traffic1* y *traffic2* detenidos

### 2.5.2.15 Análisis de aceptación - Ejecución de un pipeline de evaluación - HU-10

**Tabla 2.30.** Análisis de aceptación - HU-10 – Ejecución de un pipeline de evaluación

Terminada	Criterio de aceptación	Aceptación
	El experimentador selecciona el pipeline de evaluación que desea ejecutar, el prototipo debe mostrar una ventana con una tabla de los módulos de análisis que componen el pipeline de evaluación	Sí
Sí	Al presionar el botón para ejecutar el pipeline de evaluación, el servidor (backend) crea el archivo “ <i>docker-compose.yml</i> ” con la configuración de la composición de contenedores a ejecutar. El servidor debe ejecuta el Docker compose en base al archivo “ <i>docker-compose.yml</i> ”, si este se ha configurado correctamente	Sí
	En la interfaz se debe visualizar junto al botón de ejecución, el mensaje que indica que el pipeline está en ejecución	Sí
	En el equipo local, la interfaz de Docker desktop debe mostrar los contenedores que están ejecutándose en tiempo real	Sí
Sí	Mientras el pipeline está en ejecución, al presionar el botón “STOP”, debe desaparecer el mensaje informativo, y en la interfaz de Docker desktop, los	Sí



	contenedores que estaban en ejecución deben estar detenidos.	
--	--	--

### 2.5.3 Pruebas – Pipeline de evaluación usando la imagen del módulo de tráfico “*cliip/platform:traffic1.12*” de CLIP

Se realizaron pruebas creando un módulo de tráfico con la imagen “cliip/platform:traffic1.12” de CLIP, creando un pipeline de evaluación configurando uno o varios módulos de tráfico en este pipeline y finalmente ejecutando el pipeline de evaluación.

#### 2.5.3.1 Criterios de aceptación - Pipeline de evaluación - módulo “*traffic1.12*”

1. El experimentador debe poder crear correctamente un módulo de análisis de nombre *traffic*, *version 1.12* y con estos parámetros: *image*, *container\_name*, *volumes*, *ports*, *devices*.
2. El experimentador debe poder crear correctamente un pipeline de evaluación, de nombre “*traffic\_test*” y version 1.
3. El experimentador debe ser capaz de configurar el pipeline de evaluación “*traffic\_test*”, agregando dos configuraciones de este módulo, estableciendo los siguientes valores en sus parámetros, los parámetros con más de un valor se ingresan separados por un espacio:

#### Primer módulo traffic:

*image*: cliip/platform:traffic1.12

*container\_name*: tr01

*volumes*:

- ./config/filebeat.yml:/etc/filebeat/filebeat.yml:ro
- ./config/tr01/executor.config:/app/executor/executor.config:ro
- ./config/tr01/info.device:/app/analyze/static/info.device:ro
- /privapp/app/logs:/app/logging/log/

*ports*:

- '4001:4000'
- '8081:8080'

*devices:*

- /dev/bus/usb/003/001:/dev/bus/usb/003/001

### **Segundo módulo traffic:**

*image:* cliip/platform:traffic1.12

*container\_name:* tr02

*volumes:*

- ./config/filebeat.yml:/etc/filebeat/filebeat.yml:ro

- ./config/tr02/executor.config:/app/executor/executor.config:ro

- ./config/tr02/info.device:/app/analyze/static/info.device:ro

- /privapp/app/logs:/app/logging/log/

*ports:*

- '4002:4000'

- '8082:8080'

*devices:*

- /dev/bus/usb/003/002:/dev/bus/usb/003/002

4. El experimentador debe ser capaz de ejecutar este pipeline de evaluación. El servidor debe crear el archivo “docker-compose.yml” según la configuración del pipeline de evaluación y luego ejecuta el Docker Compose.
5. La interfaz de Docker Desktop debe mostrar que efectivamente los contenedores están en ejecución.
6. El experimentador debe ser capaz de parar la ejecución del pipeline de evaluación.

### **2.5.3.2 Pruebas funcionales - Pipeline de evaluación - módulo “*traffic1.12*”**

Presionando el botón de creación de un nuevo módulo de análisis, la interfaz mostró el formulario de creación. Se estableció el nombre *traffic*, la version 1.12 y se agregó los parámetros: *image*, *container\_name*, *volumes*, *ports*, *devices*.

## Create an analysis module

Module name  
traffic

Module version  
1.12

Parameter Name  
image

Parameter Name  
container\_name

Parameter Name  
volumes

Parameter Name  
ports

Parameter Name  
devices

+

CANCEL CREATE MODULE

**Figura 2.90** Formulario de creación de nuevo módulo con los campos llenos

En la interfaz se observó por un momento un mensaje, Figura 2.91, indicando que el módulo ha sido creado con éxito.

Parameter Name  
devices

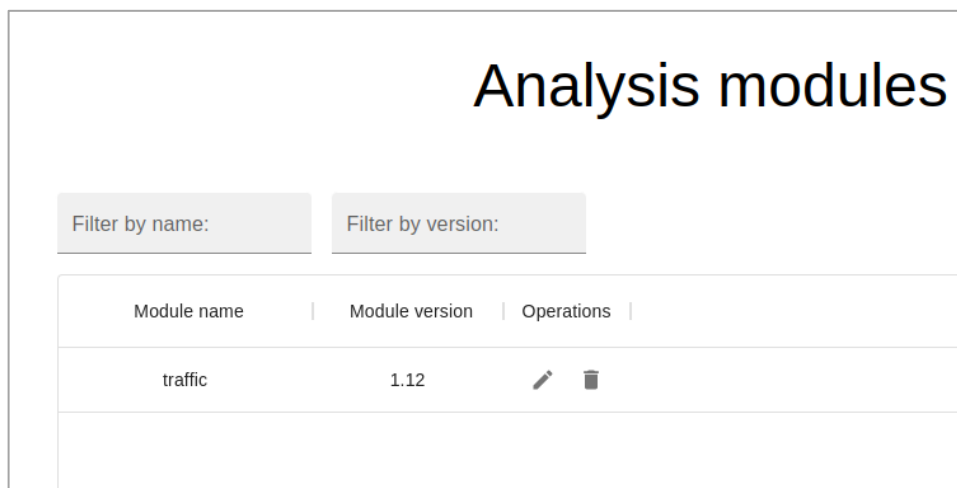
+

✓ Analysis module traffic has been created successfully

CANCEL CREATE MODULE

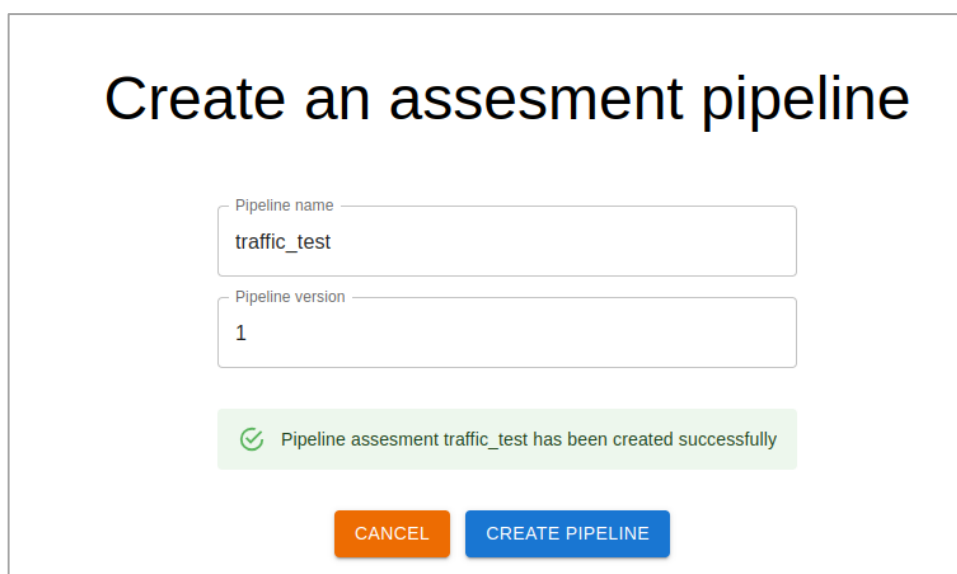
**Figura 2.91** Mensaje de creación exitosa del nuevo módulo

En la visualización de módulos apareció el módulo “traffic” creado.



**Figura 2.92** Interfaz de visualización de módulos de análisis

Accediendo a la sección de pipelines de evaluación en el menú situado en la parte superior a la izquierda. En la interfaz de pipelines de evaluación, se presionó el botón de crear pipeline, se llenó los campos nombre y version en el formulario y al presionar crear, un mensaje apareció por un momento indicando que el pipeline de evaluación se ha creado satisfactoriamente.



**Figura 2.93** Mensaje de creación exitosa del nuevo pipeline

Seleccionando el pipeline creado, en la interfaz de visualización de pipelines de evaluación, se configuro el módulo trafic seleccionándolo en la lista de módulos, en el formulario de configuración del módulo, se llenó los valores de los parámetros requeridos, Figura 2.94, este proceso se repitió una vez más para un segundo modulo según los valores de los parámetros que se muestran en la sección anterior. Figura 2.95.

## Setting Up an analysis module

Module name

Module version


image\*

container\_name\*

volumes\*

ports\*

devices\*

 Analysis module traffic has been set up successfully

**Figura 2.94** Primer módulo *traffic* configurado

En la interfaz del pipeline que se acaba de configurar, muestra una tabla con los dos módulos que se acaban de configurar. Figura 2.96.

Al presionar el botón de ejecutar, la interfaz muestra un mensaje indicando que el pipeline está ejecutándose. Figura 2.97.

El servidor crea correctamente el archivo “*docker-compose.yml*”, Figura 2.98. Pero en la consola del servidor, se ha impreso un mensaje de error al desplegar los contenedores, Figura 2.99. Finalmente, después de presionar el botón de parada, el mensaje de ejecución desapareció.

## Setting Up an analysis module

Module name

Module version

image\*

container\_name\*

volumes\*

ports\*

devices\*

✔ Analysis module traffic has been set up successfully

CANCEL
ADD MODULE

**Figura 2.95** Segundo módulo *traffic* configurado

## Setting up an assesment pipeline

Filter by name:

Filter by version:

ADD NEW

Module name	Module version	Operations
traffic	1.12	<span style="font-size: 1em;">✎</span> <span style="font-size: 1em;">🗑</span>
traffic	1.12	<span style="font-size: 1em;">✎</span> <span style="font-size: 1em;">🗑</span>

1-2 of 2 < >

STOP
EXECUTE

**Figura 2.96** Pipeline de evaluación configurado

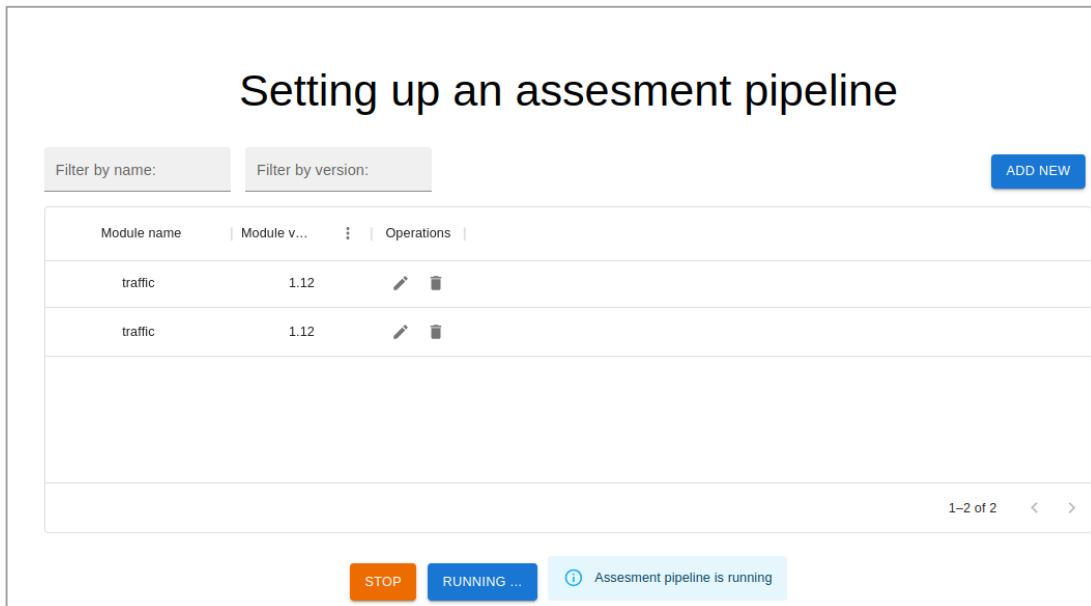


Figura 2.97 Mensaje indicando la ejecución del pipeline

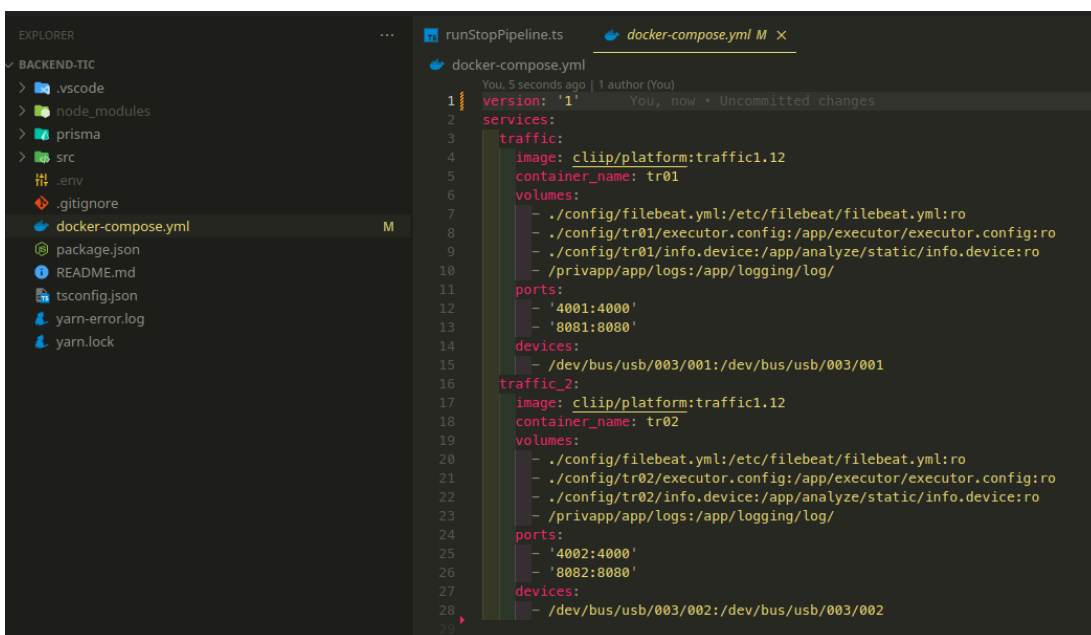


Figura 2.98 Archivo *docker-compose.yml* generado en el servidor

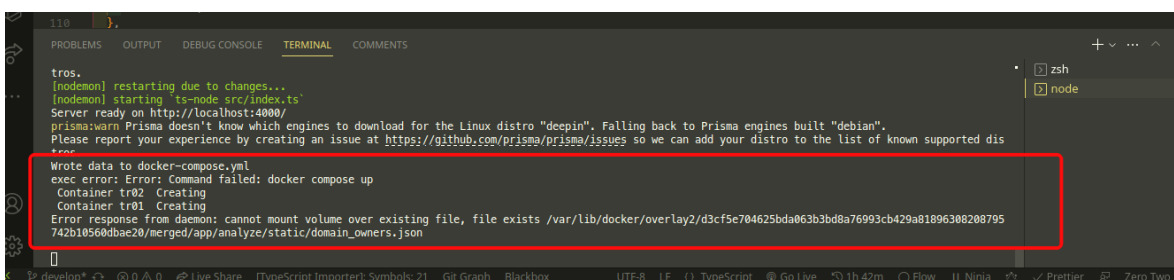


Figura 2.99 Mensaje de error al desplegar los contenedores

### 2.5.3.3 Análisis de aceptación - Pipeline de evaluación - módulo “traffic1.12”

**Tabla 2.31.** Análisis de aceptación - Pipeline de evaluación - módulo “traffic1.12”

Terminada	Criterio de aceptación	Aceptación
No	El experimentador debe poder crear correctamente un módulo de análisis de nombre <i>traffic</i> , <i>version 1.12</i> y con estos parámetros: <i>image</i> , <i>container_name</i> , <i>volumes</i> , <i>ports</i> , <i>devices</i>	Sí
	El experimentador debe poder crear correctamente un pipeline de evaluación, de nombre “ <i>traffic_test</i> ” y version 1	Sí
	El experimentador debe ser capaz de configurar el pipeline de evaluación “ <i>traffic_test</i> ”, agregando dos configuraciones de este módulo, estableciendo los valores de sus parámetros, los parámetros con más de un valor se ingresan separados por un espacio	Sí
	El experimentador debe ser capaz de ejecutar este pipeline de evaluación. El servidor debe crear el archivo “ <i>docker-compose.yml</i> ” según la configuración del pipeline de evaluación	Sí
	y luego ejecuta el Docker Compose.	No
	La interfaz de Docker Desktop debe mostrar que efectivamente los contenedores están en ejecución	No
	El experimentador debe ser capaz de parar la ejecución del pipeline de evaluación	Sí

Es posible que no se haya dado la ejecución de los contenedores con el módulo de CLIP, traffic1.12, por fallas en el mismo, ya que el mensaje de advertencia indica que no puede montar el volumen sobre un archivo existente.



#### **2.5.4 Aclaración**

La implementación de los pipelines de evaluación realizada por el prototipo de interfaz se llevó a cabo a nivel local mediante el uso de Docker Compose. Este enfoque permitió simular y verificar el proceso de despliegue de manera efectiva. Paralelamente, la opción de despliegue en la plataforma Google Cloud se presenta como una alternativa bastante similar en términos de configuración y proceso. Sin embargo, tras una evaluación de los recursos económicos actuales, se ha tomado la decisión de optar por la implementación exclusivamente a nivel local en lugar de proceder con el despliegue en Google Cloud Platform. La principal razón detrás de esta elección radica en la limitación de recursos financieros disponibles para cubrir los costos asociados con los recursos en la nube.

### **3 CONCLUSIONES Y RECOMENDACIONES**

#### **3.1 Conclusiones**

- La implementación de JSON Server como herramienta de prueba y simulación para operaciones de lectura y escritura en una base de datos ha demostrado ser una estrategia altamente efectiva en entornos de desarrollo y pruebas. Al emular una base de datos real a través de un archivo JSON, se ha logrado agilizar y simplificar el proceso de desarrollo al eliminar la necesidad de una infraestructura de base de datos completa durante las fases iniciales. Esta aproximación ha permitido probar con facilidad las interacciones de lectura y escritura, ajustar la lógica de la aplicación según sea necesario y detectar posibles problemas de manera temprana. Aunque JSON Server ofrece una solución provisional y simplificada, su flexibilidad y capacidad para imitar la funcionalidad de una base de datos lo convierten en una elección valiosa para la creación y evaluación de prototipos funcionales. Sin embargo, es importante recordar que esta implementación temporal no reemplaza la necesidad de una base de datos real en la etapa de producción, pero se erige como un recurso confiable en las etapas iniciales del proceso de desarrollo y pruebas.
- El desarrollo efectivo del prototipo de interfaz gráfica para implementar experimentos de evaluación de aspectos de privacidad en aplicaciones móviles a través de la plataforma CLIP representa un paso significativo hacia la mejora y consolidación de las capacidades de verificación de la privacidad. La capacidad

demostrada para ejecutar pipelines de evaluación utilizando Docker Compose de manera local destaca la viabilidad y la funcionalidad del prototipo.

- Aunque el prototipo actual ha alcanzado sus objetivos iniciales, es importante reconocer que existen oportunidades claras para mejoras futuras. La optimización de la interfaz gráfica, la expansión de las opciones de personalización de experimentos y la integración con servicios en la nube son áreas que podrían enriquecer aún más la herramienta.
- Con la implementación de mejoras y un proceso evolutivo relativamente constante, este prototipo tiene el potencial de transformarse en una valiosa y poderosa herramienta para la verificación exhaustiva de aspectos de privacidad en aplicaciones móviles. Al proporcionar a los desarrolladores y profesionales de seguridad una forma eficiente de evaluar y fortalecer la privacidad en sus aplicaciones, para contribuir significativamente a la protección de datos sensibles y a la construcción de aplicaciones móviles más confiables y seguras.

## 3.2 Recomendaciones

- Se podría considerar la optimización de la implementación del backend mediante la adopción de arquitecturas vanguardistas, entre las que destaca la arquitectura hexagonal. Esta última se presenta como una alternativa propicia para modularizar y desacoplar los componentes funcionales del sistema, al tiempo que fomenta una clara separación entre las capas de negocios y de infraestructura. La aplicación de dicha arquitectura puede conllevar beneficios sustanciales en términos de escalabilidad, mantenibilidad y testabilidad, al establecer una estructura que promueve la reutilización eficiente de código y facilita la incorporación de cambios sin afectar otros componentes.
- Si se está desarrollando una aplicación en React con TypeScript y se decide utilizar la librería Formik para manejar formularios, junto con la librería Yup para el esquema de validación de los datos ingresados en el formulario, es importante seguir algunas pautas para evitar problemas relacionados con el tipado. La combinación de Formik y Yup puede ser muy poderosa para crear formularios robustos con validaciones precisas, así que es esencial garantizar un desarrollo fluido y sin errores, manteniendo la coherencia del tipado asegurando que los tipos definidos en Yup para la validación coincidan con los tipos definidos en Formik, de

no coincidir el desarrollador se enfrentará a errores de tipado que probablemente tanto Formik como Yup no adviertan al desarrollador de estos errores, además se recomienda utilizar los tipos proporcionados por Yup como por ejemplo, utilizar 'yup.string' para definir validaciones de cadenas de texto y así sucesivamente para otros tipos. Además, al definir el esquema de validación, asegurarse de que los nombres de los campos en el esquema coincidan con los nombres utilizados en Formik y así las validaciones se apliquen correctamente. Por último, aprovechar las funcionalidades que da Yup para proporcionar mensajes de error personalizados que sean claros y útiles para los usuarios, en cada validación que se implemente.

- Considerar la integración futura de este prototipo con los servicios y tecnologías en la nube como la de *Google Cloud Platform* (GCP) y Kubernetes. Aprovechando las ventajas de la escalabilidad, la flexibilidad y la administración simplificada que ofrecen estas plataformas. La integración con GCP permitiría alojar la aplicación en un entorno altamente confiable y seguro, garantizando un acceso rápido y estable. Además la integración con Kubernetes facilitaría la gestión de contenedores y la gestión eficiente de los recursos, conduciendo a un despliegue más ágil y optimizado.

## 4 REFERENCIAS BIBLIOGRÁFICAS

- [1] H. Nissenbaum, "PRIVACY AS CONTEXTUAL INTEGRITY", 2004
- [2] A. N. Sánchez, "La privacidad como integridad contextual y su aplicación a las redes sociales", *Zer - Revista de Estudios de Comunicación*, vol. 20, no. 39, Dec. 2015, doi: 10.1387/zer.15531.
- [3] República del Ecuador, "Ley Orgánica de Protección de Datos Personales", 2021.
- [4] Unión Europea, "Reglamento general de protección de datos", 2016.
- [5] C. Troncoso, "*Privacy & Online Rights, Knowledge Area, Issue 1.0*", 2019.
- [6] Google, "Descripción general de Google Cloud", *Google Cloud*, 2022. [Online]. Available: <https://cloud.google.com/docs/overview?hl=es-419>. [Accessed: 31-May-2022]
- [7] Amazon, "Contenedores de Docker | ¿Qué es Docker? | AWS", *Amazon Web Services, Inc.*, 2022. [Online]. Available: <https://aws.amazon.com/es/docker/>. [Accessed: 31 - May - 2022]
- [8] *The Kubernetes Authors*, "¿Que es Kubernetes?", *Kubernetes*, 2022. [Online]. Available: <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>. [Accessed: 31- May- 2022].
- [9] *Meta Platforms, Inc.*, "React - Una biblioteca de JavaScript para construir interfaces de usuario", *Es.reactjs.org*, 2022. [Online]. Available: <https://es.reactjs.org/>. [Accessed: 31- May- 2022].
- [10] *Atlassian*, "Scrum: Qué es, cómo funciona y cómo empezar," *Atlassian*, 2023. [Online]. Available: <https://www.atlassian.com/es/agile/scrum>. [Accessed: 20 - Aug - 2023].
- [11] Prisma Data, Inc., "¿What is Prisma?," *Prisma*, 2023. [Online]. Available: <https://www.prisma.io/docs/concepts/overview/what-is-prisma>. [Accessed: 20 - Aug - 2023].
- [12] Material UI SAS, "Overview - material UI," *MUI*, 2023. [Online]. Available: <https://mui.com/material-ui/getting-started/>. [Accessed; Aug. 20 - Aug - 2023].
- [13] *Lucid Software Inc.*, "Tutorial de diagrama de actividades UML," *Lucidchart*, 2023- [Online]. Available: <https://www.lucidchart.com/pages/es/tutorial-diagrama-de-actividades-uml>. [Accessed: 20 - Aug - 2023].
- [14] *The GraphQL Foundation*, "A query language for your API," *GraphQL*, 2023. [Online]. Available: <https://graphql.org/>. [Accessed: 20 - Aug - 2023).

## **5 ANEXOS**

En esta sección se muestran los distintos anexos que complementan este trabajo de integración:

ANEXO A. Código fuente del prototipo de interfaz para el despliegue de experimentos de evaluación.

ANEXO B. Instrucciones para el despliegue local del frontend del prototipo de interfaz

ANEXO C. Instrucciones para el despliegue local del backend

ANEXO D. Demo del prototipo

## ANEXO A

### CODIGO FUENTE

El código fuente está disponible en los siguientes repositorios en github:

Frontend – Prototipo de interfaz:  
<https://github.com/Antonio-Pena/frontend>

Backend – Servidor:  
<https://github.com/Antonio-Pena/backend-tic>

También puede ser descargado en el siguiente enlace:  
[ProyectoTIC.zip](#)

[https://epnecuador-my.sharepoint.com/:u:/g/personal/antonio\\_pena\\_epn\\_edu\\_ec/ER-uiqjEDINHrIW0gvM5luIB9VINbVXo8ZsNlnltg1qzrw?e=9MTS9x](https://epnecuador-my.sharepoint.com/:u:/g/personal/antonio_pena_epn_edu_ec/ER-uiqjEDINHrIW0gvM5luIB9VINbVXo8ZsNlnltg1qzrw?e=9MTS9x)

## ANEXO B

### INSTRUCCIONES DE DESPLIEGUE DEL FRONTEND DE FORMA LOCAL

Para instalar el proyecto, siga los siguientes pasos:

1. Clonar el repositorio:

```
git clone https://github.com/Antonio-Pena/frontend.git
```

2. Instalar las dependencias:

```
cd frontend  
yarn
```

3. Iniciar el frontend:

```
cd frontend  
yarn dev
```

4. Acceder al proyecto en tu navegador web en la siguiente URL:

<http://localhost:3000/>

## ANEXO C

### INSTRUCCIONES DE DESPLIEGUE DEL BACKEND DE FORMA LOCAL

Para instalar el proyecto, siga los siguientes pasos:

1. Clonar el repositorio:

```
https://github.com/Antonio-Pena/backend-tic.git
```

2. Instalar las dependencias:

```
cd backend-tic  
yarn
```

3. Establecer las variables de entorno en el archivo .env en la ruta backend-tic/.env:

```
DATABASE_URL = "database-connection-string";
```

Actualiza la cadena de conexión a la base de datos si es necesario. Para este trabajo hemos utilizado una base de datos postgresql en Render de render.com.

4. Es necesario instalar nodemon antes de ejecutar el backend

```
npm install -g nodemon
```

También puedes instalar nodemon como dependencia de desarrollo:

```
cd backend-tic  
yarn add -D nodemon
```

5. Iniciar el backend:

```
cd backend-tic  
yarn start:dev
```

6. Accede al servidor Apollo en tu navegador web en la siguiente URL:

<http://localhost:4000/>

7. Para generar el modelo de datos con prisma se establece el siguiente comando

```
yarn db:generate
```

8. Para actualizar la base de datos con prisma el siguiente comando está configurado:



```
yarn db:push
```

9. Para gestionar la base de datos con prisma el siguiente comando está configurado:

```
yarn db:studio
```

10. Para descargar la imagen de nginx, ejecutar el siguiente comando en la terminal:

```
docker pull nginx
```

11. Para descargar la imagen del módulo de tráfico de CLIP, ejecutar el siguiente comando en la terminal:

```
docker pull cliip/platform:traffic1.12
```

## ANEXO D

### DEMO DEL PROTOTIPO

La demo del funcionamiento de este prototipo se puede encontrar en el siguiente enlace:

[DemoPrototipo.mkv](#)

<https://epnecuador->

[my.sharepoint.com/:v/g/personal/antonio\\_pena\\_epn\\_edu\\_ec/EfUVfwjNPzJMoco3cj0q2oBAYzCONzkAKGPtcRos06CiA?e=DeBpTS](https://epnecuador-my.sharepoint.com/:v/g/personal/antonio_pena_epn_edu_ec/EfUVfwjNPzJMoco3cj0q2oBAYzCONzkAKGPtcRos06CiA?e=DeBpTS)