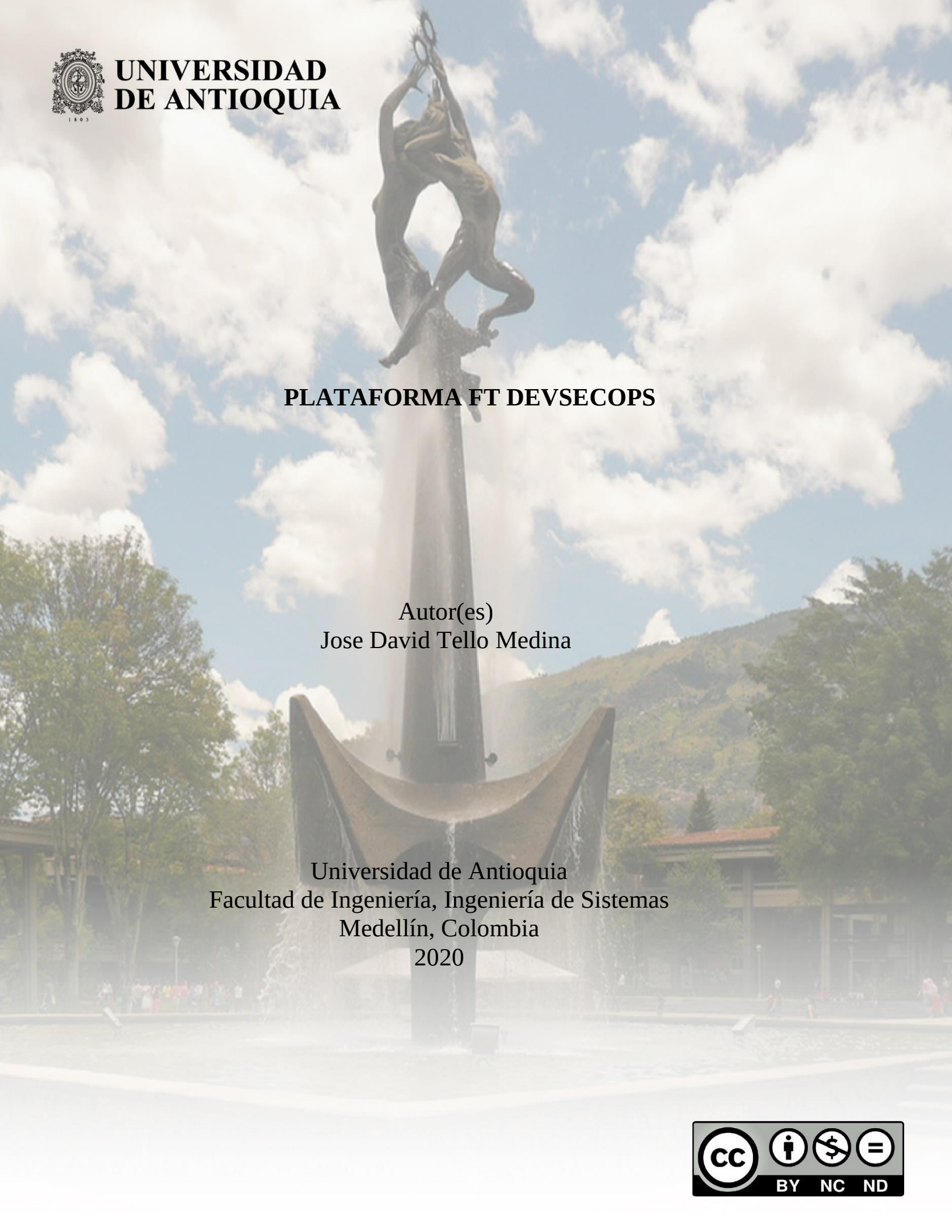


UNIVERSIDAD DE ANTIOQUIA

PLATAFORMA FT DEVSECOPS

Autor(es)
Jose David Tello Medina

Universidad de Antioquia
Facultad de Ingeniería, Ingeniería de Sistemas
Medellín, Colombia
2020

CC BY NC ND

PLATAFORMA FT DEVSECOPS

Jose David Tello Medina

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:
Ingeniería de Sistemas

Asesores (a):
Juan Carlos Ríos Paniagua
Ingeniero Electrónico

Línea de Investigación:
Seguridad de la información en el ciclo de vida del desarrollo de software

Universidad de Antioquia
Facultad de Ingeniería, Ingeniería de Sistemas.
Medellín, Colombia
2020.

Resumen

El desarrollo de software ha ido evolucionado al implementar metodologías ágiles y culturas como DevOps; sin embargo, la seguridad seguía estando al final del ciclo de vida de desarrollo. Para dicha problemática surgió "DevSecOps", que busca implementar seguridad desde el inicio de las fases de DevOps, contribuyendo así en la detección temprana y mitigación de vulnerabilidades, evitando que éstas permanezcan hasta su salida a producción

La finalidad de este proyecto es desarrollar un módulo SAST (Static Analysis Security Testing), para ser utilizado en las fases iniciales de DevSecOps, de forma que se identifiquen vulnerabilidades comunes en el código de forma automática, con el fin de que los desarrolladores puedan solucionar los hallazgos en fases pre-productivas.

Introducción

El ciclo de vida del desarrollo de software es un proceso que ha ido evolucionando con el paso del tiempo. Anteriormente se implementaban metodologías de desarrollo que debían tener cierto grado de madurez en una fase de la construcción para poder continuar con la siguiente, sin embargo, estas metodologías eran reacias al cambio, por lo que una actualización o cambio de versión del software que se estaba construyendo, suponía muchos problemas a la hora de implementarse, originando pérdidas en productividad y calidad de las empresas en los sectores que dependen de los productos de software. La aparición de las metodologías ágiles supuso un cambio positivo en el ciclo de vida de desarrollo, ya que se puede entregar un producto con características reducidas, pero que puede generar valor al usuario final, y posteriormente ir añadiendo nuevas funcionalidades consiguiendo así: flexibilidad, dinamismo y velocidad, requeridos para responder a las demandas actuales del mercado. A pesar de las ventajas que tienen las metodologías ágiles para el desarrollo de un proyecto, la seguridad seguía ejecutándose sólo en la etapa de producción, lo que suponía un alto impacto en disponibilidad y estabilidad a la hora de realizar un fix, sin mencionar los retrasos en la implementación de nuevas funcionalidades. La necesidad de poder implementar la seguridad tanto en el desarrollo como en la infraestructura ha hecho que se plantee la inclusión de técnicas y metodologías más recientes, para así sacar todas las ventajas que el agilismo pretende otorgar.

DevOps es una cultura de construcción y operación de servicios digitales, donde su principal objetivo es facilitar la comunicación entre el equipo de desarrollo (Development) y el equipo de operaciones (Operations), minimizando así los tiempos entre los cambios de software y la puesta en producción. DevOps consta de 2 fases principales que son: integración continua (por sus siglas en inglés: CI) y despliegue continuo (por sus siglas en inglés: CD). En la fase de CI se busca realizar entregas de versiones de software de forma periódica y con pruebas automatizadas, todo esto por medio de un sistema de versionamiento de código como GitHub, GitLab, BitBucket, etc. En la fase de CD se busca automatizar el proceso de despliegue del software entregado en la fase de CI, para que sean efectivos los nuevos cambios a los clientes. Sin embargo, aunque DevOps ofrece muchas ventajas en el desarrollo de software, esta cultura aún no es implementada por una

gran cantidad de empresas alrededor del mundo ya sea porque muchas de estas empresas aún manejan sistemas legacy, en los cuales tienen sus actividades core de negocio, porque aún no implementan bien las metodologías ágiles debido a las particularidades del negocio o simplemente por desconocimiento y miedo al cambio [1]. Si a este panorama sumamos el hecho de que la seguridad de los sistemas suele ser una tarea ejecutada sólo al final del ciclo de desarrollo junto con los análisis y la remediación de vulnerabilidades, esto termina por ser un proceso costoso, que ralentiza la puesta en marcha de nuevas versiones. Como consecuencia de esto surgió DevSecOps, una variante segura a DevOps que busca minimizar tiempos, costos y aumentar la seguridad en el ciclo de vida de los productos digitales [2].

La cultura DevSecOps es relativamente nueva [3] y se enfoca en realizar pruebas de seguridad en todas las etapas de DevOps, es decir, no solo se realizan pruebas con el producto ya desplegado, sino que se realiza análisis inicial y un monitoreo constante, lo que permite resolver los problemas antes de que un producto salga a producción. DevSecOps busca que los equipos de seguridad tengan más tiempo para ayudar en la próxima iniciativa de negocio en lugar de resolver problemas de vulnerabilidades en producción todo el tiempo.

FT Forensic Technology S.A.S es una empresa que busca brindar seguridad y proteger los datos de los clientes a través de sus servicios, los cuales se enfocan en integrar la seguridad a los objetivos del negocio de manera simple, dándole un alto valor a las organizaciones. Uno de los servicios que la empresa ofrece es el de pruebas de seguridad basadas en riesgo, compuesto por: pentesting, red teaming, Hacking ético a infraestructura y aplicaciones. Al ver los problemas de la implementación de los procesos de seguridad tradicional en el ciclo de vida del desarrollo con metodologías ágiles y DevOps, la empresa desea desarrollar una solución llamada FT-Platform que facilite la detección de vulnerabilidades en etapas tempranas de desarrollo, implementando pruebas de seguridad estáticas SAST (Static Application Security Test) en la etapa de integración de DevOps, dicho desarrollo busca tener un impacto a nivel latinoamericano e incursionar en otro tipo de mercados [4].

Por ende, FT-Platform es una plataforma que permitirá detectar vulnerabilidades conocidas para un desarrollo en un lenguaje determinado durante las etapas tempranas del ciclo de DevOps, esto con el fin de detectar cualquier vulnerabilidad antes de que llegue a producción, para así poder evitar tiempos de retraso, cascadas de vulnerabilidades y más problemas que pueden desprenderse de estos errores.

Para la creación de esta plataforma, se llegó a la conclusión de utilizar una arquitectura de microservicios, la cual consta de 4 funcionalidades principales: autenticación, gestión de pruebas SAST por proyectos, reportes y notificación de resultados. La arquitectura cuenta con 6 componentes principales, conformados por 4 microservicios: autenticación, pruebas, reportes y notificación, además de un componente con la función de API Gateway y otro actuando como FrontEnd. Tanto los microservicios como el API Gateway se construyeron en Python en su versión 3 y en el componente de FrontEnd se construyó con Angular en su versión 8. La infraestructura que soporta FT-Platform está a cargo de Docker, donde se desplegará en un servidor local perteneciente a la empresa FT FORENSIC TECHNOLOGY.

Tanto la arquitectura como la infraestructura fueron diseñadas conforme las necesidades de adaptabilidad y escalabilidad, al hacer que cada funcionalidad sea independiente de otras.

La metodología utilizada durante la construcción del proyecto se basó en la tradicional SCRUM, aunque adaptada a los tiempos y disponibilidades de la empresa, donde se tenía plena libertad en decisiones de diseño y donde se realizaban reuniones semanales con la persona encargada de la evaluación del proyecto por parte de la empresa. Esta metodología será explicada más adelante con más detalle.

Objetivos

Objetivo general:

Desarrollar un módulo escalable que integre parcialmente pruebas de seguridad estáticas (SAST), para la detección de vulnerabilidades durante la construcción de software, en la etapa de integración continua.

Objetivos específicos:

- Investigar las vulnerabilidades más explotadas en las aplicaciones, así como los errores de seguridad más frecuentes en el código, en la etapa de integración continua.
- Diseñar el módulo con una interfaz de usuario que presente en forma gráfica las alertas y errores encontrados por el módulo de seguridad que será construido.
- Implementar una conexión con el API de GitLab para realizar el reconocimiento de vulnerabilidades y errores de seguridad en el código.
- Construir un módulo de notificación que informe las vulnerabilidades y errores en el código detectados en la prueba.
- Integrar la autenticación con los servicios de versionamiento de código en nube de GitLab, GitHub y BitBucket, por medio del protocolo de autorización OAuth 2.0.
- Realizar pruebas funcionales del módulo desarrollado, utilizando una aplicación deliberadamente insegura (Juice-Shop del proyecto OWASP -Open Web Application Security Project-).

Marco Teórico

Para una correcta compresión de la finalidad del proyecto, es necesario explicar los conceptos relevantes que hay detrás de SAST y como es su implementación en DevOps. Primero se definirá el concepto de DevOps, DevOps es una cultura que se ha fortalecido entre las compañías, con la finalidad de hacer que la comunicación entre el equipo de desarrollo y el equipo de operaciones (despliegue e infraestructura) sea constante y automatizado, haciendo que el flujo de entregas sea continuo e implementando conceptos como la calidad del software y TDD (Test Driven Development), para facilitar la detección de errores en etapas de desarrollo tempranas y despliegue en producción frecuente, además de un código que cumpla con ciertas especificaciones de calidad que se requieran.

A pesar de los grandes avances en las metodologías de desarrollo y la inclusión de DevOps, la seguridad es una asignatura que suele dejarse para el final del ciclo de vida del desarrollo, aumentando significativamente los plazos de entrega y los costos de la corrección de las vulnerabilidades que se hallan en un sistema, ya que implementar una búsqueda, así como la solución a una vulnerabilidad en un entorno de producción puede llegar a ser complejo, porque personas con malas intenciones se pueden aprovechar vulnerabilidad, derivando en pérdidas de tiempo, dinero y esfuerzo. Es por ello que al pensar en los ciclos de vida del desarrollo con las metodologías ágiles se hace un poco tedioso integrar la seguridad, ya que en estas metodologías se prioriza la rapidez de las entregas, haciendo que tanto la calidad del código como la seguridad por cada release sea baja [5].

Con el paso del tiempo, la evolución de las necesidades de las empresas y de los clientes ha ido cambiando, es por ello que hoy en día se puede evidenciar un crecimiento en los productos y servicios web para suplir dichas necesidades. Sin embargo, con el crecimiento de los servicios online, también se han incrementado los defectos en el software, las vulnerabilidades y los ataques [6].

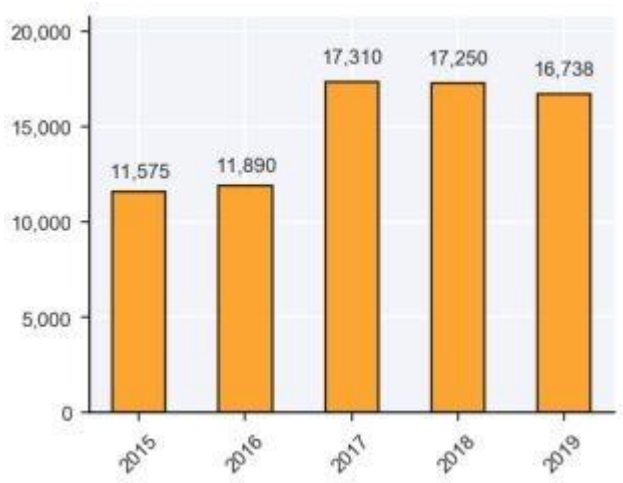


Imagen 1. *Número de vulnerabilidades descubiertas hasta 30/9 en los últimos 5 años [6].*

Dada la necesidad de incluir la seguridad en DevOps, en los últimos años surgió una solución para esta necesidad, llamada por algunos autores como SecDevOps y por otros como DevSecOps, que, en esencia, busca que la seguridad se implemente en todos los ciclos de DevOps, minimizando cada uno de los problemas previamente descritos logrando una sinergia entre la seguridad y DevOps. La definición que se adoptó para este proyecto es la definición de RedHat, donde definen que la seguridad es una responsabilidad compartida e integrada durante todo el proceso de DevOps, es por ello que se hace énfasis en crear una necesidad de implementar procesos de seguridad en los ciclos de DevOps [7].

DevSecOps consta de 4 fases de análisis y pruebas de seguridad, las cuales son: análisis de código estático (por sus siglas en inglés SAST), análisis de código dinámico (por sus siglas en inglés DAST), análisis de código interactivo (por sus siglas en inglés IAST) y análisis de código en tiempo de ejecución (por sus siglas en inglés RASP). Sin embargo, dado que la intención es crear una plataforma modular y escalable, además del tiempo (6 meses) y el número de personas encargadas (1 persona) que se tiene para el desarrollo de las funcionalidades descritas en este documento, solo se tiene en cuenta una pequeña parte de la fase SAST, donde se crea una herramienta de análisis de código estático.

SAST es una técnica usada para detectar vulnerabilidades a nivel de código, ya que implementa un conjunto de herramientas (escáneres) que detectan errores de programación por medio de patrones y flujos, eliminando así la necesidad de realizar procesos de compilación que podrían llegar a extender los tiempos de análisis y solución de estas [5][8]. Es por ello que SAST se implementa en la etapa de integración continua de DevOps, ya que permite supervisar y controlar la construcción de código en su ciclo de vida.

Para un mejor entendimiento de la construcción de la aplicación, es necesario definir que la metodología usada para el desarrollo de los distintos componentes de FT-Platform fue una metodología ágil, que se basa en hacer entregas periódicas de funcionalidades pequeñas (también llamadas release), donde se tienen una constante comunicación con el cliente (Product Owner) y se tienen tiempos de entrega cortos entre funcionalidades. Para este proyecto se utilizó SCRUM, aunque con unos cambios, donde los tiempos de entrega se acomodaron a la disponibilidad de la empresa, ya que una sola persona era la encargada de construir todas las funcionalidades. La arquitectura que se utilizó para la construcción de las interfaces de programación de aplicaciones (por sus siglas en inglés API's) fue la de microservicios, que consta de modularizar una aplicación de tal manera que sus componentes sean más pequeños e independientes entre sí, los cuales tienen la capacidad de funcionar en conjunto para llevar a cabo cualquier funcionalidad. Esta arquitectura tiene la ventaja de que es muy adaptativa a los cambios, además de que se beneficia del desarrollo con metodologías ágiles y DevOps, donde todos los microservicios eran manejados por un único componente de API Gateway, el cual se encarga de unificar y exponer los microservicios con el fin de que sean consumidos por un componente visual del lado del cliente o también llamado Front End.

Metodología

Investigación.

La investigación se llevó a cabo tomando como base fuentes de información especializadas en los temas de seguridad (proyecto OWASP, RedHat, VulnDB, CVE, etc.), estándares de autenticación y manejo de JWT (IETF), así como los diversos conceptos asociados a DevOps, DevSecOps, arquitecturas de desarrollo y las metodologías ágiles (IEEE, Researchgate, Universidades, etc.), donde se seleccionaron artículos, sitios, revistas y trabajos que tuvieran mucha relación con el tema en cuestión. Como en la mayoría de los temas investigados tenían aplicabilidad sobre el proyecto, se buscó también información en fuentes como Google y StackOverFlow, donde se encontraban ejemplos y guías de ciertos temas, que más tarde serían aplicados sobre el proyecto.

Construcción de microservicios.

Para la construcción de cada microservicio, se separaron las funcionalidades más básicas que se querían tratar, las cuales son la autenticación, el manejo y pruebas SAST de proyectos y la notificación de los resultados de las pruebas sobre los proyectos. Una vez separadas estas funcionalidades, se decidió utilizar el lenguaje de programación Python con el framework Flask, ya que se consideró que este lenguaje de programación sería el ideal gracias a su compatibilidad con los diversos motores y escáneres de vulnerabilidades.

En el lado del cliente, se decidió utilizar Angular en su versión más reciente hasta la fecha (versión 8), ya que es la librería de la que se tiene más conocimiento y con la que mejor se desempeña a la hora de construir las funcionalidades.

Para la base de datos se utilizó MariaDB, gracias a que es Open Source y con una gran cantidad de documentación presente para ésta (documentación de MySQL), así como el ORM disponible para python como lo es SQLAlchemy y para las pruebas de la aplicación se utilizó la librería Unittest de Python.

Módulo de Autenticación.

En la plataforma se usa el flujo de authorization Code Grant de Oauth 2.0 [9] de los servicios de GitHub, GitLab y BitBucket, para poder registrarse e iniciar sesión en la plataforma. Sin embargo, el manejo de la sesión y los permisos se hace con el uso de JWT (Json Web Token), el cual es manejado de acuerdo a los estándares provistos por Internet Engineering Task Force (IETF) [10]. Sin embargo, para el manejo de información sensible y en aras de mejorar la experiencia de usuario, se propone a futuro implementar el uso de cookies en conjunto con JWT, para así manejar de forma más eficiente la información y crear una mejor experiencia de usuario.

Durante la fase de implementación de este módulo, se estudiaron varias opciones para el manejo de autenticación y autorización, una de ellas fue Platform-Agnostic Security Tokens (PASETO) [11], sin embargo, al ser muy “reciente” no se realizó la implementación sobre el proyecto debido a que no es estable.

Módulo de manejo y pruebas SAST de proyectos.
En la plataforma se implementó un módulo que gestiona la importación del repositorio y la prueba sobre el repositorio importado, en este módulo se realiza la comprobación de la autenticación y autorización del usuario para verificar que pueda realizar acciones sólo sobre los proyectos a los que tiene permiso realizar acciones, además de entregar los resultados de las pruebas realizadas sobre el proyecto.

Módulo de notificación.
FT-Platform contiene un módulo de notificación el cual se activa cuando hay una petición sobre los endpoints que están disponibles. De acuerdo con la petición, se realiza la acción de notificar por medio de correo electrónico al usuario que el proyecto ha finalizado la fase de análisis y que se encuentran disponibles los resultados. Este módulo también es el encargado de realizar la notificación a la cuenta del usuario y se encargará de alertar al módulo gráfico (por medio del API Gateway) donde se encuentran los resultados de la prueba disponibles para ser vistos por el usuario.

Base de datos.
La plataforma requirió de una base de datos que almacene tanto información del usuario, como de los proyectos que han sido importados, es por ello que se creó una base de datos relacional con 9 tablas para el almacenamiento de esta información. La base de datos se implementó con MariaDB, los respectivos modelos se manejaron con la herramienta SQLalchemy (Python ORM) por parte de cada microservicio.

Interfaz gráfica.
Para la plataforma se implementó una interfaz gráfica sencilla que solo abarca la funcionalidad propuesta, ya que no es el objetivo del trabajo propuesto, sin embargo, en la construcción de cada funcionalidad se utiliza el patrón Locate, Identify, Flat, Try to be Dry (LIFT) recomendado en la documentación de angular [12]. Este módulo es el que contiene parte de la funcionalidad de autenticación de Oauth 2.0 (Obtención del código de acceso) y vistas necesarias para llevar a cabo las diversas funcionalidades.

Resultados y análisis

Al llevar a cabo la construcción de los distintos módulos de la aplicación, se logra evidenciar varios resultados interesantes, los cuales serán descritos a continuación:

En la construcción de los microservicios y el API Gateway con las funcionalidades básicas propuestas para este proyecto (autenticación, manejo de proyectos y pruebas SAST, reportes y notificación), se decidió recrear una de arquitectura de puertos y adaptadores utilizando Python, tal como se muestra en la **imagen 2**, ya que esta arquitectura al ser modular permite un bajo acoplamiento y alta cohesión. Se distribuyó de tal forma que el módulo 'Models' contiene los modelos de las entidades que se usan en el microservicio, el módulo 'Database' contiene toda la capa de persistencia y la lógica asociadas a ésta, el módulo 'Domain' contiene toda la lógica y validaciones, además de ser un conector entre el módulo de 'Service' y 'Database', el módulo service contiene la definición de los endpoints, así como los métodos asociados a estos, los cuales tiene la finalidad de recibir los datos y realizar un preprocesamiento para después ser enviados al respectivo módulo de 'Domain'. El módulo de Test es donde están contenidas todas las pruebas unitarias de código y la carpeta de 'env' es donde estará el entorno virtual del proyecto, el cual contendrá todas las librerías y configuraciones propias del lenguaje (Python).

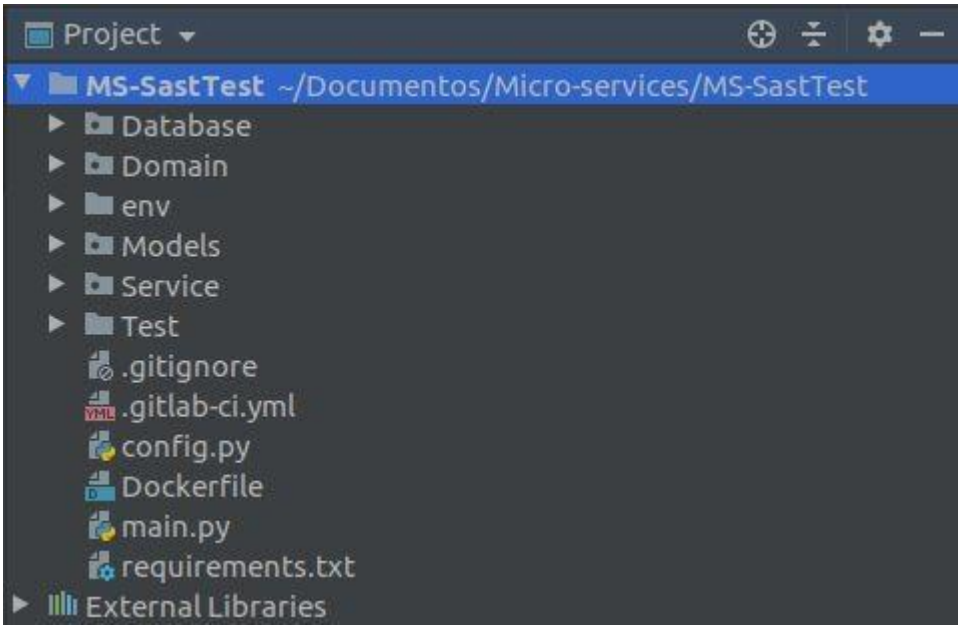


Imagen 2. *Arquitectura del microservicio.*

En la construcción del API Gateway, se realizaron los llamados de todos los microservicios por medio de sus endpoints para después ser presentados y consumidos. En los diferentes endpoints expuestos por el API Gateway, se realiza la autenticación, el listado de los proyectos asociados a un usuario, la importación del proyecto con su prueba SAST, el listado de proyectos a los que se le realizó la prueba y las vulnerabilidades encontradas. Todo esto se puede observar en las imágenes: **Imagen 3**, **Imagen 4**, **Imagen 5**, **Imagen 6** e **Imagen 7**.

Imagen 3. *Autenticación con OAuth 2.0 (GitHub).*

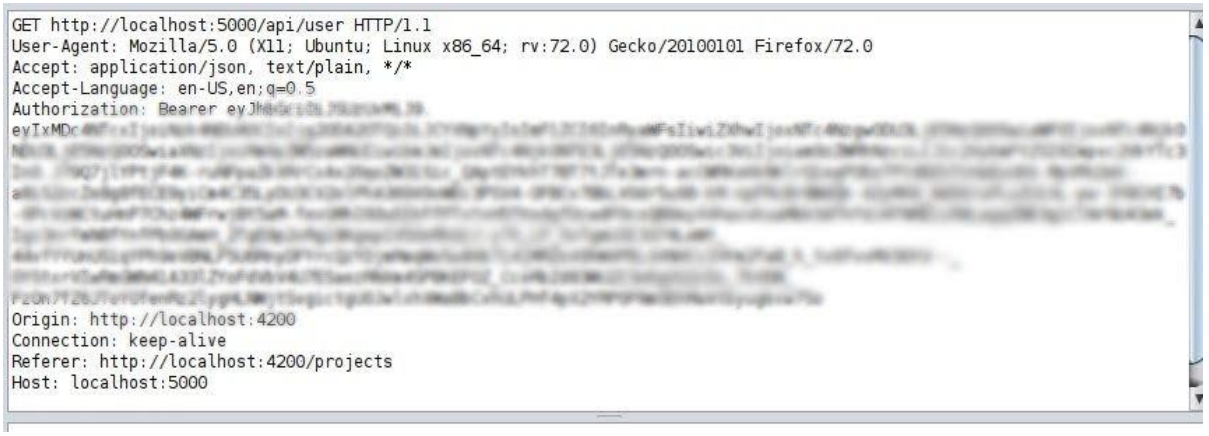


Imagen 4. *Información del usuario autenticado con listado de proyectos.*

Imagen 5. *Importación del proyecto.*

Imagen 6. *Proyectos examinados con escáneres de SAST.*

Imagen 7. *Resultados de un proyecto examinado.*

En el proceso de registro en FT-Platform se crearon dos formas de registrarse, el cual es por medio de un registro manual o por medio del protocolo OAuth 2.0. En el registro manual, solo es ingresar los datos presentados en el lado izquierdo visible en la **imagen 8**, mientras que para realizar un registro con OAuth 2.0 se debe seleccionar un de los botones presentados en la sección 'Ingresa con'. Para el servicio de autenticación se tienen diseñados cambios a futuro, donde se implementará una plataforma llamada Istio [13], que tiene dentro de sus servicios ayudar en la autenticación y comprobación de credenciales tanto de microservicios como de usuarios.

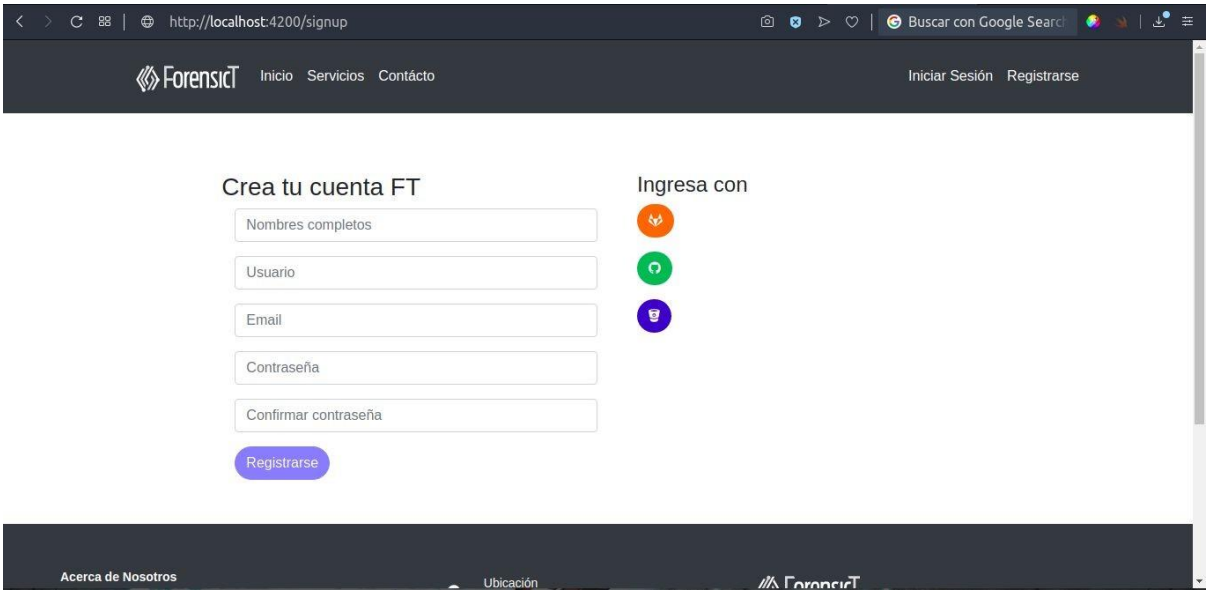


***Imagen 8**. Sección de registro de clientes.*

Al realizar clic sobre el botón verde que representa el servicio de versionamiento de GitHub, se redirige al usuario al inicio de sesión de GitHub, el cual puede verse en la **imagen 9**.

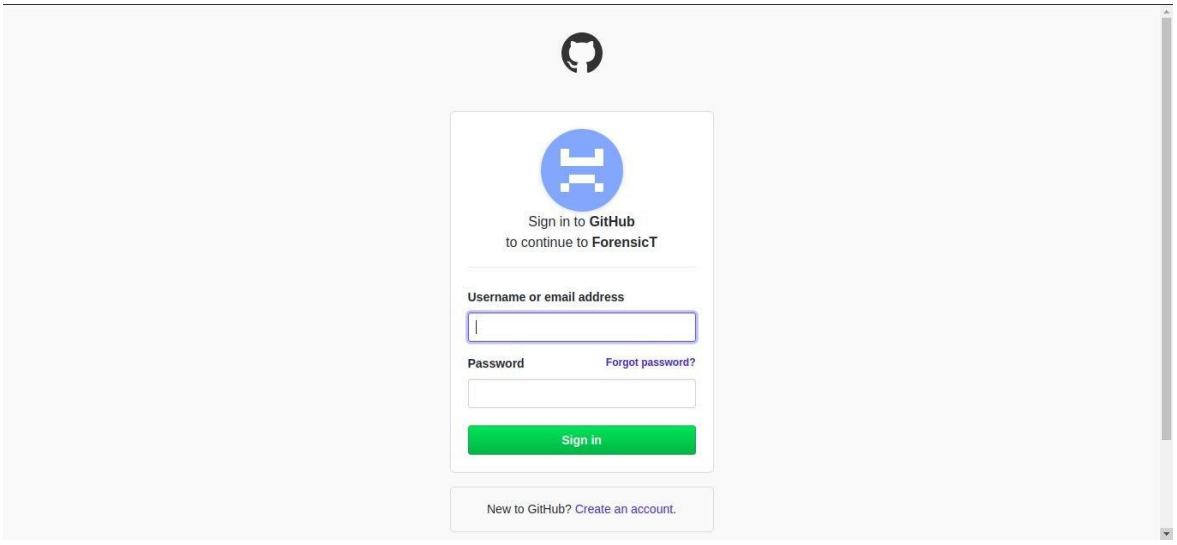


***Imagen 9.** Inicio de sesión de GitHub.*

Una vez ingresados los datos de un usuario válido en el inicio de sesión de GitHub, este redirigirá de nuevo a FT-Plaform, donde a su vez se podrá ver todos los repositorios que el usuario tiene disponibles para examinar, cómo se puede observar en la **imagen 10**.

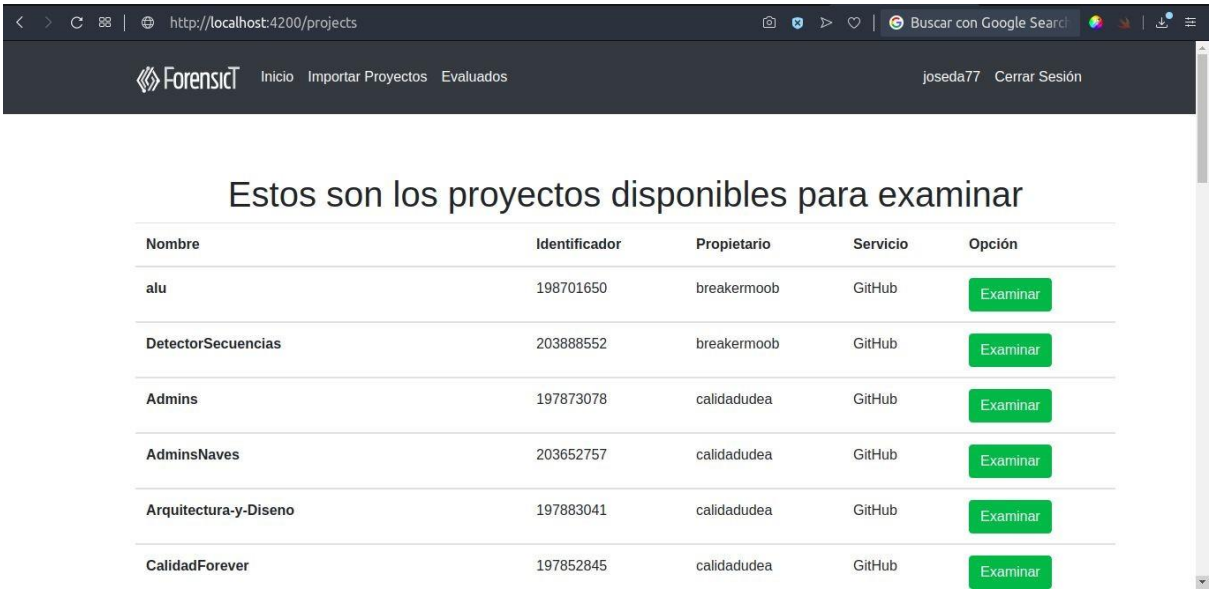


Imagen 10. *Repositorios/Proyectos disponibles para examinar del usuario.*

Si se le hace clic en el botón examinar (véase **imagen 10**), se mostrará un aviso que indicará si el proyecto está en proceso de importación/análisis (véase **imagen 11**) o si ya fue importado/analizado (véase **imagen 12**).

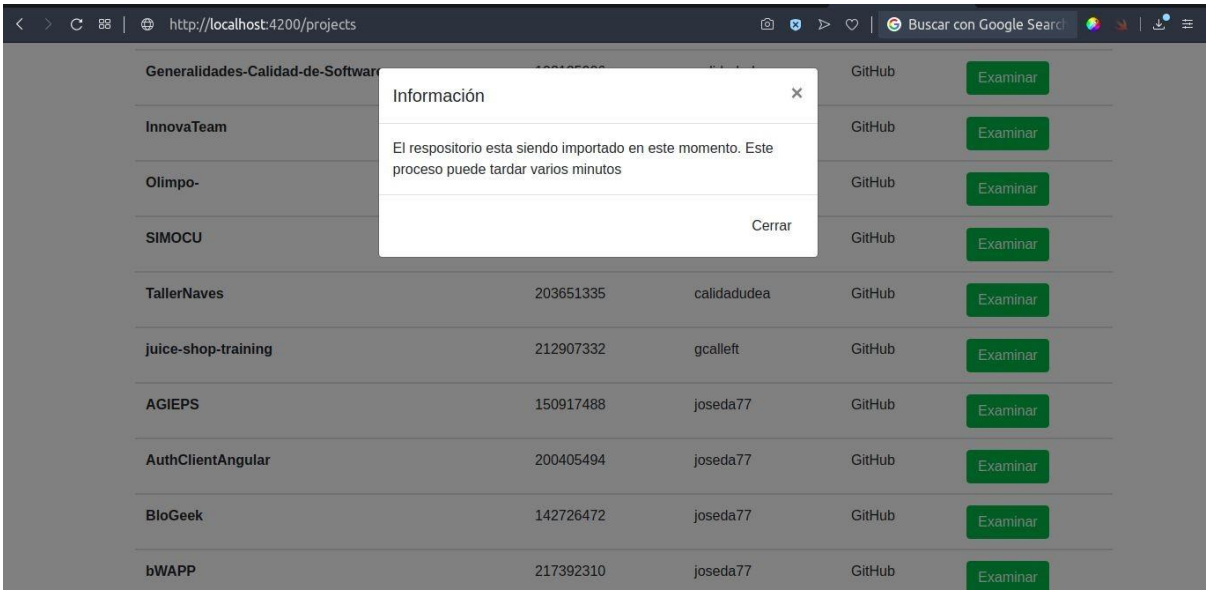


Imagen 11. *El repositorio está siendo importado.*

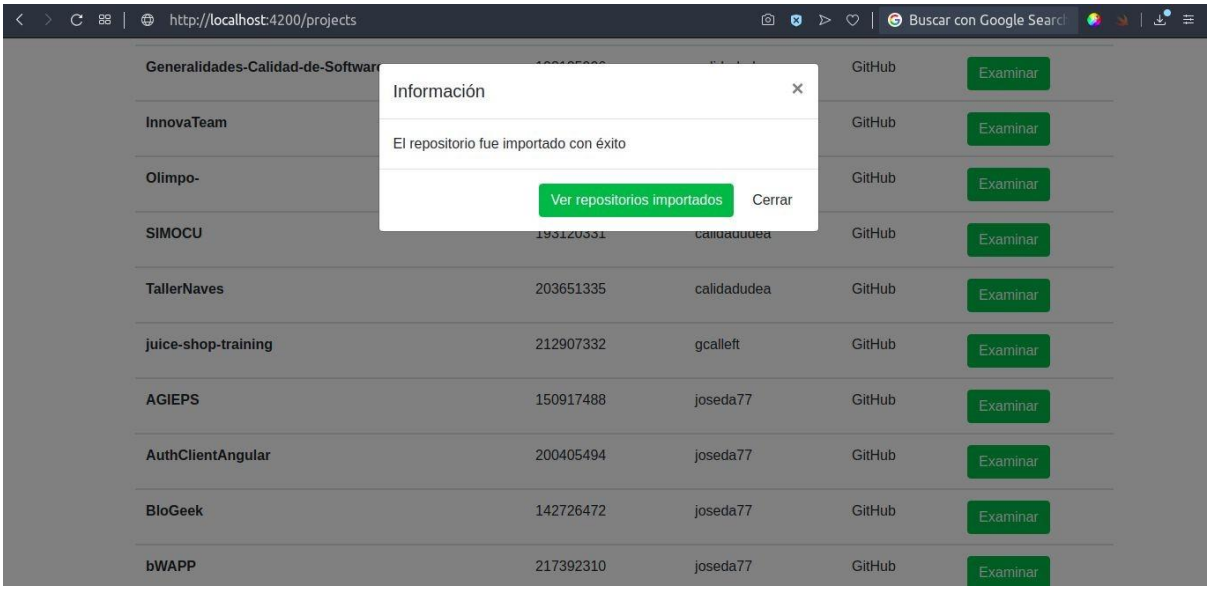


Imagen 12. *El repositorio fue importado exitosamente.*

Al realizar clic sobre el botón ‘ver repositorios’ importados (véase **imagen 12**), el usuario será redirigido a la sección de proyectos importados/evaluados, donde se listan todos los proyectos que el usuario le dio a examinar, cómo se puede observar en la **imagen 13**. En este proceso de realizar la prueba SAST sobre un proyecto como Juice-Shop, GitLab deja encolada esta petición para después ser procesada, es decir que, si se hacen múltiples peticiones de otros proyectos en un corto periodo de tiempo, la aplicación solo podrá acceder a los resultados una vez se hayan completado las solicitudes previas. El tiempo de espera puede variar de acuerdo al tamaño de proyecto y número de solicitudes, lo cual puede ser tedioso para un usuario una espera tan larga.

La solución que se propone para evitar este inconveniente es usar motores/escáner de análisis de vulnerabilidades en el código como lo son Bandit (Python), PMD (Java), Progpilot (Php), Flawfinder (C y C++), entre otros., que puedan llevar a cabo esta tarea paralelizando los procesos, para así mejorar los tiempos de respuesta de laaplicación.

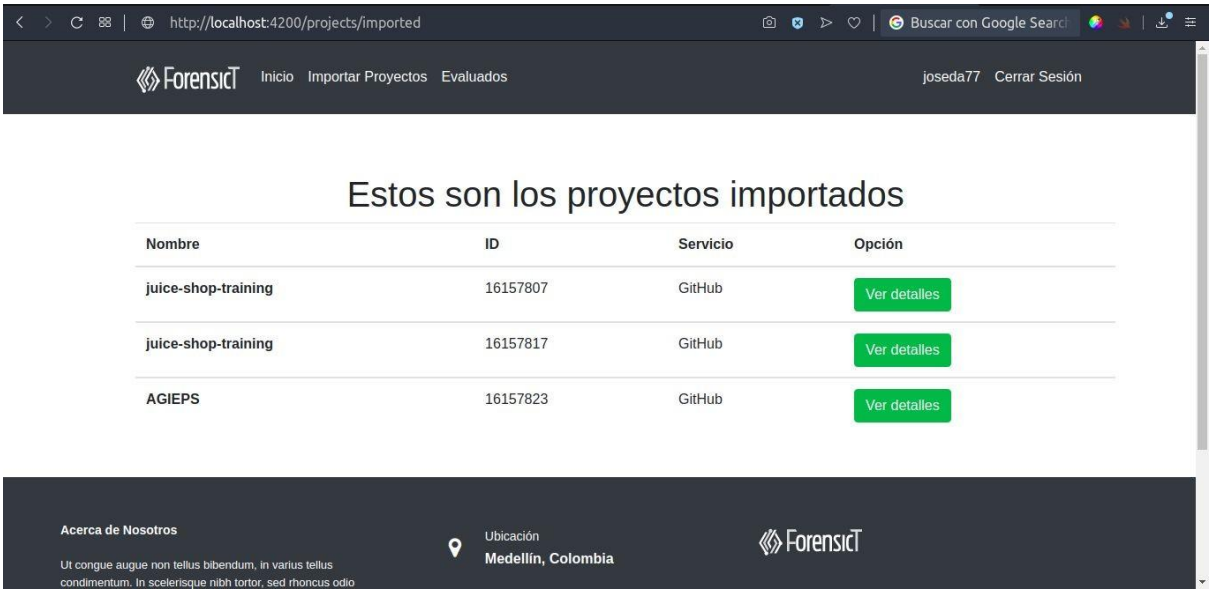


Imagen 13. *Proyectos/Repositorios examinados.*

Cuando el usuario de clic en el botón de ver detalles (véase **imagen 13**), éste mostrará un mensaje si la prueba de SAST aún está en proceso (véase **imagen 14**) o redirigirá a la vista de los resultados encontrados (véase **imagen 15**).

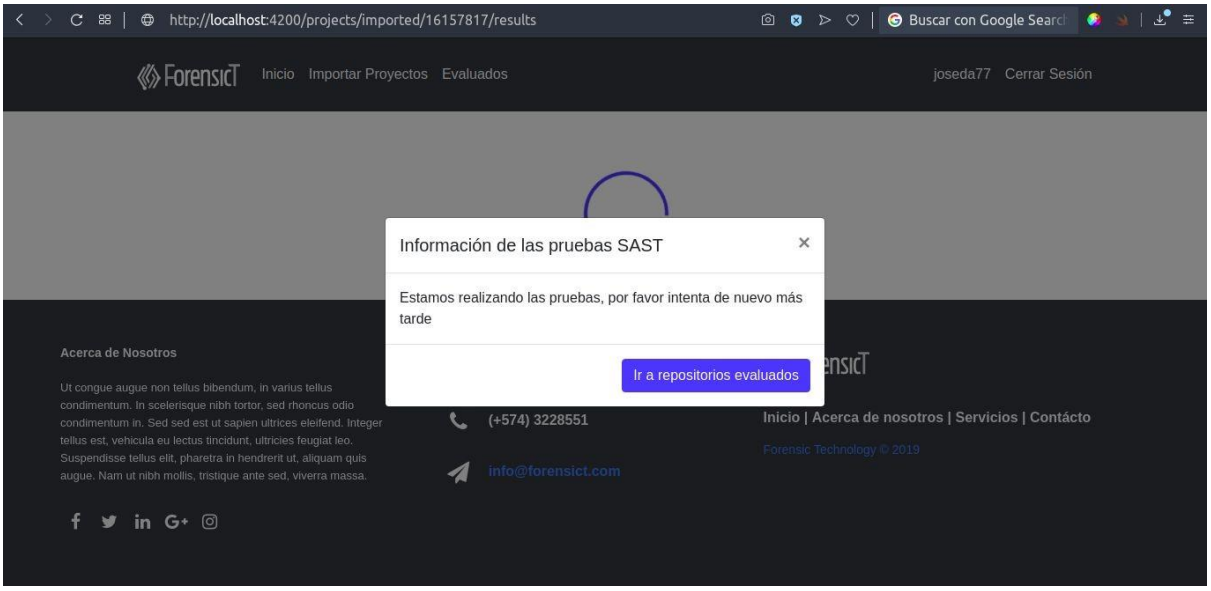


Imagen 14. *Mensaje que indica que los resultados aún no están listos.*

En el proceso de obtención de los resultados de la prueba, se evidenció problemas con la fuente de los recursos de pruebas SAST (GitLab), ya que con su última actualización hasta la fecha (12.5), se nota que los recursos cambiaron sus URI's, por ende, en su momento generó múltiples errores a la hora de hacer peticiones a los recursos de SAST, generando un error de acceso.

Al terminar de realizar la prueba de SAST, FT-Platform enviará un email al usuario notificando que los resultados de las pruebas están disponibles, donde se le informa al usuario el número del total de vulnerabilidades encontradas, tal como se puede apreciar en la **imagen 15**.

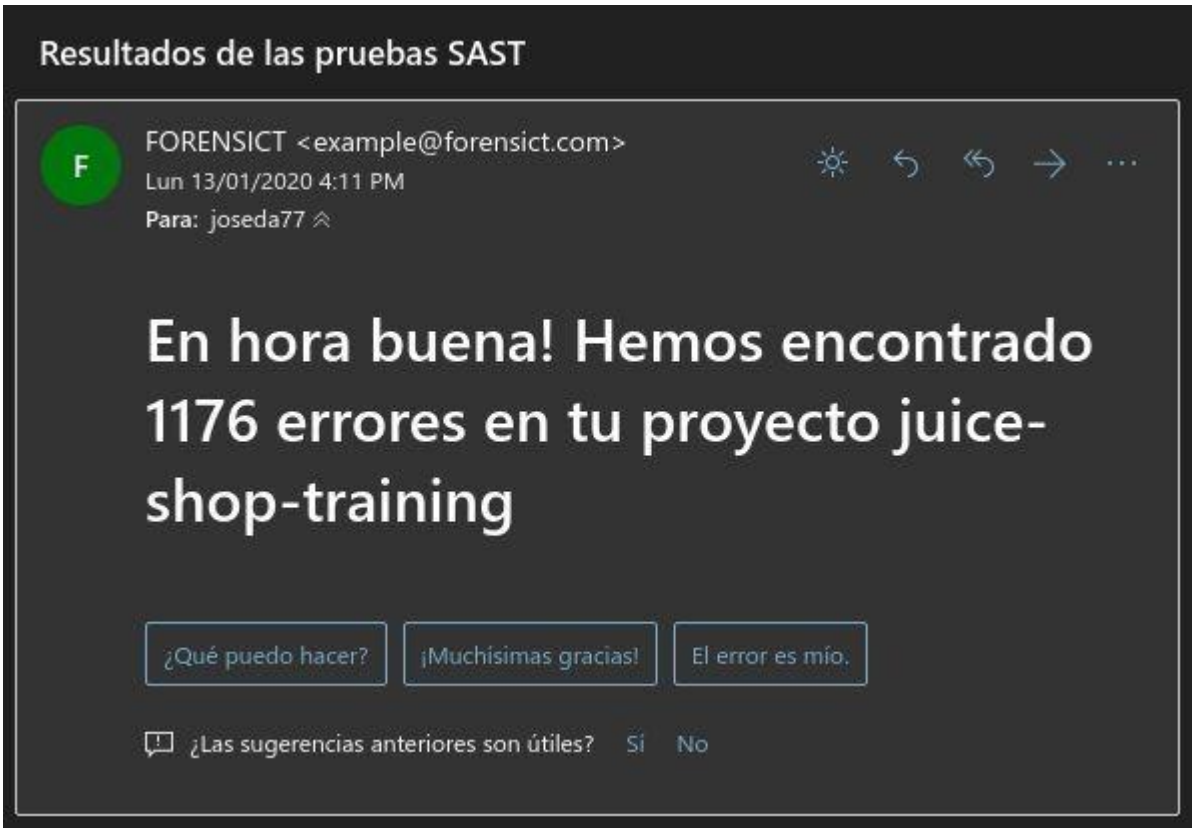


***Imagen 15**. Notificación vía email de la prueba (Caso de ejemplo en Hotmail).*

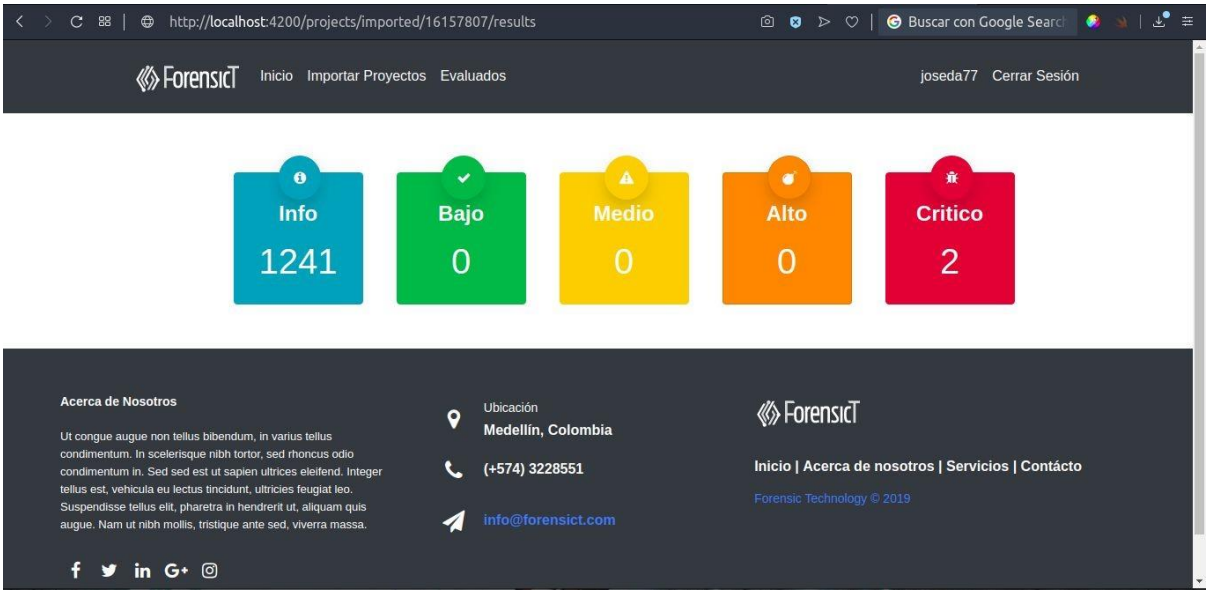
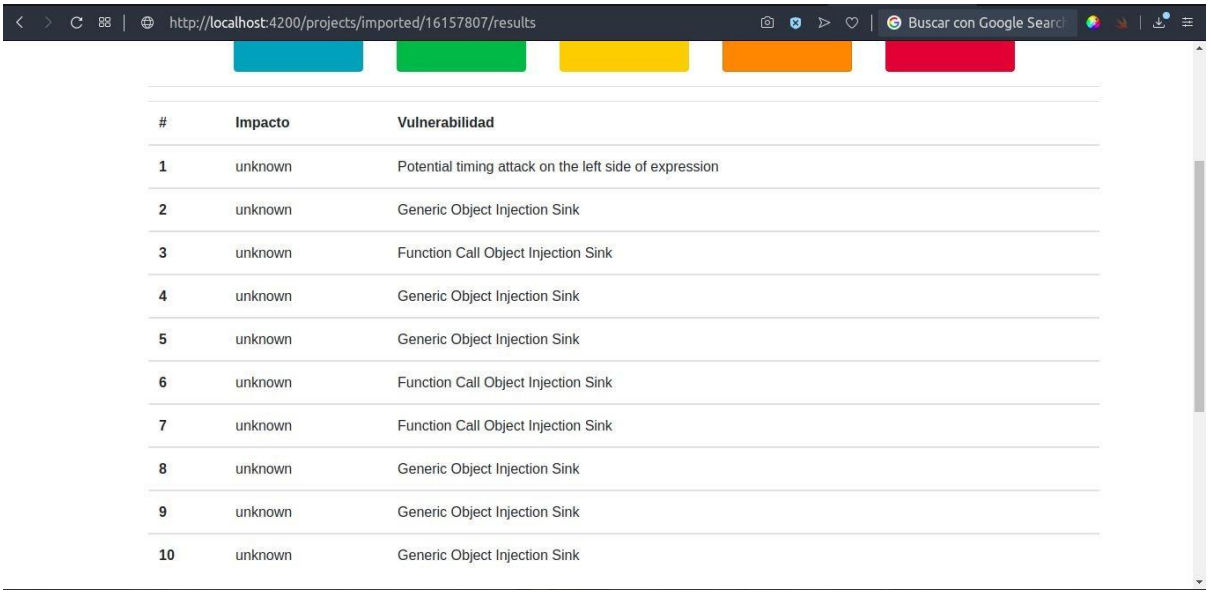


***Imagen 16.** Lista de resultados de acuerdo con su severidad.*

Al realizar clic sobre cualquier conjunto de vulnerabilidades que se pueden ver en la **imagen 16**, cada uno de estos despliegan una lista con todas las vulnerabilidades clasificadas con esa severidad, así como se puede observar en la **imagen 17**. Donde se alcanzó a observar que los resultados que el módulo de SAST de GitLab no otorga resultados muy precisos, ya que se observaron vulnerabilidades mal clasificadas o falsos positivos identificadas con la ayuda de personas expertas en seguridad de FT ForensicT. Además, al hacer una comparativa con una versión del proyecto Juice-Shop con vulnerabilidades solucionadas y el proyecto original, se evidenció que no había mucho cambio en los resultados.



#	Impacto	Vulnerabilidad
1	unknown	Potential timing attack on the left side of expression
2	unknown	Generic Object Injection Sink
3	unknown	Function Call Object Injection Sink
4	unknown	Generic Object Injection Sink
5	unknown	Generic Object Injection Sink
6	unknown	Function Call Object Injection Sink
7	unknown	Function Call Object Injection Sink
8	unknown	Generic Object Injection Sink
9	unknown	Generic Object Injection Sink
10	unknown	Generic Object Injection Sink

Imagen 17. *Lista de vulnerabilidades encontradas, clasificadas de acuerdo a su severidad.*

Si se realiza clic sobre cualquier elemento de la lista, se mostrará un recuadro con información más detallada la vulnerabilidad, tal como se muestra en la **imagen 18**.

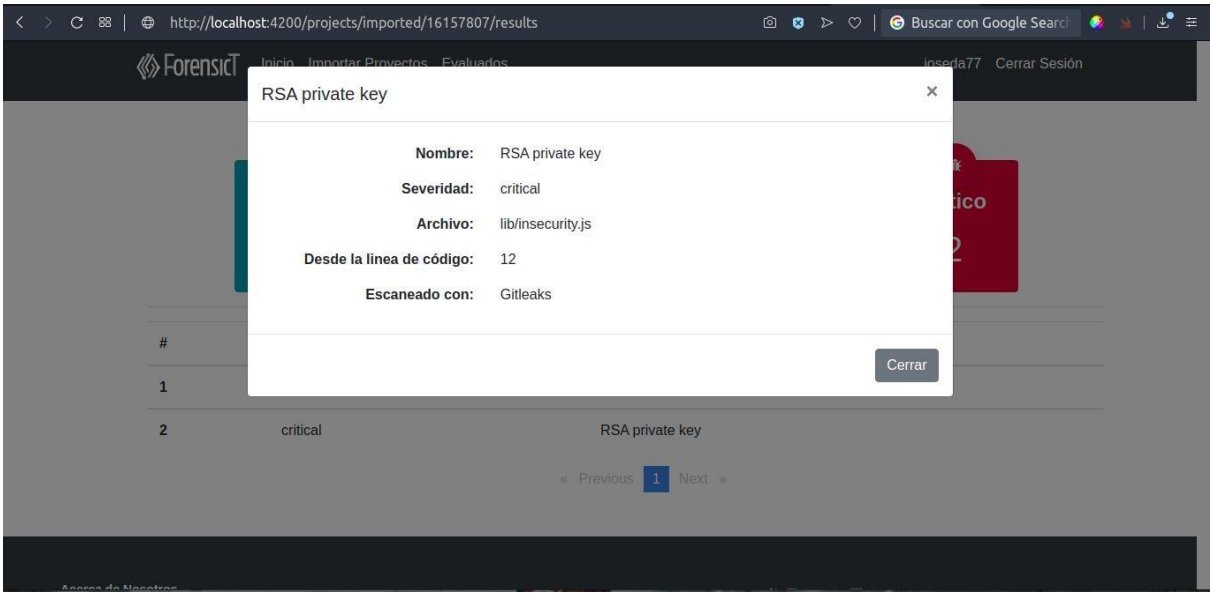


Imagen 18*. Detalles de la vulnerabilidad.*

Conclusiones

En el desarrollo de FT-Platform, se lograron evidenciar resultados que permitirán definir el rumbo del proyecto, un ejemplo de ello es que se deben evitar las dependencias de API's y servicios de terceros, ya que un cambio en estos servicios podría significar una incorrecta operabilidad en FT-Platform, tanto con errores o restricciones que éstos interpongan. Para ello se recomienda trabajar y usar herramientas open source o propias, ya que esto evita la dependencia de servicios tercerizados.

Tras realizar diversas pruebas sobre varios proyectos como Juice-Shop, bwapp [14], OWASP WEBGoat [15], se encuentra que los resultados no son 100% precisos, ya que un gran porcentaje se clasifican erradamente o existen falsos positivos. Este problema probablemente se deba a la configuración de los escáneres utilizados para esta labor. Es por ello que se recomienda siempre revisar los resultados con un experto de seguridad para evaluar la pertinencia de los datos otorgados, ya que una falla de esta índole puede afectar las actividades core de un negocio o de una empresa, provocando grandes pérdidas en los usuarios.

Al ser DevOps un tema relativamente nuevo, tiene constantes cambios mientras evoluciona, provocando que las herramientas y documentación asociada a este tema también sufra cambios, es por ello que se debe estar documentando frecuentemente de las novedades y los cambios en la cultura para no quedar atrás fácilmente.

El manejo de la autenticación y la autorización es un tema que aún está en desarrollo cuando se habla de microservicios, ya que hoy en día se encuentran diversos problemas en la implementación desde tiempos de respuesta largos hasta exposición de datos sensibles, es decir, el sacrificio de uno por mejorar el otro. Es aquí donde el uso de herramientas como librerías o frameworks debe hacerse con cuidado, pues muchas de estas manejan información sensible de los usuarios y pueden contener ciertas vulnerabilidades que pueden ser explotadas. También puede darse el caso de un mal uso por parte del desarrollador que provoque fallas en el funcionamiento de una aplicación.

Referencias Bibliográficas

[1] Kavis, M. (2014). 11 Common DevOps Bottlenecks. [online] Forbes.com. Available at: https://www.forbes.com/sites/mikekavis/2014/12/18/11-common-devops-bottlenecks/#201c40657737 [Accessed 14 Jan. 2020].

[2] Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N. (2016). DevOps. IEEE Software, 33(3), pp.94-100.

[3] Mohan, V. and Othmane, L. (2016). SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps. 2016 11th International Conference on Availability, Reliability and Security (ARES).

[4] The DevSecOps Global Skills Survey. (2020). 1st ed. [ebook] Available at: http://www.veracode.com/blog [Accessed 14 Jan. 2020].

[5] Iriz Ricote, S. (2020). Seguridad en el ciclo de vida del desarrollo del software. DevSecOps. Maestría. Universitat Oberta de Catalunya.

[6] VulnDB (2020). Vulnerability QuickView Report. VulnDB, pp.9,10,11.

[7] Redhat.com. (2020). ¿Qué es DevSecOps?. [online] Available at: https://www.redhat.com/es/topics/devops/what-is-devsecops [Accessed 14 Jan. 2020].

[8] A. D. Brucker and U. Sodan. (2014). Deploying static application security testing on a large scale. In S. Katzenbeisser, V. Lotz, and E. Weippl, editors, GI Sicherheit2014, volume 228 of Lecture Notes in Informatics, pages 91–101.

[9] Tools.ietf.org. (2020). RFC 6749 - The OAuth 2.0 Authorization Framework. [online] Available at: https://tools.ietf.org/html/rfc6749#page-31 [Accessed 14 Jan. 2020].

[10] Tools.ietf.org. (2020). RFC 7519 - JSON Web Token (JWT). [online] Available at: https://tools.ietf.org/html/rfc7519 [Accessed 14 Jan. 2020].

[11] Paseto.io. (2020). PASETO. [online] Available at: https://paseto.io/ [Accessed 14 Jan. 2020].

[12] Angular.io. (2020). Angular. [online] Available at: https://angular.io/guide/styleguide#lift [Accessed 14 Jan. 2020].

[13] Istio. (2020). Security. [online] Available at: https://istio.io/docs/concepts/security/ [Accessed 14 Jan. 2020].

[14] Itsecgames.com. (2020). itsecgames.com. [online] Available at: http://www.itsecgames.com/ [Accessed 14 Jan. 2020].

[15] Owasp.org. (2020). OWASP WebGoat. [online] Available at: https://www2.owasp.org/www-project-webgoat/ [Accessed 14 Jan. 2020].