Universidad Veracruzana

Proyecto Final

Programación Segura

Aplicación Web “Administración de Servidores”.

Integrantes:

Aldair Garrido Paz

Carlos David Uscanga Heredia

Edgar Rafael Moro Martínez

Académico: Héctor Xavier Limón Riaño

Xalapa, Ver. A 19 de Julio del año 2020

Contenido

Introducción 2
Propósito del proyecto 3
Requerimientos 4
Modelado de amenazas 7
Diagrama de Flujos de Datos (DFD) 0 7
Diagrama de Flujos de Datos (DFD) 1 8
Tablas con análisis y medidas de mitigación 9
Diagrama de Despliegue 13
Conclusiones 14
Bibliografía 15

Contenido de Ilustraciones

Ilustración 1 DFD 0 7
Ilustración 2 DFD 1 8
Ilustración 3 Diagrama de Despliegue 13

Introducción

Actualmente la seguridad, es de suma importancia en el desarrollo de sistemas, debido a la evolución de las tecnologías de la información y el crecimiento de Internet. Por tal motivo se requiere conocer las principales amenazas, riesgos y vulnerabilidades en el desarrollo de sistemas, aplicaciones, etc. Además, conocer e implementar técnicas de mitigación antes estos ataques, así como el desarrollo y análisis seguro de código.

Estos conocimientos se adquieren en la Experiencia Educativa Programación Segura, de la licenciatura en Redes y Servicios de Computo de la Universidad Veracruzana.

En este presente documento, se describirá el proyecto de la EE Programación Segura, el cual consiste en el desarrollo de la plataforma/aplicación web de monitoreo de servidores, tiene la finalidad de administrar los servidores de manera segura, utilizando tecnologías web.

La plataforma permite a administradores de servidores monitorizar servidores asociados y crear conexiones remotas a los mismos, todo a partir de una interfaz Web.

Esta aplicación Web, se conforma, de dos administradores, uno global y el de servidores, cada uno tienes distintas funcionalidades. El administrador global, se encarga de registrar a los administradores y asociar servidor especifico. Así, como eliminar y actualizar los servidores y a los administradores. Referente a los administradores de servidores se encargarán de controlar y monitorear su servidor asociado, el cual mostrara el porcentaje de uso del procesador, memoria RAM y de disco duro.

Además, debe incluir una notificación que se mandara vía telegram, por si algún servidor presenta una caída inesperada y una conexión a través de la misma interfaz web.

Para lograr esto, se implementará los requerimientos necesarios, como el framework de Django y el uso de la programación segura, utilizando tecnologías vistas en esta EE, como el cifrado, hashing, código seguro, variables de entorno, sesiones, tls etc. Además de utilizar servicios Web, con Django Rest Framework y Docker.

Otro punto, que se mencionara es el diseño, modelado de amenazas de esta plataforma, a través de los Diagramas de Flujos (DFD), tablas y diagramas de despliegue.

Propósito del proyecto

El proyecto final de la Experiencia Educativa Programación Segura, consiste en desarrollar una plataforma que permita la administración segura de servidores a través de tecnologías web. La plataforma permite a administradores de servidores monitorizar servidores asociados y crear conexiones remotas a los mismos, todo a partir de una interfaz Web.

En este sistema existen los siguientes roles y funcionalidad asociada:

Administrador global:

- Registra, actualiza y elimina servidores de la infraestructura
- Registrar, actualizar y eliminar administradores de servidores
- Asociar servidores a administradores (un servidor sólo puede ser administrado por un administrador)

Administradores de servidor:

- Iniciar sesión (sus credenciales son provistas por el administrador global por fuera del sistema, así como cualquier otra información de seguridad que necesite el administrador)
- Monitorizar servidor: por cada servidor asociado el administrador puede ver:
    - Porcentaje de uso de procesador
    - Porcentaje de uso de memoria
    - Porcentaje de uso de disco
    - Puntos extra: mandar una notificación por fuera del sistema (a telegram por ejemplo) si algún servidor cae
    - Conectarse al servidor: a través de la misma interfaz web (no usar ssh por fuera) se puede abrir una terminal hacía el servidor

Restricciones de diseño:

- Toda la interfaz de usuario será a través de páginas web
- Cada servidor debe contar con un servicio web que provea la información de monitorización

Requerimientos

A continuación, se describen algunos requerimientos, que se consideraron en la implementación de este proyecto.

Requerimientos de seguridad:

- Análisis estático de código

Consiste hacer revisiones de código para encontrar, bugs, conocer si se está realizando el código con buena calidad, hacer estimaciones del tiempo, en que se corregiría la seguridad. Logrando detectar de manera temprana los problemas de seguridad y corregirlos. Para lograr el análisis se puede utilizar el servidor Sonar qube, donde se envía el código para que se analice y muestre los resultados en su interfaz web. Con ayuda de un cliente como Sonar Scanner, se logra obtener el reporte de los resultados obtenidos.

- Variables de entorno.

Son variables dinámicas, son valores que pertenecen al contexto del proceso (programas en ejecución) y están asociadas a las variables que se crearon. Una forma de usar variables de entorno es cuando separamos el código de su configuración. Por ejemplo, utilizando un archivo. env y dentro de el se ponen las variables de entorno. Para el manejo de secretos, se puede cifrar el archivo. env con ccrypt, para cifrar y descifrar con técnicas simétricas de criptografía.

- Cifrado

El cifrado es una técnica que permite que la información cambie de representación para que alguien, que quiera interceptar la información, no vea lo que se hizo. Existe el cifrado simétrico, si lo cifro con una llave, se debe usar la misma para descifrar y el cifrado asimétrico utiliza una llave pública y privada para cifrar o descifrar.

- Hashing

Los hash o funciones de resumen son algoritmos que consiguen crear a partir de una entrada (ya sea un texto, una contraseña o un archivo) una salida alfanumérica de longitud normalmente fija. Se puede agregar un salt, para hacer más seguro el hash.

- Sesiones

Son un mecanismo para almacenar datos del usuario, que interactúa con el sistema web. Permiten destruir inactividad o un tiempo de inicio de sesión.

- CSFR

Un ataque de falsificación de solicitudes entre sitios (CSRF) se produce cuando un atacante puede hacer que el navegador de un victima envíe una solicitud HTTP a otro sitio web. Django tiene protección incorporada contra la mayoría de los tipos de ataques CSRF, siempre que se haya habilitado y utilizado cuando sea apropiado.

- TLS / HTTPS

Protocolos para hacer conexiones seguras, se entiende que el emisor y receptor su conexión pueda tener confianza de que se están comunicando con entidades adecuadas. Ya que nos brindan autenticación.

Tecnologías utilizadas:

- Django

De acuerdo con Django Software Foundation (s.f.) Django es un marco web Python de alto nivel que fomenta el desarrollo rápido, el diseño limpio y pragmático. Es gratis y de código abierto.

Entre las características de Django se encuentran:

• Django fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible.

• Django incluye docenas de extras que puede usar para manejar tareas comunes de desarrollo web. Django se encarga de la autenticación del usuario, la administración de contenido, los mapas del sitio, los canales RSS y muchas más tareas, desde el primer momento.

• Django se toma muy en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes, como la inyección de SQL, las secuencias de comandos entre sitios, la falsificación de solicitudes entre sitios y el secuestro de clics. Su sistema de autenticación de usuario proporciona una forma segura de administrar cuentas de usuario y contraseñas.

- Django REST framework.

Django REST framework es un kit de herramientas potente y flexible para crear API web.

Algunas razones por las que puede querer usar el marco REST:

- La API navegable web es una gran victoria de usabilidad para sus desarrolladores.
- Personalizable hasta el final: solo usando las vistas, basadas en funciones regulares.ñ
- Amplia documentación y excelente apoyo de la comunidad.
- Mysql

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS, por sus siglas en inglés) con un modelo cliente-servidor.

Funciones

MySQL crea una base de datos para almacenar y manipular datos, definiendo la relación de cada tabla.

Los clientes pueden realizar solicitudes escribiendo instrucciones SQL específicas en MySQL.

- Docker

Un contenedor es una unidad de software estándar que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable de un entorno informático a otro. Una imagen de contenedor Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema y configuraciones.

Modelado de amenazas

En este apartado, se mostrará los diagramas de flujos de datos (DFD), el DFD 0 o conocido por diagrama de contexto y el DFD 1. Estos nos permiten diseñar nuestro sistema, ver las funciones y el flujo datos que hay entre ellos.

Diagrama de Flujos de Datos (DFD) 0

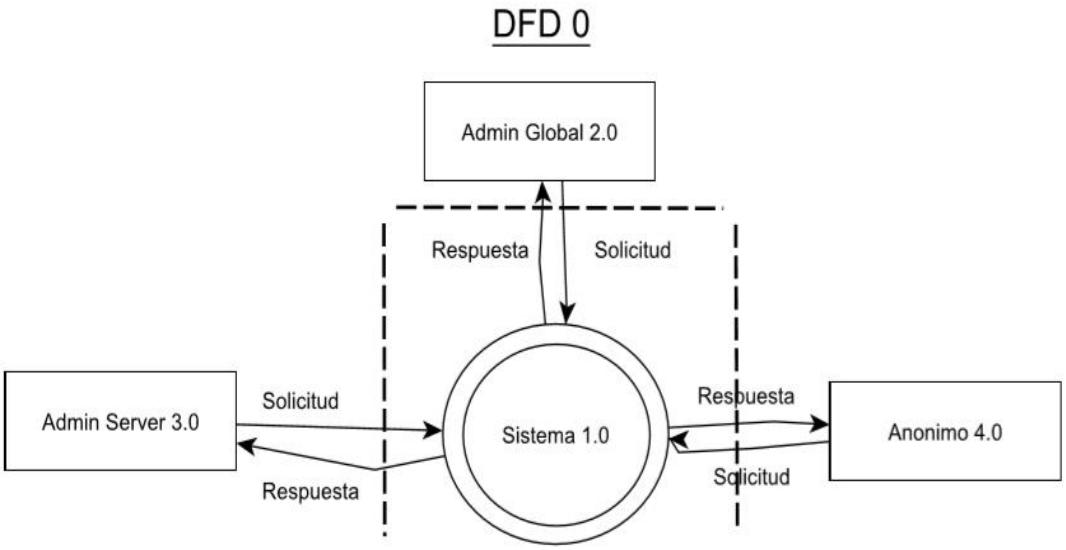

Ilustración 1 DFD 0

Diagrama de Flujos de Datos (DFD) 1

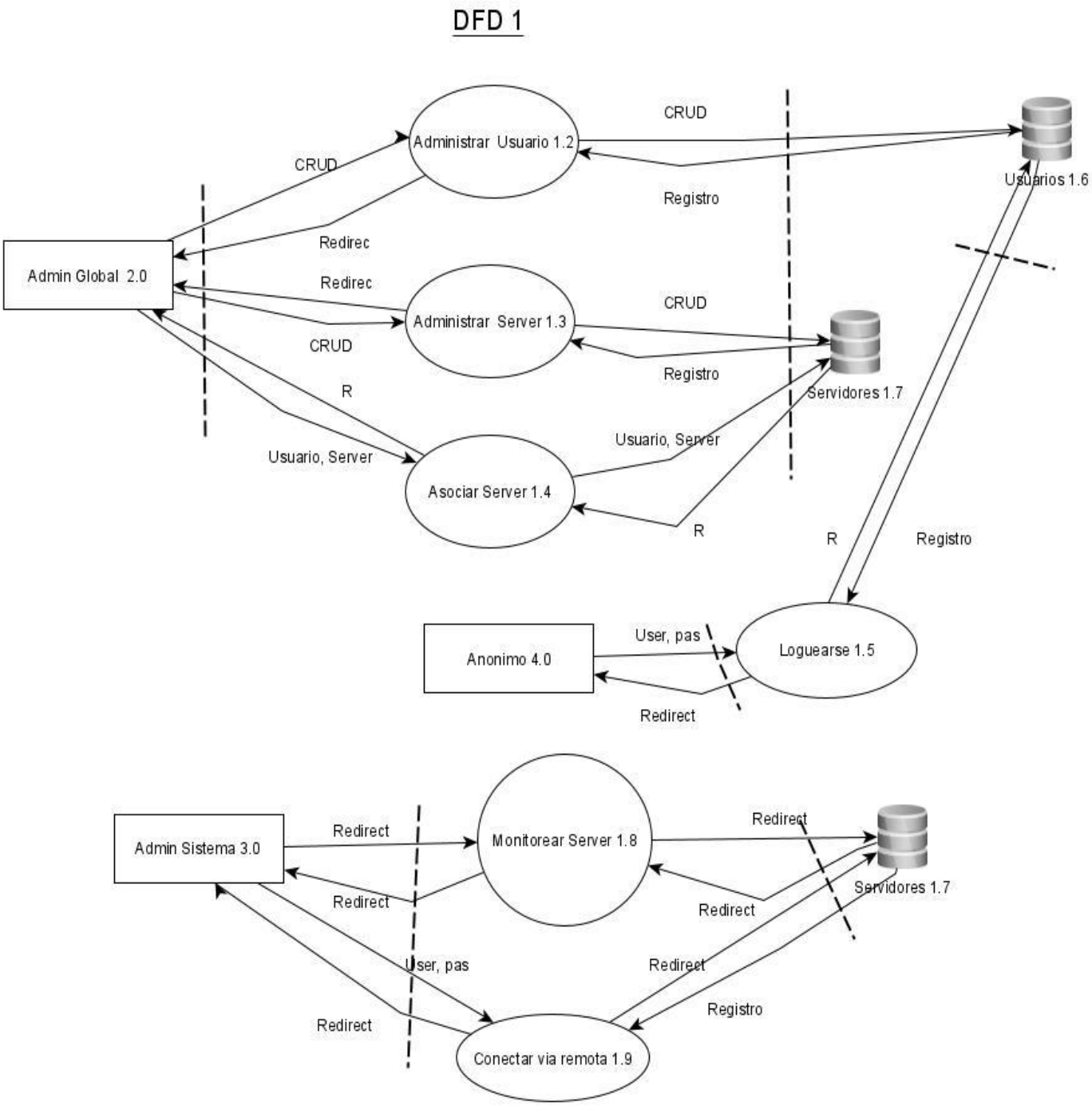

Ilustración 2 DFD 1

Tablas con análisis y medidas de mitigación

Modelado de Amenazas

Tabla 1.

Tipo de Elemento DFD	Número item DFD
Entidades Externas	Admin Global 2.0 Admin Server 3.0 Anónimo 4.0
Procesos	Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar vía remora 1.9 Monitorear Server 1.8
Almacenes	Usuarios 1.6 Servidores 1.7
Flujos de datos	Administrar Usuario 2.0->1.2->2.0 Mostrar Usuarios 1.2->1.6->1.2 Administrar Server 2.0->1.3->2.0 Mostrar Servidores 1.3->1.7->1.3 Asociar Server 2.0->1.4->2.0 Asociación 1.4->1.7->1.4 Iniciar Sesión 4.0->1.5->4.0 Mostrar Usuario 1.5->1.6->1.5 Monitorear 3.0->1.8->3.0 Revisar Servidor 1.8->1.7->1.8 Conectar vía remoto 3.0->1.9->3.0 Mostrar conexión 1.9->1.7->1.9

Tabla 2.

Tipo de amazana STRIDE	Número de Item DFD
Spoofing	**Entidades externas:** Admin Global 2.0, Admin Server 3.0, Anonimo 4.0 **Procesos:** Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar via remora 1.9 Monitorear Server 1.8

Tampering	**Almacenes:** Usuarios 1.6, Servidores 1.7 **Procesos:** Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar via remora 1.9 Monitorear Server 1.8 **Flujos de datos:** Administrar Usuario 2.0->1.2->2.0 Mostrar Usuarios 1.2->1.6->1.2 Administrar Server 2.0->1.3->2.0 Mostrar Servidores 1.3->1.7->1.3 Asociar Server 2.0->1.4->2.0 Asociación 1.4->1.7->1.4 Iniciar Sesión 4.0->1.5->4.0 Mostrar Usuario 1.5->1.6->1.5 Monitorear 3.0->1.8->3.0 Revisar Servidor 1.8->1.7->1.8 Conectar vía remoto 3.0->1.9->3.0 Mostrar conexión 1.9->1.7->1.9
Repudation	**Entidades externas:** Admin Global 2.0, Admin Server 3.0, Anónimo 4.0 **Procesos:** Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar via remora 1.9 Monitorear Server 1.8
Information disclosure	**Almacenes:** Usuarios 1.6, Servidores 1.7 **Procesos:** Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar via remora 1.9 Monitorear Server 1.8 **Flujos de datos:** Administrar Usuario 2.0->1.2->2.0 Mostrar Usuarios 1.2->1.6->1.2 Administrar Server 2.0->1.3->2.0 Mostrar Servidores 1.3->1.7->1.3 Asociar Server 2.0->1.4->2.0 Asociación 1.4->1.7->1.4 Iniciar Sesión 4.0->1.5->4.0 Mostrar Usuario 1.5->1.6->1.5 Monitorear 3.0->1.8->3.0 Revisar Servidor 1.8->1.7->1.8 Conectar vía remoto 3.0->1.9->3.0 Mostrar conexión 1.9->1.7->1.9

Denial of service	**Almacenes:** Usuarios 1.6, Servidores 1. **Procesos:** Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar vía remora 1.9 Monitorear Server 1.8
Elevation of privilages	**Procesos:** Administrar Usuario 1.2 Administrar Server 1.3 Asociar Server 1.4 Loguearse 1.5 Conectar vía remora 1.9 Monitorear Server 1.8

Tabla 3.

Componente	Tipo componente	Amenaza	Mitigación
Admin Global 2.0	Entidad Externa	S, R	bitácoras(r), autenticación, autenticación doble factor(s)
Admin Server 3.0	Entidad Externa	S, R	bitácoras(r), autenticación, Autenticación doble factor(s)
Anonimo 4.0	Entidad Externa	R	Logs(r)
Administrar Usuario 1.2	Procesos	S, T, R, I, D, E	Limitar intentos de autenticación (S, E, D), Validación de entradas y sanitización (T, I), Bitácoras(R)
Administrar Server 1.3	Procesos	S, T, R, I, D, E	Limitar intentos de autenticación (S, E, D), Validación de entradas y sanitización (T, I), Bitácoras(R)
Asociar Server 1.4	Procesos	R, D	Logs(r), Limitar tiempo del proceso(D)
Loguearse 1.5	Procesos	S, T, R, I, D, E	Limitar intentos de autenticación (S, E, D), Validación de entradas y sanitización (T, I), Bitácoras(R)
Conectar via remora 1.9	Procesos	S, T, R, I, D, E	Logs(r), Autenticación (E, D), Validación de entradas y sanitización(T, I)
Usuarios 1.6	Almacen	T, I	Hashing (I), validación de entradas(T)
Servidores 1.7	Almacen	T, I	Hashing (I), validación de entradas(T)
Administrar Usuario 2.0->1.2->2.0	Flujos de datos	T, I	HTTPS(T, I)
Mostrar Usuarios 1.2->1.6->1.2	Flujos de datos	T, I,	HTTPS(T, I)

Administrar Server 2.0->1.3->2.0	Flujos de datos	T, I	HTTPS(T, I)
Mostrar Servidores 1.3->1.7->1.3	Flujos de datos	T, I	HTTPS(T, I)
Asociar Server 2.0->1.4->2.0	Flujos de datos	T, I	HTTPS(T, I)
Asoaciación 1.4->1.7->1.4	Flujos de datos	T, I	HTTPS(T, I)
Iniciar Sesion 4.0->1.5->4.0	Flujos de datos	T, I	HTTPS(T, I)
Mostrar Usuario 1.5->1.6->1.5	Flujos de datos	T, I	HTTPS(T, I)
Monitorear 3.0->1.8->3.0	Flujos de datos	T, I	HTTPS(T, I)
Revisar Servidor 1.8->1.7->1.8	Flujos de datos	T, I	HTTPS(T, I)
Conectar via remoto 3.0->1.9->3.0	Flujos de datos	T, I	HTTPS(T, I)
Mostrar conexion 1.9->1.7->1.9	Flujos de datos	T, I	HTTPS(T, I)

Diagrama de Despliegue

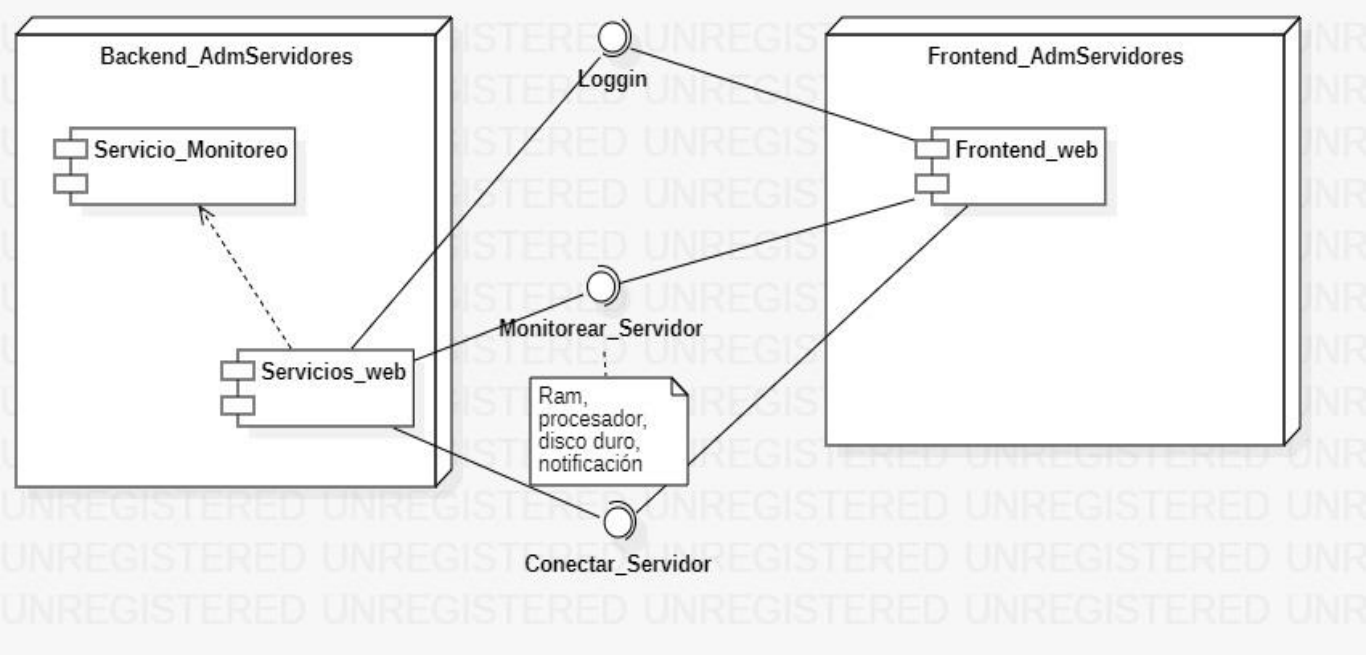

Ilustración 3 Diagrama de Despliegue

Conclusiones

Finalmente, el desarrollo de este sistema pone en practica todos los conocimientos adquiridos en la Experiencia Educativa programación segura. El cual nos permite iniciar desde un análisis, un diseño, el modelado de amenazas(a través de los DFD y tablas STRIDE), para después poner las mitigaciones antes las posibles amenazas.

Una buena práctica fue la de realizar el diagrama de despliegue, que nos ayudó a estructurar los componentes de nuestro sistema.

La realización de las tablas STRIDE, nos permitió identificar las vulnerabilidades a las cuales nuestro sistema podría ser susceptible, las tecnologías con las cuales trabajamos a lo largo del proyecto, tales como el cifrado, el uso de hash, manejo de sesiones, servicios web seguros, autorización, permisos o el throttling.

Para el desarrollo del código seguro o de calidad, se utilizó las herramientas de sonarqube y sonarscanner. Estas nos ayudaron con el análisis estático de código, para detectar posibles problemas de seguridad.

Aprendimos a trabajar con una tecnología llamada Docker, como estudiantes de la licenciatura fue bueno conocerla debido a que era un tema muy nuevo, ya que no teníamos conocimiento de esta. Gracias al académico, se pudo conocer el funcionamiento y la gran utilidad que tiene. De los últimos temas, se aprendió el uso de cifrados con tls/https para que la conexión, estuviese segura.

Finalmente, el desarrollo de este sistema nos brinda los conocimientos y las técnicas de seguridad necesarias para enfrentar escenarios reales dentro de un ámbito profesional.

Bibliografía

Django Software Foundation. (s.f.). Django. Recuperado 10 julio, 2020, de https://www.djangoproject.com/start/overview/

Docker. Why Docker? Recuperado 10 de julio, 2020 de: https://www.docker.com/