INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE INGENIERÍA EN COMPUTACIÓN
CARRERA DE INGENIERÍA EN COMPUTACIÓN

"Traductor CLI-To-REST para switches de Aruba Network"

Proyecto Final de Graduación para optar por el título
de
Ingeniero en Computación
con el grado académico de
Bachillerato Universitario

Dennis Johel Angulo Fuentes

San Jose Junio, 2022

Resumen

Actualmente, los *switches* de Aruba Network pueden ser configurados a través de una Interfaz de Línea de Comandos (*CLI*) y una Interfaz de Programación de Aplicaciones *RESTful* (*API REST*), estas interfaces afectan directamente la base de datos *OVSDB*, la cual guarda directamente la configuración del *switch*. La interfaz *CLI* fue la primera en ser desarrollada, por lo cual tiene un repertorio de pruebas mucho más grade en comparación a la *API REST*.

Este proyecto presenta una solución a la diferencia en el repertorio de pruebas sin necesidad de reducir la brecha "a mano". El proyecto solución es llamado *CLI-To-REST*, es un traductor/transformador directo de las operaciones generadas por *CLI* a métodos *HTTP* para ser usados en la *API REST*.

El sistema ha sido dividido en cinco etapas de desarrollo; la primera se conformó de pruebas de concepto para verificar la viabilidad del sistema, la segunda correspondió a generar en parte las traducciónes/tranfosmaciones para ciertos tipos de tablas, la tercera etapa (la cual se define en este documento) consta en terminar las traducciones para los tipos de tablas restantes, la cuarta etapa se conformará de presentar pruebas de concepto para integrar el traductor con el framework de pruebas de *switches* usado en Aruba Networks, y la quinta etapa constará en habilitar pruebas de la *API RESTful* aprovechando las de *CLI*.

Palabras claves: REST, RESTful, CLI, API, traductor, transformador.

Tabla de contenidos

Índice de tablas

Índice de figuras

1 Resumen ejecutivo

En este informe se da a conocer el contexto y la descripción del proyecto denominado *CLI-To-REST*, donde se explica el quehacer de la empresa donde se realiza el proyecto y los antecedentes del proyecto, así como también la descripción del problema que se busca solucionar, y ambién los objetivos y alcances del sistema planteado como solución.

A su vez, se presenta de forma textual la propuesta de arquitectura para la solución del proyecto *CLI-To-REST,* así como también utilizando un diagrama de componentes y un diagrama de paquetes.

Por último, se presenta las conclusiones y comentarios obtenidas de la realización del proyecto *CLI-To-REST*.

2 Descripción del problema

2.1 Contexto del proyecto

2.1.1 Quehacer de la empresa

Aruba Networks es una compañía subsidiaria de Hewlett Packard Enterprise (HPE) que ofrece productos, soluciones y servicios dirigidos a redes inalámbricas y empresariales, así como software de inteligencia artificial, aplicaciones para administración de redes y sistemas operativos para dispositivos de red, tales como switches. A continuación, se presentan ejemplos de productos y servicios ofrecidos por Aruba:

a. Puntos de acceso con tecnología WiFi 6
b. Puertas de enlace
c. Servicios de localización
d. Redes gestionadas en la nube
e. Switches

En Aruba Networks existen equipos de desarrollo de software con diferentes enfoques enfoques, el proyecto se realizará en el equipo de *CLI-To-REST*, el cual es un subequipo de *REST*, éste a su vez es un subequipo de *High Level & Core Software (hlcsw)* perteneciente al subequipo de *Embeeded System Software* (essw) de Aruba Networks.

2.1.2 Antecedentes del proyecto

El proyecto presentado en este documento es la tercera parte de un proyecto más grande dividido en cinco tareas. A continuación, una tabla de los objetivos de los cinco enfoques y estado de implementación:

Enfoque	¿Completada?
Presentar pruebas de concepto para confirmar la viabilidad del proyecto.	Sí
Generar las primeras traducciones de configuración para ciertas tablas, relaciones y métodos *HTTP*.	Sí
Generar las traducciones restantes de configuración para ciertas tablas, relaciones y métodos HTTP.	No (Tarea descrita en este documento)
Presentar pruebas de concepto para integrar el traductor con el Framework de Testing.	No
Habilitar pruebas de la *API RESTful* aprovechando las de *CLI*.	No

Tabla 1: Antecedentes del ptoyecto

2.2 Descripción del problema

2.2.1 Enunciado del problema

La naturaleza de los productos ofrecidos por la empresa no permite la aplicación de productos similares en el mercado debido a el sistema operativo especializado dentro de estos que lleva el nombre de *ArubaOS-CX* [1].

Existe una diferencia considerable, en términos de cantidad, entre el repertorio de testeo para la configuración del switch mediante el *CLI* y la *API RESTful*, por consiguiente, la *API RESTful* está mucho menos probada o "pulida". Lo anterior tiene dos posibles soluciones: igualar el repertorio de pruebas de la *API RESTful* (esto requiere un gran esfuerzo, tiempo y dinero, por lo que se considera como la última opción), o crear un traductor de *CLI* a la *API* dándole a los testers de los switches posibilidad de probar la *API* con todo el repertorio de pruebas de *CLI* y a su vez se evitarles la tarea de crear pruebas para la *API* ya que con solo crear la prueba para *CLI* se pueden probar ambos métodos de configuración.

2.2.2 Enunciado de la solución

Existen dos métodos para configurar el switch: *CLI* y la *API* de *RESTful*. Se creará un traductor de *CLI* a la *API RESTful* (cli-to-rest) para aprovechar el repertorio de testing disponible en la configuración del switch por CLI, para ello es necesario revisar y traducir toda la configuración del dispositivo implementada por *CLI* a su semejante en la *API*.

Actualmente, el proyecto principal está dividido en cinco partes; la primera parte constó el crear pruebas de concepto para analizar la viabilidad del proyecto, la segunda parte se empezaron a generar las traducciones/transformaciones de para ciertas tablas, la tercera (la actual, presentada en este documento) consta de continuar las traducciones/transformaciones no completadas en la segunda parte, la cuarta parte (y si no hay contratiempos en la tercera parte) consiste en generar pruebas de concepto para integrar el traductor con el Framework de Testing y la quinta parte consiste en habilitar pruebas de la *API RESTful* aprovechando las de *CLI*.

Debido a que la configuración del switch se almacena en la base de datos *OVSDB* [2] y a la manera de interactuar con la *API RESTful*, el proyecto se dividirá en tres subtareas de traducción:

a. Traducir toda configuración que implique cambios en las tablas de relación M:1. Las tablas Radius Server y Static MAC son ejemplos de configuraciones que serán abarcadas en este punto.
b. Completar las traducciones de configuración que impliquen cambios en tablas de relacion M:1 explícitas. La configuración de la tabla sFlow Collector es un ejemplo de lo abarcado en este punto.
c. Traducir toda configuración de la interfaz *System*.

Cabe aclarar que la práctica profesional consta de menos semanas (16 semanas) y la pasantía del estudiante de más (26 semanas), por lo que la tercera parte del proyecto se continuará trabajando al finalizar este proyecto.

2.2.3 Descripción de los patrocinadores

Stakeholders		
Rol	**Nombre**	**Responsabilidades**
Administrador de proyecto	Leonidas Arburola	Supervisor de tareas. Toma de decisiones sobre el rumbo del proyecto.
Arquitecta de Software	Daniela Ramírez	Definición de arquitectura del proyecto. Análisis de requerimientos.
Líder técnico	Luis Daniel Chacón	Líder del grupo de desarrollo. Atención de consultas. Apoyo al grupo de desarrollo.
Desarrollador	Dennis Angulo	Desarrollo del proyecto. Planificación de diseño e implementación.
Desarrollador	Isaac Fonseca	Desarrollo del proyecto. Planificación de diseño e implementación.

Tabla 2: Stakeholders

2.2.4 Resumen de Necesidades y Expectativas

Los requerimientos funcionales del sistema son los siguientes:

a. El sistema traducirá de CLI a métodos RESTful toda configuración que afecte al tipo de tabla 1:M Explícita
b. El sistema traducirá de CLI a métodos RESTful toda configuración que afecte al tipo de tabla M:1
c. El sistema traducirá de CLI a métodos RESTful toda configuración que afecte las interfaces de *System*

A continuación, se muestra una lista de los productos esperados del sistema:

a. Traductor de configuraciones de las tablas 1:M explícitas
b. Traductor de configuraciones de las tablas M:1
c. Traductor de configuraciones de la interfaz *System*

2.2.5 Requerimientos no funcionales

A continuación, una lista de requerimientos no funcionales del sistema propuesto en este documento:

a. Rendimiento: la traducción de las configuraciones no debe afectar el rendimiento del sistema operativo.
b. Testeabilidad: los módulos y funciones deben ser fáciles de implementar pruebas unitarias.
c. Extensibilidad: el código escrito debe ser posible de extender sin necesidad de crear regresiones.

2.2.6 Perspectiva, supuestos y dependencias del producto

Actualmente, el sistema descrito en este informe reside dentro del sistema operativo *ArubaOS-CX* [1], por lo que existe una fuerte dependencia hacia el sistema operativo. Dentro del sistema operativo existen las siguientes dependencias hacia el proyecto:

a. Interfaz de configuración del *switch* nombrado como *CLI*
b. Base de datos OVSDB [2]
c. *Sniffer* de operaciones a la base de datos *OVSDB* [2]

Las anteriores dependencias serán abordadas en la sección 3.1. Debido a que el proyecto se está desarrollando en la tercera etapa del sistema, existen

módulos dentro de este que fueron desarrollados y/o se expandieron dentro del proyecto, los cuales son:

a. *Translator*
b. *ITranslator*
c. *1:MTranslator*
d. *OperationGrouper*
e. *OutputBuilder*
f. *CLI-To-REST*

Los anteriores módulos se abordarán a detalle en la sección 3.2.

2.2.7 Características generales

El proyecto consiste en desarrollar una herramienta para traducir/transformar las operaciones a base de datos realizadas por la interfaz de configuración *CLI* a sus respectivas operaciones *HTTP* en la *API RESTful.* La herramienta quedará incrustada en el sistema operativo *ArubaOS-CX* [1], utilizado para la administración de recursos de los switches de Aruba Network.

Se desarrollará utilizando la metodología agile con la herramienta *Jira* [3], la herramienta de control de versiones *Git* [4] y la herramienta de revisión de código *Gerrit* [5].

Está involucrado un equipo de desarrollo conformado por tres personas: Luis Daniel Chacón (líder de diseño), Isaac Fonseca (desarrollador) y Dennis Angulo (desarrollador, autor del documento).

2.3 Análisis de Riesgos

Esta sección del informe no aplica para el proyecto.

2.4 Objetivos y Alcances del sistema

2.4.1 Objetivo General

Desarrollar las traducciones de las configuraciones CLI a API RESTful de los switches para completar el traductor *CLI-To-REST* y reusar las pruebas CLI del sistema operativo ArubaOS-CX de Aruba Networks.

2.4.2 Objetivos específicos

a. Desarrollar la traducción de toda configuración que implique cambios en las tablas de relación M:1.
b. Completar el desarrollo de las configuraciones que impliquen cambios en las tablas de relación M:1 explícitas.
c. Desarrollar las traducciones de las configuraciones de la interfaz *System*.

2.4.3 Alcance

Dado que el proyecto es un subproyecto del traductor mencionado anteriormente, este solo incluirá las siguientes etapas:

a. Análisis de diseño.
b. Detalles de implementación.
c. Desarrollo de la aplicación.
d. Pruebas unitarias.
e. Pruebas manuales.

Las siguientes etapas del ciclo de desarrollo no serán incluidas debido a que este proyecto es una tarea intermedia en el proyecto principal:

a. Pruebas integrales.
b. Capacitación.
c. Puesta en producción.

3 Modelo de Diseño

En esta sección se describirá la arquitectura de la solución propuesta para el proyecto *CLI-To-REST*:

3.1 Arquitectura conceptual de la solución

El sistema descrito en este punto se llama *CLI-To-REST* y forma parte del sistema operativo especializado para switches de Aruba Networks llamado ArubaOS-CX [1].

El siguiente diagrama muestra en resumen el contexto en el que se encuentra el proyecto dentro del sistema operativo y seguidamente una explicación de los componentes del diagrama:

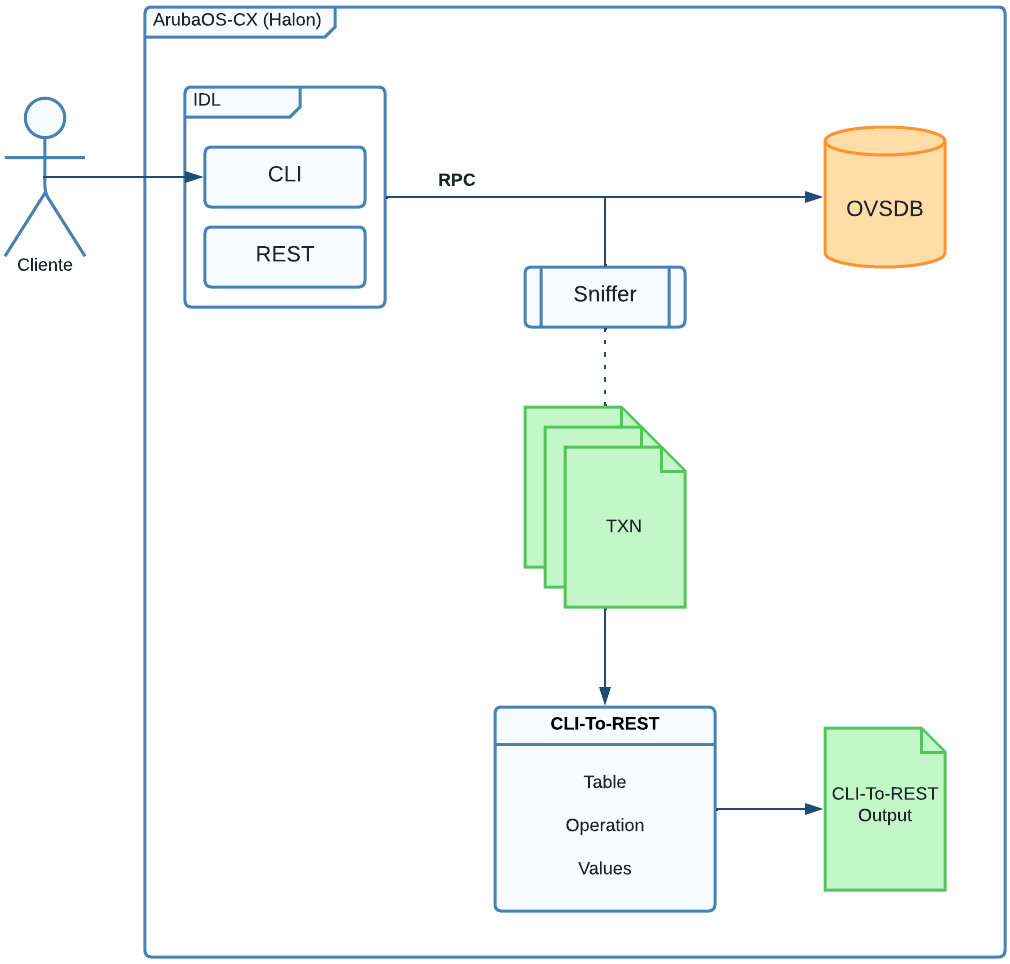

Figura 1: Diagrama contextual del sistema

Cuando un cliente de *CLI* utiliza un comando que modifica el estado de la base de datos *OVSDB* se genera y ejecuta una operación a esta, la cual es

copiada por un *sniffer* y envida a traducir/transformar en el proyecto *CLI-To-REST*.

La interfaz *CLI* es un *demonio* y se comunica con la base de datos *OVSDB* utilizando el protocolo *RPC* y está definido dentro de *IDL* [6], al igual que *REST*.

La base de datos *OVSDB* es una implementación del protocolo *Open vSwitch Database Management* [2] enfocada en las necesidades de *Aruba Networks*. OVSDB define diferentes tipos de relaciones entre tablas: las enfocadas para este proyecto son las tablas de tipo M:1, 1:M, así como las tablas *System* y *Port*.

Actualmente, la salida generada por el proyecto *CLI-To-REST* es guardada en un archivo dentro del *switch* la cual se tiene planeado mostrar dentro de la interfaz *CLI*, pero esta tarea está fuera del alcance del proyecto por lo que no se abordará más a fondo.

Cabe aclarar que actualmente el proyecto se encuentra en su tercera etapa de desarrollo, la primera etapa consistió en generar pruebas de concepto para dar el visto bueno al comienzo del proyecto y la segunda etapa consitió en desarrollar las traducciones para las tablas de tipo Root y 1:M, este último no fue completado en su totalidad.

La tercera etapa de desarrollo (la etapa actual y desarrollada en este proyecto) consiste en completar las traducciones para las tablas de relación de tipo 1:M, y desarrollar en su totalidad las traducciones para de relación de M:1 y para las tablas de tipo Interfaces.

El siguiente diagrama de componentes representa la arquitectura conceptual de la solución la cual se abordará a detalle en la sección "Los modelos de subsistemas, componentes y servicios":

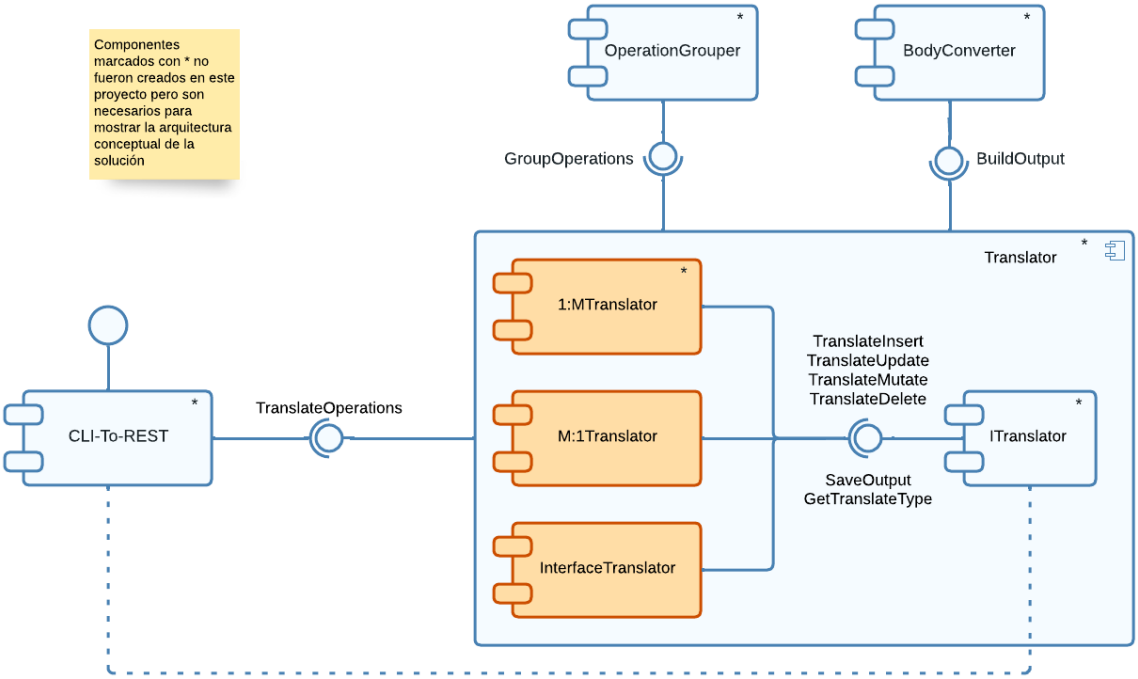

Figura 2: Diagrama de componentes del sistema

3.2 Los modelos de subsistemas, componentes y servicios

El sistema está conformado por 8 subsistemas. A continuación, una lista de los subsistemas con una explicación sobre su función a lo largo de la solución:

CLI-To-REST: subsistema orquestador encargado de recibir el archivo por el *Sniffer* para transformarlo en operaciones REST llamando a la función que se encarga de escribir el resultado de la traducción/transformación.

Translator: Implementa la función *TranslateOperations* en donde crea el *ITranslator* necesario para transformar las operaciones de CLI, llama a *GetTranslateType* del *ITranslator* creado para saber a qué función de transformación llamar (*TranslateInsert, TranslateUpdate, TranslateMutate o TranslateDelete*). Llama a *GroupOperations* para agrupar las operaciones dependientes.

ITranslator: Es una interfaz que define las funciones que un *Converter* debe implementar ya que cada tipo de relación transforma las operaciones a métodos REST a su manera. Las funciones que define son:

a. *TranslateInsert*
b. *TranslateUpdate*
c. *TranslateMutate*

d. *TranslateDelete*
e. *SaveOutput*
f. *GetTranslateType*

1:MTranslator: Componente que implementa *ITranslator* ya que maneja los insert, update, mutate y deletes de manera diferente a las demás relaciones. Se encarga de transformar las operaciones que afectan a tablas 1:M. Llama a *BuildOutput* en su implementación de la función *SaveOutput* para crear el output de la transformación.

M:1Translator: Componente que implementa *ITranslator* ya que maneja los insert, update, mutate y deletes de manera diferente a las demás relaciones. Se encarga de transformar las operaciones que afectan a tablas M:1. Llama a *BuildOutput* en su implementación de la función *SaveOutput* para crear el output de la transformación.

InterfaceTranslator: Componente que implementa *ITranslator* ya que maneja los insert, update, mutate y deletes de manera diferente a las demás tablas de la base de datos. Se encarga de transformar las operaciones que afectan a tablas *Interface* y *Port*. Llama a *BuildOutput* en su implementación de la función *SaveOutput* para crear el output de la transformación.

OperationGrouper: Implementa *GroupOperations,* agrupa las operaciones dependientes entre sí debido a que algunas operaciones que afectan las tablas de tipo 1:M pueden llegar a ser necesario agruparlas ya que afectan a dos tablas en una misma transacción.

OutputBuilder: Implementa *SaveOutput,* se encarga de guardar el output creado por los ITranslators.

3.3 Interfaces de usuario

El sistema desarrollado en este proyecto está incorporado en la interfaz CLI del sistema operativo ArubaOS-CX [1], por lo que este apartado del documento no aplica.

3.4 Diseño de base de datos, etc.

Este apartado del documento no aplica para este proyecto.

4 Conclusiones y comentarios

El sistema se completó en su totalidad, los tres objetivos propuestos para el proyecto se cumplieron sin ningún inconveniente gracias a que en gran parte la empresa proveyó de los instrumentos necesarios para el avance de los internos, las cuales son:

a. Entrenamiento, *onboarding* y/o capacitaciones.
b. Canales de comunicación con compañeros de trabajo experimentados con las herramientas usadas en el oficio.
c. Reuniones diarias, bisemanales y mensuales donde se abordaron distintos temas de conversación para facilitar la integración de los internos dentro del ambiente laboral.
d. Flexibilidad en el horario de trabajo para enfocarse en las tareas de la universidad.

Los productos entregados a la empresa consisten en módulos autodocumentados que ahora forman parte del código fuente del sistema operativo ArubaOS-CX [1]. A continuación, una descripción de los productos que fueron entregados a la empresa:

a. Traductor de configuraciones de las tablas 1:M explícitas: Este producto consiste en las secciones de códigos necesarios para traducir todas configuraciones de insert, update, mutate y delete que afectan a las tablas de tipo 1:M explícitas.
b. Traductor de configuraciones de las tablas M:1: Este producto consiste en las secciones de códigos necesarios para traducir todas configuraciones de insert, update, mutate y delete que afectan a las tablas de tipo M:1.
c. Traductor de configuraciones de la interfaz *System.* Este producto consiste en las secciones de códigos necesarios para traducir todas configuraciones de insert, update, mutate y delete que afectan a las tablas *Interface* y *Port* las cuales forman parte de las configuraciones de las interfaces de tipo *System* dentro del *switch*.

5 Minutas e informes de avance

Formulario de Visitas del Profesor Asesor

Nombre del Estudiante: Dennis Angulo Fuentes

Nombre del Profesor: Luis Montoya

Visita No. 03 Fecha: 1/06/2022

Tipo de Reunión:
( ) Presencial
( ) Telefónica
(X) Chat – Video Conferencia
( ) Otro. Especifique _____

PUNTOS TRATADOS Y ACUERDOS

1. Exposición de épicas y entregables asociados del proyecto por parte del estudiante.
2. Exposición de metas logradas junto a aproximaciones de fechas de inicio y finalización de historias de usuario por parte del estudiante.
3. Presentación final del proyecto, donde se muestra el alcance y objetivos expuestos en el anteproyecto y los ajustes que se le hicieron, así como las metas logradas y los productos entregados a la empresa.
4. Comentarios acerca de la práctica por parte del estudiante, profesor y contraparte de la empresa.

Firmas:

LUIS ARTURO MONTOYA POITEVIEN (FIRMA)
Firmado digitalmente por LUIS ARTURO MONTOYA POITEVIEN (FIRMA)
Fecha: 2022.06.07 16:36:18 -06'00'

Dennis A-F

Profesor Estudiante

6 Referencias

[1] T. Black, "ArubaOS-CX: A Modern, Programmable Network for the Mobile and IoT Age," 14 8 2017. [Online]. Available: https://blogs.arubanetworks.com/solutions/arubaos-cx-a-modern-programmable-network-for-the-mobile-and-iot-age/?highlight=arubaos-cx.

[2] B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol," December 2013. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7047.txt.

[3] Atlassian, "Funcionalidades para el desarrollo de software," [Online]. Available: https://www.atlassian.com/es/software/jira/features.

[4] Software Freedom Conservancy, "Git," [Online]. Available: https://git-scm.com/.

[5] Gerrit, "Gerrit Code Review," [Online]. Available: https://www.gerritcodereview.com/.

[6] OMG, "Interface Definition Language," 2014. [Online]. Available: https://www.omg.org/spec/IDL/4.2/.

[7] Linux Fundation, "ovsdb," [Online]. Available: https://docs.openvswitch.org/en/latest/ref/ovsdb.7/.