PROYECTO FINAL

API administrativo de horas para el servicio social.

Profesor: Rojano Caceres José Rafael

EE: Tecnologías para la integración de soluciones

Fecha: Febrero – Julio de 2022

E q u i p o 1

- Omar Alejandro Alonso Lizardi
- Nadia Itzel Bravo Guevara
- Karla Fernanda Guevara Flores

Índice

1. Introducción

En este documento se presenta el proyecto desarrollado por estudiantes de la Universidad Veracruzana de la Facultad de Estadística e Informática con los conocimientos que fueron adquiridos a lo largo de la experiencia educativa de Tecnologías para la integración de soluciones. Para este proyecto se desarrollará un API que apoye en la administración de horas que realizan los alumnos durante su servicio social así mismo como apoyo a los coordinadores para poder realizar una correcta administración de alumnos y dependencias.

Para comenzar explicaremos a continuación los protocolos aprendidos y que utilizaremos en este proyecto, los cuales son:

- **SOAP:** Es un protocolo que hace uso de diferentes procesos para que puedan comunicarse por medio de intercambios de datos en XML.
  Es un paradigma de mensajería de dirección sin estado, que puede ser utilizado para formar protocolos más completos y complejos según las necesidades de las aplicaciones que lo implementan.
- **REST:** Interfaz para conectar varios sistemas basados en el protocolo HTTP Y sirve para obtener y generar datos y operaciones, devolviéndolos en formatos como XML y JSON.
  REST se apoya en HTTP donde hace uso de:
  - **Post:** Para crear recursos nuevos.
  - **Get:** Para obtener un listado o un recurso en concreto.
  - **Put:** Para modificar.
  - **Patch:** Para modificar un recurso.
  - **Delete:** Para borrar un recurso.

2. Motivación

La motivación para desarrollar este sistema es el poder crear un API que apoye tanto a estudiantes como coordinadores en la agilización del conteo de horas que se realizan durante el servicio social, así como la reducción de uso de papel, ya que creemos que tiene un uso desmedido el cual podría controlarse mediante el uso de una API que permita llevar el control de este.

3. Problemática

Durante la lluvia de ideas que tuvimos para poder elegir un problema al cual darle solución, nos dimos cuenta de uno que la mayoría de los miembros del equipo presentaban es en cuanto al papeleo que manejan en su servicio social, ya que mensualmente se debe entregar al respectivo coordinador un reporte mensual donde se especifiquen las horas (entre otros datos) que el alumno cumplió durante el mes.

En cuanto a la parte de coordinación, encontramos el problema del papeleo ya que un coordinador no solo cuenta con un alumno, sino que se encarga de llevar la administración de todos los estudiantes de su respectiva carrera y las dependencias donde se encuentran dichos estudiantes.

El hecho de tener todos estos registros en papel lo encontramos como un riesgo de seguridad, ya que pueden ocurrir plagios o incidentes que pongan en riesgo la integridad de la información manejada en ellos.

4. Solución

Nuestra propuesta de solución para la problemática antes planteada es crear un API con 2 microservicios que haga uso uno de SOAP y el otro de REST que le permita a alumnos y coordinadores realizar esta tarea de una manera más segura y rápida. Ya que al crear este API se pueden generar diferentes clientes que hagan uso de este servicio.

En nuestro API se podrá realizar lo siguiente:

Usamos SOAP para:

- Registrar horas (POST)
- Consultar horas (tanto realizadas y por terminar) (GET)
- Generar reporte mensual (GET)

Usamos REST para:

- Registrar alumno (POST)
- Registrar dependencia (POST)
- Actualizar token alumno (PUT)
- Actualizar alumno (PUT)
- Actualizar dependencia (PUT)
- Eliminar alumno (DELETE)
- Eliminar dependencia (DELETE)
- Visualizar lista de alumnos (GET)
- Visualizar lista de dependencias (GET)

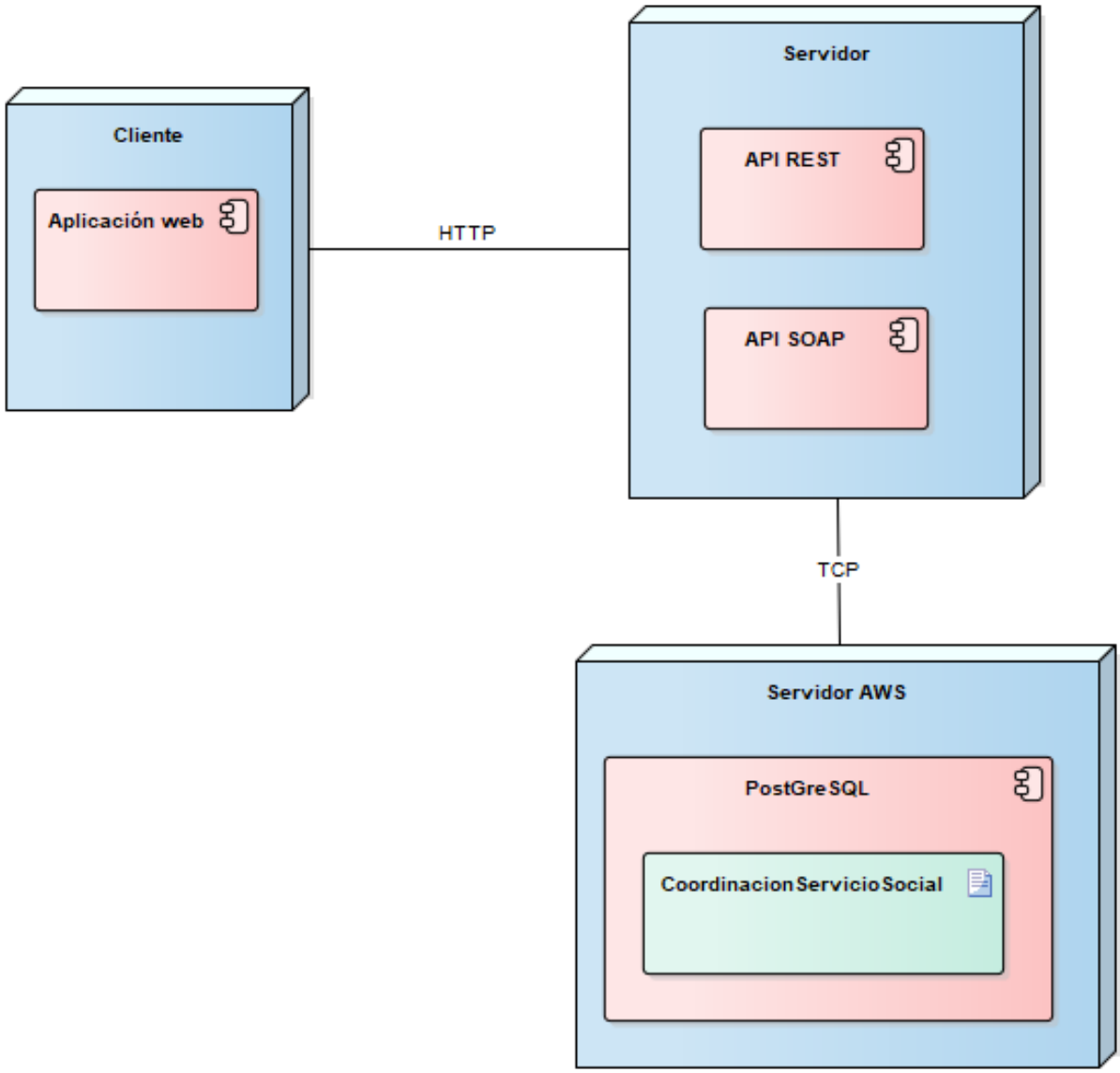
5. Costos

La empresa llamada **INFRANETWORKING** brinda soporte técnico las 24 horas para que cualquier incidencia que sus usuarios que lo contraten como hosting puedan sentirse respaldados en todo momento, siempre teniendo un respaldo para solventar sus problemas a nivel de servidor o bien cuando ocurrían errores de sus páginas.

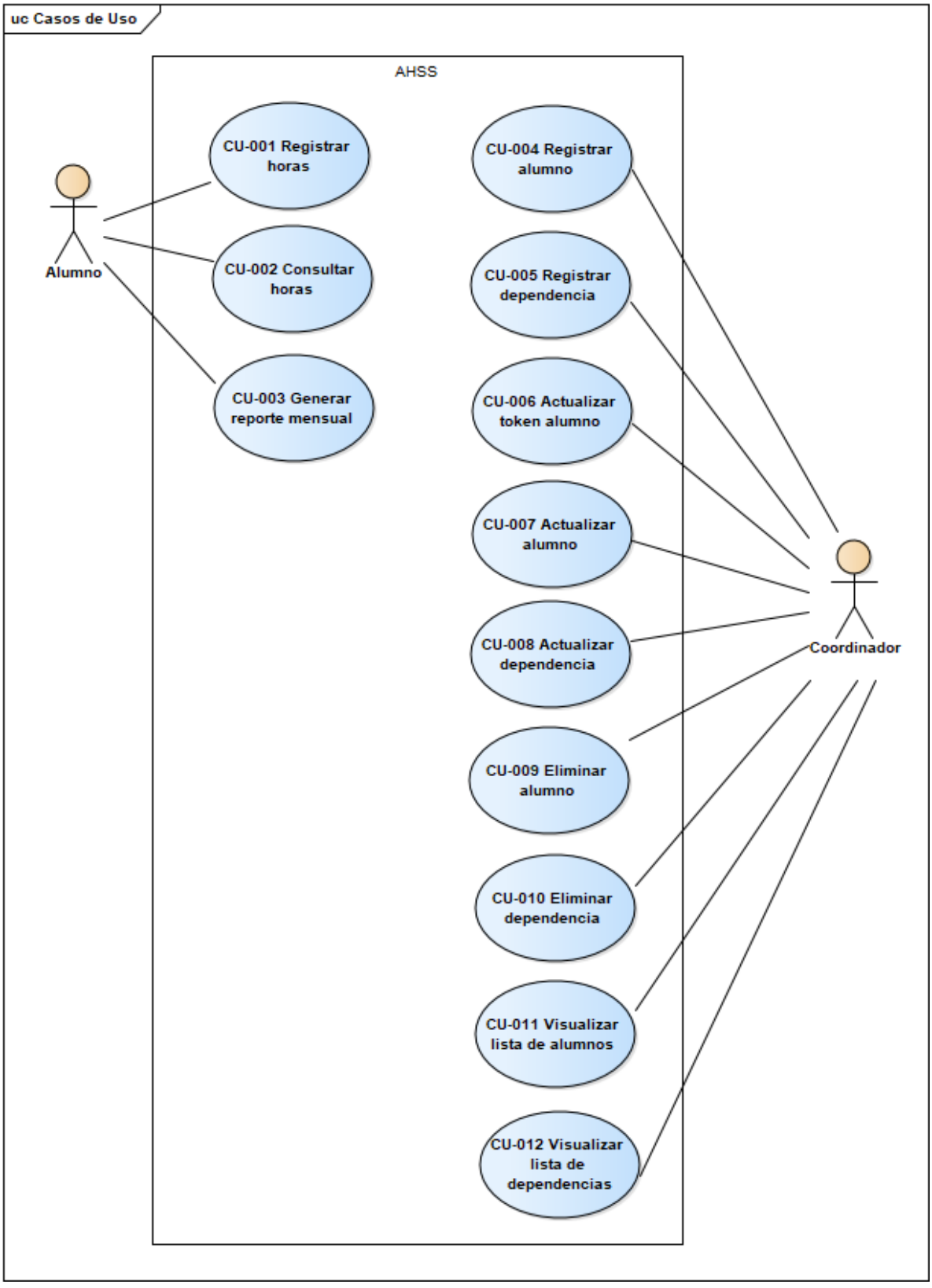
En cuanto a los servicios que ofrece está el Dominio con el costo de **17.48** dólares, Host con el costo de **99.00** dólares, Respaldos con el costo de **30.00** dólares y SSL con el costo de **29.90** dólares; con un costo total de **176.38** dólares.

Tomando en cuenta los cotos que ofrece la empresa los costos en pesos mexicanos serán los siguientes: Dominio con un costo de **342.89**, Host con un costo de **1,941.98**, Respaldos con un costo de **588.48** y SSL con un costo de **586.52**; tomando en cuenta estos resultados el costo total es **3,459.87**.

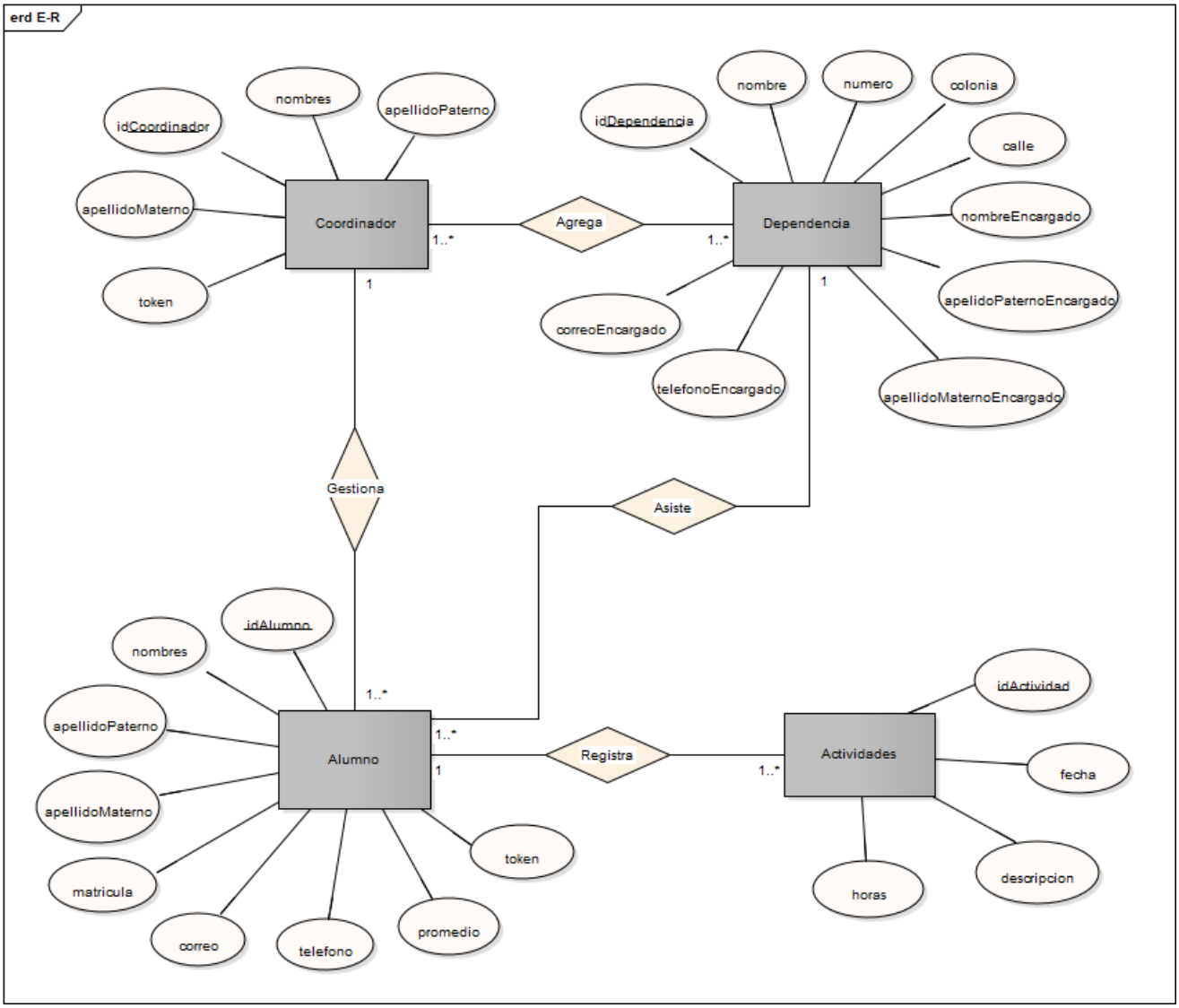
6. Diagrama de despliegue



7. Diagrama de casos de uso



8. Diagrama E-R



9. Documentación del API (SOAP Y REST)

Semana 1 (REST)

	EndPoint	PARAMS	BODY	PARAM DEVUELTOS
Actualizar token Alumno	mysterious-eyrie-42583.herokuapp.com/alumnos/{idAlumno}/actualizarToken **(PUT)** Requiere autorización header Authorization [token]	IdAlumno = El id del alumno al cual se le actualizara el token	En este caso no lleva	El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.
Actualizar alumno	mysterious-eyrie-42583.herokuapp.com/alumnos/{idAlumno}/actualizar **(PUT)** Requiere autorización header Authorization [token]	idAlumno = el id del alumno al cual se le actualizaran los datos	• **idAlumno:** de tipo int • nombres: de tipo String • **apellidoPaterno:** de tipo String • **apellidoMaterno:** de tipo String • **matricula:** de tipo String • **correo:** de tipo String • **token:** de tipo String • **promedio:** de tipo Double • **idCoordinador:** de tipo int • **idDependencia:** de tipo int	El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.

Registrar dependencias	mysterious-eyrie-42583.herokuapp.com/dependencias (**POST)**
	
	Requiere autorización header Authorization [token]
		{
		"idDependencia": 1,
		"nombre": "Dependencia 1",
		"colonia": "Colonia 1",
		"calle": "Calle 1",
		"numero": 1,
		"nombreEncargado": "Nadia",
		"apellidoPaternoEncargado": "Bravo",
		"apellidoMaternoEncargado": "Guevara",
		"correoEncargado": "nadia@correo.com",
		"telefono": "2288776655",
		"idCoordinador": 1
		}
			• **idDependencia:** de tipo int int
			• **nombre:** de tipo String
			• **colonia:** de tipo String
			• **calle:** de tipo String
			• **numero:** de tipo int
			• **nombreEncargado:** de tipo String
			• **apellidoPaternoEncargado:** de tipo String
			• **apellidoMaternoEncargado:** de tipo String
			• **correoEncargado:** de tipo String
			• **teléfono:** de tipo String
			• **idCoordinador:** de tipo int
				El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.

Registrar alumno	mysterious-eyrie-42583.herokuapp.com/alumnos (**POST)**
	
	Requiere autorización header Authorization [token]
		{
		"idAlumno": 0,
		"nombres": "omar",
		"apellidoPaterno": "Alonso",
		"apellidoMaterno": "Lizardi",
		"matricula": "1234567890",
		"correo": "correo@correo.com",
		"token": "123456",
		"promedio": 6.5,
		"idCoordinador": 1,
		"idDependencia": 1,
		"teléfono": "2288990088"
		}
			• **Id Alumno** de tipo int.
			• **Nombres** de tipo String.
			• **Apellido Paterno** de tipo String.
			• **Apellido Materno** de tipo String.
			• **Matricula** de tipo String.
			• **Correo** de tipo String.
			• **Token** de tipo String.
			• **Promedio** de tipo Double.
			• **Id Coordinador** de tipo int.
			• **Id Dependencia** de tipo int.
				El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.

			• **Teléfono** de tipo String.	

Semana 2 (REST)

	EndPoint	PARAMS	BODY	PARAM DEVUELTOS
Actualizar dependencia	mysterious-eyrie-42583.herokuapp.com/dependencias/actualizar Requiere autorización header Authorization [token]	idDependencia: el id de la dependencia que se actualizará	• **idDependencia:** de tipo int int • **nombre:** de tipo String • **colonia:** de tipo String • **calle:** de tipo String • **numero:** de tipo int • **nombreEncargado:** de tipo String • **apellidoPaternoEncargado:** de tipo String • **apellidoMaternoEncargado:** de tipo String • **correoEncargado:** de tipo String • **teléfono:** de tipo String • **idCoordinador:** de tipo int	El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.

Eliminar alumno	mysterious-eyrie-42583.herokuapp.com/alumnos/eliminar alumnos/{idAlumno}/eliminarAlumno **(DELETE)** Requiere autorización header Authorization [token]	idAlumno = el id del alumno el cual se eliminará	• **IdAlumno:** de tipo int	El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.
Eliminar dependencia	mysterious-eyrie-42583.herokuapp.com/ /dependencias/eliminar {idDependencia}/eliminarDependencia **(DELETE)** Requiere autorización header Authorization [token]	idDependencia: el id de la dependencia que se eliminará	• **idDependencia:** de tipo int int	El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.
Visualizar lista de alumnos	mysterious-eyrie-42583.herokuapp.com/alumnos Requiere autorización header Authorization [token]	Este caso no lleva	En este caso no lleva	El endpoint regresa un String, el cual contiene un mensaje de acuerdo con los resultados del proceso.

	EndPoint	PARAMS	BODY	PARAM DEVUELTOS
Buscar alumnos	BuscarAlumnosRequest	• **token:** de tipo String.	No aplica	BuscarAlumnosResponse Alumnos • **idAlumnos:** de tipo int. • **nombres:** de tipo String. • **apellidoPaterno:** de tipo String. • **apellidoMaterno:** de tipo String. • **matricula:** de tipo String. • **correo:** de tipo String. • **token:** de tipo String. • **promedio:** de tipo Double. • **idCoordinador:** de tipo int. • **idDependencia:** de tipo int. • **teléfono:** de tipo String.

Consultar Horas	ConsultarHorasRequest	• **token:** de tipo String. • **idAlumno:** de tipo int.	No aplica	ConsultarHorasResponse Alumnos • **horasRealizadas:** de tipo int. • **horasFaltantes:** de tipo int.
Registrar horas	RegistrarHorasRequest	• **idActividad:** de tipo int. • **descripción:** de tipo String. • **fecha:** de tipo String. • **horas:** de tipo int. • **idAlumno:** de tipo int. • **token:** de tipo String.	No aplica	RegistrarHorasResponse Alumnos • **idActividad:** de tipo int. • **descripción:** de tipo String. • **fecha:** de tipo String. • **horas:** de tipo int. • **idAlumno:** de tipo int. • **token:** de tipo String.
Reporte mensual	ReporteMensualRequest	• **token:** de tipo String. • **mes:** de tipo int.	No aplica	ReporteMensualResponse Alumnos • **nombreAlumno:** de tipo String. • **nombreResponsable:** de tipo String. • **nombreCoordinador:** de tipo String. • **actividad:** de tipo object. • **fecha:** de tipo String. • **horas:** de tipo int.

				• **actividad:** de tipo String. • **horasMes:** de tipo int. • **horasTotal:** de tipo int.

10. Plan de pruebas

Semana 1 (REST)

Caso de uso: Registrar Alumno					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
01	El Coordinador tiene un token valido	{ "idAlumno": 0, "nombres": "omar", "apellidoPaterno": "Alonso", "apellidoMaterno": "Lizardi", "matricula": "1234567890", "correo": "correo@correo.com", "token": "123456", "promedio": 6.5, "idCoordinador": 1, "idDependencia": 1, "teléfono": "2288990088" }	El sistema arroja el mensaje “Alumno + “nombres” + agregado con éxito”	Un nuevo alumno registrado en la base de datos.	Ninguna, funciona con normalidad
02	El coordinador no tiene un token valido	{ "idAlumno": 0, "nombres": "omar", "apellidoPaterno": "Alonso", "apellidoMaterno": "Lizardi", "matricula": "1234567890", "correo": "correo@correo.com", "token": "123456", "promedio": 6.5, "idCoordinador": 1, "idDependencia": 1, "teléfono": "2288990088" }	El sistema arroja el mensaje “Token no valido”		Ninguna, funciona con normalidad

Caso de uso: Registrar Dependencia					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**

03	El Coordinador tiene un token valido	{
		"idDependencia": 1,
		"nombre": "Dependencia 1",
		"colonia": "Colonia 1",
		"calle": "Calle 1",
		"numero": 1,
		"nombreEncargado": "Nadia",
		"apellidoPaternoEncargado": "Bravo",
		"apellidoMaternoEncargado": "Guevara",
		"correoEncargado": "nadia@correo.com",
		"telefono": "2288776655",
		"idCoordinador": 1
		}
			El sistema arroja el mensaje “Dependencia + “nombre” + agregada con éxito”	Una nueva dependencia registrada en la base de datos.	Ninguna, funciona con normalidad

04	El coordinador no tiene un token valido	{
		"idDependencia": 1,
		"nombre": "Dependencia 1",
		"colonia": "Colonia 1",
		"calle": "Calle 1",
		"numero": 1,
		"nombreEncargado": "Nadia",
		"apellidoPaternoEncargado": "Bravo",
		"apellidoMaternoEncargado": "Guevara",
		"correoEncargado": "nadia@correo.com",
		"telefono": "2288776655",
		"idCoordinador": 1
		}
			El sistema arroja el mensaje "Token no valido"		Ninguna, funciona con normalidad

Caso de uso: Actualizar token alumno					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
05	El Coordinador tiene un token valido	{ "idAlumno": 0, "nombres": "omar", "apellidoPaterno": "Alonso", "apellidoMaterno": "Lizardi", "matricula": "1234567890", "correo": "correo@correo.com", "token": "123456", "promedio": 6.5,	El sistema arroja el mensaje "Token actualizado"	Los nuevos datos del alumno se han actualizado en el a base de datos.	Ninguna, funciona con normalidad

		"idCoordinador": 1, "idDependencia": 1, "teléfono": "2288990088" }			

06	El coordinador no tiene un token valido	{
		"idAlumno": 0,
		"nombres": "omar",
		"apellidoPaterno": "Alonso",
		"apellidoMaterno": "Lizardi",
		"matricula": "1234567890",
		"correo": "correo@correo.com",
		"token": "12345",
		"promedio": 6.5,
		"idCoordinador": 1,
		"idDependencia": 1,
		"teléfono": "2288990088"
		}
			El sistema arroja el mensaje "Token no valido"		Ninguna, funciona con normalidad

Caso de uso: Actualizar alumno					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
07	El Coordinador tiene un token valido	{ "idAlumno": 0, "nombres": "omar alejandro", "apellidoPaterno": "Alonso", "apellidoMaterno": "Lizardi",	El sistema arroja el mensaje "Alumno actualizado"	Los datos del alumno se actualizan en la base de datos.	Ninguna, funciona con normalidad

		"matricula": "1234567890", "correo": "correo@correo.com", "token": "123456", "promedio": 6.5, "idCoordinador": 1, "idDependencia": 1, "teléfono": "2288990088" }			

08	El coordinador no tiene un token valido	{
		"idAlumno": 0,
		"nombres": "omar alejandro",
		"apellidoPaterno": "Alonso",
		"apellidoMaterno": "Lizardi",
		"matricula": "1234567890",
		"correo": "correo@correo.com",
		"token": "123456",
		"promedio": 6.5,
		"idCoordinador": 1,
		"idDependencia": 1,
		"teléfono": "2288990088"
		}
			El sistema arroja el mensaje "Token no valido"		Ninguna, funciona con normalidad

Caso de uso: Eliminar alumno					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
09	El Coordinador tiene un token valido	idAlumno: 1	El sistema arroja el mensaje “Alumno eliminado”	Los datos del alumno son eliminados de la base de datos	Ninguna, funciona con normalidad
10	El coordinador no tiene un token valido	idAlumno: 1	El sistema arroja el mensaje “Token no valido”		Ninguna, funciona con normalidad

Caso de uso: Eliminar dependencia					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
11	El Coordinador tiene un token valido	idDependencia: 2	El sistema arroja el mensaje “Dependencia eliminada”	Los datos de la dependencia son eliminados de la base de datos.	Ninguna, funciona con normalidad
12	El coordinador no tiene un token valido	idDependencia: 2	El sistema arroja el mensaje “Token no valido”		Ninguna, funciona con normalidad

Caso de uso: Actualizar dependencia					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**

13	El Coordinador tiene un token valido	{
		"idDependencia": 1,
		"nombre": "Dependencia 1",
		"colonia": "Colonia 1",
		"calle": "Calle 1",
		"numero": 1,
		"nombreEncargado": "Nadia Itzel",
		"apellidoPaternoEncargado": "Bravo",
		"apellidoMaternoEncargado": "Guevara",
		"correoEncargado": "nadia@correo.com",
		"telefono": "2288776655",
		"idCoordinador": 1
		}
			El sistema arroja el mensaje “Dependencia +nombre+ actualizada”	Los datos de la dependencia se actualizan en la base de datos.	Ninguna, funciona con normalidad

14	El coordinador no tiene un token valido	{ "idDependencia": 1, "nombre": "Dependencia 1", "colonia": "Colonia 1", "calle": "Calle 1", "numero": 1, "nombreEncargado": "Nadia Itzel",	El sistema arroja el mensaje “Token no valido”		Ninguna, funciona con normalidad

		"apellidoPaternoEncargado": "Bravo", "apellidoMaternoEncargado": "Guevara", "correoEncargado": "nadia@correo.com", "telefono": "2288776655", "idCoordinador": 1 }			

Caso de uso: Visualizar lista de alumnos					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
15	El Coordinador tiene un token valido	Ninguna			Ninguna, funciona con normalidad
16	El coordinador no tiene un token valido	Ninguna	El sistema arroja el mensaje "Token no valido"		Ninguna, funciona con normalidad

Caso de uso: Registrar horas					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
17	El Alumno tiene un token valido	• idActividad: de tipo int. • descripción: de tipo String. • fecha: de tipo String • horas: de tipo int. • idAlumno: de tipo int • token: de tipo String.	**RegistrarHorasResponse** • resultado: boolean	Realiza un nuevo registro en la base de datos.	Ninguna, funciona con normalidad

Caso de uso: Consultar horas					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
18	El Alumno tiene un token valido	• token: de tipo String • idAlumno: de tipo int	**ConsultarHorasResponse** • horasRealizadas: de tipo int • horasFaltantes: de tipo int.	Muestra los datos almacenados en la base de datos.	Ninguna, funciona con normalidad

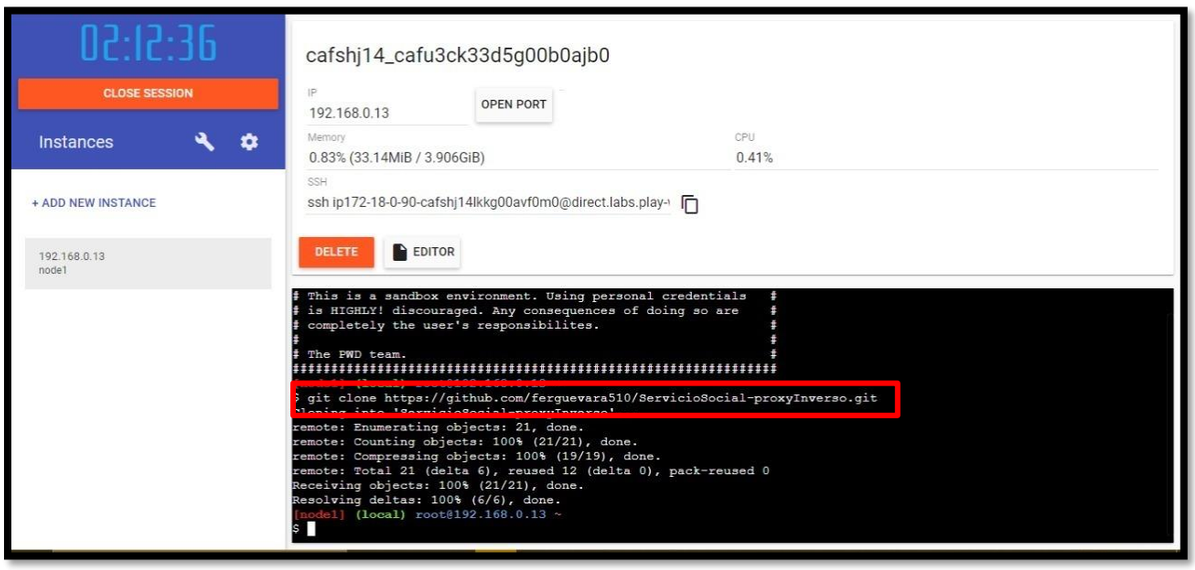
Caso de uso: Generar reporte mensual					
No.	**Condiciones de Entrada**	**Entrada**	**Salida Esperada**	**Condiciones de Salida**	**Observaciones**
19	El alumno tiene un token valido	• token: de tipo String • mes: de tipo int	**ReporteMensualResponse** • nombreAlumno: de tipo String. • nombreResponsable: de tipo String. • nombreCoordinador: de tipo String. • actividad: de tipo object. • fecha: de tipo String. • horas: de tipo int. • actividad: de tipo String. • horasMes: de tipo int. • horasTotal: de tipo int.	Realiza los cálculos para generar el reporte mensual.	Ninguna, funciona con normalidad

11. Docker

Como primer paso vamos a clonar el repositorio donde tenemos el proxy inverso en playWithDocker con el comando:

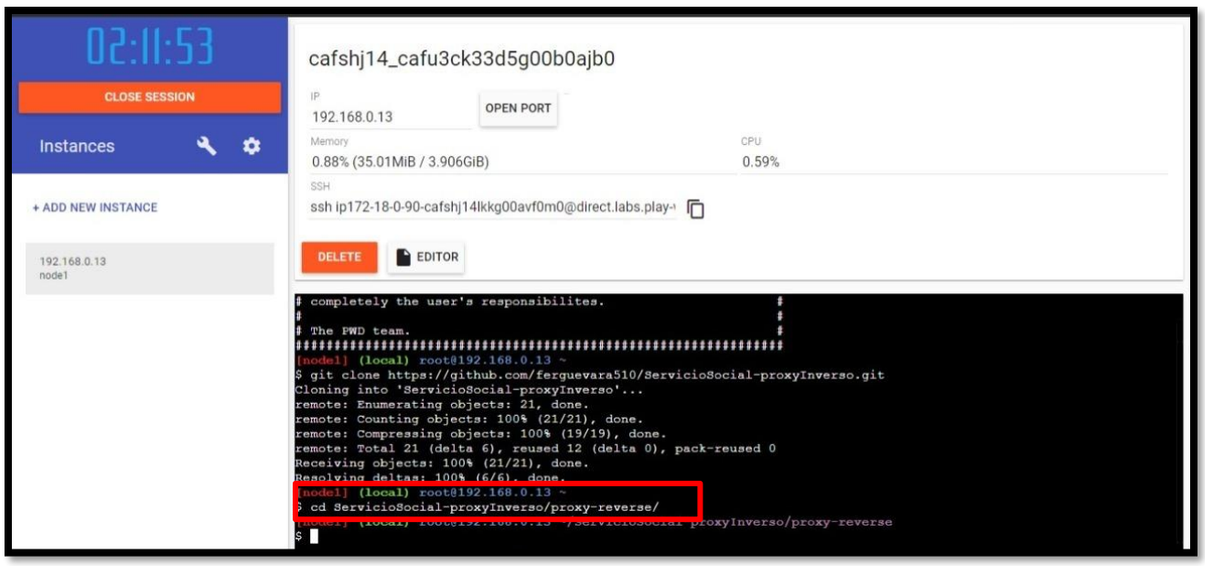
git clone https://github.com/ferguevara510/ServicioSocial-proxyInverso.git

Con eso tendremos listo y terminado el primer paso.



Como segundo paso, nos moveremos a la carpeta proxy-reverse con el siguiente comando:

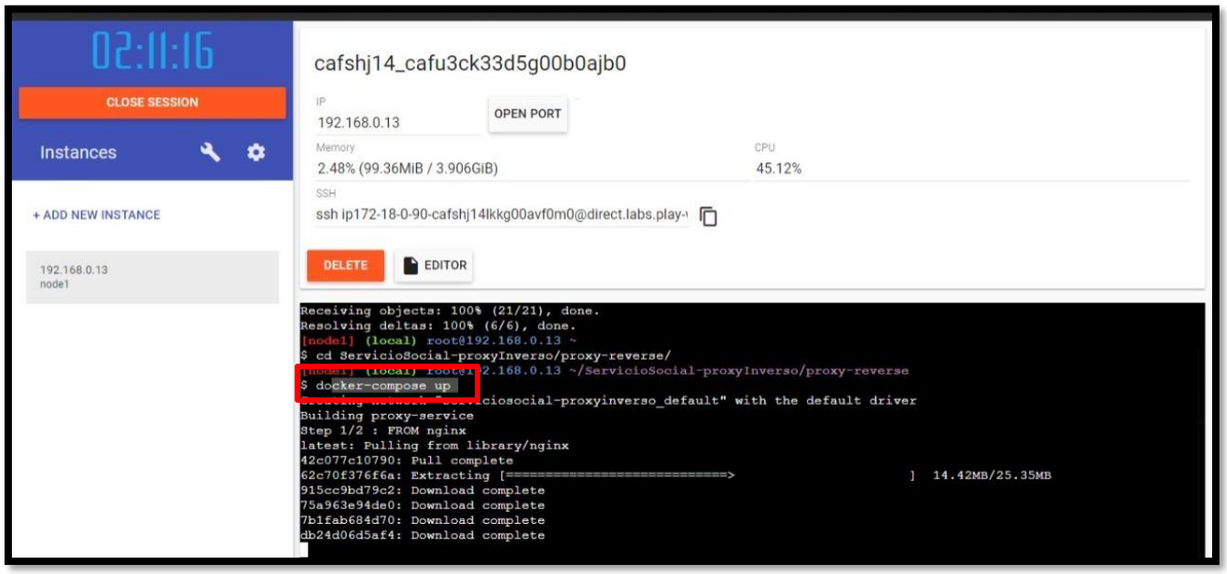
cd ServicioSocial-proxyInverso/proxy-reverse



En esa carpeta encontraremos el archivo que tiene la configuración por efecto.

Como último paso, usaremos el siguiente comando:

docker-compose up



Esto para levantar el proyecto y listo, con eso funciona nuestro proxy inverso y ponemos como ejemplo que ya podemos acceder al servicio de SOAP.