

TECNOLOGÍAS PARA LA INTEGRACIÓN DE
SOLUCIONES

EQUIPO 1

**CONTROL DE AFLUENCIA EN SITIOS
TURÍSTICOS DE XALAPA**

ABURTO LARA MAURICIO
GARCÍA APODACA JOAQUÍN ALBERTO
ORTEGA ZENTENO ARIDAI
PACHECO BAIZABAL CAROL CELINA

20/05/2022

TECNOLOGÍAS PARA LA INTEGRACIÓN DE SOLUCIONES PROYECTO

Pinacoteca Diego Rivera, Biblioteca Carlos Fuentes, Ágora de la Ciudad de Xalapa

CONTROL DE AFLUENCIA EN SITIOS TURÍSTICOS DE XALAPA

INTRODUCCIÓN

En la actualidad los sistemas web son indispensables para la administración de locales, negocios, empresas, museos, etcétera, así mismo la implementación de bases de datos permiten a estos llevar el control adecuado de la información.

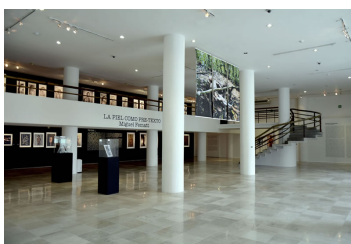
Se identificó una situación donde se puede aplicar un servicio web el cual beneficiaría a un entorno que recurramos, facilitando a los usuarios el manejo de información, aplicando los conocimientos vistos durante el curso como lo son, SOAP, REST, el uso y manejo de Docker.

MOTIVACIÓN

Promover la cultura realizando un servicio que permita tener reunidos los servicios web en un solo lugar para su fácil acceso y que agilice las búsquedas en los sitios web de estos lugares emblemáticos de la ciudad de Xalapa.

PROBLEMÁTICA

Se ubicaron tres lugares cercanos territorialmente y emblemáticos de la ciudad de Xalapa de los que se desea registrar el número de visitantes para basar esta información en la divulgación de eventos que tomen lugar en estos lugares.



SOLUCIÓN

Pinacoteca Diego Rivera

- Registrar los visitantes
 - ID
 - Nombre de persona
 - Fecha
 - Hora
 - Num de personas
- Registrar eventos por temporadas

- Registrar eventos
- Modificar
- Buscar
 - ID
 - Nombre del evento
 - Fecha
 - Hora
 - Costo
- Registrar artistas / Historial de los artistas

Biblioteca Carlos Fuentes

- Registrar lo visitantes
- Registrar el uso de la biblioteca o los servicios informáticos
- Solicitar espacio para conferencia o evento

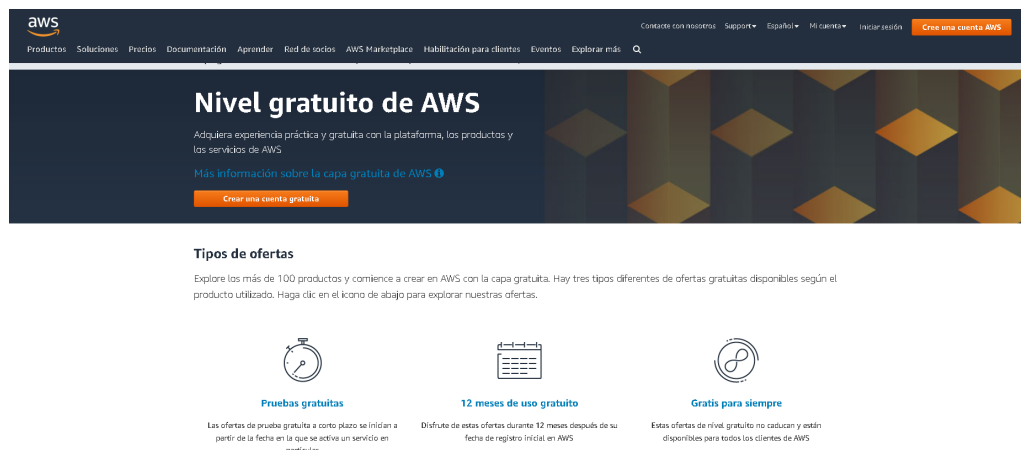
Ágora de la ciudad de Xalapa

- Registrar los visitantes
- Registrar costos de los eventos
- Modificar
- Buscar
 - ID
 - Nombre del evento
 - Fecha
 - Hora
 - Costo
- Registrar las películas que se proyectan
- Crear una proyección en específico?

HOSTING

Buscamos servicios de hosting que preferiblemente tengan un plan de servicios gratuitos donde las opciones que vimos fueron:

Amazon Web Services (AWS) (abreviado) es una colección de servicios de computación en la nube pública (también llamados servicios web) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com. Es usado en aplicaciones populares como Dropbox, Foursquare, HootSuite. Es una de las ofertas internacionales más importantes de la computación en la nube y compite directamente con servicios como Microsoft Azure, Google Cloud Platform y IBM Cloud. Es considerado como un pionero en este campo.



Nivel gratuito de AWS

Adquiera experiencia práctica y gratuita con la plataforma, los productos y los servicios de AWS.

Más información sobre la capa gratuita de AWS

[Crear una cuenta gratuita](#)

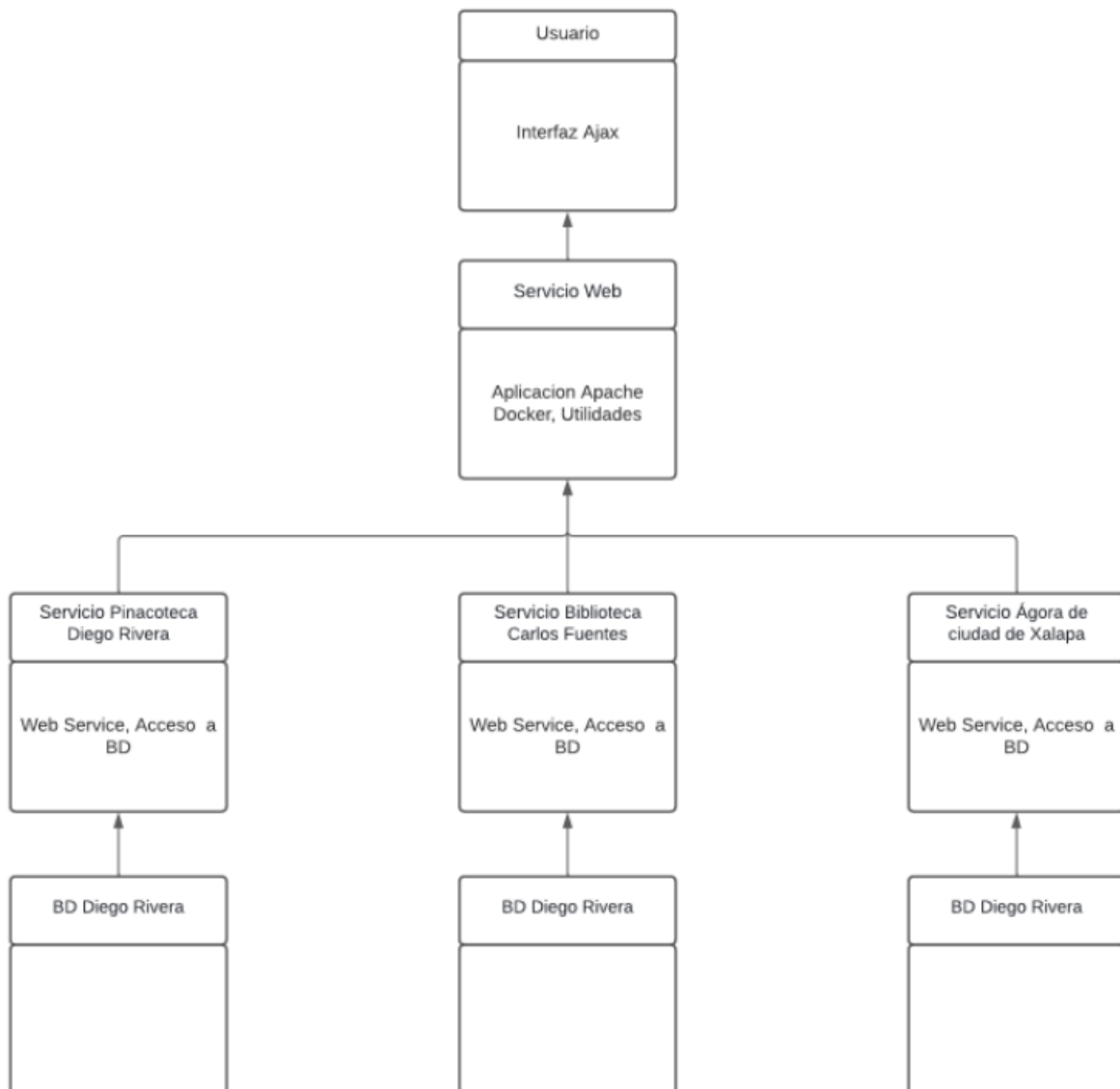
Tipos de ofertas

Explore los más de 100 productos y comience a crear en AWS con la capa gratuita. Hay tres tipos diferentes de ofertas gratuitas disponibles según el producto utilizado. Haga clic en el icono de abajo para explorar nuestras ofertas.

Pruebas gratuitas	12 meses de uso gratuito	Gratis para siempre
Las ofertas de prueba gratuita a corto plazo se inician a partir de la fecha en la que se active un servicio en particular.	Disfrute de estas ofertas durante 12 meses después de su fecha de registro inicial en AWS.	Estas ofertas de nivel gratuito no caducan y están disponibles para todos los clientes de AWS.

https://aws.amazon.com/es/free/?trk=6e90e8fa-6bd8-4a6f-be4b-3bc9e717eb2e&sc_channel=ps&sc_campaign=acquisition&sc_medium=ACQ-P|PS-GO|Brand|Desktop|SU|AWS|Core|LATAMO|ES|Text&ef_id=Cj0KCQjw1N2TBhCOARIsAGVHQc571QEEjuxk0x4CmYwDjT5cv_vUbZm3sZK4VSeWPYBTghhx-ziNqtgaAgCyEALw_wcB:G:s&s_kwid=AL!4422!3!561348326849!e!!g!!amazon%20aws&ef_id=Cj0KCQjw1N2TBhCOARIsAGVHQc571QEEjuxk0x4CmYwDjT5cv_vUbZm3sZK4VSeWPYBTghhx-ziNqtgaAgCyEALw_wcB:G:s&s_kwid=AL!4422!3!561348326849!e!!g!!amazon%20aws&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all

Modelo de Despliegue



DOCKER FILE - CONTENEDOR EN DOCKER

```
from ubuntu:20.04
workdir /app
expose 8080
cmd ["/app/script.sh"]
add ServiciosXLP/Biblioteca-0.0.1-SNAPSHOT.jar /app
run apt-get update
run apt-get install -y openjdk-8-jdk
RUN apt-get install -y mariadb-server
add script.sql /app/script.sql
RUN chmod 755 /app/script.sql
add script.sh /app/script.sh
run chmod 755 /app/script.sh
RUN /etc/init.d/mysql start; mysql < /app/script.sql
```

DOCKER FILE HEROKU

```
from rrojano/jdk8
workdir /app

cmd ["/app/script.sh"]
add Biblioteca-0.0.1-SNAPSHOT.jar /app/Biblioteca-0.0.1-SNAPSHOT.jar
run apt-get update
RUN apt-get install -y mariadb-server
add Script.sql /app/script.sql
RUN chmod 755 /app/script.sql
add Script.sh /app/script.sh
run chmod 755 /app/script.sh
RUN /etc/init.d/mysql start; mysql < /app/script.sql
```

FUNCIONAMIENTO DEL SISTEMA SOAP BIBLIOTECA “CARLOS FUENTES”

El primer sistema funcional es la Biblioteca “Carlos Fuentes” la cual se trabajó en su totalidad con SOAP y se montó en una imagen en Heroku, la cual nos permite hacer uso directo del sistema.

Se pensó en la biblioteca “Carlos Fuentes

CONTIENE

Reservaciones:

Esto consiste en administrar y tener un registro de la información referente a las reservaciones de la biblioteca. En la solución se contempló:

- Registrar Reservaciones
- Modificar Reservaciones
- Buscar Reservaciones
- Eliminar Reservaciones

Consideraciones completas que ayudan a llevar un orden y una administración adecuada de las reservaciones, permitiendo a los usuarios tener más confiabilidad y credibilidad en que serán respetadas y respaldadas sus reservaciones registradas.

Servicios

La biblioteca cuenta con diversos servicios y por ende, requieren de un orden y registro correcto, para ello se implementó como solución:

- Listar Servicios
- Agregar Servicios
- Eliminar Servicios

Esto le permite a los administradores poder conocer y manejar cada uno de los servicios funcionales dentro de la biblioteca.

Registro

Actualmente la biblioteca recibe constantemente a nuevos visitantes recurrentes, para poder tener un registro de la cantidad de visitantes que ingresan a la biblioteca cada determinado tiempo se planteó la solución utilizando el manejo de:

- Registrar Visitantes

Que ayuda a los administradores a saber cuántos visitantes y en qué fechas son más comunes.

ENDPOINTS

RESERVACIONES

RegistrarReservacionesRequest

```
public RegistrarReservacionesResponse reservar(@RequestPayload
RegistrarReservacionesRequest nombre) {
    RegistrarReservacionesResponse respuesta = new
RegistrarReservacionesResponse();
    respuesta.setRespuesta("Hola " + nombre.getNombre());
    Reservacion s = new Reservacion();
    s.setNombre(nombre.getNombre());
    s.setConcepto(nombre.getConcepto());
    s.setFecha(nombre.getFecha());
    s.setHoraInicio(nombre.getHoraInicio());
    s.setHoraFin(nombre.getHoraFin());
    s.setTiempo(nombre.getTiempo());
    Ireservaciones.save(s);
    return respuesta;
}
```

Recibe los parámetros de concepto, Fecha, Hora de inicio y fin y tiempo total para la reservación del lugar.

BuscarReservacionesRequest

```
public BuscarReservacionesResponse buscarReservaciones() {
    BuscarReservacionesResponse respuesta = new
BuscarReservacionesResponse();

    List<Reservacion> listreservaciones = (List<Reservacion>)
Ireservaciones.findAll();
    BuscarReservacionesResponse.Reservacion s = new
BuscarReservacionesResponse.Reservacion();
    for(Reservacion r : listreservaciones){
        s = new BuscarReservacionesResponse.Reservacion();
        s.setId(r.getId());
        s.setNombre(r.getNombre());
        s.setConcepto(r.getConcepto());
        s.setFecha(r.getFecha());
        s.setHoraInicio(r.getHoraInicio());
    }
}
```



```

        s.setHoraFin(r.getHoraFin());
        s.setTiempo(r.getTiempo());
        respuesta.getReservacion().add(s);
    }
    return respuesta;
}

```

Se muestran las reservaciones realizadas.

BorrarReservacionesRequest

```

@PayloadRoot(namespace = "https://Biblioteca.mx/Biblioteca",
localPart = "BorrarReservacionesRequest")
@ResponsePayload
public BorrarReservacionesResponse
borrarReservaciones(@RequestPayload BorrarReservacionesRequest
nombre) {
    BorrarReservacionesResponse respuesta = new
BorrarReservacionesResponse();
    Ireservaciones.deleteById(nombre.getId());
    respuesta.setRespuesta("Elemento Eliminado");
    return respuesta;
}

```

Se introduce el ID de la reservación que se desea eliminar

Registro de Visitantes

RegistrarVisitantesRequest

```

public RegistrarVisitantesResponse
registrarVisitantes(@RequestPayload RegistrarVisitantesRequest
nombre) {
    RegistrarVisitantesResponse respuesta = new
RegistrarVisitantesResponse();
    Registro s = new Registro();
    s.setNombre(nombre.getNombre());
    s.setFecha(nombre.getFechaDdMmAa());
    s.setParejas(nombre.getParejas());
    Iregistro.save(s);
    return respuesta;
}

```

```
}
```

Se recibe el nombre, fecha y las personas que acompañan al representante.

SERVICIOS

AgregarServicioRequest

```
public AgregarServicioResponse agregarActividad(@RequestPayload
AgregarServicioRequest agregar) {
    AgregarServicioResponse respuesta = new
AgregarServicioResponse();
    respuesta.setRespuesta("SERVICIO ANOTADO EXITOSAMENTE");
    Servicios servicio = new Servicios();
    servicio.setNombre(agregar.getNombre());
    servicio.setMotivo(agregar.getMotivo());
    servicio.setTiempo(agregar.getTiempo());
    servicio.setFecha(agregar.getFecha());
    iservicios.save(servicio);
    return respuesta;
}
```

El registro de servicios recibe un nombre, motivo (que es la razón para la que está siendo utilizado ese servicio), el tiempo que va a ser utilizado y la fecha.

ListarServicioRequest

```
public ListarServicioResponse ListarTareas() {
    ListarServicioResponse respuesta = new
ListarServicioResponse();
    Iterable<Servicios> lista = iservicios.findAll();

    for (Servicios servicio : lista) {
        ListarServicioResponse.Servicio a = new
ListarServicioResponse.Servicio();
        a.setNombre(servicio.getNombre());
        a.setId(servicio.getId());
        a.setMotivo(servicio.getMotivo());
        a.setTiempo(servicio.getTiempo());
        a.setFecha(servicio.getFecha());
        respuesta.getServicio().add(a);
    }
}
```

```
        return respuesta;
    }
```

Aquí se pueden consultar todos los registros de uso de servicios que ha habido.

EliminarServicioRequest

```
public EliminarServicioResponse eliminarActividad(@RequestPayload
EliminarServicioRequest eliminar){
    EliminarServicioResponse respuesta = new
EliminarServicioResponse();
    iservicios.deleteById(eliminar.getId());
    respuesta.setRespuesta("ACTIVIDAD ELIMINADA EXITOSAMENTE");
    return respuesta;
}
```

Aquí se pueden eliminar todos los registros de uso de servicios que ha habido.

Forma de ejecución de los contenedores

Contamos con un contenedor en docker, el cual contiene el servicio de biblioteca containerizado.

docker pull aridaioza/biblioteca

docker run -it --rm -p 8080:8080 aridaioza/biblioteca
+ /ws/biblioteca.wsdl

LINK DE HEROKU

Servicio de Biblioteca

<https://biblioteca-tis.herokuapp.com/ws/biblioteca.wsdl>

LINK DE GITHUB

<https://github.com/JoaquinGA01/ServiciosXLP>