

20/05/2022



Universidad Veracruzana

SERVICIO DE CINE

TECNOLOGÍAS PARA LA INTEGRACIÓN DE SOLUCIONES



EQUIPO #2

Preparado por:

Hernández García Abdel René.

Hernández Martínez Justin.

Tejeda Sántiz Ariana.

Catalino Piquet Bryan Usias

Catedrático:

Dr. Rojano Caceres Jose Rafael

Contenido

Introducción	4
Motivación	4
Problemática	4
Solución	5
Costos	5
• Creación de repositorio compartido:.....	5
• Servicios de nube	5
• Hosting:	5
• Peticiones para testear APIs de tipo REST	6
• Mano de obra	6
Diagrama de despliegue	7
Documentación de API SOAP y REST	8
Microservicio de Tickets y Boletos	8
EndPoint.....	8
Microservicio de Empleados	10
EndPoint.....	10
Microservicio de Dulcería	13
EndPoint.....	13
Parámetros de recepción	17
Microservicio Funciones(request)	17
Microservicio Empleados (Request)	18
Microservicio Dulceria (Request).....	21
Parámetros devueltos	24
Microservicio Tickets y Funciones (Response)	24
Microservicio Empleados (Response)	24
Microservicio Dulcería (Response).....	28
Plan de pruebas	31
Microservicio Tickets y Funciones	31
Microservicio Empleados.....	33

Microservicio Dulceria	34
Dockerfile del Microservicio Tickets y Funciones	37
Ejecutar del Microservicio Tickets y Funciones.....	37
Dockerfile del Microservicio Empleados	37
Ejecutar del Microservicio Empleados	38
Dockerfile del Microservicio dulcería.....	38
Ejecutar del Microservicio dulcería	38
<i>Proxy inverso.....</i>	39
Dockerfile	39
Documentación .html.....	39
Estilos.....	41
Acciones JS.....	44
<i>Proyecto publicado en github.....</i>	46
Microservicio Tickets y Boletos	46
Microservicio Empleados.....	46
Microservicio de dulces	46
<i>Proyecto publicado en Heroku</i>	46
Microservicio Tickets y Boletos	46
Microservicio Empleados.....	46
Microservicio Dulces	46
Proxy	46

Introducción

En el presente proyecto de Tecnologías para la integración de soluciones, tiene como objetivo evaluar y adaptar los conocimientos adquiridos durante el curso, para ello se tiene planeado realizar un servicio de API'S con distintos Microservicios, todo enfocado a un Cine para resolver algunos servicios que requiera, en el cual emplearemos las tecnologías que son SOAP y REST para su desarrollo.

Nuestro proyecto será realizado siguiendo los puntos solicitados a evaluar por parte del profesor Rojano, en este caso no nos enfocaremos como tal en el fronted, solo será por la parte del backend. En este servicio de cine se pretende poder hacer consumo de los servicios que puede ofrecer un cine, tenemos pensado que el consumo del servicio sea para el cliente y también para que el cine pueda administrar algunos detalles.

Motivación

La motivación de nuestro proyecto surge con la idea de ayudar a la facilitación y buen manejo administrativo de un cine cualquiera con ciertas funciones como compra de boletos o tickets, poder comprar fácilmente dulces y administrar a los empleados del cine.

Problemática

Una de las problemáticas que se encontraron fue en la parte administrativa de un cine por ende proponemos que nuestro servicio sirva para tras aspectos que ayuden a solucionar la organización, por ejemplo, en la parte de la dulcería en donde le permitirá al usuario comprar dulces que estén disponibles en el inventario y solo llevar un ticket al Cine de esta manera poder evitar largas filas y ahorrar tiempo.

Por otra parte, administrativamente el poder comprar boletos y ver la cartelera disponible en el sistema ahorrará bastante tiempo y permitirá que este controlado el flujo de personas en el Cine. Por último, el manejo de empleados para una adecuada organización interna del personal.

Solución

Para la solución de este problema se tiene planeado realizar un servicio de API el cual está dividido en microservicios siendo estos:

- **Servicio de Tickets o Boletos:** Este microservicio permitirá que el que se revise la cartelera disponible, ver las funciones que se harán por cada película, para ver la cartelera se tendrán atributos como (IdFuncion, Nombre de película, Hora, Fecha, Precio, Clasificación y Asientos disponibles), también se podrá comprar boletos y cambiar u modificar la función.
- **Servicio de dulcería:** Este microservicio se incorporará la opción de comprar dulces, ver el menú, el inventario y modificar el mismo
- **Servicio de Empleados:** En ese microservicio se empleará un CRUD con las siguientes opciones: Dar de Alta Empleados, Ver Empleados, Eliminar Empleados y Modificar los Empleados. Cada apartado tendrá sus distintos atributos

Costos

Las tecnologías utilizadas para nuestro proyecto que nos permitieron deducir costos son las siguientes:

- **Creación de repositorio compartido:**

Utilizamos la tecnología de GitHub para la creación de nuestro repositorio y poder todos apoyar y codificar subiendo nuestros cambios. Esta tecnología es gratuita por ende este recurso no nos costo.



- **Servicios de nube**

En este aspecto para el despliegue de nuestros microservicios utilizaremos Heroku en su versión gratuita, que nos da el almacenamiento para soportar nuestro servicio, lo cual no generó gastos adicionales.



- **Hosting:**

Para el hosting del servicio de base de datos utilizamos una versión de prueba de Clever-Cloud, la cual tampoco generó gastos.



- **Peticiones para testear APIs de tipo REST**

Para las peticiones la tecnología que ocupamos en post man es de uso gratuito y nos va a servir para consumir un recurso de un servicio web desde un cliente de manera grafica.

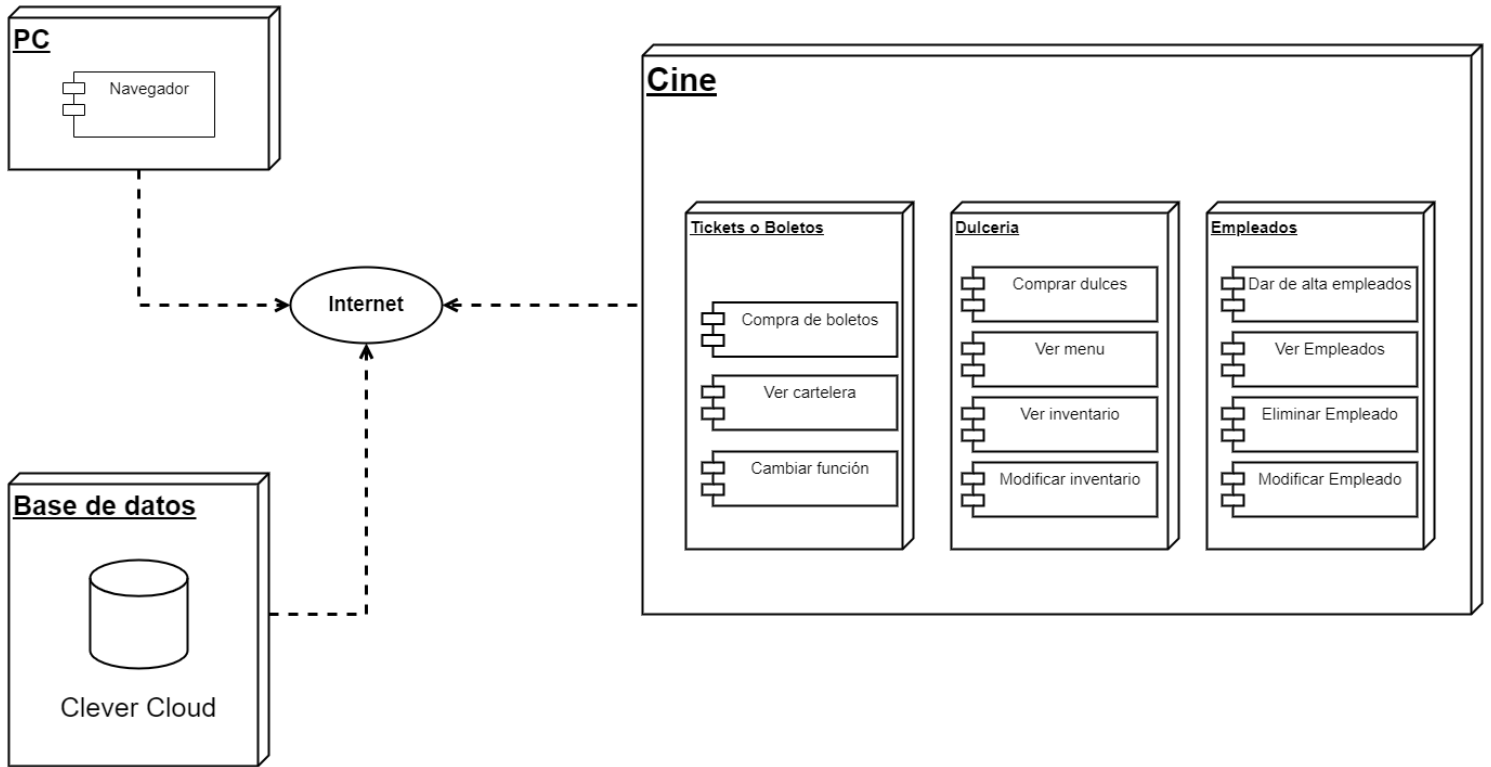


- **Mano de obra**

Para este punto no fue necesario el pago de sueldos ni la contratación debido a que somos cuatro desarrolladores trabajando en específicamente este servicio el cual fue dividido por actividades y por un lapso de tiempo considerable no se requiere mano de obra extra.



Diagrama de despliegue



Documentación de API SOAP y REST.

Microservicio de Tickets y Boletos

EndPoint

“/crearfunción”

<https://microservicio-funcion.herokuapp.com/crearFuncion>

```
//{"nombrePelicula":"El Libro De La Selva","hora":"10:45","fecha":...}
@PostMapping("/crearFuncion")
public void crearFuncion(@RequestBody Funcion funcion){
    ifuncion.save(funcion);
}
```

“/comprarBoletos”

Para el recurso de comprar un boleto para alguna funcion se hace uso de un envio post pero también se necesita pasar un parametro por la URI aquí se pasa el id de la funcion de la cual se quiere comprar boletos)

<https://microservicio-funcion.herokuapp.com/comprarBoletos?idFuncion=1>

```
@PostMapping("/comprarBoletos")
public boolean comprarBoletos(@RequestBody CompraBoleto compraBoletoRecibido , @RequestParam int idFuncion) {
    CompraBoleto compraBoleto = compraBoletoRecibido;
    Optional<Funcion> funcionOptional = ifuncion.findById(idFuncion);
    Funcion funcion = funcionOptional.get();
    if(funcion.getAsientosDisponibles() < compraBoletoRecibido.getCantidad()){
        return false;
    }else{
        compraBoleto.setCantidad(compraBoletoRecibido.getCantidad());
        compraBoleto.setFuncion(funcion);
        compraBoleto.setTotal(compraBoletoRecibido.getCantidad() * funcion.getPrecio());
        funcion.setAsientosDisponibles(funcion.getAsientosDisponibles()-compraBoletoRecibido.getCantidad());
        icompraBoleto.save(compraBoleto);
        ifuncion.save(funcion);
        return true;
    }
}
```


“/verCartelera”

Para este recurso solamente es necesario consumirlo con la siguiente URL:

<https://microservicio-funcion.herokuapp.com/verCartelera>

```
@GetMapping("/verCartelera")
public List<Funcion> verCartelera(){
    List<Funcion> cartelera = new ArrayList<>();
    Iterable<Funcion> carteleraIterable = ifuncion.findAll();
    for (Funcion funcion : carteleraIterable) {
        if(funcion.getAsientosDisponibles()>0){
            cartelera.add(funcion);
        }
    }
    return cartelera;
}
```

“/cambiarFunción”

Aquí no es necesario hacer uso de un archivo JSON ya que lo que requerimos son parametros, el id de la compra de los boletos y la nueva funcion a la que se desea cambiar.

localhost:8080/cambiarFuncion?idCompra=2&idNuevaFuncion=7

```
@PutMapping("/cambiarFuncion")
public String cambiarFuncion(@RequestParam(name="idCompra") int idCompra , @RequestParam(name = "idNuevaFuncion") int idNuevaFuncion){
    Optional<CompraBoleto> compraBoletoOptional = icompraBoleto.findById(idCompra);
    CompraBoleto compraBoleto = compraBoletoOptional.get();
    Optional<Funcion> funcionOptional2 = ifuncion.findById(idNuevaFuncion);
    Funcion funcionNueva = funcionOptional2.get();
    if(funcionNueva.getAsientosDisponibles()>compraBoleto.getCantidad()){
        Funcion funcionAntigua = compraBoleto.getFuncion();
        funcionAntigua.setAsientosDisponibles(funcionAntigua.getAsientosDisponibles()+compraBoleto.getCantidad());
        funcionNueva.setAsientosDisponibles(funcionAntigua.getAsientosDisponibles()-compraBoleto.getCantidad());
        compraBoleto.setFuncion(funcionNueva);
        icompraBoleto.save(compraBoleto);
        ifuncion.save(funcionNueva);
        ifuncion.save(funcionAntigua);
        return "Funcion cambiada exitosamente";
    }else{
        return "No hay suficientes asientos para esta función";
    }
}
```

Microservicio de Empleados

EndPoint

“Dar de Alta Empleado”

Para poder de alta un empleado, se hace uso de lo que es el request y response en relación a un archivo XSD llamado Empleados. Lo que hacemos es entrar a la siguiente pagina en heroku: <https://cine-empleados.herokuapp.com/ws/Empleados.wsdl>

y despues hacemos uso de la extensión wizdler en donde daremos clic a la opción “RegistrarEmpleado” y llenaremos los campos correspondientes que son: Nombre, Apellido, Fecha de nacimiento, Direccion, Telefono, Sexo y Estado civil; todos estos datos los recibiremos con request y los guardaremos y mostraremos con response.

Todos esos datos los guardaremos en una base de datos en la nube la cual es “Clever cloud”

```
@PayloadRoot(localPart = "RegistrarEmpleadoRequest", namespace = "https://t4is.uv.mx/Empleados")
@ResponsePayload
public RegistrarEmpleadoResponse RegistrarEmpleado(@RequestPayload RegistrarEmpleadoRequest petition) {
    RegistrarEmpleadoResponse respuesta = new RegistrarEmpleadoResponse();

    respuesta.setRNombre(petition.getNombre());
    respuesta.setRApellido(petition.getApellido());
    respuesta.setRFechaNacimiento(petition.getFechaNacimiento());
    respuesta.setRDireccion(petition.getDireccion());
    respuesta.setRTelefono(petition.getTelefono());
    respuesta.setRSexo(petition.getSexo());
    respuesta.setREstadoCivil(petition.getEstadoCivil());

    Empleado empleado = new Empleado();

    empleado.setNombre(petition.getNombre());
    empleado.setApellido(petition.getApellido());
    empleado.setFechaNacimiento(petition.getFechaNacimiento());
    empleado.setDireccion(petition.getDireccion());
    empleado.setTelefono(petition.getTelefono());
    empleado.setSexo(petition.getSexo());
    empleado.setEstadoCivil(petition.getEstadoCivil());

    iempleado.save(empleado);
    return respuesta;
}
```

“Ver Empleados”

En este apartado lo que se podrá hacer es la visualización de los distintos empleados registrados en la base de datos Clever Cloud, en este caso no ingresaremos ningún dato por parte de request, sino nomas haremos la petición para extraer los datos de la base de datos y mostrarlos en pantalla con ayuda response y del wizdler

<https://cine-empleados.herokuapp.com/ws/Empleados.wsdl>

```
@PayloadRoot(localPart = "EmpleadosRequest", namespace = "https://t4is.uv.mx/Empleados")
@ResponsePayload
public EmpleadosResponse Eventos(){
    EmpleadosResponse respuesta = new EmpleadosResponse();
    Iterable<Empleado> lista = iempleado.findAll();
    for (Empleado empleado : lista) {
        EmpleadosResponse.Empleados EmpleadoBuscar = new EmpleadosResponse.Empleados();

        EmpleadoBuscar.setId(empleado.getId());
        EmpleadoBuscar.setNombre(empleado.getNombre());
        EmpleadoBuscar.setApellido(empleado.getApellido());
        EmpleadoBuscar.setFechaNacimiento(empleado.getFechaNacimiento());
        EmpleadoBuscar.setDireccion(empleado.getDireccion());
        EmpleadoBuscar.setTelefono(empleado.getTelefono());
        EmpleadoBuscar.setSexo(empleado.getSexo());
        EmpleadoBuscar.setEstadoCivil(empleado.getEstadoCivil());

        respuesta.getEmpleados().add(EmpleadoBuscar);
    }
    return respuesta;
}
```

“Modificar Empleado”

Para poder modificar un empleado, se hace uso de lo que es el request y response en relación con un archivo XSD llamada Empleados. Lo que hacemos es entrar a la siguiente [página en heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl](https://cine-empleados.herokuapp.com/ws/Empleados.wsdl)

Y después hacemos uso de la extensión wizzler en donde daremos clic a la opción “ModificarEmpleado” y llenaremos los campos correspondientes, introduciendo el id del empleado que le deseamos modificarle los datos y los datos nuevos a reemplazar como son: Nombre, Apellido, Fecha de nacimiento, Dirección, Teléfono, Sexo y Estado civil; todos estos datos los recibiremos con request y de ahí se guardan en la base de datos de la nube Clever cloud y con response mostramos los datos ya actualizados.

```
@PayloadRoot(localPart = "ModificarEmpleadoRequest", namespace = "https://t4is.uv.mx/Empleados")
@ResponsePayload
public ModificarEmpleadoResponse modificarEmpleado(@RequestPayload ModificarEmpleadoRequest petition) {
    ModificarEmpleadoResponse respuesta = new ModificarEmpleadoResponse();

    Empleado empleado = new Empleado();

    empleado.setId(petition.getId());
    empleado.setNombre(petition.getNuevoNombre());
    empleado.setApellido(petition.getNuevoApellido());
    empleado.setFechaNacimiento(petition.getNuevaFechaNacimiento());
    empleado.setDireccion(petition.getNuevaDireccion());
    empleado.setTelefono(petition.getNuevoTelefono());
    empleado.setSexo(petition.getNuevoSexo());
    empleado.setEstadoCivil(petition.getNuevoEstadoCivil());

    iempleado.save(empleado);

    respuesta.setId(petition.getId());
    respuesta.setNombre(petition.getNuevoNombre());
    respuesta.setApellido(petition.getNuevoApellido());
    respuesta.setFechaNacimiento(petition.getNuevaFechaNacimiento());
    respuesta.setDireccion(petition.getNuevaDireccion());
    respuesta.setTelefono(petition.getNuevoTelefono());
    respuesta.setSexo(petition.getNuevoSexo());
    respuesta.setEstadoCivil(petition.getNuevoEstadoCivil());

    return respuesta;
}
```

“Eliminar Empleado”

Para poder Eliminar un empleado, se hace uso de lo que es el request y response en relación a un archivo XSD llamado Empleados. Lo que hacemos es entrar a la siguiente [pagina en heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl](https://cine-empleados.herokuapp.com/ws/Empleados.wsdl)

Y despues hacemos uso de la extensión wizzler en donde daremos clic a la opción “EliminarEmpleado” y llenaremos el campo correspondiente que en este caso es el id del empleado que deseamos eliminar. Despues de haber eliminado el empleado deseado, se nos mostrara con el response la lista de empleados que se encuentra en la base de datos Clever Cloud menos el empleado eliminado.

```
@PayloadRoot(localPart = "EliminarEmpleadoRequest", namespace = "https://t4is.uv.mx/Empleados")
@ResponsePayload
public EliminarEmpleadoResponse borrarEmpleado(@RequestPayload EliminarEmpleadoRequest peticion){

    EliminarEmpleadoResponse respuesta = new EliminarEmpleadoResponse();
    respuesta.getEmpleado().clear();

    //Eliminar datos en la base de datos de la tabla agenda
    iempleado.deleteById(peticion.getId());

    Iterable<Empleado> lista = iempleado.findAll();
    for (Empleado empleado : lista) {
        EliminarEmpleadoResponse.Empleado EmpleadoEliminar = new EliminarEmpleadoResponse.Empleado();

        EmpleadoEliminar.setId(empleado.getId());
        EmpleadoEliminar.setNombre(empleado.getNombre());
        EmpleadoEliminar.setApellido(empleado.getApellido());
        EmpleadoEliminar.setFechaNacimiento(empleado.getFechaNacimiento());
        EmpleadoEliminar.setDireccion(empleado.getDireccion());
        EmpleadoEliminar.setTelefono(empleado.getTelefono());
        EmpleadoEliminar.setSexo(empleado.getSexo());
        EmpleadoEliminar.setEstadoCivil(empleado.getEstadoCivil());

        respuesta.getEmpleado().add(EmpleadoEliminar);
    }

    return respuesta;
}
```

Microservicio de Dulcería

EndPoint

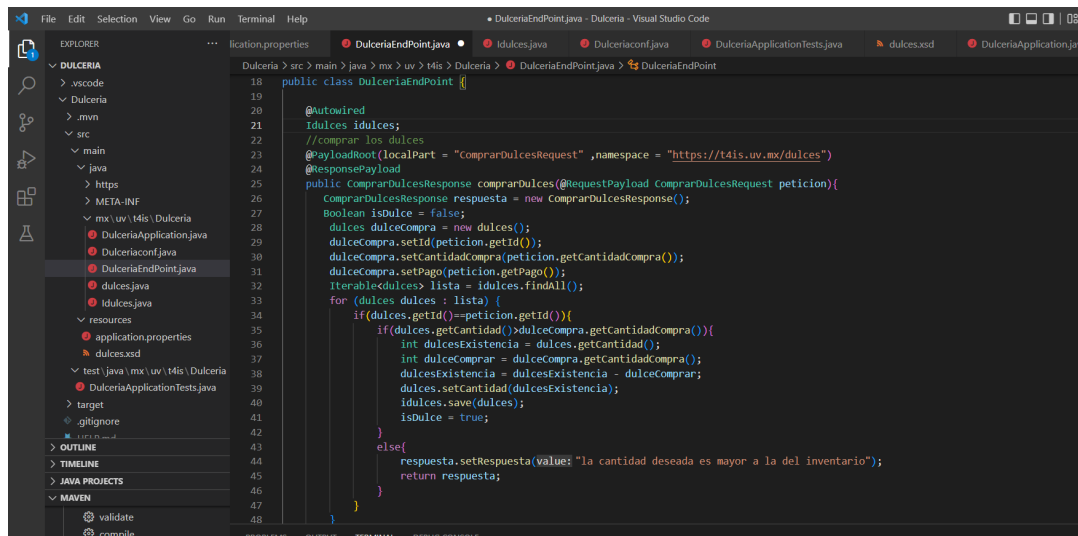
“Comprar Dulces”

Para poder comprar un dulce, se hace uso de lo que es el request y response en un archivo XSD llamado dulces. Para poder hacer dicha petición necesitamos entrar a nuestra página en heroku:

<https://dulceriati.herokuapp.com/ws/dulces.wsdl>

al estar dentro hacemos uso de la extensión widdler en donde daremos clic a la opción “ComprarDulces” y llenaremos los campos con los datos que nos pide, que son: id(del dulce), cantidad compra, pago(total de compra); todos estos datos los recibiremos con request y los guardaremos y mostraremos un mensaje de compra con response.

Los datos de los dulces están previamente guardados en un proveedor de base de datos llamado clever cloud



```
18 public class DulceriaEndPoint {
19
20     @Autowired
21     Idulces idulces;
22     //comprar los dulces
23     @PayloadRoot(localPart = "ComprarDulcesRequest", namespace = "https://t4is.uv.mx/dulces")
24     @ResponsePayload
25     public ComprarDulcesResponse comprarDulces(@RequestPayload ComprarDulcesRequest peticion){
26         ComprarDulcesResponse respuesta = new ComprarDulcesResponse();
27         Boolean isDulce = false;
28         dulces dulceCompra = new dulces();
29         dulceCompra.setId(peticion.getId());
30         dulceCompra.setCantidadCompra(peticion.getCantidadCompra());
31         dulceCompra.setPago(peticion.getPago());
32         Iterable<Idulces> lista = idulces.findAll();
33         for (Idulces dulces : lista) {
34             if (dulces.getId() == peticion.getId()) {
35                 if (dulces.getCantidad() > dulceCompra.getCantidadCompra()) {
36                     int dulceExistencia = dulces.getCantidad();
37                     int dulceComprar = dulceCompra.getCantidadCompra();
38                     dulceExistencia = dulceExistencia - dulceComprar;
39                     dulces.setCantidad(dulceExistencia);
40                     idulces.save(dulces);
41                     isDulce = true;
42                 }
43             }
44             else {
45                 respuesta.setRespuesta(valor: "la cantidad deseada es mayor a la del inventario");
46                 return respuesta;
47             }
48         }
49     }
50 }
```

“mostrarDulcesMenu”

En este apartado lo que se podrá hacer es la visualización de los distintos dulces que hay en el menú, los cuales están en la base de datos Clever Cloud, en este caso no ingresaremos datos por parte de request, solo haremos la petición para extraer los datos de la base de datos y mostrarlos en pantalla con ayuda del response

<https://dulceriati.herokuapp.com/ws/dulces.wsdl>

```

51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
DulceriaEndPoint.java
Dulceria
src
main
java
https
META-INF
DulceriaApplication.java
DulceriaConf.java
DulceriaEndPoint.java
dulces.java
idulces.java
resources
application.properties
dulces.xsd
test
java
mx
uv
t4is
Dulceria
DulceriaApplicationTests.java
target
.gitignore
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
//mostrar el menu de los dulces
@PayloadRoot(localPart = "MostrarDulcesMenuRequest", namespace = "https://t4is.uv.mx/dulces")
@ResponsePayload
public MostrarDulcesMenuResponse mostrarDulces(){
    MostrarDulcesMenuResponse respuesta = new MostrarDulcesMenuResponse();
    //se genera una lista
    Iterable<Dulces> lista = idulces.findAll();
    //se recorre la lista para sacar los dulces
    for (Dulces dulces : lista) {
        MostrarDulcesMenuResponse.Dulces e = new MostrarDulcesMenuResponse.Dulces();
        //se obtienen los dulces
        e.setId(dulces.getId());
        e.setNombreProducto(dulces.getNombreProducto());
        e.setPrecio(dulces.getPrecioProducto());
        //los guarda dulces
        respuesta.getDulces().add(e);
    }
    //regresa la respuesta
    return respuesta;
}
//mostrar el inventario de los dulces

```

“ModificarInventario”

Para poder modificar un dulce, usamos el request y response en un archivo XSD llamada dulces. Lo que hacemos es entrar a la siguiente página en heroku: <https://dulceriati.herokuapp.com/ws/dulces.wsdl>

hacemos uso de la extensión wizdler en donde daremos clic a la opción “ModificarInventario” y llenaremos los campos correspondientes, introduciendo el id del dulce que queremos modificar, y los datos nuevos todos estos datos los recibiremos con request y de se guardan el base de datos de la nube y con response mostramos un mensaje que los datos han sido actualizados.

```

100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
DulceriaEndPoint.java
Dulceria
src
main
java
https
META-INF
DulceriaApplication.java
DulceriaConf.java
DulceriaEndPoint.java
dulces.java
idulces.java
resources
application.properties
dulces.xsd
test
java
mx
uv
t4is
Dulceria
DulceriaApplicationTests.java
target
.gitignore
OUTLINE
TIMELINE
//modificacion de los dulces en el inventario
@PayloadRoot(localPart = "ModificarInventarioRequest", namespace = "https://t4is.uv.mx/dulces")
@ResponsePayload
public ModificarInventarioResponse modificarsaludo(@RequestPayload ModificarInventarioRequest petition){
    ModificarInventarioResponse respuesta = new ModificarInventarioResponse();
    Dulces dulces = new Dulces();
    dulces.setId(petition.getId());
    dulces.setNombreProducto(petition.getNombreProducto());
    dulces.setPrecioProducto(petition.getPrecio());
    dulces.setCantidad(petition.getCantidad());
    dulces.setDisponibilidad(petition.getDisponibilidad());
    idulces.save(dulces);
    respuesta.setRespuesta(value: "dulce modificado exitosamente");
    return respuesta;
}

```

“mostrarDulcesInventario”

En este apartado lo que se podrá hacer es la visualización de los distintos dulces que se encuentran dados de alta en el inventario. De nueva cuenta, en este caso no ingresaremos datos por parte de request, solo haremos la petición para extraer los datos de la base de datos y mostrarlos en pantalla con ayuda del response

<https://cine-empleados.herokuapp.com/ws/Empleados.wsdl>

```

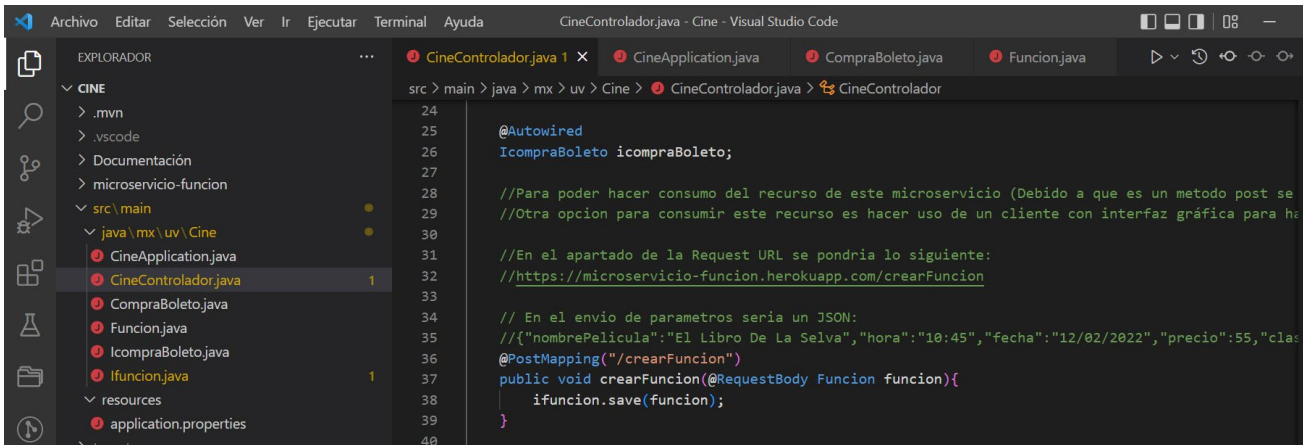
76 }
77
78 //mostrar el inventario de los dulces
79 @PayloadRoot(localPart = "MostrarDulcesInventarioRequest", namespace = "https://t4is.uv.mx/dulces")
80 @ResponsePayload
81 public MostrarDulcesInventarioResponse mostrarDulcesInventario(){
82     MostrarDulcesInventarioResponse respuesta = new MostrarDulcesInventarioResponse();
83     //se genera una lista
84     Iterable<Dulces> lista = idulces.findAll();
85     //se retorna la lista para sacar del inventario los dulces
86     for (Dulces dulces : lista) {
87         MostrarDulcesInventarioResponse.DulcesInventario e = new MostrarDulcesInventarioResponse.DulcesInventario();
88         //se obtienen los dulces dle inventario
89         e.setId(dulces.getId());
90         e.setNombreProducto(dulces.getNombreProducto());
91         e.setPrecio(dulces.getPrecioProducto());
92         e.setDisponibilidad(dulces.getDisponibilidad());
93         //los guarda
94         respuesta.getDulcesInventario().add(e);
95     }
96     //regresa la respuesta
97     return respuesta;
98 }
99
100
101

```


Parámetros de recepción.

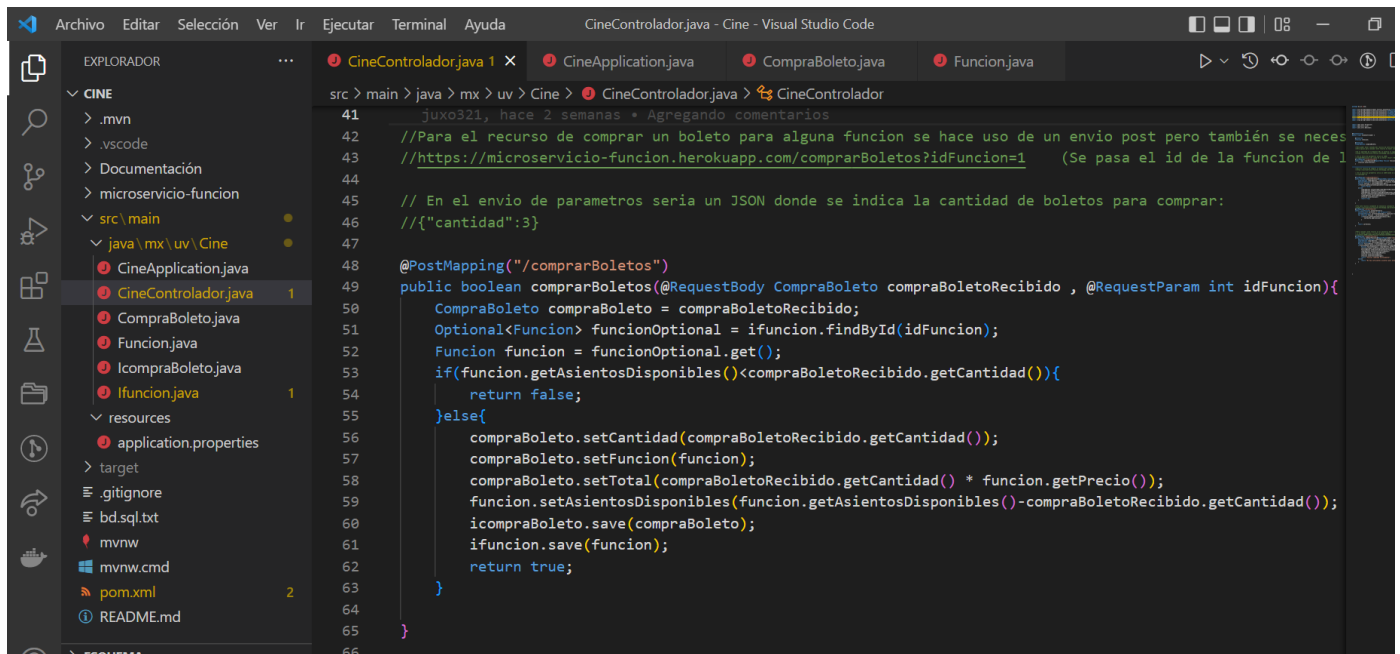
Microservicio Funciones(request)

“/crearfunción”



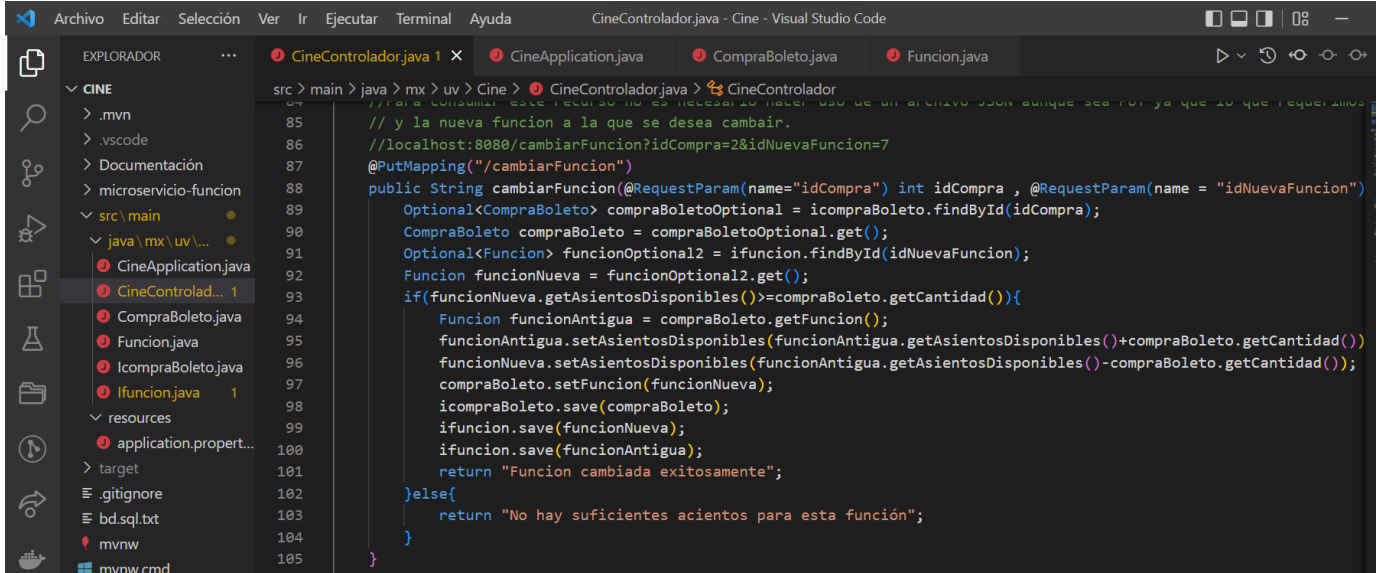
```
24
25 @Autowired
26 IcompraBoleto icompraBoleto;
27
28 //Para poder hacer consumo del recurso de este microservicio (Debido a que es un metodo post se
29 //Otra opcion para consumir este recurso es hacer uso de un cliente con interfaz gráfica para ha
30
31 //En el apartado de la Request URL se pondria lo siguiente:
32 //https://microservicio-funcion.herokuapp.com/crearFuncion
33
34 // En el envio de parametros seria un JSON:
35 //{ "nombrePelicula": "El Libro De La Selva", "hora": "10:45", "fecha": "12/02/2022", "precio": 55, "clas
36 @PostMapping("/crearFuncion")
37 public void crearFuncion(@RequestBody Function funcion){
38     ifuncion.save(funcion);
39 }
40
```

“/comprarBoletos”



```
41 //juxo321, hace 2 semanas * Agregando comentarios
42 //Para el recurso de comprar un boleto para alguna funcion se hace uso de un envio post pero también se neces
43 //https://microservicio-funcion.herokuapp.com/comprarBoletos?idFuncion=1 (Se pasa el id de la funcion de l
44
45 // En el envio de parametros seria un JSON donde se indica la cantidad de boletos para comprar:
46 //{ "cantidad": 3}
47
48 @PostMapping("/comprarBoletos")
49 public boolean comprarBoletos(@RequestBody CompraBoleto compraBoletoRecibido, @RequestParam int idFuncion){
50     CompraBoleto compraBoleto = compraBoletoRecibido;
51     Optional<Function> funcionOptional = ifuncion.findById(idFuncion);
52     Function funcion = funcionOptional.get();
53     if(funcion.getAsientosDisponibles() < compraBoletoRecibido.getCantidad()){
54         return false;
55     }else{
56         compraBoleto.setCantidad(compraBoletoRecibido.getCantidad());
57         compraBoleto.setFuncion(funcion);
58         compraBoleto.setTotal(compraBoletoRecibido.getCantidad() * funcion.getPrecio());
59         funcion.setAsientosDisponibles(funcion.getAsientosDisponibles() - compraBoletoRecibido.getCantidad());
60         icompraBoleto.save(compraBoleto);
61         ifuncion.save(funcion);
62         return true;
63     }
64 }
65
66
```

“/cambiarFunción”



```
src > main > java > mx > uv > Cine > CineControlador.java > CineControlador
85 // y la nueva función a la que se desea cambiar.
86 //localhost:8080/cambiarFuncion?idCompra=2&idNuevaFuncion=7
87 @PostMapping("/cambiarFuncion")
88 public String cambiarFuncion(@RequestParam(name="idCompra") int idCompra , @RequestParam(name = "idNuevaFuncion")
89 Optional<CompraBoleto> compraBoletoOptional = icompraBoleto.findById(idCompra);
90 CompraBoleto compraBoleto = compraBoletoOptional.get();
91 Optional<Funcion> funcionOptional2 = ifuncion.findById(idNuevaFuncion);
92 Funcion funcionNueva = funcionOptional2.get();
93 if(funcionNueva.getAsientosDisponibles()==compraBoleto.getCantidad()){
94     Funcion funcionAntigua = compraBoleto.getFuncion();
95     funcionAntigua.setAsientosDisponibles(funcionAntigua.getAsientosDisponibles()+compraBoleto.getCantidad());
96     funcionNueva.setAsientosDisponibles(funcionAntigua.getAsientosDisponibles()-compraBoleto.getCantidad());
97     compraBoleto.setFuncion(funcionNueva);
98     icompraBoleto.save(compraBoleto);
99     ifuncion.save(funcionNueva);
100     ifuncion.save(funcionAntigua);
101     return "Funcion cambiada exitosamente";
102 }else{
103     return "No hay suficientes acientos para esta función";
104 }
105 }
```

Microservicio Empleados (Request)

RegistrarEmpleado (Request)

Para poder registrar un Empleado se requerirá llenar los siguientes campos en el código XSD con ayuda de la extensión Wizrler, los campos son: Nombre, Apellido, Fecha de nacimiento, Dirección, Teléfono, Sexo y Estado Civil. Todos como campos String.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:definitions xmlns:xsd="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="https://t4is.uv.mx/Empleados" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="https://t4is.uv.mx/Empleados" targetNamespace="https://t4is.uv.mx/Empleados">
  <link type="text/css" rel="stylesheet" id="dark-mode-custom-link"/>
  <link type="text/css" rel="stylesheet" id="dark-mode-general-link"/>
  <style lang="en" type="text/css" id="dark-mode-custom-style"/>
  <style lang="en" type="text/css" id="dark-mode-native-style"/>
  <xsd:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="https://t4is.uv.mx/Empleados">
      <xsd:element name="RegistrarEmpleadoRequest">
        <xsd:complexType base="xsd:sequence">
          <xsd:sequence>
            <xsd:element name="Nombre" type="xsd:string"/>
            <xsd:element name="Apellido" type="xsd:string"/>
            <xsd:element name="FechaNacimiento" type="xsd:string"/>
            <xsd:element name="Direccion" type="xsd:string"/>
            <xsd:element name="Telefono" type="xsd:string"/>
            <xsd:element name="Sexo" type="xsd:string"/>
            <xsd:element name="EstadoCivil" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </xsd:types>
</xsd:definitions>
```

Código Empleados.xsd

```

POST https://cine-empleados.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <RegistrarEmpleadoRequest xmlns="https://t4is.uv.mx/Empleados">
      <Nombre>Rafael</Nombre>
      <Apellido>Rojano</Apellido>
      <FechaNacimiento>15 mayo del 1985</FechaNacimiento>
      <Direccion>Coatepec</Direccion>
      <Telefono>1234567890</Telefono>
      <Sexo>Masculino</Sexo>
      <EstadoCivil>Casado</EstadoCivil>
    </RegistrarEmpleadoRequest>
  </Body>
</Envelope>

```

Código RegistrarEmpleado (Request)

Empleados (Request)

En este apartado no se requiere como tal llenar un campo, sino solo darle al botón “Go” para poder mandar la petición de que se nos muestre la lista de empleados que están registrados en la Base de datos de Clever Cloud

```

<xs:element name="EmpleadosRequest"/>

```

Código Empleados.xsd

```

POST https://cine-empleados.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <EmpleadosRequest xmlns="https://t4is.uv.mx/Empleados">[any]</EmpleadosRequest>
  </Body>
</Envelope>

```

Código Empleados (request)

ModificarEmpleado (Request)

Para poder modificar un Empleado se requerirá llenar los siguientes campos en el código XSD con ayuda de la extensión Wizdler, los campos son: id, NuevoNombre, NuevoApellido, NuevaFechaNacimiento, NuevaDireccion, NuevoTelefono, NuevoSexo y NuevoEstadoCivil. Todos estos campos serán String menos el id, el será un Int (Entero)

En el caso del campo del id, hay debemos ingresar el id del Empleado al que vamos a modificar sus datos

```

▼<xs:element name="ModificarEmpleadoRequest">
  ▼<xs:complexType>
    ▼<xs:sequence>
      <xs:element name="id" type="xs:int"/>
      <xs:element name="nuevoNombre" type="xs:string"/>
      <xs:element name="nuevoApellido" type="xs:string"/>
      <xs:element name="nuevaFechaNacimiento" type="xs:string"/>
      <xs:element name="nuevaDireccion" type="xs:string"/>
      <xs:element name="nuevoTelefono" type="xs:string"/>
      <xs:element name="nuevoSexo" type="xs:string"/>
      <xs:element name="nuevoEstadoCivil" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Código Empleados.xsd

```

POST https://cine-empleados.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <ModificarEmpleadoRequest xmlns="https://t4is.uv.mx/Empleados">
      <id>[int]</id>
      <nuevoNombre>[string]</nuevoNombre>
      <nuevoApellido>[string]</nuevoApellido>
      <nuevaFechaNacimiento>[string]</nuevaFechaNacimiento>
      <nuevaDireccion>[string]</nuevaDireccion>
      <nuevoTelefono>[string]</nuevoTelefono>
      <nuevoSexo>[string]</nuevoSexo>
      <nuevoEstadoCivil>[string]</nuevoEstadoCivil>
    </ModificarEmpleadoRequest>
  </Body>
</Envelope>

```

Código ModificarEmpleado (resquest)

EliminarEmpleado (Request)

Para poder Eliminar un Empleado se requerirá llenar el siguiente campo en el código XSD con ayuda de la extensión Wzldler, el campo es el id del Empleado que deseamos eliminar.

```

▼<xs:element name="EliminarEmpleadoRequest">
  ▼<xs:complexType>
    ▼<xs:sequence>
      <xs:element name="id" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Código Empleados.xsd

```

POST https://cine-empleados.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <EliminarEmpleadoRequest xmlns="https://t4is.uv.mx/Empleados">
      <id>[int]</id>
    </EliminarEmpleadoRequest>
  </Body>
</Envelope>

```

Código EliminarEmpleado (request).

Microservicio Dulceria (Request)

comprarDulces (Request)

Para poder comprar un dulce se requerirá llenar campos en el código XSD con ayuda de la extensión Wizdler, los campos son: id, cantidadCompra(total de dulces), pago(total de la compra). Donde todos los campos son de tipo entero

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsc:definitions xmlns:wsc="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="https://t4is.uv.mx/dulces" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="https://t4is.uv.mx/dulces" targetNamespace="https://t4is.uv.mx/dulces">
  <wsc:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="https://t4is.uv.mx/dulces">
      <!-- comprar dulces -->
      <xsd:element name="ComprarDulcesRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="id" type="xsd:int"/>
            <xsd:element name="cantidadCompra" type="xsd:int"/>
            <xsd:element name="pago" type="xsd:int"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsc:types>
</wsc:definitions>

```

Código comprar dulces en el xsd

```

POST https://dulceriati.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <ComprarDulcesRequest xmlns="https://t4is.uv.mx/dulces">
      <id>[int]</id>
      <cantidadCompra>[int]</cantidadCompra>
      <pago>[int]</pago>
    </ComprarDulcesRequest>
  </Body>
</Envelope>

```

Código comprarDulces(request)

mostrarDulcesMenu (Request)

En este apartado no se requiere llenar campos, solo darle al botón “Go” para poder mandar la petición de que se nos muestre la lista de dulces del menú que ya están registrados en nuestra base de datos

```
</xs:element>
<!-- Mostrar Menu -->
<xs:element name="MostrarDulcesMenuRequest"/>
```

Código mostrarDulcesMenu en el xsd

```
POST https://dulceriati.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <MostrarDulcesMenuRequest xmlns="https://t4is.uv.mx/dulces">[any]</MostrarDulcesMenuRequest>
  </Body>
</Envelope>
```

Código mostrarDulcesMenu(request)

mostrarDulcesInventario (Request)

En este apartado no se requiere llenar campos, solo darle al botón “Go” para poder mandar la petición de que se nos muestre la lista de los dulces que ya están registrados en nuestra base de datos

```
</xs:element>
<!-- Mostrar Inventario -->
<xs:element name="MostrarDulcesInventarioRequest"/>
```

Código mostrarDulcesInventario en el xsd

```
POST https://dulceriati.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <MostrarDulcesInventarioRequest xmlns="https://t4is.uv.mx/dulces">[any]</MostrarDulcesInventarioRequest>
  </Body>
</Envelope>
```

Código mostrarDulcesInventario(request)

ModificarEmpleado (Request)

Equipo 2

Para poder modificar un dulce se requiere llenar campos en el código XSD con ayuda de la extensión Wizzler, los campos son: id, NombreProducto, cantidad, precio y disponibilidad. Todos estos campos serán String menos el id, cantidad y precio

En el id, debemos ingresar el id del dulce al que vamos a modificar sus datos

```
<!-- Modificar inventario -->
<xs:element name="ModificarInventarioRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="xs:int"/>
      <xs:element name="nombreProducto" type="xs:string"/>
      <xs:element name="cantidad" type="xs:int"/>
      <xs:element name="precio" type="xs:int"/>
      <xs:element name="disponibilidad" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ModificarInventarioResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="respuesta" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
</wsdl:types>
```

Código ModificarInventario en el xsd

```
POST https://dulceriati.herokuapp.com:443/ws
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <ModificarInventarioRequest xmlns="https://t4is.uv.mx/dulces">
      <id>[int]</id>
      <nombreProducto>[string]</nombreProducto>
      <cantidad>[int]</cantidad>
      <precio>[int]</precio>
      <disponibilidad>[string]</disponibilidad>
    </ModificarInventarioRequest>
  </Body>
</Envelope>
```

Código ModificarInventario(request)

Parámetros devueltos

Microservicio Tickets y Funciones (Response)

Microservicio Empleados (Response)

RegistrarEmpleado (Response)

La respuesta que recibimos después de aplicar RegistrarEmpleado (request), es un mensaje mostrando todos los campos con los datos del nuevo empleado que registramos en la base de datos y el estatus de la operación.

```
<?xml version='1.0' encoding='utf-8'>
  <xs:element name="RegistrarEmpleadoResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RNombre" type="xs:string"/>
        <xs:element name="RApellido" type="xs:string"/>
        <xs:element name="RFechaNacimiento" type="xs:string"/>
        <xs:element name="RDireccion" type="xs:string"/>
        <xs:element name="RTelefono" type="xs:string"/>
        <xs:element name="RSexo" type="xs:string"/>
        <xs:element name="REstadoCivil" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Código Empleados.xsd

```
POST https://cine-empleados.herokuapp.com:443/ws
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:RegistrarEmpleadoResponse xmlns:ns2="https://t4is.uv.mx/Empleados">
      <ns2:RNombre>Rafael</ns2:RNombre>
      <ns2:RApellido>Rojano</ns2:RApellido>
      <ns2:RFechaNacimiento>15 mayo del 1985</ns2:RFechaNacimiento>
      <ns2:RDireccion>Coatepec</ns2:RDireccion>
      <ns2:RTelefono>1234567890</ns2:RTelefono>
      <ns2:RSexo>Masculino</ns2:RSexo>
      <ns2:REstadoCivil>Casado</ns2:REstadoCivil>
    </ns2:RegistrarEmpleadoResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Código RegistrarEmpleado (Response)

Empleados (response)

La respuesta que recibimos en este apartado es la lista de empleados que se encuentra almacenada en la base de datos clever cloud y el estatus de la operación.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:element name="EmpleadosResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="Empleados">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:int"/>
            <xs:element name="Nombre" type="xs:string"/>
            <xs:element name="Apellido" type="xs:string"/>
            <xs:element name="FechaNacimiento" type="xs:string"/>
            <xs:element name="Direccion" type="xs:string"/>
            <xs:element name="Telefono" type="xs:string"/>
            <xs:element name="Sexo" type="xs:string"/>
            <xs:element name="EstadoCivil" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Código Empleados.xsd

POST <https://cine-empleados.herokuapp.com:443/ws>

```
<ns2:EstadoCivil>Soltero</ns2:EstadoCivil>
</ns2:Empleados>
<ns2:Empleados>
  <ns2:id>6</ns2:id>
  <ns2:Nombre>Yarel</ns2:Nombre>
  <ns2:Apellido>Baizabal</ns2:Apellido>
  <ns2:FechaNacimiento>1 de enero del 2001</ns2:FechaNacimiento>
  <ns2:Direccion>Alto Tio Diego</ns2:Direccion>
  <ns2:Telefono>1234509876</ns2:Telefono>
  <ns2:Sexo>Femenino</ns2:Sexo>
  <ns2:EstadoCivil>Soltero</ns2:EstadoCivil>
</ns2:Empleados>
<ns2:Empleados>
  <ns2:id>7</ns2:id>
  <ns2:Nombre>Adbel</ns2:Nombre>
  <ns2:Apellido>Hernandez</ns2:Apellido>
  <ns2:FechaNacimiento>16 de diciembre del 2001</ns2:FechaNacimiento>
  <ns2:Direccion>Xalapa</ns2:Direccion>
  <ns2:Telefono>2233445566</ns2:Telefono>
  <ns2:Sexo>Masculino</ns2:Sexo>
  <ns2:EstadoCivil>Casado</ns2:EstadoCivil>
</ns2:Empleados>
<ns2:Empleados>
  <ns2:id>8</ns2:id>
  <ns2:Nombre>Rafael</ns2:Nombre>
  <ns2:Apellido>Rojano</ns2:Apellido>
  <ns2:FechaNacimiento>15 mayo del 1985</ns2:FechaNacimiento>
  <ns2:Direccion>Coatepec</ns2:Direccion>
  <ns2:Telefono>1234567890</ns2:Telefono>
  <ns2:Sexo>Masculino</ns2:Sexo>
  <ns2:EstadoCivil>Casado</ns2:EstadoCivil>
</ns2:Empleados>
</ns2:EmpleadosResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request Response Done (245 ms).

Código Empleados (response)

ModificarEmpleado (Response)

La respuesta que recibimos después de aplicar ModificarEmpleado (request), es un mensaje mostrando todos los campos de un empleado con nuevos datos que se modificaron en la base de datos Clever cloud. También recibimos el estatus de la operación.

```
</xs:element>
▼<xs:element name="ModificarEmpleadoResponse">
  ▼<xs:complexType>
    ▼<xs:sequence>
      <xs:element name="id" type="xs:int"/>
      <xs:element name="Nombre" type="xs:string"/>
      <xs:element name="Apellido" type="xs:string"/>
      <xs:element name="FechaNacimiento" type="xs:string"/>
      <xs:element name="Direccion" type="xs:string"/>
      <xs:element name="Telefono" type="xs:string"/>
      <xs:element name="Sexo" type="xs:string"/>
      <xs:element name="EstadoCivil" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
▼<xs:element name="EliminarEmpleadoRequest">
  ▼<xs:complexType>
    ▼<xs:sequence>
      <xs:element name="id" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Código Empleados.xsd

```
POST https://cine-empleados.herokuapp.com:443/ws
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:ModificarEmpleadoResponse xmlns:ns2="https://t4is.uv.mx/Empleados">
      <ns2:id>2</ns2:id>
      <ns2:Nombre>Bryan</ns2:Nombre>
      <ns2:Apellido>Piquet</ns2:Apellido>
      <ns2:FechaNacimiento>13 de mayo del 2000</ns2:FechaNacimiento>
      <ns2:Direccion>Alto Lucero</ns2:Direccion>
      <ns2:Telefono>5647102938</ns2:Telefono>
      <ns2:Sexo>Masculino</ns2:Sexo>
      <ns2:EstadoCivil>Soltero</ns2:EstadoCivil>
    </ns2:ModificarEmpleadoResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Código ModificarEmpleado (response)

EliminarEmpleado (Response)

La respuesta que recibimos es la lista de empleados ya actualizada desde la base de datos Clever Cloud, mostrando los datos de cada empleado mediante los distintos campos como son el nombre, apellido, etc.

También lo recibimos como respuesta es el estatus de la operación.

```
<xs:element name="EliminarEmpleadoResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="Empleado">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:int"/>
            <xs:element name="Nombre" type="xs:string"/>
            <xs:element name="Apellido" type="xs:string"/>
            <xs:element name="FechaNacimiento" type="xs:string"/>
            <xs:element name="Direccion" type="xs:string"/>
            <xs:element name="Telefono" type="xs:string"/>
            <xs:element name="Sexo" type="xs:string"/>
            <xs:element name="EstadoCivil" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Código Empleados.xsd

POST https://cine-empleados.herokuapp.com:443/ws

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:EliminarEmpleadoResponse xmlns:ns2="https://t4is.uv.mx/Empleados">
      <ns2:Empleado>
        <ns2:id>1</ns2:id>
        <ns2:Nombre>Ariana</ns2:Nombre>
        <ns2:Apellido>Tejeda</ns2:Apellido>
        <ns2:FechaNacimiento>10 de octubre del 2001</ns2:FechaNacimiento>
        <ns2:Direccion>Xalapa</ns2:Direccion>
        <ns2:Telefono>1234567890</ns2:Telefono>
        <ns2:Sexo>Femenino</ns2:Sexo>
        <ns2:EstadoCivil>Soltero</ns2:EstadoCivil>
      </ns2:Empleado>
      <ns2:Empleado>
        <ns2:id>2</ns2:id>
        <ns2:Nombre>Bryan</ns2:Nombre>
        <ns2:Apellido>Piquet</ns2:Apellido>
        <ns2:FechaNacimiento>13 de mayo del 2000</ns2:FechaNacimiento>
        <ns2:Direccion>Alto Lucero</ns2:Direccion>
        <ns2:Telefono>5647102938</ns2:Telefono>
        <ns2:Sexo>Masculino</ns2:Sexo>
        <ns2:EstadoCivil>Soltero</ns2:EstadoCivil>
      </ns2:Empleado>
      <ns2:Empleado>
        <ns2:id>3</ns2:id>
        <ns2:Nombre>Justin</ns2:Nombre>
        <ns2:Apellido>Hernandez</ns2:Apellido>
        <ns2:FechaNacimiento>12 de enero del 2001</ns2:FechaNacimiento>
        <ns2:Direccion>El Castillo</ns2:Direccion>
        <ns2:Telefono>0987654321</ns2:Telefono>
        <ns2:Sexo>Masculino</ns2:Sexo>
        <ns2:EstadoCivil>Casado</ns2:EstadoCivil>
      </ns2:Empleado>
    </ns2:EliminarEmpleadoResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request Response Done (387 ms)

POST https://cine-empleados.herokuapp.com:443/ws

```
<ns2:EstadoCivil>Casado</ns2:EstadoCivil>
</ns2:Empleado>
<ns2:Empleado>
  <ns2:id>5</ns2:id>
  <ns2:Nombre>Horus</ns2:Nombre>
  <ns2:Apellido>Cabrera</ns2:Apellido>
  <ns2:FechaNacimiento>6 de noviembre del 1999</ns2:FechaNacimiento>
  <ns2:Direccion>Xalapa</ns2:Direccion>
  <ns2:Telefono>0987612345</ns2:Telefono>
  <ns2:Sexo>Masculino</ns2:Sexo>
  <ns2:EstadoCivil>Soltero</ns2:EstadoCivil>
</ns2:Empleado>
<ns2:Empleado>
  <ns2:id>7</ns2:id>
  <ns2:Nombre>Adebe1</ns2:Nombre>
  <ns2:Apellido>Hernandez</ns2:Apellido>
  <ns2:FechaNacimiento>16 de diciembre del 2001</ns2:FechaNacimiento>
  <ns2:Direccion>Xalapa</ns2:Direccion>
  <ns2:Telefono>2233445566</ns2:Telefono>
  <ns2:Sexo>Masculino</ns2:Sexo>
  <ns2:EstadoCivil>Casado</ns2:EstadoCivil>
</ns2:Empleado>
<ns2:Empleado>
  <ns2:id>8</ns2:id>
  <ns2:Nombre>Rafael</ns2:Nombre>
  <ns2:Apellido>Rojano</ns2:Apellido>
  <ns2:FechaNacimiento>15 mayo del 1985</ns2:FechaNacimiento>
  <ns2:Direccion>Coatepec</ns2:Direccion>
  <ns2:Telefono>1234567890</ns2:Telefono>
  <ns2:Sexo>Masculino</ns2:Sexo>
  <ns2:EstadoCivil>Casado</ns2:EstadoCivil>
</ns2:Empleado>
</ns2:EliminarEmpleadoResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request Response Done (387 ms)

Código EliminarEmpleado (response)

Microservicio Dulcería (Response)

comprarDulces (Response)

La respuesta que recibimos después de aplicar comprar (request), es un mensaje diciendo que la compra del dulce se ha hecho de manera satisfactoria

```
</xs:complexType>
</xs:element>
<xs:element name="ComprarDulcesResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="respuesta" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Código dulcesResponse en xsd

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:ComprarDulcesResponse xmlns:ns2="https://t4is.uv.mx/dulces">
      <ns2:respuesta>dulce comprado satisfactoriamente</ns2:respuesta>
    </ns2:ComprarDulcesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

ModificarInventario (response)

La respuesta que recibimos en este apartado es un mensaje en donde dice que el dulce ha sido modificado satisfactoriamente

```
</xs:element>
<xs:element name="ModificarInventarioResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="respuesta" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Código modificarInventario en xsd

```
POST https://dulceriati.herokuapp.com:443/ws
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:ModificarInventarioResponse xmlns:ns2="https://t4is.uv.mx/dulces">
      <ns2:respuesta>dulce modificado exitosamente</ns2:respuesta>
    </ns2:ModificarInventarioResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Código dulce (response)

mostrarDulcesInventario (Response)

La respuesta que recibimos al darle la opción “GO” es un detallado de todos los elementos que existen en el inventario, dentro de ellos se muestra el Id, nombre, precio, disponibilidad y la cantidad

```
<xs:element name="MostrarDulcesInventarioResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="dulcesInventario" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:int"/>
            <xs:element name="nombreProducto" type="xs:string"/>
            <xs:element name="precio" type="xs:int"/>
            <xs:element name="cantidad" type="xs:int"/>
            <xs:element name="disponibilidad" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- Modificar inventario -->
```

Código mostrarDulcesInventario en el xsd

```
POST https://dulceriati.herokuapp.com:443/ws
SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
  <ns2:MostrarDulcesInventarioResponse xmlns:ns2="https://t4is.uv.mx/dulces">
    <ns2:dulcesInventario>
      <ns2:id>1</ns2:id>
      <ns2:nombreProducto>sabritas</ns2:nombreProducto>
      <ns2:precio>10</ns2:precio>
      <ns2:cantidad>20</ns2:cantidad>
      <ns2:disponibilidad>en existencia</ns2:disponibilidad>
    </ns2:dulcesInventario>
    <ns2:dulcesInventario>
      <ns2:id>2</ns2:id>
      <ns2:nombreProducto>paleta payaso</ns2:nombreProducto>
      <ns2:precio>8</ns2:precio>
      <ns2:cantidad>6</ns2:cantidad>
      <ns2:disponibilidad>en existencia</ns2:disponibilidad>
    </ns2:dulcesInventario>
    <ns2:dulcesInventario>
      <ns2:id>3</ns2:id>
      <ns2:nombreProducto>mangomitas</ns2:nombreProducto>
      <ns2:precio>2</ns2:precio>
      <ns2:cantidad>65</ns2:cantidad>
      <ns2:disponibilidad>si hay</ns2:disponibilidad>
    </ns2:dulcesInventario>
    <ns2:dulcesInventario>
      <ns2:id>4</ns2:id>
      <ns2:nombreProducto>refresco</ns2:nombreProducto>
      <ns2:precio>14</ns2:precio>
      <ns2:cantidad>7</ns2:cantidad>
      <ns2:disponibilidad>si hay</ns2:disponibilidad>
    </ns2:dulcesInventario>
  </ns2:MostrarDulcesInventarioResponse>
</SOAP-ENV:Body>
/SOAP-ENV:Envelope
```

Código mostrarDulcesInventario (response)

mostrarDulcesMenu (Response)

La respuesta que recibimos es la lista de dulces que tenemos, teóricamente es la que se le muestra al cliente para que pueda visualizar que dulces hay. En este se puede observar el id, nombre y el precio

```
25 <xs:element name="MostrarDulcesMenuResponse">
26   <xs:complexType>
27     <xs:sequence>
28       <xs:element name="dulces" maxOccurs="unbounded">
29         <xs:complexType>
30           <xs:sequence>
31             <xs:element name="id" type="xs:int"/>
32             <xs:element name="nombreProducto" type="xs:string"/>
33             <xs:element name="Precio" type="xs:int"/>
34           </xs:sequence>
35         </xs:complexType>
36       </xs:element>
37     </xs:sequence>
38   </xs:complexType>
39 </xs:element>
```

Código mostrarDulces en xsd

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:MostrarDulcesMenuResponse xmlns:ns2="https://t4is.uv.mx/dulces">
      <ns2:dulces>
        <ns2:id>1</ns2:id>
        <ns2:nombreProducto>sabritas</ns2:nombreProducto>
        <ns2:Precio>10</ns2:Precio>
      </ns2:dulces>
      <ns2:dulces>
        <ns2:id>2</ns2:id>
        <ns2:nombreProducto>Mangomitas</ns2:nombreProducto>
        <ns2:Precio>3</ns2:Precio>
      </ns2:dulces>
      <ns2:dulces>
        <ns2:id>3</ns2:id>
        <ns2:nombreProducto>magnum</ns2:nombreProducto>
        <ns2:Precio>20</ns2:Precio>
      </ns2:dulces>
      <ns2:dulces>
        <ns2:id>4</ns2:id>
        <ns2:nombreProducto>palomitas</ns2:nombreProducto>
        <ns2:Precio>15</ns2:Precio>
      </ns2:dulces>
    </ns2:MostrarDulcesMenuResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Código mostrarDulces (response)

Plan de pruebas

El plan de pruebas aquí presentado se realizó con el propósito de identificar posibles fallas y errores de nuestro servicio web “Cine”, también uno de los propósitos es verificar el correcto funcionamiento de cada uno de los microservicios y las funcionalidades que lo componen a cada uno de ellos. De manera detallada podemos resaltar que la prueba se realizó a todos los microservicios incluidos en nuestro servicio web “Cine”, por cada funcionalidad de cada microservicio se realizó una prueba y en su gran mayoría los resultados obtenidos fueron exitosos.

Microservicio Tickets y Funciones

Requerimientos de entornos – Hardware

Hardware: Una computadora con acceso a un navegador

Red: Conexión a internet wifi

Requerimientos de entornos – Software

Software: software de para consumir un recurso de un servicio web desde un cliente de manera grafica.

Planificación y Organización

Procedimientos para las pruebas

La prueba se realizó siguiendo una serie de pasos dependiendo de la funcionalidad del microservicio.

Fecha	20 de mayo del 2022
Título	Crear función
Descripción	El administrador del cine crea las funciones de las películas
Condiciones de Entrada	<ul style="list-style-type: none">El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsd!Debe de usarse el software y colocamos las peticiones en Json:'{"nombrePelicula":"Doctor Strange","hora":"10:30","fecha":"11/02/2022","precio":50,"Clasificación":"B","asientosDisponibles":5}'
Resultado Esperado	El servicio debe guardar los datos introducidos en la base de datos y despliega todas las funciones agregadas
Resultado Obtenido	El servicio guarda los datos en la base de datos Clever cloud, muestra esos datos en la pantalla en la parte de response y muestra un estatus de la operación.

Fecha	20 de mayo del 2022
Título	Comprar boletos
Descripción	El usuario podrá ver la lista de empleados junto con sus datos, que se encuentran guardados en la BD
Condiciones de Entrada	<ul style="list-style-type: none"> El usuario debe entrar a enlace de heroku: https://microservicio-funcion.herokuapp.com/comprarBoletos?idFuncion=1 Añadir en JSON los atributos En él envío de parámetros sería un JSON donde se indica la cantidad de boletos para comprar: {"cantidad":3}
Resultado Esperado	Nos muestra que compramos con éxito los boletos
Resultado Obtenido	El servicio nos muestra la lista de empleados con sus respectivos datos y el estatus de la operación, todo en la pantalla response.

Fecha	20 de mayo del 2022
Título	Ver cartelera
Descripción	EL usuario podrá modificar los datos de cualquier empleado que se encuentre registrado en la base de datos Clever Cloud
Condiciones de Entrada	<ul style="list-style-type: none"> El usuario debe entrar al enlace de heroku: https://microservicio-funcion.herokuapp.com/verCartelera Se mostrará la cartelera disponible de acuerdo con el horario
Resultado Esperado	El servicio nos debe mostrar la cartelera disponible
Resultado Obtenido	El servicio nos muestra la lista de la cartelera con las funciones disponibles

Fecha	10 de mayo del 2022
Título	Cambiar función
Descripción	El administrador del cine cambia algún atributo de la función que ya existe
Condiciones de Entrada	<ul style="list-style-type: none"> El usuario debe entrar a link de heroku: https://microservicio-funcion.herokuapp.com/cambiarFuncion El id de la compra de los boletos y la nueva función a la que se desea cambiar. localhost:8080/cambiarFuncion?idCompra=2&idNuevaFuncion=7
Resultado Esperado	El servicio nos debe mostrar la lista de empleados guardada en la base de datos menos el empleado que eliminamos
Resultado Obtenido	El servicio nos muestra la lista de empleados guardada en la base de datos Clever Cloud menos el empleado que eliminamos. Todo esto se nos muestra en la pantalla response.

Microservicio Empleados

Recursos

Requerimientos de entornos – Hardware

Hardware: Una computadora con acceso a un navegador

Red: Conexión a internet wifi

Requerimientos de entornos – Software

Software: extensión de wizzler en nuestro navegador (para el caso de SOAP)

Planificación y Organización

Procedimientos para las pruebas

La prueba se realizó siguiendo una serie de pasos dependiendo de la funcionalidad del microservicio.

Fecha	26 de mayo del 2022
Título	Registrar Empleado
Descripción	El usuario registra un empleado llenando los campos correspondientes
Condiciones de Entrada	<ul style="list-style-type: none">• El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl• Debe usar la extensión wizzler y seleccionar la funcionalidad “RegistrarEmpleado”• Llenar los campos correspondientes en la parte de request• Y darle clic en el botón “GO”
Resultado Esperado	El servicio debe guardar los datos introducidos en la base de datos Clever Cloud y mostrarlos en pantalla en la parte de response.
Resultado Obtenido	El servicio guarda los datos en la base de datos Clever cloud, muestra esos datos en la pantalla en la parte de response y muestra un estatus de la operación.

Fecha	26 de mayo del 2022
Título	Empleados
Descripción	El usuario podrá ver la lista de empleados junto con sus datos, que se encuentran guardados en la BD
Condiciones de Entrada	<ul style="list-style-type: none">• El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl• Debe usar la extensión wizzler y seleccionar la funcionalidad “Empleados”• No se debe llenar ningún campo, solo darle clic al botón “GO” que esta en la pantalla request

Resultado Esperado	El servicio nos debe mostrar la lista de empleados con sus respectivos datos en la pantalla response
Resultado Obtenido	El servicio nos muestra la lista de empleados con sus respectivos datos y el estatus de la operación, todo en la pantalla response.

Fecha	26 de mayo del 2022
Título	Modificar Empleado
Descripción	EL usuario podrá modificar los datos de cualquier empleado que se encuentre registrado en la base de datos Clever Cloud
Condiciones de Entrada	<ul style="list-style-type: none"> • El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl • Debe usar la extensión widdler y seleccionar la funcionalidad "ModificarEmpleado" • Se deben llenar todos los campos con los datos nuevos que se desean reemplazar, sin dejar atrás que lo mas importante es ingresar el id del empleado al que le vamos a modificar sus datos. Todo esto se hace en la pantalla de request • Y por ultimo le damos clic al boton "GO"
Resultado Esperado	El servicio nos debe mostrar los campos del empleado que modificamos ya con los datos nuevos y en la BD actualizados. Todo esto en la pantalla response.
Resultado Obtenido	El servicio nos muestra los campos del empleado ya modificados, con los datos nuevos y en la base de datos actualizados. Todo esto se muestra en la pantalla response.

Fecha	26 de mayo del 2022
Título	Eliminar Empleado
Descripción	El usuario podrá eliminar cualquier empleado que se encuentre en la BD por medio de su id
Condiciones de Entrada	<ul style="list-style-type: none"> • El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl • Debe usar la extensión widdler y seleccionar la funcionalidad "EliminarEmpleado" • Se debe introducir el id del empleado que deseamos eliminar, todo esto en la pantalla request • Y damos clic al boton "GO"
Resultado Esperado	El servicio nos debe mostrar la lista de empleados guardada en la base de datos menos el empleado que eliminamos
Resultado Obtenido	El servicio nos muestra la lista de empleados guardada en la base de datos Clever Cloud menos el empleado que eliminamos. Todo esto se nos muestra en la pantalla response.

Microservicio Dulceria

Recursos

Requerimientos de entornos – Hardware

Hardware: Una computadora con acceso a un navegador

Red: Conexión a internet wifi

Requerimientos de entornos – Software

Software: extensión de wizzler en nuestro navegador (para el caso de SOAP)

Planificación y Organización

Procedimientos para las pruebas

La prueba se realizó siguiendo una serie de pasos dependiendo de la funcionalidad del microservicio.

Fecha	3 de junio del 2022
Título	Registrar Empleado
Descripción	El usuario registra un empleado llenando los campos correspondientes
Condiciones de Entrada	<ul style="list-style-type: none">• El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl• Debe usar la extensión wizzler y seleccionar la funcionalidad “RegistrarEmpleado”• Llenar los campos correspondientes en la parte de request• Y darle clic en el botón “GO”
Resultado Esperado	El servicio debe guardar los datos introducidos en la base de datos Clever Cloud y mostrarlos en pantalla en la parte de response.
Resultado Obtenido	El servicio guarda los datos en la base de datos Clever cloud, muestra esos datos en la pantalla en la parte de response y muestra un estatus de la operación.

Fecha	3 de junio del 2022
Título	Empleados
Descripción	El usuario podrá ver la lista de empleados junto con sus datos, que se encuentran guardados en la BD
Condiciones de Entrada	<ul style="list-style-type: none">• El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsdl• Debe usar la extensión wizzler y seleccionar la funcionalidad “Empleados”• No se debe llenar ningún campo, solo darle clic al botón “GO” que esta en la pantalla request
Resultado Esperado	El servicio nos debe mostrar la lista de empleados con sus respectivos datos en la pantalla response

Resultado Obtenido	El servicio nos muestra la lista de empleados con sus respectivos datos y el estatus de la operación, todo en la pantalla response.
---------------------------	---

Fecha	3 de junio del 2022
Título	Modificar Empleado
Descripción	EL usuario podrá modificar los datos de cualquier empleado que se encuentre registrado en la base de datos Clever Cloud
Condiciones de Entrada	<ul style="list-style-type: none"> • El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsd! • Debe usar la extensión widdler y seleccionar la funcionalidad "ModificarEmpleado" • Se deben llenar todos los campos con los datos nuevos que se desean reemplazar, sin dejar atrás que lo mas importante es ingresar el id del empleado al que le vamos a modificar sus datos. Todo esto se hace en la pantalla de request • Y por ultimo le damos clic al boton "GO"
Resultado Esperado	El servicio nos debe mostrar los campos del empleado que modificamos ya con los datos nuevos y en la BD actualizados. Todo esto en la pantalla response.
Resultado Obtenido	El servicio nos muestra los campos del empleado ya modificados, con los datos nuevos y en la base de datos actualizados. Todo esto se muestra en la pantalla response.

Fecha	3 de junio del 2022
Título	Eliminar Empleado
Descripción	El usuario podrá eliminar cualquier empleado que se encuentre en la BD por medio de su id
Condiciones de Entrada	<ul style="list-style-type: none"> • El usuario debe entrar a link de heroku: https://cine-empleados.herokuapp.com/ws/Empleados.wsd/ • Debe usar la extensión widdler y seleccionar la funcionalidad "EliminarEmpleado" • Se debe introducir el id del empleado que deseamos eliminar, todo esto en la pantalla request • Y damos clic al boton "GO"
Resultado Esperado	El servicio nos debe mostrar la lista de empleados guardada en la base de datos menos el empleado que eliminamos
Resultado Obtenido	El servicio nos muestra la lista de empleados guardada en la base de datos Clever Cloud menos el empleado que eliminamos. Todo esto se nos muestra en la pantalla response.

Dockerfile del Microservicio Tickets y Funciones

Código del archivo Dockerfile

```

From rrojano/jdk8
workdir /app
#expose 8080
cmd ["/app/ejecutar.sh"]
add app/Funciones-0.0.1-SNAPSHOT.jar /app/Funciones-0.0.1-SNAPSHOT.jar
add ejecutar.sh /app/ejecutar.sh
run chmod 755 /app/ejecutar.sh

```

Ejecutar del Microservicio Tickets y Funciones

Código del archivo ejecutar.sh

```

#!/bin/sh
/usr/bin/java -jar -Dserver.port=$PORT Funciones-0.0.1-SNAPSHOT.jar

```

Dockerfile del Microservicio Empleados

Código del archivo Dockerfile

```
From rrojano/jdk8
workdir /app
#expose 8080
cmd ["/app/ejecutar.sh"]
add app/EmpleadosBd-0.0.1-SNAPSHOT.jar /app/EmpleadosBd-0.0.1-
SNAPSHOT.jar
add ejecutar.sh /app/ejecutar.sh
run chmod 755 /app/ejecutar.sh
```

Ejecutar del Microservicio Empleados

Código del archivo ejecutar.sh

```
#!/bin/sh
/usr/bin/java -jar -Dserver.port=$PORT EmpleadosBd-0.0.1-SNAPSHOT.jar
```

Dockerfile del Microservicio dulcería

Código del archivo Dockerfile

```
From rrojano/jdk8
workdir /app
#expose 8080
cmd ["/app/ejecutar.sh"]
add app/Dulceria-0.0.1-SNAPSHOT.jar /app/Dulceria-0.0.1-SNAPSHOT.jar
add ejecutar.sh /app/ejecutar.sh
run chmod 755 /app/ejecutar.sh
```

Ejecutar del Microservicio dulcería

Código del archivo ejecutar.sh

```
/usr/bin/java -jar -Dserver.port=$PORT Dulceria-0.0.1-SNAPSHOT.jar
```

Proxy inverso

```
server {  
listen $PORT;  
location /boletos/ {  
proxy_pass https://microservicio-funcion.herokuapp.com/;  
}  
location /empleados/ {  
proxy_pass https://cine-empleados.herokuapp.com/ws/Empleados.wsdl;  
}  
location /dulceria/ {  
proxy_pass https://dulceriati.herokuapp.com/ws/dulces.wsdl;  
}  
}
```

Dockerfile

```
FROM nginx add proxy/proxy.conf /etc/nginx/conf.d/default.conf add proxy/index.html  
/etc/nginx/html/index.html add proxy/Documentacion.html  
/etc/nginx/html/Documentacion.html add proxy/LogoUV.png  
/etc/nginx/html/LogoUV.png CMD sed -i -e 's/$PORT/"$PORT"/g'  
/etc/nginx/conf.d/default.conf && nginx -g 'daemon off;'
```

Documentación .html

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Documetación</title>  
</head>  
<body>  
  <h1>SERVICIO DE CINE</h1>  
  <br>  
  <h2>Problematica</h2>  
  <p>Una de las problemáticas que se encontraron fue en la parte administrativa de un cine por  
ende proponemos que nuestro servicio  
sirva para tras aspectos que ayuden a solucionar la organización, por ejemplo en la parte de la  
dulcería en donde le permitirá al
```

usuario comprar dulces que estén disponibles en el inventario y solo llevar un ticket al Cine de esta manera poder evitar largas filas y ahorrar tiempo.</p>

<p>Por otra parte, administrativamente el poder comprar boletos y ver la cartelera disponible en el sistema ahorrará bastante tiempo y permitirá que este controlado el flujo de personas en el Cine. Por último, el manejo de empleados para una adecuada organización interna del personal. </p>

[17:14]

<h2>Solucion</h2>

<p>Para la solución de este problema se tiene planeado realizar un servicio de API el cual esta dividido en microservicios siendo estos:</p>

<h3>

Microservicio de Tickets o Boletos:

</h3>

<p>Este microservicio permitirá que el que se revise la cartelera disponible, ver las funciones que se haran por cada pelicula, para ver la cartelera se tendran atributos como (IdFuncion, Nombre de pelicula, Hora, Fecha, Precio, Clasificación y Asientos disponibles), tambien se podra comprar boletos y cambiar u modificar la función.</p>

<h3>

Microservicio de Empleados:

</h3>

<p>En ese microservicio se empleará un CRUD con las siguientes opciones: Dar de Alta Empleados, Ver Empleados, Eliminar Empleados y Modificar los Empleados. Cada apartado tendrá sus distintos atributos </p>

<h3>

Microservicio de Dulceria:

</h3>

<p> Este microservicio se incorporar la opcion de comprar dulces, ver el menu, el inventario y modificar el mismo</p>

</body>

</html>

Estilos

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

.navbar {
  overflow: hidden;
  background-color: #090793;
}

.navbar a {
  float: left;
  font-size: 16px;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-family: 'calibri';
}

.subnav {
  float: left;
  overflow: hidden;
}

.subnav .subnavbtn {
  font-size: 16px;
  border: none;
  outline: none;
  color: white;
  padding: 14px 16px;
  background-color: inherit;
  font-family: 'calibri';
  margin: 0 auto;
}
```

```
.navbar a:hover, .subnav:hover .subnavbtn {  
  background-color: #04D831;  
}
```

```
.subnav-content {  
  display: none;  
  position: absolute;  
  left: 0;  
  background-color: #04D831;  
  width: 100%;  
  z-index: 1;  
}
```

```
.subnav-content a {  
  float: left;  
  color: white;  
  text-decoration: none;  
}
```

```
.subnav-content a:hover {  
  background-color: #eee;  
  color: black;  
}
```

```
.subnav:hover .subnav-content {  
  display: block;  
}
```

Canva

```
.sidenav {  
  height: 100%;  
  width: 0;  
  position: fixed;  
  z-index: 1;  
  top: 0;  
  left: 0;  
  background-color: black;  
  overflow-x: hidden;  
  transition: 0.5s;  
  padding-top: 60px;  
}
```

```
.sidenav a {
```

```

padding: 8px 8px 8px 32px;
text-decoration: none;
font-size: 25px;
color: white;
display: block;
transition: 0.3s;
}

.sidenav a:hover {
  color: black;
}

.sidenav .closebtn {
  position: absolute;
  top: 0;
  right: 25px;
  font-size: 36px;
  margin-left: 50px;
}

#main {
  transition: margin-left .5s;
  padding: 16px;
}

@media screen and (max-height: 450px) {
  .sidenav {padding-top: 15px;}
  .sidenav a {font-size: 18px;}
}
</style>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!--script src="acciones.js"></script-->
  <title>Pantalla Principal</title>
</head>
<body>

```

Acciones JS

```
<script>
//Acciones relacionado con el funcionamiento del patron de diseño Off Canva

function openNav() {
  document.getElementById("mySidenav").style.width = "250px";
  document.getElementById("main").style.marginLeft = "250px";
  document.body.style.backgroundColor = "rgba(0,0,0,0.4)";
}

function closeNav() {
  document.getElementById("mySidenav").style.width = "0";
  document.getElementById("main").style.marginLeft= "0";
  document.body.style.backgroundColor = "white";
}
</script>
<!--Este menu sale a pantalla cuando das clic en un boton-->
<!--Todo esto es por el patron de diseño Off Canvas-->

<div id="mySidenav" class="sidenav">
  <a href="javascript:void(0)" class="closebtn" onclick="closeNav()">&times;</a>
<!--&times; Muestra una X-->

  <!--img src="Perfil.jpeg" alt="Bryan" width="200" height="300"-->
  <br><br>
  <a href="#">Facultad de Estadística e Informática</a>
  <a href="#">Av. Xalapa s/n, Obrero Campesina, 91020 Xalapa-Enríquez,
Ver.</a>

</div>
<div id="main" class="navbar">
  <div class="perfil">
    <span style="cursor:pointer;" onclick="openNav()"></span>
  </div>
  <a href="Documentacion.html">Breve Documentacion</a>
  <div class="subnav">
    <button class="subnavbtn">Microservicios<i class="fa fa-caret-
down"></i></button>
    <div class="subnav-content">
      <a href="https://microservicio-funcion.herokuapp.com/%22%3ETickets y
Boletos</a>
      <a href="https://cine-
empleados.herokuapp.com/ws/Empleados.wsd/%22%3EEmpleados</a>
      <a href="https://dulceriati.herokuapp.com/ws/dulces.wsd/%22%3EDulceria</a>
```

```

    </div>
</div>
<div class="subnav">
  <button class="subnavbtn">Repositorios<i class="fa fa-caret-
down"></i></button>
  <div class="subnav-content">
    <a href="https://github.com/juxo321/Cine%22%3ERepositorio Tickets y
Boletos</a>
    <a
href="https://github.com/bryanUCP/Cine_Empleados.git%22%3ERepositorio
Empleados</a>
    <a href="https://github.com/Adbel79/Dulceria.git ">Repositorio Dulceria</a>
  </div>
</div>
</div>

<br>
<br>
<h1>Proyecto Tecnologías para la Integración de Soluciones - Equipo 2</h1>
<br>
<h2>Integrantes:</h2>
<h3>
<br>
<li>
  Bryan Usías Catalino Piquet
</li>
<h3>
<br>
<li>
  Ariana Tejeda Santiz
</li>
<h3>
<br>
<li>
  Justin Martinez Hernández
</li>
<h3>
<br>
<li>
  Adbel Rene Garcia Hernández
</li>

</body>
</html>

```

Proyecto publicado en github

Microservicio Tickets y Boletos

[juxo321/Cine \(github.com\)](https://github.com/juxo321/Cine)

Microservicio Empleados

https://github.com/bryanUCP/Cine_Empleados.git

Microservicio de dulces

<https://github.com/Adbel79/Dulceria.git>

Proyecto publicado en Heroku

Microservicio Tickets y Boletos

<https://microservicio-funcion.herokuapp.com/>

Microservicio Empleados

<https://cine-empleados.herokuapp.com/ws/Empleados.wsdl>

Microservicio Dulces

<https://dulceriati.herokuapp.com/ws/dulces.wsdl>

Proxy

<https://proxyinversotc.herokuapp.com/>