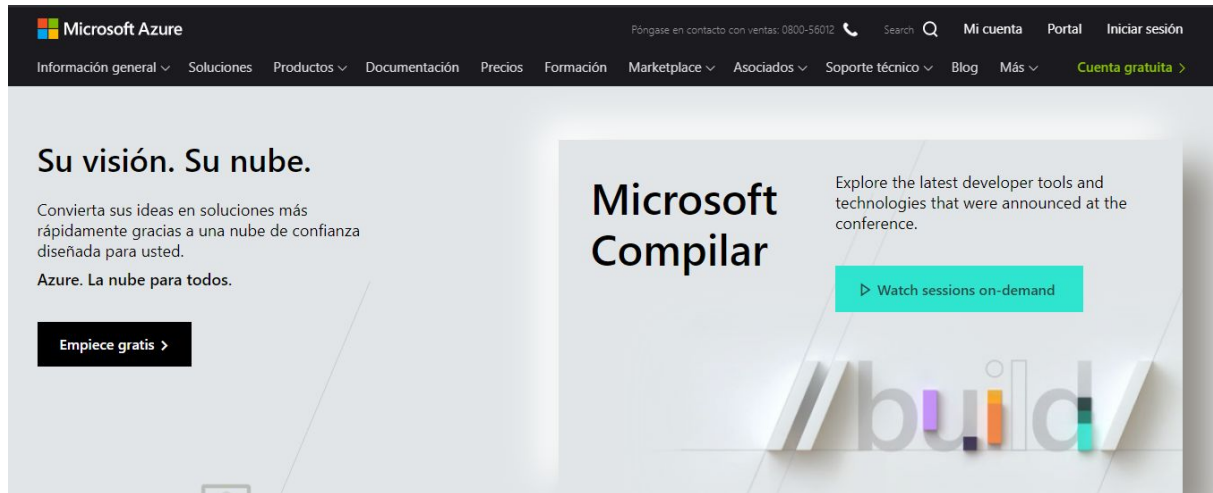


# Face APi + Java EE

Nos dirigimos a la siguiente url

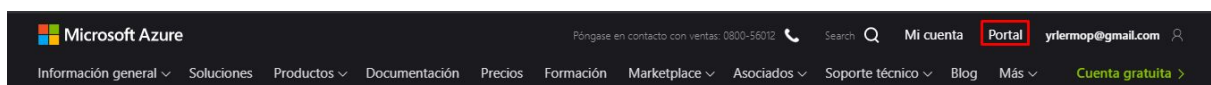
<https://azure.microsoft.com/es-es/>



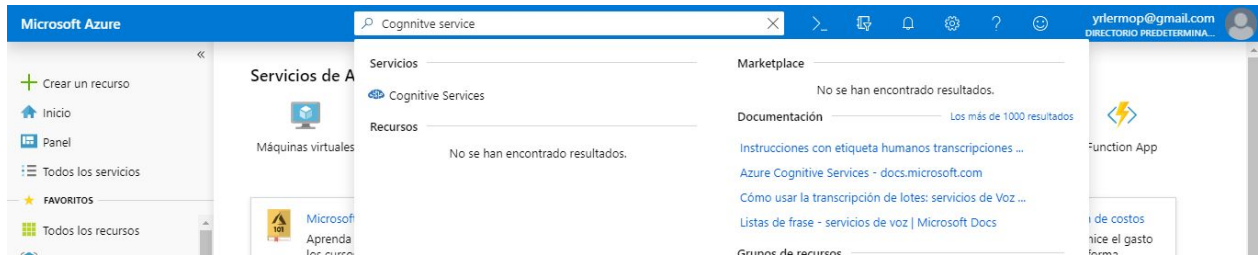
Iniciamos sesión



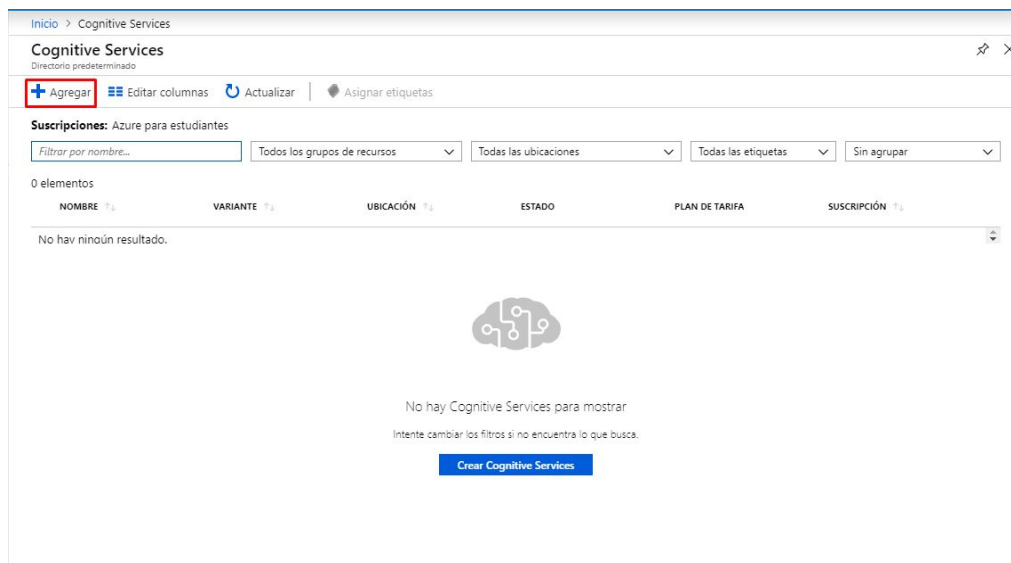
Ingresamos a nuestro portal



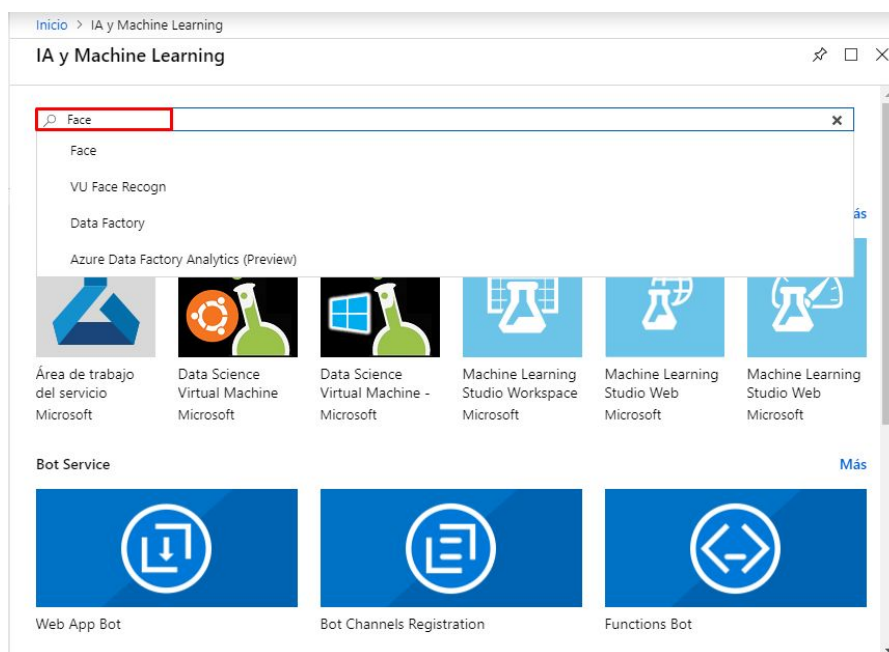
## Buscamos Cognitive Service



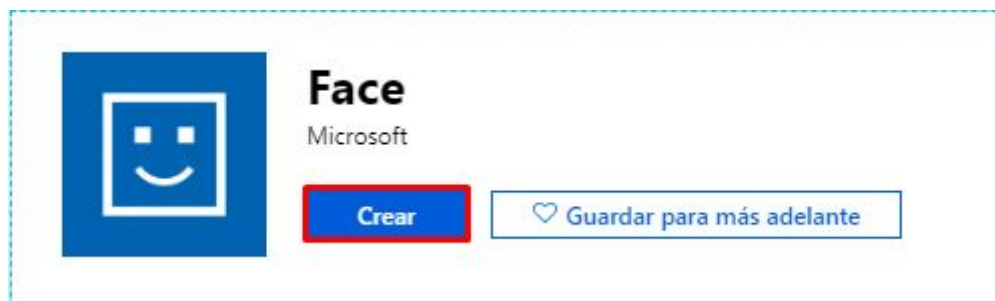
## Agregamos un nuevo servicio Cognitivo



## Buscamos Face



Creamos el servicio cognitivo



Nombre del servicio

\* Nombre  
Demo

Si tenemos la cuenta de estudiante nos aparece predeterminado

\* Suscripción  
Azure para estudiantes

Agregamos la ubicación del servicio

\* Ubicación  
(EE. UU.) Centro-sur de EE. UU.

El plan de tarifa, seleccionamos F0

\* Plan de tarifa ([Ver todos los detalles de los precios](#))  
S0 (10 Llamadas por segundo)

Creamos un nuevo grupo de recurso o si ya tenemos un grupo de recurso

\* Grupo de recursos  
Seleccionar existentes...  
Crear nuevo

Le damos crear

Crear

Después de haber creado el servicio nos dirigimos a recursos

✓ Se completó la implementación

[Ir al recurso](#)



Nombre de implementación: Microsoft.CognitiveServicesFace  
Suscripción: [Azure para estudiantes](#)  
Grupo de recursos: [Prueba](#)

DETALLES DE IMPLEMENTACIÓN [\(Descargar\)](#)

Hora de inicio: 24/5/2019 8:53:48

Duración: 12 segundos

Id. de correlación: 929c0af0-689d-440d-ac60-e7ccda4a2503

RECURSO	TIPO	ESTADO	DETALLES DE LA OPE...
✓ Demo	Microsoft.Cognitiv...	OK	<a href="#">Detalles de la operac</a>

Click en información general

**Demo - Inicio rápido**  
Cognitive Services

Buscar (Ctrl+F)

**Información general**

Registro de actividad  
Control de acceso (IAM)  
Etiquetas  
Diagnosticar y solucionar pr...

ADMINISTRACIÓN DE RECURSOS

Claves  
Inicio rápido  
Plan de tarifa  
Facturación por suscripción  
Propiedades  
Bloqueos  
Exportar plantilla

Supervisión  
Alertas

Explore la guía de inicio rápido para empezar a usar Face.  
Descubre el nuevo soporte técnico para contenedores de Docker en Azure Cognitive Services (VERSIÓN PRELIMINAR)

**1** Obtener las claves  
Cada llamada API y cada activación de contenedor de Docker de Face requiere una clave de suscripción. Para la API web, esta clave se debe pasar a través de un parámetro de cadena de consulta o se debe especificar en el encabezado de la solicitud. Para el contenedor de Docker, la clave se debe pasar a través del comando de Docker.  
[Claves](#)

**2a** Ejecutar el contenedor de Docker (VERSIÓN PRELIMINAR)  
Face también está disponible como contenedor de Docker que puede extraer y usar directamente como parte de la aplicación. El contenedor de Face permite obtener la misma funcionalidad enriquecida que el servicio Face, pero empaquetado en un contenedor de Docker.  
[Compatibilidad con contenedores en Azure Cognitive Services](#)  
[Contenedor de Face](#)

**2b** O realizar una llamada API a este extremo: <https://southcentralus.api.cognitive.microsoft.com/face/v1.0>  
Obtenga información detallada sobre cada propiedad y método de la API. Pruebe las claves con la consola de pruebas integrada sin

Buscamos el token de conexión

**Demo**  
Cognitive Services

Buscar (Ctrl+F)

Configuración no disponible Eliminar

Grupo de recursos (cambiar)  
[Prueba](#)

Tipo de API  
Face

Estado  
Activo

Plan de tarifa  
Estándar

Ubicación  
Centro-Sur de EE. UU.

Suscripción (cambiar)  
[Azure para estudiantes](#)

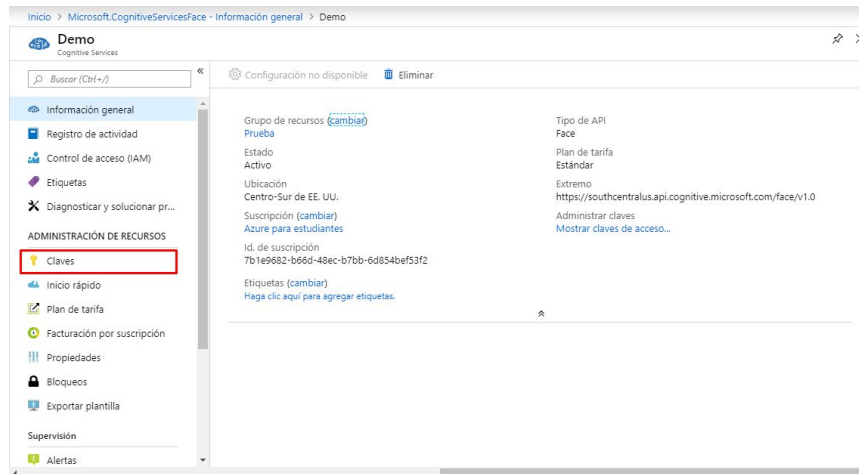
Id. de suscripción  
7b1e9682-b66d-48ec-b7bb-6d054bef53f2

Etiquetas (cambiar)  
[Haga clic aquí para agregar etiquetas.](#)

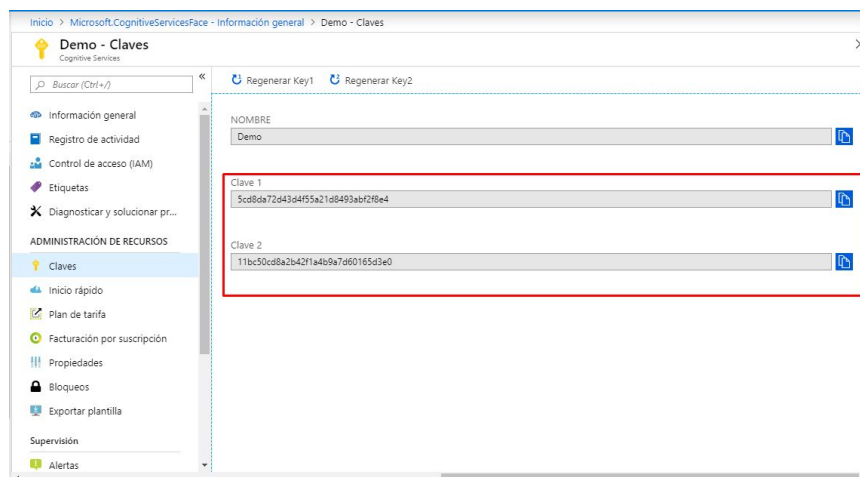
Extremo  
<https://southcentralus.api.cognitive.microsoft.com/face/v1.0>

Administrar claves  
[Mostrar claves de acceso...](#)

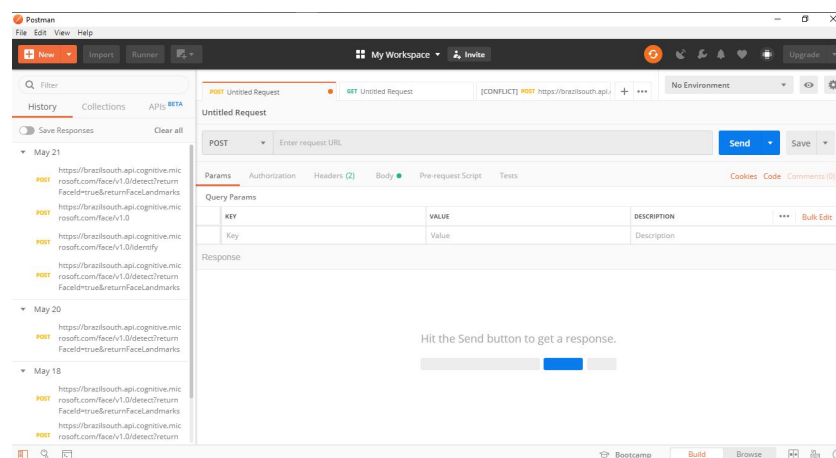
Ahora nos dirigimos a **Claves**



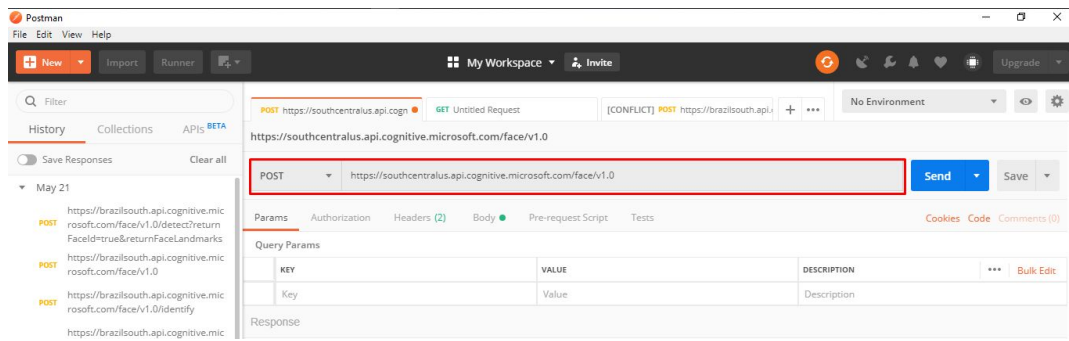
Podemos escoger cualquiera de estas dos claves para la conexión



Para probar la api de conexión abriremos **POSTMAN** (Opcional)



Copiamos el token de conexión y pegamos



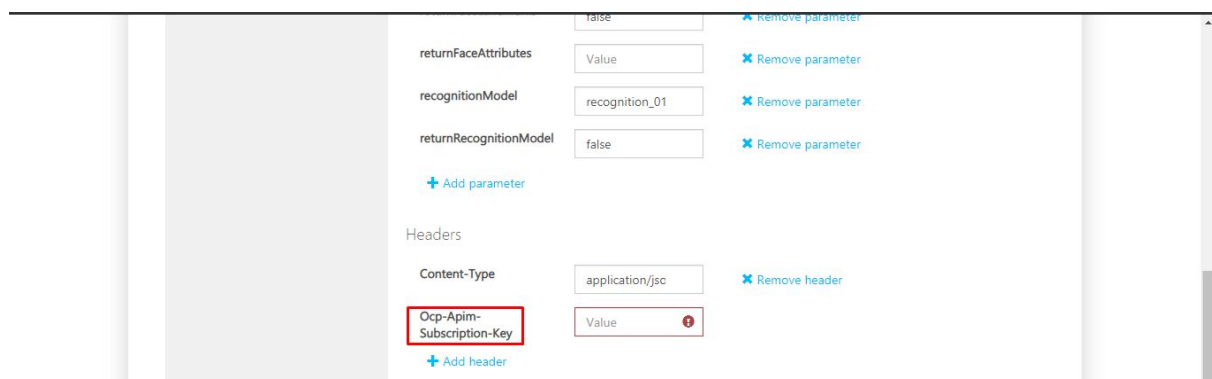
Nos dirigimos a la siguiente url

<https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395236>

Ahí buscamos según en donde hayamos creado el servicio



Copiamos el siguiente texto



Agregamos los parametros segun la pagina

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Cookies

Code

Comments

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	returnFaceId	true			
<input checked="" type="checkbox"/>	returnFaceLandmarks	false			
<input checked="" type="checkbox"/>	recognitionModel	recognition_01			
<input checked="" type="checkbox"/>	returnRecognitionModel	false			
<input checked="" type="checkbox"/>	returnFaceAttributes	age,gender,smile,facialHair,glasses,emotion,hair,make...			
	Key	Value	Description		

Agregamos la Clave

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Cookies

Code

Comments

▼ Headers (2)

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets ▼
<input checked="" type="checkbox"/>	Content-Type	application/json				
<input checked="" type="checkbox"/>	Ocp-Apim-Subscription-Key	83be250b81834f99bf636ab44a73b955				
	Key	Value	Description			

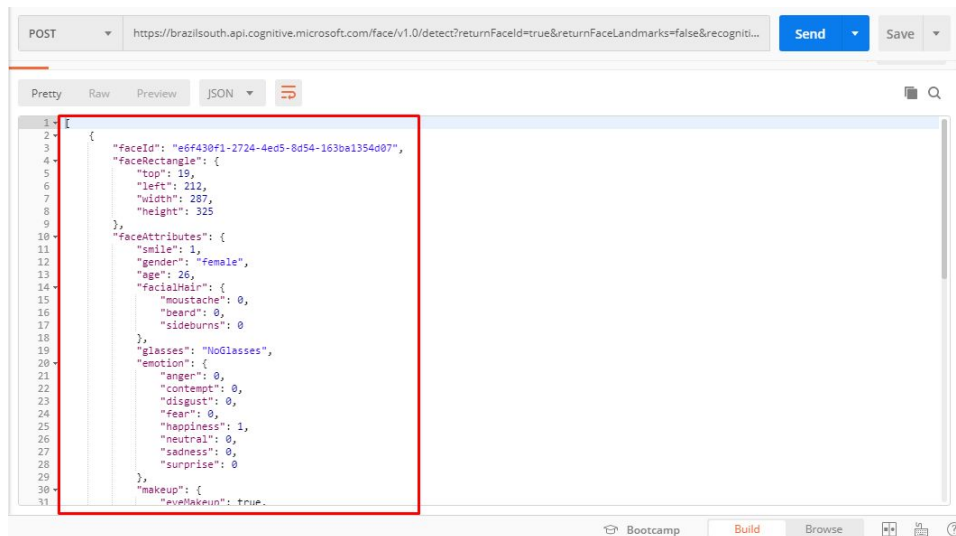
Agregamos una url de prueba

Params	Authorization	Headers (10)	Body	Pre-request Script	Tests	Cookies	Code	Comments
● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON (application/json) Beautify								
1	{							
2	"url": "http://cosmeticdentistofmichigan.com/wp-content/uploads/2011/05/shutterstock_59848516.jpg"							
3	}							

y enviamos para probar el token

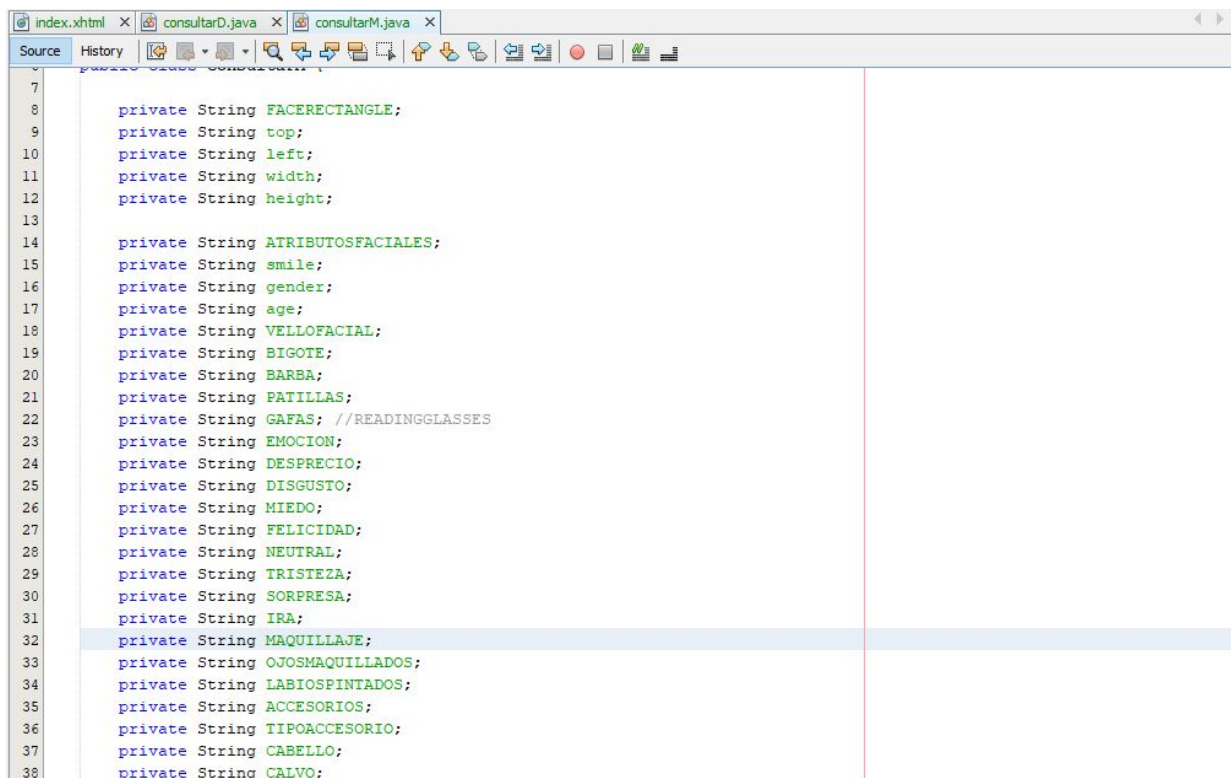
POST	https://brazilsouth.api.cognitive.microsoft.com/face/v1.0/detect?returnFaceId=true&returnFaceLandmarks=false&recogniti...	Send	Save
------	---------------------------------------------------------------------------------------------------------------------------	------	------

Ahí podemos ver los datos que nos envía el token



## Integración a Java EE

Declaramos las variables en el modelo





Creamos un método en el dao para hacer la consulta con la API

DAO:

Agregamos la URL, Clave y tipo de consulta

```
public class consultarD {  
  
    public consultarM consultar(String url) throws IOException {  
        consultarM variablesModel = new consultarM();  
        HttpClient httpClient = new DefaultHttpClient();  
        try {  
            StringEntity body = new StringEntity("{\"Url\": \"" + url + "\"}");  
            JsonParser converter = new JsonParser();  
            String emocion[] = "anger,contempt,disgust,fear,happiness,neutral,sadness,surprise".split(",");  
            HttpPost request = new HttpPost("https://brazilsouth.api.cognitive.microsoft.com/face/v1.0/detect?return");  
            request.addHeader("Ocp-Apim-Subscription-Key", "83be250b81834f99bf636ab44a73b955");  
            request.addHeader("Content-Type", "application/json");  

```

Traemos los atributos de la API

```
            request.setEntity(body);  
            HttpResponse response = httpClient.execute(request);  
            HttpEntity entity = response.getEntity();  
            JSONArray array = converter.parse(EntityUtils.toString(entity)).getAsJSONArray();  
            JSONObject object = array.get(0).getAsJsonObject();  
            JSONObject attributes = object.getAsJsonObject("faceAttributes");  
            JSONObject emotion = attributes.getAsJsonObject("emotion");  
            JSONObject velloCara = attributes.getAsJsonObject("facialHair");  
            JSONObject maquillaje = attributes.getAsJsonObject("makeup");  
            JSONObject pelo = attributes.getAsJsonObject("hair");  
            JSONArray colorPelo = pelo.getAsJSONArray("hairColor");  
            String lentes = attributes.get("glasses").getString();  
            String edad = attributes.get("age").getString();  
            String genero = attributes.get("gender").getString();  
            double bigote = velloCara.get("moustache").getAsDouble();  
            double barba = velloCara.get("beard").getAsDouble();  
            double patillas = velloCara.get("sideburns").getAsDouble();  
            String ojosPintados = maquillaje.get("eyeMakeup").getString();  
            String labiosPintados = maquillaje.get("lipMakeup").getString();  
            double calvo = pelo.get("bald").getAsDouble();  
            String invisible = pelo.get("invisible").getString();  
            String nameEmotion = null;
```

Validamos y traducimos los datos que traemos de la API

```
//  
            String colorPerlo = colores.get("color").getString();  
            System.out.println(json1);  
            variablesModel.setEMOCION(nameEmotion);  
            variablesModel.setBIGOTE(bigote > 0.5 ? "Sí" : "NO");  
            variablesModel.setBARBA(barba > 0.5 ? "Sí" : "NO");  
            variablesModel.setPATILLAS(patillas > 0.5 ? "Sí" : "NO");  
            variablesModel.setGAFAS(lentes.equals("NoGlasses") ? "SIN LENTES" : "CON LENTES");  
            variablesModel.setAge(edad);  
            variablesModel.setGender(genero.equals("female") ? "MUJER" : "HOMBRE");  
            variablesModel.setOJOSMAQUILLADOS(ojosPintados.equals("false") ? "NO" : "Sí");  
            variablesModel.setLABIOSPINTADOS(labiosPintados.equals("false") ? "NO" : "Sí");  
            variablesModel.setCALVO(calvo > 0.5 ? "Sí" : "NO");  
            variablesModel.setINVENSIBLE(invisible.equals("false") ? "NO" : "Sí");  
            variablesModel.setCOLORPELO(colorPerlo);
```

## Controlador:

Creamos dos variables “URL”, “Resultado” y instanciamos el modelo

```
private String urlImage;  
private String resultado;  
private consultarD dao = new consultarD();  
consultarM modelo = new consultarM();
```

Creamos un método para hacer la consulta enviando la URL

```
public void consultar() throws Exception {  
    try {  
        modelo = dao.consultar(urlImage);  
        modelo.setLINK(urlImage);  
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "CONSULTA"  
    } catch (IOException ex) {  
        throw ex;  
    }  
}
```

## Vista:

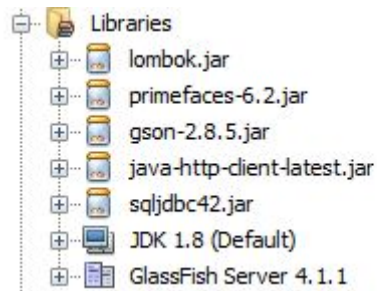
Creamos un inputtext para ingresar la URL y un botón para hacer la consulta.

```
</p:growl>  
<p:panelGrid columns="2" id="dlg">  
    <p:inputText value="#{consultarC.urlImage}" placeholder="Url de la Imagen"/>  
    <p:commandButton id="btnConsultar" value="Consultar" actionListener="#{consultarC.consultar()}" update="form"/>  
    <p:commandButton id="btnLimpiar" value="Limpiar" actionListener="#{consultarC.clear()}" update="form"/>  
    <br/>  
    <span style="font-size: 50px;">#{consultarC.resultado}</span>  
    <p:blockUI block="form" trigger="btnConsultar" />  
      
</p:panelGrid columns="2" id="pnlg">
```

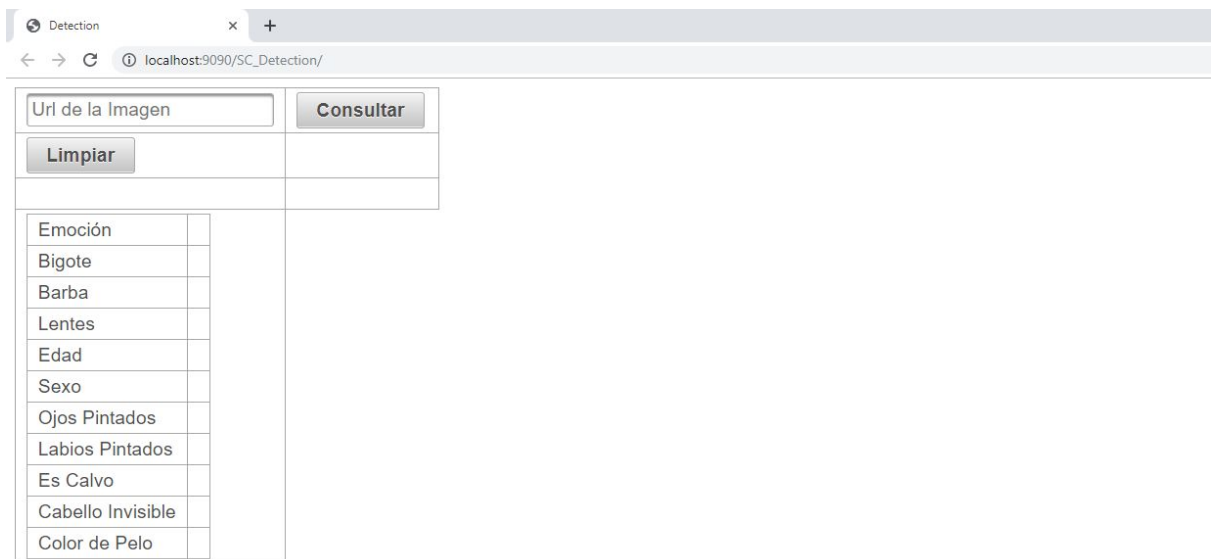
Creamos outputlabel para mostrar los datos

```
<p:panelGrid columns="2" id="pnlg">
    <p:outputLabel value="Emoción"/>
    <p:outputLabel value="#{consultarC.modelo.EMOCION}"/>
    <p:outputLabel value="Bigote"/>
    <p:outputLabel value="#{consultarC.modelo.BIGOTE}"/>
    <p:outputLabel value="Barba"/>
    <p:outputLabel value="#{consultarC.modelo.BARBA}"/>
    <p:outputLabel value="Lentes"/>
    <p:outputLabel value="#{consultarC.modelo.GAFAS}"/>
    <p:outputLabel value="Edad"/>
    <p:outputLabel value="#{consultarC.modelo.age}"/>
    <p:outputLabel value="Sexo"/>
    <p:outputLabel value="#{consultarC.modelo.gender}"/>
    <p:outputLabel value="Ojos Pintados"/>
    <p:outputLabel value="#{consultarC.modelo.OJOSMAQUILLADOS}"/>
    <p:outputLabel value="Labios Pintados"/>
    <p:outputLabel value="#{consultarC.modelo.LABIOSPINTADOS}"/>
    <p:outputLabel value="Es Calvo"/>
    <p:outputLabel value="#{consultarC.modelo.CALVO}"/>
    <p:outputLabel value="Cabello Invisible"/>
    <p:outputLabel value="#{consultarC.modelo.INVENSIBLE}"/>
    <p:outputLabel value="Color de Pelo"/>
    <p:outputLabel value="#{consultarC.modelo.COLORPELO}"/>
</p:panelGrid>
```

Librerías necesarias



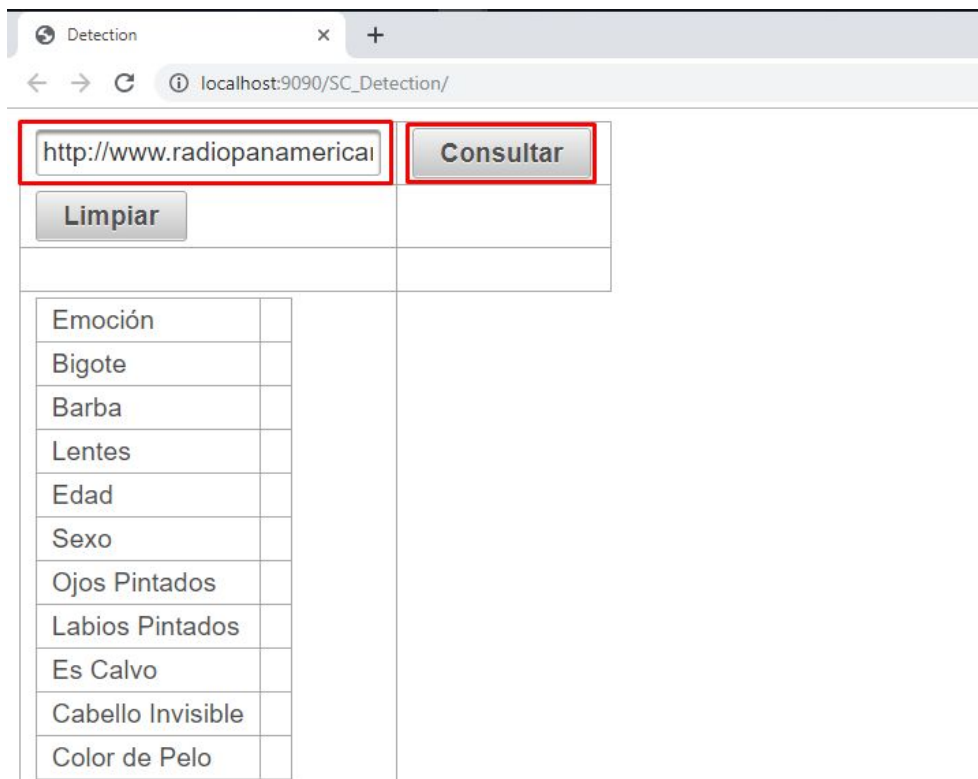
## Ejecutamos el proyecto



The screenshot shows a web browser window with the address bar displaying "localhost:9090/SC\_Detection/". The application interface includes a text input field labeled "Url de la Imagen", a "Consultar" button, and a "Limpiar" button. Below these is a table with 12 rows and 2 columns, listing various facial features for detection.

Emoción	
Bigote	
Barba	
Lentes	
Edad	
Sexo	
Ojos Pintados	
Labios Pintados	
Es Calvo	
Cabello Invisible	
Color de Pelo	


Agregamos la URL y hacemos la consulta



This screenshot shows the same application interface as the previous one, but with the URL "http://www.radiopanamericana" entered into the "Url de la Imagen" input field. The "Consultar" button is highlighted with a red box, indicating the next step in the process.

Emoción	
Bigote	
Barba	
Lentes	
Edad	
Sexo	
Ojos Pintados	
Labios Pintados	
Es Calvo	
Cabello Invisible	
Color de Pelo	

Resultado

		
Emoción		
Bigote	NO	
Barba	NO	
Lentes	SIN LENTES	
Edad	21.0	
Sexo	HOMBRE	
Ojos Pintados	NO	
Labios Pintados	NO	
Es Calvo	NO	
Cabello Invisible	NO	
Color de Pelo	black	