



**ITESO**

UNIVERSIDAD JESUITA DE GUADALAJARA

# HOME SERVICES SYSTEM - HSS

---

## REPORTE 2

**Equipo número:** 2

**Presentado por:**

Fernando Daniel Hernández Amaya y José Antonio Rodríguez Suárez

**Correo Electrónico:**

[fhernandez.amaya@iteso.mx](mailto:fhernandez.amaya@iteso.mx), [jose.rodriguezs@iteso.mx](mailto:jose.rodriguezs@iteso.mx)

**Para el curso de:** Cloud Computing (Development)

**Impartida por:** Mtro. Ojeda Orozco Miguel Angel

**Fecha:** 23/02/2023

**Versión:** 1.2.0

## HOME SERVICES SYSTEM

### MARCO TEÓRICO

En la actualidad existe un sin numero de aplicaciones para distintos servicios para la vida cotidiana, ya sea para adquirir algún servicio de entretenimiento, comprar en línea, acceder a la banca electrónica, etc, sin embargo hay ciertos sectores que aún no han sido tan explotados, aprovechando las tecnologías de la información, uno de ellos son los servicios domésticos o mejor llamados oficios que requiere cualquier vivienda. Entendiendo como servicios para el hogar como: plomería, jardinería, carpintería, fontanería y electricidad.

### JUSTIFICACIÓN

El propósito de este proyecto es crear una solución que le ayude a cualquier persona a tener acceso a una cartera de servicios digital para el hogar, en el cuál pueda localizar, consultar y valorar al proveedor de dichos servicios que más le convenga de acuerdo al costo/beneficio.

### REQUERIMIENTOS

Desde la perspectiva funcional de esta solución, se seguirá una arquitectura cliente- servidor, donde el cliente será el que consuma datos puros o procesados del servidor.

#### Requerimientos funcionales de la aplicación cliente

A continuación, se describen las funcionalidades de la aplicación cliente:

- **Sección de registro del usuario** final (la persona que busca un servicio domestico) a dicha aplicación
- **Manejo de sesiones** (login/logout) para que el usuario final pueda entrar con un email y password que previamente registro en dicho sistema.
- **Sección Home:** La cuál será la raíz o sección principal de la aplicación cliente, el cuál mostrará información elemental del usuario, cómo su user id y un mensaje de bienvenida, y los siguientes accesos:
  - Home
  - Servicios
  - Valoraciones
  - Comentarios
  - Soporte
  - Aviso de privacidad

- Sección Servicios: En dicha sección el usuario podrá navegar y visualizar los la información de contacto de cada proveedor de servicios, dependiendo de la categoría (plomería, jardinería, carpintería, fontanería y electricidad) que se seleccione por medio de una lista desplegable, dicha información sería la siguiente:
  - Nombre del proveedor
  - Dirección del negocio o sucursal
  - Horario
  - Email
  - Link de su página principal
  - Botón “Contratado”
- Sección Valoraciones: En esta sección el usuario tendrá la oportunidad de calificar la calidad del servicio de dicho proveedor contratado, con la finalidad de enriquecer la experiencia del servicio del proveedor por medio de tu raking de estrellas, en donde 5 sería máxima satisfacción y 1 la mínima, esto en diferentes widgets clasificados por categorías: “calidad”, “limpieza”, “tiempo de entrega” y “respeto y cordialidad”, los cuales aparecerán automáticamente, en cuanto se haya contratado al proveedor.
- Sección comentarios: Se contará con un sencillo pero necesario sistema de comentarios, esto para darle la libertad al usuario de dar su punto de vista a alguno de nuestros proveedores o incluso al sistema como tal, el cuál indicará el usuario y la fecha en que sé publicó dicho comentario, siguiendo una serie de reglas “de convivencia” o “respeto”.
- Soporte: En está sección se creará un canal de comunicación directa y por email con la finalidad de dar una respuesta clara y oportuna a todos los usuarios de la aplicación.
- Aviso de privacidad: Es una sección únicamente informativa para la protección de datos sensibles de nuestros usuarios o clientes para efectos de legalidad.

## DESARROLLO DEL PROYECTO

### 1. ANALISIS DE REQUERIMIENTOS

Para la parte del análisis de los requerimientos, tuvimos que analizar desde las necesidades o el por qué elegimos dicho proyecto, todo viene desde una problemática y esta problemática se debe a que en muchas ciudades de México no existe una herramienta o instrumento que califique o valore el trabajo que realizan algunas personas que ofrece algún servicio como un oficio doméstico, como puede ser: carpintería, plomería, electricidad, pintura o jardinería, indagando por la web, si encontramos algunas soluciones que se

ofrecen en la Marketplace y apple store de Android e iOS pero al analizarlas, nos percatamos que sólo ofrecen e intentan gestionar una cartera de proveedores y sólo eso, así que por

dicha razón además por solucionar un problema real es que decidimos construir dicha aplicación.

Además de que surgió desde una situación cotidiana y social, ya que platicando en equipo, surgió el tema que siempre ha sido un problema navegar con diferentes proveedores de servicios sobretodo porque son oficios, no existe un tabulador como tal de precios, los proveedores ponen el precio y la calidad, la fecha de entrega no siempre se cumple por parte del proveedor, dependiendo que tanta confianza se le tenga con el proveedor ya que algunos no le dan seguimiento a su trabajo, o piden un adelanto y ya no lo finalizan, etc.

Los factores que determinaron o en los que podemos generar un impacto son los siguientes:

- ✓ Definir un canal de comunicación y cartera de proveedores, es decir visibilidad
- ✓ Contar con una herramienta de valoración
- ✓ Concentrar la información en una sola herramienta
- ✓ Por parte del cliente, tener una herramienta que le respalde y le ayude a elegir al proveedor que desee
- ✓ Por parte del proveedor contar con un medio que le ayude a venderse en base a su trabajo y comentarios (recomendaciones)

En cuanto la factibilidad del proyecto, se concretaron ciertas funcionalidades para cumplir con las fechas de entrega para dicho proyecto final, en dado caso de querer adjuntar o añadir funcionalidad sería en un sprint o periodo adicional.

## **2. DEFINICION DEL DISEÑO Y ARQUITECTURA DE LA SOLUCIÓN**

Para el diseño y arquitectura de la solución nos enfocamos en 2 elementos esenciales para explicar de la mejor manera el flujo y la estructura del proyecto, los cuales son el diagrama de flujo y el diagrama de arquitectura que a continuación les presentamos.

### **2.1 Flujo de datos – HSS**

En la Fig. 1 y Fig. 2 podemos apreciar el flujo de datos de la aplicación Home Services Systems.

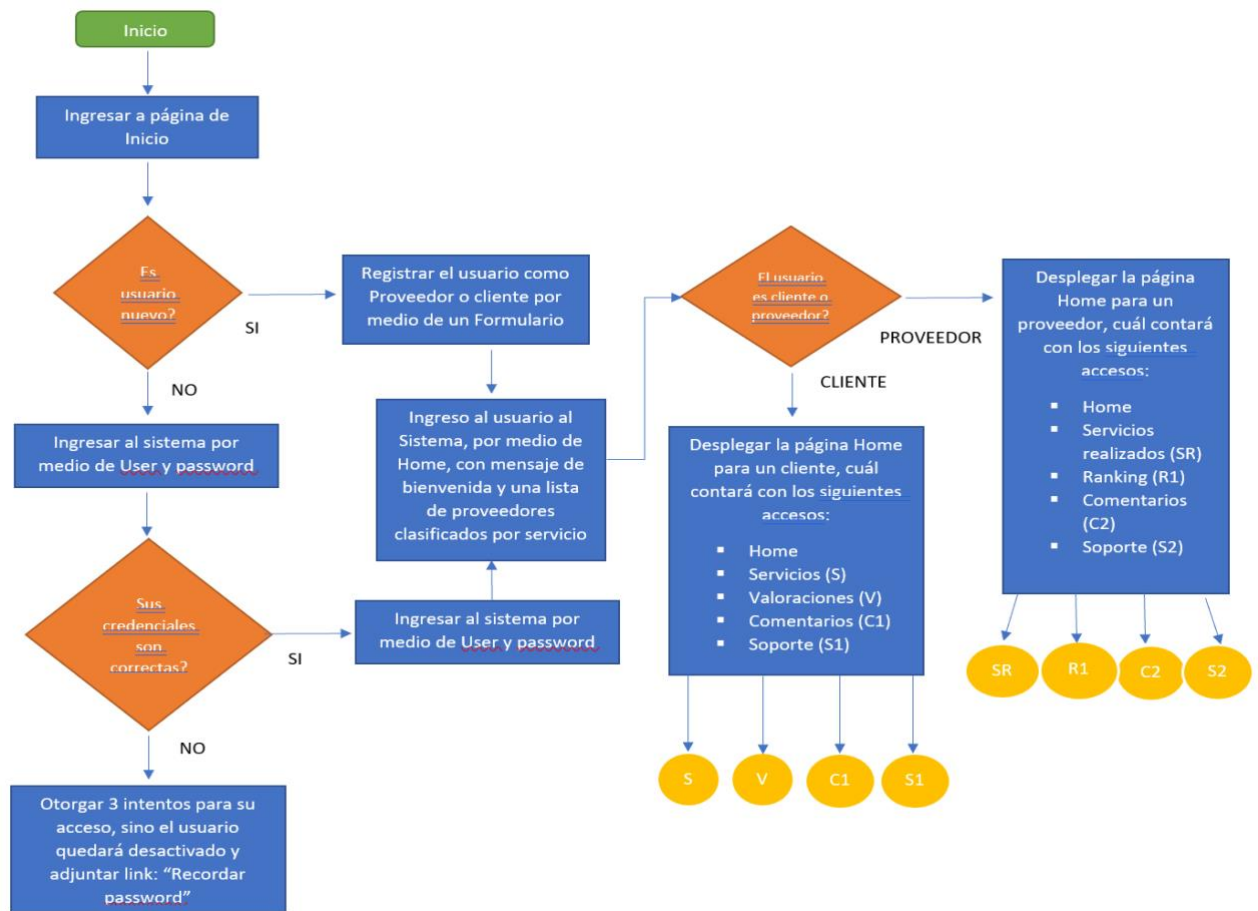


Fig. 1 Primera parte del diagrama de datos del proyecto HSS.

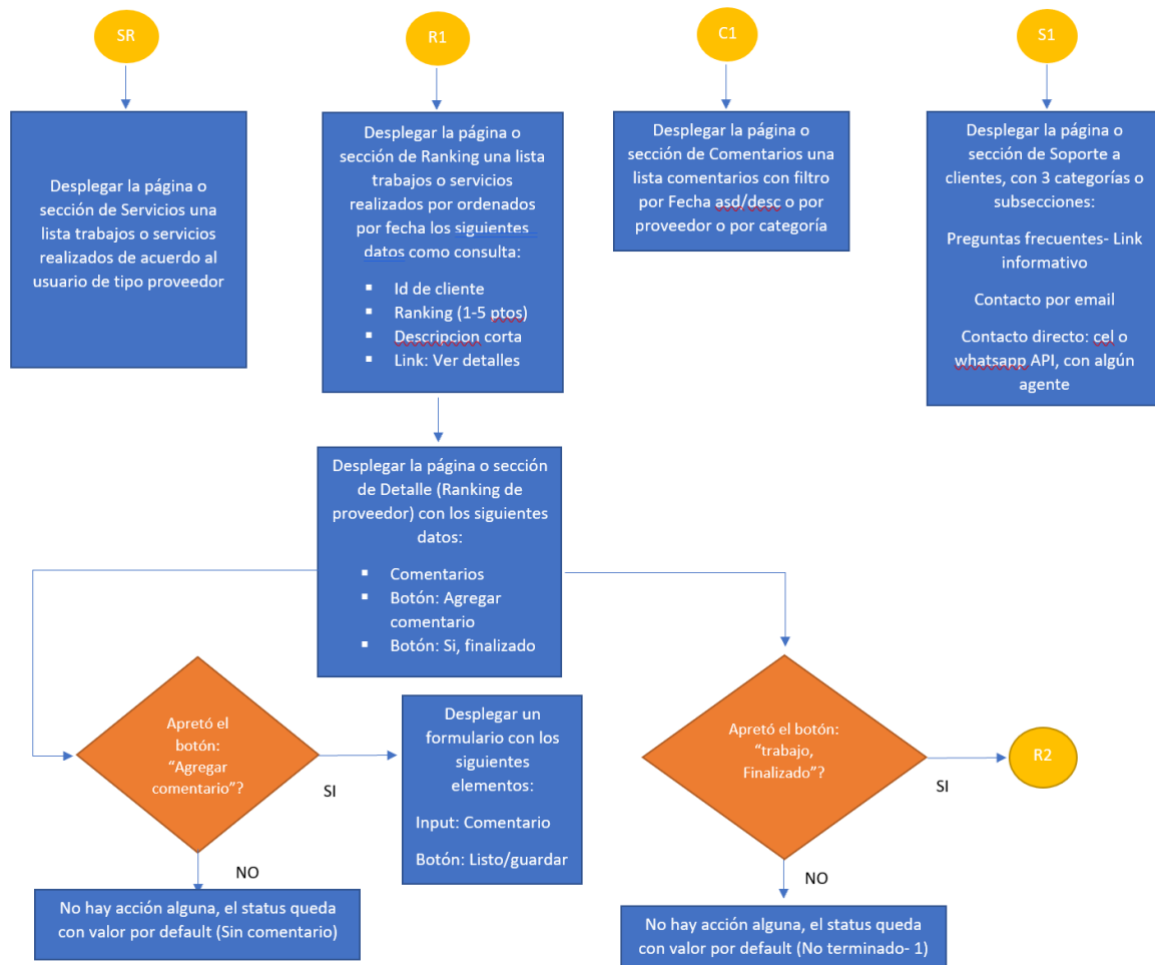


Fig. 2 Segunda parte del diagrama de datos del proyecto HSS.

## 2.2 Diagrama de Arquitectura del proyecto HSS

En la Fig. 3 podemos apreciar la arquitectura del proyecto HSS, donde se muestran los diferentes servicios que incluye dicha solución.

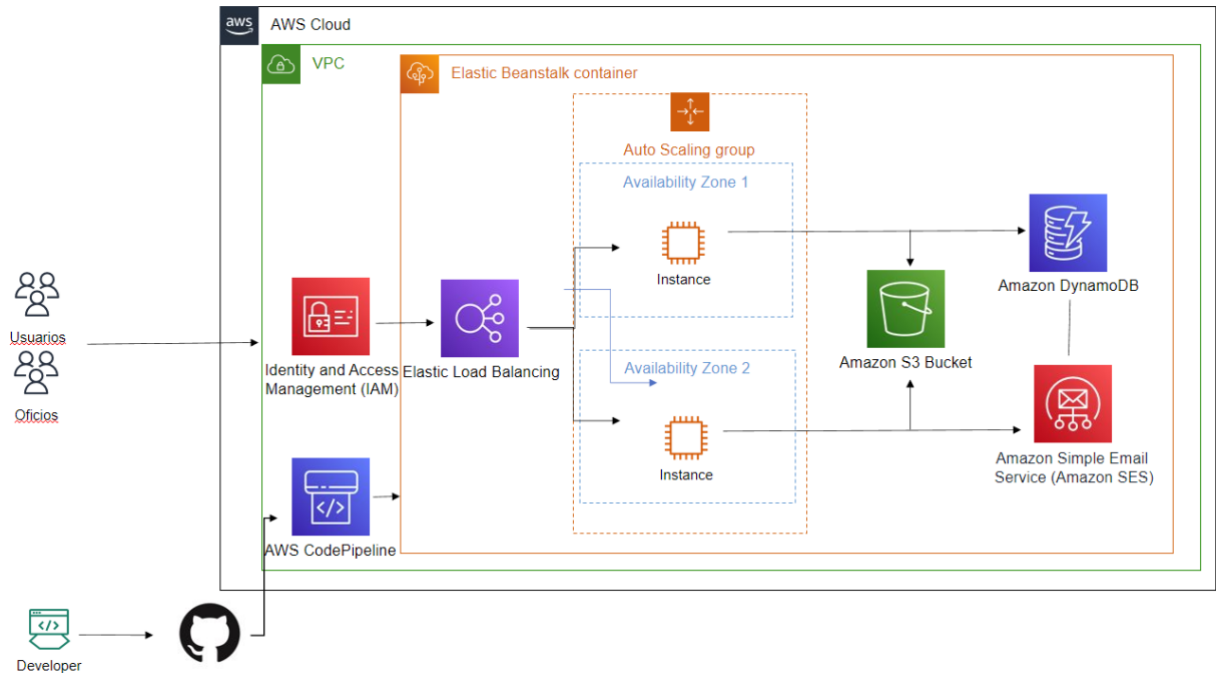


Fig. 3 Diagrama de arquitectura del proyecto HSS

De acuerdo a la Fig. 3 se definieron dichos servicios para atender las siguientes funcionalidades:

- Beanstalk es un servicio que nos permite desplegar y escalar aplicaciones web de forma fácil.
- Amazon Route 53 es un servicio de DNS que nos va a permitir asociar un nombre de dominio con la dirección IP de la aplicación web. Podremos asegurar que la aplicación se mantenga disponible y que los usuarios puedan acceder a ella utilizando un nombre fácil de recordar.
- ELB distribuye las cargas entre varias instancias de EC2. Así podemos asegurar que la aplicación siempre esté disponible y que los usuarios puedan acceder incluso si una de estas instancias falla.
- EC2 nos va a proporcionar instancias de máquinas virtuales que serán utilizadas para ejecutar la aplicación web.
- S3 bucket, con este servicio se podrá implementar para subir archivos tales como PDFs de cotizaciones para formalizar algún trabajo.
- DynamoDB nos podrá servir para almacenar los datos de los usuarios y oficinas.

- SNS nos podrá mandar notificaciones de algún nuevo trabajo disponible en nuestra area y si ha encontrado un nuevo servicio en nuestra area
- API Google Maps, este será útil para poder determinar el servicio que se ofrece en el area determinada.

### 3. CREACIÓN Y CONFIGURACIÓN DE SERVICIOS Y DESARROLLO

La aplicación web ha sido desarrollada con Node.js 18, empleando Elastic Beanstalk, a continuación, se describe con mayor detalle cómo fue utilizado cada uno de los servicios mencionados

#### Elastic Beanstalk

Lo primero fue crear una aplicación desde el panel de control con Elastic Beanstalk, posteriormente se te solicita que crees un entorno para tu aplicación, seleccionamos el entorno, en este caso de entorno servidor web como se puede ver en la Fig. 4

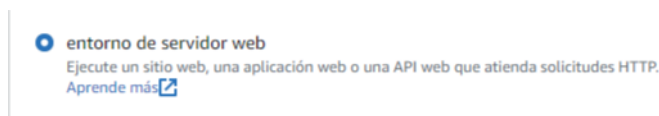


Fig 4. Configuración para un entorno web

Configuramos el entorno, eligiendo la plataforma sobre la cual trabajaremos, en este caso Node.js, dentro de las opciones específicas de Node.js nos permite trabajar con la versión 16 y 14, en nuestro caso elegimos la versión 16 que es la más reciente.

Aun no teníamos escrito ningún código por lo que nuestra mejor opción para el código fue el de prueba que nos ofrece la misma plataforma de Elastic Beanstalk, como se puede ver en la Fig. 5.

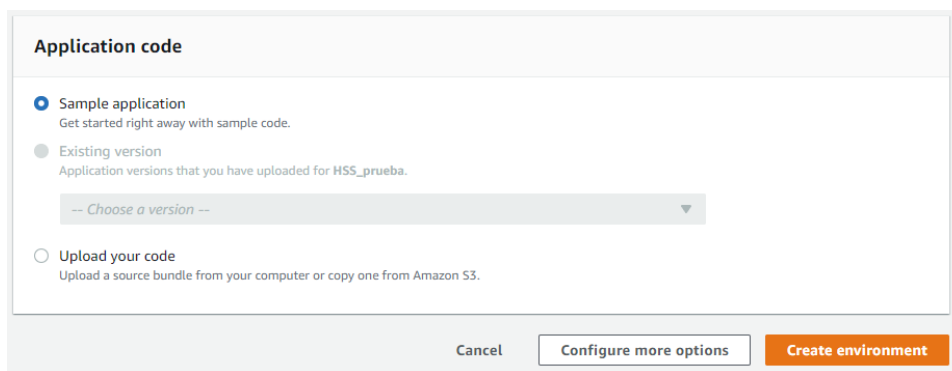


Fig. 5 Configuración de un servicio de Beanstalk sin código

Beanstalk toma un tiempo para ir creando nuestro entorno, por lo que tenemos que esperar un par de minutos



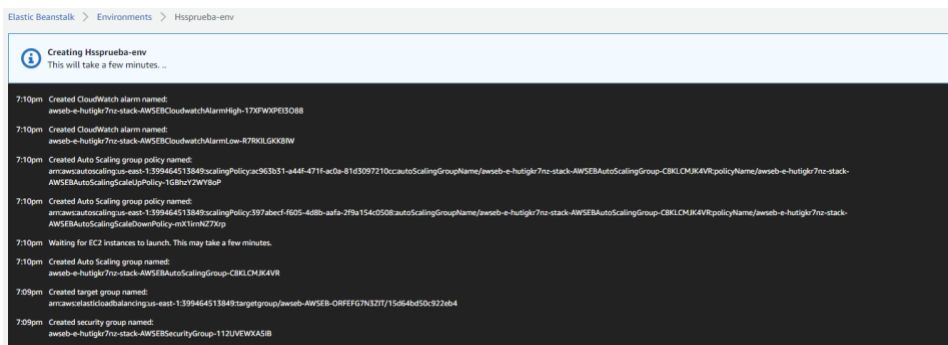


Fig. 6 Ventana de un servicio Beanstalk en proceso de creación.

Posterior a la pantalla de espera podemos ver que todo se ha creado de manera correcta de manera administrada por beanstalk lo cual nos facilita mucho el trabajo y hemos obtenido la pagina principal de nuestro entorno, donde si ponemos de las modificaciones realizadas, la salud de nuestro entorno, la plataforma que estamos utilizando, así como una liga desde la cual podemos visitar nuestra página web, como se puede apreciar en la Fig. 7.

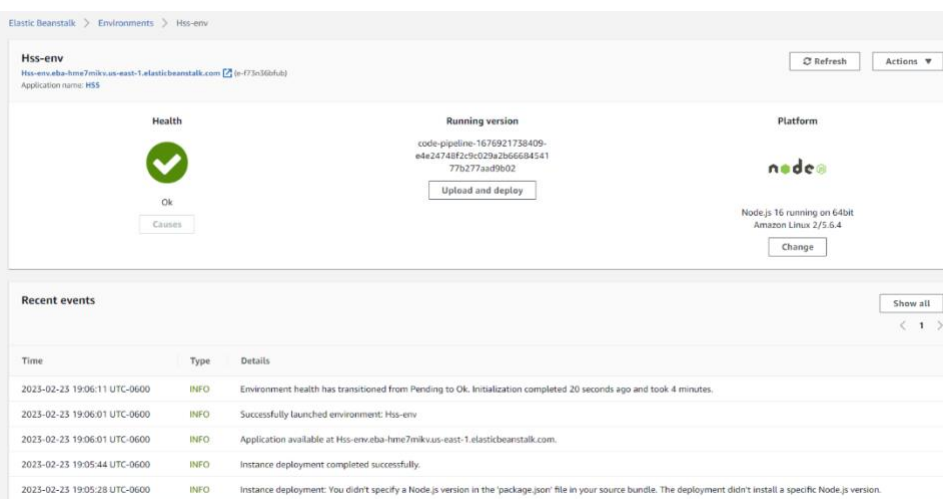


Fig. 7 Ventana de detalle de la creación de un servicio de Beanstalk.

En nuestra primera visita a la URL proporcionada podremos encontrar esta página demo que nos ha desplegado como se puede ver en la Fig. 8.

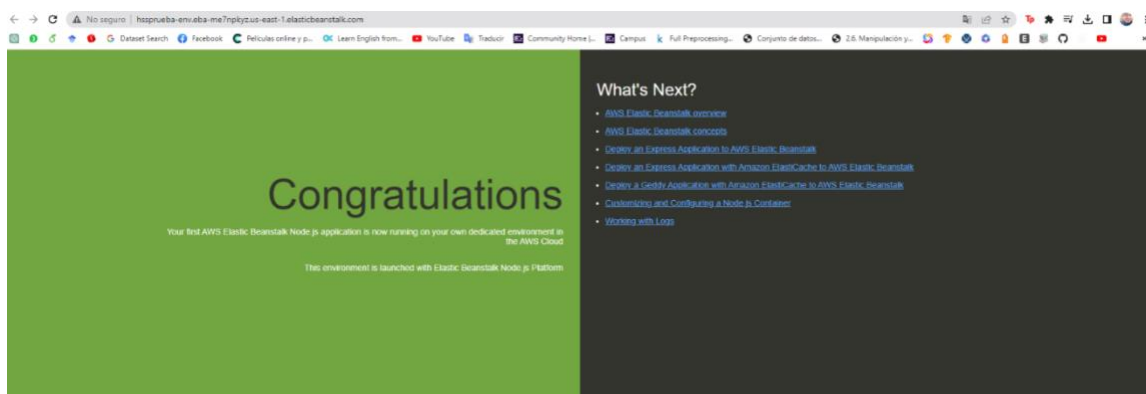


Fig. 8 Página web de confirmación de la creación de un servicio de Beanstalk.

Ahora lo que necesitamos es subir nuestro código que estamos desarrollando, por lo que emplearemos el servicio de CodePipeline.

## GitHub

El primer paso fue crear un repositorio público



Fig. 9 Creación de un repositorio público

El primer código que desplegamos fue la configuración más básica que hacía referencia a nuestro index, así como a nuestro servidor además de las librerías que pretendíamos utilizar más básicas, con este *Hello world* pretendíamos ver la impresión en pantalla del mensaje para corroborar que nuestro código se estaba subiendo y leyendo de manera correcta.

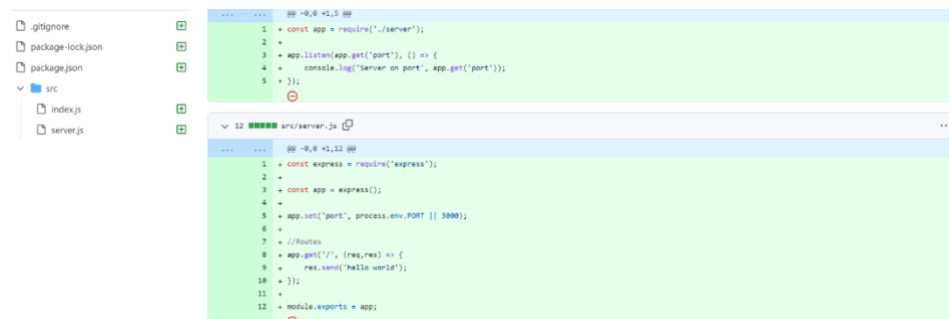
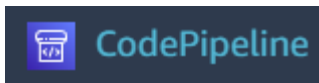


Fig. 10 Probando el servicio de Beanstalk

## CodePipeline

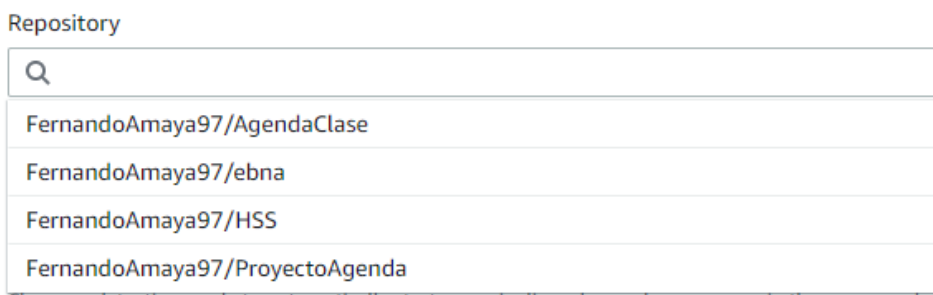
Desde el panel de control de AWS seleccionamos el servicio de CodePipeline, y nos dirigimos a crear un nuevo pipeline



Elegimos un nombre a nuestro pipeline (hay que tener en cuenta que debe ser único), y el nombre de nuestro rol, que se puede dejar el que te crea por defecto.

A continuación, elegimos la fuente de donde va a extraer nuestro código, y conectamos con nuestro repositorio las credenciales necesarias para poder extraer de ahí el código que se va a desplegar

Seleccionamos el repositorio específico como se puede observar en la Fig. 11.



**Fig. 11 Ventana para seleccionar repositorios**

Así como una de las ramas ya existentes en tu código desde la cual tomara el código subido, así mismo te pregunta si deseas que detecte cada *push* hecho a la rama de *GitHub* o que *CodePipeline* realice revisiones cada cierto tiempo para ver si ha habido un cambio, en este caso elegimos: Cada que se haga un cambio en la rama seleccionada.

Para hacer nuestro deploy elegimos Benastalk, elegimos la misma región del pipeline, el nombre de nuestra aplicación y nuestro entorno, nos aparecerá un resumen de configuración, creamos nuestro pipeline y comenzará a realizar el primer despliegue como se puede observar en la Fig. 12.

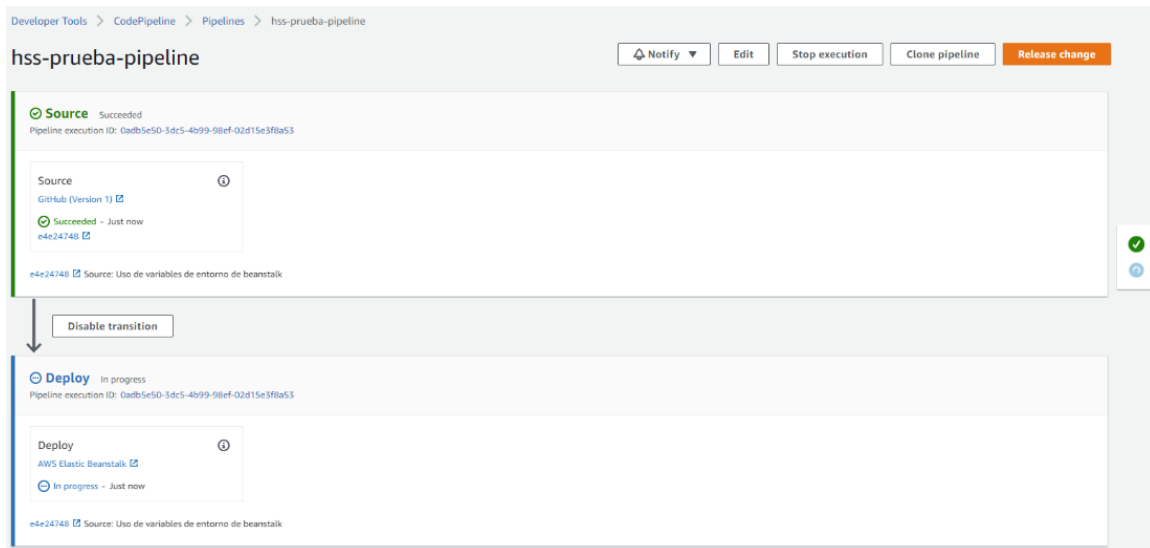


Fig. 12 Ventana de detalle de la configuración de un Pipeline

En nuestro caso, el primer despliegue falló, debido a que no encontraba nuestro archivo index en la raíz de nuestra carpeta HSS, por lo que fue necesario especificarle que index se encontraba dentro de la carpeta *src* y con *.npmrc* le indicamos a *npm* que ejecutara los scripts de instalación de paquetes con permisos de superusuario como se puede observar en la Fig. 13.

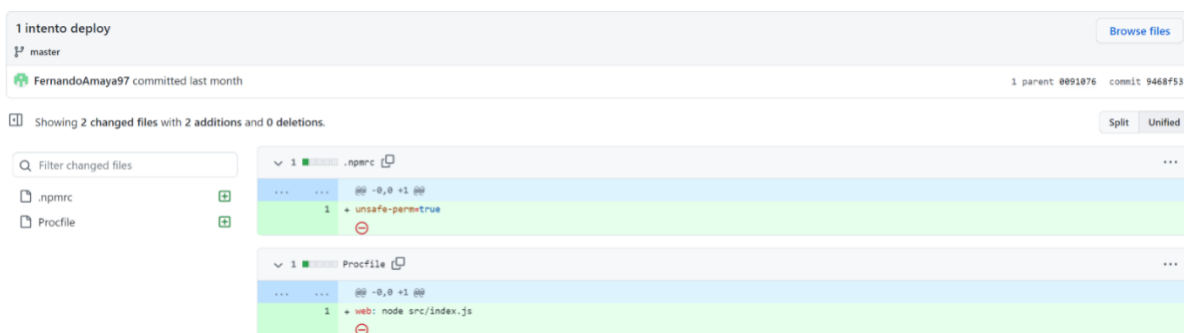


Fig. 13 Primer intento de Deploy en la carpeta root de HSS.

Con esto fue posible corregir el error en el deploy para posteriormente obtener esta pantalla en nuestro pipeline, en donde nos confirma que ha sido desplegada nuestra aplicación de manera correcta.

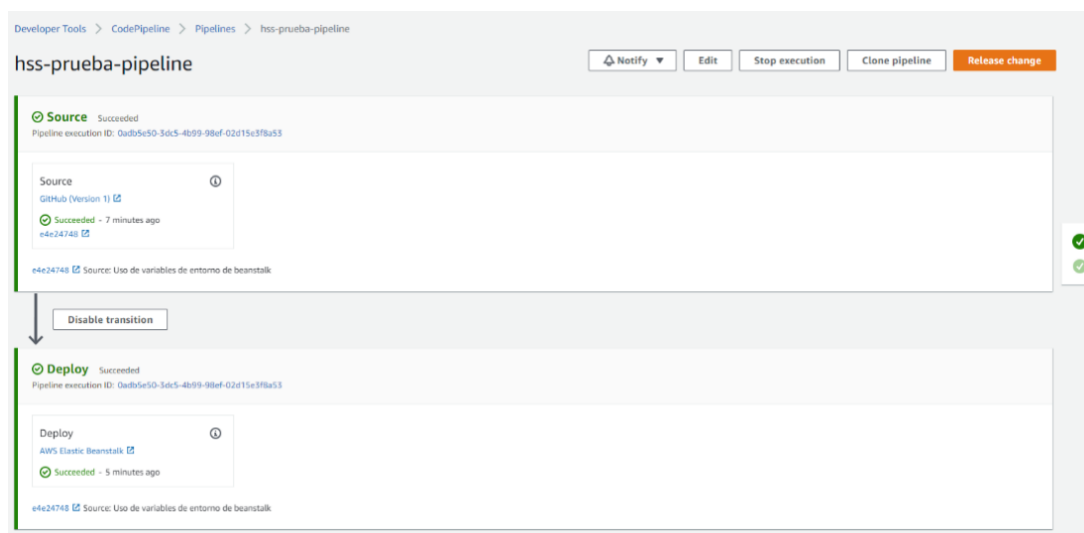


Fig. 14 Ventana de confirmación de Deploy de HSS.

## AWS SDK

Para poder conectar con los diferentes servicios de AWS nuestra aplicación, fue necesario emplear *AWS SDK (Software Development Kit)* que es un conjunto de herramientas, bibliotecas y *APIs* que nos permiten a los desarrolladores interactuar con los servicios de AWS desde nuestras aplicaciones y software.

Nos proporciona una amplia variedad de lenguajes de programación con los que podemos trabajar: *Java, Python, Ruby, PHP, .NET, Node.js, Go* y muchos otros, de esa manera podemos acceder de manera programática a AWS, automatizar tareas integrar aplicaciones y construir soluciones personalizadas en la nube de AWS, como se puede mostrar en la Fig. 15.

## SDK de AWS para JavaScript

Desarrolle e implemente aplicaciones con AWS SDK para JavaScript. El SDK proporciona compatibilidad con TypeScript de primera clase y facilita la llamada a los servicios de AWS mediante las API de JavaScript idiomáticas para crear aplicaciones web móviles, web y Node.js.

Instalar desde NPM

### Desarrollar aplicaciones del lado del servidor



Desarrolle aplicaciones web modernas del lado del servidor utilizando Node, Node Modules y el servidor Node HTTP. Acceda a los servicios de AWS directamente desde sus aplicaciones RESTful utilizando los marcos de Node.js.

[Comience con AWS SDK para JavaScript para Node.js »](#)

### Desarrollar aplicaciones web



Desarrolle aplicaciones front-end modernas y acceda a los servicios de AWS directamente desde el código JavaScript que se ejecuta en su navegador.

[Comience con AWS SDK para JavaScript en el navegador »](#)

### Desarrollar aplicaciones móviles



Desarrolle aplicaciones móviles modernas con React Native y acceda a los servicios de AWS directamente desde sus dispositivos móviles iOS y Android.

[Comience con AWS SDK para JavaScript en React Native »](#)

Fig. 15 Ventana para la instalación de algún SDK en algunas tecnologías web.

Lo primero fue instalarlo mediante ***npm install aws-sdk***

```
"aws-sdk": "^2.1312.0",
```

Posterior a esto fue necesario llamarlo en aquellos archivos en donde iba a ser utilizado

```
2 const AWS = require("aws-sdk");
```

## DynamoDB

Empleamos *DynamoDB* para poder construir nuestra base de datos en la cual almacenaremos los datos de cada uno de nuestros proveedores. El primer paso para poder utilizar DynamoDB ha sido crear nuestra tabla desde la consola



Fig. 16 Acceso para la creación de una base de datos DynamoDB

Desde la pantalla principal elegimos crear una nueva tabla, en nuestro caso emplearemos una tabla específicamente para almacenar a los proveedores y una tabla distinta para poder almacenar a los clientes por separado, dejamos la configuración de la tabla que tiene por default, procedemos a crear la tabla, una vez disponible el status de nuestra nueva tabla cambia a Activa como se puede observar en la Fig. 17.

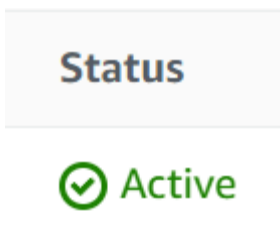


Fig. 17 Ventana para la instalación de algún SDK en algunas tecnologías web.

Ahora que es posible utilizar nuestra tabla podemos interactuar con ella desde *AWS-SDK*, para lo que es necesario tener habilitado un usuario con los permisos para poder agregar un ítem a la tabla, para esto es necesario implementar el servicio de *IAM*, una vez teniendo el *ACCESS KEY* de usuario es posible manipular la tabla, nos conectamos a *DynamoDB* desde *AWS-SDK* como podemos apreciar en la Fig. 18.

```
const dynamodb = new AWS.DynamoDB.DocumentClient({
  region: 'us-east-1',
  accessKeyId: '',
  secretAccessKey: ''
}); //intenta conectarse a dynamodb
```

Fig. 18 Conexión a DynamoDB desde código.

Creamos nuestras variables del formulario de registro, véase Fig. 19.

```
const {correo, password, nombre, oficio, dc, direccion, ciudad, estado, cp} = req.body;
const createdAt = new Date().toISOString();
const id = v4();
```

Fig. 19 Creación de las variables del formulario en este caso para Proveedores.

Y pasamos los valores de cada una de estas variables a nuestro ítem de la tabla de *DynamoDB* como se puede observar en la Fig. 20.

```
const newProv = {
  'id': id,
  'correo': correo,
  'password': password,
  'nombre': nombre,
  'oficio': oficio,
  'dc': dc,
  'direccion': direccion,
  'ciudad': ciudad,
  'estado': estado,
  'cp': cp,
  'createdAt': createdAt,
  'urlFoto': result['Location']
};

const params = {
  TableName: "proveedor",
  Item: newProv,
};
```

Fig. 20 Pasarela de valores desde variables a Items de la base de datos creada.

Esto se hace a través de un método *put* que es pasado como una promesa e imprimimos cualquier error que se haya podido producir, como podemos apreciar en la Fig. 21.

```
try {
  await dynamodb.put(params).promise();
  console.log("Success");
  res.render('proveedores/login_prov');
} catch (err) {
  console.log("Error", err);
  res.status(500).send("Error al crear el proveedor");
}
```

Fig. 21 Promesa de método Put mediante un bloque Try-catch.





Para poder trabajar de manera conjunta los dos miembros del equipo del proyecto, creamos desde una cuenta adicionalmente un segundo usuario, como se puede observar en la Fig. 22.

Tony-FullAccess

Fig. 22 Icono del nuevo acceso creado.

Con este segundo usuario teníamos la posibilidad de hacer las modificaciones necesarias a los servicios de AWS, con todos los permisos necesarios, véase la Fig. 23.

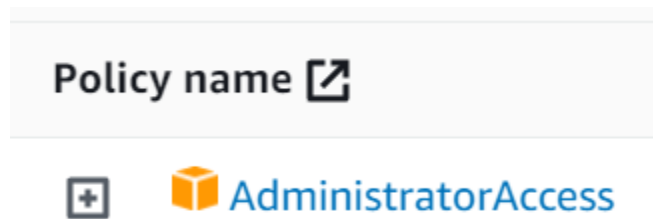


Fig. 23 Permisos creados al nuevo colaborador.

Desde la consola de IAM creamos un nuevo usuario, y elegimos el nombre de este, para este caso en concreto creamos un usuario que serán nuestros usuarios de la aplicación, véase Fig 24.

hss-user

Fig. 24 Acceso de nuevo usuario creado desde IAM.

En los permisos que se le van a asignar a *hss-user* elegimos un *FullAccess* tanto para *DynamoDB* como para *S3* esto nos permitirá poder subir y bajar o modificar nuestra foto de perfil, así como nuestros datos personales como usuario como podemos ver en la Fig. 25.

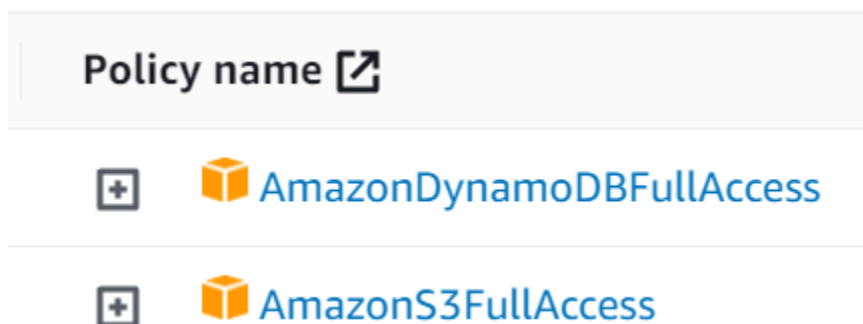


Fig. 25 Accesos para la configuración de la gestión de datos en S3.

Para poder utilizar los servicios de AWS con *AWS SDK* era necesario emplear un Access Key por lo que se le creó al usuario *hss-user* desde la sección de *Security Credentials*, véase Fig. 26.

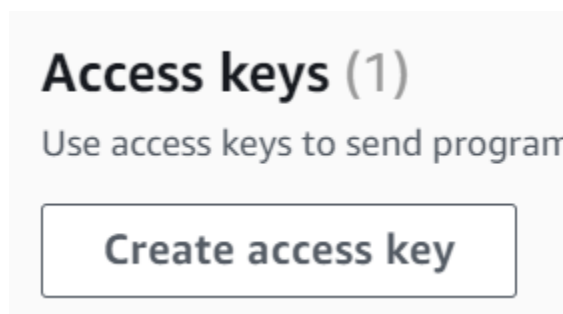



Fig. 26 Ventana para la creación de un Access key.

En el caso de las notificaciones que serán enviadas por medio del servicio de SNS se creó el usuario

## hss-notificaciones

Fig. 27 Acceso para el servicio de notificaciones SNS.

Al cual se le dieron los permisos correspondientes a la función para la cual se va a utilizar, véase Fig. 28.

Policy name 

  AmazonSESFullAccess

Fig. 28 Acceso para otorgar permisos para notificaciones SES.

### S3

El servicio de S3 se emplea específicamente una cubeta, Fig 29, en la cual vamos a almacenar la foto de perfil de nuestros usuarios, por lo cual usaremos de manera concurrencia el usuario hss-user de IAM ya configurado, cabe resaltar que esta cubeta no tiene ninguna característica especial, se ha definido únicamente como públicos todos los archivos.

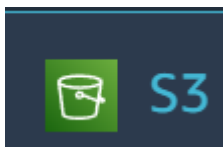


Fig. 29 Icono del servicio S3

hss-profileimages

Fig. 30 Acceso para la gestión de imagenes

Para poder subir nuestra fotografía de perfil ha sido necesario colocar nuestra configuración inicial con las credenciales de hss-user

```
const S3 = require('aws-sdk/clients/s3');

const s3 = new S3({
  region: 'us-east-1',
  accessKeyId: '',
  secretAccessKey: ''
});
```

Fig. 31 Declaración del servicio S3 en el código.

Nuestro formulario permite la subida de imágenes, como se puede ver en la Fig. 32.

```
const result = await uploadFile(req.file)
```

Fig. 32 Declaración de una constante para permitir subir imágenes a S3, desde código.

Por medio de la función *uploadfile* especificamos el *bucket* al que se subirá nuestra fotografía de perfil, y definimos que se va a subir de manera pública para que posteriormente podamos acceder a ella, como podemos apreciar en la Fig.33.

```
//subir fotos de perfil a s3
function uploadFile(file) {
  const fileStream = fs.createReadStream(file.path)

  const uploadParams = {
    Bucket: 'hss-profileimages',
    Body: fileStream,
    Key: file.filename,
    acl: 'public-read' //Agregamos que ACL para hacer publica la imagen
  }

  return s3.upload(uploadParams).promise()
}
exports.uploadFile = uploadFile;
```

Fig. 33 Creación de una función para subir imágenes de perfil por medio del servicio S3.

Definimos el parámetro *urlfoto* en nuestra tabla de dynamoDB donde almacenaremos la *URL* de nuestra imagen de perfil para poder requerirla de manera posterior, ya sea para verla o editarla, véase Fig. 34.

```
'urlFoto': result['Location']
```

Fig. 34 Variable en dónde almacenaremos la URL de la imagen por subir.

SES

Para el sistema de notificaciones fue implementado SES, el cual nos va a permitir mandar diferentes alertas a nuestros usuarios, así como la bienvenida cuando se han registrado a nuestra aplicación, véase Fig. 35.

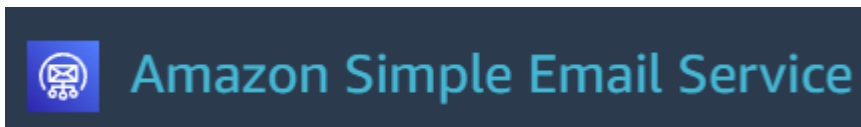


Fig. 35 Icono del servicio de Amazon SNS

Para esto ha sido necesario registrar un correo para nuestra aplicación y registrarla como una identidad verificada, este correo será desde el que se mandaran los correos de manera automatizada, véase Fig. 36.

<input type="checkbox"/>	Identity	<input type="checkbox"/>	Identity type	<input type="checkbox"/>	Identity status
<input type="checkbox"/>	homeservicessystem@gmail.com		Email address		Verified

Fig. 36 Ventana de confirmación del servicio SNS verificado

Configuramos el servicio desde nuestra aplicación con las credenciales correspondientes, así como la versión y región en donde se usará

```
const ses = new AWS.SES({
  apiVersion: '2010-12-01',
  region: 'us-east-1',
  accessKeyId: '',
  secretAccessKey: ''
});
```

Fig. 37 Configuración de servicio SNS desde código fuente.

Configuramos nuestro mensaje que será enviado a nuestros usuarios que se han registrado como proveedores al correo con el que han llenado el formulario y definimos cual es nuestro correo de envío, como se puede apreciar en la Fig. 38.

```
const paramsSES = {
  Destination: {
    ToAddresses: [correo]
  },
  Message: {
    Body: {
      Text: {
        Data: 'Gracias por registrarte como Proveedor de servicios a nuestra aplicación web HSS'
      }
    },
    Subject: {
      Data: 'Confirmación de registro'
    }
  },
  Source: 'homeservicessystem@gmail.com'
}
```

Fig. 38 Configuración del mensaje que será enviado a los proveedores una vez dados de alta.

Creamos la función que se encargará de mandar el correo de bienvenida a un nuevo proveedor, y visualizamos únicamente que ha sido enviado de manera correcta, véase Fig. 39.

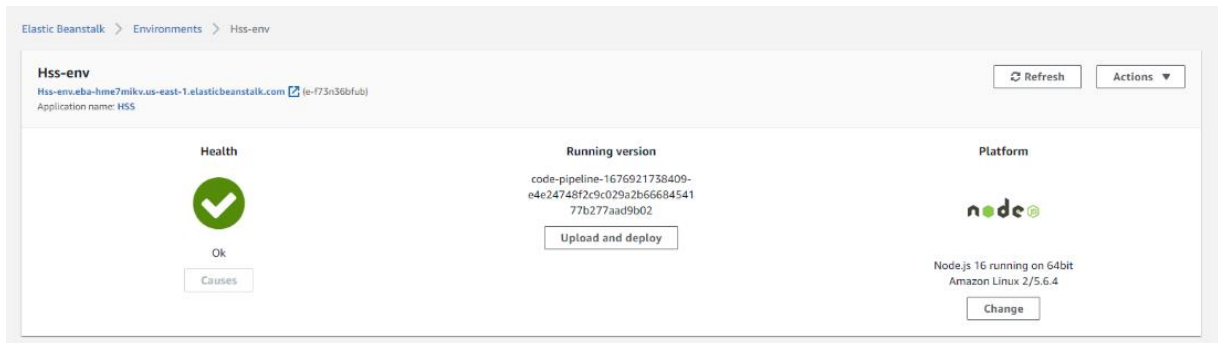
```
ses.sendEmail(paramsSES, function (err, data) {
  if (err) {
    console.log(err);
  } else {
    console.log('Correo electrónico enviado correctamente');
  }
});
```

Fig. 39 Bloque de código sólo para validar que se haya enviado el email al proveedor.

#### 4. TESTEO DE LA SOLUCIÓN

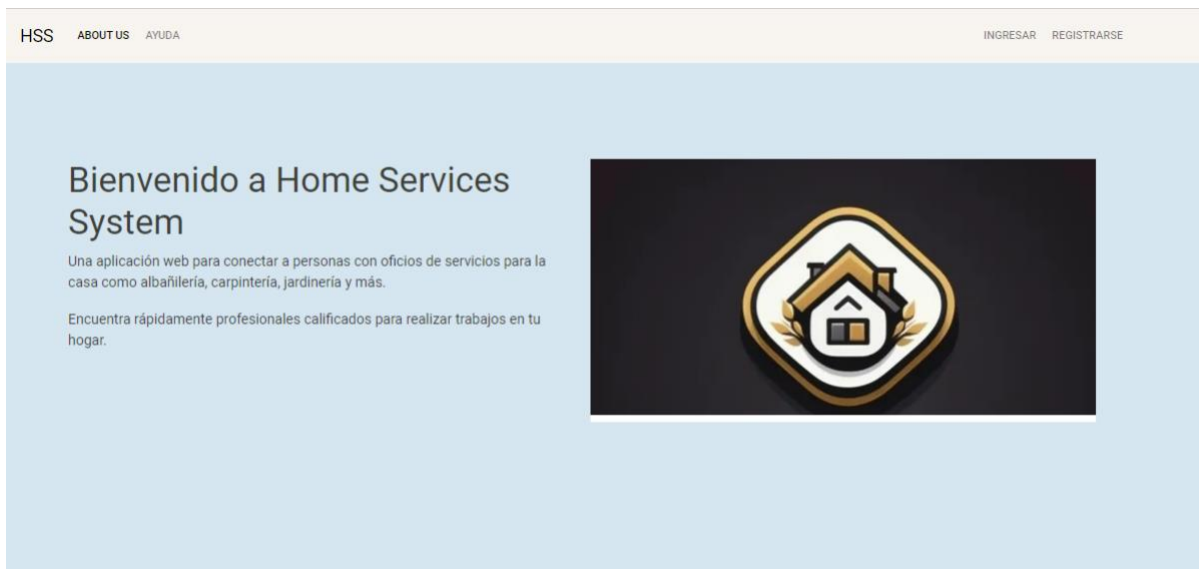
Para poder probar que nuestra aplicación se encuentra funcionando en estos momentos accedemos a la URL que nos proporciona en entorno de Beanstalk al poner a funcionar nuestra aplicación como podemos observar en la Fig. 40.

Link de acceso: <http://hss-env.eba-hme7mikv.us-east-1.elasticbeanstalk.com/>



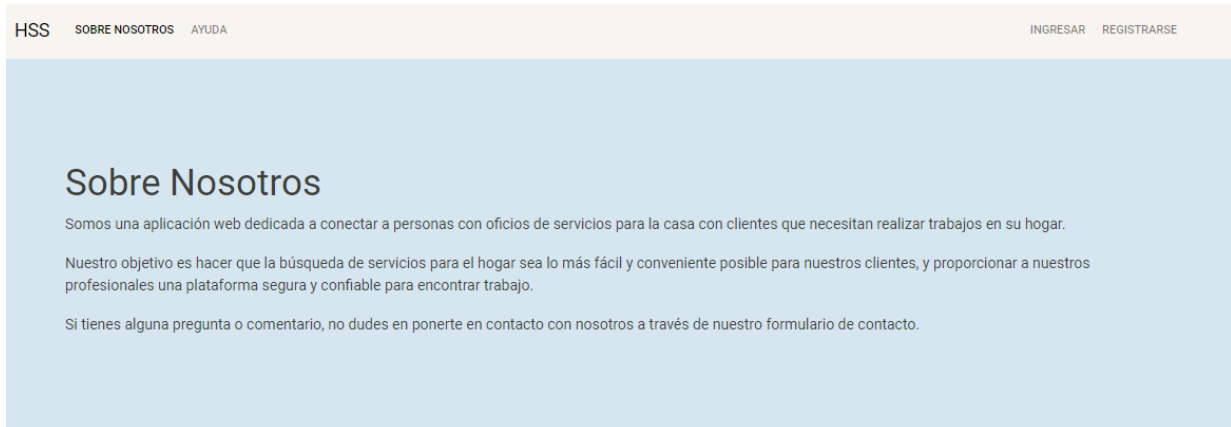
**Fig. 40 Ventana de detalle al crear la URL que nos proporciona Beanstalk**

Donde tenemos funcionando nuestra pantalla de inicio como se puede ver en la Fig. 41.



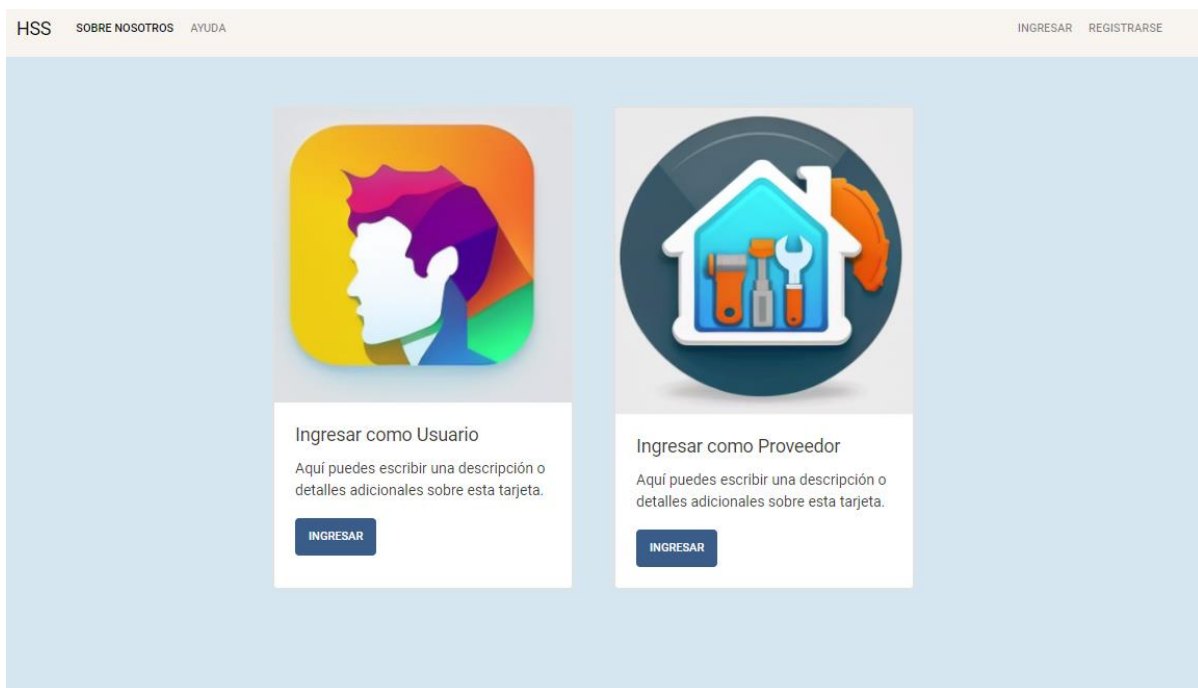
**Fig. 41 Página de inicio de la aplicación web: Home Services Systems**

En la sección Sobre Nostros podemos encontrar una breve descripción de lo que pretende ser nuestra aplicación, véase Fig. 42.



**Fig. 42** Página “Sobre Nosotros” de la aplicación web: Home Services Systems

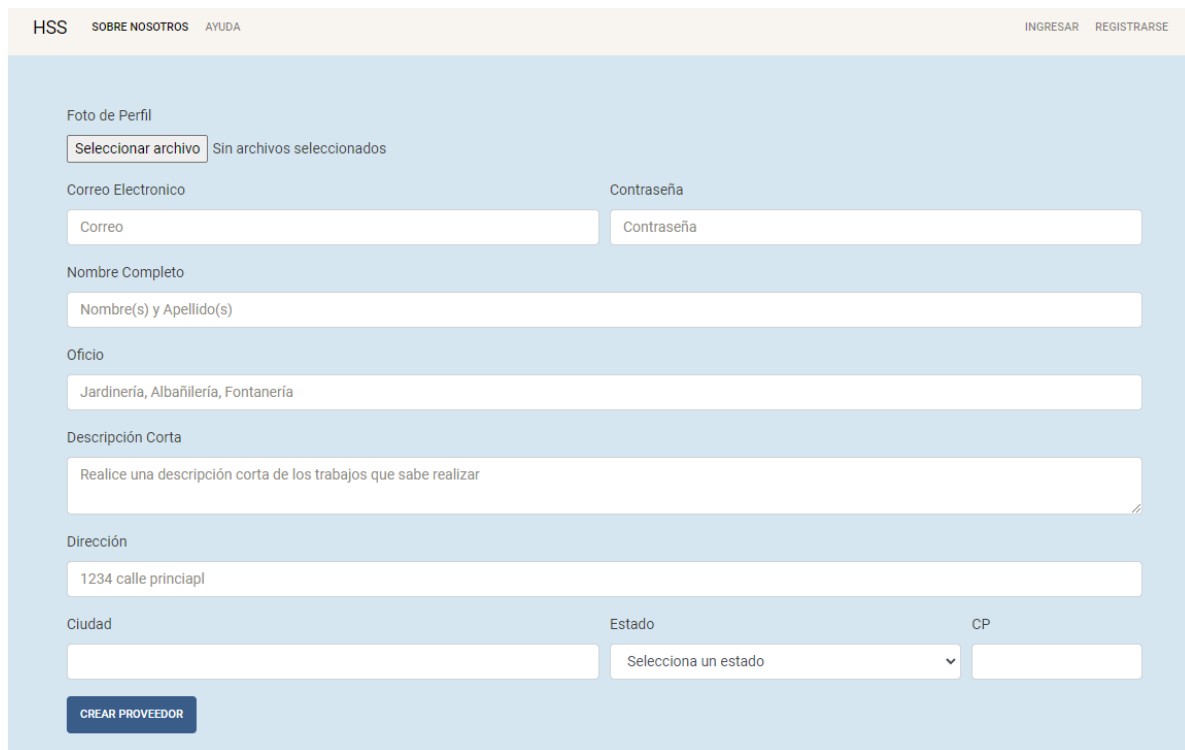
Tenemos habilitado el botón de ingresar, donde elegimos nuestro rol ya sea como usuario o proveedor de servicios, véase Fig. 43.



**Fig. 43** Página para seleccionar un rol de la aplicación web: Home Services Systems

Y en la sección de registro, tenemos por ahora creado el formulario correspondiente para el registro de un nuevo proveedor de servicios, véase Fig. 44.





HSS    SOBRE NOSOTROS    AYUDA    INGRESAR    REGISTRARSE

Foto de Perfil  
Seleccionar archivo Sin archivos seleccionados

Correo Electronico    Contraseña  
Correo    Contraseña

Nombre Completo  
Nombre(s) y Apellido(s)

Oficio  
Jardinería, Albañilería, Fontanería

Descripción Corta  
Realice una descripción corta de los trabajos que sabe realizar

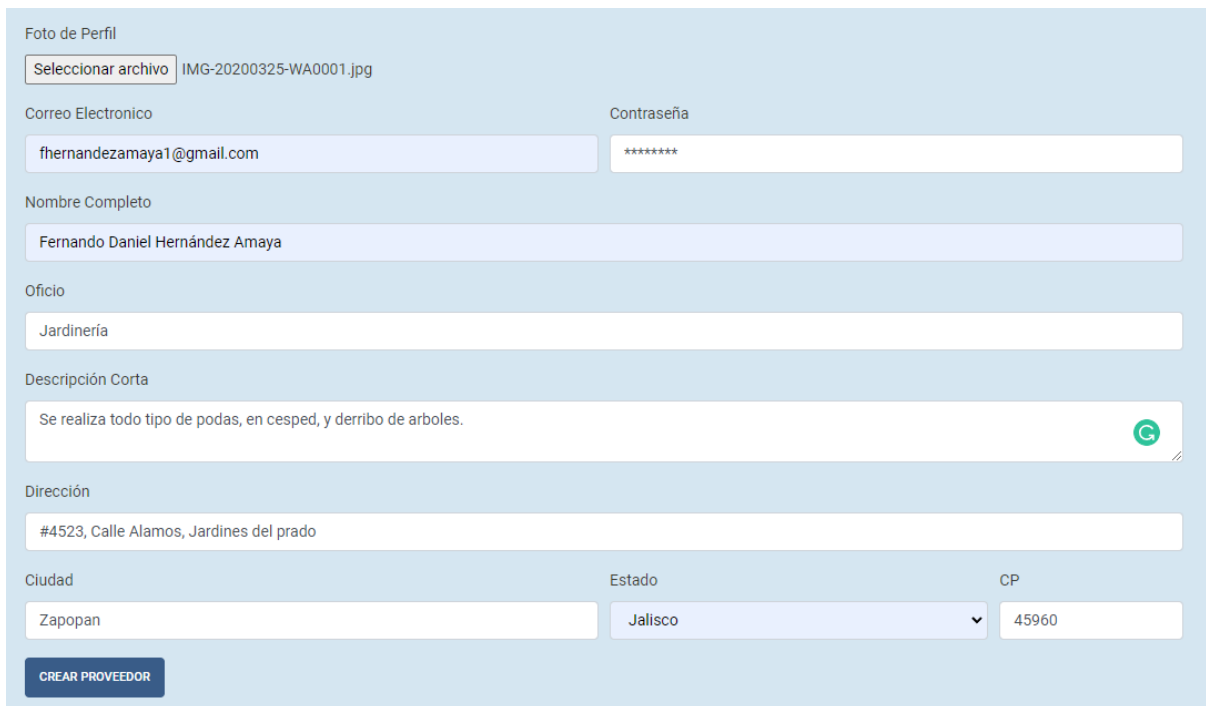
Dirección  
1234 calle principi

Ciudad    Estado    CP  
    Selecciona un estado   

CREAR PROVEEDOR

**Fig. 44** Formulario de la página “Resgistrarse” de la aplicación web: HSS para proveedores.

En nuestro formulario de registro llenamos los campos respectivos con cada uno de los datos solicitados, y procedemos a presionar crear proveedor como podemos observar en la Fig. 45.

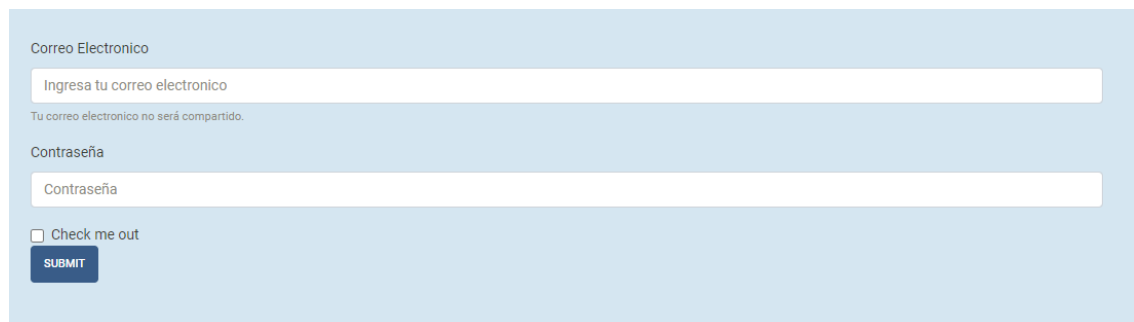


The registration form for providers includes the following fields and elements:

- Foto de Perfil:** A button labeled "Seleccionar archivo" and a text field showing the filename "IMG-20200325-WA0001.jpg".
- Correo Electronico:** A text field containing "fhernandezamaya1@gmail.com".
- Contraseña:** A password field with masked characters "\*\*\*\*\*".
- Nombre Completo:** A text field containing "Fernando Daniel Hernández Amaya".
- Oficio:** A text field containing "Jardinería".
- Descripción Corta:** A text area containing "Se realiza todo tipo de podas, en cesped, y derribo de arboles." with a green circular icon on the right.
- Dirección:** A text field containing "#4523, Calle Alamos, Jardines del prado".
- Ciudad:** A text field containing "Zapopan".
- Estado:** A dropdown menu showing "Jalisco".
- CP:** A text field containing "45960".
- CREAR PROVEEDOR:** A blue button at the bottom left.

**Fig. 45** Formulario de la página “Resgistrarse” para proveedores.

En este momento será registrado nuestro nuevo usuario en nuestra base de datos y se podrá redireccionar a la página de *login* para poder acceder, véase Fig. 46.



The login window for providers includes the following fields and elements:

- Correo Electronico:** A text field with the placeholder "Ingresa tu correo electronico". Below it, a small note states "Tu correo electronico no será compartido."
- Contraseña:** A password field with the placeholder "Contraseña".
- Check me out:** An unchecked checkbox.
- SUBMIT:** A blue button at the bottom left.

**Fig. 46** Ventana login para el rol de proveedores.

Se ha enviado un correo de confirmación de suscripción a la plataforma por medio de correo electrónico, véase Fig. 47.

## Confirmación de registro Recibidos x



homeservicessystem@gmail.com a través de amazonses.com

para mí ▼

Gracias por registrarte como Proveedor de servicios a nuestra aplicación web HSS

**Fig. 47 Email de confirmación al registro del sistema HSS.**

Por la parte del desarrollador podemos confirmar que los datos han sido registrados en nuestra base de datos de DynamoDB, como se puede observar en la Fig. 48.

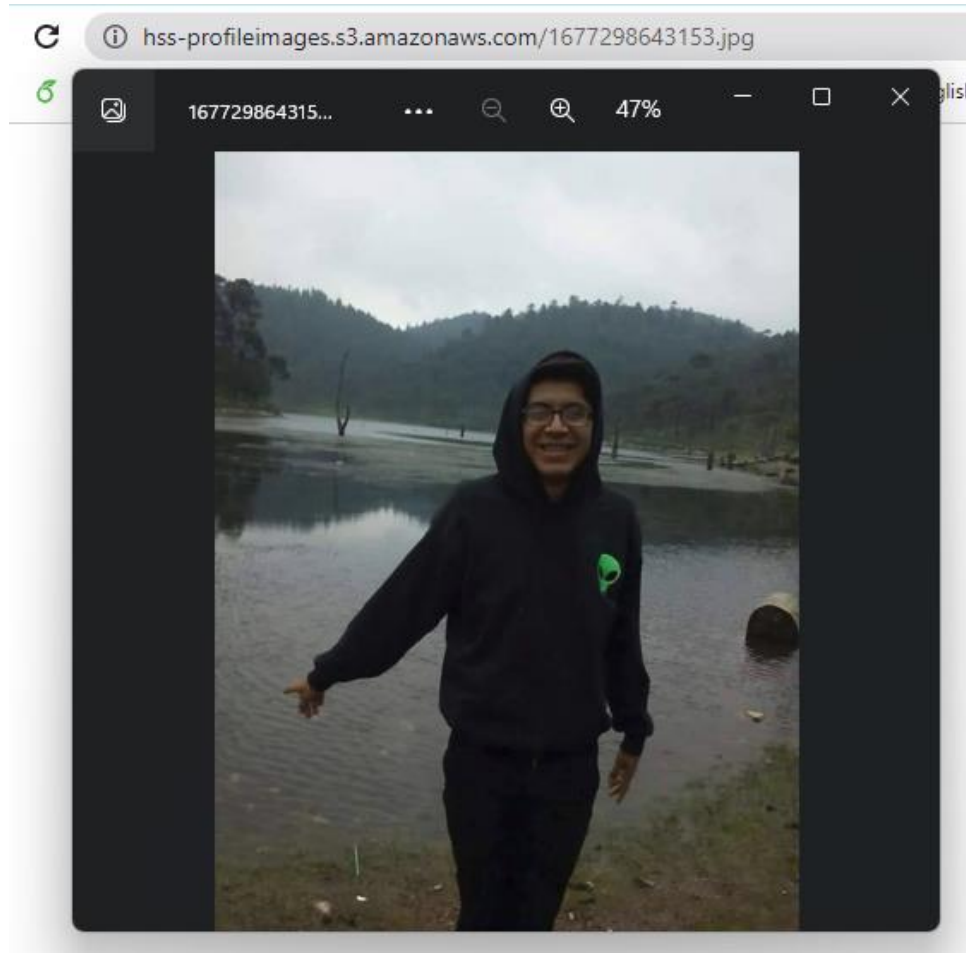
Attributes			A
Attribute name	Value	Type	
id - Partition key	b3b09dce-c618-483b-aa03-cb5b06951f12	String	
correo - Sort key	fhernandezamaya1@gmail.com	String	
ciudad	Zapopan	String	
cp	45960	String	
createdAt	2023-02-25T04:17:23.155Z	String	
dc	Se realiza todo tipo de podas, en césped, y derribo de árboles.	String	
direccion	#4523, Calle Alamos, Jardines del prado	String	
estado	Jalisco	String	
nombre	Fernando Daniel Hernández Amaya	String	
oficio	Jardinería	String	
password	*****	String	
urlFoto	https://hss-profileimages.s3.amazonaws.com/1677298643153.jpg	String	

**Fig. 48 Registro de los datos ingresados al sistema HSS desde DynamoDB.**

Consultamos la URL que nos arroja nuestro item de urlFoto por medio del link:

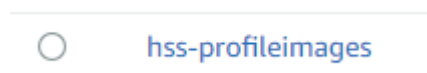
<https://hss-profileimages.s3.amazonaws.com/1677298643153.jpg>

Y al pegarla en el buscador es posible acceder a dicha fotografía, como se puede apreciar en la Fig. 49.



**Fig. 49 Imagen cargada desde el sistema HSS y vista desde bucket S3**

Al consultar directamente con el servicio de AWS en el bucket creado como destino, de acuerdo a la Fig. 50.



**Fig. 50 Acceso a las imágenes de perfil de HSS**

Confirmamos que efectivamente se encuentra almacenada y disponible como se puede ver en la Fig. 51.

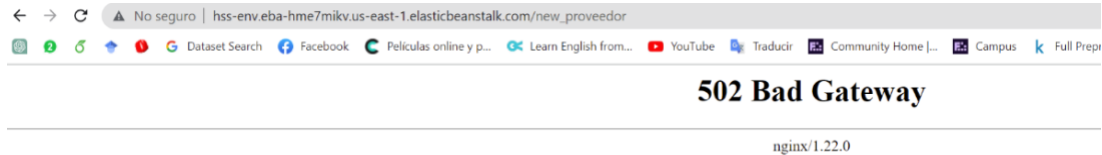


**Fig. 51 Confirmación de imagen de perfil almacenada.**

## 5. PROBLEMAS O RETOS IDENTIFICADOS

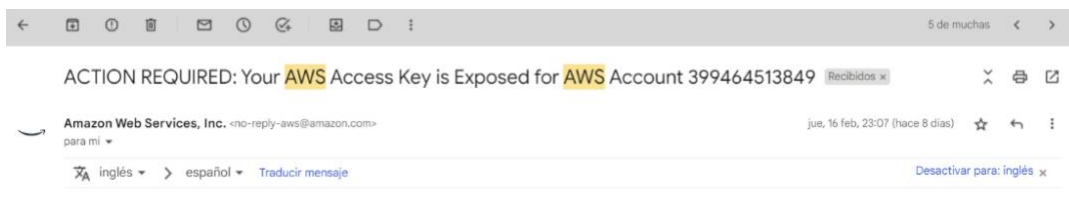
Uno de los problemas que más problemas nos causó durante el desarrollo del proyecto hasta el momento ha sido que comenzamos utilizando un *framework* llamado *serverless*, el cual supuestamente te ayuda a crear de manera más sencilla la interacción entre los servicios, debido a nuestra falta de expertiz creímos que nos serviría sin embargo tuvimos que dejar de usarlo y volver a lo más básico y lógico que era utilizar el servicio de AWS-SDK que ya nos proporciona mismo AWS.

Como se puede apreciar en la Fig. 52 había un problema de comunicación entre los servidores lo cual no se pudo resolver a pesar de hacer un seguimiento de log de errores.



**Fig. 52 Problema de comunicación entre servidores**

Por otra parte, se había dejado a la vista el código en el repositorio de manera pública mis *Access Key* de AWS por lo cual recibí constantes correos para que protegiera mis credenciales y no las estuviera dejando a la vista de personas por internet, véase Fig. 53.




**Fig. 53 Email recibido de problema con exposición de código de forma publica.**

Pero esto fue resuelto de la siguiente manera:

Desde la aplicación creada desde el servicio de Beanstalk accedemos, posteriormente seleccionamos nuestro entorno desplegado, accedemos al menú de configuración, en la sección de Software seleccionamos editar, existe una opción llamada *Environment Properties*, vease Fig. 54 en donde es posible cargar de manera segura nuestras variables de entorno de manera segura y que nuestra aplicación siga funcionando sin ningún problema.

### Environment properties

The following properties are passed in the application as environment properties. [Learn more](#) 

Name
ID
KEY
REGION

Fig. 54 Ventana para la configuración de propiedades de enviroment.

## COSTO

A continuación, se despliegan los costos calculados por medio de la plataforma de AWS *pricing calculator*, Como se pueden observar en la Fig. 55 y Fig 56.

24/2/23, 23:17

AWS Pricing Calculator

Contact your AWS representative:  
<https://aws.amazon.com/contact-us/>

Export date: 24/2/2023

Language: English

Estimate title: My Estimate

Estimate URL: <https://calculator.aws/#/estimate?id=dfbd87bfe82ad7586211ce4c960c405812ce4f7d>

Estimate summary		
Upfront cost	Monthly cost	Total 12 months cost
<b>180.00 USD</b>	<b>58.74 USD</b>	<b>884.91 USD</b>
		Includes upfront cost

### Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon EC2	No group applied	US East (N. Virginia)	0.00 USD	7.59 USD
<b>Description:</b> Instancias disponibles <b>Config summary:</b> Tenancy (Shared Instances), Operating system (Linux), DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month), Workload (Consistent, Number of instances: 1), Snapshot Frequency (No snapshot storage), Enable monitoring (disabled), Advance EC2 instance (t3.micro), Pricing strategy (On-Demand)				
Elastic Load Balancing	No group applied	US East (N. Virginia)	0.00 USD	22.27 USD
<b>Description:</b> ELB <b>Config summary:</b> Number of Application Load Balancers (1)				
Amazon Simple Storage Service (S3)	No group applied	US East (N. Virginia)	0.00 USD	0.23 USD
<b>Description:</b> Bucket para fotos de perfil <b>Config summary:</b> S3 Standard storage (10 GB per month), S3 Standard Average Object Size (16 MB)				

Fig. 55 Ventana de detalle sobre la estimación de costos de los servicios de AWS requeridos (parte 2).

24/2/23, 23:17

AWS Pricing Calculator

<b>Amazon DynamoDB</b>	No group applied	US East (N. Virginia)	180.00 USD	28.64 USD
<b>Description:</b> Base de datos DynamoDB <b>Config summary:</b> Table class (Standard), Average item size (all attributes) (1 KB), Write reserved capacity term (1 year), Read reserved capacity term (1 year), Data storage size (10 GB)				
<b>Amazon Simple Email Service (SES)</b>	No group applied	US East (N. Virginia)	0.00 USD	0.01 USD
<b>Description:</b> Notificaciones por SES <b>Config summary:</b> Email messages sent from EC2 (100 per month)				

Acknowledgement

AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. [Learn more](#)

Fig. 56 Ventana de detalle sobre la estimación de costos de los servicios de AWS requeridos (parte 1).

## CONCLUSIONES

Al momento de proponer una solución tecnológica a un posible o potencial cliente, siempre es primeramente importante entender qué realmente requiere y enseguida estimar qué tanto puede llegar a escalar dicha aplicación así para tener una estrategia en dado caso de que la aplicación tenga un crecimiento exponencial o mucho mayor a como se tenía previsto, en este caso la tecnología que nos podrá asegurar una aplicación auto escalable, segura, durable y disponible fue proponer el servicio de Elastic Beanstalk en conjunto con una Elastic Load Balancer principalmente, de acuerdo las características solicitadas.

Con este proyecto pudimos aprender desde la forma en cómo se crean y configuran los servicios de AWS necesarios para dicho proyecto, así como la creación y manejo de permisos para otros colaboradores, esto cuando se trabaja en equipo lo cual fue algo nuevo pero muy interesante, con esto nos podemos dar cuenta que realmente es relativamente fácil y rápido aunque siempre se requiere tiempo al principio. Con dicha experiencia para el desarrollo de aplicaciones web escalables, es necesario un conjunto de buenas prácticas como comunicación y gestión de tareas.

## BIBLIOGRAFÍA





- [1] *Website & Web App Deployment - AWS Elastic Beanstalk - AWS.* (s.f.). Amazon Web Services, Inc. <https://aws.amazon.com/elasticbeanstalk/>
- [2] *AWS Pricing Calculator.* (s.f.). AWS Pricing Calculator. <https://calculator.aws/#/addService>
- [3] *AWS SDK for JavaScript.* (s.f.). Amazon Web Services, Inc. <https://aws.amazon.com/sdk-for-javascript/>
- [4] *AWS | Almacenamiento de datos seguro en la nube (S3).* (s.f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/s3/>
- [5] *Amazon DynamoDB.* (s.f.). Amazon Web Services, Inc. <https://aws.amazon.com/pm/dynamodb/>
- [6] *Plataforma de emails | Correo web | Amazon Simple Email Service (SES).* (s.f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/ses/>
- [7] *Administración de identidades | IAM | AWS.* (s.f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/iam/>