Examen José Dan Aguilar Luna

En este documentado se dará una solución digital para el siguiente caso.

Caso:

La empresa “Automotriz SPS” no tiene el control de los cambios que realizan los usuarios de negocio a los parámetros establecidos para las reglas de descuento, por lo cual, requiere tener el rastreo de dichas actividades. El sistema de administración de reglas requiere un servicio de bitácora en el cual se pueda enviar dicha información, dicha solución se encuentra dentro de la red corporativa.

Este será resuelto con las siguientes herramientas de trabajo:

-Elastic search (La herramienta será Kibana Instancia de base de datos que nos permitirá almacenar la información de manera eficiente y centralizada)

-Api desarrollo en C# (Desarrollar la solución de manejo de la base de datos y la interacción entre el usuario y la aplicación)

-Oracle SOA suite (esta será nuestra infraestructura que nos permitirá tener nuestra solución en un modelo de capas de servicios web).

Estas herramientas nos darán la facilidad de mantener un control de usuarios y de control de cambios de descuentos como lo marca el caso anterior.

Empezamos definiendo los campos de los datos que requerimos para el manejo de información

Nuestras tablas de información serán constituidas de la siguiente manera:

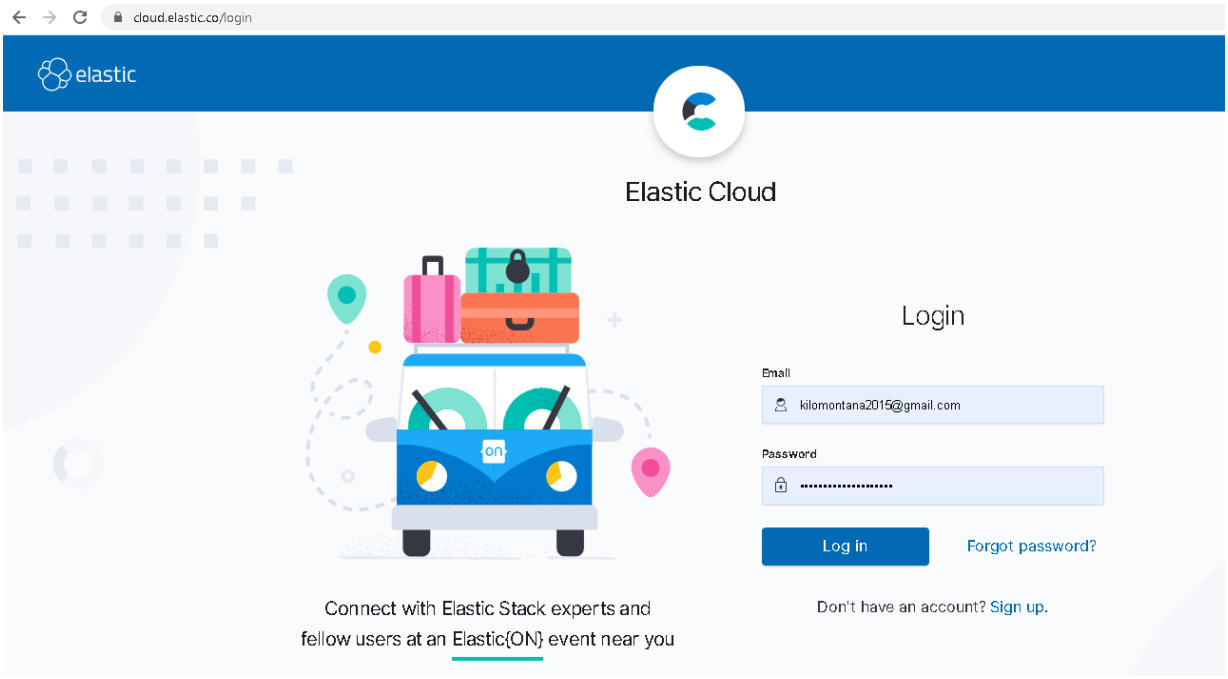
Tabla de productos	
identificador	valor
Id Producto	Int
Nombre	String
Precio	Float
descuento	float

Tabla de bitácoras	
identificador	valor
Id	Int
usuario	String
Fecha Modificación	Date
Modificado producto	String
Notas	Arreglo de notas

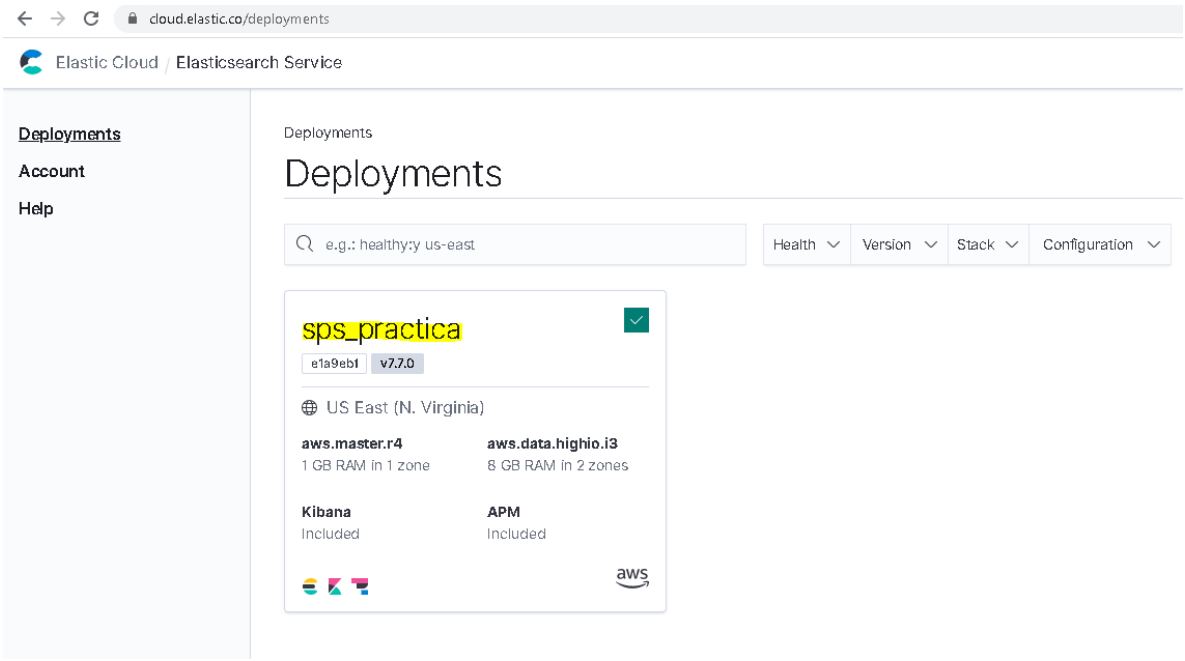
Cada uno de estos campos tendrá lugar en el control de cambios de cada producto.

Empezaremos definiendo los índices de cada tabla en Kibana de la siguiente maneja.

Primero entramos en la plataforma de kibana esto lo haremos a través de Elastic search, por tanto primero hay que entrar a Elastic Cloud:

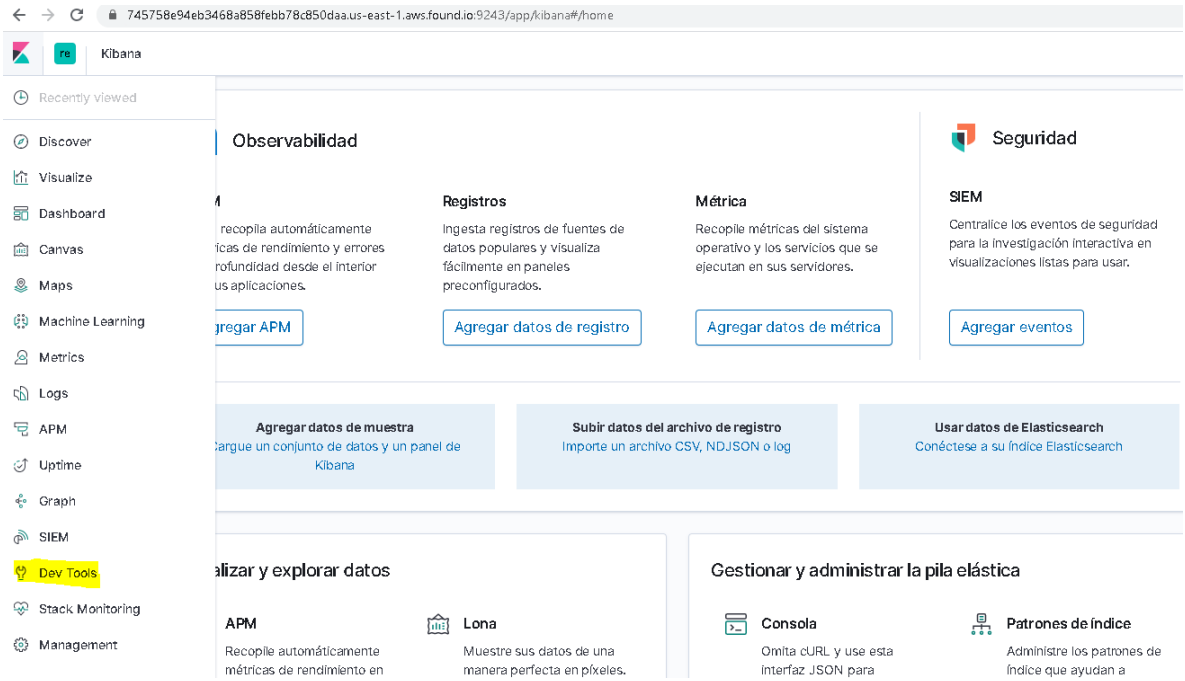


Una vez dentro buscamos nuestras instancias en Kibana (si no existe una instancia en kibana hay que crearla):



Al ingresar a kibana nos pedirán el usuario y la contraseña generada al momento de crear nuestra instancia:

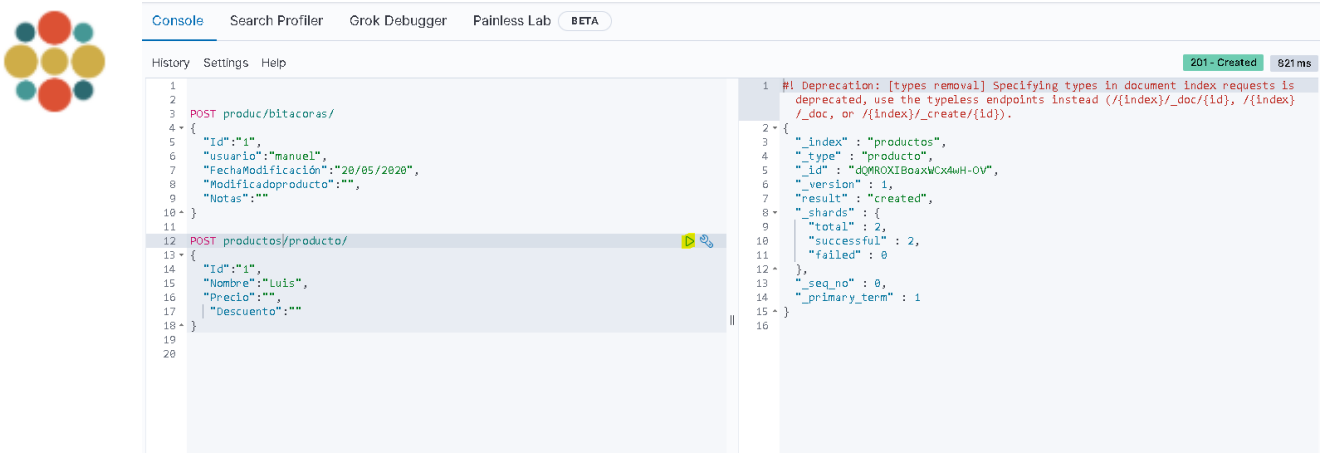
Una vez dentro veremos un panel de control del lado izquierdo hay que buscar Dev Tools que nos permitirá empezar a crear nuestros índices:



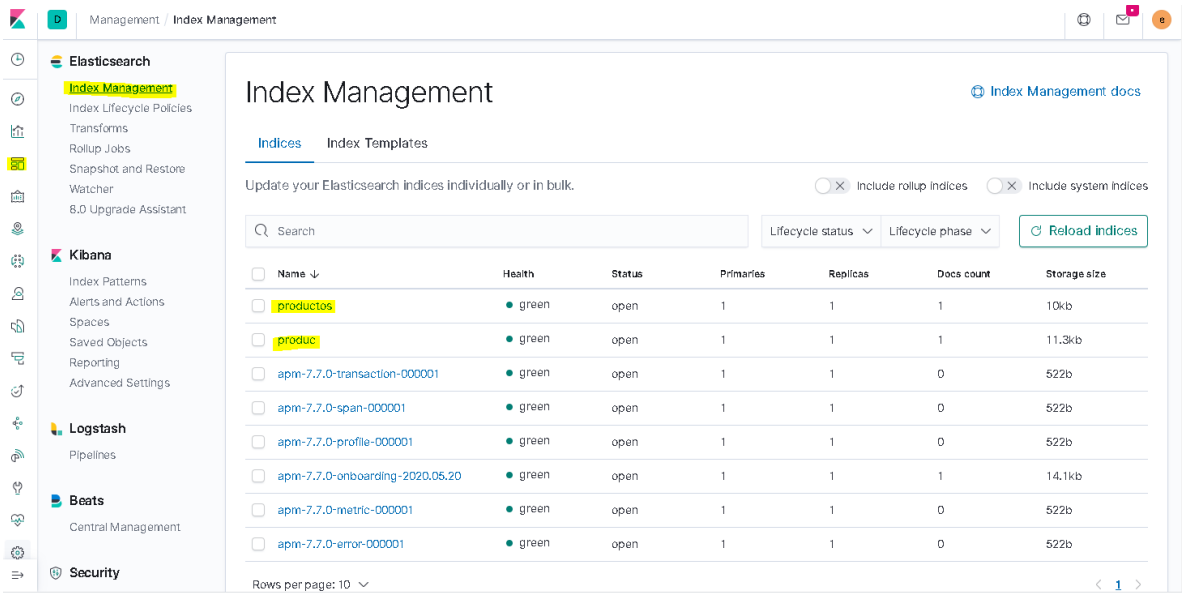
En esta parte empezaremos creando nuestros índices y las tablas con lo anterior visto de la siguiente manera:

```
POST produc/bitacoras/
{
  "Id":"1",
  "usuario":"manuel",
  "FechaModificación":"20/05/2020",
  "Modificadoproducto":"",
  "Notas":""
}

POST productos/producto/
{
  "Id":"1",
  "Nombre":"Luis",
  "Precio":"",
   "Descuento":""
}
```



Accionando el botón verde señalado en la anterior imagen nos permitirá tener instanciadas nuestro índice de información y podremos ver la opción de manager nuestra informacion de la siguiente manera:



Para verificar que nuestra informacion este en Kibana colocamos las siguientes sentencias de en la parte de Dev Tool KQL que es perteneciente a kibana para la búsqueda de información:

```
GET /produc/_search?q=*
GET /productos/_search?q=*
```

Esto nos permite tener visualización de la informacion en los índices y que nos lo presenta en un formato Json.

Ya teniendo nuestras instancias de índices podremos empezar a desarrollar nuestra API Rest para la comunicación es nuestra información.

El manejo de informacion con las Apis nos facilita la funciones de comunicación entre los diferentes sistemas por tanto tenemos la siguiente solución desarrollada en c#.

El sistema se compone por 2 modelos de objetos que nos permitirán tener el control de los objetos que se han de enviar, además de contar con 4 métodos que serán los controladores principales de nuestras aplicaciones consumidas desde la web.

Nuestra aplicación será probada desde Postman que nos permitirá enviar objetos en Json y realizar peticiones a nuestra aplicación.

Nuestros modelos son:

```
namespace ApiSolucionDescunetos.Controllers
{
    6 referencias
    public class logUsuarios
    {
        4 referencias
        public int Id { get; set; }
        3 referencias
        public string usuario { get; set; }

        3 referencias
        public DateTime FechaModificacion { get; set; }
        0 referencias
        public string Modificadoproducto { get; set; }

        3 referencias
        public string[] Notas { get; set; }

    }
}
```

Este es nuestro modelo para guardar nuestras bitácoras

```
namespace ApiSolucionDescunetos.Modelos
{
    8 referencias
    public class Productos
    {
        3 referencias
        public int IdProducto { get; set; }
        4 referencias
        public string nombreProducto{get;set;}
        3 referencias
        public double precio { get; set; }
        4 referencias
        public double descuento { get; set; }
    }
}
```

Este será nuestro modelo para controlar nuestros productos y revisar cual es el precio y descuento de cada uno.

Nuestros métodos para nuestra Api serán:

```
[HttpGet]
0 referencias
public IActionResult Get()
{
    return Ok(new object[] {
        new { Mensaje= "Bienvenido ",Metodos="GetProducto(string id) \n"+"GetBitacora(string id) \n"+"AgregarProducto(Productos
          nuevoCliente)\n"+"ModificarDescuento(string producto, string descuento, string usuario)\n"},
    });

}
```

Este método solo envía un Json con los métodos de nuestra api.

```
[HttpGet("{id}")]
0 referencias
public IActionResult GetProducto(string id)
{
    FnControldescuentos rpCli = new FnControldescuentos();

    var cliRet = rpCli.ObtenerProductos(id);


    if (cliRet == null)
    {
        var nf = NotFound("El usuario " + id.ToString() + " no existe.");
        return nf;
    }

    return Ok(cliRet);
}
```

Este método busca los productos por el Id asignado a cada producto

```
[HttpGet("{id}")]
0 referencias
public IActionResult GetBitacora(string id)
{
    FnControldescuentos rpCli = new FnControldescuentos();

    var cliRet = rpCli.ObtenerProductos(id);


    if (cliRet == null)
    {
        var nf = NotFound("El producto " + id.ToString() + " no existe.");
        return nf;
    }

    return Ok(cliRet);
}
```

Este metodo me permite buscar las bitacoras de los usuarios por su Id

```
[HttpPost("agregar")]
1 referencia
public IActionResult AgregarProducto(Productos nuevoCliente)
{
    try
    {
        FnControldescuentos rpCli = new FnControldescuentos();
        rpCli.AgregarProducto(nuevoCliente);
        return CreatedAtAction(nameof(AgregarProducto), nuevoCliente);
    }
    catch (Exception ex) { return BadRequest("Fallo actualización" + ex); }
}
```

Este método nos permite agregar nuevos productos, este solo puede invocarse desde Postman.

```
[HttpPost("{producto},{descuento},{usuario}")]
1 referencia
public IActionResult ModificarDescuento(string producto, string descuento, string usuario)
{
    try
    {
        FnControldescuentos rpCli = new FnControldescuentos();
        rpCli.ModificarDescuentoProdcuto(Convert.ToInt32(producto), Convert.ToDouble(descuento), Convert.ToInt32(usuario));
        return CreatedAtAction(nameof(ModificarDescuento), producto, descuento, usuario);
    }
    catch (Exception ex){ return BadRequest("Fallo actualización" + ex); }
}
```

Este método nos permite actualizar los descuentos de cada producto enviándole los parámetros producto, descuento y usuario.

Este método nos garantizara que toda modificación quede registrada en los índices de bitácora.

Nuestro Api seria: https://localhost:44322/ELK/ e invocaremos a los diferentes métodos que tenemos en nuestra solución:

Recuperar la informacion de productos:
https://localhost:44322/ELK/GetBitacora/{Idusuario}

Recuperar la informacion de bitácoras:
https://localhost:44322/ELK/GetProducto/{IdProducto}

Recuperar la informacion de productos:
https://localhost:44322/ELK/AgregarProductos/{Productonuevo}

Recuperar la informacion de productos:
https://localhost:44322/ELK/ModificarDescuento/{Idproducto},{descuento},{Idusuario}

Nota: No fue posible hacer captura de pantalla porque tuve un problema al correr la solución en mi computadora, una disculpa de ante mano sin embargo el código esta en mi repositorio.

https://github.com/JoseDanAL/Solucion-SPS-Examen

Por ultimo montamos nuestra solución en nuestro ambiente de SOA para tener nuestro ciclo de servicios completos y que nos permite tener un control de nuestro servicio y de nuestra base de datos.