



## EVALUACIÓN FULLSTACK

### PL/SQL

#### ENTREGA

Si construyo múltiples scripts, estos deben estar organizados y numerados en el orden que deben ser ejecutados, con el objetivo de facilitar la replicación de los objetos de BD en el ambiente de calificación.

Debe enviar la ruta del repositorio público utilizado (GitHub, GitLab, Bitbucket, etc.) al correo electrónico que le suministro esta prueba.

Las pruebas que no sean enviadas dentro del tiempo establecido o cuya ruta del repositorio no sea accesible, no serán calificadas.



Tiempo: 1 Día

## REGLAS

### Reglas

- Debe utilizar **Oracle 11gR2 o superior**
- Definir los índices que considere necesarios.
- No utilizar **cursores implícitos**.
- Todos los procedimientos, funcionalidades, records y tablas PLSQL que se construyan deben estar contenidas o pertenecer a un **paquete** de base de datos.
- Las sentencias **DML** y **DDL** utilizadas deben ser consignadas en un archivo con extensión .sql.

## HABILIDADES

### SQL



### PL/SQL



### Best Practices



### GIT



## CONTEXTO

### Problemática

En el creciente mundo empresarial y comercial, las empresas son los nodos medulares de la economía, por eso es imprescindible lograr disponer de los datos suficientes para analizar el patrimonio fluctuante del mercado, razón por la cual el gremio de comercio desea tener a su disposición una herramienta que les permita conocer de forma rápida y centralizada la información de los comerciantes y sus respectivos establecimientos, con el objetivo de analizar el mercado y tomar decisiones orientadas al progreso colectivo.

En este punto es donde entramos nosotros, para este lograr esta gran hazaña nos han solicitado construir una aplicación para condensar la información de los comerciantes y establecimientos con el objetivo de apoyar los procesos operativos esenciales de la agremiación nacional de comercio.



# RETO 01

## Modelo de Datos

Construir un modelo de datos donde se pueda organizar y almacenar la siguiente información:

- **Usuario:** Nombre, Correo Electrónico, Contraseña y Rol (**Administrador o Auxiliar de Registro**)
- **Comerciante:** Nombre o razón social, Municipio, Teléfono (**Opcional**), Correo Electrónico (**Opcional**), Fecha de Registro y Estado (**Activo o Inactivo**)
- **Establecimiento:** Nombre del Establecimiento, Ingresos (debe permitir valores con dos decimales) y Número de Empleados y el Comerciante Dueño del Establecimiento

### Notas:

- Debe normalizar el modelo, si y solo si, lo considera necesario
- Las entidades **Comerciante** y **Establecimiento** deben incluir los campos de auditoría (**Fecha de actualización y Usuario**)

# RETO 02

## Secuencias, Identificadores y Auditoría

- El identificador único de cada tabla debe ser generado por una **secuencia** de Base de Datos y cada secuencia de Base de Datos debe ser gestionada por un **trigger**.
- Los campos de auditoría (Fecha de actualización y Usuario) de cada entidad deben ser actualizados mediante un **trigger** por cada sentencia DML de inserción y actualización que se realice sobre la entidad respectiva

# RETO 03

## Datos Semilla

Generar los datos semilla para las entidades de **Usuarios**, **Comerciantes** y **Establecimientos**, respetando las siguientes cantidades:

**Usuarios:** 2 registros (uno por cada rol)

**Comerciantes:** 5 registros

**Establecimientos:** 10 registros

### Notas:

- Procure que sean datos aleatorios y diferentes para cada registro
- La cantidad de establecimientos por comerciante debe ser aleatoria



# RETO 04

## Reporte de Comerciantes

Construir una función que retorne un **cursor referenciado o por referencia** con la información de los comerciantes **activos**, bajo las siguientes condiciones:

- La estructura del **cursor referenciado o por referencia** debe contener el Nombre o razón social, Municipio, Teléfono, Correo Electrónico, Fecha de Registro, Estado, Cantidad de Establecimientos, Total Ingresos y Cantidad de Empleados
- Los campos **Cantidad de Establecimientos**, **Total Ingresos** y **Cantidad de Empleados** deben ser calculados en base a los establecimientos asociados al comerciante
- Los **comerciantes** deben estar ordenados de forma **descendente** por la **cantidad de establecimientos** que tengan asociados
- Puede construir las funciones auxiliares que considere necesarias



## REGLAS

### Reglas

- Debe utilizar **Java 11 o superior**
- Debe utilizar **Spring Boot**
- Debe construir mínimo **una prueba unitaria** que considere necesaria utilizando **JUnit**
- Debe implementar **Swagger** o compartir la colección de **Postman** utilizada
- Utilizar arquitectura limpia y principios **SOLID**
- Se debe construir el **Docker File** para implementar el despliegue contenerizado utilizando **Docker**

## HABILIDADES

### Java



### Spring Boot



### JUnit



### Best Practices



### GIT



## EVALUACIÓN FULLSTACK SPRING BOOT

### ENTREGA

Debe enviar la ruta del repositorio público utilizado (GitHub, GitLab, Bitbucket, etc.) al correo electrónico que le suministro esta prueba.

Las pruebas que no sean enviadas dentro del tiempo establecido o cuya ruta del repositorio no sea accesible, no serán calificadas.



Tiempo: 1 Día

## CONTEXTO

### Problemática

En el creciente mundo empresarial y comercial, las empresas son los nodos medulares de la economía, por eso es imprescindible lograr disponer de los datos suficientes para analizar el patrimonio fluctuante del mercado, razón por la cual el gremio de comercio desea tener a su disposición una herramienta que les permita conocer de forma rápida y centralizada la información de los comerciantes y sus respectivos establecimientos, con el objetivo de analizar el mercado y tomar decisiones orientadas al progreso colectivo.

En este punto es donde entramos nosotros, para este lograr esta gran hazaña nos han solicitado construir una aplicación para condensar la información de los comerciantes y establecimientos con el objetivo de apoyar los procesos operativos esenciales de la agremiación nacional de comercio.



# RETO 05

## Web API - Seguridad

- Crear un **endpoint** para realizar la **autenticación (login)** del usuario, recibiendo en el cuerpo de la petición los siguientes datos:
  - Correo Electrónico
  - Contraseña
- Implementar **JWT**, de tal forma que al realizar el proceso de **autenticación (login)** se genere un **token** cuyo tiempo de expiración sea de **1 hora**.
- Implementar **Autorización** por roles (**Administrador** y **Auxiliar de Registro**) para controlar el acceso a cada endpoint del API.
- Se debe utilizar el mapeo de entidades estándar o un ORM (**Hibernate** o **Spring Boot JPA**)

### Notas:

- El **endpoint** de **login** debe ser **público**, es decir, no debe solicitar un **token JWT** para su ejecución
- Debe implementar una política de **CORS** para asegurar el consumo controlado de las APIs

# RETO 06

## Web API - Listas de Valores

- Crear un **endpoint** para retornar una lista de valores para el campo **Municipio**, la cual será utilizada en el **CRUD** de la entidad - **Comerciantes**
- Implementar **Cache en Memoria** para evitar accesos adicionales a la base de datos **[Opcional]**
- Se debe utilizar el mapeo de entidades estándar o un ORM (**Hibernate** o **Spring Boot JPA**)

### Notas:

- Todos estos **endpoints** deben ser **privados**, es decir, deben solicitar un **token JWT** para su ejecución
- Implementar estandarización de la respuesta HTTP de los endpoints



# RETO 07

## Web API - CRUD - Comerciante

- Crear un **endpoint** para cada una de las operaciones **CRUD** de la entidad - **Comerciante**
  - Consulta Paginada (5 registros por página por defecto)
  - Consultar por Id
  - Crear
  - Actualizar
  - Eliminar (Solo el rol de **administrador** puede visualizar y ejecutar esta acción)
- Las operaciones de **crear** y **actualizar** deben solicitar los campos: Nombre o razón social, Municipio, Teléfono (**Opcional**), Correo Electrónico (**Opcional**), Fecha de Registro y Estado
- La consulta **paginada** debe permitir filtrar por **Nombre o razón social, Fecha de Registro y Estado**
- Las operaciones de **crear** y **actualizar** deben modificar los campos de auditoría (**Fecha de actualización** y **Usuario**) de la entidad, pero sin solicitar esta información por **body** de la petición
- Crear un **endpoint** para modificar el estado del comerciante utilizando el método HTTP **PATCH**
- Se debe utilizar el mapeo de entidades estándar o un ORM (**Hibernate** o **Spring Boot JPA**)

### Notas:

- Implementar validaciones respectivas (tipos de datos, obligatoriedad y formato de correo electrónico) para verificar la información suministrada en los **endpoints** de **creación** y **actualización**
- Todos estos **endpoints** deben ser **privados**, es decir, deben solicitar un **token JWT** para su ejecución
- Implementar estandarización de la respuesta HTTP de los **endpoints**

# RETO 08

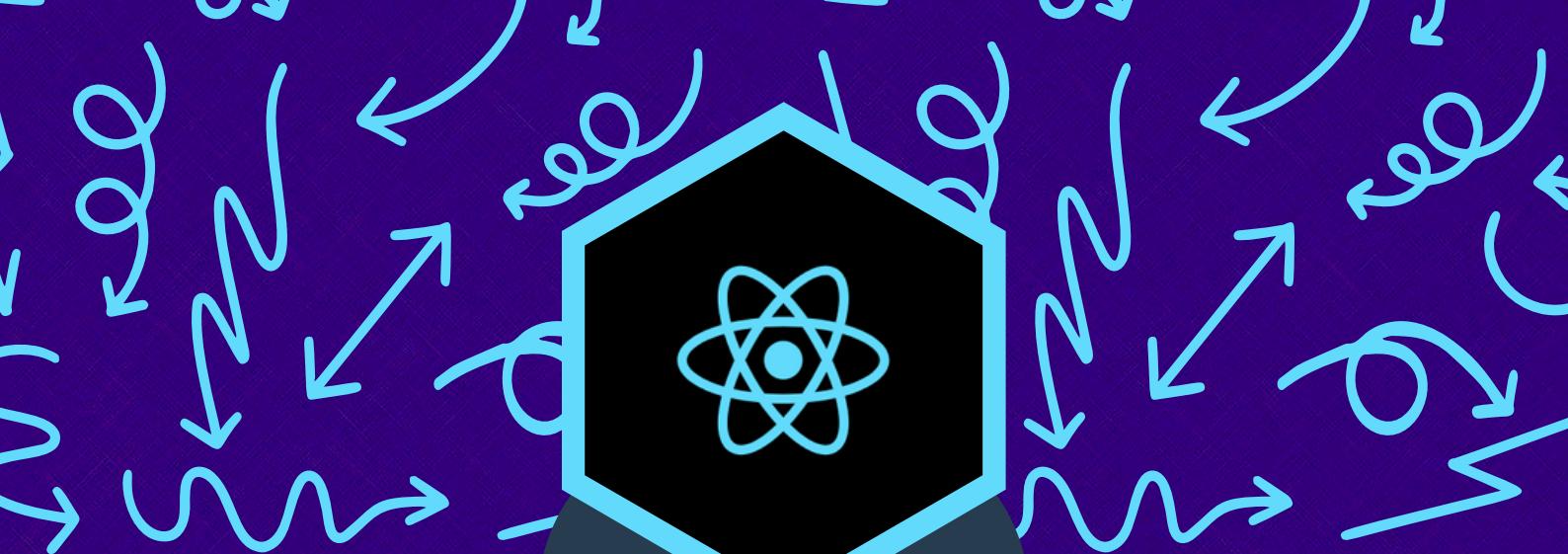
## Web API - Reporte Comerciantes

Crear un **endpoint** que permita generar un **archivo plano (.csv)** con la información de los comerciantes **activos**, bajo las siguientes condiciones:

- La estructura del **archivo plano (.csv)** debe contener el Nombre o razón social, Municipio, Teléfono, Correo Electrónico, Fecha de Registro, Estado, Cantidad de Establecimientos, Total Ingresos y Cantidad de Empleados
- Los campos **Cantidad de Establecimientos**, **Total Ingresos** y **Cantidad de Empleados** deben ser calculados en base a los establecimientos asociados al comerciante
- Para la generación del archivo plano se debe utilizar como separador el carácter **pipe ( | )**, es decir, la barra vertical
- Se deben utilizar o consumir la función construida en el **Reto 4** para este propósito

### Notas:

- El **endpoint** debe ser **privado**, es decir, debe solicitar un **token JWT** para su ejecución
- Solo debe ser accesible a usuarios con el rol - **Administrador**
- Implementar estandarización de la respuesta HTTP del **endpoint**



## REGLAS

### Reglas

- Debe utilizar **React 18 o superior**
- Debe utilizar un sistema de gestión para el estado global, la elección se deja a consideración del desarrollador.
- Debe construir mínimo **una prueba unitaria** que considere necesaria utilizando **Jest**
- Tener en cuenta las recomendaciones del **OWASP TOP 10** para la seguridad.

## HABILIDADES

### HTML Y CSS



### TypeScript



### React



### Best Practices



### GIT



## EVALUACIÓN FULLSTACK REACT

### ENTREGA

Se debe entregar adicional al repositorio con el código fuente frontend, un compilado para producción comprimido, por el medio que el desarrollador considere (link de descarga, adjunto al correo, etc).

Debe enviar la ruta del repositorio público utilizado (GitHub, GitLab, Bitbucket, etc.) al correo electrónico que le suministro esta prueba.

Las pruebas que no sean enviadas dentro del tiempo establecido o cuya ruta del repositorio no sea accesible, no serán calificadas.



Tiempo: 1 Día

## CONTEXTO

### Problemática

En el creciente mundo empresarial y comercial, las empresas son los nodos medulares de la economía, por eso es imprescindible lograr disponer de los datos suficientes para analizar el patrimonio fluctuante del mercado, razón por la cual el gremio de comercio desea tener a su disposición una herramienta que les permita conocer de forma rápida y centralizada la información de los comerciantes y sus respectivos establecimientos, con el objetivo de analizar el mercado y tomar decisiones orientadas al progreso colectivo.

En este punto es donde entramos nosotros, para este lograr esta gran hazaña nos han solicitado construir una aplicación para condensar la información de los comerciantes y establecimientos con el objetivo de apoyar los procesos operativos esenciales de la agremiación nacional de comercio.



# RETO 09

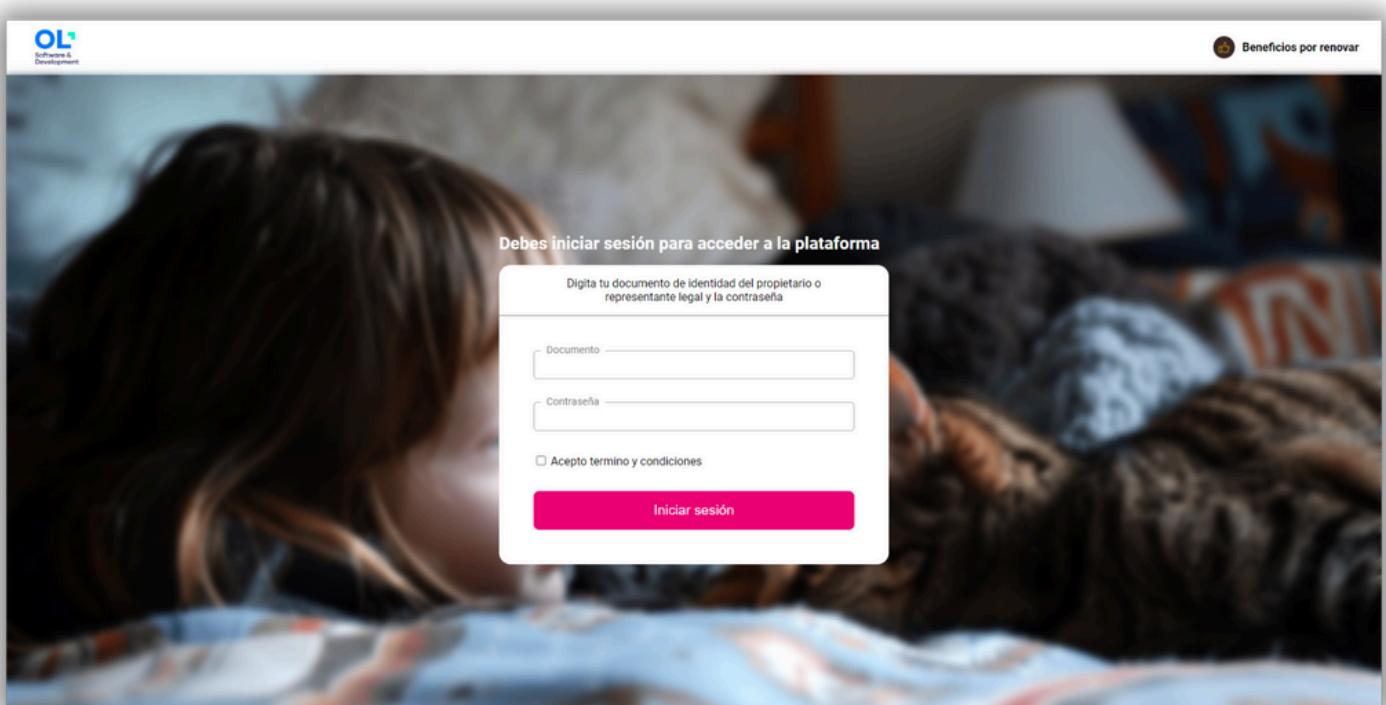
## Página - Login

Crear una **página o interfaz de usuario** que permita realizar el proceso de autenticación y sirva como punto de entrada a la aplicación, bajo las siguientes condiciones:

- Debe tener la siguiente estructura o campos **obligatorios**:
  - Correo Electrónico (Usuario)
  - Contraseña
  - Checkbox para aceptar los términos y condiciones antes de poder iniciar sesión
- Una vez que el usuario inicie sesión, en la parte superior de la página debe existir un encabezado donde se visualice: **Nombre del Usuario y Rol (Administrador o Auxiliar de Registro)**
- El proceso de autenticación se debe realizar por medio de **JWT** sin enviar los datos en bruto, es decir, el usuario y contraseña no se deben poder capturar desde el debug de los navegadores
- Se debe utilizar o consumir el **endpoint** construido en el **Reto 5** para este propósito

### Notas:

- El **diseño y arquitectura de la información** debe respetar el **prototipo** suministrado a continuación, **Excepto** la imagen de fondo (Niña & Gato), esa se deja a criterio del desarrollador, la que considera más pertinente.





# RETO 10

## Página - Home

Crear una **página o interfaz de usuario** que permita visualizar los **comerciantes** registrados en la aplicación, bajo las siguientes condiciones:

- Debe disponer de una tabla con la siguiente estructura o columnas: Nombre o Razón Social, Teléfono, Correo Electrónico, Fecha Registro, Cantidad de Establecimientos, Estado y Acciones
- La tabla debe tener un **paginado** que permita seleccionar la cantidad de ítems por página, la cantidad permitida de registros por pagina deben ser **5, 10 y 15**.
- En la columna de acciones, debe haber botones que permitan
  - **Editar registro** (Redirigir al usuario al formulario de edición detallado en el **Reto 11**)
  - **Activar/Inactivar registro** (Se debe invocar o consumir el **endpoint** construido en el **Reto 7**)
  - **Eliminar registro** (Solo el rol de **administrador** puede visualizar y ejecutar esta acción)
- Disponer de un botón que permita **Descargar Reporte en CSV** que debe invocar o consumir el **endpoint** construido en el **Reto 8** para poder obtener los datos necesarios para construir el archivo en el formato indicado. Solo el rol de **administrador** puede visualizar y ejecutar esta acción
- Disponer de un botón que permita **crear** un nuevo comerciante, esto debe llevar al usuario al formulario de creación detallado en el **Reto 11**
- Disponer de una opción que permita **cerrar sesión** (la ubicación de la opción es decisión del desarrollador)
- Se deben utilizar o consumir los **endpoints** construidos en el **Reto 7** y **Reto 8** que considere necesarios para este propósito

### Notas:

- El **diseño y arquitectura de la información** debe respetar el **prototipo** suministrado a continuación

OL Software & Development

Lista Formulario | Crear Formulario | Beneficios por renovar | Bienvenido! John Doe Administrador

### Listado de Formularios Creados

Razón Social	Teléfono	Correo Electrónico	Fecha Registro	No. Establecimientos	Estado	Acciones
Empresa 1	+57 123 456 7890	empresa1@example.com	2024-06-01	5	Activo	
Empresa 2	+57 098 765 4321	empresa2@example.com	2024-06-02	3	Inactivo	
Empresa 3	+57 321 654 9870	empresa3@example.com	2024-06-03	4	Activo	
Empresa 4	+57 567 890 1234	empresa4@example.com	2024-06-04	2	Activo	
Empresa 5	+57 234 567 8901	empresa5@example.com	2024-06-05	6	Inactivo	
Empresa 6	+57 345 678 9012	empresa6@example.com	2024-06-06	7	Activo	
Empresa 7	+57 456 789 0123	empresa7@example.com	2024-06-07	8	Inactivo	
Empresa 8	+57 567 890 1234	empresa8@example.com	2024-06-08	9	Activo	
Empresa 9	+57 678 901 2345	empresa9@example.com	2024-06-09	10	Activo	
Empresa 10	+57 789 012 3456	empresa10@example.com	2024-06-10	11	Activo	

Items:

Prueba Técnica De Uso Exclusivo de OLSoftware S.A.



# RETO 11

## Página - Formulario (Creación y Actualización)

Crear una **página o interfaz de usuario** que permita **crear y/o actualizar** la información de los comerciantes y sus respectivos establecimientos, bajo las siguientes condiciones:

- Debe disponer de una tabla con la siguiente estructura o columnas:
  - Nombre o Razón Social
  - Municipio o Ciudad (**Selector o Lista de Valores**)
  - Teléfono (**Opcional**)
  - Correo Electrónico (**Opcional**)
  - Fecha de Registro (**Selector de fecha**)
  - ¿Posee establecimientos? (**CheckBox**)
- Solo al **editar un comerciante**, dentro de la pantalla del formulario, en el **footer**, se debe poder visualizar la **sumatoria** de los **ingresos** y la **sumatoria** de la **cantidad de empleados** de todos los establecimientos (Previamente cargados en **Reto 3**) que se encuentren asociados al comerciante seleccionado de acuerdo al **diseño o prototipo**; dado que la gestión de los establecimientos **NO disponen de API's ni interfaz**, esta información debe ser cargada por BD a través de la data semilla solicitada en el **Reto 3**
- Se deben utilizar o consumir los **endpoints** construidos en el **Reto 6** y **Reto 7** que considere necesarios

### Notas:

- Todos los campos del formulario deben poseer su correspondiente validador de tipos de dato (número, fecha, email, etc.), longitud y obligatoriedad para que el usuario pueda ingresar datos con formato correcto y dar un feedback al usuario en caso de generarse un error; el mecanismo de feedback se deja libre para que el desarrollador lo escoja, puede ser toats, modal, texto de apoyo en el input, entre otros.
- El **diseño y arquitectura de la información** debe respetar el **prototipo** suministrado a continuación:

OL Software & Development

Bienvenido! John Doe Administrador

Empresa 1

Datos Generales

Razón Social \*

Departamento \* Amazonas

Ciudad \* Alcalá

Teléfono

Correo electrónico

Fecha de registro \*

Posee establecimientos?

Total Ingresos Formulario: \$100.000.000.000

Cantidad de empleados: 999

Si ya ingresaste todos los datos, crea tu formulario aquí

Enviar Formulario