Jenifer Paola Rodriguez Villamizar

Documentación prueba técnica Mercado Libre

El programa se desarrolló en GO, a continuación, se explica los detalles relevantes del desarrollo:

Funcionalidades y objetivos de los paquetes:

Package service : proporciona servicios para interactuar con la base de datos y realizar operaciones como persistir datos, escanear la base de datos, obtener la clasificación de la base de datos y administrar palabras.

- **Estructuras en el paquete:**

Existen dos estructuras principales en este paquete:

- ✓ **Database:** Representa el servicio de base de datos y proporciona métodos para interactuar con la base de datos.
- ✓ **IDatabase:** Define la interfaz para el servicio de base de datos.

- **Funciones en el paquete:**

Funciones definidas en el paquete:

- ✓ **NewDatabase:** Crea una nueva instancia del servicio de base de datos.
- ✓ **Persist:** Almacena las credenciales de una base de datos en la tabla *`credentials`* de la base de datos `*databasecredentials*`.
  - → Recibe un objeto `*PersistRequest*` y devuelve el ID del último registro insertado.
- ✓ **Scan:** Escanea la base de datos, clasifica la información y la persiste en las tablas correspondientes en la base de datos `*databaseclasification*`.
  - → Recibe el ID de la base de datos a escanear.
- ✓ **GetClassification:** Obtiene la clasificación de la base de datos a partir del ID proporcionado. Devuelve un objeto `*EsquemasData*` con la información clasificada.
- ✓ **AddNewWord:** Agrega una nueva palabra a la tabla `*privateWord*` en la base de datos `privateData`.
  - → Recibe la palabra a agregar y devuelve el ID del último registro insertado.
- ✓ **GetWordList:** Obtiene la lista de palabras almacenadas en la tabla `*privateWord*` de la base de datos `*privateData*`.
  - → Devuelve un objeto `WordListResponse` con la lista de palabras clave a validar para realizar la clasificación de las columnas.
- ✓ **clasificarColumna:** Función auxiliar que clasifica una columna de acuerdo a una lista de palabras.
  - → Recibe la lista de palabras y el nombre de la columna a clasificar. Devuelve la clasificación de la columna.

→ Toma la lista de palabras y un nombre de columna como parámetros. Itera sobre la lista de palabras y compara cada palabra con el nombre de la columna, buscando coincidencias. Si una palabra coincide total o parcialmente con el nombre de la columna, se considera una clasificación válida por lo cual se agrega la clasificacion a la columna.

→ La función utiliza una función auxiliar llamada `contienePalabras2` para realizar la comparación. Si se encuentra una clasificación válida, se devuelve la clasificación en mayúsculas. Si no se encuentra ninguna clasificación válida, se devuelve "N/A" para indicar que no se pudo clasificar la columna.

→ La función `contienePalabras2` verifica si el nombre de la columna contiene al menos una cantidad mínima de palabras de la lista. También utiliza otra función auxiliar llamada `contieneName` para verificar si la lista de palabras contiene la palabra "name", lo cual es una condición adicional para considerar una clasificación válida pues en caso de contener la palabra name se tendrá que contener al menos 2 palabras de la lista para considerarlo una clasificación valida, esto con el fin de que no se clasifique como LastName o firstName palabras como tableName o userName.

Nota: Este paquete utiliza el controlador de MySQL para interactuar con la base de datos.

Package controller : El paquete controller contiene los controladores que manejan las solicitudes HTTP y definen los enrutamientos de la aplicación.

- **Controlador Persist**

El controlador Persist es responsable de almacenar las credenciales de una base de datos en la base de datos.

- **Método HTTP y ruta**
- **Método:** POST
- **Ruta:** /api/v1/database

Ejemplo de entrada:

```
{
  "host": "localhost",
   "port": 3306,
    "username": "root",
    "password": "MySQLPassword2023"
}
```

Ejemplo de salida:

Salida exitosa: Status 200

```
{
    "Id": 1
}
```

Error:

```
{
  "error": "<Mensaje de error>"
}
```


- **Controlador Scan**

El controlador Scan se encarga de escanear una base de datos específica y clasificar la información encontrada.

- **Método HTTP y ruta**

→ **Método:** POST

→ **Ruta:** /api/v1/database/scan/:id

Parámetros de entrada

El controlador espera recibir el ID de la base de datos como parte de la URL.

Respuestas

El controlador no devuelve ningún cuerpo de respuesta. Solo establece el estado de la respuesta HTTP.

Código de estado: 201 (Created)

- **Controlador GetClassification**
- **Método HTTP y ruta**

→ **Método:** GET

→ **Ruta:** /api/v1/database/scan/:id

Parámetros de entrada

El controlador espera recibir el ID de la base de datos como parte de la URL.

Respuestas

El controlador devuelve una respuesta JSON que contiene la clasificación de la base de datos.

Código de estado: 200 (OK)

Cuerpo de respuesta: Objeto JSON que representa la clasificación de la base de datos.

Ejemplo:


```
{
  "databaseId": "1",
  "last_scan": "2023-07-13T17:55:57Z",
  "Esquemas": [{
    "EsquemaName": "databasecredentials",
    "Tablas": [{
      "TableName": "credentials",
      "columnas": [{
          "columnName": "dbhost",
```

```
            "tipo": "varchar",
            "Clasificacion": "N/A"
          },
          {
            "columnName": "dbpassword",
            "tipo": "varchar",
            "Clasificacion": "PASSWORD"
          },
          {
            "columnName": "dbport",
            "tipo": "int",
            "Clasificacion": "N/A"
          },
          {
            "columnName": "dbusername",
            "tipo": "varchar",
            "Clasificacion": "USERNAME"
          },
          {
            "columnName": "id",
            "tipo": "int",
            "Clasificacion": "N/A"
          }
        ]
      }]
  }]
}
```


- **Controlador GetWordList**

El controlador GetWordList obtiene la lista de palabras clave a validar.

- **Método HTTP y ruta**

→ **Método:** GET
→ **Ruta:** /api/v1/privateData/wordlist

Respuestas:

El controlador devuelve una respuesta JSON que contiene la lista de palabras clave.

Código de estado: 200 (OK)

Cuerpo de respuesta: Objeto JSON que representa la lista de palabras clave.

Ejemplo:


```
{
  "wordList": [
    "PASSWORD",
    "IP_ADDRESS",
    "LAST_NAME",
    "FIRST_NAME",
    "CREDIT_CARD_NUMBER",
```

```
    "USERNAME",
    "EMAIL_ADDRESS"
]
}
```

- **Controlador AddNewWord**

El controlador AddNewWord agrega una nueva palabra a la lista de palabras clave

- **Método HTTP y ruta**

→ **Método:** POST
→ **Ruta:** /api/v1/privateData/wordlist/:word
→

Parámetros de entrada:

El controlador espera recibir la palabra a agregar como parte de la URL.

Respuestas

El controlador devuelve una respuesta JSON que contiene el ID de la palabra agregada.

Código de estado: 200 (OK)

```
{
  "id": <ID>
}
```

Consideraciones adicionales:

- En la dirección meli_golang_gin_basic_app\postman se encuentra la colección de postman con la cual se pueden probar las dependencias
- En la direccion \meli_golang_gin_basic_app\scripts se encuentran los scrips para la creacion de las tablas