2020

Gestor Múltiple de Bases de Datos

Pablo Brañuelas

Castellano

08/06/2020

PROYECTO FIN DAM

INDICE

1. MEMORIA

a) Diseño de la aplicación

El diseño está basado en contenedores y cajas, usando un sistema de GridLayout.

La idea es que se pueda ver fácilmente las bases de datos y al clicar sobre ellas aparezcan sus tablas y vistas.

Mientras que en el lado derecho es donde se realizan las consultas y donde se muestran los resultados obtenidos y también el número de filas obtenidas.

En la parte superior hay un menú donde se pueden establecer la conexión a los diferentes tipos de bases de datos (MySQL, MariaDB, SQLITE, Firebird, SQL Server, PostgreSQL), consultar la sintaxis de cada sistema de base de datos, dar formato al texto, exportar e importar bases de datos, guardar los resultados de las consultas realizadas, cargar archivos para mostrar gráficos de datos, actualizar y vaciar los diferentes contenedores,una conexión a un Web Service escrito en Java Spring y una opción de salir de la aplicación.

Las versiones utilizadas en este proyecto son:

- Python 3.8.2.
- MySQL 8.0.19.
- MariaDB 10.4.
- SQLITE 3.
- Dremio 3.3.0.
- Firebird 3.0.
- SQL SERVER 2017.
- PostgreSQL 11.7.

b)Funcionamiento de la aplicación

La aplicación está basada en los diferentes gestores de base de datos que existen; por ejemplo DBeaver.

La aplicación te permite conectarte a distintos gestores de bases de datos como MySQL, MariaDB, SQLITE, Firebird, SQL Server, PostgreSQL, Dremio.

Algunas de las funcionalidades que incluye son:

- Exportar e importar bases de datos de cualquier gestor usado en la aplicación.
- Guardar las consultas realizadas en distintos formatos: CSV, PDF, HTML.
- Cargar archivos csv y xls para obtener gráficos a partir de la información que contienen.
- Refrescar bases de datos y tablas.
- Dar estilo a las distintas ventanas cambiado el tamaño, tipo y color de la letra además del fondo, según desee el usuario.
- Sistema de ayudas saber la sintaxis correcta a usar dependiendo del gestor que estés utilizando.
- Api de Google para saber cuántas veces se ha buscado una palabra en cada región.
- Palabras más buscadas al año.
- Web Service escrito en Java y Spring ,al cual se llama desde la interfaz escrita en Python.

c) Comunicación Rest

➢ Google Trends

Usando la api de Google Trends(pytrends) a través de un formulario escrito en Python puedo obtener las tendencias de un año deseado y las veces que se ha buscado esa palabra en cada región del mundo.
Para poder mostrarlo en una tabla convierto en DataFrame en un array del cual mediante dos bucles extraigo y los muestro en un QtableWidget.

➢ Python Spring

Creando un proyecto escrito en Java usando el framework de Spring y a través de métodos get y post paso a través de formularios escritos en Python y la librería urllib paso los parámetros y la url los cuales recibe la aplicación de Spring y realiza las operaciones de Crear ,Mostrar,Actualizar,Buscar y Eliminar Usuarios.
Para que el proyecto funcione hay que indicar e instalar una serie de dependencias en un archivo llamado "pomp.xml".

Las dependencias indicadas han sido:

a) spring-boot-starter-web.Se utiliza para desarrollar servicios web con Tomcat incorporado.

b) tomcat-embed-jasper.Es el servidor integrado de Tomcat.

c) spring-boot-starter-tomcat.Es el encargado de arrancar el servidor tomcat.

d) javax-servlet.Es la dependencia que contiene las peticiones request que se realizan al servidor a través de la Interfaz HttServletRequest.

e) spring-boot-starter-data-jpa.Es una caraccteristica de Spring Boot que sirve para conectar la aplicación de Spring con la base de Datos.

f) mysql-java-connector.Es el conector con la base de datos. En la etiqueta “<version>” indicamos la versión del controlador que vamos a usar.

➢ Dremio

- **¿Qué es Dremio**

  Es el motor que usa el Data Lake para procesar los datos.

- **¿Qué es Data Lake?**

  Es un repositorio centralizado que permite almacenar datos estructurados y no estructurados. Es una ubicación central en la que almacenar todos los datos independientemente de la fuente o el formato.

  A cada elemento del Data Lake se le asigna un identificador único.

Los beneficios que ofrece el Data Lake son:

- Centralización de datos procedente de distintas fuentes .Una vez reunidas y agrupadas se combinan y procesan utilizando big data, búsquedas y análisis.
- Permite conservar los datos sin preocuparse por la falta de espacio.

- **Funcionalidad dentro del Proyecto**

Desde la propia interfaz web de Dremio que esta en localhost:9047 el usuarios crea la conexión a la base de datos deseada. Otra de las opciones que dispone es cargar archivos XLS,CSV o JSON y realizar consultas sobre ellos.

Desde la intefaz del proyecto se puede conectar a Dremio y realizar consultas sobre la conexión creada desde la interfaz web(por ejemplo ,si te conectas a Hadoop desde la web,en la interfaz del proyecto puedes hacer consultas sobre las distintas tablas que tenga la base de datos).Otra característica que tiene Dremio es que no tiene tablas propias,sino que las tablas que crea son a partir de consultas que tu haces a la base de datos y tienen que guardar en “$scratch” y esto se te guarda en tu directorio del ordenador en la ruta “AppData\Local\Dremio\pdfs\scratch” dentro de una carpeta con el nombre de la tabla y dentro 2 archivos:

Uno con extensión .crc y otro .parquet que son los que contienen la información del resultado de la sintaxis “create table $scratch.nombretabla as consulta”.

d)Seguridad

Para poder acceder a las bases de datos y sus diferentes opciones necesitas haber iniciado sesión.

Para ello hay una ventana de inicio de sesión conectada a distintas librerías de python, para verificar el usuario y contraseña.

A nivel interno cada gestor tiene su propia forma de controlar que puede hacer y que permisos tiene cada usuario a través de roles y permisos.

2. Línea de Investigación

Para realizar el proyecto he tenido que aprender:

1. Como conectar los distintos gestores de bases de datos y que no entren en conflicto al usarlos en otros sistemas operativos como Linux.

2. Librerías de Python que permitan guardar las consultas en distintos formatos.

3. Librerías de Python para poder leer los archivos csv y xls y poder mostrar sus columnas y dibujar los gráficos.

4. Las características propias y la sintaxis de cada uno de los gestores utilizados y como realizar por línea de comandos las exportaciones y las importaciones de las bases de datos.

5. Librería para usar Google Trends con Python.

6. Como conectar y consumir un Web Service escrito en Java y Spring desde Python.

7. Distintas opciones de consultas sobre archivos csv y xls que permite Dremio a través de Azure Storage y creación de tablas en Dremio.

8. En futuras ampliaciones se puede añadir:
    a. MongoDB.
    b. Apache Cassandra.
    c. TimeScaleDB.
    d. Apache Hadoop.
    e. DB2.
    f. Oracle.
    g. Dremio Ampliado con Azure Storage.

3. Accesibilidad

Para mejorar la accesibilidad en la aplicación he añadido una opción para que cada contenedor pueda configurar el estilo que el usuario necesite.

Además de agregar dos configuraciones especificas para personas con problemas de visión y daltonismo.

Los criterios seguidos son:

- Para personas con problemas de visión según los criterios de accesibilidad tiene que usarse:

  - Alto contraste. Para ello he establecido como fondo el color negro y color de fuente el verde.

  - Tipo de fuente y tamaño de letra. No hay un criterio prestablecido.He optado por Arial y tamaño 30 pixeles.

- Para personas con daltonismo el criterio seguido es buscar características comunes que no afecten a los distintos tipos de daltonismo.

  Para ello he usado la web https://www.toptal.com/designers/colorfilter donde te filtra según tipo de daltonismo y te muestra como lo vería cada persona los colores.

  - Color de Fondo. El elegido es “Sky blue”.Ya que las personas con daltonismo si ven ese color (salvo aquellas que sufren un daltonismo acromático).

- Fuente de la letra. Como no hay un criterio preestablecido he escogido Arial 20 píxeles.

- Color de Letra. Bajo el mismo criterio del color de fondo; es decir un color que sea fácilmente visible y buscando un buen contraste, para facilitarles la visión he escogido el color negro.

4. BIBLIOGRAFÍA

- **MYSQL**

  - Conectar Python con Mysql: https://unipython.com/acceso-a-bases-de-datos-con-pyodbc/

  - Documentación:

    https://www.w3schools.com/sql/default.asp

    https://dev.mysql.com/doc/refman/8.0/en/

- **MARIADB**

  - Conectar Python con MariaDB: https://unipython.com/acceso-a-bases-de-datos-con-pyodbc/
    Lo único que cambia es el puerto Mysql escucha en el 3306 y MariaDB en el 3310

- Documentación:
  https://mariadb.com/kb/en/documentation/

- **SQLITE**

  - Conectar Python con SQLITE:
    https://pynative.com/python-sqlite/
  - Documentación:
    https://www.sqlite.org/keyword_index.html

    https://www.sqlite.org/docs.html

- **FIREBIRD**

  - Conectar python con Firebird:
    https://firebirdsql.org/file/documentation/drivers_documentation/python/fdb/getting-started.html

  - Documentación:
    - Crear bases de datos:
      https://www.firebirdsql.org/manual/es/qsg15-es-creating.html

    - Agregar usuarios :
      - https://firebird21.wordpress.com/2015/07/11/agregando-el-usuario-sysdba-en-firebird-3/

        - https://firebird21.wordpress.com/2013/04/21/agregando-modificando-y-borrando-usuarios/

    - Mostrar tablas y vistas: https://firebird21.wordpress.com/2013/03/04/nombres-de-todas-las-tablas-y-de-todas-las-vistas/

    - Estructura de una tabla: https://firebird21.wordpress.com/2013/03/04/nombres-de-todas-las-columnas-de-una-tabla/

- **SQL SERVER**

  - Conectar Python con SQL SERVER: https://docs.microsoft.com/es-es/sql/connect/python/pyodbc/step-3-proof-of-concept-connecting-to-sql-using-pyodbc?view=sql-server-ver15

  - Documentación:

    https://docs.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver15

    - Realizar copias de seguridad: https://docs.microsoft.com/es-es/sql/linux/sql-server-linux-backup-and-restore-database?view=sql-server-2017

- **POSTGRESQL**

  - Conectar Python con PostgreSQL: https://www.neoguias.com/como-conectarse-postgresql-python/

  - Documentación: https://www.postgresql.org/docs/

    - Ver Bases de datos: http://www.postgresqltutorial.com/postgresql-show-databases/
    - Mostrar tablas y vistas: http://www.postgresqltutorial.com/postgresql-show-tables/

    - Mostrar campos de una tabla: http://www.postgresqltutorial.com/postgresql-describe-table/

- **DREMIO**

  - Conectar Dremio con Python: https://docs.dremio.com/client-applications/python.html

  - Documentación: https://docs.dremio.com/

    - Crear Tablas: https://docs.dremio.com/sql-reference/sql-commands/tables.html

- Cargar archivos csv y xls y realizar consultas SQL desde la interfaz web: https://www.dremio.com/tutorials/visualizing-your-first-dataset-with-tableau-and-dremio/

➢ Que es el Data Lake y que beneficios tiene:
https://www.powerdata.es/data-lake

https://epicartsagency.com/ventajas-de-data-lake/

- **ACCESIBILIDAD**

➢ Baja Visión:
http://accesibilidadweb.dlsi.ua.es/?menu=deficit-visual-baja-vision

➢ Daltonismo:
https://cuidatuvista.com/daltonismo-problemas-vision-de-colores-rojo-verde/

➢ Web Efecto Daltonismo:
https://www.toptal.com/designers/colorfilter

➢ Mejorar Accesibilidad para daltónicos:
http://accesibilidadweb.dlsi.ua.es/?menu=deficit-visual-daltonismo