**Documentación del Proyecto: Sistema de Limpieza y Almacenamiento de Datos con MySQL**

* **Autor(es)**

Ing. Dary Alexander Guale Pilay

* **Fecha de creación**10/18/2024
* **Versión del documento**

Versión 1.0.0.0

Contenido

[Introducción: 4](#_Toc180144397)

[Propósito del documento. 4](#_Toc180144398)

[Alcance del proyecto. 4](#_Toc180144399)

[Breve descripción del sistema o producto. 4](#_Toc180144400)

[El flujo de trabajo incluye las siguientes etapas: 4](#_Toc180144401)

[1. Creación del entorno de trabajo 4](#_Toc180144402)

[2. Lectura de datos: 4](#_Toc180144403)

[3. Limpieza de datos: 4](#_Toc180144404)

[4. Concatenación archivos: 5](#_Toc180144405)

[Procesamiento vertical y horizontal: 5](#_Toc180144406)

[5. Procesamiento vertical: 5](#_Toc180144407)

[6. Procesamiento horizontal: 5](#_Toc180144408)

[7. Salida de datos: 5](#_Toc180144409)

[Requisitos: 5](#_Toc180144410)

[Requisitos funcionales y no funcionales. 5](#_Toc180144411)

[Requisitos funcionales. 5](#_Toc180144412)

[Requisitos y no funcionales. 5](#_Toc180144413)

[Especificaciones del sistema 5](#_Toc180144414)

[Arquitectura del Sistema: 5](#_Toc180144415)

[1. Capa de Ingesta 6](#_Toc180144416)

[2. Capa de procesamiento de datos (Limpieza y transformación): 6](#_Toc180144417)

[3. Capa de Almacenamiento y Exportación: 6](#_Toc180144418)

[Diagrama de arquitectura. 6](#_Toc180144419)

[Diseño del Sistema: 7](#_Toc180144420)

[Implementación: 7](#_Toc180144421)

[4. Descripción del entorno de desarrollo. 7](#_Toc180144422)

[5. Lenguaje de programación: 7](#_Toc180144423)

[6. Entorno de Desarrollo Integrado (IDE). 7](#_Toc180144424)

[7. Entorno virtual. 7](#_Toc180144425)

[8. Configuración de bases de datos 8](#_Toc180144426)

[9. Ejecución de Pipeline scripts .bat 8](#_Toc180144427)

[Documentacion Tecnica 8](#_Toc180144428)

[Herramientas y tecnologías utilizadas 8](#_Toc180144429)

[1. Creación del Entorno Virtual (venv) 8](#_Toc180144430)

[Contenido del archivo requirements.txt: 9](#_Toc180144431)

[Instalación de Dependencias: 9](#_Toc180144432)

[Estructura del Directorio del Proyecto 9](#_Toc180144433)

[Manual de ejecución del Pipeline de limpieza 10](#_Toc180144434)

[Preparación de los scripts para el proceso de limpieza 11](#_Toc180144435)

[10. Ejecución del Pipeline 13](#_Toc180144436)

[Resultados Esperados del Proyecto 15](#_Toc180144437)

[Diseño de un Data WareHouse 15](#_Toc180144438)

[Introducción 15](#_Toc180144439)

[Alcance 15](#_Toc180144440)

[Arquitectura del Data WareHouse 16](#_Toc180144441)

[Relaciones entre Tablas 16](#_Toc180144442)

[Diagrama ERD (Entity-Relationship Diagram) 17](#_Toc180144443)

[Relaciones en el Modelo Dimensional 17](#_Toc180144444)

[Relación entre Dim\_Fecha y Hechos\_Licencias: 17](#_Toc180144445)

[Relación entre Dim\_Ubicación y Hechos\_Licencias: 17](#_Toc180144446)

[Relación entre Dim\_Tipo\_Licencia y Hechos\_Licencias: 18](#_Toc180144447)

[Integración con Datos Limpios 18](#_Toc180144448)

[Aplicación de la Metodología Inmon: 19](#_Toc180144449)

[Comprender la Tercera Forma Normal (3NF) 19](#_Toc180144450)

[Identificación de Entidades y Atributos 19](#_Toc180144451)

[Descripción de la Metodología Inmon 19](#_Toc180144452)

[Descripción de la Normalización en 3NF 19](#_Toc180144453)

[Transformación desde el Modelo Estrella 20](#_Toc180144454)

[Documentación de la Transformación 21](#_Toc180144455)

# Introducción:

## Propósito del documento.

Este documento tiene como objetivo describir el proceso de desarrollo y las principales características del proyecto, dónde se realizará carga, limpieza y procesamiento de datos utilizando Python y Apache Spark. También detallará el propósito, el alcance y las funcionalidades que llevará al sistema implementado. Así también como las técnicas y herramientas empleadas para el desarrollo de este proyecto. Este documento servirá como guía para. Entender el flujo de trabajo y la arquitectura, los resultados esperados del sistema.

## Alcance del proyecto.

El proyecto abarca el desarrollo de un sistema de limpieza y el procesamiento de datos masivos utilizando Python y Apache Spark. El objetivo principal es garantizar que los datos provenientes de un conjunto de archivos CSV y JSON puedan ser limpiados, transformado y preparado para su respectivo análisis.

Los procesos claves incluyen eliminación de valores nulos, creación de columnas derivada y ejecución de cálculo, normalización de los datos y conversión de datos, Este sistema se implementa en un entorno de procesamiento distribuido capaz de manejar grandes volúmenes de datos.

## Breve descripción del sistema o producto.

El sistema desarrollado es una solución de limpieza y procesamiento de datos basada en Apache, Spark y Python, que son herramientas orientadas a la transformación eficiente de los datos a gran escala. El proyecto se centra en el procesamiento de un conjunto de dataset que contienen información sobre terrazas comerciales, urbanas, incluyendo de dimensiones, ubicaciones, permiso, horario y operaciones.

# El flujo de trabajo incluye las siguientes etapas:

1. Creación del entorno de trabajo**:** Se instalarán todas las librerías y herramientas adecuadas para el desarrollo del proyecto
2. Lectura de datos: Se utilizará un conjunto de archivos como fuente de entrada donde se cargarán los datos con Apache Spark y convertirlo en un DataFrame distribuido.
3. Limpieza de datos: eliminación de filas con valores nulos. Más de 50% de sus columnas. También se identifican y se corrigen inconsistencia en los datos.
4. Concatenación archivos: A partir de archivos limpios y existentes, se generan uniones y concatenación de archivos

## Procesamiento vertical y horizontal:

1. Procesamiento vertical:se realizan transformaciones en las columnas como normalización y limpieza de datos
2. Procesamiento horizontal:Se añaden nuevos registros mediante la carga de múltiples archivos y se combinan datasets.
3. Salida de datos:Los datos procesados se exportan y se almacenan en formato apto para su análisis posterior en bases de datos SQL.

# Requisitos:

# Requisitos funcionales y no funcionales.

Los requisitos funcionales definen las funciones y capacidades del sistema que debe cumplir para lograr los objetivos del proyecto.

## Requisitos funcionales.

1. Lecturas de archivos
2. Limpieza de datos
3. Transformación de datos
4. Almacenamientos de datos procesados
5. Escalabilidad del procesamiento

## Requisitos y no funcionales.

Los requisitos no funcionales describen las características de calidad y restricciones bajo los cuales se debe operar el sistema.

1. Rendimiento
2. Disponibilidad y confianza
3. Escalabilidad
4. Mantenibilidad
5. Seguridad
6. Compatibilidad

# Especificaciones del sistema

# Arquitectura del Sistema:

La arquitectura del sistema describe como los diferentes componentes interactúan y se comunican para realizar el procesamiento de data. En este proyecto, la arquitectura está basada en un modelo distribuido que utiliza Apache Spark para limpieza, transformación y análisis de datos. Como una integración hacia el sistema de almacenamiento y bases de datos.

## Capa de Ingesta

* Los archivos que contienen toda la información se almacenan en un sistema local
* Se utiliza Apache Spark para leer y cargar los datos desde este sistemas de archivos hacia un dataframe distribuido

## Capa de procesamiento de datos (Limpieza y transformación):

* Utilizando PySpark, los datos se limpian eliminando filas con datos faltantes o duplicados y se transforman mediante la creación de nuevas columnas derivadas.
* El sistema utiliza una serie de reglas para realizar operaciones sobre los datos como cálculos de sumas multiplicación y otros.

## Capa de Almacenamiento y Exportación:

* Los datos procesados Exportan a archivos CSV limpios y se almacenan en base de datos SQL o No SQL para su posterior análisis.

# Diagrama de arquitectura.

A continuación, se muestra un diagrama de la arquitectura que describe el flujo de datos:

Datos Crudos

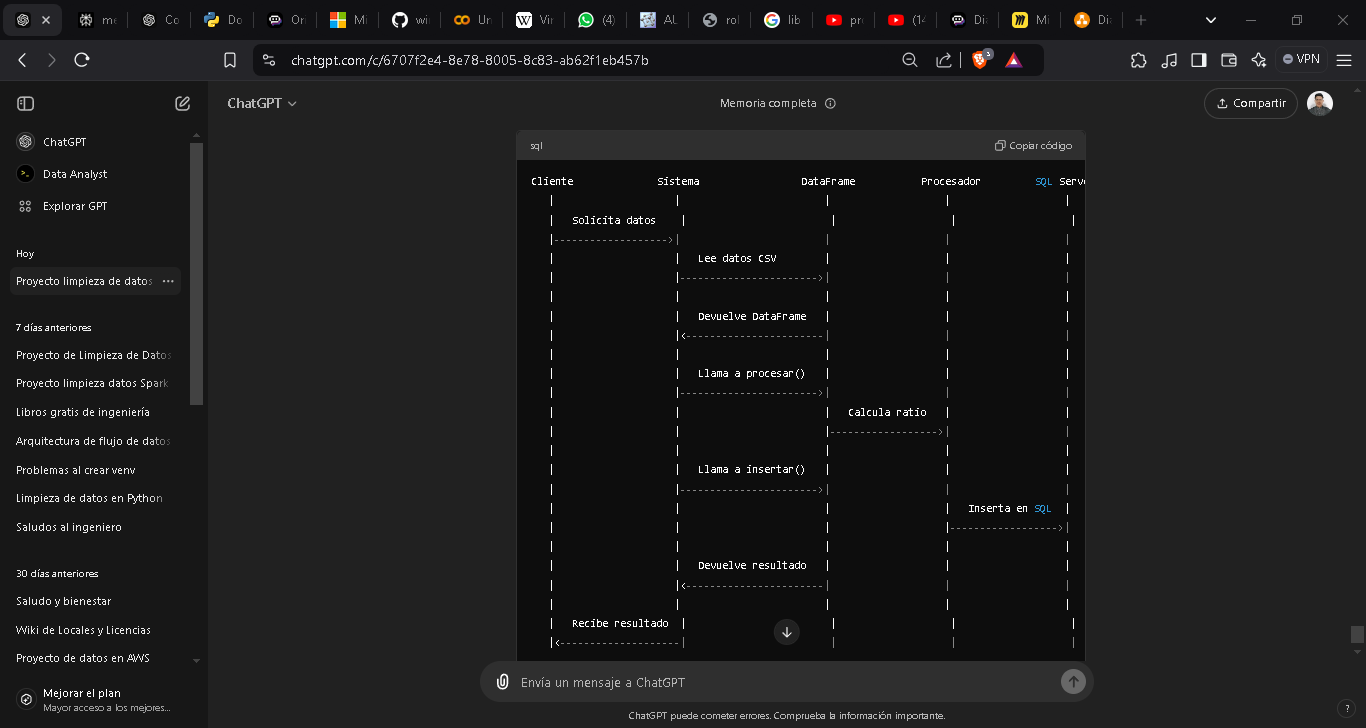
Limpieza de Datos (Script)

Transformación (Calculo de Ratios y Nuevas Columnas)

Creación de Tablas en SQL (Si no existen)

Inserción de Datos en SQL

# Diseño del Sistema:

****Diagrama de Secuencia de inserción de los datos a SQL (clase, secuencia, etc.).

# Implementación:

## Descripción del entorno de desarrollo.

* El entorno de desarrollo para este proyecto está configurado de la siguiente manera.

## Lenguaje de programación:

* Python es el lenguaje principal utilizado para desarrollar, escribir de limpieza y transformación.
* Spark principal componente para el procesamiento de datos a grandes volumen.

## Entorno de Desarrollo Integrado (IDE).

* Se utilizó visual Studio Code (VSCode) para el desarrollo del Código en Python aprovechando su integración con entornos virtuales.

## Entorno virtual.

* El proyecto utiliza un entorno virtual de Python (venv) para gestionar las dependencias y paquetes de manera aislada, asegurando que las versiones de las librerías utilizadas no interfieran con otras aplicaciones.
* Los paquetes principales instalados en el enfermo mental incluyen:
* Pyspark Para el procesamiento de transformación de datos con Apache Spark.
* Pymsql y pyodbc Para conectarse a la base de datos es el QSL o MySql para consultas necesarias.

## Configuración de bases de datos

* SQL Server 2019 es la versión del servidor de la base de datos utilizada para almacenar los datos limpios. La conexión se establece mediante los drivers adecuados configurando el servidor con usuarios y contraseñas necesarias para acceder a la instancia.

## Ejecución de Pipeline scripts .bat

* El scripts principal es (app.bat) donde se ejecuta en la línea de comando, activando todo el entorno y ejecutando la aplicación.
* Este entorno garantiza que el proyecto sea portable, controlado y fácil de replicar en otros equipos o entorno de producción

# Documentacion Tecnica

# Herramientas y tecnologías utilizadas

## 1. Creación del Entorno Virtual (venv)

Para aislar las dependencias del proyecto, creamos un entorno virtual utilizando venv en el directorio del proyecto. Esto nos permitió tener un entorno controlado y evitar conflictos con otras instalaciones globales de Python en el sistema

Pasos:

1-Navegamos al directorio donde se encuentra el proyecto:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

2-Creamos el entorno virtual con el siguiente comando:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Esto creó un nuevo directorio llamado venv, que contiene los archivos del entorno virtual.

3-Activamos el entorno virtual (para sistemas Windows):

Texto

Descripción generada automáticamente

**2**. Archivo requirements.txt

Creamos el archivo requirements.txt para especificar las dependencias que se deben instalar en el entorno virtual. Este archivo contenía las librerías necesarias para el proyecto, tales como pyspark, mysql-connector-python, y pandas.

## Contenido del archivo requirements.txt:

**Interfaz de usuario gráfica, Texto

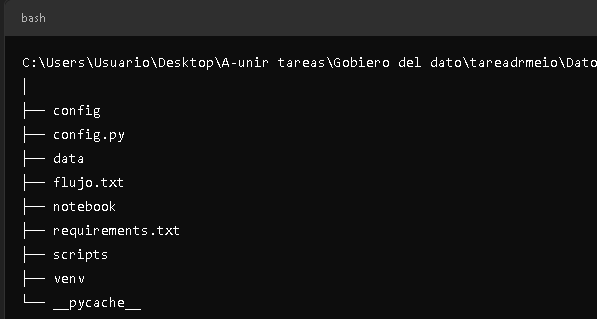
Descripción generada automáticamente con confianza media**

## Instalación de Dependencias:

Una vez que el entorno virtual estuvo activo, ejecutamos el siguiente comando para instalar todas las dependencias listadas en el archivo requirements.txt:

## Estructura del Directorio del Proyecto

Después de crear el entorno y ejecutar los requerimientos, el proyecto debería tener la siguiente estructura:

****

* **config/:** Contiene configuraciones adicionales.
* **config.py:** Archivo de configuración principal que centraliza credenciales y rutas.
* **data/:** Directorio donde se almacenan los datos en bruto.
* **flujo.txt:** Documento que describe el flujo de trabajo.
* **notebook/:** Directorio para los Jupyter notebooks si fuera necesario.
* **requirements.txt**: Lista de dependencias.
* **scripts/:** Contiene scripts Python para la extracción, limpieza y almacenamiento de datos.
* **venv/:** El entorno virtual donde se instalaron las dependencias.
* **pycache/:** Directorio generado automáticamente para almacenar los archivos compilados de Python.

# Manual de ejecución del Pipeline de limpieza

Debes tener configurado un entorno virtual (venv) en Python con las dependencias necesarias instaladas, incluyendo pyspark y pymssql/pyodbc.

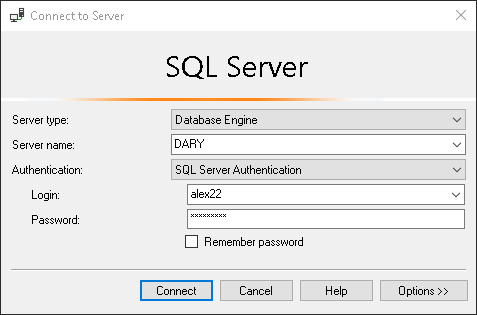
Como ya tenemos listo el entorno realizaremos los siguientes pasos:

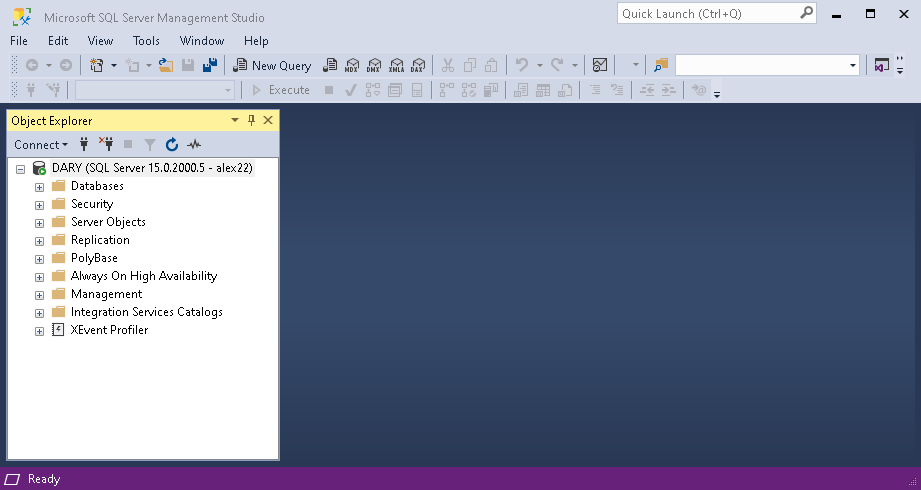
* 1. Texto

     Descripción generada automáticamenteDebemos estar en la ruta correcta donde creamos nuestro entorno
  2. Texto

     Descripción generada automáticamenteIngresamos el siguiente comando para poder trabajar dentro del entorno y preparar el proceso de limpieza   
     # .\venv\Scripts\activate  
       
       
       
       
       
     Podemos observar que ingreso correctamente al entorno .
  3. **Acceso a la Base de Datos SQL Server**:

Nos aseguramos de tener las credenciales correctas para conectarte a la base de datos de SQL Server donde se almacenarán los datos limpios.

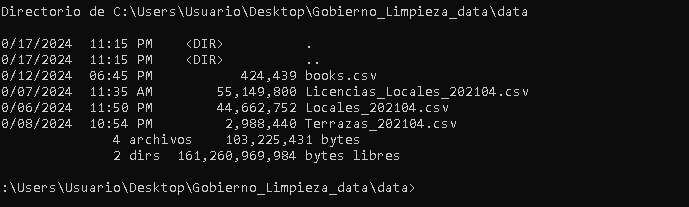




Las credenciales donde se guardarán los archivos limpios son correctas

* 1. **Archivos CSV de Entrada**:

Los archivos deben estar ubicado en la ruta correcta para que el script pueda leerlo.



# Preparación de los scripts para el proceso de limpieza

* 1. Creamos un archivo config donde estén las rutas de los archivos a limpiar y las credenciales de la bases de datos

Texto

Descripción generada automáticamenteUsamos un archivo de configuración porque es una práctica recomendada en el desarrollo de software que mejora la seguridad, mantenibilidad, facilidad de uso y separación de responsabilidades. Es esencial para gestionar las credenciales y configuraciones necesarias para que el proyecto funcione correctamente.

* 1. Un reloj digital en la pantalla

     Descripción generada automáticamente con confianza mediaLimpieza y transformación de los datos   
     -Funciones clave utilizadas en el proceso de limpieza y transformación de los datos   
     Carga de datos

Filtrados de nulos

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteNormalizar columnas

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza mediaCalcular Ratio

Texto

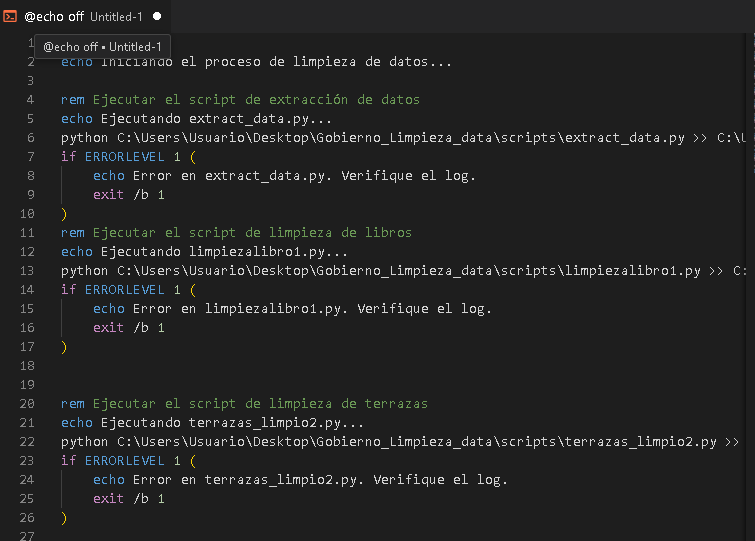
Descripción generada automáticamente con confianza mediaConvertir horas en formato adecuado

* 1. **Resultados Esperados**

Un conjunto de datos limpio y normalizado que puede ser utilizado para análisis posteriores o ser insertado en una base de datos SQL Server.

## Ejecución del Pipeline

Se utiliza el archivo por lotes**run\_app.bat**

El archivo run\_app.bat es un script de Windows que se utiliza para ejecutar automáticamente el pipeline de carga, limpieza y procesamiento de datos. Este archivo permite a los usuarios iniciar todo el proceso con un solo comando, simplificando la ejecución del proyecto.   


* 1. **Ejecución del Pipeline**
     1. Texto

        Descripción generada automáticamenteIngresamos al entorno del proyecto
     2. **Texto

        Descripción generada automáticamente**Ingresamos a la ruta donde se encuentra nuestro archivo **run\_app.bat**
     3. **Texto

        Descripción generada automáticamente con confianza media**Dentro de la ruta ejecutamos el archivo **run\_app.bat**
     4. Texto

        Descripción generada automáticamenteEl proceso se ejecutó correctamente   
        Interfaz de usuario gráfica, Texto

        Descripción generada automáticamente
     5. Interfaz de usuario gráfica, Texto, Aplicación

        Descripción generada automáticamenteVerificamos los datos se cargaron correctamente

# Resultados Esperados del Proyecto

1. Archivos Limpiados y Normalizados: Los datasets deben estar libres de registros nulos excesivos y duplicados, asegurando que todos los valores sean consistentes.
2. Datos Integrados: El resultado final debe incluir un dataset integrado que combine información relevante sobre terrazas y licencias, listo para análisis posteriores.
3. Informes Generados: Se generarán logs que documentan cada paso del proceso, permitiendo a los desarrolladores rastrear cualquier problema o inconsistencia que surja durante la ejecución.
4. Facilidad para Análisis Futuro: Los datos deben estar estructurados y listos para ser utilizados en herramientas analíticas o reportes, facilitando decisiones basadas en datos precisos.

Esta documentación proporciona una visión general clara del proyecto, sus objetivos, el flujo del proceso y las funciones utilizadas, así como instrucciones sobre cómo ejecutarlo adecuadamente y los resultados esperados.

# Diseño de un Data WareHouse

# Introducción

La creación de este proyecto de **Data** **Warehouse** tiene su origen en el trabajo previo realizado en el proceso de limpieza de datos sobre licencias y terrazas. Durante este proceso, se desarrolló un **pipeline de limpieza de datos** que unificó, depuró y estructuró la información extraída de distintas fuentes. Este pipeline fue clave para asegurar la calidad y consistencia de los datos, eliminando duplicados, corrigiendo errores de codificación y normalizando las estructuras.

Una vez los datos estuvieron limpios y preparados, fueron utilizados como base para la construcción de este **Data Warehouse**, diseñado para soportar análisis avanzados y generar reportes útiles para la toma de decisiones estratégicas.

# Alcance

El Data Warehouse creado abarca las siguientes áreas de datos:

* **Licencias de terrazas**: Información detallada sobre las licencias emitidas, sus fechas y características.
* **Ubicaciones**: Datos geográficos que permiten analizar la distribución de las terrazas en diferentes ciudades y estados.
* **Tipos de licencias**: Clasificación de las licencias según su tipo.
* **Información temporal**: Fechas asociadas a las licencias, permitiendo realizar análisis por año, mes, trimestre, y día.

Este Data Warehouse es el resultado del procesamiento de datos limpios provenientes del pipeline de limpieza, garantizando la integridad y fiabilidad de la información para análisis futuros.

# Arquitectura del Data WareHouse

El diseño del Data Warehouse para este proyecto se ha basado en una arquitectura de estrella (*star schema*), lo cual es ampliamente utilizado en entornos de análisis de datos debido a su simplicidad y eficiencia para realizar consultas y análisis multidimensionales.

Diagrama

Descripción generada automáticamente

El modelo consta de dos tipos principales de tablas:

* **Tablas de Hechos (Facts)**: Contienen los eventos o transacciones principales que se desean analizar. En este caso, la tabla de hechos almacena las licencias emitidas para terrazas.
* **Tablas de Dimensiones (Dimensions)**: Describen los atributos o características de los hechos, como ubicación, tipo de licencia, y tiempo, permitiendo desglosar y categorizar los eventos.

## Relaciones entre Tablas

Las relaciones en este Data Warehouse están definidas como claves foráneas entre las tablas de hechos y las tablas de dimensiones. El diagrama ERD presentado anteriormente describe las siguientes relaciones:

* **Relación entre Hechos\_Licencias y Dim\_Ubicación**: Relación de uno a muchos. Cada licencia está asociada a una ubicación específica mediante la columna Ubicacion\_ID.
* **Relación entre Hechos\_Licencias y Dim\_Tipo\_Licencia**: Relación de uno a muchos. Cada licencia está vinculada a un tipo específico de licencia mediante la columna Tipo\_Licencia\_ID.
* **Relación entre Hechos\_Licencias y Dim\_Fecha**: Relación de uno a muchos. Cada licencia está asociada a una fecha de emisión mediante la columna Fecha\_ID.

# Diagrama ERD (Entity-Relationship Diagram)

**Diagrama

Descripción generada automáticamente**

# Relaciones en el Modelo Dimensional

## Relación entre Dim\_Fecha y Hechos\_Licencias:

* Tipo: Uno a Muchos
* Descripción: Cada fecha (en Dim\_Fecha) puede estar asociada a múltiples licencias (en Hechos\_Licencias). Por ejemplo, varias licencias pueden ser otorgadas en la misma fecha.

## Relación entre Dim\_Ubicación y Hechos\_Licencias:

* Tipo: Uno a Muchos
* Descripción: Cada ubicación (en Dim\_Ubicación) puede tener múltiples licencias asociadas. Por ejemplo, múltiples licencias pueden ser emitidas para la misma ubicación.

## Relación entre Dim\_Tipo\_Licencia y Hechos\_Licencias:

* Tipo: Uno a Muchos
* Descripción: Cada tipo de licencia (en Dim\_Tipo\_Licencia) puede estar asociado a múltiples licencias. Por ejemplo, muchas licencias pueden ser del mismo tipo (residencial, comercial, etc.).

## Integración con Datos Limpios

La construcción de este Data Warehouse se basa en los datos limpios y preprocesados provenientes del pipeline de limpieza, lo que asegura que las tablas de hechos y dimensiones no contengan duplicados, errores de codificación o inconsistencias. Este enfoque garantiza la calidad y precisión de los análisis derivados del Data Warehouse.

Imagen que contiene Tabla

Descripción generada automáticamente

Como vemos los datos se insertaron correctamente extrayéndolos de los datos limpios que se guardaron anterior mente después de la limpieza

# Aplicación de la Metodología Inmon:

Para implementar un modelo de Data Warehouse en tercera forma normal (3NF) utilizando la metodología de Inmon, debes seguir un enfoque que garantice la eliminación de redundancias y la normalización completa de las tablas.

## Comprender la Tercera Forma Normal (3NF)

La 3NF implica que:

1. **Estar en 2NF**: Todos los atributos no clave son totalmente dependientes de la clave primaria.
2. **No tener dependencias transitivas**: Ningún atributo no clave debe depender de otro atributo no clave.

## Identificación de Entidades y Atributos

Para el dataset Licencias\_Terrazas\_Integradas, podemos identificar las siguientes entidades:

* **Licencias**: Información sobre cada licencia.
* **Ubicaciones**: Detalles sobre las ubicaciones.
* **Tipos de Licencia**: Clasificación de las licencias.
* **Fechas**: Información desglosada sobre fechas.

## Descripción de la Metodología Inmon

En esta sección, se describe la implementación de un modelo de Data Warehouse siguiendo la Metodología Inmon, que se basa en el diseño de un modelo de datos normalizado en tercera forma normal (3NF). El objetivo es garantizar la eliminación de redundancias, promover la consistencia de los datos, y asegurar que las tablas estén completamente normalizadas.

Este enfoque es diferente al modelo estrella implementado inicialmente, donde la desnormalización mejora el rendimiento de las consultas analíticas, pero puede llevar a la duplicación de datos.

## Descripción de la Normalización en 3NF

La **tercera forma normal (3NF)** busca que cada columna de una tabla dependa únicamente de la clave primaria de esa tabla, eliminando cualquier dependencia transitiva o redundancia. Esto significa dividir las tablas existentes en estructuras más pequeñas y específicas, manteniendo integridad referencial mediante claves foráneas.

Al aplicar esta metodología a los datos del proyecto, se hace necesario rediseñar las tablas para cumplir con los principios de normalización. Se crearon nuevas tablas para representar de manera más eficiente las relaciones entre las entidades, como Estado, Ciudad, Ubicación, y se ajustaron las relaciones entre ellas para minimizar la duplicación de datos.

## Transformación desde el Modelo Estrella

El modelo estrella original utilizado en este proyecto, que facilitaba la consulta de datos, ahora debe ser normalizado. A continuación, se detalla el proceso de transformación:

1. **Dim\_Ubicación** en el modelo estrella incluía información sobre el estado y la ciudad. En el nuevo diseño 3NF, estas características se dividen en tres tablas: Estado, Ciudad, y Ubicación.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. **Dim\_Tipo\_Licencia**: Esta tabla se mantiene, ya que los tipos de licencias ya son una entidad independiente. Sin embargo, podría subdividirse si se identifica alguna redundancia adicional.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **Dim\_Fecha**: La tabla de fechas también se mantiene, ya que cumple con las reglas de normalización al tener únicamente dependencias sobre la clave primaria (Fecha\_ID).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. **Hechos\_Licencias**: En el modelo estrella, esta tabla de hechos contiene claves foráneas hacia las dimensiones desnormalizadas. En el modelo 3NF, ahora contiene claves foráneas hacia las nuevas tablas normalizadas (Ubicación, Ciudad, Estado, etc.).

Texto

Descripción generada automáticamente

## Documentación de la Transformación

La transición del modelo estrella al modelo en 3NF involucró:

* **Descomposición** de las dimensiones de ubicación en tablas más pequeñas (Estado, Ciudad, Ubicación).
* **Actualización de la tabla de hechos** para referenciar las nuevas tablas normalizadas.
* **Modificación de las relaciones** entre tablas, asegurando el uso de claves foráneas para mantener la integridad referencial.

Este rediseño permite la **escalabilidad** y el **mantenimiento** del Data Warehouse a largo plazo, a costa de un mayor esfuerzo en la construcción de consultas más complejas y en el tiempo de procesamiento.