

FOXTECH

INFORME DE MIGRACIÓN DE DATOS



Introducción



Configuración sistemas gestores de bases de datos



Evidencia de la migración de la base de datos en diferentes motores (SQLite, PostgreSQL, M...

Introducción

INTEGRANTES:

- Andrea Lenés

Introducción

A continuación, podrá encontrar un informe detallado sobre el proceso de migración de datos, las herramientas utilizadas, el paso a paso y las configuraciones de los SGBD.

Objetivo general

Documentar el proceso de migración de datos.

Objetivos específicos

- Delimitar las herramientas utilizadas para la migración.
- Guiar paso a paso la migración de datos.
- Describir la configuración de los SGBD.

Justificación

En ocasiones es necesario migrar datos, sin embargo, hay algunas cosas que cambian de un SGBD a otro, por ello es importante realizar un informe de migración que permita guiar a los interesados en realizar tal proceso. En ese sentido, se presenta el informe de migración de datos

Configuración sistemas gestores de bases de datos

En la carpeta del proyecto de django crearemos un archivo llamado db.py en el cual se almacenara la configuración de cada gestor de base de datos que vamos a utilizar, cada gestor estará representado como una variable constante con los siguientes campos:

- **Engine:** Motor y su respectivo driver para realizar la conexión con la base de datos.
- **Name:** Nombre de la base de datos.
- **User:** Usuario registrado en el gestor de la base de datos.
- **Password:** Contraseña del usuario registrado en el gestor de la base de datos.
- **Host:** Url o dirección IP en donde esta alojado la base de datos.
- **Port:** Puerto de la url o dirección IP en donde esta alojado la base de datos



SQLite no tendrá una configuración tan extensa debido a que este motor de base de datos no maneja usuarios ni vistas.

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

SQLITE = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3')
    }
}

POSTGRESQL = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'Foxtech',
        'USER': 'postgres',
        'PASSWORD': '1234',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}

MYSQL = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'Foxtech',
        'USER': 'root',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '3306'
    }
}
```

Evidencia de la migración de la base de datos en diferentes motores (SQLite, PostgreSQL, MySQL)

Una vez creado el archivo db.py en donde se almacenara la configuración de la conexión a los tres motores de bases de datos, al principio del archivo settings.py se procederá a importar la librería del sistema operativo y se importara el archivo db.py al cual le colocaremos como alias "db".

```
import os
import Foxtech.db as db
```

Migración a SQLite

1. En el archivo settings.py en la variable constante "DATABASES", se procederá a llamar a la variable que contiene la configuración de la conexión con el motor de la base de datos en este caso SQLite.

```
DATABASES = db.SQLITE
```

2. Una vez especificado en el archivo settings.py a que motor vamos a realizar la conexión, en la terminal procederemos a ejecutar el siguiente comando para realizar la migración de la base de datos a SQLite.

```
python3 manage.py migrate
```

3. Abrimos la base de datos en el gestor de SQLite para confirmar que la migración se realizo de manera correcta.

Name	Type	Schema
Tables (19)		
auth_group		CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE)
auth_group_permissions		CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED)
auth_permission		CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, "codename" varchar(100) NOT NULL)
auth_user		CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL, "is_superuser" bool NOT NULL, "username" varchar(150) NOT NULL UNIQUE, "first_name" varchar(30) NULL, "last_name" varchar(30) NULL, "email" varchar(254) NULL)
auth_user_groups		CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED)
auth_user_user_permissions		CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED)
category		CREATE TABLE "category" ("id_category" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(55) NOT NULL, "description" varchar(255) NULL)
django_admin_log		CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "action_time" datetime NOT NULL, "object_id" text NULL, "object_repr" varchar(200) NOT NULL, "change_message" text NOT NULL)
django_content_type		CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varchar(100) NOT NULL)
django_migrations		CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) NOT NULL, "applied" datetime NOT NULL)
django_session		CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NULL)
header_ordered		CREATE TABLE "header_ordered" ("id_header_ordered" integer NOT NULL PRIMARY KEY, "order_date" datetime NOT NULL, "state" bool NOT NULL, "payment_reference" varchar(45) NOT NULL, "payment_type" integer NOT NULL REFERENCES "payment_type" ("id_payment_type") DEFERRABLE INITIALLY DEFERRED)
order_detail		CREATE TABLE "order_detail" ("id_order_detail" integer NOT NULL PRIMARY KEY, "quantity" integer NOT NULL, "subtotal" integer NOT NULL, "total" integer NOT NULL, "header_ordered_id" integer NOT NULL REFERENCES "header_ordered" ("id_header_ordered") DEFERRABLE INITIALLY DEFERRED)
payment_type		CREATE TABLE "payment_type" ("id_payment_type" integer NOT NULL PRIMARY KEY, "description" varchar(45) NOT NULL)
product		CREATE TABLE "product" ("id_product" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(55) NOT NULL, "description" varchar(255) NULL, "unit_price" integer unsigned NOT NULL CHECK ("unit_price > 0"))
sqlite_sequence		CREATE TABLE "sqlite_sequence" ("name" text PRIMARY KEY, "seq" integer NOT NULL)
type_accounting_document		CREATE TABLE "type_accounting_document" ("id_type_accounting_document" integer NOT NULL PRIMARY KEY, "nature" varchar(45) NOT NULL)
type_of_delivery		CREATE TABLE "type_of_delivery" ("id_type_of_delivery" integer NOT NULL PRIMARY KEY, "description" varchar(45) NOT NULL)
unit_of_measure		CREATE TABLE "unit_of_measure" ("id_unit_of_measure" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(45) NOT NULL)
Indices (21)		
Views (0)		
Triggers (0)		

Migración a PostgreSQL

Migración a PostgreSQL

1. En el archivo settings.py en la variable constante "DATABASES", se procederá a llamar a la variable que contiene la configuración de la conexión con el motor de la base de datos en este caso PostgreSQL.

```
DATABASES = db.POSTGRESQL
```

2. Una vez especificado en el archivo settings.py a que motor vamos a realizar la conexión, en la terminal procederemos a ejecutar el siguiente comando para realizar la migración de la base de datos a PostgreSQL.

```
python3 manage.py migrate
```

3. Abrimos la base de datos en el gestor pgAdmin4 para confirmar que la migración se realizó de manera correcta.

Migración a MySQL

1. En el archivo settings.py en la variable constante "DATABASES", se procederá a llamar a la variable que contiene la configuración de la conexión con el motor de la base de datos en este caso MySQL.

```
DATABASES = db.MYSQL
```

2. Una vez especificado en el archivo settings.py a que motor vamos a realizar la conexión, en la terminal procederemos a ejecutar el siguiente comando para realizar la migración de la base de datos a MySQL.

```
python3 manage.py migrate
```

3. Abrimos la base de datos en el gestor MySQL Workbench para confirmar que la migración se realizó de manera correcta.