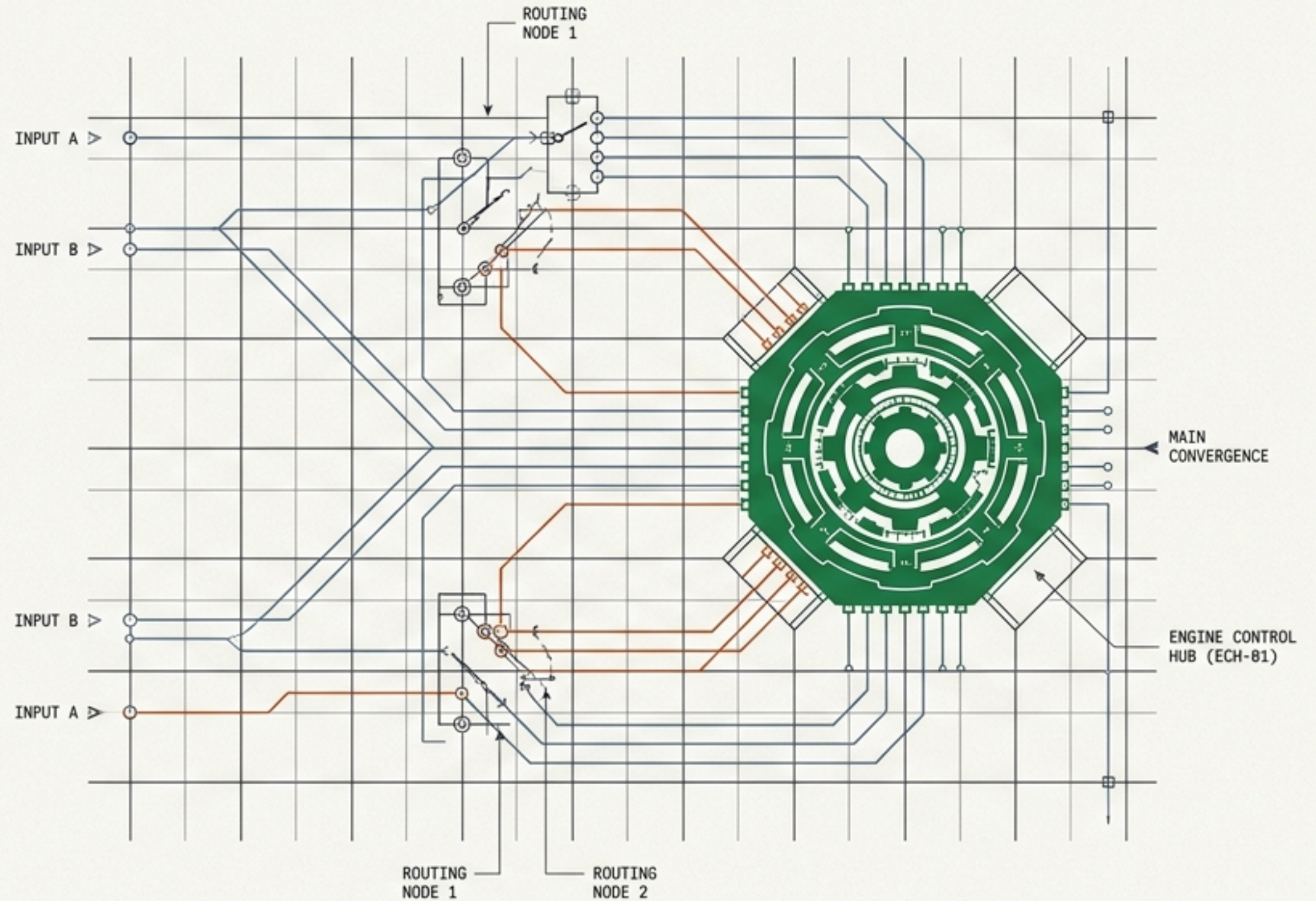


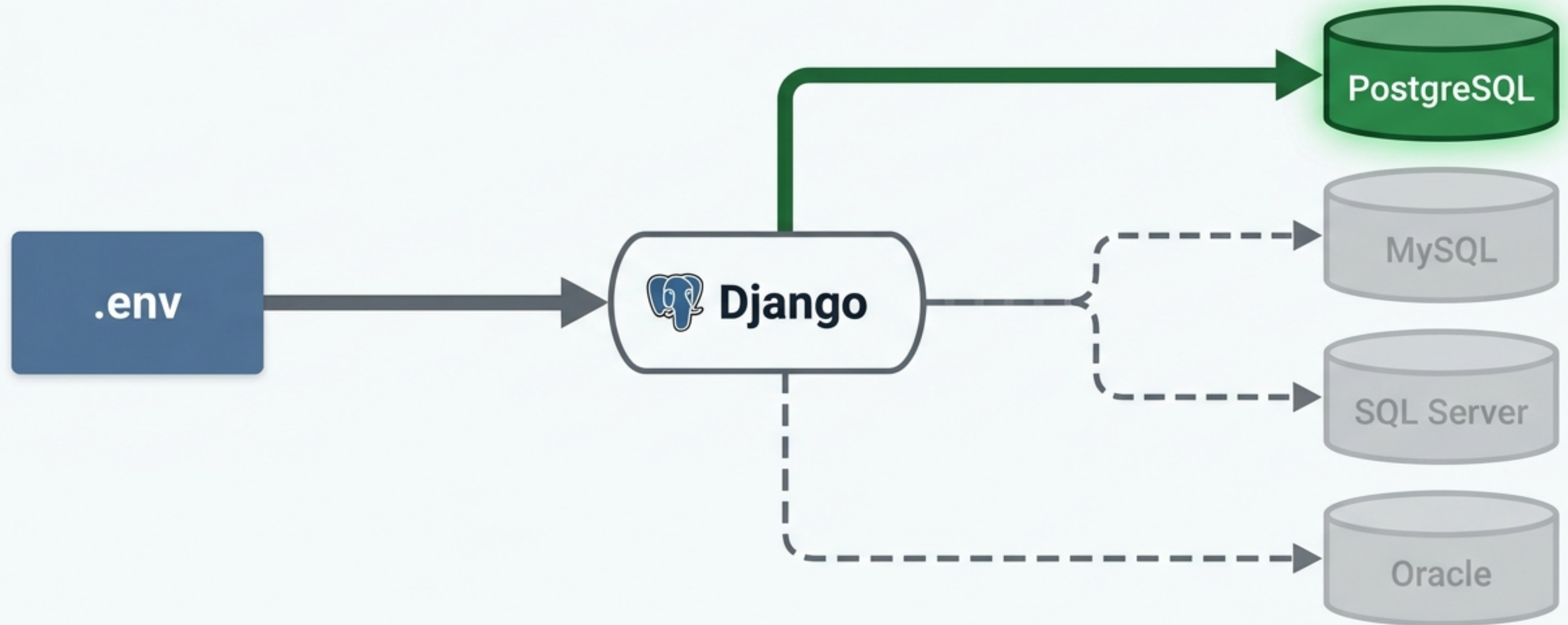
ISS-01

Settings y selección del motor



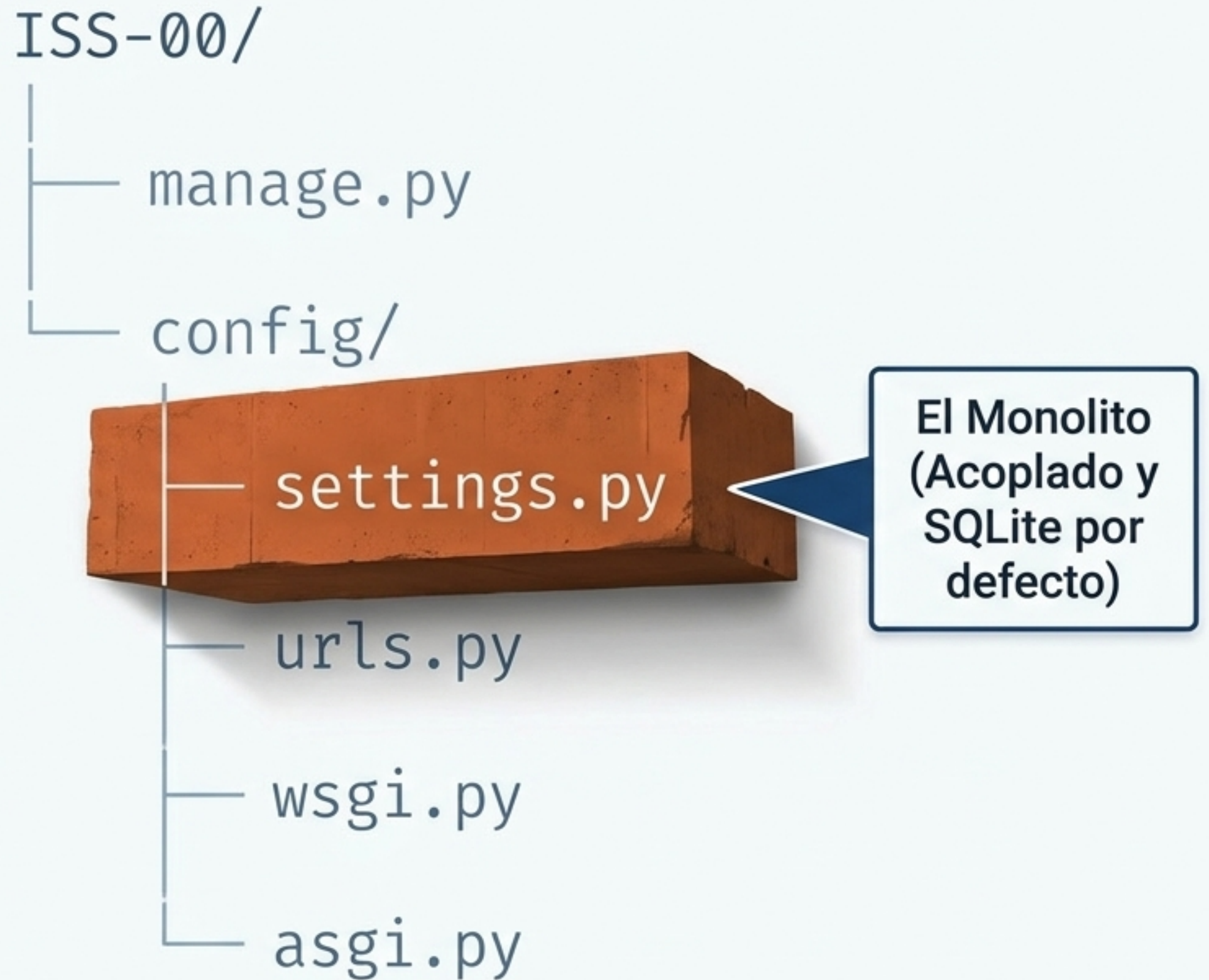
¿Qué construiremos en esta unidad?

Selección dinámica de un solo motor de base de datos por ejecución, inyectado desde el entorno **sin modificar código ni escribir SQL**.



Contexto y Punto de Partida

El entorno aislado está activo, pero el proyecto depende de un archivo **monolítico e inflexible**.



Herramientas vs. Controladores

Django no incluye los drivers de conexión por defecto. Instalamos la lectura de entorno y los traductores para cada dialecto SQL.

Caja 1: Entorno

`python-dotenv==1.2.3`

Expone `load_dotenv()` para leer el `.env` antes de que Django arranque.

Caja 2: Motores de Base de Datos

`psycopg[binary]==3.3.6`

Driver C compilado para PostgreSQL.

`mysqlclient==2.3.0`

Driver oficial documentado por Django.

`mssql-django==2.0.0`

Registra el motor mssql (requiere ODBC Driver 18).

`oracledb==26.0.1`

Reemplaza al deprecado `cx_Oracle`.

Evolución Estructural: De Archivo a Paquete

Un archivo y un directorio no pueden llamarse igual. Eliminamos el monolito para construir una arquitectura modular basada en paquetes.

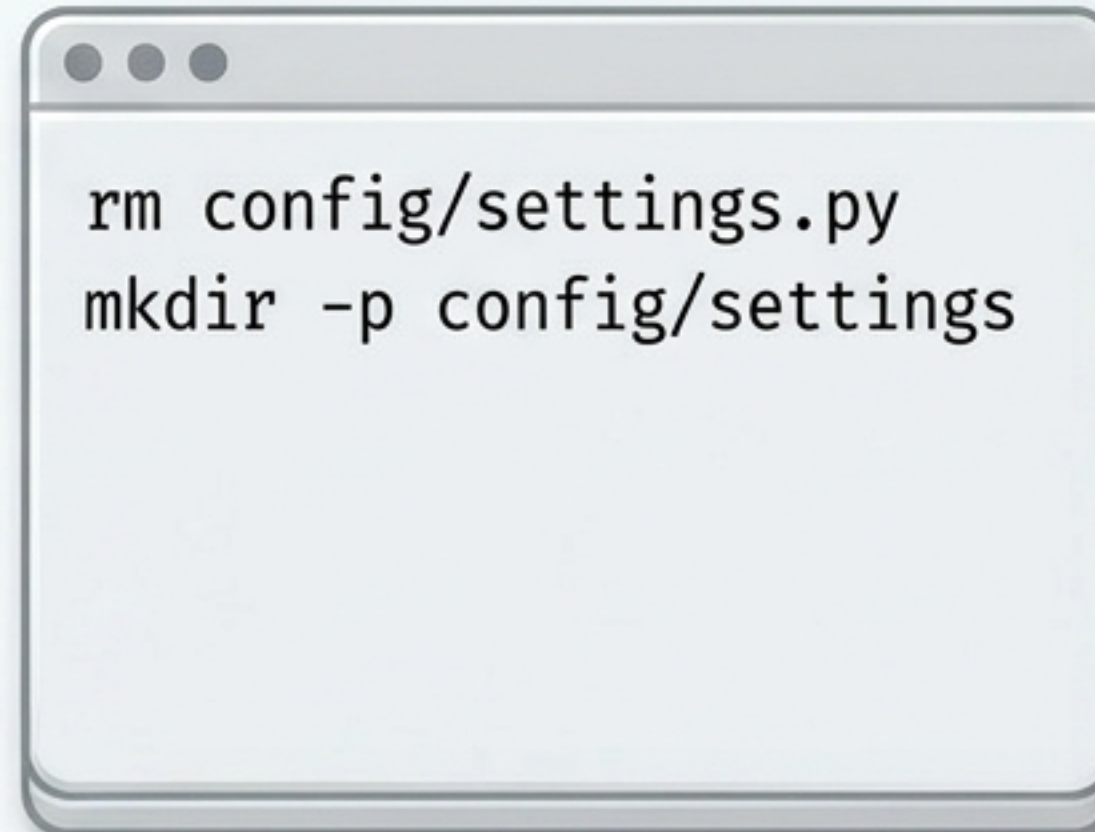
Antes



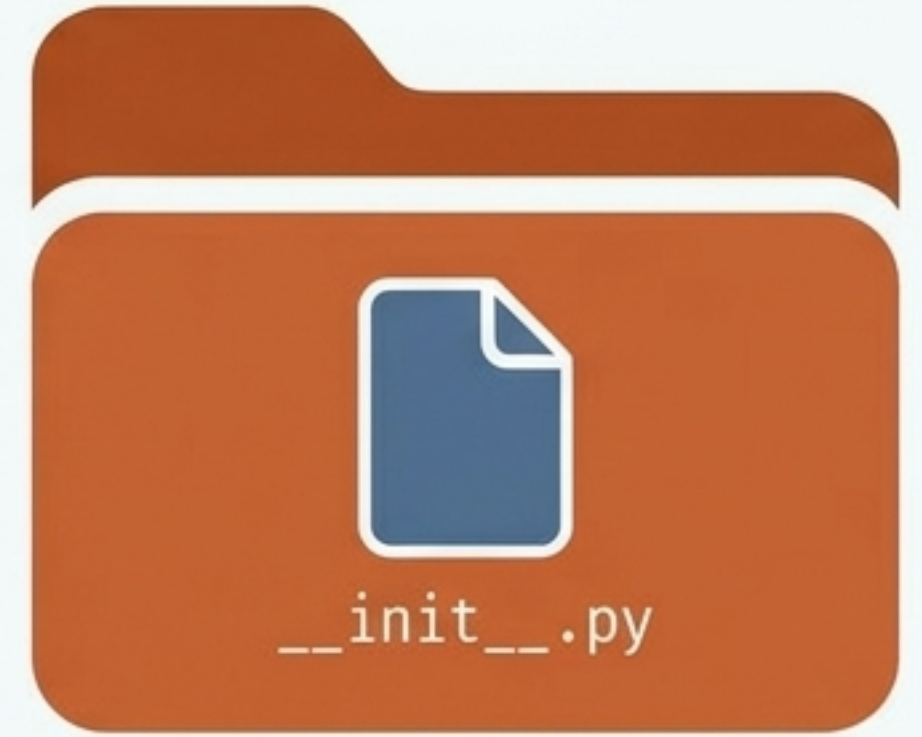
`config/settings.py`



Transición



Después



`config/settings/`

El Panel de Control Externo (.env)

Separación estricta de secretos y configuración. El proceso ignora la plantilla y lee las variables activas.

.env

```
DJANGO_SECRET_KEY=your-secret-key-goes-here  
DJANGO_DEBUG=True
```

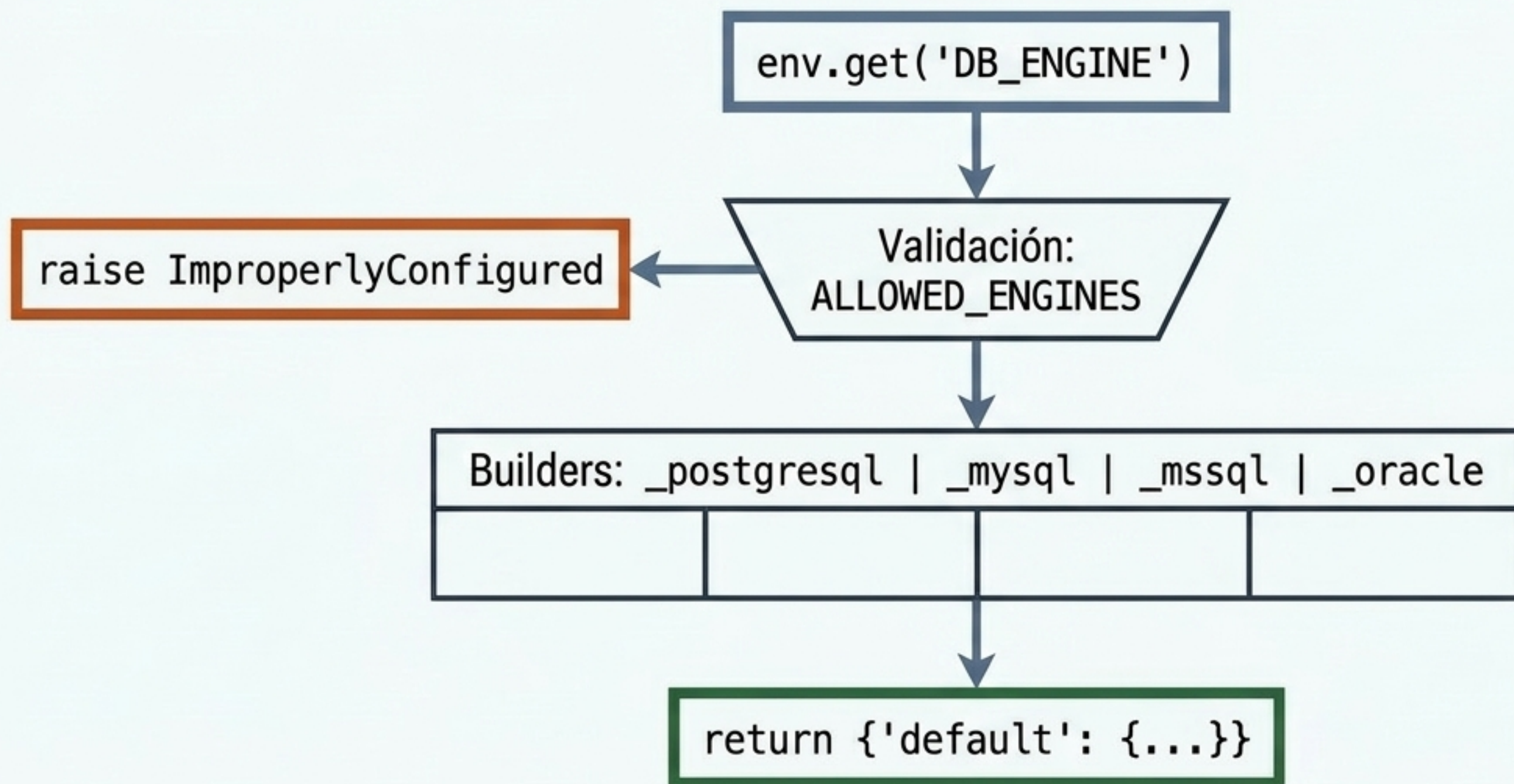
```
DB_ENGINE=postgresql
```

```
POSTGRES_DB=mydatabase  
POSTGRES_USER=myuser  
POSTGRES_PASSWORD=mypassword  
POSTGRES_HOST=localhost  
POSTGRES_PORT=5432
```

```
# MYSQL_DB=mysql_db  
# MYSQL_USER=mysql_user  
# MYSQL_PASSWORD=mysql_pass  
# MYSQL_HOST=127.0.0.1  
# MYSQL_PORT=3306
```

El Módulo de Traducción: config/database.py

Intercepta la variable de entorno, valida contra motores permitidos y delega la construcción del diccionario de conexión.



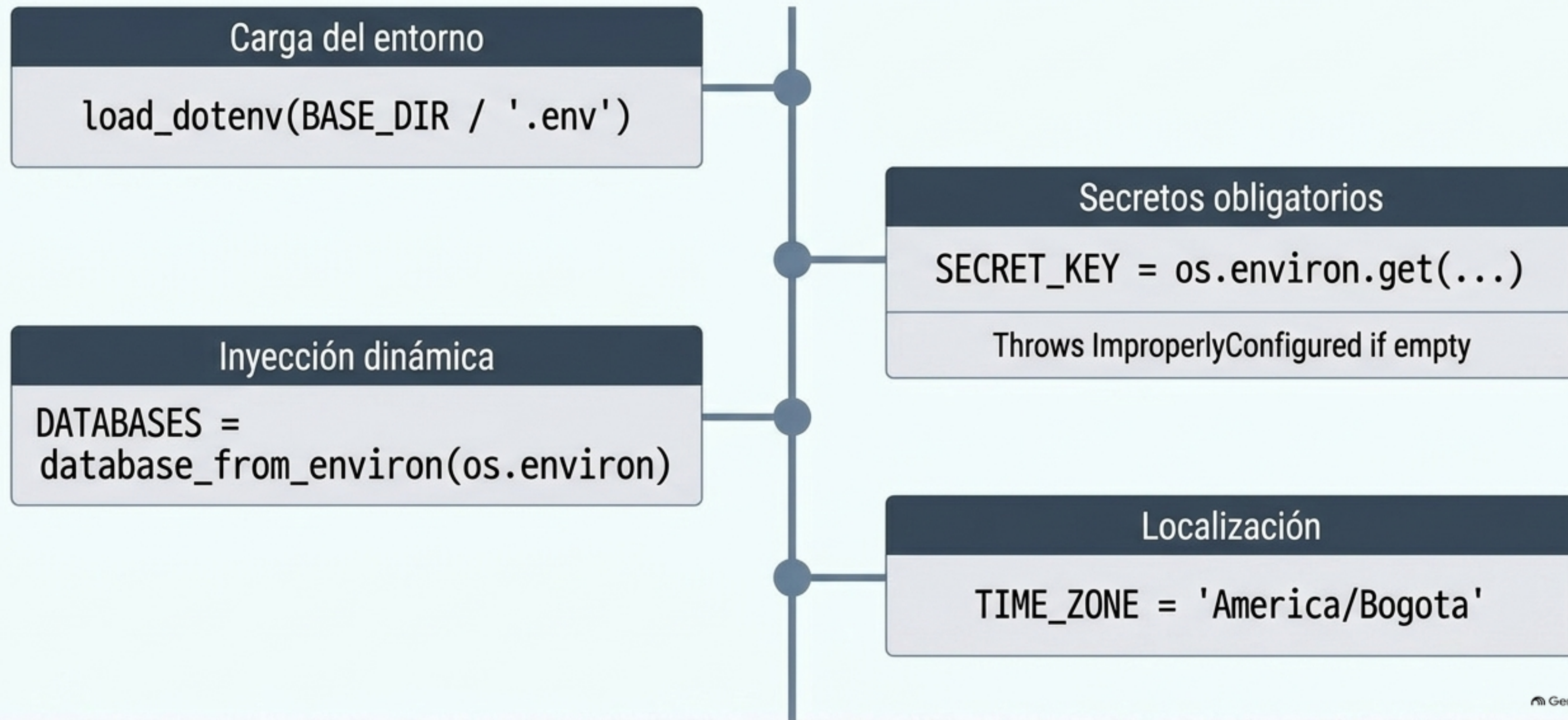
Matriz Diagnóstica: Constructores de Motor

Cada función constructora exige sus propias variables y configura el motor con especificaciones únicas.

Motor	Cadena ENGINE	Puerto por Defecto	Configuraciones Especiales
PostgreSQL	<code>django.db.backends.postgresql</code>	5432	Ninguna
MySQL	<code>django.db.backends.mysql</code>	3306	<code>OPTIONS: {'charset': 'utf8mb4'}</code>
SQL Server	<code>mssql</code>	1433	<code>driver='ODBC Driver 18...', extra_params='...'</code>
Oracle	<code>django.db.backends.oracle</code>	(Vacío)	NAME ensambla un DSN con SERVICE_NAME

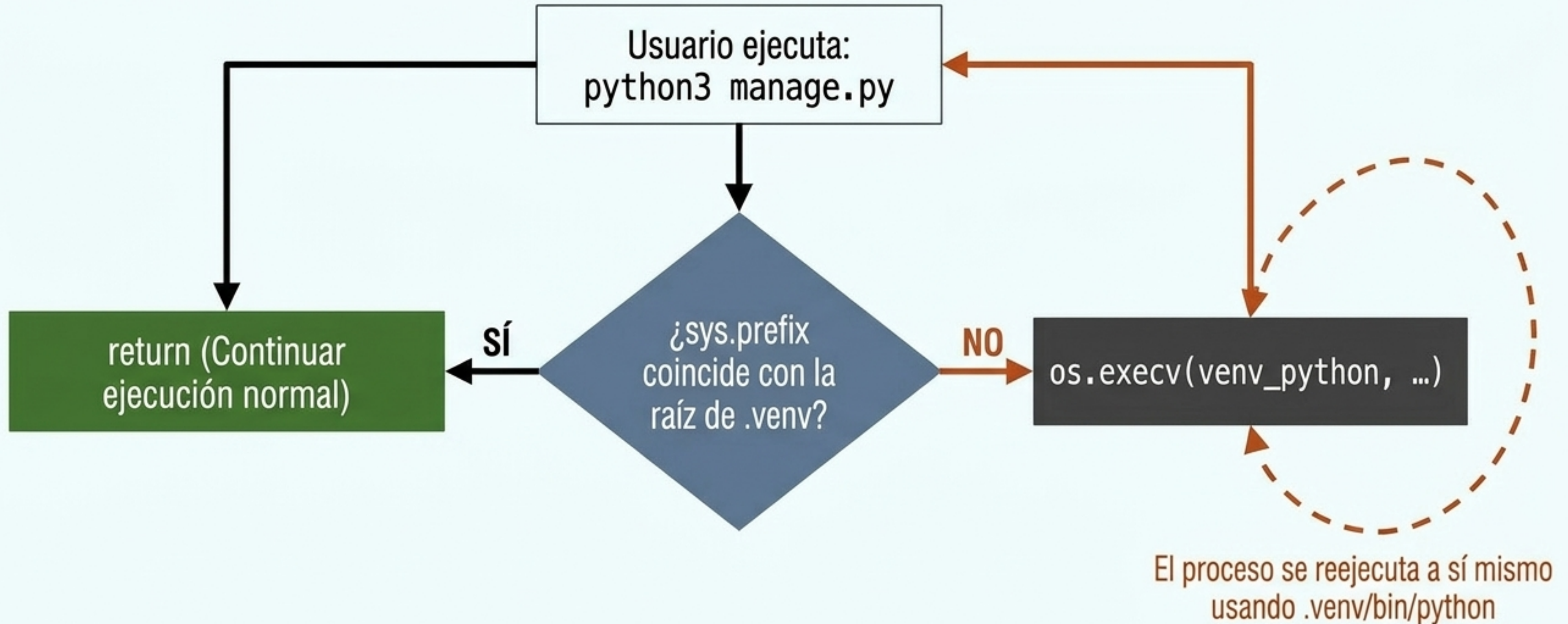
El Nuevo Cerebro: settings/__init__.py

El paquete orquesta la lectura segura del entorno, la inyección del motor y la localización.



Inteligencia de Arranque: `use_project_python()`

Garantiza el aislamiento: redirige la ejecución al entorno virtual de forma automática si se lanza con el Python del sistema.



Parcheando los Puntos de Entrada

Inyectamos inteligencia antes de que el framework comience su ciclo de vida en `manage.py`, `wsgi.py` y `asgi.py`.



`manage.py`

```
def main():
```

Autodetección de intérprete

```
    use_project_python()
```

Carga de variables

```
    load_dotenv()
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', ...)
```

`wsgi.py / asgi.py`

```
from dotenv import load_dotenv
```

```
...
```

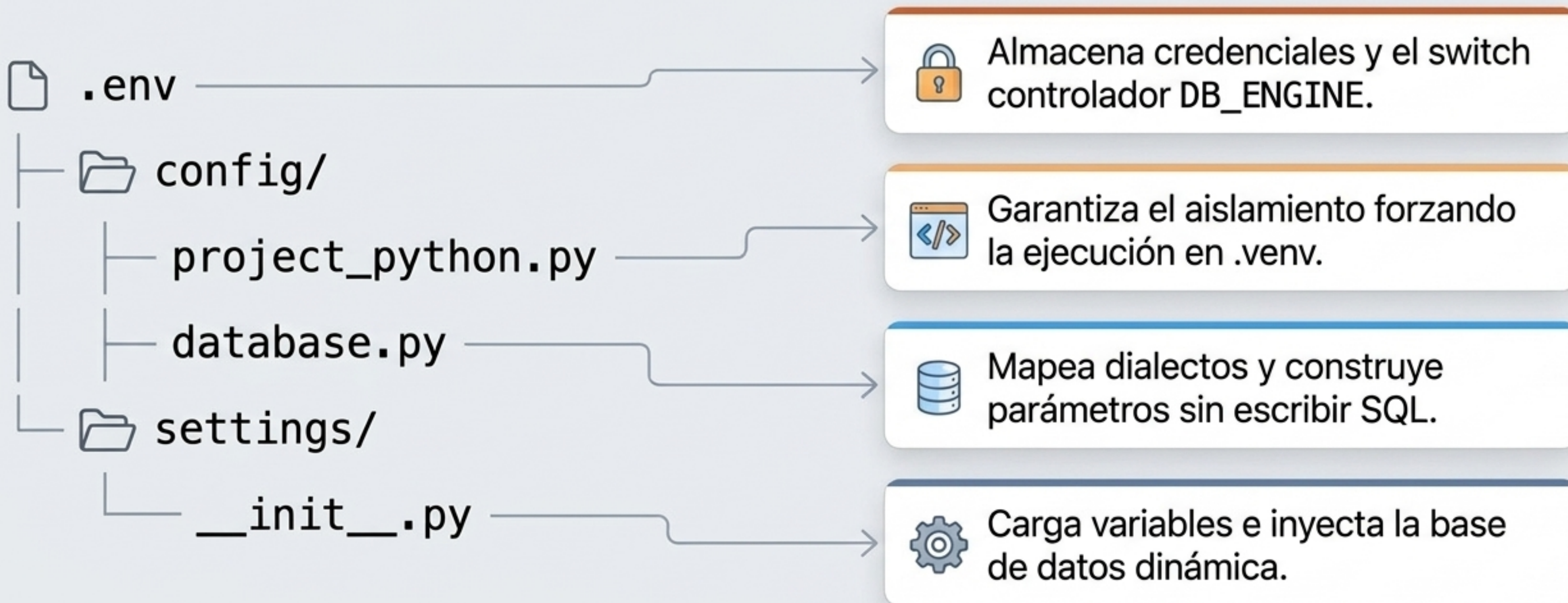
Intercepta antes de settings

```
load_dotenv()
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', ...)
```

Síntesis Arquitectónica: Estado Resultante

El proyecto transiciona de un script generado por defecto a una arquitectura modular y configurable.



Flujo Macro de Ejecución

El viaje secuencial de la configuración: desde el disco hasta la capa de persistencia.



Conceptos Clave de Django

Piezas fundamentales del framework operadas durante esta refactorización.

Card 1

ImproperlyConfigured

Excepción nativa de Django. Se utiliza para detener el arranque de forma segura e inmediata si faltan secretos críticos o si la configuración de la base de datos es inválida.

Card 2

Alias 'default'

El ORM de Django exige que al menos una conexión de base de datos responda al alias 'default' en el diccionario. Nuestro módulo oculta completamente qué motor físico hay detrás de este alias.

Card 3

USE_TZ = True

Localización. Combinado con 'America/Bogota', asegura que las fechas se guarden en la base de datos en formato UTC absoluto, pero se operen y presenten en la hora local correcta.

Diagnóstico y Solución de Errores

Identificación temprana de fallos de configuración durante el arranque del entorno.

Síntoma / Error	Causa Raíz	Solución
 <code>ModuleNotFoundError: No module named 'dotenv'</code>	Instalación de dependencias realizada fuera del entorno virtual (.venv).	 Ejecutar <code>'source .venv/bin/activate'</code> y reinstalar requerimientos.
 Django arranca pero <code>SECRET_KEY</code> está vacío	Se omitió la invocación de <code>load_dotenv()</code> en los puntos de entrada.	 Revisar los parches de inserción en <code>manage.py</code> , <code>wsgi.py</code> y <code>asgi.py</code> .
 <code>ImproperlyConfigured: DB_ENGINE='sqlite' no es válido</code>	El motor definido en el entorno no está en la tupla <code>ALLOWED_ENGINES</code> .	 Corregir el valor en <code>.env</code> a <code>postgresql</code> , <code>mysql</code> , <code>mssql</code> u <code>oracle</code> .

Criterios de Aceptación (AC-01)

Condiciones estrictas que la construcción debe cumplir para considerar superada la unidad.

☐ **AC-01-01** postgresql, mysql, mssql y oracle producen el ENGINE documentado.

☐ **AC-01-02** Un motor desconocido falla inmediatamente lanzando ImproperlyConfigured.

☐ **AC-01-03** La falta de una variable obligatoria interrumpe el arranque antes de intentar conectar.

☐ **AC-01-04** El DSN del motor Oracle se ensambla correctamente integrando SERVICE_NAME.

Verificación y Evidencias

Validamos la lógica de enrutamiento y decisión dinámica sin abrir conexiones físicas al motor.

```
$ python manage.py test  
apps.security.tests.DatabaseSelectionTests  
.....  
-----  
Ran 4 tests in 0.012s  
  
OK
```

Nota Arquitectónica:

Se utiliza SimpleTestCase. El test evalúa únicamente los diccionarios generados por la función `database_from_environ()`, garantizando que la lógica de selección de motor (EVI-01-01) sea correcta antes de intentar cualquier migración.

GATE: Settings y Motor

Matriz de aprobación técnica. Todos los criterios deben alcanzar el estado PASS para continuar.

Criterio	Verificación	Evidencia	Resultado
AC-01-01	Test de verificación por motor	EVI-01-01	PASS
AC-01-02	Test de inyección de motor inválido	EVI-01-01	PASS
AC-01-03	Test de validación de variable vacía	EVI-01-01	PASS
AC-01-04	Aserción de ensamble de SERVICE_NAME	EVI-01-01	PASS

Defensa Oral: Preparación

Preguntas de comprensión arquitectónica sobre la configuración dinámica implementada. (No memorísticas).

¿Por qué delegamos la validación de variables a `database_from_environ` y lanzamos `ImproperlyConfigured` en lugar de dejar que la base de datos falle físicamente al intentar conectar?

Si un administrador olvida activar `.venv` en la terminal, pero ejecuta `'python3 manage.py runserver'`, ¿qué mecanismo salva la ejecución y cómo lo hace exactamente?

¿Por qué el script `project_python.py` verifica el valor de `'sys.prefix'` en lugar de simplemente buscar la ruta absoluta del ejecutable actual en el sistema de archivos?