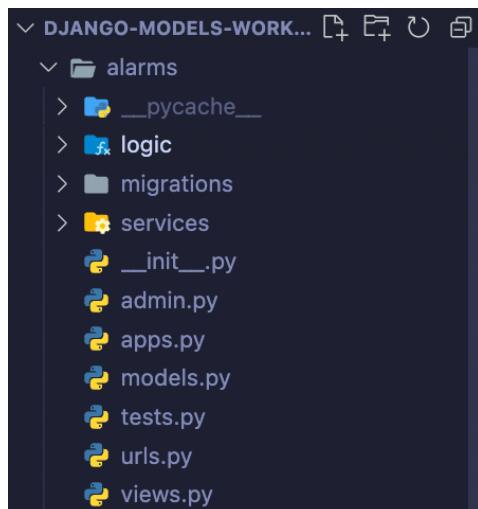


Pasos para la creación de una nueva aplicación “Alarms”:

1. Estructura de archivos de la nueva aplicación:



2. Código de la nueva aplicación:

```
from django.contrib import admin
from .models import Alarm
admin.site.register(Alarm)
```

The screenshot shows a code editor with the file `settings.py` open. The code defines various Django settings, including the secret key, debug mode, allowed hosts, installed apps, middleware, root URLconf, and templates.

```
SECRET_KEY = '+n*zlf1bmm@zuxw0o10uq^no2d77wa%iyay8$$46$82zhlkw63!'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'measurements',
    'variables',
    'alarms',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'monitoring.urls'

TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [],
    'APP_DIRS': True,
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
        ],
    },
}
```

The screenshot shows a code editor with the following structure:

- EXPLORER**: Shows the project structure:
 - DJANGO-MODELS-WORKSHOP
 - alarms
 - __pycache__
 - logic
 - migrations
 - services
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - measurements
 - __pycache__
- ... (Ellipsis)
- models.py (selected tab)
- admin.py
- settings.py

The `models.py` file content is as follows:

```
from django.db import models
from measurements.models import Measurement

class Alarm(models.Model):
    name = models.CharField(max_length=50, default="Undefined Alarm")
    measurement = models.ManyToManyField(Measurement)

    def __str__(self):
        return '{}'.format(self.name)
```

3. Ejecución de aplicación:

```
...rchiecture-and-design/django-models-workshop on ✘ master [!?!] via 🐄 v3.18.6
→ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
measurements.Measurement: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
    HINT: Configure the DEFAULT_AUTO_FIELD setting or the MeasurementsConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
variables.Variable: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
    HINT: Configure the DEFAULT_AUTO_FIELD setting or the VariablesConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 2 issues (0 silenced).
August 20, 2022 - 13:49:44
Django version 3.2.6, using settings 'monitoring.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[20/Aug/2022 13:50:11] "GET / HTTP/1.1" 200 10697
[20/Aug/2022 13:50:11] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[20/Aug/2022 13:50:11] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[20/Aug/2022 13:50:11] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
[20/Aug/2022 13:50:11] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
Not Found: /favicon.ico
[20/Aug/2022 13:50:11] "GET /favicon.ico HTTP/1.1" 404 2114
[20/Aug/2022 13:50:15] "GET /admin HTTP/1.1" 301 0
[20/Aug/2022 13:50:15] "GET /admin/ HTTP/1.1" 302 0
[20/Aug/2022 13:50:15] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[20/Aug/2022 13:50:16] "GET /static/admin/css/base.css HTTP/1.1" 200 19513
[20/Aug/2022 13:50:16] "GET /static/admin/css/nav_sidebar.css HTTP/1.1" 200 2271
[20/Aug/2022 13:50:16] "GET /static/admin/css/login.css HTTP/1.1" 200 939
[20/Aug/2022 13:50:16] "GET /static/admin/css/responsive.css HTTP/1.1" 200 18545
[20/Aug/2022 13:50:16] "GET /static/admin/js/nav_sidebar.js HTTP/1.1" 200 1368
[20/Aug/2022 13:50:21] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 2377
[20/Aug/2022 13:50:23] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[20/Aug/2022 13:50:23] "GET /admin/ HTTP/1.1" 200 7178
[20/Aug/2022 13:50:23] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 380
[20/Aug/2022 13:50:23] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331
[20/Aug/2022 13:50:23] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
^[[
```

The screenshot shows the Django administration interface. On the left, there's a sidebar with links for 'ALARMS', 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users), 'MEASUREMENTS' (Measurements), and 'VARIABLES' (Variables). Each section has a '+ Add' and 'Change' button. On the right, there are two panels: 'Recent actions' which lists recent measurements and variables added, and 'My actions' which lists specific actions taken like adding 20.0% Measurement, 20.0% Measurement, 15.0 C Measurement, Oxygen Variable, Moisture Variable, and Temperature Variable.

4. Creación de varios registros de measurements:

Django administration

WELCOME, ZEJIRAN [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Home > Measurements > Measurements > 20.0 %

ALARMS

Alarms [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

MEASUREMENTS

Measurements [+ Add](#)

VARIABLES

Variables [+ Add](#)

Change measurement

20.0 %

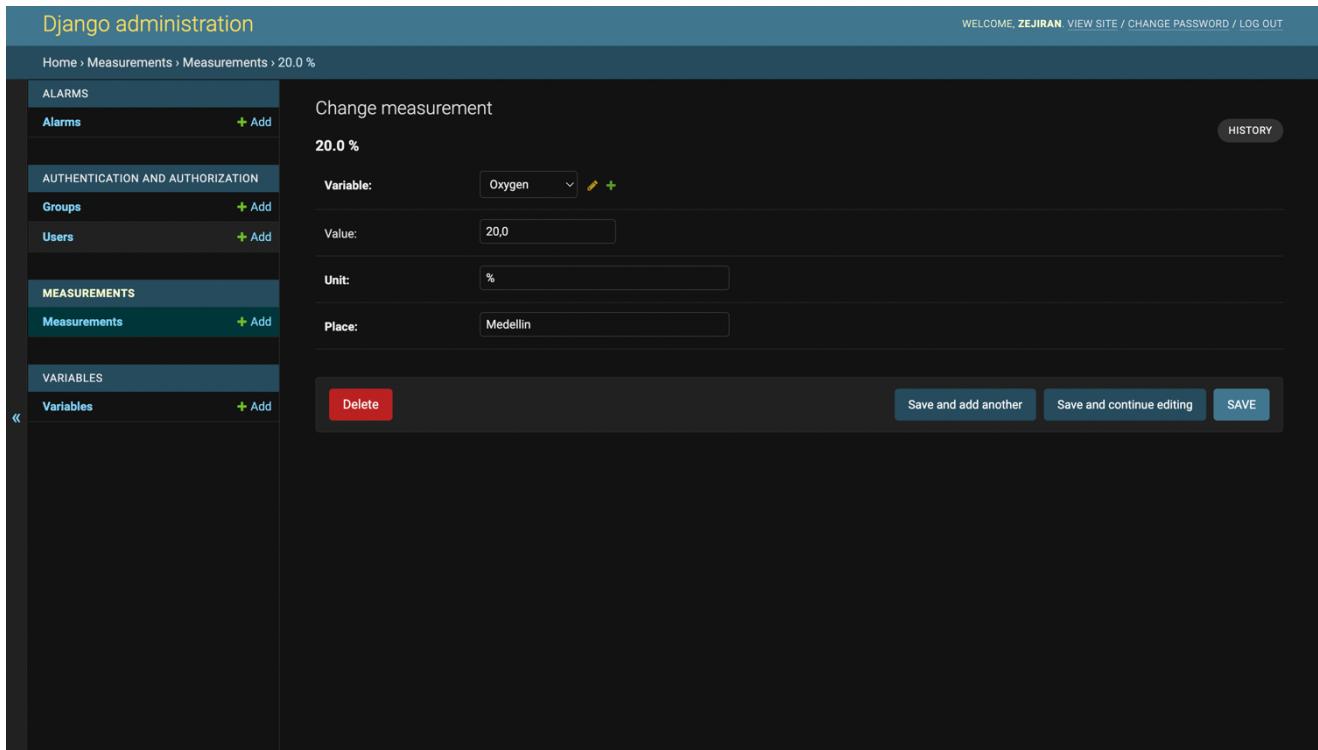
Variable: Oxygen [Edit](#) [+](#)

Value: 20,0

Unit: %

Place: Medellin

Delete [Save and add another](#) [Save and continue editing](#) [SAVE](#)



Django administration

WELCOME, ZEJIRAN [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Home > Measurements > Measurements

ALARMS

Alarms [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

MEASUREMENTS

Measurements [+ Add](#)

VARIABLES

Variables [+ Add](#)

Select measurement to change

Action: [—](#) Go 0 of 3 selected [ADD MEASUREMENT +](#)

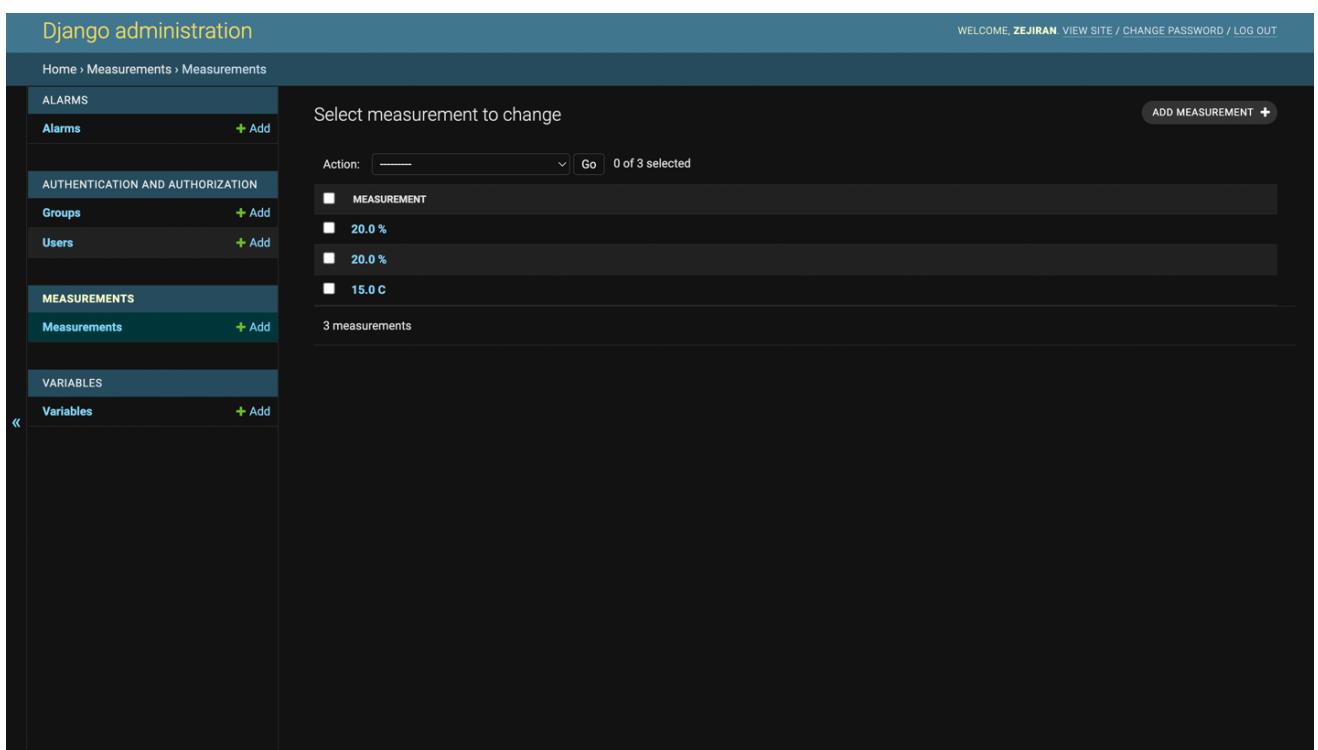
MEASUREMENT

20.0 %

20.0 %

15.0 C

3 measurements



5. Creación de registros de alarms:

Django administration

Home > Alarms > Alarms > Add alarm

ALARMS

Alarms [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

MEASUREMENTS

Measurements [+ Add](#)

VARIABLES

Variables [+ Add](#)

Add alarm

Name: Cali Alarm

Measurement: [15.0 C](#) [20.0 %](#) [20.0 %](#)

Hold down "Control", or "Command" on a Mac, to select more than one.

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

Django administration

Home > Alarms > Alarms

ALARMS

Alarms [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

MEASUREMENTS

Measurements [+ Add](#)

VARIABLES

Variables [+ Add](#)

The alarm "Cali Alarm" was added successfully.

Select alarm to change

Action: [——](#) Go 0 of 2 selected

ALARM

Cali Alarm

Bogotá Alarm

2 alarms

6. Relación ManyToMany en alarms:

En esta captura de pantalla se evidencia como la alarma “Cali Alarm” puede seleccionar varios measurements al mismo tiempo.

The screenshot shows the Django admin interface for the 'ALARMS' model. On the left, there's a sidebar with 'Groups' and 'Users' under 'AUTHENTICATION AND AUTHORIZATION', 'Measurements' under 'MEASUREMENTS', and 'Variables' under 'VARIABLES'. The main area is titled 'Change alarm' for 'Cali Alarm'. The 'Name:' field has 'Cali Alarm' entered. The 'Measurement:' field contains three selected items: '15.0 C', '20.0 %', and '20.0 %'. A note below says 'Hold down "Control", or "Command" on a Mac, to select more than one.' At the bottom, there are four buttons: 'Delete', 'Save and add another', 'Save and continue editing', and 'SAVE'.

Preguntas

- Investigue que es un ORM y defínalo con sus propias palabras.

R// Un ORM (Object Relational Mapping) es un software que permite transformar las queries de SQL a un sistema más similar a las estructuras de los lenguajes de programación orientados a objetos. De esta forma, se facilita la manipulación de las bases de datos con métodos o estructuras de datos previamente definidas sin tener que escribir directamente las sentencias de SQL, desde por ejemplo el ORM de Django.

A continuación se adjunta una captura de pantalla que muestra primero una query hecha con SQL y luego la misma query pero con el ORM Diesel (<https://diesel.rs/>):

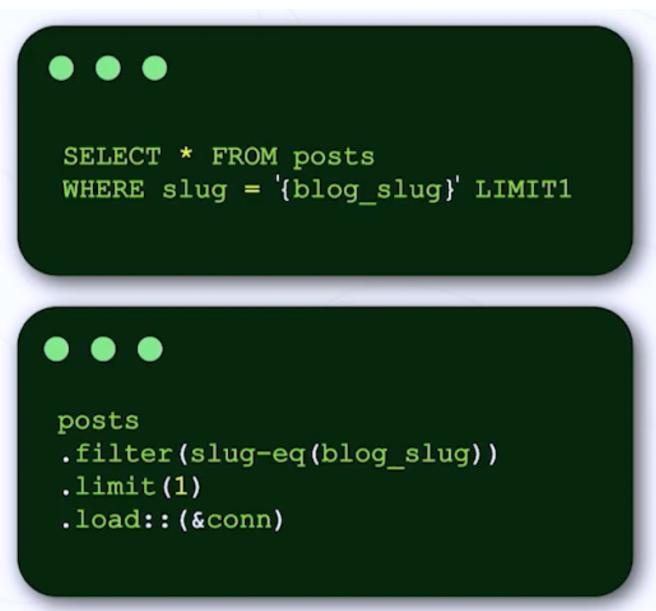


Figura 1. Comparación SQL vs. ORM

- Ventajas y desventajas de usar un ORM/ODM como el que trae integrado Django.

R//

Ventajas:

- Se genera una abstracción de la base de datos utilizada, causando que, independientemente de la elección de base de datos, el código en el ORM siempre será el mismo. De esta forma, no tendremos que realizar cambios si estamos usando el ORM de Django y deseamos cambiar de base de datos.
- Un ORM permite la reutilización del código al permitir la creación de objetos de datos que tienen métodos, utilizables incluso desde aplicaciones distintas.
- Un ORM evita que debamos conocer y dominar SQL, en cambio nos permite centrarnos en el lenguaje de backend que estemos utilizando, en nuestro caso Python junto a Django.

Desventajas:

- En entornos con una necesidad de alto rendimiento y manejo de grandes volúmenes de datos, utilizar un ORM podría agregar tiempos de espera entre procesos al requerir una capa extra de comunicación y por lo tanto mermar el rendimiento.
 - Aumenta la curva de aprendizaje y posiblemente el tiempo de desarrollo inicial al requerir que los programadores necesiten aprender la nueva sintaxis del ORM a utilizar.
 - Los desarrolladores que utilizan solamente los ORMs pueden tener una menor comprensión de lo que ocurre realmente por debajo y por lo tanto podrían llegar a tener menos control de los datos y la información.
- Consigne el significado de DTO y DAO y explique a cuál pertenece el ORM de Django.

R//

DTO: Estructura de datos que permite pasar información entre servicios o aplicaciones.

Un ejemplo de esto, es JSON que permite el intercambio de datos a través de gran parte de la web bajo un mismo estándar.

DAO: Es un patrón de diseño que permite proveer una interfaz a algún mecanismo de persistencia. Consiste en utilizar un objeto de acceso de datos para abstraer y encapsular todas las operaciones a la fuente de datos. De este modo, se encarga de manejar la conexión con la base de datos, ya sea para obtener o almacenar nuevos datos.

➔ El ORM de Django pertenece a DAO debido a que los modelos no son solamente objetos de datos, sino que permiten ejecutar todas las operaciones CRUD propias de la abstracción de los DAO a las bases de datos.

- Investigue 2 alternativas al ORM integrado de Django para bases de datos relacionales como PostgreSQL.

R//

- **SQLAlchemy:** utiliza la implementación Data Mapper que separa la estructura de la base de datos con la de los objetos, no como el ORM de Django que cada fila en la base de datos está directamente mappeada a un objeto en el código. Este ORM es recomendado de utilizar cuando se tiene muchas reglas de negocio y restricciones de aplicación, para que así se tenga mayor libertad de diseño. Por otro lado, el ORM de Django es recomendado cuando se desea avanzar rápidamente en el desarrollo en un negocio que solo necesite de operaciones CRUD sencillas y sin reglas complejas.
- **PonyORM:** es otra alternativa que en comparación a Django provee el patrón IdentityMap, el manejo automático de transacciones, el caching automático de objetos y queries, por último, ofrece el acceso a escribir queries de una manera más similar a SQL, junto a otras de las funcionalidades SQL que otros ORM no proveen como escribir SQL plano en caso de ser necesario.

Bibliografía

- <https://picodotdev.github.io/blog-bitix/2015/05/alternativa-a-hibernate-u-orm-y-ejemplo-de-jooq/>
- <https://www.hwlibre.com/orm-object-relational-mapping/>
- https://tutorial.djangogirls.org/es/django_orm/
- <https://java-white-box.blogspot.com/2014/02/ORMQueesunORM.html>
- <https://codigofacilito.com/articulos/orm-explicacion>
- <https://www.quora.com/What-is-the-difference-between-DAO-and-DTO>
- [https://rjcodeadvance.com/patrones-de-software-que-es-patron-de-diseno-parte-2/#:~:text=Patr%C3%B3n%20Data%20Access%20object%20\(DAO,para%20obtener%20y%20almacenar%20datos.](https://rjcodeadvance.com/patrones-de-software-que-es-patron-de-diseno-parte-2/#:~:text=Patr%C3%B3n%20Data%20Access%20object%20(DAO,para%20obtener%20y%20almacenar%20datos.)
- <https://www.eversql.com/django-vs-sqlalchemy-which-python-orm-is-better/>
- <https://docs.ponyorm.org/>