@fpftech.educacional

FPFtech
Escola Tecnológica

FPFtech
Escola Tecnológica

Explicando o Settings.py

É um arquivo do Django que contém diversas configurações, como por exemplo: Aplicações instaladas, configuração de banco de dados, middleware, timezone e outros.

settings.py

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'core',
]
```

settings.py

```python
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',
]
```

Explicando o urls.py

É um módulo responsável pelas rotas do Django

urls.py

```python
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('', include('core.urls'), name='clientes'),
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls')),
]
```

Explicando o wsgi.py

Web Server Gateway Interface – Interface de Porta de Entrada do Servidor Web é uma especificação para uma interface simples e universal entre *servidores* web e *aplicações* Web

wsgi.py

```python
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'temp_course.settings')

application = get_wsgi_application()
```

Explicando o wsgi.py

Imagine que você criou uma aplicação Django. Para que ela funcione na web, ela precisa se conectar com um servidor web, como o Gunicorn, uWSGI ou Apache (com mod_wsgi).
Mas:
O servidor web fala uma "língua" (HTTP).
O Django fala outra (Python).
O WSGI é o "intérprete" que conecta os dois. É uma ponte entre o servidor e o seu código Python.

Como funciona na prática?

- Um cliente (navegador) faz uma requisição HTTP.
- O servidor web recebe essa requisição.
- O servidor chama a função WSGI da sua aplicação Django (application no wsgi.py).
- O Django processa a requisição e retorna uma resposta (HTML, JSON, etc).
- O servidor web envia essa resposta para o navegador.

WSGI é a interface que permite que servidores web falem com aplicações Python, como o Django
Ele é necessário para colocar sua aplicação no ar de forma profissional.

Explicando o wsgi.py

Asynchronous Server Gateway Interface – Segue o mesmo raciocínio do WSGI porém para tarefas assíncronas.

asgi.py

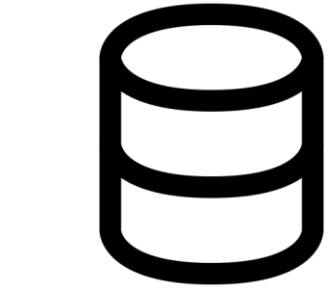
```python
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'temp_course.settings')

application = get_asgi_application()
```

Ciclo de Requests

FPFtech
Escola Tecnológica

GET https://dummyjson.com/users/1

```python
from rest_framework import serializers
from .models import MockUser

class MockUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = MockUser
        fields = ['id', 'username', 'email']
```



request

endpoint
urls

viewsets

Serializers
JSON=Django

models

database

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MockUserViewSet

router = DefaultRouter()
router.register(r'mock-users', MockUserViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```python
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```

```python
from django.db import models

class MockUser(models.Model):
    username = models.CharField(max_length=100)
    email = models.EmailField(unique=True)

    def __str__(self):
        return self.username
```

Entendendo o arquivo models.py

É onde é definida os modelos de dados da aplicação. Um modelo é uma classe Python que herda da biblioteca django.db.models.Model e cada modelo mapeia diretamente para uma tabela no banco de dados.

models.py

```python
from django.db import models

# Create your models here.

class Cliente(models.Model):

    name = models.CharField(

        db_column='tx_name',

        max_length=100,

        null=False

    )

    age = models.IntegerField(

        db_column='tx_age',

        null=False

    )
```

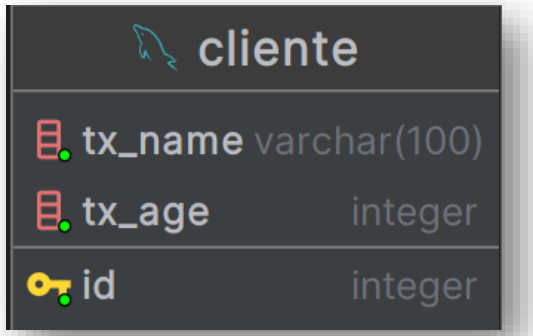


Tabela do banco de dados

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

Realizando uma Herança com um modelo Genérico.

```python
from django.db import models

class ModelBase(models.Model):
    id = models.BigAutoField(
        db_column='id',
        null=False,
        primary_key=True
    )
    created_at = models.DateTimeField(
        db_column='dt_created',
        auto_now_add=True,
        null=True
    )
    modified_at = models.DateTimeField(
        db_column='dt_modified',
        auto_now=True,
        null=True
    )
    active = models.BooleanField(
        db_column='cs_active',
        null=False,
        default=True
    )

    class Meta:
        abstract = True
        managed = True
```

Classe pai

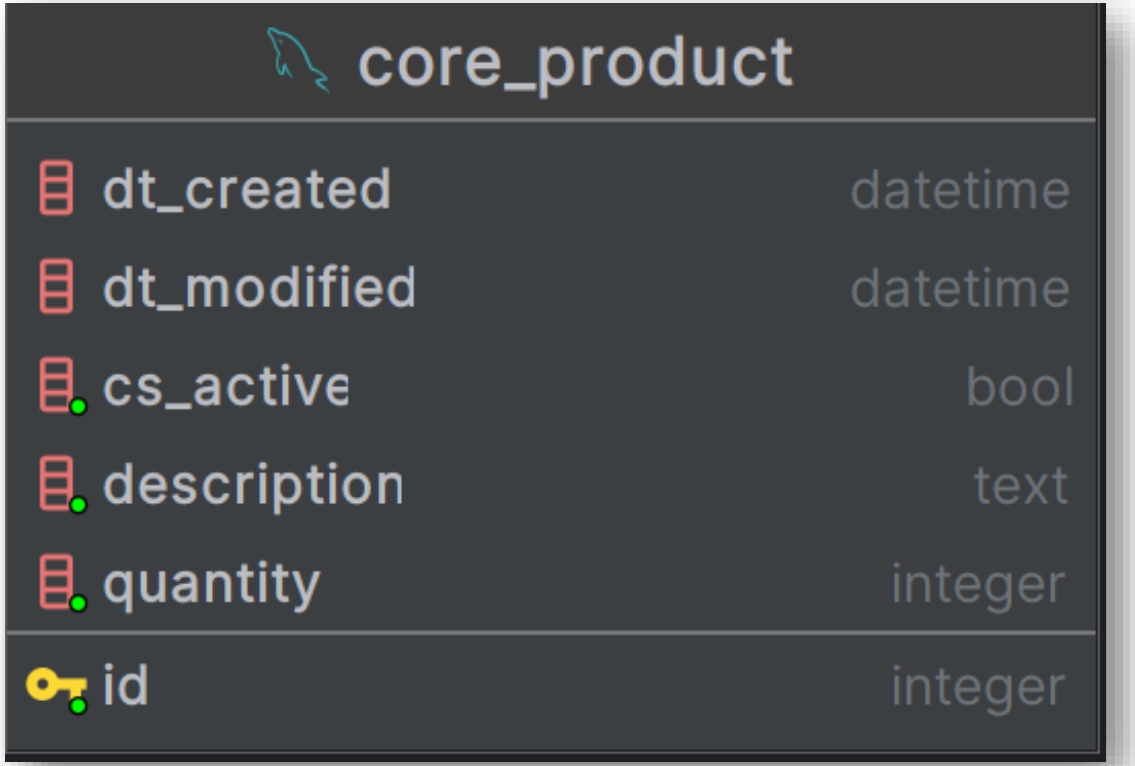
Classe filha

```python
class Product(ModelBase):

    description = models.TextField(
        db_column='description',
        null=False
    )
    quantity = models.IntegerField(
        db_column='quantity',
        null=False,
        default=0
    )
```

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

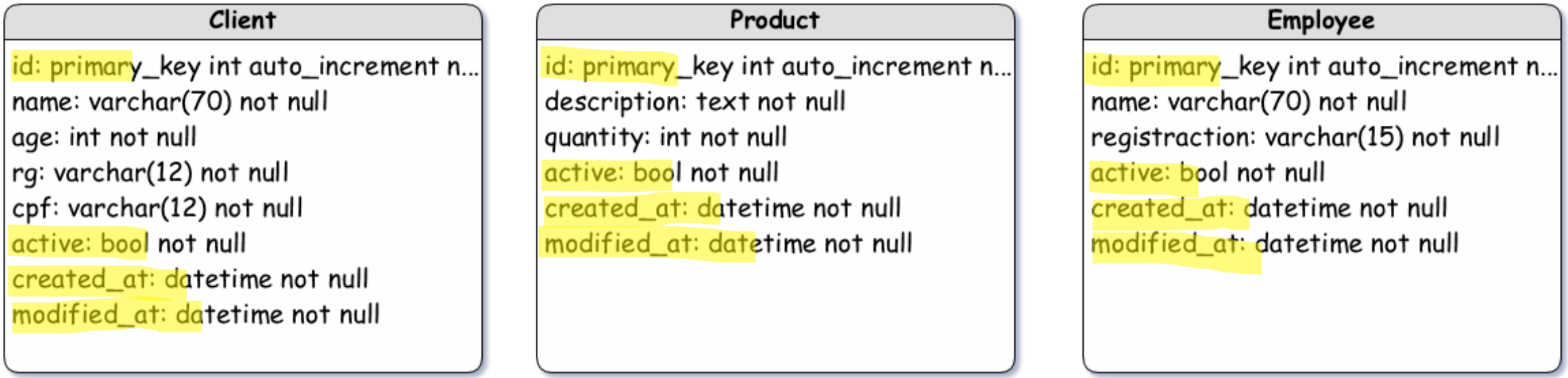
Resultado na tabela do banco de dados.



Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

Exercício: Dentro do seu aplicativo, crie uma classe abstrata Modelo Base com as colunas “id, active, created_at, modified_at” e herde nas classes abaixo:



Entendendo o arquivo models.py

```
from django.db import models

class ModelBase(models.Model):
    id = models.BigAutoField(
        db_column='id',
        null=False,
        primary_key=True
    )
    created_at = models.DateTimeField(
        db_column='dt_created',
        auto_now_add=True,
        null=True
    )
    modified_at = models.DateTimeField(
        db_column='dt_modified',
        auto_now=True,
        null=True
    )
    active = models.BooleanField(
        db_column='cs_active',
        null=False,
        default=True
    )

    class Meta:
        abstract = True
        managed = True
```

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

Exercício: Dentro do seu aplicativo, crie uma classe abstrata Modelo Base com as colunas “id, active, created_at, modified_at” e herde nas classes abaixo:

Client
id: primary_key int auto_increment n...
name: varchar(70) not null
age: int not null
rg: varchar(12) not null
cpf: varchar(12) not null
active: bool not null
created_at: datetime not null
modified_at: datetime not null

Product
id: primary_key int auto_increment n...
description: text not null
quantity: int not null
active: bool not null
created_at: datetime not null
modified_at: datetime not null

Employee
id: primary_key int auto_increment n...
name: varchar(70) not null
registraction: varchar(15) not null
active: bool not null
created_at: datetime not null
modified_at: datetime not null

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

```
from django.db import models

class ModelBase(models.Model):
    id = models.BigAutoField(
        db_column='id',
        null=False,
        primary_key=True
    )
    created_at = models.DateTimeField(
        db_column='dt_created',
        auto_now_add=True,
        null=True
    )
    modified_at = models.DateTimeField(
        db_column='dt_modified',
        auto_now=True,
        null=True
    )
    active = models.BooleanField(
        db_column='cs_active',
        null=False,
        default=True
    )

    class Meta:
        abstract = True
        managed = True
```

```
class Client(ModelBase):

    name = models.CharField(
        db_column='description’,
        max_lenght=70,
        null=False
    )
    age = models.IntegerField(
        db_column='age',
        null=False
    )
    rg = models.CharField(
        db_column= ‘rg’,
        max_lenght=12,
        null=False
    )
     cpf = models.CharField(
        db_column= ‘cpf’,
        max_lenght=12,
        null=False
    )
```

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

Exercício: Dentro do seu aplicativo, crie uma classe abstrata Modelo Base com as colunas “id, active, created_at, modified_at” e herde nas classes abaixo:

Client
id: primary_key int auto_increment n...
name: varchar(70) not null
age: int not null
rg: varchar(12) not null
cpf: varchar(12) not null
active: bool not null
created_at: datetime not null
modified_at: datetime not null

Product
id: primary_key int auto_increment n...
description: text not null
quantity: int not null
active: bool not null
created_at: datetime not null
modified_at: datetime not null

Employee
id: primary_key int auto_increment n...
name: varchar(70) not null
registraction: varchar(15) not null
active: bool not null
created_at: datetime not null
modified_at: datetime not null

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

```
from django.db import models

class ModelBase(models.Model):
    id = models.BigAutoField(
        db_column='id',
        null=False,
        primary_key=True
    )
    created_at = models.DateTimeField(
        db_column='dt_created',
        auto_now_add=True,
        null=True
    )
    modified_at = models.DateTimeField(
        db_column='dt_modified',
        auto_now=True,
        null=True
    )
    active = models.BooleanField(
        db_column='cs_active',
        null=False,
        default=True
    )

    class Meta:
        abstract = True
        managed = True
```

```
class Client(ModelBase):

    name = models.CharField(
        db_column='description’,
        max_lenght=70,
        null=False
    )
    age = models.IntegerField(
        db_column='age',
        null=False
    )
    rg = models.CharField(
        db_column= ‘rg’,
        max_lenght=12,
        null=False
    )
     cpf = models.CharField(
        db_column= ‘cpf’,
        max_lenght=12,
        null=False
    )
```

```
class Product(ModelBase):

    description = models.TextField(
        db_column=‘tx_descricao',
        null=False
    )
    quantity = models.IntegerField(
        db_column=‘nb_quantidade',
        null=False,
        default=0
    )
```

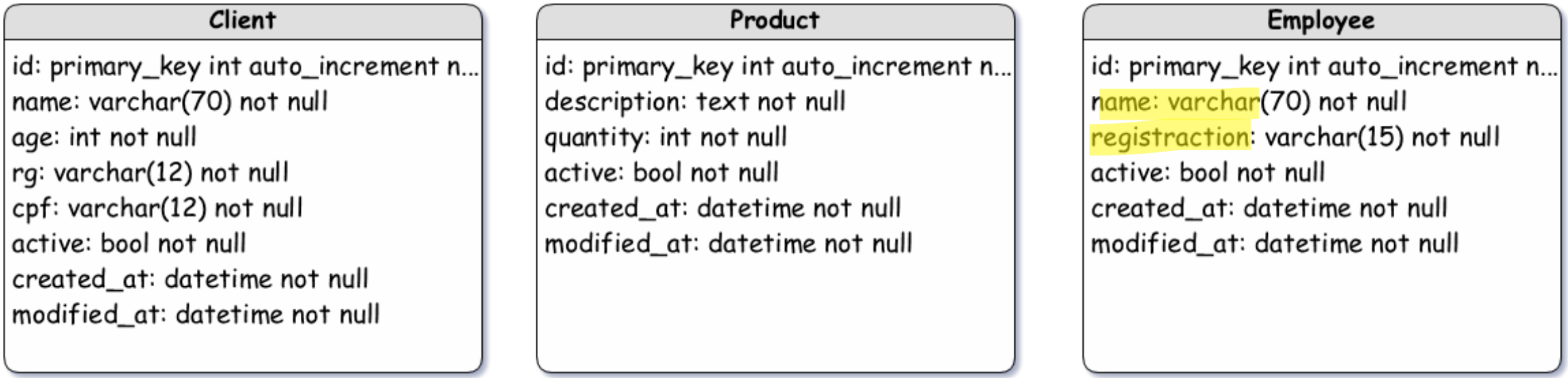
Diferença CharField e TextField

Característica	`CharField`	`TextField`
Tamanho máximo	**Obrigatório** (`max_length=...`)	**Opcional** (pode ser texto ilimitado)
Armazenamento	Usado para **strings curtas** (nome, CPF, email, etc.)	Usado para **textos longos** (descrições, observações, etc.)
Banco de dados	Geralmente vira um campo `VARCHAR(N)`	Vira um campo `TEXT` ou equivalente
Validação	Valida o tamanho no lado do Python	Não valida automaticamente o tamanho

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

Exercício: Dentro do seu aplicativo, crie uma classe abstrata Modelo Base com as colunas “id, active, created_at, modified_at” e herde nas classes abaixo:



Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

```
from django.db import models

class ModelBase(models.Model):
    id = models.BigAutoField(
        db_column='id',
        null=False,
        primary_key=True
    )
    created_at = models.DateTimeField(
        db_column='dt_created',
        auto_now_add=True,
        null=True
    )
    modified_at = models.DateTimeField(
        db_column='dt_modified',
        auto_now=True,
        null=True
    )
    active = models.BooleanField(
        db_column='cs_active',
        null=False,
        default=True
    )

    class Meta:
        abstract = True
        managed = True
```

```
class Client(ModelBase):

    name = models.CharField(
        db_column='description’,
        max_lenght=70,
        null=False
    )
    age = models.IntegerField(
        db_column='age',
        null=False
    )
    rg = models.CharField(
        db_column= ‘rg’,
        max_lenght=12,
        null=False
    )
     cpf = models.CharField(
        db_column= ‘cpf’,
        max_lenght=12,
        null=False
    )
```

```
class Product(ModelBase):

    description = models.TextField(
        db_column='description',
        null=False
    )
    quantity = models.IntegerField(
        db_column='quantity',
        null=False,
        default=0
    )
```

```
class Employee(ModelBase):

    name = models.CharField(
        db_column=‘tx_nome’,
        max_length=70,
        null=False
    )
    registration = models.CharField(
        db_column=‘tx_registro’,
        max_length=15,
        null=False
    )
```

Entendendo o arquivo models.py

Foreign Key: Chave de Referência de uma tabela em outra tabela. Como criar no models ?

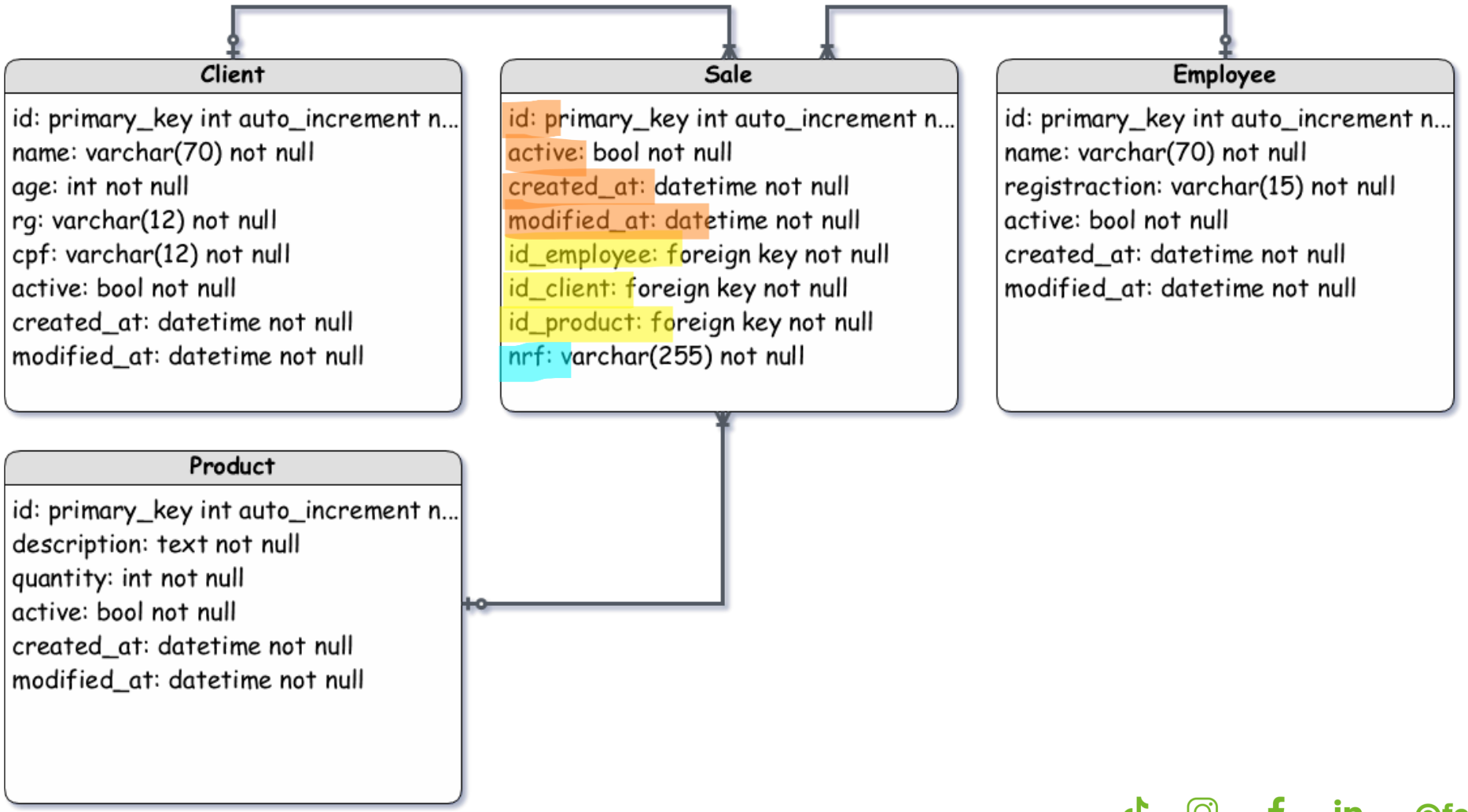
```
from django.db import models

class Author(ModelBase):
    name = models.CharField(max_length=100)

class Book(ModelBase):
        title = models.CharField(max_length=200)
        author = models.ForeignKey(
            Author,
            db_column='author_id',
            null=False,
            on_delete=models.DO_NOTHING
        )
```

```
id_employee: foreign key not null
id_client: foreign key not null
id_product: foreign key not null
nrf: varchar(255) not null
```

Entendendo o arquivo models.py

FPFtech
Escola Tecnológica

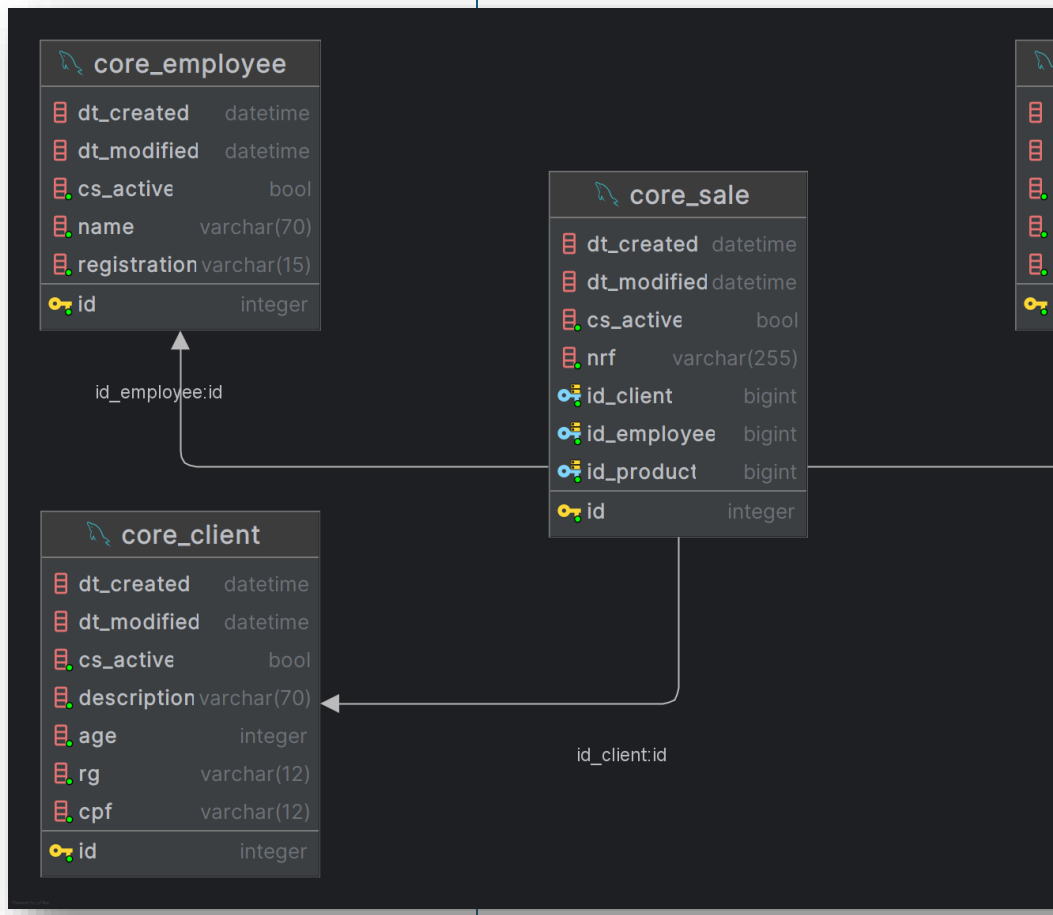
Exercício: Faça uma nova classe Sale(venda) e associe os ids dos demais nessa tabela:



```
class Sale(ModelBase):

    product = models.ForeignKey(
        Product,
        db_column='id_product',
        null=False,
        on_delete=models.DO_NOTHING
    )

    client = models.ForeignKey(
        Client,
        db_column='id_client',
        null=False,
        on_delete=models.DO_NOTHING
    )

    employee = models.ForeignKey(
        Employee,
        db_column='id_employee',
        null=False,
        on_delete=models.DO_NOTHING
    )
    nrf = models.CharField(
        db_column='tx_nrf',
        max_length=255,
        null=False
    )
```

Entendendo o arquivo models.py

```
class Sale(ModelBase):                                          )

    product = models.ForeignKey(
        Product,
        db_column='id_product',
        null=False,
        on_delete=models.DO_NOTHING
    )

    client = models.ForeignKey(
        Client,
        db_column='id_client',
        null=False,
        on_delete=models.DO_NOTHING
    )
    employee = models.ForeignKey(
        Employee,
        db_column='id_employee',
        null=False,
        on_delete=models.DO_NOTHING
    )

    nrf = models.CharField(
        db_column='tx_nrf',
        max_length=255,
        null=False
```



Salvar no banco postgres

FPFtech
Escola Tecnológica

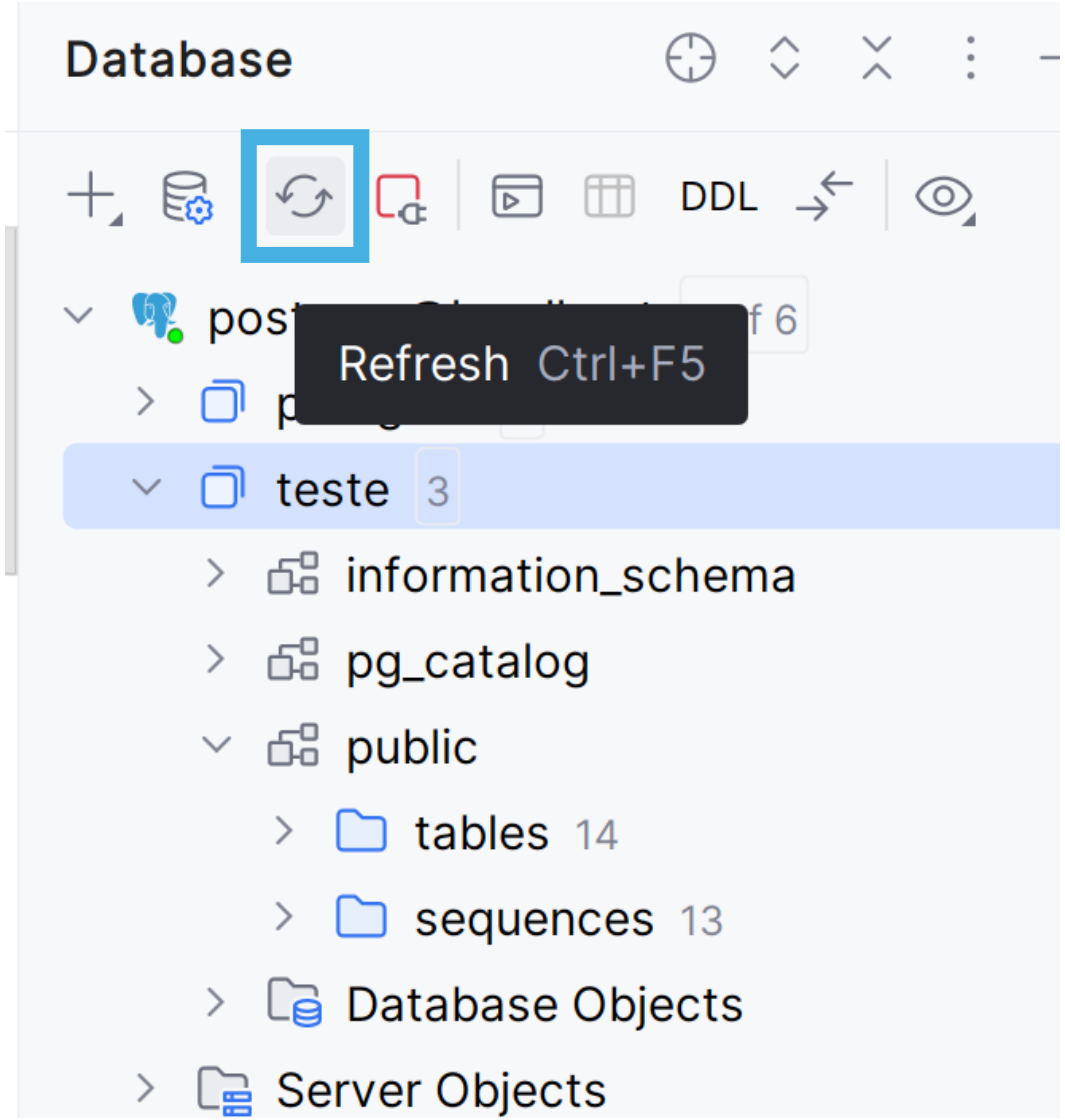
```
ects\djangoProject> python manage.py showmigrations
```

```
ts\djangoProject> python manage.py makemigrations
```

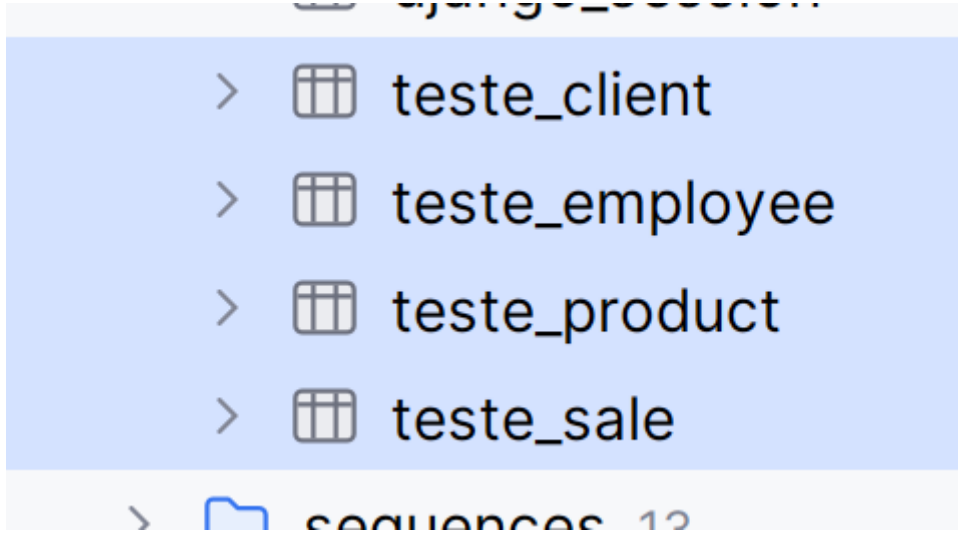
```
ts\djangoProject> python manage.py migrate
```

```
ects\djangoProject> python manage.py showmigrations
```

Ver alterações no banco de dados postgres.



Novas tabelas estão aqui (limpas ainda):



FPFtech
Escola Tecnológica

Obrigado!

FPFtech
Escola Tecnológica

@fpftech.educacional