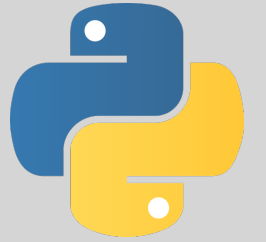




Django



Aplicação para controle de acesso a uma empresa ou a um condomínio

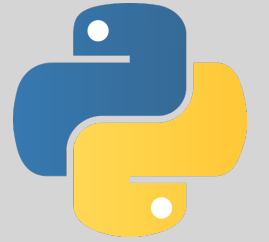
Defina um projeto chamado Controle de Visitantes

Após a definição do projeto crie uma aplicação chamada usuarios

A estrutura básica do projeto deve ser:

```
controle_visitantes
env
usuarios
```

Django



Definindo um modelo para os usuários

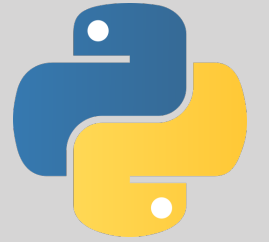
Uma camada de “modelo” em uma aplicação é uma parte da aplicação onde são definidas classes que são uma representação exata das tabelas do banco de dados.

No Django essa camada será definida na classe `models.py` dos aplicativos.

Antes de definir, vamos registrar o aplicativo no projeto
No arquivo `settings.py`, alterar a variável `INSTALLED_APPS`

```
INSTALLED_APPS += [  
    'usuários',  
]
```

Django

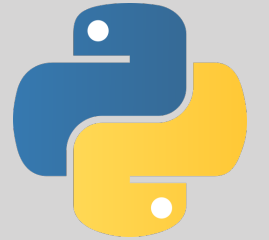


Escrevendo a classe de modelo (models.py)

```
from django.contrib.auth.models import(  
    BaseUserManager,  
    AbstractBaseUser,  
    PermissionsMixin,  
)
```

Este trecho de código realiza a importação das classes que serão usadas para criação do modelo.

Django



Escrevendo a classe de modelo (models.py)

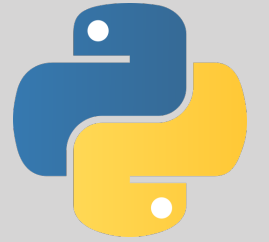
```
class Usuario(AbstractBaseUser, PermissionsMixin):  
    email = models.EmailField(  
        verbose_name = "E-mail do usuário",  
        max_length = 100,  
        unique = True,  
    )
```

verbose_name - atributo que define um label ou uma descrição para o campo

max_length - define o tamanho máximo que esse campo poderá armazenar

unique - define que o campo não poderá ter o seu valor duplicado

Django



Adicionando mais campos

Para que a classe `Usuario` possa ser usada para autenticação no sistema e ela deve ter alguns campos exigidos pelo Django para que a autenticação funcione.

`is_active`



Vai indicar se um usuário está ativo no sistema.

`is_staff`



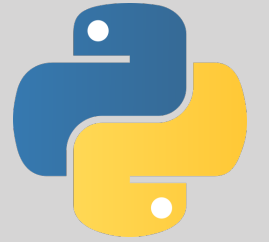
Vai indicar se um usuário faz parte da equipe de funcionários.

`is_superuser`



Vai indicar se um usuário tem poderes de administração.

Django

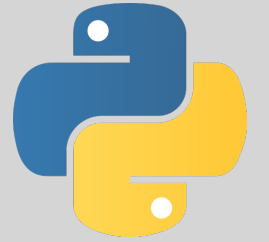


```
is_active = models.BooleanField(  
    verbose_name = "Usuário está ativo",  
    default = True,  
)
```

```
is_staff = models.BooleanField(  
    verbose_name = "Usuário é da equipe",  
    default = False,  
)
```

```
is_superuser = models.BooleanField(  
    verbose_name = "Usuário administrador",  
    default = False,  
)
```

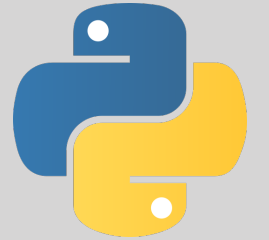
Django



Indicando para o Django o atributo que deve ser usado para autenticação

```
USERNAME_FIELD = "email"
```


Django



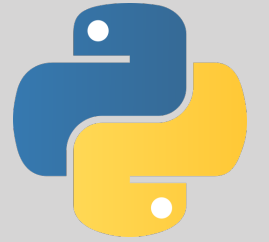
A classe Meta... Esta classe é comum a todos os modelos do Django e deve ser definida como uma classe interna. Ela será usada para definir algumas informações para o Django, os metadados.

```
class Meta:
    verbose_name = "Usuário"
    verbose_name_plural = "Usuários"
    db_table = "usuario"

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

O método `__str__` é obrigatório e será executado sempre que a classe for instanciada retornando um valor para fins de exibição.

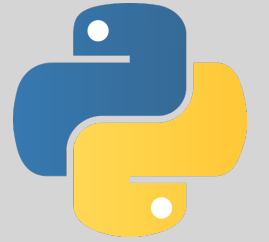
Django



Agora, a definição da classe manager, acima da classe de modelo

```
class UsuarioManager(BaseUserManager):  
  
    def create_user(self, email, password=None):  
        usuario = self.model(  
            email = self.normalize_email(email)  
        )  
  
        usuario.is_active = True  
        usuario.is_staff = False  
        usuario.is_superuser = False  
  
        if password:  
            usuario.set_password(password)  
  
        usuario.save()  
        return usuario
```

Django



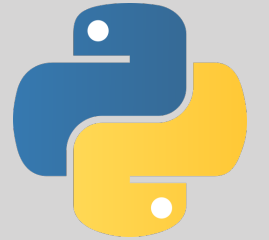
```
def create_superuser(self, email, password):
    usuario = self.create_user(
        email = self.normalize_email(email),
        password = password,
    )

    usuario.is_active = True
    usuario.is_staff = True
    usuario.is_superuser = True

    usuario.set_password(password)

    usuario.save()
    return usuario
```

Django



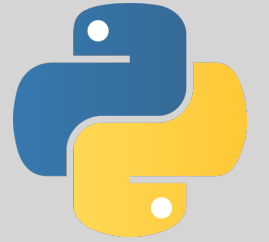
Com isso criamos um manager com a função de criar usuários e através da variável `objects` (que é usada pelo Django), informa-se na classe `Usuário` qual a classe que será responsável pelo manager.

```
objects = UsuarioManager()
```

O próximo passo é informar ao Django para ele utilizar essa classe como modelo para os usuários. Abra o arquivo `settings` e crie a variável:

```
AUTH_USER_MODEL = "usuarios.Usuario"
```

Django



Criando o banco de dados

```
python manage.py makemigrations usuarios
```

```
python manage.py migrate
```

Sempre que houver alteração ou criação de um modelo é necessário que estes comandos sejam executados.

Django



O primeiro usuário...

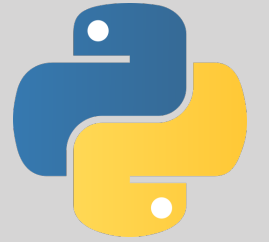
```
python manage.py createsuperuser
```

Após o cadastro do primeiro usuário...

```
python manage.py runserver
```

```
127.0.0.1:8000/admin
```

Django



O primeiro usuário...

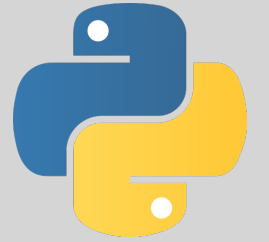
Abrir `admin.py` (não é necessário parar o servidor)

Importar a classe: `from usuarios.models import Usuario`

Adicionar: `admin.site.register(Usuario)`

Atualize o site

Django



Algumas configurações...

Alterando o idioma

Abrir o arquivo settings.py

Alterar LANGUAGE_CODE para pt-br

Alterando o fuso horário

Abrir o arquivo settings.py

Alterar TIME_ZONE para America/Sao_Paulo