

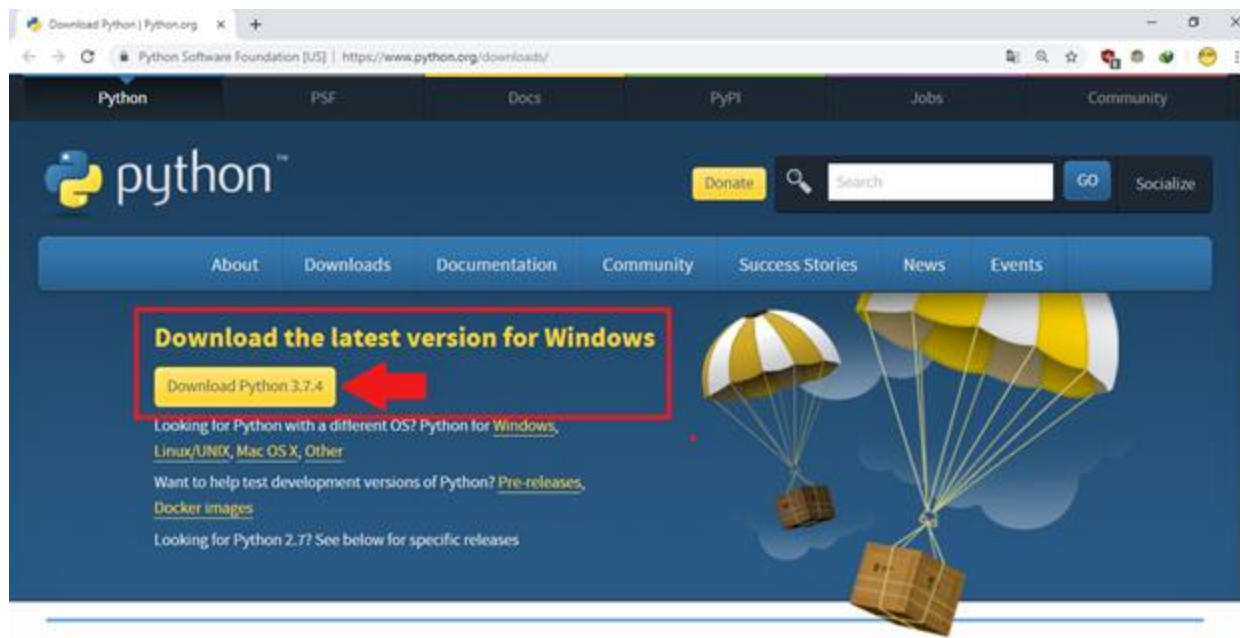
**Universidade Federal de Goiás**  
**Instituto de Informática**  
**Prof. Ronaldo Martins da Costa**





# Instalando o Python

- <http://python.org.br/instalacao-windows/>
- <https://python.org.br/instalacao-linux/>
- <https://python.org.br/instalacao-mac/>





# IMPORTANTE

- Algunas linguagens utilizam {} para identificar blocos de comandos
- O Python utiliza posição das colunas para definir blocos

```
def nome-da-funcao(parametros):  
    for a in range(1, 100):  
        if (a/10 == 5):  
            print("Valor de 50")  
        else:  
            print("Valor diferente de 50")  
        print("Abaixo do IF")  
    print("Fora do FOR")  
    return
```

```
function register()  
{  
    if (isEmpty($_POST)) {  
        $msg = '';  
        if ($_POST['user_name']) {  
            if ($_POST['user_password_new']) {  
                if ($_POST['user_password_new'] == $_POST['user_password_repeat']) {  
                    if (strlen($_POST['user_password_new']) > 5) {  
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {  
                            if (preg_match('/[a-z\d](2,64)/i', $_POST['user_name'])) {  
                                $user = read_user($_POST['user_name']);  
                                if (!isset($user['user_name'])) {  
                                    if ($_POST['user_email']) {  
                                        if (strlen($_POST['user_email']) < 65) {  
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {  
                                                create_user();  
                                                $_SESSION['msg'] = 'You are now registered so please login';  
                                                header('Location: ' . $_SERVER['PHP_SELF']);  
                                                exit();  
                                            } else $msg = 'You must provide a valid email address';  
                                        } else $msg = 'Email must be less than 64 characters';  
                                    } else $msg = 'Email cannot be empty';  
                                } else $msg = 'Username already exists';  
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';  
                        } else $msg = 'Username must be between 2 and 64 characters';  
                    } else $msg = 'Password must be at least 6 characters';  
                } else $msg = 'Passwords do not match';  
            } else $msg = 'Empty Password';  
        } else $msg = 'Empty Username';  
        $_SESSION['msg'] = $msg;  
    }  
    return register_form();  
}
```



# Ambiente Virtual

- Um ambiente virtual é uma ferramenta para “encapsular” um projeto, dependências e bibliotecas em um único lugar
- Este recurso permite você configurar um ambiente específico para o projeto e não interferir nas dependências de outros projetos
- O comando a seguir cria um ambiente virtual chamado `BigData-env`

```
python -m venv BigData-env
```





# Ambiente Virtual

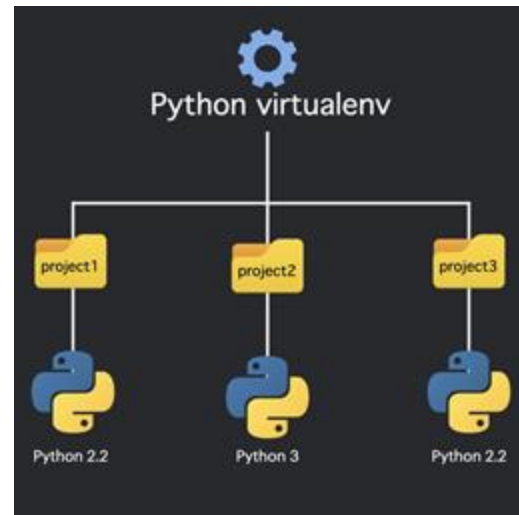
## ● Ativando um ambiente virtual

- No Windows

```
BigData-env\Scripts\activate.bat
```

- No Linux e MacOS

```
source BigData-env/bin/activate
```

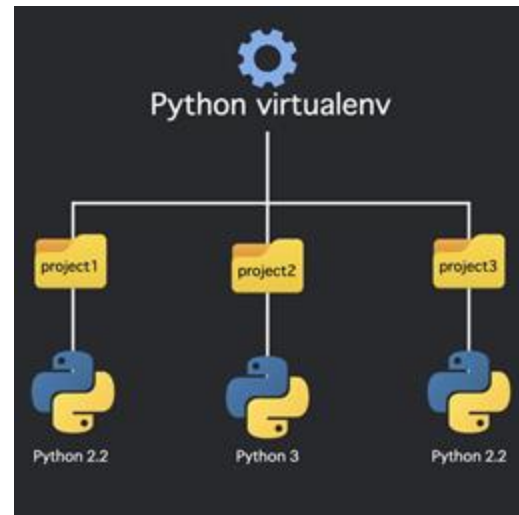




# Ambiente Virtual

## ● Desativando um ambiente virtual

```
deactivate
```





# Framework Django

- O Django é um framework que pode ser utilizado para desenvolver aplicações de forma rápida e eficiente
- Desenvolvedores precisam codificar funcionalidades semelhantes em cada aplicação que escrevem. O Django torna esse trabalho mais fácil, agrupando as diferentes funções em uma grande coleção de módulos reutilizáveis, chamados de framework de aplicações
- O framework Django é utilizado para organizar e escrever o código de maneira mais eficiente e para reduzir significativamente o tempo de desenvolvimento





## Verificando o ambiente

- Antes de iniciar as instalações é recomendado realizar uma verificação do ambiente existente
- Comando para verificar se existe alguma instalação prévia do django

```
python -m django --version
```

- Comando para atualizar o pip

```
python -m pip install --upgrade pip
```







## Instalando o django

- A instalação do django pode ser feita de forma bastante simples com o pip

```
pip install django
```





# Iniciando um projeto

- Iniciando um projeto com o nome **bdpratico**

```
django-admin startproject bdpratico
```

- A seguinte estrutura de pastas será criada

```
bdpratico
  bdpratico
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```





# Testando a instalação

django

View [release notes](#) for Django 4.1

- ☉ Dentro da pasta **bdpratico** execute o comando

```
python manage.py runserver
```

- ☉ Abra o navegador de sua preferência e execute

```
http://localhost:8000  
ou  
http://127.0.0.1:8000
```



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



**Django Documentation**  
Topics, references, & how-to's



**Tutorial: A Polling App**  
Get started with Django



**Django Community**  
Connect, get help, or contribute



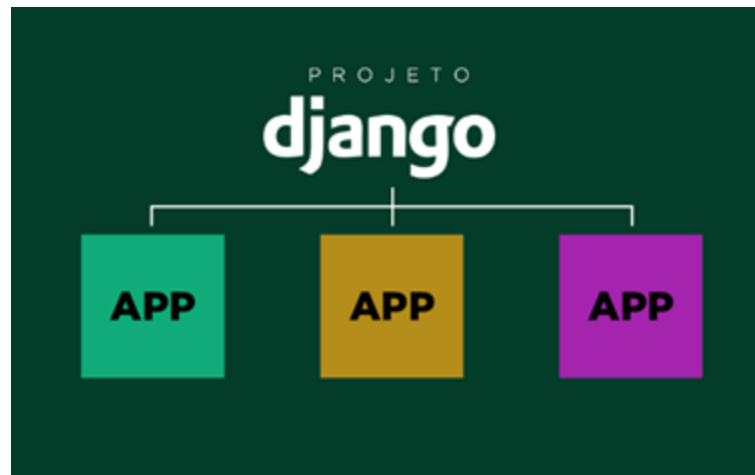
## Criando uma App

- Para criar um App dentro do projeto, basta utilizar o comando

```
python manage.py startapp exemplo01
```

- Para criar outro App dentro do mesmo projeto, basta utilizar o comando especificando o nome da nova App

```
python manage.py startapp exemplo02
```



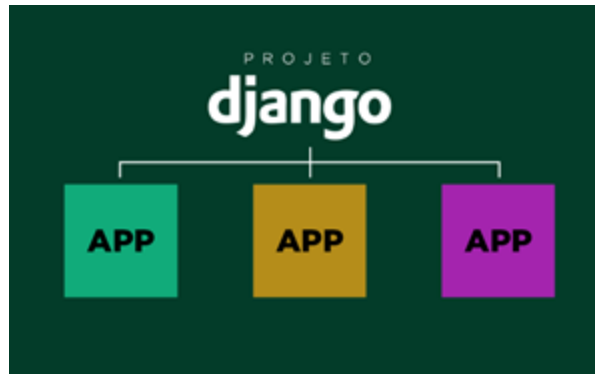


## Testando os App's

- Edite o arquivo `bdpratico/urls.py` deixando-o da seguinte forma:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('', include('exemplo01.urls')),
    path('exemplo01/', include('exemplo01.urls')),
    path('exemplo02/', include('exemplo02.urls')),
    path('admin/', admin.site.urls),
]
```





## Testando os App's

- Dentro da pasta bdpratico/exemplo01 crie um arquivo chamado urls.py deixando-o da seguinte forma:

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```





## Testando os App's

- Edite o arquivo views.py dentro da pasta bdpratico/exemplo01 deixando-o da seguinte forma:

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("EXEMPLO 01.")
```





## Testando os App's

- Dentro da pasta bdpratico/exemplo02 crie um arquivo chamado urls.py deixando-o da seguinte forma:

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```







## Testando os App's

- Edite o arquivo views.py dentro da pasta bdpratico/exemplo02 deixando-o da seguinte forma:

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("AGORA EH O EXEMPLO 02.")
```





## Pronto para Testar

● Agora é só testar

```
python manage.py runserver
```

```
http://localhost:8000/
```

```
http://localhost:8000/exemplo01/
```

```
http://localhost:8000/exemplo02/
```

☒ DONE!



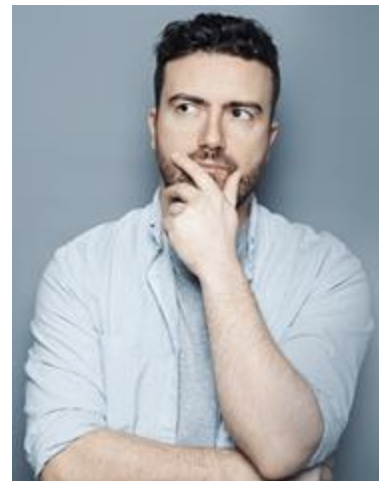
## Entendendo o que aconteceu...

```
python manage.py runserver
```

```
bdpratico
  bdpratico
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

1

- Ao executar o Django além de executar as configurações que estão dentro do arquivo settings.py, uma pré-compilação dos projetos é executada





## Investigando o Settings.py

```
DEBUG = True
```

- A opção **True** permite que erros e informações de Debug sejam exibidas no navegador quando elas ocorrerem
- Quando o projeto estiver sendo executado em ambiente de produção, é aconselhável que a variável seja *setada* para **False**





## Investigando o Settings.py

```
ALLOWED_HOSTS = []
```

- Dentro dos colchetes [] deve ser listada a relação de IP's ou endereços que terão acesso para execução dos projetos.
- Se você colocar ["\*"], o projeto estará aberto para qualquer pessoa que desejar executá-lo, desde que você possua um servidor de páginas (como por exemplo o apache) e um IP válido em sua máquina





## Investigando o Settings.py

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'exemplo01',  
    'exemplo02',  
]
```

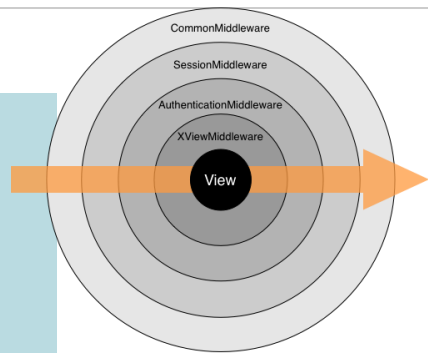


- Nesta lista devem estar todas as APP's que o projeto possuir
- O Django já traz algumas das APP's por default, por exemplo as do conjunto de aplicações do **contrib**



## Investigando o Settings.py

```
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',  
]
```

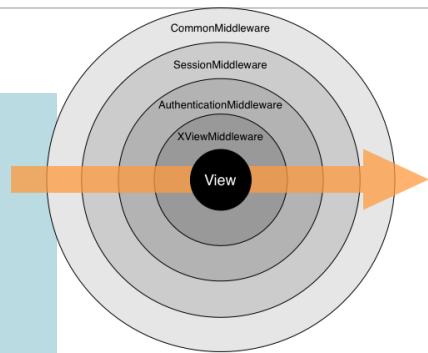


- O Middleware é uma lista que contém todos os **Hook's** dentro do processamento de requisição/resposta do Django. Ele é um sistema de *plugins* de entrada ou saída do Django



# Investigando o Settings.py

```
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',  
]
```



- 🟡 **Hook** consiste em uma série de técnicas utilizadas para modificar ou melhorar o comportamento de um componente de software através da interceptação de chamadas de funções, mensagens ou eventos passados entre componentes de software

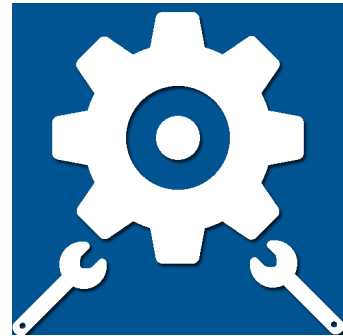




## Investigando o Settings.py

```
ROOT_URLCONF = 'bdpratico.urls'
```

- Especifica o primeiro arquivo de rotas (urls.py) a ser executado pelo projeto





# Investigando o Settings.py

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



- Aqui são especificados as aplicações que geram o HTML de sua aplicação dinamicamente



## Investigando o Settings.py

```
WSGI_APPLICATION = 'bdpratico.wsgi.application'
```

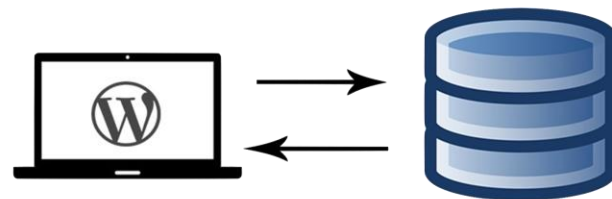
- WSGI (Web Server Gateway Interface) é uma especificação de interface que permite a comunicação entre o servidor e sua aplicação python
- Em português “Interface de Porta de Entrada do Servidor Web”, é uma especificação para uma interface simples e universal entre servidores web e aplicações web ou frameworks para a linguagem de programação Python





## Investigando o Settings.py

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```



- Configuração de conexão com o banco de dados do projeto, por *default* o Django traz a conexão com o sqlite
- Os parâmetros de conexão com o banco são:

```
'ENGINE': 'django.db.backends.oracle',  
'NAME': 'nome-do-usuario',  
'USER': 'user-name',  
'PASSWORD': 'senha',  
'HOST': '??????.??????.?????',  
'PORT': '?????',
```

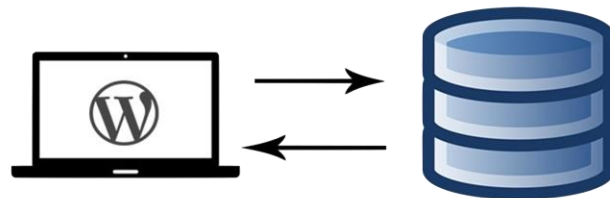


## Investigando o Settings.py

- A documentação do Django apresenta informações para conexão com os bancos de dados:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

- Além destas conexões nativas, é possível conectar a outros bancos de dados através de conexão ODBC





## Investigando o Settings.py

```
AUTH_PASSWORD_VALIDATORS = [  
    {  
        ...
```



- Apresenta as configurações padrão para validação de senha do Django



## Investigando o Settings.py

```
STATIC_URL = 'static/'
```

- Especifica a pasta onde ficarão armazenados os arquivos estáticos do projeto, como as imagens e os arquivos HTML





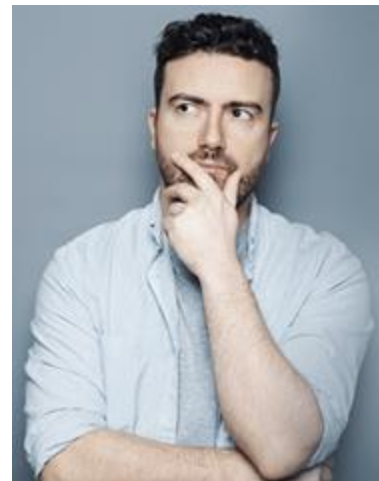
## Entendendo o que aconteceu...

```
python manage.py runserver
```

```
bdpratico
  bdpratico
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

2

- Após executado o settings.py, o próximo passo são os arquivos urls.py







## Rotas - urls.py

`http://localhost:8000`

```
bdpratico
  bdpratico
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

2

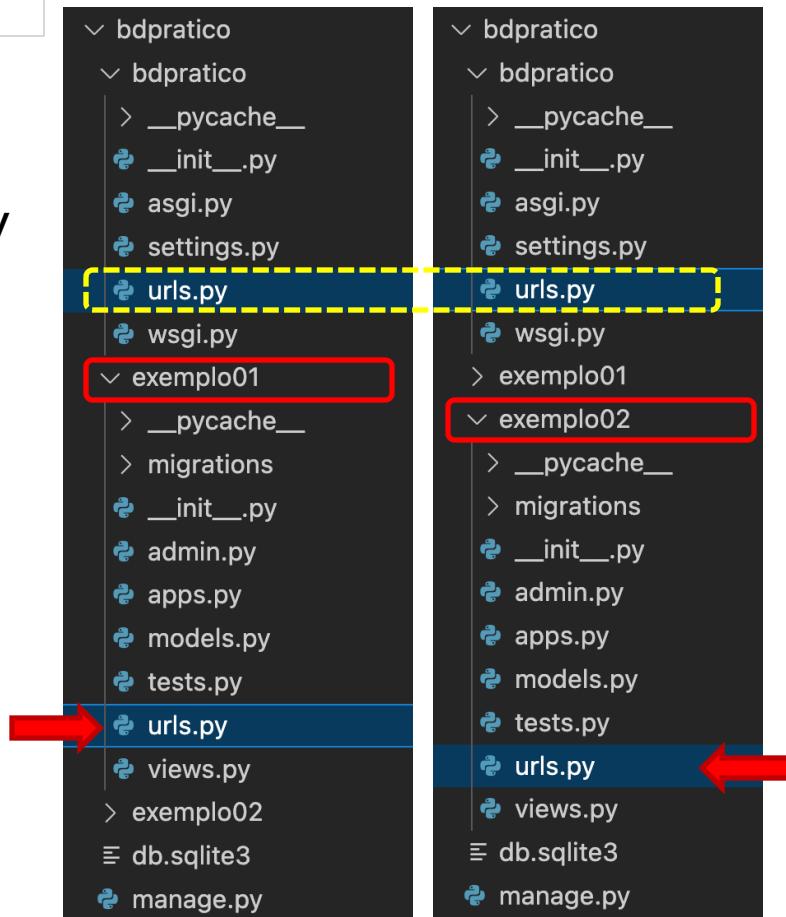
- Após executado o `settings.py`, o próximo passo são os arquivos `urls.py`





## Rotas - urls.py

- Seu projeto vai ter o arquivo **urls.py** na pasta base do projeto, que neste exemplo chamamos de bdpratico e um arquivo **urls.py** dentro de cada APP criada, como demonstrado na figura





## Rotas - urls.py

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('', include('exemplo01.urls')),
    path('exemplo01/', include('exemplo01.urls')),
    path('exemplo02/', include('exemplo02.urls')),
    path('admin/', admin.site.urls),
]
```

bdpratico/urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
]
```

bdpratico/exemplo01/urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
]
```

bdpratico/exemplo02/urls.py





## Entendendo o que aconteceu...

`http://localhost:8000/exemplo01/`

```
bdpratico
  bdpratico
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  exemplo01
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    admin.py
    urls.py
    views.py
  manage.py
```

3





## views.py

- Uma **view** é uma função do Python que recebe uma solicitação da Web e retorna uma resposta da Web
- Essa resposta pode ser o conteúdo HTML de uma página da Web, um redirecionamento, um erro 404, um documento XML ou uma imagem . . . ou qualquer coisa
- A **view** contém qualquer lógica necessária para retornar essa resposta

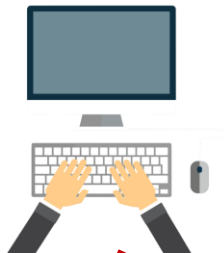
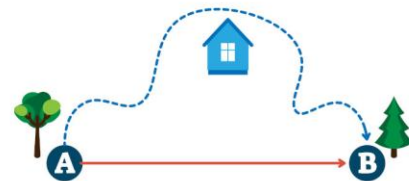


## O caminho até aqui...

```
from django.shortcuts import render
from . import views
from django.http import HttpResponse

def index(request):
    return HttpResponse("EXEMPLO 01.")
```

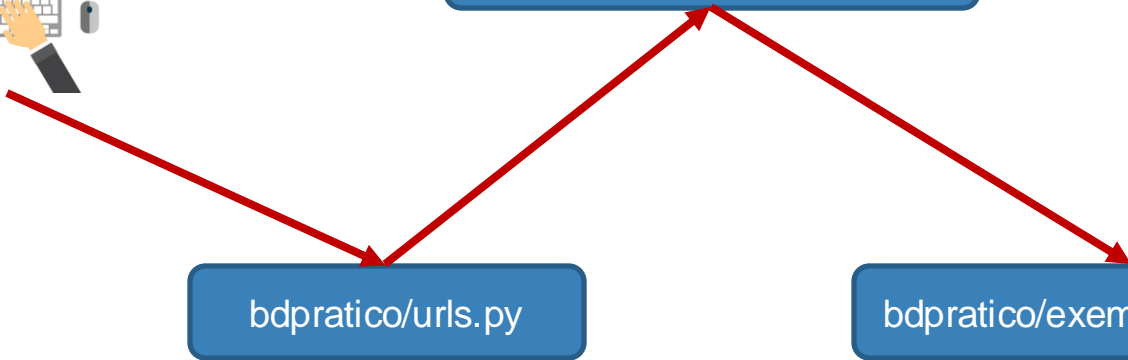
bdpratico/exemplo01/views.py



bdpratico/exemplo01/urls.py

bdpratico/urls.py

bdpratico/exemplo01/views.py





## Inicializando o DB

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions. Run **'python manage.py migrate'** to apply them.

Operations to perform:

Apply all migrations: admin, auth, contenttypes, sessions

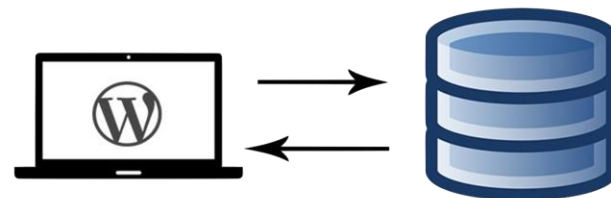
Running migrations:

Applying contenttypes.0001\_initial... OK  
Applying auth.0001\_initial... OK  
Applying admin.0001\_initial... OK  
Applying admin.0002\_logentry\_remove\_auto\_add... OK  
Applying admin.0003\_logentry\_add\_action\_flag\_choices... OK  
Applying contenttypes.0002\_remove\_content\_type\_name... OK  
Applying auth.0002\_alter\_permission\_name\_max\_length... OK  
Applying auth.0003\_alter\_user\_email\_max\_length... OK  
Applying auth.0004\_alter\_user\_username\_opts... OK  
Applying auth.0005\_alter\_user\_last\_login\_null... OK  
Applying auth.0006\_require\_contenttypes\_0002... OK  
Applying auth.0007\_alter\_validators\_add\_error\_messages... OK  
Applying auth.0008\_alter\_user\_username\_max\_length... OK  
Applying auth.0009\_alter\_user\_last\_name\_max\_length... OK  
Applying auth.0010\_alter\_group\_name\_max\_length... OK  
Applying auth.0011\_update\_proxy\_permissions... OK  
Applying auth.0012\_alter\_user\_first\_name\_max\_length... OK  
Applying sessions.0001\_initial... OK

A execução deste comando inicializa o DB criando as tabelas de controle utilizadas pelos APP's django



# Inicializando o DB



db

Name	Used
> auth_group (0)	3.13%
> auth_group_per...	3.13%
> auth_permission ...	3.13%
> auth_user (0)	3.13%
> auth_user_group...	3.13%
> auth_user_user_...	3.13%
> django_admin_lo...	3.13%
> django_content_...	3.13%
> django_migratio...	3.13%
> django_session (0)	3.13%
sqlite_master (29)	9.38%
sqlite_sequence ...	3.13%
freelist	0.00%

Database | **Data** | SQL | Design | HexWindow

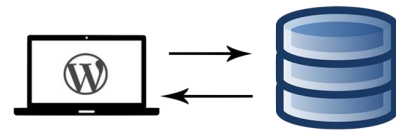
Filter in any column

id	name
----	------





## Criando uma Tabela (models.py)



- O arquivo models.py que fica dentro de cada APP criado faz toda a integração da camada dos Modelos de um projeto
- Altere o arquivo bdpratico/exemplo01/models.py com o conteúdo a seguir

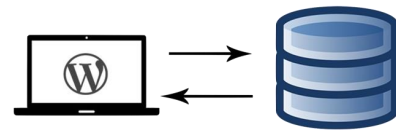
```
class pessoa(models.Model):
    nome = models.CharField(max_length=50, null=False, blank=False, verbose_name='Nome')
    email = models.CharField(max_length=50, null=False, blank=False, verbose_name='eMail')
    celular = models.CharField(max_length=20, null=True, blank=True, verbose_name='celular')
    funcao = models.CharField(max_length=30, null=True, blank=True, verbose_name='Funcao')

    def __str__(self):
        return self.nome

class Meta:
    ordering = ['nome']
```



## Criando uma Tabela (models.py)



- Tendo feito a atualização do arquivo models.py, basta executar o comando para criar a migração do modelo do respectivo APP para o DB

```
python manage.py makemigrations exemplo01
```

- Se o arquivo models.py estiver correto, o resultado deve ser:

```
python manage.py makemigrations exemplo01
Migrations for 'exemplo01':
  exemplo01/migrations/0001_initial.py
    - Create model pessoa
```

- Basta então executar o comando de migração do modelo para o DB

```
python manage.py migrate
```



## O Admin



- O APP de administração do Django pode usar os modelos para criar automaticamente uma área com o propósito de criar, visualizar, atualizar e excluir registros no DB
- O APP de administração por *default* já vem instalado no projeto

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'exemplo01',  
    'exemplo02',  
]
```



## O Admin



- Para começar a utilizar o APP de administração basta criar o usuário Admin
- **Lembre-se de memorizar a senha criada**

```
python manage.py createsuperuser
```

- Caso a senha seja muito simples (como eu fiz) o Django avisa porém não impede que seja criada

```
Username (leave blank to use 'ronaldocosta'): admin
Email address: ronaldocosta@ufg.br
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```



# O Admin

## Django administration

Username:

Password:



Log in

Site administration | Django site

localhost:8000/admin/

## Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

### Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add   Change

Users

+ Add   Change

Recent actions

My actions

None available



## O Admin

- Para definir que o APP de administração vai gerenciar a tabela pessoa criada, basta alterar o arquivo admin.py da APP exemplo01

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

- É necessário “importar” os modelos criados





# O Admin

## ● Visualizando o resultado



**Django administration**

Site administration

**AUTHENTICATION AND AUTHORIZATION**

<b>Groups</b>	<a href="#">+ Add</a>	<a href="#">Change</a>
<b>Users</b>	<a href="#">+ Add</a>	<a href="#">Change</a>

**EXEMPL001**

<b>Pessoas</b>	<a href="#">+ Add</a>	<a href="#">Change</a>
----------------	-----------------------	------------------------



## O Admin – Customizando...

- Para customizar a visualização de uma tabela no APP de administração, basta acrescentar algumas definições no arquivo admin.py



```
from django.contrib import admin
from .models import *

class PessoaCustomizado(admin.ModelAdmin):
    list_display = ('nome', 'email', 'celular', 'funcao', )

admin.site.register(pessoa, PessoaCustomizado)
```





# O Admin

## Visualizando o resultado



Django administration

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

Home > Exemplo01 > Pessoas

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

EXEMPLO01

- Pessoas + Add

Select pessoa to change

ADD PESSOA +

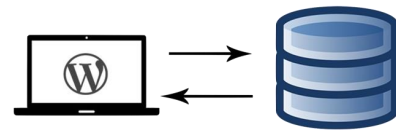
Action:  Go 0 of 4 selected

<input type="checkbox"/>	NOME	EMAIL	CELULAR	FUNCAO
<input type="checkbox"/>	Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109	Atendente
<input type="checkbox"/>	João da Silva	joasilva@gmail.com	(12) 34567-8901	Aluno
<input type="checkbox"/>	João da Silva	joasilva2@gmail.com	(12) 12121-1212	Duplicidade
<input type="checkbox"/>	Ronaldo Martins da Costa	ronaldocosta@gmail.com	(62) 98271-6303	Professor

4 pessoas



## Alterando uma tabela existente



- Altere o models.py acrescentando o campo “ativo” no modelo **pessoa**

```
class pessoa(models.Model):  
    nome = models.CharField(max_length=50, null=False, blank=False, verbose_name='Nome')  
    email = models.CharField(max_length=50, null=False, blank=False, verbose_name='eMail')  
    celular = models.CharField(max_length=20, null=True, blank=True, verbose_name='celular')  
    funcao = models.CharField(max_length=30, null=True, blank=True, verbose_name='Funcao')  
    ativo = models.BooleanField(default=True, verbose_name='Ativo')
```

- Lembre-se de migrar para o DB as alterações feitas

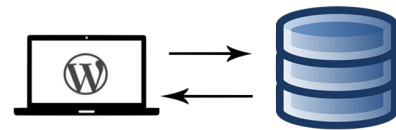
```
python manage.py makemigrations exemplo01
```

- E depois

```
python manage.py migrate
```



## Alterando uma tabela existente



- Tendo feito a atualização do arquivo `models.py`, basta executar o comando para criar a migração do modelo do respectivo APP para o DB

```
python manage.py makemigrations exemplo01
```

- Se o arquivo `models.py` estiver correto, o resultado deve ser:

```
python manage.py makemigrations exemplo01
Migrations for 'exemplo01':
  exemplo01/migrations/0001_initial.py
    - Create model pessoa
```

- Basta então executar o comando de migração do modelo para o DB

```
python manage.py migrate
```



# O Admin

## Visualizando o resultado



Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home › Exemplo01 › Pessoas

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- Groups** + Add
- Users** + Add

**EXEMPLO01**

- Pessoas** + Add

Select pessoa to change

ADD PESSOA +

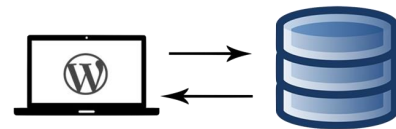
Action:  Go 0 of 4 selected

<input type="checkbox"/>	NOME	EMAIL	CELULAR	FUNCAO	ATIVO
<input type="checkbox"/>	Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109	Atendente	✓
<input type="checkbox"/>	João da Silva	joaosilva@gmail.com	(12) 34567-8901	Aluno	✓
<input type="checkbox"/>	João da Silva	joaosilva2@gmail.com	(12) 12121-1212	Duplicidade	✓
<input type="checkbox"/>	Ronaldo Martins da Costa	ronaldocosta@gmail.com	(62) 98271-6303	Professor	✓

4 pessoas



## Customizando a exibição



- É possível ordenar a exibição dos campos com mais de um campo/coluna da tabela, para tal, basta ajustar a class meta do arquivo models.py

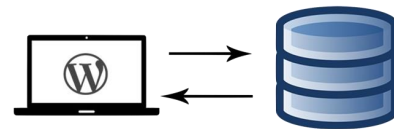
```
class pessoa(models.Model):
    nome = models.CharField(max_length=50, null=False, blank=False, verbose_name='Nome')
    email = models.CharField(max_length=50, null=False, blank=False, verbose_name='eMail')
    celular = models.CharField(max_length=20, null=True, blank=True, verbose_name='celular')
    funcao = models.CharField(max_length=30, null=True, blank=True, verbose_name='Funcao')
    ativo = models.BooleanField(default=True, verbose_name='Ativo')

    def __str__(self):
        return self.nome

class Meta:
    ordering = ['nome', 'funcao',]
```



## Customizando Campos



- Vamos criar mais um campo em nosso modelo chamado “**nascimento**” que armazenará a data de nascimento da pessoa
- Mas ao exibir na tela do admim vamos mostrar a idade calculada em anos referente a data de hoje

```
class pessoa(models.Model):
    nome = models.CharField(max_length=50, null=False, blank=False, verbose_name='Nome')
    email = models.CharField(max_length=50, null=False, blank=False, verbose_name='eMail')
    celular = models.CharField(max_length=20, null=True, blank=True, verbose_name='celular')
    funcao = models.CharField(max_length=30, null=True, blank=True, verbose_name='Funcao')
    nascimento = models.DateField(null=True, blank=True, verbose_name='Nascimento')
    ativo = models.BooleanField(default=True, verbose_name='Ativo')

    def __str__(self):
        return self.nome

    class Meta:
        ordering = ['nome', 'funcao',]
```

```
python manage.py makemigrations exemplo01
```

```
python manage.py migrate
```



# O Admin

## 🔴 Visualizando o resultado



Django administration WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) > [Exemplo01](#) > [Pessoas](#) > Add pessoa

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- [Groups](#) + Add
- [Users](#) + Add

**EXEMPLO01**

- [Pessoas](#) + Add

### Add pessoa

**Nome:**

**EMail:**

**Celular:**

**Funcao:**

**Nascimento:**  Today | ←

Note: You are 3 hours behind server time.

☒ Ativo

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



# Customizando Campos

- O arquivo admin.py deve ser ajustado para:



```
from django.contrib import admin
from .models import *

class PessoaCustomizado(admin.ModelAdmin):
    list_display = ('nome', 'email', 'celular', 'funcao', 'calcula_idade', 'ativo', )

    @admin.display(description='Idade')
    def calcula_idade(self, obj):
        from datetime import date
        hoje = date.today()
        idade = hoje.year - obj.nascimento.year
        return idade

admin.site.register(pessoa, PessoaCustomizado)
```





# Renderizando páginas HTML

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



Screenshot of the Django administration interface showing the 'Pessoas' (People) list. The browser address bar shows 'localhost:8000/admin/exemplo01/pessoa/'. The page title is 'Django administration'. The left sidebar shows the navigation menu with 'Pessoas' selected. The main content area displays a table of people with columns: NOME, EMAIL, CELULAR, FUNCAO, NASCIMENTO, IDADE, and ATIVO. A red arrow points to the bottom of the table, indicating the total count of 6 pessoas.

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Exemplo01 > Pessoas

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

EXEMPLO01

- Pessoas + Add

Select pessoa to change

Action: [dropdown] Go 0 of 6 selected

<input type="checkbox"/>	NOME	EMAIL	CELULAR	FUNCAO	NASCIMENTO	IDADE	ATIVO
<input type="checkbox"/>	Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109	Médico	Dec. 30, 1998	25	✓
<input type="checkbox"/>	Joao da Silva	joasilva2@gmail.com	(98) 76543-2109	Médico	July 9, 1975	48	✗
<input type="checkbox"/>	Joao da Silva	joasilva@gmail.com	(12) 34567-8901	Médico	May 6, 2002	21	✓
<input type="checkbox"/>	Nova Pessoa	novapessoa@gmail.com	2238729387	Nov Função	Jan. 1, 2023	0	✓
<input type="checkbox"/>	Outra Nova	outranova@gmail.com	7373463748	78643876	Jan. 23, 2023	0	✓
<input type="checkbox"/>	Ronaldo Martins da Costa	ronaldocosta@ufg.br	(62) 98271-6303	Professor	Aug. 27, 1971	52	✓

6 pessoas



# Permissões do Admin

- A APP de administração do Django fornece um sistema de usuários e grupos eficiente



Django administration WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Authentication and Authorization > Users > medico01

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- Groups [+ Add](#)
- Users [+ Add](#)
- EXEMPLO01
- Pessoas [+ Add](#)

**Change user** HISTORY

**medico01**

**Username:**   
Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

**Password:**   
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).


**Personal info**

**First name:**

**Last name:**

**Email address:**

**Permissions**

- ☒ **Active**  
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- ☒ **Staff status**  
Designates whether the user can log into this admin site. 
- ☐ **Superuser status**  
Designates that this user has all permissions without explicitly assigning them.



# Permissões do Admin

- Através desta interface é possível configurar as permissões de RWDX em cada modelo da aplicação



The screenshot shows the Django Admin interface for configuring permissions. It is divided into two main sections: 'Groups' and 'User Permissions'.

**Groups Section:**

- Available groups:** A list of groups with a search filter. The groups listed are 'Médicos', 'Pacientes', and 'Tecnologia de Informação'.
- Chosen groups:** An empty list for groups that have been selected.
- Buttons:** 'Choose all' and 'Remove all'.
- Text:** 'The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.'

**User Permissions Section:**

- Available user permissions:** A list of permissions with a search filter. The permissions listed are: 'auth | user | Can change user', 'auth | user | Can delete user', 'auth | user | Can view user', 'contenttypes | content type | Can add content type', 'contenttypes | content type | Can change content type', 'contenttypes | content type | Can delete content type', 'contenttypes | content type | Can view content type', 'exemplo01 | pessoa | Can add pessoa', 'exemplo01 | pessoa | Can delete pessoa', 'sessions | session | Can add session', 'sessions | session | Can change session', 'sessions | session | Can delete session', and 'sessions | session | Can view session'.
- Chosen user permissions:** A list of permissions that have been selected. The permissions listed are: 'exemplo01 | pessoa | Can change pessoa' and 'exemplo01 | pessoa | Can view pessoa'.
- Buttons:** 'Choose all' and 'Remove all'.
- Text:** 'Specific permissions for this user. Hold down "Control", or "Command" on a Mac, to select more than one.'



# Permissões do Admin

- ⦿ Ao configurar um usuário sem acesso para adicionar novos registros em um modelo, o botão Add é retirado da interface



Django administration

WELCOME, **PRIMEIRO NOME** [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Exemplo01 > Pessoas

Start typing to filter...


**EXEMPL001**

**Pessoas**

Select pessoa to change

NOME	1 ▲	EMAIL	CELULAR	FUNCAO	2 ▲	IDADE	ATIVO
Fulano de Tal		fulanotal@gmail.com	(98) 76543-2109	Atendente		48	✓
João da Silva		joaosilva@gmail.com	(12) 34567-8901	Aluno		21	✓
João da Silva		joaosilva2@gmail.com	(12) 12121-1212	Duplicidade		25	✓
Ronaldo Martins da Costa		ronaldocosta@gmail.com	(62) 98271-6303	Professor		52	✓

4 pessoas





## Permissões do Admin

- Configurando o usuário sem acesso para remover registros em um modelo, o botão Delete é retirado da interface



Change pessoa

**João da Silva** HISTORY

Nome:

E-mail:


Celular:

Funcao:

Nascimento:  Today |

Note: You are 3 hours behind server time.

☒ Ativo

 Save and continue editing SAVE



## Customizando Ações no Admin

- É possível criar ações customizadas para o APP Admin do Django
- Altere seu arquivo `admin.py` e veja os resultados





## Customizando Ações no Admin



```
from django.contrib import admin
from .models import *
```

```
@admin.action(description="Habilitar Registros Selecionados")
def habilitar_pessoas(ModelAdmin, request, queryset):
    for p in queryset:
        p.ativo = True
        p.save()
```

```
@admin.action(description="Desabilitar Registros Selecionados")
def desabilitar_pessoas(ModelAdmin, request, queryset):
    queryset.update(ativo=False)
```

```
class PessoaCustomizado(admin.ModelAdmin):
    list_display = ('nome', 'email', 'celular', 'funcao', 'nascimento', 'calcula_idade', 'ativo')
    actions = [habilitar_pessoas, desabilitar_pessoas]
```

```
@admin.display(description='Idade')
def calcula_idade(self, obj):
    from datetime import date
    hoje = date.today()
    idade = hoje.year - obj.nascimento.year
    return idade
```

```
admin.site.register(pessoa, PessoaCustomizado)
```



# O Admin

## Visualizando o resultado



Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Exemplo01 > Pessoas

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- Groups** + Add
- Users** + Add

**EXEMPLO01**

- Pessoas** + Add

Select pessoa to change

ADD PESSOA +

Action: ☒ ----- Go 0 of 4 selected

- Delete selected pessoas
- Habilitar Registros Seleccionados
- Desabilitar Registros Seleccionados

	CELULAR	FUNCAO	NASCIMENTO	IDADE	ATIVO
<input type="checkbox"/> Fulano da Silva	76543-2109	Aluno	Dec. 30, 1998	25	✓
<input type="checkbox"/> Joao da Silva	(12) 34567-8901	Diretor	May 6, 2002	21	✓
<input type="checkbox"/> Joao da Silva	(98) 76543-2109	Duplicidade	July 9, 1975	48	✓
<input type="checkbox"/> Ronaldo Martins da Costa	(62) 98271-6303	Professor	Aug. 27, 1971	52	✓

4 pessoas



**Universidade Federal de Goiás**  
**Instituto de Informática**  
**Prof. Ronaldo Martins da Costa**

