

# Envio de datos

- [API PGD](#)

# API PGD

## API PGD

O envio de dados (planos individuais) para o governo é feito por uma subaplicação externa ao sistema Polare disponível no endereço <https://github.com/djangonauta/dj-polare-pgd>. Essa aplicação também é utilizada para visualizar a formatação dos dados a serem enviados e pode ser utilizada para consultas e análise de planos individuais.

## Instalação

Como se trata de um aplicação [django](#) / [python](#) disponibilizada no [github](#), os seguintes passos para a instalação são necessários:

1. Instalação do interpretador python e as dependências do ambiente
2. Instalação do Git e clonagem do projeto
3. Instalação e configuração das dependências do Projeto

### Instalação do interpretador python e as dependências do ambiente

As versões mais novas de sistemas Linux (como [Ubuntu](#)) provavelmente já possuem o interpretador Python3 instalado. A aplicação citada anteriormente foi testada utilizando a versão mais recente da linguagem Python no momento que esta documentação é escrita (Versão 3.11.2).

Para gerenciar múltiplas versões do interpretador Python (incluindo as versões mais novas) recomendamos utilizar [Pyenv](#).

O comando a seguir instala as dependências (pacotes Python) do ambiente utilizando o gerenciador de pacotes [Pip](#):

```
pip3 install -U pip setuptools wheel bpython
```

### Instalação do Git e clonagem do projeto

O Git pode ser encontrado no endereço <https://git-scm.com/downloads>. Em sistemas linux (Ubuntu) o seguinte comando pode ser utilizado para instalá-lo:

```
sudo apt install git -y
```

Para clonar o projeto o seguinte comando pode ser utilizado:

```
git clone git@github.com: djangonauta/dj-polare-pgd.git
```

### Note

O comando anterior baixa e salva o projeto em um diretório chamado `dj-polare-pgd`

## Instalação e configuração das dependências do projeto

As configurações do projeto são feitas via variáveis de ambiente e/ou adicionando as seguintes variáveis no arquivo localizado em `dj-polare-pgd/projeto/settings/.env`:

```
SECRET_KEY=' cixd( 5rsi2$0h@z7m2juljajcm$f8c&yy7m7mzqtc73! +e@(- '
```

```
DATABASE_URL=' postgres: //postgres: admin@localhost: 5432/polaredb'
```

```
ADMINS=' admin=admin@domain. com'
```

```
EMAIL_URL=' consolemail: //: @'
```

```
CACHE_URL=' redis: //127. 0. 0. 1: 6379'
```

```
BROKER_URL=' amqp: //usuario: senha@localhost: 5672/projeto'
```

```
DISABLE_ACCOUNT_REGISTRATION=' False'
```

```
ACCOUNT_EMAIL_VERIFICATION=' none'
```

```
API_PGD_LOGIN_URL=' http: //hom. api. programadegestao. economia. gov. br /auth /jwt /login'
```

```
API_PGD_PLANO_TRABALHO_URL=' http: //hom. api. programadegestao. economia. gov. br /plano_trabalho'
```

```
API_PGD_CREDENCIAIS=' username=usuario@edu. br, password=senha'
```

As URLs referentes a configurações contendo usuário e senha possuem o seguinte formato:

`protocolo://usuario:senha@host:porta/recurso`.

A variável SECRET\_KEY é necessária para o funcionamento do framework Django especialmente relacionada com segurança. O seguinte site pode ser utilizado para gerar essa string: <https://djecrety.ir/>.

A variável DATABASE\_URL contém as configurações de acesso ao banco de dados postgresql. Recomendamos que inicialmente seja utilizada uma nova base (backup) do sistema Polare. A configuração apresentada tenta conectar em um banco de dados postgresql chamado `polaredb-backup` rodando em `localhost` na porta `5432` utilizando o usuário `postgres` e a senha `admin`. O responsável pela distribuição da aplicação deve alterar esses valores conforme as especificidades de sua instituição.

A variável ADMINS configura os usuários admin do sistema no formato `usuario1=usuario1@domain.edu.br,usuario2=usuario2@domain.edu.br`. Esses usuários normalmente são notificados por email quando erros acontecem no sistema.

A variável EMAIL\_URL configura o servidor de email utilizado para enviar mensagens. Essa aplicação no momento não trabalha enviando emails. A configuração apresentada configura o sistema para enviar mensagens para a linha de comando.

As variáveis API\_PGD\_\*\_URL são utilizadas para conexão e envio de dados para a API PGD do governo. As URLs apresentadas apontam para o ambiente local de desenvolvimento dessa api descrito em <https://github.com/economiagovbr/api-pgd>. A variável API\_PGD\_CREDENCIAIS deve ser definida conforme cadastrada no ambiente de desenvolvimento, ou obtida para o ambiente de produção. As credenciais de produção podem ser obtidas na documentação oficial <https://api-programadegestao.economia.gov.br/docs>

#### Note

Para configurar e levantar localmente a aplicação de envio de dados para testes da API PGD acesse a documentação oficial disponível em <https://github.com/economiagovbr/api-pgd>

#### Note

As credenciais (login e senha) obtidas junto ao ministério da economia podem ser utilizadas no ambiente de homologação online <http://hom.api.programadegestao.economia.gov.br/>.

As dependências do projeto são gerenciadas utilizando `pipenv`. Inicialmente deve ser criado o ambiente virtual a partir do diretório `dj-polare-pgd` utilizando o seguinte comando:

```
pipenv shell
```

### Note

O comando anterior também ativa o ambiente virtual.

Após a criação do ambiente virtual as dependências podem ser instaladas com o comando:

```
pipenv install --dev
```

### Note

No ambiente de produção, como as dependências de desenvolvimento não são necessárias, o comando seria `pipenv install`.

## Execução da aplicação

Com as dependências instaladas e as variáveis de ambiente configuradas é possível executar a aplicação no modo desenvolvimento com o comando:

```
invoke
```

### Note

O comando acima executa a task default pyinvoke chamada "run\_server" (definida no arquivo tasks.py). Muitas tasks pyinvoke servem para executar outros comandos por por conveniência. É possível executar a aplicação utilizando o comando padrão django `./manage.py runserver 0.0.0.0:8000 --settings projeto.settings.development`

A aplicação deve estar rodando em `localhost:8000`:

Api Root

# Api Root

OPTIONS

GET

The default basic root view for DefaultRouter

GET /api/v1/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "plano-individual": "http://localhost:8000/api/v1/plano-individual/",
  "plano-individual-proteg": "http://localhost:8000/api/v1/plano-individual-proteg/"
}
```

É necessário criar um usuário administrador para visualizar os endpoints contendo os planos individuais serializados no formato json.

Para criar um usuário administrador deve-se utilizar o seguinte comando e fornecer os dados que o utilitário necessita (usuário, email e senha):

```
./manage.py createsuperuser
```

## Note

O comando acima pode ser utilizado no mesmo terminal que executou a aplicação, mas para tanto a mesma precisa ser finalizada com o atalho no teclado Control + C

## Warning

É necessário que o ambiente virtual esteja ativado e com as dependências instaladas para que os comandos relacionados com a aplicação funcionem corretamente.

Com pelo menos um usuário admin criado é possível logar na aplicação clicando no link login no canto superior direito:

 images/login.png unknown

Após logar na aplicação é possível visualizar os planos individuais no formato json esperado pela API PGD do governo clicando no link <http://localhost:8000/api/v1/plano-individual/>:

Api Root / Listagem de Planos Individuais

## Listagem de Planos Individuais

Lista de Planos Individuais ativos no sistema Polare

Filtros

OPTIONS

GET ▾

« 1 2 3 ... 61 »

GET /api/v1/plano-individual/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "count": 902,
  "next": "http://localhost:8000/api/v1/plano-individual/?page=2",
  "previous": null,
  "results": [
    {
      "cod_plano": 2,
      "matricula_siape": "██████████",
      "cpf": "██████████",
      "nome_participante": "████████████████████████████████████████",
      "cod_unidade_exercicio": 11011801,
      "nome_unidade_exercicio": "████████████████████████████████████████",
      "modalidade_execucao": 2,
      "carga_horaria_semanal": 40,
      "data_inicio": "2022-12-12",
      "data_fim": "2022-12-31",
      "carga_horaria_total": 840,
      "horas_homologadas": 40,
      "atividades": [
        {
          "id_atividade": 2834,
          "nome_atividade": "Indício TCU",
          "faixa_complexidade": "ALTA",
          "tempo_presencial_estimado": 0,
          "tempo_presencial_programado": 0,

```

É possível fazer consultas de planos individuais clicando no botão “filter”. A aplicação busca pelas chaves `matricula_siape`, `nome_participante` e `modalidade_execucao` (presencial, remoto e híbrido - valores 1, 2 e 3 respectivamente).

## Envio de dados

O envio de dados para a API PGD do governo é feita utilizando o seguinte comando:

```
./manage.py enviar_dados
```

### Note

O comando anterior consulta todos os planos individuais ativos e faz solicitações http assíncronas utilizando as urls definidas na seção de configuração (`API_PGD_LOGIN_URL` e `API_PGD_PLANO_TRABALHO_URL`).

No momento que esta documentação é escrita ocorre um erro ao enviar os dados para o banco de dados da API PGD localmente (desenvolvimento). Esse erro acontece porque alguns código de unidade do instituto são muito longos, ultrapassando o limite do tipo de dados inteiro da coluna `cod_unidade_exercicio` da tabela `public.plano_trabalho` do banco `api_pgd`.

Para contornar esse problema localmente, o tipo de dados da referida coluna foi alterado de int para bigint utilizando a SQL a seguir (conectado no banco da aplicação de recebimento de dados api\_pgd):

```
api_pgd=# alter table public.plano_trabalho alter cod_unidade_exercicio type bigint;
```

#### Note

O problema citado anteriormente foi resolvido pela equipe da API PGD na *issue* <https://github.com/gestaogovbr/api-pgd/issues/72>

---

© Copyright 2023, UFRN/IFPA. Revision `ce91cf46`.

Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).