**DataViva**

Ambiente de Desenvolvimento

V0.1

Núcleo de Sistemas e Gestão

Abril 2013

Histórico de Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| DATA | VERSÃO | DESCRIÇÃO | AUTOR |
| 08/04/2013 | 1.0 | Criação do documento | Mario H. Mol |
| 31/10/2013 | 1.1 | Versão base | Mario H. Mol |

ÍNDICE

[1. INTRODUÇÃO 4](#_Toc371670114)

[2. Banco de Dados 5](#_Toc371670115)

[3. SCM 6](#_Toc371670116)

[4. INSTALAÇÃO 7](#_Toc371670117)

[4.1. Python 7](#_Toc371670118)

[4.1.1. PIP 7](#_Toc371670119)

[4.1.2. Dependências 8](#_Toc371670120)

[4.2. Flask 9](#_Toc371670121)

[4.3. Redis 10](#_Toc371670122)

[4.4. Ferramenta de Desenvolvimento 10](#_Toc371670123)

[4.5. Dependências 12](#_Toc371670124)

[4.6. Aplicativo 13](#_Toc371670125)

[4.7. FLASK 14](#_Toc371670126)

[5. DEPLOY 15](#_Toc371670127)

[6. CACHE 16](#_Toc371670128)

INTRODUÇÃO

Consideramos como ambiente de desenvolvimento todos os preparativos de máquina, processos, ferramentas e documentação para que o produto possa ser instalado para consulta de usuários ou para que o desenvolvedor possa trabalhar em alterações.

Neste caso o desenvolvedor utilizará uma máquina Windows, já que é o padrão da estação de trabalho para o Governo de Minas. O ambiente de produção será descrito tanto para ambiente Linux como para ambiente Windows, já que podemos ter servidores em diferentes sistemas operacionais.

Vamos iniciar falando de ferramentas utilizadas para suportar todo este ambiente de desenvolvimento e por fim vamos descrever os procedimentos de instalação na máquina de um desenvolvedor.

1. Banco de Dados

O banco de dados utilizado nesta aplicação é o *MySQL* 5.5.25a e nele tem a estrutura de dados utilizado pela aplicação construída em *Flask*. Este framework em si não precisa de tabelas de sistema no banco de dados, mas algumas tabelas genéricas foram criadas, como por exemplo, para autenticação de usuário. Portanto neste caso teremos tabelas de sistema e a massa de dados já calculados a partir das bases fornecidas pelo Estado.

Existe um driver próprio para conexão entre *Python* e o banco de dados *MySQL*. Este driver pode ser encontrado para diversos sistemas operacionais e normalmente só a instalação deste driver é o suficiente para fazer funcionar.

* http://sourceforge.net/projects/mysql-python/files/

Depois de instalado o driver devemos então fazer a instalação do banco de dados da aplicação no servidor. Existe uma cópia completa do banco de dados com informações de todos os produtos e sua origem de produção desde o ano de 2000. Para realizar desenvolvimentos utilizamos uma base de dados mais enxuta, com cerca de 1 GB, diferente dos 70 Gb que ocupa a base de dados com todas as informações.

De posse deste arquivo com extensão SQL, com as instruções para montar o banco de dados, execute o seguinte comando para realizar a instalação local:

* Primeiro acesse a pasta de Bin do seu Mysql Server, aqui a pasta de sugestão:
  + C:\Program Files\MySQL\MySQL Server 5.6\bin>
* Depois execute o comando para importar, da seguinte forma:
  + "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe" --user=root --password=123456 --port=3306 ps < DataMinas.sql
  + Fazer o mesmo para a base de dados mte\_tables.sql

SCM

Temos hoje um servidor virtualizado com a ferramenta *GITLAB*, que tem o controle de versionamento utilizando tecnologia *GIT* e também com uma interface Web para gerenciamento. Para mais informações de como utilizar o GIT e as ferramentas de apoio, consulte o documento Manual GIT.doc do Escritório de Prioridades Estratégicas.

O repositório pode ser acessado através do link:

* <https://github.com/alexandersimoes/dataviva>

Para realizar a atualização dos fontes diretamente pela fornecedora, a Growth Ventures, é necessário realizar uma conexão com o servidor:

* growth-ventures.com (204.9.221.112)

Exemplo para atualização das informações do repositório:

* git pull <https://github.com/alexandersimoes/dataviva.git> master
  + Existe uma branch master com o código do sistema, uma versão do dataviva completa
  + E uma branch chamada scripts para o processo de cálculo

Neste documento vamos utilizar a pasta padrão de instalação do DataViva conforme abaixo, portanto os demais exemplos que precisar acessar estes fontes partirão desta pasta:

* C:\Users\m1165564\Documents\Projetos\dataviva\source\www

INSTALAÇÃO

Neste capítulo teremos instruções de instalação das plataformas utilizadas nesta solução, bem como suas dependências.

* 1. Python

Vamos agora realizar o procedimento de instalação do *Python*, linguagem em que a plataforma *Flask* foi escrita. Realize o download da versão compatível *Python* 2.7.3 utilizando o site oficial do fornecedor:

* http://www.python.org

Descompacte o Python em uma pasta do Windows, tomamos como padrão:

* C:\Python27\

Neste caso é necessário adicionar no PATH do Windows a pasta do Python, para que ele possa ser instanciado de qualquer local:

* Acesse o "Meu Computador" e com o direito do mouse acesse "Propriedades"
* Entre na opção à esquerda chamada “Configurações avançadas do sistema”
* Acesse a opção Clique na guia (aba) "Avançados" e no botão "Variáveis de ambiente"
* Encontra a variável PATH e adicione ao final do texto as pastas C:\Python27; C:\Python27\Scripts;
* Adicione em variáveis de usuário as variáveis http\_proxy e https\_proxy com o valor <http://proxy.prodemge.gov.br:8080>
* Feche o CMD e abra novamente para executar a instalação do PIP.
  + 1. PIP

Existe um programa para realizar a instalação automática de módulos do Python, chamado PIP. Neste caso módulos são pequenos programas escritos em Python que realizar tarefas em particular, para exemplificar temos o módulo de MySQL, que faz a conexão entre o Python e bancos de dados da tecnologia MySQL. O PIP realiza o download do módulo em repositórios oficiais na internet, descompacta, compila e instala no computador em que é executado.

A instalação do PIP é realizada através do Easy Install, e para isso realize então o procedimento abaixo:

* Entre na pasta:
  + C:\Python27\Scripts
* Salve o seguinte arquivo nesta pasta:
  + <http://python-distribute.org/distribute_setup.py>
* Instale o Easy Install executando o comando:
  + python distribute\_setup.py
  + Se o proxy bloquear o download do conteúdo que será instalado, copie o endereço do arquivo no terminal, cole no browser e faça o download, copie o arquivo baixado para a pasta Scripts e execute o comando novamente.
* Por fim instale o PIP por meio do Easy Install
  + easy\_install pip

Com o PIP instalado é possível instalar os módulos executando o comando da seguinte forma:

* pip install nomeModulo

Vamos dizer que surgiu a seguinte mensagem ao executar a aplicação em Python:

* ImportError: No module named flask.ext.sqlalchemy
  + Para resolver basta instalar o módulo requisitado, como no comando abaixo:
  + pip install SQLAlchemy

Caso não seja possível baixar o programa, talvez por falta de conectividade ou pelo fato do proxy barrar, poderá aparecer a seguinte mensagem:

* Download error on http://github.com/mitsuhiko/flask-babel: timed out -- Some packages may not be found!

Neste caso você pode entrar no link fornecido na mensagem de erro, baixar o módulo e instalar manualmente como no exemplo do comando abaixo:

* pip install path\_to\_file

Para baixar o módulo é necessário ter um proxy de internet configurado no computador, para isso pode ser necessário configurar da seguinte forma:

* Exemplo de uso de proxy com usuário e senha
  + pip install lib --proxy http://user:pass@proxyAddress:proxyPort
    1. Dependências

Temos aqui as bibliotecas necessárias a serem instaladas no Python, para suportar a aplicação.

* pip install python-openid
* pip install Mako
* pip install pytz
* pip install blinker
* pip install httplib2
* pip install flup
* pip install alembic
  + # http://bitbucket.org/zzzeek/alembic
* pip install oauth2
  + # <http://github.com/simplegeo/python-oauth2>
* Install pywin32

Para a comunicação com o banco de dados, inicialmente podemos tentar utilizar o PIP:

* pip install MySQL-python

Para a instalação do MySQLdb é possível realizar através do pip:

* pip install MySQLdb

Caso mesmo após a instalação a aplicação retornar erro de não encontrar o módulo MySQLdb, é possível tentar instalar utilizando uma versão já compilada:

* soemin.googlecode.com/files/MySQL-python-1.2.3c1.win32-py2.7.exe

A última opção é compilar o módulo na máquina. Para isso instale o MinGW no C: e utilizando o instalador no endereço:

* http://sourceforge.net/projects/mingw/files/

Depois adicione no PATH do Windows a pasta:

* C:\MinGW\bin

Depois descompacte os fontes do módulo MySQLdb e execute o comando abaixo:

Existe também bibliotecas utilizadas para a exportação:

* http://ftp.gnome.org/pub/gnome/binaries/win32/librsvg/2.26/

A instalação do componente de banco de dados pode ocasionar erros, principalmente em ambientes Windows. Para isso existe um pacote do MySQL para o Python, compatível com Windows 64 Bits, que pode ser instalado separadamente, ou seja, sem o PIP.

* <http://www.codegood.com/archives/129>
  + Instalar em máquinas 64 bits o arquivo de 32 bits, pois o de 64 bits dá problema por não reconhecer o path do Python, caso o python instalado na máquina seja a versão 32bit.
  + MySQL-python-1.2.3.win32-py2.7.exe

Em ambientes linux a instalação deste componente pode se feito com o comando:

* apt-get install python-mysqldb (Linux Ubuntu, ...)
  1. Flask

O *Flask* utilizando neste ambiente é o mesmo de produção, versão 0.9, e a instalação deve ser realizada através do PIP, que foi explicado em capítulos anteriores.

Para instalar o Flask execute da seguinte forma:

* pip install Flask

Mais informações sobre as extensões do Flask podem ser encontradas em:

* http://flask.pocoo.org/extensions/

Os módulos necessários são:

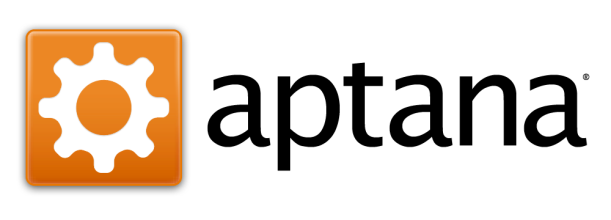
* pip install Flask-SQLAlchemy
* pip install Flask-Login
* pip install Flask-OpenID
* Localization
  + pip install Flask-Babel
* Utilidades para criar formulários (Form helper)
  + pip install Flask-WTF
* Indexação de texto complete para o banco de dados
  + pip install Flask-whooshalchemy
  + Houve casos que o o PIP não conseguiu instalar este plugin, sendo assim é necessário baixar diretamente do git
    - <https://github.com/gyllstromk/Flask-WhooshAlchemy>
    - E depois rodar o comando apontando para este arquivo
      * Pip install arquivo.zip
* pip install Flask-Oauth
* pip install Flask-Mail
  + https://bitbucket.org/danjac/flask-mail
* pip install Flask-Script
  + http://github.com/techniq/flask-script
* Flask-alembic connector
  + https://github.com/tobiasandtobias/flask-alembic.git
  + pip install flask-alembic
  1. Redis

Instalação dos plug-ins do Redis para o Python, mas antes confirme que o Redis está instalado, vide capítulo 8.

* pip install redis
  1. Ferramenta de Desenvolvimento

Existem alguns programas para que o desenvolvedor possa utilizar para auxiliar na manutenção do código fonte. Optamos por utilizar como padrão o *Aptana*, por ter diversos recursos de auxílio ao desenvolvedor, é compatível com *Python*, é gratuito e tem um vasto número de usuários ativos.

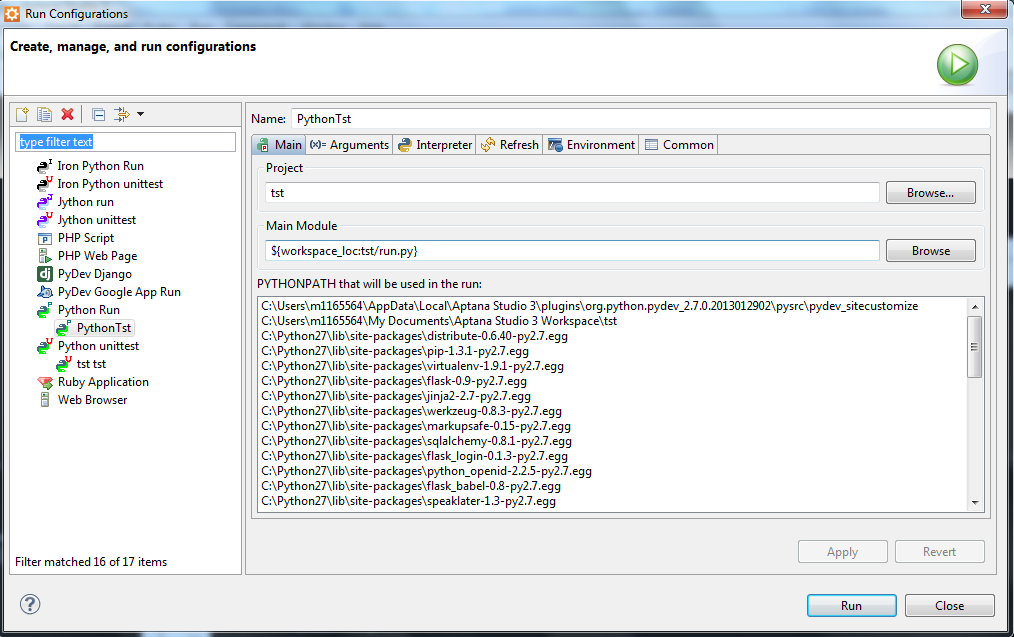
* <http://www.aptana.com/>

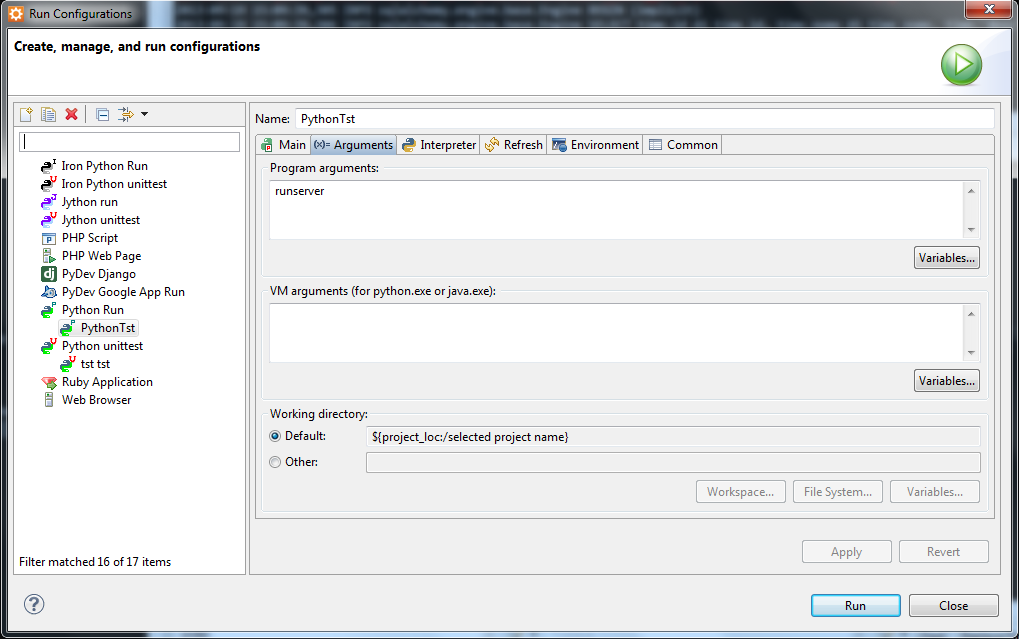


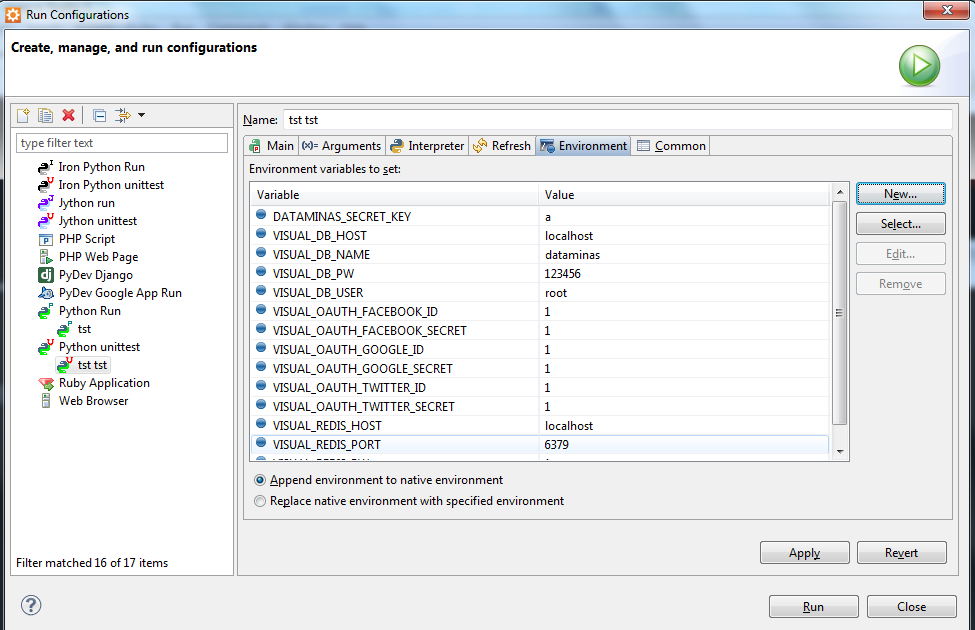
A versão do *Aptana Studio 3* tem todos os módulos importantes para trabalhar neste projeto por padrão e funciona muito bem em Windows, portanto baixe na página do fornecedor e realize a instalação. Com o *Aptana* aberto acesso o menu File -> Import Project e abra a pasta *WWW* que está dentro do diretório com os fontes do projeto, descrito no capítulo sobre SCM. Veja que na pasta raiz do projeto existe um arquivo chamado .project e outro .pydevproject.

Estes arquivos contem configurações para abrir um projeto *PyDev*, mas caso você não tenha estes arquivos, basta criar um novo projeto no *Aptana* como *PyDev* e então colar os arquivos da pasta WWW para a pasta deste projeto criado.

Para executar o projeto, basta acessar Run -> Run Configurations e adicionar um Python Run, considerando as configurações ilustradas nos print’s abaixo:







Quando for necessário para o serviço, utilizar o parar do Eclipse não necessariamente mata o processo do servidor python que foi iniciado. Para isso, é necessário abrir o gerenciador de tarefas e matar o processo manualmente.

* 1. Dependências

Algumas bibliotecas, fora as já disponibilizadas pelo Python e Flask, são utilizadas neste projeto e chamamos tal situação de dependências. Atualmente temos as seguintes dependências:

* rsvg-convert python lib
  + <https://github.com/jmcb/python-rsvg-dependencies>
  1. Aplicativo

Após termos instalado todas as bibliotecas, frameworks e banco de dados, temos finalmente a instalação do sistema que está sendo construído. Existem algumas particularidades na instalação do aplicativo, como por exemplo.

* Configura o Proxy
  + set HTTP\_PROXY=http://proxy.prodemge.gov.br:8080
* Exemplo com Usuário e Senha
  + HTTP\_PROXY=http://user:password@your-company-proxy.com:8080
* VARIÁVEL DE AMBIENTE: If you don't have rights to modify the global system variables (you can only do so if you have local Admin rights), simply add it to your user-level variables.
  + Set it from My Computer > Properties > Advanced > Environment Variables (or "Advanced Properties" if in Windows 7)

Variáveis de ambiente que devem ser registradas para o sistema funcionar. No caso do linux basta utilizar:

* SET VISUAL\_DB\_NAME=ps.

Já no caso de ambiente Windows, as chaves são criadas via tela através do caminho:

* Botão direto em Computador -> Propriedades
* Acesse a opção "Configurações avançadas do sistema"
* Abrirá na aba "Avançado" e abaixo terá a opção "Variáveis de Ambiete"
* No Box de "Variáveis de Ambiente", adicione as chaves abaixo
  + VISUAL\_OAUTH\_GOOGLE\_ID 1
  + VISUAL\_OAUTH\_GOOGLE\_SECRET 1
  + VISUAL\_OAUTH\_TWITTER\_ID 1
  + VISUAL\_OAUTH\_TWITTER\_SECRET 1
  + VISUAL\_OAUTH\_FACEBOOK\_ID 1
  + VISUAL\_OAUTH\_FACEBOOK\_SECRET 1
  + VISUAL\_DB\_USER root
  + VISUAL\_DB\_PW 123456
  + VISUAL\_DB\_HOST localhost
  + VISUAL\_DB\_NAME dataminas
  + VISUAL\_REDIS\_HOST localhost
  + VISUAL\_REDIS\_PORT 6379
  + VISUAL\_REDIS\_PW

Caso queira alterar as configurações de conexão com

* 1. FLASK

Depois de instalado o driver devemos então criar as tabelas de sistemas utilizando o comando:

* python manage.py syncdb

Para iniciar o servidor utilize:

* python run.py runserver
* python run.py runserver 0.0.0.0:8000

DEPLOY

Temos hoje um servidor virtualizado com a ferramenta *GITLAB*, que tem o controle de versionamento *GIT* com uma interface Web para gerenciamento. Mas será necessário também neste projeto o uso de uma ferramenta de integração contínua, que será adicionado ao servidor de repositório, no caso o <http://200.198.6.6/users/sign_in>

* <http://git-scm.com/downloads/guis>
* http://windows.github.com/

CACHE

O sistema de cache utilizado no PS é o Redis e este tipo de sistema trabalha com a ideia de chave e valor, ou seja, você salva uma chave e o valor dela para quando necessitar novamente, entrega a chave e o sistema de cache retorna o valor.

Existem outros padrões de cache Web onde é realizado uma cópia de toda a tela, portanto se dois usuários acessarem ou mesmo endereço de internet, para o segundo usuário ele apenas entrega a cópia da tela visto pelo usuário anterior. Desta forma a requisição de visualização da tela não chega nem ao servidor, ou seja, o sistema de cache trabalha entre o cliente e a aplicação.

Realize a instalação do Redis no site oficial, tomando cuidado para a arquitetura do seu computador, 32 ou 64 bits.

* http://redis.io/download
  + redis-2.4.6-setup-64-bit.exe

Redis para Windows 64 Bits é uma versão que ainda não é suportada pelos desenvolvedores oficiais. Mesmo assim existe na comunidade de desenvolvedores uma versão que está estável, mas ainda não recomendada para produção. Portanto para ambiente de desenvolvimento é uma ótima opção, para isso baixe o arquivo em:

* https://github.com/dmajkic/redis/downloads

Após realizar a instalação é possível inicial executando o arquivo redis-server.exe que fica dentro da pasta instalada. Ao executar ele manterá a janela de comando do Windows aberta, e passará a funcionar na pasta padrão, mesmo utilizada pela aplicação para realização do cache.

A inicialização do service deve ser feito manualmente, executando o arquivo redis-server.exe , encontrado na pasta onde foi instalado o Redis.

Para limpar o cache do Redis, processo chamado de Clean redis, faça:

* python -c "import redis; r = redis.Redis(); r.flushdb();"