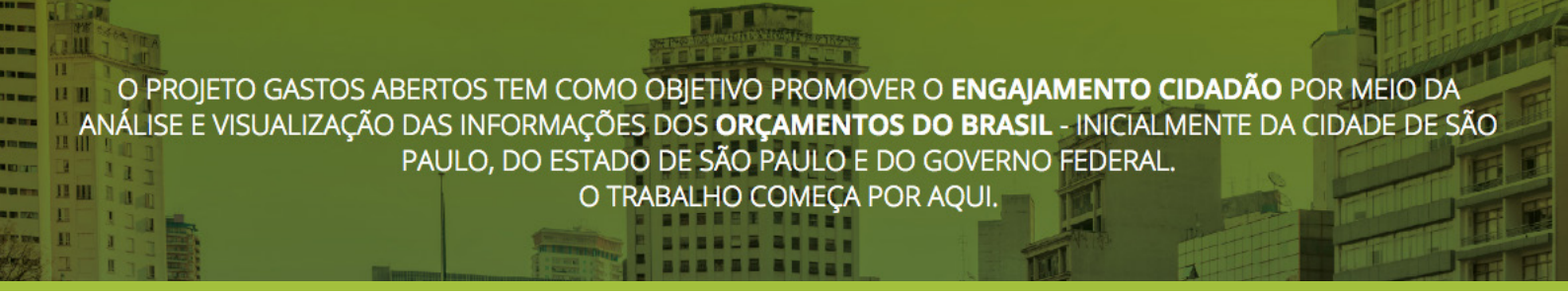
gastos abertos

Guia **CKAN**
para desenvolvedores

#gastosabertos
gastosabertos.org



Indice

Indice

Mod. 01. Orientações Gerais

Sobre o CKAN

CKAN é a sigla para **Global Knowledge Archive Network** e é um aplicativo de **código aberto** para o **armazenamento e distribuição** de dados, como planilhas e o conteúdo do código de banco de dados.

A plataforma vem sendo usada por diversos portais de dados abertos ao redor do mundo, bem como por governos que almejam aumentar o grau de transparência de suas administrações, facilitando o acesso aos dados de maneira aberta e estruturada. O código fonte do CKAN é aberto e mantido pela organização internacional Open Knowledge International e pode ser acessado no site oficial da plataforma, neste endereço: *http://ckan.org/developers/docs-and-download/*

Características

- Publicar e encontrar conjuntos de dados;
- Armazenar e gerenciar dados;
- Envolver-se com os usuários e outros;
- Personalizar e estender através de uma API aberta.

Objetivo Geral

O Guia do CKAN, têm como objetivo, capacitar usuários e desenvolvedores na implantação, setup, hospedagem, criação, gerenciamento e visualização.

Objetivos específicos

O objetivo específico é que o desenvolvedor seja capacitado para instalar uma instância do CKAN, consiga utilizar as ferramentas de customização, seja capaz de realizar manutenções na plataforma, que tenha conhecimento prático e teórico para utilizar as ferramentas disponíveis no sistema.

BASE

Esta parte do guia tem como objetivo nivelar o conhecimento sobre o que são dados abertos e o objetivo do CKAN como ferramenta.

O QUE SÃO DADOS ABERTOS?

Segundo a definição da Open Knowledge, em suma, dados são abertos quando qualquer pessoa pode livremente usá-los, reutilizá-los e redistribuí-los, estando sujeito a, no máximo, a exigência de creditar a sua autoria e compartilhar pela mesma licença.
Isso geralmente é satisfeito pela publicação dos dados em formato aberto e sob uma licença aberta.

OPEN KNOWLEDGE

A Open Knowledge, antes conhecida como Open Knowledge Foundation, é uma organização sem fins lucrativos que promove conhecimento livre.
https://pt.wikipedia.org/wiki/Open_Knowledge

Os dados abertos também são pautados pelas três leis e oito princípios.

As três leis

O especialista em políticas públicas e ativista de dados abertos David Eaves propôs as seguintes "leis":

1. Se o dado não pode ser encontrado e indexado na Web, ele não existe;
2. Se não estiver aberto e disponível em formato compreensível por máquina, ele não pode ser reaproveitado;
3. Se algum dispositivo legal não permitir sua replicação, ele não é útil.

As leis foram propostas para os Dados Abertos Governamentais, mas aceita-se sua aplicação aos dados abertos de forma geral.

Os oito princípios

Em 2007, um grupo de trabalho de 30 pessoas reuniu-se na Califórnia, Estados Unidos da América, para definir os princípios dos Dados Abertos Governamentais. Chegaram num consenso sobre os seguintes 8 princípios:

1. **Completos.** Todos os dados públicos são disponibilizados. Dados são informações eletronicamente gravadas, incluindo, mas não se limitando a, documentos, bancos de dados, transcrições e gravações audiovisuais. Dados públicos são dados que não estão sujeitos a limitações válidas de privacidade, segurança ou controle de acesso, reguladas por estatutos.
2. **Primários.** Os dados são publicados na forma coletada na fonte, com a mais fina granularidade possível, e não de forma agregada ou transformada.
3. **Atuais.** Os dados são disponibilizados o quão rapidamente seja necessário para preservar o seu valor.
4. **Acessíveis.** Os dados são disponibilizados para o público mais amplo possível e para os propósitos mais variados possíveis.

5. **Processáveis por máquina.** Os dados são razoavelmente estruturados para possibilitar o seu processamento automatizado.

6. **Acesso não discriminatório.** Os dados estão disponíveis a todos, sem que seja necessária identificação ou registro.

7. **Formatos não proprietários.** Os dados estão disponíveis em um formato sobre o qual nenhum ente tenha controle exclusivo.

8. **Livres de licenças.** Os dados não estão sujeitos a regulações de direitos autorais, marcas, patentes ou segredo industrial. Restrições razoáveis de privacidade, segurança e controle de acesso podem ser permitidas na forma regulada por estatutos.

Além disso, o grupo afirmou que a conformidade com esses princípios precisa ser verificável e uma pessoa deve ser designada como contato responsável pelos dados Apesar dos princípios terem sido pensados para os Dados Abertos Governamentais, pode-se aplicá-los, também, a Dados Abertos de modo geral (com a possível exceção do primeiro, já que este trata de dados do poder público).

Fonte: http://dados.gov.br/dados-abertos

Linked Data

O conceito de **linked data** (dados ligados entre si) é um conjunto de práticas introduzidas por Tim Berners-Lee em suas notas de Arquitetura web "Linked Data", com função de publicar e estruturar dados na Web. Estas práticas vêm sido cada vez mais adotadas levando à criação do que conhecemos como Web de dados. No contexto de Web Semântica, a função não é somente lançar os dados, é fazer com que a pessoa e a máquina possam explorar a Web de Dados.

Os 4 princípios de linked data

Um documento web é construído sobre um pequeno conjunto de padrões simples, utilizando URIs como mecanismo global e único de identificação, HTTP como mecanismo de acesso universal, e HTML como formato de conteúdo. Daí a Web é construída sobre o princípio de manter hiperlinks entre documentos da Web Baseando-se nisto, Berners-Lee criou suas notas, que ficaram conhecidas como "Os princípios da Linked Data". São eles:

1. Use URIs para nomear as coisas
2. Use URIs HTTP para que as pessoas possam procurar o desejado
3. Quando alguém olha para um URI, forneça informações úteis, usando os padrões (RDF *, SPARQL)
4. Incluir links para outros URIs. Para que eles possam descobrir explorar mais as coisas

Fonte: https://pt.wikipedia.org/wiki/Linked_data

Meta data

O **CKAN** foi desenvolvido para os metadados.

Metadados, ou Metainformação, são dados sobre outros dados. Um item de um metadado pode dizer do que se trata aquele dado, geralmente uma informação inteligível por um computador. Os metadados facilitam o entendimento dos relacionamentos e a utilidade das informações dos dados.

Web semântica, é uma web "inteligente", capaz de conceder um significado a um arquivo que será disponibilizado para outros utilizadores, podendo ser usado como fonte de pesquisa.

No CKAN a utilização é para descrever melhor o que são as bases de dados, além do nome do próprio arquivo.

RDF

Resource Description Framework (RDF) é uma linguagem para representar informação na Internet. Arquivos RDF são modelos ou fontes de dados, também conhecidos como metadata, tecnologia endossada e recomendada pela W3C desde fevereiro de 1999, tendo como principais objetivos criar um modelo simples de dados, com uma semântica formal, usar o vocabulário URI-based. Os arquivos RDF têm três componentes básicos: recurso, propriedade e indicação, o que torna a linguagem altamente escalável.

- **Recurso:** Qualquer coisa que pode conter um URI, incluindo as páginas da web, assim como elementos de um documento XML.
- **Propriedade:** Um recurso que tenha um determinado nome e possa ser utilizado como uma propriedade.
- **Indicação:** consiste na combinação de um recurso, de uma propriedade, e de um valor.

Atualmente, os principais buscadores utilizam RDFs fornecidos pelos websites para otimizar os rankings de indexação, dessa forma, os sites que contenham tais informações podem ser melhor qualificados nos resultados de busca.

Licença de dados livres

É recomendado que todo dado seja distribuído com alguma licença, há licenças especificas para dados, e é importante ressaltar que elas são especificas para dados e não deve ser utilizada licenças existentes para software livre ou conteúdo livre.

- **Open Data Commons Public Domain Dedication and License (PDDL)**
  http://opendatacommons.org/licenses/pddl
- **Open Data Commons Open Database License**
  http://opendatacommons.org/licenses/odbl
- **Database Contents License**
  http://opendatacommons.org/licenses/dbcl
- **Open Government License**
  http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3

Mod. 02. Conceitos básicos CKAN

O objetivo é explicar o workflow de funcionamento do CKAN e suas principais características.

Quem utiliza ?

Há um site utilizado para informar a maioria dos sites que utiliza o CKAN

- **http://instances.ckan.org/**
  Neste site é possível visualizar instancias atuais do CKAN

Referências e documentações disponíveis na Internet sobre o CKAN

Há diversos fóruns e manuais sobre o CKAN na internet, porém a documentação oficial esta disponível neste endereço: **http://docs.ckan.org**

Repositório de código fonte do CKAN

O código fonte do CKAN é aberto, e está disponível em **https://github.com/ckan/ckan**.
A licença é a *GNU Affero General Public License*.

Datasets & Resources

Para o CKAN, os dados são publicados em unidades chamadas "bases de dados" (datasets). Um conjunto de dados é um pacote de dados - por exemplo, poderia ser as estatísticas de criminalidade para uma região, os valores de gastos para um departamento do governo, ou as leituras de temperatura de várias estações meteorológicas. Quando os usuários pesquisarem os dados, os resultados das pesquisas que virão, serão conjuntos de dados individuais.

Um conjunto de dados contém duas coisas:

1. Informações ou "metadados" sobre os dados. Por exemplo, o título e editora, data, em que formato ele está disponível, sob qual licença ele é liberado, etc.
2. Uma série de "recursos", que mantêm os dados em si. Para o CKAN não é importante em que formato os dados estão, um recurso pode ser uma planilha CSV ou Excel, arquivo XML, documento PDF, arquivo de imagem, dados vinculados no formato RDF, etc., o CKAN pode armazenar o recurso internamente, ou armazená-lo simplesmente como um link, o recurso próprio pode estar em outro lugar na web. Um conjunto de dados pode conter qualquer número de recursos. Por exemplo, diferentes recursos podem conter os dados para anos diferentes, ou eles podem conter os mesmos dados em diferentes formatos.

Usuários, Organizações e Autorizações

Usuários CKAN podem registrar contas de usuários e logar no sistema (dependendo da configuração local), o login não é necessário para procurar e encontrar dados, mas é necessário para todas as funções de publicação: Datasets podem ser criados, editados, etc. pelos usuários, com permissões apropriadas.

Normalmente, cada conjunto de dados é propriedade de uma "organização". O CKAN pode ter qualquer número de organizações. Por exemplo, se o CKAN está sendo usado como um portal de dados por um governo nacional, as organizações podem ser diferentes departamentos governamentais, cada um dos quais publica dados. Cada organização pode ter seu próprio fluxo de trabalho e autorizações, permitindo-lhes gerir o seu próprio processo de publicação.

Administradores de uma organização podem adicionar usuários individuais a ele, com diferentes funções, dependendo do nível de autorização necessário. Um usuário em uma organização pode criar um conjunto de dados de propriedade da organização. Na configuração padrão, esse conjunto de dados é inicialmente privado e visível apenas para outros usuários na mesma organização. Quando ele está pronto para publicação, ele pode ser publicado através de um simples botão. Isso pode exigir um nível de autorização mais alto dentro da organização.

Os datasets normalmente não podem ser criados, exceto dentro das organizações É possível, no entanto, configurar CKAN para permitir que conjuntos de dados não pertence a qualquer organização. Tais conjuntos de dados podem ser editados por qualquer usuário logado, criando a possibilidade de uma DataHub wiki-like.

FileStore

É onde o conteúdo dos arquivos é armazenado, é a funcionalidade que permite que arquivos sejam enviados para o CKAN como upload.

DataStore

É uma extensão, opcional, que é utilizado para armazenar os dados estruturados dos recursos do CKAN, desta maneira permitindo a visualização dos dados de maneira mais inteligente através do "Data Explorer". Além de que oferece uma API para que desenvolvedores possam explorar os dados hospedados no CKAN de maneira mais completa.

O *DataStore* é distinto, porém complementar ao FileStore. Em contraste com a FileStore que fornece armazenamento "blob" de arquivos inteiros com nenhuma forma de acessar, ou partes de consulta desse arquivo, o DataStore é como um banco de dados no qual elementos de dados individuais são acessíveis. Para ilustrar esta distinção, considere armazenar um arquivo de planilha como um documento CSV ou Excel. No FileStore este arquivo seria armazenado diretamente, seria o Excel. Para acessá-lo você deve baixar o arquivo como um todo. Em contrapartida, se os dados da planilha são armazenados no DataStore, seria capaz de acessar as linhas da planilha individuais através de uma API simples na web, bem como ser capaz de fazer consultas sobre o conteúdo da planilha.

EXERCÍCIOS

1. Criar uma conta e publicar um dataset no http://demo.ckan.org
2. Criar uma visualização via tabelas e gráficos
3. Realizar o download de uma tabela

Mod. 03. Infra-estrutura básica e características

Descreve quais as tecnologias utilizadas para desenvolver o CKAN e uma reflexão de como manter e implementar.

Como o CKAN foi desenvolvido? Quais tecnologias foram utilizadas?

O CKAN foi desenvolvido na linguagem **Python**, utiliza o **banco de dados PostgreSQL** e o **Apache Solr**. Os desenvolverem utilizam o git para gerenciar o código fonte.

É sugerido utilizar o Apache e o Nginx como servidores web. O Apache fica responsável por executar a aplicação via WSGI, que é um padrão Python para descrever como um aplicativo web pode se comunicar com um servidor WEB e auxilia na performance do aplicativo.

Infra-estrutura mínima e como dimensionar uma infra-estrutura ideal

Para uma instancia CKAN comum, é sugerido no mínimo um servidor que atenda estes requisitos:

- **CPU cores 2**
- **GB of RAM 4**
- **GB of disk space 60**

Se você irá executar o frontend e o backend em um mesmo servidor, provavelmente você irá precisar de 4(+) CPU cores.

Para cenários maiores, é interessante que o frontend e o backend do CKAN estejam fisicamente separados.

Instalação

Descreve como funciona o processo de instalação, atualização e backup de uma instancia do CKAN.

Vagrant

Para o treinamento, recomendamos a utilização do vagrant. Caso for utilizar, esta documentação foi baseada nas seguintes imagens:

CentOS 7.0
http://opscode-vm-bento.s3.amazonaws.com/vagrant/virtualbox/opscode_centos-7.0_chef-provisionerless.box

Ubuntu Server 12.04 amd64
http://goo.gl/8kWkm

Caso queira acessar portas que estejam nas máquinas virtuais, você deve editar o arquivo Vagrantfile, e configurar pelo parametro:

```
config.vm.network "forwarded_port", guest: 80, host: 8080
```

Depois de alterar o arquivo, a máquina virtual deve ser reiniciada.

Atenção, não é recomendando a utilização do vagrant em produção, apenas para treinamento ou desenvolvimento.

Como as versões são organizadas nas novas releases do CKAN?

CKAN segue um ciclo de lançamento previsível de modo que os usuários podem depender de versões estáveis de CKAN, e podem planejar suas atualizações para novas versões.

Cada versão tem um número da versão da forma M.m (por exemplo. 2.1) ou M.m.p. (por exemplo. 1.8.2), em que M é a versão principal, m é a versão menor e p é o número da versão de correções. Existem três tipos de lançamento:

Principais Lançamentos

Major Releases: Lançamentos importantes, como CKAN 1.0 e 2.0 CKAN, incrementar o número de versão principal. Essas versões contêm grandes mudanças na base de código CKAN, com refatorações significativas e alterações de quebra, por exemplo na API ou nos modelos. Estes lançamentos são muito pouco frequentes.

Minor Releases: Lançamentos menores, tais como CKAN 1.8 e 2.1 CKAN, incrementar o número de versão secundárias. Essas versões não são tão impactantes como grandes lançamentos, mas talvez contenham mudanças que não sejam retro compatíveis. O Changelog irá documentar quaisquer alterações de quebra. Nosso objetivo é lançar uma versão menor de CKAN aproximadamente a cada três meses.

Patch Releases: Lançamento para correções, como CKAN 1.8.1 ou 2.0.1 CKAN, incrementar o número da versão do patch. Estes lançamentos não quebram compatibilidade com versões anteriores, eles só incluem correções de bugs e otimizações e recursos. Estes lançamentos não contêm:

- As alterações do esquema do banco de dados ou migrações
- Mudanças de interface Função
- Mudanças na interface do Plugin
- Novas dependências
- Refatorações grandes ou novos recursos em partes críticas do código

Iniciando a instalação

Há três maneiras de você instalar o CKAN:

- Através dos pacotes do sistema operacional;
- Pelos códigos fontes;
- Utilizando imagens Docker.

Instalação baseada em pacotes do sistema operacional

Para instalar via pacotes do sistema operacional, é no caso que você queira executar apenas uma instancia do CKAN no servidor e irá utilizar a distribuição Ubuntu 12.04 64-bit.

Instalação de pacotes

Para este tipo de instalação é necessário realizar a instalação de todas as dependências do sistema operacional, neste caso é utilizado o **Ubuntu 12.04**.

```
apt-get update
apt-get install -y nginx apache2 libapache2-mod-wsgi libpq5 \
postgresql solr-jetty \
openjdk-6-jdk python-pastescript curl \
python-dev python-sqlalchemy python-pylons python-pip
```

Instalação do CKAN

O CKAN disponibiliza um pacote para o Debian na arquitetura amd64, de todas as versões superiores à 2.0.

```
wget http://packaging.ckan.org/python-ckan_2.4_amd64.deb
dpkg -i python-ckan_2.4_amd64.deb
```

Configuração dos serviços

É necessário configurar o Jetty, com o caminho correto para o JAVA, a porta utilizada pelo serviço e manter para que ele seja inicializado automaticamente pelo sistema operacional, para isto edite o arquivo **"/etc/default/jetty",** e altere o parâmetros:

```
NO_START=0
JETTY_PORT=8983
JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64/
```

Feito isto, vamos configurar o esquema correto para a plataforma do CKAN no Solr.

```
mv /etc/solr/conf/schema.xml /etc/solr/conf/schema.xml.bak
ln -s /usr/lib/ckan/default/src/ckan/ckan/config/solr/schema.xml /etc/solr/
conf/schema.xml
service jetty restart
```

Configuração do banco de dados

Antes da criação do banco é necessário realizar uma configuração no CKAN especifica, edite o arquivo **"/etc/ckan/default/production.ini"** e configure o parâmetro "ckan. site_url" e descomente "solr_url".

```
ckan.site_url = http://localhost
solr_url = http://127.0.0.1:8983/solr
```

Quando executar o primeiro comando abaixo, será solicitado uma senha para o usuário "ckan_default". Lembre-se dela para posteriormente, colocar no arquivo de configuração do CKAN.

```
sudo -u postgres createuser -S -D -R -P ckan_default
sudo -u postgres createdb -O ckan_default ckan_default -E utf-8
```

Para testar, se o banco foi criado veja através de "sudo -u postgres psql -l"
Verifique no arquivo de configuração **"/etc/ckan/default/production.ini"**, o parâmetro **sqlalchemy.url** com os dados de usuário e senha para acesso ao banco.

```
sqlachemy.url = postgresql://ckan_default:pass@localhost/ckan_default
```

Depois disto, inicie o banco.

```
ckan db init
```

Configurando o FileStore

```
mkdir -p /var/lib/ckan/default
chown www-data /var/lib/ckan/default
chmod u+rwx /var/lib/ckan/default
```

Edite o arquivo **"/etc/ckan/default/production.ini"** e descomente a linha:

```
ckan.storage_path = /var/lib/ckan/default
```

Configurando o DataStorage

Quando executar o primeiro comando, será solicitado uma senha para o usuário "ckan_default". Lembre-se dela para posteriormente, colocar no arquivo de configuração do ckan.

```
sudo -u postgres createuser -S -D -R -P -l datastore_default
sudo -u postgres createdb -O ckan_default datastore_default -E utf-8
```

Edite o arquivo **"/etc/ckan/default/production.ini"** e descomente as linhas "ckan.datastore.write_url" e "ckan.datastore.read_url". Adicione também na lista de plugins "datastore" pelo parametro "ckan.plugins". Um exemplo, destes parâmetros configurados seria:

```
ckan.plugins = stats text_view image_view recline_view datastore
ckan.datastore.write_url = postgresql://ckan_default:pass@localhost/datastore_default
ckan.datastore.read_url = postgresql://datastore_default:pass@localhost/datastore_default
```

```
service apache2 restart
ckan datastore set-permissions
```

Reinicie os serviços.

```
service nginx restart
service apache2 restart
service jetty restart
```

Para realizar o teste se o DataStorage foi realizado corretamente, execute o comando abaixo e veja se irá retornar um JSON.

```
curl -X GET
"http://127.0.0.1:80/api/3/action/datastore_search?resource_id=_table_metadata"
```

Acompanhe os logs do Apache para entender qual tipo de problema você pode estar passando na utilização da aplicação.

Instalação baseada nos códigos fontes (CentOS)

Para este tipo de instalação é necessário realizar a instalação de todas as dependências do sistema operacional, neste caso é utilizado o CentOS 7.0.

```
# yum -y install wget policycoreutils-python
# yum -y install vim psmisc lynx
# yum -y install xml-commons subversion mercurial postgresql-server\
postgresql-devel postgresql python-devel libxslt libxslt-devel \
libxml2 libxml2-devel python-virtualenv gcc gcc-c++ make \
java-1.6.0-openjdk-devel java-1.6.0-openjdk tomcat tomcat-webapps \
tomcat-admin-webapps xalan-j2 unzip policycoreutils-python \
mod_wsgi httpd git
```

Nossa instalação será baseada em um usuário do sistema, que será "ckan". Este usuário será criado com uma shell "/sbin/nologin" e com diretório home sendo "/usr/lib/ckan".

```
# useradd -m -s /sbin/nologin -d /usr/lib/ckan -c "CKAN User" ckan
```

Precisamos criar o diretório, com permissão de leitura para todos, pois eventualmente o servidor web irá necessitar acessar ele.

```
# chmod 755 /usr/lib/ckan
```

Vamos alterar o ambiente para o usuário ckan.

```
# su -s /bin/bash - ckan
```

Vamos instalar um ambiente isolado Python através do virtualenv, e ativar ele.

```
virtualenv --no-site-packages default
. default/bin/activate
```

Agora, nossa instalação irá iniciar, vamos realizar o download e a instalação da versão 2.2.1 do CKAN, e depois instalar os módulos Python que sejam necessários.

```
pip install --ignore-installed -e \
"git+https://github.com/okfn/ckan.git@ckan-2.2.1#egg=ckan"
pip install --ignore-installed -r default/src/ckan/pip-requirements-docs.txt
```

Configuração do PostgreSQL

Habilite o PostgreSQL para ser iniciado com o sistema operacional.

```
# systemctl enable postgresql.service
```

O comando abaixo, é necessário apenas uma vez, é para criação do PGDATA. Pode ser, que em versões antigas do RedHat ou CentOS, este comando não seja necessário.

```
# service postgresql initdb
```

Edite o arquivo "/var/lib/pgsql/data/pg_hba.conf" para que usuário sejam autenticados através de senha e o usuário postgres se mantenham através de ident. As alterações mais relevantes, são:

```
local   all         postgres                ident
local   all         all                     md5
# IPv4 local connections:
host    all         all         127.0.0.1/32      md5
# IPv6 local connections:
host    all         all         ::1/128           md5
```

Vamos iniciar o servidor do PostgreSQL.

```
# systemctl start postgresql.service
```

Vamos criar o usuário e o banco para utilização do CKAN, no primeiro comando será solicitado uma senha, lembre-se que iremos utilizar esta senha nos arquivos de configuração posteriormente.

```
# sudo -u postgres createuser -S -D -R -P ckan_default
# sudo -u postgres createdb -O ckan_default ckan_default -E utf-8
```

Criar a configuração do CKAN no sistema

Vamos criar o diretório onde teremos os arquivos de configuração do CKAN.

```
# mkdir -p /etc/ckan/default
# chown -R ckan /etc/ckan/
```

Com o usuário do CKAN, vamos criar o arquivo de configuração inicial.

```
# su -s /bin/bash - ckan
. default/bin/activate
cd /usr/lib/ckan/default/src/ckan
paster make-config ckan /etc/ckan/default/development.ini
```

Para finalizar, iremos editar o arquivo "/etc/ckan/default/development.ini" e alterar os seguintes parâmetros:

```
sqlalchemy.url = postgresql://ckan_default:pass@localhost/ckan_default
ckan.site_id = default
solr_url = http://127.0.0.1:8080/solr/ckan-schema-2.0
ckan.site_url = http://localhost
```

Instalando e configurando o Apache Solr

É recomendado a utilização 1.4.1 do Apache Solr com o CKAN. Faça o download pelo comando:

```
# curl http://archive.apache.org/dist/lucene/solr/1.4.1/apache-solr-1.4.1.tgz | tar xzf -
```

Crie os diretórios para atender todos os cores.

```
# mkdir -p /usr/share/solr/core0 /var/lib/solr/data/core0 /etc/solr/core0
# mkdir -p /usr/share/solr/core1 /var/lib/solr/data/core1 /etc/solr/core1
```

Copie o Apache Solr para o diretório.

```
# cp apache-solr-1.4.1/dist/apache-solr-1.4.1.war /usr/share/solr
```

Copie um exemplo de configuração do Apache Solr para o diretório core0.

```
# cp -r apache-solr-1.4.1/example/solr/conf /etc/solr/core0
```

Edite o arquivo "/etc/solr/core0/conf/solrconfig.xml" e altere as seguintes instruções:

```
<dataDir>${dataDir}</dataDir>
```

Copie a configuração do core0 para o core1, e crie os links simbólicos necessários.

```
# cp -r /etc/solr/core0/conf /etc/solr/core1
# ln -s /etc/solr/core0/conf /usr/share/solr/core0/conf
# ln -s /etc/solr/core1/conf /usr/share/solr/core1/conf
```

Vamos utilizar o schema do CKAN.

```
# rm -f /etc/solr/core0/conf/schema.xml
# ln -s /usr/lib/ckan/default/src/ckan/ckan/config/solr/schema-2.0.xml /etc/solr/
core0/conf/schema.xml
# rm -f /etc/solr/core1/conf/schema.xml
# ln -s /usr/lib/ckan/default/src/ckan/ckan/config/solr/schema-1.4.xml /etc/solr/
core1/conf/schema.xm
```

Crie um novo arquivo "/etc/tomcat/Catalina/localhost/solr.xml", e adicione o conteúdo abaixo.

```
<Context docBase="/usr/share/solr/apache-solr-1.4.1.war" debug="0"
privileged="true" allowLinking="true" crossContext="true">
<Environment name="solr/home" type="java.lang.String" value="/usr/share/solr"
override="true" />
<Context/>
```

Crie um arquivo novo "/usr/share/solr/solr.xml" com o conteúdo:

```
<"solr persistent="true" sharedLib="lib>
<"cores adminPath="/admin/cores>
core name ="ckan-schema-2.0" instanceDir="core0"> <property>
<name="dataDir" value="/var/lib/solr/data/core0" /></core
core name="ckan-schema-1.4" instanceDir="core1"> <property>
<name="dataDir" value="/var/lib/solr/data/core1" /></core
<cores/>
<solr/>
```

Configure as permissões dos diretórios.

```
# chown -R tomcat:tomcat /usr/share/solr /var/lib/solr
```

Vamos ativar os serviços do tomcat.

```
# systemctl enable tomcat.service
# systemctl start tomcat.service
```

Para testar se o tomcat e o Apache Solr estão executando corretamente, acesse os endereços no servidor:

```
curl -X GET http://127.0.0.1:8080/
curl -X GET http://127.0.0.1:8080/solr
```

Criação das tabelas no banco de dados

Executar o comando no ambiente do usuário ckan.

```
# su -s /bin/bash - ckan
. default/bin/activate
cd default/src/ckan
paster db init -c /etc/ckan/default/development.ini
```

É provável que você veja alguns erros, mas no final deve conter a mensagem **"Initialising DB: SUCCESS"**.

Configurando o FileStore

```
mkdir -p /var/lib/ckan/default
chown apache /var/lib/ckan/default
chmod u+rwx /var/lib/ckan/default
```

Edite o arquivo **"/etc/ckan/default/development.ini"** e descomente a linha:

```
ckan.storage_path = /var/lib/ckan/default
```

Configurando o datastore

A configuração do datastore é opcional.

Quando executar o primeiro comando, será solicitado uma senha para o usuário "ckan_default". Lembre-se dela para posteriormente, colocar no arquivo de configuração do CKAN.

```
sudo -u postgres createuser -S -D -R -P -l datastore_default
sudo -u postgres createdb -O ckan_default datastore_default -E utf-8
```

Edite o arquivo **"/etc/ckan/default/development.ini"** e descomente as linhas "ckan.datastore.write_url" e "ckan.datastore.read_url". Adicione também na lista de plugins "datastore" pelo parametro "ckan.plugins". Um exemplo, destes parâmetros configurados seria:

```
ckan.plugins = stats text_preview recline_preview datastore
ckan.datastore.write_url = postgresql://ckan_default:pass@localhost/datastore_
default
ckan.datastore.read_url = postgresql://datastore_default:pass@localhost/
datastore_default
```

Feio isto, é necessário setar as permissões corretas no banco através:

```
# . /usr/lib/ckan/default/bin/activate
# paster --plugin=ckan datastore set-permissions -c /etc/ckan/default/
development.ini postgres
# deactivate
```

Este comando pode exigir que o usuário esteja no sudoers, caso seja o caso, inclua temporiamente as permissões necessárias em "/etc/sudoers".

Para testar, se esta tudo funcionando perfeitamente com a integração do datastore, faça estes comandos:

```
curl -X GET "http://127.0.0.1:80/api/3/action/datastore_search?resource_id=_table_metadata"
```

Crie um dataset "test" na interface web, depois execute o comando abaixo para criar um resource no dataset "test".

```
curl -X POST http://127.0.0.1:80/api/3/action/datastore_create -H "Authorization: {API KEY}" -d '{"resource": {"package_id": "{PACKAGE ID}"}, "fields": [ {"id": "a"}, {"id": "b"} ], "records": [ { "a": 1, "b": "xyz"}, {"a": 2, "b": "zzz"} ]}'
```

O retorno deve conter um JSON com algo parecido:

success": **true**, "result": {"resource": {"url": "_datastore_only_resource"}," ... "resource_id": "**d6f4e008-ff16-432a-9769-c49f4dee39b9**", "fields": [{"type": "int", "id": "a"}, {"type": "text", "id": "b"}], "method": "insert", "records": [{"a": 1, "b": "xyz"}, {"a": 2, "b": "zzz"}]}

Veja o ID do resource criado, você irá utilizar ele nos próximos comandos.

Quando você editar na interface web, irá notar que terá mais dados lá, e o que criado recentemente. Caso queira visualiazar via linha de comando, também faça isto:

```
curl -X GET "http://127.0.0.1:80/api/3/action/datastore_search?resource_id=**RESOURCE_ID**"
```

Finalmente, para remover o resource criado:

```
curl -X POST http://127.0.0.1:80/api/3/action/datastore_delete -H "Authorization: **API_KEY**" -d '{"resource_id": "**RESOURCE_ID**"}'
```

Todos os retornos contêm um JSON.

Para remover o dataset criado:

Configuração Repoze.who

Ele deve ser acessível do mesmo diretório do arquivo de configuração.

```
$ ln -s /usr/lib/ckan/default/src/ckan/who.ini /etc/ckan/default/who.ini
```

Configuração do Apache

Crie o arquivo **"/etc/ckan/default/apache.wsgi**" com o conteúdo:

```
import os
('activate_this = os.path.join('/usr/lib/ckan/default/bin/activate_this.py)
execfile(activate_this, dict(__file__=activate_this))

from paste.deploy import loadapp
config_filepath = os.path.join(os.path.dirname(os.path.abspath(__file__)),
'development.ini')
from paste.script.util.logging_config import fileConfig
fileConfig(config_filepath)
application = loadapp('config:%s' % config_filepath)
```

O modwsgi irá redirecionar todas as requisições do seu navegador para este script, que irá tratar todas as requisições direto com sua instância CKAN.

Crie o arquivo "**/etc/httpd/conf.d/ckan_default.conf**", para configurar o Apache.

```
WSGISocketPrefix /var/run/wsgi
<VirtualHost 0.0.0.0:80>
ServerName default.yourdomain.com
ServerAlias www.default.yourdomain.com
WSGIScriptAlias / /etc/ckan/default/apache.wsgi
# Pass authorization info on (needed for rest api).
WSGIPassAuthorization On
# Deploy as a daemon (avoids conflicts between CKAN instances).
WSGIDaemonProcess ckan_default display-name=ckan_default processes=2 threads=15
WSGIProcessGroup ckan_default
# Add this to avoid Apache show error:
# "AH01630: client denied by server configuration: /etc/ckan/default/apache.wsgi"
<Directory /etc/ckan/default>
Options All
AllowOverride All
Require all granted
<Directory/>
ErrorLog /var/log/httpd/ckan_default.error.log
CustomLog /var/log/httpd/ckan_default.custom.log combined
<VirtualHost/>
```

Habilite o apache para iniciar automaticamente com seu sistema operacional, e depois inicie ele.

```
chkconfig httpd on
systemctl start httpd.service
```

Para testar, basta acessar a instância do CKAN.

```
curl -X GET http://127.0.0.1:80/
```

Acompanhe os logs do Apache para entender qual tipo de problema você pode estar tendo na utilização da aplicação.

Instalação baseada no Docker

A equipe do CKAN ainda esta trabalhando com a distribuição do software através do Docker.

Busque por mais informações na documentação do CKAN ou em **http://hub.docker.com**

Mod. 05. Atualização do CKAN

Atualizações na sua instância são importantes de serem realizados de tempos em tempos, você pode se inscrever em uma lista de e-mails para receber todas as novas versões do CKAN no seu e-mail em:
https://lists.okfn.org/mailman/listinfo/ckan-announce

Para saber qual a versão atual do seu CKAN, basta acessar a URL:

```
curl -X GET http://127.0.0.1/api/util/status
```

O retorno será um JSON como este:

```
ckan_version": "2.2.1", "site_url": "http://localhost", "site_description": ""}
"site_title": "CKAN", "error_emails_to": "no-reply@eokoe.com", "locale_default":
"en", "extensions": ["stats", "text_preview", "recline_preview", "datastore",
"googleanalytics"]}
```

Para atualizar dos códigos fontes, você pode efetuar através do repositório git oficial do CKAN. Com estes comandos, você pode atualizar da versão "2.2.1" para "2.2.3".

```
root@localhost httpd]# su -s /bin/bash - ckan]
Last login: Tue Sep  1 11:23:54 UTC 2015 on pts/0
ckan@localhost ~]$ . default/bin/activate]
(default)[ckan@localhost ~]$ cd /usr/lib/ckan/default/src/ckan
(default)[ckan@localhost ckan]$ git fetch
(default)[ckan@localhost ckan]$ git checkout release-v2.2.3
Previous HEAD position was ab89c51... Update version number for 2.2.1 release
Branch release-v2.2.3 set up to track remote branch release-v2.2.3 from origin.
Switched to a new branch 'release-v2.2.3'
default)[ckan@localhost ckan]$ pip install --upgrade -r requirements.txt)
...
(default)[ckan@localhost ckan]$ python setup.py develop
running develop
...
Processing dependencies for ckan==2.2.3
Finished processing dependencies for ckan==2.2.3
```

Realizado toda atualização, basta reiniciar o servidor WEB.

Para atualizações maiores, veja as alterações no ChangeLog e a documentação em: **http://docs.ckan.org/en/latest/maintaining/upgrading/index.html**

Como realizar o backup do banco de dados?

O backup ideal do CKAN contempla seu banco de dados no PostgreSQL e o diretório do FileStore.

Há um comando para o banco de dados para as versões mais recentes do CKAN.

```
paster --plugin=ckan db dump --config=/etc/ckan/default/development.ini my_
ckan_database.pg_dump
```

Como realizar o monitoramento da aplicação ?

A melhor maneira para monitorar o CKAN é através de seus processos e serviços. Além de monitorar requisições reais, tais elas como:

```
curl -X GET http://127.0.0.1:80/api
```

EXERCÍCIOS

4 Realizar a instalação baseada em pacotes do sistema operacional

5 Realizar a instalação baseado em códigos fontes

6 Atualizar a versão do CKAN instalado via códigos fontes da versão "2.2.1" para "2.2.3"

Notificações por e-mail

CKAN pode enviar notificação para os usuários, como por exemplo quando o usuário tem uma atualização no seu dashboard. Uma vez habilitado, as notificações serão enviadas por e-mail pelo adminstrador, cada usuário pode configurar se deseja receber ou não estas notificações. Porém, para habilitar a notificação por e-mail o administrador deve:

Configurar uma tarefa no agendador "cron", para solicitar via API o envio dos e-mails pendentes. Na maioria dos UNIX, você pode realizar pelo comando "crontab -e" e adicionar uma nova linha no final do arquivo. Para mais informações veja o manual do crontab, "man crontab".

Lembre-se de fazer isto, quando a instalação for pels códigos fontes, de realizar isto pelo usuário que está sendo executado o CKAN, no nosso caso "ckan". Caso a instalação seja feita pelo pacote do sistema operacional, este comando deve estar configurado no usuário "root".

Adicione a linha no cron do usuário:

```
hourly source /usr/lib/ckan/default/bin/activate && echo '{}' | /usr/lib/ckan/@
default/bin/paster --plugin=ckan post -c /etc/ckan/default/development.ini /api/
action/send_email_notifications > /dev/null
```

Você utilizar @daily, @weekly ou @monhy no lugar de @hourly.

Você deve verificar as configurações do CKAN das notificações de e-mail, nenhum e-mail será enviado se o período especificado em "ckan.email_notifications_since config" for maior (padrão é 2 dias), por isto tente fazer com que a tarefa no crontab seja executada com a maior frequência possível. @hourly e @daily são boas escolhas.

Como este comando é via API, ele pode ser invocado de uma máquina que não esteja executando o CKAN.

Lembre-se, o parâmetro "ckan.activity_streams_email_notifications" deve estar como True, pois caso contrário, nenhum e-mail irá sair.

Verifique se o parâmetro "ckan.site_url" está correto, pois todos os links nos e-mails gerados irão com este endereço para o usuário clicar.

O parâmetro "ckan.site_title" é o nome que irá ser utilizado como origem do e-mail configurado em "smtp.mail_from".

Em um servidor em produção recomendamos utilizar um servidor de e-mail local, a configuração do servidor de e-mail deve ser realizada no arquivo de configuração do CKAN, pelos parâmetros:

```
ckan.activity_streams_enabled = true
ckan.activity_list_limit = 31
ckan.activity_streams_email_notifications = true
ckan.email_notifications_since = 2 days

Email settings ##

email_to = temp@eokoe.com
error_email_from = temp@eokoe.com
smtp.server = smtp.gmail.com:587
smtp.starttls = True
smtp.user = temp@eokoe.com
********** = smtp.password
smtp.mail_from = temp@eokoe.com
```

Para que as configurações sejam utilizadas, você deve reiniciar o servidor web.

Apps & Ideas

Esta funcionalidade permite que sejam compartilhadas notícias, ideias e referencias no geral sobre os datasets disponível na plataforma.

Ela opção é habilitada por padrão, e pode ser desabilitada pelo parâmetro ckan.dataset.show_apps_ideas no arquivo de configuração.

Page View Tracking

O CKAN pode armazenar as páginas visitadas no seu site e com estes dados permitir:

- Ordenar os datasets por popularidade
- Destacar com highlights datasets e resources populares
- Mostrar a lista dos datasets mais populares
- Exportar dados para um arquivo CSV

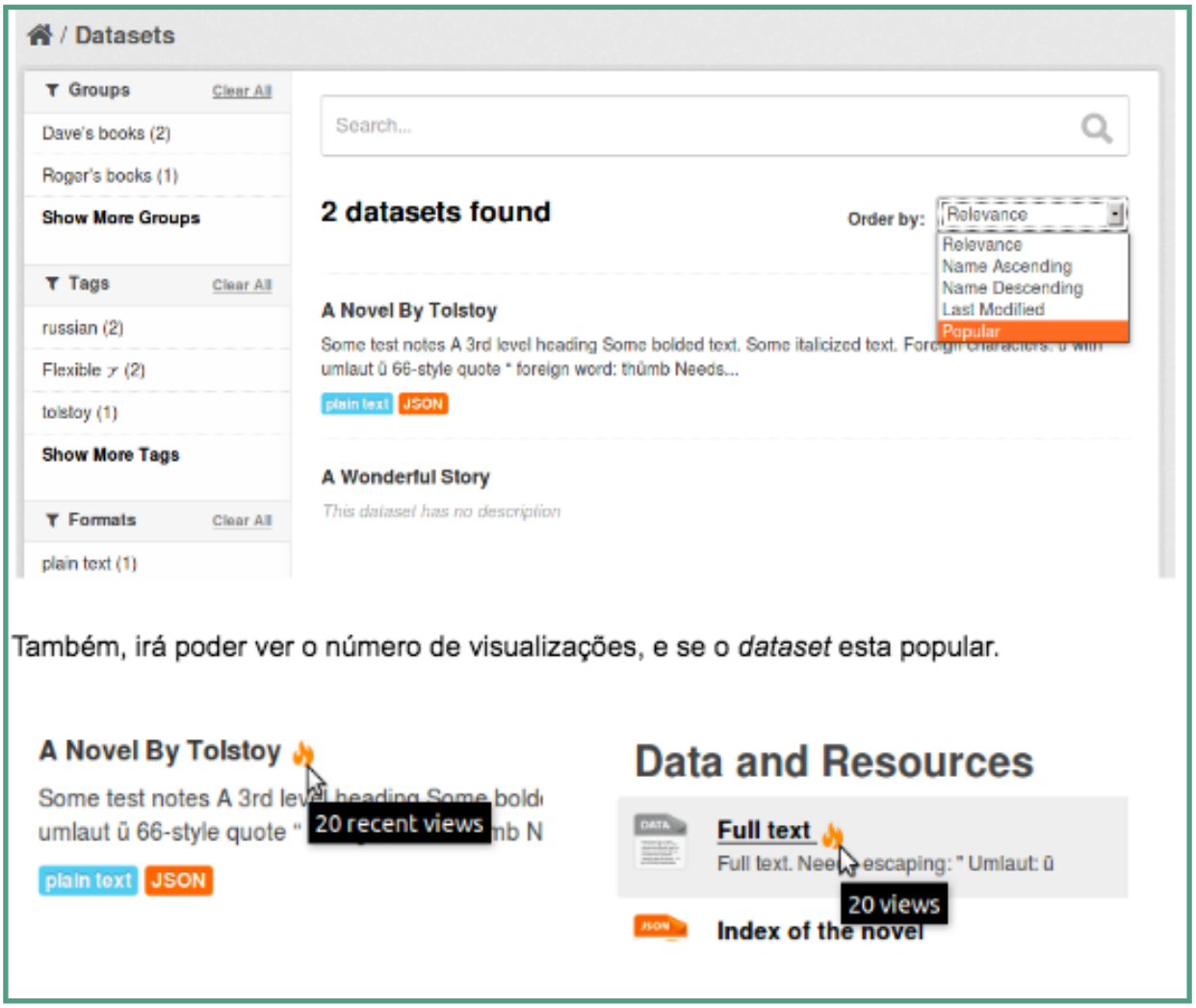
Para habilitar esta funcionalidade é muito simples, basta editar o arquivo de configuração do CKAN, e adicionar o parâmetro:

```
[app:main]
ckan.tracking_enabled = true
```

Esta funcionalidade é disponível de maneira assincrona, ou seja, é necessário executar comandos para que os dados sejam atualizados periodicamente. Uma boa maneira de realizar isto é adicionar os comandos no crontab, pode ser através do "crontab -e".

```
hourly source /usr/lib/ckan/default/bin/activate && /usr/lib/ckan/default/bin/@
paster --plugin=ckan tracking update -c /etc/ckan/default/development.ini && /
usr/lib/ckan/default/bin/paster --plugin=ckan search-index rebuild -r -c /etc/ckan/
default/development.ini
```

É necessário reinciar o aplicação para que a funcionlidade esteja disponível. Uma vez habilitado o tracking, você irá poder ordenar os datasets pela popularidade.



API

Demonstra o funcionando da API do CKAN.

A API do CKAN é útil para desenvolvedores que querem criar códigos para interagir com a plataforma.

A documentação completa está disponível em:
http://docs.ckan.org/en/latest/api/index.html

Autenticação e API Key

Algumas funcionalidades requerem autorização. A API utiliza as mesmas autorizações e configurações da interface web, então se um um usuário tem autorização para realizar alguma coisa na web, ele poderá realizar na API.

Para encontrar sua chave, basta realizar um login no CKAN via interface web e visitar o seu perfil.

EXERCÍCIO

7 Buscar as 10 tags mais utilizadas em http://demo.ckan.org

Extensões externas

Stats

Esta extensão analisa o banco de dados do CKAN e mostra algumas tabelas e gráficos com estatísticas sobre o seu site.

Para ativar ou desativar, basta editar o seu arquivo de configuração e adicionar "stats" no parâmetro "ckan.plugins".

Para visualizar as estatísticas, basta acessar a URI "/stats" do seu site. Um exemplo, veja em: *http://demo.ckan.org/stats e http://dados.gov.br/stats*.

Google Analytics

Com esta extensão é possível enviar informações para o Google Analytics, assim como retirar informações do Google Analytics para inserir em páginas do CKAN.

Para mais informações, veja em: *https://github.com/ckan/ckanext-googleanalytics*

Harvest

Com esta extensão é disponibilizada uma estrutura para o CKAN prover ou buscar informações entre instâncias do CKAN, algumas outras extensões trabalham baseado nela quando exigem comunicação entre instâncias do CKAN.

Para mais informações, veja em: *https://github.com/ckan/ckanext-harvest*

Linked Data & RDF

O ckanext-dcat é uma extensão que proporciona o CKAN expor e consumir metadados de outros catálogos utilizando documentos RDF que são serializados em DCAT. O "Data Catolog Vocabulary" (DCAT) é um vocabulário em RDF desenvolvido para facilitar a interoperabilidade entre catálogo de dados que são publicados na WEB. Para maiores informações sobre o DCAT veja no site da W3C: *http://www.w3.org/TR/vocab-dcat*

Para mais informaçõs, veja **https://github.com/ckan/ckanext-dcat**

CKAN & IOTA

Com esta extensão é possível sincronizar dados de uma plataforma IOTA com uma instância do CKAN.

Para mais informações, veja **https://github.com/eokoe/ckanext-iota**

EXERCÍCIO

8 Instalar o plugin do Google Analytics, utilizando o ID UA-1010101-1.

Customização

Objetivo é oferecer uma maneira de customizar o CKAN e manter apto para atualizações futuras.

Algumas customizações podem ser realizadas diretamente no arquivo de configuração do CKAN.

```
## Front-End Settings
ckan.site_title = CKAN
ckan.site_logo = /base/images/ckan-logo.png
ckan.site_description = The easy way to get, use and share data
ckan.favicon = /images/icons/ckan.ico
ckan.main_css = /base/css/my.css
```

Vejam todas as opções em **http://docs.ckan.org/en/latest/maintaining/configuration.html#front-end-settings**

Para maioria dos cenários a alteração do CSS funciona muito bem, porém há outros casos onde há necessidade de alterar a estrutura das páginas, e para isto sugerimos a criação de um tema.

A criação de um tema no CKAN, significa a criação de uma extensão. Uma boa referência de como isto pode ser realizado está disponível online na documentação do CKAN.

http://docs.ckan.org/en/latest/theming/templates.html

EXERCÍCIOS

9 Customizar o logotipo.

10 Alterar cores padrões das páginas do CKAN.


gastos
abertos
gastosabertos.org