

# Presupuestos del Gobierno de La Rioja

*Documentación Técnica*

8/12/2023. Versión 1.0: Primera versión.

# Introducción

El objetivo de la aplicación es ofrecer una visualización de los Presupuestos del Gobierno de La Rioja suficientemente intuitiva como para ser comprendida por personas sin experiencia previa, pero haciendo a la vez disponibles los detalles de cada elemento del presupuesto para las personas interesadas en profundizar más. La aplicación muestra la realidad del presupuesto en su conjunto, cubriendo tanto el lado de los ingresos como el de los gastos, y tanto las cantidades previstas como las finalmente realizadas (cuando la información esté disponible).

Las principales funcionalidades de la aplicación son:

- Visualización de gastos e ingresos presupuestados, de forma jerárquica y según las cuatro clasificaciones usadas en los presupuestos: funcional (para qué se gasta), económica (en qué se gasta, o cómo se ingresa) y orgánica/institucional (quién gasta/ingresa).
- Mostrar la información de los programas presupuestarios al máximo nivel existente, el nivel de partida.
- Mostrar la evolución de los presupuestos.
- Búsqueda de texto libre en los presupuestos, facilitando encontrar cualquier dato.

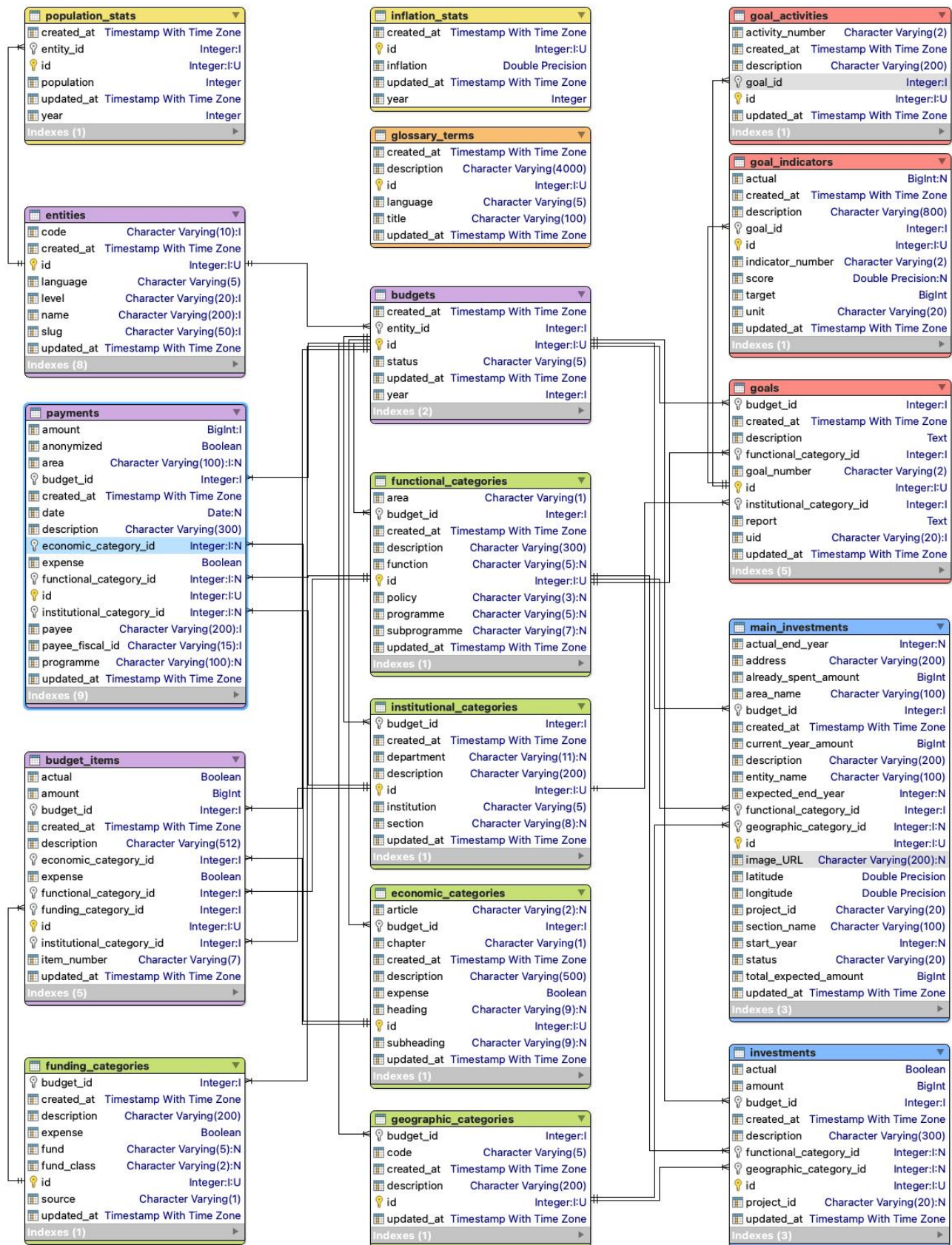
# Arquitectura del software

La aplicación web está desarrollada sobre Python 2.7, sobre la plataforma de código abierto Django, utilizando la arquitectura MVC (Model-View-Controller) común en aplicaciones web.

- El modelo se almacena en las tablas de una base de datos relacional utilizando el componente de persistencia incluido por defecto en Django. Actualmente la aplicación sólo es compatible con PostgreSQL 9+. El modelo de datos detallado se incluye en la siguiente sección.
- Las páginas generadas por la aplicación están compuestas por *markup* compatible con HTML5, así como por código Javascript. Las visualizaciones están realizadas sobre la librería Javascript de código abierto D3, cuyo formato de salida es SVG, compatible sin necesidad de *plugins* adicionales con Firefox, Chrome, Safari e Internet Explorer 9, así como con las versiones móviles de estos navegadores. La aplicación no incluye ningún formato ni componente multimedia adicional, como Flash, vídeo o audio.
- El código de los controladores — encargados de procesar las peticiones de los usuarios, recuperar la información del modelo y entregársela a las vistas — está desarrollado en Python.

La aplicación no incluye ningún proceso de segundo plano (*daemons* o servidores adicionales), ni tareas *batch* que deban ejecutarse regularmente. Sí que incluye una serie de procesos de carga de datos, escritos en Python, como se detalla en la sección “Instalación”.

# Esquema de datos



# Instalación

Esta sección explica cómo instalar la aplicación web en un nuevo entorno.

## Entrega del software y gestión de configuraciones

El código de la aplicación está dividido en dos repositorios públicos:

- La parte principal de la aplicación, compartida por todos los despliegues en distintas administraciones (por ejemplo, comunidades de Aragón y Canarias, o los Ayuntamientos de Coruña y Barcelona), está disponible en Github, en la siguiente dirección: <https://github.com/civio/presupuesto>.
- Además, un segundo repositorio recoge las modificaciones específicas de la instalación para el Gobierno de La Rioja: identidad corporativa, datos y modificaciones solicitadas durante el desarrollo. El segundo repositorio está disponible en <https://github.com/civio/presupuesto-larioja>.

El primer paso para proceder a la instalación es descargar el contenido del primer repositorio en el directorio donde se vaya a instalar la aplicación. A continuación debe descargarse el segundo repositorio dentro del primero:

```
$ git clone https://github.com/civio/presupuesto.git
$ cd presupuesto
$ git clone https://github.com/civio/presupuesto-larioja.git
```

## Dependencias

La aplicación se ejecuta sobre una infraestructura técnica con las siguientes características principales:

- Servidor web Apache
- Base de datos PostgreSQL 9+
- Python 2.7

La aplicación necesita instalar una serie de paquetes Python, que se detallan en **requirements/server.txt**. Las principales son:

- Django
- Jinja2
- psycopg2
- django\_compressor

Las dependencias pueden ser instaladas utilizando gestores de paquetes de Python como **virtualenv** o **pip**. Usando este último, por ejemplo, haríamos:

```
$ pip install -r requirements/server.txt
```

El componente 'psycopg2' es el responsable de la conectividad con la base de datos Postgres, y tiene una parte nativa que debe ser compilada en el momento de la instalación.

## Configuración del acceso a la base de datos

La aplicación lee la configuración para su acceso a base de datos del fichero '**local\_settings.py**', que debe crearse en cada despliegue, basándose en el contenido del fichero **local\_settings.py**, incluido en el repositorio. Así, dado el siguiente fichero de configuración:

```
ENV = {
    'THEME': 'presupuesto-larioja',

    'DEBUG': False,
    'TEMPLATE_DEBUG': False,

    # Database
    'DATABASE_NAME': 'vpresup',
    'DATABASE_USER': 'user',
    'DATABASE_PASSWORD': 'password',
    'DATABASE_PORT': '5453',
    'SEARCH_CONFIG': 'unaccent_spa',

    # Caching
    'CACHES': {
        'default': {
            'BACKEND': 'django.core.cache.backends.locmem.LocMemCache'
        }
    }
}
```

La aplicación se conectará al esquema de base de datos '**vpresup**' usando el usuario '**user**' y la contraseña '**password**'.

Una vez creado el esquema -usando las herramientas de gestión de Postgres, algo que no se cubre en este documento de instalación- podemos continuar con la instalación. Desde el directorio de la aplicación creamos las tablas de la base de datos de forma automática:

```
$ python manage.py migrate
```

## Carga del glosario

La aplicación incluye un glosario de términos relacionados con la economía y los presupuestos. Estos términos se actualizan a partir de ficheros de texto CSV, en castellano '**presupuesto-larioja/data/glosario\_es.csv**', cuyo formato es el siguiente:

- Título/Término
- Descripción

Existe un script para manipular esta información:

```
python manage.py load_glossary --language=<idioma>
```

Al instalar la aplicación por primera vez debemos cargar los términos del glosario para que estén disponibles:

```
$ python manage.py load_glossary --language=es
Cargando glosario de .../data/glosario_es.csv...
Cargando término 'Inflación'...
...
```

## Carga de la lista de administraciones

La aplicación necesita disponer de la lista de administraciones cuyos presupuestos se están mostrando. Esta lista se actualizan a partir de un fichero de texto CSV, '**presupuesto-larioja/data/entidades.csv**', cuyo formato es el siguiente:

- Identificador único
- Tipo de entidad (comunidad autónoma, comarca o municipio)
- Nombre de la entidad

Existe un script para manipular esta información:

```
python manage.py load_entities
```

Al instalar la aplicación por primera vez debemos cargar la lista de organismos para que estén disponibles:

```
$ python manage.py load_entities
Cargando lista de organismos de .../data/entidades.csv...
...
```

## Carga de estadísticas oficiales

La aplicación utiliza dos estadísticas oficiales — número de habitantes de comarcas y municipios, así como inflación anual en Madrid — a la hora de mostrar la información presupuestaria, permitiendo hacer cálculos de gasto per-cápita, y ajustando las cantidades a la inflación. Estos datos estadísticos se actualizan a partir de dos ficheros de texto CSV. El primero, '**presupuesto-larioja/data/inflacion.csv**', tiene el siguiente formato:

- Año
- Inflación anual

El segundo, '**presupuesto-larioja/data/poblacion.csv**', contiene las siguientes columnas:

- Identificador del organismo (comarca, municipio o comunidad autónoma)



- Nombre del organismo
- Año
- Número de habitantes

Existe un script para manipular esta información:

```
python manage.py load_stats
```

Al instalar la aplicación por primera vez debemos cargar las estadísticas para que estén disponibles:

```
$ python manage.py load_stats
Cargando estadísticas oficiales de ../data/estadisticas.csv...
Cargando estadísticas oficiales para el año 2012...
Cargando estadísticas oficiales para el año 2013...
...
```

## Carga de los datos presupuestarios de La Rioja

La aplicación lee la información presupuestaria y de ejecución a partir de los ficheros descargados del portal de datos abiertos del Gobierno de La Rioja. Los ficheros se almacenan en la carpeta '**data**' de la aplicación, desde donde pueden ser cargados y analizados mediante el comando:

```
python manage.py load_budget <año_a_cargar> --language=<idioma>
```

Así, para cargar los datos del año 2016 en castellano, desde el directorio donde se haya instalado la aplicación:

```
$ python manage.py load_budget 2016 --language=es
Cargando presupuesto de ../data/es/municipio/2016...
Cargando lista de secciones de ../data/es/municipio/2016/estructura_organica.csv...
Cargando jerarquía económica de ../data/es/municipio/2016/estructura_economica.csv...
...
```

A la hora de estimar el espacio de almacenamiento en base de datos necesario para el funcionamiento de la aplicación podemos fijarnos exclusivamente en el espacio utilizado por los presupuestos, ya que son el único elemento de tamaño no despreciable (frente al glosario y las estadísticas oficiales):

- Los ficheros CSV con los datos presupuestarios de un año ocupan menos de 1MB por año.
- Una base de datos PostgreSQL con los presupuestos de 2011 a 2017 cargados, tanto en castellano como en euskera, ocupa, en el entorno de desarrollo, aproximadamente 10MB.

Ya que la aplicación no almacena ninguna información en el día a día, podemos esperar un crecimiento en su espacio necesario de aproximadamente 2MB al año.

## Rendimiento

Algunas de las consultas que la aplicación realiza en la base de datos implican leer una cantidad considerable de datos: así, para calcular el total de gasto para una política concreta, como “Sanidad”, es necesario leer y sumar todas las líneas del presupuesto incluidas en esa política (varios cientos). Pero toda la información que maneja la aplicación es estática — no se realiza ninguna actualización en ningún momento —, y es independiente del usuario de la web, por lo que puede ser cacheada indefinidamente sin ningún problema, utilizando los mecanismos estándares de Django. Así, el impacto de la aplicación en el rendimiento del servidor es mínimo después de haber servido las primeras peticiones.

La aplicación utilizará el caché en memoria incluido por defecto en Django, sin ninguna dependencia con servicios externos de caché tipo ‘memcached’.

# Mantenimiento

## Dependencias con otros componentes

La aplicación no tiene ninguna dependencia con ningún otro servicio o componente del entorno, más allá del servidor web y la base de datos. La aplicación:

- no manda correos
- no muestra vídeos, ni en streaming ni como descarga
- no tiene cuentas de usuario, por lo que no se conecta a ningún servidor de autenticación (LDAP o similar)

## Otros procesos y tareas

La aplicación no necesita ejecutar ningún proceso (*daemon*) adicional a los encargados de servir las peticiones de los usuarios. Tampoco necesita configurar ningún tipo de tarea periódica / *batch*.

## Seguridad y accesos

La información disponible en la aplicación está formada por datos públicos ya disponibles en la web de Datos Abiertos del Ayuntamiento de Madrid. No existe, por tanto, ninguna información confidencial.

Por otro lado, la aplicación no permite crear ningún tipo de cuenta de usuario, ni almacena ninguna información de sus usuarios.

Las tareas de mantenimiento (carga de datos) se realizan vía una conexión SSH al servidor. El control de acceso de la aplicación es, por tanto, el de las credenciales necesarias para acceder al servidor vía SSH.

## Tareas regulares de mantenimiento

La única tarea regular de mantenimiento que precisa la aplicación es la rotación y/o limpieza de sus *logs* periódicamente. El tamaño de estos *logs* es similar al de cualquier otra aplicación web, y depende exclusivamente del volumen de tráfico que reciba, por lo que es difícil de estimar a priori.

No existen tareas adicionales, ya que:

- La información mostrada por la aplicación es estática, no crece en el día a día, por lo que no son necesarios procesos de limpieza y archivo en la base de datos.
- Al no existir cuentas de usuario, no existen peticiones de alta, baja o soporte.
- Al no existir datos privados, en caso de fallo grave del sistema (pérdida del sistema de ficheros o base de datos) la aplicación puede restaurarse a partir de cualquier copia de seguridad. Si no existieran o se perdieran es posible reinstalar

el sistema desde cero, a partir del código y datos almacenados en Github — cómo se explicó en la sección previa — sin que se pierda información.

### Tareas extraordinarias de soporte

Anualmente será necesario actualizar la información almacenada en la aplicación, para añadir nuevos presupuestos. Esta tarea se compone de los siguientes pasos:

- Descargar los datos del nuevo presupuesto de la web de datos abiertos, y agregarlos al repositorio de gestión de versiones de la aplicación.
- Actualizar los ficheros '`data/inflacion.csv`' y '`data/poblacion.csv`', añadiendo la población de Madrid y la inflación en ese año. (El fichero actualmente contiene los enlaces a las páginas del INE donde se actualizan estos datos.)
- Desplegar la nueva versión del repositorio en producción.
- Cargar el nuevo presupuesto en la base de datos, utilizando la tarea '`load_budget[año]`' descrita anteriormente. Si disponemos además de información de ejecución presupuestaria o recaudación podemos cargar ésta utilizando las tareas correspondientes descritas anteriormente.
- Cargar las nuevas estadísticas en la base de datos, utilizando la tarea '`load_stats`' descrita anteriormente.
- Por último, necesitamos indicarle a la aplicación que limpie su caché, para que los nuevos datos sean visibles: para aplicaciones Django desplegadas vía Apache esto se hace "tocando" el fichero `project/wsgi.py` (`touch project/wsgi.py`).