*E1.8*

Tecnologías Utilizadas en Callejero de Uruguay

*v1.0*

27 Febrero 2020

Para cualquier consulta o cuestión pendiente sobre la oferta la persona de contacto es:

|  |  |  |
| --- | --- | --- |
| Nombre | Teléfono | e-mail |
| José Vicente Higón Valero | +34 600 55 44 74 | [jvhigon@gvsig.com](mailto:jvhigon@gvsig.com) |

Información general:

Asociación gvSIG

CIF G98125362

Calle Ángel Guimerá 61, puerta 3

[http://www.gvsig.com](http://www.gvsig.com/)

Control de cambios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Responsable | Organismo | Descripción del cambio | Fecha |
| v.1.0 | Nacho Brodin | Asoc gvSIG | Documento inicial | 27/02/2020 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Índice**

[1. Tecnologías](#_gjdgxs) **4**

[Componentes cartográficos](#_gzqfnhr9cxxs) 4

[Almacenamiento](#_vqdhxkn7t0c3) 6

[Seguridad](#_asoey0t2eltr) 7

[Servicios a través de API](#_a6wllipc3zq4) 8

[Servidores de datos](#_j79uigqsib35) 9

[Comunicaciones](#_t4aj8iqz6azt) 11

# 1. Tecnologías

Este apartado describe las tecnologías utilizadas dentro del proyecto de Callejero, así como del componente de administración del mismo que está incluido dentro de gvSIGOnline por lo que muchas de las tecnologías utilizadas en gvSIGOnline son las usadas en la administración del callejero. Se clasifican las tecnologías por grupos, componentes cartográficos, almacenamiento, seguridad, servicios a través del API, servidores de datos y comunicaciones:

## Componentes cartográficos

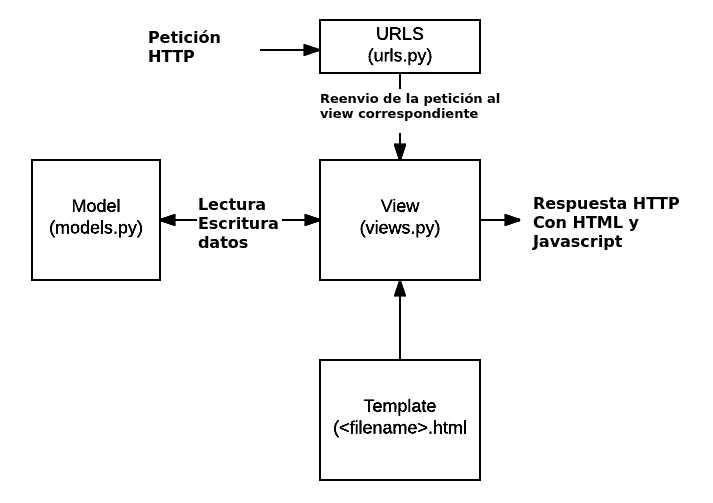
* Geoportal: se trata de gvSIG Online, aplicación web basada en Python-Django que se ejecuta como servicio de systemd a través del software uWSGI.



La tecnología utilizada en el visor es HTML con Javascript utilizando el mecanismo de template de Django para la parte de administración de gvSIGOnline incluyendo el administrador de direcciones del callejero. El siguiente diagrama muestra el principal flujo de datos y entidades necesarias para ser implementadas cuando se maneja una respuesta/petición en HTTP en gvSIGOnline.

Los principales elementos que se crean son:

* Mapeadores URL para reenviar las URLs admitidas (y cualquier información codificada en las URLs) a las funciones de vista apropiadas.
* Funciones de vista para obtener los datos solicitados desde los modelos, crear una página HTML que muestre los datos, y devolverlo al usuario para que lo vea en el navegador.
* Plantillas usadas por las vistas para renderizar los datos.



El cliente web utiliza el “trío tecnológico” formado por html, javascript y css para el renderizado. Html como lenguaje hipertexto para la elaboración de las páginas web, javascript como lenguaje de programación interpretado que permite interactuar con el navegador y CSS como manejo de estilos en el navegador web.

* Mapas-vec: componente encargado de servir los mapas de fuentes de datos vectoriales (WMS y WMTS) y los servicios de objetos geográficos (WFS). Se empleará el software Geoserver y Geowebcache que se ejecutan en el contenedor de Servlets Tomcat.



Geoserver es un servidor de código abierto desarrollado en Java que permite publicar y editar datos geoespaciales. Diseñado para la interoperabilidad, publica datos de las principales fuentes de datos espaciales usando estándares abiertos. Geoserver integra con Geowebcache para la generación de datos cacheados. Esto es importante para no sobrecargar el servidor.

* Mapas-ras: componente encargado de servir mapas de fuentes de datos raster (WMS y WMTS) mediante Geoserver y Geowecache.
* Print: componente con el software Mapfish encargado de generar los PDF para imprimir.



* Estándares OGC: Las comunicaciones cartográficas se utilizan realizando los protocolos del Open Geospatial Consortium. Estos se utilizan desde gvSIGOnline y son los siguientes
  + WFS - Web Feature Service o Servicio de entidades vectoriales que proporciona la información relativa a la entidad almacenada en una capa vectorial (cobertura) que reúnen las características formuladas en la consulta.
  + WMS - Web Map Service o Servicio de mapas en la web que produce mapas en formato imagen a la demanda para ser visualizados por un navegador web o en un cliente simple.
  + WCS - Web Coverage Service o Servicio de coberturas en la web (datos raster).
  + CSW - Web Catalogue Service o Servicio de catálogo.
  + WMTS - Servicio de mosaico de mapas web para servir mosaicos de mapas georreferenciados calculados en tiempo de ejecución o preprocesados ​​a través de Internet.



## Almacenamiento

* BBDD: base de datos geoespacial PostgreSQL/Postgis. La base de datos principal e histórico se encuentran en esquemas separados y se implementan triggers que disparan código en procedimientos almacenados en la base de datos desarrollados en PLSQL. Este es un lenguaje que permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones.



Las consultas a la base de datos se realizan por tres vías fundamentalmente:

El servicio de consulta de callejero se conecta a través de JDBC con Spring utilizando JdbcTemplate.



El administrador de direcciones se conecta por dos medio. Por un lado, utiliza el sistema de modelos interno de Django que permite a gvSIGOnline mapear tablas de Postgres en objetos Python. Este método se utiliza para la base de datos de la propia aplicación. Por otro, lado se utiliza conexión Odbc principalmente para conectar a la base de datos de callejero ya que las tablas no se mapean al modelo de Django. ODBC es el Open DataBase Connectivity, estándar de acceso a las bases de datos. El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos almacene los datos.

En cualquier caso cualquier tipo de consulta acaba convirtiéndose finalmente en peticiones SQL qué son las que se soportan en una base de datos relacional como Postgres.

* Filesystem: componente que servirá de sistema de ficheros compartido para el cluster, típicamente con acceso NFS. También ofrecerá el acceso a imágenes y recursos para descarga.

## Seguridad

* WAF: será el encargado de realizar de proxy reverse de todos los servicios de la capa de aplicación. También realiza tareas de seguridad (a través de mod\_security) y de Firewall (a través de firewalld)



* HTTPS: La comunicación con el servidor web así como los servicios se hace a través del protocolo seguro https. Este es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto. Utiliza un cifrado basado en la seguridad de textos SSL/TLS para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo se consigue que la información sensible (usuario y claves de paso normalmente) no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar.



## Servicios a través de API

El API definido tanto para los servicios de Callejero como para el administrador de direcciones se realiza con el protocolo REST. Este es un protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar). Geoserver también dispone de un API REST para la realización de acciones de consulta y edición desde gvSIGOnline.



Para la documentación del API REST se utiliza Swagger. Swagger es una especificación abierta para definir API REST. El documento Swagger especifica la lista de recursos disponibles en la API REST y las operaciones a las que se puede llamar en estos recursos. El documento Swagger especifica también la lista de parámetros de una operación, que incluye el nombre y tipo de los parámetros, si los parámetros son necesarios u opcionales, e información sobre los valores aceptables para estos parámetros. Además, el documento Swagger puede incluir un esquema JSON adicional que describa la estructura del cuerpo de la solicitud que se envía a una operación en una API REST, y el esquema JSON describe la estructura de los cuerpos de respuesta devueltos de una operación.



## Servidores de datos

Django: gvSIGOnline y el administrador de direcciones está desarrollado con Django. Este es un framework de desarrollo web de código abierto escrito en Python que sigue el patrón de diseño Modelo–vista–template. En este se recibe una petición del navegador, se ejecuta la vista correspondiente y se renderiza el Template para que el navegador muestre el HTML resultante.



Los servicios de callejero está implementados utilizando Spring. Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Por lo tanto el lenguaje de programación en el que están desarrollados dichos servicios es Java. Spring proporciona facilidades para el manejo de los siguientes aspectos. Aunque no todos son relevantes ni utilizados en el contexto del proyecto si proporciona un framework de apoyo sobre todo en conectividad, control de dependencias y gestión de transacciones con la base de datos..

En concreto, se utiliza Spring Boot para la implementación de los servicios de búsqueda de direcciones. Spring Boot permite generar microservicios de una manera sencilla, y publicarlos en un contenedor de Servlets o generar un “fat-jar”, un fichero .jar con todo lo necesario para ejecutar el servidor embebido junto con las dependencias que se necesiten.

Spring Boot permite también una integración sencilla con otra tecnología que hemos usado en el proyecto: OpenShift. La integración es a nivel de compilación y creación de la imagen como a la hora de ofrecer servicios internos a OpenShift para monitorizar el estado de la imagen (POD).

### Configuración del Servicio de SpringBoot:

El servicio de DireccionesUy se configura en base a un fichero application.properties y unas variables de entorno:

En el fichero application.properties encontramos:

server.port = ${OPENSHIFT\_CALLEJEROUY\_PORT}

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect

spring.datasource.url = jdbc:postgresql://${OPENSHIFT\_CALLEJEROUY\_DB\_HOST}:${OPENSHIFT\_CALLEJEROUY\_DB\_PORT}/tsubasa

spring.datasource.username = ${OPENSHIFT\_CALLEJEROUY\_DB\_USERNAME}

spring.datasource.password = ${OPENSHIFT\_CALLEJEROUY\_DB\_PASSWORD}

spring.jpa.hibernate.ddl-auto=create-drop

logging.level.org.hibernate.SQL=ERROR

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=ERROR

management.endpoints.web.exposure.include=\*

Las variables de entorno que definen el servicio empiezan con OPENSHIFT y definen el puerto en el que se quiere poner el servicio, la IP o host de la base de datos, el puerto de esa base de datos y el usuario y contraseña para conectar a la base de datos.

Hay otra variable de entorno que es opcional:

${OPENSHIFT\_CALLEJEROUY\_APIS\_ALLOWED}

Si se define, se pueden configurar los servicios que van a devolver resultados.

Por ejemplo, si asignamos OPENSHIFT\_CALLEJEROUY\_APIS\_ALLOWED = “v1”, solo las nuevas versiones de los servicios se van a poder usar.

“v0” solo para la versión compatible con la versión vieja. Si se desea más nivel de detalle, por ejemplo solo v0 SugerenciaCalleCompleta y Localidades, podríamos asignar “Localidades, SugerenciaCalleCompleta”.

Si la variable no está definida (por defecto), todos los servicios están habilitados.



Los servicios de callejero están implementados utilizando Java. Este es un lenguaje de programación orientado a objetos. La idea de Java es que pueda realizarse programas con la posibilidad de ejecutarse en cualquier contexto, en cualquier ambiente, siendo así su portabilidad uno de sus principales características. Una de las mayores ventajas que aporta java dentro del proyecto es la amplia disponibilidad de librerías y la fácil conexión con la base de datos Postgres a través de JDBC.



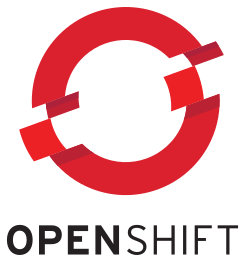
## CI / CD

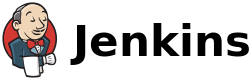
Los servicios de callejero se compilan y se publican siguiendo un esquema de CI/CD (Continuous Integration / Continuous Deployment). En AGESIC se utiliza OPENSHIFT, una plataforma para la gestión de contenedores desarrollada por RedHat que permite manejar de una manera integrada Kubernetes y Docker.

No vamos a entrar en detalle para describir todas las posibilidades de OpenShift, hay un documento en el que se explica con más detalle cómo se integran todos los componentes, pero de manera abreviada, existe un “pipeline”, una serie de flujo de trabajo como el que sigue:

* Partimos del código fuente en el GitLab de AGESIC.
* En OpenShift hemos generado unas imágenes que permiten compilar, testear y analizar el código fuente.
* Se genera una imagen basada en CentOS con el servicio SpringBoot.
* Esa imagen pasa por distintos entornos donde se prueba y testea. Una vez validada, se publica en el entorno adecuado.

Dentro del “pipeline”, es de vital importancia Jenkins, un servidor de automatización que se encarga de orquestar todo el proceso.





## Comunicaciones

Todas las comunicaciones de consulta del API de callejero, así como el API de gvSIGOnline se realiza a través de HTTP en su versión segura HTTPS, como ya se ha comentado.

Las comunicaciones con la base de datos a través de sus múltiples formas (ODBC, JDBC, Django) se realiza a través de la comunicación por sockets a través del puerto 5432 con la base de datos Postgres. Socket es una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.