Diseño e Implementación de un sistema de URL-Shortening compuesto de un servidor de REST-WS sobre CloudFoundry y una aplicación cliente desarrollada en GWT

Diseño del sistema

Autor: Carlos Yagüe Martín-Lunas

1. Introducción

Este sistema corresponde a un prototipo de un sistema de URL-Shortening basado en la utilización de la plataforma CloudFoundry.com y de tecnologías como GWT, Spring, MongoDB, REST/SOAP, Maven, etc.

El sistema está compuesto por dos componentes claramente diferenciados:

- Servidor de WS (`url-shortener-ws`): Se trata de una aplicación desplegada en CloudFoundry.com que tiene en su contexto dos servicios web basados en la arquitectura REST:
  - `shortenURL(longUrl)`: Este servicio web recibe como argumento una URL larga y devuelve como resultado una URL corta.
  - `expandURL(shortUrl)`: Este servicio web hace la operación inversa, dada una URL corta, devuelve una URL larga.

  Dicha aplicación está desplegada sobre la plataforma CloudFoundry.com en la siguiente ruta: http://url-shortener-ws.cloudfoundry.com/

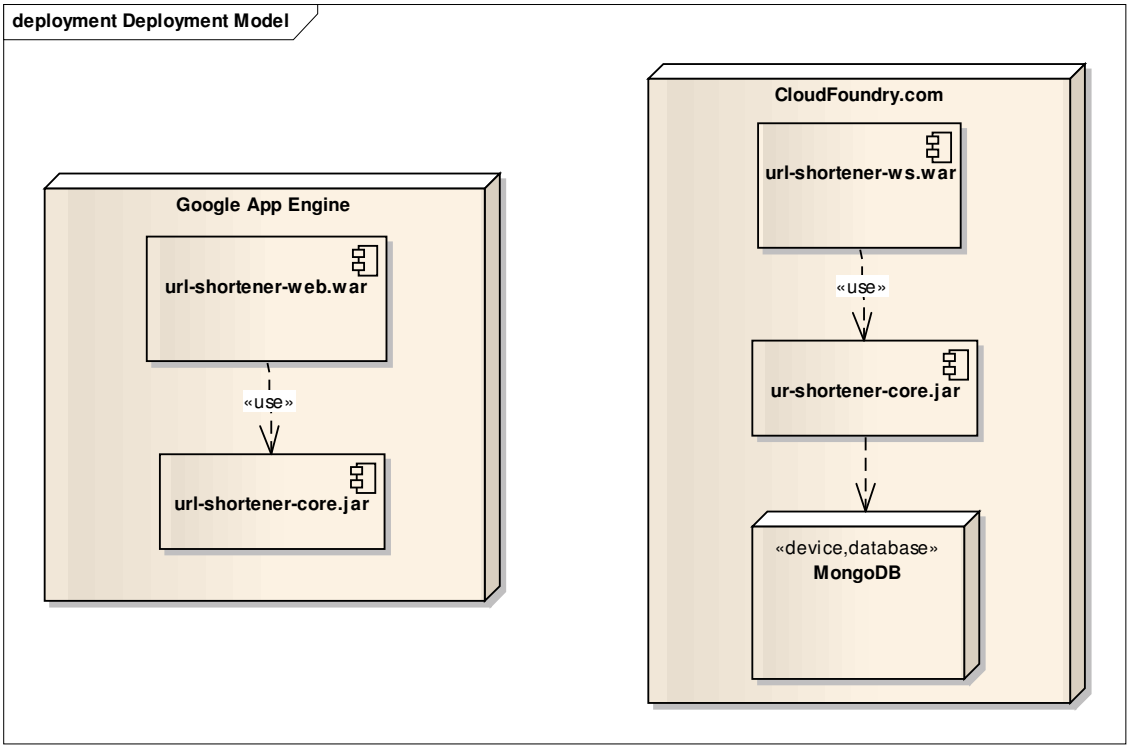
- Cliente de WS (`url-shortener-web`): Corresponde a una aplicación web que hace las veces de cliente del componente anterior. Esta aplicación está desarrollada en GWT (*Google Web Toolkit*) y también se encuentra desplegada en una plataforma de Google, en la *Google App Engine*: http://orgurlshortener.appspot.com/

En los siguientes apartados se explica en detalle ambos componentes.

2. Estructura física del sistema

Físicamente, el sistema corresponde a dos aplicaciones independientes desplegadas sobre dos plataformas distintas. Ambas aplicaciones utilizan una misma librería (`url-shortener-core`) que ofrece los servicios de lógica de negocio comunes.

El siguiente diagrama de despliegue muestra la relación entre los componentes desarrollados, las plataformas de ejecución y el SGBD (Sistema Gestor de Base Datos) utilizado.

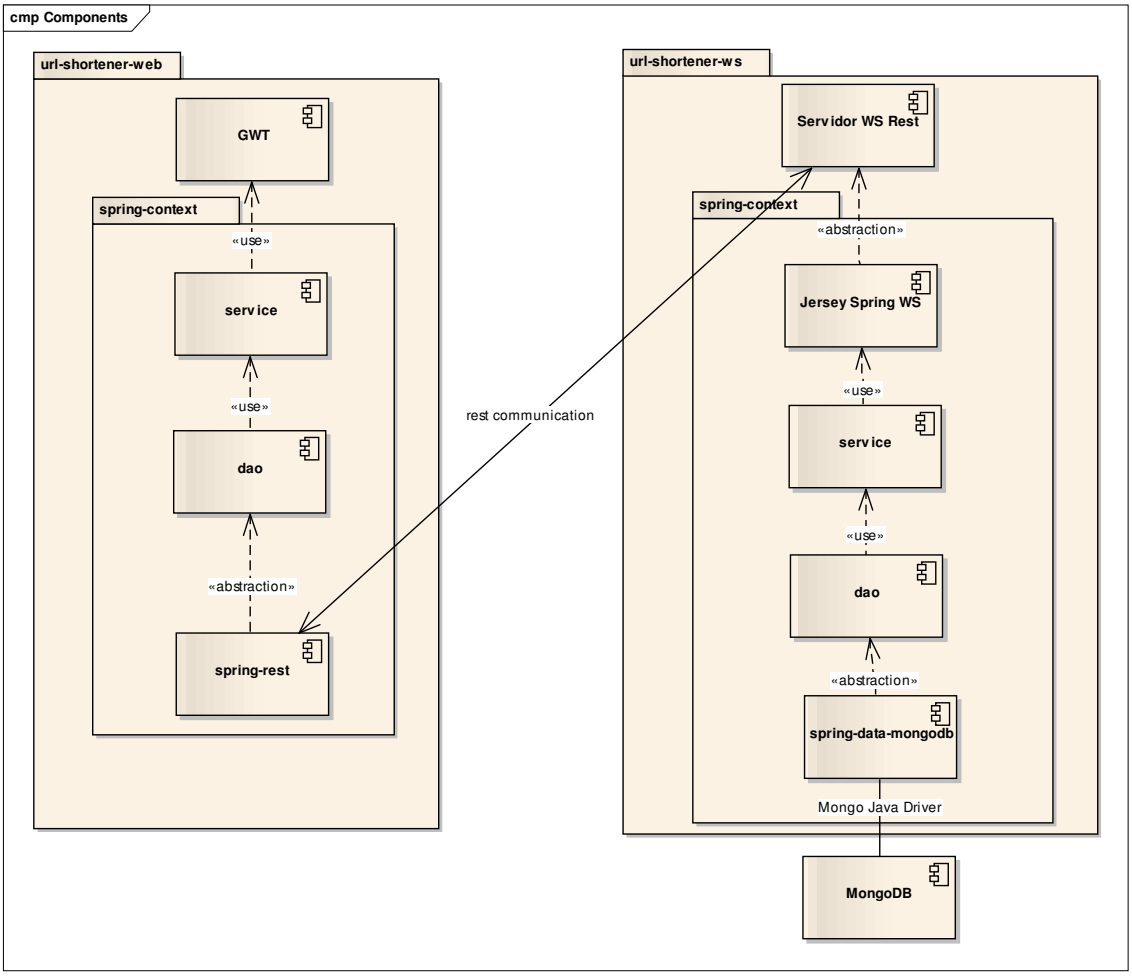


Como SGBD se ha utilizado MongoDB (http://www.mongodb.org/) que corresponde una base de datos orientada a documentos. Dicho SGBD tiene como cualidad principal la de minimizar la utilizar uniones entre tablas ya que soporta entre sus tipos de datos tanto documentos como Arrays lo que mejora su velocidad de consulta e inserción.

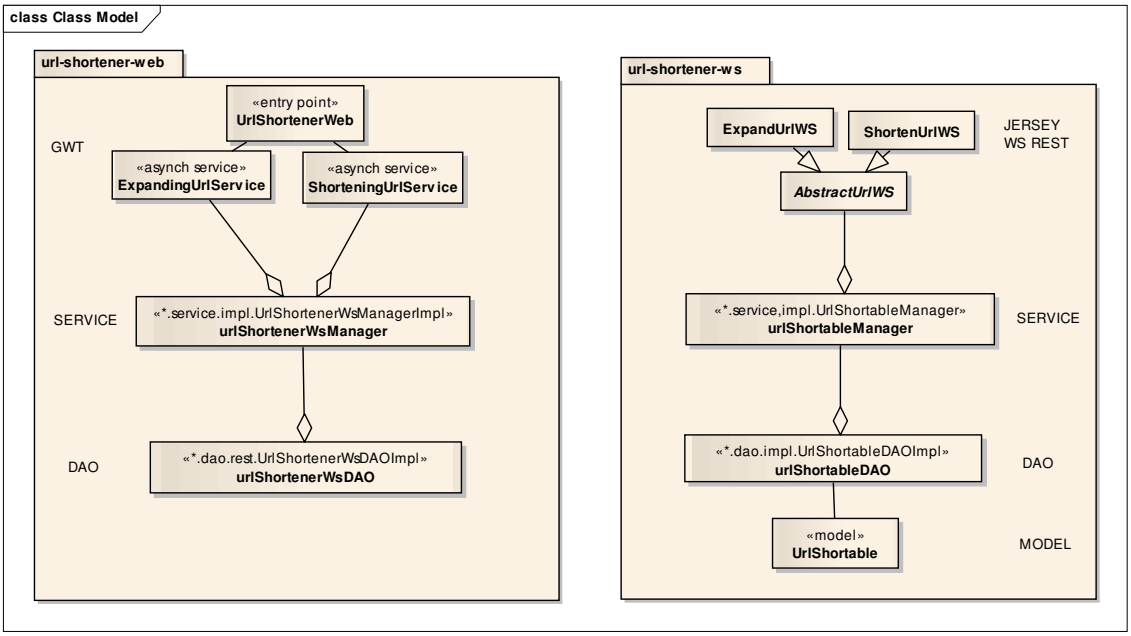
La utilización de este SGBD es adecuada para aplicaciones que funcionan en la plataforma Cloud Foundry ya que dicha plataforma cuenta con dicho SGBD entre sus servicios.

3. Estructura lógica del sistema

A continuación se muestra la estructura lógica de ambas aplicaciones. Como se observa en el siguiente gráfico, se muestran tanto las tecnologías empleadas como su diferenciación funcional.



Ambas aplicaciones utilizan Spring Framework como mecanismo de inyección de dependencias, lo que genera un subcontexto a nivel de aplicación denominado *"Contexto de Spring"*. Las dependencias se definen en el siguiente diagrama:



- **url-shorterner-web**
  - GWT: Capa de presentación implementada en GWT. Se trata de un punto de entrada o *entry point* `UrlShortenerWeb` que tiene asociado dos servicios asíncronos:
    - `ShorteningUrlService`: Servicio asíncrono que se comunicará con el servidor de WS para recortar una URL larga.
    - `ExpandingUrlService`: Servicio asíncrono que se comunicará con el servidor de WS para expandir una URL corta.

    Ambos servicios dependen del servicio `urlShortenerManager` incluido en la capa de servicios.

  - Contexto de Spring:

    - SERVICE: Capa de servicios de lógica funcional. Se usará el manager `urlShortenerManager`. Este servicio depende del DAO de servicios web `urlShortenerWsDAO`.

    - DAO: Capa de acceso a datos. Se usará el DAO `urlShortenerWsDAO` que realizará la comunicación con el servidor de WS.
      - SPRING-REST. Tecnología con la que se ha implementado el DAO `urlShortenerWsDAO`. Spring Rest corresponde únicamente a la plantilla definida por Spring que cumple la arquitectura REST. Dicha funcionalidad se

encuentra definida en el módulo SPRING-WEB.

- **url-shortener-ws**
  - JERSEY SPRING WS: Servicios Web implementados utilizando el API de REST:
    - `ExpandUrlWS`: Servicio Web de expansión de URLs.
    - `ShortenUrlWS`: Servicio Web encargado de recortar las URLs.

    Ambos WS tienen como dependencia al *manager* `urlShortableManager`.

  - SERVICE: Servicios de lógica de negocio que aportan la funcionalidad de recortar/expandir URLs: `urlShortableManager`

  - DAO: Capa de acceso a datos. Se usará el DAO `urlShortableDAO` que accederá al SGBB MongoDB.
    - SPRING-DATA-MONGOBD: Tecnología utilizada para implementar el DAO `urlShortableDAO`. Es un API de Spring preparado para interactuar con el driver Java de MongoDB.

  - MODEL: Entidad mapeada con la *collection* `urlShortable` de la BBDD de MongoDB

4. Entorno de desarrollo

Se han utilizado para desarrollar este sistema el siguiente conjunto de herramientas:

- JDK 1.6.0_17
- IDE: Eclipse Indigo
- Plugins Eclipse
    - M2Eclipse 1.0.0
    - Google Suite Plugin 2.3.3
    - Subclipse 1.6.18
    - HTTP4e 4.2.5
- Compilación/Despliegue: Maven 3.0.2
- Servidor de aplicaciones: Apache Tomcat 6
- SGBD: MongoDB 1.8.3
- Forja/SVN: CodeGoogle: http://code.google.com/p/openurlshortener/
- Despliegues en CloudFoundry: Ruby 1.9.2

5. Instalación y configuración

Para generar los Wars basta con utilizar Maven sobre el directorio url-shortener:

```
$url-shortener> mvn clean install
```

Partiendo de los siguientes WARs su instalación es simple. Estos son los puntos a tener en cuenta:

- `url-shortener-web-1.0.war`
  - Plataforma de destino: *Google App Engine*
  - Para configurar el servidor de WS se debe especificar en la propiedad `CLOUD_HTTP_SERVER` del siguiente fichero de configuración: `/WEB-INF/classes/application.properties`
  - Herramienta de despliegue: Google Suite Plugin 2.3.3

- `url-shortener-ws-1.0.war`
  - Plataforma de destino: *CloudFoundry.com*
  - No requiere configurar nada. En lo que respecta a la configuración la base de datos MongoDB, la plataforma CloudFoundry se encarga de configurar el acceso a un SGBD MongoDB por si solo.
  - Herramienta de despliegue: Herramienta VMC de Ruby.