Trabajo de Fin de Grado

Grado en Ingeniería Informática

# Un portal de transparencia para datos libres


**Autor**
Germán Martínez Maldonado

**Tutor**
Juan Julián Merelo Guervós




Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación


—

Granada, 7 de junio de 2015

# Prefacio

## Un portal de transparencia para datos libres

Germán Martínez Maldonado

## Resumen

**Palabras clave**: software libre, transparencia, datos abiertos, sistema de control de versiones, aprovisionamiento, tests, integración continua, despliegue automático

Se pretende desarrollar una plataforma para la transparencia que sea desplegable en un infraestructura física o virtual basándose en el trabajo realizado en el portal UGR Transparente, respaldada por los datos abiertos publicados en la plataforma OpenData UGR.

Este desarrollo tendrá como objetivo que el resultado sea una plataforma totalmente basada en el software libre que se pudiera adaptar con facilidad a otro organismo, teniendo en cuenta funcionalidades y requisitos obligatorios, además de aspectos de accesibilidad y escalabilidad.

La herramientas básicas a usar serán un sistema de control de versiones y un sistema de desarrollo colaborativo que albergue el proyecto, aque además use dicho sistema de control de versiones. También se quiere que la plataforma se pueda desarrollarr y administrar simultáneamente, por lo que para convertir todo esto es un proceso más ágil e ininterrumpido se usarán otras herramientas que permitan lo siguiente:

- Realizar aprovisionamiento de las infraestructuras.
- Validación mediante tests unitarios.
- Comprobación de conflictos mediante integración continua.
- Actualizaciones mediante despliegue automático.

# A transparency portal for open data

Germán Martínez Maldonado

## Extended abstract

**Keywords**: free software, transparency, opendata, version control system, provisioning, tests, continuous integration, automated deployment

It is intended to develop a platform for transparency that is deployable on a physical or virtual infrastructure based on work done on the site UGR Transparente, backed by open data published in the Open Data UGR.

This development aims that the result is a platform completely based on free software that could be easily adapted to another organization, taking into consideration features and mandatory requirements, as well as issues of accessibility and scalability. The platform which presents the data will develop in Node.js, which is a programming environment that operates at runtime based on the Google's V8 JavaScript engine; also will use Express for web application development and Jade to generate HTML files based on templates. Moreover, the platform that contains the data is based on CKAN, a open-source data portal platform developed by The Open Knowledge Foundation, a not-for-profit organisation that promotes creating and sharing knowledge freely.

Being free software, this development is not limited to the working group that started the work, but rather is focused on that anyone can make their contribution to the project; therefore, for all this can be handled is necessary a version control system, in this case Git will be used, which is practically a standard in this area. In addition, it will also be used a recognized and open collaborative based development platform as is GitHub, which is also integrated with Git, providing ease in the development and distribution of work, because any who access the project repository can freely copy it for its open license.

An important point to facilitate system administration is the provisioning. Provisioning a machine, as the name suggests, is to provide to the machine all the resources needed for its performance, in our case we refer to all the software necessary for that the platform developed works properly in such infrastructure. In the case of transparency portal, as the entire project is hosted on GitHub, we can set up a utility like Ansible to download the project and then install on the machine all the necessary software so that the platform can function properly. In contrast, such as the open data portal

is not a own development, their provisioning will consist of installing and customizing CKAN for the University of Granada.

During the development of any software application, new features are introduced to the software gradually, so must be ensured that the new features do not compromise the overall stability of the project. For this purpose the unit tests are written, which could be considered as small programs within the system that handles check by assertions and behavior patterns that all elements operate as they should. To verify that our unit tests are sufficient also need to pass a coverage test that tells us that all the functionality of our platform are properly validated by the corresponding unit test. There are many libraries that allow both actions, but for the facility of work between them, unit test will be passed with Mocha and coverage test will be with Istanbul.

Once we have the tests written, we don't need concern ourselves with run them manually, we have the option of running in a external platform every time we make changes and directly get the results, this is continuous integration. Continuous integration will be made with Travis CI and the operation is very simple: every time we make a change in GitHub, Travis CI download the latest version of the code, builds it and passes the tests we have written, to finish by returning the results of the tests that will make us know whether changes have produced a conflict in the system.

The last thing to keep in mind is about deploying the changes automatically on our server, can be a tedious procedure having to manually access to the infrastructure of our platform every time that we want to apply the new changes we have made in the application, so we'll use the tool Flightplan for automatic deployment; only with specify our server as the target we can set a series of tasks that will make that automatically for the updates we're making become effective on the main machine.

By using these tools we are getting an application in which development and management are closely related and automated, which makes the application much more reliable and easy recovery in case of problems.

Yo, **Germán Martínez Maldonado**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado (*Un portal de transparencia para datos libres*) en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Germán Martínez Maldonado

Granada, a 7 de junio de 2015

D. **Juan Julián Merelo Guervós**, Profesor del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado ***Un portal de transparencia para datos libres***, ha sido realizado bajo su supervisión por **Germán Martínez Maldonado**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 7 de junio de 2015.

**El tutor:**

**Juan Julián Merelo Guervós**

# Agradecimientos

[Poner aquí agradecimientos]

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción y motivación

La transparencia en los organismos públicos es importante.
Los desarrollos de software con despliegue continuo son interesantes.
(Pendiente)

# Capítulo 2

# Objetivos

El objetivo de este proyecto es el de obtener un portal de transparencia para la Universidad de Granada basado en el software libre que además permita una metodología de desarrollo en el que continuamente se puedan añadir nuevas funcionalidades a la vez que se van testeando, para después de una fácil integración culminar con un despliegue automático. Este portal no almacenará los propios datos abiertos, si no que hará de presentación de los datos que estarán contenidos en otra plataforma destinada únicamente a dicho fin (OpenData UGR).

Un resumen de los principales objetivos a alcanzar son:

- **OBJ-1.** Promover la idea de que los desarrollos bajo software libre en general y en la administración pública en particular ayudan a generar confianza y demuestran compromiso con la transparencia.
- **OBJ-2.** Solucionar los problemas de generación de las tablas con los elementos de información.
- **OBJ-3.** Implementar en la plataforma UGR Transparente una metodología de desarrollo continuo que se base en el uso de tests unitarios para cada una de las funcionalidades que se vayan añadiendo, test de cobertura que evalúen los test unitarios, integración continua ante cambios introducidos, despliegue automático de las actualizaciones que se vayan produciendo y aprovisionamiento software para la infraestructura.

Además como objetivo secundario tendremos:

- **OBJ-4.** Estudiar la posibilidad de hacer una instalación totalmente personalizada del portal de datos de código abierto CKAN para los datos abiertos de la Universidad de Granada, además de un aprovisionamiento que permitiera que esta instalación se pudiera realizar automáticamente en una infraestructura.

Destacar en los aspectos formativos previos más utilizados para el desarrollo del proyectolos conocimientos sobre infraestructuras virtuales para el tema de gestión de configuraciones, ingeniería de software para el análisis del proyecto e ingeniería de servidores para la realización de pruebas desde el aspecto hardware.

## 2.1. Alcance de los objetivos

La aplicación resultante visualmente será idéntica que la versión actual de UGR Transparente, pero internamente habrá cambiado totalmente la metodología de desarrollo, pasando de un desarrollo sin control a uno totalmente controlado mediante pruebas unitarias e integración continua. Además los cambios ya no se tendrá que aplicar manualmente en el servidor, sino que se hará uso de un despliegue automático para tal fin.

Esta aplicación se usará de base en la Universidad de Granada, pero además el objetivo del desarrollo es que se pueda exportar para su uso en cualquier organización sin demasiada dificultad, de ahí que se vaya a estudiar también la posibilidad de personalización de CKAN, ya que esto permitiría generar una gran plataforma de liberación de datos totalmente personalizable según el ámbito y las necesidades.

## 2.2. Interdependencia de los objetivos

Todos los objetivos son independientes entre sí, pero el primer objetivo (**OBJ-1**) es el principal motivador de este proyecto, por lo que aún sin representar el desarrollo de ningún trabajo en concreto es el que va a escudar y avalar el desarrollo de los otros. En aspectos más relacionados con la realización del proyecto, el segundo objetivo (**OBJ-2**) es el que se solucionará de forma inmediata ya que impide el correcto funcionamiento del portal. El resto de objetivos serán tratados en mayor medida durante los capítulos de implementación y pruebas.

# Capítulo 3

# Planificación

## 3.1. Fases y entregas

### 3.1.1. Fases

Como este va a ser un proyecto que ya cuenta con un trabajo previo realizado, la fase inicial de gestión va a ser muy breve porque se parte de que ya se han realizado varias reuniones y el proyecto está parcialmente funcionando, por lo que no se parte de cero, es una ampliación de un proyecto inicial.

- **Fase 1:** Especificaciones del proyecto
- **Fase 2:** Planificación
- **Fase 3:** Análisis y diseño
- **Fase 4:** Implementación
- **Fase 5:** Pruebas
- **Fase 6:** Documentación

### 3.1.2. Lista de entregas

Se harán una serie de breves informes sobre el contenido de cada una de las fases de planificación del proyecto.

- **Fase 1:** Especificación del proyecto.
    - Descripción: Se establecen los objetivos a cumplir para que el desarrollo del proyecto se considere completado.
    - Tipo: informe.

- **Fase 2:** Planificación.
  - Descripción: Se desarrolla la documentación con toda la planificación del desarrollo del proyecto.
  - Tipo: informe.
- **Fase 3:** Análisis y diseño.
  - Descripción: Todos los aspectos del proyectos son analizados para concretar la forma de desarrollarlo.
  - Tipo: informe.
- **Fase 4:** Implementación.
  - Descripción: Con el proyecto ya planificado y diseña se pasa programar todo lo necesario para cumplir los objetivos.
  - Tipo: software.
- **Fase 5:** Pruebas.
  - Descripción: Una vez este todo programado, se pasa a validar con diferentes procedimientos que el proyecto funciona correctamente tomando como referentes unas métricas propias.
  - Tipo: informe y software.
- **Fase 6:** Documentación.
  - Descripción: Para finalizar el proyecto se realiza toda la documentación informativa y explicativa.
  - Tipo: informe.

## 3.2. Estructura de Descomposición del Trabajo

El diagrama de Estructura de Descomposición del Trabajo (figura 3.1) es una descomposición jerárquica de las diferentes fases y entregas en las que está planificado el proyecto.

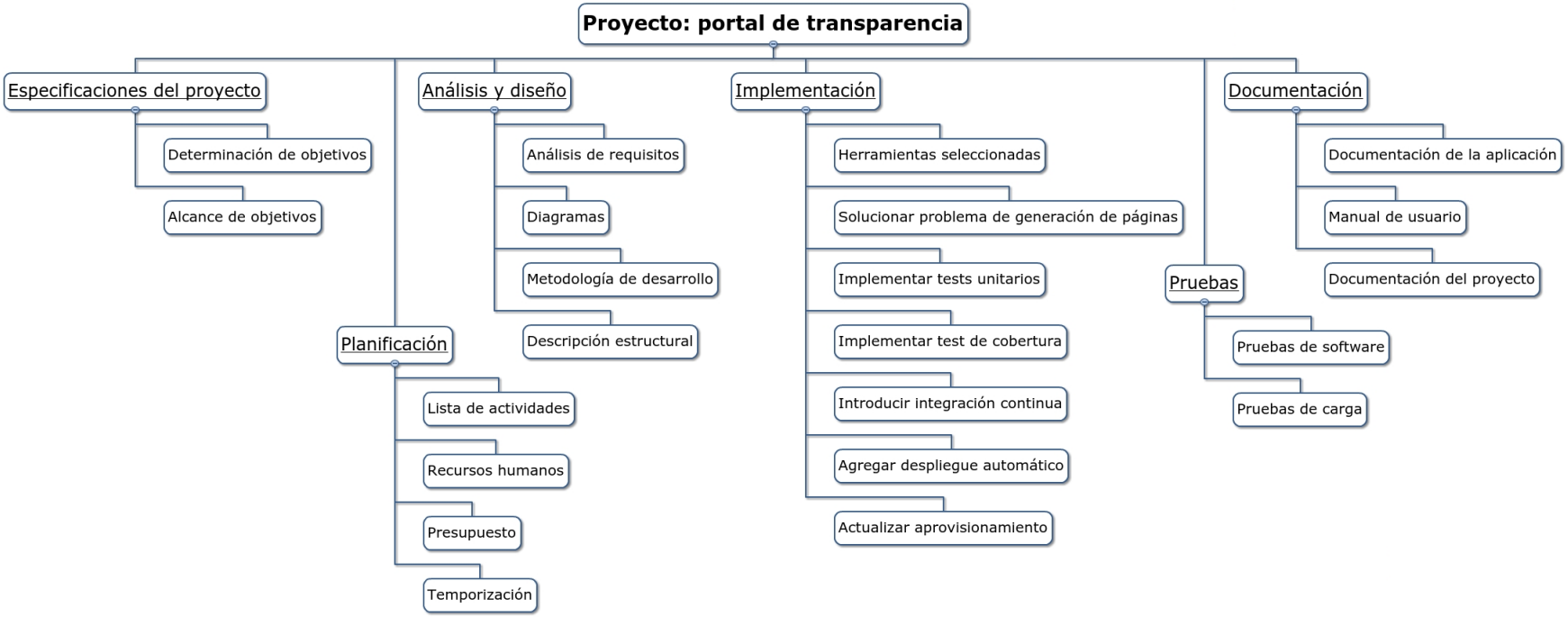


Figura 3.1: Diagrama de Estructura de Descomposición de Trabajo

## 3.3. Lista de actividades

Las actividades que se vayan a desarrollar en cada una de las fases para cada una de las entregas se va a listar junto con una estimación del tiempo que deberían tomar en ser cumplidas.

- **Especificaciones del proyecto:**
  - Determinación de objetivos.
  - Determinación de requisitos.
  - *Estimación: 6 horas*
- **Planificación:**
  - Lista de actividades.
  - Recursos humanos.
  - Presupuesto.
  - Temporización.
  - *Estimación: 10 horas*
- **Análisis y diseño:**
  - Análisis de requisitos.
  - Diagramas.
  - Metodología de desarrollo.
  - Descripción estructural.
  - *Estimación: 30 horas*
- **Implementación:**
  - Herramientas seleccionadas.
  - Solucionar problema de generación de páginas.
  - Implementar tests unitarios.
  - Implementar test de cobertura.
  - Introducir integración continua.
  - Agregar despliegue automático.
  - Actualizar aprovisionamiento.
  - *Estimación: 200 horas*

- **Pruebas:**
  - Pruebas de software.
  - Pruebas de carga.
  - *Estimación: 20 horas*
- **Documentación:**
  - Documentación de la aplicación.
  - Manual de usuario.
  - Documentación del proyecto.
  - *Estimación: 30 horas*

## 3.4. Recursos humanos

Como este es un proyecto que comenzó su desarrollo en la Oficina de Software Libre de la Universidad de Granada, cuento con el apoyo de todos sus colaboradores, además del resto de becarios que se encuentran realizando prácticas en empresa en ella como es mi situación actual, además, siendo el tutor de este proyecto el director de la propia oficina.

## 3.5. Presupuesto

Una de las ventajas de usar software libre es que no es necesario adquirir licencias por las que haya que pagar para realizar su uso, como todas las herramientas que usen (al igual que las que se generen) serán software libre, el coste en software para el desarrollo será cero.

El único recurso necesario es el servidor en que estará instalada la plataforma, servidor ya que adquirido anteriormente, por lo que tampoco es necesario considerarlo un gasto a afrontar de cara al desarrollo.

## 3.6. Temporización

[Pendiente]

# Capítulo 4

# Análisis y diseño

## 4.1. Análisis de requisitos

Todo software se desarrolla para cubrir una necesidad, por lo que en este apartado vamos a describir los requisitos que se estiman necesarios para cubrir los objetivos propuestos.

### 4.1.1. Descripción de los actores

Los actores implicados serán dos: el **desarrollador** y el **usuario**.

El **desarrollador** será el encargado de solucionar los problemas de visualización de los datos en la plataforma, además de portar el desarrollo actual de la plataforma a un entorno de desarrollo continuo. También asumirá la administración de la plataforma ya que este tipo de rol está muy integrado con las labores de despliegue en el desarrollo continuo.

El **usuario** de la aplicación será cualquier persona que tenga interés por conocer datos internos de la Universidad de Granada fácilmente. El usuario no pertenece a ningún público objetivo concreto, por lo que no se tiene que considerar que tenga una experiencia previa en navegador por sitios web.

### 4.1.2. Requisitos Funcionales

Los requisitos funcionales son las características que tiene que implementar el sistema para cubrir todas las necesidades de los distintos usuarios.

Al usuario lo único que le interesa es ver una página web estática con la información que desea consultar, para ello el desarrollador deberá hacer que sea posible que se generen siempre las tablas con los elementos de información.

Por otra parte, el desarrollador quiere integrar el sistema en un desarrollo continuo, por lo que añadirá tests unitarios, test de cobertura, integración continua, despliegue automático y aprovisionamiento con tal fin.

- **RF-1.** Administración portal:
  - **RF-1.1.** Arrancar el portal mediante un script.
  - **RF-1.2.** Detener el portal mediante un script.
- **RF-2.** Acceso información:
  - **RF-2.1.** Consultar tablas con elementos de información.
- **RF-3.** Desarrollo continuo:
  - **RF-3.1.** Realizar tests unitarios.
  - **RF-3.2.** Realizar test de cobertura.
  - **RF-3.2.** Usar integración continua.
  - **RF-3.3.** Usar despliegue automático.
  - **RF-3.4.** Usar aprovisionamiento.

### 4.1.3. Requisitos no Funcionales

Los requisitos no funcionales son las características propias del desarrollo, pero que no tienen que estar relacionadas con su funcionalidad.

- **RN-1.** Toda la programación del portal se hará en Node.js y los módulos que se usen deben ser instalables a través de su gestor de paquetes NPM.
- **RN-2.** El portal se iniciará y se detendrá mediante scripts lanzados con NPM.
- **RN-3.** Para iniciar la ejecución del portal es necesario que reciba el puerto de escucha del servidor y la dirección de acceso.
- **RN-4.** Todos los módulos se ejecutarán desde scripts lanzamos con NPM.
- **RN-5.** Los tests unitarios se realizarán en base a comportamientos esperados y valores de estados recibidos como contestación a las peticiones que se realicen al portal.
- **RN-6.** Los tests unitarios tienen que recibir las páginas del portal para ejecutarse.

- **RN-7.** El test de cobertura tiene que tener una automatización integrable con los tests unitarios.
- **RN-8.** La integración continua se ejecutará automáticamente con cada cambio que se haga en la programación del portal.
- **RN-9.** El despliegue automático se realizará mediante conexiones SSH.
- **RN-10.** Tanto para el despliegue automático como para el aprovisionamiento es necesario indicar el usuario que lo realiza y el destino en el que se realiza.

### 4.1.4. Requisitos de Información

Los requisitos de información se refieren a la información que es necesaria almacenar en el sistema. La única información relevante que se va a almacenar son los datos descriptivos y de enlace de cada uno de los elementos del portal OpenData UGR que se van a mostrar en UGR Transparente.

- **RI-1.** Datos abiertos.
  - Información sobre cada uno de los elementos que se van a mostrar en el portal de transparencia como datos abiertos.
  - Contenido: nombre, categoría, conjunto de datos, enlace a OpenData UGR, enlace al recurso.

## 4.2. Modelos de casos de uso

Aunque ya se ha indicado que la parte funcional ya se encuentra implementada de forma previa a este proyecto, se van a incluir unos modelos de caso de uso simples para dar un visión más clara del funcionamiento general de la plataforma UGR Transparente.

### 4.2.1. Descripción básica de actores

- **Ac-1.** Desarrollador
  - Descripción: Encargado del desarrollo y administración de la plataforma.
  - Características: Su trabajo está en el lado del servidor que genera la página, nunca trabaja desde el lado del cliente.
  - Relaciones: Ninguna.
  - Atributos: Ninguno.
  - Comentarios: Es el encargado de que la información se muestre en el portal y de añadir funcionalidades a la plataforma.

- **Ac-2.** Usuario
  - Descripción: Persona que usa la plataforma que consulta datos.
  - Características: Es el usuario común que accederá a la página.
  - Relaciones: Ninguna.
  - Atributos: Ninguno.
  - Comentarios: El usuario no es necesario que tenga ningún conocimiento previo al uso de la plataforma, simplemente accederá y consultará los datos que sean de su interes.

### 4.2.2. Descripción casos de uso

- **CU-1.** Iniciar plataforma
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias:
  - Precondición: El servidor esté detenido.
  - Postcondición: El portal será accesible públicamente.
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Arrancar el servidor del portal UGR Transparente.
  - Resumen: Cuando se ejecuta el script de inicio de la aplicación, arranca el servidor y el portal será accesible desde internet.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: da orden de que se inicie la plataforma. | | |
| | | *2a* | Comprueba que el servidor está detenido e inicia la plataforma. |

Tabla 4.1: Curso normal de CU-1. Iniciar plataforma

| **Curso alterno** | |
|---|---|
| *2b* | Si el servidor está funcionando, no se ejecuta el script de arranque del servidor. |

Tabla 4.2: Curso alterno de CU-1. Iniciar plataforma

- **CU-2.** Detener plataforma
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias:
  - Precondición: El servidor esté funcionando.
  - Postcondición: El portal deja de estar operativo.
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Detener el servidor del portal UGR Transparente.
  - Resumen: Cuando se ejecuta el script de detención de la aplicación, se detiene el servidor y el portal deja de estar accesible desde internet.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: da orden de que se detenga la plataforma. | | |
| | | *2a* | Comprueba que el servidor está iniciado y detiene la plataforma. |

Tabla 4.3: Curso normal de CU-2. Detener plataforma

| **Curso alterno** | |
|---|---|
| *2b* | Si el servidor no está funcionando, no se ejecuta el script de parada del servidor. |

Tabla 4.4: Curso alterno de CU-2. Detener plataforma

- **CU-3.** Consultar datos
  - Actores: Usuario
  - Tipo: Primario, Esencial
  - Referencias:
  - Precondición: Existan archivos con los datos abiertos.
  - Postcondición:
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0

- Propósito: El usuario consulta los datos mostrador en el portal UGR Transparente.
- Resumen: El usuario que accede al portal de transparencia, navega a través de las distintas categorías y selecciona aquella en la que está interesado en conocer.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Usuario: consulta información de una subcategoría del portal. | | |
| | | *2* | Se generan las tablas con todos los elementos de información de la subcategoría correspondiente. |
| | | *3* | Se muestran en la página todos las tablas generadas con los elementos de información. |

Tabla 4.5: Curso normal de CU-3. Consultar datos

- **CU-4.** Realizar tests unitarios
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias:
  - Precondición: Existan tests unitarios.
  - Postcondición:
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Realizar tests unitarios para comprobar las funcionalidades de la aplicación.
  - Resumen: Cada vez que se añadan nuevas páginas o funcionalidades al portal, se comprueba que funcionan correctamente.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: da orden de ejecutar los tests unitarios. | | |
| | | *2a* | Comprueba que hay tests unitarios y los ejecuta. |

Tabla 4.6: Curso normal de CU-4. Realizar tests unitarios

| **Curso alterno** | |
|---|---|
| *2b* | Si no hay tests unitarios creados, el sistema no hace nada. |

Tabla 4.7: Curso alterno de CU-4. Realizar tests unitarios

- **CU-5.** Realizar tests de cobertura
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias: (CU-4.) Realizar tests unitarios
  - Precondición: Existan tests unitarios.
  - Postcondición:
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Realizar test de cobertura para comprobar calidad de los tests unitarios.
  - Resumen: Para comprobar si los tests unitarios cumplen correctamente con su función se analiza el porcentaje del código que está cubierto por los mismos.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: da orden de ejecutar test de cobertura. | | |
| | | *2a* | Comprueba que hay tests unitarios y los ejecuta. |
| | | *3* | Se ejecuta el test de cobertura en base a los tests unitarios ejecutados. |

Tabla 4.8: Curso normal de CU-5. Realizar test de cobertura

| **Curso alterno** | |
|---|---|
| *2b* | Si no hay tests unitarios creados, el sistema no hace nada. |

Tabla 4.9: Curso alterno de CU-5. Realizar test de cobertura

- **CU-6.** Usar integración continua
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias: (CU-4.) Realizar tests unitarios
  - Precondición: Existan test unitarios.
  - Postcondición: Se genera un informe con el resultado de los tests unitarios.
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Comprobar si los cambios en la aplicación provocan errores.
  - Resumen: Cuando se efectúan cambios en la aplicación automáticamente se ejecutan los tests unitarios para comprobar si los cambios introcudidos provocan conflictos en la plataforma.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: guardar cambios en el repositorio. | | |
| | | *2a* | Comprueba que hay tests unitarios y comienza la verificación de la integración continua. |
| | | *3* | Ejecuta los tests unitarios para cada entorno definido. |

Tabla 4.10: Curso normal de CU-6. Usar integración continua

| **Curso alterno** | |
|---|---|
| *2b* | Si no hay tests unitarios creados, el sistema no hace nada. |

Tabla 4.11: Curso alterno de CU-6. Usar integración continua

- **CU-.7** Usar despliegue automático
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias: (CU-1.) Iniciar plataforma
  - Precondición:
  - Postcondición: Los cambios introducidos son aplicados en la plataforma.
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Aplicar automáticamente los cambios realizados a la aplicación.
  - Resumen: Para no tener que acceder manualmente al servidor de la plataforma y tener que desplegar los cambios introducidos, desde el entorno de desarrollo desplegamos automáticamente los cambios en el servidor.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: ordenar despliegue automático. | | |
| | | *2* | Conectar al servidor. |
| | | *3* | Crear copia de seguridad. |
| | | *4a* | Comprobar si hay cambios en el repositorio de la plataforma y descargarlos en dicho caso. |
| | | *5* | Iniciar la plataforma con los cambios aplicados. |

Tabla 4.12: Curso normal de CU-7. Usar despliegue automático

| **Curso alterno** | |
|---|---|
| *4b* | Si no hay cambios aplicables, se continua con el proceso. |

Tabla 4.13: Curso alterno de CU-7. Usar despliegue automático

- **CU-.8** Usar aprovisionamiento
  - Actores: Desarrollador
  - Tipo: Primario, Esencial
  - Referencias: (CU-1.) Iniciar plataforma
  - Precondición:
  - Postcondición: El portal queda instalado en la infraestructura seleccionada.
  - Autor: Germán Martínez Maldonado
  - Versión: 1.0
  - Propósito: Instalar automáticamente el portal en una infraestructura dada.
  - Resumen: Se instalarán automáticamente todos los elementos necesarios para poner en funcionamiento el portal en cualquier infraestructura que se indique.

| **Curso normal** | | | |
|---|---|---|---|
| **Actor** | | **Sistema** | |
| *1* | Desarrollador: ordenar despliegue automático. | | |
| | | *2* | Conectar al servidor. |
| | | *3* | Comprobar aplicación necesaria. |
| | | *4a* | Se comprueba si la aplicación necesaria está instalada, instalándola en caso de que no lo esté. |
| | | *5* | Descarga la plataforma desde el repositorio. |
| | | *6* | Instalar todas las dependencias de la plataforma. |
| | | *7* | Se establecen los parámetros de acceso al portal desde internet. |
| | | *8* | Con todo preparado, se inicia la plataforma. |

Tabla 4.14: Curso normal de CU-8. Usar aprovisionamiento

| **Curso alterno** | |
|---|---|
| *4b* | Si la aplicación está instalada, se continua con el proceso. |

Tabla 4.15: Curso alterno de CU-8. Usar aprovisionamiento

## 4.3. Diagrama de paquetes

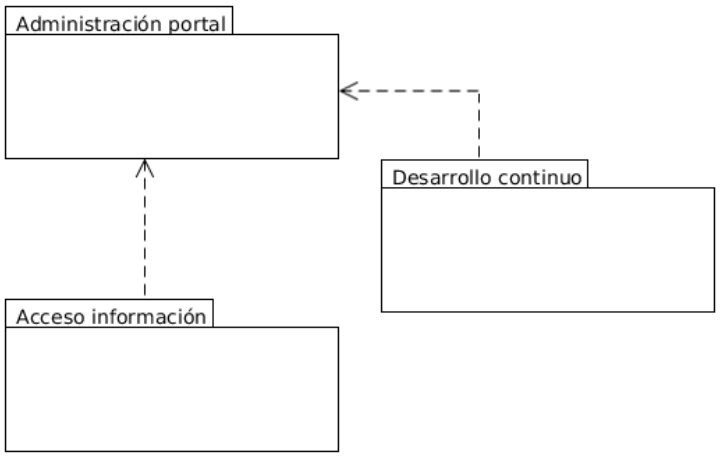


Figura 4.1: Diagrama de paquetes

## 4.4. Diagramas de casos de uso

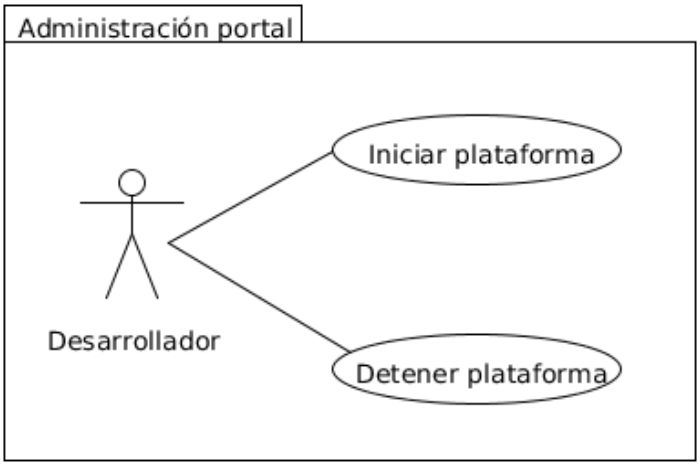


Figura 4.2: Diagrama de casos de uso: paquete Administración portal

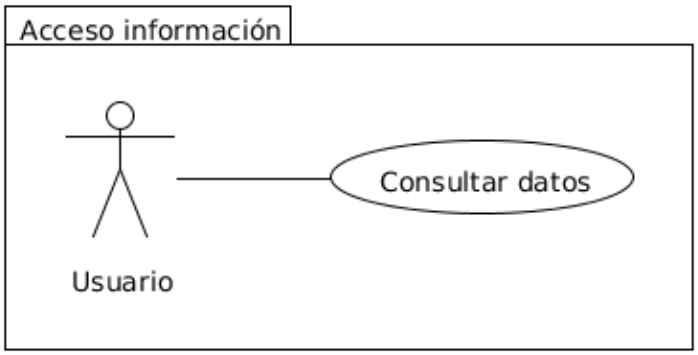


Figura 4.3: Diagrama de casos de uso: paquete Acceso información

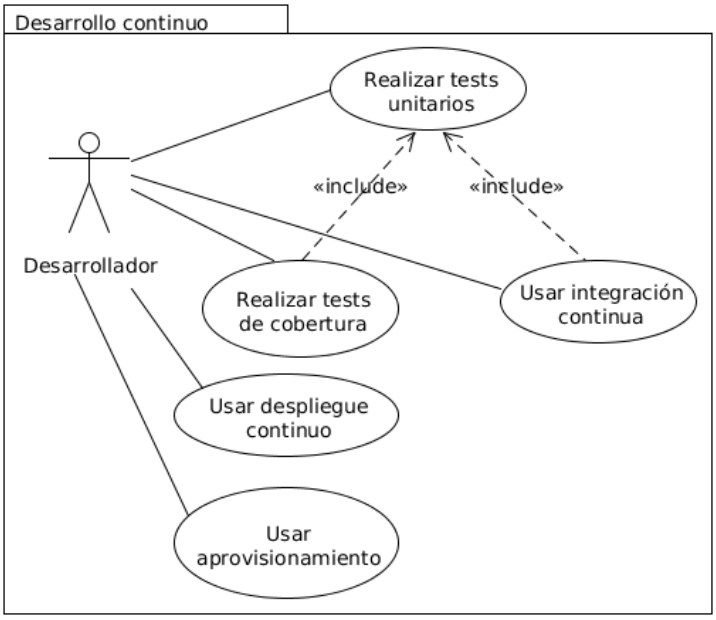


Figura 4.4: Diagrama de casos de uso: paquete Desarrollo continuo

## 4.5. Diagramas de actividad

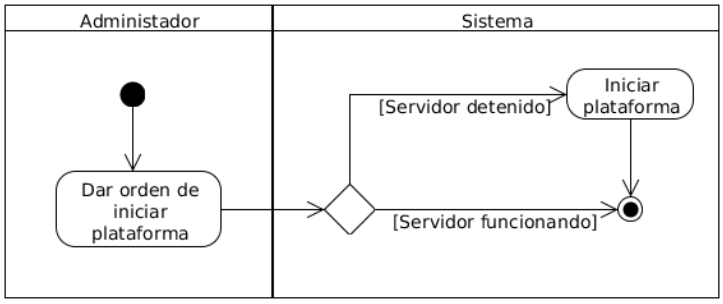


Figura 4.5: Diagrama de actividad CU-1. Iniciar plataforma

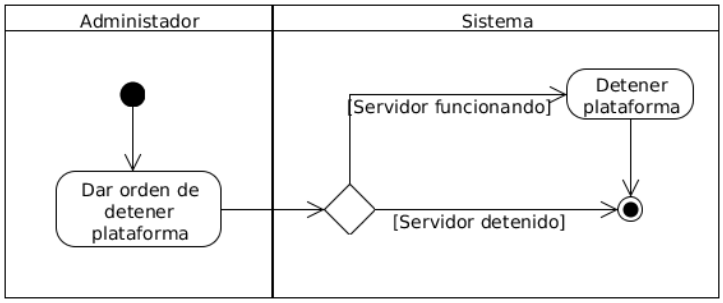


Figura 4.6: Diagrama de actividad CU-2. Detener plataforma

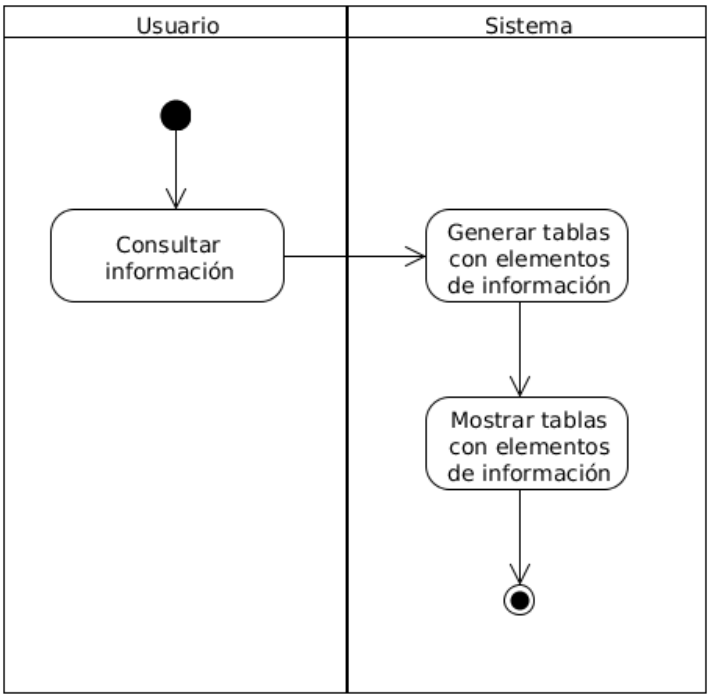


Figura 4.7: Diagrama de actividad CU-3. Consultar datos

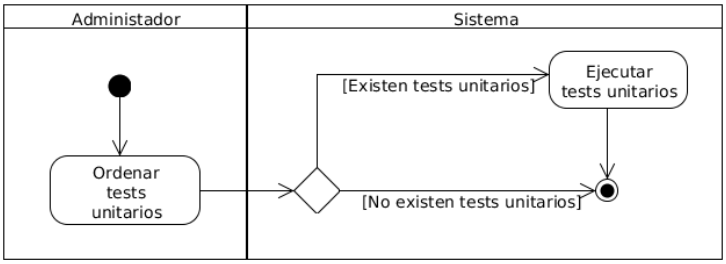


Figura 4.8: Diagrama de actividad CU-4. Realizar tests unitarios

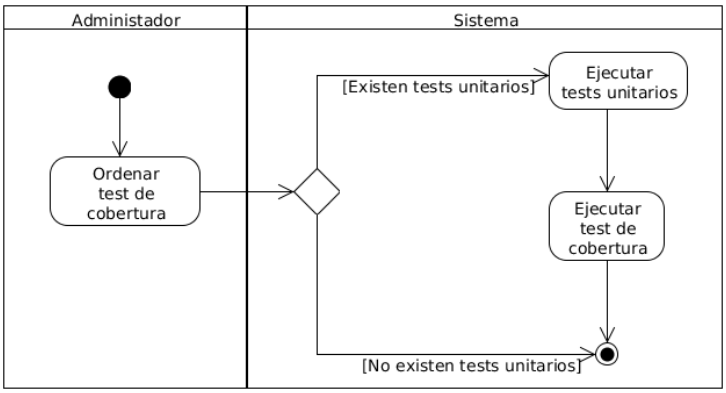


Figura 4.9: Diagrama de actividad CU-5. Realizar tests de cobertura

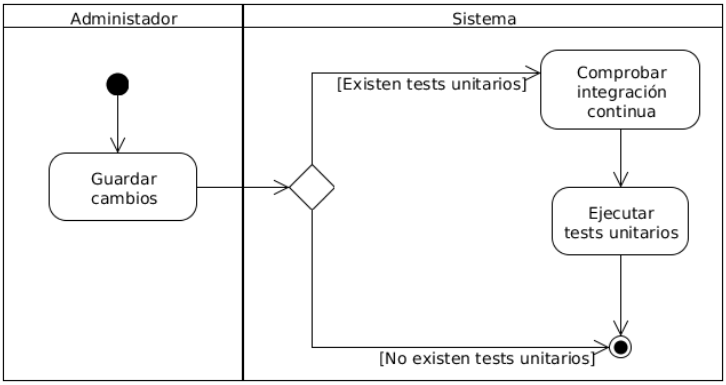


Figura 4.10: Diagrama de actividad CU-6. Usar integración continua

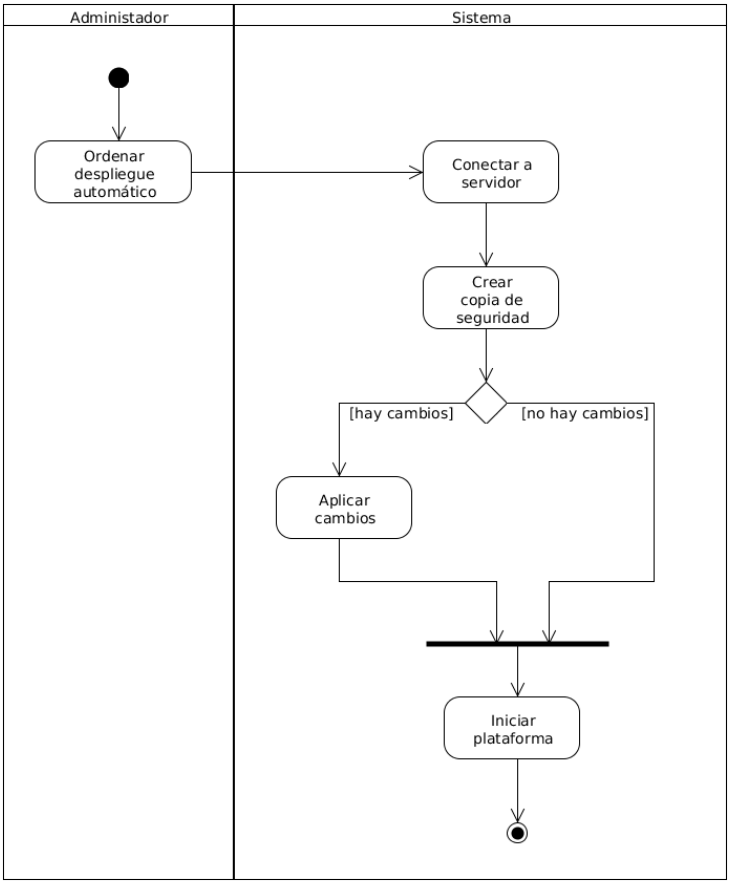


Figura 4.11: Diagrama de actividad CU-7. Usar despliegue continuo

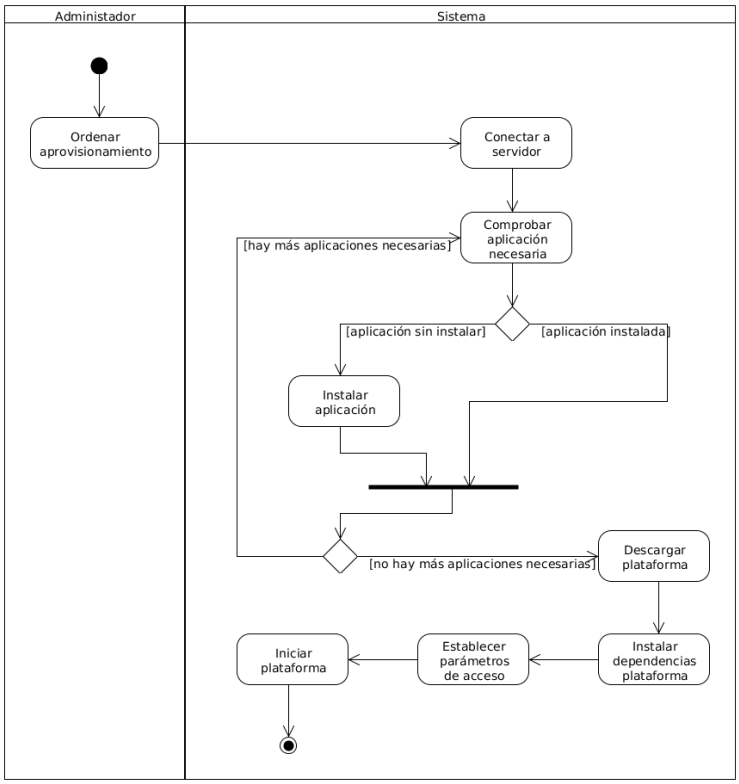


Figura 4.12: Diagrama de actividad CU-8. Usar aprovisionamiento

## 4.6. Diagramas de secuencia

[Pendiente]

## 4.7. Diseño de los procedimientos

[Provisional]

- Sin clases
- Scripts requeridos y exportados
- Todo lo programado fuera de la aplicación principal son procedimientos aislados.