“Instalador para soluciones desarrolladas empleando el Marco de Trabajo Symfony2.”

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Angel Luis Rodríguez Molina

Tutor: Ing.René R. Bauta Camejo

La Habana, 2014

Resumen

La Universidad de las Ciencias Informáticas se encuentra inmersa en un proceso de reestructuración de su estructura productiva. Como parte de este proceso se ha definido Symfony2 como arquitectura de referencia para las aplicaciones desarrolladas sobre PHP y como estrategia a mediano plazo se planea fusionar Symfony2 con las potencialidades de Sauxe. Por esta razón gran cantidad de productos utilizarán Symfony2 como base, por lo que resulta de gran importancia la instalación y configuración de este marco de trabajo para su uso en el despliegue de aplicaciones en los proyectos productivos de la universidad. Durante esta debe configurarse una serie de tecnologías que utiliza para su funcionamiento y se deben realizar muchas tareas complejas en las que se pierde valioso tiempo e interrumpen el flujo de trabajo que acontece hoy en los proyectos productivos.

Debido a esto se propone crear un instalador para soluciones desarrolladas empleando el marco de trabajo Symfony2 como solución a dicha situación. Este permitirá facilitar el trabajo de instalación de las tecnologías necesarias para el despliegue de una aplicación sobre Symfony2 agilizando y mejorando este proceso.

En este documento está fundamentada la necesidad de la creación de un instalador de este tipo, se describen las herramientas utilizadas, el diseño de la solución, las características del producto desarrollado y las pruebas a las que se vio sometido.

Palabras clave

Instalador web, marco de trabajo, Symfony2, configuración.

Declaración de Autoría

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Angel Luis Rodríguez Molina

_____

Ing.Rene Rodrigo Bauta Camejo

Le agradezco a la revolución por haberme brindado la posibilidad de estudiar y superarme.

A mi toda mi familia, en especial a mi madre y mi padre por estar ahí siempre.

A todas las personas que me ayudaron y me apoyaron durante toda la carrera.

A todo el que me brindó su ayuda para la realización de este trabajo y todo el que me cayó arriba para que lo terminara.

A todos mis amigos por estar ahí y molestarme cada vez que tuvieran un chance.

A mis padres por siempre estar ahí para mí y ser esa guía que toda persona necesita.

Índice

Índice

Índice de Figuras

Índice de Tablas

Índice

Introducción

La Universidad de las Ciencias Informáticas (UCI) se destaca por ser una entidad que posee centros productivos en los cuales se desarrollan productos que impactan en áreas como la gestión de procesos empresariales, aduanales, financieros, fiscales entre otros. Bajo el principio de alcanzar la soberanía tecnológica muchos de estos centros emplean tecnologías libres para el desarrollo de sus productos. Entre los lenguajes más utilizados se encuentra PHP (Hipertext Preprocesor) el cual se emplea en la creación de aplicaciones web. Para el desarrollo de aplicaciones web existen varios marcos de trabajo (MT). Un marco de trabajo es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Estos permiten agilizar y dar mayor robustez al proceso de desarrollo. Entre los MT más usados en la actualidad para el desarrollo de aplicaciones web empleando el lenguaje de PHP se encuentra Symfony2, siendo este el MT más empleado en la UCI.

Symfony2 brinda facilidades a la hora de crear una aplicación web compleja, ya que automatiza las tareas más comunes del proceso de implementación permitiendo al desarrollador centrarse en los aspectos específicos de cada aplicación. Su flexibilidad permite satisfacer las necesidades de los profesionales y usuarios avanzados por igual. Posee herramientas diseñadas para que la vida de un desarrollador sea mucho más fácil, entre ellas la barra de herramientas de depuración web, así como soporte nativo para entornos de desarrollo, páginas de error detallados o incluso de seguridad nativa. Además es un MT que posee abundante documentación.

La UCI está inmersa en un proceso de reestructuración de su estructura productiva. Como parte de este proceso se están definiendo arquitecturas de referencias para las distintas tecnologías en las que se desarrolla. La Dirección General de Producción (DGP) definió usar como arquitecturas de referencias para PHP Symfony y Sauxe, teniendo como objetivo final fusionar estas dos y crear una en la que la base tecnológica sea Symfony y que incluya las potencialidades de Sauxe. Teniendo en cuenta estos elementos, a mediano plazo, los desarrollos que se hagan empleando PHP utilizarán como base a Symfony.

Teniendo en cuenta lo anteriormente planteado, a mediano plazo, el número de productos que tengan como base tecnológica Symfony aumentará por lo que es necesario prestar atención a las configuraciones mínimas necesarias a realizar en los servidores donde estos se vayan a

desplegar. Para el despliegue de un producto desarrollado sobre Symfony es necesaria su instalación, durante la cual se deben configurar una serie de tecnologías que utiliza para su funcionamiento. Se debe realizar cambios en importantes archivos del sistema por lo cual cometer un error podría repercutir en que no se pueda utilizar el marco de trabajo o incidir directamente en la eficiencia o estabilidad del sistema operativo.

El MT requiere de la instalación de PHP con extensiones para optimizar su uso como por ejemplo el acelerador PHP APC y php-intl. Se debe instalar Apache2 con algunas extensiones necesarias para el buen funcionamiento de Symfony2, como libapache2-mod-proxy-html con la cual se compila el módulo mod_rewrite, el cual permite la reescritura de URL. PostgreSQL también debe estar instalado y configurado por lo que deben hacerse cambios en varios de sus archivos de configuración. Para mejorar el rendimiento y visualizar las aplicaciones con más seguridad se debe configurar un VirtualHost, ya que así se le puede brindar servicio de alojamiento web con una disminución en el costo de tales servicios y por lo tanto su accesibilidad por mayor número de usuarios.

Además se requiere realizar cambios en el archivo php.ini del servidor y el de la línea de comandos ya que Symfony2 utiliza a veces distintos archivos para la web y la línea de órdenes de la terminal. También se deben otorgar los permisos correctos al usuario que utiliza Apache para poder acceder a los directorios donde se guarda la caché y los logs de Symfony2.

Todo este proceso debe hacerse de forma manual para instalar Symfony2 y se puede tornar engorroso además de que requiere conocimientos y tiempo por parte del equipo de despliegue.

A partir de la problemática antes descrita se identificó como **problema a resolver**:

¿Cómo mejorar el proceso de instalación de las aplicaciones desarrolladas empleando el Marco de Trabajo Symfony2 para facilitar el proceso de despliegue?

La investigación se enmarca en el **objeto de estudio**:

Proceso de instalación y configuración en aplicaciones web.

Tomando como **campo de acción**:

Proceso de instalación de las aplicaciones desarrolladas sobre el Marco de Trabajo Symfony2.

Objetivo general: Desarrollar un componente para el Marco Trabajo Symfony2 que mejore el proceso de instalación y configuración de las aplicaciones para facilitar el proceso de despliegue.

Objetivos específicos

- Elaborar el marco teórico de la investigación relacionado con el proceso de instalación de las aplicaciones para identificar debilidades y aspectos reutilizables.
- Realizar el análisis y diseño de la solución para facilitar la implementación del componente.
- Implementar el instalador para soluciones desarrolladas empleando el Marco de Trabajo Symfony2 para mejorar su proceso de despliegue.
- Validar el instalador para soluciones desarrolladas empleando el Marco de Trabajo Symfony2 mediante pruebas de software para asegurar la calidad del producto.
- Realizar un pre-experimento para validar el cumplimiento del objetivo planteado.

Idea a Defender: Si se desarrolla un componente para el Marco Trabajo Symfony2 que mejore al proceso de instalación de las aplicaciones desarrolladas empleando dicho Marco de Trabajo se facilitará el proceso de despliegue.

Métodos Científicos:

- Métodos Teóricos:

☐ Método sistémico

Consiste en estudiar el objeto mediante la determinación de sus componentes o como componente del objeto a que pertenece, así como la relación entre ellos que conforma una realidad como totalidad (1).

Se utilizó el método sistémico en el estudio de los tipos de instaladores que se pueden utilizar para dar solución al problema planteado. Al estudiarlos por separado se analizaron las características propias de cada uno identificando sus fortalezas y propiedades que podrían ser utilizadas para conformar la solución.

☐ Método hipotético-deductivo.

Desempeña un papel esencial en el proceso de verificación de la hipótesis, tiene un gran valor heurístico, pues permite adelantar y verificar nuevas hipótesis de la realidad, permite inferir conclusiones y establecer predicciones a partir del sistema de conocimientos que ya se posee (1).

Se utilizó este método para identificar las características esenciales de los instaladores web estudiados y que podrían ser incorporadas al instalador para el MT Symfony2 y de esta manera

predecir muchas de las situaciones que se pueden presentar durante el diseño e implementación de la solución atendiendo a las características de la aplicación en cuestión.

- Métodos Empíricos:

☐ Método de observación:

La observación científica es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado (1).

Mediante este método se pudo observar la complejidad que puede adquirir la configuración de las tecnologías que requiere Symfony2 para su funcionamiento en los proyectos productivos de la UCI.

Estructura del trabajo:

El presente documento posee la siguiente estructura: Resumen, Índice, Introducción, tres Capítulos, Conclusiones Generales, Bibliografía y Anexos.

Capítulo 1: En este capítulo se tratan los elementos referentes al estado del arte, tales como la investigación sobre los instaladores existentes en la actualidad y la comparación entre ellos, la metodología de desarrollo a usar, el análisis de las tecnologías y herramientas a utilizar.

Capítulo 2: Se presenta el modelo conceptual que describe la solución. Las técnicas utilizadas para la captura de requisitos, los requisitos funcionales capturados y su respectiva validación. Además se describe la solución propuesta con todos los diagramas propuestos por la metodología de desarrollo elegida para la mejor comprensión de la problemática presentada. En este capítulo se utilizan métricas para medir la calidad del diseño del software.

Capítulo 3: En este capítulo se presenta el diagrama de componentes, los estándares de codificación utilizados así como características de la solución obtenida. Contiene además las pruebas realizadas al producto incluyendo las pruebas de caja blanca y de caja negra utilizadas para validar la calidad del producto final.

Capítulo 1: Fundamentación teórica.

1.1 Introducción

En este capítulo se realiza una investigación sobre los tipos de instaladores que existen para elegir las características que se pueden utilizar de cada uno. Además se realiza un estudio sobre los instaladores web que existen en la actualidad analizando las características particulares de cada uno y valorando los aspectos reutilizables que tengan. Se describen las tecnologías y herramientas que se usarán durante el análisis e implementación, así como la metodología de software a utilizar. Culminado el estado del arte se brinda una propuesta de solución al problema planteado.

1.2 Estudio de los tipos de instaladores existentes

Se realizó una investigación sobre los distintos tipos de instaladores que existen analizando las características de cada uno con el objetivo de seleccionar la vía más factible para darle solución al problema presentado.

1.2.1 Instalador exe

Los ficheros ".exe", que no son más que ficheros ejecutables que realizan determinadas operaciones cuando se les activa como copiar archivos, descargar documentos o instalar programas. Para comenzar la instalación se ejecuta el fichero y se siguen las instrucciones de la aplicación. Es importante saber que para lograr la instalación mediante este tipo de ficheros los ordenadores deben estar equipados con MS- DOS, Microsoft Windows, OS/2 y ReactOS (2).

1.2.2 Instalador Bash

Bash (Bourne again shell) es un programa informático cuya función consiste en interpretar órdenes. Está basado en la shell (intérprete de comandos) de Unix y es compatible con POSIX (Portable Operating System Interface). Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de GNU con Linux. Con un instalador bash se pueden utilizar todos los comandos propios de Linux, por lo que se le prestará especial atención debido a la política de migración al software libre asumida en los centros productivos de la UCI (3).

1.2.3 Instalador web

Para instalar desde la web es necesario copiar los archivos a una estación de trabajo y realizar las configuraciones pertinentes para que desde el navegador se pueda acceder a la página web de instalación. Eso conlleva a la configuración de Apache, la creación de un VirtualHost para acceder a la aplicación garantizando una mayor seguridad y eficiencia, y la instalación de las extensiones necesarias de PHP que permitan la visualización de las páginas web. Desde un instalador web se puede configurar la conexión a un gestor de base de datos por lo cual se vuelve una parte indispensable al tener que realizar operaciones con bases de datos.

1.3 Estudio de los instaladores web existentes

Los instaladores web son poco comunes, por lo cual se desarrolló una investigación sobre los instaladores web existentes en la actualidad que puedan proporcionar una guía para la creación del instalador para soluciones desarrolladas empleando el MT Symfony2. Se tuvieron en cuenta los siguientes aspectos para su evaluación: tecnologías que utilizan, la forma de presentar el contenido al usuario y la posibilidad de integración con Symfony2.

1.3.1 Instalador web para Mantis

Mantis es una herramienta de gestión de incidencias (*bugtrucker*) de código abierto, se utiliza para testear soluciones, hacer un registro histórico de alteraciones y gestionar equipos remotamente.

Su instalador está escrito en PHP y requiere una base de datos, la base de datos recomendada para la ejecución de Mantis es MySql y requiere de un servidor Apache. Además tiene software y requisitos de hardware modestos por lo que puede trabajar en máquinas sin grandes prestaciones. Se requiere un equipo que sea capaz de ejecutar el software de un servidor. La aplicación es accesible solamente por el navegador. Es multiplataforma, ya que es operable en Windows, Linux y sistemas operativos MacOS. Posee una interfaz gráfica bastante sencilla y entendible. Una buena práctica que realiza es chequear la versión de PHP que posee la máquina donde va a instalarse Mantis, evitando así que el sistema se instale sin las condiciones necesarias para su uso (4).

1.3.2 Instalador web para Joomla!

Se encarga de instalar Joomla desde la web. Los requerimientos mínimos para proceder a la instalación son: debe poseer un servidor web Apache 2.x o superior, servidor de bases de datos: MySQL 5.0.4 o superior, intérprete del lenguaje PHP 5.3 o superior y soporte XML. Se necesita tener operativa una aplicación cliente de FTP. El proceso de instalación se inicia a través de un navegador Joomla!, este puede ser usado con la mayoría de los navegadores: Internet Explorer, Mozilla Firefox, Safari, Netscape, Opera.

El instalador presenta una interfaz de usuario sencilla, que va informando de los pasos para la instalación. Como primer paso se selecciona del idioma en el que se va a instalar. El siguiente paso se encarga de revisar si el servidor cumple con los requisitos previos de instalación incluyendo los valores recomendados de la configuración de PHP; lo cual asegura que la instalación se realice correctamente. Luego se introducen los datos para la creación de la base de datos. Por último permite crear el primer usuario de Joomla! (5).

1.3.3 Instalador web para Cedrux

El instalador web para el Sistema Integral de Gestión Cedrux es un producto diseñado con el propósito de instalar Cedrux mediante la web permitiendo realizar las configuraciones necesarias durante el proceso de instalación. Está creado sobre tecnologías libres y está programado en su capa de negocio en PHP, utiliza un servidor web Apache, PostgreSQL como gestor de base de datos.

La interfaz fue creada usando la librería de ExtJs y es accesible desde el navegador. El producto ofrece funcionalidades de instalación avanzadas para seleccionar los subsistemas a instalar creando además la base de datos sobre la cual trabajará. Fue desarrollado con el modelo de desarrollo de software tecnológico propuesto por el Departamento de Tecnología del CEIGE (6).

1.3.4 Instalador web para el sistema GINA

El instalador para el Sistema de Gestión Integral de Aduanas (GINA) ofrece funcionalidades avanzadas para la instalación como la selección de subsistemas a instalar, siendo capaz de gestionar las dependencias entre ellos. Posee un módulo que permite la configuración del servidor donde se va a instalar el sistema GINA y permite empaquetar las herramientas necesarias para este proceso. Está desarrollado sobre Symfony1, utilizando para la creación de

las interfaces de usuario ExtJs y requiere de un servidor web Apache (7).

A continuación se muestra el resultado del estudio de los instaladores web existentes atendiendo a los indicadores seleccionados:

Tabla 1: Comparación de los instaladores web existentes.

Indicadores		**Mantis**	**Joomla!**	**Cedrux**	**GINA**
Tecnologías que utiliza	Lenguaje de programación	PHP	PHP	PHP	PHP
	Marco de Trabajo	-	-	ZendFramework, ExtJs	Symfony1, ExtJs
	Servidor web	Apache	Apache	Apache	Apache
	Gestor de base de datos	MySql	MySql	PostgreSQL	Oracle
	Otras		Cliente FTP		
Forma de presentar el contenido		regular	Muy buena	buena	buena
Integración con Symfony2		no	no	no	no

1.4 Valoración de los instaladores analizados

Después de haber realizado un estudio de los principales tipos de instaladores existentes y de los instaladores web elegidos atendiendo a sus características y a los indicadores previamente seleccionados se concluye lo siguiente:

Todos los instaladores utilizan como servidor de base de datos Apache y configuran la base de datos que van a utilizar durante la instalación, aunque sólo el instalador web para Cedrux utiliza el gestor de base de datos PostgreSQL. Sin embargo el instalador web para Cedrux usa ZendFramework como MT y está diseñado para brindar una solución específica por lo que no puede reutilizarse.

Por su parte el instalador para el Sistema de GINA está implementado utilizando Symfony1, pero Symfony2 posee demasiadas diferencias estructurales con su predecesor por lo que no podría migrarse o reutilizarse esta solución. Este instalador también permite la configuración de la estación de trabajo donde se va a instalar el sistema GINA, característica que aporta eficiencia y seguridad al instalador ya que esta queda lista para el uso del sistema. Se decide entonces crear

una herramienta con estas características como parte del instalador para soluciones desarrolladas empleando el MT Symfony2, esta realizaría la configuración de las tecnologías que requiere Symfony2. Con este objetivo se decide utilizar un instalador bash, mediante este se puede instalar y configurar las tecnologías necesarias utilizando órdenes propias de la línea de comandos de los sistemas Linux, mediante las cuáles se pueden realizar todas estas tareas.

Se midió el indicador presentación del contenido al usuario atendiendo a la forma en que el instalador presenta la información, clasificando en muy buena, buena, regular y mala. Todos los instaladores poseen en general una interfaz bastante amigable y entendible, en especial el instalador web para Joomla!, el cual va informando al usuario sobre las operaciones que debe ir realizando para la instalación indicando los pasos a realizar uno a uno. Se utilizará de guía la forma en que se guía al usuario durante el proceso de instalación del instalador de Joomla!.

Además de todo lo antes analizado, ninguno de los instaladores estudiados cumple con la característica de poder integrarse con Symfony2. Por lo tanto se pudo constatar que ninguno de ellos se puede reutilizar o puede dar solución a la problemática presentada.

1.5 Modelo de desarrollo de software

Un modelo de Desarrollo de Software es una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software.

El centro CEIGE propone el modelo de desarrollo de software para el CEIGE. Este modelo fue elaborado por dicho centro basado en principios y buenas prácticas propuestas por el SEI, y las metodologías SCRUM y RUP pero adaptados a las características específicas de la UCI. Este detalla el ciclo de vida de los proyectos con la incorporación de los distintos subprocesos dictados por el Nivel II de CMMI (*Capability Maturity Model Integration*), certificación obtenida por el centro en julio de 2011 y reconocida por el SEI (*Software Engineering Institute*) como aval de la calidad de su proceso de desarrollo de software. Se detallan, por tanto, los artefactos a generar en cada momento independientemente de las herramientas o métodos que se utilicen para ello. En general proporciona una guía para regir el proceso de desarrollo de software tecnológico, centrado en la arquitectura y logrando una mayor especialización por parte del equipo de desarrollo *(8)*.

El modelo de desarrollo consta de dos fases: El Inicio o Estudio preliminar y el Desarrollo esta última dividiéndose en varias disciplinas. En la Figura 1 se puede observar el ciclo de vida de los

proyectos desarrollados usando el modelo de desarrollo propuesto por CEIGE.

Figura 1: Ciclo de vida de proyectos del CEIGE

Para la creación del Instalador para soluciones desarrolladas empleando del MT Symfony2 se transitará por la fase de Desarrollo. La fase consta de varias disciplinas, en la disciplina del modelado del negocio se identificarán los procesos que se llevarán a cabo y se desean automatizar obteniéndose el modelo de dominio. En la disciplina de requisitos se preparará todo lo necesario para la correcta captura de requisitos y luego su validación, se especifican los requisitos funcionales y no funcionales obteniéndose el documento de Descripción de requisitos.

Durante el análisis y diseño es modelado el sistema para que soporte todos los requisitos, se lleva a cabo el refinamiento y aprobación de estos, se analizarán todos los artefactos generados en las etapas previas para lograr su comprensión y así seleccionar la arquitectura de datos y del sistema, se identifican activos reutilizables, se definirá la arquitectura de datos seleccionando las

herramientas de trabajo y el dominio de los datos, se elaborarán los diseños de casos de prueba.

En la disciplina de implementación se analizarán los artefactos generados durante el análisis y diseño, ya que estos guiarán la programación, se implementarán los componentes y se probarán para garantizar su pleno funcionamiento. Durante las pruebas internas se entregarán los artefactos, se realizarán pruebas exploratorias, las iteraciones de pruebas, se planificarán y ejecutarán las pruebas definidas por el departamento al producto.

1.6 Tecnologías propuestas

1.6.1 Symfony2:

Symfony2 es un MT diseñado para optimizar el desarrollo de las aplicaciones web que está diseñado para poder soportar el patrón Modelo Vista Controlador, el cual separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony2 sigue una política de tipo *LTS* (*Long Term Support*), o sea que las versiones estables se mantienen durante 3 años sin cambios pero con una continúa corrección de los errores conocidos. Utiliza una licencia *MIT* con la que se pueden hacer aplicaciones web comerciales, gratuitas y/o de software libre. Está desarrollado completamente en PHP 5.3, es compatible con la mayoría de gestores de bases de datos y se puede ejecutar en plataformas Unix, Linux y en plataformas Windows. Puede satisfacer las necesidades de los profesionales y usuarios avanzados dada su flexibilidad, es muy accesible y posee abundante documentación. Es muy utilizado actualmente en los proyectos productivos de la UCI (9).

1.6.2 Doctrine 2.2.1

Doctrine es un mapeador de objetos relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un Sistema de Gestión de Bases de Datos. Brinda la posibilidad de mapear una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases

(convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.

Doctrine necesita un bajo nivel de configuración para empezar un proyecto. Puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros marcos de trabajo. Tiene diversos comportamientos del modelo (conjuntos anidados, internacionalización, *log*, índice de búsqueda) que permiten una mayor accesibilidad a los datos. Symfony2 viene equipado con Doctrine para el manejo de la base de datos (10).

1.6.3 ExtJs 4.2

ExtJs es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como *AJAX2*, *DHTML3* y *DOM4*. Permite crear aplicaciones complejas utilizando componentes predefinidos.

Es compatible con la mayoría de los navegadores permitiendo crear páginas e interfaces web dinámicas. Se encuentra fundamentalmente en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. Se ajusta perfectamente a la concepción de interfaz que se desea para el instalador para soluciones desarrolladas empleando el MT Symfony2 (11).

1.6.4 Apache 2.2

El servidor Apache HTTP es el estándar en la entrega de servicios web, posee gran posibilidad de configuración, robustez y estabilidad. Apache se basa en una plataforma de servicio web de fuente abierta originalmente desarrollada para servidores de Linux/Unix, pero se configuró posteriormente para que funcione con Windows y otros sistemas operativos. El servidor web Apache es gratuito y es muy rápido *(12)*.

1.6.6 PostgreSQL 9.0

PostgreSQL es un motor de base de datos relacional orientada a objetos y de código abierto, es potente y robusto. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y *joins* de gran tamaño. Es multiplataforma

(disponible para Linux y UNIX en todas sus variantes y Windows 32/64bit). Se usará para la creación de la base de datos de la aplicación (13).

1.6.5 Lenguajes de Programación

PHP 5.3.6

Es un lenguaje interpretado de propósito general ampliamente usado, de código abierto especialmente adecuado para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. Además posee capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destacándose su conectividad con PostgreSQL (14).

JavaScript

Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Gran parte de su programación está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas, entre otros. Permite la programación de pequeños scripts y de programas más grandes orientados a objetos, con funciones y estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web para poder acceder a ellos y modificarlos dinámicamente. Es compatible con la mayoría de los navegadores modernos, es soportado por Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros. Es implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas (15).

1.7 Herramientas utilizadas

1.7.1 Mozilla Firefox

Firefox es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es

software libre, publicado bajo una triple licencia GPL/LGPL/MPL. Es un navegador rápido y eficiente, permite instalarle varios Plugins para mejorar sus funcionalidades. Se le puede incorporar el Plugin Firebug, el cual edita, depura y monitorea CSS, HTML y JavaScript de cualquier página web, lo cual es muy útil en la programación web y puede utilizarse para capturar los errores del código durante la implementación del instalador web (16).

1.7.2 Visual Paradigm 8.0

Es una herramienta UML (Lenguaje Unificado de Modelado) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Es fácil de instalar y actualizar y compatible entre ediciones. Soporte ORM: generación de objetos Java desde la base de datos. Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, Vista, 7,8), Linux, Mac OS X, Solaris o Java (17).

Se utiliza el UML para la creación de los diagramas propuestos por el modelo de desarrollo de software utilizado. Mediante esta herramienta se crearon los diagramas de clases con estereotipos web, diagramas de secuencia, los prototipos de interfaz de usuario y el diagrama de componentes.

1.8 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, por lo que es de suma importancia aplicarlos en la construcción del diseño de un sistema. Un patrón de diseño resulta ser una solución a un problema de diseño, se debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Además es reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. Con su uso, se pretende establecer un lenguaje común entre los programadores, contribuir a la reutilización, ahorrar tiempo en la implementación y obtener un producto con calidad (18).

1.8.1 Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP se codifican algunos de ellos, que se aplican al preparar los diagramas de interacción, cuando se asignan las responsabilidades o durante ambas actividades. Existen nueve patrones GRASP los cuales son: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador, Polimorfismo, Fabricación pura, Indirección y No hables con extraños (18).

1.8.2 Patrones GOF

Los patrones de diseño Gang of Four (GoF) que definen una estructura de clases que tratan una situación en particular. Los patrones GOF se clasifican en 3 categorías: creacionales, estructurales y de comportamiento.

Patrones de creación: Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de qué clase crear o cómo crearla. Entre ellos se encuentran Abstract Factory, Builder, Factory Method, Prototype y Singleton (18).

Patrones estructurales: Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Entre ellos se encuentran Adapter, Bridge, Composite, Decorator, Facade y Proxy (18).

Patrones de comportamiento: Los patrones de comportamiento estudian las relaciones de las llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal. Entre ellos se encuentran Chain of Responsibility, Command, Interpreter, Mediator y Memento (18).

1.9 Patrones Arquitectónicos

El patrón Modelo-Vista-Controlador (MVC) fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados. Este separa el modelado

del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir separa en tres capas diferentes los datos de una aplicación, la interfaz de usuario, y la lógica de control (18).

Estas capas son:

El Modelo: Esta capa administra el comportamiento y los datos del dominio de la aplicación, maneja los datos y controla todas sus transformaciones.

Vista: Esta capa maneja la visualización de la información, presenta los datos representados por el modelo mediante la interfaz de usuario en un formato adecuado para que el usuario pueda interactuar con ellos.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo, actúa sobre los datos del modelo y sobre los cambios en la vista respondiendo a eventos y acciones del usuario.

1.10 Conclusiones del capítulo

- El estudio de los instaladores web existentes permitió comprobar que ninguno de ellos puede dar solución al problema planteado, sin embargo se identificaron fortalezas y aspectos que sirvieron para sentar las bases para el desarrollo de esta investigación.
- El estudio realizado sobre las tecnologías y herramientas permitió identificar cuáles eran más idóneas para el desarrollo de la investigación.

Capítulo 2: Análisis y Diseño.

2.1 Introducción

En este capítulo se realiza la captura de requisitos funcionales, su validación, se generan los artefactos y diagramas necesarios para la mejor comprensión de los requisitos funcionales obtenidos y del producto que se desea desarrollar. También se describen todas las clases necesarias para el diseño de la aplicación y los prototipos de interfaz de usuario.

2.2 Modelo conceptual

Un modelo conceptual en la resolución de problemas e ingeniería de software, es un modelo de todos los temas relacionados con un problema específico. En él se describen las distintas entidades, sus atributos, papeles y relaciones, además de las restricciones que rigen el dominio del problema, este se crea con el fin de representar el vocabulario y los conceptos clave del dominio del problema (19).

En la Figura 2 se muestra el diagrama del modelo de dominio perteneciente al instalador para soluciones desarrolladas empleando el MT Symfony2.

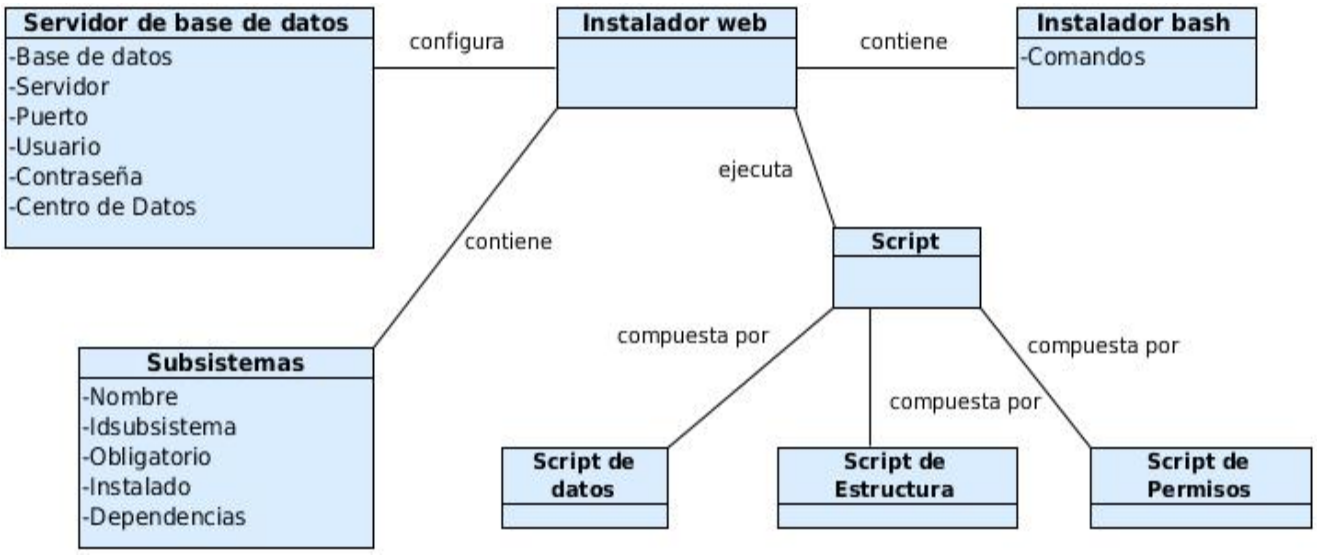


Figura 2: Modelo conceptual del Instalador

A continuación se muestra una descripción de los principales conceptos asociados:

Instalador: Si se busca la definición de instalar se encontrará que es incorporar en una computadora una aplicación o un dispositivo para poder ser utilizado. Hasta las aplicaciones más elaboradas suelen tener un programa instalador que facilita el proceso de instalación.

Por tanto un instalador se define como un programa que copia archivos, configura herramientas y ficheros para permitir el funcionamiento correcto de una aplicación o programa determinado (20).

Instalador bash: se encarga de instalar todas las tecnologías y servicios necesarios para el correcto funcionamiento del instalador web quedando instalado Symfony2 con todas las configuraciones que necesita para su funcionamiento. Este realiza todas sus operaciones mediante comandos secuenciales propios del sistema Linux, y descarga las herramientas necesarias desde el repositorio de Linux.

Instalador web: terminada la ejecución del script bash se ejecuta el instalador web, el cual es accesible desde el navegador y mediante una interfaz amigable permite la configuración de la base de datos y la instalación de los subsistemas.

Servidor de base de datos: el instalador web configura el servidor de base de datos suministrándole los siguientes datos: el servidor y el puerto que utilizará, el nombre y la contraseña de un usuario con permisos para crear y manipular bases de datos, el nombre de la base de datos a crear y el valor de el centro de datos.

Subsistemas: el instalador web posee información de los subsistemas que va a instalar: el nombre del subsistema, un identificador único, las dependencias que posee, si es obligatoria su instalación y si se encuentra instalado.

Script: son las instrucciones que se ejecutan durante la instalación. Los scripts son archivos adicionales que contienen las sentencias que deben ser ejecutadas en la base de datos.

Script de Datos: Estos contienen los datos necesarios para el funcionamiento del sistema.

Script de Estructura: Forman la estructura necesaria para el funcionamiento de la base de datos y las relaciones entre todos los componentes.

Script de Permisos: Se ejecutan cuando ya está creada correctamente la base de datos con sus usuarios y roles. Son los que otorgan los permisos necesarios y los privilegios para el uso de la base de datos.

2.3 Requisitos de Software

2.3.1 Técnicas para la captura de requisitos

Para realizar una eficiente captura de requisitos respondiendo a las necesidades de los proyectos productivos se utilizaron las siguientes técnicas de captura de requisitos:

Tormenta de ideas

Mediante esta técnica de trabajo en grupo, se pretende obtener el mayor número de ideas o soluciones a cuestiones planteadas en el menor tiempo posible, aprovechando la capacidad creativa de las personas. Se emplea fundamentalmente cuando se requieren soluciones creativas e ideas innovadoras, después de agotar los esfuerzos individuales sin obtener resultados satisfactorios (21).

Observación de sistemas semejantes

Otra técnica de extracción de requisitos es realizar el estudio de soluciones semejantes. Probablemente no permitan obtener los requisitos directamente, pero sí puede dar una buena base para saber qué se pretende buscar y obtener una colección de requisitos de partida que permita agilizar la toma de requisitos (22).

Mediante el estudio y análisis de los instaladores web existentes en la actualidad se pudo hallar características o funcionalidades comunes de estos, las cuáles pueden ser utilizadas y por tanto brindan una guía para la captura de los requisitos funcionales correspondientes al instalador web.

2.3.2 Requisitos Funcionales

Luego de analizar la problemática planteada y utilizando las técnicas para la captura de requisitos descritos se han obtenido los siguientes requisitos funcionales:

- Instalar y configurar tecnologías para arquitectura de referencia utilizando Symfony2.
- Crear base de datos a partir de la configuración definida.
- Instalar Subsistemas.

Descripción del requisito funcional crear base de datos a partir de configuración definida:

Precondiciones	Han sido copiadas en la PC todas las carpetas y archivos del instalador y el script bash instalador.sh ha culminado su ejecución.
Flujo de eventos	
Flujo básico Crear base de datos a partir de configuración definida.	
1	Se introducen los parámetros de configuración de la base de datos. (Base de Datos, Servidor, Puerto, Usuario, Contraseña)
2	El usuario presiona el botón **Siguiente**.
3	El sistema valida los datos introducidos (ver validación 1).
4	El sistema comprueba que los datos introducidos son correctos para la conexión con la base de datos.
5	El sistema realiza la conexión al servidor de base de datos.
6	El sistema crea una nueva base de datos con el nombre suministrado.
7	Se registran los parámetros en el archivo parameters.yml.
8	El sistema muestra el mensaje: “La base de datos ha sido creada satisfactoriamente.”
9	El sistema re direcciona hacia la siguiente página del instalador.
10	Concluye el requisito.
Pos-condiciones	
1	Se creó la base de datos y se configuró el archivo parameters.yml.
Flujos alternativos	
Flujo alternativo 3.a Información errónea	
1	El sistema señala los campos con datos incorrectos.
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	NA
Flujo alternativo 3.b Información incompleta	
1	El sistema muestra el mensaje: “Debe llenar todos los campos.” y señala los campos vacíos.
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	NA
Flujo alternativo 4.a Datos erróneos en los parámetros de configuración	

1	El sistema muestra un mensaje de error informando cuál es el parámetro de conexión erróneo.	
2	Volver al paso 1del flujo básico.	
Pos-condiciones		
1	NA	
Flujo Alternativo *.a		
1	El usuario presiona el botón **Limpiar**.	
2	Se limpia el contenido de todos los campos.	
Pos-condiciones		
1	NA	
Validaciones		
1	Ver Modelo Conceptual. (..\Tesis\Artefactos\CEIGE_Modelo_conceptual.doc)	
Conceptos	**Servidor de base de datos**	Visibles en la interfaz: Base de datos, Servidor, Puerto, Usuario, Contraseña, Centro de datos. Utilizados internamente: NA
Requisitos especiales	NA	
Asuntos pendientes	NA.	

2.3.3 Requisitos no funcionales

Los requisitos no funcionales hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo. Un requisito no funcional especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema y no su comportamiento específico (23).

Usabilidad

El sistema será creado para especialistas pero podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo del sistema operativo Linux.

Eficiencia

Los tiempos de respuesta y velocidad de procesamiento serán rápidos.

Soporte

La aplicación contará antes de su puesta en marcha con un período de pruebas para detectar posibles errores y se le dará mantenimiento de acuerdo a las necesidades de uso por parte de los centros productivos que lo usen para el despliegue.

Políticos Culturales (CUL)

El producto no debe contener palabras en otros idiomas.

Restricciones de diseño.

Se utilizaran las tecnologías propuestas por el centro.

Ambiente

Para que el sistema funcione es necesario que el sistema operativo utilizado sea Linux.

2.3.4 Validación de requisitos

La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente, es decir, es el proceso de evaluar un sistema o componente durante o al final del proceso de desarrollo para determinar si este satisface los requerimientos especificados. Demuestra que el producto cumplirá con el uso que tiene definido, asegurando que se está construyendo lo correcto (26).

La validación de requisitos tiene como misión demostrar que los requisitos obtenidos definen realmente el sistema que el usuario necesita o el cliente desea. Se chequea la consistencia, completitud, realismo y verificabilidad de estos.

Revisión técnica formal

La revisión técnica formal del documento de requisitos consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos por varias personas, revisando anomalías y omisiones. Los conflictos, contradicciones, errores y omisiones deben señalarse durante la revisión y registrarse formalmente. Con ello solamente se puede validar la correcta interpretación de la información transmitida (27).

Se creó un equipo compuesto por los analistas del departamento y se analizaron todos los requisitos obtenidos y su respectiva descripción para capturar fallos o incongruencias.

Prototipos de interfaz de usuario

Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema y valorar si cumple con las necesidades. Es un diseño en menor porción que verificará que las especificaciones han sido realizadas de acuerdo con los requisitos. Utilizando la herramienta Visual Paradigm se crearon los prototipos de interfaz de usuario (27).

Para un mejor entendimiento de las vistas a las que se quiere arribar y un mayor entendimiento con el usuario final se crearon los siguientes prototipos de interfaz de usuario validando así los requisitos funcionales capturados.

En la Figura 3 se muestra la propuesta de interfaz de usuario para capturar los datos referentes a la configuración de la base de datos.

Figura 3: Interfaz de Usuario Configurar Base de Datos

También se muestra el prototipo de interfaz de usuario para la selección de los subsistemas, el cual muestra en una lista los subsistemas disponibles para instalar permitiendo seleccionar los que se desean instalar (Figura 4).

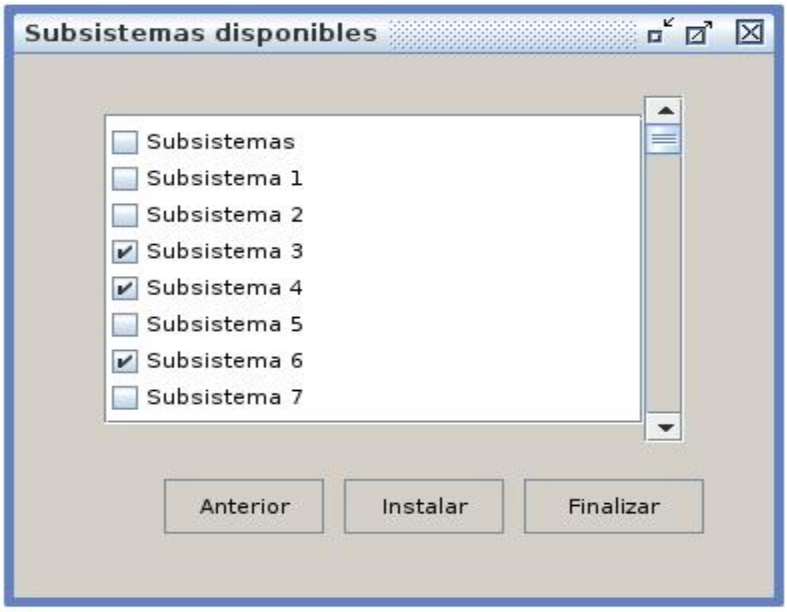


Figura 4: Interfaz de usuario Seleccionar Subsistemas

2.4 Patrones de diseño

2.4.1 Utilización de los Patrones GRASP

En la realización del instalador para soluciones desarrolladas empleando el marco de trabajo Symfony2 se utilizaron los siguientes patrones:

Patrón experto: existen clases que poseen la información necesaria para cumplir con las responsabilidades. El método cargarSubsistemasAction de la clase SubsistemasController posee la información necesaria para obtener los datos requeridos de los subsistemas que se encuentran disponibles y enviar la respuesta a la vista quien se encarga de mostrar estos datos.

Patrón bajo acoplamiento: con este patrón se aumenta la reutilización asignando las responsabilidades a las clases encargadas disminuyendo así las dependencias entre clases. Las clases InstaladorController y SubsistemasController no dependen entre si, encargándose cada una de sus tareas de forma independiente.

Patrón alta cohesión: se utiliza para que la complejidad se mantenga manejable, así que se

asignan las responsabilidades de manera que la cohesión se mantenga alta. Symfony2 permite una alta cohesión ya que separa la lógica de la presentación y el modelo por lo que todos los procesos y funciones se encuentran en las clases Controller. Desde un formulario se envían los datos sobre los subsistemas seleccionados a la clase SubsistemasController y este realiza todas las operaciones necesarias con esta información.

Patrón no hables con extraños: se utiliza para evitar que las clases posean dependencias innecesarias. Cada entidad se hace cargo de sus funciones sin interferir o utilizar funciones de otras clases que no necesite. Para el manejo de los parámetros de la base de datos que utilizan las dos clases controladoras del instalador web se utiliza el servicio sensio_distribution.webconfigurator evitando así que tengan ambas clases que estar utilizando métodos de la otra.

Patrón controlador: las peticiones no se hacen directamente, todas son manejadas por el controlador frontal de Symfony2 nombrado app_dev.php**,** el cual posee la configuración necesaria para realizar el correcto enrutamiento de las peticiones realizadas.

2.4.2 Utilización de los Patrones GOF

A continuación se describen los patrones GoF utilizados para la creación del instalador para soluciones desarrolladas empleando el marco de trabajo Symfony2:

Facade: Proporciona una interfaz única que simplifica los servicios generales del sistema, definiendo un comportamiento independiente de donde se vaya a utilizar, siendo esta más fácil de utilizar (18).

Existe un archivo llamado layout.html.twig el cual posee la plantilla global, la cual contiene la interfaz que es usada por todas las vistas evitando tener que repetir el mismo código para cada interfaz. Todas las vistas heredan de este archivo.

Mediator: Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente (18).

No se utilizan directamente las clases del bundle de terceros SensioDistributionBundle, sino que se accede a las funcionalidades de ellas utilizando un servicio por lo que se pueden utilizar sus

métodos sin instanciar las clases.

2.5 Patrones arquitectónicos

Symfony2 está diseñado para soportar la utilización del patrón MVC. Este patrón fue utilizado en la creación del instalador de la siguiente manera:

Symfony2 utiliza este patrón para lograr una mayor organización del código y que el desarrollo de las aplicaciones sea rápido. Las clases controladoras se nombran de la siguiente forma: nombreController.php y poseen la lógica de la aplicación. Estas manejan los datos que le son suministrados desde la vista o el modelo y realizan las operaciones necesarias manipulando estos datos.

La vista se encarga de mostrar la información generada al usuario, permitiéndole interactuar con ella y transformarla. Se crean las interfaces usando ExtJs a partir de la información suministrada por las controladoras.

La información que utiliza la aplicación se encuentra en archivos xml ubicados en los directorios del bundle o son enviadas desde los formularios de la vista.

2.6 Diagrama de clases de diseño con estereotipos web

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real (18).

El diagrama de clases de diseño con estereotipos web modela los elementos web del sistema con las interacciones entre ellos y sus relaciones incluyendo los estereotipos los cuales le otorgan una particularidad, un tipo a cada clase dentro de la estructura.

Se obtuvo un diagrama por cada requisito funcional, en el presente documento se muestran el correspondiente al requisito crear base de datos a partir de la configuración definida.

2.6.1 Crear base de datos a partir de la configuración definida

En la Figura 5 se muestra el diagrama correspondiente al requisito Crear base de datos a partir de la configuración definida.

En el diagrama se observa como la página cliente llamada principal.html.twig extiende de la página layout.html.twig la cual contiene la interfaz de la aplicación usando la librería ExtJs y hojas de estilo css.

La página cliente utiliza un archivo javascript para crear el formularios encargado de capturar los datos referidos a la configuración de la base de datos y enviarlos a al controlador frontal de la aplicación, quien ha su vez envía estos datos a la clase InstaladorController encargada de utilizarlos realizando todas las operaciones lógicas para la creación de la base de datos.

La clase InstaladorController hereda de la clase SymfonyController y utiliza un servicio del software de terceros Sensio-DistributionBundle, el cual permite manipular los datos del archivo parameters.yml (el encargado de manejar la conexión a la base de datos en Symfony2) para de esta manera modificarlo con los datos introducidos por el usuario.

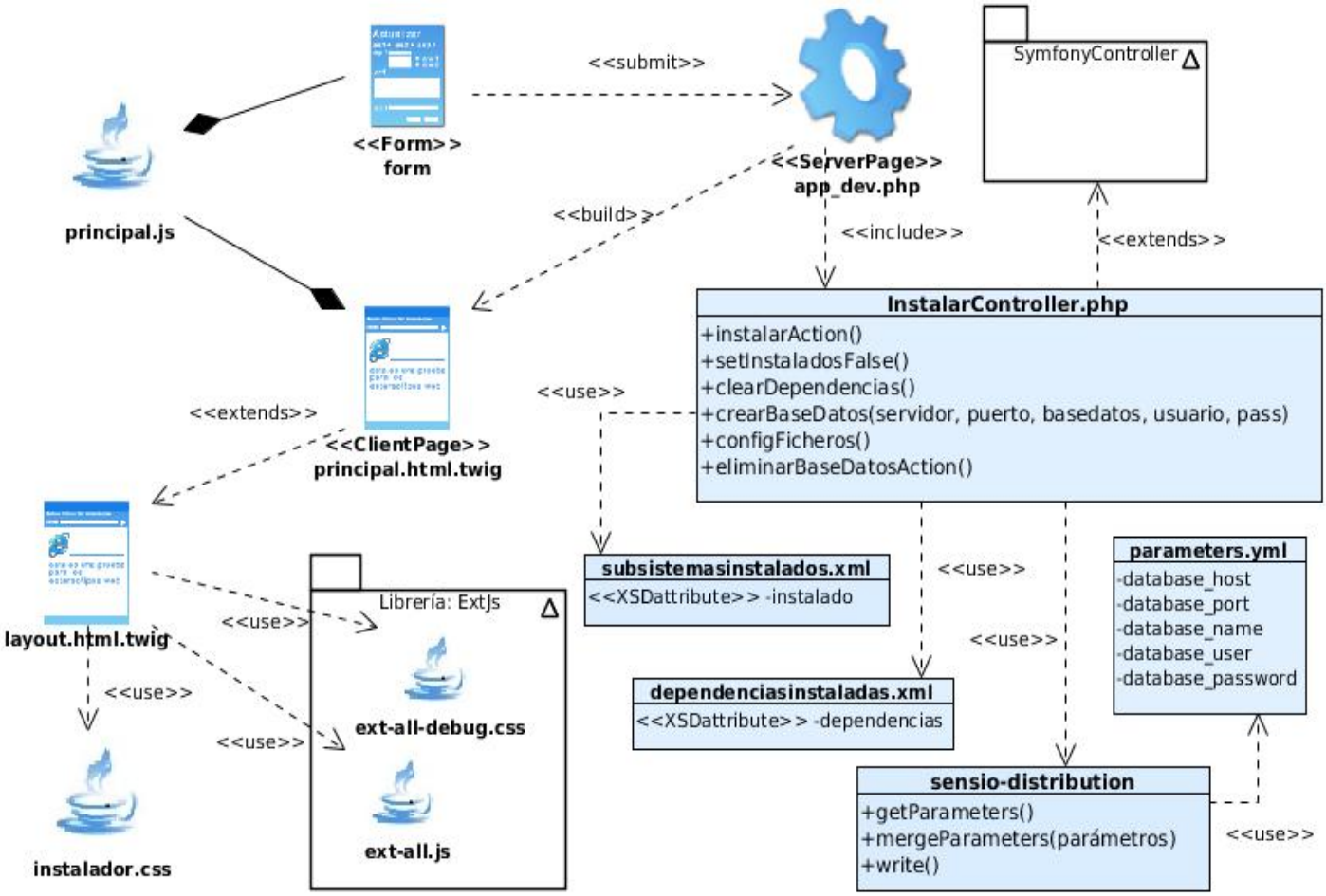


Figura 5: Diagrama de clases Crear base de datos a partir de la configuración definida.

2.7 Diagrama de interacción

El diagrama de interacción, representa la forma en la que los actores y objetos se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. Muestran una interacción, que consiste de un conjunto de objetos y sus relaciones, incluyendo los mensajes que puedan ser realizados entre ellos. Son importantes para modelar los aspectos dinámicos de un sistema y para construir sistemas ejecutables a través de ingeniería hacia adelante e ingeniería inversa (24).

2.7.1 Diagrama de secuencia

Los diagramas de secuencia describen la interacción entre los objetos de una aplicación a través del tiempo y los mensajes recibidos y enviados por ellos. Se modela para cada caso de uso. Muestran gráficamente los eventos que originan los actores dentro de un sistema y el orden en el que se comunican con las clases interfaces y la controladora. Se hace énfasis en el paso de mensajes, en cómo se desenvuelven sobre el tiempo, lo cual es una manera útil para visualizar el comportamiento dinámico en el contexto de un escenario de un caso de uso (25).

Se muestra el diagrama de secuencia correspondiente al requisito crear base de datos a partir de la configuración definida (Figura 6). Donde se observa con claridad el orden en que ocurren las operaciones, de donde parte cada una y las clases que intervienen en cada momento.

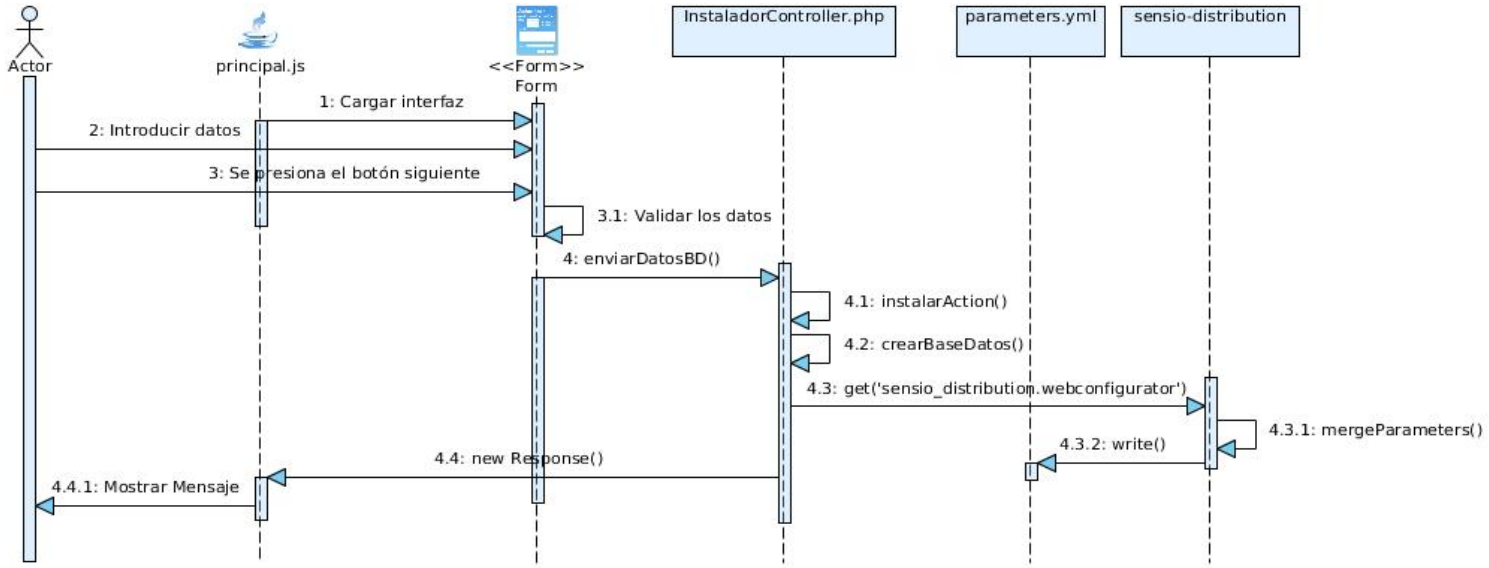


Figura 6: Diagrama de secuencia Crear base de datos a partir de la configuración definida.

2.8 Métricas de software

Para evaluar la calidad del diseño del software se han creado unas métricas básicas que desde un esquema sencillo brindan la posibilidad de analizar y evaluar la calidad del diseño de un software atendiendo a los atributos siguientes:

Responsabilidad: consiste en la responsabilidad que posee una clase para el modelado de un concepto o dominio.

Complejidad de implementación: consiste en el grado de dificultad que puede poseer la implementación del diseño de clases.

Reutilización: consiste en el grado de reutilización que puede tener una clase dentro del diseño del software.

Acoplamiento: consiste en el grado de dependencia que puede tener una clase con otras. Está muy ligada a la característica de reutilización.

Complejidad del mantenimiento: consiste en el grado de esfuerzo que será necesario para realizar un arreglo, una mejora o una corrección de algún error del diseño de un software.

Cantidad de pruebas: consiste en el número o el grado de esfuerzo para realizar las pruebas que aseguren la calidad del producto diseñado (26).

2.8.1 Tamaño operacional de clases

El tamaño operacional de clases (TOC) está definido por la cantidad de procedimientos que posea una clase. Esta métrica fue propuesta por Lorenz y Kidd (27).

Tabla 2: Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC).

Atributos	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.

Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	<= Prom.

Tabla 3: Atributos de calidad de la métrica TOC

Atributo de Calidad	Modo en que Afecta
Responsabilidad	El aumento del TOC implica el aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	El aumento del TOC implica el aumento de la complejidad de implementación de la clase.
Reutilización	El aumento del TOC implica la disminución del grado de reutilización de la clase.

Tabla 4: Resultados de la aplicación de la métrica TOC

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
InstaladorController	6	Baja	Baja	Alta
SubsistemasController	9	Media	Media	Media

En las siguientes gráficas se muestra el resultado obtenido de la aplicación de la métrica TOC:

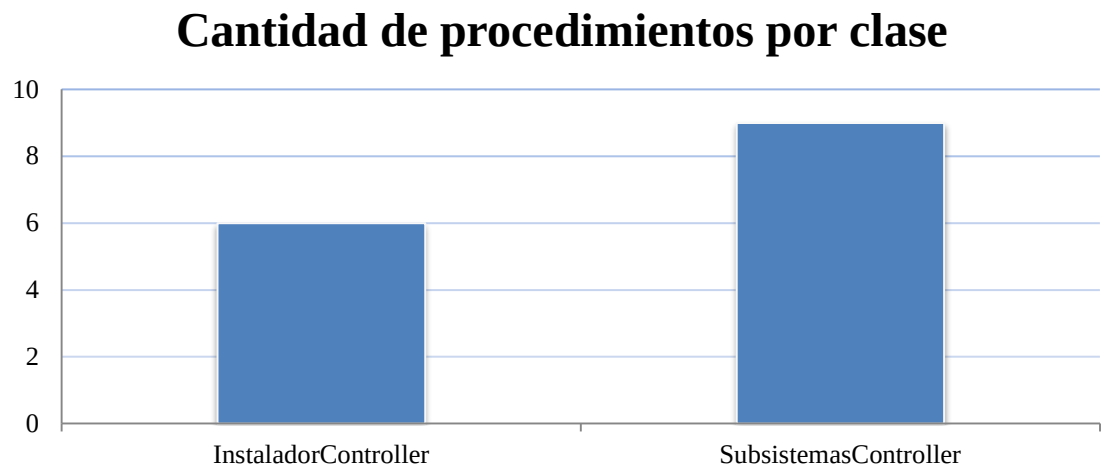


Figura 7: Cantidad de procedimientos por clase para aplicación de la métrica TOC

Porciento de clases por intervalos de procedimiento obtenido durante la aplicación de la métrica TOC:

Figura 8: Cantidad de clases por intervalos de procedimientos

Porciento de clases por categoría del atributo Responsabilidad obtenido durante la aplicación de la métrica TOC:

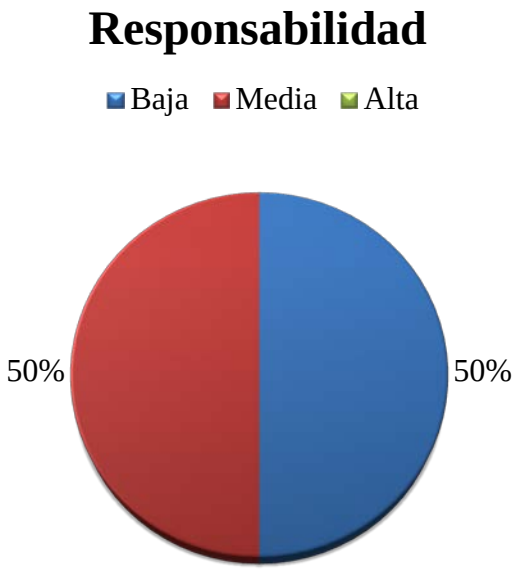


Figura 9: Porciento de clases por categoría del atributo Responsabilidad

Porciento de clases por categoría del atributo Complejidad obtenido durante la aplicación de la métrica TOC:

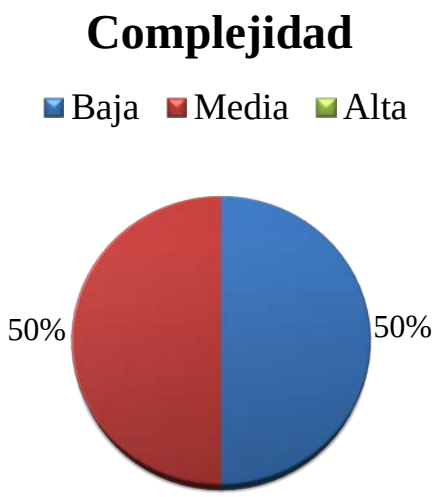


Figura 10: Porciento de clases por categoría del atributo Complejidad

Porciento de clases por categoría del atributo Reutilización obtenidos durante la aplicación de la métrica TOC:

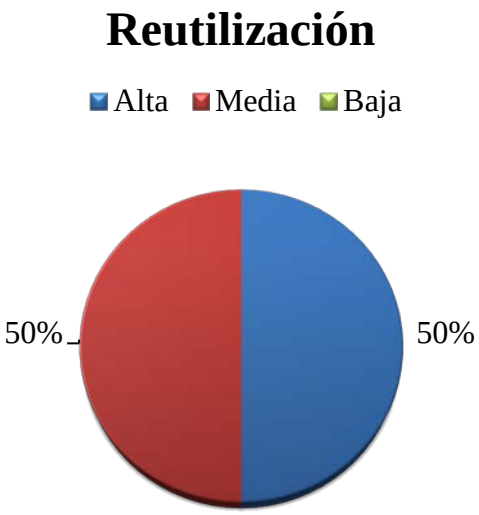


Figura 11: Porciento de clases por categoría del atributo Reutilización

Como se muestra en la Figura 8 todas las clases poseen entre 6 y 10 procedimiento, lo que propicia que los niveles de responsabilidad y complejidad nunca llegan a ser elevados,

manteniéndose bajos o medios, tal como lo demuestran las Figuras 9 y 10, esto potencia que la reutilización de estas clases sea elevada lo que permite señalar que existe una calidad aceptable en el diseño.

2.8.2 Relaciones entre clases

La métrica Relaciones entre clases (RC) está dada por el número de relaciones presentes entre una clase con otra (26).

Tabla 5: Rango de valores para los criterios de evaluación de la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 6: Relaciones entre clases

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento de las RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento de las RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento de las RC implica una disminución en el

	grado de reutilización de la clase.
Cantidad de pruebas	Un aumento de las RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 7: Resultados de la aplicación de la métrica RC

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cantidad de Pruebas
InstaladorController	1	Bajo	Baja	Alta	Baja
SubsistemasController	1	Bajo	Baja	Alta	Baja

En las Figuras 13, 14, 15 y 16 se muestran gráficamente los resultados obtenidos de la aplicación de la métrica RC para los atributos cantidad de dependencias, acoplamiento, cantidad de pruebas y reutilización.

La siguiente gráfica representa los resultados obtenidos de la aplicación de la métrica RC:

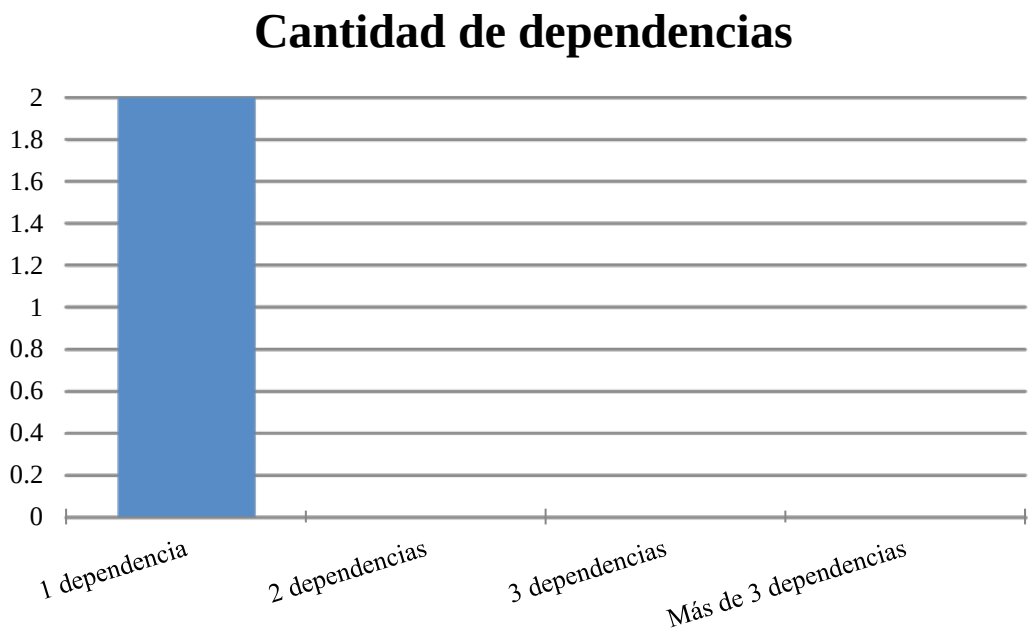


Figura 12: Resultados obtenidos de la aplicación de la métrica RC

Gráfica que representa el porciento de clases con respecto a la cantidad de dependencias que poseen:

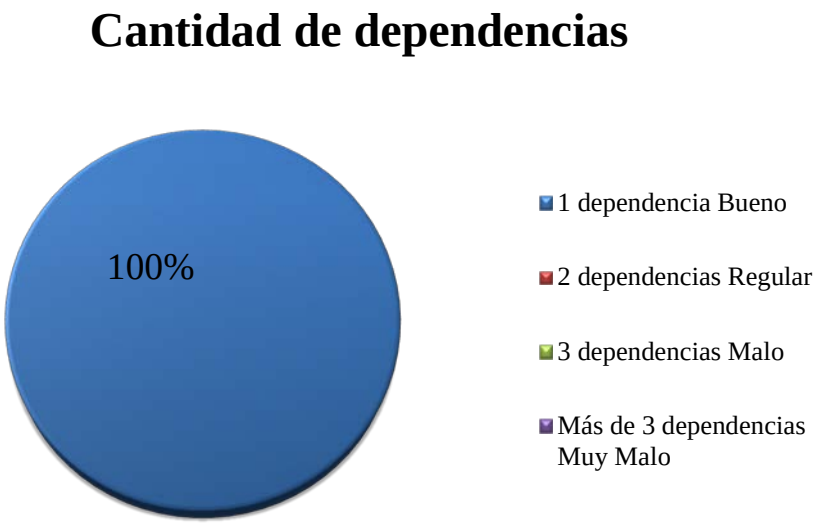


Figura 13: Porciento de clases por cantidad de dependencias

En la gráfica se muestra el porciento de clases con respecto al atributo acoplamiento, que se obtuvieron durante la aplicación de la métrica RC:

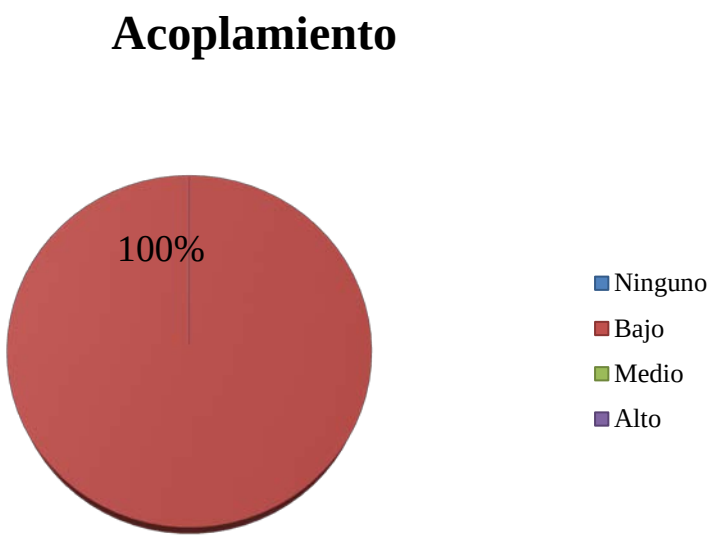


Figura 14: Porciento de clases por categoría del atributo Acoplamiento

En la gráfica se muestra el porciento de clases con respecto al atributo complejidad de mantenimiento obtenido durante la aplicación de la métrica RC:

Figura 15: Porciento de clases atributo Complejidad del mantenimiento

En la gráfica se muestra el porciento de clases con respecto a la cantidad de pruebas, que fue obtenido durante la aplicación de la métrica RC:

Figura 16: Porciento de clases por categoría del atributo Cantidad de pruebas

En la gráfica se muestra el porciento de clases con respecto a la reutilización obtenidos durante la aplicación de la métrica RC:

Figura 17: Porciento de clases por categoría del atributo Reutilización

La aplicación de la métrica RC mostró que las clases poseen muy poca dependencia entre si como se muestra en la Figura 12. El 100% de las clases poseen valores bajos para los atributos acoplamiento entre clases, cantidad de pruebas y complejidad de mantenimiento por lo cual el mantenimiento de estas no será muy complejo y la reutilización para el 100% de las clases es alta. Este resultado demuestra que los atributos de calidad se encuentran en niveles satisfactorios para la métrica RC.

2.9 Conclusiones del capítulo

- La aplicación de técnicas para la captura de requisitos posibilitó identificar las funcionalidades del instalador a partir de las necesidades del cliente.
- La especificación de requisitos brindó una mayor claridad de las funcionalidades que debía incluir el instalador.
- La validación de los requisitos identificados, a partir de la aplicación de revisiones técnicas formales, demostró que los requisitos estaban descritos de forma correcta y clara.
- La generación de los diagramas correspondientes al diseño de la aplicación facilitó la comprensión de las relaciones entre las clases presentes en el modelo.
- La aplicación de las métricas TOC y RC para la validación del diseño arrojó como resultado que el diseño es sencillo y que no presenta una alta complejidad de implementación.

Capítulo 3: Implementación y Pruebas.

3.1 Introducción

En este capítulo se muestra el diagrama de componentes obtenido, se describen los estándares de codificación utilizados, así como una descripción de las principales características de la solución obtenida. Además se abordan los distintos tipos de pruebas a las que se le aplicaron al instalador para asegurar su calidad y funcionalidad.

3.2 Diagrama de Componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces (28).

En la Figura 18 se pueden apreciar los diferentes componentes que conforman la solución con las relaciones de uso entre ellos.

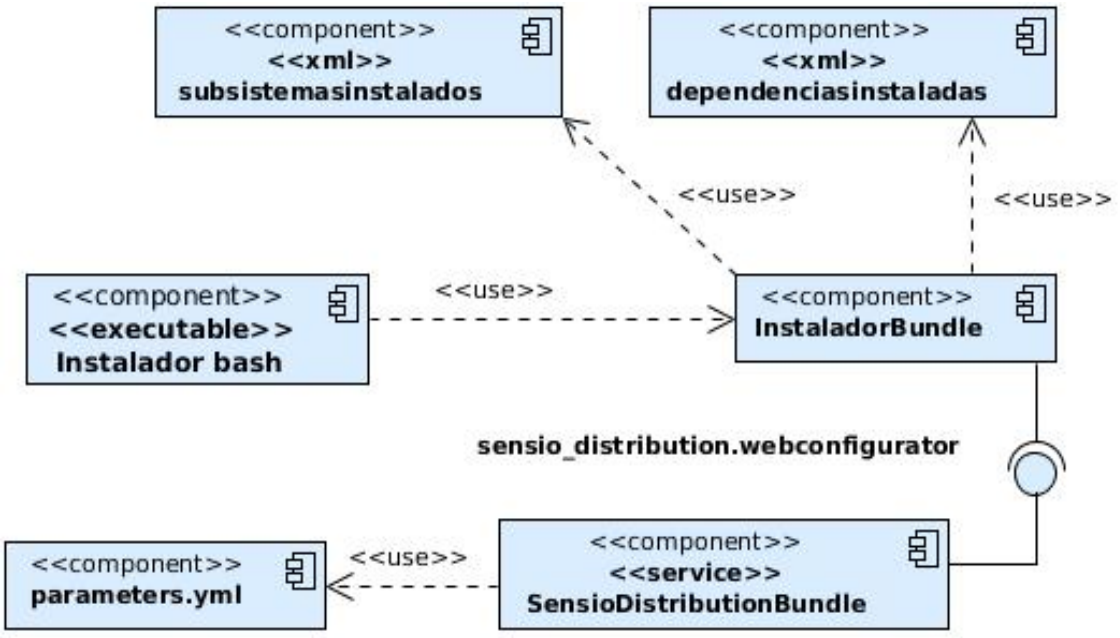


Figura 18: Diagrama de Componentes

El componente Instalador bash es un ejecutable que realiza una serie de operaciones en el Shell (intérprete de comandos) de Linux y finalizadas estas ejecuta el navegador web Firefox con la url configurada para mostrar la página de bienvenida de InstaladorBundle; bundle encargado de la creación de la base de datos, el cual usa a su vez el servicio sensio_distribution.webconfigurator propio del software de terceros SensioDistributionBundle, el cual manipula la información del archivo parameters.yml. InstaladorBundle también usa los archivos subsistemasinstalados.xml y dependenciasinstaladas.xml manejando la información contenida en ellos.

3.3 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software (29).

Durante la implementación del instalador para soluciones desarrolladas empleando el marco de trabajo Symfony2 se utilizó para nombrar métodos la nomenclatura CamelCase, la cual consiste en escribir palabras compuestas eliminando los espacios intermedios y poniendo minúscula la primera letra y mayúsculas las demás primeras letras de cada palabra. Ejemplos del uso de la nomenclatura CamelCase son los métodos cargarSubsistemasAction(), obtenerScripts() y crearBaseDatos().

Para el nombre de las clases se utilizó la nomenclatura UpperCamelCase, la cual es parecida a la CamelCase pero además la primera letra del nombre de la clase se escribe mayúscula. Ejemplos del uso de esta nomenclatura son las clases InstaladorController y SubsistemasController.

3.4 Tratamiento de los errores

El tratamiento de los errores en cualquier aplicación web es importante para garantizar el funcionamiento correcto de todos los procesos. La identificación y captura de los errores permite reconocer los problemas que pueden ocurrir durante la ejecución de las operaciones y realizar una acción al respecto.

3.4.1 Tratamiento de los errores en el instalador bash

Para el instalador bash es importante el tratamiento de los errores ya que actúa sobre archivos del sistema y un error podría incidir directamente sobre la estabilidad del sistema operativo.

Con el objetivo de realizar sus funciones correctamente y evitar errores el script realiza una serie de verificaciones antes de continuar su ejecución. Este comprueba que está siendo ejecutado por un usuario root (usuario con permisos de administrador en Linux), debido a que realiza operaciones que sólo pueden ser utilizadas por un usuario de este tipo. Además verifica que está siendo ejecutado desde la carpeta raíz del proyecto Symfony que va a ser utilizado debido a que requiere utilizar variables utilizando esa dirección. De igual manera comprueba que todas las carpetas y archivos necesarios se encuentran en ese directorio. Si no cumple con alguna de estas condiciones el script muestra los respectivos mensajes de error y finaliza.

En la Figura 19 se muestra el código que asegura que se cumplan estas condiciones y en las Figuras 20 y 21 se muestran los mensajes de error asociados a estas validaciones.

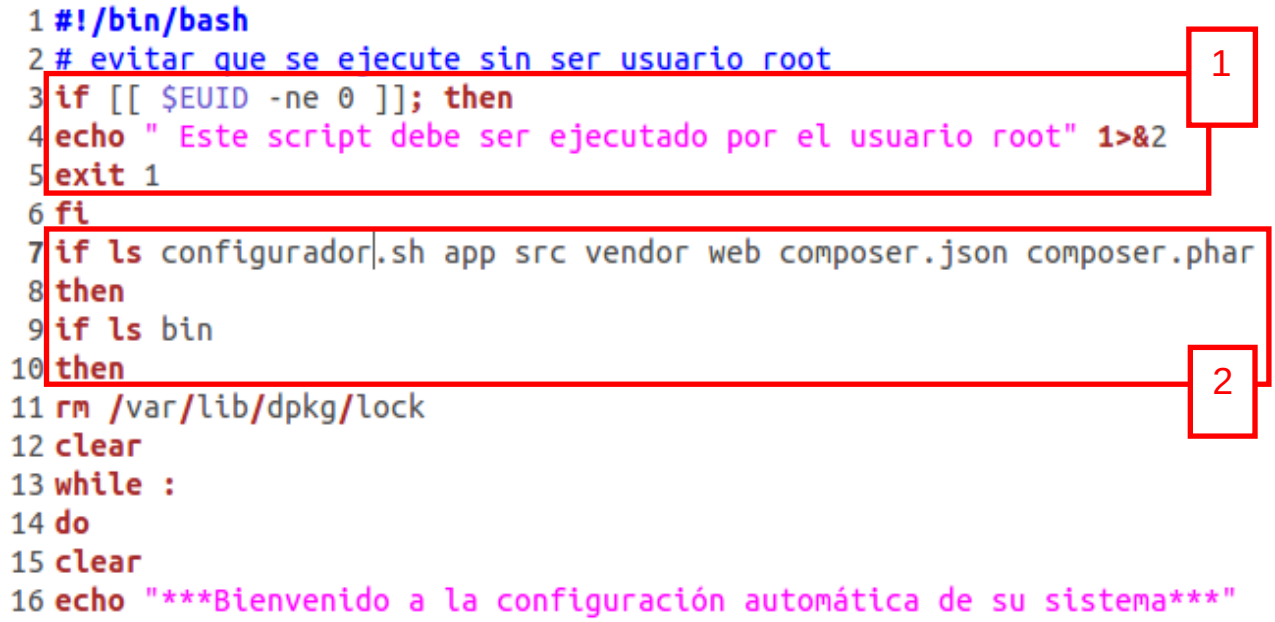


Figura 19: Fragmento de código del script bash

En “1” se asegura que el script sea ejecutado por un usuario con permisos de root, en caso contrario muestra un mensaje de error y finaliza, en “2” se asegura que todos los archivos del instalador se hallan en la carpeta raíz del proyecto y que la terminal se encuentre en esa carpeta (/ruta/hasta/Symfony/) y no se esté ejecutando desde otra dirección.

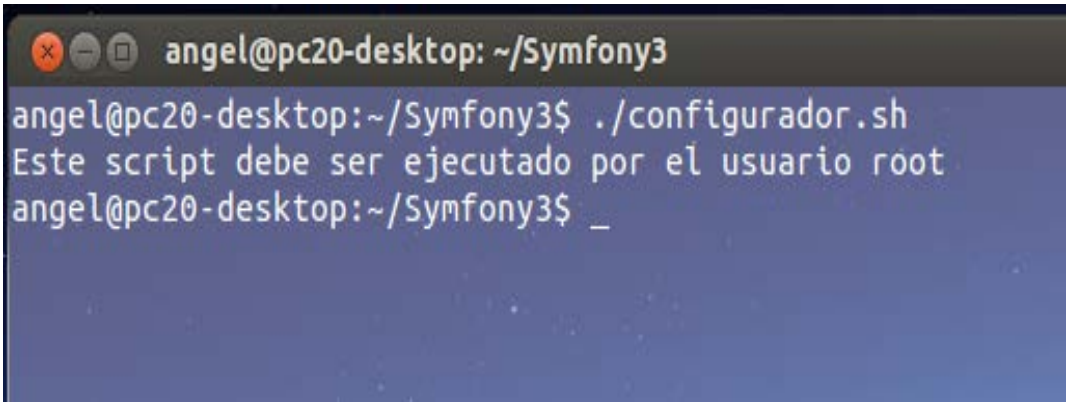


Figura 20: Error cuando el script es ejecutado por usuario sin permisos de root

En la Figura 21 se muestra el mensaje de error que devuelve el script al ejecutarlo desde otra dirección, en este caso desde el directorio /home/angel de la siguiente manera:

root@pc20-desktop:/home/angel# ./Symfony3/instalador.sh

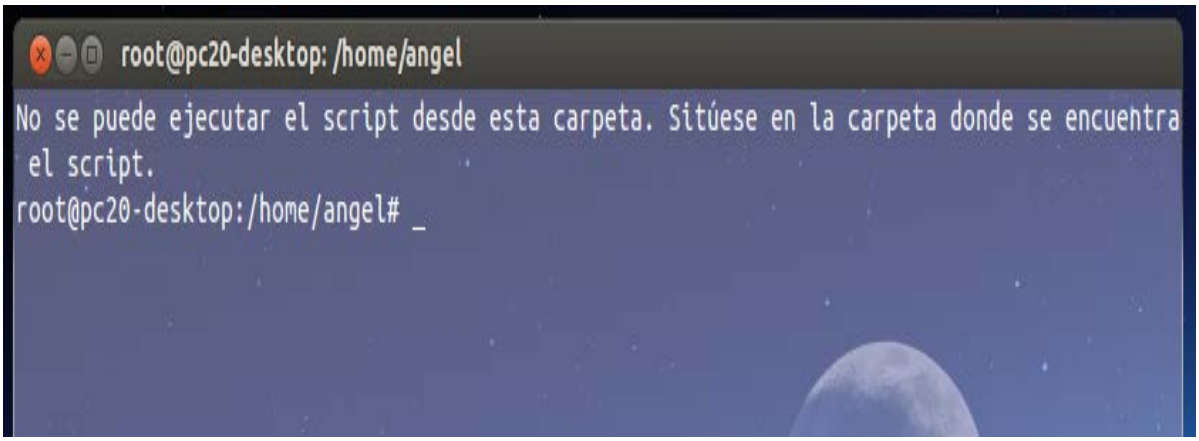


Figura 21: Error cuando la terminal no se encuentra en la carpeta raíz

3.4.2 Tratamiento de los errores en el instalador web

Para evitar los errores más comunes, se realiza la validación de datos en las vistas utilizando JavaScript se comprueba que los datos cumplan con reglas determinadas por expresiones regulares, evitando así que se introduzcan valores inválidos como letras en campos donde solo se admiten números. En la Figura 22 se muestra como se señala el campo con los datos inválidos mostrando la información relacionada al error.

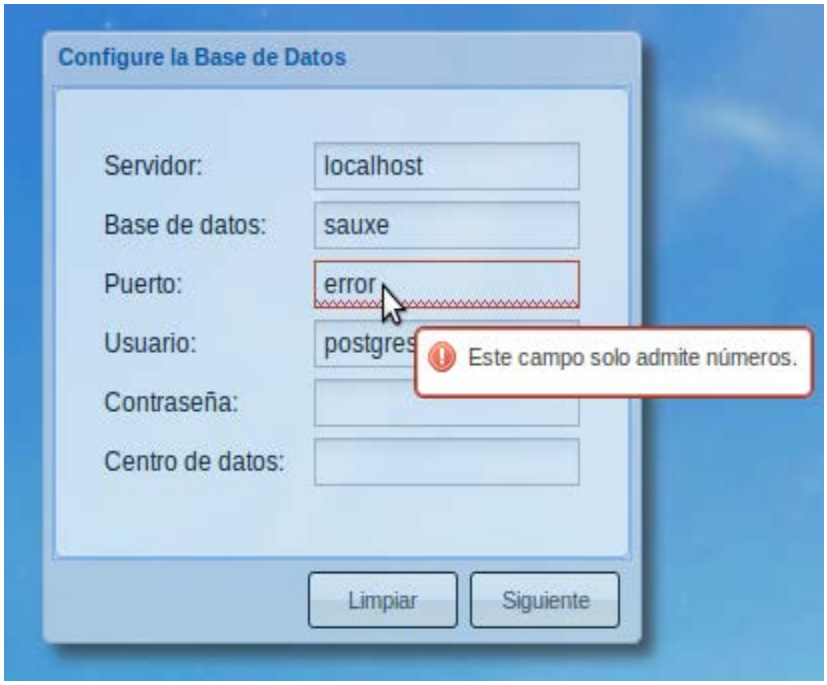


Figura 22: Validación de datos en las vistas

Para el tratamiento de errores se utiliza también la captura de excepciones utilizando las palabras claves *try*, *catch* como se muestra en la Figura 23, donde se trata de establecer una conexión a la base de datos dentro del bloque *try* y en caso de ocurrir algún fallo se captura el error que devuelve el objeto *PDO* en el bloque *catch* y se muestra al usuario con la información referente como se muestra en la Figura 24.

```
    try {
     $conexion = new \PDO("pgsql:host=$servidor;port=$puerto", $usuario, $pass);
} catch (\Exception $e) {
     array_push($errores,"<b style='background-color:#ff1500'> No se pudo establecer la conexion con el servidor d
     $conexion=null;
    return array(1,$mensajes,$errores);
}
```

Figura 23: Tratamiento de errores mediante try-catch

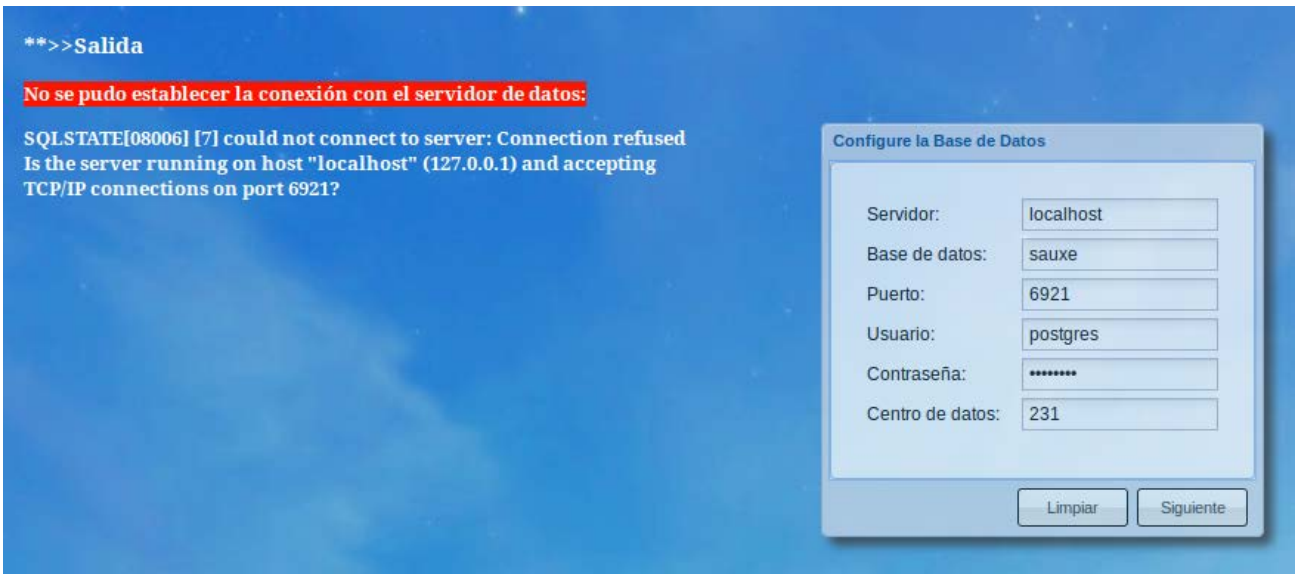


Figura 24: Mostrando las excepciones capturadas

De esta forma en caso de ocurrir cualquier error el usuario puede obtener la información necesaria para saber de donde proviene y en el caso de los usuarios más avanzados poder corregirlo.

3.5 Descripción de la solución

El instalador para soluciones desarrolladas empleando el MT Symfony2 es un producto diseñado con el propósito de la instalación y despliegue de productos informáticos que utilicen como base tecnológica Symfony2.

Posee un instalador bash diseñado para ser ejecutado previamente, ya que este instala y configura correctamente Apache, PostgreSQL y PHP con todas las extensiones necesarias para la ejecución del instalador web y el uso del MT Symfony2.

Para ejecutarlo debe habérsele otorgado los permisos de ejecución necesarios al script. Luego se debe ubicar la consola en la carpeta donde se encuentra el archivo instalador.sh (el directorio raíz de Symfony) y ejecutarlo de la siguiente manera: ./instalador.sh o bash instalador.sh. Si la consola se encuentra en otra ubicación o carpeta y se trata de ejecutar el script bash desde allí este mostrará un error informando que se debe estar ubicado en la carpeta raíz del proyecto para ejecutarlo. El script además chequea que todos los archivos del instalador se encuentran en dicha ubicación y que el usuario que lo ejecuta tiene permisos de root (administrador del sistema Unix). En caso contrario el script muestra el mensaje correspondiente y finaliza.

Al ejecutar el instalador bash este presenta un menú para que el usuario elija que tipo de

configuración desea (Figura 25). Se recomienda elegir la opción “Desinstalar completamente, instalar y configurar” la cual desinstalará y eliminará los archivos de configuración de Apache, PostgreSQL y PHP que puedan causar conflictos con las configuraciones que propone el script bash.

En caso de que el usuario ya posea configuraciones en estos archivos y no quiera perderlas puede elegir la opción 2 que solamente realizará los cambios necesarios para el uso de Symfony2 y el del instalador web. Esta opción no es recomendada, ya que si se había realizado alguna modificación a una propiedad que el script deba utilizar en un archivo de configuración este no podrá reconocerlo y por tanto no podrá configurar esa propiedad como lo requiere Symfony2. Por tanto en caso de elegirse la opción 2 y al culminar su ejecución el instalador bash el instalador web o Symfony2 presenten problemas se recomienda volver a ejecutar el archivo instalador.sh pero esta vez eligiendo la opción 1.

Cuando el script finaliza ejecuta el navegador web firefox con la url que configuró para el instalador web presentando la página de bienvenida, donde informa que la instalación de Symfony2 a culminado con éxito y permite ver el archivo config.php (propio de Symfony2) el cual informa sobre las recomendaciones específicas del servidor donde fue instalado.

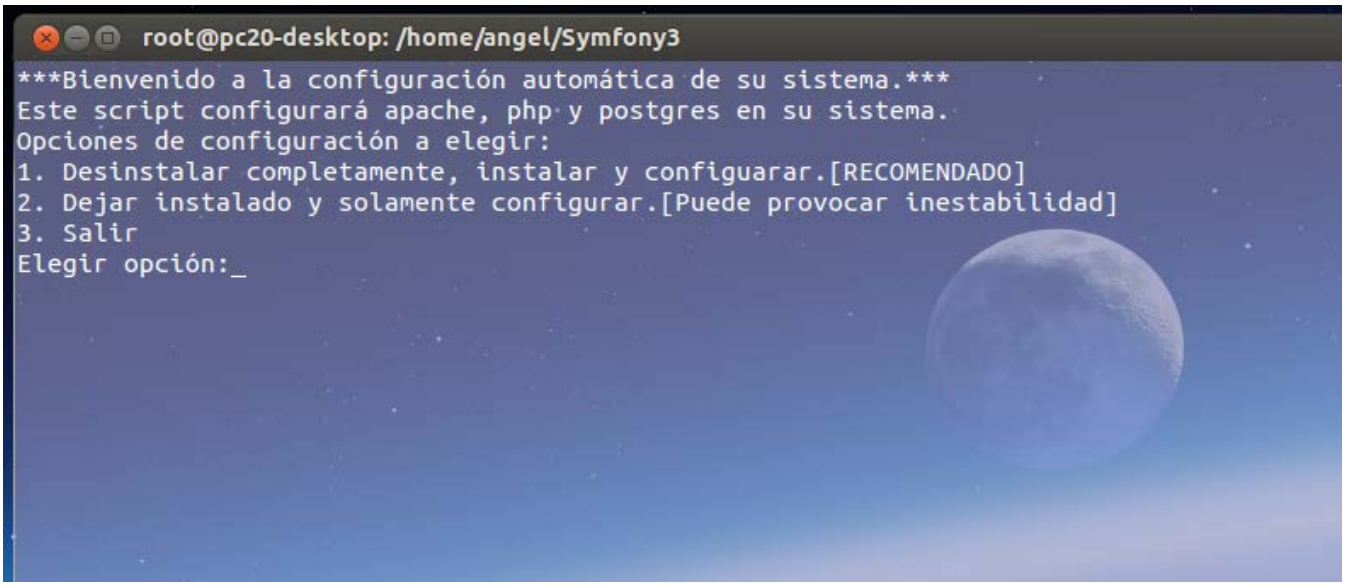


Figura 25: Menú del instalador bash

Al presionar el botón **Siguiente** se re direcciona hacia la página encargada de obtener los datos para la configuración de la conexión a la base de datos y el nombre y el nombre de la base de

datos a crear (Ver Figura 26). Al presionar **Siguiente** el sistema realiza la conexión con los parámetros suministrados y crea una base de datos con el nombre introducido, en caso de error se le informa al usuario. Al crearse la base de datos se muestra un mensaje informando que la base de datos ha sido creada y se re direcciona para la siguiente página correspondiente a la instalación de los subsistemas.

Figura 26: Formulario para configurar base de datos

En esta página el instalador muestra una lista con los subsistemas disponibles a instalar brindándole la opción al usuario de elegir los que desea instalar (Figura 27). Se puede presionar el botón **Finalizar** con el cual se realizan las operaciones y en caso de no haber error el sistema finaliza su ejecución o se puede elegir presionar **Instalar**, con el cual al terminar de instalar los subsistemas elegidos permite, volver a seleccionar. El sistema ejecuta los scripts necesarios de acuerdo a los subsistemas elegidos y muestra los mensajes correspondientes al éxito o fallo de las operaciones.

El sistema se puede adaptar a cualquier estructura organizacional, cambiando la configuración del archivo subsistemasinstalados.xml, el cual contiene el nombre de los subsistemas, un identificador único para cada subsistema, el nombre de los scripts que deben ejecutarse para la creación de las tablas, *triggers* y relaciones correspondientes e información sobre su estado: si se encuentra instalado y si es de carácter obligatoria su instalación. De esa manera se pueden añadir o quitar subsistemas según la entidad que planee realizar el despliegue. El programa con la información almacenada en ese archivo se encarga de ejecutar los scripts necesarios y en caso de existir algún error en los scripts que le son suministrados lo informa convenientemente.

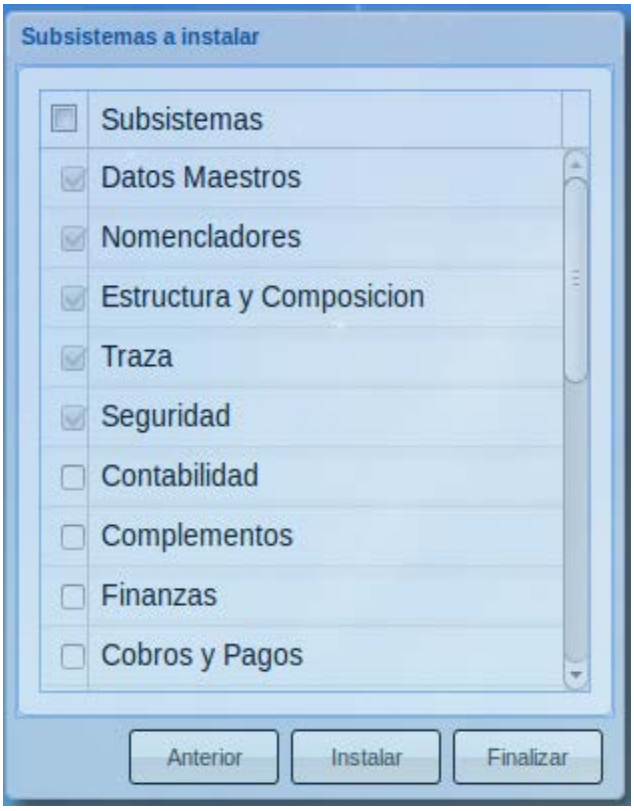


Figura 27: Formulario para elegir subsistemas a instalar

3.6 Pruebas

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Son un conjunto de situaciones o condiciones ante las cuales un programa debe responder satisfactoriamente para que pueda afirmarse que cumple con sus especificaciones u objetivos. Es una actividad más en el proceso de control de calidad. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas (30).

3.6.1 Pruebas de caja blanca

En las pruebas de caja blanca se comprueban los componentes internos. Comprueba los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones. Es un método de diseño que usa la estructura de control descrita como parte del

diseño al nivel de componentes para derivar los casos de prueba. Al realizar las pruebas de caja blanca se garantiza que:

- Se han ejercitado al menos una vez todas las rutas independientes dentro del módulo.
- Se ejecuten los lados verdadero y falso de todas las decisiones básicas.
- Se ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- Se ejecuten estructuras de datos internos para asegurar su validez (31).

Dentro de las de las pruebas de caja blanca se incluye la prueba del camino básico.

Prueba del camino básico

Este método permite que el diseñador de casos de prueba obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución (31).

A continuación se muestra el código del método crearBaseDatos perteneciente a la clase InstaladorController numerado:

```
protected function crearBaseDatos($servidor,$puerto,$basedatos,$usuario,$pass,$errores)
 {//-------------------------------------------------------------------------------------1
       try {//-------------------------------------------------------------------------2
        $conexion = new \PDO("pgsql:host=$servidor;port=$puerto", $usuario, $pass); //----2
    } catch (\Exception $e) {//-----------------------------------------------------------3
         array_push($errores,"<b style='background-color:#ff1500'> No se pudo establecer la conexion
         $conexion=null;//--------------------------------------------------------------4
        return array(1,$errores);//-----------------------------------------------------(9)
    }//-----------------------------------------------------------------------------------4
      $sql = "CREATE DATABASE $basedatos  WITH OWNER = postgres ENCODING = 'UTF8';";//----5
    $conexion->exec($sql);//--------------------------------------------------------------5
    $error = $conexion->errorInfo();//----------------------------------------------------6
    if (count($error[2]) != null){//------------------------------------------------------7
        array_push($errores,"<b style='background-color:#ff1500'> En estos momentos no se puede crear
         $conexion=null;//--------------------------------------------------------------8
        return array(1,$errores);//-----------------------------------------------------(9)
    }//-----------------------------------------------------------------------------------8
    return array(0,$errores);//-----------------------------------------------------------9
}
```

Figura 28: Código del método crearBaseDatos

A partir de este método se genera el siguiente gráfico de flujo:

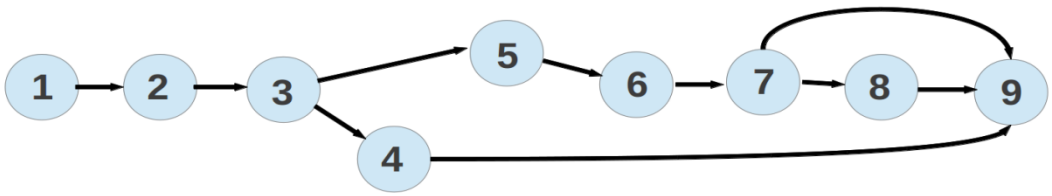


Figura 29: Gráfico de flujo correspondiente al método crearBaseDatos

Se calcula la complejidad ciclomática, la cual proporciona una medida cuantitativa de la complejidad lógica de un programa. Esta define el número de rutas independientes en el conjunto básico de un programa. Una ruta independiente es cualquier ruta de un programa que ingresa por lo menos un nuevo conjunto de instrucciones de procesamiento (31).

Cálculo de la complejidad ciclomática:

Fórmula 1:

A: cantidad total de aristas del grafo.

N: cantidad total de nodos del grafo.

V (G) = (A-N) + 2

V (G) = (10-9) + 2

V (G) = 3

Fórmula 2:

P: cantidad total de nodos predicados (nodos de los que parten dos o más aristas).

V (G) = P + 1

V (G) = 2 + 1

V (G) = 3

Fórmula 3:

R: cantidad total de regiones existentes en el grafo, se incluye el área exterior del grafo, como una región más.

V (G) = R

V (G) = 3

En el código presentado se generan las siguientes rutas independientes:

Tabla 8: Caminos básicos generados

Número	Camino Básico
1	1-2-3-4-9
2	1-2-3-5-6-7-9
3	1-2-3-5-6-7-8-9

Casos de prueba para las rutas independientes halladas:

Tabla 9: Caso de prueba para el camino básico #1

Ruta #1: 1-2-3-4-9	
Descripción	Ocurre un error conectándose a la base de datos.
Entrada	Datos para la conexión con la base de datos ($servidor, $puerto, $usuario, $contraseña, $basedatos) y un array vacío ($errores).
Resultado esperado	Se retorna un array con el valor 1(error) y el array $errores con el mensaje: “No se pudo establecer la conexión con el servidor de datos” y la respectiva excepción lanzada.
Resultado	Satisfactorio.

Tabla 10: Caso de prueba para el camino básico #2

Ruta #2: 1-2-3-5-6-7-9	
Descripción	Es creada la base de datos.
Entrada	Datos para la conexión con la base de datos ($servidor, $puerto, $usuario, $contraseña, $basedatos) y un array vacío ($errores).
Resultado esperado	Se retorna un array con el valor 0 (éxito) y el array $errores vacío.
Resultado	Satisfactorio.

Tabla 11: Caso de prueba para el camino básico #3

Ruta #3: 1-2-3-5-6-7-8-9	
Descripción	Ocurre un error creando la base de datos.
Entrada	Datos para la conexión con la base de datos ($servidor, $puerto, $usuario, $contraseña, $basedatos) y un array vacío ($errores).
Resultado esperado	Se retorna un array con el valor 1(error) y el array $errores con el mensaje: “En estos momentos no se puede crear la base de datos” y la respectiva excepción lanzada.
Resultado	Satisfactorio.

3.6.2 Pruebas funcionales

Las pruebas funcionales se centran en los requisitos funcionales llevándose a cabo sobre la interfaz del software. En estas pruebas se ejecuta cada caso de uso o función usando datos válidos e inválidos para verificar el comportamiento de la aplicación: que se obtengan los resultados esperados cuando se inserten datos correctos y que sean desplegados los mensajes de error correspondientes cuando se usen los datos inválidos (32).

Con estas se pretende capturar errores de interfaz, de estructuras de datos o acceso a bases de datos externas, funciones incorrectas o errores de rendimiento. Para la realización de estas pruebas es necesario realizar un correcto diseño de casos de prueba.

Casos de Pruebas

En la ingeniería del software, los casos de prueba son un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito a menos que un requisito tenga requisitos secundarios. En ese caso, cada requisito secundario deberá tener por lo menos un caso de prueba (33).

Se crearon los documentos de diseño de casos de prueba para cada requisito funcional. A continuación se muestra el diseño de casos de prueba para el requisito funcional crear base de datos a partir de la configuración definida.

Condiciones de ejecución: Se debe poseer un usuario y contraseña con permisos para realizar operaciones en el servidor de base de datos.

Tabla 12: Descripción de casos de prueba para el requisito funcional crear base de datos a partir de la configuración definida.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Crear base de datos introduciendo datos válidos.	Se introducen los parámetros para la configuración de la base de datos y el sistema crea la base de datos con el nombre proporcionado.	Se creó la base de datos y se configuró el archivo parameters.yml.	Se introducen los datos que pide la interfaz. El usuario presiona el botón **Siguiente**. El sistema valida los datos introducidos. El sistema comprueba que los datos introducidos son correctos para la conexión con la base de datos. Si los datos son correctos el sistema los registra. El sistema realiza la conexión al servidor de base de datos. El sistema crea una nueva base de datos con el nombre suministrado. Se registran los parámetros en el archivo parameters.yml. El sistema muestra el mensaje: “La base de datos ha sido creada satisfactoriamente.” El sistema redirige hacia la siguiente página del instalador. Concluye el requisito.
EC 1.2 Crear base de datos introduciendo datos inválidos.	Se introducen varios datos inválidos y el sistema los señala.	El Sistema señala los campos inválidos.	Se introducen los parámetros de configuración de la base de datos. El usuario presiona el botón **Siguiente**. El sistema valida los datos introducidos. El sistema señala los campos con datos incorrectos.
EC 1.3 No se introducen todos los datos.	Se dejan campos vacíos y el sistema los señala.	El Sistema muestra un mensaje de error y señala los campos vacíos.	Se introducen los parámetros de configuración de la base de datos. El usuario presiona el botón **Siguiente**. El sistema valida los datos introducidos. El sistema muestra el mensaje: “Debe llenar todos los campos.” y señala los campos vacíos.
EC 1.4 Se introducen datos incorrectos para la conexión con la base de datos.	Se introducen datos válidos pero incorrectos para la conexión con la base de datos y el sistema muestra los errores.	El Sistema muestra los errores y las excepciones capturadas.	Se introducen los parámetros de configuración de la base de datos. El usuario presiona el botón **Siguiente**. El sistema valida los datos introducidos. El sistema muestra un mensaje de error informando cuál es el parámetro de conexión erróneo.
EC 1.5 Se presiona el botón **Limpiar**.	Se presiona el botón Limpiar y el sistema limpia los campos.	El Sistema muestra los errores y las excepciones capturadas.	Se presiona el botón Limpiar. Se limpia el contenido de todos los campos.

Descripción de las variables a utilizar en el caso de prueba:

Tabla 13: Descripción de las variables para el requisito funcional crear base de datos a partir de la configuración definida

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Base de datos	Campo de texto(20)	No	Letras y Números
2	Servidor	Campo de texto(20)	No	Letras y Números
3	Puerto	Campo de texto(20)	No	Números
4	Usuario	Campo de texto(20)	No	Letras y Números
5	Contraseña	Campo de texto(20)	No	Letras y Números
6	Centro de datos	Campo de texto(20)	No	Números

A continuación se muestran un juego de datos que se utilizará para probar los escenarios capturados.

Tabla 14: Juego de datos a probar para el requisito funcional crear base de datos a partir de la configuración definida

Escenario	Base de Datos	Servidor	Puerto	Usuario	Contraseña	Centro de datos
EC 1.1 Crear base de datos introduciendo datos válidos.	V	V	V	V	V	V
	nombreBD	localhost	5432	postgres	postgres	235
EC 1.2Crear base de datos introduciendo datos inválidos.	I	V	V	V	V	I
	123Symfony	localhost	5432	postgres	postgres	centro
EC 1.3 No se introducen todos los datos.	I	V	V	V	V	V
	NA	localhost	5432	postgres	postgres	263
EC 1.4 Se introducen datos incorrectos para la conexión con la base de datos.	V	I	V	V	V	V
	Symfony	localhast	5432	postgres	postgres	963
EC 1.5 Se presiona el botón **Limpiar**.	V	V	V	V	V	V
	NA	NA	NA	NA	NA	NA

Resultado de las pruebas funcionales

Se realizaron tres iteraciones de las pruebas funcionales utilizando los diseños de casos de prueba generados detectándose varias no conformidades las cuales fueron clasificadas en errores de aplicación y errores ortográficos. Durante la primera iteración se hallaron 8 no conformidades, 6 referidas a las funcionalidades y resultados de las operaciones y 2 problemas con la ortografía en los mensajes de la aplicación. Durante la segunda se encontraron 2 no conformidades referentes a la aplicación, siendo resueltas todas para la tercera iteración (ver Figura 30).

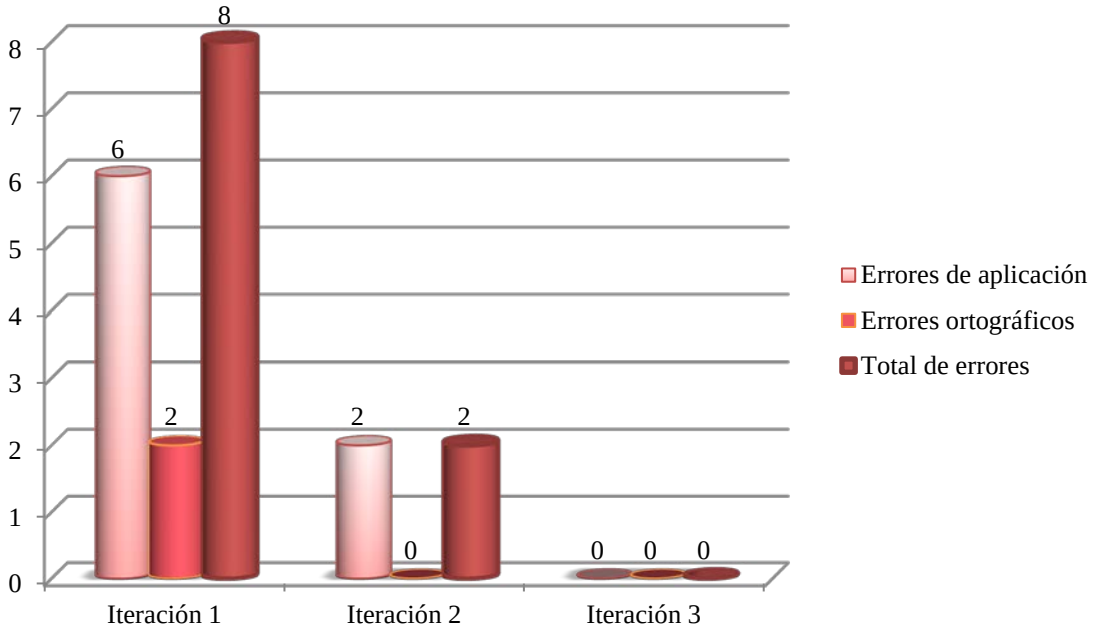


Figura 30: Resultados de la aplicación de las pruebas de caja negra

3.7 Validación de la solución

Con el objetivo de validar la solución se realizó un pre-experimento. Se eligió un equipo de desarrollo con el objetivo de realizar la instalación de los subsistemas Acaxia y Estructura y Composición de forma manual y luego utilizando el instalador desarrollado. Los subsistemas elegidos poseen los scripts necesarios para la creación de sus bases de datos y ya cuentan con versiones implementadas para Symfony2.

La validación se separó en dos procesos independientes, uno de los cuales medirá la instalación y configuración de tecnologías para el uso de Symfony2 y el otro la configuración de la base de

datos e instalación de los subsistemas. Para estimar cuantitativamente los resultados se utilizan los indicadores cantidad de errores; que viene dado por el número de errores que ocurrieron durante la ejecución de los procesos antes mencionados, y el tiempo utilizado para la realización de todas las actividades, dado en minutos.

3.7.1 Descripción del pre-experimento

A continuación se describen los pasos realizados para cada proceso:

Proceso 1. Instalación y configuración de las tecnologías para el correcto despliegue de Symfony2.

De manera manual:

Se debe haber lanzado la terminal de comandos y haberse autenticado como usuario root.

Configuraciones obligatorias (configuraciones completamente necesarias):

Paso 1: Se instala Apache2.

apt-get install apache2

Paso 2: Instalar PHP con las extensiones necesarias.

apt-get install php5 php-cli php5-gd php5-ldap php-soap php5-xsl php5-pgsql php5-xdebug php5-intl php5-json php5-cli php5-sqlite php5-mcrypt

/etc/init.d/apache2 restart

Paso 3: Instalar y habilitar modrewrite, que permite la sobre-escritura de url y es obligatorio para utilizar Symfony2.

apt-get install libapache2-mod-proxy-html

a2enmod rewrite

/etc/init.d/apache2 restart

Paso 4: En caso de haberse instalado, desinstalar el módulo libapache2-mod-php5filter, el cual impide la utilización de la barra de herramientas de depuración web.

apt-get remove libapache2-mod-php5filter

/etc/init.d/apache2 restart

Paso 5: Realizar cambios en los ficheros php.ini

sudo gedit /etc/php5/apache2/php.ini

Se busca la línea que dice: ;date.timezone = y se reemplaza por date.timezone = America/Havana.

Se realizan las mismas configuraciones para el archivo php.ini correspondiente a la línea de comandos.

sudo gedit /etc/php5/cli/php.ini

sudo /etc/init.d/apache2 restart

Paso 6: Se configura un VirtualHost.

touch /etc/apache2/sites-available/nombre

sudo gedit /etc/apache2/sites-available/nombre

Una vez abierto el archivo nombre se procede a introducir la siguiente configuración del VirtualHost optimizada para el uso de Symfony2 introduciendo en %puerto el valor del puerto a utilizar y en %ruta la ruta hasta el directorio web del proyecto Symfony a utilizar.

NameVirtualHost *:%puerto

Listen %puerto

<VirtualHost *:%puerto >

ServerName localhost

DocumentRoot "%ruta "

DirectoryIndex app_dev.php

<Directory "%ruta">

AllowOverride None

Allow from All

<IfModule mod_rewrite.c>

Options -MultiViews

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f

RewriteRule ^(.*)$ app.php [QSA,L]

</IfModule>

</Directory>

KeepAlive On

MaxKeepAliveRequests 200

KeepAliveTimeout 5

</VirtualHost>

Guardar el archivo y crear el enlace simbólico que habilita el uso del VirtualHost

ln -fs /etc/apache2/sites-available/nombre /etc/apache2/sites-enabled

Reiniciar apache.

/etc/init.d/apache2 restart

Paso 7: Le otorgo los permisos a los ficheros del proyecto.

sudo chown www-data:www-data ruta/hasta/directorio/Symfony -R

sudo chmod 777 ruta/hasta/directorio/Symfony –R

Paso 8: Instalar PostgreSQL

apt-get install postgresql-9.1

Paso 9: Configurar los el archivo pg_hba.conf

sudo gedit /etc/postgresql/9.1/main/pg_hba.conf

Anadir la línea “host all all 10.0.0.0/8 md5” a la configuración de las conexiones locales.

Paso 10: Configurar el archivo postgresql.conf

sudo gedit /etc/postgresql/9.1/main/postgresql.conf

Sustituir la línea #listen_addresses= ‘ localhost ’ por listen_addresses= ’ * ’.

Paso 11: Cambiarle la contraseña al usuario postgres.

su postgres

psql

ALTER ROLE postgres with password ‘postgres’;

\q

exit

Paso 12: Otorgar permisos a los directorios app/cache y app/logs.

apt-get install acl

sudo setfacl -R -m u:root:rwx -m u:`whoami`:rwx app/cache app/logs

sudo setfacl -dR -m u:root:rwx -m u:`whoami`:rwx app/cache app/logs

Paso 13: Crear los enlaces a la carpeta **web/bundles** para poder visualizar el contenido de la carpeta **public** de los bundles.

php app/console assets:install --symlink

Configuraciones recomendadas (configuraciones para optimizar el uso de Symfony2):

Paso 14: Instalar el acelerador de PHP APC:

sudo apt-get install php-apc

sudo gedit /etc/php5/apache2/php.ini

Y se copia lo siguiente al final del archivo.

[APC]

extension=apc.so

apc.apc.stat = 0

apc.include_once_override = 1

apc.shm_size = 64

Se guarda y se reinicia apache2.

sudo /etc/init.d/apache2 restart

Paso 15: Cambios en los ficheros php.ini para optimizar el uso de Symfony2:

sudo gedit /etc/php5/apache2/php.ini

Se cambian las siguientes líneas.

short_open_tag = On por short_open_tag = Off.

magic_quotes_gpc = On por magic_quotes_gpc = Off.

register_globals = On por register_globals = Off.

session.autostart = On por session.autostart = Off.

max_input_nesting_level = (valor) por max_input_nesting_level = 250.

Realizar los mismos cambios en el php.ini correspondiente a la línea de comandos.

sudo gedit /etc/php5/apache2/php.ini

sudo /etc/init.d/apache2 restart

Paso 16: Se cambia el valor de xdebug.max_nesting_level

sudo gedit /etc/php5/cli/conf.d/xdebug.ini

Y se copia la línea xdebug.max_nesting_level=250.

Utilizando el instalador:

Se debe haber lanzado la terminal de comandos y haberse autenticado como usuario root.

Paso 1: Ubicar la terminal en la carpeta del instalador bash.

cd ruta/hasta/carpeta/instalador

Paso 2: Ejecutar el instalador bash.

./instalador.sh

Paso 3: Elegir la opción a ejecutar.

Se escribe 1, 2 ó 3 de acuerdo a la operación que se desee realizar. En este caso 1.

Paso 4: Cambiarle la contraseña al usuario postgres.

su postgres
psql
ALTER ROLE postgres with password ‘postgres’;
\q
exit

Proceso 2. Configuración de la conexión a la base de datos, creación de la base de datos e instalación de los subsistemas.

De forma manual:

Se debe tener instalada la herramienta pgAdmin.

Paso 1: Crear una conexión.

Se busca el menú File->Add Server, y luego se introducen los valores para el nombre de la conexión, el servidor, el puerto, el usuario y la contraseña.

Paso 2: Crear la base de datos.

Usando el pgAdmin se presiona el botón secundario del mouse sobre 'Databases' y seleccionando la opción New Database, luego introducir el nombre en la ventana que se muestra.

Paso 3: Instalar los subsistemas.

Vía 1: Accediendo en el pgAdmin en la barra de menú a Tools->Query tool y realizando las consultas de forma manual se siguen los siguientes pasos:

Paso 3.1: Crear los esquemas correspondientes.

Paso 3.2: Se crean las tablas.

Paso 3.3: Se crean las secuencias.

Paso 3.4: Se crean las llaves primarias.

Paso 3.5: Se crean las funciones.

Paso 3.6: Se crean los *triggers*.

Paso 3.7: Se insertan los datos en las tablas.

Paso 3.8: Se crean los roles.

Paso 3.9: Se asignan los permisos.

Vía 2: Teniendo los scripts y conociendo el orden en el que se deben ejecutar se accede en el pgAdmin a la barra de herramientas, en Tools->Query tool se abre el Query Editor. En este se accede en la barra de Menú a File->Open, y se busca la ubicación del script correspondiente seleccionándolo y una vez cargado se ejecuta. El orden de los scripts es el siguiente:

Paso 3.1: Scripts de estructura.

Paso 3.2: Scripts de datos.

Paso 3.3: Scripts de permisos.

Debido a la alta complejidad de la vía 1 se eligió par realizar la vía 2 para comparar sus resultados con los del instalador.

Utilizando el instalador:

Paso 1: Crear la base de datos.

En la interfaz se introducen los parámetros de conexión servidor, puerto, usuario y contraseña, se introduce el nombre de la base de datos a crear. Se presiona Siguiente.

Paso 2: Instalar los subsistemas.

Se seleccionan en la interfaz los subsistemas a instalar. Se presiona Instalar.

3.7.2 Resultados del pre-experimento

Se realizó ambos procesos en una máquina con las siguientes prestaciones: Memoria RAM: 975,4 MiB, Procesador: Intel® Core™2 Duo CPU E4500 @ 2.20GHz × 2 con el sistema operativo Ubuntu Versión 12.04 (precise) de 32-bit. Se utilizaron las vías propuestas obteniéndose los siguientes resultados de acuerdo a los indicadores analizados:

Las pruebas fueron realizadas por un equipo de desarrollo con cierta habilidad, conocimientos y experiencia utilizando estas tecnologías e instalando Symfony2.

Descripción del resultado del pre-experimento

Proceso 1:

Durante la instalación de forma manual:

Se realizaron uno a uno todos los pasos antes descritos ocurriendo los algunos errores por las siguientes causas:

Olvido de pasos: se olvidó crear el enlace simbólico al fichero con el VirtualHost configurado por lo cual este no se activó. Se olvidó reiniciar el servicio apache después de realizar algunos cambios por lo cual estos no surtieron efecto. Se realizaron los cambios al fichero php.ini que utiliza el servidor web, sin embargo se olvidó hacerlos en el fichero que usa la línea de comandos. Esto se detectó al ejecutar desde la terminal el siguiente comando php app/check.php, el cual es propio del

MT Symfony2 y comprueba que se cumplen todos los requisitos para el uso de la terminal de comandos para PHP.

Errores configurando: en el VirtualHost configurado la propiedad Listen debía tener el valor del puerto que se utilizaba para el sitio, sin embargo tenía otro valor por lo cual apache no reconocía el puerto. Además se eligió inicialmente un puerto que estaba siendo utilizado por otro VirtualHost por lo cual apache mostraba un error informando que el puerto estaba actualmente en uso.

El proceso demoró 33 minutos empleando tiempo en la identificación y corrección de los errores cometidos.

Durante la instalación usando el instalador bash:

Se realizaron los pasos descritos anteriormente obteniéndose un resultado satisfactorio sin errores en 3 minutos aproximadamente.

Proceso 2:

Durante la creación de la base de datos de forma manual:

Realizando la vía 2 ocurrió un error al alterar el orden de uno de los scripts. Luego se ejecutó el script necesario y todo ocurrió con normalidad hasta obtener un resultado satisfactorio en 6 minutos.

Durante la creación de la base de datos utilizando el instalador web:

Se realizaron los pasos descritos creando la base de datos sin contratiempos en 20 segundos.

A continuación se muestra una tabla con los resultados obtenidos:

Tabla 15: Resultados del pre-experimento.

Indicadores		**Cantidad de errores**	**Tiempo necesario**
Proceso 1	Manualmente	5	33 minutos
	Instalador	0	3 minutos
Proceso 2	Manualmente	1	6 minutos
	Instalador	0	20 segundos

Con estos resultados se pudo corroborar que el instalador para soluciones desarrolladas empleando el MT Symfony2 aporta seguridad y rapidez a los procesos en los que interviene como se muestra en las Figuras 31 y 32.

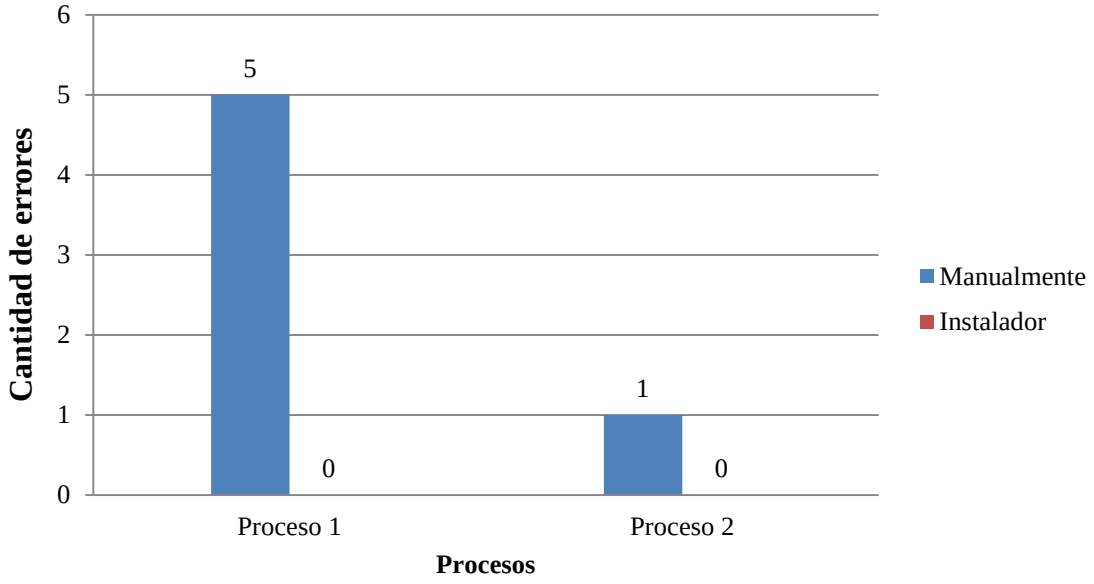


Figura 31: Resultados para el indicador cantidad de errores.

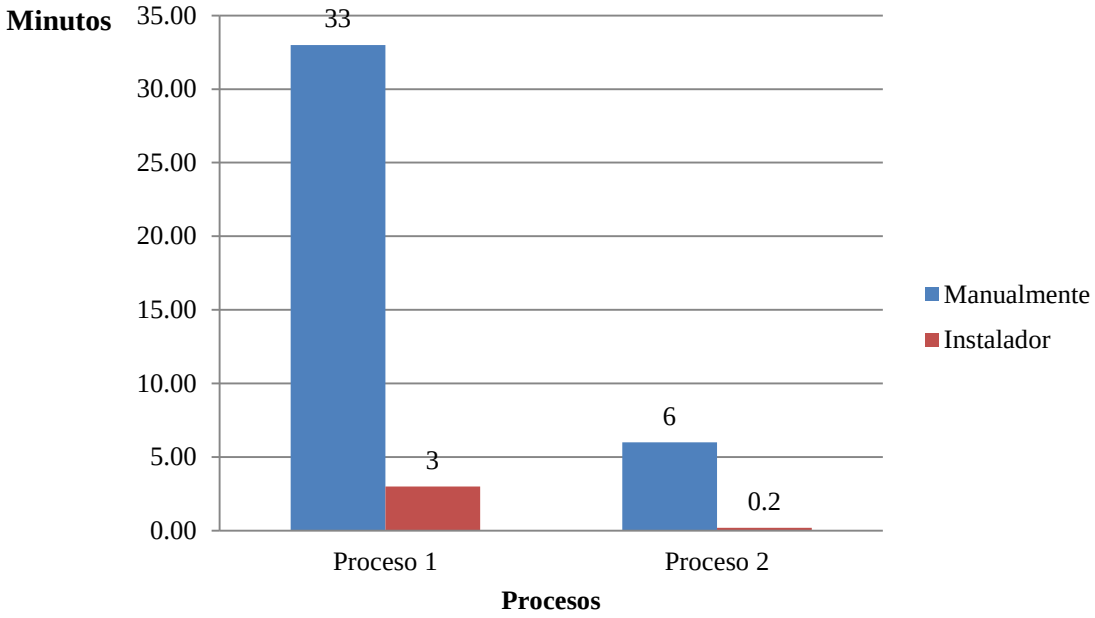


Figura 32: Resultados para el indicador tiempo utilizado.

3.8 Validación de la herramienta

Para la validación de la herramienta se tuvieron en cuenta los siguientes aspectos: el alcance de los requisitos y la aceptación del cliente.

Alcance de los requisitos: se identificaron 3 requisitos funcionales, implementándose el 100%.

Aceptación del cliente: para validar que se cumplieron todas las especificaciones planteadas por el cliente se creó una comisión con el objetivo de realizarle una revisión técnica a la solución creada. Esta determinó que el instalador cumple con el objetivo para el cual fue creado y está listo para su uso. La comisión emitió el acta de liberación correspondiente.

3.9 Conclusiones del capítulo

- El uso de estándares de codificación permitió la construcción de un código fuente más legible y de fácil mantenimiento.
- Las pruebas de caja blanca y caja negra realizadas al producto permitieron detectar deficiencias en la implementación que fueron corregidas garantizando la calidad del producto final.
- La realización del pre experimento permitió constatar que la solución mejora el proceso de despliegue de las aplicaciones desarrolladas empleando el MT Symfony2 teniendo en cuenta los indicadores de cantidad de errores y tiempo quedando demostrado de esta forma el cumplimiento del objetivo de la investigación.

Conclusiones generales

Luego de culminada la realización de este trabajo se arribaron a las siguientes conclusiones:

- El estudio realizado sobre los instaladores web existentes arrojó como resultado que ninguno daba solución al problema planteado pero si contaban con fortalezas que sirvieron de punto de partida para la realización de esta investigación.
- El análisis y diseño del instalador web permitió la construcción de los artefactos necesarios para lograr una correcta comprensión de las funcionalidades que debía incluir el instalador así como la colaboración entre las diferentes clases de la solución. Tanto el análisis como el diseño fueron validados a través de técnicas y métricas que aseguraron su calidad.
- La aplicación de estándares de codificación durante la implementación aseguró la obtención de un código fuente legible y de fácil mantenimiento.
- La aplicación de pruebas de caja blanca y caja negra permitió validar desde el punto de vista funcional la calidad de la solución desarrollada.
- La realización del pre-experimento permitió demostrar la mejoría en el proceso de despliegue de las soluciones desarrolladas sobre el marco de trabajo Symfony2 a partir de la evaluación de los indicadores cantidad de errores y tiempo.

Recomendaciones

Con el fin de enriquecer la solución propuesta se sugieren las siguientes recomendaciones:

- Hacer encuestas a los desarrolladores que lo utilizan preguntándoles principalmente qué otras funcionalidades podrían añadírsele.
- Realizar un seguimiento de la evolución del MT Symfony2 para incorporar nuevas herramientas que aporten mayor eficiencia al instalador.

Bibliografía

1. **Gonzáles, Rolando Alfredo Hernández y Sayda Coello.** *El Proceso de Investigación Científica.* s.l. : Editorial Universitaria, 2011.

2. **Abel, Peter.** *Lenguaje Ensamblador y Programación para PC IBM y compatibles.* ISBN: 986-880-708-7.

3. **Garrels, Machtelt.** *Bash Guide for Beginners, Second Ed.* s.l. : Fultus Corporation, 2010.

4. Kioskea. [En línea] [Citado el: 10 de 12 de 2013.] httpÑ//es.kioskea.net/faq/2639-instalacion-de -mantis-bug-traker..

5. Joomla!, la web en entornos educativos. [En línea] [Citado el: 11 de 12 de 2013.] http>//www.ite.educacion.es/formacion/materiales/99/cd/mod_02/instalacion_de_joomla_instalacin_web.html.

6. **Velázquez, Ricardo Pérez.** *Instalador WEB para el Sistema Integral de Gestión Cedrux.* Ciudad de La Habana : s.n., 2010.

7. **Alemañy, Julio César Cuscó.** *Desarrollo del instalador para el Sistema de Gestión Integral Aduanera 1.0.* La Habana : s.n., 2013.

8. **Obregón, William Gonzáles.** *CEIGE-Modelo de Desarrollo de Software V1.2.* 2013.

9. Symfony para Todos. [En línea] [Citado el: 02 de 02 de 2014.] http://symfony.cubava.cu/introduccion/symfony-2/..

10. EcuRed. [En línea] [Citado el: 16 de 02 de 2014.] http://www.ecured.cu/index.php/Doctrine..

11. CodeManía. [En línea] [Citado el: 02 de 02 de 2014.] http://codemania.cubava.cu/2013/09/que-es-extjs/..

12. eHow. [En línea] [Citado el: 16 de 02 de 2014.] http://www.ehowenespanol.com/funciona-servidor-web-apache-como_92268/..

13. PostgreSQL-es. [En línea] [Citado el: 22 de 02 de 2014.] http://www.postgresql.org.es/sobre_postgresql..

14. MDN Mozilla Developer Network. [En línea] [Citado el: 16 de 02 de 2014.] https://developer.mozilla.org/es/docs/User:Marti1125..

15. EcuRed. [En línea] [Citado el: 22 de 02 de 2014.]

www.ecured.cu/index.php/JavaScript,www.inforibera.net/smxpace/javascript.html..

16. Internet. [En línea] [Citado el: 02 de 02 de 2014.] http://mdai.cep.edu.uy/internet.html.

17. **Infante, Daniel.** Geocities. [En línea] [Citado el: 22 de 02 de 2014.] www.geocities.ws/daniel.infante/fase2/t1.html..

18. **Larman, Craig.** *UML y Patrones, Introducción al análisis y diseño orientado a objetos.* s.l. : Prentice Hall.

19. Cyclopaedia.net. [En línea] [Citado el: 25 de 02 de 2014.] http://es.cyclopaedia.net/wiki/Modelo_de_dominio..

20. Alegsa.com.ar. [En línea] 10 de 04 de 2014. http://www.alegsa.com.ar/Dic/instalar.php.

21. Buenas Tareas. [En línea] [Citado el: 20 de 04 de 2014.] http://www.buenastareas.com/materias/herramientas-para-pulir-un-manuscrito/0.

22. Adictos al Trabajo. [En línea] [Citado el: 12 de 04 de 2014.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=RM#_Toc260742341.

23. SoftQaNetwork. [En línea] [Citado el: 26 de 03 de 2014.] http://www.softqanetwork.com/requisitos-no-funcionales-nfr.

24. DocIRS. [En línea] [Citado el: 05 de 03 de 2014.] www.docirs.cl/uml.htm.

25. Altova. [En línea] [Citado el: 12 de 04 de 2014.] http://www.altova.com/es/umodel/sequence-diagrams.html.

26. **Baryolo, Oiner Gómez.** *Solución informática de autorización en entornos multientidad y multisistema.* La Habana : s.n., 2010.

27. **Doria, Heidi González.** *Las Métricas de Software y su Uso en la Región.* México : Derechos Reservados ©2001., 2001.

28. Developer Network. [En línea] [Citado el: 02 de 03 de 2014.] http://msdn.microsoft.com/es-es/library/dd409390.aspx..

29. Developer Network. [En línea] [Citado el: 11 de 4 de 2014.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

30. prezi. [En línea] [Citado el: 03 de 04 de 2014.] http://prezi.com/fdkxfkwcrb5o/copy-of-mapa-conceptual-de-tecnicas-de-pruebas-y-mantenimiento-de-software/.

31. **Pressman, Roger S.** *Ingeniería del software, Un enfoque práctico.*

32. **Peña, Yadira Machado.** *Planificación, Diseño y Ejecución de Pruebas Funcionales.* Habana : s.n., 2011.

33. logysof. [En línea] [Citado el: 15 de 03 de 2014.] http://logysoft.wordpress.com/type/aside/..

34. **Rodríguez, Lisandra Vega.** *Propuesta del Plan para la verificación y validación de requisitos en el proyecto ERP-Cuba.* La Habana : s.n., 2010.

35. EcuRed. [En línea] [Citado el: 02 de 02 de 2014.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software#Introducci.C3.B3n_a_las_Metodolog.C3.ADas_.C3.81giles..

36. Microsoft. [En línea] [Citado el: 11 de 12 de 2013.] http://www.microsoft.com/es-mx/download/details.aspx?id=35.

37. Librosweb. [En línea] [Citado el: 11 de 05 de 2014.] http://librosweb.es/composer/capitulo_1.html.

38. EcuRed. [En línea] [Citado el: 20 de 01 de 2014.] www.ecured.cu/index.php/Validación_de_Requisitos.

Anexos

Anexo 1: Descripción del requisito funcional Instalar y configurar tecnologías para arquitectura de referencia utilizando Symfony2

Precondiciones	Han sido copiadas en la PC todas las carpetas y archivos del instalador y el script bash instalador.sh.
Flujo de eventos	
Flujo básico Instalar y configurar tecnologías para arquitectura de referencia utilizando Symfony2	
11	Se ejecuta el script desde la terminal.
12	El script comprueba que fue ejecutado por un usuario con permisos de root.
13	El script comprueba que fue ejecutado desde la carpeta raíz del proyecto Symfony.
14	El script muestra un menú con las opciones de configuración.
15	El usuario elige la opción 1.
16	Para mayor seguridad el script elimina las tecnologías con sus ficheros de configuración.
17	El script instala las tecnologías necesarias: Apache2, PostgreSQL y PHP-5.
18	El script configura las tecnologías necesarias: Symfony2, Apache2, PostgreSQL y PHP-5.
19	El script ejecuta el navegador con la página de bienvenida al instalador web.
20	Concluye el requisito.
Pos-condiciones	
3	Fueron instaladas y configuradas las tecnologías requeridas para la arquitectura de referencia utilizando Symfony2.
4	Queda abierto el navegador web firefox con la url del instalador web.
Flujos alternativos	
Flujo alternativo 2.a Usuario sin permisos de root	
1	El script muestra el mensaje "Este script debe ser ejecutado por un usuario con permisos de root"
2	El script finaliza su ejecución.
3	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	NA
Flujo alternativo 3.a Ejecutado desde otra carpeta	
1	El script muestra el mensaje "No se puede ejecutar el script desde esta carpeta. Faltan archivos del instalador o no se encuentra en la carpeta raíz del proyecto".

2	El script finaliza su ejecución.	
3	Volver al paso 1 del flujo básico.	
Pos-condiciones		
2	NA	
Flujo alternativo 5.a Se elige la opción 2		
3	Ir al paso 7 del flujo básico.	
Pos-condiciones		
1	NA	
Flujo Alternativo 5.b Se elige la opción 3		
1	El script finaliza su ejecución.	
Pos-condiciones		
1	NA	
Validaciones		
2	NA	
Conceptos	**Instalador bash**	Visibles en la interfaz: NA Utilizados internamente: NA
Requisitos especiales	NA	
Asuntos pendientes	NA.	

Anexo 2: Descripción del requisito funcional Instalar Subsistemas.

Precondiciones	Debe haber sido creada la base de datos.
Flujo de eventos	
Flujo básico Instalar Subsistemas	
1	El sistema carga los subsistemas disponibles a instalar y los muestra en la interfaz marcando por defecto los subsistemas obligatorios e instalados. (Nomencladores, Datos maestros, Seguridad, Estructura y composición y Trazas)
2	El usuario selecciona los subsistemas a instalar.
3	El usuario presiona el botón Finalizar.
4	El sistema ejecuta los scripts correspondientes.

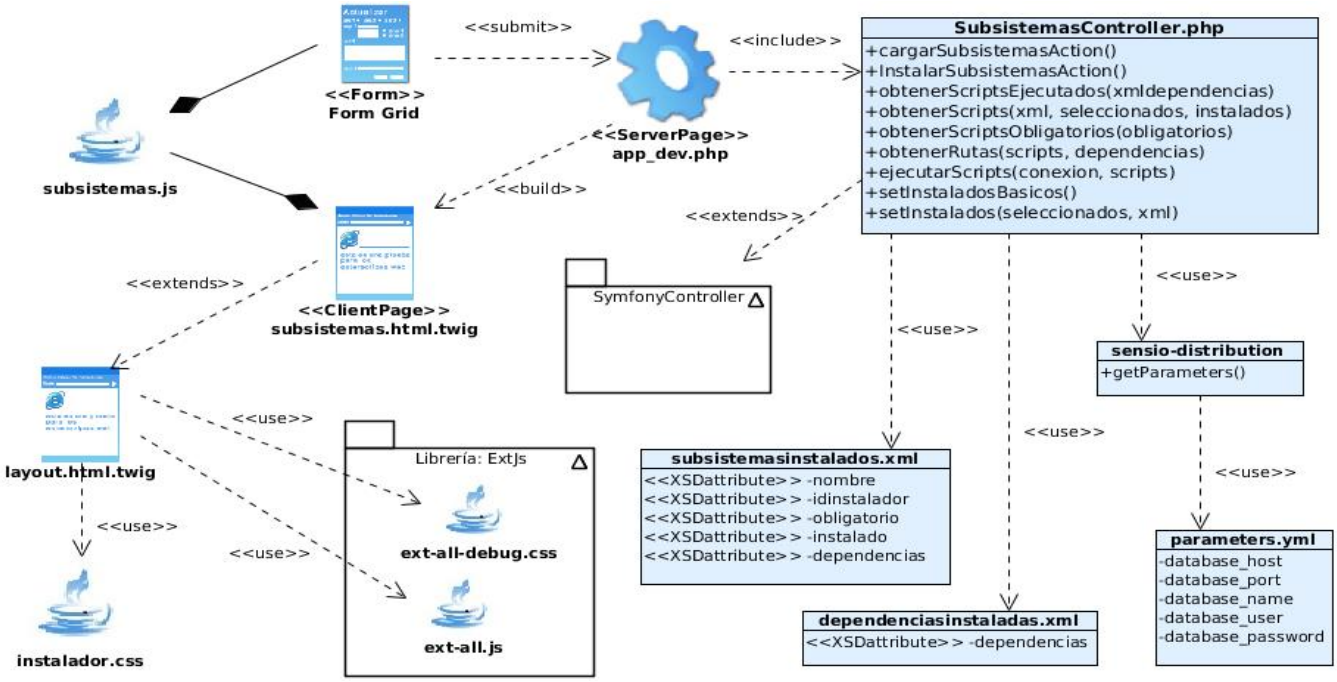
5	El sistema muestra un mensaje confirmando que se instalaron los subsistemas seleccionados.
6	El sistema finaliza la ejecución.
7	Concluye el requisito.
Pos-condiciones	
1	Se instalaron los subsistemas seleccionados.
Flujos alternativos	
Flujo alternativo 2.a No se ha selecciona ningún subsistema.	
1	Se presiona **Finalizar**.
2	Finaliza la ejecución del instalador
Pos-condiciones	
1	NA
Flujo alternativo 2.b No se ha selecciona ningún subsistema.	
1	Se presiona el botón **Instalar**.
2	El sistema muestra el mensaje “Debe seleccionar al menos un módulo”.
3	Volver al paso 2 del flujo básico.
Pos-condiciones	
1	NA
Flujo alternativo 3.b Se presiona el botón Instalar	
1	El sistema ejecuta los scripts correspondientes.
2	El sistema muestra un mensaje confirmando que se instalaron los subsistemas seleccionados.
3	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	Se instalaron los subsistemas seleccionados.
Flujo Alternativo *.a Se presiona el botón Anterior	
1	Se muestra un mensaje de confirmación con el siguiente texto: “Esta acción eliminará la base de datos creada. ¿Desea continuar?”.
2	El usuario selecciona Si.
3	Se prepara la conexión con la base de datos y se procede a la eliminación de la base

	de datos creada utilizando los parámetros registrados en el archivo parameters.yml.	
4	Se elimina la base de datos registrada.	
5	Se muestra el mensaje “La base de datos ha sido eliminada satisfactoriamente”.	
6	Se re direcciona hacia la página de configuración de la base de datos.	
7	Concluye el requisito.	
Pos-condiciones		
1	Ha sido eliminada la base de datos.	
2	Aparece la página de la configuración de la base de datos en el navegador.	
Flujo Alternativo *.a.2.a Se Se selecciona No		
1	Se cancela la acción.	
Pos-condiciones		
1	NA	
Flujo Alternativo *.a.3.a Ocurre un error durante la conexión o eliminación de la base de datos.		
1	Se muestra el mensaje de error “Han ocurrido errores eliminando la base de datos, Se recomienda eliminar la base de datos manualmente”.	
2	Se re direcciona hacia la página de configuración de la base de datos.	
3	Concluye el requisito.	
Pos-condiciones		
1	No se elimina la base de datos.	
2	Aparece la página de la configuración de la base de datos en el navegador.	
Conceptos	**Subsistemas**	Visibles en la interfaz: Nombre subsistema. Utilizados internamente: Idsubsistema. Instalado. Obligatorio.
Requisitos especiales	NA	

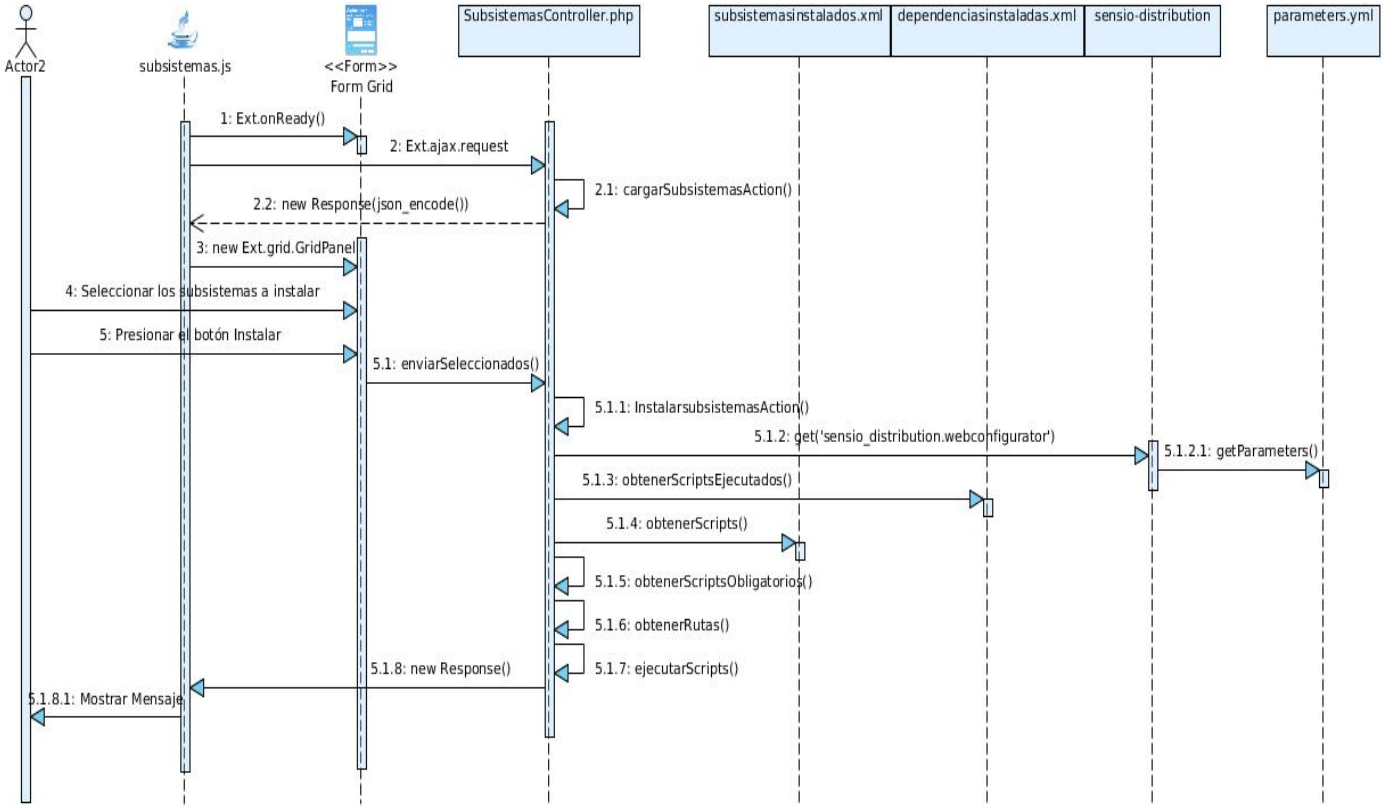
Asuntos pendientes	NA.

Anexo 3: Validación de atributos.

Nombre	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**	**Restricciones**	
					Clases válidas	Clases no válidas
Base de Datos	Es el nombre de la base de datos que se va a crear.	varchar	No	Si	Letras y números siempre comenzando con letras, letras.	Espacios, caracteres especiales.
Servidor	El servidor de base de datos que se utilizará.	varchar	No	No	Letras. Números	Espacios, caracteres especiales.
Puerto	Puerto que se va a utilizar de la base de datos.	número entero	No	No	Números.	Letras, espacios, caracteres especiales.
Usuario	Nombre del usuario que va a trabajar en la base de datos.	varchar	No	No	Letras.	Espacios, Números, caracteres especiales.
Contraseña	Contraseña del usuario que va a trabajar con la base de datos.	varchar	No	No	Letras, números, caracteres especiales, espacios.	N/A
Centro de datos	Valor que se le otorga a la base de datos a crear para la réplica de datos.	número	No	No	Números.	Letras, espacios, caracteres especiales.



Anexo 4: Diagrama de clase con estereotipos web Instalar Subsistemas



Anexo 5: Diagrama de secuencia Instalar Subsistemas