

**VERSIÓN 1.0**

**SISTEMA DE CONTROL DE PESO**

**SISTEMA DE CONTROL DE PESO**

**SISTEMA DE CONTROL DE PESO**

**PROYECTO SCP**

**PROYECTO SCP**

**Proyecto SCP**

**PROYECTO SCP**

**PROYECTO SCP**

**Sistema de gestión de peso**

**Plan de Desarrollo Software y Documentación Técnica**

# Versión 1.0

**Hoja de control del documento**

|  |  |  |  |
| --- | --- | --- | --- |
| **Título** | Plan de desarrollo de software y documentación técnica. Sistema de control de peso  SCP | | |
| **Descripción** | SCP es una aplicación en red desarrollada para facilitar el control de trazabilidad del transporte en las carreteras intradepartamentales, a través de registros de datos para su posible análisis. | | |
| **Autor** | Víctor Paúl Téllez Pérez | | |
| **Responsable** | Ing. Juan Gabriel Bermúdez Hernández | | |
| **Organización** | System and Solutions. | | |
| **Categoría** | Documentación técnica | | |
| **Palabras clave** | Trazabilidad del transporte, software, SCP | | |
| **Versión** | 1. | **Fecha** | 16 de enero de 20 |
| **Estadísticas** | 8766729 bytes, 137782 caracteres, 92 páginas, 2761 párrafos, 24445 palabras | | |
| **Edición** | 0 revisiones | | |

**Historial de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 2020-11-16 | 1.0 | Versión inicial | Víctor Paúl Téllez Pérez |

**Tabla de contenidos**

1. [Introducción 7](#_TOC_250051)
   1. [Propósito 7](#_TOC_250050)
   2. [Usuarios 7](#_TOC_250049)
   3. [Alcance 8](#_TOC_250048)
   4. [Resumen 8](#_TOC_250047)
   5. [Advertencias 9](#_TOC_250046)
   6. [Glosario, acrónimos y definiciones 9](#_TOC_250045)
2. [Vista general del proyecto 11](#_TOC_250044)
   1. [Propósito, alcance y objetivos 11](#_TOC_250043)
   2. [Restricciones y suposiciones 12](#_TOC_250042)
      1. Restricciones 12
      2. Suposiciones 13
   3. [Aclaraciones previas 13](#_TOC_250041)
      1. Especificación 14
      2. Diseño y arquitectura 15
      3. Programación 15
      4. Prueba 15
      5. Mantenimiento 15
   4. [Ciclo de vida 15](#_TOC_250040)
   5. [Normativa y protocolos aplicables al ciclo de vida 17](#_TOC_250039)
   6. [Metodología 18](#_TOC_250038)
   7. [Entregables del proyecto 23](#_TOC_250037)
3. [Organización del proyecto 25](#_TOC_250036)
   1. [Recursos humanos 25](#_TOC_250035)
   2. [Jerarquía RRHH 26](#_TOC_250034)
4. [Gestión de procesos en el proyecto 29](#_TOC_250033)
   1. [Pendiente 29](#_TOC_250032)
5. [Infraestructura 34](#_TOC_250029)
6. [Estructura física de la información 36](#_TOC_250028)
7. [Estructura lógica de la información 40](#_TOC_250027)
   1. [Elementos 40](#_TOC_250026)
   2. [Relaciones entre elementos 40](#_TOC_250025)
   3. [Taxonomías 41](#_TOC_250024)
   4. [Metadatos 42](#_TOC_250023)
8. [Diseño de la arquitectura del sistema 46](#_TOC_250022)
   1. [Descripción general 46](#_TOC_250021)
   2. [Ingeniería de requisitos 50](#_TOC_250020)
      1. Requisitos funcionales 51
      2. Requisitos organizacionales 52
      3. Requisitos no funcionales 52
   3. [Implementación de la solución 52](#_TOC_250019)
9. [Resumen de procesos del sistema 55](#_TOC_250018)
   1. [Actores y casos de uso. Modelos básicos 55](#_TOC_250017)
      1. Actores 55
      2. Casos de uso 56
   2. [Procesos 59](#_TOC_250016)
      1. Procesos principales 59
      2. Procesos auxiliares 60
      3. Procesos del framework 60
10. [Interfaces del sistema 61](#_TOC_250015)
    1. [Back-end 61](#_TOC_250014)
    2. [Front-end 62](#_TOC_250013)
11. [Componentes SW 64](#_TOC_250012)
    1. [Componentes básicos 64](#_TOC_250011)
    2. [Estructura SW 64](#_TOC_250010)
    3. [Algoritmos y librerías básicas 66](#_TOC_250009)
       1. Algoritmos 66
       2. Librerías 67
12. [Gestión de riesgos 69](#_TOC_250008)
13. [Verificación de la calidad 71](#_TOC_250007)
    1. [Evaluación del nivel de calidad 72](#_TOC_250006)
       1. Herramientas de medición de calidad 72
       2. Matriz de control de calidad 73
       3. Informe de errores 74
       4. Evaluación 74
14. [Anexo gráfico 75](#_TOC_250005)
15. [Anexo documental 89](#_TOC_250004)
    1. [Informe sobre cobertura geográfica 89](#_TOC_250003)
    2. [Informe sobre LOC y complejidad SW 89](#_TOC_250002)
    3. [Normativa aplicada 90](#_TOC_250001)
    4. [Bibliografía 90](#_TOC_250000)

**Tabla de ilustraciones**

Ilustración 1. Jerarquía metodológica en SCP 19

Ilustración 2. Ciclo SCRUM 21

Ilustración 3. Desarrollo secuencial y solapado 22

Ilustración 4. Disciplinas y fases RUP en SCP 23

Ilustración 5. Interacciones entre actores en SCP 28

Ilustración 6. Modelo simplificado SI SCP 47

Ilustración 7. Comparativa de test de velocidad MariaDB y MySQL 50

Ilustración 8. Procesos GC según PMI 71

Ilustración 9. EF como subproyecto de EX. Mapa conceptual 75

Ilustración 10. EF: Esquema de flujo simplificado. Mapa conceptual 76

Ilustración 11. EF. Esquema procesual. Mapa conceptual 77

Ilustración 12. Stack de WordPress 78

Ilustración 13. EF. Modelo de datos 79

Ilustración 14. Relaciones entre los 3 sistemas principales de EF 80

Ilustración 15. Registro ENTIDADES. Relaciones 81

Ilustración 16. Registro PERSON. Relaciones 82

Ilustración 17. Registro PUBLICACIONES. Relaciones 82

Ilustración 18. Registro EMPRESA. Relaciones 83

Ilustración 19. Registro EXPOSICIÓN. Relaciones 84

Ilustración 20. Interfaz back-end 85

Ilustración 21. Interfaz front-end 86

Ilustración 22. Diagrama UML genérico. Modelo de casos de uso 87

Ilustración 23. Servidor admin.expofinder.es. Tráfico últimos 12 meses 87

Ilustración 24. Servidor admin.expofinder.es. Uso CPU últimos 12 meses 88

Esta página ha sido intencionadamente dejada en blanco.

# Introducción

El presente documento (desde ahora, PDSDT) está destinado a describir el plan de desarrollo (en adelante PDS) de la aplicación denominada Sistema de Control de Peso (en adelante SCP) y sus especificaciones técnicas (en adelante DT). Este documento provee una visión global del enfoque de desarrollo propuesto y una descripción abreviada de las mencionadas especificaciones.

El proyecto ha sido realizado y descrito en PDSDT basándose en una estructura de tipo *Rational Unified Process* (RUP), que junto con el Lenguaje Unificado de Modelado UML constituye la meto- dología estándar más utilizada para el análisis, diseño, implementación y documentación de sis- temas orientados a objetos. Por ello PDSDT utiliza la terminología propia de dicha metodología e incluye el modelo del negocio y el alcance de SCP, identifica los actores y casos de uso y esboza un plan de negocio para determinar qué recursos deben ser asignados a cada tarea.

El enfoque de desarrollo propuesto constituye una configuración del proceso RUP de acuerdo a las características de SCP, seleccionando los roles de los participantes, las actividades a realizar y los artefactos (entregables) que serán generados. PDSDT es, a su vez, uno de los artefactos RUP.

## Propósito

El propósito de PDS es proporcionar la información necesaria para controlar el proyecto SCP y describir su funcionalidad, además de dotarlo de una base teórica susceptible de revisión, modificaciones y mejoras. El propósito de DT es describir con cierto nivel de detalle las soluciones técnicas adoptadas para implementar adecuadamente PDS de acuerdo a lo en él estipulado.

Téngase en cuenta que existe documentación adicional detallada, que incluye el código fuente y los diagramas UML completos empleados durante el proceso de desarrollo que no figura en PDSDT. Los objetivos esenciales son:

* + - Documentar textualmente de forma sucinta pero clara los elementos esenciales que configuran el proyecto SCP.
    - Referenciar las fases esenciales del proyecto: análisis previo, desarrollo primario, pruebas funcionales, puesta en marcha y corrección de errores y/o disfuncionalidades.
    - Informar sobre el estado actual de la ejecución del proyecto, situándolo con respecto a la planificación inicial.
    - Evaluar el grado de ajuste del estado actual mencionado en el punto anterior y las posibilidades de conclusión exitosa del proyecto SCP en tiempo y forma.

## Usuarios

PDSDT está concebido como herramienta de trabajo y referencia, y no como texto con finalidad divulgativa o para usuarios finales. La audiencia prevista para el mismo, por tanto, tiene un marcado perfil técnico. Los actores previstos para la consulta son:

* + - *Responsables SCP*: como herramienta de seguimiento y para control del proceso de desarrollo.
    - *Desarrolladores*: como guía de actividades, calendario de fases y ejecución**.**
    - *Auditores*: para verificar el ajuste de lo proyectado a la realidad de la implementación.
    - *DBA*: para el manejo y control de los datos recopilados en las bases de datos.

## Alcance

PDSDT no pretende abarcar la descripción exhaustiva de ninguno de los apartados PDS y DT. Para ello se establece como artefacto una posterior versión, cuya entrega está prevista a la finalización del desarrollo del proyecto SCP. El detalle de las iteraciones se describe en los planes individuales de cada una, que figurarán detalladamente en el artefacto final y que ahora se mencionan en [4.2.](#_bookmark13) La base para la planificación de las citadas iteraciones reside en la estructura de subprocesos que se describe en DT (véase [9](#_bookmark23)).

Dadas las características específicas del proyecto SCP (dependiente del desarrollo de SCP), la captura de información para ingeniería de requisitos no se ha realizado, como suele ser habitual, a través de un *stakeholder* representante del MTI (en adelante RMTI) sino que ha sido realizada a partir de las estipulaciones de los actores responsables, que detentan, en este caso concreto, la calidad de responsables también en el MTI. Por tanto, los requerimientos se han establecido en un primer momento mediante una estimación aproximada, y una vez comenzado el proyecto y duran- te la fase inicial se ha ido refinando a partir de las entregas de prototipos de artefactos. Está pre- vista la edición de sucesivas versiones de PDSDT (anteriores a la final) para adecuarse a tales circunstancias.

## Resumen

De acuerdo a lo especificado, PDSDT se organiza en secciones según el siguiente esquema operativo, en el que cada apartado va precedido por el ámbito al que pertenece (PDS o DT):

* + - **PDS. Vista general del proyecto**. Proporciona una descripción del propósito, alcance y objetivos de SCP, estableciendo los artefactos que se producen y utilizan durante la ejecución del proyecto. Se incluye el modelo de ciclo de vida adoptado y la metodología de desarrollo.
    - **PDS. Organización del proyecto**. Describe la estructura organizacional y funcional de los recursos humanos implicados en el desarrollo de SCP.
    - **PDS. Gestión de procesos del proyecto**. Explica los costos temporales y la planificación estimada, define las fases e hitos de SCP y describe cómo se realiza su seguimiento.
    - **PDS. Infraestructura**. Estipula los requisitos mínimos necesarios para poder implementar la solución propuesta de manera que se garantice el adecuado funcionamiento del conjunto.
    - **DT. Estructura física de la información**. Describe la organización de las tablas y relaciones de la BD.
    - **DT. Estructura lógica de la información**. Dadas las características específicas de la implementación de SCP, establece la significación de cada elemento semántico en el conjunto de datos almacenados.
    - **DT. Diseño de la arquitectura del sistema**. Descripción de los patrones y *frameworks*

utilizados y las razones para ello.

* + - **DT. Resumen de procesos del sistema**. Descripción de los procesos y subprocesos implicados y sus referencias al modelo de datos.
    - **DT. Interfaces del sistema**. Descripción sucinta de los interfaces principales con mención expresa de los actores implicados.
    - **DT. Componentes SW**. Descripción de los diferentes componentes software empleados y sus relaciones. Se incluyen las referencias a librerías externas y SW de terceros.
    - **DT. Gestión de riesgos**. Descripción del plan de seguridad activa y proactiva.
    - **DT. Verificación de la calidad**. Descripción de los mecanismos de certificación y validación de la información almacenada.

## Advertencias

Por incluir los apartados correspondientes a PDS y DT en el mismo documento, cabe la posibilidad de que determinados apartados de PDSDT puedan prestarse a confusión o no resulten claros. La correcta interpretación del presente texto debe realizarse tomando en cuenta las llamadas y referencias cruzadas de unos apartados a otros, donde se contiene la información significativa. Por ejemplo, puesto que es imposible realizar una estimación de esfuerzo basada en casos de uso si no se conoce la estructura de datos, los procesos y los casos mismos, es conveniente, tal y como se establece e[n 4.1,](#_bookmark12) consultar la información suministrada en [7](#_bookmark19) y [9](#_bookmark23).

## Glosario, acrónimos y definiciones

Con el fin de clarificar la terminología utilizada en el presente documento, se adjunta un breve prontuario de acrónimos, siglas y términos de uso específico.

|  |  |
| --- | --- |
| **Baseline** | Registro estático del estado de cada artefacto al final de cada iteración, estableciendo una versión o *release* |
| **Bug** | En el contexto de desarrollo, error descubierto en un programa |
| **CDU** | Caso de uso |
| **RMTI** | Actor que realiza el papel de representante del MTI o “cliente” (no en el sentido comercial de la palabra) |
| **CMS** | *Content Management System* |
| **CRUD** | *Create, Read, Update, Delete*. Acrónimo del proceso de gestión de registros en una base de datos |
| **Changelog** | Registro detallado de los cambios realizados en los artefactos |
| **CBS** | *Control de Basculas* |
| **DP** | *Data Person.* |
| **PCT** | Punto de control o estación de bascula |
| **DRAE** | Diccionario de la Real Academia Española |
| **DT** | Documentación Técnica |
| **SCP** | Sistema de control de Peso |
| **S&S** | System and Solutions |
| **MTI** | Ministerio de Transporte e Infraestructura |
| **FW** | *Framework*. Terminología de arquitectura de desarrollo |
| **GC** | Gestión de calidad |

|  |  |
| --- | --- |
| **HW** | *Hardware* |
| **IEC** | *International Electrotechnical Commission* |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **INTECO** | Instituto Nacional de Tecnologías de la Comunicación |
| **ISO** | *International Organization for Standarization* |
| **KISS** | *Keep It Simple, Stupid*. Lema de trabajo que aboga por la implantación de la “navaja de Oc- kham” como principio esencial en la adopción de soluciones |
| **LP** | *Lean Programming*. Acrónimo de un tipo de metodología ágil |
| **OS** | *Operating System* |
| **OSI** | *Open Source Initiative*. También, según el contexto, puede aludir al modelo *Open System Interconnection* descrito por ISO/IEC 7498-1 |
| **PDS** | Proyecto de Desarrollo de Software |
| **PDSDT** | Documento actual (Proyecto de Desarrollo de Software y Documentación Técnica) |
| **PMBOK** | *Project Management Body of Knowledge*. Estándar desarrollado por el *Project Management Institute* para gestionar los proyectos y medir su calidad y éxito |
| **PMI** | *Project Management Institute* |
| **POO** | Programación Orientada a Objetos. También conocido por sus siglas inglesas, OOP |
| **RAD** | *Rapid Application Development* |
| **Red** | Con mayúscula inicial, Internet |
| **Release** | Término inglés para *lanzamiento* o liberación de una versión de un producto SW o “artefac- to”, en terminología RUP |
| **RRHH** | Recursos humanos |
| **RUP** | *Rational Unified Process* |
| **SaaS** | *Software as a Service* |
| **SC** | Solicitud de cambio. En contexto de procesos (véase [9)](#_bookmark23), Sistema de Captura |
| **SCRUM** | Metodología organizativa relacionada con los sistemas ágiles de desarrollo |
| **SEO** | *Search Engine Optimization* |
| **SESC** | *Software Engineering Standards Committee* |
| **SGBD** | Sistema Gestor de Bases de Datos. También se conoce como DBMS |
| **SI** | Sistema de Información |
| **Skeumórfico** | Interfaz de usuario basado en la simulación de objetos realistas (sobreados, 3D,…) |
| **SPICE** | *Software Process Improvement Capability Determination* |
| **Stakeholder** | Parte interesada en el proceso de desarrollo, como interno o como CL, en terminología SCRUM |
| **SW** | *Software* |
| **UMA** | Universidad de Málaga |
| **UML** | *Unified Modeling Language* |
| **Workflow** | O *flujo de trabajo*. Estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincroni-  zan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento. Ter- minología propia de RUP |
| **XP** | *eXtreme Programming*. Acrónimo de un tipo de metodología ágil |

# Vista general del proyecto

## Propósito, alcance y objetivos

El propósito del proyecto de desarrollo de *software* SCP, un proyecto que realice la recopilación de los datos de cada uno de los medios de transporte que llegue una báscula del Ministerio de Transporte e Infraestructura (Nicaragua), también posibilita la realización de otros proyectos de esta misma institución, en conjunto con la recopilación de los datos de peso de los medios de transporte, así mismo alguna activada que se realice después de la recopilación de los datos, como son Multas, Advertencias, Sanciones, etc.

El proyecto formara parte de las líneas de trabajo emprendidas en conjunto con otros desarrolladores, cuyo objetivo específico es la promoción, desarrollo y consolidación en Nicaragua de las sobre las vías de transporte terrestres, acuática y de construcción de todas las areas de nicaragua, Asimismo, este proyecto forma parte de las iniciativas promovidas por la Red Interna- cional de Estudios Digitales sobre la Cultura Artística, ReArte.Dix, la primera organización interna- cional dedicada a este fin en el contexto hispanohablante, financiada por la Universidad de Málaga como parte de su plan propio de investigación y como parte de los proyectos asociados al Campus de Excelencsia Internacional Andalucía Tech.

El proyecto SCP exige que se realicen las acciones oportunas para que un sistema lo más automatizado posible capture datos que estén al alcance del objeto a analizar: peso en bascula, datos del conductor, datos del medio de transporte y la actividad consecuente. Los datos deben poder, a posteriori, ser susceptibles de recibir un análisis cuantitativo y/o cualitativo que permita extraer conclusiones científicamente o económicamente válidas y, por tanto, generar nuevo conocimiento, a través de reportes.

Para cumplir adecuadamente con el cometido, las características requeridas por el sistema SCP deberán ser las siguientes:

* ***Captura automatizada de información*.**
  + Localización de fuentes de información primarias a partir de las cuales se obtendrán los datos.
  + Puesta a disposición de los datos capturados a los actores encargados de su consolidación y revisión.
  + Establecimiento del proceso de captura como tarea automatizada y repetitiva del peso obtenido en basculas.
  + Establecimiento de filtros previos que afinen el proceso de captura en función de la pertinencia de los contenidos localizados en las fuentes remotas de la Red.
* ***Consolidación de la información localizada***.
  + Revisión del conjunto de capturas por parte de los actores encargados de ello, descartando la información capturada no relevante a los efectos del propósito de SCP.
* Completado de la información considerada esencial en cada PCT localizada y que esté ausente en los datos primarios capturados de las ubicaciones remotas de la Red.
* Almacenamiento de los datos localizados y revisados en un SGBD con garantías de seguridad, fiabilidad y posibilidades de recuperación.
* Normalización de la información almacenada mediante revisiones que garanticen la recuperabilidad y homologabilidad de los datos y permitan su clasificación y ordenación.
* ***Explotación de la información****.*
  + Generación de un formato exportable para su uso por parte de los actores externos a SCP y pertenecientes a MTI, regido por normativa de intercambio lo más estándar posible, adaptada a las peculiaridades de la estructura y arquitectura de datos de la aplicación.
  + Creación de un mecanismo de exportación de los datos consolidados y normalizados a los actores externos intervinientes que formen parte de EX y estén habilitados para ello previa adecuada salvaguarda de identificación y validez da las solicitudes de acceso.
  + Creación de un mecanismo de explotación directa de la información cuantificable de los datos coleccionados, disponible desde el apropiado interfaz de SCP y paralela a cualquier otra implementación que al respecto puedan llevar a cabo los oportunos actores de SCP.

## Restricciones y suposiciones

Si bien no existe, como en el caso de los proyectos de carácter comercial o no académico, un conjunto de restricciones muy específicas, relacionadas con la infraestructura del MTI y su disponibilidad económica y temporal, la propia existencia de unos plazos de término para la entrega de la documentación, condiciona el desarrollo de SCP, así como el presupuesto derivado de la dotación económica que reciba S&S.

Este apartado [2.2](#_bookmark4) podrá ser modificado en sucesivas versiones de PDSDT de forma que se adapte a las condiciones evolutivas de SCP.

* + 1. **Restricciones**

Derivan de las condiciones de adjudicación de la ayuda y de los recursos humanos, temporales y materiales disponibles por el Ministerio de transporte e infraestructura.

* + - * La recopilación de datos para la codificación del sistema de parte del equipo técnico de S&S.
      * Colaboración de parte del equipo de MTI para brindar la debida información que se solicite.
      * Previamente a la finalización, un grupo de actores deberá realizar el proceso de análisis de la información custodiada por SCP.
      * Por las condiciones MTI como la entidad que hará uso del sistema, dará acceso a un grupo humano con la adecuada formación para acometer la tarea de actores necesarios en SCP con perfiles muy específicos, pero no existe un excesivo margen de maniobra para retribuir la colaboración de dichos actores, de manera que SCP puede verse sometido a picos de productividad y rendimiento no planificados, derivados de la compatibilidad de tareas por parte de los mencionados actores.
      * Al tratarse de una entidad gubernamental dependiente del presupuesto general de la república, SCP debe utilizar recursos HW y/o SW compatibles.
      * Por necesidad de cumplir adecuadamente los plazos estipulados SCP debe estar a pleno rendimiento y completamente funcional para permitir la explotación de los datos coleccionados con antelación en un periodo de 3.5 meses una vez aprobado este PDSDT.
      * Para que la validez de los estudios cuantitativos y/o cualitativos derivados del uso de los datos coleccionados transferidos desde SCP a los actores oportunos sea efectiva, la información recopilada deberá abarcar de la forma más exhaustiva posible un conjunto normas cuya significación permita elevar las conjeturas derivadas de la investigación operativa de los datos a la categoría de conclusiones científicamente contrastables.
    1. **Suposiciones**
       - SCP supone la existencia de fuentes primarias de información disponibles en la Red, de las que es posible extraer un conjunto relevante de datos que tratar. La no existencia de dichas fuentes ya es, de por sí, significativa.
       - Dada la naturaleza y el enfoque que se adopta como solución proyectual para SCP (véase [5](#_bookmark17)), todos los actores que intervengan en el proceso de consolidación y almacenamiento de datos deben disponer del HW y la conectividad adecuados para acceder a la Red.
       - SCP no necesita una ubicación física o “domicilio” para desarrollarse, pudiendo intervenir los actores desde cualquier punto, lo que por una parte supone una evidente ventaja, pero por otra puede ser contraproducente, al no existir un “set” HW/SW estandarizado que se empleará como herramienta de trabajo por parte de los actores.
       - Se supone que todos los actores implicados en los procesos de grabación, normalización, verificación y revisión aludidos en [4](#_bookmark11) disponen de la adecuada formación, no necesariamente completa en ambos aspectos pero sí obligatoriamente en el primero, tanto en DP y CBS como en el manejo de herramientas ofimáticas simples, no siendo posible para el plan de SCP contar con una fase de formación genérica de sus actores, si bien sí la hay, en formato cíclico, para la específica.

## Aclaraciones previas

La norma IEEE Std 610 define software como “programas, procedimientos y documentación y da- tos asociados, relacionados con la operación de un sistema informático”. PDSDT utiliza tal definición como referencia y, por tanto, considera la existencia de tres componentes en el conjunto de desarrollo.

* + - **Programas**. Conjuntos de instrucciones que proporcionan la funcionalidad deseada cuan- do son ejecutadas por el ordenador. Están escritos usando lenguajes específicos que los ordenadores pueden leer y ejecutar, tales como lenguaje ensamblador, Basic, C, C#, Visual Basic … Los programas también pueden ser generados usando generadores de programas.
    - **Datos**. Elementos significativos portadores de la información que se pretende tratar mediante el empleo de programas.
    - **Documentos**. Imprescindibles para que los actores dispongan de una explicación de cómo usar los programas (manuales de usuario y de operatoria) y de cómo intervenir sobre ellos para ajustar su funcionalidad.

Según la clasificación de SW recogida en el documento “Ingeniería de Software: metodologías y ciclos de vida” (INTECO 2009)2, la aplicación de SCP puede considerarse como integrante de las siguientes categorías:

* + - **Software de aplicaciones**: se usan para proveer servicios a clientes y ejecutar negocios de forma más eficiente. El software de aplicaciones puede ser un sistema pequeño o uno grande integrado.
    - **Software de gestión**: el proceso de la información comercial constituye la mayor de las áreas de aplicación del software. Los sistemas discretos (por ejemplo: nóminas, cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión (SIG) que accede a una o más bases de datos que contienen in- formación comercial. Las aplicaciones en esta área reestructuran los datos existentes pa- ra facilitar las operaciones comerciales o gestionar la toma de decisiones.
    - **Software basado en web**: las páginas web buscadas por un explorador son software que incorpora instrucciones ejecutables y datos.

La **ingeniería de software** dispones de una definition de IEEE SESC (1990)3 del siguiente tenor: “Software engineering is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software”.

En la ingeniería de software, la **ingeniería de requisitos** comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un SW nuevo o modificado, tomando en cuenta los diversos requerimientos de las partes interesadas, que pueden entrar en conflicto entre ellos. El propósito de la ingeniería de requisitos es hacer que los mismos alcancen un estado óptimo antes de alcanzar la fase de diseño en el proyecto. Los buenos requisitos deben ser medibles, comprobables, sin ambigüedades o contradicciones, etc. La norma 830-1988 de IEEE regula los procedimientos de la ingeniería de requisitos.

El estándar IEEE-1074 para los procesos de vida del software describe el conjunto de actividades y procesos obligatorios para el desarrollo y mantenimiento de SW. Tiene como objetivo establecer un marco común para el desarrollo de modelos para el proceso de construcción. EF estructura su proceso vital en fases.

* + 1. **Especificación**

Escribir detalladamente el SW en desarrollo, en una forma matemáticamente rigurosa. Cubierta mediante PDSDT.

* + 1. **Diseño y arquitectura**

Describir las pautas o patrones a los que se adecua el mecanismo de funcionamiento de la aplicación, incorporando consideraciones de la implementación tecnológica y obteniendo un modelo cercano a la programación orientada a objetos. Cubierta mediante PDSDT.

* + 1. **Programación**

Reducir un diseño a código ejecutable en un ordenador.

* + 1. **Prueba**

Verificar que SW realiza correctamente las tareas indicadas en la especificación del proyecto

1. INTECO. *Ingeniería de Software: metodologías y ciclos de vida*. Laboratorio Nacional de Calidad del Software. Instituto Nacional de Tecnologías de la Comunicación. 2009.
2. IEEE. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE std 610.12-1990, 1990.
   * 1. **Mantenimiento**

Corregir errores y mejorar SW para solventar fallos descubiertos y tratar con nuevos requisitos. Puede ser de cuatro tipos: *perfectivo* (mejorar la calidad interna de los sistemas), *evolutivo* (incorporaciones, modificaciones y eliminaciones necesarias en un producto software para cubrir la expansión o cambio en las necesidades del usuario), *adaptativo* (modificaciones que afectan a los entornos en los que el sistema opera, por ejemplo, cambios de configuración del HW, SW de base, gestores de base de datos, comunicaciones) y *correctivo* (corrección de errores).

Estas fases se corresponden, de acuerdo a lo normalizado en [*Ilustración 4. Disciplinas y fases*](#_bookmark7)[*RUP en SCP*](#_bookmark7) (ver [2.7](#_bookmark6)), respectivamente, con lo expresado en la siguiente matriz.

|  |  |
| --- | --- |
| **FASES IEEE-1074** | **FASES RUP** |
| Especificación | Inicial |
| Diseño y arquitectura | Elaboración |
| Programación | Construcción |
| Prueba | Transición |
| Mantenimiento | Transición |

## Ciclo de vida

A los efectos PDSDT, se entenderá por ***ciclo de vida*** o ***paradigma*** el conjunto de fases por las que pasa el sistema que se está desarrollando desde que nace la idea inicial hasta que el software es retirado o remplazado (muere). Las fases mencionadas se desarrollan de forma sucesiva, y pueden estar compuestas por tareas panificables. Cada fase se construye agrupando tareas (actividades elementales), que pueden compartir un tramo determinado del tiempo de vida de un proyecto. La agrupación temporal de tareas impone requisitos temporales correspondientes a la asignación de recursos (humanos, financieros o materiales).

Según el modelo de ciclo de vida, la sucesión de fases puede ampliarse con bucles de *feedback*, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones a los resultados intermedios que se van produciendo (retroalimentación).

Un concepto adicional, que se usa en PDSDT, es el de ***entregables***. Por tales deben entenderse los productos intermedios que generan las fases. Pueden ser materiales o inmateriales (documentos, SW). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.

El modelo de ciclo de vida adoptado en el desarrollo de SCP es el de **prototipos**, propio de la metodología ágil utilizada para todo el proceso completo (ver [2.6](#_bookmark5)). Según los principios de este modelo, el RMTI a menudo define un conjunto de objetivos generales para SW, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficiencia de un algoritmo, de la calidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción hombre-máquina. En estas y en otras muchas situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque.

El ciclo de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y RMTI encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para RMTI. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa RMTI y se utiliza para refinar los requisitos del SW a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades de RMTI, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

Entre las ventajas, citadas por INTECO 2009, figuran:

* + - Ofrece visibilidad del producto desde el inicio del ciclo de vida con el primer prototipo. Esto puede ayudar al RMTI a definir mejor los requisitos y a ver las necesidades reales del producto. Permite introducir cambios en las iteraciones siguientes del ciclo. Permite la realimentación continua del RMTI.
    - El prototipo es un documento vivo de buen funcionamiento del producto final. El RMTI reacciona mucho mejor ante el prototipo, sobre el que puede experimentar, que no sobre una especificación escrita.
    - Este modelo reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.

El documento citado también describe algunos inconvenientes de este tipo de ciclo, pero, dadas las características específicas de SCP, el equipo de desarrollo ha concluido que eran poco aplicables en este caso concreto y, por tanto, su impacto en el resultado final es despreciable. Tales inconvenientes pueden describirse como sigue:

* + - Es posible que el conjunto procesual de desarrollo sea lento. Además, puede llevar a que se realicen fuertes inversiones en un producto desechable ya que los prototipos se descartan. Ambas contrariedades carecen de relevancia en el caso SCP, puesto que tanto la inversión está predeterminada desde el primer momento, y la relación con RMTI es con intermediarios, es decir un tanto indirecta.
    - Con este modelo pueden surgir problemas con RMTI, que ve funcionando versiones del prototipo, pero puede desilusionarse porque el producto final aún no ha sido construido. El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y de mantenimiento que tiene con el RMTI. No es el caso de SCP, por tratarse de un equipo meramente de desarrollo por así decirlo, un prototipo representaría un eslabón más del equipo de diseño.

## Normativa y protocolos aplicables al ciclo de vida

La norma ISO/IEC 12207 define un modelo de ciclo de vida como un marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto software, y que abarca toda la vida del sistema, desde la definición de sus requisitos hasta el final del uso.

Esta norma agrupa las actividades que pueden llevarse a cabo durante el ciclo de vida del SW en procesos principales, de apoyo y organizativos. Cada proceso del ciclo de vida está dividido en un conjunto de actividades y cada una de ellas en un conjunto de tareas. Con afán de simplificación y adecuación a las necesidades específicas del proyecto SCP, el equipo de desarrollo resumió la estructura de la mencionada norma, dejando solamente los procesos y actividades de aplicación en el caso. Por ejemplo, la actividad de adquisición tiene el sentido ya que SCP se trata de un trabajo personalizable, una “venta” de software.

Téngase en cuenta que esta agrupación de procesos no responde a un mecanismo secuencial, donde uno precede a otro, sino que pueden ejecutarse de forma simultánea. Por ejemplo, las actividades de solución de problemas y mejora van unidas al tiempo de operación.

A continuación, se describe la jerarquía de procesos y tareas tal y como se implanta en SCP.

* + - **Procesos principales**: dan servicio a las partes más importantes del ciclo. Los principales son:
      * ***Suministro***. Asimilada a la compra del *HW* necesario para llevar a término el proyecto.
      * ***Desarrollo***. Creación del *SW* para la funcionalidad de la aplicación final.
      * ***Operación***. Incluye el proceso de captura automatizado, la consolidación y grabación de datos, su normalización y verificación, además de las tareas propias de administración del aplicativo (parámetros, opciones por defecto,…).
      * ***Mantenimiento***. Tareas propias de la administración del sistema. Garantizan el adecuado funcionamiento del OS empleado en el servidor o alojamiento web, del motor SGBD y de otros elementos comunes de infraestructura, imprescindibles para el funcionamiento correcto del proyecto.
    - **Procesos de apoyo**: contribuyen a la puesta en funcionamiento de otros procesos como parte esencial de los mismos, con un propósito bien definido y al éxito y calidad del proyecto. Son empleados y ejecutados por otros procesos según sus necesidades. Los procesos de apoyo son:
      * ***Documentación*.** Tanto la propia del diseño del proyecto de SW y de su proyecto matriz como los manuales de usuario. Realizada a medias entre el equipo de desarrollo y los actores implicados en verificación y validación.
      * ***Gestión de la configuración*.** Asimilada a las tareas de administración del sistema.
      * ***Verificación*.** Comprobación de la calidad de los datos capturados automáticamente y completado de los mismos con información obtenida por cualquier medio válido (preferentemente, disponible de forma irrestricta en la Red).
      * ***Validación***. **o** ***Normalización***. Adecuación de la información contenida en los datos ya validados a la reglamentación que sobre los mismos exista (clasificaciones, taxonomías,).
      * ***Auditoría***. Realizada por actores externos a SCP, preferiblemente por aquellos que dieron seguimiento a todo el proceso de desarrollo.
      * ***Solución de problemas*.** Ajuste fino del funcionamiento de SW aplicativo, de oficio

o a petición de cualquier actor implicado en su uso.

* + - **Procesos organizativos**: se emplean para establecer e implementar una infraestructura construida por procesos y personal asociado al ciclo de vida, y para mejorar de forma cíclica el conjunto. Los más destacados son:
      * ***Gestión***. Coordinación de los equipos formados por los grupos de actores tal y como se describen en [3](#_bookmark8). Comunicaciones de alcance global para el conjunto de recursos humanos implicados. Relaciones con el ministerio de transporte e infraestructura.
      * ***Infraestructura***. Gestión del suministro de HW/OS/SW imprescindible para la implementación y la implantación. Monitorización periódica y sistemática del estado del conjunto servidor (HW/OS/SW).
      * ***Mejora***. Llevada a cabo a partir de sugerencias en reuniones periódicas de tipo *sprint* según la metodología SCRUM/XP. Implementada por el equipo de desarrollo y supervisada por el *Product Owner* de SCP.
      * ***Formación***. Específica para los actores implicados en verificación y validación. No incluye, por tratarse de un conocimiento previo necesario, preparación en manejo de herramientas ofimáticas.

## Metodología

Definido por el DRAE como “conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal”, en el contexto del DBA utilizamos el término *metodología* para hacer referencia al grupo normativizado de procedimientos para abordar las tareas que componen las fases del ciclo de vida. Cada una de esas tareas puede ser abordada y resuelta de múltiples maneras, con distintas herramientas y utilizando distintas técnicas, y justamente al conjunto de todos esos instrumentos para acometer su realización, conjunto que se rige por unos determinados protocolos, es a lo que el presente documento alude cuando se habla de metodología.

Se suele establecer en la actualidad una distinción entre las *metodologías tradicionales* y las *ágiles*. Cada una de ellas tiene aplicación dependiendo de las características del proyecto abordado. En el caso de SCP, el equipo de desarrollo optó por las denominadas “ágiles”.

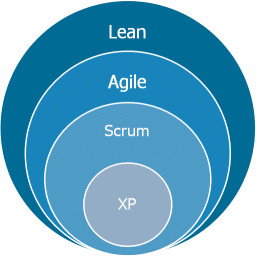
Entre las tradicionales, desechadas para usarlas de base del trabajo en SCP desde el primer momento, figuran algunas como el *Desarrollo de Sistemas de Jackson* (JSD), muy propia de los años 80 del pasado siglo, o, de las mismas fechas, la *Ingeniería de la Información* o la británica *Structured System Analysis and Design Method* (SSADM). El Ministerio de Administraciones Públicas promociona *METRICA* (actualmente en su versión 3) como estándar para el trabajo de organismos oficiales.

Por contraste con los sistemas anteriores, caracterizados por la rigidez de sus procedimientos y la sobreabundancia de entregables documentales, el **desarrollo ágil** de SW utiliza un proceso iterativo como base para abogar por un punto de vista más ligero y más centrado en las personas que en el caso de las soluciones tradicionales. Los usos ágiles emplean retroalimentación en lugar de planificación, como principal mecanismo de control. La retroalimentación se canaliza por medio de pruebas periódicas y frecuentes versiones del software. De hecho, se trata de metodologías aplicables a numerosos tipos de actividades humanas, y no solamente al desarrollo de SW.

Hay muchas variantes de los procesos ágiles, pero en EF el equipo de desarrollo ha procurado ceñirse a dos modelos esenciales: **eXtreme Programming (XP)** y **Lean Programming**. En ambos casos, es posible (aunque el desarrollo de la pertinente argumentación queda fuera del alcance del presente documento) hacer compatibles tales modelos con el estándar ISO/IEC 15504 o SPICE, considerado uno de los más fiables para el establecimiento de mejoras en el complejo conjunto de especificaciones de la ingeniería de software.

En el caso de la *programación extrema (XP)*, las fases se realizan en pasos muy cortos (o "continuos") con respecto al anterior. Se crean pruebas automatizadas para proveer metas concretas al desarrollo. Después se programa el código, que será completo cuando todas las pruebas se superan sin errores. El diseño y la arquitectura emergen a partir de la refactorización del código, y lo realizan los propios desarrolladores. El sistema, incompleto, pero funcional, se despliega para su

demostración al RMTI a través de los usuarios directos (al menos uno de los cuales pertenece al equipo de desarrollo).



Esta metodología es muy conveniente para SCP por cuanto, como se verá en [3,](#_bookmark8) existe un grupo de usuarios finales, que pueden considerarse parte del RMTI, que trabajan como revisores y grabadores de datos capturados y actúan como parte interesada en el ciclo de desarrollo.

*Lean Software Development*, también conocido como *Lean Programming* es un conjunto de técnicas que engloban una metodología de desarrollo ágil de software orientado a conseguir exactamente lo que necesita el RMTI. Es una evolución del Método Toyota de Producción aplicado al desarrollo y que esta muy de moda en-

Ilustración 1. Jerarquía metodológica en EF

tre los equipos de desarrollo en *startups*. Principalmente consiste en ciclos de evolución de SW incrementales en los que se posponen las decisiones lo más posible hasta haber obtenido un feedback del cliente y así reaccionar lo más rápido y eficazmente a sus necesidades. Se fundamenta en tener un equipo potente y comprometido y el principio de aprendizaje continuo sobre el producto final.

El desarrollo basado en LP se debe su nombre a la palabra inglesa que puede traducirse por “esbelto”, pero también por “apoyado”. Ambas cosas coinciden en una metodología cimentada a su vez sobre las buenas prácticas industriales (de Toyota, en concreto)4. LP basa su estructura en siete importantes principios, muy relacionados con los ciclos diarios de desarrollo que se popularizaron tras la publicación del libro *Microsoft Secrets5*, que hacía público el sistema de desarrollo incremental con ciclos diarios utilizado por la célebre empresa norteamericana. Estos principios, a los que SCP se adhiere, son:

* + - Eliminar todo aquello que no sea útil para el desarrollo: burocracia innecesaria, retrasos en entregas, código redundante,…
    - Aprendizaje continuo y ampliado. De los errores también se aprende.
    - No adoptar prematuramente decisiones que comprometan el desarrollo.
    - Reaccionar a la máxima velocidad posible ante modificaciones, ampliaciones o cambios de rumbo en el proyecto y –cómo no- ante los *bugs*.
    - Potenciar el trabajo en equipo y la cohesión entre los diferentes agentes y roles que interactúan entre sí en SCP.
    - Elevar la integridad conceptual a la categoría de rumbo principal del proyecto. Este término se emplea en SCP para aludir a dos cuestiones:
      * Cada artefacto debe funcionar a la perfección en coordinación con los demás, es decir, que la ejecución de un código no puede realizarse correctamente a costa de otro.
      * Cada actor que intervenga como verificador o supervisor debe asumir (por cuanto se verá en lo especificado en [8](#_bookmark20)) que la calidad de los datos que custodia depende exclusivamente de su escrupuloso cumplimiento de las normas de revisión y grabación
    - Ver siempre SCP como un conjunto único, un único proyecto, de forma que nunca se pierda la perspectiva global, que ha de garantizar tanto la integridad conceptual como el aprendizaje continuo y las respuestas rápidas.

Para el trabajo efectivo, SCP adopta una técnica “suavizada” de desarrollo **SCRUM**, un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Presentado por Ken Schwaber a mediados de la década de los noventa del pasado siglo en una conferencia internacional sobre POO, recoge el estudio de dos investigadores japoneses que habían detectado las pautas de funcionamiento básicas en grandes empresas industriales como Xerox, 3M, Canon y otras. El nombre alude a un tipo de puesto de jugador en las tácticas del deporte del rugby.

SCRUM se basa en la adopción de ***roles*** por parte de quienes intervienen en el proceso de desarrollo (tanto de la parte RMTI como del equipo que construye el proyecto). Cada actor interviene según las pautas del rol que se le ha asignado. Los procesos se realizan empleando ***artefactos*** y los hitos de desarrollo se denominan ***eventos***. La gestión del proyecto se realiza mediante reuniones cíclicas cuyo contenido es referenciado documentalmente de carácter periódico en cuanto a temporalidad y cíclico en cuanto a contenido.

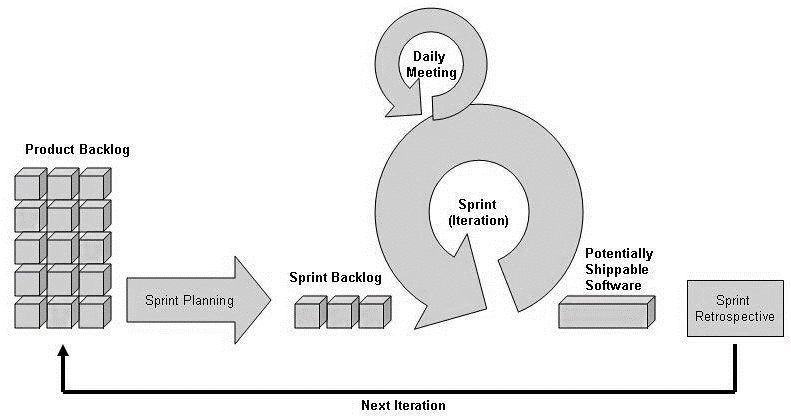


Ilustración 2. Ciclo SCRUM

1. A pesar de lo que comúnmente se cree, los que comenzaron a popularizar el concepto *lean* no fueron los japoneses, sino los laboratorios del MIT, en especial tras la publicación de su célebre libro “La máquina que cambió el mundo” (1990). Sin embargo, fue el matrimonio formado por Mary y Tom Poppendieck los primeros en atribuir el concepto al SW, a través de su obra *Lean Development Software: An Agile Toolkit for Software Development Managers (The Agile Software Development Series)* (2003, Addison-Wesley, EEUU).
2. M. Cusmano y M. Selby. *Microsoft Secrets*. EEUU: HarperCollins Business, 1997.

SCRUM implica, por parte del equipo de desarrollo, la asunción de unos principios, a los que SCP, por las circunstancias concretas de su generación (es un proyecto de recopilación de datos cuya finalidad es lanzar alertas de cuando se incumple los reglamentos de vías terrestres del MTI), se adhiere de forma natural. Dichos principios esenciales son:

* + Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto. Muy conveniente, por cuanto el objetivo final concreto de SCP es “territorio inexplorado todavía, pero se tiene una breve esencia”, y es importante no dar por sentados los planteamientos iniciales. SCP debe “tener cintura” para lidiar con frecuentes cambios de orientación que pueden llegar a afectar al paradigma.
  + Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados. De ahí la importancia de la relación académica inherente a los recursos humanos que constituyen el personal de trabajo.
  + Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada. La limitación de tiempo, la premura de un *back-end* inexorable, tanto más cuanto, como se ha mencionado, ningún miembro del equipo puede disponer de dedicación exclusiva, aconsejan acelerar los procesos y simultanearlos.

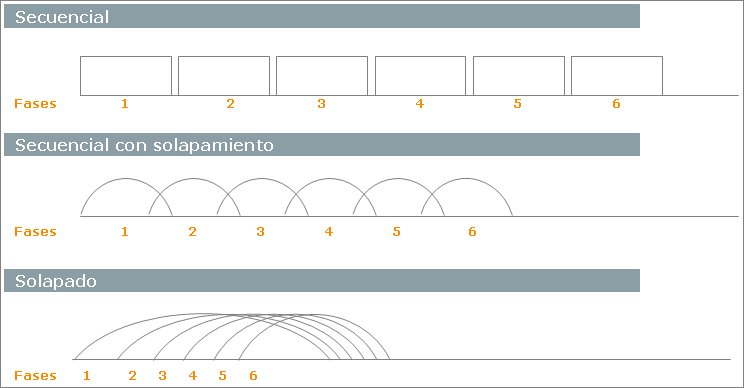
Un principio adicional, no obstante que clave, en el planteamiento SCRUM es la asunción como pauta de desarrollo de que durante un proyecto RMTI puede cambiar de idea sobre lo que necesita o desea obtener, alterando así los requerimientos iniciales e influyendo en la disciplina de la ingeniería de requisitos, y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Adhiriéndose plenamente a ello, SCP adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes. Los “saltos” evolutivos se llevan a término en ciclos, denominados ***Sprints***.

Ilustración 3. Desarrollo secuencial y solapado

SCP no implementa el modelo SCRUM completo (es posible obtener documentación adicional al respecto en Palacio 20156) por las características de los recursos humanos implicados (carecen en su totalidad de dedicación plena al proyecto), que imposibilitan un plan de reuniones tan estricto como el modelado original requiere. La plasmación modélica de SCP utiliza los siguientes roles:

|  |  |  |
| --- | --- | --- |
| **ROL** | **ACTOR** | **CARACTERÍSTICAS** |
| Scrum Master | Desarrollador jefe | Facilita la aplicación de SCRUM y gestionar  cambios. Coordina con Team. |
| Product Owner | Supervisor jefe | Representa a los *stakeholders* (interesados externos o internos). |
| Team | Desarrolladores | Realizan los artefactos determinados en las  Reuniones. |

Las reuniones en SCRUM siguen una pauta poco flexible, y por ello SCP ha simplificado su jerarquía reduciéndolas a una por ciclo (***Sprint Planning Meeting***), que, por razones de premura y organizativas, suelen realizarse una vez cada quince días, en vez de con periodicidad mensual, como suele ser la tónica común.

El ***Product Backlog***, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar, viene en SCP predeterminado desde el inicio por la matriz aceptada de requerimientos, y, como determina el método, se altera en función de los resultados del ***Sprint Backlog*** para adaptar el conjunto de características a la evolución de las necesidades de RMTI, a las especificidades del trabajo de los diferentes actores de SCP que también son cambiantes según se lleve a cabo el desempeño de los roles concretos y a los límites temporales y presupuestarios.

## Entregables del proyecto

A continuación, se indican y describen cada uno de los *artefactos* que han sido, son o serán generados y utilizados por el proyecto y que constituyen los ***entregables***. Esta lista describe la configuración RUP desde la perspectiva de artefactos, adaptada al método SCRUM modificado propio de SCP.

Es preciso destacar, no obstante, que, de acuerdo a la filosofía RUP (y de cualquier procedimiento iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del tiempo de desarrollo, con lo cual sólo al término de un proceso es posible disponer de una versión definitiva y completa de ellos. Sin embargo, el resultado de cada iteración, el ***Sprint Backlog*** y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos.

1. J. Palacio. *Gestión de proyectos Scrum Manager (Scrum Manager I y II). Versión 2.5.1*. España, 2015.

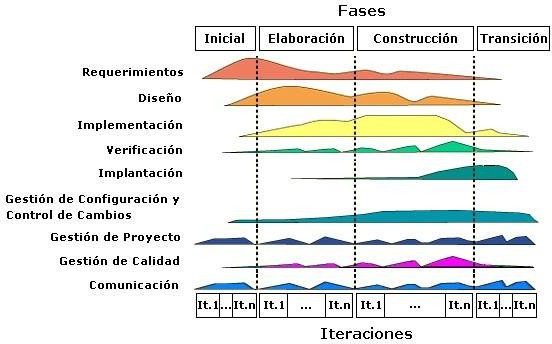


Ilustración 4. Disciplinas y fases RUP en SCP

La estructura de procesos y secuencias que el principio RUP impone agrupa las tareas que se realizan en ***disciplinas*** (en el diagrama, los elementos que figuran a la izquierda), y la flecha temporal se basa en la realización de ***iteraciones*** de las tareas reunidas en ***fases***. En SCP, y puesto que, como ha quedado establecido ya desde [1,](#_bookmark0) el *constructo* lógico del proyecto se apoya en una planificación *Rational Unified*, a la vez que sigue principios SCRUM, LP y XP, los entregables se ciñen a lo estipulado en el modelo, de manera que tanto RMTI como el propio *Team* y los diferentes *Managers* puedan realizar un seguimiento detallado que permita aplicar los principios LP sobre el aprendizaje, la rapidez de reacción y la integridad. El diagrama representa como una línea gráfica con picos y valles la intensidad de las actividades prevista desde el principio para el despliegue de SCP. Los entregables se agrupan tal como figura en la siguiente tabla.

|  |  |  |  |
| --- | --- | --- | --- |
| **DISCIPLINA** | **ENTREGABLE** | **PREVISIÓN** | **OBSERVACIONES** |
| Gestión del proyecto | PDS | Noviembre 2020 | Incluido en PDSDT. En [3](#_bookmark8) |
| Diseño (Modelado de  negocio) | Casos de uso agrupados | Noviembre 2020 | Incluido en PDSDT. En [4](#_bookmark11) |
| Diseño (Modelado de  negocio) | Modelo de datos | Noviembre 2020 | Parcialmente incluido en PDSDT  (modelos físico y lógico). En [6](#_bookmark18) y [7](#_bookmark19) |
| Requerimientos (Requisitos) | Restricciones y suposiciones | Noviembre 2020 | Incluido en PDSDT. En [2](#_bookmark2) |
| Requerimientos (Requisitos) | Matriz de requisitos | Noviembre 2020 | Incluido en PDSDT |
| Implementación | Modelo de implementación | Noviembre 2020 | Incluido en PDSDT. En [8](#_bookmark20) |
| Verificación | Modelo de control de errores | Noviembre 2020 | Incluido en PDSDT. En [10](#_bookmark25) |
| Implantación | Sistema | Pendiente | Instalado en servidor |
| Implantación | Modelos de interfaces | Noviembre 2020 | Incluido en PDSDT. En [10](#_bookmark25) |
| Implantación | Manuales de usuario | Pendiente | No incluidos en PDSDT |
| Gestión de calidad | Matriz de PMBOK | Noviembre 2020 | Incluido en PDSDT. En [13](#_bookmark30) |
| Comunicación | Publicación de artículos y organización de  *workshop* | Pendiente | No incluidos en PDSDT |

Nótese que no todas las disciplinas implican la existencia de entregables y que, por la propia naturaleza del proyecto SCP, algunos dependen de la participación de RMTI.

# Organización del proyecto

## Recursos humanos

La estructura organizativa de SCP depende de dos principales entidades S&S y MTI cabe consultar la pertinente disposición del personal responsable del seguimiento así mismo los detalles sobre actores “externos” que intervengan en el procesado analítico de los datos consolidados en SCP. El perfil del personal asignado se agrupa en **equipos** de la siguiente manera.

## Supervisión

*Nuria Rodríguez Ortega*. NRO. Cargo: Investigadora principal. Tareas: Coordinación y di- rección. Especialidades: Historia del Arte Digital, Humanidades Digitales, Análisis relacio- nal.

*María Luisa Bellido Gant*. MLBG. Cargo: Investigadora principal. Tareas: Coordinación y dirección. Especialidades: Historia del Arte Digital, Humanidades Digitales, Datos, Coordi- nación.

## Desarrollo y administración del sistema

*Víctor Paúl Téllez Pérez*. VPTP. Cargo: Investigador. Tareas: Desarrollo de aplicaciones. Especialidades: Conocimientos Informáticos, Humanidades Digitales, Informática, Desarrollo.

## Revisión

*Juan Gabriel Bermúdez Hernández*. JGBH. Cargo: Investigador Informático. Ing. De sistemas Tareas: Coordinación de equipo. Datos. Especialidades: Director informático, Analista de datos, Coordinación.

## Grabación y verificación

*María Casas González*. MCG. Cargo: Investigadora colaboradora. Becaria. Tareas: Datos. Especialidades: Historia del Arte Digital, Humanidades Digitales, Datos.

*Miguel Ángel Sánchez Badillo*. MASB. Cargo: Investigador colaborador. Becario. Tareas: Datos. Especialidades: Historia del Arte Digital, Humanidades Digitales, Datos.

Dadas las específicas condiciones de SCP, su ausencia de objetivo comercial o lucrativo y su vocación científica, el **perfil** profesional y/o académico de los diferentes miembros de los grupos se adecua a las necesidades propias de la ejecución de las tareas que les son asignadas.

|  |  |  |
| --- | --- | --- |
| **EQUIPO** | **PERFIL** | **ESPECIALIDADES** |
| Supervisión |  |  |
| Desarrollo | Ingeniero de sistemas |  |
| Administración del sistema |  | SW y/o Telecomunicaciones |
| Revisión |  | . |
| Grabación | Ingeniero de sistemas o Ingeniero de computación (especialidad en gestión de datos). |  |
| Verificación |  | . |

Si bien cae fuera del alcance de PDSDT abundar sobre la existencia de un mecanismo de relación laboral como vínculo entre los distintos miembros de los diferentes grupos y el proyecto, sí que cabe señalar que, los *stakeholder* carecen de remuneración dineraria por su participación en SCP y pueden solamente establecer una dedicación parcial, mientras que los restantes perfiles sí acceden a retribuciones mediante contratos especiales con S&S. Ello redunda, evidentemente, en beneficio de la intensidad de la dedicación de dichos perfiles, no obstante lo cual las limitaciones presupuestarias derivadas de RMTI (en este caso, el propio MTI) no permiten ampliar el horario de trabajo, pero se deben de realizar las entregas a tiempo y forma establecido en el plan de trabajo. A los efectos organizativos dentro de SCP, la **matriz de atribución de roles, actores, equipos y dedicación** puede consultarse en la siguiente tabla.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EQUIPO** | **ACTOR** | **ROL** | **DEDICACIÓN** | **REMUNERACIÓN** | **ASIGNACIÓN** |
| Supervisión | Supervisor | Product Owner | Parcial | No |  |
| Desarrollo | Desarrollador | Team | Total | Si | VPTP |
| Administración del  sistema | Administrador | Team Manager | Parcial | No |  |
| Revisión | Revisor | Team | Total | Sí | JGBH |
| Grabación | Grabador verificador | Team | 20 | Sí |  |
| Verificación | Grabador verificador | Team | 20 | Sí |  |

## Jerarquía RRHH

La fluidez del funcionamiento de los diferentes equipos depende enteramente de una bien planificada estructura jerárquica que permita a unos actores realizar tareas que perfeccionen las llevadas a cabo por otros. De esa forma se refuerza la cohesión entre los miembros, se mantiene la integridad conceptual y se facilita la eliminación de tareas inútiles como comunicaciones en bucle o mediante escalafones complejos. El entramado de rangos y comunicaciones se recogen en la siguiente tabla y el esquema adjunto. Se incluye el actor “no humano” servidor por hallarse implicado un actor perteneciente a los equipos RRHH.

|  |  |  |
| --- | --- | --- |
| **ACTOR** | **→ ACCIÓN →** | **ACTOR** |
| Administrador | Informa de cambios de estado en el servicio | Supervisor |
| Administrador | Informa de cambios de estado en el servicio | Grabador verificador |
| Administrador | Informa de cambios de estado en el servicio | Revisor |
| Administrador | Informa de cambios de estado en el servicio | Desarrollador |
| Administrador | Monitoriza estado del servicio | Servidor |
| Administrador | Realiza optimizaciones específicas | Servidor |
| Administrador | Realiza optimizaciones periódicas | Servidor |
| Desarrollador | Informa sobre novedades en funcionamiento de SCP | Grabador verificador |
| Desarrollador | Informa sobre novedades en funcionamiento de SCP | Revisor |
| Desarrollador | Informa sobre novedades en funcionamiento de SCP | Supervisor |
| Desarrollador | Informa sobre novedades en funcionamiento de SCP | Administrador |
| Desarrollador | Solicita ajustes u optimizaciones del servicio | Administrador |
| Desarrollador | Valida calidad de manuales de usuario | Revisor |
| Grabador verificador | Informa de errores en programación | Desarrollador |
| Grabador verificador | Solicita modificaciones en valores de referencia | Revisor |
| Grabador verificador | Verifica y graba | Servidor |
| Revisor | Corrige errores de verificación y/o grabación | Grabador verificador |
| Revisor | Elabora y entrega manuales de usuario | Grabador verificador |
| Revisor | Elabora y entrega manuales de usuario | Supervisor |
| Revisor | Informa de errores en programación | Desarrollador |
| Revisor | Informa de errores en verificación y/o grabación | Grabador verificador |
| Revisor | Informa de errores en verificación y/o grabación | Supervisor |
| Revisor | Pide autorización para modificar valores de referencia | Supervisor |
| Supervisor | Autoriza modificar valores de referencia | Revisor |
| Supervisor | Informa de errores en programación | Desarrollador |
| Supervisor | Pide modificaciones en funcionalidad de SCP | Desarrollador |
| Supervisor | Solicita optimizaciones en prestaciones | Administrador |
| Supervisor | Valida calidad final de datos | Revisor |

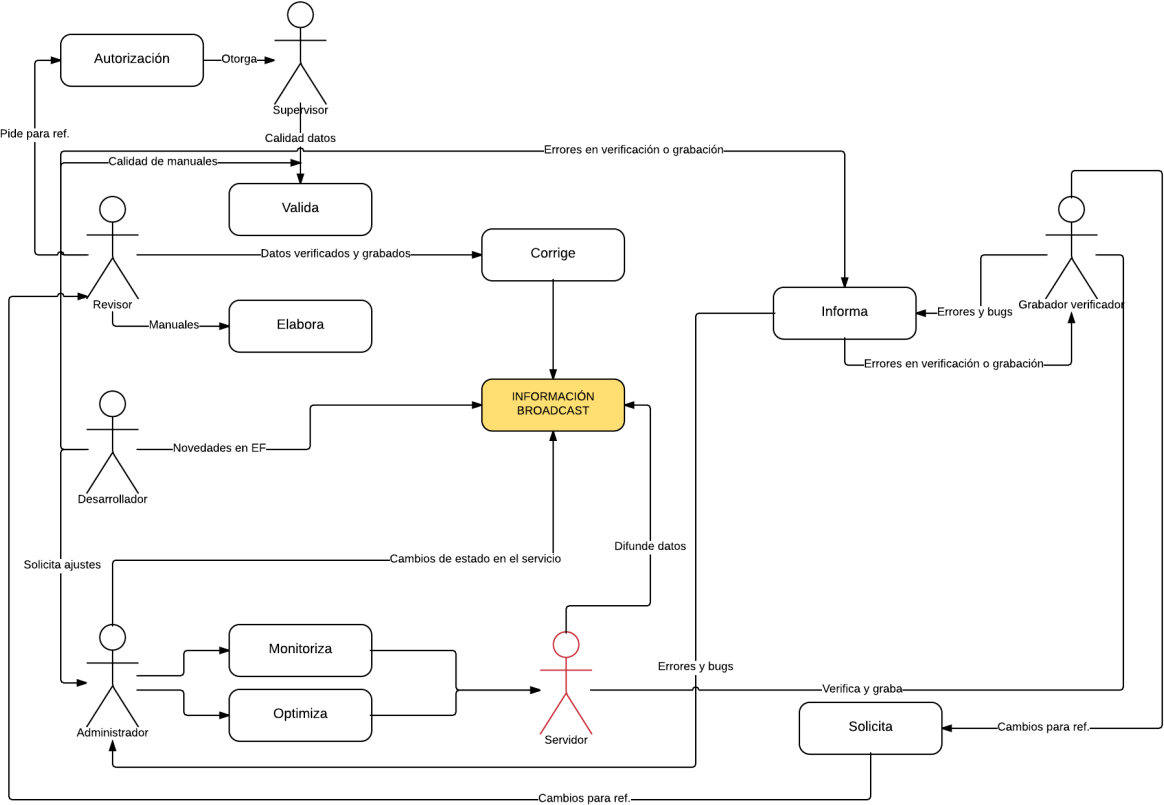


Ilustración 5. Interacciones entre actores en SCP

El elemento en color amarillo del diagrama (“Información *broadcast*”) representa las interacciones en las que un único actor realiza una acción o entrega un artefacto cuyos destinatarios son la totalidad de los integrantes de los equipos de trabajo.

# Gestión de procesos en el proyecto

## Pendiente

# 5 Infraestructura

La modalidad de trabajo cliente/servidor, adoptada como solución funcional para SCP tal como se describe [en 8,](#_bookmark20) implica el uso del siguiente HW por parte de los diferentes actores del sistema.

## Lado cliente

Equipo informático con cualquier sistema operativo capaz de soportar conexiones a Internet (OSX, Windows, Linux,…). Mínimo 4 Gb RAM. EL espacio de almacenamiento local es irrelevante.

Navegador basado en Webkit o Blink (Google Chrome 40 ó superior, Safari, Firefox… o bien Microsoft Edge).

Conexión de banda ancha a la red (xDSL, FTTH, FTTC…).

Velocidad de red mínima 5 MB.

## Lado servidor

Equipo de marca indiferente, con resiliencia eléctrica y almacenamiento.

RAID. Sistema operativo Indiferente, con acceso *root*.

SGBD SQLServer

Conexión de alta velocidad a Internet. Dominio Internet de tipo “.com”

**Alternativa**

Equipo informático con cualquier sistema operativo capaz de soportar conexiones a Internet (OSX, Windows, Linux,…). Mínimo 4 Gb RAM. EL espacio de almacenamiento local es irrelevante.

6 GB de ram

512 De almacenamiento.

Conexión de banda ancha a la red (xDSL, FTTH, FTTC…).

Velocidad de red mínima 10 MB.

De acuerdo con los requerimientos solicitados por S&S, que se expresan en [8](#_bookmark20), y para el correcto funcionamiento del sistema SCP, *Product Owner*/*Supervisor* esta en derecho de declinar esta propuesta y está obligado a realizar una contra propuesta que se ajuste a la posibilidad de adquisición y la modalidad que mejor convenga según su presupuesto.

# Estructura física de la información

Según se expresa en el análisis de requisitos y en la arquitectura consecuentemente adoptada (ver [8](#_bookmark20)), existirá una capa física de estructura de los datos, que se corresponde con la configuración de tablas en el SGBD empleado, y otra lógica, que ajusta las peculiaridades de los requerimientos a lo determinado en la anterior. En realidad, la capa lógica hace las veces de interfaz entre los requerimientos y el sistema implementado en SCP.

La organización responde a las suposiciones basadas en la información brindada con anterioridad y pequeños flujos de trabajos que se han observación en algunos puntos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tabla: Personas** | | | | | |
| **Campo** | **Tipo** | **Nulos** | **Clave** | **Por defecto** | **Extra** |
| persona\_id | integer |  | PRI |  | auto\_increment |
| first\_name | varchar(50) |  | IND | NULL |  |
| second\_name | varchar(50) |  | IND | NULL |  |
| first\_last\_name | varchar(50) |  |  | NULL |  |
| second\_last\_name | varchar(50) |  |  | NULL |  |
| Identificacion | varchar(50) |  |  |  |  |
| email | varchar(100) | YES |  | NULL |  |
| photo | varchar(max) | YES |  | NULL |  |
| birth\_date | date | YES |  | NULL |  |
| gender | int | YES |  | NULL |  |
| cellphone | int | YES |  | NULL |  |
| number\_phone | Int | YES |  | NULL |  |
| country\_phone\_code | varchar(10) | YES |  | NULL |  |
| **Índices** |  |  |  |  |  |
| **Nombre** | **Tipo** | **Campos** |  |  |  |
| PRIMARY | PRIMARY | persona\_id |  |  |  |
| **Tabla: Vehículos** | | | | | |
| **Campo** | **Tipo** | **Nulos** | **Clave** | **Por defecto** | **Extra** |
| Vehiculo\_id | int |  | PRI | NULL | auto\_increment |
| Vehiculo\_color | varchar(20) | YES |  | NULL |  |
| Num\_chasis | varchar(50) |  | UNIQUE | 0000-00-00 00:00:00 |  |
| Num\_motor | varchar(50) |  | UNIQUE | 0000-00-00 00:00:00 |  |
| Num\_placa | varchar(50) |  | UNIQUE | NULL |  |
| Toneladas\_Max | double |  |  | NULL |  |
| Ejes | int |  |  | NULL |  |
| Marca | varchar(50) | YES |  | NULL |  |
| Modelo | varchar(50) | YES |  | NULL |  |
| Num\_circulacion | varchar(50) |  |  | NULL |  |
| **Índices** |  |  |
| **Nombre** | **Tipo** | **Campos** |
| PRIMARY | PRIMARY | Vehiculo\_id |
| UNIQUE | UNIQUE | Num\_chasis |
| UNIQUE | UNIQUE | Num\_motor |
| UNIQUE | UNIQUE | Num\_placa |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Empresas Transporte** | | | | | | | | | |
| **Campo** | | | **Tipo** | **Nulos** | | | **Clave** | **Por defecto** | **Extra** |
| Empresa\_transp\_id | | | int |  | | | PRI |  | auto\_increment |
| nombre | | | varchar(200) |  | | |  |  |  |
| Cellphone | | | int |  | | |  | NULL |  |
| country\_phone\_code | | | varchar(10) | YES | | |  | NULL |  |
| Persona\_id | | | int |  | | | FK persona |  |  |
| **Índices Nombre** | **Tipo** | | | **Campos** |
| PRIMARY | PRIMARY | | | Empresa\_transp\_id |
| FORENGKEY | FORENGKEY | | | Persona\_id |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Conductores** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Conductor\_id | | | int | |  | PRI | |  | auto\_increment |
| Person\_id | | | int | |  | FK persona | |  |  |
| Num\_licencia | | | Varchar(50) | |  |  | | NULL |  |
| Categorías\_vehiculos | | | Varchar(50) | |  |  | | NULL |  |
| Nacionalidad | | | Varchar(50) | |  |  | | NULL |  |
| Poliza\_id | | | int | |  | FK poliza | |  |  |
| Empresa\_transp\_id | | | int | |  |  | |  |  |
| **Índices Nombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Conductor\_id | | |
| FORENGKEY | FORENGKEY | | Persona\_id | | |
| FORENGKEY | FORENGKEY | | Poliza\_id | | |
| FORENGKEY | FORENGKEY | | Empresa\_transp\_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Pólizas** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Poliza\_id | | | int | |  | PRI | |  | auto\_increment |
| Num\_Poliza | | | varchar(30) | |  | UNIQUE | |  |  |
| Num\_seguro | | | varchar(30) | |  | UNIQUE | |  |  |
| Compania | | | varchar(50) | |  |  | | NULL |  |
| Fecha\_vencimiento | | | Datetime | |  |  | | NULL |  |
| Tipo\_seguro\_id | | | Int | |  | FK tiposeguro | |  |  |
| **Índices**  **Nombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Poliza\_id | | |
| UNIQUE | UNIQUE | | Num\_Poliza | | |
| UNIQUE | UNIQUE | | Num\_seguro | | |
| FORENGKEY | FORENGKEY | | Tipo\_seguro\_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Tipos de seguro** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Tipo\_seguro\_id | | | int | |  | PRI | |  | auto\_increment |
| Tipo\_seguro | | | varchar(50) | |  |  | | null |  |
| **Índices**  **Nombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Tipo\_seguro\_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Puntos de control** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Punto\_id | | | int | |  | PRI | |  | auto\_increment |
| Nombre\_Punto | | | varchar(50) | |  | UNIQUE | |  |  |
| Bascula\_id | | | int | |  | FK Basculas | |  |  |
| Ubicacion | | | varchar(250) | |  |  | |  |  |
| cellphone | | | int | | YES |  | | NULL |  |
| number\_phone | | | Int | | YES |  | | NULL |  |
| Persona\_id | | | int | |  | FK persona | |  |  |
| **Índices**  **Nombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Punto\_id | | |
| UNIQUE | UNIQUE | | Nombre\_Punto | | |
| FORENGKEY | FORENGKEY | | Persona\_id | | |
| FORENGKEY | FORENGKEY | | Bascula\_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Basculas** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Bascula\_id | | | int | |  | PRI | |  | auto\_increment |
| Modelo | | | varchar(50) | |  | UNIQUE | |  |  |
| Capacidad | | | varchar(250) | |  |  | |  |  |
| Unidad de medida | | | varchar(5) | | YES |  | | NULL |  |
| Marca | | | varchar(50) | | YES |  | | NULL |  |
| Foto | | | varchar(250) | |  |  | |  |  |
| **Índices**  **Nombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Bascula\_id | | |
| UNIQUE | UNIQUE | | Modelo | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Países** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Pais\_id | | | int | |  | PRI | |  | auto\_increment |
| Nombre | | | varchar(50) | |  | UNIQUE | |  |  |
| **ÍndicesNombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Pais\_id | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Departamentos** | | | | | | | | |
| **Campo** | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Departamento\_id | | int | |  | PRI | |  | auto\_increment |
| Nombre | | varchar(50) | |  | UNIQUE | |  |  |
| Pais\_id | | int | |  | FK paises | |  |  |
| **ÍndicesNombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Departamento \_id | | |
| FORENGKEY | FORENGKEY | | Pais \_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Municipios** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Municipio\_id | | | int | |  | PRI | |  | auto\_increment |
| Nombre | | | varchar(50) | |  | UNIQUE | |  |  |
| Departamento\_id | | | int | |  | FK Departamento | |  |  |
| **ÍndicesNombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Municipio \_id | | |
| FORENGKEY | FORENGKEY | | Departamento \_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Usuarios.** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Usuario\_id | | | int | |  | PRI | |  | auto\_increment |
| Persona\_id | | | int | |  | FK persona | |  |  |
| Nivel\_id | | | int | |  | FK niveles | |  |  |
| Usuario | | | Varchar(50) | |  |  | |  |  |
| Contraseña | | | Varchar(15) | |  |  | |  |  |
| **ÍndicesNombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Usuario\_id | | |
| FORENGKEY | FORENGKEY | | Persona\_id | | |
| FORENGKEY | FORENGKEY | | Nivel\_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Niveles.** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Nivel\_id | | | int | |  | PRI | |  | auto\_increment |
| Nivel | | | int | |  | FK persona | |  |  |
| Nombre | | | int | |  | FK niveles | |  |  |
| Descripcion | | | Varchar(50) | |  |  | |  |  |
| **ÍndicesNombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Usuario\_id | | |
| FORENGKEY | FORENGKEY | | Persona\_id | | |
| FORENGKEY | FORENGKEY | | Nivel\_id | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla: Pesadas.** | | | | | | | | | |
| **Campo** | | | **Tipo** | | **Nulos** | **Clave** | | **Por defecto** | **Extra** |
| Pesada\_id | | | int | |  | PRI | |  | auto\_increment |
| Punto\_id | | | int | |  | FK punto\_control | |  |  |
| Usuario\_id | | | int | |  | FK ususario | |  |  |
| Conductor\_id | | | int | |  | FK conductore | |  |  |
| Vehiculo\_id | | | int | |  | FK Vehiculo | |  |  |
| Peso | | | Double | |  |  | |  | 0 |
| Tara | | | Double | |  |  | |  | 0 |
| Destino | | | Varchar(250) | |  |  | |  |  |
| **ÍndicesNombre** | **Tipo** | | **Campos** | | |
| PRIMARY | PRIMARY | | Pesada \_id | | |
| FORENGKEY | FORENGKEY | | Punto \_id | | |
| FORENGKEY | FORENGKEY | | Usuario\_id | | |
| FORENGKEY | FORENGKEY | | Conductor\_id | | |
| FORENGKEY | FORENGKEY | | Vehiculo\_id | | |
|  |  | |  | | |

No se incluyen las tablas que dependen de la estructura interna de funcionamiento de SQL y que

no están relacionadas con la operatoria de SCP.

# Estructura lógica de la información

A continuación, se suministra la información básica sobre la estructura lógica del sistema de información de SCP. En PDSDT se usa el término *sistema de información* (en adelante, SI) con el mismo sentido descrito por la Encyclopaedia Britannica: "*an integrated set of components for collecting, storing, and processing data and for delivering information, knowledge, and digital products*".

## Elementos

Por razones de terminología propia de SCP, en PDSDT no se emplea el término *entidad* en el sentido propio de una descripción de un modelo de datos, porque podría plantear confusiones con uno de los modelos de registro utilizados. En su lugar, se usa ***elemento***.

El ámbito y el alcance, así como su nomenclatura, de los elementos del SI se delimitan de acuerdo a las siguientes especificaciones:

* + - **Empresa de Transporte EMPT**. Se entiende como empresa de transporte (según DRAE) "cualquier corporación, compañía, institución, etc., tomada como persona jurídica que se encarga de la logística de cargas". En este caso será considerado como tal cualquier organismo con entidad legal que se dedique al rubro de transporté de cargas pesadas asi se utilizará en el contexto de Sistema de control de Peso, bien como un origen de datos HTML, a partir de cuyos resultados el sistema explora la Red en búsqueda de información sobre pesos y cargas de diferentes medios de transportes que transiten las diferentes vías terrestres al pasar en un punto control así mismo el medio de carga deberá estar vinculado a un conductor con una empresa se transporte.
    - **Personas PER**. Se entiende como persona en el contexto DT a quienes interactúan como actores de una acción en el sistema por ejemplo (usuario, Conductor, responsable de punto de control).
    - **Vehículos VEH**. Se entiende como vehículo en el contexto DT cualquier medio de transporte que logre movilizar a un apersona u objeto a un destino distante. Su uso "natural" es para transportar cargas.
    - **Conductor CON**. Se entiende como conductor a toda persona que maneja el mecanismo de dirección de un vehículo o, simplemente, los mandos cuando carece de la misma.  en términos mercantiles, comerciales o de negocio.
* **Póliza** **POL**. La póliza de seguro es un contrato entre una persona u entidad, como asegurado y la compañía de seguros, que establece los derechos y obligaciones de ambos, en relación al seguro contratado. Relaciones entre elementos.
* **Tipos de Seguros TS**. Se trata de tipos de seguro a la multiplicidad de riesgos que pueden ser tratados mediante un contrato de seguro.
* **Punto de Control PDC**. Los puntos de control son definidos por EL CODEX como un punto crítico de control (PCC) como "una etapa donde se puede aplicar un control y que sea esencial para evitar o eliminar un peligro a la inocuidad del alimento o para reducirlo a un nivel aceptable". En el contexto de SCP un punto de control es un lugar sobre la ruta de transito en el cual se chequeará la el peso del vehículo al cual se le aplicará control.
* **Basculas BSC**. Se define como Instrumento para medir pesos, generalmente grandes, que consiste en una plataforma donde se coloca lo que se quiere pesar, un sistema de palancas que transmite el peso a un brazo que se equilibra con una pesa, y un indicador que marca el peso. Este sera el principal hito del sistema que se encarga de recopilar la principal información del sistema de información.
* **Usuarios USR**. Este elemento tiene el derecho de usar de una cosa ajena con unas limitaciones determinadas.
* **Niveles LVL.** Los niveles brindan acceso a determinadas areas de un SI.
* **Pesadas PSD.** Son iteraciones del proceso de pesaje de los vehículos en el cual se recopila la mayor parte de la información que sera posteriormente analizada para realizar una toma de decisión.

Los datos obtenidos se atienen al modelo expresado en las imágenes diagramáticos E/R, que se muestran [en 14.](#_bookmark33) En esencia, el SI agrupa la información de acuerdo al siguiente criterio (el símbolo → significa *se relaciona con*):

## Empresa de transporte

* + - Empresa de transporte → Conductor:
    - Empresa de transporte [1→∞]. Vehículos.
    - Empresa de transporte [1→∞]. Pesadas.
    - Empresa de transporte [1→∞]. Puntos de control.
    - Empresa de transporte → Personas:
    - Empresa de transporte [1→1]. La persona encargada.
    - Empresa de transporte [1→1]. Localidad u nacionalidad.

## Vehículos

* + - Vehículos → Conductor:
    - Vehículos [1→1]. Conductor.
    - Vehículos [1→1]. Póliza.
    - Vehículos [1→∞]. Pesadas.

## Personas

* + - Personas → Empresas de Transporte:
    - Personas [1→1]. Representante de empresa
    - Personas → Puntos de control:
    - Personas [1→∞]. Usuarios
    - Personas [1→1]. Representante de Punto de control.
    - Personas →Conductor:
    - Personas [1→1]. Conductor.
    - Dependencia [1→1]. La persona (derecha) es la principal en una relación descendente de dependencia de la persona.

## Polizas

* + - Polizas → Tipos de seguros.
    - Polizas [1→1]. Tipo de seguro que ejerce la poliza.

## Conductor

* + - Conductor → Conductor.
    - Conductor [1→∞]. Polizas.

## Puntos de control

* + - Puntos de control → Basculas
    - Puntos de control [1→1]. Basculas.
    - Puntos de control → persona.
    - Puntos de control [1→1]. Persona.
    - Puntos de control [1→1]. Municipio.
    - Puntos de control [1→∞]. Pesadas.

## Usuarios

* + - Usuarios → persona
    - Puntos de control [1→1]. persona.
    - Puntos de control [1→1]. Nivel.

# Diseño de la arquitectura del sistema

Desde su fase de diseño, y de acuerdo con el análisis de ingeniería de requisitos realizado a partir de las especificaciones planteadas por EX, EF se concibió como un aplicativo, adherido al para- digma **cliente-servidor** siguiendo un modelo **SaaS** —si bien, obviamente, sin los condicionantes económicos—, realizado sobre una estructura a modo de FW, entendiéndolo a la manera en que DocForge10 lo define: "un conjunto de código fuente o bibliotecas que proporcionan una funcionali- dad común a toda una clase de aplicaciones".

## Descripción general

Por facilidad de implementación, PHP es el lenguaje de desarrollo elegido, sobre una plataforma Unix/Linux. Durante la fase de diseño se realizaron pruebas con diversas distribuciones *framework* bien conocidas y utilizadas para desarrollos RAD, como *CakePHP*, *CodeIgniter* o *Laravel*. Sin em- bargo, pronto surgió (y, evidentemente se afianzó) la posibilidad de utilizar un conocido CMS como **WordPress** para esa finalidad.

La prensa *online* especializada anda ciertamente dividida en cuanto a la posibilidad de usar WP como FW, con posturas a menudo encontradas y, hasta cierto punto, escasamente razonadas en numerosas ocasiones. Sin embargo es evidente que WP ofrece todo un complejo API, basado en la exposición de sus rutinas mediante mecanismos de *hooking* y filtrado, y facilita, como parte de su diseño interno, la realización tareas y operaciones tan esenciales como:

* + - Completa y razonablemente segura gestión de usuarios.
    - Excelente motor de plantillas, basado en la abstracción de las mismas en un conjunto de- nominado "tema", que permite su explotación coordinada y estética y funcionalmente co- herente.
    - Mecanismo sólido para el registro de errores.
    - Posibilidad de localización e internacionalización completa mediante el uso de *gettext*

según principios i18n.

* + - Completo API para la gestión de formularios y la validación de datos.
    - Facilidades para la administración de envío y recepción de archivos, con inserción de me- tadatos que permiten su localización e identificación.
    - Consistente grupo de rutinas y clases para la manipulación de imágenes.
    - Gestión fácil de URLs amigables para los motores de búsqueda y SEO.
    - Implementación de un completo *subset* de procedimientos XML-RPC.
    - Uso de *hooks*, clases abstractas, prototipos y mecanismos de tipo *plugin* que permiten ampliar o modificar la funcionalidad básica.

1. Multiple (wiki). "Web application framework". Docforge. Consultado 2015-01-19.
   * Sistema semiautomatizado de ejecución planificada de código.
   * Robusto y eficiente motor de *caching*, esencial para obtener buenos resultados en cuanto a tiempos de operación con el equipo de trabajo.

Adicionalmente, al tratarse de un FW acogido a la distribución bajo licencia GPLv2 (o posterior), resulta fácilmente mantenible, reuniendo todo el esfuerzo conjunto de una amplísima comunidad de desarrolladores y usuarios, de manera que la observación colectiva permite una rápida actua- ción frente a cualquier vulnerabilidad detectada. Según el buscador Technorati, el 48% de sus *top 100 blogs* usan dicho software para su funcionamiento, y 74.652.825 sedes en la Red lo emplean como su sistema principal. Se han realizado 96 versiones hasta la fecha, y su núcleo cuenta con más de 40 traducciones.

Por todo lo expresado, la decisión sobre el FW a emplear resultó claramente favorable a WP. Y desde el principio, de acuerdo con lo dicho sobre el mantenimiento de un aceptable nivel de segu- ridad, se trató de NO EMPLEAR NINGÚN MECANISMO QUE INTERFIRIESE CON LA

ESTRUCTURA INTERNA Y LA FUNCIONALIDAD NATIVA DE WP, de manera que fuese sencillo y exento de trabajo adicional o de riesgos (perder lo realizado, abrir "puertas traseras" o recuperar vulnerabilidades que se creían eliminadas) actualizar el núcleo del sistema sin mermar por ello su rendimiento y las garantías de aislamiento de peligros externos.

En [14](#_bookmark33) se muestran imágenes que ilustran adecuadamente la estructura funcional en capas de WP.

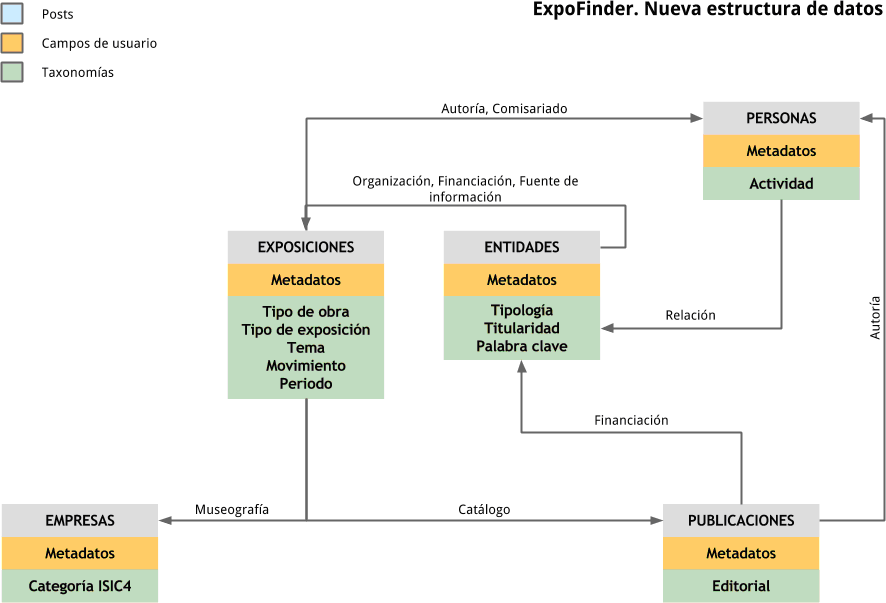
A partir de lo recogido en [7,](#_bookmark19) es posible expresar en forma simplificada la estructura del SI EF en la imagen que se muestra a continuación, a modo de diagrama relacional entre elementos.

Ilustración 6. Modelo simplificado SI EF

A partir de ella, se pueden establecer las relaciones entre los diversos dominios de información presentes en los requisitos, tal y como se exhiben en las siguientes tablas.

|  |  |  |
| --- | --- | --- |
| **ELEMENTO** | **OBTIENE** | **DE** |
| Exposiciones | Fuente de información | Entidades |
| Exposiciones | Organización | Entidades |
| Exposiciones | Financiación | Entidades |
| Exposiciones | Autoría | Personas |
| Exposiciones | Comisariado | Personas |
| Exposiciones | Museografía | Empresas |
| Exposiciones | Catálogo | Publicaciones |
| Personas | Relación | Entidades |
| Publicaciones | Autoría | Personas |
| Publicaciones | Financiación | Entidades |

WP está excelentemente diseñado, y jugando con su estructura ya se han realizado al menos dos proyectos de envergadura para transformarlo en un modelo lo más universal posible que permita realizar desarrollos sobre el. Aunque, en realidad, y de acuerdo con lo expresado por Tom McFar- lin11 en su conocida página “tuts+”, es más una *foundation* (cimiento) que un *framework*.

Y tal vez no ande escaso de razón: un FW consiste en un conjunto de convenciones -como dónde deben guardarse los archivos- así como de bibliotecas y herramientas -pongamos por caso, una capa de abstracción de base de datos- que nos permiten fácilmente comenzar a trabajar en una aplicación. En resumen, proporciona los medios por los cuales un programa puede ser construido partiendo de cero, a partir del esquema de base de datos hasta su *front-end*. Sin embargo, una *foundation* permite extender una aplicación ya existente. WP tiene sus propios mecanismos inter- nos bien definidos, y la *foundation* simplemente amplía su funcionamiento o lo aprovecha en su propio beneficio. Y eso es lo que EF realiza con WP: aprovecha la base de datos predefinida, las APIs disponibles, y el sistema de plantillas para la visualización de datos para construir soluciones usando un aplicativo que ya está plenamente funcional.

Existen en la Red proyectos que basan su funcionamiento en implementar un mecanismo MVC (*Model, View, Controller*) sobre WP, que indudablemente hacen gala de una gran calidad y que usan una metodología perfectamente adaptada a los principios más adecuados de ingeniería del software. Pero fueron descartados para EF porque hacían excesivamente complejo el desarrollo y obligaban a “forzar la máquina”, teniendo que realizar adaptaciones y modificaciones en el núcleo principal (*core*) de WP, lo que suponía renunciar a algunas de sus ventajas:

* + Una base de datos con un esquema organizativo flexible y muy sólido.

1. T. McFarlin. *Using WordPress for Web Application Development: An Introduction*. 2013. <http://code.tutsplus.com/tutorials/using->

wordpress-for-web-application-development-an-introduction--wp-33828. Consultado 20-1-2015.

* + Una capa de aplicación principal con numerosos *hooks* que permiten aprovechar al máxi- mo su funcionalidad.
  + La facilidad para gestionar tareas en las caras de servidor y cliente, y asumiendo roles de administrador o usuario.

Los elementos y puntos fuertes de WP como FW que a continuación se relacionan se incorporan a EF como núcleo fundamental sobre el que construir la arquitectura SW.

* + Comunidad de desarrollo, soporte y documentación. WP dispone de una gran comunidad y un foro de soporte activo. Ya se trate de un problema con la instalación, la configuración, desarrollo o cualquier otra cosa, normalmente cualquier problema se resuelva satisfacto- riamente con la ayuda de la comunidad de apoyo.
  + Gestión de usuarios. La seguridad no es, en el caso de aplicaciones abiertas a su uso en Internet, un problema menor. WP tiene un módulo de gestión de usuarios excelente, que se encarga de toda la funcionalidad, tal como el registro de usuarios y de inicio de sesión, gestión de roles, asignación de capacidades a diferentes roles y la creación de nuevas so- luciones.
  + Pantallas de administración. WP ofrece todo un interfaz, que permite crear páginas pro- pias de configuración personalizada para poder hacer frente a una amplia serie de requisi- tos.
  + Funcionamiento CRUD. Supone un enorme ahorro de “energía programadora” y de tiem- po contar con un muy contratado mecanismo que controla absolutamente todo el proceso de dar de alta, de baja, modificar o leer cualquier registro o conjunto de registros almace- nados.
  + Completísimo juego de funciones internas perfectamente aprovechables. Incluyen desde la gestión de la internacionalización (mediante mecanismos *gettext*) a través de un avan- zado control i18n hasta la “desinfección” preventiva de textos, bloqueando de esa forma posibles ataques de tipo *SQL injection* y variantes.
  + Administración de medios. WP se ocupa de la carga de archivos y la gestión de los me- dios audiovisuales, incluyendo su securización.
  + Extensibilidad y escalabilidad. Desde el punto de vista del desarrollo, extensibilidad y es- calabilidad constituyen las dos piedras de toque esenciales para adoptar una decisión so- bre la infraestructura o el paradigma a emplear. Todos sabemos que lo que empieza como un pequeño proyecto acaba ampliándose ad infinitum. Su mecanismo API interno, basado en *hooks*, filtros, preprocesadores y *callbacks*, permiten intervenir en prácticamente cual- quier operación que WP tenga predefinida.
  + Enrutamiento de URIs y URLs “SEO friendly”. Son características de gran importancia a la hora de optar por una modelización u otra. La trascendencia de enrutar adecuadamente las llamadas a las distintas estructuras de la aplicación es innegable. Por otra parte, la es- tructura de la URL mediante la cual se identifica un contenido juega un papel muy impor- tante en el mecanismo SEO, decisivo ala hora de que un proyecto cumpla con las condi- ciones de la web semántica, y WordPress implementa magníficamente ambas facilidades.
  + *Caching*. Si se está buscando una aplicación de alto rendimiento que pueda servir muchas

peticiones simultáneas y manejar decenas de miles de usuarios, es necesario disponer de un mecanismo de almacenamiento en caché eficiente pero que no sacrifique la precisión en aras de las prestaciones. WP y su API de *transients* proporcionan funcionalidad de al- macenamiento en caché a nivel de la base de datos que se utilizará en la aplicación.

* + Plantillas. La presentación o *front-end* no es un mero adorno. Mediante una peculiar arqui- tectura basada en funciones y con extenso uso de *hooks* y retrollamadas, WP dispone de una potentísima mecánica de formateo final de páginas, los temas, de gran flexibilidad tanto en su cara administrativa como en la de CL. Una misma página, por ejemplo puede disponer de varias plantillas que modifiquen o amplíen la experiencia del usuario.

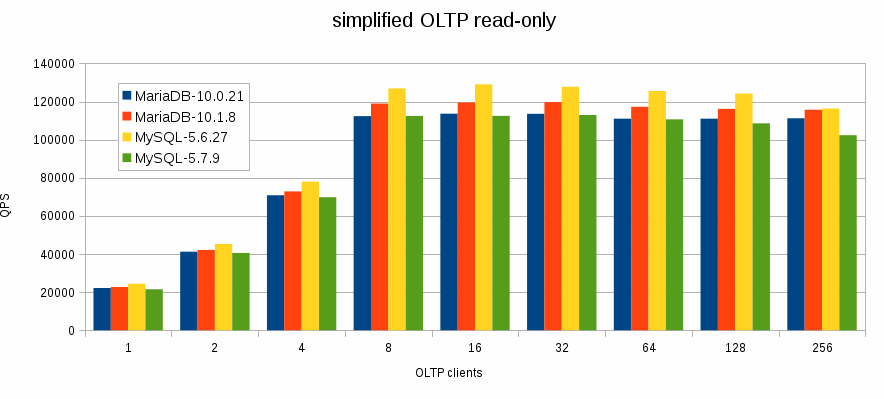
No es, ni mucho menos, despreciable la capacidad de WP en cuanto se refiere a escalabilidad. La posibilidad de redimensionar la operatividad de EF es un punto importante a su favor. WP puede trabajar con dos SGBD: MySQL y MariaDB, heredero éste último del anterior, que obtiene unas ci- fras de velocidad y capacidad de respuesta en ejecución más que notables.

Ilustración 7. Comparativa de test de velocidad MariaDB y MySQL

Los inconvenientes más destacados pasan por una cierta rigidez estructural si se desea aprove- char la magnífica configuración de los *custom post*, los metadatos y las taxonomías, jerárquicas o no, que en ocasiones dificulta bastante la posibilidad, por ejemplo, de establecer vocabularios con- trolados o tesauros. No obstante, tales obstáculos pueden ser considerados menores en compa- ración con las capacidades de que WP hace gala.

## Ingeniería de requisitos

Dadas las peculiaridades y el ambiente de EF (subproyecto al servicio de EX), el actor *Supervi- sor*12 se encarga de la especificación de requisitos. La definición de los mismos fue expresada por NRO en el documento de solicitud adjuntado al proyecto EX. Según ese texto, se trataba de explo-

1. Recuérdese que, en el caso de EF, coincide con el *Product Owner* de la filosofía SCRUM, lo que le faculta para decidir los obje-

tivos finales en función de su pertinencia para la obtención de resultados en EX.

tar *la función dinamizadora de la cultura artística desarrollada por los museos y las galerías me- diante las exposiciones temporales, entendidas éstas como fenómenos complejos de actores en relación dinámica. Es un proyecto que también se enfoca a mejorar la sociedad facilitando el ac- ceso y la reutilización de contenidos culturales digitales, y en este sentido plantea como uno de sus resultados desarrollar una plataforma web que proporcionará información de interés al usua- rio, sea aficionado al arte, agente de viajes, hotelero, galerista, conservador de museos, comisario de exposiciones o investigador*. Por tanto, se imponía desde un primer momento desarrollar una aplicación accesible desde la Red capaz de cubrir las necesidades que se planteaban.

En el caso de EF, por tanto, no se han utilizado metodologías al uso, tales como las entrevistas, talleres, lluvia de ideas o contrato formalizado como fuente de especificaciones de requerimientos, sino un proceso analítico a partir de lo mencionado. Dicho proceso ha sido acometido por NRO y MLBG en calidad de *Supervisoras*, y ha sido concretado en un documento, a partir del cual se ha elaborado la estructura lógica y física del proyecto EF, y que no se adjunta en el presente PDSDT por formar parte de la documentación de EX.

Mediante la consulta de dicho documento se ha realizado la **matriz de requisitos y procesos** que a continuación se recoge, según su clasificación. Téngase en consideración que dicha matriz in- cluye referencias a los **procesos** y subprocesos del sistema, que están especificados en [9](#_bookmark23). De acuerdo a la importancia o la trascendencia a los propósitos de EX, los requisitos se han clasifica- do en *funcionales*, *organizacionales* y *no funcionales*. En todos los casos, la información que de ellos depende puede ser o no de carácter **crítico**. La **atribución** de un requisito a determinado ac- tor o proyecto explicita para quién debe realizarse la solución propuesta en cada caso y qué ele- mento es responsable de su verificación final.

* + 1. Requisitos funcionales

|  |  |  |  |
| --- | --- | --- | --- |
| **REQUISITO** | **ATRIBUCIÓN** | **PROCESO** | **CRÍTICO** |
| Recopilar información sobre entidades susceptibles de producir  información acerca de ETA | EX | Búsqueda13  Grabación | Sí |
| Agregar a la información de entidades URIs que puedan ser usadas para automatizar el proceso de recogida de información | EX | Grabación | Sí |
| Automatizar la recogida de información sobre ETA mediante  proceso sin intervención humana de carácter cíclico | EX | Captura | Sí |
| Seleccionar de forma automática la información sobre ETA recogida de manera que solo se almacene aquella que pueda  ser relevante | EX | Filtrado | Sí |
| Intervenir con actores pertenecientes a RRHH dotados de la adecuada formación en DH y DAH para eliminar cualquier in-  formación sobre ETA que el proceso automatizado haya podido incluir de forma inadecuada | EX | Verificación | Sí |
| Completar la metainformación pertinente para la información  sobre ETA automáticamente capturada, filtrada y seleccionada | EX | Grabación | Sí |
| Facilitar la transferencia de la información recopilada y verifica- | EX | Exportación | Sí |

1. El proceso de búsqueda debe entenderse como actividad exclusivamente realizada por RRHH, sin intervención del sistema in- formático. Se trata de utilizar los conocimientos en DH y DAH de los actores *Grabador verificador* y *Revisor* para localizar po-

sibles fuentes de información a partir de las cuales obtener una lista de URIs desde las que capturar datos sobre ETA.

|  |  |  |  |
| --- | --- | --- | --- |
| **REQUISITO** | **ATRIBUCIÓN** | **PROCESO** | **CRÍTICO** |
| da a actores externos de EX para su posterior uso en tareas de  análisis |  |  |  |
| Facilitar el acceso para consulta, monitorización y análisis (a menor escala que en el caso del realizado por los actores de  EX) a la información recopilada | EF | Interfaz público | No |

* + 1. Requisitos organizacionales

|  |  |  |  |
| --- | --- | --- | --- |
| **REQUISITO** | **ATRIBUCIÓN** | **PROCESO** | **CRÍTICO** |
| Poder realizar clasificaciones taxonómicas sobre la información  y metainformación recopilada | EX | Grabación | Sí |
| Poder explotar la información y metainformación recopilada desde un punto de vista de su cobertura temporal | EX | Grabación | Sí |
| Poder explotar la información y metainformación recopilada  desde un punto de vista de su cobertura geográfica | EX | Grabación | Sí |

* + 1. Requisitos no funcionales

|  |  |  |  |
| --- | --- | --- | --- |
| **REQUISITO** | **ATRIBUCIÓN** | **PROCESO** | **CRÍTICO** |
| Garantizar la seguridad de acceso y de conservación de la in-  formación recopilada | Desarrollador | Administración | Sí |
| Garantizar la coherencia de la metainformación recopilada me-  diante tesauros y vocabularios controlados | EF  Desarrollador | Grabación | Sí |
| Utilizar tecnología que aúne a la vez simplicidad y potencia bajo  la filosofía KISS | Desarrollador | Administración | Sí |
| Permitir que el sistema de filtrado automático disponga de un mecanismo de aprendizaje basado en procedimientos heurísti-  cos | Desarrollador | Filtrado | No |
| Garantizar la disponibilidad de acceso a los equipos de trabajo,  con un mínimo de “tres nueves”14 | Desarrollador  Administrador | Administración | No |

## Implementación de la solución

De acuerdo con lo expresado en [8.1,](#_bookmark21) EF adopta como arquitectura la construcción de un artefacto sustentado en WP utilizado como FW. No obstante, es necesario precisar que hay tres formas de construir la estructura de la aplicación con las citadas premisas.

**Interviniendo sobre el núcleo funcional (*core*) de WP y adaptándolo a las necesidades de EF**. Descartado desde un principio para poder conseguir el mantenimiento de seguridad y funcio- nalidad derivado de las actualizaciones periódicas del FW, como se ha dicho en [8.1,](#_bookmark21) sin necesidad de acciones específicas.

**Utilizando un *plugin***. Un *plugin* o complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecuta- da por la aplicación principal e interactúan por medio de la API. También se lo conoce como *plug- in*, *add-on*, *conector* o *extensión*. WP utiliza extensamente estos mecanismos, con el beneficio agregado de que su funcionamiento es absolutamente independiente de la presentación, acabado

1. La disponibilidad es usualmente expresada como un porcentaje del tiempo de funcionamiento en un año dado. La regla de los

“tres nueves” implica que el sistema deberá estar disponible todo el tiempo salvo 99,9% = 43.8 minutos/mes u 8,76 horas/año.

o formato externo de la aplicación sobre ellos construida. Su API está disponible por completo mediante tres sistemas:

* + **Hooks**. O “ganchos”. Permiten al *plugin* incorporarse al sistema como un elementos más sin necesidad de modificar la estructura *core* interna.
  + **Acciones**. Son causadas por eventos específicos que ocurren en WP, como publicar un *post*, cambiar de plantilla (“tema”), o mostrar una pagina en el panel de administración. Un plugin puede responder al evento al ejecutar una función PHP, que a su vez puede llevar a cabo una o varias de las siguientes acciones:
    - Modificar datos en la base de datos.
    - Enviar un mensaje de correo
    - Modificar lo que es mostrado en la pantalla del navegador (ya sea al administra- dor o a cualquier otro usuario)
  + **Filtros**. Son funciones a través de las cuales WP transmite datos en ciertos puntos de la ejecución, justo antes de emprender alguna acción con esos datos (como agregarlos al SGBD o enviarlos a la pantalla del navegador). Los filtros actúan como interfaz o punto de paso intermedio entre la base de datos y el navegador cuando WP esta generando pagi- nas, y entre el navegador y la base de datos cuando WP esta agregando registros. La mayoría de los datos entrantes y salientes en WP pasan a través de por lo menos un filtro, establecido por diseño.

**Utilizando un tema**. Es la solución adoptada. Se define un “tema” como un conjunto de archivos que trabajan coordinados para crear el diseño y funcionalidad de una sede WP. Cada tema puede ser diferente, ofreciendo muchas opciones para cambiar al instante la apariencia y/o la funcionali- dad de la aplicación. Ofrece numerosas ventajas:

* + Permite crear una apariencia única y personalizada de una aplicación.
  + Permite disponer de la ventaja de una mecánica basada en plantillas (*templates*), etique- tas de plantillas (*template tags*) para generar diferentes resultados y apariencias del sitio web, lo que aproxima el funcionamiento de WP al de una estructura MVC.
  + Para presentar los resultados en un tema, WP utiliza un bucle por diseño (*the WordPress Loop*). La intervención sobre él también posibilita actuar sobre los resultados obtenidos en cada consulta.
  + Es posible utilizar plantillas alternativas para características específicas de la aplicación, tales como páginas por taxonomía o páginas de archivo.
  + Para rápidamente cambiar entre dos disposiciones de sitio, o aprovecharse de un selector de tema o estilo Theme or style switcher para permitir a los propietarios de sitio el cambiar la apariencia de tu sitio.
  + Separa los estilos de presentación y archivos de plantilla de los del sistema, de forma tal que es posible realizar intervenciones y ajustes sin cambios drásticos para la presentación visual del sitio.

Los temas tienen disponibles todos los *hooks*, las acciones y los filtros del API WP. La descripción de la mecánica de integración con el *core* de WP aparece recogida en [11.](#_bookmark27)

En [14](#_bookmark33) figura una representación gráfica del *stack* o pila de capas de funcionamiento de WP. A través de dicha representación es posible conocer la forma en que interactúan los diferentes ele- mentos de un tema con el núcleo operativo. A continuación se muestra una matriz de los subsis- temas WP utilizados desde EF para implementar su funcionalidad.

|  |  |
| --- | --- |
| **SUBSISTEMA WP** | **FUNCIONALIDAD EF** |
| Ajax | Control de funciones ejecutadas en cliente |
| CURL | Captura automatizada de datos |
| Database | Acceso directo a SGBD. Mantenimiento |
| Mail | Comunicación con usuarios (seguridad) |
| REST | Exportación de datos. Ajax (funciones ejecutadas en cliente) |
| RSS/Atom Reader | Captura automatizada de datos |
| Sanitization | Seguridad frente a intrusiones |
| Usuarios | Acceso granular a funcionalidades EF. Seguridad de acceso |
| WP Cookies | Control de acceso. Control de sesiones |
| WP Cron Jobs | Captura automatizada de datos. Mantenimiento |

La estructura detallada de la solución arquitectónica del tema figura en [11.](#_bookmark27)

# 9 Resumen de procesos del sistema

Dado el limitado alcance de PDSDT en la versión actual (1.2), en el presente apartado se han simplificado los casos de uso y la descripción de los principales procesos se exhibe agrupada por funcionalidad y extensibilidad.

Como ha quedado determinado en [8.2,](#_bookmark22) el conjunto completo del proyecto EX incluye dos grandes bloques de trabajo, que determinan, en gran medida, el ritmo y el formato de su despliegue:

* Por una parte, un “motor” que posibilite la extracción y el almacenamiento de la informa- ción sobre ETA.
* Por otra, dicho “motor” debe poder exportar los registros conseguidos de forma que sea factible su explotación en plataformas diseñadas para el análisis de grandes volúmenes de datos mediante técnicas de descubrimiento de relaciones y patronizado de estructuras reticulares, tales como el *cowording*, trazado de redes, etc.

Esas condiciones determinan la creación de **tres sistemas** dentro del conjunto de EF:

|  |  |  |  |
| --- | --- | --- | --- |
| **ABRV** | **SISTEMA** | **NOMBRE** | **TAREAS** |
| SC | Captura Filtrado | Beagle | ∞ Leer URIs almacenadas en entidades y construir lista |
| ∞ Buscar en lista información sobre ETA |
| ∞ Filtrar información sobre ETA y “aprender” sobre calidad de información |
| ∞ Guardar en borrador información encontrada |
| SG | Grabación | ExpoFinder Admin | μ Verificación de validez de información capturada |
| μ Completado de metainformación sobre información capturada |
| μ Grabación de metainformación e información capturada |
| SE | Exportación | ExpoFinder Export | α Construcción de tablas de datos exportables |
| α Generación de formatos exportables |

***Clave***: **∞** = automática cíclica; **α** = automática; **μ** = manual

La denominación *Beagle* se asigna en honor al buque con el que Charles Darwin recorrió medio mundo y por su signi- ficado en inglés: “sabueso”.

## Actores y casos de uso. Modelos básicos

Se denomina caso de uso (CDU en adelante) a una descripción de las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participan se deno- minan *actores*. En el ámbito de la ingeniería del software, la expresión alude a la secuencia de in- teracciones que deben realizarse entre un sistema y sus actores en respuesta a un evento que ini- cia un actor sobre el propio sistema.

* + 1. Actores

La lista de actores que intervienen en EF se muestra a continuación.

|  |  |
| --- | --- |
| **NOMBRE** | **DESCRIPCIÓN** |
| Administrador | Usuario que gestiona el SO del servidor y el funcionamiento interno de WP |
| Beagle | Proceso automatizado cíclico que captura la información procedente de las URIs RSS y  HTML, filtra la que alcanza un nivel mínimo para ser considerada válida y la graba como borrador para ponerla a disposición de *Grabador verificador* |
| Grabador verificador | Valida la información capturada por Beagle. Descarta aquella que pueda haber sido filtrada como adecuada por error de Beagle. Tiene derechos de modificación de la información de la cual es responsable. Carece de derechos de modificación de la información verificada y  grabada por otros usuarios de su tipo. Tiene derechos de modificación de la información |

|  |  |
| --- | --- |
| **NOMBRE** | **DESCRIPCIÓN** |
|  | capturada por Beagle |
| Revisor | Usuario que verifica la correcta aplicación de la normativa sobre coherencia de la informa- ción. Tiene derechos de modificación de la información almacenada por *Grabador verifica-*  *dor* |
| Sistema | Sistema de software EF |
| Supervisor | Usuario que valida la información, autoriza sus modificaciones y evalúa el éxito de EF. Tie- ne derechos de modificación de la información almacenada por *Grabador verificador* |
| Usuario externo | Usuario con un nivel de autenticación mínimo (identificación, sin derechos de acceso a back-end) |
| Usuario no pertenecien-  te a RRHH | Usuario que no se ha identificado frente al sistema. No puede acceder a front-end ni a  back-end |
| WP | API de WordPress a disposición de EF |

La equivalencia entre los actores pertenecientes a los grupos de RRHH y los usuarios estándar ofrecidos por el API WP, que se utilizan en EF como mecanismo de seguridad y acceso, se expre- sa en la siguiente tabla.

|  |  |  |
| --- | --- | --- |
| **ACTOR** | **TIPO DE USUARIO WP** | **DERECHOS15** |
| Usuario externo | [Sin acceso a EF] | [Sin acceso a EF] |
| Usuario no perteneciente a RRHH | Suscriptor | R T. Front-end |
| Grabador verificador | Autor | CRUD P. Front-end y back-end |
| Revisor | Editor | CRUD T. Front-end y back-end |
| Administrador | Administrador | CRUD T. Front-end y back-end |
| Supervisor | Editor | CRUD T. Front-end y back-end |

* + 1. Casos de uso

En el contexto de EF, y teniendo en cuenta que, tal y como se ha establecido desde [1.3](#_bookmark1), el alcan- ce del presente artefacto PDSDT es limitado, se expresan los casos de forma genérica, adaptán- dolos a los principales procesos del sistema.

|  |  |
| --- | --- |
| Nombre | **Lectura de URIs** |
| Descripción | SC selecciona registros de tipo entidad que contengan metainformación sobre URIs RSS y/o  HTML. Con la selección elabora una lista de URIs. |
| Actores | Beagle |
| Precondiciones | Deben existir registros de entidades. Debe estar grabada metainformación sobre entidades. La  metainformación debe incluir URIs. |
| Flujo normal | SC consulta SGBD usando API WP para realizar la selección. |
| Flujo alternativo | SC dispara una condición de error usando API WP. |
| Postcondiciones | Captura de información debe disponer de una lista (array) de URIs RSS y HTML válidas. |
| Referencias | Captura de información |

|  |  |
| --- | --- |
| Nombre | **Captura de información** |
| Descripción | SC lee la lista de URIs. Secuencialmente, y mediante el acceso a la lista de URIs, consulta, usando el API WP, la información almacenada y extrae la información relevante. |
| Actores | Beagle, WP. |

15 Clave de derechos de acceso a información en SGBD. C=Creación; R=Lectura, U=Actualización; D=Borrado. T=acceso a los

registros generados por cualquier actor; P=acceso solo a los registros creados por el propio actor.

|  |  |
| --- | --- |
| Precondiciones | Debe existir la lista de URIs. |
| Flujo normal | SC lee una URI. Mediante API WP extrae el resultado (protocolo http, estado y contenido). |
| Flujo alternativo | SC dispara una condición de error usando API WP. |
| Postcondiciones | Filtrado deba disponer de un contenido HTML o XML analizable. |
| Referencias | Lectura de URIs, Filtrado |

|  |  |
| --- | --- |
| Nombre | **Filtrado** |
| Descripción | SC usa la información relevante obtenida por Captura de información para transformarla en texto y aplicar el algoritmo de selección. Si la información obtiene o sobrepasa el nivel mínimo, se transfie- re a Aprendizaje como válida; en caso contrario, se descarta y se transfiere a Aprendizaje como no  válida. |
| Actores | Beagle. |
| Precondiciones | Debe existir la información relevante. Debe poder ser transformada desde su formato original (XML  o HTML) en texto. |
| Flujo normal | SC lee la información relevante. Evalúa el contenido mediante aplicación de algoritmo de filtro. Transfiere resultados a Aprendizaje. |
| Flujo alternativo | SC dispara una condición de error usando API WP. |
| Postcondiciones | Aprendizaje debe disponer de un contenido en formato texto. |
| Referencias | Captura de información, Aprendizaje |

|  |  |
| --- | --- |
| Nombre | **Aprendizaje** |
| Descripción | SC usa la información filtrada. Si procede de Filtrado como válida, aplica algoritmo clasificador ba-  yesiano ingenuo (CBI) y la clasifica como válida. En caso contrario, aplica el mismo algoritmo y la clasifica como no válida. |
| Actores | Beagle, Sistema |
| Precondiciones | Debe existir la información filtrada y evaluada. |
| Flujo normal | SC lee la información evaluada. Aplica algoritmo CBI y la clasifica. |
| Flujo alternativo | SC dispara una condición de error usando API WP. |
| Postcondiciones | Grabación debe disponer de respuesta completa proporcionada por Captura de información y del  resultado del filtro y de la clasificación. |
| Referencias | Grabación |

|  |  |
| --- | --- |
| Nombre | **Grabación** |
| Descripción | SC recoge la información transferida por Aprendizaje. Añade la metainformación (entidad origen de URI usada para capturar la información, fecha y hora de captura,…). Graba un registro de tipo ex-  posición en SGBD marcado como “borrador”. Atribuye dicho registro al actor Grabador verificador que grabó la información de la entidad origen. |
| Actores | Beagle, Sistema |
| Precondiciones | Debe existir la información capturada, filtrada y aprendida. Debe existir la metainformación asocia- da. |
| Flujo normal | SC lee la información capturada, filtrada y aprendida. Graba una exposición en formato “borrador”.  Asigna la exposición a usuario correspondiente a Grabador verificador responsable de la entidad origen. |
| Flujo alternativo | SC dispara una condición de error usando API WP. |
| Postcondiciones | Grabador verificador debe disponer de un registro con la información capturada y la metainforma-  ción en formato borrador. |
| Referencias | Aprendizaje, Verificación |

|  |  |
| --- | --- |
| Nombre | **Verificación** |
| Descripción | Grabador verificador lee registros grabados por Grabación asignados a su usuario WP. Verifica que  la información capturada es válida. Si no lo es, utiliza el flujo alternativo. Completa la metainforma- |

|  |  |
| --- | --- |
|  | ción obligatoria y la que sea posible no obligatoria. Cambia el formato a “publicado”. Graba el resul-  tado. |
| Actores | Grabador verificador, Sistema |
| Precondiciones | Debe existir un registro guardado en formato “borrador” asignado al usuario WP que identifica a  actor. |
| Flujo normal | Grabador verificador lee registro guardado. Verifica validez de datos. Caso de no ser correctos, utiliza el flujo alternativo. Completa la metainformación obligatoria y, en su caso, no obligatoria re- curriendo a búsquedas en la Red y cualesquier otros procedimientos externos que sean necesa-  rios. Cambia el formato del registro a “publicado”. Graba el registro. |
| Flujo alternativo | Grabador verificador constata que el registro guardado no cumple las condiciones para ser válido.  Lo marca como “papelera”. |
| Postcondiciones | Grabador verificador debe disponer de un registro con la información capturada y la metainforma-  ción en formato borrador. |
| Referencias | Grabación, Revisión |

|  |  |
| --- | --- |
| Nombre | **Grabación auxiliar** |
| Descripción | Inducida por Verificación. Si es preciso grabar cualquier registro de entidad, persona, publicación o empresa, Grabador verificador procede a cumplimentar la información y la metainformación propia  de cada caso específico. El registro, una vez guardado, es susceptible de Revisión. |
| Actores | Grabador verificador, Revisor |
| Precondiciones | Debe existir un registro guardado en formato “borrador” asignado a Grabador verificador. De las circunstancias concretas de dicho registro, debe desprenderse la necesidad de grabar información  y/o metainformación adicional de tipo entidad, persona, publicación o empresa. |
| Flujo normal | Grabador verificador completa la información y la metainformación obligatoria y, en su caso, no obligatoria recurriendo a búsquedas en la Red y cualesquier otros procedimientos externos que  sean necesarios. Cambia el formato del registro a “publicado”. Graba el registro. |
| Flujo alternativo | No contemplado. |
| Postcondiciones | Revisor debe disponer de un registro grabado de tipo entidad, persona, publicación o empresa y en  formato “publicado”. |
| Referencias | Revisión |

|  |  |
| --- | --- |
| Nombre | **Revisión** |
| Descripción | Revisor lee registro grabado y en formato “publicado”. Comprueba la adecuación de la metainfor-  mación grabada a la normativa sobre taxonomía y metainformación (vocabularios controlados). En caso de no adecuación a norma, utiliza flujo alternativo. |
| Actores | Grabador verificador, Revisor, Supervisor |
| Precondiciones | Debe existir un registro guardado en formato “publicado” asignado a Grabador verificador. |
| Flujo normal | Revisor lee registro grabado por Grabador verificador como “publicado”. Verifica validez de datos. |
| Flujo alternativo | Revisor constata que la metainformación grabada en el registro guardado no se adecua a la norma-  tiva sobre taxonomía y metainformación (vocabularios controlados). Emite una SC a Supervisor. |
| Postcondiciones | Revisor debe disponer de un registro grabado y en formato “publicado”. |
| Referencias | Verificación, Borrado, Supervisión |

|  |  |
| --- | --- |
| Nombre | **Supervisión** |
| Descripción | Supervisor recibe una SC de Revisor. Evalúa si se trata de una actuación errónea o hay que modi- ficar el vocabulario controlado o las normas de verificación. En el primer caso, utiliza el flujo alterna-  tivo. En el segundo, gestiona el alta de la metainformación en el vocabulario controlado o altera la normativa en consecuencia. |
| Actores | Grabador verificador, Revisor, Supervisor |
| Precondiciones | Debe existir una SC de Revisor. |
| Flujo normal | Supervisor recibe una SC de Revisor. Evalúa positivamente la solicitud. Realiza los cambios en el vocabulario controlado o modifica las normas |

|  |  |
| --- | --- |
| Flujo alternativo | Supervisor recibe una SC de Revisor. Evalúa negativamente la solicitud. Envía mensaje por men-  sajería interna a Revisor y a Grabador verificador para que se efectúe Borrado. |
| Postcondiciones | Supervisor debe disponer de una SC. Debe existir un registro grabado y en formato “publicado”. |
| Referencias | Revisión, Borrado |

|  |  |
| --- | --- |
| Nombre | **Borrado** |
| Descripción | Por iniciativa propia durante Verificación o a instancias de Supervisor tras Supervisión, Grabador  verificador borra un registro no válido. |
| Actores | Grabador verificador, Revisor, Supervisor, Sistema |
| Precondiciones | Debe existir un mensaje por mensajería interna de Supervisor. |
| Flujo normal | Grabador verificador recibe un mensaje por mensajería interna para borrado de un registro. Graba- dor verificador cambia su estado a “papelera”. Grabador verificador “vacía” la Papelera de su usua- rio WP. Sistema borra el registro del CBI como positivo. Sistema da de alta el registro en CBI como  negativo. |
| Flujo alternativo | SC dispara una condición de error usando API WP. |
| Postcondiciones | El registro erróneo debe ser eliminado con efectividad de la SGBD. |
| Referencias | Supervisión |

|  |  |
| --- | --- |
| Nombre | **Exportación** |
| Descripción | A través de API WP, Sistema recibe una solicitud REST de preparar una exportación de datos. La solicitud incluye parámetros sobre tipo de exportación. Sistema verifica autenticidad de la solicitud.  En caso de verificación positiva, sigue flujo normal. En caso de verificación negativa, sigue flujo alternativo. |
| Actores | Sistema, Usuario externo |
| Precondiciones | Debe existir una solicitud REST. La solicitud debe superar el control de seguridad e identificación. |
| Flujo normal | La solicitud REST es evaluada como válida (identificación, formato de salida y tipo de datos). Sis- tema selecciona los datos solicitados. Sistema da formato a dichos datos de acuerdo a la solicitud.  Sistema devuelve un archivo con la información solicitada. |
| Flujo alternativo | Sistema devuelve un archivo en formato XML-RPC con la condición de error. |
| Postcondiciones | El archivo devuelto debe ser legible. |
| Referencias | Ninguna |

En [14](#_bookmark33) se incluye un diagrama global y genérico del modelo de casos de uso.

## Procesos

Los procesos de EF se agrupan en **principales**, derivados de los CDU, y **auxiliares**, necesarios para el funcionamiento del sistema. Cada proceso se corresponde con un elemento dentro del conjunto de interfaces.

* + 1. Procesos principales

Generados a partir de los casos de uso. Su equivalencia dentro de los sistemas de EF se expresa en la siguiente tabla.

|  |  |  |
| --- | --- | --- |
| **CDU** | **PROCESO** | **SISTEMA** |
| Lectura de URIs | Captura | Beagle |
| Captura de información | Captura | Beagle |
| Filtrado | Filtrado | Beagle |
| Aprendizaje | Filtrado | Beagle |
| Grabación | Grabación | Beagle |
| Grabación auxiliar | Grabación | ExpoFinder Admin |
| Verificación | Grabación | ExpoFinder Admin |
| Supervisión | Grabación | ExpoFinder Admin |

|  |  |  |
| --- | --- | --- |
| **CDU** | **PROCESO** | **SISTEMA** |
| Borrado | Grabación | ExpoFinder Admin |
| Exportación | Exportación | ExpoFinder Export |

* + 1. Procesos auxiliares

Generados a partir de las necesidades derivadas de los casos de uso y de los requisitos. Se citan en la siguiente tabla.

|  |  |
| --- | --- |
| **PROCESO** | **DESCRIPCIÓN** |
| Ayuda contextual | Adaptada a cada pantalla del interfaz back-end. Disponible para todos los usuarios WP |
| Mensajería interna | Disponible para todos los usuarios WP |
| Avisos administrativos | Emitidos por Supervisor o Administrador. Legibles para todos los usuarios WP |
| Control de calidad y  Actividad | Evaluación automática de errores formales, estadísticas de trabajo y sesiones por usuario  WP. Disponible para todos los usuarios WP |
| Evolución | Evaluación automática del nivel de éxito de EF. Disponible para todos los usuarios WP |
| Mantenimiento | Realización de tareas de optimización, inicialización y mejora de los datos guardados en  SGBD. Monitorización de estado de memoria y microprocesador en servidor. Disponible solo para Administrador |
| Ajuste | Modificación y control de los objetivos numéricos finales de EF. Control de otros parámetros  básicos de la aplicación. Disponible solo para Administrador. |
| Consultas | Generación de *queries* SQL para uso en front-end. Disponible solo para Administrador. |

* + 1. Procesos del *framework*

No se incluye en PDSDT una descripción detallada de los procesos inherentes al FW, que abar- can ámbitos como la seguridad, el acceso, la gestión CRUD, la gestión de plantillas, etc. Reco- mendamos consultar la información al respecto en la página https://codex.wordpress.org/.

# Interfaces del sistema

Como ha quedado establecido en [9,](#_bookmark23) existen dos grandes bloques de interfaces que permiten ac- ceder a la aplicación EF. No todos los interfaces se corresponden con tareas. Por ejemplo, los procesos *Captura* y *Filtrado*, ambos integrantes de Beagle, al ser ejecutados por el proceso del sistema WP Cron Jobs no precisan disponer de una “cara” en pantalla o en consola. En realidad, sí que existe, a efectos de depuración, una salida STD para ambos empleando el interfaz CLI de PHP, pero no forma parte de una especificación formal, sino de un simple mecanismo de control.

En esta descripción de interfaces NO SE INCLUYE INFORMACIÓN sobre elementos propios del FW WP (gestión de usuarios, plugins, apariencia, actualizaciones,…) por considerarlo fuera del al- cance del presente artefacto PDSDT.

|  |  |
| --- | --- |
| **INTERFAZ** | **DESCRIPCIÓN** |
| Back-end | Interfaz administrativo. Permite operaciones CRUD, ajustes y monitorización del estado del proyecto |
| Front-end | Interfaz público16. Permite búsquedas de datos sobre todos los tipos de registro y consultas  de carácter estadístico. |

## Back-end

Se ciñe estrictamente al interfaz diseñado por el FW WP, pero añade pantallas y elementos de menú propios, alterando algunas de las informaciones ofrecidas por defecto por WP por razones de seguridad17. Dentro del back-end se pueden distinguir dos tipos de pantalla:

* + - **Pantallas CRUD**. Se usan para la gestión y el mantenimiento de todos los tipos de *cus- tom posts*: *entidades*, *personas*, *publicaciones*, *empresas*, *exposiciones* y el de uso ex- clusivo de Administrador *consultas*.
    - **Pantallas informativas**. Se usan para informar a los usuarios sobre datos que pueden ser de su interés: *Mensajería*, *Control de calidad*, *Actividad*, *Evolución* y (de uso exclusivo de Administrador) *Mantenimiento*.

Las pantallas CRUD incluyen, básicamente, un complejo formulario con los diversos tipos de cam- pos que almacenan la información y la metainformación de cada registro. Esos campos pueden ser:

* + - Campos de texto no editables. Solo muestran información.
    - Campos de texto editables. Longitud limitada. Pueden contener mecanismos de verifica- ción para ajuste a normas y vocabularios controlados (Geonames, Fundación Getty,…).
    - Campos georreferenciados. Incluyen un mapa (en esta versión de PDSDT procedente de

1. Esa expresión (público) no es especialmente exacta. Por razones inherentes a la estructura del proyecto EX, desde un principio se ha adoptado la decisión de que solamente puedan acceder al front-end los usuarios WP autorizados con el tipo “Suscrip- tor”. Por tanto, para consultar datos (no ya para intervenir sobre ellos) es preciso estar identificado. El usuario puramente ex- terno no tiene acceso a ninguna información. Esta situación previsiblemente evolucionará con el tiempo.
2. Por ejemplo, se sustituye la pantalla WP conocida como “Desktop” por una de elaboración propia, que incluye información de los

procesos Mensajería interna, Avisos administrativos, Control de calidad, Actividad, Evolución y Mantenimiento.

Google Maps), un campo de sólo lectura con las coordenadas y un campo editable con la dirección, sincronizados ambos con un marcador de posición que se muestra en el mapa.

* + Campos autorreferenciales. Muestran una lista de registros ya guardados en el sistema.
  + Campos selectores. Botones de radio, casillas de verificación y desplegables. Permiten selecciones simples o múltiples.

Cada formulario CRUD incluye un mecanismo “nonce” de verificación de origen por razones de seguridad e integridad de la información. Algunos formularios (como el de entidades) incluyen pro- cesos de supervisión o verificación externa. Todos los formularios incluyen mecanismos de super- visión de la calidad de ciertos campos, según las reglas de integridad y las normativas.

* + Campos cuyo contenido sea una URI18. Verificación de su adecuación a norma. En el caso de las URIs RSS y HTML del registro de tipo entidad, Sistema realiza una consulta remota usanto el API WP para verificar que la dirección conduce a un extremo cuya respuesta http es válida y el contenido de sus datos es coherente con la solicitud (XML en caso de URIs RSS y HTML en caso de URIs de ese tipo).
  + Campos cuyo contenido sea un número de teléfono. Verificación de que solo contiene ca- racteres numéricos.
  + Campos que siguen la norma **2 \* ;**. Aplicable a nomenclaturas geográficas basadas en Geonames. Verifica que contengan 3 bloques de texto separados por punto y coma (;).
  + Campos que siguen la norma **1 \* :**. Aplicable a campos autorreferenciados. Verifica que contenga un bloque numérico separado de un bloque de texto por un signo de dos puntos seguido de un espacio en blanco (: ).

En [14](#_bookmark33) se incluyen volcados de modelos de pantalla del interfaz back-end.

## Front-end

En el momento de elaborar la versión actual de PDSDT, se encuentra aún en fase de desarrollo y no ha sido completado. Para la elaboración se ha seguido una plantilla basada en el FW CSS3 + HTML5 + JavaScript Bootstrap 3.0, por la simplicidad de sus elementos, basados en un formato de bloques de color planos, sin simulación de relieve (no *skeumórficos*). Básicamente se agrupa en 4 grandes bloques.

* + - Pantalla de presentación. Simple *splash* de conexión. Con información resumida sobre las estadísticas de grabación de los distintos tipos de CPs y sus etiquetas más significativas, así como sobre el número de URIs HTML y RSS/Atom activas. Además, información gráfi- ca sobre:
      * Grado de finalización del proyecto.
      * Porcentaje de hombres y mujeres registrados en los CP de tipo *Personas*.
      * Las 5 relaciones más fuertes entre distintos tipos de registro (diagrama Sankey).

1. Según el esquema oficial de URIs de IANA. Véase [http://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml.](http://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml)

o Mapa con las 30 ciudades más activas (con más exposiciones o entidades).

* + Menú Proyecto. Incluye información resumida sobre EX, EF, equipo de trabajo y estado del proyecto EF resumido en estadísticas más relevantes.
  + Menú Documentos. Incluye información sobre EX, EF y otro material de carácter técnico pero de tono divulgativo.
  + Menú Consultas. Incluye dos submenús.
    - Banco de datos. Acceso detallado a los campos *no restringidos* de los distintos CPs. Incluye listado y ficha de detalle, con mapa de ubicación en caso de que sea posible ofrecerlo, y enlaces a otros elementos y CPs. Permite búsqueda de texto libre sobre cualquier campo, filtrable por tipo de CP.
    - Análisis con tres pestañas, que muestran los datos obtenidos de las consultas es- tadísticas y de agrupación mencionadas en [9.2.2](#_bookmark24) en tres formatos:
      * Tabla simple exportable como XLS, CSV, PDF o al portapapeles del SO, ordenable y buscable.
      * Tabla manipulable por el usuario para crear un conjunto de referencias cruzadas (pivotable) y mostrarlo en formato gráfico o de tabla simple. Im- primible y exportable a formato CSV.
      * Gráfico de barras, líneas o áreas para representar la tabla simple. Dispo- ne de subgráfico para seleccionar zona de interés.
  + Menú Usuario. Permite acceder al interfaz Back-end con los derechos inherentes al perfil del usuario que haya iniciado la sesión y consultar el estado de la evaluación (datos indi- viduales).
  + Barra de estado, con información de tiempos de acceso y recursos empleados para cada página, así como enlaces utilitarios y obligatorios (información legal y créditos y licencias) más acceso al informa de errores y la evaluación (véase [13.1.3](#_bookmark31) y [13.1.4](#_bookmark32)).
  + Visita guiada. El usuario puede elegir realizar un “paseo virtual” automatizado por los ele- mentos principales del interfaz. La visita puede ser abandonada o retomada a voluntad en cualquier momento.

# Componentes SW

La estructura de componentes SW de EF viene determinada por el análisis de requisitos, la arqui- tectura y los procesos inherentes al proyecto. Véase al respecto [2.2,](#_bookmark4) [8](#_bookmark20) y [9](#_bookmark23).

Como principio rector, dada la naturaleza científica y académica de los proyectos EF y EX, se ha adoptado el de usar única y exclusivamente componentes no propietarios de software libre.

## Componentes básicos

La aplicación se construye empleando **PHP**, acrónimo de *Hypertext Preprocessor*. Se trata de un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML en forma de scripts ejecutables desde el extremo servidor.

**WordPress** es, en palabras de sus autores, “es una poderosa plataforma de publicación personal, y viene con una gran cantidad de características incorporadas, diseñadas para hacer tan fácil, pla- centera y atractiva como sea posible la experiencia de publicar en Internet”. Se distribuye bajo li- cencia estándar GPL.

**MySQL** es un SGBD relacional, multihilo y multiusuario, cuya compañía matriz desde enero de 2008 es subsidiaria de Sun Microsystems, y ésta a su vez de Oracle Corporation desde abril de 2009. Está desarrollado como software libre en un esquema de licenciamiento dual.

**MariaDB** es un SGBD derivado de MySQL con licencia GPL y absolutamente compatible en cuan- to a instrucciones y lenguaje con el mismo. Está desarrollado por el fundador de MySQL y la co- munidad de desarrolladores de software libre. En el momento de redactar la versión 1.1 de PDSDT, EF está funcionando empleando MySQL como gestor de bases de datos, pero está pre- vista la migración a MariaDB.

**Ubuntu** es un sistema operativo basado en GNU/Linux y que se distribuye como software libre. Su patrocinador, Canonical, es una compañía británica propiedad del empresario sudafricano Mark Shuttleworth. Ofrece el sistema de manera gratuita, y se financia por medio de servicios vincula- dos al sistema operativo.

EF utiliza ampliamente las posibilidades de la POO mediante el empleo de clases, subclases y funciones. Toda la nomenclatura de código está normalizada mediante el prefijo “csl\_” en varia- bles de alcance global, funciones, clases y constantes. La grafía del código sigue las normas de estilo de WP en cuanto a estilos de redacción (Allman preferentemente, K&R ocasionalmente).

Las versiones en uso de los componentes en el momento de redactar el presente artefacto PDSDT son las siguientes.

|  |  |
| --- | --- |
| **COMPONENTE** | **VERSIÓN** |
| PHP | 5.6.11-1ubuntu3.1 |
| WordPress | 4.4.1 es-ES |
| MySQL | 5.6.27-0ubuntu1 |
| Ubuntu | Ubuntu Server 15.04 "Vivid Vervet". Kernel 3.14.32-xxxx-std-ipv6-64 #5 SMP Tue Sep 8 17:41:55  CEST 2015 x86\_64 |

## Estructura SW

Como ya se ha mencionado en [8,](#_bookmark20) EF se construye como extensión sobre el FW WP. La construc- ción se realiza adoptando el formato de “tema”, con la denominación específica **CSL**.

El tema consiste en un directorio, que se instala en la raíz física de la instalación de WP, en la

subcarpeta wp-contents/themes. No se recoge en PDSDT la organización normalizada de un tema estándar WP, pero sí se detalla la composición propia de CSL.

Archivos obligatorios: functions.php. index.php, styles.php

functions.php. Contiene las inclusiones de librerías, la lógica del programa, la inicialización y las llamadas a las clases que generan los *custom post types*.

index.php. Contiene la lógica de arranque para front-end. Se organiza en tres bloques: footer, sidebar y WP Loop. Se ha optado por insertar TODA la estética del interfaz en este único archivo en vez de emplear múltiples plantillas.

styles.css. Contiene los estilos CSS3 básicos. Puede ser complementado con otras hojas en cascada.

El directorio (o carpeta) CSL se organiza de acuerdo al siguiente esquema.

assets

css img js

keywords logs tools

includes

languages libraries

Readability XMLRPC

b8 csl\_custom\_admin\_posts\_helper.php csl\_custom\_ajax\_helper.php csl\_custom\_dashboard\_chat.php csl\_custom\_dashboard\_helper.php csl\_custom\_data\_retrieval\_helper.php csl\_custom\_definitions\_helper.php csl\_custom\_global\_helper.php csl\_custom\_idle\_autologout.php csl\_custom\_local\_avatars.php csl\_custom\_log\_helper.php csl\_custom\_mail\_helper.php csl\_custom\_post\_helper.php csl\_custom\_profiling\_tools.php csl\_custom\_scripts\_helper.php csl\_custom\_secure\_xmlrpc\_server.php csl\_custom\_shortcodes\_helper.php csl\_custom\_survey\_helper.php csl\_custom\_terms\_management\_helper.php

\_csl\_cron\_beagle.php changelog.txt functions.php index.php license.txt readme.txt screenshot.png style.css

Los directorios figuran en color rojo. El contenido de las carpetas se describe a continuación.

|  |  |
| --- | --- |
| **CARPETA** | **CONTENIDO** |
| assets | Cualquier elemento auxiliar en el funcionamiento del programa |
| css | Hojas de estilo |
| img | Imágenes de uso en EF |
| keywords | Archivos textuales con palabras clave utilizadas en el proceso Filtrado |
| logs | Archivos textuales con los registros de las operaciones automatizadas cíclicas regidas por WP  Cron Jobs, como Beagle |
| includes | Librerías y archivos con prototipos de clases y funciones de uso común. Los archivos, en la raíz  de esta carpeta, prefijados por “csl\_” contienen dichas clases |

## Algoritmos y librerías básicas

* + 1. Algoritmos

Los procesos de EF emplean determinados algoritmos. A continuación los describimos.

## Búsqueda de palabras clave en texto

Se basa en la existencia de una lista de palabras clave, clasificada por idiomas, que se hace coin- cidir con el detectado (a través de la estructura XML o HTML y no de la cabecera HTTP) en el do- cumento a analizar. El proceso es el siguiente.

* + - * Mediante el API WP para CURL y para captura de RSS y/o Atom se obtiene el contenido de la URI explorada.
      * Al contenido se le aplica un filtro para la extracción de contenido principal basado en Rea- dability (ver lista de carpetas más arriba). Se obtiene en codificación UTF-8.
      * Al contenido principal así extraído se le aplica un filtro basado en el API WP KSES para eliminar caracteres diacríticos y reducir los elementos no analizables. Se obtiene un texto depurado.
      * Sobre el texto depurado se aplica estadística TF-IDF *(Term Frequency - Inverse Docu- ment Frequency*), basada en la Asunción de Luhn y en IDF de Spärck Jones mediante pe- so binario para TF y peso unario para IDF. Está previsto adaptar un sistema basado en normalización doble K para TF y máximo de frecuencia inversa para IDF.
      * El resultado se compara con la lista de palabras válidas mediante intersección de *arrays*.
      * Si el valor total de los *arrays* intersecados iguala o supera el umbral mínimo, el documento se considera válido.

## Evaluación del grado de éxito del proyecto

Su algoritmo es la aplicación de lo expresado en [4.1.](#_bookmark12) Es posible consultar más elementos emplea- dos en Verificación de la calidad en [13.](#_bookmark30)

## Detección y recuento de errores formales

Se basa en una compleja consulta SQL sobre las tablas estándar suministradas por el API WP, que no se reproduce en PDSDT por su elevada extensión (650 líneas en formato normalizado). Básicamente busca ausencias de metainformación obligatoria o de cumplimiento de normas (véa- se [10.1](#_bookmark26)).

El sistema calcula una tasa media de error por usuario para todos aquellos registros que al menos tengan uno, y una tasa individual, para cada Grabador verificador. Los cálculos finales se emplean también en la verificación de la calidad (véase [13](#_bookmark30)).

Se clasifican en los siguientes tipos:

* + - * Duplicados: Posible título duplicado
      * Obligatorios: Ausencia de uno ó más campos obligatorios
      * Geoname: Error en uno o más campos de la convención de Geonames para entidades lo- cales (regla 2 \* ;)
      * Autorreferencia: Error en uno o más campos de la convención de autorreferencia (regla 1

\* :)

* + - * Taxonomías: Ausencia de una o más taxonomías obligatorias

## Clasificación bayesiana ingenua

Se emplea la librería de Tobias Leupold B8 (véase [11.2](#_bookmark28)). Un clasificador de Bayes ingenuo asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica, dada la clase variable. Los CBIj se pueden entrenar de manera muy eficiente en un entorno de aprendizaje supervisado19, que es modelo que tal librería emplea, de manera que obtiene una “opinión probabilística” acerva de la validez o ausencia de ella de determinado texto capturado. Se utiliza en el contexto de los procesos Filtrado (Beagle) y Bo- rrado.

* + 1. Librerías

Como se ha dicho anteriormente, en ningún apartado de [11](#_bookmark27) se hace mención de los elementos que forman parte del API de WP, usado en el proyecto como FW. EF emplea dos categorías de li- brerías: para ejecución en servidor y para ejecución en cliente. Todas las usadas disponen de li- cencias que las caracterizan como SW libre, pudiendo ser dispuestas de acuerdo a los términos establecidos en cada caso.

* Para ejecución en servidor (PHP)

## PHP Readability

* + - Release: 0.21. 12 de febrero de 2014.
    - Licencia: MIT.
    - Autor: feelinglucky.
    - Descripción: Port de la librería homónima de Arc90 (<http://graceco.de/readability/>).
    - Función: Selección de texto relevante en página HTML.

## B8

* + - Release: 0.6.1. 12 de marzo de 2014.
    - Licencia: GPL 2.
    - Autor: Tobias Leupold.
    - Descripción: Clasificador Bayesiano Ingenuo basado en PHP y MySQL.
    - Función: Aplicar CBI a textos recuperados por Beagle y filtrados.
* Para ejecución en cliente (Javascript)

## PivotTable.js

* + - Release: 2.0.2. 12 de enero de 2016.
    - Licencia: MIT.

19 Técnica matemática que posibilita deducir o detectar la existencia de una función (o “patrón oculto”) a partir de datos suministra- dos a modo de entrenamiento, que suelen ser pares de objetos (vectores, habitualmente), que trabajan mediante sus compo-

nentes: una representa los datos de entrada y la otra el resultado esperado de salida.

* Autor: Nicolas Kruchten.
* Descripción: Formateador de *arrays* como tablas de referencias cruzadas configurable por el usuario final
* Función: Parte de la funcionalidad ofertada por front-end (Inteligencia empresarial).

## DataTables

* + Release: 1.10.10. Vía CDN.
  + Licencia: MIT.
  + Autor: SpryMedia Ltd.
  + Descripción: Formateo de tablas a partir de *arrays*.
  + Función: Parte de la funcionalidad ofertada por front-end (Inteligencia empresarial).

## C3

* Release: 0.4.10. 17 de marzo de 2015.
* Liencia: MIT.
* Autor: Masayuki Tanaka.
* Descripción: Trazado de gráficos a partir de *arrays*. Depende de D3.js <= 3.5.0.
* Función: Parte de la funcionalidad ofertada por front-end (Inteligencia empresarial).

# 12 Gestión de riesgos

Se entiende por riesgo en el contexto PDSDT cualquier acontecimiento futuro o no predecible que pueda afectar de forma negativa a EF. No todos los riesgos tienen la misma probabilidad de ocu- rrir, ni el mismo impacto, por lo que a la hora de tratarlos DT opta por seguir una pauta establecida como normal.

En EF se clasifican los riesgos de acuerdo a la matriz que a continuación se expone, basada en los estudios de Pressman20, clasificándolos por su tipología de acuerdo a las fases expresadas en [4.2.](#_bookmark13)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TIPO** | **ELABORACIÓN** | **CONSTRUCCIÓN** | **TRANSICIÓN** | **MANTENIMIENTO** |
| Técnicos | Requisitos | Diseño  Implementación Interfaz | Verificación | Control de calidad |
| Del proyecto | RRHH  Tiempo Requisitos | RRHH  Tiempo | RRHH  Tiempo |  |
| Del negocio |  | Apoyo  Presupuesto | Presupuesto | Presupuesto |

Como base de trabajo para la gestión de riesgos, EF usa la **estrategia proactiva** (no reactiva) con los siguientes principios:

* Se realiza un estudio de los riesgos potenciales (conocidos y desconocidos).
* Se miden las posibilidades de que se conviertan en reales y se cuantifican los problemas que puedan surgir.
* Se redacta una lista priorizada de los riesgos detectados.
* Se realiza el plan de gestión de riesgos, es decir, se planifica el proyecto para evitar los riesgos o minimizarlos.
* En caso de no dar resultado se dispondrá de planes de contingencia.

En cuanto al plan de gestión de riesgos, se abordan de acuerdo a la clasificación propuesta por Boehm21 (modelo en espiral).

|  |  |
| --- | --- |
| **VALORACIÓN DEL RIESGO** | **CONTROL DEL RIESGO** |
| Identificación | Planificación de la gestión |
| Análisis de las causas | Resolución |
| Priorización según gravedad | Monitorización del resultado de la solución adoptada |

La estrategia proactiva adoptada sigue unos **modelos preventivos** basados en los **principales factores de riesgo**, que se identifican a continuación. Para los restantes sucesos imponderables y con implicaciones de riesgo, Supervisor, en coordinación con Administrador y Desarrollador, pre- via reunión, adoptarán las medidas de corrección oportunas en cada caso.

|  |  |
| --- | --- |
| **FACTOR RIESGO** | **ACCIONES PREVENTIVAS** |

1. PRESSMAN, R. S. *Ingeniería del Software: un enfoque práctico*. 2005, McGraw-Hill
2. BOHEM B. ["A Spiral Model of Software Development and Enhancement](http://csse.usc.edu/csse/TECHRPTS/1988/usccse88-500/usccse88-500.pdf)". *IEEE Computer*, IEEE, 1988.

|  |  |
| --- | --- |
| **FACTOR RIESGO** | **ACCIONES PREVENTIVAS** |
| Pérdida de información | Copias de seguridad automatizadas en servidor, con ciclo incremental diario, a  partir de las herramientas ofrecidas por SO y MySQL. Uso del recurso proporcio- nado por la instalación RAID 5 del servidor. |
| Pérdida de código | Copias de seguridad automatizadas en servidor, con ciclo incremental diario, a  partir de las herramientas ofrecidas por SO. Uso del recurso proporcionado por la instalación RAID 5 del servidor. |
| Acceso no deseado a información | Uso del API WP para asegurar la identificación de cualquier usuario externo. |
| Falta de competencia adecuada de miembros de RRHH | Revisión de vinculación laboral y/o académica con los miembros afectados. Re- novación de plantilla. |
| Extracción de información no autorizada | Durante proceso Exportación. Aseguramiento de las transacciones XML-RPC,  facilitadas por API WP, mediante clave asimétrica única por usuario autorizado para la extracción y operación. |

# Verificación de la calidad

Para la verificación de la calidad se recurre al modelo popularizado por PMBOK, en una versión simplificada donde solamente se toman en cuenta determinados parámetros, tanto desde el punto de vista CL como desde el de Desarrollador.

El *Project Management Institute* (PMI) distingue tres procesos en la gestión de la calidad (GC) de un proyecto, que EF procura seguir.

**Planificación [GCP]**: Identificación de los requisitos de calidad y/o normas del proyecto, docu- mentando la manera en que se demostrará el cumplimiento de los mismos.

**Aseguramiento [GCA]**: Auditoría de los requisitos de calidad y los resultados obtenidos a partir de medidas de control de calidad, a fin de garantizar que se utilicen definiciones operacionales y normas adecuadas. También implica el perfeccionamiento y la mejora del proceso con el fin de eliminar las actividades que no agreguen valor.

**Control [GCC]**: Monitorización y registro de los resultados de la ejecución de las actividades de calidad, a fin de evaluar el desempeño y recomendar cambios necesarios. Permite validar que los entregables y el trabajo cumplen con los requisitos especificados.

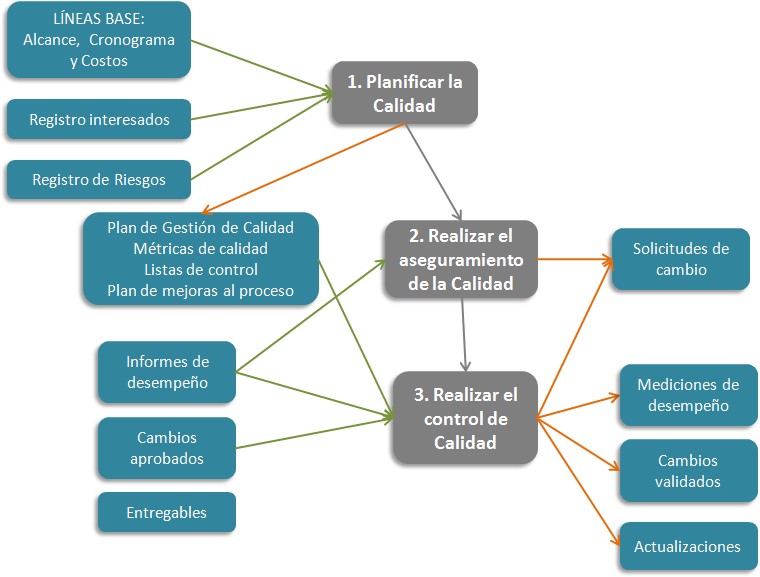


Ilustración 8. Procesos GC según PMI

EF considera que GC es una herramienta esencial para medir el grado de éxito, y por ello imple- menta el núcleo de los tres procesos mencionados a través de diversos mecanismos. GCP está presente tanto en el propio diseño del proyecto como en la arquitectura y la funcionalidad interna de EF. GCA se ha implementado mediante una política de información continua sobre la calidad y

el estado del proyecto. GCC utiliza un conjunto de métricas para obtener una valoración final del nivel de calidad.

Además de los establecidos por PBMOK, y dada la naturaleza científica y académica tanto de EF como de su matriz EX, además de la cualidad del proyecto de ser un PDS, existen condicionantes de calidad adicionales. En la siguiente tabla se detallan las acciones de verificación que se reali- zan para culminar cada proceso.

|  |  |  |  |
| --- | --- | --- | --- |
| **PROCESO** | **ACCIONES** | | **EN PDSDT** |
| Planificación | **1** | **Cumplimiento de restricciones y suposiciones**. Se verifica que todos  los puntos establecidos se cumplen | [2.2](#_bookmark4) |
| **2** | **Cumplimiento de requisitos**. Se verifica que todos los requisitos iniciales se cumplen | [8.2](#_bookmark22) |
| **3** | **Cumplimiento del plan de fases**. Se verifica que los entregables cumplen  el plan establecido. | [4.2.1](#_bookmark14) |
| **4** | **Cumplimiento de tabla de esfuerzo**. Se verifica que la grabación alcanza las cifras estipuladas | [4.1](#_bookmark12) |
| Aseguramiento | **5** | **SCRUM. Reuniones de Sprint**. Se discute sobre la información de errores  individual. Se discute sobre grado de cumplimiento de esfuerzo | [2.6](#_bookmark5) |
| Control | **6** | **Información en tiempo real. Grado de cumplimiento de esfuerzo**. En  interfaz back-end. Accesible a todos los usuarios WP | [10.1](#_bookmark26) |
| **7** | **Información en tiempo real. Número de errores por actor**. En interfaz  back-end. Accesible a todos los usuarios WP | [10.1](#_bookmark26) |
| **8** | **Suministro de información. Errores formales individuales por actor**.  En interfaz back-end. Accesible a todos los usuarios WP | [10.1](#_bookmark26) |
| Condicionantes adicionales | **9** | **Cobertura geográfica mínima**. Dado el alcance del proyecto EX (ámbito europeo), se verifica que las zonas geográficas previstas son cubiertas | N/A |
| **10** | **Disponibilidad SW**. Se verifica y monitoriza el ancho de banda y el acce-  so al servidor durante el tiempo de duración del proyecto | [5](#_bookmark17) |
| **11** | **Calidad SW**. Se verifica la calidad del código entregado mediante herra- mientas de métrica estándar | [11](#_bookmark27) |

## Evaluación del nivel de calidad

Con el fin de conocer el grado de calidad alcanzado por EF en su estado en el momento de entre- gar el artefacto PDSDT actual, a continuación se detallan los procesos GC considerados, las ac- ciones verificadoras realizadas y los resultados obtenidos. En primer lugar se citan los requisitos GC mensurables [RGC] mediante técnicas cuantificadoras, las herramientas empleadas, su ubica- ción en EF y los resultados a la fecha de entrega de la versión actual de PDSDT. En la tabla final y en la descripción de las herramientas se especifican los condicionantes de calidad y el nivel de cumplimiento [NC] (0 >= nivel de cumplimiento <= 1).

* + 1. Herramientas de medición de calidad

|  |  |
| --- | --- |
| Nombre | **1 Grado global de finalización** |
| Procedencia | Acción 1 |
| Herramienta | Cálculo global de nivel de finalización |
| Procedimiento | Detallado en [4.1](#_bookmark12) |
| Nivel debido | 54% |
| Nivel alcanzado | 60% |
| Cumplimiento NC | 1 |

|  |  |
| --- | --- |
| Nombre | **2 Número de errores global y por actor** |
| Procedencia | Acción 7 |

|  |  |
| --- | --- |
| Herramienta | Cálculo global de nivel de errores y cálculo individual por actor |
| Procedimiento | Detallado en [11.3](#_bookmark29) |
| Nivel debido | 5% |
| Nivel alcanzado | 9% |
| Cumplimiento NC | 0,6 |

|  |  |
| --- | --- |
| Nombre | **3 Cobertura geográfica mínima** |
| Procedencia | Acción 9 |
| Herramienta | Informe específico (ver [15)](#_bookmark34) |
| Procedimiento | Especificado por CL. Verificación de que está cubierto el 100% de los municipios >= 20.000 habi-  tantes en España |
| Nivel debido | 100% |
| Nivel alcanzado | 94% |
| Cumplimiento NC | 0,9 |

|  |  |
| --- | --- |
| Nombre | **4 Disponibilidad SW** |
| Procedencia | Acción 10 |
| Herramienta | Informe específico (ver [15)](#_bookmark34) |
| Procedimiento | Especificado por Desarrollador. Nivel de disponibilidad de acceso al servidor y funcionalidad de  aplicación de 99,9% |
| Nivel debido | 99,9% |
| Nivel alcanzado | 99,8% |
| Cumplimiento NC | 1 |

|  |  |
| --- | --- |
| Nombre | **5 Calidad SW** |
| Procedencia | Acción 11 |
| Herramienta | Informe específico (ver [15)](#_bookmark34) |
| Procedimiento | Especificado por Desarrollador. Complejidad ciclomática por LLOC <= 0,35 |
| Nivel debido | 0,35 |
| Nivel alcanzado | 0,38 |
| Cumplimiento NC | 0,9 |

* + 1. Matriz de control de calidad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROCESO** | **ACCIÓN** | | | **CUMP** |
| Planificación | 1 | Restricciones y suposiciones | Verificado por CL y herramienta 1 | 1 |
| Planificación | 2 | Requisitos | Verificado por CL | 1 |
| Planificación | 3 | Plan de fases | Verificado por CL | 1 |
| Planificación | 4 | Tabla de esfuerzo | Verificado por CL y herramienta 1 | 1 |
| Aseguramiento | 5 | Reuniones Sprint | Verificado por CL | 1 |
| Control | 7 | Errores por actor | Verificado por CL y herramienta 2 | 0,6 |
| Condicionantes  adicionales | 9 | Cobertura geográfica | Verificado por CL y herramienta 3 | 0,9 |
| Condicionantes  adicionales | 10 | Disponibilidad SW | Verificado por Desarrollador y herra-  mienta 4 | 1 |
| Condicionantes adicionales | 11 | Calidad SW | Verificado por Desarrollador y herra- mienta 5 | 0,9 |
| **GRADO TOTAL DE CALIDAD ALCANZADA A FECHA DE ENTREGA DE PDSDT** | | | | **0,93** |

En consecuencia, y a la vista de los resultados obtenidos, el grado de cumplimiento alcanzado (93% de los objetivos previstos) se considera **satisfactorio** desde el punto de vista de PDSDT.

* + 1. Informe de errores

La necesidad de implementar un adecuado seguimiento de los errores se solventa mediante el es- tablecimiento de un formulario de notificación, categorizado y con campos de cumplimentación por defecto (solo lectura) o selectiva, que es enviado automáticamente por correo electrónico a la ad- ministración del sistema, y al que se puede acceder desde cualquier página del interfaz *front-end*.

También se ha establecido un formulario de contacto simple que permite, con un procedimiento similar, obtener información o enviar sugerencias a la administración del sistema.

* + 1. Evaluación

EF establece un mecanismo de evaluación mediante un formulario, que se muestra de forma sin- gularizada para cualquier página del interfaz *front-end*, que conserva la valoración que cada usua- rio de tipo suscriptor (con acceso a EF) realiza de ella, mediante un sistema de puntuación simple (mostrado en la tabla siguiente) que se basa en cuatro aspectos, con un rango posible entre un mínimo de 0 (cero) y un máximo de 36. Los resultados se conservan de manera individualizada y EF calcula los promedios y una valoración global del aplicativo. A tal información solamente tendrá acceso el administrador del sistema.

|  |  |  |
| --- | --- | --- |
| **ASPECTO** | **DESCRIPCIÓN** | **PUNTUACIÓN** |
| Diseño | El diseño de los elementos gráficos y textuales de la página es correcto y  adecuado a su funcionalidad. | De 0 a 9 |
| Funcionalidad | La funcionalidad que debe ofrecer la página sirve para cumplir su cometido  en el conjunto de acciones propias de la aplicación. | De 0 a 9 |
| Utilidad | La utilidad que ofrece la página es adecuada (sirve para cumplir un propósi- to necesario para el buen funcionamiento de la aplicación). | De 0 a 9 |
| Velocidad | La velocidad con la que la página se genera u ofrece resultados es acepta-  ble. | De 0 a 9 |

# Anexo gráfico

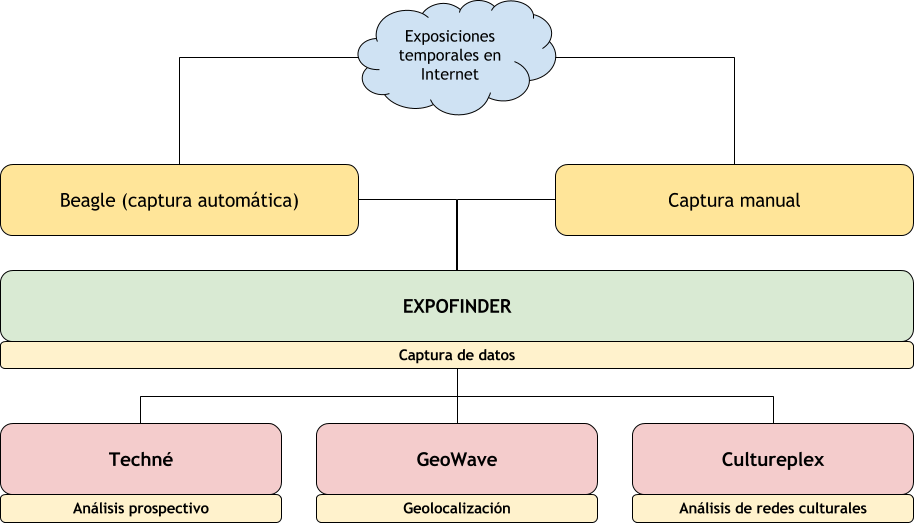


Ilustración 9. EF como subproyecto de EX. Mapa conceptual

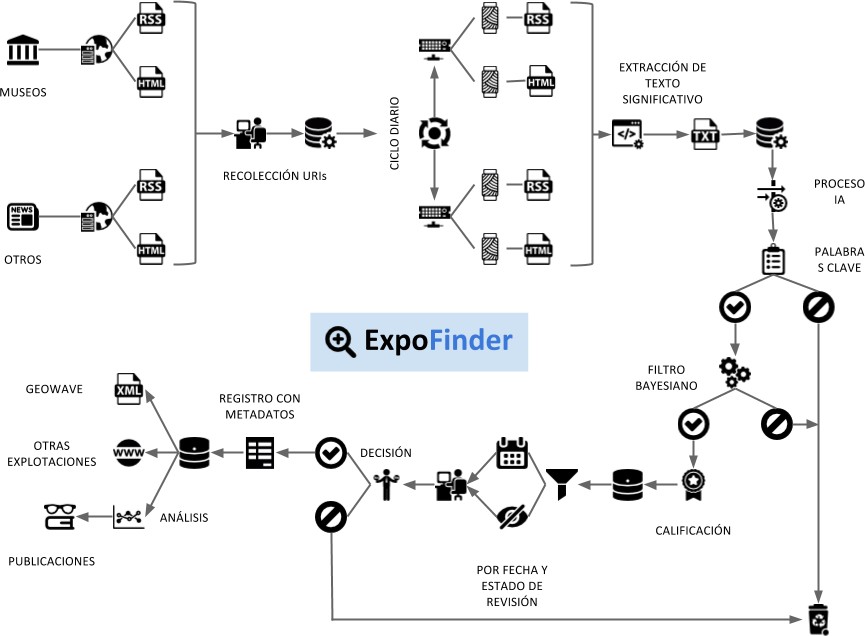


Ilustración 10. EF: Esquema de flujo simplificado. Mapa conceptual

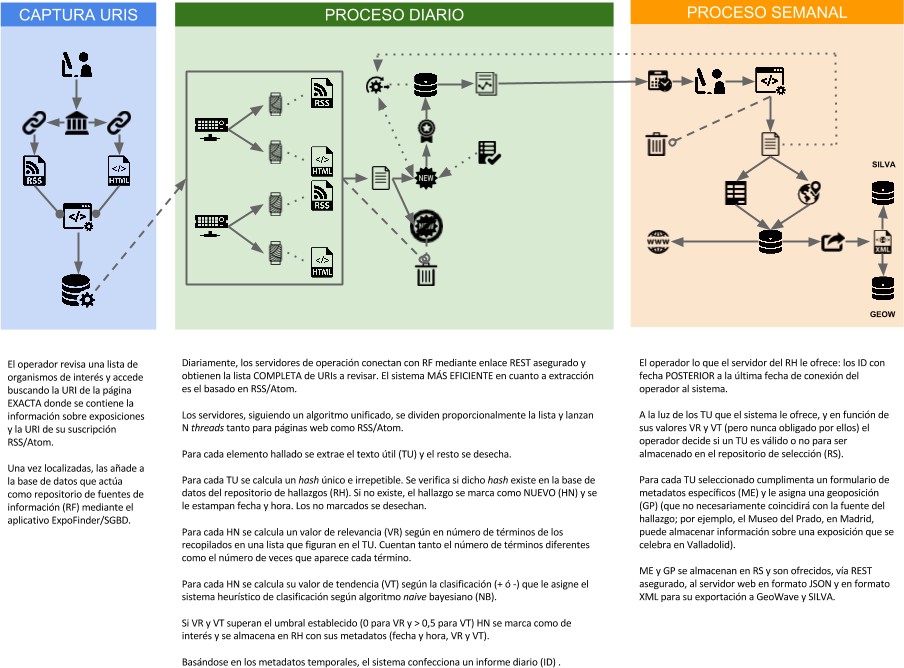


Ilustración 11. EF. Esquema procesual. Mapa conceptual

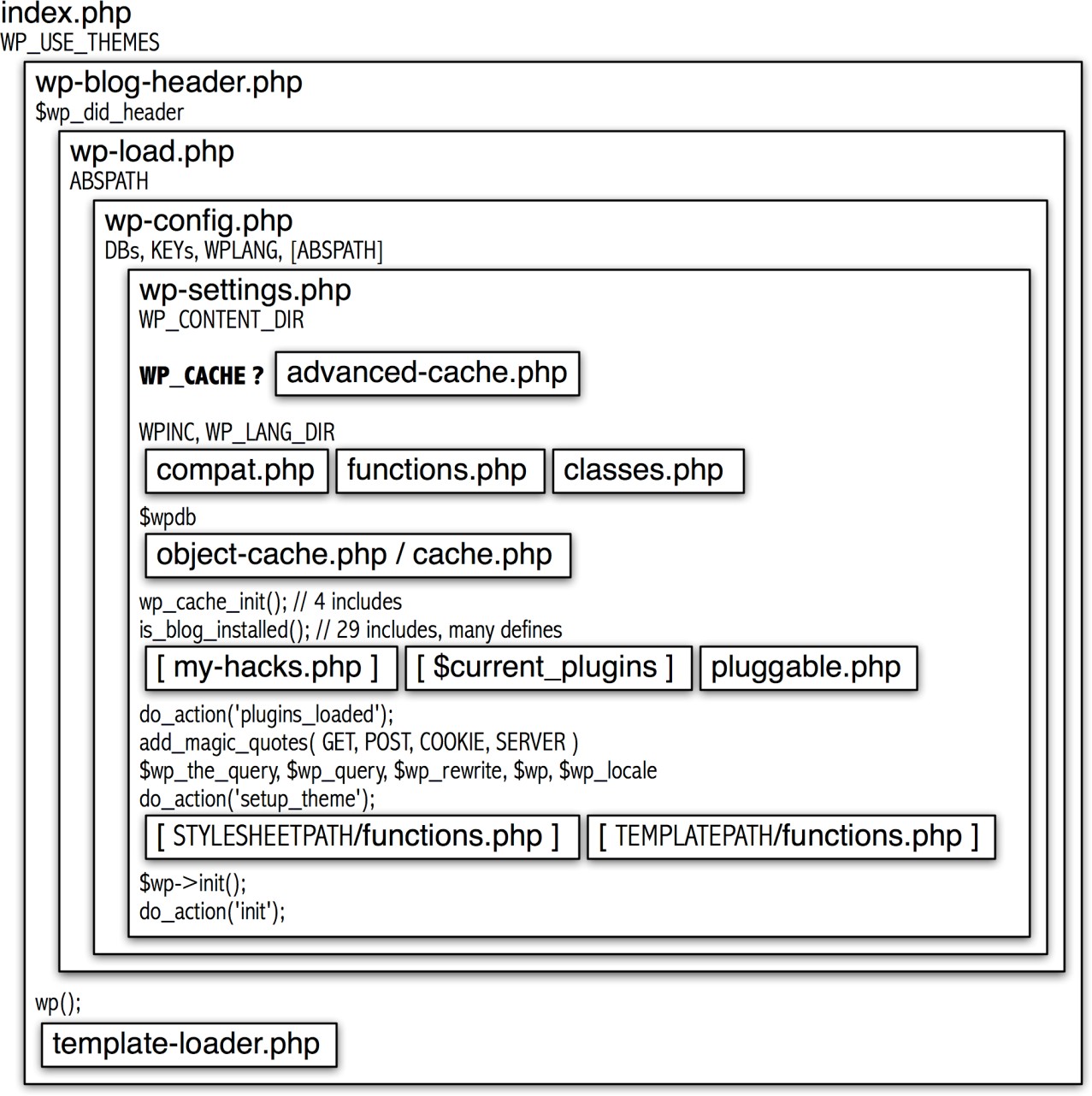


Ilustración 12. Stack de WordPress

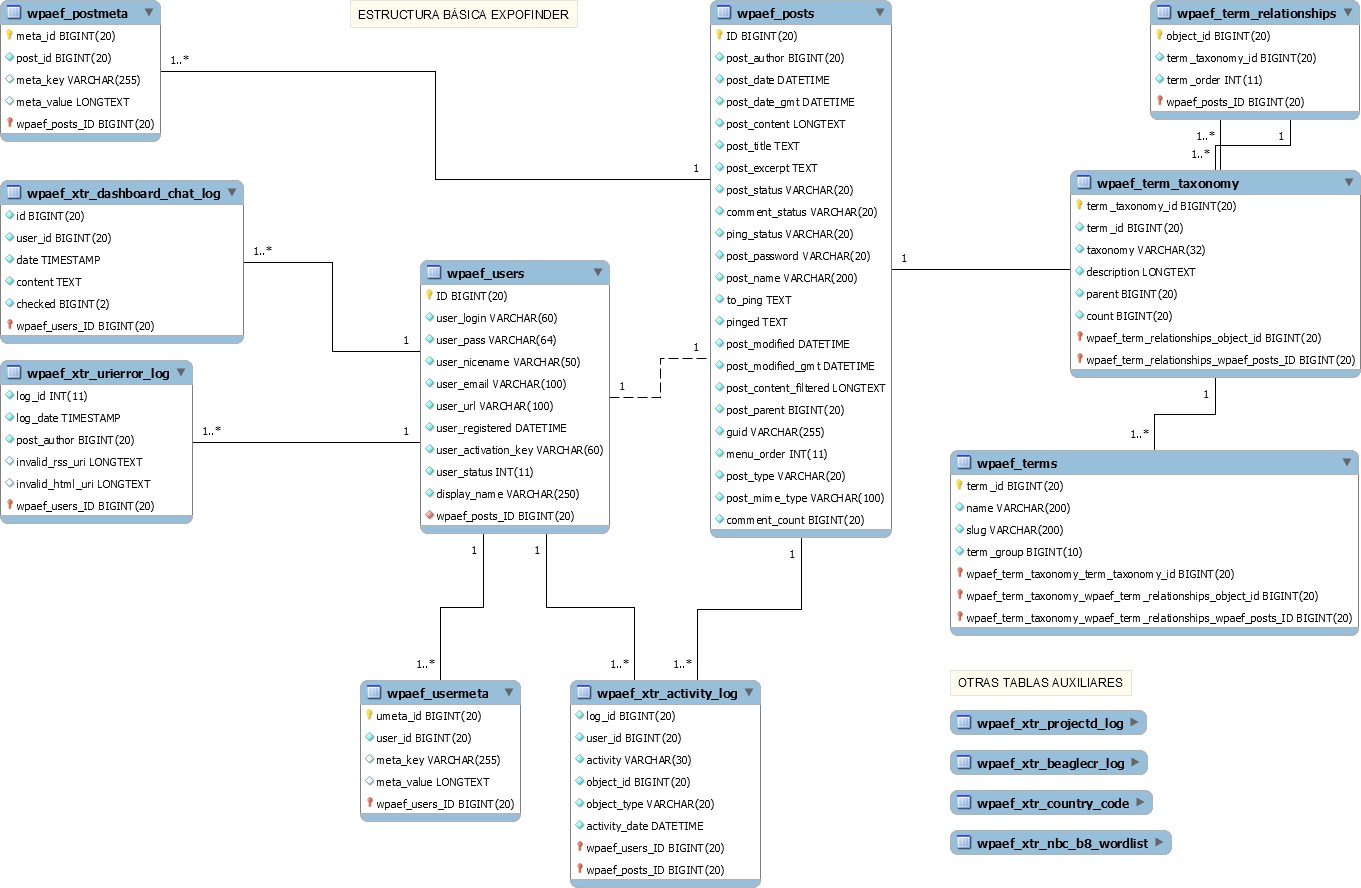


Ilustración 13. EF. Modelo de datos

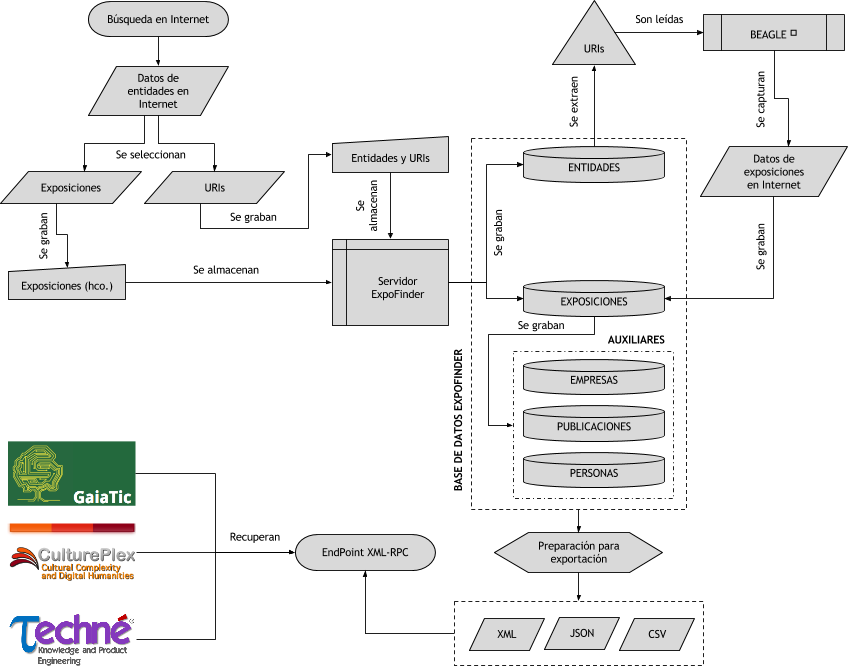


Ilustración 14. Relaciones entre los 3 sistemas principales de EF

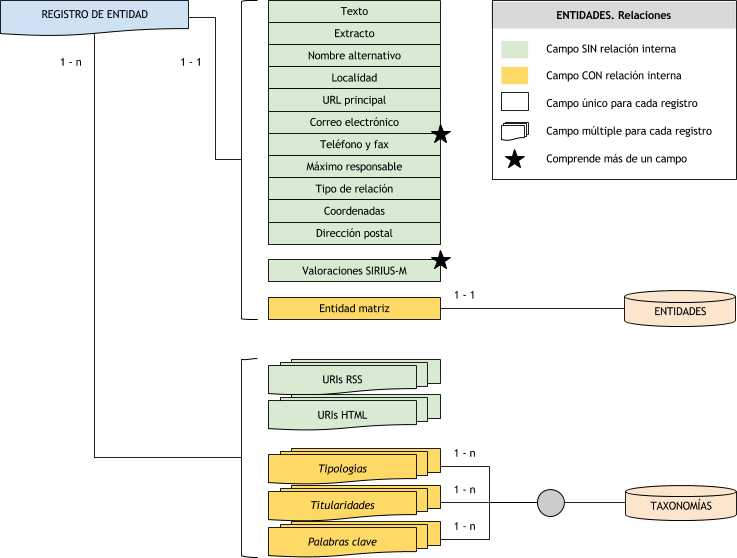


Ilustración 15. Registro ENTIDADES. Relaciones

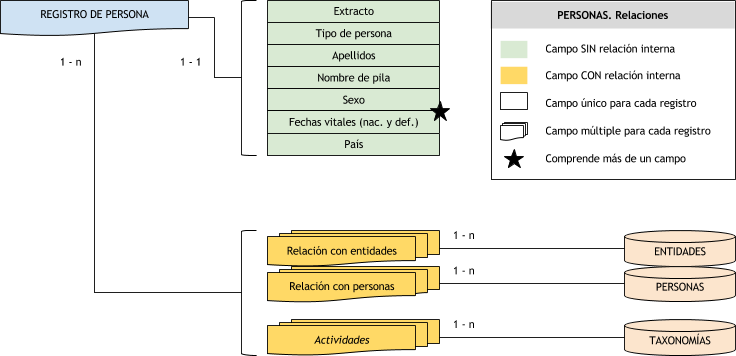


Ilustración 16. Registro PERSON. Relaciones

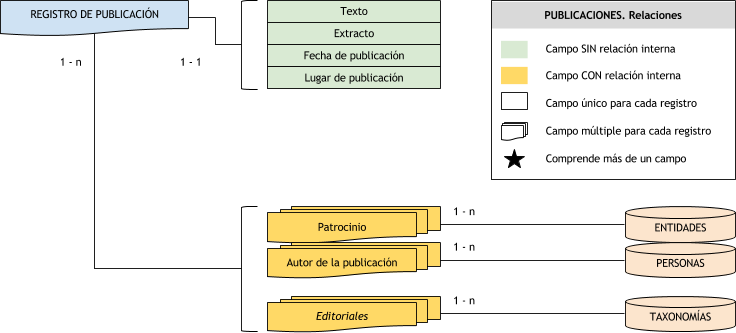


Ilustración 17. Registro PUBLICACIONES. Relaciones

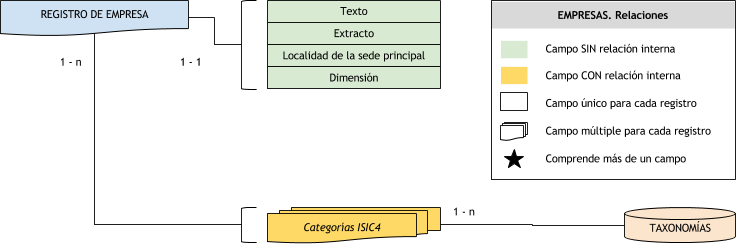


Ilustración 18. Registro EMPRESA. Relaciones

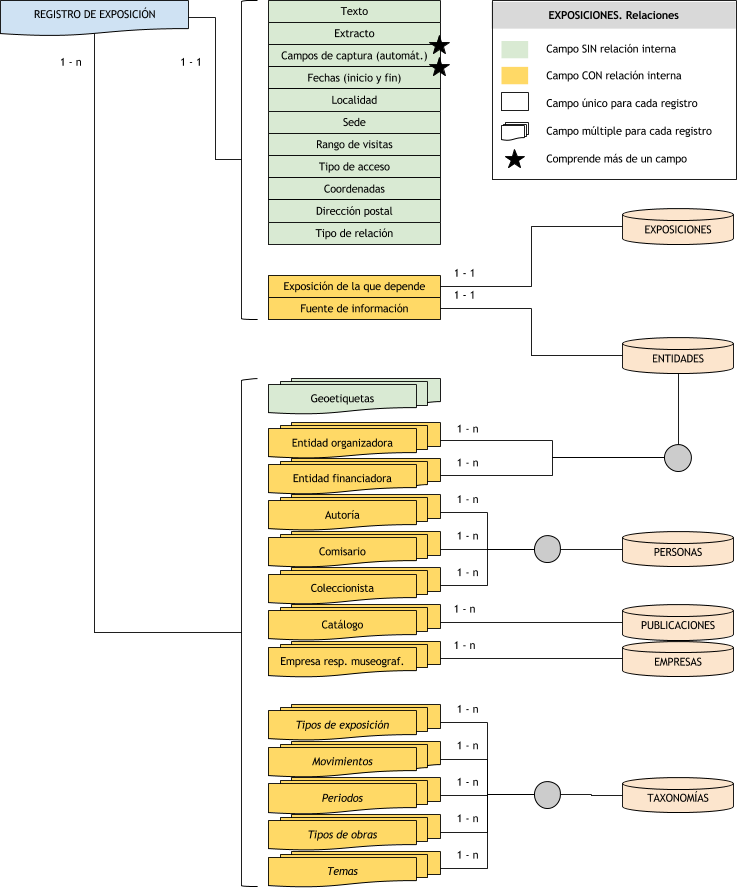


Ilustración 19. Registro EXPOSICIÓN. Relaciones

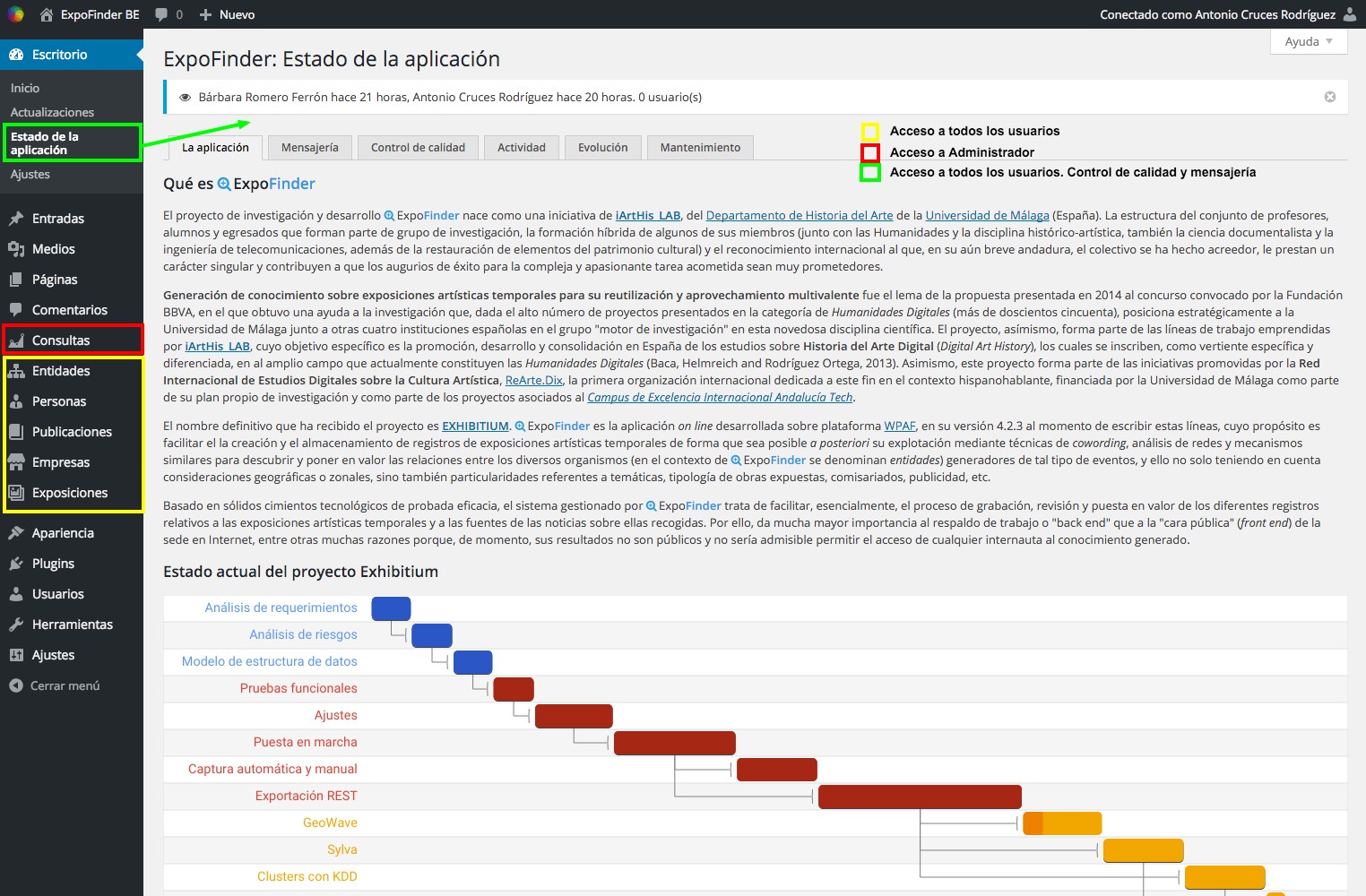


Ilustración 20. Interfaz back-end

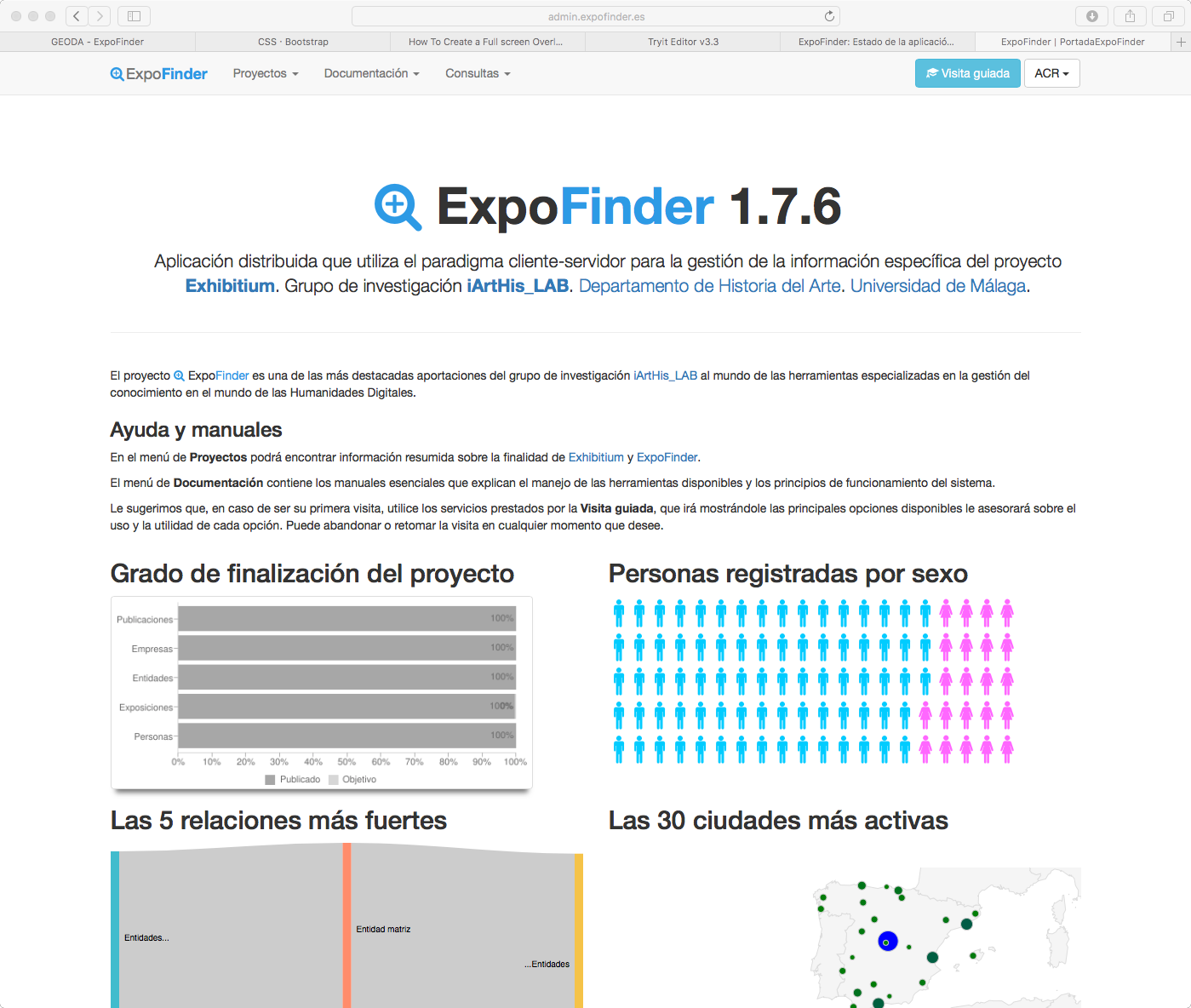
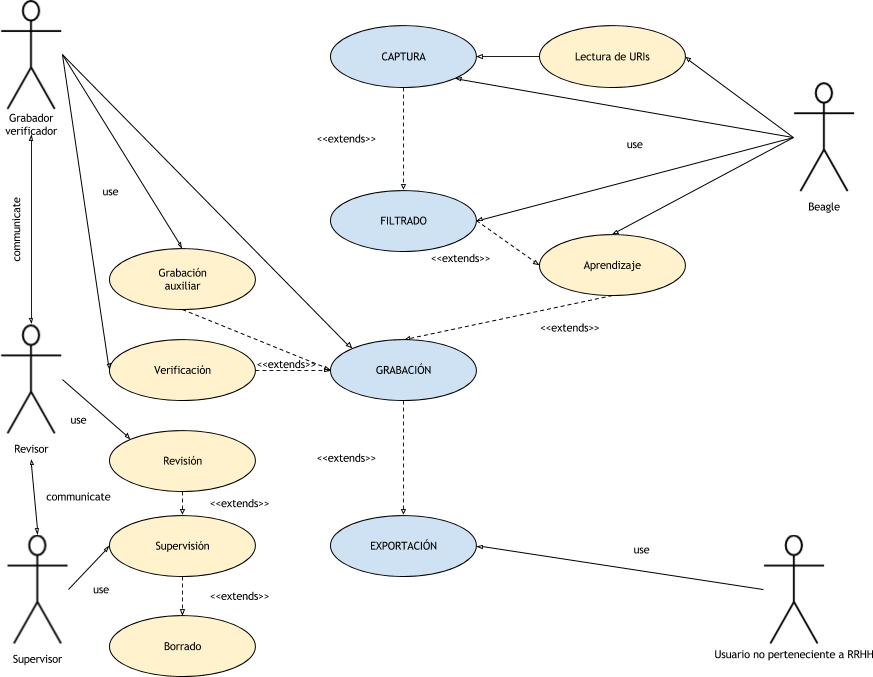


Ilustración 21. Interfaz front-end



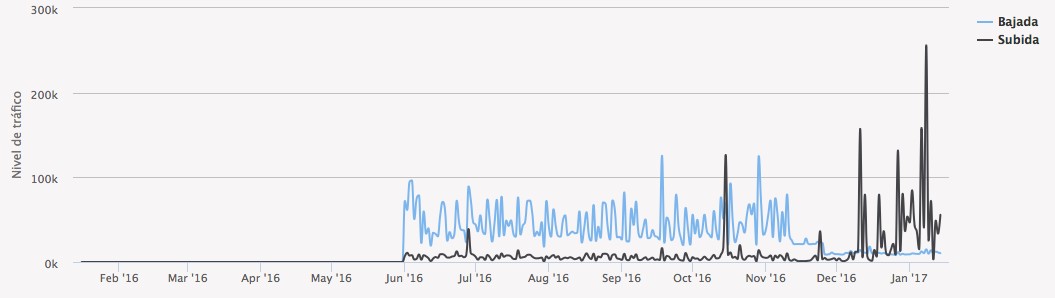
Ilustración 22. Diagrama UML genérico. Modelo de casos de uso

Ilustración 23. Servidor admin.expofinder.es. Tráfico últimos 12 meses

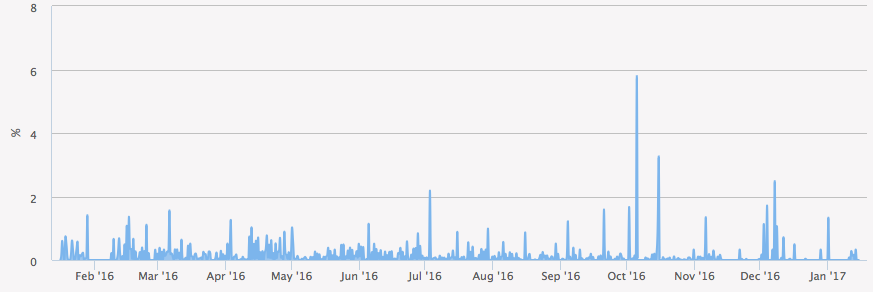
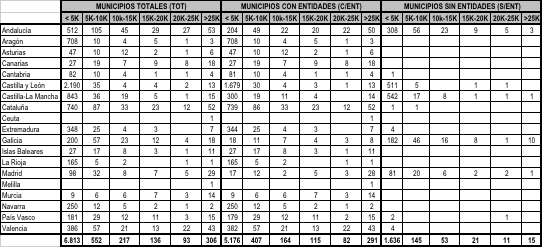


Ilustración 24. Servidor admin.expofinder.es. Uso CPU últimos 12 meses

# Anexo documental

## Informe sobre cobertura geográfica

Datos analizados directamente de SGBD a 30 de diciembre de 2015. Datos sobre municipios pro- cedentes de Instituto Nacional de Estadística (INE). Padrón consolidado 2014.

## Informe sobre LOC y complejidad SW

Herramienta de cálculo: PHP LOC 2.1.3 (Sebastian Bergmann). Fecha de captura: 14 de enero de 2017.

|  |  |  |
| --- | --- | --- |
| Directories | 10 |  |
| Files | 42 |
| Size  Lines of Code (LOC) | 26651 |
| Comment Lines of Code (CLOC) | 4046 | (15.18%) |
| Non-Comment Lines of Code (NCLOC) | 22605 | (84.82%) |
| Logical Lines of Code (LLOC) | 7033 | (26.39%) |
| Classes | 2592 | (36.85%) |
| Average Class Length | 83 |  |
| Minimum Class Length | 4 |  |
| Maximum Class Length | 370 |  |
| Average Method Length | 8 |  |
| Minimum Method Length | 0 |  |
| Maximum Method Length | 108 |  |
| Functions | 3596 | (51.13%) |
| Average Function Length | 11 |  |
| Not in classes or functions | 845 | (12.01%) |
| Cyclomatic Complexity | | |
| Average Complexity per LLOC | 0.38 | |
| Average Complexity per Class | 39.87 | |
| Minimum Class Complexity | 2.00 | |

Maximum Class Complexity Average Complexity per Method

Minimum Method Complexity

Maximum Method Complexity

203.00

5.11

1.00

58.00

Dependencies

Structure

|  |  |  |
| --- | --- | --- |
| Global Accesses | 2677 |  |
| Global Constants | 2249 | (84.01%) |
| Global Variables | 190 | (7.10%) |
| Super-Global Variables | 238 | (8.89%) |
| Attribute Accesses | 1839 |  |
| Non-Static | 1839 | (100.00%) |
| Static | 0 | (0.00%) |
| Method Calls | 914 |  |
| Non-Static | 859 | (93.98%) |
| Static | 55 | (6.02%) |

|  |  |  |
| --- | --- | --- |
| Namespaces | 0 |  |
| Interfaces | 0 |
| Traits | 0 |
| Classes | 31 |
| Abstract Classes | 2 | (6.45%) |
| Concrete Classes | 29 | (93.55%) |
| Methods Scope  Non-Static Methods | 268  233 | (86.94%) |
| Static Methods | 35 | (13.06%) |
| Visibility | | |
| Public Methods | 189 | (70.52%) |
| Non-Public Methods | 79 | (29.48%) |
| Functions | 307 |  |
| Named Functions | 299 | (97.39%) |
| Anonymous Functions | 8 | (2.61%) |
| Constants | 119 |  |
| Global Constants | 85 | (71.43%) |
| Class Constants | 34 | (28.57%) |

## Normativa aplicada

La confección del artefacto PDSDT ha seguido la normativa sugerida y especificada en ISO/IEC 12207.

## Bibliografía

A. Piscitelli. “Humanidades digitales y nuevo normal educativo”. *TELOS (Cuadernos de Comunica- ción e Innovación)*, pp. 10 y ss., junio-septiembre 2015.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE std 610.12-1990, 1990.

INTECO. *Ingeniería de Software: metodologías y ciclos de vida. Laboratorio Nacional de Calidad del Software*. España: Instituto Nacional de Tecnologías de la Comunicación. 2009.

ISO/IEC. *ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes*. EEUU: 2008.

J. Palacio. *Gestión de proyectos Scrum Manager (Scrum Manager I y II). Versión 2.5.1*. España, 2015.

M. Cusmano y M. Selby. *Microsoft Secrets*. EEUU: HarperCollins Business, 1997.

M. Poppendieck y T. Poppendieck. *Lean Development Software: An Agile Toolkit for Software De- velopment Managers (The Agile Software Development Series)*. EEUU: Addison-Wesley, 2003.

PMI. *A Guide to the Project Management Body of Knowledge*. EEUU, 2012.

T. McFarlin. *Using WordPress for Web Application Development: An Introduction*. 2013.<http://code.tutsplus.com/tutorials/using-wordpress-for-web-application-development-an-> introduction--wp-33828. Consultado 20-1-2015.