Pliego de Prescripciones Técnicas para la Contratación del Servicio de Evolución Tecnológica y Funcional de las Aplicaciones Informáticas de la Universitat de València.

ÍNDICE

Contenido

1.-Antecedentes

La Universitat de València (en adelante UV) dispone una amplia cartera de aplicaciones, que da soporte y abarca tanto la propia infraestructura del entorno tecnológico, como la gestión académica (oficial y propia), investigación, económica, personal, reservas de espacios y administración electrónica, entre otras. Estas aplicaciones dan respuesta a cambios organizativos, normativos y procedimentales de las áreas donde aplican, así como de nuevos requerimientos, exigiendo por tanto de un esfuerzo continuo de evolución.

También dispone de un amplio conjunto de procedimientos para la consulta y explotación de datos, así como de informes (destinados a la dirección de la UV o a otros organismos oficiales) que de forma constante requieren de actualizaciones motivados por obligaciones normativas.

El Servicio de Informática de la Universitat de València (en adelante SIUV) dedica una importante parte de sus recursos a asegurar el correcto funcionamiento de las aplicaciones, pero no dispone de los recursos suficientes para soportar el mantenimiento evolutivo de todas ellas, y es por tanto objeto de esta licitación la contratación de los servicios de evolución funcional y tecnológica de su cartera de aplicaciones y consulta de datos.

2.- Objeto del contrato

El objeto del contrato es la prestación del servicio de evolución tecnológica y funcional del conjunto de aplicaciones de gestión, portal de acceso a servicios, procedimientos de la Sede Electrónica y de explotación de datos de la Universitat de València para asegurar su correcto funcionamiento y su evolución para dar servicio a los diferentes órganos y unidades administrativas de la Universitat de València.

Se debe destacar, por lo excepcional de la situación actual, que en el año 2017 se ha finalizado la migración de la plataforma tecnológica de la UV a un nuevo entorno (sistemas físicos, sistemas operativos, bases de datos, etc.), por lo que, dentro de los servicios de evolución tecnológica, objeto de este contrato, se considerarán también tareas de migración de aplicaciones ya existentes a un nuevo escenario. La migración de las aplicaciones se llevará a cabo de acuerdo a la planificación que el SIUV prevea y de acuerdo a la arquitectura de desarrollo software basada en el Anexo Framework_CRUE-TIC de este pliego.

Por último, incidir en que todas las aplicaciones y servicios ofrecidos permitirán ofrecer aplicaciones multilingües, accesibles y eficientes por su usabilidad y cumpliendo la normativa en cuanto seguridad se refiere.

En el objeto del contrato se incluyen tanto aplicaciones de desarrollo propio, como los servicios de integración y comunicación de estas aplicaciones con aplicaciones de terceros y

está organizado en 2 lotes, indicando los desarrollos inicialmente previstos por la Universitat de València y que deberán incluir además todos los cambios normativos y funcionales derivados de la gestión de la Universitat durante el periodo de vigencia de este contrato.

LOTE1: APLICACIONES DE LAS ÁREAS DE INFRAESTRUCTURA, GESTIÓN ACADÉMICA Y UNIVERSITARIA.

A continuación, se describen los conjuntos de aplicaciones integradas en cada una de las áreas asociadas a este lote. Se incluye, a título orientativo, una previsión inicial de tareas a abordar en cada uno de los ámbitos. La ejecución de estas tareas se planificará durante el periodo de duración del contrato. La planificación real podrá verse afectada por nuevas necesidades de la UV o por cambios normativos, por lo que, podrán incluirse nuevas tareas o no ejecutar alguna de las inicialmente previstas.

Con carácter general, para todas las aplicaciones incluidas en este lote, deberá considerarse la realización de tareas orientadas a:

- ✓ Mantenimiento correctivo y evolutivo de las aplicaciones.
- ✓ Actuaciones derivadas de la aplicación en la UV de las leyes 39/2015 de Procedimiento Administrativo Común de las AAPP y 40/2015 de Régimen Jurídico del Sector Público, en las que se establece que la tramitación electrónica de los procedimientos debe constituir la actuación habitual de todas las Administraciones, tanto en su relación con los ciudadanos como en la gestión interna y en los intercambios de información entre distintos organismos.
- ✓ Desarrollo de servicios de integración y comunicación con otras aplicaciones de la UV o de organismos externos que así lo requieran.
- ✓ Generación de informes y mecanismos que faciliten la explotación de datos.
- ✓ Migración/Adaptación de aplicaciones a la nueva arquitectura de desarrollo.

⇨ A) ÁREA DE INFRAESTRUCTURA DE APLICACIONES

1.- Aplicaciones de infraestructura. Comprende el conjunto de servicios comunes de las aplicaciones, aplicación de gestión de perfiles de usuarios en las aplicaciones, portal de acceso a las aplicaciones, procedimientos para obtención de datos y listados en base a consultas parametrizadas. Previsión de tareas a realizar en este ámbito:

- ✓ Nuevas funcionalidades en la aplicación de usuarios de aplicaciones: Definición de nuevos roles, perfiles. Integración con la aplicación de gestión de recursos humanos y

con el gestor de identidades. Funcionalidades del administrador de cada unidad administrativa. Mantenimiento de tablas generales y parámetros de las aplicaciones.

- ✓ Definición y adaptación de las hojas de estilo en todas las aplicaciones.
- ✓ Unificación del acceso a manuales de usuario y notificación de incidencias.
- ✓ Desarrollo de servicios web para la integración con aplicaciones que lo requieran.
- ✓ Mantenimiento de las herramientas de desarrollo (incluyendo Rational *Business Developer).*

En esta área se podrán incluir nuevos servicios, aplicaciones o herramientas de desarrollo que puedan ser de utilidad al conjunto de aplicaciones de la Universitat.

⇨ B) ÁREA DE GESTIÓN ACADÉMICA

2.-Planes de Estudio. Gestión y mantenimiento de los Planes de Estudio correspondientes a los estudios y titulaciones de la Universitat. Mantiene la estructura, vigencia, tipología de estudios, asignaturas, estructura y organización de estudios universitarios correspondientes a los antiguos planes conducentes a títulos de Licenciaturas, Diplomaturas, Doctorado y a los actuales de Grado, Máster y Postgrado y los nuevos estudios de Doctorado. Previsión de tareas a realizar en este ámbito:

- ✓ Adaptación a la nueva estructura de Planes de Estudio: Gestión de nuevas estructuras, Agencias evaluadoras, Convenios, temporalidad y vigencia. Integración de datos entre Planes y Verifica. Definición de versiones de los planes.

3.- Guías Docentes de asignaturas. Gestión y mantenimiento y publicación de las guías docentes de las asignaturas.

- ✓ Nuevos desarrollos para adaptar las guías docentes a los procesos de seguimiento de titulaciones de las agencias de evaluación estatal y autonómica.
- ✓ Funcionalidades de exportación masiva de guías docentes de titulación.
- ✓ Almacenamiento y consulta de guías docentes en el gestor documental de la UV.

4.- Oferta de Curso Académico. Gestión y mantenimiento de las Titulaciones ofertadas, objetivos académicos, asignaturas-grupos y subgrupos según la modalidad de la docencia, restricciones, itinerarios, gestión de capacidades y cupos para matrícula, agrupaciones y conjuntos horarios. Visión por Centro y Titulación.

- ✓ Nuevas funcionalidades en la Oferta de Curso Académico: Agrupaciones, ayudas sobre

campos modificables al cerrar la oferta, nuevas funciones de validación.

5.- Gestión de calendario, espacios y reservas. Gestión y mantenimiento de las estructuras que representan el catálogo de espacios y su uso. Reservas de espacios para docencia y otras actividades. Mantenimiento de Calendarios, Unidades gestoras y gestión de las reservas. Integración con la docencia, exámenes y otras actividades.

- ✓ Nuevas consultas en línea sobre ocupación de espacios.

6.- Plan de Ordenación Docente. Gestión y mantenimiento de las estructuras para la gestión, definición y seguimiento del plan de ordenación docente de cada curso académico. Carga Docente de los Departamentos. Visión por Departamento y Centro. Gestión de las dedicaciones del personal docente.

- ✓ Nuevas funcionalidades de informativas orientadas a centro, departamento y profesorado.

7.- Preinscripción y acceso a estudios universitarios. Integración de las estructuras de estudiantes de nuevo ingreso a los estudios universitarios, datos de preinscripción, integración con la aplicación de llamamientos para gestión de listas de espera. Preinscripción a estudios de máster y postgrado (master y doctorado). Integración con los procedimientos de ordenación de la matrícula y con las solicitudes de cursos de adaptación a grado.

- ✓ Nuevos desarrollos para integración de la aplicación con los procedimientos de administración electrónica.

8.- AutoMatrícula y Gestión de Matrícula. Gestión y mantenimiento de la aplicación de matrícula para todos los estudios oficiales de la Universitat de València.

- ✓ Revisión anual de adaptación a nuevos criterios de matrícula.
- ✓ Integración con datos de preinscripción, ordenación y procedimientos de administración electrónica.
- ✓ Mantenimiento y desarrollo de procedimientos para la explotación de la información.
- ✓ Integración con Servicios de Interoperabilidad de otras Administraciones Públicas.

9.- Programas de movilidad. Gestión de las solicitudes para programas de movilidad. Gestión de convocatorias y programas: Erasmus, Promoe, SICUE, Erasmus Practicum, etc. Gestión de Coordinadores. Contrato de estudios. Gestión de las becas y pagos.

- ✓ Nuevos desarrollos en Aplicación de Movilidad: nuevos Programas Internacionales,

mejoras en la explotación de datos, justificaciones y listados. Gestión de becarios de movilidad. Incoming: Certificados, integración de datos y mejoras. Outgoing: Revocación de renuncias, cambios de titulación, respetar dependencias entre movilidad, programas y solicitudes. Gestión de Coordinadores.

10.- Gestión Expediente Académico. Gestión y mantenimiento de las estructuras que conforman el expediente académico de los estudiantes: datos de inicio o formación del expediente, identificación de los estudios y titulaciones, información generada por las aplicaciones de matrícula, calificaciones de las actas y resoluciones de adaptación, convalidación, programas internacionales, reconocimientos, etc. Gestión de simultaneidad de estudios, visión del expediente, itinerarios, rendimiento académico, finalización de estudios.

- ✓ Nuevos desarrollos para la notificación de eventos vinculados con cambios en el expediente de los estudiantes a través de la plataforma de mensajería de la universidad.
- ✓ Ampliación del expediente para incorporar datos referentes a conocimientos de idiomas, grupos específicos de docencia, realización de prácticas externas,... de acuerdo a la aplicación normativa del SET.

11.- Certificaciones. Mantenimiento y definición de las certificaciones académicas sobre el expediente, matrícula, situaciones especiales, ayudas al estudio, pago de tasas, calificaciones y nota media y demás certificados.

- ✓ Cambios en los bloques del certificado, presentación y cambios de formato.
- ✓ Certificación del idioma de docencia.
- ✓ Incorporación de todos los certificados a la Sede electrónica.

12.- Gestión de tasas y recibos. Mantenimiento de las estructuras para la gestión de los precios públicos correspondientes a las tasas de matrícula y secretaría. Mantenimiento de grupos de tasa, conceptos, tipos de exención y precios. Valoración de la matrícula. Gestión de recibos de los estudiantes por lugar de gestión. Generación de remesas y gestión de cobros. Justificación de cobros y enlaces contables.

- ✓ Nuevos desarrollos para adaptar la aplicación a los cambios en la gestión de los recibos de matrícula y secretaria.
- ✓ Simplificación del proceso de actualización de tasas.

13.- Gestión de Títulos y Suplemento Europeo al Título. Mantenimiento de las estructuras correspondientes a los procedimientos de solicitud, depósito, inscripción, tramitación al registro nacional de títulos, e impresión de títulos universitarios.

- ✓ Mejoras al mantenimiento de títulos. Estudios previos en doctorado. Normalización de municipios de nacimiento con la codificación del Ministerio. Controles de alta repetida, validaciones de datos requeridos (apellidos, nombre y pasaporte).
- ✓ Modificaciones en el mantenimiento de títulos de doctores. Mantenimiento de las menciones. Generación del SET. Integración del pago con tarjeta.

14.- Gestión de Tribunales, Tesis, Trabajos fin de Estudios. Aplicación para el mantenimiento de las estructuras de datos y procedimientos de gestión de los tribunales de tesis y trabajos de finalización de estudios.

- ✓ Nuevos desarrollos para adaptar la aplicación a los estudios de Grado, Master y Doctorado, de acuerdo con la normativa actual.

15.- Gestión de Prácticas externas: Aplicación para la gestión de las solicitudes, adjudicaciones, certificación y gestión de las prácticas correspondientes a estudios oficiales.

- ✓ Adaptaciones para la generalización de la aplicación a las prácticas correspondientes a todas las titulaciones.

16.- Actas de evaluación docente. Gestión y mantenimiento para la cumplimentación de las actas oficiales de calificación de las asignaturas impartidas en la Universitat.

- ✓ Incorporación de la firma electrónica.
- ✓ Almacenamiento en el gestor documental.

17.- Aplicación de ordenación de las citas de matrícula. Publicación de los resultados y criterios personalizados sobre la asignación de la cita para la matrícula del curso.

- ✓ Mantenimiento evolutivo y correctivo de la aplicación.

18.- Aplicación de gestión de becas. Mantenimiento de becas solicitadas, situación de tramitación y gestión de cobros.

- ✓ Mantenimiento evolutivo y correctivo de la aplicación.

19.- Aplicación de gestión de estudios interuniversitarios. Intercambio de datos académicos y personales de estudiantes de titulaciones interuniversitarias en las que participa la UV.

- ✓ Mantenimiento evolutivo y correctivo de la aplicación.

En esta área se podrán incluir nuevas aplicaciones del mismo ámbito que la Universitat requiera desarrollar.

⇨ C) ÁREA DE GESTIÓN UNIVERSITARIA

20.- Gestión de cursos y actividades complementarias en la formación universitaria. Aplicación para la gestión, oferta y matricula de cursos y actividades ofertadas por diferentes Servicios de la Universitat como actividades complementarias para los estudiantes.

- ✓ Oferta de Actividades y Cursos. Incorporación de los cursos en el expediente personal. Integración de la aplicación con la matrícula de actividades y cursos. Nuevas consultas en línea y listados. Certificaciones.
- ✓ Unificación en la misma aplicación de ofertas de cursos y actividades realizadas desde diferentes servicios de la UV.

21.- Gestión de la Formación y Expediente Personal. Mantenimiento de un expediente que incorpore todas las acciones formativas y cursos realizados por el personal de la Universitat en sus distintas modalidades. Solicitud, baremación, matrícula, certificación de dichas actividades.

- ✓ Automatización de los informes basados en las encuestas de evaluación de las actividades.
- ✓ Desarrollo de nuevos certificado y pago de actividades.
- ✓ Integración de actividades y cursos de formación de diversas fuentes (Servei d'Esports, Servei d'Estudiants y Servei de Política Lingüística) e históricos en el expediente de Personal.

22.- Gestión de pagos. Aplicación para el pago de tasas y precios públicos por Internet mediante pasarela de pagos y terminales de pago con tarjeta (TPV).

- ✓ Implantación de terminales con tarjeta y desarrollos para pagos de recibos en las dos modalidades (presencial y no presencial) en aquellas aplicaciones y/o servicios que lo requieran.

23.- *Tarjeta Universitaria.* Gestión y mantenimiento de tarjetas de estudiantes y personal de la Universitat.

- ✓ Adaptaciones a la nueva Tarjeta Universitaria Inteligente.
- ✓ Comunicación de datos con los proveedores.
- ✓ Incorporación de certificados digitales en la tarjeta.

24.- <u>Gestión de la dedicación horaria</u>. Gestión y control de fichajes del personal de la Universitat.

- ✓ Nuevas tipologías de horas: formación.
- ✓ Modificaciones derivadas de cambios en la gestión de personal.

25.- <u>Gestión de Formación del Servei de Política Lingüística (SPL)</u>. Gestión de exámenes y pruebas de nivel, matrícula y certificación en cursos.

- ✓ Migración datos desde el Servicio Estudiantes y transformación de acuerdo a los niveles del Marco de Referencia Europeo.
- ✓ Migración/Adaptación a la nueva arquitectura de desarrollo.
- ✓ Nuevo desarrollo para incorporar el perfil Centro Autoaprendizaje o perfil del profesor para asistencias y notas. Habilitar el acceso de mantenimiento a las Asistencias y las Actas si se trata de Profesor de esa Edición.

26.- <u>Gestión Traducciones</u>. Gestión de peticiones de traducciones y/o correcciones del Servei de Política Lingüística.

- ✓ Mantenimiento evolutivo y correctivo de la aplicación.

27.- <u>Reservas de espacios para instalaciones deportivas</u>. *Gestión y mantenimiento del espacio de los Campos de Deporte de la Universitat de València.*

- ✓ Gestión de solicitudes de reservas.
- ✓ Integración con el catálogo de espacios.
- ✓ Consulta, mantenimiento y confirmación de las reservas realizadas.
- ✓ Pagos con tarjeta e integración con la pasarela de pagos y terminales.

28.- <u>Aplicación para el escrutinio y publicación de resultados de las elecciones a Rector</u>. Gestión y mantenimiento del escrutinio en las elecciones a Rector abierto a toda la comunidad universitaria.

- ✓ Generalización a elecciones en otros órganos de representación.

En esta área se podrán incluir nuevas aplicaciones del mismo ámbito que la Universitat requiera desarrollar.

LOTE 2: APLICACIONES DE LAS ÁREAS DE GESTIÓN ADMINISTRATIVA, ADMINISTRACIÓN ELECTRÓNICA Y GESTIÓN DE LA INVESTIGACIÓN.

A continuación, se describen los conjuntos de aplicaciones integradas en cada una de las áreas asociadas a este lote. Se incluye, a título orientativo, una previsión inicial de tareas a abordar en cada uno de los ámbitos. La ejecución de estas tareas se planificará durante el periodo de duración del contrato. La planificación real podrá verse afectada por nuevas necesidades de la UV o por cambios normativos, por lo que, podrán incluirse nuevas tareas o no ejecutar alguna de las inicialmente previstas.

Con carácter general, para todas las aplicaciones incluidas en este lote, deberá considerarse la realización de tareas orientadas a:

- ✓ Mantenimiento correctivo y evolutivo de las aplicaciones.
- ✓ Adaptaciones derivadas de la aplicación en la UV de las leyes 39/2015 de Procedimiento Administrativo Común de las AAPP y 40/2015 de Régimen Jurídico del Sector Público, en las que se establece que la tramitación electrónica de los procedimientos debe constituir la actuación habitual de todas las Administraciones, tanto en su relación con los ciudadanos como en la gestión interna y en los intercambios de información entre distintos organismos.
- ✓ Desarrollo de servicios de integración y comunicación con otras aplicaciones de la UV que lo requieran.
- ✓ Generación de informes y mecanismos que faciliten la explotación de datos.
- ✓ Migración/Adaptación de aplicaciones a la nueva arquitectura de desarrollo.

⇨ D) ÁREA DE GESTIÓN ADMINISTRATIVA

En esta área, la UV dispone de aplicaciones tanto de desarrollo propio como de terceros que requieren evolutivos y mantenimientos para hacer frente a cambios normativos. La integración de todas estas aplicaciones con otros sistemas de información es también tarea clave en esta área.

29.- GTI (Gestión y Tratamiento de Ingresos). Aplicación destinada a la gestión de los ingresos de la Universitat de Valencia.

- ✓ Integración con otros aplicativos de la Universitat de València.

30.- Servicios de integración con aplicaciones de terceros de gestión administrativa en la UV: SICUV-Gestión Contable (T-Systems), UXXI-Gestión de RRHH (OCU), Licit@-Gestión de la Contratación, Gestión de Órganos Colegiados.

⇨ E) ÁREA DE ADMINISTRACIÓN ELECTRÓNICA

El sistema de Información que configura la plataforma de Administración electrónica de la Universitat está formado por un grupo de componentes que proporcionan las funcionalidades necesarias para llevarla a cabo. Estos componentes son tres aplicativos denominados: TRAMITEM, ENTREU y TRES.

31.- Entreu (ENtorn de TRamitación Electrónica de la Universitat). Sede electrónica de la Universitat de Valencia. Es el portal a través del que los miembros de la comunidad universitaria y cualquier ciudadano van a poder acceder a la información, servicios y trámites electrónicos de la Administración de la Universidad de Valencia.

32.- Tramitem. Aplicación generalista destinada a la Gestión de Circuitos de Gestión y tareas vinculadas. Es motor de tramitación que permite gestionar el workflow de los procedimientos administrativos de la Universitat.

Las aplicaciones Entreu y Tramitem constituyen la base de la administración electrónica en la Universitat de València y están desarrolladas sobre la plataforma tecnológica TACTICA. Dentro del contrato se incluye:

- ✓ Desarrollo de nuevos trámites en el ámbito de la administración electrónica de la UV.
- ✓ Desarrollo de un entorno con procedimientos de tramitación interna a la Universitat.
- ✓ Adaptación a las hojas de estilos corporativas para la Sede Electrónica incluyendo aspectos de accesibilidad y usabilidad.
- ✓ Adaptación a la gestión documental y cuadros de clasificación de la Universitat.
- ✓ Adaptación de procedimientos ya existentes a las leyes 39/2015 y 40/2015.

33.- TRES (Tratamiento del Registro de Entrada y Salida). Aplicación que da soporte al registro presencial y telemático.

- ✓ Adaptación a la gestión documental y cuadros de clasificación de la Universitat.
- ✓ Integración con los servicios de registro telemático del MINHAP.

En esta área se podrán incluir nuevas aplicaciones del mismo ámbito que la Universitat requiera desarrollar.

⇨ F) ÁREA DE INVESTIGACIÓN

34.- MECENAS (Mecanización CENtralizada de Ayudas y Subvenciones). Aplicación destinada al control y justificación de ayudas y subvenciones del personal investigador. Y a todas las gestiones derivadas.

35.- PACTUM. Gestión de contratos y convenios firmados y promovidos por la Universitat, su control y su justificación.

36.- SABIO. Aplicación que permite definir estructuras de investigación y agrupar los investigadores en ellas.

- ✓ Los desarrollos sobre estas tres aplicaciones estarán subordinados a las necesidades y requisitos establecidos por el Servicio de Investigación de la Universitat de València. No obstante, se prevé el mantenimiento evolutivo y correctivo de todas ellas.

37.- FIU. Aplicación para la gestión de los fondos de investigación universitaria.

38.- *IPC.* Gestión del índice de producción científica.

39.- *GREC.* Aplicación para la gestión de la memoria de investigación.

En esta área se podrán incluir nuevas aplicaciones del mismo ámbito que la Universitat requiera desarrollar.

3.-Plataforma Tecnológica

LOTE 1

Tras la migración acontecida en 2017, la plataforma tecnológica es la siguiente:

Servidor de aplicaciones : Websphere sobre servidores Linux .

Gestor de base de datos : Oracle RAC

Servidores web: servicios http y https sobre servidores apache con Linux.

Aplicaciones: gran parte de la lógica de negocio está desarrollada en el lenguaje de alto nivel EGL de IBM. Para desplegar las aplicaciones, se generan estos programas en lenguaje java y se despliegan en un repositorio compartido por las distintas JVMs. Un cargador dinámico de clases se encarga de revisar que clases han cambiado y cargarlas en las JVMs de los servidores de aplicaciones WebSphere. Control de versiones con CVS.

El objetivo de la UV a medio plazo es adaptar todas las aplicaciones de este lote de acuerdo con la arquitectura basada en el Anexo Framework_CRUE-TIC.

LOTE 2:

Servidor de aplicaciones : Websphere sobre servidores Linux .

Gestor de base de datos : Oracle RAC

Servidores web: servicios http y https sobre servidores apache con Linux.

Aplicaciones: Lenguaje de desarrollo Java según los estándares de la arquitectura JEE en 3 capas. Control de versiones con CVS.

El objetivo de la UV a medio plazo es adaptar todas las aplicaciones de este lote de acuerdo con la arquitectura basada en el Anexo Framework_CRUE-TIC.

4.- Descripción del modelo de prestación servicio

La Universitat de València, siguiendo las prácticas comúnmente extendidas en los procesos de externalización de la producción de software, estructura la contratación del servicio en tres fases, con el fin de asegurar la calidad de los servicios prestados y asegurar la libre concurrencia de empresas prestadoras de servicios:

✓ **Fase de transición**: se trata de un periodo inicial que permite la adquisición de información y documentación de los desarrollos y la implantación de la metodología de trabajo. La duración de este periodo será máxima de 2 meses.

✓ **Fase de prestación del servicio**: se trata del periodo normal de prestación del servicio.

✓ **Fase de devolución del servicio**: se trata de la fase final de la prestación del servicio que permite la recuperación del servicio por parte del personal de la UV, o bien, por parte de un tercer prestador del servicio. La duración de este periodo será máxima de 2 meses.

De cara a describir el modelo de prestación del servicio, se enumeran a continuación los actores que intervienen en el proceso de producción de software, que en algunos casos se corresponden con personas y en otros con equipos de trabajo:

- **Dirección SIUV**: que asume el rol de dirección del contrato.

- **Responsable de proyecto del SIUV**: persona que tiene asignada la responsabilidad de jefe de proyecto por parte del SIUV.

- **Equipo de soporte del SIUV**: conjunto de personas que ofrecen servicios de soporte a las aplicaciones, por ejemplo, tareas de despliegue, administración de sistemas, explotación, etc.

- **Jefatura de proyecto de la empresa contratista:** persona que tiene asignada la responsabilidad de jefatura de proyecto por parte de la empresa contratista.

- **Equipo de desarrollo de la empresa contratista**: conjunto de personas que la empresa contratista pone al servicio del contrato.

- **Usuario/a funcional**: persona o conjunto de personas con la potestad de definir los requerimientos de evolución de una aplicación (pueden ser grupos diferentes de personas para las aplicaciones dentro del alcance de este contrato).

- **Usuario/a final**: conjunto de personas usuarias de la aplicación, y que por tanto pueden reportar incidencias o sugerencias de funcionamiento.

- **Figuras de promotor e interlocutor**: es el promotor, dentro de la UV, el centro, servicio, vicerrectorado o entidad universitaria similar que lidera el proyecto. Para facilitar la comunicación con el promotor del proyecto, se designará por parte del promotor la figura del interlocutor, que centralizará por parte del promotor no sólo los requisitos de usuario, sino permitirá una gestión más ágil del proyecto en términos de cambios solicitados por el usuario y validación de los mismos.

4.1.- Fase de transición del servicio

Se define un periodo inicial de toma de control de los trabajos a realizar por parte de la empresa contratista, en el que de manera gradual se iniciará la prestación del servicio. Durante esta fase no se aplicarán penalidades derivadas del incumplimiento del Acuerdo de Nivel de Servicio.

Durante este periodo la Universitat o, en su caso, el proveedor saliente realizará la devolución del servicio finalizado al nuevo contratista.

La Universitat pondrá a disposición del contratista el código fuente y la documentación disponible de todas las aplicaciones. El contratista dedicará los recursos necesarios para completar la documentación de las aplicaciones que la UV determine como prioritarias.

En esta fase se deberán acordar las fuentes de datos y los criterios para construir el cuadro de mandos de indicadores de seguimiento del proyecto.

4.2.- Fase de prestación del servicio

En esta fase el adjudicatario tiene la responsabilidad total de la prestación del servicio y se exigirá el cumplimiento del Acuerdo de Nivel de Servicio de este contrato. La empresa contratista se compromete a llevar a cabo los trabajos previstas en el plan de Calidad incluido en su oferta.

Tanto la planificación de los trabajos a realizar como la priorización temporal de los mismos estarán condicionadas por las necesidades de gestión la UV y de los cambios normativos, internos o de rango superior, que se puedan producir durante el período de ejecución del contrato y que afecten a las aplicaciones objeto del mismo. Así mismo, la planificación de los trabajos a realizar dentro del contrato estará restringida al número de horas contratadas.

Dada la naturaleza de los trabajos a desarrollar, se puede diferenciar por una parte un conjunto de trabajos continuados que se prolongan a lo largo de toda la prestación del servicio como son los trabajos de coordinación, gestión del servicio, implementación de evolutivos de corta duración (fijando un tiempo menor a 40 horas de desarrollo como referencia), resolución de incidencias que puedan surgir de los evolutivos y desarrollos, tareas de soporte, formación, documentación de aplicaciones, etc., y por otra un conjunto de trabajos no continuados como son mantenimientos evolutivos de mayor entidad, recodificaciones u optimizaciones de código, tareas de migración y nuevos desarrollos.

De este modo, la UV define un modelo de prestación de servicio compuesto por una parte dedicada a los trabajos continuados, y por otra parte dedicada a los trabajos bajo petición (no continuados) y que se detallan a continuación:

Trabajos continuados

Los trabajos continuados será el conjunto de actuaciones, habitualmente no programables, resultado de la operación diaria de las aplicaciones, y que tendrán básicamente los siguientes orígenes:

-Peticiones de trabajo demandadas por la persona responsable de proyecto del SIUV.

-Incidencias de funcionamiento reportadas por los usuarios funcionales o usuarios finales de las aplicaciones, a través de los diferentes canales de soporte habilitados por la Universitat.

-Incidencias de disponibilidad, rendimiento o debidas a actuaciones planificadas en los sistemas que albergan las aplicaciones.

-Pequeñas tareas de mejora (usabilidad, rendimiento, ajuste, etc.)

Debido a la amplia variabilidad de las acciones a acometer se definen a continuación las premisas fundamentales que deben regir la prestación del servicio:

-La persona responsable de proyecto del SIUV debe ser informada de todas las actuaciones en curso o planificadas, pudiendo definir prioridades, urgencias y descartes y ajustar el número de horas imputadas. También deberá establecer el nivel de severidad: grave o normal.

-Cualquier propuesta de evolución por parte de usuarios funcionales, usuarios finales o de la propia empresa contratista debe contar con la aprobación de la persona responsable de proyecto del SIUV para la asignación de prioridades e implementación.

-Con el fin de agilizar los tiempos de respuesta, la empresa contratista deberá actuar de forma autónoma para resolver incidencias de funcionamiento de las aplicaciones siempre y cuando estas actuaciones no impliquen riesgos colaterales de funcionamiento de otras aplicaciones, desviación de recursos que pongan en riesgo otras tareas del equipo de trabajo u otros riesgos que requieran de la validación de la persona responsable de proyecto del SIUV, que deberá estar informada en todo momento de estas actuaciones.

-Será la empresa contratista la que inicialmente valore el nivel de gravedad de las incidencias, pero la jefatura de proyecto tendrá la potestad de modificar el nivel de gravedad.

Todas las tareas acometidas por la empresa contratista deberán ser convenientemente registradas, con los siguientes datos, entre otros: número de actuación, fecha y hora de la solicitud, fecha y hora de resolución de cara al usuario final, gravedad, origen de la petición, esfuerzo invertido, descripción de la actuación, proceso de despliegue (cómo, cuándo se desplegará el cambio en el entorno de preproducción/producción).

Trabajos bajo petición

Los trabajos bajo petición serán planificados periódicamente por la persona responsable de proyecto del SIUV, para lo cual contará con el soporte de la empresa contratista principalmente para la valoración de los esfuerzos requeridos para las actuaciones (ver Estimación de Costes de Tareas en horas en Anexo de Estimación de Costes). Durante las reuniones de seguimiento de cada periodo se definirán las actuaciones previstas para el siguiente periodo.

Las premisas fundamentales que deben regir la prestación del servicio serán las siguientes:

-La persona responsable de proyecto del SIUV planificará el calendario de tareas a desarrollar.

-La prestación de los trabajos bajo petición requerirá una valoración previa por parte de la empresa en términos perfiles/horas, para la cual se deberán utilizar mecanismos de estimación que independicen al máximo la relación existente entre las estimaciones de esfuerzo y el equipo de personas implicado en realizar dicha estimación.

-Las valoraciones que se deriven para cada uno de los trabajos bajo petición, deberán ser aprobadas por la persona responsable de proyecto del SIUV.

-Cualquier actuación aprobada para su ejecución deberá contar con una completa documentación funcional, técnica y de pruebas para poder ser considerada como entregada.

-Todas las actuaciones identificadas (ya sean finalmente aprobadas o no) deberán ser registradas para poder tener un inventario completo de actuaciones en cola, descartas, en proceso de ejecución o ya terminadas.

4.3.- Fase de devolución del servicio

La fase final de devolución del servicio es la que permite la recuperación del servicio por parte del personal de la Universitat de València, o bien, por parte de un tercer prestador del servicio.

Una vez finalizado el contrato, se define un periodo en el que el proveedor saliente pondrá disposición de la Universitat el código fuente, la documentación y el conocimiento disponible sobre todas las aplicaciones desarrolladas o modificadas durante el periodo de prestación del servicio. El contratista saliente dedicará los recursos necesarios para completar la documentación de las aplicaciones que la UV determine como prioritarias.

El objeto de esta fase es permitir el mantenimiento continuado de las aplicaciones en los periodos de transición entre empresas contratistas diferentes y se considera una fase crítica en la prestación del servicio.

5.- Condiciones de la prestación del servicio

5.1.- Acuerdo de Nivel de Servicio

Con objeto de garantizar la efectividad del servicio prestado de manera objetiva y cuantitativa, la Universitat de València establece un Acuerdo de Nivel de Servicio que permitirá evaluar la calidad del servicio de forma periódica. Los Indicadores de Nivel de Servicio son los siguientes (ver Estimación de costes de tareas en horas en Anexo de Estimación de Costes para la clasificación de las tareas).

Tiempos:

- ✓ **I1:** % de incidencias en producción con prioridad Crítica resueltas en menos de 4 horas laborables respecto a todas las incidencias de este tipo **≥ 95%.**
- ✓ **I2:** % de incidencias en producción con prioridad Urgente resueltas en menos de 8 horas laborables respecto a todas las incidencias de este tipo **≥ 95%.**
- ✓ **I3**: % de incidencias en producción con prioridad Normal resueltas en menos de 24 horas laborables respecto a todas las incidencias de este tipo **≥ 90%.**

Errores:

- ✓ **I4:** % de incidencias derivadas de errores imputables al adjudicatario respecto del total de incidencias del período**≤ 5%.**
- ✓ **I5:** % de incidencias de errores generadas por corrección de otro error o recurrencia del mismo del total de incidencias del período **≤ 2%.**

Entregables:

- ✓ **I6: %** de desviación media por exceso de tiempo (fecha de finalización) en el último entregable de todos los proyectos con respecto a la planificación inicial **≤ 10%.**
- ✓ **I7: %** de desviación media por exceso de coste (horas) en el último entregable de todos los proyectos con respecto a la planificación inicial **≤ 10%.**

5.2.- Herramienta de gestión del servicio

Las herramientas de soporte a la gestión y a la operativa del servicio son esenciales en la prestación del presente servicio. Por este motivo, la UV dispone de una herramienta conjunta de gestión del servicio:

- JIRA para la gestión de los proyectos de desarrollo y en el que se define el flujo de los

distintos estados en que puede estar una tarea.

- Confluence para la gestión del conocimiento.

Adicionalmente, los usuarios de las aplicaciones utilizan la herramienta de ticketing Request Tracker para reportar incidencias de éstas en el entorno de producción.

A partir de los datos recogidos en las anteriores herramientas, se elaborará los Indicadores de Nivel de Servicio a valorar en cada reunión de seguimiento del proyecto.

5.3.- Distribución del trabajo por perfiles

Para la ejecución de los trabajos descritos en el presente pliego se ha estimado la carga de trabajo especificada en la siguiente tabla en función del perfil profesional:

Lote 1

Perfil	Horas mínimas (anual)
Analista	3.750
Programador/a	11.250

Lote 2

Perfil	Horas mínimas (anual)
Analista	3.750
Programador/a	11.250

El rol de Jefatura de Proyecto deberá ser asumido por una de las personas con perfil Analista en el equipo de trabajo. La dedicación al contrato de la jefatura de Proyecto y de los Analistas será coherente y equilibrada con las horas establecidas para el perfil Programador/a en la propuesta ofertada. Los recursos se mantendrán vinculados al contrato hasta la finalización del mismo o hasta que las tareas imputadas consuman la totalidad de horas objeto del mismo. Para que así sea, la empresa contratista se compromete a distribuir adecuadamente la carga de trabajo entre los miembros (perfiles) del equipo durante el periodo de ejecución del contrato.

Si durante la ejecución del contrato la empresa contratista propusiera el cambio de alguna de las personas del equipo de trabajo, el cambio deberá ser obligatoriamente aprobado a la UV para proceder a dar de baja las autorizaciones de acceso a los recursos UV a los que estuviese

autorizado para el desempeño de sus tareas. Además, el cambio no deberá repercutir en los Acuerdos de Nivel de Servicio y, por tanto, se deberá asegurar el mantenimiento de la calidad del servicio. Por lo tanto, los cambios en el equipo de trabajo no se considerarán como justificación para el incumplimiento de los Acuerdos de Nivel de Servicio.

En cualquier caso, la formación del nuevo personal que se adscriba al proyecto correrá a cargo de la empresa contratista, que deberá anticipar convenientemente los cambios de personal que se pudieran producir.

5.4.- Lugar de trabajo

Los trabajos se realizarán de forma preferente en las instalaciones de la empresa, sin perjuicio de que puedan realizarse también en las de la UV, a decisión de ésta.

La empresa deberá proveer a su equipo de trabajo del material informático y de oficina que se considere necesario para la correcta prestación del servicio. En caso de necesidad por la que la prestación del servicio deba realizarse de manera puntual en las instalaciones de la UV, la empresa proveerá a cada técnico al menos del mobiliario necesario y un ordenador personal dotado con el hardware y software que necesite.

5.5.- Metodología de trabajo

La UV entiende que la prestación de servicios de software debe tener un marco metodológico común a aplicar por las diferentes empresas prestadoras de servicios, de forma que se obtenga una prestación de servicio homogénea en los diferentes ámbitos. Por ello, el SIUV sigue una metodología de desarrollo de aplicaciones alineada con las tendencias actuales en el área de Gestión de proyectos TIC y le facilitará a la empresa contratista toda la documentación necesaria para su correcta aplicación. La empresa contratista, por su parte, se compromete a la utilización de la metodología definida.

Dado que la metodología debe contemplar todos los aspectos del ciclo de vida de desarrollo de software, pero a su vez no debe suponer una pesada carga que ralentice los trabajos, el marco metodológico diferenciará las tareas y entregables a realizar para los trabajos continuados, que serán menores que las tareas y entregables a realizar para los trabajos de desarrollo de mayor envergadura (trabajos bajo petición). Esta metodología de desarrollo de proyectos es abierta y no está relacionada con ningún producto comercial y/o privativo para el desarrollo de aplicaciones.

En este sentido, la empresa contratista deberá asegurar la correcta ejecución y documentación de las siguientes tareas asociadas a cada desarrollo.

Tareas:

- ✓ Toma de requisitos y especificación del sistema.

- ✓ Análisis funcional/orgánico.
- ✓ Diseño técnico.
- ✓ Desarrollo de la solución.
- ✓ Planificación (ETD), ejecución y documentación de las pruebas.
- ✓ Puesta en preproducción.
- ✓ Soporte.

Documentos:

- ✓ Planificación general del proyecto (Plan de Proyecto y Entregables).
- ✓ Toma de requisitos y especificación del sistema.
- ✓ Análisis funcional/orgánico (casos de uso).
- ✓ Diseño técnico (diagrama de secuencia, clases).
- ✓ Manual técnico del software desarrollado.
- ✓ Plan y resultados de las pruebas.
- ✓ Manual de usuario.
- ✓ Manual de explotación.
- ✓ Documentación de despliegue.

Cada una de las tareas realizadas dentro del contrato se considerará finalizada con la elaboración de estos documentos, que constituyen el *Objeto Entregable*.

En el caso de que se trate de tareas continuadas propias al mantenimiento de los aplicativos, el alcance de este Objeto Entregable se limitará a los elementos efectivamente modificados y a la documentación asociada a los mismos.

Los desarrollos software se ajustarán a los criterios técnicos y de arquitectura definidos por la UV. En el caso en que la empresa adjudicataria utilice herramientas o elementos software propios para desarrollar sus trabajos deberá hacerse bajo los siguientes preceptos:

- Conocimiento y aceptación por parte de la UV.

- Cesión de los derechos de uso de estas tecnologías a la UV, sin coste económico adicional, con objeto de asegurar el futuro mantenimiento y evolución de las aplicaciones así desarrolladas, por parte de la propia UV o de terceras empresas contratadas por ella a tal fin.

- Documentación y formación sobre las tecnologías empleadas al personal de la UV, sin coste económico adicional.

5.6.- Control y seguimiento

Para obtener una mejor coordinación y seguimiento del servicio, con carácter previo al inicio de los trabajos, se deberá mantener una reunión de arranque entre la jefatura de Proyecto por parte de la empresa y el personal del SIUV, en la que se presente y detalle el contenido de la oferta que ha resultado contratista. Se revisará la documentación relativa a las normas de desarrollo para los correctivos y evolutivos, a los modelos de estimación y a las herramientas de control y seguimiento de proyecto.

Periódicamente, con carácter bimensual, se realizarán reuniones de seguimiento cuyos objetivos serán:

1.- Valorar y aprobar las tareas realizadas en base a los Objetos Entregables de cada una de ellas. Las tareas certificadas favorablemente podrán ser facturadas en el siguiente periodo.

2.- Planificar el trabajo pendiente de realizar.

3.- Realizar el seguimiento del ANS (Acuerdo de Nivel de Servicio).

Previo a cada reunión de seguimiento, la empresa contratista presentará un informe de las tareas realizadas, incluyendo una relación de las horas empleadas por cada uno de los perfiles del proyecto e informará sobre los valores de los indicadores de Nivel de Servicio y su adecuación al ANS. Se deberá incluir la información tanto para el periodo en cuestión como su acumulado desde el inicio del contrato, así como su planificación hasta la finalización del proyecto.

Durante las reuniones de seguimiento se evaluará la calidad del servicio prestado y se marcarán prioridades en la planificación de los desarrollos. En caso de que no fuera posible el cumplimiento de alguno de los Indicadores de Nivel de Servicio por causas ajenas a la empresa contratista o porque la muestra utilizada para el cálculo de los indicadores no fuera significativa, así se hará constar en los informes de seguimiento requiriendo de la aprobación por parte de la Universitat.

De forma complementaria, se celebrarán reuniones de dirección con la periodicidad que se establezca en el inicio del proyecto, a las que acudirá como mínimo la dirección del SIUV y de la empresa contratista, donde se tratarán aspectos como la definición de estrategias y objetivos de alto nivel, la revisión del balance de los niveles de servicio establecidos, la revisión de potenciales conflictos o problemas que no puedan ser resueltos por el comité de seguimiento, el seguimiento del nivel de satisfacción de los usuarios, etc.

De acuerdo con la metodología de la Universitat, cada uno de los proyectos desarrollados en el marco del contrato, contará con un responsable de proyecto del SIUV que realizará reuniones de seguimiento con la jefartuea de proyecto de la empresa contratista. Dependiendo de la tipología del proyecto, estas reuniones serán regulares (mensuales) o asociadas a los hitos/entregables al proyecto. Adicionalmente, el responsable del proyecto del SIUV realizará secuencias de control para monitorizar el estado en tiempo y forma del proyecto. Dichas acciones se realizarán de forma periódica (15 días) y requerirán de datos puntuales de evolución del proyecto de la empresa contratista.

5.7.- Calidad

Es un objetivo prioritario de la UV asegurar la calidad de los servicios prestados. La UV podrá llevar a cabo actividades de control de calidad del software entregado por parte de la empresa, apoyadas por la utilización de herramientas que permitan obtener métricas para determinar la calidad del desarrollo. Estas actividades podrían incluir verificación estática del código fuente, verificación de código duplicado y código innecesario, verificación de dependencias, verificación de documentación del código, entre otros.

La empresa contratista responderá de la correcta realización de los trabajos contratados y de los defectos que en ellos hubiere. La UV podrá rechazar total o parcialmente los trabajos realizados, en la medida que no respondan a lo especificado en los objetivos de la planificación o aquellos que no superasen los controles de calidad o no estuvieran dentro de los límites definidos por el acuerdo de nivel de servicio pues se observara una desviación en los indicadores propuestos. Las tareas derivadas de errores imputables a la empresa contratista no serán en ningún caso objeto de facturación.

Durante el transcurso del contrato, la UV podrá realizar encuestas de satisfacción dirigida a los usuarios funcionales y finales. Sus resultados se utilizarán para diseñar medidas correctivas con el fin de mejorar el servicio.

6.- Propiedad intelectual y otros condicionantes.

6.1.- Propiedad intelectual

Corresponderá a la UV cualesquiera derechos de explotación derivados de la Ley de Propiedad Intelectual, tanto de los programas, módulos, servicios o componentes software así como de la diferente documentación asociada a los mismos, obligándose la empresa contratista a respetar en todo momento lo dispuesto en esta cláusula.

Estos derechos corresponden a la UV de forma indefinida, en exclusiva, para un ámbito territorial mundial y respecto de cualesquiera modalidades de explotación existentes.

La empresa contratista deberá garantizar que los servicios prestados a la UV, en virtud del documento contractual, no infringen ni vulneran los derechos de propiedad intelectual y/o industrial, o cualesquiera otros derechos legales o contractuales de terceros.

6.2.- Tratamiento de datos de carácter personal

El adjudicatario tendrá la condición de encargado del tratamiento al efecto de lo dispuesto en la legislación sobre protección de datos personales. En virtud de ello:

1.- Deberá cumplir con lo dispuesto por la Disposición adicional vigésima sexta sobre "Protección de datos de carácter personal" del Real Decreto Legislativo 3/2011, de 14 de noviembre, por el que se aprueba el texto refundido de la Ley de Contratos del Sector Público, por el artículo 12 de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal y por los artículos 20 y siguientes del Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal.

2.-Deberá facilitar la tramitación y firma del contrato para el acceso a los datos por cuenta de terceros al que se refieren las normas citadas.

En particular:

a) Deberá acreditar su capacidad para cumplir con sus obligaciones en esta materia de modo que se pueda realizar una elección diligente del encargado en los términos del artículo 20 del Real Decreto 1720/2007.

Tales capacidades podrán demostrarse entre otras formas mediante:

- Acreditación de la inscripción de sus propios ficheros ante el Registro General de Protección de Datos Personales de la Agencia Española de Protección de Datos.
- Exhibición o certificación de informes de auditoría que acrediten el cumplimiento normativo y/o de seguridad.
- Acreditación de que su personal ha sido debidamente formado.
- Acreditación de la adhesión a estándares comúnmente admitidos en materia de seguridad o privacidad, y cuando ello fuere posible, exhibición de su certificación de cumplimiento.
- Declaración de sus políticas de seguridad en aquello que pudiera afectar al objeto del contrato objeto de licitación.

b) Deberá facilitar cuando se le requiera la información necesaria para la redacción definitiva del citado contrato.

c) Deberá firmar el citado contrato necesariamente antes del desarrollo de actividades que comporten acceso a datos.

6.3.- Otras obligaciones

La empresa contratista está obligada al cumplimiento de las normas sobre uso de recursos TIC de la Universitat de València que le sean de aplicación (http://www.uv.es/uvweb/servei-informatica/ca/normativa-procediments/reglament-us-recursos-tic-uv-1285874221018.html), así los siguientes reglamentos referentes a la seguridad de la información:

-Política de seguridad de la información:

http://www.uv.es/sgeneral/Reglamentacio/Doc/Adm_Electronica/Y10.pdf

-Reglamento de seguridad de la información:

http://www.uv.es/sgeneral/Reglamentacio/Doc/Adm_Electronica/Y11.pdf

Para ello deberán firmar un compromiso expreso de su cumplimiento, así como y acuerdo de confidencialidad. También deberá firmar un compromiso de cumplimiento de todas aquellas normas y protocolos de actuación que se deriven de la adaptación al ESQUEMA NACIONAL de SEGURIDAD por parte de la Universitat de València.

Valencia, 22 de noviembre de 2017

Fdo.: Fuensanta Doménech Roda
Directora del Servei d'Informàtica

TIC CRUE

CONFERENCIA DE RECTORES DE LAS UNIVERSIDADES ESPAÑOLAS
COMISIÓN SECTORIAL TIC

Anexo Framework_CRUE-TIC

ESTÁNDAR D.O. CRUE-TIC: ARQUITECTURA DE DESARROLLO PARA APLICACIONES WEB DE LA CRUE

Comisión Sectorial TIC de la CRUE

Madrid, 6 de Noviembre de 2015

1. ÍNDICE

2. ALCANCE

El alcance de este documento se centra en describir una arquitectura recomendada para el desarrollo de aplicaciones web de la CRUE, teniendo en cuenta las tendencias más novedosas en el ámbito del Cloud Computing y la movilidad, así como la reutilización de dicha arquitectura para futuros desarrollos.

Para ello se han seguido los siguientes objetivos:

- Sentar una base sólida y actualizada de las tecnologías de desarrollo para la definición de la arquitectura bajo la denominación D.O. CRUE-TIC. Partiendo sobre el estado del arte, escoger aquellas tecnologías y metodologías más idóneas.
- Definir las tecnologías de las diferentes capas horizontales y desarrollar los componentes necesarios para ensamblar dichas capas de manera que los desarrollos de los distintos módulos de las aplicaciones sean rápidos y robustos, y que incluyan las características propias del uso de tecnologías Cloud y relacionadas con la movilidad.

El documento está dividido en dos partes. En la primera se describe una propuesta de pila tecnológica de arquitectura de aplicaciones, donde se enumeran, para cada capa de aplicación, los frameworks y tecnologías que consideramos más apropiadas (salvo algún caso que se justifica convenientemente). En general, todas las librerías y frameworks tienen algún tipo de licencia *"open source"*, salvo en algunos casos en los que consideramos que no existe una alternativa *"open source"* equiparable. Este es el caso de las herramientas de pruebas Siesta y SOAPUI (la versión profesional); dado que son herramientas de pruebas, no influyen en que el código fuente siga siendo "*open source*" y los costes de licenciamiento de las versiones "*profesionales*" (tienen versiones gratuitas) son asequibles.

En la segunda parte se describen los tipos y las capacidades de las arquitecturas cloud y los requisitos que deben cumplir las aplicaciones a construir para poder aprovecharse de sus características.

3. ARQUITECTURA DE APLICACIONES ORIENTADA A SERVICIOS

Con el objeto de desarrollar nuevas aplicaciones Web , se propone una arquitectura de referencia adaptada a la elaboración de software para las necesidades del día a día de las universidades. Está basada en los últimos estándares de JAVA, suficientemente consolidados y probados con éxito en experiencias de desarrollo recientes o en curso.

Está basada en un paradigma de Orientación a Servicios, en el cual la posible complejidad de los componentes Software queda encapsulada como servicios que ofrecen una "*interface*" simple de invocación y persistente en el tiempo. Esto permite la modificación o cambio de ciertos servicios sin afectar al resto del sistema. Este enfoque aporta las siguientes características técnicas:

- Permite segmentar y controlar el desarrollo en diferentes fases manteniendo operativo todo el sistema mientras se van incorporando o substituyendo los diferentes componentes que lo conforman.
- Permite una rápida integración de componentes proporcionando disponibilidad y tolerancia a errores.
- Proporciona alto rendimiento con gran capacidad de procesamiento y baja latencia, incorporando tecnologías de procesamiento paralelo.
- Permite el balanceo constante de carga mediante servidores en clúster que redundan en su fiabilidad y garantizan el servicio.
- Permite escalar aplicaciones por lo que se podrán integrar diferentes entornos de trabajo.

Este enfoque de la solución permite el desarrollo de los sistemas en base a reutilización de componentes y facilitando su posterior reconfiguración y evolución.

3.1 Arquitectura empresarial

La arquitectura empresarial (*enterprise*) actúa como marco en el que las respectivas arquitecturas técnicas de los diferentes sistemas se apoyan para cumplir con los requisitos técnicos y de negocio.

Las distintas capas de esta arquitectura se describen a continuación:

- **Procesos**

  En esta capa se definen los procesos de negocio de la Arquitectura Empresarial. Un proceso de negocio puede estar compuesto por la orquestación de diferentes servicios de negocio o flujos de navegación localizados en diferentes Sistemas.

- **Integración**

  En esta capa se realiza la integración de los diferentes procesos, sistemas, soluciones y recursos que componen la Arquitectura Empresarial.

- **Sistemas**

  En esta capa se definen los diferentes sistemas que componen una arquitectura empresarial. Un sistema es un conjunto de componentes que cumple de forma autónoma y desacoplada una

funcionalidad de negocio. La arquitectura de sistemas debe garantizar que el sistema cumple todos los requisitos necesarios para cumplir los requisitos globales de la arquitectura empresarial.

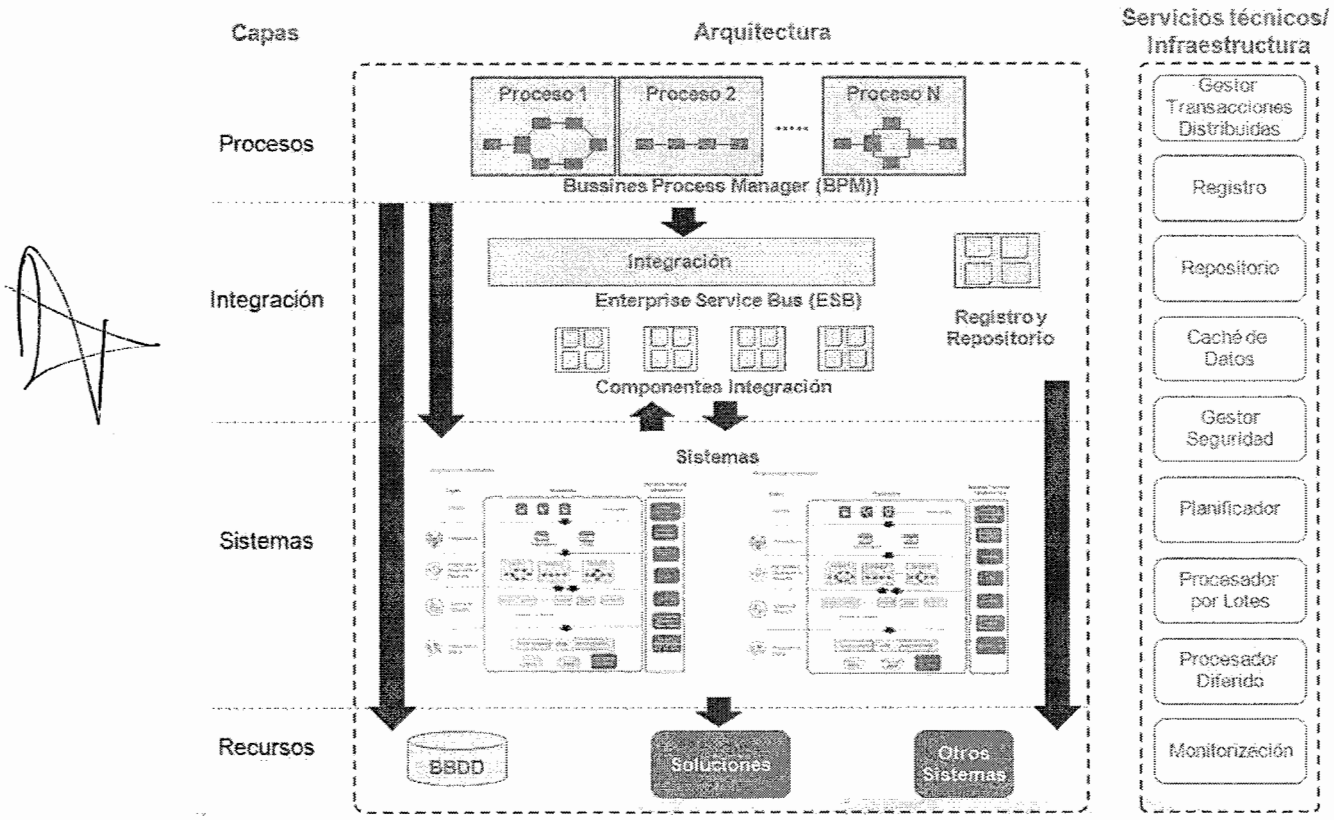
- **Recursos**

En esta capa se encuentran todos los recursos y soluciones encargados que almacenar los diferentes elementos del Sistema de Información gestionado por la Arquitectura Empresarial.

- **Servicios Empresariales**

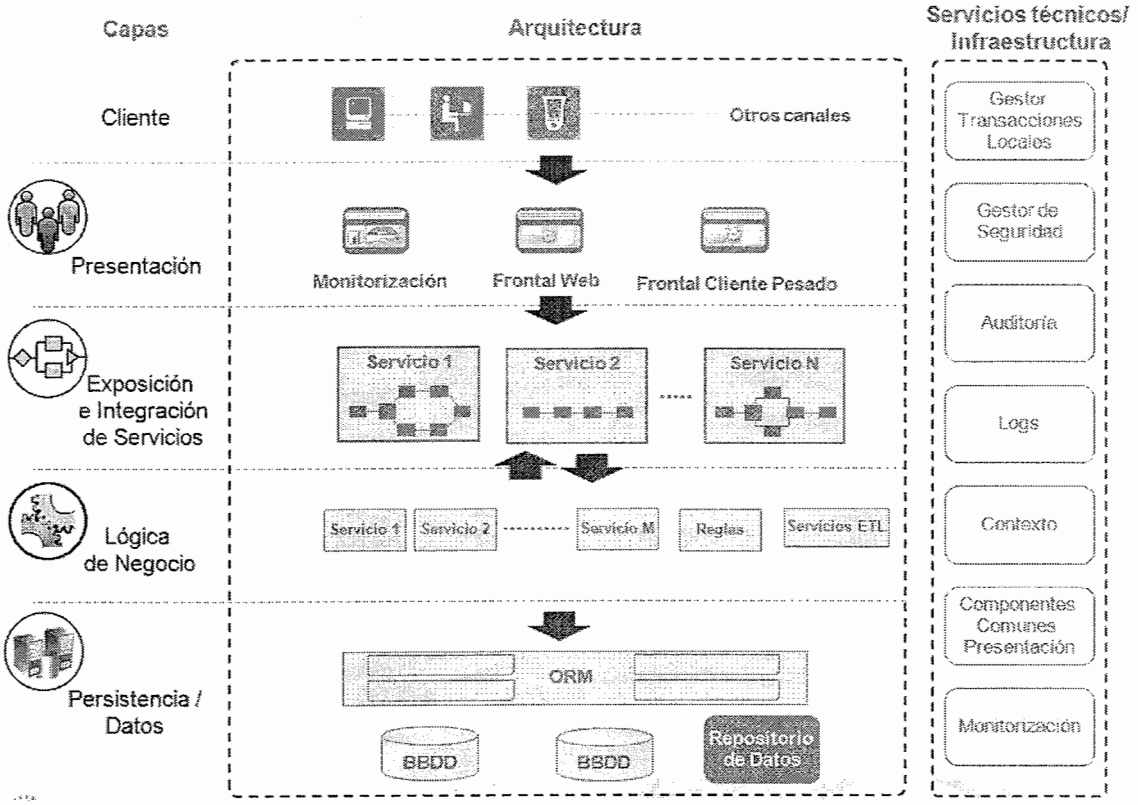
En esta capa se encapsulan todos los servicios empresariales encargados de cumplir los requisitos globales de la Arquitectura Empresarial.

A continuación se muestra de manera gráfica como quedan organizadas las diferentes capas que conforman dicha arquitectura:

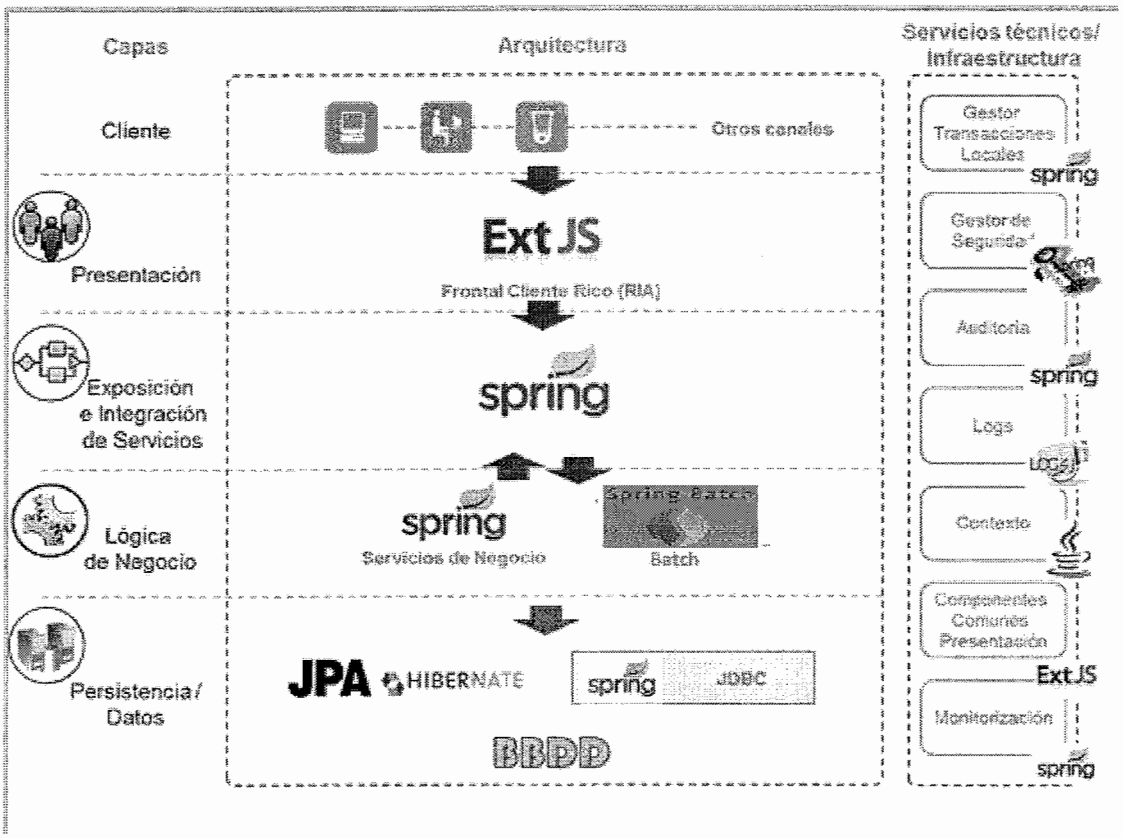


3.1.1 Arquitectura técnica básica

Si nos enfocamos en el detalle de cada sistema, podemos identificar una arquitectura técnica, que estaría conformada por los siguientes elementos, que cubrirían todas las capas de los desarrollos a realizar.



Este diagrama se materializa en los siguientes elementos software para cada una de las diferentes capas:



3.1.1.1 Capa de presentación Web. EXTJS

En las aplicaciones web actuales, uno de los principales factores de éxito es que el producto desarrollado proporcione una **buena experiencia de usuario**, con una interfaz atractiva, interactivamente rica y que suponga una mejora en la productividad y eficiencia con las que el usuario completa las tareas para las que se ha desarrollado la aplicación.

FrameWork ExtJS

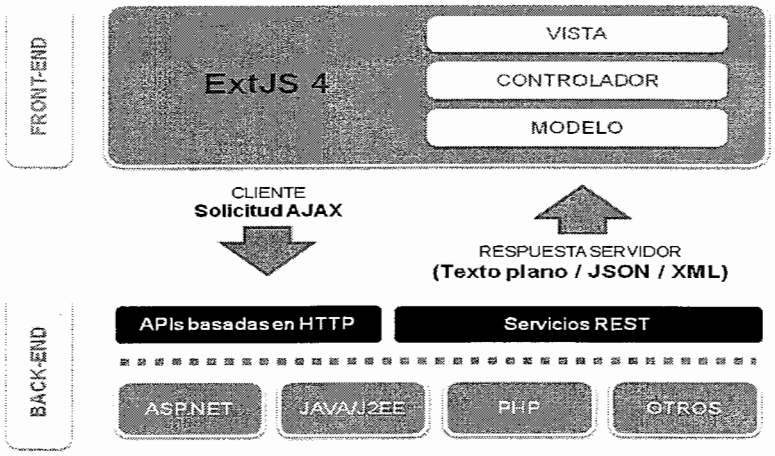
Actualmente, en cuanto al contexto de uso de aplicaciones, podríamos distinguir dos escenarios que van ganando peso:

- Aplicaciones web RIA (Rich Internet Aplications)
- Aplicaciones para dispositivos móviles.

Aunque poco a poco irá existiendo mayor convergencia entre ambas, aún siguen existiendo diferencias en cuanto a las capacidades hardware de los dispositivos móviles y los PC's que aconsejan aplicar un diseño gráfico y de interacción específicos y apropiados para cada ámbito. En ese sentido, para la presente oferta, se ha optado por el uso del framework javascript ExtJS v.5.0.

El uso de ExtJS nos permite un desacoplamiento de la tecnología importante como se puede ver en el siguiente diagrama:

Independencia del back-end



Beneficios:

Desacoplamiento tecnológico

El framework ExtJS se adapta al enfoque de separar y desacoplar perfectamente la capa de presentación del resto de capas de la arquitectura. En este sentido, esta separación se realiza de forma física. La UI (User Interface) se encarga de manejar la interacción del usuario completamente desde el lado cliente y consume, envía los datos e interacciona con el lado servidor, mediante llamadas AJAX a servicios REST.

De esta forma los servicios de negocio pueden ser reutilizables desde el lado cliente con la ventaja de:

- Poder crear distintas vistas que representen y/o manejen los datos de diferentes formas
- Aprovechar al máximo las capacidades visuales e interactivas de los distintos dispositivos existentes
- Poder implementar la UI con distintos lenguajes o tecnologías sin tener que modificar los servicios de negocio.

Paralelización de Roles Técnicos

ExtJS facilita la separación de los ficheros y recursos de la aplicación con los que trabajan los maquetadores y programadores FrontEnd, de los ficheros y recursos que se manejan en el ámbito de los programadores del BackEnd. De esta manera, se pueden desarrollar en paralelo la interfaz gráfica y los servicios de negocio, obligando de esta manera a establecer los contratos entre ambas capas en fases iniciales del desarrollo.

En otras tecnologías, tipo Struts, JSP's, JSF..., la parte de código correspondiente a la presentación, tanto de maquetación como de lógica de presentación (etiquetas HTML y javascript), acaba siendo transformada en otras etiquetas y entremezclada con el código correspondiente a la parte del servidor o de negocio.

En ese contexto, comúnmente, los maquetadores construyen las pantallas en html/CSS y posteriormente los programadores Java los convierten, p.ej. a JSP, lo que acaba en ocasiones produciendo problemas o inconvenientes como:

1. Trabajo extra para los programadores java que frecuentemente tienen que convertir las etiquetas de los ficheros html recibidos desde maquetación, a los tags específicos de las librerías de componentes que estén utilizando en el back-end.
2. Dificultad en el mantenimiento, corrección y pulido de la maquetación de las pantallas, una vez hecha la transformación explicada anteriormente, si la persona encargada del mismo, no conoce perfectamente tanto HTML y CSS como java.

Documentación

Otro de los puntos fuertes de ExtJS, es la calidad de la documentación del API de desarrollo, con información muy detallada de los distintos componentes y con numerosos ejemplos con código fuente disponible.

Curva de aprendizaje

Debido a que el framework abarca todas las partes del desarrollo del FrontEnd, su aprendizaje puede requerir inicialmente un esfuerzo superior al necesario en otras librerías o toolkits más básicos, pero este esfuerzo es rápidamente compensado por el resultado que se obtiene del mismo.

Soporte

Aunque dispone de licencia opensource (GPLv3), ofrece también modalidades de pago con soporte incluido, que pueden incluir la solución de bugs en un intervalo corto de horas. Aparte, existe una comunidad amplia de desarrolladores, por lo que es fácil encontrar soluciones a posibles problemas o respuestas a las dudas más frecuentes.

Relación Esfuerzo/Resultado

Las aplicaciones desarrolladas con ExtJS muestran un salto cualitativo en cuanto a interactividad, flexibilidad y acabado respecto a las desarrolladas con otras tecnologías.

Además, es crossbrowser, por lo que reduce en un elevado porcentaje el tiempo de desarrollo que habitualmente requiere en otras tecnologías o librerías, la adaptación a los distintos navegadores o incluso las distintas versiones de un mismo navegador.

Tendencia tecnológica

ExtJS se encuentra alineado con las tendencias actuales de desarrollo FrontEnd por diversos motivos:

- Permite utilizar el patrón Modelo Vista Controlador y una programación orienta a eventos acorde al tipo de aplicaciones que se demanda hoy en día
- Aporta un amplio catálogo de componentes de interfaz con un elevado grado de interactividad y flexibilidad a la hora de programarlos y configurarlos.
- Los componentes están basados en HTML y javascript y por tanto, están soportados de forma nativa por todos los navegadores (Explorer, Chrome, Firefox, Safari, etc.) sin la necesidad de instalar plugins adicionales.
- Los componentes son compatibles con HTML5 y CSS3, lo que garantiza la evolución futura de las aplicaciones y el aprovechamiento de las nuevas capacidades que los navegadores modernos irán introduciendo según vayan evolucionando los propios estándares.

Este punto es especialmente importante atendiendo a que la tendencia hacia la que se dirigen los distintos tipos de dispositivos móviles, tablets, etc. es a la adopción del HTML(5) y javascript como tecnologías con los que construir las interfaces de usuario de sus aplicaciones, incluso facilitando API's que permiten el acceso a los componentes hardware de los dispositivos, aproximándose así al rendimiento y funcionalidades conseguidos en desarrollos nativos.

Dependencia de Terceros

La amplitud del catálogo de componentes que ofrece de base el framework hace que, en la mayor parte de los desarrollos, no sea necesario buscar plugins externos que incorporen algún nuevo componente o que extiendan alguna funcionalidad.

Reutilización

Varios factores influyen en que el framework facilite la reutilización de componentes:

- La librería de componentes que trae de base cubren la mayor parte de las necesidades que las aplicaciones requieren.
- Es sencillo extender los componentes base ampliando las funcionalidades y el comportamiento original.
- Se pueden componer, a partir de componentes elementales, macrocomponentes o vistas completas fácilmente reutilizables entre distintas partes del proyecto o en distintos proyectos.

Catálogo de UI-Widgets y Componentes

ExtJS, aporta un **conjunto base muy amplio de ui-widgets** (componentes con los que construir pantallas del estilo de Árboles desplegables, Grids de datos, Tabs, Accordions, ventanas flotantes etc.) por lo que rara vez se hace necesario aplicar plugins de terceras partes.

La parte dedicada a controles de manejo de **Layout** (disposición de elementos en las ventanas y formularios) facilita la maquetación líquida, lo que permite el control y adaptación de las ventanas a la resolución de cada usuario en los casos en los que así se requiera.

Otro aspecto muy importante a tener en consideración, es la flexibilidad que ofrece para la personalización del look&feel de los componentes. De esta manera, es posible crear y configurar temas visuales que mediante clases CSS estándar adaptan los ui-widgets a la imagen de marca de la empresa o corporación para la que se ha desarrollado la aplicación.

- Permite personalizar de forma fácil el Look&Feel de los componentes mediante creación de temas visuales
- El estilo visual que trae por defecto está muy depurado por lo que incluso personas sin ningún conocimiento de maquetación pueden generar ventanas con un buen acabado
- Los UI-widgets ofrecen muchos atributos y métodos que los hacen muy configurables y adaptables a las necesidades interactivas
- Los UI-widgets se pueden extender fácilmente
- Es un framework completo (no una simples librerías de componentes poco integradas entre sí) por lo que no sólo aporta controles visuales y de manejo de layout sino que también ofrece una metodología de trabajo, facilita el acceso a los datos y a las conexiones al servidor, la gestión de las llamadas AJAX, etc.
- Ofrece un nivel de reutilización muy alto.
- Es cross-browser, eliminando la necesidad de que sea el programador Front el que tenga que meter las habituales excepciones de programación en las ventanas en función de la versión de Ms Explorer, Firefox, etc.

API de desarrollo

El API de desarrollo ofrece, de manera muy **consistente** entre todos los objetos, el acceso a las numerosas propiedades, atributos, métodos y eventos de los componentes, facilitando tanto el desarrollo de las aplicaciones, al permitir adaptarlos a los requisitos de la aplicación, como al aprendizaje en el empleo del framework.

CSS 3

Las hojas de estilo en cascada o (Cascading Style Sheets, siglas CSS). Se utilizan como lenguaje estándar para la capa de presentación, materializando cambios en el aspecto y forma de documentos escritos en HTML y XHTML. Aunque también entre sus aplicaciones se encuentran también documentos XML, SVG y XUL. Destacar también que se encuentra bajo la supervisión W3C.

CSS es la última versión del estándar. Cabe destacar que es completamente compatible con versión anteriores CSS1, CSS2 y CSS2.1.

Vamos a destacar las que se consideran las ventajas más destacadas del estándar CSS:

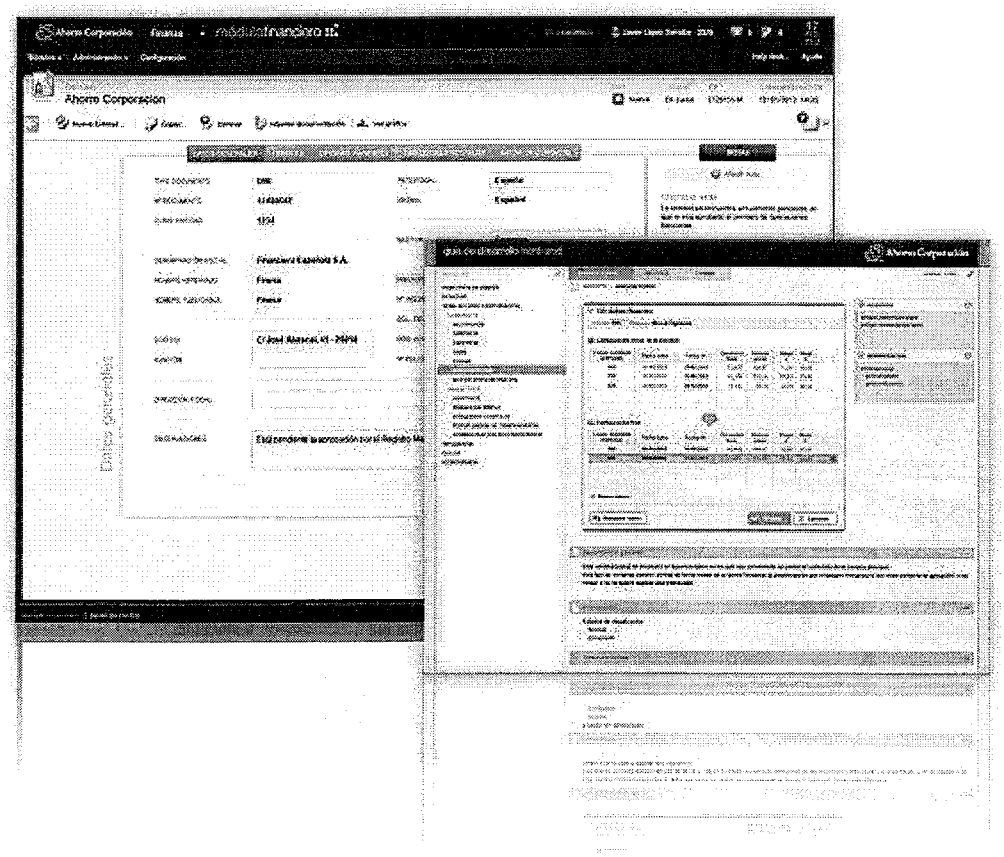
- Al ser un lenguaje estándar es soportado por todos los navegadores de última generación.
- Permite la gestión centralizada de los estilos de presentación. Tanto para aplicaciones web como para clientes pesados, como es el caso de la nueva tecnología para la capa de presentación de Java conocida como JavaFX, que utiliza como base CSS haciendo extensiones sobre la misma.
- Facilita el desarrollo, ya que independiza lo que es el contenido de la presentación. Permitiendo que los diseñadores puedan abordar cambios sin que afecte al negocio.
- Es un lenguaje sencillo y de fácil aprendizaje. Además es estático y no necesita ser compilado. Así que se pueden asumir cambios en caliente que reviertan en la presentación sin precompilar.
- Otras de sus propiedades es la reutilización para distintos documentos a través del uso de selectores.
- La comunidad que hace uso de CSS3 está en auge, pudiéndose encontrar gran cantidad de información sobre su utilización, desarrollo y buenas prácticas. Además de multitud de ejemplos.
- Mediante la característica "slicer" de la SDK de Ext JS, ahora se puede ajustar el posicionamiento multinavegador, a nivel de pixel, utilizando sólo CSS3. Ya no es necesaria la generación de "sprites" personalizados, ajuste de las reglas CSS, etc.

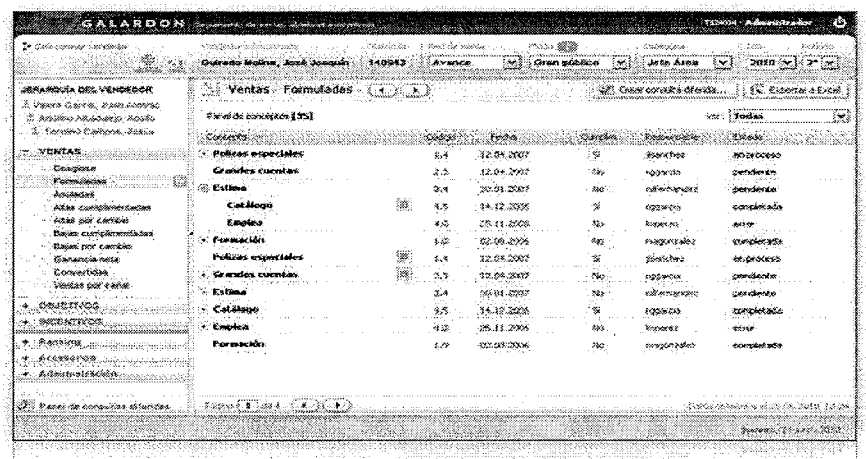
Ahora vamos a particularizar las características o módulos más a avanzados del CSS3 entre los que destacamos:

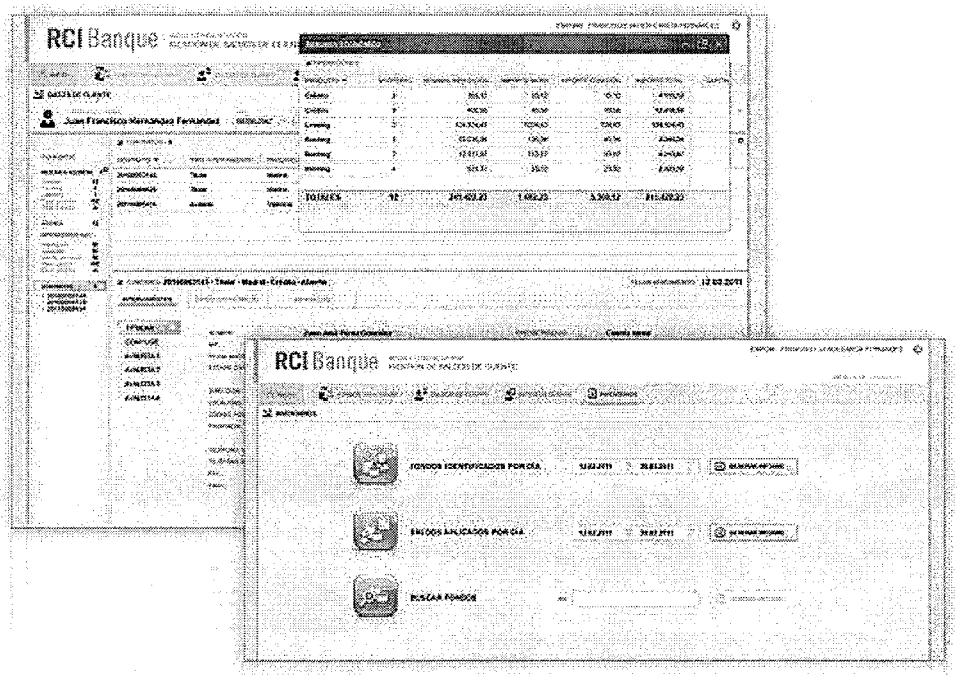
- Selectores, patrones usados para seleccionar elementos sobre los que aplicar un cambio de estilo.
- Modelo de caja, es esencialmente una caja que envuelve alrededor los elementos HTML, y se compone de: márgenes, bordes, relleno y el contenido real.
- Nuevas propiedades para los Fondos y Bordes
- Valores de imagen y contenido de reemplazo.
- Efectos sobre texto como aplicación de sombras o autoajustado de textos a los contenedores.

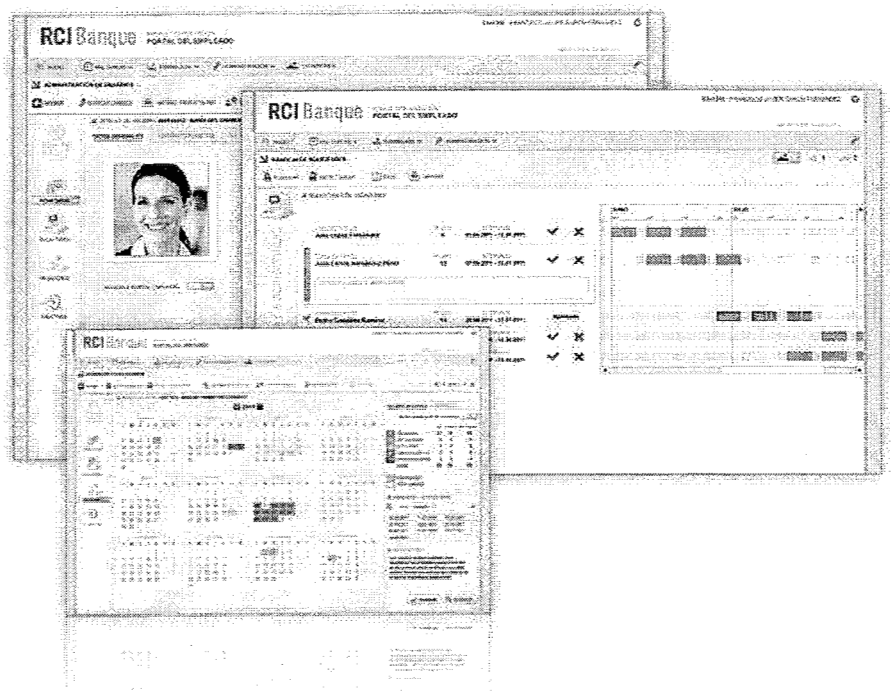
- Transformaciones 2D y 3D. Una transformación es un efecto que permite a un elemento cambiar la figura, el tamaño o la posición. Así podremos escalar, girar mover o hacer los elementos elásticos.
- Transiciones, permitiendo añadir un efecto de cambio de estilo de uno de otro, sin la utilización de animaciones Flash o JavaScript.
- Tiene la capacidad de crear animaciones, para que puedan reemplazar a las imágenes animadas o GIF, animaciones Flash y JavaScripts.
- Diseño de columna múltiple.
- Nuevas Intefaces de usuario, como el redimensionado de elementos, autoajustado de elementos a un área, o el Outline-offset que permita aplicar bordes más allá del propio componente.

Capturas de algunas aplicaciones desarrolladas con la capa de presentación propuesta.

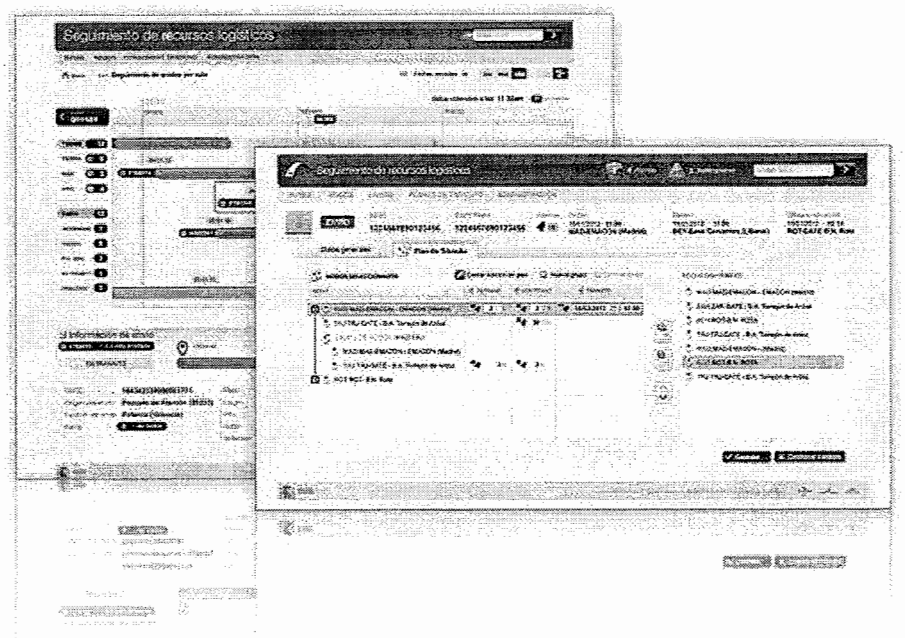


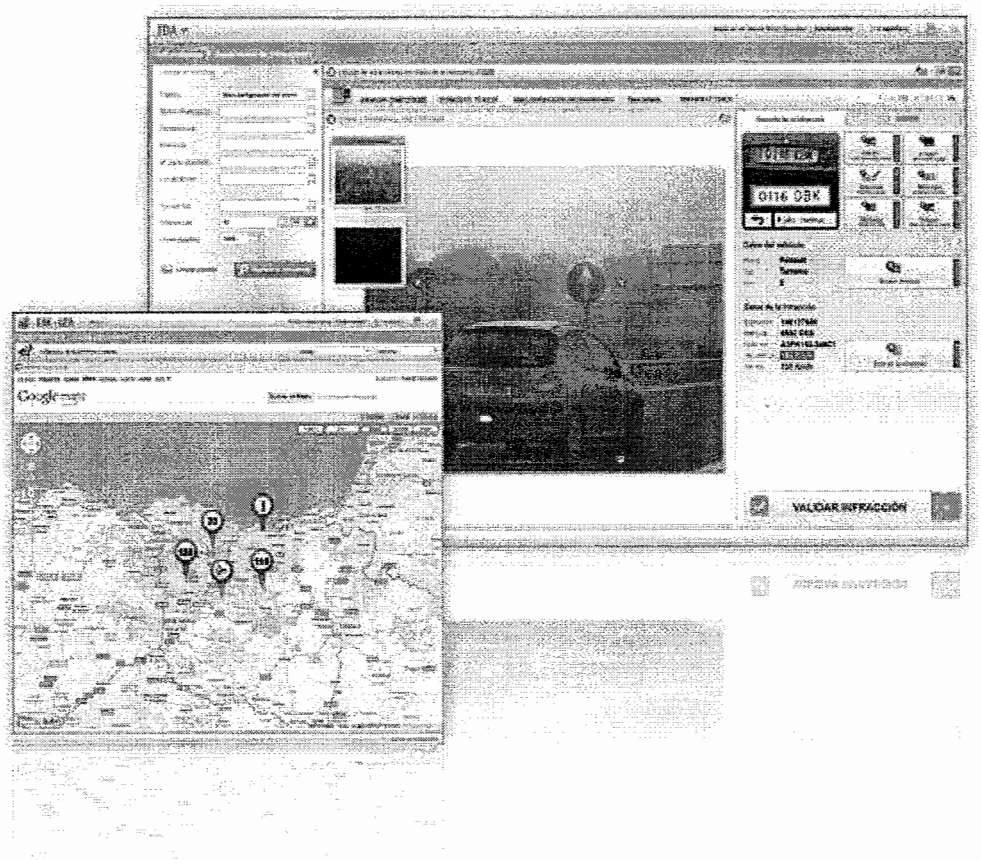





RCI Banque
RCI Banque
RCI Banque


sicap


Seguimiento de recursos logísticos
Seguimiento de recursos logísticos


0116 OBK
VALIDAR INFRACCIÓN

3.1.1.2 Capa de presentación Móvil.

La selección de un marco de trabajo para el desarrollo de aplicaciones para móviles (*smartphones*) y *tablets* varía mucho en función de los requisitos y necesidades. Básicamente existen tres tipos de aplicaciones en función de la tecnología *target* empleada:

1. **Aplicaciones nativas**

   Una app nativa, en principio es una aplicación que se construye en el lenguaje nativo de cada dispositivo. Por eso, si vamos desarrollar una App nativa tendremos que utilizar un lenguaje diferente para cada Sistema Operativo. Los lenguajes de programación serán por tanto los siguientes:

   - iOS: Objective C
   - Android: Java
   - Windows: C# y Visual Basic .NET.
   - BlackBerry 10: C++

   En cualquier caso depende del nivel y experiencia del equipo de desarrollo y de que el código resultante de su trabajo sea el correcto, pero en principio, una App nativa es la opción cuyo resultado es el más robusto y fluido ya que se desarrolla directamente para integrarse en el Sistema Operativo. La experiencia de usuario será la mejor posible (si se diseña la aplicación correctamente, claro) ya que su funcionamiento, rendimiento y respuesta será el más inmediato de todas las opciones de desarrollo incluso en los diseños más complejos y personalizados.

Ventajas	Inconvenientes
• Acesso completo al dispositivo (GPS, cámara, acelerómetro, etc.)	• Mayor coste: profesionales especialistas, repetir la construcción de la app por cada plataforma, actualizaciones de SO implican cambios en la APP, etc.
• La mejor experiencia de usuario posible	
• Notificaciones "*push*"	
• Funcionamiento online-offline, con sincronización automática	

2. **Aplicaciones híbridas**

   Consisten en apps que proporcionan ellas mismas el navegador web del dispositivo (no usan el navegador nativo) y además proporcionan un api de acceso a recursos del dispositivo

(cámara, gps, contactos, Etc.). Para su desarrollo se utilizan frameworks de desarrollo basados en lenguajes de programación web (HTML, CSS y JavaScrpt).

En este tipo de Apps el grado de integración con el SO dependerá del framework de desarrollo utilizado y como de abierto sea el SO, teniendo cada uno de ellos sus ventajas e inconvenientes. Actualmente con esta opción existe bastante acceso al hardware del teléfono e incluso en algunos casos a las librerías del SO, pero lo cierto es que de momento no se ha conseguido igualar la respuesta y la experiencia de usuario de una App nativa.

Su desarrollo es más económico que las aplicaciones nativas, y es una opción clara para llegar al mayor número de usuarios repartidos en las diferentes plataformas y dispositivos aunque por el momento sus limitaciones son claras.

<table>
<tr><th>Ventajas</th><th>Inconvenientes</th></tr>
<tr><td>• Instalación (y desinstalación nativa). Se distribuye a través de los "marketplaces"</td><td rowspan="2">• Experiencia de usuario más propia de la web que de app nativa.</td></tr>
<tr><td>• Mismo código para múltiples plataformas (como las app web)</td></tr>
<tr><td>• Acceso a las características HW (cámara, gps, etc.) y recursos del SO del móvil (contactos, sistema de ficheros, etc.) mediante JavaScript.</td><td>• Poca integración visual en el SO en que se ejecuta.</td></tr>
</table>

3. Aplicaciones web

Son aquellas que están desarrolladas usando lenguajes para el desarrollo web como html, css y javascript y un framework para el desarrollo de aplicaciones web, como por ejemplo jquery mobile, Sencha Touch, Kendo UI, entre otros. Se podría decir que este tipo de aplicaciones son muy usadas para ofrecer accesibilidad a contenidos desde cualquier dispositivo, sin importar el sistema operativo, ya que solo se necesita contar con un navegador para acceder a esta. En resumen; son **webs a las que se accede a través de una URL en el navegador del dispositivo** (Chrome, Safari, etc.) **y se adapta al tamaño de la pantalla para que tenga aspecto de navegación App.** Los navegadores de los móviles permiten crear un acceso directo en nuestro escritorio de esta web de manera que parezca que se ha instalado en el dispositivo.

Ventajas	Inconvenientes
• Multiplataforma	• Sólo funciona online
• Bajo coste	• Acceso **MUY** limitado a los recursos del dispositivo.
• Publicación directa (no app store)	
• Siempre actualizada	• La experiencia de usuario y latencia son peores que en app's nativas.
• Mejora la capacidad de reutilización.	

Por otro lado, existen diferentes frameworks para desarrollar cada una de los tipos de apps mencionadas. En este sentido, existe una categoría más de tipo de aplicación; app nativa desarrollada en un *framework* multiplataforma. En este caso es el *framework* de desarrollo el que nos aisla del HW y SO del dispositivo, realizando un único desarrollo para todas las plataformas y generando varias apps nativas (una por cada plataforma soportada por el *framework*). Naturalmente impone ciertas restricciones en el desarrollo de las aplicaciones.

Por otra parte, algunas empresas ofrecen soluciones "enterprise" (MEAP o Mobile Enterprise Application Platform), que ofrecen servicios más allá del desarrollo:

- Seguridad
- Conectividad de integración
- Gestión y distribución de las aplicaciones
- Frameworks multiplataforma nativa (algunos)

A continuación realizamos una enumeración de algunos de los frameworks disponibles, por cada tipo.

MEAP

1. **Xamarin.** Permite desarrollar aplicaciones nativas para IOS, Android y Windows en C#. Ofrece además servicios añadidos de pruebas, monitorización de sesiones y usuarios, seguridad y pruebas automáticas (Xamarin TestCloud permite realizar pruebas de integración continua sobre cientos de dispositivos).
2. **Kony.** Es una plataforma que soporta el desarrollo de aplicaciones móviles de todo tipo (nativas, híbridas y web).
   - Ofrece servicios de backend mediante API's RESTful.
   - Proporciona servicios de autenticación, notificaciones push y sincronización de datos offline entre otros.

- Gestiona la distribución de apps, tanto a través de *marketplaces* públicos como privados.

3. **Appcelerator Titanium.** Permite manejar el ciclo de vida completo de una app, ofreciendo herramientas y servicios para depuración de errores, pruebas automáticas, despliegue, monitorización de errores y obtención de datos analíticos. A través de su herramienta de desarrollo, permite generar aplicaciones nativas construidas con su propio framework "*Alloy*" (javascript) pero a su vez soporta desarrollos nativos en java (para Android) y Objective-C (para IOS). Ofrece un entorno de desarrollo (Appcelerator Studio) y servicios de backend (ArrowDB, que proporciona servicios comunes para crear, modificar, consultar y borrar objetos comunes a muchas aplicaciones, como usuarios, lugares y fotografías y ArrowPush, que permite realizar notificaciones push)

Frameworks de desarrollo web híbrido

Existen innumerables frameworks de desarrollo híbrido (hemos inventariado 14, pero hay muchos más; Ionic, Emy, Famo.us, PhoneGap, KendoUI, JQuery Mobile, JQTouch, OnsenUI, PhoneJS, Enyo, Intel App Framework, lavaca, mgwt, lungo). A continuación describimos los que son más representativos y creemos más adecuados, tanto por la comunidad de usuarios que los utilizan, su esquema de licenciamiento (open source) y el soporte empresarial que poseen.

1. **Sencha Touch**

Es el framework de desarrollo para móviles y tabletas de Sencha, la empresa que está detrás de Extjs. El objetivo central de este framework es la creación de aplicaciones móviles que se parezcan los máximo posible tanto en rendimiento como en *look-and-feel* a las aplicaciones nativas. Permite desarrollar aplicaciones web e híbridas, integrándose en este último caso con Cordova y Phonegap. Tiene las siguientes características:

- Proporciona más de 50 componentes gráficos
- Incluye librerías para mostrar gráficos (incluso
- Soporte para la integración con cualquier backend, ya que la capa de datos está perfectamente aislada de la tecnología de obtención de los mismos.

2. **IONIC**

Es un framework que está orientado sobre todo a la interfaz gráfica e interacción con el usuario. Por lo tanto, no es un framework destinado a ser usado de forma independiente sino en combinación con otros frameworks como PhoneGap, Cordova o Trigger.io.

3. **PhoneGap y Cordova**

Estos dos frameworks se han mencionado repetidamente en los párrafos anteriores.

PhoneGap es una plataforma open source que permite a los desarrolladores empaquetar aplicaciones híbridas y por otro lado ofrece un api javascript para poder acceder a los sensores/hardware (cámara, gps, etc.) u otros recursos (contactos, sistema de ficheros, etc) de los dispositivos en los que se ejecuta. En 2011 Adobe (que es la empresa que mantiene PhoneGap) creó a partir de PhoneGap el proyecto Apache Cordova, con lo que a partir de ese

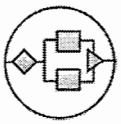
momento PhoneGap utiliza Cordova internamente como "*motor*", ofreciendo PhoneGap servicios de valor añadido como PhoneGap Build, que permite empaquetar las apps en la nube.

Conclusiones

Establecer un framework de referencia para el desarrollo de aplicaciones móviles es prácticamente imposible. El motivo es que la tecnología web no está lo suficientemente madura como para ofrecer garantías de rendimiento y look-and-feel adecuados para la mayoría de las aplicaciones. Por otro lado el uso de frameworks nativos o plataformas que generan código nativo deberían estudiarse para aplicaciones muy concretas donde, bien por el uso continuado y por los requisitos técnicos (gráficos, interacción, realidad aumentada, etc.) sea necesario aprovechar al máximo las capacidades de los dispositivos. Es por ello que recomendamos realizar una consultoría específica (o esperar a fase de oferta), en la que realizar un análisis de los casos de uso a movilizar, los requisitos de rendimiento y usabilidad, etc. para poder definir el/los frameworks a utilizar.

Por último indicar que grandes empresas u organizaciones que apostaron fuerte por el desarrollo de apps web para móviles en HTML5 y javascript, han migrado al cabo de cierto tiempo a aplicaciones nativas. Este es el caso de Facebook (2012), Linkedin(2013) y Wikipedia(2014) migración sus *apps* web o híbridas a *apps* nativas. El motivo principal que aducen es que no existen todavía herramientas que permitan depurar problemas de rendimiento, por lo que es muy difícil optimizar las aplicaciones basadas en HTML5 y javascript.

3.1.1.3 Capa de Exposición e integración de Servicios



La capa de Exposición de servicios estará compuesta por servicios basados en Spring y una capa por encima de esta de servicios web Rest encargada de realizar la comunicación entre la capa de presentación y la capa de servicios Spring.

Spring MVC

Contenedor ligero que actúa de soporte y de unión con el resto de elementos de la arquitectura, haciendo uso además del conjunto de módulos que posee para garantizar diversos componentes solicitados.

Las principales características de Spring son:

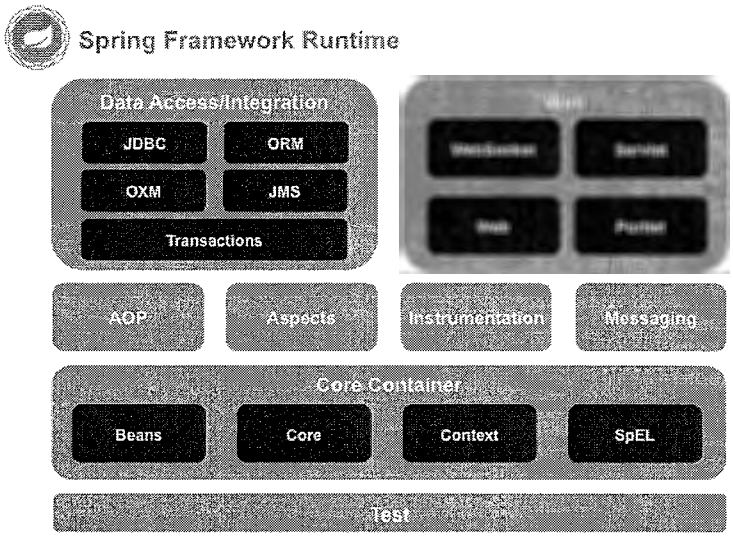
- **Inyección de dependencias**: Spring consigue un débil acoplamiento gracias a la inyección de dependencias (DI). El contenedor inyecta las dependencias durante la instanciación de los objetos que gestiona, de ésta forma, éstos no tienen que buscar las referencias de los otros objetos que usen, por lo que se reduce el acoplamiento, facilitando el mantenimiento y las pruebas.

- **Orientación a aspectos**: la AOP nos permite separar la lógica de negocio de los servicios de sistema transversales, tales como la auditoría, el logging y la gestión de transacciones. De esta manera, los objetos de aplicación únicamente contienen lógica de negocio, y mediante aspectos, definimos de manera externa los servicios transversales.

- **Contenedor**: Spring es un contenedor puesto que contiene, gestiona el ciclo de vida y gestiona las configuraciones de los objetos de aplicación. Permite declarar cómo será la instanciación de un objeto: *singleton* (objeto único), *prototype* (un nuevo objeto por cada llamada, etc.; la configuración de los mismos (propiedades), así como la asociación existente entre los objetos.

- **Framework**: Spring es un framework compuesto por diversos módulos permitiendo así la creación de aplicaciones empresariales.

Spring Framework está compuesto por un conjunto de 20 módulos. Los módulos se agrupan en Core Container, Data Access/Integration, Web, AOP, Instrumentation y Test tal y como se puede apreciar en la siguiente figura:



Generalmente, cuando se emplean casi todos los módulos, se tiene lo necesario para construir cualquier aplicación empresarial. Si bien, Spring permite la integración con otros frameworks y librerías.

Todos los módulos se encuentran definidos sobre el contenedor Core, que es el responsable de la instanciación, ciclo de vida y configuración de los beans.

- **Core Container:** Se encuentra compuesto por los módulos *Core*, *Beans*, *Context* y *Expression Language*.

  Core y *Beans*: son las piezas fundamentales del framework, ya que garantizan el IoC y la inyección de dependencias. La *BeanFactory* consiste en una implementación del patrón de factoría, eliminando la necesidad de definir singletons y permitiendo desacoplar la configuración y la especificación de dependencias de la lógica actual del programa.

  Context: este módulo añade soporte para la internacionalización, propagación de eventos, carga de recursos estáticos y la creación transparente de contextos. Adicionalmente, provee servicios tales como email, JNDI, integración con EJBs, invocaciones remotas y planificadores.

  Expression Language: permite consultar y manipular grafos de objetos en tiempo de ejecución. Se trata de una extensión al unified expresión language especificado en la

especificación JSP 2.1. El lenguaje permite acceder/modificar valores de propiedades, invocar métodos, acceder al contexto de los arrays, colecciones e indexadores, operadores lógicos y aritméticos, etc.

- **Data Access/Integration:** Se encuentra compuesto por los módulos JDBC, JMS, ORM, OXM y *Transaction*.

  JDBC: facilita una capa de abstracción eliminando la necesidad de escribir y parsear códigos de error específicos por proveedor de base de datos.

  JMS: este módulo permite las funcionalidades de envío y recepción de mensajes de *Queues* y *Topics*. También permite el consumo asíncrono de mensajes mediante los MDPs (Message *Driven Pojos*).

  ORM: Spring ofrece un amplio soporte para el trabajo con diferentes motores de persistencia (Hibernate, TopLink, IBatis) y también para trabajar con JPA permitiendo configurar estos contenedores y su transaccionalidad desde el contexto de Spring.

  OXM: capa abstracta que facilita el mapeo entre objetos y XML usando JAXB, Castor, XMLBeans, JiBX y XStream.

  Transaction: el módulo soporta la gestión de transacciones de manera tanto programática como declarativa (en base a configuración o anotaciones). Puede ser usado en combinación con ORM y con JDBC.

- **Web:** Encontramos los módulos *Web*, *Web-Servlet*, *Web-Struts* y *Web-Portlet*.

  Web: ofrece funcionalidades web tales como la subida de archivos multiparte desde formularios, la inicialización del contenedor IoC usando servlet listeners... Contiene también las partes relacionadas con el soporte de invocación remota de los servicios basados en web.

  Web-Servlet: contiene la implementación Spring MVC para las aplicaciones web.

  Web-Struts: permite integrar una aplicación Struts dentro de una aplicación Spring.

  Web-Portlet: ofrece la implementación MVC para ser usada en un entorno de portlets.

- **AOP:** El módulo de Spring AOP ofrece una implementación de programación orientada a aspectos (AOP Alliance-compliant) y también ofrece soporte para AspectJ.
- **Test:** Ofrece soporte para testar los componentes de Spring con JUnit y TestNG. También ofrece mock objects para poder probar el código de manera aislada.

Al ofrecer Spring soporte a todas las capas de desarrollo, permite el desarrollo de toda la aplicación **basado en Frameworks**:

- Proporciona al programador un **modo de trabajo**.
- Facilita la **colaboración**, puesto que estandariza el código, a nivel de componentes, permitiendo ahorrar tiempo e incrementando la calidad del software.

El objetivo central de Spring es permitir que objetos de negocio y de acceso a datos sean reusables, no atados a servicios JEE específicos. Estos objetos pueden ser reutilizados tanto en entornos JEE (web o EJB), aplicaciones standalone, entornos de pruebas, etc. sin ningún problema.

Spring funciona en entornos J2SE y en entornos JEE ligeros. Spring es portable entre diferentes servidores de aplicaciones y contenedores de servlets, como por ejemplo Websphere, Weblogic, JBoss, Resin, Geronimo, Tomcat y Jetty. Actualmente, tanto el desarrollo de aplicaciones web como de servicios con Spring permite escalabilidad, balanceo de carga y alta disponibilidad.

3.1.1.4 Capa de persistencia – jpa + hibernate

Java Persistence API, más conocida por sus siglas JPA, es la API estándar para la Plataforma Java/JEE de persistencia para bases de datos relacionales.

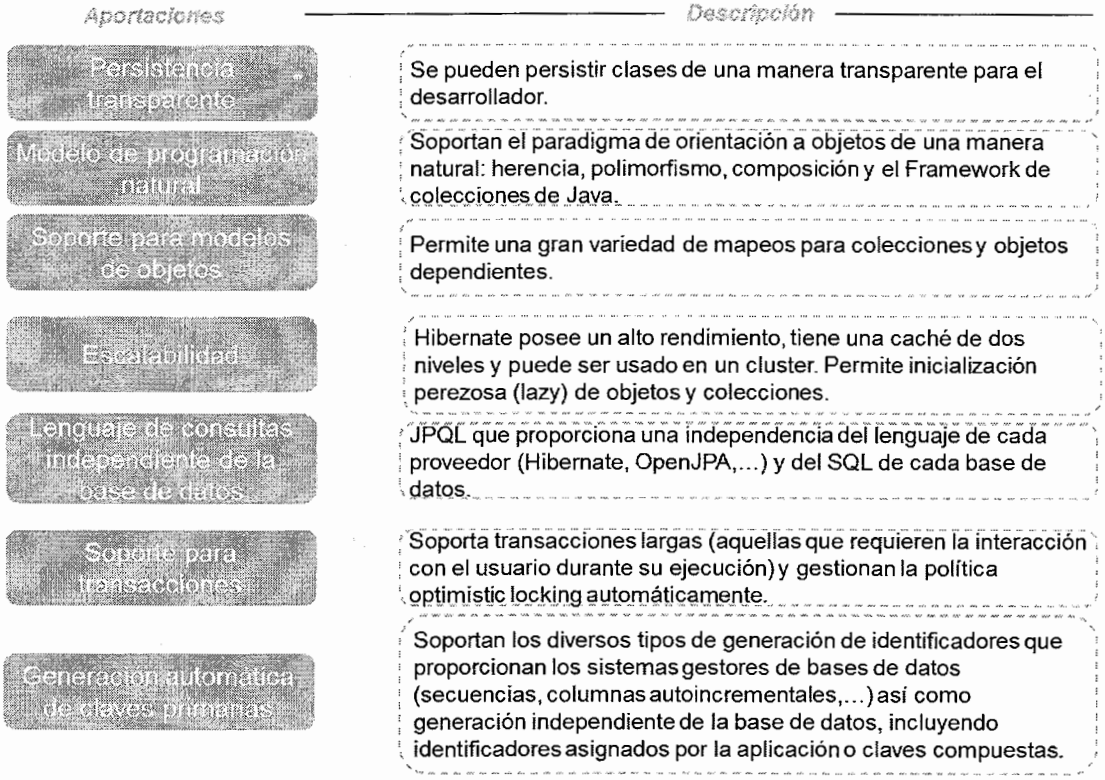
En el contexto de Persistencia cubre tres áreas:

- La API en sí misma, definida en javax.persistence.package
- La Java Persistence Query Language (JPQL)
- Metadatos objeto/relacional

Existen numerosas implementaciones del API JPA, siendo la más usada (estándar de facto) en el mundo JEE Hibernate. Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Hibernate permite desarrollar clases persistentes a partir de clases comunes, incluyendo asociación, herencia, polimorfismo, composición y colecciones de objetos. El lenguaje de consultas de Hibernate HQL (Hibernate Query Language), diseñado como una mínima extensión orientada a objetos de SQL, proporciona un puente elegante entre los mundos objetual y relacional. Hibernate también permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

Soporta prácticamente todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones standalone.



3.1.1.4.1 Pruebas unitarias

Las pruebas unitarias garantizan que los componentes, de forma individual y aislada, se comportan según lo esperado. Para cada una de las capas de las aplicaciones se desarrollan test unitarios para mantener la consistencia dentro de cada capa.

Se recomiendan los siguientes frameworks:

En java: Junit + Mockito

Se propone utilizar como framework de pruebas unitarias **Junit** y como framework para mockear aquello que no estemos probando se usará **Mockito**.

Junit es la herramienta de pruebas unitarias más conocida y empleada en la comunidad java. Desarrollada inicialmente por Erich Gamma y Kent Beck (precursor del proceso de desarrollo basado en pruebas *TDD*), es sencilla de usar, muy ligera y bastante potente, con un desarrollo activo que periódicamente añade mejoras como el uso de reglas generales para reducir el código de *setup-teardown* repetitivo (*@Rule*), la inclusión de nuevas formas de expresar las aserciones (*@Theories*), etc.

Mockito es un framework utilizado para simular las respuestas de aquellos componentes con los que se integra el componente que estamos probando. De esta manera podemos centrarnos en probar toda la funcionalidad de un componente sin tener que levantar la aplicación o preocuparnos del comportamiento de otros componentes que no estamos probando (p.e., que la BBDD esté en determinado estado, que un servidor esté levantado, etc.)

En javascript: Jasmine + PhantomJS + JSLint

Como se ha mencionado anteriormente, extjs implementa el patrón MVC y, a partir de la versión 5, también implementa el patrón MVVM). Los componentes de código donde las pruebas unitarias tienen sentido serían el controlador (MV**C**), el controlador de modelo/vista (MVVM) y las vistas; sobre el modelo no hay nada que probar (no tiene comportamiento). Los frameworks recomendados para realizar estas pruebas son los siguientes:

Jasmine. Se utiliza para codificar las pruebas unitarias en javascript, mediante una sintaxis orientada a comportamiento; ofrece mecanismos análogos a junit para preparar y limpiar el entorno antes y después de cada test, realizar las aserciones, utilizar "*test doubles*", etc.

JSLint. Dado que javascript es un lenguaje interpretado, los errores de sintaxis sólo aparecen en ejecución. JSLint es una librería javascript que verifica que la sintaxis de código javascript. JSLint se puede ejecutar con maven, mediante el plugin jslint-maven-plugin.

PhantomJS. Es un navegador sin interfaz gráfica basado en webkit que se ejecuta por línea de comandos y se puede manejar usando un API javascript. Es necesario ya que las vistas actualizan el DOM, y por tanto es preciso ejecutar los tests en el contexto de un navegador. Existen varios plugins de maven que permiten lanzar tests de jasmine sobre phatomjs.

La selección de las herramientas mencionadas anteriormente garantizan, en la parte que les compete, las propiedades básicas que deben poseer las pruebas unitarias. Sin embargo es labor del programador codificar las pruebas (o elegir las pruebas Estas propiedades se referencian a menudo mediante el acrónimo FIRST:

- **Fast**. Todos los frameworks mencionados son muy ligeros y pueden ejecutar una o varias pruebas unitarias tanto desde los entornos de desarrollo como mediante herramientas de construcción como maven. Por otra parte, si a la hora de codificar una prueba esta tarde del orden de segundos, dicha prueba no se debería calificar como prueba unitaria y debería ejecutarse en otro grupo de pruebas.
- **Independent**. Las pruebas deben ser independientes, lo que significa que se pueden ejecutar en cualquier orden. Junit ofrece métodos de preparación (setUp) y limpieza (tearDown) para gestionar aquellas variables de estado de la prueba que se comparten entre los tests (principalmente objetos mock). Junit ofrece también "*reglas*", que permiten agrupar el código repetitivo de preparación y limpieza, haciendo los tests más mantenibles.
- **Repeatable**. Indica que las pruebas se puedan ejecutar en cualquier entorno (desarrollo, integración, etc.). Ninguno de los frameworks mencionados influye en este aspecto; es misión del programador construir pruebas unitarias de manera que no dependan del entorno.
- **Self Validating.** Es la propia prueba la que debe indicar si ha pasado o ha fallado, sin requerir ninguna intervención humana para validar la prueba. Junit proporciona métodos y anotaciones para definir las postcondiciones de una prueba (aserciones). Mockito por su parte permite verificar que la implementación del componente está de acuerdo a unas especificaciones (métodos verify para verificar las integraciones).
- **Timely.** Indica que las pruebas se deben definir **ANTES** de la implementación. Está relacionado con TDD, (Kent Beck, 2008).

3.1.1.4.2 Pruebas Integradas

Se recomienda desarrollar test integrados entre cada una de las capas de la aplicación para confirmar la consistencia entre capas. Este tipo de pruebas compruebas que los componentes de la aplicación funcionan correctamente actuando en conjunto.

Son pruebas dependientes del entorno en el que se ejecutan. Si fallan, puede ser que el código sea correcto pero que haya un cambio en el entorno.

Capa de presentación

Se recomienda usar **Siesta** para realizar las pruebas integradas de la capa de presentación (mockeando la capa de servicios, usando *proxies* locales). Es un producto parecido a Selenium, que permite "*testear*" el DOM y simular interacciones del usuario con la interfaz gráfica. Se puede utilizar junto con cualquier librería javascript que manipule el DOM - jQuery, Ext JS, NodeJS, Dojo, YUI etc. Permite realizar verificaciones de todo tipo, desde aserciones simples de comparación entre objetos hasta verificaciones de visibilidad de los componentes en la pantalla. Es recomendable que por lo menos se realicen pruebas integradas de aquellos componentes o *widgets* que sean comunes a varias pantallas.

Capa de negocio

En este punto se recomienda usar JUnit + Spring Test para realizar pruebas de integración, simplemente no implementando mocks o stubs, y centrándose en probar el comportamiento de los componentes en su conjunto. Dado que el mantenimiento de los test de integración que prueban la capa de persistencia son bastante costosos (es preciso preparar y restablecer los datos de la BBDD para los tests), se recomienda elegir cuidadosamente las pruebas para equilibrar el valor que aportan dichas pruebas con el coste que representan.

Pruebas integradas desde la capa de servicios

Otra estrategia bastante común consiste en probar los servicios REST más representativos mediante alguna herramienta de pruebas de servicios (como SOAP-UI). La elección de estos servicios debe realizarse de forma cuidadosa, intentando maximizar el número de integraciones cubiertas.

3.1.1.4.3 Pruebas Funcionales

Mediante las pruebas funcionales se valida el cumplimiento del software desarrollado contra las funciones detalladas en el documento de requisitos, persiguiendo reducir los defectos en la etapa de operación y permitiendo la corrección de los mismos a costos reducidos al ser encontrados en etapas tempranas. Se les llama también pruebas de caja negra, ya que los responsables de pruebas enfocan su atención a las respuestas del sistema en función de los datos de entrada y su resultado en los datos de salida. Estas pruebas requieren integración con todas las capas y sistemas, salvo aquellos sistemas externos que deban ser "taponados", bien porque no estén dedicados exclusivamente al entorno de pruebas funcionales, bien porque ni siquiera existan en dicho entorno.

Dentro de las pruebas funcionales distinguimos dos partes:

- Documentación de Pruebas: Proponemos el uso de la herramienta **TestLink.** De esta forma se tiene trazabilidad entre los requisitos establecidos y las pruebas funcionales asociadas a cada uno de ellos.
- Automatización de Pruebas: Proponemos Siesta integrado con **Selenium** (***web driver***) como herramienta de automatización de pruebas funcionales. De esta forma se realizan de una forma sencilla y rápida tanto las pruebas funcionales de regresión como las nuevas definidas en cada iteración del desarrollo.

3.1.1.4.4 Pruebas de rendimiento

Con las pruebas de rendimiento se pretende medir la capacidad de respuesta en un entorno esperado (pruebas de carga), degradación (pruebas de rendimiento) y límites del sistema (pruebas de stress) software, principalmente.

Se propone el uso de **Apache JMeter.** Se trata de una herramienta Open Source diseñada para analizar y medir el rendimiento de un sistema mediante la ejecución de pruebas funcionales de carga. Estas son algunas de sus características:

- Multiplataforma: Dado que es un framework java, funciona en aquellos sistemas donde exista una máquina virtual de java (descartando sistemas embebidos). Los sistemas sobre los que el equipo de desarrollo de JMETER testea las versiones son los siguientes: Windows 8, Windows 7 , Mac OSX, Mac OS, Linux, Windows XP, FreeBSD, OpenVMS y Solaris SPARC. La lista detallada de las plataformas de testeo de JMETER están enumeradas en https://wiki.apache.org/jmeter/JMeterAndOperatingSystemsTested. Esto no quiere decir que no funcione en otros sistemas operativos. Como se indica en el manual de usuario de JMETER, es una aplicación Java 100% que sólo utiliza el API estándard de Java, por lo que debería ejecutarse correctamente en cualquier plataforma Java. Por otra parte, cada versión de JMETER indica a partir de qué versión de Java es compatible (lo que se indica también en el enlace mencionado). A partir de la versión 2.9 es necesario como mínimo Java 6.
- Permite actuar sobre distintos tipos de servidores: Web (HTTP, HTTPS), SOAP, Base de datos (JDBC), LDAP, JMS, Email (POP3 e IMAP).
- Multihilo: Ejecución de pruebas de carga de forma concurrentes simulando acceso simultáneo de conjuntos de usuarios.
- Permite identificar desviaciones típicas en tiempos de respuesta.
- Monitorización de respuestas en gráficas online.
- Testeo distribuido.

3.1.2 Frameworks adicionales

Además del "*stack*" tecnológico base descrito, existen otros aspectos de arquitectura y funcionales que son necesarios o que aparecen habitualmente y que conviene abordar mediante frameworks o productos adicionales.

3.1.2.1 Seguridad

En el ámbito de la aplicación es preciso realizar un control de autenticación del usuario y autorización de las acciones que puede realizar. En este aspecto recomendamos el uso de **Spring Security**, que tiene las siguientes características:

- Autenticación. Soporta los mecanismos de autenticación HTTP Basic, HTTP Digest, HTTP X509, LDAP, OpenID, basada en formulario, CAS, Kerberos, NTLM, "*Remember me*", etc.
- Autorización. Soporta la verificación en dos niveles; peticiones HTTP (por defecto, mediante "*matchers*" basados en patrones) y sobre métodos java (mediante anotaciones y aspectj). Soporta tanto roles como ACL's.
- Características Web avanzadas, como gestión de la sesión web (timeouts, control de sesiones concurrentes, protección contra ataques "*sesión fixation*").

3.1.2.2 Procesamiento Batch

Es común en prácticamente todos los sistemas de envergadura que tarde o temprano surja la necesidad de procesamiento por lotes, por ejemplo, para importar o exportar datos en/de un sistema (bases de datos a ficheros y viceversa) o realizar transformaciones masivas en los datos de una BBDD.

Una aplicación batch procesa datos de manera automática por lo que debe ser robusta y fiable, ya que en el proceso no existe ninguna interacción humana que soluciones los errores. Dado que contra mayor sea el volumen de datos que se deba procesar, mayor será el tiempo que se emplee en procesarlos, es necesario tener en cuenta el rendimiento desde el principio, ya que normalmente estas aplicaciones deben ejecutarse en una ventana de tiempo. Basada en esta descripción, estos son los requisitos de una aplicación batch:

- Volumen alto de datos
- Automatización
- Robustez
- Fiabilidad
- Buen Rendimiento

Para cubrir todos estos requisitos y posibles necesidades recomendamos Spring Batch, un framework integral de procesamiento por lotes con un API consistente para el desarrollo de aplicaciones batch robustas, que además se integra perfectamente con spring. Estas son sus principales características:

- Modelo de dominio consistente (*"JobLauncher", "JobRepository", "Job", "Step", "ItemReader, ItemProcessor", "ItemWriter"*)
- Gestión de transacciones (una transacción por "paso")
- Procesamiento basado en "trozos" (en terminología Spring Batch se denominan "chunks")
- Entrada/Salida declarativa
- Los procesos se pueden parar, arrancar y rearrancar
- Permite reintentos sobre errores (o ignorarlos)
- Interfaz Web de Administración (Spring Batch Admin)

3.1.2.3 Informes

Prácticamente en todos los sistemas existe la necesidad de generar informes a partir de los datos generados. Dentro del mundo opensource, la herramienta que recomendamos es JasperReports.

JasperReports es el motor de informes de código abierto más popular en la comunidad Java. Es capaz de obtener datos de multitud de fuentes de datos para producir documentos *"pixel-perfect"* que pueden ser visualizados, impresos o exportados a una gran variedad de formatos, entre los que se encuentran HTML, PDF, Excel, OpenOffice y Word. Se encuentra bajo licencia libre LGPLv3.

La librería está totalmente escrita en Java lo que permite una fácil integración en aplicaciones Java o JavaEE. JasperReports forma parte de *JasperSoft Business Intelligence Suite*, que es un conjunto de herramientas integradas para la gestión y creación de informes. Esta librería es el core de los productos *Report Designer, Jaspersoft Studio* y *JasperReports Server*.

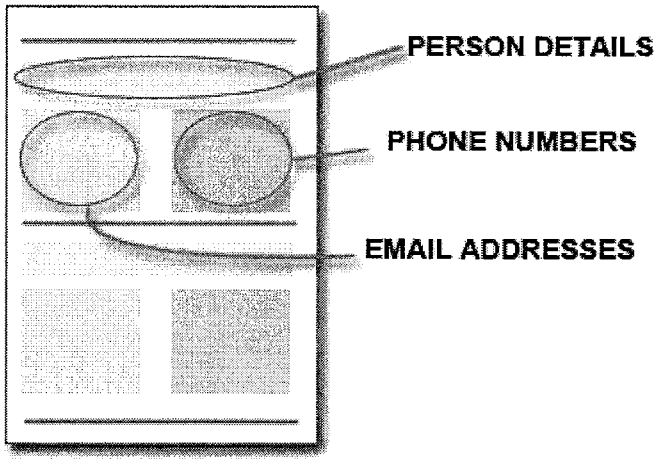
Características

A continuación se enumeran las características y puntos fuertes que hacen que de JasperReport el motor de informes open source más popular. Se hace especial hincapié en los componentes de diseño de informes y su diseñador de informes, formatos de salida, fuentes de datos, escalabilidad e integración con terceros.

Características de Diseño

JasperReport se basa en el paradigma de creación de informes orientados a página y con precisión "*al pixel*" lo cual nos permite crear informes de alta complejidad de forma sencilla. Contiene una completa librería de componentes estándar que pueden incorporarse al informe desde el diseñador, tales como tablas, gráficos, widgets, etc. A continuación se enumerar las características más importantes:

- Soporte para Dashboards, tablas, tablas con referencias cruzadas, gráficos, indicadores y widgets.
- Los "*SubReports*" o SubInformes permite incluir informes dentro de otros informes. Este es uno de los puntos fuertes de JasperReports ya que se pueden crear diseños complejos en un solo documento utilizando diferentes fuentes de datos.



- Creación de diseños interactivos gracias a los elementos de tabla interactivos, marcadores de PDF, gráficos e hiperenlaces.
- JasperReport soporta una gran varidad de formatos de salida entre los que se encuentran PDF, RTF, XML, XLS, CSV, HTML, XHTML, TXT, DOCX y OpenOffice.
- Librerías de estilos y soporte CSS que facilitan la incorporación de estilos definidos en web corporativas e independizar la definición del diseño.
- Incluye otras características como soporte para la integración de códigos de barras, componentes para la rotación visual del texto, impresión condicional, "*cross-tabbin*" o distribución multicolumna.

Características de la herramienta de Diseño

La herramienta de diseño de informes iReport/JasperSoft Studio permite la creación de informes de forma interactiva para JasperReport. Proporciona todas las paletas de elementos necesarias para la creación de informes: gráficos, imágenes, tablas, así como las herramientas necesarias el acceso a fuentes de datos y un diseñador de consultas. Algunas de sus características:

- Código 100% Java y OpenSource con licencia EPL (Eclipse Public License).
- Basado en Eclipse lo que disminuye la curva de aprendizaje ya que Eclipse es un IDE muy extendido entre los desarrolladores Java.
- Vista previa durante la creación del informe que facilita la detección de errores en fases iniciales del desarrollo.

- Soporte para el 98% de las etiquetas de JasperReports.
- Recopilador y exportador integrados.
- Incluye asistentes para la creación de informes y sub-informes con multitud de plantillas y estilos.

Internacionalización

JasperReport permite crear informes para herramientas globales mediante la internacionalización de textos, moneda y signos de puntación en función la configuración del usuario local.

Fuentes de Datos

Una de las fases más importante en la creación de informes es la recopilación de información de terceros. JasperReport tiene la capacidad de acceder a múltiples fuentes de datos de diferentes tipos como pueden ser bases de datos (RDBMS o NoSQL), ficheros, WebServices o servidores de datos mediante una serie de conectores estándar. En el caso de no querer utilizar estos conectores estándar JasperReport proporciona las herramientas necesarias para acceder a través de conectores personalizados. Por último, que se pueden utilizar múltiples fuentes de datos de diferentes tipos en un informe.

A continuación destacamos algunos de las fuentes de datos que proporciona JasperReport:

- *Database JDBC connection*
- *XML file data source*
- *JavaBeans set data source*
- *Custom JRDataSource*
- *File CSV data source*
- *JRDataSourceProvider*
- *Hibernate connection*
- *Spring-loaded Hibernate connection*
- *EJBQL connection*
- *Mondrian OLAP connection*
- *Query Executor mode*
- *Empty data source*
- *Custom iReport connection*
- *XMLA server connection*

Escalabilidad e integración con terceros

En este apartado vamos a comentar las capacidades de escalabilidad e integración con terceros. En el punto anterior ya se comentó el acceso a fuentes de datos de terceros y la capacidad que tiene JasperReport para trabajar con multitud de tipos y su integración dentro de los informes. A continuación comentamos las más destacadas:

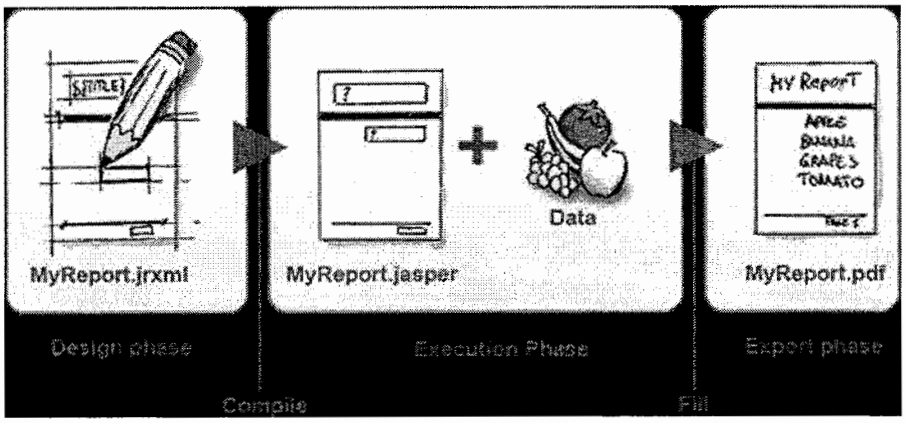
- No existe tamaño máximo de los informes lo que permite la creación de documentos de gran tamaño e elementos gráficos de calidad.
- Report Visualizer están optimizados para gestionar de forma correcta la memoria y el rendimiento de E/S

- Query Governors protegen los recursos del sistema, para ello se estima el costo de la consulta antes de ejecución y evitar consultas que superar unos límites de tiempo especificados.
- Cuando existe requisitos de alto nivel en la gestión de reportes existe la posibilidad de importar los informes a JasperServer. JasperServer un servidor de informes que facilita la centralización de informes, análisis de datos, seguridad, auditoria o análisis OLAP y expone diferentes servicios web (REST, SOAP) que simplifica la integración de los informes en aplicaciones web o móviles.
- Soporte para definición de funciones y expresiones mediante Scripting en lenguaje Java, JavaScript y Groovy). Estas funciones pueden ser invocadas al inicio y final de cada una de las etapas de generación del informe (Informe, Página, Columna o Grupo).
- La integración de JFreeChart con JasperReport permite ampliar el abanico de posibilidad para generación de graficos. JFreeChart es un framework opensource que facilita la creación de gráficos complejos de forma sencilla.

Por último, indicar que JasperReport es un proyecto estable que cuenta con una comunidad muy activa que le permite evolucionar y mejorar. Gracias a su popularidad existe gran cantidad de documentación de calidad y foro donde se pueden consultar dudas o problemas que le han surgido a otros desarrolladores.

Arquitectura

En el siguiente apartado se muestra de forma rápida el ciclo de vida de un informe en JasperReport, centrándonos en los elementos que forman parte de la arquitectura de creación y ejecución del motor de informes. En la siguiente imagen se muestran las fases (Diseño, ejecución y exportación) de un informe:



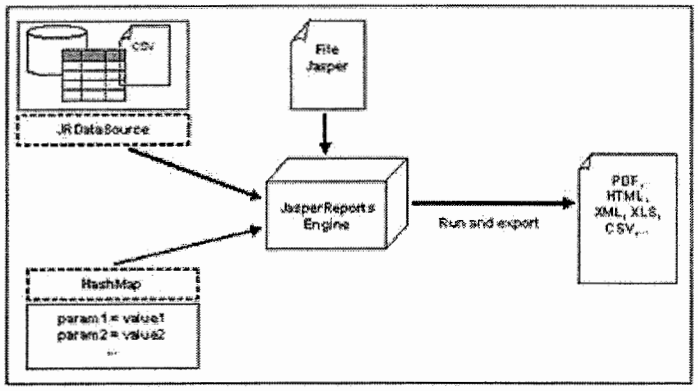
Fase de Diseño

Como su propio nombre indica se realiza el diseño del informe por parte del desarrollador codificándolo en XML utilizando las etiquetas y atributos definidos por JasperReports. Durante esta fase es recomendable la utilización JasperSoft Studio para estructurar correctamente los elementos del informe, definir las fuentes de datos y los parámetros de entrada.

Una vez finalizado el diseño se debe compilar el fichero JRXML para obtener un informe que pueda ser interpretado por el motor de informes (fichero *.jasper).

Fase de Ejecución

Como se indica en puntos anteriores el motor de informes de JasperReport funciona de forma similar en aplicaciones Java y JavaEE, por tanto, el comportamiento en similar. Durante esta fase el motor de reglas es capaz de interpretar el informe compilado obtenido en la fase de diseño para generar un informe imprimible en un formato determinado. Entre las tareas que se realizan está la obtención de los parámetros de entrada y la obtención de los datos que alimentan el informe en el caso de que se haya definido que el informe tiene datos dinámicos, sumatorios, paginado, ...



Fase de Exportación

Si durante la fase de Ejecución no se ha producido ningún error el motor de reglas está en disposición de exportar el informe a alguno de los formatos permitidos. Una práctica habitual es enviar como parámetro de entrada el formato de salida del informe. Una vez generado JasperReport retorna el informe a la aplicación para que realizan las operaciones deseadas con el mismo: almacenar, imprimir o visualizar.

3.2 Arquitectura hexagonal

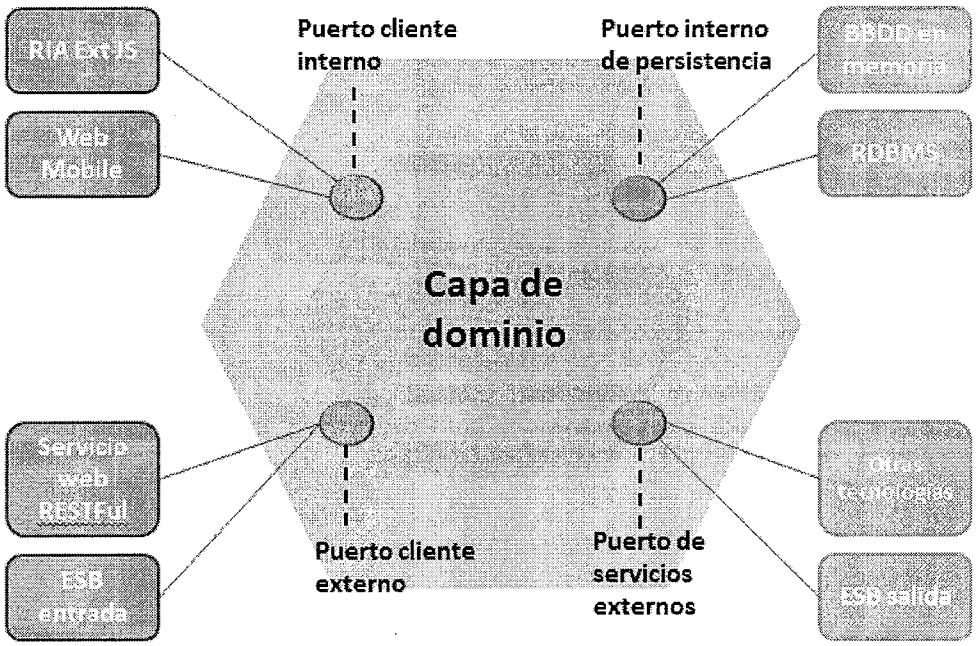
Si el sistema a construir fuese de gran extrensión con diferentes y numerosas áreas funcionales, se propone seguir un modelo de arquitectura hexagonal (también conocida como de puertos y adaptadores) en la construcción de cada una de las aplicaciones de servicios que contenga.

La Arquitectura Hexagonal define capas conceptuales de responsabilidad de código y, a continuación, señala la manera de desvincular el código entre las capas. Los objetivos son dos; permitir que una aplicación, pueda ser usada tanto por usuarios, programas, procesos por lotes o pruebas automáticas y que, a su vez, pueda ser probada de forma autónoma, e independiente de los dispositivos y bases de datos que finalmente usará.

Una arquitectura hexagonal tiene tres capas — **modelo de dominio, puertos y adaptadores** — con el **modelo de dominio** como parte central, que contiene toda la lógica y las reglas de la aplicación. En la capa de dominio no hay relaciones ni responsabilidades tecnológicas, como podrían ser contextos HTTP o llamadas a base de datos, permitiendo que no le afecten cambios en la tecnología.

Rodeando el modelo de dominio se encuentra la capa de **puertos** recibir todas las solicitudes que corresponden a un caso de uso que orquesta el trabajo en el modelo de dominio. La capa de puertos ejerce de frontera, con las entidades del modelo de dominio en el interior y las entidades externas en el exterior.

Por último, rodeando la capa de puertos se encuentra la capa de **adaptadores**, donde se realiza la integración tecnológica y la transformación de los datos de entrada/salida. Por ejemplo, con una solicitud HTTP el adaptador la transforma en una llamada al dominio y realiza a continuación un "*marshalling*" de la respuesta del dominio de nuevo al cliente mediante HTTP. En el adaptador no hay lógica de negocio; su única responsabilidad es una transformación técnica entre el mundo externo y el del dominio. Cualquier adaptador que se adhiere al protocolo de un puerto puede utilizarlo y varios adaptadores pueden utilizar el mismo puerto.



En la figura se representa un ejemplo de arquitectura hexagonal, con cuatro puertos. El hexágono hace hincapié en el hecho de que hay múltiples puertos, tanto de entrada al sistema como de salida del mismo. Los adaptadores de capa de presentación (RIA ExtJS y web mobile) utilizan los puertos primarios, mientras que los puertos secundarios permiten al sistema interactuar con los adaptadores en la capa de infraestructura.

3.3 Alternativas y Justificaciones

En este apartado se pretende realizar un repaso a las recomendaciones expuestas, justificándolas y presentando alguna alternativa, en aquellos casos en los que se haya considerado oportuno.

En la capa de presentación web existen innumerables frameworks pero pocos que tengan las características de extjs; 100% javascript, una empresa detrás que ofrezca un soporte y una estrategia sólida y una librería de componentes (gráficos y de soporte) tan extensa. Sin embargo hay alternativas, las más importantes serían las siguientes:

1. **ZK**. Es un framework muy potente, con más de 200 componentes "*out-of-the-box*" y con soporte empresarial. El principal problema es que está orientado a servidor, el MVC se implementa en un java y lenguajes de marcado principalmente. Los equipos especialistas en programación de presentaciones web necesitan tener el control "*a nivel de pixel*" de la interfaz gráfica, y esto sólo se consigue trabajando directamente con html, css y javascript. Es un framework más apropiado para programadores con poca experiencia en desarrollos "front" que para especialistas.

2. **AngularJS.** Está teniendo una rápida difusión, y consideramos hacer un seguimiento de esta tecnología, pero adolece desde nuestro punto de vista de los siguientes problemas:

    a. Utiliza plantillas HTML para enlazar la vista con el controlador y el modelo. Esto puede suponer un problema cuando se trabaja en equipo, debido a la inherente falta de modularidad de las páginas html.

    b. No proporciona una librería de componentes de gráficos. Estos componentes son proporcionados por terceros, como AngularUI, Kendo o Wijmo, pero no alcanzan el número de componentes proporcionado por extjs.

El resto de tecnologías (GWT, Vaadin, Ember.js, jexpresso, dojo, etc.) o son comparables a Extjs o los hemos descartado por razones similares a las expuestas en los casos de ZK o angular.

Cabe mencionar una iniciativa de la compañía Red Hat; en lenguaje de programación Ceylon. Es un lenguaje orientado a objetos, con tipado fuerte y, lo más importante, diseñado para poder ejecutarse en las máquinas virtuales de java y javascript. De esta manera se pretende unificar el lenguaje de programación de *front y backend's.* Todavía no dispone de librería gráfica, pero es una plataforma a tener en cuenta en el futuro a medio/largo plazo.

Con respecto al ecosistema Spring y a pesar que el stack JEE estándar ha ido paulatinamente incorporando características importadas de spring (CDI, Batch 1.0) pero Spring, dada su naturaleza independiente y no limitada a definir un estándar (comités, *"papers", etc.*) añade prácticamente cada año nuevas características tanto en el core como en proyectos del ecosistema (spring integration, spring batch, spring boot, etc.). Con un enfoque de arquitectura hexagonal y con cuidado en la programación, evitando o aislando referencias explícitas a componentes del ecosistema spring, se pueden minimizar las dependencias con Spring. Además, Spring intenta adoptar los estándares definidos en JEE; por ejemplo, las anotaciones *@Inject* y *@Named* definidas en **JSR-299** se implementaron en la versión 3.0 de spring.

En cuanto a la capa de persistencia, se recomienda JPA 2.X (hibernate 4.3 o superior). Es la tecnología de persistencia estándar en java y el framework de referencia, que en la versión 4 introdujo el soporte a bases de datos multi-tenant. Este concepto se define como la capacidad de particionar virtualmente los "*clientes*" (*tenants*) de una aplicación empresarial en lugar de almacenar todos sus datos en un espacio común. Este principio permite mejoras en la gestión, seguimiento e incluso la seguridad y es muy útil para los grandes proveedores de servicios. Las compañías que ofrecen infraestructuras de nube pueden beneficiarse de múltiples clientes también. Hay varias formas de implementar este principio, que incluyen:

1. Una base de datos y / o esquema diferente para cada cliente.

2. La misma base de datos/esquema para todos los clientes, pero con una columna extra (por ejemplo tenant_id) en todas las tablas que se pueden utilizar para filtrar los datos

Hibernate soporta el primer método en la versión 4.0. El soporte para el segundo método (es decir, discriminador multicliente) está prevista en la próxima versión.

En relación a las pruebas unitarias en java, los frameworks recomendados son las opciones clásicas. Una alternativa interesante a junit es spock; Spock es un framework de pruebas y especificación para Java y Groovy (de hecho nació en la comunidad groovy). Lo bueno de spock es que permite especificar las pruebas unitarias mediante el estilo "**given-when-then**" de las herramientas BDD, lo cual mejora la legibilidad del resultado (en junit es habitual definir nombres de métodos de prueba larguísimimos con objeto de indicar las precondiciones, la acción y postcondiciones). Con Spock no sería necesario utilizar mockito, ya que tiene su propia implementación para implementar "*test doubles*". Sin embargo consideramos que dependerá de la experiencia del equipo con herramientas BDD o en programación con groovy, la curva de aprendizaje puede ser demasiado exigente.

Con respecto a los frameworks de pruebas unitarias en capa de presentación, son los recomendados por Sencha y no se han analizado con el suficiente rigor otras alternativas.

En cuanto a pruebas de rendimiento, Apache JMETER es la referencia en el mundo opensource. No hemos encontrado otra herramienta libre que tenga la misma funcionalidad y que siga desarrollándose y ampliándose (la última versión es de marzo de 2015).

3.3.1 SOAP vs REST

Lo primero de todo conviene aclarar que cuando hablamos de SOAP y REST estamos hablando de cosas diferentes. SOAP es un protocolo de aplicación en el que se define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Normalmente este intercambio se produce utilizando HTTP, pero puede utilizar otros mecanismos (JMS, TCP, etc.). REST (Representational State Transfer) por contra se refiere a la forma de construir servicios en un entorno distribuidos. **Es un estilo de arquitectura software** para sistemas hipermedia distribuidos, como la World Wide. En la mayoría de las ocasiones la construcción de servicios REST, debido a la influencia de Internet (no olvidemos que el protocolo HTTP fue definido en primera instancia por Roy Fielding, que también fue el que definió "***REST***" en su tesis doctoral del año 2000) y la cada vez mayor exposición de servicios en la misma por parte de organizaciones, el protocolo que se utiliza es HTTP(S).

El paradigma utilizado en la inmensa mayoría de servicios web basados en SOAP es RPC (*Request Procedure Call*). Cuando se diseña un API en RPC se utilizan "verbos" para indicar las operaciones, exponiendo la funcionalidad como **llamadas a funciones** que aceptan parámetros, e invoca estas funciones a través del método HTTP que parezca más adecuado - un 'GET' para consulta, un 'PUT' para una "asignación", etc., pero el nombre del verbo es puramente incidental y no tiene relación verdadera sobre las funcionalidades realmente implementadas, ya que se llama a una URL diferente cada vez. Los códigos de retorno se codifican a medida como parte del contrato de servicio.

En REST, el concepto central es el **recurso** y las operaciones están prefijadas y estandarizadas para obtener y modificar el estado de dicho recurso y que **aplican a todos los recursos** (en HTTP sería los métodos HTTP; GET, PUT, POST, DELETE, PATCH, etc.). Para ello además es necesario tener una sintaxis universal de identificación de los recursos (en la Web es la URI), un protocolo sin estado (en la Web es HTTP) y la capacidad de manejar varios medios de representación (hipermedia) de dichos recursos (por ejemplo, XML, JSON, imágenes, páginas, etc.).

Dicho esto, nuestra recomendación es utilizar una arquitectura REST para la construcción de servicios (que se expongan hacia otros sistemas, aplicaciones web o incluso en la WWW) en los que **no haya requisitos específicos de seguridad** y en los que el protocolo HTTP(S) sea posible, las **operaciones sean sin estado** y se diseñen los recursos lo más "***RESTFul***" posible, utilizando JSON como formato de representación. JSON es un formato ligero de intercambio de datos análogo a XML pero mucho más ligero. En medidas experimentales se ha comprobado que XML genera hasta un 80% más de *overhead* que JSON, sin comprimir. Spring MVC proporciona los mecanismos técnicos para implementar servicios REST.

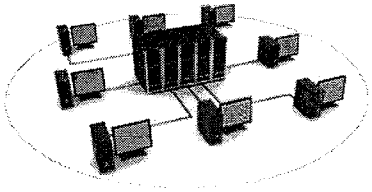
En cuanto a SOAP, nuestra opinión es limitar su ámbito a servicios con necesidades especiales de seguridad (cifrado con firma digital), servicios con estado (información contextual o estado conversacional) o servicios que se consuman desde otros sistemas y que además requieran la definición de un contrato estricto, ya que existen tanto los estándares (WS-Security, WS-Coordination, WS-Realibity, etc.) y los frameworks para poder implementar estas necesidades específicas. Dicho esto, en la mayoría de las aplicaciones de gestión, este tipo de servicios son una minoría.

4. ARQUITECTURA CLOUD

El hecho que las universidades unifiquen sus ecosistemas tecnológicos bajo la arquitectura recomendada, reportará varios beneficios tanto para las distintas universidades, como para sus usuarios. Con el objetivo de maximizar dichos beneficios, se hace fundamental incluir el paradigma del Cloud Computing como parte del nuevo ecosistema, permitiendo optimizar los recursos tanto económicos como de trabajo.

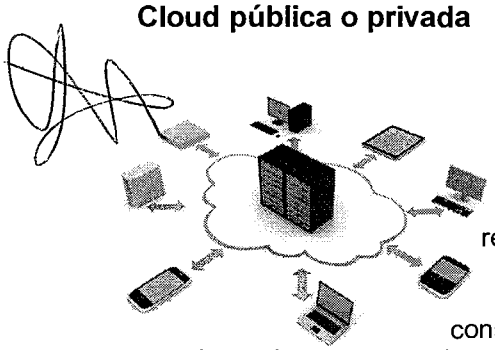
Con esto no queremos decir que todas las aplicaciones deberían estar en la nube, puesto que en función de varios aspectos tales como para que uso fue desarrollada, el acceso por parte de los usuarios a la información que contiene o incluso la interconexión con otras fuentes de datos, nos debería inducir a una valoración respecto si debemos incluirla o no en Cloud, y en caso afirmativo, si dicha Cloud debería ser privada, pública o híbrida.

In house



Adecuado para aplicaciones sin acceso externo o con un uso no colaborativo. Entendemos que será el modelo menos usado por parte de las universidades, puesto que no permite el compartir los recursos suponiendo unos costes más elevados.

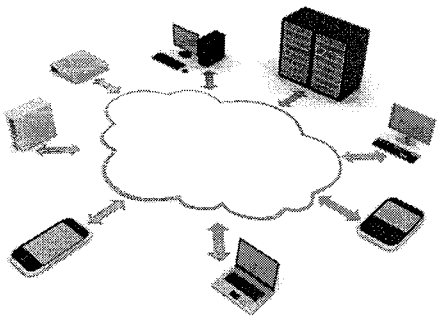
Cloud pública o privada



Son apropiadas para aquellos ecosistemas donde todos sus elementos podrían encontrarse en la nube. Son modelos aconsejados para nuevos desarrollos, puesto que permite la optimización de los recursos, tanto a nivel de infraestructura como de aplicaciones.

La decisión respecto a si es pública o privada, deberá tomarse considerando todas las características, aunque la arquitectura propuesta permite el uso de ambas. Así mismo, hay algunos factores que debemos tomar en especial consideración:

- Económico. La nube privada precisa de una inversión en infraestructura mayor, en previsión de la demanda del conjunto de aplicaciones alojadas en la misma. Además los costes operativos de la nube privada tienden a ser superiores.



- Escalabilidad. La nube pública nos dará unas capacidades muy superiores y que debemos tener en cuenta para aplicaciones que pudieran tener picos de demanda de recursos importantes.

- Seguridad. En líneas generales la seguridad de ambas nubes son similares, por lo que no es un elemento decisorio, aunque para algunos usuarios sí que puede ser importante el control sobre los datos y la infraestructura que proporciona la nube privada.

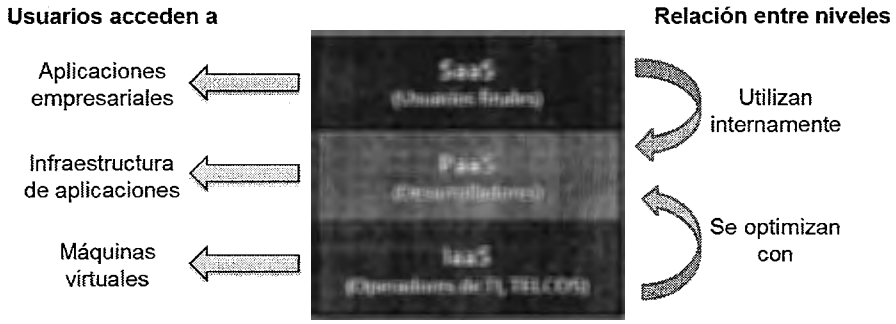
Cloud híbrida

La Cloud híbrida nos permite beneficiarnos de aquello que nos interese tanto de la nube pública como privada. Además, es una opción muy interesante si tenemos algún elemento (aplicación, base de datos, etc.) que no podemos alojar en una nube pública, bien sea por sus características técnicas o del fabricante o bien sea por decisión de la propia organización.

Por ello, es una opción cada vez más utilizada por las organizaciones, y si bien no se considere utilizar en un primer término, sí que es importante, contar con una arquitectura y unos proveedores que nos permitan realizarla en cuanto se precise.

4.1 Usando el Cloud Computing

Es importante cuando hablamos de Cloud, que lo hagamos de la pila completa, puesto que será la única forma de poder aprovechar por completo sus capacidades.



Pila Cloud

Uso de la pila Cloud

No vamos a entrar a describir cada una de estas capas, puesto que ya son bien conocidos, pero sí que puede ser interesante realizar algunas recomendaciones a la hora de seleccionar los proveedores más apropiados o configurar nuestro propio servicio. Dicha decisión, puede condicionarnos en aspectos tan importantes como los SLA, tecnologías de desarrollo o una posible migración de una cloud a otra.

Respecto a la arquitectura propuesta, veamos algunas recomendaciones que se deberían seguir a la hora de seleccionar los proveedores en las tres capas:

IaaS

Como ya se ha comentado, la arquitectura propuesta permite la construcción tanto de una Cloud pública como privada, así como la creación de una cloud híbrida si fuese necesario. La mayoría de los proveedores de IaaS están habilitados para dar servicio a la arquitectura propuesta, por lo que la selección del mismo se debería centrar en otros aspectos como SLA, localización geográfica, precio y los servicios que ofrecen.

Aun así, es importante antes de seleccionar nuestro proveedor, nos aseguremos que cumple con los requerimientos necesarios para desplegar nuestra arquitectura. Podemos encontrarnos conexiones que no sean posibles, localizaciones del proveedor que penalicen mucho el ancho de banda o que no cumpla con la legislación vigente en nuestro país.

En este sentido, las tres grandes empresas a nivel mundial (AWS Amazon, Microsoft Azure y Google Cloud) ofrecen servicios de cloud pública a precios muy competitivos.

Si en cambio buscamos soluciones de Cloud privadas, reutilizando las infraestructuras ya existentes en las universidades, nos podríamos decantar principalmente por una de estas dos soluciones que permiten la construcción de todo tipo de Clouds, pero en servidores propios:

- **Openstack:** Solución Open Source bajo licencia Apache pero evolucionado por grandes empresas como Red Hat o HP convirtiéndola en líder del mercado Open Source. Bajo una estructura modular, provee servicios para administrar todo aquello relacionado con un IaaS, dando salida a nuevos servicios de forma semestral. Además, diversas empresas desarrollan soluciones para su uso sobre OpenStack proporcionando soluciones bastante completas.

  El que haya tantas empresas detrás de esta solución, mitiga el principal problema que suele tener este tipo de soluciones, como es el soporte. En cambio suelen hacer desarrollos complementando la solución Open Source, los cuales tan sólo ellos mantienen, por lo que complica el cambiar de proveedor del soporte una vez puesto en marcha.

- **vCloud:** De la californiana VMWare, es la solución de virtualización de máquinas que más se ha extendido por el mundo. Podemos encontrar multitud de empresas que van servicios sobre esta tecnología dada su robustez, aunque resulta algo más cara que OpenStack (incluso incluyendo el soporte) u otras soluciones como RackSpace.

  Aunque inicialmente fue creado para la construcción de cloud privadas, permite la creación de cloud públicas e híbridas con una alta seguridad.

PaaS

Conviene destacar que cuando se habla de PaaS, no debemos limitarlo al ámbito del desarrollo de aplicaciones. Hay herramientas en este nivel del Cloud, que potencian los servicios ya ofrecidos por el IaaS, como pueden ser las herramientas que se describen en el siguiente apartado.

En cuanto a la selección del proveedor, debemos comprobar que permite los desarrollos y la ejecución con la arquitectura propuesta. A pesar de tratarse de una arquitectura con tecnologías muy populares, no todos lo aceptan como sería el caso de la plataforma de desarrollo de aplicaciones Java CloudBees. También es importante que nos permita abstraernos del proveedor de IaaS, permitiendo las migraciones a aquellos que consideremos oportuno.

Nuevamente como ocurría en el IaaS, es importante que antes de contratar a nuestro proveedor, nos aseguremos que puede cumplir con la arquitectura de servicios planteada, permitiendo que las SaaS mantengan distintos niveles de integrando con el PaaS, aunque esto conlleve un uso parcial de las capacidades Cloud.

Por todo ello y siguiendo las recomendaciones de Gartner en su Magic Quadrant de PaaS del 2014, podemos destacar los siguientes:

- **GPaaS (Indra Sistemas):** Plataforma de la española Indra Sistemas, destaca por su independencia tecnológica con otros proveedores, tanto a nivel de IaaS, como lenguajes de programación, bases de datos, frameworks de desarrollo, etc..

  Siendo una plataforma nativa cloud, incluye entre sus servicios un servidor de aplicaciones, base de datos y tecnologías BPM, portal y middleware combinadas con herramientas para construir, desplegar, ejecutar y explotar aplicaciones y servicios. GPaaS además provee una variedad de características cloud como escalabilidad elástica, autoaprovisionamiento, multitenancy de base de datos y servidor de aplicaciones y capacidades para el procesamiento extremo de transacciones.

  Incluye la herramienta GIM, que permite la automatización del despliegue configurando el middleware (Jboss, WebSphere, etc.) y de las aplicaciones desplegadas a partir de las preconfiguraciones definidas de despliegue.

También incluye un marketplace de aplicaciones y servicios llamado ICB, que gestiona el proceso completo desde la publicación de un servicio, hasta el proceso de compra, disponibilización, generación de contrato y valoración por el consumo o prestación del servicio.

- **Agile Apps Live (SoftwareAG):** Proviene de la californiana Long Jump que fue adquirida por la alemana SoftwareAG. Permite el despliegue tanto en cloud públicas como privadas o híbridas, así como on premise, está enfocada a Big Data Cloud, movilidad y tecnologías de colaboración social.

  AgileApps Live es una plataforma multitenant Cloud nativa. Puede ser desplegado en una arquitectura de dos capas pero con la restricción de que debe usar el servidor de aplicaciones Tomcat® y la base de datos MySQL®. También es posible configurarse para alta disponibilidad.

- **WSO2:** Desarrolla aplicaciones de software open source enfocadas en proveer una arquitectura orientada a servicios (SOA) para desarrolladores profesionales. Cuenta con aproximadamente 85 empleados distribuidos en sus sedes de USA, UK y Sri Lanka, siendo California (USA) su sede principal.

  Todo su software es open source y liberado bajo la licencia Apache 2.0. Su oferta está basada en el soporte, entrenamiento y servicios de consultoría. Su producto es completamente nativo Cloud y se publicitan como el único PaaS Open Source que existe actualmente para despliegues privados, públicos e híbridos.

  Su oferta Cloud está enfocada en el producto WSO2 App Factory el cual es una plataforma multi-tenant, elástica y con capacidad de autoaprovisionamiento que permite crear, correr y gestionar aplicaciones empresariales. Proporciona también la posibilidad de consumir apps y APIS a través de un market.

- **Heroku:** Fue creada inicialmente con el objetivo de soportar solamente Ruby pero posteriormente se ha extendido el soporte a Java, Node.js, Scala, Clojure, Python y Php. En 2010 Salesforce compra la empresa dejándola como una subsidiaria de la misma. En 2011 introdujeron Heroku para Facebook y actualmente soporta Cloudant, Membase, MongoDB y Redis, además de la norma PostgreSQL, tanto como parte de la plataforma y como un servicio independiente.

  Heroku se constituye como una plataforma cloud que permite el despliegue de aplicaciones desarrolladas en distintos lenguajes (Clojure, Java, Node.js, Play, Python, Ruby, Scala) y que sus aplicaciones se despliegan sobre componentes que se ejecutan en contenedores a los que llama Dynos.

  Heroku permite reescalar el número de dynos de cada tipo de forma sencilla aunque es el usuario el que debe hacerlo. Lo que sí permite es apagar de forma automática determinados dynos cuando llevan un tiempo sin recibir trabajo y de encenderlos cuando les llegan nuevas tareas o peticiones.

  Opcionalmente, heroku permite instalar determinados add-ons a elegir entre una larga lista (envío de SMS, almacenamiento clave-valor, integración con herramientas y sistemas de almacenamiento externos, etc.).

SaaS

En cuanto a las aplicaciones, nos podemos encontrar con el desarrollo de nuevas aplicaciones o con aplicaciones ya desarrolladas.

En cuanto a las aplicaciones ya desarrolladas, es importante partir de una arquitectura que cumpla los parámetros de la arquitectura propuesta, para a continuación analizar en cada caso cual es la mejor forma de migrarla a la nube. Este análisis es recomendable realizarlo siguiendo alguna de las metodologías que existen en el mercado, las cuales partiendo del hecho que no todas las

aplicaciones son candidatas a ser migradas, deben indicarnos qué modelo de transformación es el más adecuado en cada caso, siguiendo aspectos técnicos, económicos, de negocio o complementándolos con factores subjetivos de los propios usuarios.

Próximos pasos

Como resultado de dicho análisis, si se tomara la decisión de migrar aplicaciones, tendremos que realizar una de las siguientes opciones:

- Portar: llevar sobre el IaaS la aplicación sin realizar cambio alguno, de tal forma que pueda adquirir capacidades cloud del IaaS.
- Refactorizar: llevar sobre el PaaS realizando aquellas adaptaciones que el PaaS seleccionado requiera, adquiriendo las capacidades Cloud del PaaS.
- Revisar: cambiar parte del código con el objetivo de lograr potenciar alguna característica Cloud que consideremos interesante para la aplicación.
- Reconstruir: en aquellas aplicaciones que se consideren necesarias pero que no permita ninguna de las opciones anteriores.

A la hora de desarrollar una nueva aplicación, tendremos la opción de usar alguna de las plataformas de desarrollo que proporciona el PaaS, aunque muchas permiten en desarrollo sobre frameworks de desarrollo tradicionales, como el propuesto en la arquitectura, lo cual nos puede simplificar la adopción puesto que estaremos "desarrollando en Cloud" de forma transparente para nuestros equipos de desarrollo.

Recomendaciones en el desarrollo de aplicaciones Cloud

A la hora de realizar estos desarrollos, es importante que tengamos en cuenta las siguientes recomendaciones:

- **Uso de infraestructura elástica**
  - No basta con que la aplicación se ejecute correctamente en un único nodo de servicio; la aplicación debe soportar su ejecución en modo Cluster. En este sentido, la aplicación debe soportar que distintas peticiones de la misma sesión HTTP sean atendidas por distintos nodos del cluster, que se están ejecutando en Máquinas Virtuales Java y Máquinas Virtuales de Sistema Operativo diferentes. Debe ponerse especial cuidado en el tratamiento de las siguientes situaciones:
    - Si la aplicación realiza escrituras/lecturas en recursos en disco que tienen un ciclo de vida superior y gran petición, no se deben realizar en el disco local de los servidores de aplicación. Cualquier operación de lectura-escritura en recursos distribuidos en red debe tener en cuenta la naturaleza concurrente y distribuida de la misma.
    - Debe tenerse en cuenta que los distintos nodos que atienden peticiones dentro de una misma sesión no comparten la memoria RAM. No debemos hacer uso de una caché acoplada dentro de la misma aplicación. Debería de hacer uso de un servidor externo de caché. En cualquier caso los mecanismos de caché utilizados deben soportar su distribución en cluster. La aplicación debería de ser "stateless", y no asumir que distintas peticiones van a tener objetos en memoria generados por peticiones anteriores.
    - En línea con lo anterior, idealmente, la aplicación debería de ser "stateless", es decir, no hacer uso de sesiones HTTP o hacerlo únicamente para garantizar el "Failover de Login", y consecuentemente no guardar información en ellas que no sea necesaria para esto último. Dicha información (la almacenada en la sesión http) debe ser

minimizada y siempre debe ser serializable. Se debe comprobar que los mecanismos de transmisión en cluster de la sesión HTTP funcionan correctamente en escenarios de failover (Cuando cae un nodo del cluster, la sesión se ha replicado previamente en el resto de nodos, de manera que los usuarios logados en el nodo caído no necesitan logar de nuevo, si no se ha producido el timeout de sus sesiones.

- **Aplicaciones multitenant**
  - Se debe evitar el uso de Servicios de Tipo EJB (Statefull o Stateless) prefiriendo servicios tipo REST. Para aplicaciones multitenant que requieran seguridad intra-tenant es conveniente verificar la ejecución de la aplicación en modo Máquina Virtual Java seguro, y en el caso de otros entornos no JVM con una configuración de seguridad adecuada para varios tenant compartiendo el sistema operativo.
  - Es conveniente que todos los aspectos "dependientes de tenant" de la configuración de la aplicación sean externalizados a ficheros de configuración tipo texto.
  - Si vamos a tener algún comportamiento en la lógica de la aplicación dependiente de tenant o de contexto de ejecución de una instancia de la misma, es conveniente implementarla utilizando el mecanismo de inyección de dependencias contra una interfaz bien definida. Por ejemplo, si cada tenant o contexto puede presentar variaciones en lo que hace al cálculo de impuestos, por ejemplo dependiendo de su país, definiremos una interfaz de clase para el cálculo de impuestos, y generaremos una implementación de ese interfaz para cada región.

- **Integración con otros módulos externos a la Cloud**
  - Si esta es una aplicación que requiere una integración con dispositivos anclados en los PCs de los clientes, y por consiguiente, requiere que se ejecute un cliente pesado, habrá que revisar el modelo de distribución de la aplicación, y si resulta conveniente la conversión de estos dispositivos.

- **Uso de bases de datos**
  - Es cierto que determinadas base de datos tradicionales, puede tener ciertos problemas para ser usadas en Cloud, especialmente de licenciamiento. Para ello, hay proveedores de IaaS que ofrecen el servicio integrado, de tal forma que se puede contratar la provisión de la base de datos, incluyendo la licencia. Por ello se recomienda revisar que el proveedor de infraestructura y de plataforma, soportan todas las tecnologías que se desean usar, antes de proceder a su contratación.

4.1.1 Herramientas

Así mismo, dentro de las tres capas del cloud, se pueden encontrar diversas herramientas que nos van a ayudar desde la gestión de la infraestructura, a desarrollar una aplicación, o incluso a desplegar y mantener dicha aplicación. De todas ellas, vamos a destacar tres que consideramos que son especialmente importantes de cara a poder mantener y explotar la arquitectura propuesta en las universidades, y que por tanto se deberían tener en cuenta a la hora de seleccionar los proveedores:

4.1.1.1 Portal de Provisión

Por medio de una interfaz permite gestionar las instancias del sistema de una forma sencilla e intuitiva. El portal además de gestionar los recursos disponibles y ofrecidos por el IaaS, debe permitir tanto lanzar/parar instancias como gestionar de forma ordenada los ficheros de configuración de los despliegues, permitiendo la gestión de los ficheros de despliegue como si de un proyecto software se tratara.

Esta funcionalidad es ofrecida por todos los proveedores de IaaS recomendados en este documento, pero cabe destacar las funcionalidades adicionales que aportan los PaaS también incluidos en el documento, y que normalmente están enfocadas hacia la gestión de las aplicaciones.

4.1.1.2 Gestor de Despliegue

En algunos casos se ofrece incluido dentro del portal de provisión, y en otros se ofrece como una herramienta adicional, las cuales suelen ofrecer un mayor número de capacidades. Se encarga de la orquestación tanto del despliegue de las instancias, como de la configuración de la aplicación y de la comunicación e integración con la capa del IaaS.

Esta pieza asegura el despliegue coordinado dentro del Sistema Operativo incluyendo la gestión de las dependencias tecnológicas, y una vez el software está desplegado, procede a realizar las configuraciones pertinentes para dejar el software operativo y listo para su uso.

Cabe prestar especial atención a las licencias de las diferentes tecnologías, puesto que no todos los gestores de despliegue lo permiten. Debemos revisarlo con nuestro proveedor del soporte antes de contratar los servicios.

4.1.1.3 Monitorización

El servicio de monitorización nos permite detectar y prever posibles problemas dentro de la plataforma, utilizando una arquitectura de sensores que monitorizan desde dentro del Sistema Operativo anfitrión y comunica a la infraestructura de monitorización el estado del sistema y de procesos específicos que son de especial interés para el software administrado.

Este elemento ofrece un valor añadido proveyendo información para que en el caso necesario pueda tomar decisiones tales como la presentación de alarmas, comunicación con sistemas externos o la asistencia a la toma de decisiones sobre la infraestructura por los operarios del sistema.

Además en aplicaciones multitenant, nos permite realizar una gestión a nivel de tenant, controlando todo lo que ocurre en nuestros entornos, de cara tanto a la facturación como a la gestión por entiendad (universidad, por ejemplo) o incluso usuario perteneciente a cada una de ellas.

4.2 Qué nos puede proporcionar el Cloud

Cuando migramos al Cloud una aplicación desarrollada bajo una arquitectura tradicional, es posible que no nos permita utilizar alguna de las capacidades Cloud, especialmente el multitenant. Si esto ocurriera, existen en el mercado metodologías de migración de aplicaciones, que nos podrían indicar si la adopción de dicha aplicación al Cloud es posible y además cual es el valor que nos aportaría.

Estas herramientas son de gran valor puesto que nos prevén los resultados finales, pudiendo centrar nuestros esfuerzos en migrar aquellas aplicaciones que nos vayan a aportar un mayor valor.

4.2.1 Elasticidad

El reparto elástico de recursos de computación es un aspecto clave del Cloud Computing. No deberíamos hablar de un sistema Cloud si este no dispone de elasticidad, aunque dicho sistema se encuentre hospedado y se pueda consumir con un servicio.

Podremos decir que la computación es elástica, si sus recursos (CPU, memoria, canales, threads y pools de conexiones, etc.) son asignados a una instancia de una aplicación que los requiera y que puedan ser desasignados en cuanto baje dicho requerimiento para pasar a ser asignados a otra instancia o simplemente dejar de ser consumidos.

En aquellos entornos elásticos que son multitenant (una instancia está asociada a un tenant), cada instancia de una aplicación va a usar únicamente los recursos que necesite, teniendo disponibilidad de los mismos siempre dentro de los límites establecidos por las políticas de consumo, y permitiendo que escale en función de sus necesidades.

Las principales diferencias que vamos a encontrar entre entornos elásticos (Cloud) y entornos que siguen el modelo de servicios dedicados (hosted), a nivel de usuario son:

- Entorno con una mayor agilidad (destacando en picos de la demanda no programados).
- Elimina costes que se producen en entornos hosted por la sobredimensión de los entornos para dar respuesta a los picos.
- Permite la facturación acorde con los niveles de actividad de la aplicación.
- Gran flexibilidad en la regulación del auto-servicio del rendimiento y las políticas de costes, ya que las políticas pueden ser relativamente afinadas al detalle y los cambios son efectivos en tiempo real.

En cuanto a las ventajas que aporta la elasticidad a nivel del proveedor de los servicios:

- Un mayor uso de los recursos físicos.
- A pesar que disminuye la necesidad de adquirir recursos físicos.
- Posibilidad de facturar por servicio a nivel de tenant.
- Mayor satisfacción del cliente, al aumentar la capacidad de ofrecer servicios y su disponibilidad.
- Capacidad para ofrecer más elecciones y más parámetros controlados en los SLA, ofreciendo una facturación ajustada al consumo.

4.2.2 Escalabilidad

Podemos definir la propiedad de escalabilidad de un sistema Cloud, como la capacidad o característica que tiene el sistema para incrementar o reducir la cantidad de recursos destinados a dar un determinado servicio, en base a la demanda – o una previsión de esta demanda - que sufre el servicio. Un requisito básico es que el sistema debe ser capaz de proveer esta propiedad de forma automática y parametrizable.

Podemos completar la definición de la escalabilidad de un servicio en base a distintos criterios:

- Escalabilidad Vertical: La instancia de la imagen solo puede escalar verticalmente, apropiado para instancias que contienen servicios que no escalan horizontalmente.
- Escalabilidad Horizontal: La imagen se instancia múltiples veces en función de la necesidad del sistema y la carga actual.
- Escalabilidad Híbrida: Modelo en el que se puede optar por ambas opciones dependiendo de las necesidades del proyecto. Por ejemplo partir de una instancia pequeña y cuando llegue a una instancia grande, escalar horizontalmente.

Por ello es importante definir qué tipo de servicio nos interesa y contratar una combinación de proveedores Cloud (IaaS+PaaS) que nos permita utilizar el tipo de escalabilidad que se precise para cada una de nuestras aplicaciones.

4.2.3 Multitenant

Para entornos con varios usuarios (departamentos, investigadores, etc.) el multitenant puede ser una característica de gran utilidad por los ahorros que puede aportar. Permite la ejecución de tenants de diferentes usuarios sobre un mismo servidor o máquina virtual, logrando así un uso más óptimo del número de servidores precisos.

Esto es posible gracias al aislamiento lógico entre los distintos usuarios de las aplicaciones desplegadas, incluyendo distintas aplicaciones o múltiples instancias de una misma aplicación en un único runtime. Además de los ahorros de infraestructura, también proporciona ahorros en el mantenimiento y soporte de las aplicaciones.

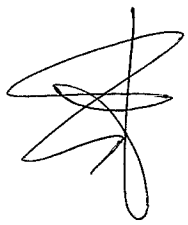
Por otro lado, nos proporciona nuevas herramientas de gestión permitiendo la trazabilidad del uso de los recursos a nivel de tenant, llegando a identificar el coste realizado por recurso. También permite operaciones como "multitenant anidada" y control de versiones para operaciones especiales de los distintos departamentos o unidades funcionales.

4.2.4 Alta disponibilidad

Para aquellas aplicaciones que no podemos prescindir de ellas durante algunos periodos de tiempo por cortos que estos sean, es importante provisionarlas en modo de Alta disponibilidad, que nos garantice la mayor disponibilidad posible. Para ello, el cloud proporciona modelos que permiten optimizar el uso de los recursos.

Es importante que dicho servicio se ofrezca no sólo a nivel de aplicación, sin en todos los servicios de la plataforma. Así mismo, nos debe permitir diversas configuraciones de despliegue en función de nuestras necesidades. Estas configuraciones pueden ser tanto en formato stand-alone, como en configuraciones de activo / pasivo y activo / activo, dependiendo de la carga que deba sufrir el sistema y los niveles de servicio requeridos.

Centrándonos en la alta disponibilidad de la aplicación administrada, los hipervisores suelen ofrecer servicios a nivel de plataforma, como pueda ser la definición de grupos de disponibilidad dentro de clúster de servidores virtualizados. Aquí el PaaS nos puede aportar un valor añadido, puesto que para mismas instancia de las aplicaciones gestionadas por el PaaS se puede pedir al gestor del hipervisor que estas no estén en las mismas máquinas físicas. De igual forma, para aplicaciones donde el tiempo de comunicación entre instancias es altamente crítico se pueden usar las características del hipervisor relacionadas con la cercanía lógica de instancias y la reducción de latencias.



Fresanta Daveiech Roda

22. Nov. 2017

ANEXO ESTIMACIÓN DE COSTES

CLASIFICACION DE LAS TAREAS

En JIRA se han establecido tres niveles de prioridad, estableciéndose la siguiente tabla:

PRIORIDAD	Severidad		
FECHA	1	2	3
1	[illegible]	[illegible]	[illegible]
2	[illegible]	Urgente	Urgente
3	[illegible]	Normal	Normal

donde,

el parámetro correspondiente a la **FECHA** determina el periodo en el que la aplicación requiere un uso masivo. Por ejemplo: para la aplicación de automatrícula será en los meses de julio, agosto y septiembre.

En cuanto a la **SEVERIDAD**:

Severidad	Definición
1	**Completa pérdida de servicio, no existen alternativas:** El usuario no puede realizar sus funciones y no existe ninguna alternativa de operación disponible.
2	**Completa pérdida de servicio del negocio pero existen alternativas de actuación**: En esta situación, el usuario no puede utilizar las aplicaciones tal y como fueron concebidas pero existe algún mecanismo de operación que puede funcionar como alternativa temporal. También se podrían encuadrar aquí aquellas incidencias que suponen, o pueden suponer, una amenaza a futuro para la estabilidad del entorno.
3	**Pérdida parcial de servicio del negocio**: Se refiere a situaciones donde el usuario (o cliente) no puede ejecutar algunas funciones específicas de la aplicación.

Para las tareas NO clasificadas como ERRORES, la prioridad quedará establecida por la proximidad de la fecha FIN en la que ésta deba estar instalada en PROD.

ESTIMACION DE COSTE DE LAS TAREAS

Para realizar la estimación se dividen los trabajos en:

- Nuevos Desarrollos y
- Mantenimientos.

a) Nuevos Desarrollos

PANTALLA DE CONSULTA	
Nivel de complejidad	**Definición**
Simple (S)	Página de consulta sin controles
Medio (M)	Página estática compleja : ■ muchos flujos ■ los campos sobrepasan los límites de la pantalla (necesidad de scroll) o Minoría de reglas de presentación complejas (<3): ■ Cálculos ■ Condicionamiento función del usuario ■ Maestro/detalle de combos...
Complejo (C)	Paginas dinámicas teniendo presencia significativa de reglas de presentación complejas (3<reglas<10): ■ Cálculos ■ Condicionamiento función del usuario ■ Maestro/detalle de combos

PANTALLA DE CREACION/MODIFICACION -FORMULARIOS	
Nivel de complejidad	**Definición**
Simple (S)	Página de modificación sin controles Minoría de reglas de presentación complejas (<3) Minoría de controles complejos: ■ Correlación entre campos ■ Validaciones...

	Pantalla sencilla de actualización de objetos
Medio (M)	Presencia significativa de reglas de presentación complejas (3<reglas<10) Presencia significativa de controles complejos (3<controles<10): ■ Correlación entre campos ■ Validaciones... Pantalla sencilla de actualización de objetos
Complejo (C)	Presencia importante de reglas de presentación complejas (10<reglas<16) Presencia importante de controles complejos (10<controles<16) Tratamiento de datos especifico complejo: ■ Modificación de datos ■ Transcodificacion

PANTALLA DE LISTADO	
Nivel de complejidad	**Definición**
Simple (S)	■ Incorpora hasta 6 campos ■ Incorpora entre 1 y 2 controles ■ No incluye reglas de presentación complejas
Medio (M)	■ Incorpora entre 7 y 10 campos. ■ Incorpora entre 3 y 6 controles. ■ Incluye entre 1 y 3 reglas de representación complejas.
Complejo (C)	■ Incorpora más de 10 campos. ■ Incorpora entre 7 y 10 controles. ■ Incluye entre 4 y 6 reglas de representación complejas.

INFORMES	
Nivel de complejidad	**Definición**
Simple (S)	■ Se modifican/incorporan entre 1 y 3 campos o ■ Se modifica/incorpora 1 regla de presentación compleja
Medio (M)	■ Se modifican/incorporan entre 4 y 6 campos o ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas

Complejo (C)	■ Se modifican/incorporan entre 7 y 10 campos o ■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas

FUNCIONES EGL	
Nivel de complejidad	**Definición**
Simple (S)	■ Menos 2 operaciones ■ Menos de 2 DAOs ■ Menos de 2 Reglas de gestión complejas (más de 1 DAO implicado)
Medio (M)	■ De 2 a 5 operaciones ■ De 2 a 5 DAOs ■ De 2 a 5 Reglas de gestión complejas
Complejo (C)	■ Más de 5 operaciones ■ Más de 5 DAOs ■ Más de 5 Reglas de gestión complejas

SERVICIO WEB DE CONSULTA	
Nivel de complejidad	**Definición**
Simple (S)	■ Se consultan hasta 6 campos ■ Sin reglas de comprobación complejas.
Medio (M)	■ Se consultan hasta 10 campos ■ Entre 1 y 3 reglas de comprobación complejas.
Complejo (C)	■ Se consultan más de 10 campos ■ Entre 4 y 6 reglas de comprobación complejas.

SERVICIO WEB DE LISTADO	
Nivel de complejidad	**Definición**
Simple (S)	■ Incorpora hasta 6 campos ■ Incorpora entre 1 y 2 controles ■ No incluye reglas de comprobación complejas
Medio (M)	■ Incorpora entre 7 y 10 campos. ■ Incorpora entre 3 y 6 controles. ■ Incluye entre 1 y 3 reglas de comprobación complejas.
Complejo (C)	■ Incorpora más de 10 campos. ■ Incorpora entre 7 y 10 controles.

	■ Incluye entre 4 y 6 reglas de comprobación complejas.

GENERACIÓN DE SCRIPTS SQL	
Nivel de complejidad	**Definición**
Simple (S)	Scripts de inserción, actualización, borrado y consulta que cumplan: ■ Número de cruces entre tablas por sentencia <3 ■ Número de instrucciones Sql < 3 ■ Número de campos en cada sentencia Sql < 10 ■ Sin tablas auxiliares ■ Sin transformaciones de datos (negocio)
Medio (M)	Scripts de inserción, actualización, borrado y consulta que cumplan: ■ Número de cruces entre tablas por sentencia entre 3 y 6 tablas ■ Número de instrucciones Sql entre 3 y 6 sentencias ■ Número de campos en cada sentencia entre 10 y 20 ■ Tablas auxiliares < 3 ■ Transformaciones de datos < 10 por sentencia (negocio)
Complejo (C)	Scripts de inserción, actualización, borrado y consulta que cumplan: ■ Número de cruces entre tablas entre 6 y 12 tablas ■ Número de instrucciones Sql entre 6 y 12 sentencias ■ Número de campos en cada sentencia entre 20 y 40 ■ Tablas auxiliares entre 3 y 10 ■ Transformaciones de datos entre 10 y 20 por sentencia (negocio)

Tabla de valoración de nuevos desarrollos

Nuevos Desarrollos			
Complejidad del elemento	**Simple**	**Medio**	**Complejo**
Pantalla de Consulta	6	10	12
Pantalla de Creación/Modificación Formularios	12	20	28
Pantalla de Listado	7	10	14
Informe	11	22	31
Función EGL	9	14	32
Integración / Servicios WEB de consulta	5	8	10
Integración / Servicios WEB de Listados	7	12	17
Generación de Scripts SQL	3	7	14

ATENCION
En este modelo se podrán contemplar excepciones puntuales cuando determinados componentes excedan la norma habitual. En estos casos se realizará una estimación "a medida" de dicho componente que será siempre validada y, si procede, aprobada por la UV. Se tendrá en cuenta también que en las tareas repetitivas o adaptaciones se aplicará un factor de valoración a la baja.

b) Mantenimientos

PANTALLA DE CONSULTA		
Complejidad del elemento	**Impacto del Mantenimiento**	**Tareas de mantenimiento asociadas**
Simple (S)	Bajo	■ Se modifican/incorporan entre 1 y 3 campos
	Medio	■ Se modifican/incorporan entre 4 y 6 campos
	Alto	■ Se modifican/incorporan entre 7 y 10 campos
Medio (M)	Bajo	■ Se modifican/incorporan entre 1 y 3 controles, ó ■ Se modifica/incorpora 1 regla de presentación compleja
	Medio	■ Se modifican/incorporan entre 4 y 6 controles, ó ■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 7 y 10 controles, ó ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas
Complejo (C)	Bajo	■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas (lleva implícito la modificación/inclusión de entre 1 y 3 campos)
	Medio	■ Se modifican/incorporan entre 4 y 6 reglas de presentación complejas (lleva implícito la modificación/inclusión de entre 4 y 6 campos)
	Alto	■ Se modifican/incorporan entre 7 y 10 reglas de presentación complejas (lleva implícito la modificación/inclusión de entre 7 y 10 campos)

PANTALLA DE CREACION/MODIFICACION - FORMULARIO		
Complejidad del elemento	**Impacto del Mantenimiento**	**Tareas de mantenimiento asociadas**
Simple (S)	Bajo	■ Se modifican/incorporan entre 1 y 3 campos
	Medio	■ Se modifican/incorporan entre 4 y 6 campos, o ■ Se modifica/incorpora 1 regla de presentación compleja
	Alto	■ Se modifican/incorporan entre 7 y 10 campos, o ■ Se modifica/incorpora entre 2 y 3 reglas de presentación complejas
Medio (M)	Bajo	■ Se modifican/incorporan entre 1 y 3 controles, ó ■ Se modifica/incorpora entre 1 y 3 reglas de presentación compleja
	Medio	■ Se modifican/incorporan entre 4 y 6 controles, ó ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 7 y 10 controles, ó ■ Se modifican/incorporan entre 7 y 10 reglas de presentación complejas
Complejo (C)	Bajo	■ Se modifican/incorporan entre 1 y 5 controles, ó ■ Se modifica/incorpora entre 1 y 5 reglas de presentación complejas
	Medio	■ Se modifican/incorporan entre 6 y 10 controles, ó ■ Se modifica/incorpora entre 6 y 10 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 11 y 16 controles, ó ■ Se modifica/incorpora entre 11 y 16 reglas de presentación complejas

PANTALLA DE LISTADO		
Complejidad del elemento	**Impacto del Mantenimiento**	**Tareas de mantenimiento asociadas**
Simple (S)	Bajo	■ Se modifican/incorporan entre 1 y 3 campos
	Medio	■ Se modifican/incorporan entre 4 y 6 campos
	Alto	■ Se modifican/incorporan entre 7 y 10 campos
Medio (M)	Bajo	■ Se modifican/incorporan entre 1 y 3 controles, ó ■ Se modifica/incorpora 1 regla de presentación compleja
	Medio	■ Se modifican/incorporan entre 4 y 6 controles, ó ■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 7 y 10 controles, ó ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas
Complejo (C)	Bajo	■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas (lleva implícito la modificación/inclusión de entre 1 y 3 campos)
	Medio	■ Se modifican/incorporan entre 4 y 6 reglas de presentación complejas (lleva implícito la modificación/inclusión de entre 4 y 6 campos)
	Alto	■ Se modifican/incorporan entre 7 y 10 reglas de presentación complejas (lleva implícito la modificación/inclusión de entre 7 y 10 campos)

INFORMES		
Complejidad del elemento	**Impacto del Mantenimiento**	**Tareas de mantenimiento asociadas**
Simple (S)	Bajo	■ Se modifican/incorporan entre 1 y 3 campos o ■ Se modifica/incorpora 1 regla de presentación compleja
	Medio	■ Se modifican/incorporan entre 4 y 6 campos o ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 7 y 10 campos o ■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas
Medio (M)	Bajo	■ Se modifican/incorporan entre 1 y 3 campos o ■ Se modifica/incorpora 1 regla de presentación compleja
	Medio	■ Se modifican/incorporan entre 4 y 6 campos o ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 7 y 10 campos o ■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas
Complejo (C)	Bajo	■ Se modifican/incorporan entre 1 y 3 campos o ■ Se modifica/incorpora 1 regla de presentación compleja
	Medio	■ Se modifican/incorporan entre 4 y 6 campos o ■ Se modifican/incorporan entre 3 y 6 reglas de presentación complejas
	Alto	■ Se modifican/incorporan entre 7 y 10 campos o ■ Se modifican/incorporan entre 1 y 3 reglas de presentación complejas

FUNCIONES EGL		
Complejidad del elemento	**Impacto del Mantenimiento**	**Tareas de mantenimiento asociadas**
Simple (S)	Bajo	■ Menos 2 operaciones ■ Menos de 2 DAOs ■ Menos de 2 Reglas de gestión complejas (más de 1 DAO implicado)
	Medio	■ De 2 a 5 operaciones ■ De 2 a 5 DAOs ■ De 2 a 5 Reglas de gestión complejas
	Alto	■ Más de 5 operaciones ■ Más de 5 DAOs ■ Más de 5 Reglas de gestión complejas
Medio (M)	Bajo	■ Menos 2 operaciones ■ Menos de 2 DAOs ■ Menos de 2 Reglas de gestión complejas (más de 1 DAO implicado)
	Medio	■ De 2 a 5 operaciones ■ De 2 a 5 DAOs ■ De 2 a 5 Reglas de gestión complejas
	Alto	■ Más de 5 operaciones ■ Más de 5 DAOs ■ Más de 5 Reglas de gestión complejas
Complejo (C)	Bajo	■ Menos 2 operaciones ■ Menos de 2 DAOs ■ Menos de 2 Reglas de gestión complejas (más de 1 DAO implicado)
	Medio	■ De 2 a 5 operaciones ■ De 2 a 5 DAOs ■ De 2 a 5 Reglas de gestión complejas
	Alto	■ Más de 5 operaciones ■ Más de 5 DAOs ■ Más de 5 Reglas de gestión complejas

SCRIPTS SQL		
Complejidad del elemento	**Impacto del Mantenimiento**	**Tareas de mantenimiento asociadas**
Simple (S)	Bajo	■ Se modifican/incorporan 1 cruce por sentencia ■ Se modifica/incorpora entre 1 instrucción ■ Se modifica/incorpora entre 1 y 3 campos
	Medio	■ Se modifican/incorporan 1 o 2 cruces por sentencia ■ Se modifica/incorpora entre 1 o 2 instrucciones ■ Se modifica/incorpora entre 3 y 6 campos
	Alto	■ Se modifican/incorporan hasta 3 cruces por sentencia ■ Se modifica/incorpora hasta 3 instrucciones ■ Se modifica/incorpora entre 6 y 10 campos
Medio (M)	Bajo	■ Se modifican/incorporan 1 cruce por sentencia ■ Se modifica/incorpora entre 1 instrucción ■ Se modifica/incorpora entre 1 y 3 campos ■ Se modifica/incorpora 1 tabla auxiliar ■ Se modifica/incorpora 1 hasta 3 transformaciones de datos
	Medio	■ Se modifican/incorporan 1 o 2 cruces por sentencia ■ Se modifica/incorpora entre 1 o 2 instrucciones ■ Se modifica/incorpora entre 3 y 6 campos ■ Se modifica/incorpora hasta 2 tablas auxiliares ■ Se modifica/incorpora 3 hasta 6 transformaciones de datos
	Alto	■ Se modifican/incorporan hasta 3 cruces por sentencia ■ Se modifica/incorpora hasta 3 instrucciones ■ Se modifica/incorpora entre 6 y 10 campos ■ Se modifica/incorpora hasta 3 tablas auxiliares ■ Se modifica/incorpora 6 hasta 10 transformaciones de datos
Complejo (C)	Bajo	■ Se modifican/incorporan 1 cruce por sentencia ■ Se modifica/incorpora entre 1 instrucción ■ Se modifica/incorpora entre 1 y 3 campos ■ Se modifica/incorpora 1 tabla auxiliar ■ Se modifica/incorpora 1 hasta 3 transformaciones de datos
	Medio	■ Se modifican/incorporan 1 o 2 cruces por sentencia ■ Se modifica/incorpora entre 1 o 2 instrucciones ■ Se modifica/incorpora entre 3 y 6 campos ■ Se modifica/incorpora hasta 2 tablas auxiliares ■ Se modifica/incorpora 3 hasta 6 transformaciones de datos
	Alto	■ Se modifican/incorporan hasta 3 cruces por sentencia ■ Se modifica/incorpora hasta 3 instrucciones ■ Se modifica/incorpora entre 6 y 10 campos ■ Se modifica/incorpora hasta 3 tablas auxiliares ■ Se modifica/incorpora 6 hasta 10 transformaciones de datos

Tabla de valoración de mantenimientos

La siguiente tabla muestra los esfuerzos en horas PTU, esto es, en horas de programación y test unitarios asociados a cada tipo de modificación en función de la complejidad del elemento a mantener y del impacto del mantenimiento sobre el elemento:

Complejidad del elemento	**Sencillo**			**Medio**			**Complejo**		
Impacto del mantenimiento	B	M	A	B	M	A	B	M	A
Pantalla de consulta	3	4,5	6	5	7	11	6	8	14
Pantalla de Creación/Modificación formularios	6	10	14	10	17	24	16	25	36
Pantalla de listado	3,5	5	6,5	5	7	11	7	10	15
Informe	4	6	11	9	14	20	15	23	30
Función EGL	5	7,5	13	7,5	13	17	18	16	36
Integración / servicio WEB consultas	2,5	4	6	4,5	7	10	6	9,5	12
Integración / servicios WEB listados	3,5	5,5	7	6,5	10	13	9	15	20
Generación de Scripts SQL	1	3	5	2	6	12	4	8	16

De esta forma, se consigue obtener una estimación del esfuerzo de mantenimiento, de cada uno de los elementos atómicos, esto es, las pantallas de consulta, de creación/modificación, informes y funciones host.

A los valores establecidos en la tabla anterior se aplicaran una serie de factores correctores en función de la documentación existente y de la entidad responsable de su desarrollo, así se identifican las siguientes circunstancias:

- Se aplicará una **reducción de un 30 %** sobre el tiempo de trabajo estipulado, cuando el elemento haya sido creado por la empresa dentro del presente contrato.
- Se aplicará una **reducción de un 15 %** sobre el tiempo de trabajo estipulado, cuando el elemento haya sido creado por la empresa fuera del presente contrato.
- Se aplicará un **incremento de un 20 %** sobre el tiempo estipulado cuando el elemento haya sido creado por una empresa/entidad diferente del adjudicatario y no exista documentación del elemento a mantener.

ATENCION

En este modelo se podrán contemplar excepciones puntuales cuando determinados componentes excedan la norma habitual. En estos casos se realizará una estimación "a medida" de dicho componente que será siempre validada y, si procede, aprobada por la UV. Se tendrá en cuenta también que en las tareas repetitivas o adaptaciones se aplicará un factor de valoración a la baja.

22 · Nov · 2017