**ESCUELA SUPERIOR DE INGENIERÍA**

**INGENIERO EN INFORMÁTICA**

**jUCAbox**

**jukebox Social**

AUTOR(A): Sergio Ruiz Piulestán

30 de septiembre de 2017

**ESCUELA SUPERIOR DE INGENIERÍA**

**INGENIERO EN INFORMÁTICA**

**jUCAbox**

**jukebox Social**

AUTOR(A): Sergio Ruiz Piulestán DIRECTOR(A): Iván Ruiz Rube

Cádiz, 30 de septiembre de 2017

***Agradecimientos***

Me gustaría agradecer y dedicar estas líneas a mi pareja, familia, amigos y en especial a mis tutores de proyecto, por su gran ayuda y tiempo.

A todos ellos por el apoyo ofrecido en los momentos difíciles.

# Índice general

### [Prolegómeno](#_bookmark0)

#### [Introducción](#_bookmark1) 5

* 1. [Motivación](#_bookmark2) 5
  2. [Alcance](#_bookmark3) 5
  3. [Glosario de Términos](#_bookmark4) 5
  4. [Organización del documento](#_bookmark5) 5

#### [Planiftcación](#_bookmark6) 7

* 1. [Metodología de desarrollo](#_bookmark7) 7
  2. [Planificación del proyecto](#_bookmark8) 7
  3. [Organización](#_bookmark9) 7
  4. [Costes](#_bookmark10) 7
  5. [Riesgos](#_bookmark11) 8
  6. [Aseguramiento de calidad](#_bookmark12) 8

### [Desarrollo](#_bookmark13)

#### [Requisitos del Sistema](#_bookmark14)

* 1. [Situación actual](#_bookmark15) 13
     1. [Procesos de Negocio](#_bookmark16) 13
     2. [Entorno Tecnológico](#_bookmark17) 13
     3. [Fortalezas y Debilidades](#_bookmark18) 13
  2. [Necesidades de Negocio](#_bookmark19) 13
     1. [Objetivos de Negocio](#_bookmark20) 14
     2. [Procesos de Negocio](#_bookmark21) 14
  3. [Objetivos del Sistema](#_bookmark22) 14
  4. [Catálogo de Requisitos](#_bookmark23) 14
     1. [Requisitos funcionales](#_bookmark24) 14
     2. [Requisitos no funcionales](#_bookmark25) 14
     3. [Reglas de negocio](#_bookmark26) 14
     4. [Requisitos de información](#_bookmark27) 14
  5. [Alternativas de Solución](#_bookmark28) 14
  6. [Solución Propuesta](#_bookmark29) 15

1. [Análisis del Sistema](#_bookmark30)
   1. [Modelo Conceptual](#_bookmark31) 17
   2. [Modelo de Casos de Uso](#_bookmark32) 17
      1. [Actores](#_bookmark33) 17
   3. [Modelo de Comportamiento](#_bookmark34) 17
   4. [Modelo de Interfaz de Usuario](#_bookmark35) 17
2. [Diseño del Sistema](#_bookmark36)
   1. [Arquitectura del Sistema](#_bookmark37) 19
      1. [Arquitectura Física](#_bookmark38) 19
      2. [Arquitectura Lógica](#_bookmark39) 19
   2. [Parametrización del software base](#_bookmark40) 20
   3. [Diseño Físico de Datos](#_bookmark41) 20
   4. [Diseño detallado de Componentes](#_bookmark42) 20
   5. [Diseño detallado de la Interfaz de Usuario](#_bookmark43) 20
3. [Construcción del Sistema](#_bookmark44)
   1. [Entorno de Construcción](#_bookmark45) 21
   2. [Código Fuente](#_bookmark46) 21
   3. [Scripts de Base de datos](#_bookmark47) 21
4. [Pruebas del Sistema](#_bookmark48)
   1. [Estrategia](#_bookmark49) 23
   2. [Entorno de Pruebas](#_bookmark50) 23
   3. [Roles](#_bookmark51) 23
   4. [Niveles de Pruebas](#_bookmark52) 23
      1. [Pruebas Unitarias](#_bookmark53) 23
      2. [Pruebas de Integración](#_bookmark54) 24
      3. [Pruebas de Sistema](#_bookmark55) 24
      4. [Pruebas de Aceptación](#_bookmark56) 24
5. [Epílogo](#_bookmark57)
6. [Manual de implantación y explotación](#_bookmark58)
   1. [Introducción](#_bookmark59) 29
   2. [Requisitos previos](#_bookmark60) 29
   3. [Inventario de componentes](#_bookmark61) 29
   4. [Procedimientos de instalación](#_bookmark62) 29
   5. [Pruebas de implantación](#_bookmark63) 29
   6. [Procedimientos de operación y nivel de servicio](#_bookmark64) 29
7. [Manual de usuario](#_bookmark65)
   1. [Introducción](#_bookmark66) 31
   2. [Instalación](#_bookmark67) 31
   3. [Uso del sistema](#_bookmark68) 31

#### [Manual del desarrollador](#_bookmark69)

* 1. [Introducción](#_bookmark70) 33
  2. [Preparación del entorno de trabajo](#_bookmark71) 33
  3. [Consideraciones generales sobre el desarrollo](#_bookmark72) 33
  4. [Instrucciones para construcción y despliegue](#_bookmark73) 33

#### [Conclusiones](#_bookmark74)

* 1. [Objetivos alcanzados](#_bookmark75) 35
  2. [Lecciones aprendidas](#_bookmark76) 35
  3. [Trabajo futuro](#_bookmark77) 35

#### [Bibliografía](#_bookmark78) 37

#### [Información sobre Licencia](#_bookmark81)

# Índice de figuras

Figura 2.1. Diagramas de Gantt

# Índice de tablas

**Parte I**

**Prolegómeno**

**Capítulo 1**

**Introducción**

El presente documento detalla el proceso que se ha seguido para la creación de la herramienta jUCAbox.

En la actualidad, muchos locales tienen sistemas de reproducción musical. Cuando un cliente quiere solicitar alguna canción especial, tiene que ir al encargado de la música y decirle el nombre de la canción y el artista que desea. Conllevaría la interrupción de la labor desempeñada por el encargado de la música, así como la dificultad por el ruido y el ambiente en el local de poder transmitir que canción desea.

Por otro lado, está creciendo la necesidad de solicitar unas canciones antes de realizer un evento social, como una boda, una communion, o simplemente una fiesta entre amigos. Por lo que el organizador, debería pedir, mediante correos, mensajes y/o personalmente, las canciones que le gustaría que sonaran en el evento.

La herramienta **jUCAbox** es un Jukebox social, una aplicación pensada para pubs, discotecas y eventos sociales.

Con **jUCAbox**, se pretende paliar estos problemas. El organizador del evento, o el encargado del sistema musical del local, registrarían su evento/local en la herramienta y automaticamente esta disponible para que los usuarios puedan solicitar/registrar las canciones deseadas. Evitando el tráfico de información que pueda ser susceptible de perderse.

Con **jUCAbox** se podrá interactuar con el DJ en tiempo real para solicitar canciones, intercambiar mensajes con otros usuarios de la herramienta, e incluso, el propio local podrá añadir promociones para favorecer a que el cliente use la herramienta.

A su vez, se podrán enviar canciones a una lista de reproducción futura, no teniendo que ser en el momento del evento, para su posterior validación. Este listado de canciones, son todas las disponibles en el ámbito de Spotify, ya que la herramienta se conecta a través de un API proporcionado por la misma.

**jUCAbox** se compone de 3 partes:

* **Búsqueda de canciones**: El usuario podrá buscar todas las canciones disponibles en Spotify, para luego enviarla a un evento de todos los registrados en la herramienta. Esta funcionalidad no requiere que el usuario esté registrado en jUCAbox. Si quiere guarder información de que canciones ha enviado, o guarder lugares y artistas favoritos, entonces si deberá registrarse.
* **Búsqueda de lugares**: El usuario podrá buscar todos los lugares dados de alta en la herramienta, para poder ver una descripción más detallada del mismo y canciones enviadas, top de canciones entre rangos de fechas. El usuario podrá crear lugares y administrarlos, para controlar las canciones que le envien y postear mensajes a los usuarios que tengan al lugar como favorito.
* **Usuario**: Tendrá un apartado para ver los datos referente a su usuario. Log de actvidad, lugares y artistas favoritos, y asi como los amigos que tenga agregados para empezar un chat si están en linea.

## **Motivación**

La motivación para realizar esta aplicación han sido diversas.

Conocer nuevos frameworks de desarrollo web, nuevos modelos de datos no convencionales, como son los modelos de base de datos no relaciones, y poder explotar toda esa información mediante API Rest full.

El sistema utilizado se define como SPA, Simple Page Application, por el cual la navegación es muy fluida y la actualizacion de los datos ese n tiempo real. Por el cual es muy importante para el contexto en el que se basa la herramienta. Todo sistema con interacción social requiere de un tiempo de respuesta muy corto, uno de los puntos Fuertes de los frameworks utilizados.

La principal motivación de este proyecto es poner en práctica los conocimientos adquiridos a lo largo de las asignaturas de la Titulación de Ingeniería en Informática así como conocer nuevos lenguajes de programación que nos permita ampliar nuestros conocimientos dentro del abanico de posibilidades que nos ofrece la informática hoy en día y aprender a desenvolverse en situaciones desconocidas hasta el momento por uno mismo.

## **Alcance**

El objetivo principal de este proyecto es que el usuario pueda solicitar una canción de manera autónoma y que el administrador del local/evento pueda llevar un control de las canciones enviadas y a su vez poder utilizarlo como portal de promociones hacia los usuarios registrados, pudiendo promover el uso de la aplicación.

De modo que se obtendrán estadísticas de las canciones más solicitadas, para poder adaptar las canciones futuras a los gustos de las personas que frecuenten el local.

## **Glosario de Términos**

Esta sección debe contener una lista ordenada alfabéticamente de los principales términos, acró- nimos y abreviaturas específicos del dominio del problema, especialmente de los que se considere que su significado deba ser aclarado. Cada término, acrónimo o abreviatura deberá acompañarse de su definición.

A continuación mostramos algunas definiciones, acrónimos y abreviaturas que irán apareciendo a lo largo de la memoria, para así facilitar la comprensión de algunos conceptos.

**Angular**: Framework javascript para desarrollo de páginas Web basado en el modelo MVC. La primera version de este Framework, fue llamada AngularJS, fue creada por google y se desarrollaba utilizando javascript. El 15 de Septiembre de 2016, aparecería la version 2.0, no siendo un simple evolutivo de la primera version, si no que dejaba de lado javascript, para utilizer typescript, como lenguaje base para su implementación. Los sistemas desarrollados por Angular se denominan SPA, en el cual se pasa la lógica de negocio al cliente en vez de al servidor. Mejorando tiempos de espera y latencia con la aplicación.

**TypScript**: Es un lenguaje de programación de código abierto, desarrollado por Microsoft. Se basa en javascript, añadiendole tipado a las variables y clases. Es un lenguaje, en el cual su salida compilada es lenguaje javascript.

**MongoDB**: Es un sistema de gestión de base de datos no relacionales basadas en JSON, orientado a documentos. Es una base de datos de código abierto nacida en 2007.

**Express**: Es una infraestructura para Node.js en el cual proporciana una serie de caracteristicas para conectar el servidor Node con la base de datos. Se suele utilizar con MongoDB, mendiante mongoose, una librería javascript para facilitar la integración.

**Node**.**js**: Es un intérprete de javascript en el lado del servidor basado en eventos basado en el motor V8 desarrollado por Google. Con lo cual permite al usuario desarrollar en el servidor con el mismo lenguaje de programación del cliente, siendo su curva de aprendizaje minima.

**QlikView**: Es una plataforma de Business Intelligence que sirve, para mostrar y analizar datos de manera inmediata. Se utiliza como cuadro de mandos, para poder reporter datos a gran escala, y que cada usuario pueda filtrar y ordenar la información como desee.

**Git**: Repositorio de código utilizado para la gestión de versiones. Fue creado pensando en la eficiencia y confiabilidad del mantenimiento de aplicaciones, cuando estas tienen un número elevado de ficheros Fuente.

**Spotify**: Es una herramienta multiplataforma utilizada para la reproducción de música vía Streaming.

**Api Rest**: Es un sistema por el cual los programadores pueden interactuar con una aplicación y obtener y enviar datos de manera ordenada y planificada. Se basa en obtener y enviar datos mediante JSON y peticiones HTTP.

## **Organización del documento**

El documento está divido en once capítulos en los cuales se irán describiendo las distintas etapas que se han seguido a la hora de desarrollar el proyecto.

En este primer capítulo, ofrece una visión algo general de lo que este proyecto abarcará. Definiremos los límites que esperamos alcanzar en la aplicación a desarrollar al final del documento.

En el segundo de los capítulos indicaremos la metodología utilizada para el desarrollo de la herramienta, asi como la planificación y organización del proyecto. Abarcaremos los coste y riesgos que conllevarían su implantación.

En los cinco capítulos siguientes se abordarán las distintas fases que comprenden cualquier desarrollo de software bajo el modelo de desarrollo lineal secuencial, que son:

Requisitos del sistema, Análisis, Diseño, Construcción del sistema y Pruebas.

**Requisitos del sistema** Se defininen los requisitos necesarios para poder realizar el sistema, asi como las necesidades de negocio y studio de las solición propuesta

**Análisis**. Fase en la cual se estudia la información que deberá manejar el sistema así como los procesos a desarrollar sobre estos y la interconexión entre ambos.

**Diseño**. Etapa consistente en la estructuración de los datos y en el diseño de los algoritmos que tratarán dichas estructuras de información. En esta fase, se obtiene ya una representación del software previa a la implementación que podrá dar una idea de la calidad que tendrá este.

**Construcción del sistema e Implementación**. Paso en el que se transformará la representación del software generada en el paso anterior a un lenguaje entendible por la máquina.

**Pruebas**. Previa a la implantación del software se harán necesarias unas pruebas que garanticen la calidad del sistema. Esta fase se prorrogará incluso estando ya el software en funcionamiento, ya que no hay mejor prueba que un entorno real capaz de simular todas y cada una de las situaciones existentes.

En el octavo capítulo describiremos un completo manual para la implantación del sistema, donde explicamos como realizar el proceso de instalación.

En el noveno capítulo describiremos un completo manual para el usuario, mostrando de manera detallada e ilustrada como realizar las distintas funcionalidades.

En el capítulo diez detallaremos el manual del desarrollador, modelo de trabajo utilizado, consideraciones a la hora de desarrollar e indicaremos las instrucciones para el despliegue del sistema.

En el capítulo once analizaremos como ha ido el desarrollo completo del sistema, si se han cumplido las expectativas y los distintos problemas que nos hemos encontrado.

Y por ultimo describiremos el software usado para la realización de todo el proyecto, bibliografía usada y la licencia aplicada a la aplicación.

**Capítulo 2**

**Planificación**

## **Metodología de desarrollo**

La metodología utilizada para la elaboración del proyecto se basa en el Racional Unified Process (RUP), que traducido es proceso racional unificado. Este proceso de desarrollo software unido al lenguaje unificado de modelado (UML) componen la metodología habitual estándar para el análisis, implementación y documentación de sistemas orientados a objetos.

El proceso unificado racional (RUP) se utiliza como en nuestro caso para desarrollar sistemas orientados a objetos, basado originalmente en el Proceso unificado (UP) y que utiliza Lenguaje unificado de modelado (UML) para su notación.

Consta de cuatro fases:

* **Inicio**: Se plantea y determina las características del proyecto.
* **Diseño**: Donde se realiza el diseño, requisitos, análisis.
* **Implementación**: Donde se realiza la implementación.
* **Pruebas**: Periodo de pruebas para comprobar que todo funciona correctamente.

Como hemos mencionamos anteriormente, en el desarrollo de nuestro proyecto seguimos un enfoque orientado a objetos. La programación orientada a objetos es un paradigma de la ingeniería del software que basa la arquitectura de los sistemas en los objetos y sus interacciones. Estos objetos representan las entidades físicas y conceptuales del mundo real y están estrechamente relacionados con la aplicación a construir.

Basado en el desarrollo incremental, donde todas los roles o actores se involucran más en el Proyecto. Haciendo entregas parciales del software, evaluandose como partes individuales, puediendo reconstruirse partes anteriores conforme se elaboran las nuevas.

De esta forma se determinan las funcionalidades importantes y las comprobaciones son constantes a lo largo del desarrollo, permite una implementación con refinamientos sucesivos y con cada incremento se agrega nueva funcionalidad, se cubren nuevos requisitos o bien se mejora la versión previamente implementada del sistema.

## **Planifiicación del proyecto**

Estimación temporal y definición del calendario básico (hitos principales e iteraciones). Desarro- llo de la planificación detallada, utilizando un diagrama de Gantt. Los diagramas de Gantt que se vean correctamente (girados y divididos si hace falta).

Se debe incluir una comparación cuantitativa del tiempo y el esfuerzo realmente invertido frente al estimado y planificado. Estos datos pueden recogerse del sistema de gestión de tareas empleado para el seguimiento del proyecto.

Las etapas llevadas a cabo para realizar este Proyecto son las ya mencionadas: Análisis, Diseño, Implementación, Pruebas y documentación.

El detalle de las fases llevadas a cabo son:

Etapa 1 - **Análisis**:

* Tecnología a usar
* Formación
* Visión Global
* Definición alcance total

Etapa 2 - **Diseño**:

* Modelo de datos
* Fase 1 – Usuarios / roles
* Fase 2 – Buscar Canciones
* Fase 3 – Lugares
* Fase 4 – Envío y Reproducción de canciones
* Fase 5 – Mensajería
* Fase 6 – Reporting

Etapa 3 - **Implementación**:

* Fase 1 – Usuarios / roles
* Fase 2 – Buscar Canciones
* Fase 3 – Lugares
* Fase 4 – Envío y Reproducción de canciones
* Fase 5 – Mensajería
* Fase 6 – Reporting

Etapa 4 - **Pruebas**:

* Fase 1 – Usuarios / roles
* Fase 2 – Buscar Canciones
* Fase 3 – Lugares
* Fase 4 – Envío y Reproducción de canciones
* Fase 5 – Mensajería
* Fase 6 – Reporting

Etapa 5 – **Documentación**.

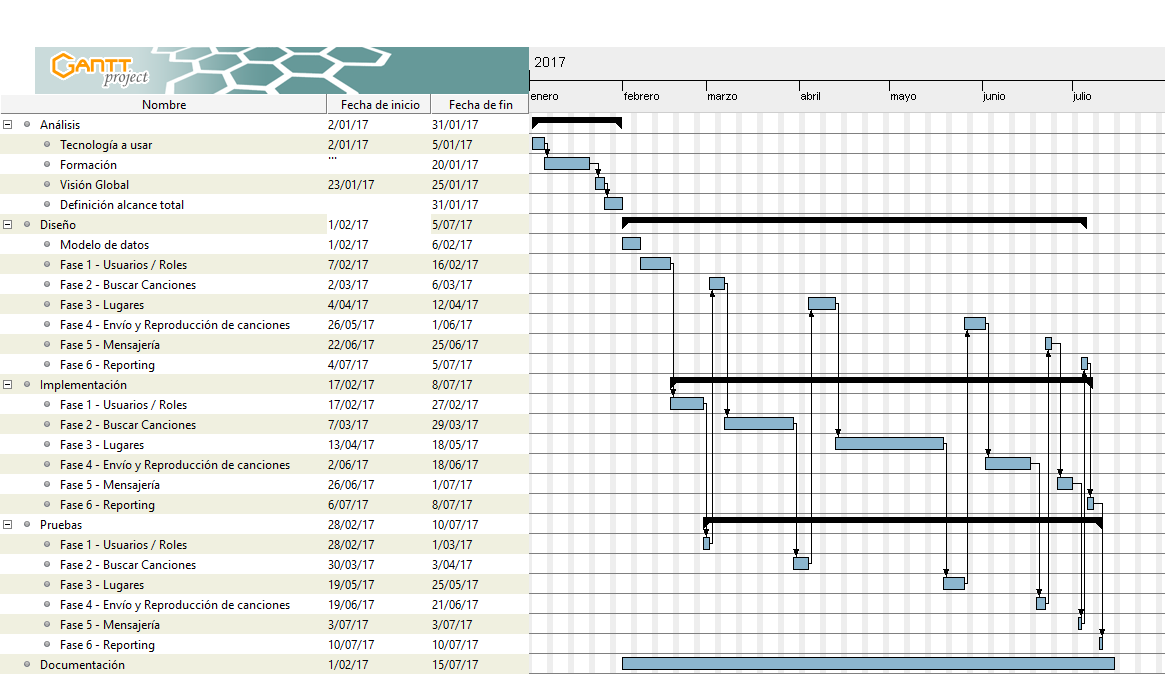


Figura 2.1. Diagramas de Gantt

La duración esperada de cada etapa se ha correspondido con la realizada real.

## **Organización**

Relación de las personas (roles) involucradas en el proyecto y de cómo se estructuran las relacio- nes entre las mismas para ejecutar el proyecto. Relación de los recursos inventariables utilizados en el proyecto: equipamiento informático (hardware y software), herramientas empleadas, etc.

Para la realización de la aplicación, han estado involucrados los distintos roles:

* Analista:
  + Encargado de obtener los requisitos funcionales necesarios para la elaboración de la herramienta
* Desarrollador:
  + Encargado de llevar a cabo los requisitos obtenidos por el analísta
* Encargado de pruebas
  + Encargado de evaluar la herramienta, tanto de fallos técnicos, como requisitos que no se hayan desarrollado de forma correcta.

Para poder desarrollar la herramient jUCAbox se han utilizado los frameworks propuestos por el fork MEAN stack.

* **MongoDB:** Sistema de base de datos no relacionales basada en JSON. Utiliza una version propia llamada BSON.
* **Express:** Framework encargado de conectar MongoDB con el servidor WEB
* **Angular2:** Framework basado en MVC utilizado para la parte Cliente de la aplicación WEB. Se utiliza para el desarrollo de páginas SPA (Single Page Application). Se desarrolla con typescript, una versión libre de javascript creada por Microsoft.
* **Node.js:** Framework utilizado como servidor WEB. Se utiliza javascript para el desarrollo de las reglas del servidor.

A su vez, se ha empleado otra sistema de gestión de base de datos, para el envío en tiempo real de los mensajes entre usuarios. **FireBase,** un sistema de dase de datos no relacionales basada en JSON. La gran diferencia que tiene con MongoDB es que se trata de un sistema en tiempo real.

A su vez, se han desarrollado una serie de informes, mediante la herramienta **QlikView**. Una herramienta capaz de transformar la información en tiempo real y que se encuentra alojada en la nube.

El hardware utilizado para el desarrollo de esta herramienta ha sido un portatil Lenovo ideapad 100 con un procesador intel i5 con 8gb Ram y disco duro de 256 GB.

## **Costes**

Estudio y presupuesto de los costes de los recursos (humanos y materiales) descritos anterior- mente, necesarios para el proyecto.

Para el cálculo de costes de personal pueden consultarse las tablas salariales de la UCA para el personal técnico de apoyo contratado laboral [[CCOO, 2010](#_bookmark79)], o bien otras más ajustadas a la realidad. El cálculo del coste del personal del proyecto debe hacerse en personas-mes, y luego hacer la correspondencia al coste monetario.

El coste final del producto, sería la suma del coste de los recursos humanos, con los recursos materiales.

Los **recursos materiales** se dividen en dos:

* Recursos software
  + Fork utilizado para el desarrollo Mean Stack. Todo lo utilizado es software libre, por lo tanto el coste de los recursos software es 0€.
* Recursos hardware
  + Para el desarrollo de la herramienta se ha utilizado un portátil Lenovo ideapad con un precio de 585€. Fuente: <https://www.pccomponentes.com/lenovo-ideapad-110-15isk-intel-core-i5-6200u-8gb-1tb-r5-m430-156>
  + Para el despliegue de la aplicación, se utilizaría un sistema en la nube basado en AWS de amazon. El precio para el alojamiento de la aplicación depende del uso de la herramienta, y el impacto social que pueda tener. Todos estos servidores son escalables, es decir, en el momento que se necesiten más recursos se pueden contratar, y automáticamente serían añadido. El modelo utilizado para el despliegue de esta herramienta sería Amazon Ec2 bajo demanda, con un almacenamiento estandar de 50TB/mes de uso de datos, y un servidor Linux t2.medium con unas especificaciones de 2 CPU virtuales y 4 GB de ram, serían 34.41€ al mes.

Los **recursos humanos** necesarios para el desarrollo serían:

* **Analista**: Sueldo medio anual de un analista funcional es de 45.000€. Siendo la hora a 23€
* **Desarrollador:** Sueldo medio anual de un desarrollador web es de 35.000€. Siendo la hora a 18€.
* **Encargado de Pruebas**:Sueldo medio anual de un especialista encargado de realizar pruebas técnicas y funcionales es de 30.000€. Siendo la hora a 15€.

Coste desarrollo de **jUCAbox:**

* Equipos necesarios para el desarrollo:
  + 3 Lenovo ideapad: 1.755€
* Recursos humanos necesarios:
  + Analista:
    - Horas destinadas al anális y documentación: 888
    - Coste total: 20.424€
  + Desarrollador:
    - Horas destinadas al anális y documentación: 808
    - Coste total: 14.544€
  + Encargado de pruebas:
    - Horas destinadas al anális y documentación: 760
    - Coste total: 11.400 €
  + Coste total: 46.368€
* Coste total: 48.123€

Coste anual demanteniemiento **jUCAbox:**

* Recursos materiales:
  + 412,92€ por alojamiento.
* Recursos humanos:
  + Desarrolladore:
    - Sería necesario un desarrollador para incidencias y mejoras a media jornada.
    - Coste total: 26.892€

## **Riesgos**

Enumeración de los riesgos del proyecto, indicando su posible impacto (efecto que la ocurrencia del citado riesgo tendría en el desarrollo del proyecto) y la probabilidad de ocurrencia. Una vez los riesgos son identificados y priorizados, hay que definir los planes necesarios para reducir los efectos del riesgo una vez se haya materializado o disminuir que este ocurra.

Principales riesgos:

* **No disponibilidad del sistema por la caida del servidor**
  + Para no sufrir dependencia de un único servidor, se utiliza el balanceo de servidores. Su estructura se compone de 3 servidores. El primer servidor es el que se encarga de gestionar la comunicación de la base de datos con el cliente Web. El Segundo hace una replica en tiempo real de los datos del primero. Y el tercero se mantiene a la espera de posibles daños.
  + Cuando en el primer servidor se produce un fallo, comprometiendo la integridad de la aplicación, el Segundo servidor se pone como maestro y el tercero ocupa el puesto que este deja, siendo el que replica la información del servidor.
  + Cuando el servidor vuelve a la normalidad, se convierte en el tercero de ellos, esperando que se produzcan posibles fallos.
* **Pérdida de datos**
  + Para ello, como se ha comentado antes, en el grupo de servidores, el Segundo servidor no estaría esperando a que el primero falle para ocupar su lugar. Serviría también de almacen de copia de seguridad a corto plazo, pudiendo recuperar la información si fuera necesario.

## **Aseguramiento de calidad**

En esta sección se incluirán las actividades y tareas relacionadas con el aseguramiento de caldad a realizar durante el desarrollo del software. Se incluirán los estándares, prácticas y normas aplicables durante el desarrollo del software.

También, deberán recogerse los diferentes tipos de revisiones, verificaciones y validaciones que se van a llevar a cabo, los criterios para la aceptación o rechazo de cada producto y los procedi- mientos para implementar acciones correctoras o preventivas.

**Parte II**

**Desarrollo**

En esta parte se debe describir el desarrollo del proyecto siguiendo la metodología empleada. Sus capítulos no deben ser una descripción exhaustiva de todos los documentos, diagramas, código fuente y, en general, entregables generados, sino más bien una explicación resumida del desarrollo, estructurada según las etapas principales del proceso de ingeniería. Deben seleccionarse aquellos diagramas, fragmentos de código y secciones de los entregables que sean más significativos para dicha explicación. La totalidad de los entregables resultado del proyecto se ubicarán en los anexos y/o en el material en CD/DVD que acompañe al proyecto.

**Capítulo 3**

**Requisitos del Sistema**

En esta sección se detalla la situación actual de la organización y las necesidades de la misma, que originan el desarrollo o mejora de un sistema informático. Luego se presentan los objetivos y el catálogo de requisitos del nuevo sistema. Finalmente se describen las diferentes alternativas tec- nológicas y el análisis de la brecha entre los requisitos planteados y la solución base seleccionada, si aplica.

## **Situación actual**

Esta sección debe contener información sobre la situación actual de la organización para la que se va a desarrollar el sistema software.

### Procesos de Negocio

Esta sección debe contener información sobre los modelos de procesos de negocio actuales, que suelen ser la base de los modelos de procesos de negocio a implantar.

### Entorno Tecnológico

Esta sección debe contener información general sobre el entorno tecnológico en la organización del cliente antes del comienzo del desarrollo del sistema software, incluyendo hardware, redes, software, etc.

### Fortalezas y Debilidades

Esta sección debe contener información sobre los aspectos positivos y negativos del negocio actual de la organización para la que se va a desarrollar el sistema software.

## **Necesidades de Negocio**

Esta sección debe contener información sobre los objetivos de negocio de clientes y usuarios, incluyendo los modelos de procesos de negocio a implantar.

### Objetivos de Negocio

Esta sección debe contener los objetivos de negocio que se esperan alcanzar cuando el sistema software a desarrollar esté en producción.

### Procesos de Negocio

Esta sección, debe contener los modelos de procesos de negocio a implantar, que normalmente son los modelos de procesos de negocio actuales con ciertas mejoras.

## **Objetivos del Sistema**

Esta sección debe contener la especificación de los objetivos o requisitos generales del sistema.

## **Catálogo de Requisitos**

Esta sección debe contener la descripción del conjunto de requisitos específicos del sistema a desarrollar para satisfacer las necesidades de negocio del cliente.

### Requisitos funcionales

Descripción completa de la funcionalidad que ofrece el sistema.

### Requisitos no funcionales

Descripción de otros requisitos (relacionados con la calidad del software) que el sistema deberá sa- tisfacer: portabilidad, seguridad, estándares de obligado cumplimiento, accesibilidad, usabilidad, etc.

### Reglas de negocio

En el desarrollo del sistema, hay que tener en cuenta las denominadas reglas de negocio, es decir, el conjunto de restricciones, normas o políticas de la organización que deben ser respetadas por el sistema, las cuales suelen ser cambiantes.

### Requisitos de información

En esta sección se describen los requisitos de gestión de información (datos) que el sistema debe gestionar.

## **Alternativas de Solución**

En esta sección, se debe ofrecer un estudio del arte de las diferentes alternativas tecnológicas que permitan satisfacer los requerimientos del sistema, para luego seleccionar (si procede) la herramienta o conjunto de herramientas que utilizaremos como base para el software a desarrollar.

## **Solución Propuesta**

Si se ha optado por utilizar un software de base, debemos identificar y medir las diferencias entre lo que proporciona este software y los requisitos definidos para el proyecto.

El resultado de este análisis permitirá identificar cuáles de éstos requisitos ya están solventa- dos total o parcialmente por el sistema base y cuales tendremos que diseñar e implementar la propuesta de solución.

**Capítulo 4**

**Análisis del Sistema**

Esta sección cubre el análisis del sistema de información a desarrollar, haciendo uso del lenguaje de modelado UML.

## **Modelo Conceptual**

A partir de los requisitos de información, se desarrollará un diagrama conceptual de clases UML, identificando las clases, atributos, relaciones, restricciones adicionales y reglas de derivación ne- cesarias.

## **Modelo de Casos de Uso**

A partir de los requisitos funcionales descritos anteriormente, se emplearan los casos de uso como mecanismo para representar las interacciones entre los actores y el sistema bajo estudio. Para cada caso de uso deberá indicarse los actores implicados, las precondiciones y postcondiciones, los pasos que conforman el escenario principal y el conjunto de posibles escenarios alternativos.

### Actores

En este apartado se describirán los diferentes roles que juegan los usuarios que interactúan con el sistema. Los actores pueden ser roles de personas físicas, sistemas externos o incluso el tiempo (eventos temporales).

## **Modelo de Comportamiento**

A partir de los casos de uso anteriores, se crea el modelo de comportamiento. Para ello, se rea- lizarán los diagramas de secuencia del sistema, donde se identificarán las operaciones o servicios del sistema. Luego, se detallará el contrato de las operaciones identificadas.

## **Modelo de Interfaz de Usuario**

En esta sección se deberá incluir un prototipo de baja fidelidad o mockup de la interfaz de usuario del sistema. Además, es preciso elaborar un diagrama de navegación, reflejando la secuencia de pantallas a las que tienen acceso los diferentes roles de usuario y la conexión entre éstas.

**Capítulo 5**

**Diseño del Sistema**

En esta sección se recoge la arquitectura general del sistema de información, la parametrización del software base (opcional), el diseño físico de datos, el diseño detallado de componentes software y el diseño detallado de la interfaz de usuario.

## **Arquitectura del Sistema**

En esta sección se define la arquitectura general del sistema de información, especificando la infra- estructura tecnológica necesaria para dar soporte al software y la estructura de los componentes que lo forman.

### Arquitectura Física

En este apartado, describimos los principales elementos hardware que forman la arquitectura física de nuestro sistema, recogiendo por un lado los componentes del entorno de producción y los componentes de cliente.

Se debe incluir un modelo de despliegue en el cual se describe cómo los elementos software son desplegados en los elementos hardware. También se incluyen las especificaciones y los requisitos del hardware (servidores, etc.), así como de los elementos software (sistemas operativos, servicios, aplicaciones, etc.) necesarios.

### Arquitectura Lógica

La arquitectura de diseño especifica la forma en que los artefactos software interactúan entre sí para lograr el comportamiento deseado en el sistema. En esta sección se muestra la comunicación entre el software base seleccionado, los componentes reutilizados y los componentes desarrollados para cumplir los requisitos de la aplicación. También, se recogen los servicios de sistemas externos con los que interactúa nuestro sistema. Se debe incluir un diagrama de componentes que muestre en un alto nivel de abstracción los artefactos que conforman el sistema.

Existen diferentes patrones o estilos arquitectónicos. En los sistemas web de información es común la utilización del patrón Layers (Capas), con el cual estructuramos el sistema en un número apropiado de capas, de forma que todos los componentes de una misma capa trabajan en el mismo nivel de abstracción y los servicios proporcionados por la capa superior utilizan

internamente los servicios proporcionados por la capa inmediatamente inferior. Habitualmente se tienen las siguientes capas:

**Capa de presentación (frontend)** Este grupo de artefactos software conforman la capa de presentación del sistema, incluyendo tanto los componentes de la vista como los elementos de control de la misma.

**Capa de negocio** Este grupo de artefactos software conforman la capa de negocio del sistema, incluyendo los elementos del modelo de dominio y los servicios (operaciones del sistema).

**Capa de persistencia** Este grupo de artefactos software conforman la capa de integración del sistema, incluyendo las clases de abstracción para el acceso a datos (BD o sistema de ficheros) o a sistemas heredados.

Es común que a la capa de negocio y de datos de los sistemas web, se denomine conjuntamente como backend o modelo de la aplicación.

Opcionalmente, podemos disponer de un conjunto de artefactos software que pueden ser usados por elementos de cualquiera de las capas del sistema y que fundamentalmente proporcionan servicios relacionados con requisitos no funcionales (calidad).

## **Parametrización del software base**

En esta sección, se detallan las modificaciones a realizar sobre el software base, que son requeridas para la correcta construcción del sistema. En esta sección incluiremos las actuaciones necesarias sobre la interfaz de administración del sistema, sobre el código fuente o sobre el modelo de datos.

## **Diseño Físico de Datos**

En esta sección se define la estructura física de datos que utilizará el sistema, a partir del modelo de conceptual de clases, de manera que teniendo presente los requisitos establecidos para el sistema de información y las particularidades del entorno tecnológico, se consiga un acceso eficiente de los datos. La estructura física se compone de tablas, índices, procedimientos almacenados, secuencias y otros elementos dependientes del SGBD a utilizar.

## **Diseño detallado de Componentes**

Para cada uno de los módulos funcionales del sistema debemos realizar un diagrama de secuencia, para definir la interacción existente entre las clases de objetos que permitan responder a eventos externos.

## **Diseño detallado de la Interfaz de Usuario**

En esta sección se detallarán las interfaces entre el sistema y el usuario, incluyendo un prototipo de alta fidelidad con el diseño de la IU. Se definirá el comportamiento de las diferentes pantallas, indicando qué ocurre en los distintos componentes visuales de la interfaz cuando aparecen y qué acciones se disparan cuando el usuario trabaja con ellas.

**Capítulo 6**

**Construcción del Sistema**

Este capítulo trata sobre todos los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

## **Entorno de Construcción**

En esta sección se debe indicar el marco tecnológico utilizado para la construcción del sistema: entorno de desarrollo (IDE), lenguaje de programación, herramientas de ayuda a la construcción y despliegue, control de versiones, repositorio de componentes, integración contínua, etc.

## **Código Fuente**

Organización del código fuente, describiendo la utilidad de los diferentes ficheros y su distribución en paquetes o directorios. Asimismo, se incluirá algún extracto significativo de código fuente que sea de interés para ilustrar algún algoritmo o funcionalidad específica del sistema.

## **Scripts de Base de datos**

Organización del código fuente, describiendo la utilidad de los diferentes ficheros y su distribución en paquetes o directorios. Asimismo, se incluirá el script de algún disparador o un procedimiento almacenado, que sea de interés para ilustrar algún aspecto concreto de la gestión de la base de datos.

**Capítulo 7**

**Pruebas del Sistema**

En este capítulo se presenta el plan de pruebas del sistema de información, incluyendo los dife- rentes tipos de pruebas que se han llevado a cabo, ya sean manuales (mediante listas de compro- bación) o automatizadas mediante algún software específico de pruebas.

## **Estrategia**

En esta sección se debe incluir el alcance de las pruebas, hasta donde se pretende llegar con ellas, si se registrarán todas o sólo aquellas de un cierto tipo y cómo se interpretarán y evaluarán los resultados. También, se incluirá el procedimiento a seguir para las pruebas de regresión, esto es, la repetición de ciertas pruebas para comprobar que nuevos cambios que se vayan introduciendo no originen errores en el software ya probado.

## **Entorno de Pruebas**

Incluir en este apartado los requisitos de los entornos hardware/software donde se ejecutarán las pruebas.

## **Roles**

Describir en esa sección cuáles serán los perfiles y participantes necesarios para la ejecución de cada uno de los niveles de prueba.

## **Niveles de Pruebas**

En este sección se documentan los diferentes tipos de pruebas que se han llevado a cabo, ya sean manuales o automatizadas mediante algún software específico de pruebas.

### Pruebas Unitarias

Las pruebas unitarias tienen por objetivo localizar errores en cada nuevo artefacto software desarrollado, antes que se produzca la integración con el resto de artefactos del sistema.

### Pruebas de Integración

Este tipo de pruebas tienen por objetivo localizar errores en módulos o subsistemas completos, analizando la interacción entre varios artefactos software.

### Pruebas de Sistema

En esta actividad se realizan las pruebas de sistema de modo que se asegure que el sistema cumple con todos los requisitos establecidos: funcionales, de almacenamiento, reglas de negocio y no funcionales. Se suelen desarrollar en un entorno específico para pruebas.

#### Pruebas Funcionales

Con estas pruebas se analiza el buen funcionamiento de la implementación de los flujos normales y alternativos de los distintos casos de uso del sistema.

#### Pruebas No Funcionales

Estas pruebas pretenden comprobar el funcionamiento del sistema, con respecto a los requisitos no funcionales identificados: eficiencia, seguridad, etc.

### Pruebas de Aceptación

El objetivo de estas pruebas es demostrar que el producto está listo para el paso a producción. Suelen ser las mismas pruebas que se realizaron anteriormente pero en el entorno de producción. En estas pruebas, es importante la participación del cliente final.

**Parte III**

**Epílogo**

En esta última parte quedarán recogidas las conclusiones y los manuales necesarios para el manejo de la aplicación resultado del desarrollo. Si se ha realizado algún tipo de evaluación de la solución proporcionada, más allá de las pruebas del sistema, también deberá venir recogida en un capítulo separado dentro de esta parte. Pueden consultarse diversos tipos de evaluaciones sobre sistemas de información en [[Hevner et al., 2004](#_bookmark80)]: casos de estudio, análisis estático, análisis dinámico, simulación, experimento controlado, etc.

**Capítulo 8**

**Manual de implantación y explotación**

Las instrucciones de instalación y explotación del sistema se detallan a continuación. Este manual resulta de aplicación en aquellos casos en que se requiere realizar la instalación del software objetivo de este proyecto sobre algún entorno de servidor o cuando su instalación no sea trivial.

## **Introducción**

Resumen de los principales objetivos, ámbito, características y alcance del software desarrollado.

## **Requisitos previos**

Requisitos hardware y software para la correcta instalación del sistema.

## **Inventario de componentes**

Lista de los componentes hardware y software que se incluyen en la versión del producto.

## **Procedimientos de instalación**

Procedimientos de instalación y configuración de cada componente hardware y software (base y desarrollado) para asegurar la correcta instalación y explotación del sistema, así como aquellos procedimientos necesarios de migración/carga de datos.

## **Pruebas de implantación**

Descripción de las pruebas a realizar después de la instalación del sistema.

## **Procedimientos de operación y nivel de servicio**

Procedimientos necesarios para asegurar el correcto funcionamiento, rendimiento, disponibilidad y seguridad del sistema: back-ups, chequeo de logs, etc. También, es preciso indicar claramen- te aquellas actuaciones precisas necesarias para el mantenimiento preventivo del sistema y así prevenir posibles fallos en el mismo.

**Capítulo 9**

**Manual de usuario**

Las instrucciones de uso del software se detallan a continuación. Este manual se dirige al usuario final del software objetivo de este proyecto.

## **Introducción**

Resumen de los principales objetivos, ámbito, características y alcance del software desarrollado.

## **Instalación**

Se detallarán los pasos necesarios para la obtención e instalación del software, así como los requisitos previos de hardware y software.

## **Uso del sistema**

Describir todos los aspectos necesarios para una utilización efectiva y eficiente de las caracterís- ticas del sistema por parte de los usuarios.

**Capítulo 10**

**Manual del desarrollador**

A continuación se recogen las instrucciones necesarias para evolucionar el software. Este manual está dirigido a los desarrolladores que pretenden extender o modificar el código fuente, con el fin de incorporar nuevas funcionalidades o modificar las ya existentes. A lo largo de este capítulo se deberán hacer referencias explicitas a aquellos epígrafes de los capítulos de Diseño, Construcción y Pruebas del Sistema que resulten de interés.

## **Introducción**

Resumen de los principales objetivos, ámbito, características y alcance del software desarrollado.

## **Preparación del entorno de trabajo**

Descripción de los requisitos (hardware y software) previos. Datos de interés relativos al control de versiones del software. Detalles sobre la instalación en local del entorno de desarrollo y, si fuesen necesarios, de otros componentes como bases de datos, servidores de aplicaciones, etc.

## **Consideraciones generales sobre el desarrollo**

Aspectos importantes a tener en cuenta a la hora de modificar y extender el código fuente, guías de estilo, etc. Asimismo, se detallarán las directrices que sean de aplicación a la hora de realizar pruebas sobre las nuevas mejoras introducidas.

## **Instrucciones para construcción y despliegue**

Secuencia de pasos requeridos para llevar a cabo la compilación del código fuente y así poder construir y depurar el software sobre una máquina de desarrollo.

**Capítulo 11**

**Conclusiones**

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

## **Objetivos alcanzados**

Este apartado debe resumir los objetivos generales y específicos alcanzados, relacionándolos con todo lo descrito en el capítulo de introducción.

## **Lecciones aprendidas**

A continuación, se detallan las buenas prácticas adquiridas, tanto tecnológicas como procedi- mentales, así como cualquier otro aspecto de interés.

Resumir cuantitativamente el tiempo y esfuerzo dedicados al proyecto a lo largo de su desarrollo que escribir un sencillo ’he trabajado mucho en este proyecto’.

## **Trabajo futuro**

En esta sección, se presentan las diversas áreas u oportunidades de mejora detectadas durante el desarrollo del proyecto y que podrán ser abarcadas en futuras versiones del software.

Los elementos aquí descritos deben estar en relación con lo relatado en el apartado de objetivos y alcance del proyecto descritos en la introducción.

# Bibliografía

[CCOO, 2010]CCOO (2010). Tablas salariales 2010 IV Convenio Colectivo. [http://www.uca.](http://www.uca.es/sindicato/ccoo/documentos/tabla-salarial-pas-laboral-2010.pdf) [es/sindicato/ccoo/documentos/tabla-salarial-pas-laboral-2010.pdf](http://www.uca.es/sindicato/ccoo/documentos/tabla-salarial-pas-laboral-2010.pdf).

[Hevner et al., 2004]Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105.

Información sobre Licencia

Incluir aquí la información relativa a la licencia seleccionada para la documentación y software del presente proyecto.