UNIVERSIDAD HISPANOAMERICANA

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

PROPUESTA DE MÉTODOS Y ESTÁNDARES DE DESARROLLO EN LENGUAJE JAVA PARA LA UNIVERSIDAD DE COSTA RICA, SEDE RODRIGO FACIO, EN EL PERIODO 2017

MAURICIO SALAS CHAVES

DIRECTORA: YENORY ROJAS HERNÁNDEZ

I SEMESTRE, 2017

Contenido

Figuras

Tablas

DECLARACIÓN JURADA

DECLARACIÓN JURADA

Yo **Mauricio Salas Chaves**, mayor de edad, portador de la cédula de identidad número **2-0706-0790** egresado de la carrera de **Ingeniería Informática** de la Universidad Hispanoamericana, hago constar por medio de éste acto y debidamente apercibido y entendido de las penas y consecuencias con las que se castiga en el Código Penal el delito de perjurio, ante quienes se constituyen en el Tribunal Examinador de mi trabajo de tesis para optar por el título de **Licenciatura**, juro solemnemente que mi trabajo de investigación titulado: **Propuesta de métodos y estándares de desarrollo en lenguaje java para la Universidad de Costa Rica, sede Rodrigo Facio en el periodo 2017**, es una obra original que ha respetado todo lo preceptuado por las Leyes Penales, así como la Ley de Derecho de Autor y Derecho Conexos número 6683 del 14 de octubre de 1982 y sus reformas, publicada en la Gaceta número 226 del 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte; articulo 70. Es permitido citar a un autor, transcribiendo los pasajes pertinentes siempre que éstos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor de la obra original. Asimismo, quedo advertido que la Universidad se reserva el derecho de protocolizar este documento ante Notario Público. en fe de lo anterior, firmo en la ciudad de San José, a los **25** días del mes de **julio** del año dos mil **diecisiete**.

Mauricio Salas Chaves
Firma del estudiante
Cédula 2-0706-0790

CARTA DEL TUTOR

CARTA DEL TUTOR

Heredia, 25 Julio, 2017

Departamento de Registro
Ingeniería en Sistemas
Universidad Hispanoamericana

Estimado señor o señora:

El estudiante **MAURICIO SALAS CHAVES**, cédula de identidad número 2-0706-0790, me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado **PROPUESTA DE MÉTODOS Y ESTÁNDARES DE DESARROLLO EN LENGUAJE JAVA PARA LA UNIVERSIDAD DE COSTA RICA, SEDE RODRIGO FACIO EN EL PERIODO 2017**, el cual ha elaborado para optar por el grado académico de licenciatura.
En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación, antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos, conclusiones y recomendaciones.

De los resultados obtenidos por el postulante se obtiene la siguiente calificación:

a)	Original del tema	10%	8%
b)	Cumplimiento de entrega de avances	20%	18%
c)	Coherencia entre los objetivos, los instrumentos aplicados y los resultados de la investigación	30%	27%
d)	Relevancia de las conclusiones y recomendaciones	20%	19%
e)	Calidad, detalle del marco teórico	20%	18%
	Total		90%

En virtud de la calificación obtenida se avala el traslado al proceso de lectura.

Atentamente,

Ing. Erick López Ch. M.R.I.
1-0993-0088

CARTA DEL LECTOR

CARTA DE LECTOR

Universidad Hispanoamericana
Sede Heredia
Carrera Ingeniería Informática

Estimado señor

El estudiante Mauricio Salas Chaves, cédula de identidad: 2-0706-0790, me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado " PROPUESTA DE MÉTODOS Y ESTÁNDARES DE DESARROLLO EN LENGUAJE JAVA PARA LA UNIVERSIDAD DE COSTA RICA, SEDE RODRIGO FACIO EN EL PERIODO 2017", el cual ha elaborado para obtener su grado de Licenciatura en Ingeniería Informática.

He revisado y he hecho las observaciones relativas al contenido analizado, particularmente lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; asimismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación. Así mismo, se realizaron las modificaciones solicitadas a nivel de nombre de proyecto y objetivos.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.

Atte.

Firma 

Nombre Ing. José Roberto Santamaría Sandoval

Cédula 1-1178-0664

CARTA DEL FILÓLOGO

26 de agosto del 2017

Señores
Universidad Hispanoamericana
Facultad de Ingeniería
Escuela de Ingeniería Informática

Estimados señores:

Leí y corregí el Trabajo Final de Graduación denominado: **Propuesta de métodos y estándares de desarrollo en lenguaje Java para la Universidad de Costa Rica, sede Rodrigo Facio, en el periodo 2017,** elaborado por el estudiante Mauricio Salas Chaves, para optar por la Licenciatura en Ingeniería Informática.

Corregí el trabajo en aspectos tales como: construcción de párrafos, vicios del lenguaje que se trasladan a lo escrito, ortografía, puntuación y otros relacionados con el campo filológico, y desde ese punto de vista considero que está listo para ser presentado como Trabajo Final de Graduación, por cuanto cumple con los requisitos establecidos por la Universidad.

Cordialmente,

Licda. Ginette Fonseca Vargas
Filóloga
Carné 10993

DEDICATORIA

Dedico este trabajo, principalmente, a Dios, por haberme dado la vida y permitirme haber llegado hasta este momento tan importante de mi formación profesional.

De igual forma, dedico esta tesis a mis padres, que han inculcado buenos hábitos y valores, lo cual me ha ayudado a salir adelante en todo momento.

AGRADECIMIENTO

Quiero agradecerle a Dios, quien me ha mantenido siempre por el buen camino y me ha llenado de buenas oportunidades.

También a mis padres, por su gran esfuerzo para darme una buena educación y estar siempre a mi lado en todo momento.

INTRODUCCIÓN

En este trabajo se realizará una propuesta de estándares y métodos de desarrollo en lenguaje Java para el Área de Desarrollo de Sistemas del Centro de Informática, en la Universidad de Costa Rica, Sede Rodrigo Facio. Este departamento realiza el desarrollo de sus sistemas en lenguaje ASP .Net y Winforms, esta metodología se ha mantenido estable por al menos 9 años, permitiendo crear una arquitectura robusta y madura, donde también se cuenta con manuales, los cuales han sido creados para el desarrollador explicando el paso a paso en un proyecto. Gracias a esto el enfoque de todos los desarrolladores de esa área se centraliza en la elaboración de requerimientos basados en una lista de estándares y métodos de desarrollo ya definidos, permitiendo además la facilidad de mantenimiento sobre los sistemas ya terminados.

Céspedes (2016) indica que en los últimos años la Universidad de Costa Rica ha iniciado un plan de migración a *software* libre, donde los distintos sectores que conforman esta universidad buscan como ser partícipes de esta ideología. En el Área de Desarrollo de Sistemas del Centro de Informática se dio la orden de cambiar a la plataforma Java para todos los nuevos proyectos que se asignen a partir de este año 2017, lenguaje sobre el cual se tiene un conocimiento básico y no ha sido utilizado para desarrollar ningún proyecto, generando un atraso sobre todos los proyectos nuevos ya que no se cuenta con una arquitectura y una metodología de desarrollo para este lenguaje. Con este proyecto se pretende dar un mejor rumbo a todos los nuevos proyectos que se desarrollen en este lenguaje, para que sean desarrollados sobre un mismo marco de trabajo y cuenten con las mismas virtudes de los proyectos realizados en los otros lenguajes.

CAPÍTULO 1: Planteamiento del tema

1.1. Definición del problema

> El Centro de Informática, en su papel de ente rector en materia de tecnologías de información y comunicación para la Universidad de Costa Rica, funge como ente catalizador del cambio a través de proyectos de migración de equipos a sistemas operativos y plataforma de ofimática libre. De igual forma, como medio para apoyar el proceso de migración, busca el cambio de la plataforma de desarrollo a tecnologías Open Source, con la finalidad de disminuir el monto de inversión de licenciamiento y mantener un nivel de seguridad y productividad alto en las aplicaciones desarrolladas. (Arias, s.f.)

El Área de Desarrollo de Sistemas de la Universidad de Costa Rica es una pequeña sección del Centro de Informática que actualmente se divide en 3 pequeños equipos, los cuales se encargan de realizar aplicaciones para la misma universidad, donde se busca agilizar muchas de las actividades que se realizan en la institución, manejando un gran volumen de datos.

Entre estos se encuentra el sistema para el Examen General Básico Clínico, el cual maneja todo el proceso de certificación de títulos para personas que estudiaron medicina general en el extranjero, el sistema de Órdenes de Trabajo, el cual maneja toda la gestión de trabajos de reparación o preventivos, como lo es el arreglo de tuberías, entre otros. Todos estos sistemas han sido de gran éxito y están desarrollados bajo un mismo esquema de trabajo, lo cual facilita el mantenimiento en caso de algún fallo o bien facilita que el desarrollo de un sistema nuevo sea más fácil y rápido para los desarrolladores.

Debido al cambio de lenguaje solicitado por la alta gerencia, el inicio de los nuevos proyectos de este año 2017 se ha visto afectado, ya que no se cuenta con un esquema de trabajo definido, lo cual causa atrasos en el desarrollo debido a los grandes periodos de tiempo en investigación que es necesario invertir, además, al no tener una línea definida cada uno de los sistemas se ve desarrollado de manera antojadiza por parte del desarrollador, lo cual en el futuro va a causar una mayor dificultad para darles soporte, debido a las diferentes técnicas o métodos de desarrollo de parte de cada uno de los desarrolladores.

El equipo de trabajo no cuenta con un amplio conocimiento de la herramienta, por lo que no conocen a fondo las ventajas o limitaciones de este nuevo lenguaje sobre el que se está utilizando actualmente. Debido a lo anterior, surge como pregunta: ¿Cuáles técnicas de desarrollo de *software* en Java, usadas actualmente en el mercado, se pueden utilizar para optimizar el proceso en la Universidad de Costa Rica, sede Rodrigo Facio, para el año 2017?

1.2. Justificación del proyecto

La Universidad de Costa Rica se ha posicionado como una de las instituciones líderes y con más experiencia en el uso de tecnologías abiertas. Gracias a esta experiencia se ha brindado asesoría e información a instituciones tales como: La Asamblea Legislativa, la Contraloría General de la República, la CCSS, Municipalidades, entre otras. (Universidad de Costa Rica, s.f.)

Las autoridades universitarias han expresado abiertamente su apoyo al uso de software libre por medio de las actas de sesión número 5302 y 5574 del Consejo

Universitario y fomentan su adopción a través de la migración de la plataforma ofimática y otras herramientas para las cuales existe un equivalente que no perjudique el rendimiento del funcionario. (Semanario Universidad, 2011)

1.2.1. Motivos de migración a *software* libre

Según la UCR (2014) algunas de las razones por las que se está realizando la migración a *software* libre son las siguientes:

1.2.1.1. Aprovechar los recursos

Con el uso de aplicaciones libres es posible redistribuir parte del presupuesto universitario que se usa para la adquisición de *software* privativo e invertirlo en la compra de programas de cómputo especializados y de índole científica.

Por ejemplo: Del 2008 al 2010, el presupuesto utilizado para adquirir la aplicación Microsoft Office correspondió al 40% del presupuesto para compra de *software* administrado por la Comisión Institucional de Equipamiento (CIEQ). Al migrar de Microsoft Office a alguna aplicación de ofimática libre como Libre Office, puede utilizarse ese presupuesto en la compra de *software* especializado.

En contraste, las solicitudes de *software* para el 2010 son seis veces más del presupuesto con el que cuenta la CIEQ, por lo que en la universidad existen necesidades no cubiertas en la adquisición de programas de cómputo para docencia e investigación en temas especializados como robótica, análisis celular y genético, diseño asistido por computadora, sistemas de información geográfica, entre otros.

1.2.1.2. Proteger el acceso a la información

La utilización de formatos abiertos como OpenDocument (ODF) permite crear archivos que pueden ser manipulados con varias aplicaciones, de tal manera que la persona usuaria pueda elegir entre varias opciones. Además, los formatos abiertos están respaldados por un estándar que no depende de un único proveedor y pueden ser utilizados para desarrollar nuevas aplicaciones.

1.2.1.3. Tener autonomía y no depender de un único proveedor

Al utilizar *software* privativo la Universidad se ve condicionada a elegir solo entre ciertos proveedores de *software*.

Una vez adquirida una aplicación, las opciones para soporte, mantenimiento y actualizaciones son muy limitadas y en ocasiones únicas; y si se desea cambiar de proveedor, es posible que sea necesario cambiar también la aplicación utilizada, lo que trae consigo gastos adicionales.

Esto genera relaciones de dependencia tecnológica con proveedores comerciales que limitan la autonomía que pueda tener la Universidad respecto a las aplicaciones que soportan los procesos de producción y circulación de la información universitaria.

1.2.1.4. Generar conocimiento implementando nuevas aplicaciones

La Universidad, al adquirir un compromiso con el *software* libre, se convierte en un

actor activo dentro de una comunidad mundial de universidades e instituciones públicas que promueven, usan y desarrollan *software* libre, lo que es coherente con su naturaleza de institución educativa a cargo de la producción y enseñanza de conocimientos.

Debido a esto, varios sectores que conforman la Universidad de Costa Rica buscan siempre mantenerse en un continuo cambio y en la búsqueda de nuevas tecnologías que sea *Open Source*, además es de acatamiento obligatorio la puesta en marcha del algún plan de migración según se indica en la resolución R-289-2014 (Salom, 2014).

Además, el Centro de Informática debe ser uno de los pilares en este tipo de decisiones y también debe dar el ejemplo, ya que es donde se realiza la mayor cantidad de investigaciones tecnológicas y de nuevas tendencias. En este sector se encuentran distintas áreas, las cuales tienen un enfoque más definido, entre ellas están el área de desarrollo de sistemas, área de diseño, área de *software* libre, área de mantenimiento a equipos, área de base de datos, entre otras; esta primer área mencionada se ha dado a la tarea de cambiar el lenguaje de desarrollo a Java, al ser este lenguaje nuevo para los equipos de desarrollo, es necesario implementar una arquitectura desde el inicio la cual sea estable y muy similar a la que se encuentra actualmente implementada para el lenguaje de ASP .NET y WinForms, razón por la cual surge la necesidad de realizar amplias investigaciones con el fin de que no exista ninguna afectación a la hora de la migración total del lenguaje y se logre sacar la mayor ventaja de cambio.

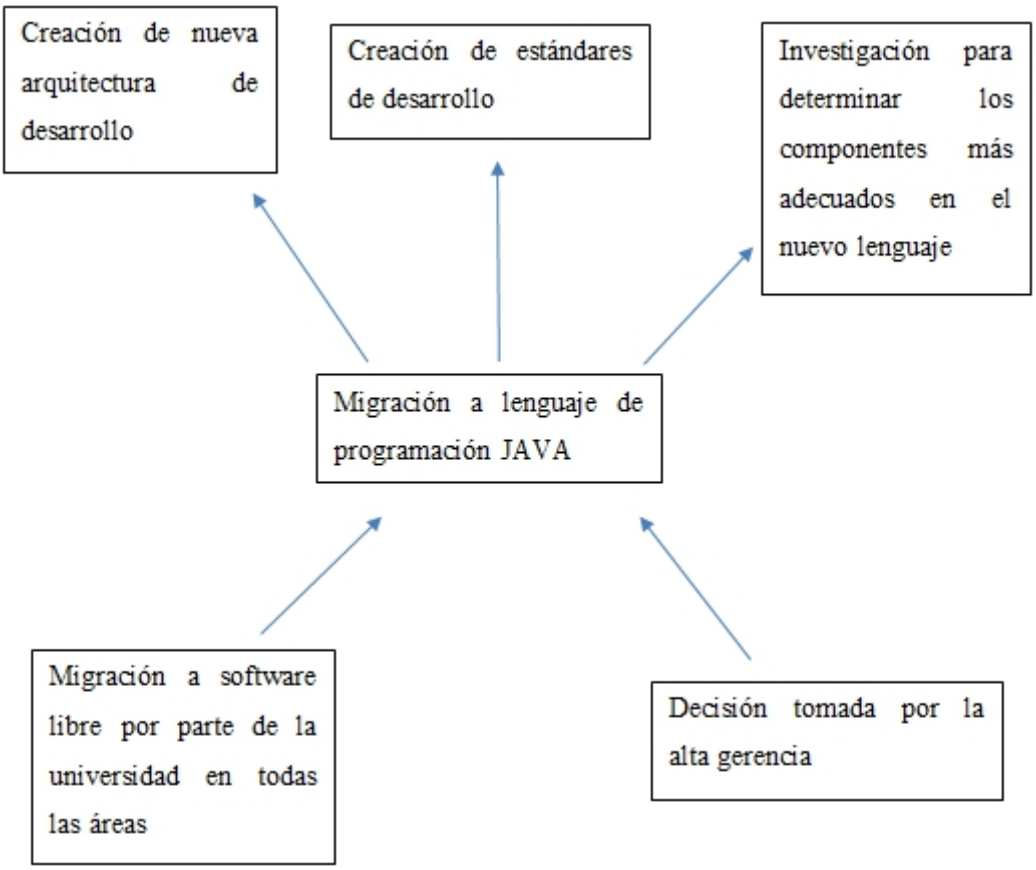


Figura 1. Resumen del problema

Fuente: Elaboración propia.

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. Objetivo general

- Proponer estándares y metodologías de desarrollo de *software* en lenguaje Java para la Universidad de Costa Rica en la sede Rodrigo Facio, mediante la aplicación de diferentes técnicas de desarrollo utilizadas en el mercado para la optimización del proceso en la entidad.

1.3.2. Objetivos específicos

- Diagnosticar la situación actual de la organización en la aplicación de estándares y metodologías de desarrollo en “ASP .net”.
- Determinar puntos de mejora en el proceso de desarrollo actual y aplicarlos en la nueva propuesta sobre el lenguaje Java.
- Estructurar estándares y una metodología de desarrollo sobre un proyecto base para verificar su funcionamiento
- Implementar los estándares y metodología de desarrollo propuestos en el primer módulo del proyecto “Sistema de órdenes de producción”.

1.4. MARCO DE REFERENCIA EMPRESARIAL Y CONTEXTUAL

La Universidad de Costa Rica se fundó en el año 1940 (Universidad de Costa Rica, 2016), nace como una institución docente y de cultura superior. Abrió sus puertas inicialmente en el barrio capitalino González Lahmann con apenas 719 estudiantes y con el avance del tiempo fueron creadas las actuales sedes y recintos que conforman esta institución, siendo encabezados por su campus principal, la Ciudad Universitaria Rodrigo Facio, ubicada en San Pedro de Montes de Oca.

Además, como institución autónoma, se encuentra conformada actualmente, por una comunidad de profesores, funcionarios administrativos y estudiantes, donde su principal actividad es la enseñanza, cuenta con una amplia gama de carreras universitarias y postgrados. Entre sus otras funciones se puede nombrar la investigación, el estudio, la creación artística y la difusión del conocimiento.

En un estudio realizado por la empresa Quacquarelli Symonds (2016), se obtuvo el resultado de que para el año 2016 la Universidad de Costa Rica está considerada dentro de las mejores universidades de América Latina, en el puesto número 18.

1.4.1. Misión

> Generar las transformaciones que la sociedad requiere para alcanzar un desarrollo integral, basado en el respeto a las diferencias ideológicas y culturales, la justicia social y el mejoramiento de la calidad de vida, mediante la formación de profesionales humanistas y la acción conjunta de la Docencia, Investigación, Acción social, Servicios estudiantiles y Administración, desde la región occidental del país. (UCR, s.f.)

1.4.2. Visión

> La Sede de Occidente, con base en su vocación y experiencia de desarrollo universitario, mantendrá y desempeñará un papel protagónico en el campo de la educación superior, mediante la autoevaluación constante, para proponer con innovación y compromiso social las transformaciones que demanda la sociedad, al formar profesionales con excelencia académica y humanista que favorezcan la inclusión social, la equidad y la justicia. (UCR, s.f.)

1.4.3. Organización académica

La Universidad de Costa Rica se encuentra conformada por seis áreas académicas, las cuales se pueden subdividir en facultades o escuelas, a continuación se detalla cómo se encuentra conformada cada una de estas áreas.

1.4.3.1. Área de Artes y Letras

- Facultad de Letras
- Escuela de Filosofía de la Universidad de Costa Rica
- Escuela de Lenguas Modernas de la Universidad de Costa Rica
- Escuela de Filología, Lingüística y Literatura de la Universidad de Costa Rica
- Facultad de Bellas Artes
- Escuela de Artes Dramáticas de la Universidad de Costa Rica
- Escuela de Artes Plásticas de la Universidad de Costa Rica
- Escuela de Artes Musicales de la Universidad de Costa Rica

1.4.3.2. Área de Ciencias Sociales

- Facultad de Ciencias Sociales
- Escuela de Ciencias de la Comunicación Colectiva de la Universidad de Costa Rica
- Escuela de Psicología de la Universidad de Costa Rica
- Escuela de Ciencias Políticas de la Universidad de Costa Rica
- Escuela de Trabajo Social de la Universidad de Costa Rica
- Escuela de Historia de la Universidad de Costa Rica

- Escuela de Geografía de la Universidad de Costa Rica
- Escuela de Antropología de la Universidad de Costa Rica
- Escuela de Sociología de la Universidad de Costa Rica
- Facultad de Derecho
- Facultad de Ciencias Económicas
- Escuela de Administración de Negocios de la Universidad de Costa Rica
- Escuela de Administración Pública de la Universidad de Costa Rica
- Escuela de Economía de la Universidad de Costa Rica
- Escuela de Estadística de la Universidad de Costa Rica
- Facultad de Educación
- Escuela de Formación Docente de la Universidad de Costa Rica
- Escuela de Orientación y Educación Especial de la Universidad de Costa Rica
- Escuela de Bibliotecología y Ciencias de la Información
- Escuela de Educación Física y Deportes de la Universidad de Costa Rica

1.4.3.3. Área de Ciencias Básicas

- Facultad de Ciencias
- Escuela de Física de la Universidad de Costa Rica
- Escuela de Matemática de la Universidad de Costa Rica
- Escuela de Biología de la Universidad de Costa Rica
- Escuela Centroamericana de Geología
- Escuela de Química de la Universidad de Costa Rica

1.4.3.4. Área de Ciencias Agroalimentarias

- Facultad de Ciencias Agroalimentarias
- Escuela de Agronomía de la Universidad de Costa Rica
- Escuela de Zootecnia de la Universidad de Costa Rica
- Escuela de Economía y Agronegocios de la Universidad de Costa Rica
- Escuela de Tecnología de Alimentos de la Universidad de Costa Rica

1.4.3.5. Área de Ingenierías

- Facultad de Ingeniería
- Escuela de Arquitectura
- Escuela de Ingeniería Civil de la Universidad de Costa Rica
- Escuela de Ingeniería Eléctrica de la Universidad de Costa Rica
- Escuela de Ingeniería Industrial de la Universidad de Costa Rica
- Escuela de Ingeniería Mecánica de la Universidad de Costa Rica
- Escuela de Ingeniería Química de la Universidad de Costa Rica
- Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica
- Escuela de Ingeniería Agrícola de la Universidad de Costa Rica
- Escuela de Ingeniería Topográfica de la Universidad de Costa Rica
- Escuela de Ingeniería en Marina Civil con énfasis en Ingeniería Náutica y transporte marítimo, Ingeniería Marina y Radioelectrónica

1.4.3.6. Área de Salud

- Facultad de Medicina
- Escuela de Enfermería
- Escuela de Medicina de la Universidad de Costa Rica
- Escuela de Nutrición de la Universidad de Costa Rica
- Escuela de Salud Pública de la Universidad de Costa Rica
- Escuela de Tecnologías en Salud de la Universidad de Costa Rica
- Facultad de Farmacia
- Facultad de Microbiología
- Facultad de Odontología

Además se cuenta con la Escuela de Estudios Generales, que tiene a su cargo la formación humanística por medio de cursos obligatorios para todo estudiante en su primer año universitario.

1.5. ALCANCES Y LIMITACIONES

1.5.1. Alcances

- El primer entregable del proyecto es un diagnóstico de la situación actual de la Universidad de Costa Rica en la aplicación de estándares y metodologías de desarrollo en el lenguaje ASP .net.
- El segundo entregable del proyecto es la identificación de puntos de mejora para lograr aplicarlos a la nueva propuesta de estándares y metodología de desarrollo en el lenguaje Java, en la cual se detalle la estructura básica de un proyecto, manera de declarar un variable (prefijos), método, función, entidad,

sentencia SQL, manejo de excepciones, manejo de referencias, estándar para definir un objeto a nivel de base de datos.

- El tercer entregable del proyecto es un demo o proyecto base donde se apliquen todos los estándares propuestos, autenticación, manejo de opciones de menú del sistema, recuperación de información desde la base de datos y un catálogo base con las operaciones básicas (agregar, modificar, eliminar) partiendo de lo que se encuentre en el mercado actualmente.
- El cuarto entregable del proyecto es aplicar todo el conocimiento adquirido durante la investigación en el nuevo sistema en desarrollo de la Universidad de Costa Rica, sede Rodrigo Facio, el cual tiene el nombre de “Sistema de órdenes de producción” sobre el primer módulo del sistema, que consiste en la solicitud de órdenes de producción de parte de cualquier funcionario de la institución.

1.5.2. Limitaciones

- La implementación de los servidores que van a contener los aplicativos en lenguaje Java y el manejo de control de cambios que se va realizando conforme avanza un proyecto corresponde a una tarea única del arquitecto de sistemas de la Universidad de Costa Rica, por lo que no se van a considerar estos puntos durante la investigación.
- El diseño de los estilos que conllevan los sistemas, el estándar del tamaño de las fuentes y la combinación de colores corresponden a una tarea del área de diseño interno del Centro de Informática.

1.6. CRONOGRAMA DE ACTIVIDADES

Tabla 1. Diseño del Proyecto de Graduación

Nombre de la tarea	Duración	Comienzo	Fin
Documento de investigación sobre lenguaje Java y Visual Basic .Net	**1 mes**	**01/01/2017**	**31/01/2017**
Ventajas del lenguaje Java y Visual Basic	4 días	01/01/2017	04/01/2017
Ventajas Java	2 días	01/01/2017	02/01/2017
Ventajas Visual Basic	2 días	03/01/2017	04/01/2017
Desventajas del lenguaje Java y Visual Basic	4 días	05/01/2017	08/01/2017
Desventajas Java	2 días	05/01/2017	06/01/2017
Desventajas Visual Basic	2 días	07/01/2017	01/01/2017
Costo de licencias en Visual Basic	1 día	09/01/2017	09/01/2017
Costos relacionados con Java	1 día	10/01/2017	10/01/2017
Implementación a nivel empresarial de Java	1 semana	11/01/2017	18/01/2017
Requisitos	5 días	19/01/2017	23/01/2017
Manejo de control de versiones de un proyecto en Java	4 días	24/01/2017	27/01/2017
Team foundation (Visual Basic)	2 días	28/01/2017	29/01/2017
Maven (Java)	2 días	30/01/2017	31/01/2017
Documento con propuesta de estándares de programación	**1 mes**	**01/02/2017**	**28/02/2017**
Estructura básica de un proyecto	2 semanas	01/02/2017	15/02/2017
Variables	2 días	01/02/2017	02/02/2017
Métodos	2 días	03/02/2017	04/02/2017
Funciones	2 días	05/02/2017	06/02/2017
Servicio web	3 días	07/02/2017	09/02/2017
Entidad	2 días	10/02/2017	11/02/2017

Nombre de la tarea	Duración	Comienzo	Fin
Sentencia SQL	1 día	12/02/2017	12/02/2017
Manejo de excepciones	2 días	13/02/2017	14/02/2017
Manejo de referencias	1 día	15/02/2017	15/02/2017
Definición de objetos en la base de datos	1 semana	16/02/2017	22/02/2017
Vistas	2 días	16/02/2017	17/02/2017
Procedimientos	2 días	18/02/2017	19/02/2017
Funciones	2 días	20/02/2017	21/02/2017
Triggers	1 día	22/02/2017	22/02/2017
Revisión con el arquitecto de sistemas de la Universidad de Costa Rica	6 días	23/02/2017	28/02/2017
Creación de proyecto base	2 meses	01/03/2017	30/04/2017
Autenticación en el sistema	4 días	01/03/2017	04/03/2017
Opciones de menú	2 días	05/03/2017	07/03/2017
Recuperación de información desde la base de datos	1 semana 4 días	08/03/2017	18/03/2017
Listado	1 semana	19/03/2017	25/03/2017
Consulta de datos	4 días	19/03/2017	22/03/2017
Filtrado de información	3 días	23/03/2017	25/03/2017
Formulario	2 semanas	26/03/2017	08/04/2017
Agregar un registro	1 semana	26/03/2017	05/04/2017
Modificar un registro	2 días	06/07/2017	07/04/2017
Eliminar un registro	1 día	08/04/2017	08/04/2017
Manejo *web service*	4 días	09/04/2017	12/04/2017
Pruebas del sistema	1 semana	13/04/2017	20/05/2017
Revisión con el arquitecto de sistemas de la Universidad de Costa Rica	3 días	21/05/2017	24/05/2017
Documentación del proyecto	6 días	25/05/2017	30/05/2017
Implementación de conocimientos en el primer módulo del “Sistema de	**4 meses**	**01/05/2017**	**31/08/2016**

Nombre de la tarea	Duración	Comienzo	Fin
órdenes de producción”			
Reunión con la analista encargada del sistema para análisis de requerimientos	1 semana	01/05/2017	06/05/2017
Desarrollo de requerimientos	2 meses y medio	06/05/2017	22/07/2017
Uso de la propuesta de estándares			
Uso de la metodología de desarrollo propuesta			
Revisión de los requerimientos	2 semanas	23/07/2017	05/08/2017
Aprobación de parte de la analista del sistema	1 semana	23/07/2017	30/07/2017
Aprobación de parte del usuario experto	1 semana	31/07/2017	05/08/2017
Creación de manuales del sistema	4 semanas	06/08/2017	31/08/2017

Fuente: Elaboración propia.

CAPÍTULO II: MARCO TEÓRICO

2.1. CONCEPTOS BÁSICOS EN *SOFTWARE*

La carrera de Ingeniería Informática es una base muy amplia sobre la cual lo que se busca es formar personas capaces de crear soluciones de cómputo y comunicaciones, que sean capaces de procesar información de una manera ágil y automática, en la cual todas las personas que desean tener conocimientos relacionados con esta área se van a encontrar con varias ramificaciones, las cuales en conjunto cumplen con esta finalidad, depende del gusto de cada persona elegir cuál o cuáles son las más atractivas, con el fin de enfocar sus estudios, especializarse y posteriormente aplicar sus conocimientos ya en el ámbito laboral.

Con el paso de los años la informática se ha convertido en uno de los pilares para todas las grandes empresas, debido a los grandes volúmenes de información que se manejan en la actualidad y al constante cambio de los intereses del cliente, debido a esto se realizan grandes inversiones en esta área.

Según Cohen y Asín (2009), en la actualidad podemos encontrarnos con dos tipos de programas o *software*, el primer tipo son todos los que comprenden un conjunto de subprogramas que permiten interactuar con el sistema computacional, como ejemplos de esta clase de programas podemos encontrar los sistemas operativos como Windows, Linux o Solaris. Por otro lado se encuentran los programas de aplicación, constituidos a su vez por subprogramas que resuelven problemas funcionales para todos sus usuarios, por ejemplo Microsoft Office, o un sistema desarrollado para apoyar en el proceso de toma de decisiones o un sistema transaccional (p. 304).

Los sistemas y las tecnologías de información son una nueva herramienta presente en las organizaciones, que se suma a otras áreas como las finanzas, la contabilidad, los recursos humanos, la logística y las operaciones; además, comprenderlas, así como su uso, es de suma importancia, ya que son un componente vital en el éxito de los negocios y organizaciones. En específico, los sistemas de información (SI) han venido a cambiar la forma en que operan las organizaciones actuales. A partir de su uso se logran importantes mejoras, como la automatización de los procesos operativos que proporcionan información de apoyo al proceso de toma de decisiones y, lo que es más importante, su implantación facilita el logro de ventajas competitivas. De este modo la aplicación de las tecnologías de información en los negocios constituye un campo de estudio fundamental para la ciencia de la administración y gestión de negocios (Cohen & Asín, 2009, p. 2).

El impacto de los sistemas y las tecnologías de información es muy grande ya que cambia muchos de los procesos internos y la forma de operar de las empresas, buscando facilitar las tareas y agilizar procesos para una mejor toma de decisiones. Todo esto de una manera única y cada vez más acelerada, ya que conforme pasa el tiempo los equipos son capaces de procesar una mayor cantidad de transacciones en una menor cantidad de tiempo.

2.1.1. Sistemas de información (SI)

Los sistemas de información son varios procesos relacionados entre sí, los cuales se encargan de llevar a cargo una función mediante el procesamiento de información, facilitando también la toma de decisiones. Entre las definiciones que existen para este término, podemos encontrar:

> Un sistema de información se puede definir técnicamente como un conjunto de componentes relacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control en una organización. (Instituto Técnico de Sonora, s.f.)

Los SI se pueden dividir en dos tipos, uno de ellos es una solución creada por algún fabricante, la cual recibe el nombre de *Enterprise Resource Planning* (ERP), la cual trata de responder a la mayor cantidad de procesos o necesidades de una empresa para una tarea en específico, y se vende o entrega como producto ya terminado a las empresas, las cuales se deben adaptar a las funcionalidades que ya trae implementado el SI; por otro lado están los sistemas hechos a la medida, los cuales toma más tiempo implementar ya que nacen a partir de una idea y es necesario crearlos desde 0, además son más costosos debido a todo el proceso de desarrollo, pero cumplen con todas las necesidades de una empresa gracias a que son personalizados completamente, estos últimos son los que se van a tratar más a fondo para este trabajo debido al tema del proyecto.

2.1.2. *Enterprise Resource Planning* (ERP)

Para este concepto podemos encontrar muchas definiciones de distintos autores, pero todas llegan al mismo punto.

"El ERP es un sistema integral de gestión empresarial que está diseñado para modelar y automatizar la mayoría de procesos en la empresa (área de finanzas, comercial, CMR, logística, producción, etc.)" (LIDER IT Consulting, s.f., p. 2).

Un ERP es un sistema o paquete ya creado por un fabricante, que cumple con una funcionalidad específica, por ejemplo, el paquete de Office de Microsoft, el cual tiene un procesador de texto (Word), hojas de cálculo (Excel), entre otros; por otro lado también podemos usar como ejemplo el servicio de mensajería y de video llamadas Skype, el cual es adquirido por las empresas para facilitar una tarea.

Una de las principales razones por la cual este tipo de programa es adquirido es porque no es viable desarrollar un sistema personalizado para tareas como las descritas anteriormente, ya que se debe considerar el tiempo de desarrollo, el cual para un ERP no existe y el costo, que es menor en este tipo de sistemas y finalmente el producto final, si escogemos un sistema personalizado o un ERP va a ser muy similar.

2.1.3. Sistemas hechos a la medida

Los sistemas hechos a la medida son aplicativos que cubren necesidades presentes y futuras específicas para una empresa, moldeándose a una operación ágil y eficiente, lo cual le genera grandes beneficios a las empresas. Para poder desarrollar un sistema, se debe estudiar muy detalladamente los procesos, para detectar problemas o posibles afectaciones que se vayan a tener en el futuro, y para que el producto final cumpla realmente con las expectativas y se le pueda sacar el mayor provecho.

Una vez que se cuente con toda esta información se puede proceder con el ciclo de desarrollo de sistemas para conseguir el producto deseado.

A menudo en las empresas surge la pregunta: ¿Cuál es la mejor opción, desarrollar un sistema o buscar uno ya hecho? Todo esto depende de la necesidad de la empresa y las alternativas con las que cuenta el mercado para dar una solución; sin embargo, ninguna de las dos posibilidades es mejor que la otra o debe descartarse completamente, es necesario realizar un estudio exhaustivo para poder así llegar a una conclusión (MADO, 2017).

2.2. DESARROLLO DE *SOFTWARE*

El desarrollo de *software* ha ido evolucionando con el paso de los años, buscando facilitar la metodología y manera de desarrollar para todos los involucrados en un nuevo sistema.

Dentro de los cambios más notables se encuentra en primer lugar los lenguajes de programación, los cuales se han ido simplificando, facilitando la tarea del desarrollador, donde inicialmente se encontraban lenguajes como Cobol o Ensamblador cuya programación es más extensa y compleja que la actual para realizar una simple tarea.

Por otro lado se encuentra la interfaz que se muestra al usuario final, la cual también busca ser más amigable y de fácil uso.

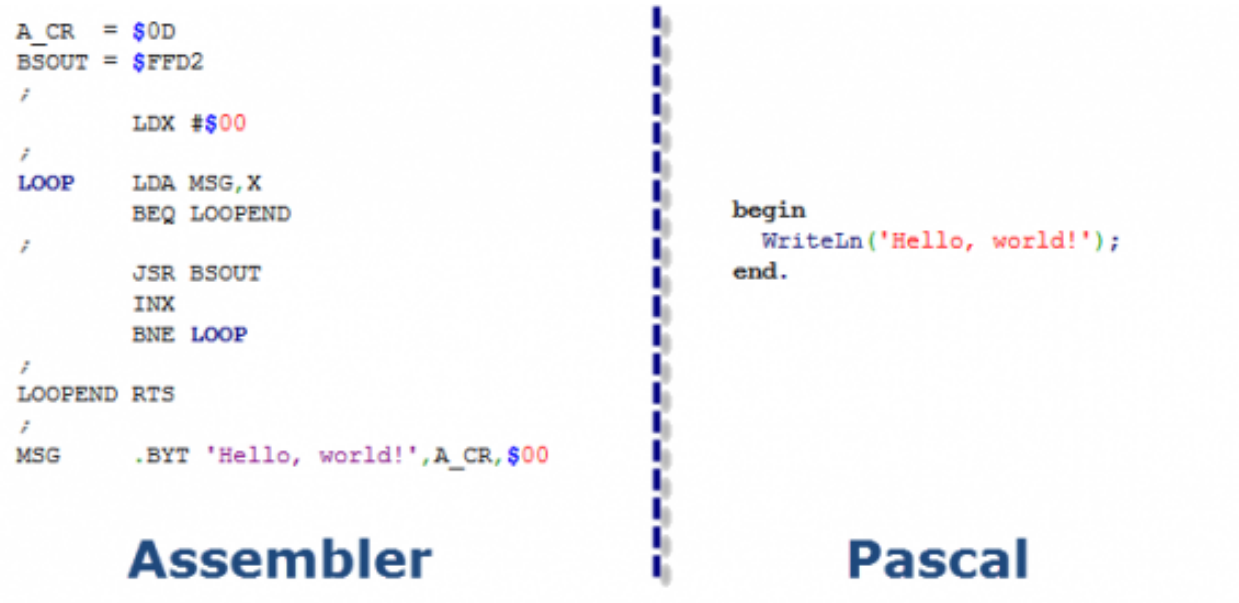


Figura 2. Ejemplo “Hola mundo” en Ensamblador y Pascal

Fuente: Elaboración propia,

A continuación se detalla más temas relacionados con el desarrollo de *software*.

2.2.1. Personas que participan en el desarrollo de un sistema

En el desarrollo de aplicaciones existen varias personas involucradas, las cuales podemos clasificar en cinco grandes grupos:

2.2.1.1. Propietarios

Son aquellas personas que patrocinan y promueven los sistemas de información. Entre las funciones de los propietarios está fijar el presupuesto y los plazos para el desarrollo y mantenimiento de los sistemas de información, y dar el visto bueno al sistema de información final.

En función del tamaño del sistema de información que se intenta desarrollar, los propietarios de sistemas pueden pertenecer a distintos niveles jerárquicos dentro de la organización. En el desarrollo de los sistemas más grandes, los propietarios son directivos que están situados en lo más alto de la jerarquía de la compañía. En el desarrollo de sistemas de tamaño medio, los propietarios suelen ser directivos medios o ejecutivos, mientras que en sistemas más pequeños es bastante común encontrar directivos medios y supervisores como propietarios de sistemas.

Debido a la distancia existente entre los propietarios de sistemas y el desarrollo y mantenimiento de los sistemas de información, los propietarios suelen participar en el proyecto en términos muy generales, sin entrar en detalle (Fernández, 2006, p. 16).

Zachmann (1987) expone que una de las claves en el éxito de cualquier proyecto de sistemas de información es el compromiso de los propietarios del sistema en su desarrollo. La implicación de los propietarios de sistemas favorece la creación de un compromiso por parte de los subordinados hacia el proyecto, así como hacia su éxito.

2.2.1.2. Usuarios

Según la Real Academia Española (RAE) (2017) un usuario es: “Que usa algo” y si partimos de esta definición enfocándolo al área de sistemas de información, son todas aquellas personas que utilizan los sistemas de información de una forma regular para manejar información con un fin. Entre todos los grupos de individuos que participan en el desarrollo de un sistema de información, los usuarios son el de mayor volumen de personas.

Así mismo, los usuarios deben ser considerados como el grupo de individuos más importantes en el desarrollo de sistemas de información, ya que serán los que tendrán que trabajar diariamente sobre él, y decidirán si cumple con las necesidades que tiene el negocio. Por tanto, es necesario el compromiso de los usuarios de sistemas para tratar de definir bien lo que necesitan, con sus restricciones, y también identificar de forma oportuna los problemas que se vayan encontrando durante el desarrollo del sistema.

Los sistemas de información pueden ser utilizados por una gran cantidad de individuos con objetivos y necesidades muy diversas. Por ese motivo puede ser interesante agrupar a los usuarios de sistemas en grupos y subgrupos en función de la relación con la empresa. Para empezar se puede distinguir entre usuarios internos a la organización y usuarios externos a ella.

Los usuarios internos son todas aquellas personas que están desarrollando el sistema de información, y que en la mayoría de ocasiones son los destinatarios principales del nuevo sistema.

El segundo grupo lo conforman los usuarios externos a la organización que pueden ser clasificados a su vez como clientes, proveedores, aliados y finalmente, trabajadores externos (Fernández, 2006, p. 17).

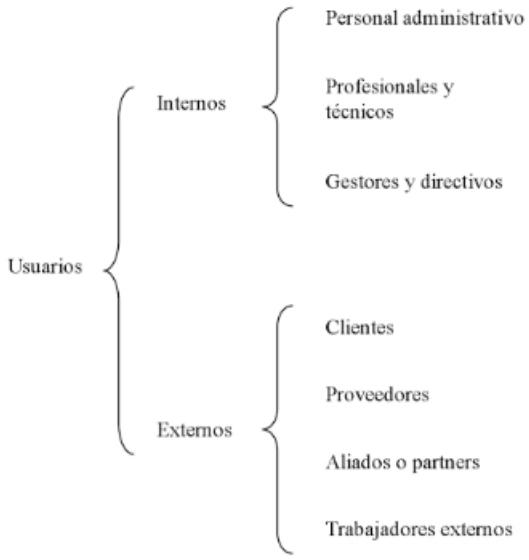


Figura 3. Tipología de usuarios de sistemas

Fuente: Fernández, 2006.

2.2.1.3. Programadores

A partir de especificaciones de diseño de soft*w*are, los programadores escriben, verifican y arreglan instrucciones detalladas que configuran programas de computación. La ejecución de estos en equipos de computación produce los resultados que constituyen los objetivos del sistema. También conciben, diseñan y verifican estructuras lógicas para resolver problemas mediante computadores. Estos programadores pueden provenir de diversos niveles de educación formal o informal, aunque buena parte de ellos proviene de educación superior incompleta, y a veces se los identifica por la tecnología que emplean. Una formación posible en el marco de la educación media es la de Técnicos en Programación de Computadores (Instituto Nacional de Educación Tecnológica, 2006, p. 2).

2.2.1.4. Analistas

Los analistas de sistemas informáticos adaptan y diseñan sistemas de información para ayudar a las empresas trabajar de forma más rápida y eficiente. Trabajan en estrecha colaboración con personal de todas las categorías para averiguar los problemas que surjan en el sistema existente, y para cumplir con las expectativas del cliente a la hora de crear un nuevo sistema. Los analistas producen una especificación para un sistema que satisfaga las necesidades de la empresa (Educaweb, s.f.).

2.2.1.5. Project Manager

Los *Project Manager* son considerados agentes de cambio, todas las metas del proyecto las convierten en propias y usan sus capacidades y experiencia para alentar a su equipo de trabajo, además se caracterizan por su capacidad para trabajar bajo presión y se sienten cómodos bajos los cambios y la complejidad sobre ambientes dinámicos, permitiendo de esta manera delegar funciones sobre su equipo de trabajo y que el proyecto sea exitoso (PMI, 2017).

2.2.2. Ciclo de vida de desarrollo de sistemas

El ciclo de vida de desarrollo de sistemas es una guía para todas aquellas personas que se dedican al área de desarrollo en la informática, donde se define, en una serie de pasos estructurados, cómo es la construcción de una aplicación. Estas etapas tienen un orden lógico y cada una de ellas cumple una función muy importante; pueden variar de un autor a otro, pero en su esencia buscan llegar a un mismo fin.

Según Kendall y Kendall (2005), el ciclo de vida de desarrollo de sistemas lo podemos definir de la siguiente manera:

> El ciclo de vida de vida del desarrollo de sistemas (SDLC, Systems Development life cycle) es un enfoque por fases para el análisis y el diseño cuya premisa principal consiste en que los sistemas se desarrollan mejor utilizando un ciclo especifico de actividades del analista y el usuario. (p. 10)

Según Berzal (2005), las etapas del ciclo de vida de sistemas son un reflejo del proceso que se sigue para resolver cualquier tipo de problema. Desde hace muchos años atrás se veía una estructura básica de lo que conocemos hoy como el ciclo de vida de desarrollo de sistemas, como lo describe George Polya en 1945 en su libro “How to solve it” con las siguientes etapas:

- Comprender el problema.
- Plantear una posible solución.
- Llevar a cabo la solución planteada.
- Comprobar que el resultado obtenido sea el correcto.

Las etapas que aparecieron posteriormente responden a procesos más relacionados con el desarrollo de sistemas informáticos y la industria. (p. 3)

Además, todas estas etapas adicionales pueden variar en número dependiendo del autor, a pesar de que la finalidad es la misma.

2.2.2.1. Etapas del ciclo de vida de desarrollo de sistemas

Cáceres (2006) menciona las etapas del ciclo de vida de desarrollo de sistemas de la siguiente manera:

- Identificación de problemas, oportunidades y objetivos, donde se busca cumplir con varios puntos: los planes, la viabilidad del proyecto, los problemas, las oportunidades, las normas y el propósito del sistema, logrando así obtener una estructura definida con lo que se desea realizar.
- Estudiar y analizar el sistema donde los analistas estudian los problemas, oportunidades y normas que enmarquen los procesos, hechos y limitaciones con el fin de tener un mayor panorama sobre lo que se va a realizar.
- Definir las necesidades de los usuarios y establecer prioridades con el fin de establecer un sistema funcional y que satisfaga todas las necesidades del usuario.
- Diseño del sistema recomendado, donde se evalúa si existen soluciones alternativas y especificar cuál sería la solución informática en caso de ser necesario, tomando en cuenta la viabilidad técnica, operativa y económica, luego adquirir el equipo y programas necesarios para, finalmente, diseñar e integrar el nuevo sistema.
- Desarrollo del sistema partiendo de toda la información recopilada, donde también se realice la respectiva documentación del sistema para facilitar su entendimiento de parte de otras personas.
- Pruebas y correcciones del sistema con ayuda de los usuarios expertos para recibir retroalimentación y asegurar la calidad del proyecto.

- Implementación y evaluación del sistema asegurando que funcione en un ambiente adecuado y de la forma óptima.

2.2.3. Arquitectura para el desarrollo de sistemas

Según Moreno (2014) las arquitecturas para el desarrollo de sistemas nacen debido a la complejidad reciente de las aplicaciones, los sistemas distribuidos y los sistemas abiertos basados en componentes, donde se busca comprender y mejorar la estructura de las aplicaciones complejas, reutilizar dicha estructura o parte de ella para resolver problemas similares, planificar la evolución de la aplicación identificando las partes mutables e inmutables de esta, así como los costos de los posibles cambios, analizar la corrección de la aplicación y su grado de cumplimiento respecto a los requisitos iniciales y permitir el estudio de algunas propiedades específicas de dominio.

En otras palabras, una arquitectura de desarrollo de sistemas lo que busca es facilitar el desarrollo de una aplicación, ya que se define cómo se debe manejar los datos, cómo se deben manejar los componentes, cómo es la interacción del sistema y posibles soluciones para problemas similares. Todo esto agiliza el trabajo de los desarrolladores ya que se cuenta con muchas cosas ya definidas, además en caso de ser necesario realizar algún mantenimiento en un sistema, es más fácil para cualquier persona dar una solución eficiente ya que todo se maneja sobre un mismo estándar, facilitando la comprensión de lo que se hizo en su momento y corregirlo.

A la hora de definir una arquitectura es necesario tomar en cuenta varios puntos de vista, ya que las diferentes personas que participan en un proyecto tienen distintas

expectativas, enfocadas en las necesidades de esa persona; a continuación se muestra un cuadro que resume las necesidades de estos actores:

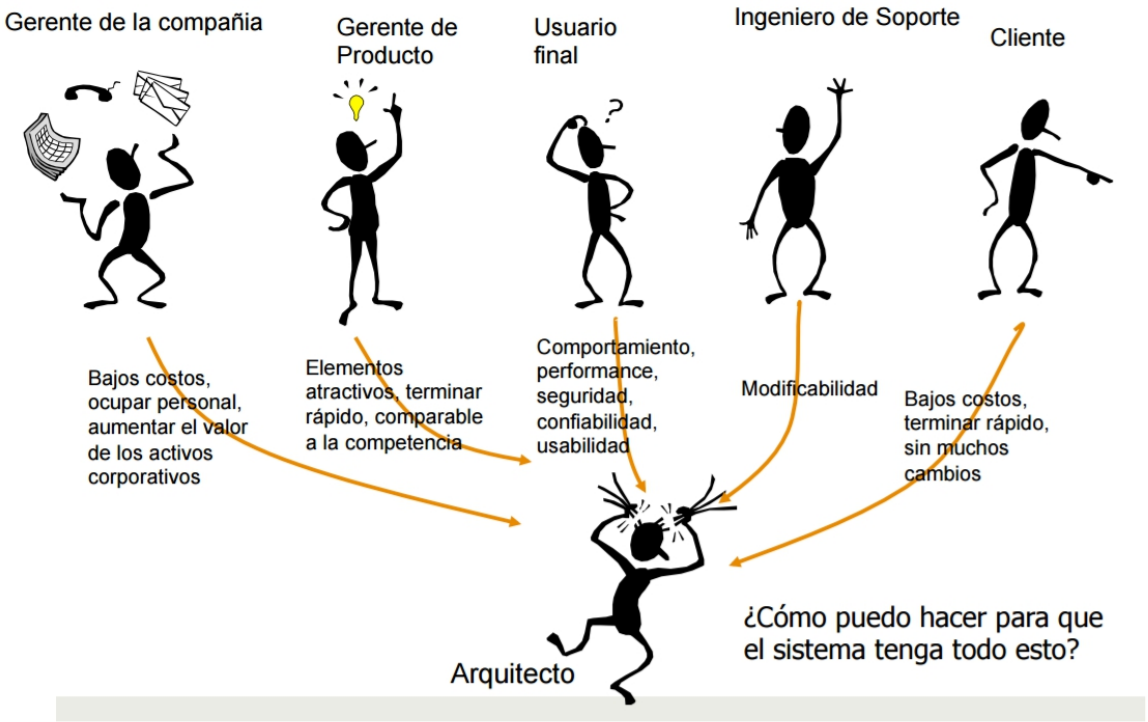


Figura 4. Componentes de una arquitectura de *software*

Fuente: Barraza, s.f.

Además se debe considerar el tipo de sistema por desarrollar, dentro de los cuales podemos encontrar:

- Aplicaciones monoprocesadoras: son todas aquellas que se ejecutan en una sola máquina y esta no se comunica con otras aplicaciones, por ejemplo, un procesador de texto.
- Aplicaciones embebidas: se ejecutan en un entorno computarizado especial y requiere diseño de *hardware* y *software*, por ejemplo, un teléfono móvil.

- Aplicaciones de tiempo real: tiene entre sus especificaciones requerimientos temporales y son de naturaleza reactiva, por ejemplo, el *software* de un radar.
- Aplicaciones distribuidas: se ejecutan en múltiples procesadores y requieren acceso a internet para poder comunicarse entre sí, por ejemplo, un sistema de votaciones.

2.2.4. Scrum

Es un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90. Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos. Scrum muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo, de modo que podamos mejorar.

El marco de trabajo Scrum consiste en los Equipos Scrum, roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso. Las reglas de Scrum relacionan los eventos, roles y artefactos, gobernando las relaciones e interacciones entre ellos (Schwaber & Sutherland, 2013, p. 4).

2.2.4.1. El equipo de trabajo

Dentro de la metodología de trabajo de Scrum se encuentran definidos ciertos roles o grupos de personas, donde, cada una de ellas cumple un papel especifico.

2.2.4.1.1. Propietario del producto

El propietario del producto (*product owner*) es quien toma las decisiones del cliente. Su responsabilidad es el valor del producto.

Para simplificar la comunicación y toma de decisiones es necesario que este rol recaiga en una única persona. Si el cliente es una organización grande, o con varios departamentos, puede adoptar la forma de comunicación interna que considere oportuna, pero en el equipo de desarrollo solo se integra una persona en representación del cliente, y esta debe tener el conocimiento suficiente del producto y las atribuciones necesarias para tomar las decisiones que le corresponden (Scrum Manager, 2016).

2.2.4.1.2. Equipo de desarrollo

Grupo de personas que de manera conjunta desarrollan el producto del proyecto. Tienen un objetivo común, comparten la responsabilidad del trabajo que realizan (así como de su calidad) en cada iteración y en el proyecto.

El tamaño del equipo es entre 5 y 9 personas. Por debajo de 5 personas cualquier imprevisto o interrupción sobre un miembro del equipo compromete seriamente el compromiso que han adquirido y, por tanto, el resultado que se va a entregar al cliente al finalizar la iteración. Por encima de 9 personas, la comunicación y colaboración entre todos los miembros se hace más difícil y se forma subgrupos (Proyectos agiles, s.f.).

2.2.4.1.3. Scrum master

Según Garzás (2014), el Scrum Master es responsable de asegurar que se cumplen los valores y las prácticas de Scrum, se asegura de que la velocidad del equipo no se vea frenada por problemas en el proceso ágil y de que los problemas, obstáculos, impedimentos, se resuelvan (no siempre de resolverlos él, pero sí de preocuparse porque se resuelvan).

Este rol también es conocido como *coach* del equipo. También hay quien conoce al Scrum Master como el "protector del equipo y del proceso ágil". De ahí, obviamente, que debe tener amplia formación en agilidad y actualizarse frecuentemente.

Además el Scrum Master elimina obstáculos que dificultan al equipo para lograr el objetivo del *sprint*. El Scrum Master potencia la productividad.

2.2.4.2. Eventos de Scrum

En Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (*time-boxes*), de tal modo que todos tienen una duración máxima y se detallan a continuación.

2.2.4.2.1. *Sprint*

Según Rouse (2015), un *sprint* es un periodo de tiempo definido durante el cual un grupo de tareas deben ser completadas y estar listas para ser revisadas al final de este.

Cada *sprint* inicia con una reunión de planificación, en la cual se llega a un acuerdo sobre cuáles tareas se van a llevar a cabo durante el *sprint*.

2.2.4.2.2. Reunión de planificación

Según OBS Business School (2016), la reunión de planificación de un *sprint* es un evento de tiempo variable. Para un *sprint* de un mes tiene ocho horas de duración. Para *sprints* más cortos, el evento es proporcionalmente más corto. Por ejemplo, para un *sprint* de dos semanas, las reuniones de planificación de *sprint* son de cuatro horas de duración.

En esta reunión se define la funcionalidad en el incremento planeado y cómo el Equipo de Desarrollo creará este incremento y la salida de este trabajo es definir el Objetivo del *sprint*.

La reunión de planificación de *sprint* tradicionalmente consta de dos partes, cada una de la mitad de tiempo de duración de la Reunión de Planificación, respondiendo a las siguientes dos preguntas:

- ¿Qué va a ser entregado en el incremento resultante del próximo *sprint*?
- ¿Cómo se va a realizar el trabajo seleccionado?

2.2.4.2.3. Scrum diario

Según Casanova (2015), la reunión diaria, como su propio nombre indica, consiste en reunirse todos los días el equipo Scrum completo, para que cada miembro secuencialmente hable durante 2-3 minutos y responda a estas 3 preguntas:

- ¿Qué hiciste ayer
- ¿Qué harás hoy?
- ¿Hay algún impedimento?

Los objetivos que se busca al responder a estas tres preguntas son los siguientes:

- Compartir con el equipo el compromiso para avanzar hacia el objetivo del *sprint*.
- Tomar decisiones coordinadas entre todos para eliminar obstáculos que impidan llegar a ese objetivo.

2.2.4.2.4. Revisión del *sprint*

La reunión de revisión de *sprint* es un evento de un máximo de cuatro horas de duración para *sprints* de un mes y proporcionalmente menor en *sprints* más cortos.

Durante esta reunión se inspeccionan los elementos del *Product Backlog* incluidos en el *sprint* para valorar si cumplen su definición de completado.

El dueño del producto es el encargado de decidir si un ítem cumple o no con la definición de completado y puede requerir la ayuda del equipo de desarrollo para

valorar cuestiones técnicas como, por ejemplo, el nivel de cobertura de pruebas unitarias del proyecto.

Los elementos del *Product Backlog* pueden estar completos o no, pero en ningún caso podrán estar parcialmente completos. Aquellos elementos que no hayan sido completamente terminados, o que presenten deficiencias, volverán al *Product Backlog* para ser estimados y priorizados de nuevo.

En la práctica suele ocurrir que la mayoría de elementos no completados en el *sprint* que termina sean puestos en la cima de la pila del *sprint* que empieza, pues suelen ser tareas bien definidas y a las que en muchas ocasiones solo les restan unas pocas horas de dedicación; no obstante, esto no se cumple siempre o no nos interesa siempre, por ello es conveniente que vuelvan al *Backlog* y sean replanteadas (Barba, 2015).

2.2.4.2.5. Retrospectiva del *sprint*

La reunión retrospectiva tiene lugar el último día del *sprint*, tras la reunión de revisión del *sprint*. En esta reunión, el equipo examina y explora su funcionamiento en los procesos de Scrum. En función de este análisis, el equipo podría tomar la decisión de adaptar sus procesos para mejorar su propia eficacia, productividad, calidad y satisfacción. Esta reunión y las mejoras resultantes son fundamentales para lograr una organización ágil y autónoma.

Si el equipo no completó todos los casos de usuario asignados al *sprint*, en la reunión retrospectiva se abordarán los motivos. El equipo determinará si puede adaptar

sus procesos de modo que la probabilidad de que surjan esos problemas sea menor. También deben abordarse los problemas que afectaron la eficacia global del equipo, la productividad, la calidad y el grado de satisfacción del grupo con respecto al proyecto (Microsoft, s.f.).

2.2.4.3. Herramientas de Scrum

2.2.4.3.1. Lista de objetivos (*Product Backlog*)

Proyectos Ágiles (s.f.) explica esta herramienta como una lista que representa la visión y expectativas del cliente respecto de los objetivos y entregas del producto o proyecto. El cliente es el responsable de crear y gestionar la lista (con la ayuda del Facilitador y del equipo, quien proporciona el coste estimado de completar cada requisito).

- Contiene los objetivos/requisitos de alto nivel del producto o proyecto, que se suelen expresar en forma de historias de usuario. La lista está priorizada balanceando el valor que cada requisito aporta al negocio frente al coste estimado que tiene su desarrollo, es decir, basándose en el Retorno de la Inversión (ROI).
- En la lista se indican las posibles iteraciones y las entregas esperadas por el cliente (los puntos en los cuales desea que se le entreguen los objetivos/requisitos completados hasta ese momento), en función de la velocidad de desarrollo del equipo involucrado en el proyecto. Es conveniente que el contenido de cada iteración tenga una coherencia, de manera que se reduzca el esfuerzo de completar todos sus objetivos.

- La lista también debe considerar los riesgos del proyecto e incluir los requisitos o tareas necesarios para mitigarlos.

Área de requisitos	Requisitos	Origen	Valor	Estimación inicial	Factor Ajuste	Estimación ajustada	Iteración:	1	2	3	4	5
							Pendiente:	225	170	114	57	
Área X	Requisito A	Marketing	2000	15		15		15	0			
Área Z	Requisito B	Producción	1750	20		20		20	0			
Área Y	Requisito C	Ventas	1500	20		20		20	0			
	Iteración 1		5250	55		55		55	0	0	0	0
Área Z	Requisito C	Producción	1250	15	0,2	18		18	18	0		
Área X	Requisito D	Producción	1250	20		20		20	20	0		
Área Z	Requisito E	Marketing	1000	15	0,2	18		18	18	0		
	Iteración 2		3500	50		56		56	56	0	0	0
	Primera entrega		8750	105		111		111	56	0	0	0
Área X	Requisito F	Marketing	1250	20	0,2	24		24	24	24	0	
Área Y	Requisito G	Marketing	750	15		15		15	15	15	0	
Área Y	Requisito H	Ventas	750	15	0,2	18		18	18	18	0	
	Iteración 3		2750	50		57		57	57	57	0	0
Área Z	Requisito I	Producción	700	15	0,2	18		18	18	18	18	
Área Y	Requisito J	Marketing	500	10	0,5	15		15	15	15	15	
Área Y	Requisito K	Ventas	500	20	0,2	24		24	24	24	24	
	Iteración 4		1700	45		57		57	57	57	57	0
	Segunda entrega		4450	95		114		114	114	114	57	0

Figura 5. Ejemplo de lista de objetivos

Fuente: Proyectos Ágiles, s.f.

2.2.4.3.2. Tablero Kanban

El tablero Kanban es una herramienta que permite controlar el avance de los *sprints*, viendo de una forma más gráfica el avance de todos los involucrados con el proyecto.

Por lo general cuenta con 3 secciones: “por hacer”, donde se ubican inicialmente todas las tareas del *sprint;* “en ejecución”, donde se encuentran las tareas que están siendo ejecutadas actualmente y “finalizadas”, donde se colocan todas las tareas terminadas.

2.2.4.3.3. Historias de usuario

Según Scrum Manager (2014), es una descripción de una funcionalidad que debe incorporar un sistema de *software*, y cuya implementación aporta valor al cliente. La estructura de una historia de usuario está formada por:

- Nombre breve y descriptivo.
- Descripción de la funcionalidad en forma de diálogo o monólogo del usuario, describiendo la funcionalidad que desea realizar.
- Criterio de validación y verificación que determinará para considerar terminado y aceptable por el cliente el desarrollo de la funcionalidad descrita.
- Información que resulte necesaria por el modelo de implementación: Prioridad, riesgo, tamaño, etc.

2.3. LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación es un conjunto de órdenes e instrucciones que se dan al ordenador para que resuelva un problema o ejecute una determinada misión. En los primeros tiempos de la informática, la programación se efectuaba en el único lenguaje que entiende el microprocesador: su propio código binario, también denominado lenguaje máquina o código máquina.

Pero la programación en lenguaje máquina resulta muy lenta y tediosa, pues los datos e instrucciones se deben introducir en sistema binario y, además, obliga a conocer las posiciones de memoria donde se almacenan los datos. Debido a esto, cometer errores a la hora de desarrollar una aplicación es muy común, y a la hora de

revisar el código toma una gran cantidad de tiempo debido al gran número de líneas de código (Trigo, s.f.).

Con el paso del tiempo la tecnología fue avanzando y los lenguajes de programación también, siendo cada vez más sencillos y fáciles de aprender, dejaron de ser de bajo nivel (lenguaje de máquina), cambiando a una estructura que se adapta más al pensamiento del ser humano (alto nivel) donde se incorpora también un poco más el diseño gráfico, dejando de lado solo las líneas de código.

La cantidad de lenguajes de programación que existen en la actualidad es muy extensa, tanto que se puede comparar con los lenguajes humanos, aunque existen algunos que sobresalen y son utilizados por un gran número de personas.

2.3.1. ASP .NET

Según Microsoft (2017), ASP.NET es un modelo de desarrollo web unificado que incluye los servicios necesarios para crear aplicaciones web empresariales con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el *Common Language Runtime* (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del *Common Language Runtime*, seguridad de tipos, herencia, etc. Dentro de las características que podemos encontrar están las siguientes:

- Marco de trabajo de página y controles
- Compilador de ASP.NET
- Infraestructura de seguridad
- Funciones de administración de estado
- Configuración de la aplicación
- Supervisión de estado y características de rendimiento
- Capacidad de depuración
- Marco de trabajo de servicios web XML
- Entorno de *host* extensible y administración del ciclo de vida de las aplicaciones
- Entorno de diseñador extensible

Además para trabajar con una aplicación web ASP.NET, se debe utilizar un explorador para realizar solicitudes al servidor web que aloja la aplicación. Las aplicaciones web ASP.NET se alojan normalmente utilizando *Internet Information Services* (IIS) como servidor web, a excepción de cuando ejecutamos la aplicación de manera local en el equipo, mediante la aplicación Visual Studio, el cual se encarga de realizar la simulación sin necesidad de instalar IIS.

2.3.1.1. Ventajas

Según Bien (2009), ASP .Net es un lenguaje del cual se pueden obtener grandes beneficios cuando ya se tienen otros productos que pertenecen a la misma empresa (Microsoft), como por ejemplo el sistema operativo, ya que su IDE para desarrollo (Visual Studio) fue creado para este sistema originalmente, por otro lado se puede integrar con otros productos de esta empresa como Sharepoint, Exchange y el paquete de Office.

Entre de las ventajas para este lenguaje podemos encontrar:

- Se consiguen todos los productos de un mismo vendedor (Microsoft).
- El desarrollo de aplicaciones es rápido.
- La mayoría de herramientas y soluciones se encuentran en *Microsoft Developer Network* (MSDN).
- Cuenta con un servidor web llamado *Internet Information Services* (IIS). para el control de versiones en el desarrollo de aplicaciones.

2.3.1.2. Desventajas

En http://www.lawebdelprogramador.com/foros/ASP/381741-Ventajas-y-desventajas-del-asp.html (2004) se indica que dentro de las desventajas para este lenguaje, están las siguientes:

- No es 100% multiplataforma.
- Requiere de un servidor de Microsoft (IIS).
- Las licencias para utilizar el IDE Visual Studio tienen un costo.

2.3.1.3. Costos

El costo para desarrollar en ASP .Net depende mucho de la necesidad o del ambiente donde se va a implementar, para empezar se debe tomar en cuenta el costo de las licencias para Visual Studio, donde la versión Enterprise ronda los 3 millones de colones según la tienda de Microsoft, la cual proporciona un conjunto completo de recursos para crear, desplegar y administrar aplicaciones en plataformas como Windows, iOS, Android y Linux, acceso a casi todos los programas de servidor de

Microsoft anteriores o actuales, créditos Azure para desarrollo y pruebas en la nube, cursos técnicos y más, sin costo adicional.

Por otro lado está el servicio de hosting o almacenamiento web de la aplicación, el cual, según SmarterAsp (2017), tiene un costo mensual entre 1500 colones y 5000 colones, dependiendo de las necesidades o cantidad de aplicaciones.

2.3.1.4. Manejo de versiones

Visual Studio Team Services y *Team Foundation Server* (TFS) proporcionan dos modelos de control de versiones: Git, que es el control de versiones distribuido y *Team Foundation Version Control (TFVC)* , que es el control de versiones centralizado.

2.3.1.4.1. Git (distribuido)

Según Microsoft (2017) es un sistema de control de versiones distribuido. Cada desarrollador tiene una copia del repositorio de origen en su máquina. Los desarrolladores pueden subir cada conjunto de cambios desde su máquina y realizar operaciones de control de versiones, como el historial, y comparar sin una conexión de red. Cuando necesite cambiar de contexto, puede crear un *host* local privado. Puede cambiar rápidamente de una rama a otra para pivotar entre diferentes variaciones de su base de código. Posteriormente, puede fusionar, publicar o disponer de la sucursal.

2.3.1.4.2. TFVC (centralizado)

Según Microsoft (2017) es un sistema centralizado de control de versiones. Normalmente, los miembros del equipo solo tienen una versión de cada archivo en sus

máquinas. Los datos históricos se mantienen únicamente en el servidor. Las sucursales se basan en rutas de acceso y se crean en el servidor.

TFVC tiene dos modelos de flujo de trabajo:

- Espacios de trabajo del servidor : Antes de realizar cambios, los miembros del equipo descargan públicamente los archivos. La mayoría de las operaciones requieren que los desarrolladores se conecten al servidor. Este sistema facilita el bloqueo de los flujos de trabajo.
- Espacios de trabajo locales: Cada miembro del equipo toma una copia de la última versión de la base de código con ellos y funciona sin conexión según sea necesario. Los desarrolladores verifican sus cambios y resuelven los conflictos según sea requerido.

2.3.2. Java

Este lenguaje fue creado inicialmente por Sun Microsystems Inc. en 1990, como un proyecto para desarrollar un sistema que controlara electrodomésticos, se pretendía crear un *hardware* polivalente, con un sistema operativo eficiente (SunOS) y un lenguaje de desarrollo denominado Oak (roble), pero este proyecto finalizó en 1992 debido a su alto costo en relación con otros proyectos del mismo tipo que ya existían, siendo declarado como un fracaso.

En 1995 el proyecto fue retomado y gracias a una acertada decisión de distribuir libremente el lenguaje por Internet, se popularizó, lográndose que una gran cantidad de programadores lo depuraran y terminaran de perfilar según sus necesidades.

El nombre de este lenguaje tuvo que ser cambiado ya que existía otro lenguaje llamado Oak. Se buscaba un nombre que evocara la esencia de la tecnología (viveza, animación, rapidez, interactividad…). Java fue elegido de entre muchísimas propuestas. No es un acrónimo, sino más bien es un nombre inspirado en algo que a muchos programadores les gusta beber en grandes cantidades: una taza de café (Java en argot inglés americano). De esta forma, Sun lanzó las primeras versiones de Java a principios de 1995.

Debido a la explosión tecnológica de estos últimos años, Java ha desarrollado soluciones personalizadas para cada ámbito tecnológico, agrupando cada uno de esos ámbitos en una edición distinta: J2SE (*Java 2 Stantard Edition*), J2EE (*Java 2 Enterprise Edition*), J2ME (*Java 2 Micro Edition*) (López, 2008).

2.3.2.1. Kit de desarrollo de Java (JDK)

La mayoría de los lenguajes de programación se caracterizan por ser interpretados o compilados, lo que determina cómo serán ejecutados en una computadora. Java tiene la característica de ser al mismo tiempo compilado e interpretado. El compilador es el encargado de convertir el código fuente de un programa en un código intermedio llamado *bytecode,* que es independiente de la plataforma en que se trabaje y que es ejecutado por el intérprete de Java que forma parte de la Máquina Virtual de Java (*Java Virtual Machine*).

Es por eso que se dice que Java tiene la característica de ser multiplataforma, pue s el mismo *bytecode* (o archivo *.class), es interpretado por una máquina virtual

diferente y adecuada para cada plataforma, es decir que lo que cambia no es el archivo compilado, sino la máquina virtual que lo va a interpretar; a diferencia de C, por ejemplo, que el código fuente tiene que ser compilado para cada plataforma (Universidad de Chile, s.f.).

2.3.2.2. Fases de creación y ejecución de programas en Java

Según Berzal (2009), las fases de creación y ejecución se realizan de la siguiente manera:

- La primera fase es el editor, en donde se crea un programa con la ayuda de un editor, el cual se almacena en un fichero con la extensión .java.
- La segunda fase es el compilador, el compilador lee el archivo .java y se detecta si hubo errores sintácticos, posteriormente se generan los *bytecodes* que se almacenan en ficheros .class.
- La tercera etapa es el cargador de clases, se leen los ficheros .class y los *bytecodes* pasan de disco a memoria principal.
- La cuarta etapa es el verificador de *bytecodes*, donde se comprueba que todos los *bytecodes* sean válidos y que no violen las restricciones de seguridad de la Máquina Virtual de Java.
- La quinta etapa es el intérprete de *bytecodes*, la Máquina Virtual de Java (JVM) lee los *bytecodes* y los traduce al lenguaje que el ordenador entiende.

2.3.2.3. Características

Según Sun Microsystems, Java cuenta con las siguientes características:

- Sencillo: Elimina la complejidad de los lenguajes como C y da paso al contexto de los lenguajes modernos orientados a objetos. Aunque la sintaxis de Java es muy similar a C y C++, que son lenguajes a los que una gran mayoría de programadores están acostumbrados a emplear.
- Orientado objetos: La filosofía de programación orientada a objetos es diferente a la programación convencional (imperativa o procedural). Su nivel de abstracción facilita la creación y mantenimiento de programas. Existen muchas referencias que dan una introducción a esta forma de programar.
- Independiente a la arquitectura y portable. Al compilar un programa en Java, el código resultante es un tipo de código binario conocido como Java *bytecodes*. Este código es interpretado por diferentes computadoras de igual manera, por lo que únicamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura de ordenador específica. Como el código compilado de Java es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- Robusto: Java simplifica la gestión de la memoria dinámica. Por ejemplo, ya no es necesaria la liberación explícita, el intérprete de Java la lleva a cabo automáticamente cuando detecta que una variable dinámica ya no es usada por el programa. Por otra parte, impide que un puntero Java apunte a una dirección de memoria no válida, los punteros (referencias) Java son seguros y deterministas: o bien apuntan a un elemento correctamente alojado en memoria o bien tienen el valor nulo. Finalmente, el acceso a la memoria es supervisado por el intérprete de tal

manera que no es posible acceder a zonas de memoria no autorizadas sin provocar un error. Por ejemplo, no es posible escribir fuera de los límites de un vector.

- Seguro: El sistema de Java tiene ciertas políticas que evitan que se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los denominados *applets*, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- Multitarea: Un lenguaje que soporta múltiples *threads*, hilos o tareas, es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo. El soporte y la programación de hilos en Java está integrado en la propia sintaxis del lenguaje.
- Dinámico: En Java no es necesario cargar completamente el programa en memoria, sino que las clases compiladas pueden ser cargadas bajo demanda en tiempo de ejecución (*dynamic binding*). Esto proceso permite la carga de código bajo demanda.

2.3.2.1. Desventajas

Pérez (2015) indica que la sintaxis de este lenguaje es bastante compleja en comparación con otros lenguajes, siendo esta la mayor y casi única desventaja que encuentran las personas para este lenguaje, otro punto que se podría considerar como desventaja es la necesidad de tener el JDK instalado en la máquina donde se va a utilizar, ya que se necesita el intérprete de la aplicación para poderse ejecutar.

2.3.2.2. Costos

Los costos para desarrollar en Java solo se reflejan cuando queremos alguna función adicional o especifica que no viene con el paquete principal, como por ejemplo *Java*

Service Wrapper, según Tanuki Software (2017), esta herramienta permite que Java se ejecute como un servicio Windows, monitoreando el estado de la aplicación y el JVM, con un costo que empieza a partir del millón de colones.

2.3.2.3. Manejo de versiones

Para Java existen muchas formas para manejar versiones de código, cada una de ellas depende del IDE que se esté utilizando, por ejemplo Eclipse cuenta con una herramienta llamada Subclipse, la cual podemos descargar y configurar, otra opción es mediante Netbeans, donde Wong (2012) nos explica que necesitamos dos aplicaciones, una llamada VisualSVN, herramienta sobre la cual creamos el repositorio de datos, y TortoiseSVN, la cual se encarga de subir o descargar los archivos desde el repositorio.

2.3.2.4. *Java Server Faces* (JSF)

Según la Universidad de Alicante (2014), JSF es un *framework* MVC (Modelo-Vista-Controlador) basado en el API de Servlets que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el *framework* Facelets. Facelets se define en la especificación 2 de JSF como un elemento fundamental de JSF que proporciona características de plantillas y de creación de componentes compuestos. Antes de la especificación actual se utilizaba JSP para componer las páginas JSF.

Este *framework* utiliza las páginas Facelets como vista, objetos Javabean como modelos y métodos de esos objetos como controladores. El servlet FacesServlet realiza toda la tediosa tarea de procesar las peticiones HTTP, obtener los datos de

entrada, validarlos y convertirlos, colocarlos en los objetos del modelo, invocar las acciones del controlador y *renderizar* la respuesta utilizando el árbol de componentes.

2.3.2.5. *Primefaces*

Según Anghel (2016), Primefaces es un conjunto completo de más de 100 componentes JSF UI compatibles con HTML5, los cuales poseen gran apariencia y soportan capacidades AJAX, además se pueden configurar de forma *responsive* y son compatibles con cualquier navegador o dispositivos.

Dicho conjunto de objetos ofrece una alternativa diferente para todos los desarrolladores que trabajen en JSF, la cual es muy potente y que además cuenta con soporte completo de parte de los creadores.

CAPÍTULO III: MARCO METODOLÓGICO

3.1. CONSIDERACIONES DEL PROYECTO

Este proyecto cuenta con una finalidad aplicada y naturaleza cuantitativa, donde se busca dar una base para la creación de una arquitectura robusta, mediante la propuesta de estándares y una metodología de desarrollo, buscando comparar, mediante el primer módulo del “Sistema de Órdenes de Producción”, la velocidad de desarrollo en comparación con la metodología actual de la UCR en el lenguaje ASP .net.

3.2. FUENTES Y SUJETOS DE INFORMACIÓN

3.2.1. Fuentes primarias

Las fuentes de esta tesis serán principalmente las siguientes:

- Trabajadores del Centro de Informática de la Universidad de Costa Rica, quienes en su mayoría llevaron un curso de JAVA impartido por un ente externo.
- Libro “Java: A Beginner's Guide, Sixth Edition” del autor Herbert Schild.
- Libro “Learn Java in One Day and Learn It Well (Learn Coding Fast) (Volume 4)”, del autor Jamie Chan.

3.2.2. Fuentes secundarias

Se tomará las siguientes fuentes:

- Revistas científicas
- Tesis
- Información obtenida de internet.

3.2.3. Sujetos de información

Tabla 2. Sujetos de información

Puesto laboral o descripción general	Profesión u oficio	Experiencia	Relación con el tema
Director	Ingeniero en sistemas	30 años	Director de proyectos en la UCR
Arquitecto de sistemas	Ingeniero en sistemas	10 años	Arquitecto en la UCR
Analista de sistemas	Ingeniera en sistemas	7 años	Analista de sistemas a cargo del proyecto de Órdenes de Producción
Desarrollador	Ingeniero en sistemas	3 años	Desarrollador en la UCR

Fuente: Elaboración propia.

3.3. TÉCNICAS Y HERRAMIENTAS

Para poder cumplir con los objetivos planteados en la investigación, se requiere la aplicación de instrumentos de recolección de la información que será utilizada para el análisis de la situación por estudiar y el planteamiento de soluciones.

La encuesta se puede definir como: "… serie de preguntas que están dirigidas a una porción representativa de una población, y tiene como finalidad averiguar estados de opinión, actitudes o comportamientos de las personas ante asuntos específicos" (Significados, s.f.).

Para esta investigación se aplicará las siguientes herramientas:

- Una encuesta estructurada con 11 preguntas cerradas para conocer la percepción de los trabajadores respecto al cambio de lenguaje en la institución y medir su conocimiento en el lenguaje Java (ver apéndice 1).
- Investigación bibliográfica con el fin de ver ejemplos de cómo desarrollan en Java otras personas y tomar ideas que se puedan aplicar al proyecto.
- Reuniones con los compañeros de trabajo con el fin de ir presentando los avances en la metodología y estándares para tomar la opinión y retroalimentación de cada uno.

3.4. VARIABLES DE INVESTIGACIÓN

Tabla 3. Variables de investigación

Objetivos específicos	Variables asociadas	Descripción
Comparar el lenguaje actual “ASP .Net” con la nueva propuesta “*Java Primefaces*”	ASP .Net Java Ventajas Limitaciones Características Costos	Crear un documento donde se compare ambos lenguajes para identificar las diferencias entre uno y otro, además de sus ventajas.
Crear estándares de desarrollo para el lenguaje JAVA	Estándares Bases de datos Código	Crear un documento donde se explique la propuesta de estándares y métodos de desarrollo sobre el lenguaje Java.
Proponer metodologías de desarrollo mediante un proyecto base	Proyecto base Base de datos Código	Crear un proyecto base donde se utilicen los métodos y estándares propuestos con el fin de

Objetivos específicos	Variables asociadas	Descripción
		verificar la funcionalidad
Implementar los estándares y metodologías de desarrollo en el proyecto “Sistema de órdenes de producción”, sobre el primer módulo del sistema	*Java Primefaces* Órdenes de producción Metodología propuesta Estándares propuestos	Desarrollo del primer módulo del sistema de Órdenes de Producción en lenguaje Java, haciendo uso de los estándares y métodos de desarrollo propuestos

Fuente: Elaboración propia.

3.5. DISEÑO DE INVESTIGACIÓN

3.5.1. Etapa 1

En la etapa 1 se realiza una revisión de la situación actual de la Universidad de Costa Rica en el manejo de estándares y metodologías de desarrollo en lenguaje ASP .net.

3.5.2. Etapa 2

En la etapa 2 se identifica cuáles son los puntos de mejora sobre el proceso de desarrollo actual para aplicarlos sobre la nueva propuesta en lenguaje JAVA.

3.5.3. Etapa 3

En la etapa 3 se va a elaborar un proyecto base o prototipo donde se hará uso de la propuesta de estándares de programación, tomando ejemplos encontrados en la web de técnicas de programación sobre este lenguaje, con el fin de ver el funcionamiento en

una aplicación sencilla de todos los componentes en conjunto, y hacer una mayor investigación sobre ciertos puntos donde se encuentren deficiencias en caso de ser necesario.

3.5.4. Etapa 4

En la etapa 4 se va a realizar una implementación de conocimientos en el primer módulo del “Sistema de órdenes de producción”, donde se va a hacer uso de la metodología y estándares propuestos.

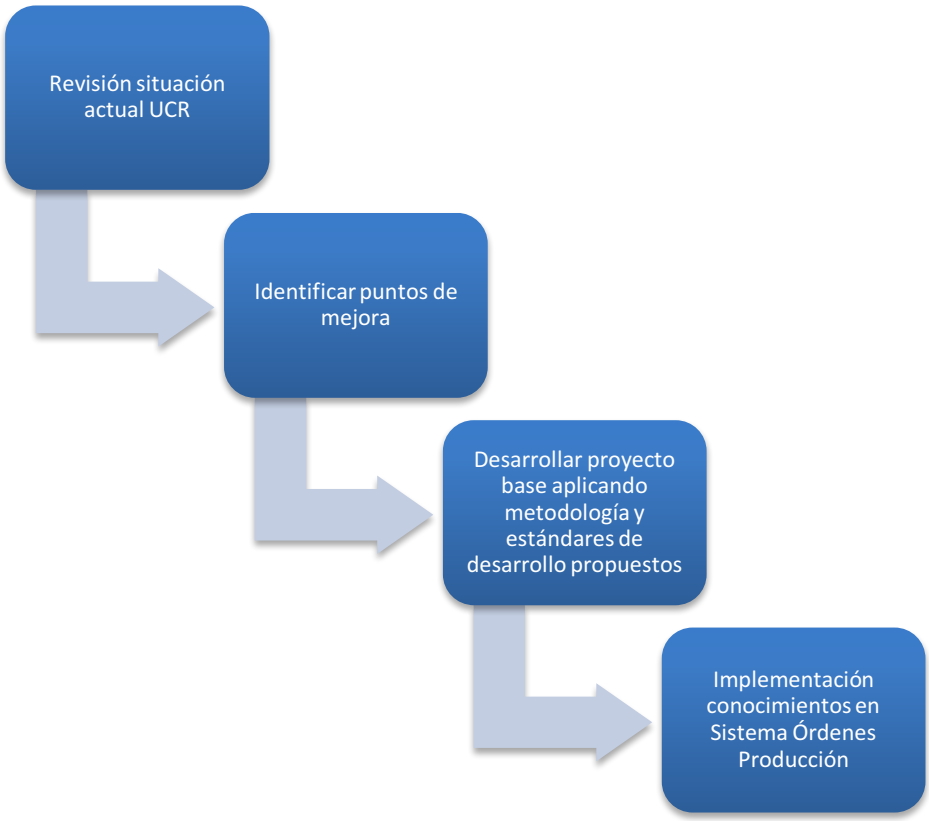


Figura 6. Diseño de investigación

Fuente: Elaboración propia.

CAPÍTULO IV: DIAGNÓSTICO

4.1. DIAGNÓSTICO SITUACIÓN ACTUAL

En esta sección se explica a modo muy general la situación actual de la UCR en la forma como se maneja el desarrollo de sistemas en lenguaje ASP .net, con el fin de identificar posibles puntos de mejora e incorporarlos en la nueva propuesta.

Actualmente el equipo de desarrollo cuenta con una arquitectura establecida para el lenguaje ASP .net, donde se define el manejo de estándares a la hora de crear un objeto de base de datos y en el código de cada sistema, también se cuenta con formas ya establecidas para implementar ciertas funcionabilidades básicas a la hora de programar, por ejemplo, el manejo de listados de información, reportes de sistema, entre otros. Finalmente se cuenta con 3 ambientes (desarrollo, pruebas y producción) y un equipo de control de calidad, el cual se encarga de aprobar cada uno de los requerimientos antes de ser mostrados al usuario interesado en el sistema.

El esquema de trabajo actual cuenta con un director de proyectos, un equipo de analistas de sistemas y un equipo de desarrolladores, dependiendo de la magnitud de cada proyecto el director asigna uno o dos analistas, quienes se encargan de realizar las reuniones con el usuario final con el fin de hacer un levantamiento de requerimientos. Una vez completado este proceso de la misma forma se asigna una cantidad determinada de desarrolladores y los analistas proceden a repartir un grupo de tareas o requerimientos a cada desarrollador, buscando cubrir una o dos semanas de trabajo.

Luego de haber iniciado el desarrollo de los requerimientos, el analista consulta de manera aleatoria el estado de los requerimientos al desarrollador para ir midiendo el avance e ir asignando más carga de trabajo.

Durante el periodo de desarrollo, cada desarrollador puede consultar al analista de sistemas o al arquitecto de sistemas dudas de programación o respecto a la definición de cada requerimiento, repitiendo este ciclo hasta terminar el proyecto.

Figura 7. Situación actual para desarrollar en la UCR

Fuente: Elaboración propia.

4.2. DIAGNÓSTICO ADMINISTRATIVO U OPERATIVO

El proceso de desarrollo actual no cuenta con documentación, pero sí es conocido por todos los trabajadores, con el fin de todos seguir con la misma metodología y además se encuentra muy bien definido.

Todos los sistemas que se implementan por el área de informática son desarrollos internos para la misma institución y para el nacimiento de todo nuevo proyecto se debe seguir un proceso.

Primero se procede a plantear la idea en las reuniones anuales entre directores de diferentes áreas de la institución, definiendo así la cartera de proyectos para el nuevo año que empieza. Una vez que alguno de estos proyectos ha sido asignado al equipo de trabajo, se procede a realizar un levantamiento de requerimientos de parte del analista de sistemas que haya sido asignado.

Cuando ya se tiene una cantidad avanzada de todos los requerimientos del sistema se procede a iniciar con el desarrollo del proyecto, donde se solicita al arquitecto de sistemas crear la base de datos, según las necesidades del analista, y crear el proyecto base, a partir del cual los desarrolladores inician su trabajo.

Debido a que ya se cuenta con estándares y métodos de desarrollo muy bien definidos en el lenguaje ASP .net, es muy fácil llevar un control de cómo va el avance del proyecto, además de la facilidad para llevar a cabo cada uno de los requerimientos.

Todo este trabajo se realiza sobre el programa Visual Studio, manejando un control de versiones mediante el TFS (*Team Foundation Server*) e inicialmente en un ambiente de desarrollo. Esto con el fin de llevar un mejor control y respaldo de lo que se vaya creando, además de facilitar la integración en caso de ser más de un desarrollador participando en el mismo proyecto.

Una vez que un requerimiento ha sido finalizado por el desarrollador, probado por él y probado por el analista encargado, se procede a hacer un pase al ambiente de pruebas.

Para realizar este tipo de pases entre ambientes, se encuentra una persona designada, la cual dos veces por semana atiende todas las solicitudes de los desarrolladores o analistas para hacer este proceso.

Una vez que el código deseado se encuentra en el ambiente de pruebas, un equipo de control de calidad y el usuario experto o usuario final de cada sistema se encarga nuevamente de realizar las pruebas respectivas, con el fin de dar el visto bueno o dar al equipo de desarrollo la retroalimentación necesaria para mejorar la herramienta, con el fin de lograr lo requerido y finalmente hacer un pase a producción.

Cuando ya se ha finalizado una gran cantidad o todos los requerimientos del sistema, es decisión del usuario experto, en coordinación del director de su área, decidir cuándo habilitar el sistema para uso público a todos los usuarios interesados.

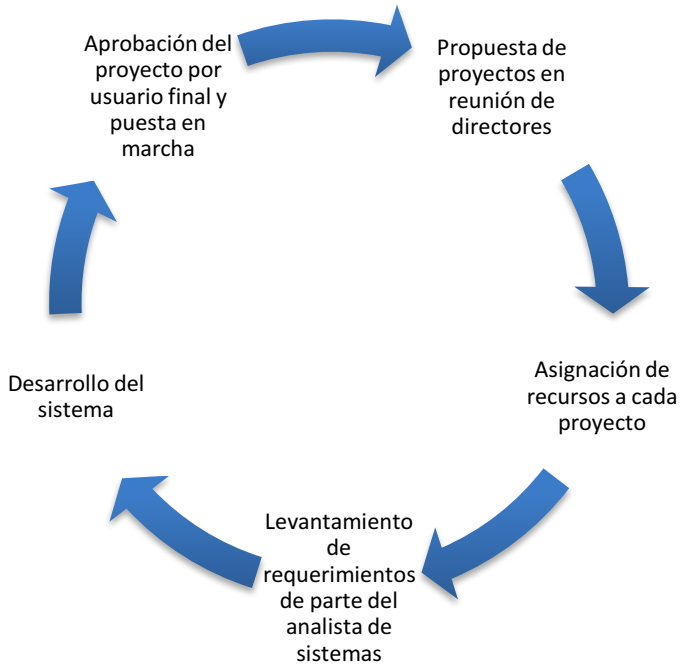


Figura 8. Ciclo de desarrollo sistemas en la UCR

Fuente: Elaboración propia.

A pesar de que este proceso se encuentra muy bien definido, existen ciertos contratiempos o problemas con el cambio de lenguaje para los usuarios, algunos de ellos son:

- Atrasos en los tiempos estimados para cada uno de los proyectos que inicien en desarrollo.
- Función parcial de algún requerimiento del usuario, debido al desconocimiento del lenguaje por parte de los involucrados en el desarrollo del sistema.

4.3. DIAGNÓSTICO TÉCNICO

Según González (2017), la Universidad de Costa Rica cuenta con 13 servidores físicos y 14 servidores virtuales distribuidos entre los 3 ambientes de desarrollo de sistemas (desarrollo, pruebas y producción) de la siguiente manera según su rol:

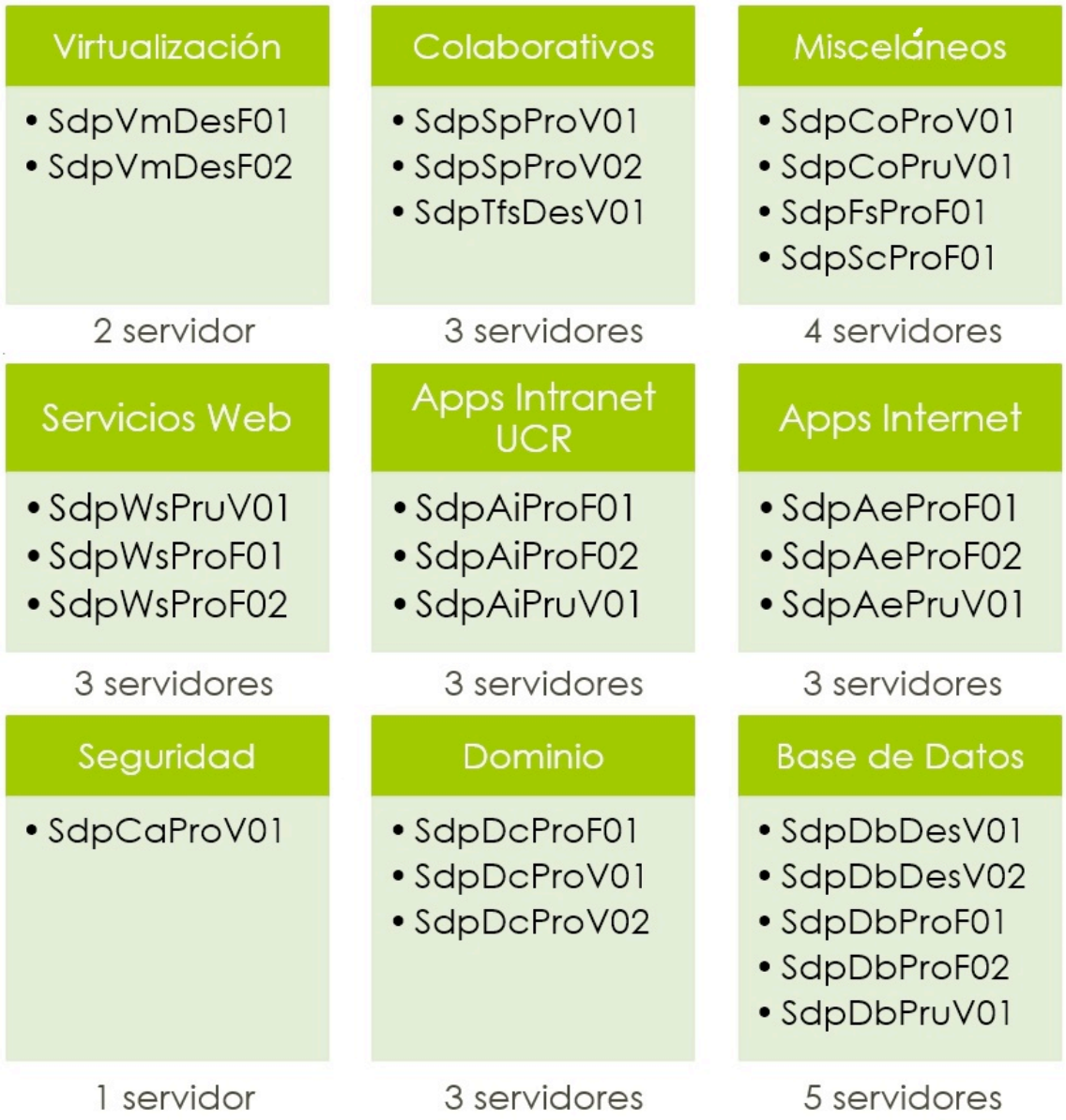


Figura 9. Distribución de servidores según su rol en la UCR

Fuente: González, 2017.

El nombre de estos servidores se encuentra conformado por los siguientes términos:

-Sdp = Sección de proyectos	-Ca = Servidor de Seguridad
-Dc = Servidor de Dominio	-Db = Servidor de Base de datos
-Ws = Servidor de Servicios web	-Ai = Servidor de App de intranet
-Ae = Servidor de App de internet	-Vm = Servidor de Virtualización
-Sp / Tfs = Servidor de Colaborativos	-Co / Fs / Sc = Servidor de Misceláneos
-Pro = Ambiente de producción	-Pru = Ambiente de pruebas
-Des = Ambiente de desarrollo	-V = Servidor virtual
-F = Servidor físico	

Además, estos equipos cuentan con las siguientes especificaciones:

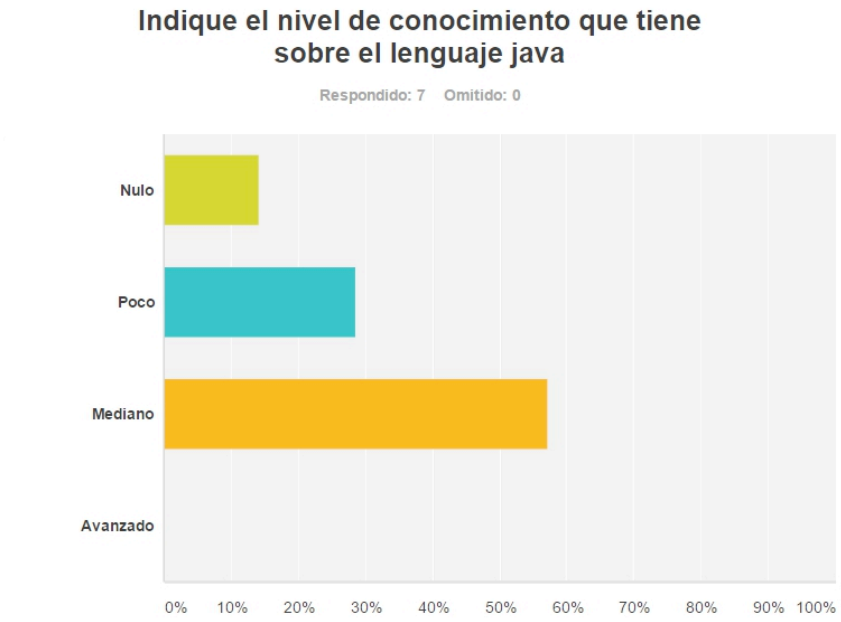
- 2 Xeon X5647
- 16 GB de RAM
- 3 HD Raid 5 (1 TB)
- Windows Server 2008 R2
- Oracle 11g

Por otro lado, a nivel de programas se utiliza IIS 10.0, ASP .net 4.5 y Visual Studio 2012 para el desarrollo de aplicaciones en la institución.

Dichos servidores están actualmente equipados para todos los sistemas que se desarrollan en ASP .net, la problemática que se detecta es que se necesita acondicionar los servidores al lenguaje Java, de forma que se pueda seguir un proceso parecido al que se maneja actualmente con el otro lenguaje.

4.4. DIAGNÓSTICO DE PERCEPCIÓN

Se realizó una encuesta acerca la situación actual sobre el lenguaje Java, en algunos miembros del equipo de desarrollo de manera selectiva, en la cual participó el director de proyectos, e arquitecto de sistemas, 4 analistas y un desarrollador; los resultados obtenidos fueron los siguientes:

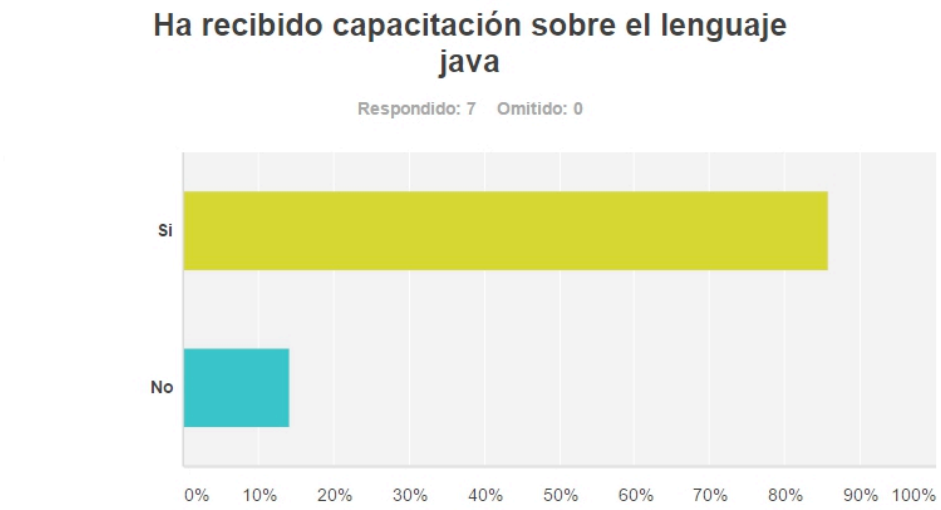


Opciones de respuesta	Respuestas	
Nulo	14,29%	1
Poco	28,57%	2
Mediano	57,14%	4
Avanzado	0,00%	0
Total		7

Figura 10. Nivel conocimiento en Java

Fuente: Elaboración propia.

De acuerdo con figura 6, la mayor parte de los trabajadores del Centro de Informática tiene conocimientos básicos en Java, facilitando que se adapten a la nueva plataforma.

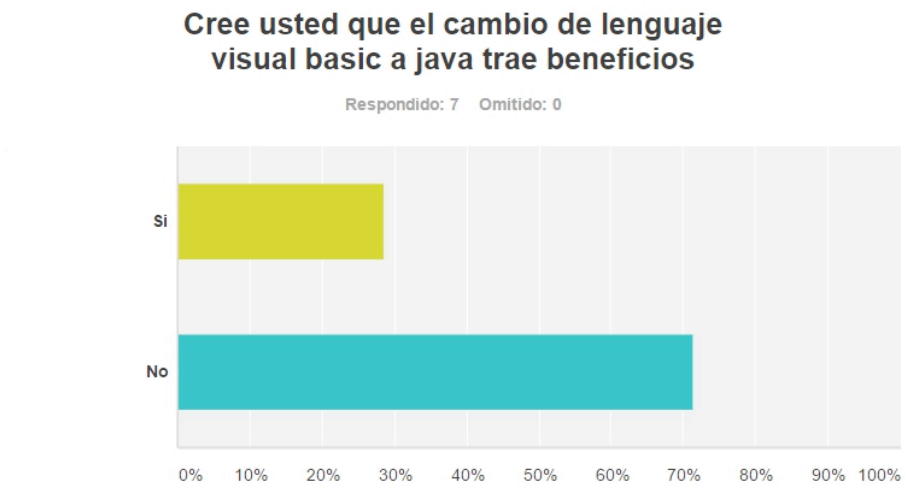


Opciones de respuesta	Respuestas	
Si	85,71%	6
No	14,29%	1
Total		7

Figura 11. Capacitación en Java

Fuente: Elaboración propia.

De conformidad con la Figura 7, solo una persona no ha recibido ninguna capacitación sobre el lenguaje Java, siendo esta la que reciba mayor afectación con el cambio.

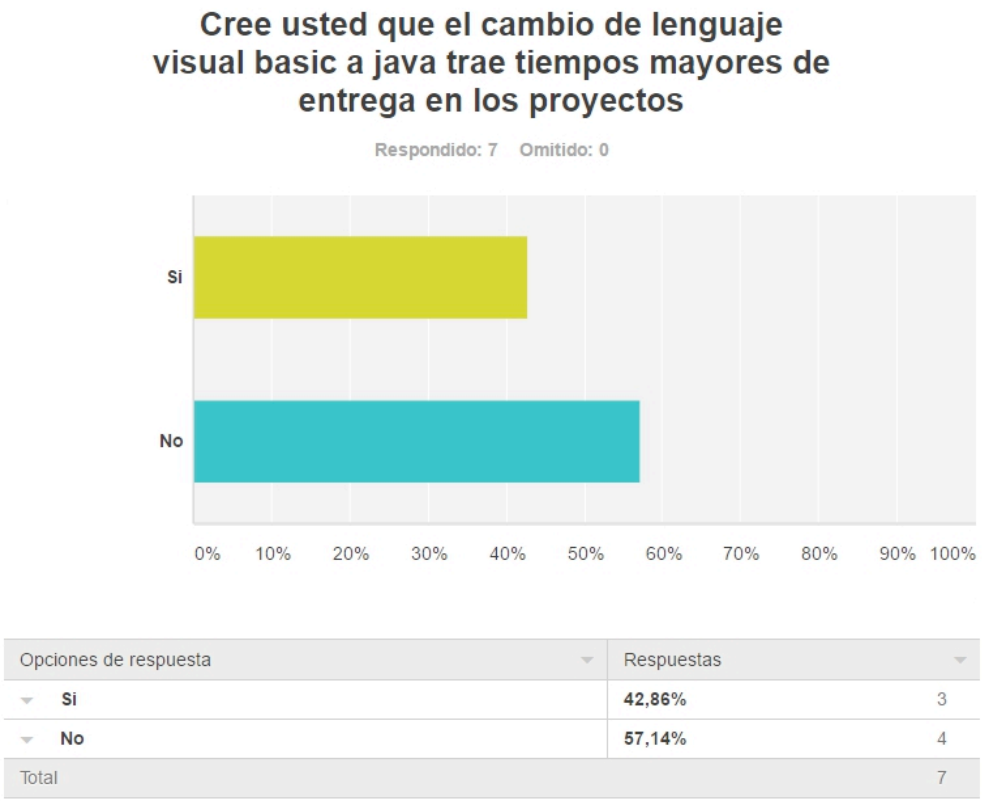


Opciones de respuesta	Respuestas	
Si	28,57%	2
No	71,43%	5
Total		7

Figura 12. Beneficios en Java

Fuente: Elaboración propia.

En la Figura 8 se observa que la mayoría de las personas encuestadas no ven un beneficio con el cambio de plataforma, mostrando una resistencia al cambio.

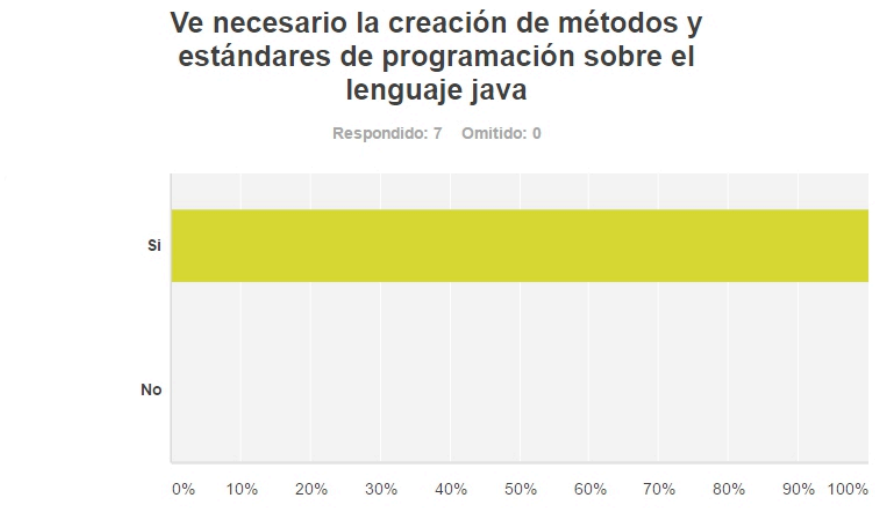


Opciones de respuesta	Respuestas	
Si	**42,86%**	3
No	**57,14%**	4
Total		7

Figura 13. Tiempo entrega de proyectos en Java

Fuente: Elaboración propia.

En la Figura 9 se observan opiniones muy divididas en cuanto a los tiempos de entrega sobre nuevos proyectos desarrollados sobre la plataforma Java, donde la mayoría no cree que llegue a haber una afectación sobre este punto.

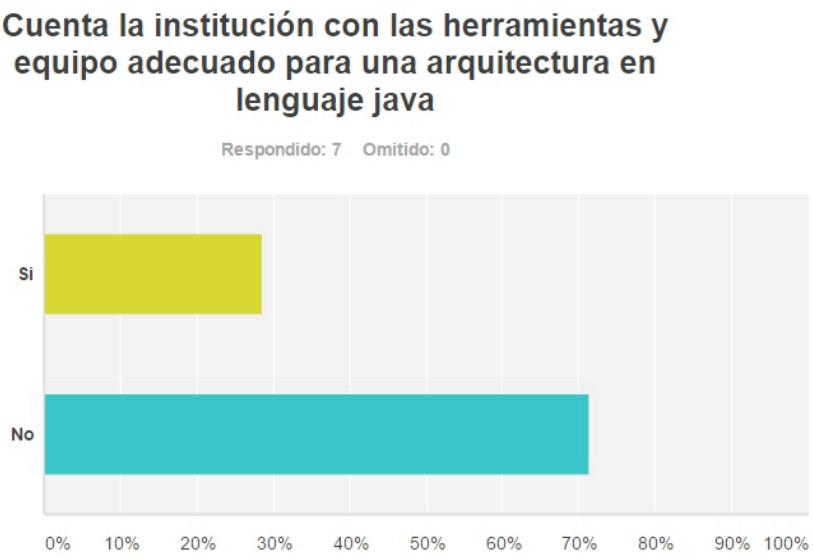


Opciones de respuesta	Respuestas	
Si	100,00%	7
No	0,00%	0
Total		7

Figura 14. Opinión sobre métodos y estándares en Java

Fuente: Elaboración propia.

En la Figura 10, el 100% de los encuestados ve necesaria la creación de métodos y estándares de programación, ya que esto facilita el desarrollo y posteriormente el mantenimiento de las aplicaciones.



Opciones de respuesta	Respuestas	
Si	28,57%	2
No	71,43%	5
Total		7

Figura 15. Capacidad de la institución sobre Java

Fuente: Elaboración propia.

En la Figura 11 se observa que la mayor parte de los encuestados considera que la UCR no cuenta aún con el equipo adecuado para empezar a trabajar sobre la nueva plataforma, y que hay ciertos puntos que deben estar primero concluidos para poder iniciar, por ejemplo, la creación de métodos y estándares de desarrollo.

4.5. BRECHAS O CONCLUSIONES DEL DIAGNÓSTICO

En conclusión, se puede mejorar la metodología actual estructurándola de una manera diferente, con el fin de manejar un mejor orden y control. Además, se requiere tomar en consideración los puntos que se detallan en el siguiente cuadro antes de iniciar el desarrollo en el Sistema de Órdenes de Producción.

Tabla 4. Brechas del diagnóstico

Situación actual	**Brecha**	**Situación deseada**
No se conoce los beneficios y desventajas del nuevo lenguaje	Se requiere conocer los beneficios y desventajas para sacar provecho al nuevo lenguaje	Conocimiento de los beneficios y desventajas del lenguaje Java
No existen estándares para el desarrollo de aplicaciones en Java	Se requiere crear estándares útiles para ser usados por todos los interesados	Estándares de desarrollo de aplicaciones en Java
No existen métodos de desarrollo para funciones básicas en Java	Se requiere crear métodos de desarrollo donde se integren funcionalidades básicas	Métodos de desarrollo en Java
No se cuenta con un prototipo base o la aplicación de métodos y estándares en lenguaje Java	Se requiere crear un prototipo para ver el funcionamiento de los	Prototipo base hecho en Java

Situación actual	Brecha	Situación deseada
	métodos y estándares en conjunto	

Fuente: Elaboración propia.

CAPÍTULO V: DISEÑO Y DESARROLLO DEL PROYECTO

5.1. REQUERIMIENTOS POR DESARROLLAR

- Estructurar de mejor manera el proceso de desarrollo con el fin de nivelar las cargas de manera equitativa a todos los desarrolladores mediante una nueva metodología.
- Aplicar estándares en la creación de objetos y métodos a nivel de código y nivel de base de datos.
- Crear un proyecto base aplicando la nueva propuesta, con el fin de ver la funcionalidad de un sistema desarrollado en Java.
- Desarrollar el primer módulo del Sistema de Órdenes de Producción en Java y comparar el tiempo de desarrollo con proyectos previamente desarrollados en ASP .net.

5.2. METODOLOGÍA DE DESARROLLO PROPUESTA

Para la metodología de desarrollo se propone una estructura basada en la metodología ágil Scrum, ya que en la UCR se está tratando de implementar este modelo y la mayoría de empleados directos de la institución cuentan con cursos de capacitación en esta metodología, con el fin de que los productos se entreguen en un menor tiempo de desarrollo, sin afectar la calidad.

Una de las razones por las cuales aún no se ha podido implementar este modelo en su totalidad es debido a que el equipo de desarrollo se encuentra conformado por trabajadores directos de la institución y trabajadores externos, los cuales no necesariamente conocen esta metodología de trabajo, además los clientes o interesados en los productos son los mismos funcionarios de esta institución, quienes

en la mayoría de los casos ocupan el cargo de directores de un departamento o jefes, y su disponibilidad para una reunión se ve afectada muchas veces debido a su función.

Por otro lado, existen ciertas plazas de trabajo ya establecidas dentro de la institución, las cuales no calzan con esta metodología de desarrollo, por lo que en esta propuesta se toman en consideración.

5.2.1. Equipo de trabajo

El equipo de trabajo se va a ver conformado por 5 diferentes actores, los cuales se detallan a continuación.

5.2.1.1. Director de proyectos

Es la persona a la cual se le asigna una cantidad determinada de proyectos, partiendo del criterio propio y experiencia se encarga de asignar los recursos para llevar a cabo cada uno de los proyectos propuestos.

5.2.1.2. Usuario experto

Es la persona que tiene el conocimiento de cómo debe funcionar un proyecto determinado, conociendo todas las necesidades del departamento o personas interesadas.

Dependiendo del tamaño del proyecto, esta persona se suele convocar a reuniones cuando ya se tiene un módulo terminado o hay un avance significativo de pantallas para poder realizar pruebas y dar el visto final a los requerimientos.

5.2.1.3. Analista de sistemas

Es la persona encargada de llevar a cabo todo el análisis de un proyecto, donde por medio de reuniones iniciales se establecen los requerimientos que van a conformar el sistema, los cuales se documentan de forma tal que el desarrollador entienda y los pueda elaborar, además se procede a diseñar un modelo de base de datos.

Una vez que el proyecto se encuentra en desarrollo, el analista se encarga de dar soporte o aclarar dudas a los desarrolladores y de gestionar las reuniones con el usuario experto en caso de ser necesario refrescar algún punto o mostrar productos terminados.

5.2.1.4. Arquitecto de sistemas

El arquitecto de sistemas lleva el control de todos los sistemas, velando por el buen funcionamiento de los existentes, genera nuevas propuestas o mejoras para los nuevos proyectos y se encarga de revisar junto con el analista los requerimientos y el modelo de base de datos que esta persona vaya creando para un nuevo proyecto, con el fin de retroalimentar al analista y asegurar que la calidad del producto sea la óptima.

Además el arquitecto acompaña también a los desarrolladores y analistas durante todo el proceso de creación de un sistema, con el objetivo de evacuar dudas que no se puedan resolver.

5.2.1.5. Desarrollador

Es la persona que se encarga de llevar a cabo todos los requerimientos de un sistema, bajo el apoyo de todo el equipo.

5.2.2. Estructura de la metodología

En cada proyecto que inicia, el analista debe realizar un listado de requerimientos junto con el usuario experto del sistema, ir enumerándolos en prioridad y agregándolos a un *Product Backlog*, los cuales, una vez terminados, deben ser aprobados por el arquitecto de sistemas.

Luego de este paso el analista de sistemas procede a crear una base de datos, primero en alguna aplicación para modelar, como por ejemplo Power Designer, la cual debe ser también aprobada por el arquitecto. Una vez finalizada su creación se procede ya a replicar en el motor de base de datos.

Una vez que se finaliza con estos primeros pasos el director de proyectos procede a asignar desarrolladores al proyecto y se inicia un ciclo, el cual se maneja en forma de *sprint* hasta que se finalice el proyecto, cada *sprint* va a tener una semana de duración y se encuentra conformado por la siguiente estructura:

5.2.2.1. Reunión de planificación

En esta reunión participa el director de proyectos, el analista de sistemas y los desarrolladores del proyecto y representa el inicio de un *sprint*, la extensión de dicha reunión depende de la cantidad de desarrolladores asignados del proyecto, el tiempo estimado por desarrollador para llevar a cabo una reunión de planificación es de 1 hora, en la cual se ve la lista de requerimientos por asignar y se estiman los tiempos de desarrollo por requerimiento.

Para definir la cantidad de requerimientos que va a tener un desarrollador, en el *sprint* se toma la cantidad de horas que va a laborar la persona como tope, el analista procede a ir explicando los requerimientos y estimando los posibles tiempos de desarrollo junto con el desarrollador para llenando ese total de horas.

Dentro del total de horas se debe descontar al menos 2 horas para atención de consultas del desarrollador y demás imprevistos durante la semana, además la duración de las reuniones diarias, reunión de planificación y reunión de cierre y retrospectiva o alguna otra actividad que afecte el *sprint*.

Para cada uno de los requerimientos asignados a los desarrolladores se crea un papel donde se coloca el nombre del requerimiento, tiempo estimado y posteriormente, al finalizar, el tiempo real. Además se crean otros papeles que representan subtareas para completar este requerimiento igual con su tiempo estimado y posteriormente su tiempo real.

La creación de estos papeles es con el fin de generar evidencia y pegarlos posteriormente en una tabla de Kanban.

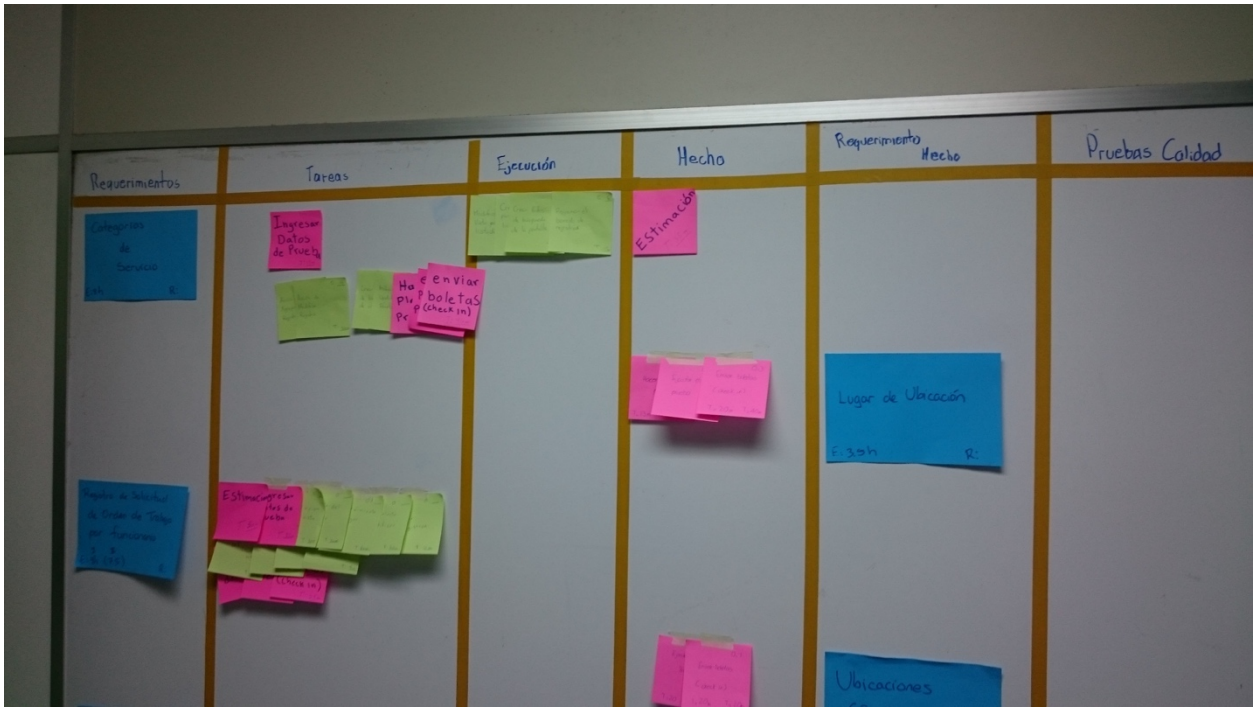


Figura 16. Ejemplo tablero Kanban

Fuente: Elaboración propia.

5.2.2.2. Reuniones diarias

Las reuniones diarias tienen una duración máxima de 5 minutos y se efectúan todos los días del *sprint*, participan el analista y el desarrollador y se ven los siguientes puntos:

- ¿Que se hizo ayer?
- ¿Qué se va a hacer hoy?
- ¿Qué problemas se han tenido?

5.2.2.3. Reunión de cierre y retrospectiva

Esta reunión tiene una duración aproximada de 4 horas, se ejecuta al finalizar el *sprint* y participa el director de proyectos, el analista y el desarrollador, donde se ve inicialmente los siguientes puntos, con el fin de ir mejorando el equipo de trabajo:

- ¿Qué se hizo bien?
- ¿Qué se hizo mal?
- ¿Qué se debería empezar a hacer?
- ¿Qué se debería dejar de hacer?
- ¿Qué se debería seguir haciendo?

Una vez que se ha llegado a un acuerdo se procede a hacer una revisión rápida de los requerimientos terminados, con el fin de determinar si se necesitan ajustes e ir incorporando todo el sistema.

5.2.3. Pruebas integrales

Cuando ya se tiene una gran cantidad de requerimientos desarrollados se procede a realizar una reunión de pruebas integrales, la cual tiene una duración aproximada de 4 horas, en la cual el analista, junto al desarrollador, realiza los distintos flujos del sistema con las pantallas que ya se tienen desarrolladas, con el fin de verificar el correcto funcionamiento del sistema y presentarle al usuario experto un producto más limpio y libre de errores.

5.2.4. Reunión con el usuario experto

Esta reunión tiene una duración aproximada de 4 horas y se lleva a cabo cuando ya se realizaron pruebas integrales exitosas, participa el usuario experto, el analista y en algunos casos el desarrollador, con el fin de mostrar el sistema al usuario experto y obtener el visto bueno de los requerimientos o retroalimentación en caso de ser necesario.

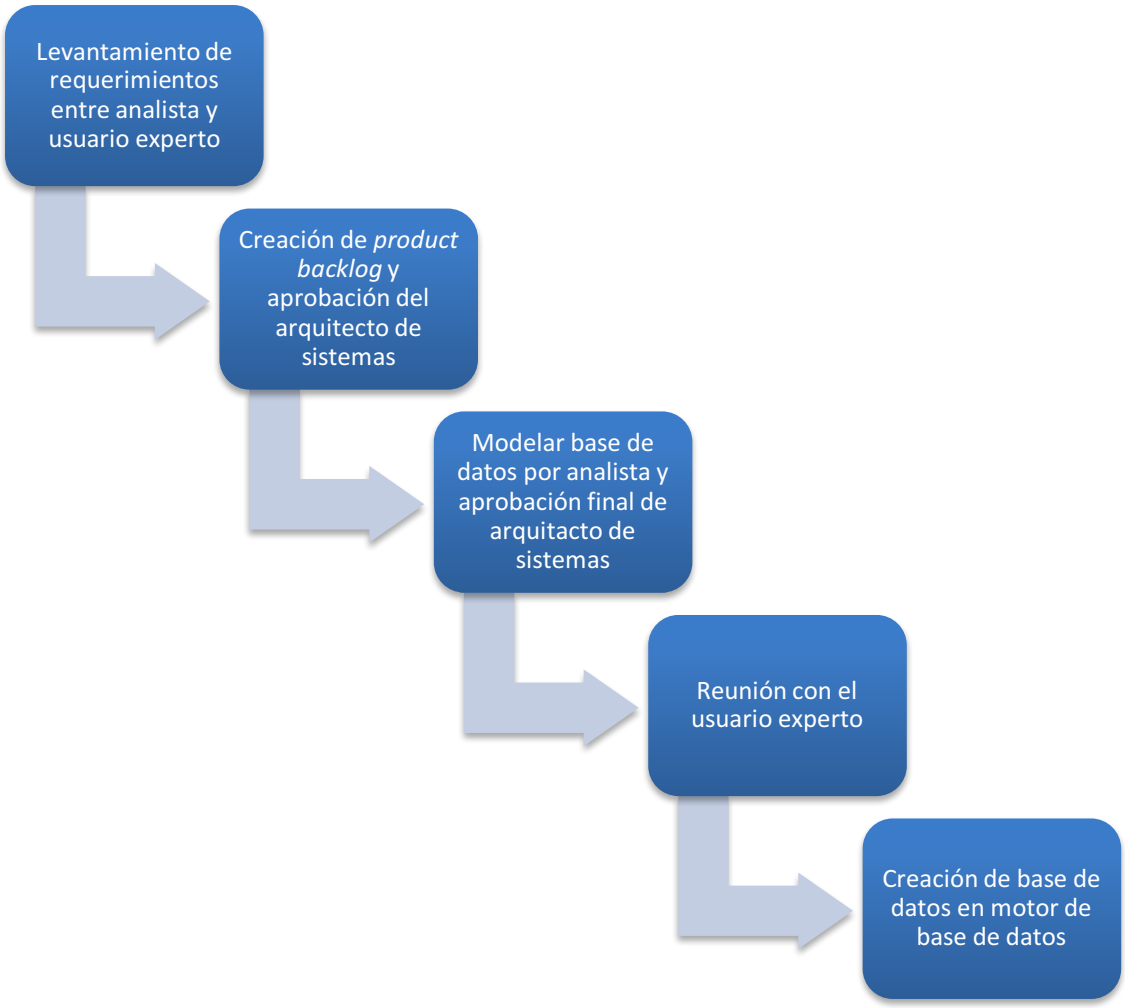


Figura 17. Cuadro 1 resumen metodología

Fuente: Elaboración propia.

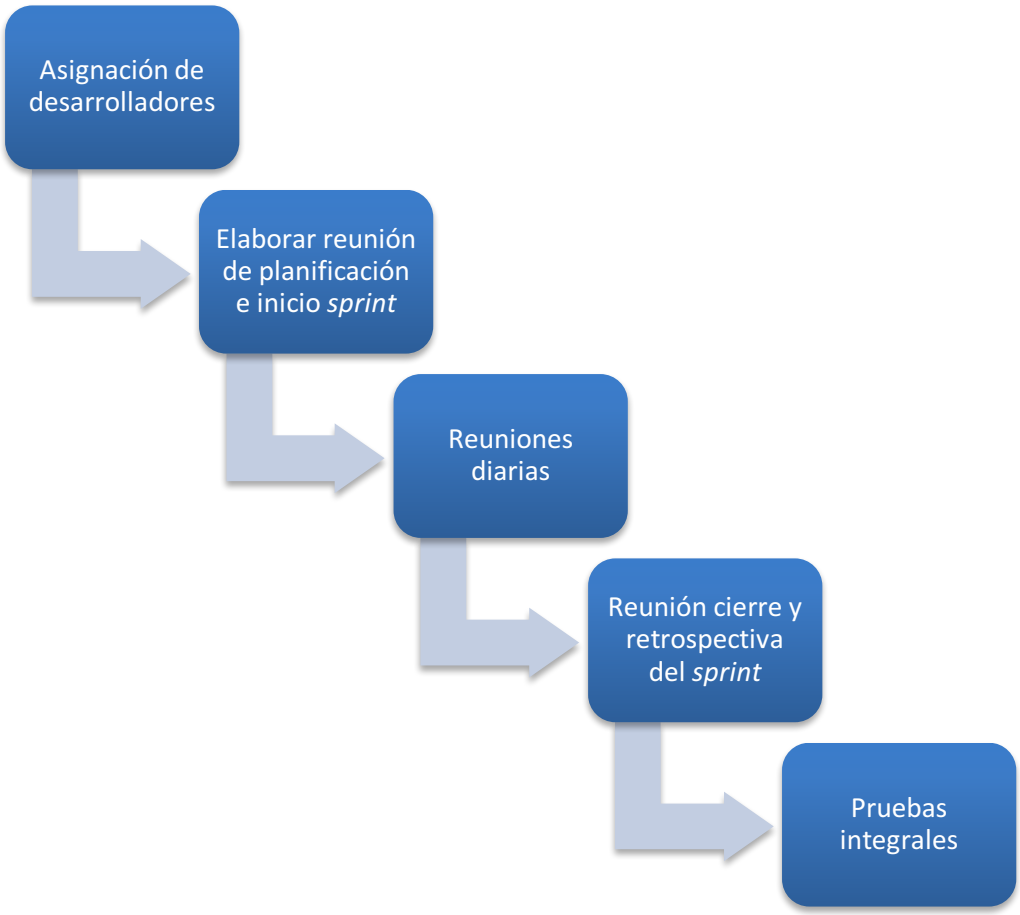


Figura 18. Cuadro 2 resumen metodología

Fuente: Elaboración propia.

5.3. PROPUESTA DE ESTÁNDARES DE DESARROLLO

Mediante esta propuesta de estándares de desarrollo se busca facilitar el mantenimiento del código, tanto a nivel de base de datos como a nivel de programación, otorgando una mayor facilidad para entender lo realizado por cualquier programador.

5.3.1. Base de datos

La siguiente estructura propuesta es para una base de datos MySql.

5.3.1.1. Creación de procedimientos

Todos los procedimientos creados en la base de datos deben tener la siguiente estructura:

"pr_" + [nombre del procedimiento]

Un ejemplo de procedimiento puede ser "pr_crearUsuario" o "pr_resultadoVotaciones".

5.3.1.2. Creación de vistas

Todas las vistas creadas en la base de datos deben tener la siguiente estructura: "v_" + [nombre de la vista]

Un ejemplo de vista puede ser "v_votosPersona" o "v_funcionarios".

5.3.1.3. Nombre de variables

Todas las variables definidas en el sistema deben tener la siguiente estructura: "vl"+ [indicador de tipo de dato] + "_" + [descripción de la variable].

Tabla 5. Indicador de tipo de dato de la variable

Letra	Significado
c	Variable de tipo carácter
n	Variable de tipo numérico
d	Variable de tipo fecha
b	Variable de tipo booleano
o	Variable de tipo objeto

Fuente: Elaboración propia

5.3.2. Código

5.3.2.1. Creación de objetos

Se recomienda organizar cada uno de estos archivos en paquetes o carpetas según su funcionalidad o módulo del sistema al que corresponde, con el fin de evitar tener todo revuelto en una sola ubicación.

Tabla 6. Creación de objetos

Capa del proyecto	Nombre del objeto	Descripción
Intermedia	ent[Nombre de la tabla]	Archivos generados para el manejo de la base de datos a través de *hibernate*
	Dal[Nombre de la tabla]	Archivos contenedores de los métodos para interactuar con la base de datos (agregar, modificar, consultar, eliminar, entre otros)
Superior	lst[Nombre de la pantalla]	Archivo *.xhtml con el diseño de un listado
	frm[Nombre de la pantalla]	Archivo *.xhtml con el diseño de un formulario
	lst[Nombre de la pantalla]Controller	Archivo *.java contenedor de los métodos del listado
	frm[Nombre de la pantalla]Controller	Archivo *.java contenedor de los métodos del formulario

Fuente: Elaboración propia.

5.3.2.2. Nombre de variables

Todas las variables definidas en el sistema deben tener la siguiente estructura: v + [indicador de alcance] + [indicador de tipo de dato] + "_" + [descripción de la variable].

Tabla 7. Indicador de alcance de la variable

Letra	Significado
l	Alcance local: se utiliza para variables declaradas dentro de un método o función.
g	Alcance global: Se utiliza para variables de acceso global de la clase.

Fuente: Elaboración propia.

Tabla 8. Indicador de tipo de dato de la variable

Letra	Significado
c	Variable de tipo carácter
n	Variable de tipo numérico
d	Variable de tipo fecha
b	Variable de tipo booleano
o	Variable de tipo objeto

Fuente: Elaboración propia.

Algunos ejemplos para la creación de variables son los siguientes:

- vlc_nombreEmpleado
- vlo_lista

5.3.2.3. Nombre de parámetros

Todos los parámetros definidos en el sistema deben tener la siguiente estructura:

pv + [indicador de tipo de dato] + “_” + [descripción del parámetro].

Tabla 9. Indicador de tipo de dato del parámetro

Letra	Significado
c	Parámetro de tipo carácter
n	Parámetro de tipo numérico
d	Parámetro de tipo fecha
b	Parámetro de tipo booleano
o	Parámetro de tipo objeto

Fuente: Elaboración propia.

Algunos ejemplos para la creación de parámetros son los siguientes:

- pvc_nombreEmpleado
- pvo_lista

5.4. PROPUESTA DE ESTRUCTURA BÁSICA DE UN PROYECTO

A continuación se detalla la estructura básica de un proyecto, donde se puede ver las capas y cómo se encuentra conformada cada una de ellas.

5.4.1. Base de datos

Elemento básico de todo sistema de información.

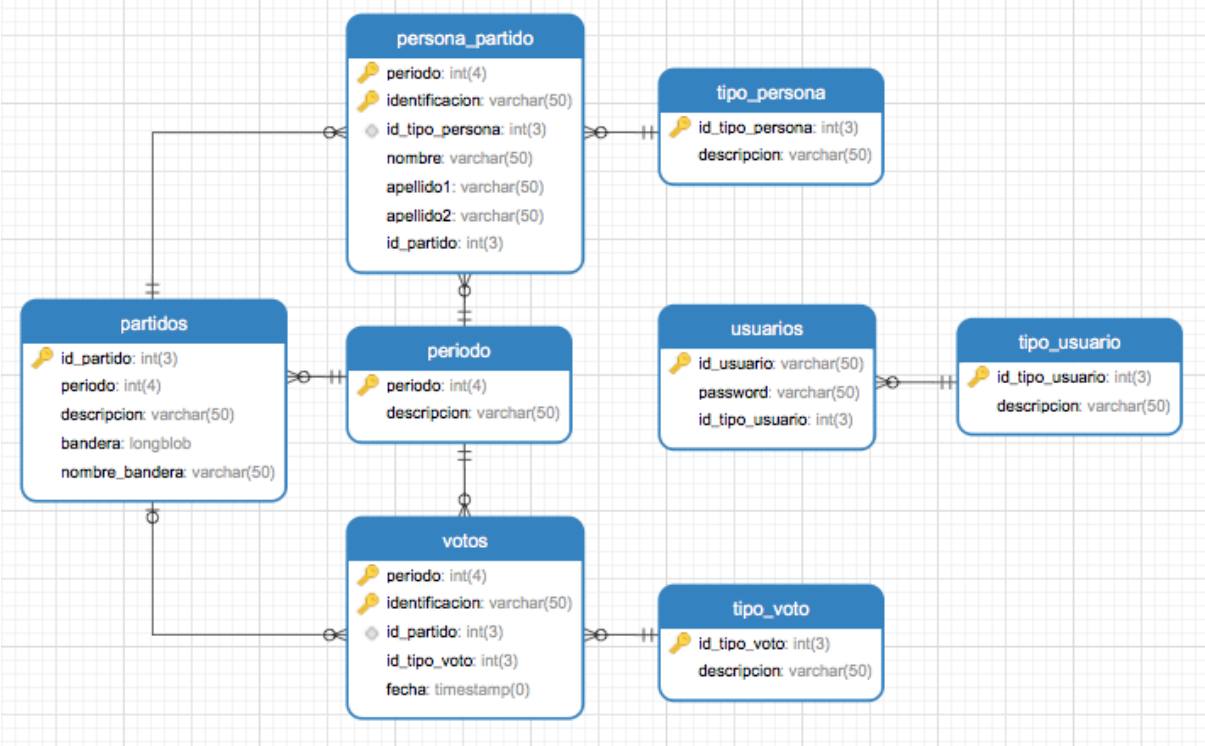


Figura 19. Ejemplo de diseño de una base de datos

Fuente: Elaboración propia.

5.4.2. Capa intermedia

Esta capa se encarga de realizar la comunicación entre la base de datos y el sistema, como propuesta para este proyecto se utiliza el componente *hibérnate,* el cual facilita el manejo de la información y las operaciones básicas (agregar, modificar, eliminar y consultar).

La estructura propuesta son dos paquetes, uno llamado “nombre del proyecto”.ent, el cual contiene todas las entidades del proyecto, y otro paquete llamado “nombre del proyecto”.dal, con todos los métodos por utilizar (agregar, modificar, eliminar, consultar, entre otros) y finalmente el archivo “persistance.xml”, con la conexión a la base de datos.

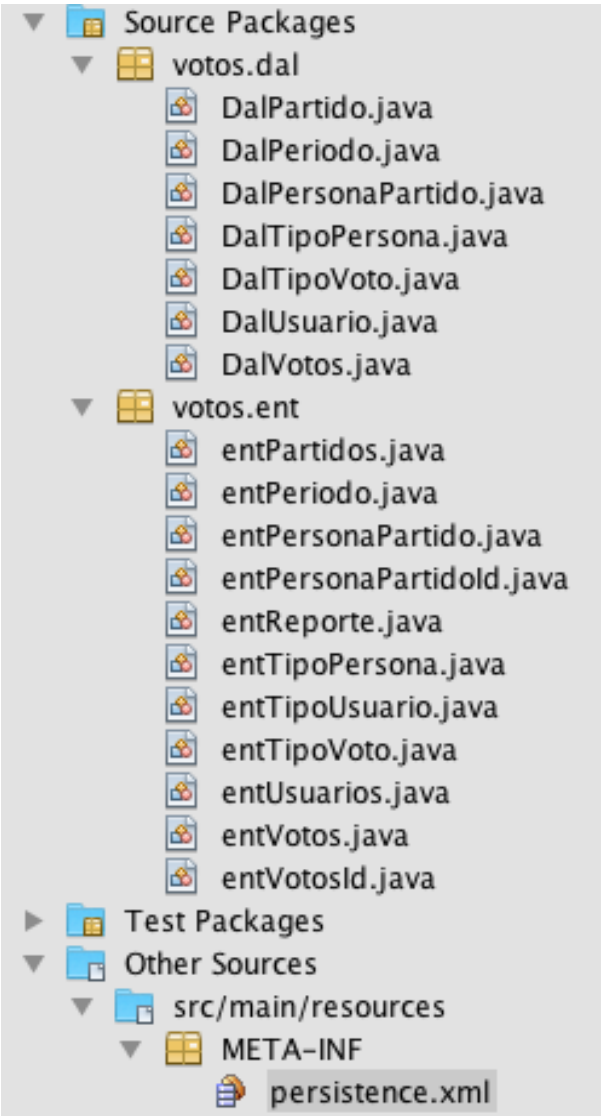


Figura 20. Ejemplo de estructura capa intermedia

Fuente: Elaboración propia.

5.4.2.1. Archivo persistance.xml

Este archivo es necesario cuando se utiliza *hibernate* y es en el cual se configura la conexión a la base de datos, además se indica cuáles entidades creadas en el proyecto (archivos ubicados en la sección “votos.ent”, ver figura 17), van a ser utilizadas en el sistema, por lo que debemos asegurarnos de que se encuentren todas las entidades incluidas.

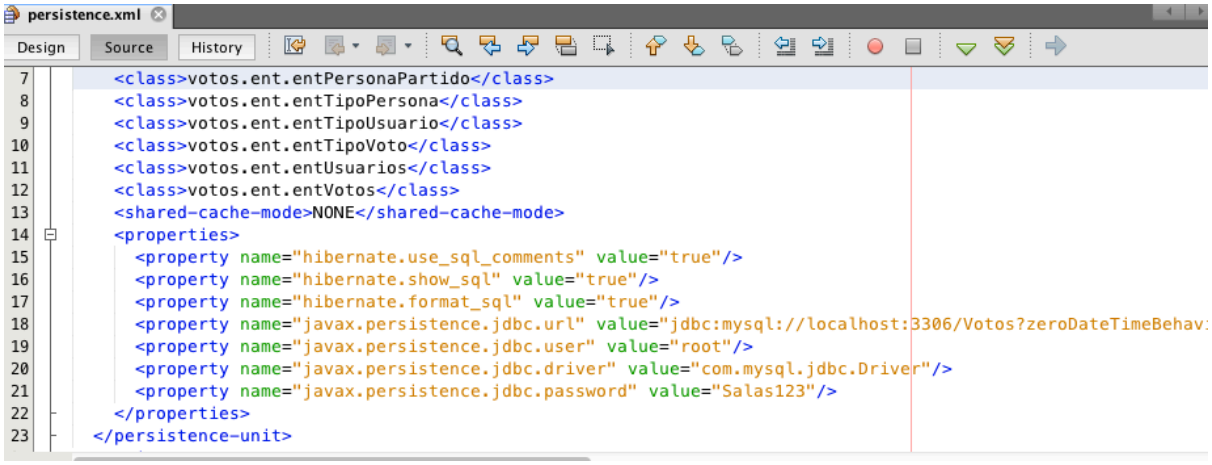


Figura 21. Ejemplo de archivo persistance.xml

Fuente: Elaboración propia.

5.4.2.2. Entidades

Cada uno de estos archivos es un mapeo de una tabla de la base de datos, copiando toda su estructura, pero en lenguaje Java. Estos archivos se encuentran en la ubicación “votos.ent” (ver Figura 18).

Estos archivos pueden ser generados de manera automática, la manera se explica más adelante y solo debemos añadir manualmente las consultas a la base de datos, para obtener listados de información o una entidad como tal partiendo de condiciones definidas.

```
@NamedQueries(value = {
    @NamedQuery(
            name = entPartidos.LISTAR_POR_PERIODO,
            query = "SELECT p FROM entPartidos p WHERE p.periodo.periodo = :periodo"),
    @NamedQuery(
            name = entPartidos.LISTAR_TODOS,
            query = "SELECT p FROM entPartidos p"),
    @NamedQuery(
            name = entPartidos.OBTENER_REGISTRO,
            query = "SELECT p FROM entPartidos p WHERE p.idPartido = :idPartido")
})
```

Figura 22. Ejemplo consultas a la base de datos

Fuente: Elaboración propia.

5.4.2.3. Métodos

Existe un archivo con métodos por cada entidad del proyecto, con el fin de separar mejor el sistema, estos archivos se encuentran en la ubicación “votos.dal” (ver Figura 13) y en cada uno de ellos se programan las acciones básicas (agregar, modificar, eliminar y consultar) o algún método extra según la necesidad del caso.

A continuación se muestra un ejemplo de las operaciones básicas, las variaciones de cada uno de estos métodos para otras entidades o proyectos son el nombre de la unidad de persistencia (nombre que se le da a la configuración de la conexión de base de datos en el archivo persistance.xml, que para el ejemplo se utiliza UP_VOTOS) y el nombre de la entidad (entPartidos).

5.4.2.3.1. Método agregar registro

```
public void InsertarRegistro(entPartidos pvo_Partido) throws Exception {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("UP_VOTOS");
    EntityManager vlo_em = emf.createEntityManager();

    vlo_em.getTransaction().begin();
    vlo_em.persist(pvo_Partido);
    vlo_em.flush();
    vlo_em.getTransaction().commit();
}
```

Figura 23. Ejemplo método agregar

Fuente: Elaboración propia.

5.4.2.3.2. Método modificar registro

```
public void ModificarRegistro(entPartidos pvo_Partido) throws Exception {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("UP_VOTOS");
    EntityManager vlo_em = emf.createEntityManager();

    vlo_em.getTransaction().begin();
    vlo_em.merge(pvo_Partido);
    vlo_em.flush();
    vlo_em.getTransaction().commit();
}
```

Figura 24. Ejemplo método modificar

Fuente: Elaboración propia.

5.4.2.3.3. Método borrar registro

```
public void BorrarRegistro(entPartidos pvo_Partido) throws Exception {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("UP_VOTOS");
    EntityManager vlo_em = emf.createEntityManager();

    vlo_em.getTransaction().begin();
    entPartidos vlo_EntAux = vlo_em.merge(pvo_Partido);
    vlo_em.remove(vlo_EntAux);
    vlo_em.flush();
    vlo_em.getTransaction().commit();

}
```

Figura 25. Ejemplo método borrar

Fuente: Elaboración propia.

5.4.2.3.4. Método obtener registro

```
public entPartidos ObtenerRegistro(int pvn_IdPartido) throws Exception {
    try {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("UP_VOTOS");
        EntityManager vlo_em = emf.createEntityManager();

        return vlo_em.createNamedQuery(entPartidos.OBTENER_REGISTRO, entPartidos.class)
                .setParameter("idPartido", pvn_IdPartido).getSingleResult();

    } catch (NoResultException e) {
        return null;
    }

}
```

Figura 26. Ejemplo método obtener

Fuente: Elaboración propia.

5.4.2.3.5. Método listar todos

```
public List<entPartidos> ListarTodos() throws Exception {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("UP_VOTOS");
    EntityManager vlo_em = emf.createEntityManager();

    return vlo_em.createNamedQuery(entPartidos.LISTAR_TODOS, entPartidos.class).getResultList();

}
```

Figura 27. Ejemplo método listar todos

Fuente: Elaboración propia.

5.4.2.4. Generar entidades de forma automática

Para el siguiente método se utilizó el IDE NetBeans e *Hibernate*, para lograr construir los archivos de forma automática.

5.4.2.4.1. Crear conexión de base de datos

- En el menú superior de Netbeans, nos vamos a la pestaña “Services” > *Databases* > *New Connection…*<

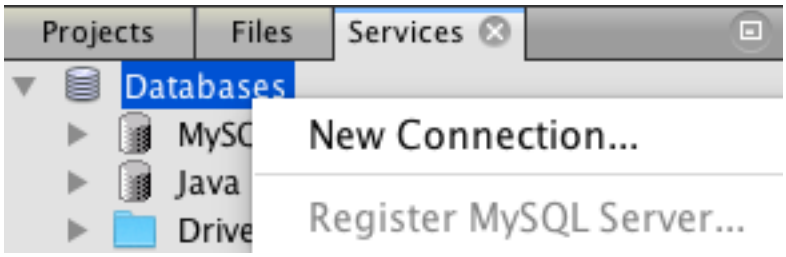


Figura 28. Crear conexión de base de datos

Fuente: Elaboración propia.

- Seleccionamos el *driver* dependiendo del motor de base de datos escogido para la aplicación y presionamos *Next.*

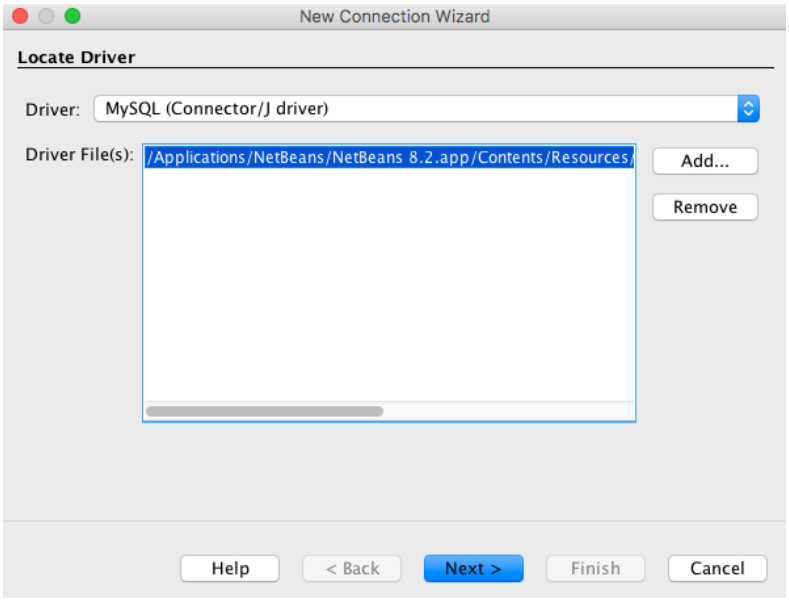


Figura 29. Seleccionar *driver*

Fuente: Elaboración propia.

- Se llenan los datos relacionados con la conexión > Presionamos el botón *Test connection* para asegurarnos de que todo funcione correctamente > Presionamos el botón *Finish.*

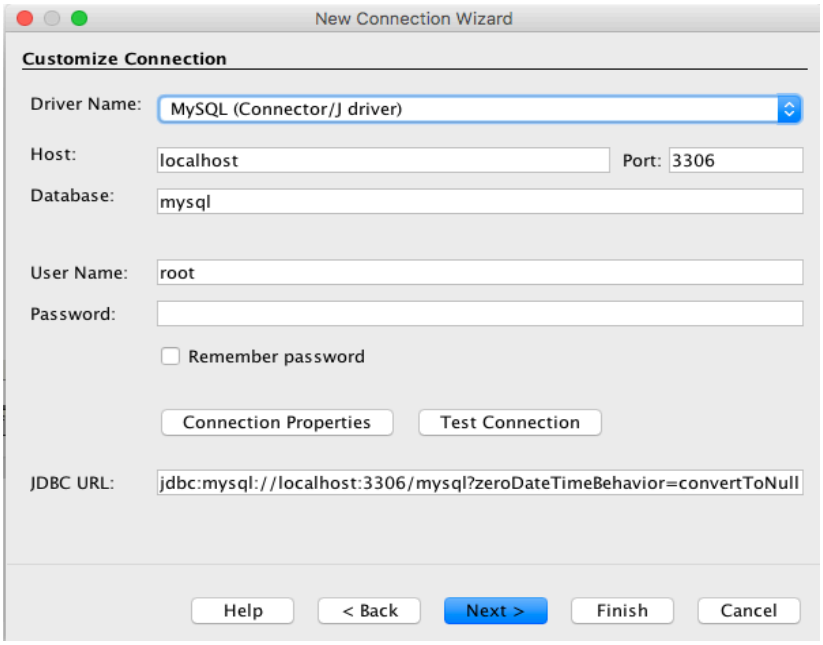


Figura 30. Llenar datos relacionados con la conexión

Fuente: Elaboración propia.

5.4.2.4.2. Crear proyecto web

- Creamos un proyecto Java > *Web application*
- Clic derecho propiedades en el proyecto web y nos vamos a la pestaña de *Framework* para agregar el *Hibernate*

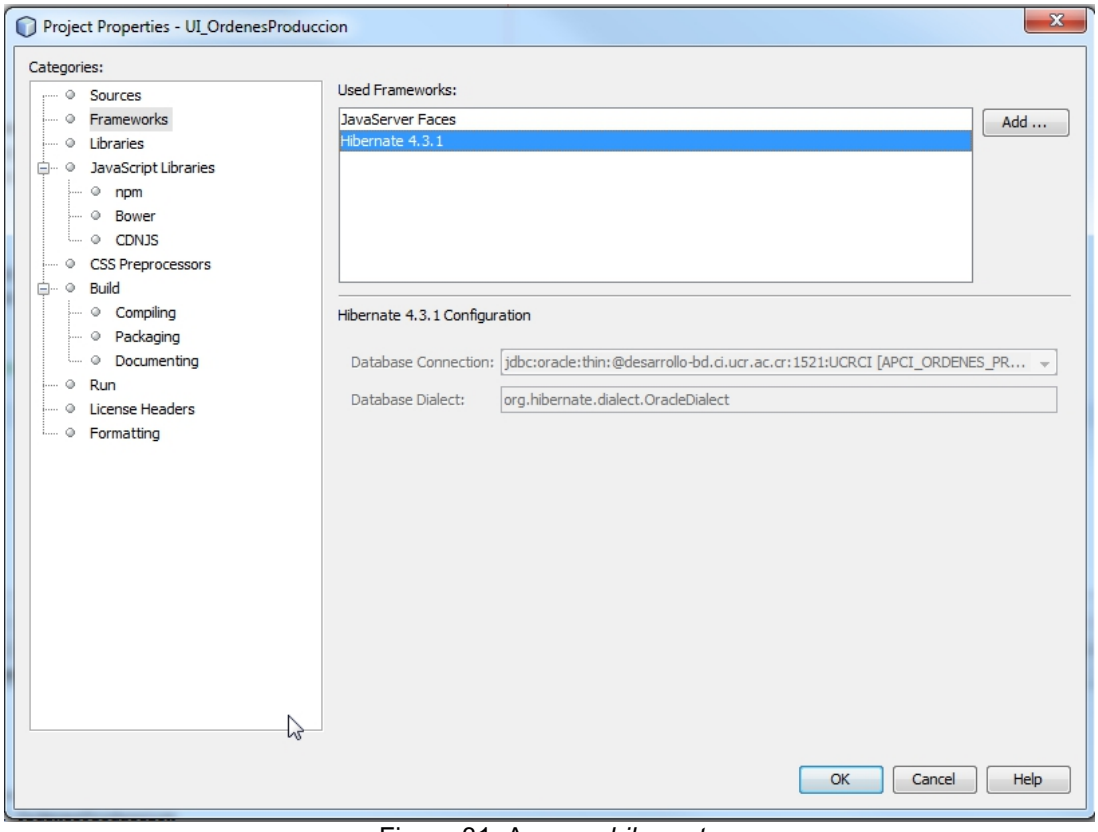


Figura 31. Agregar *hibernate*

Fuente: Elaboración propia.

5.4.2.4.3. Crear archivo hibernate.cfg.xml

- Clic derecho en el proyecto > Seleccionamos *New* > Seleccionamos *Other.*
- Buscamos el archivo sobre la carpeta *Hibernate* y lo creamos.

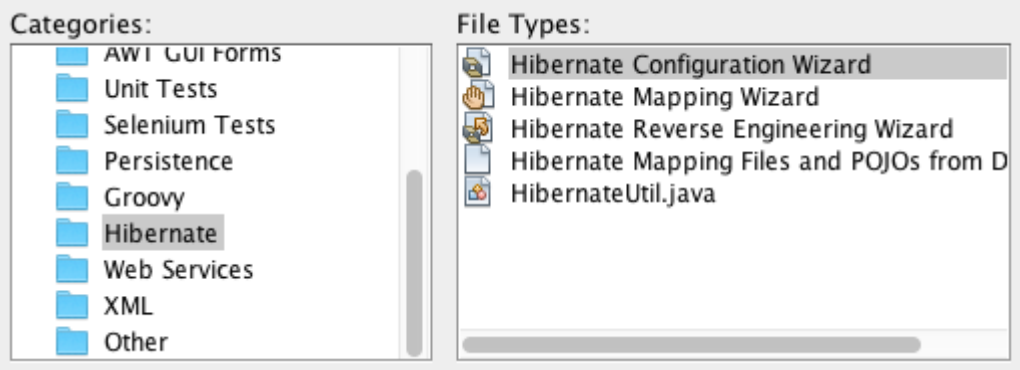


Figura 32. Agregar archivo hibernate.cfg.xml

Fuente: Elaboración propia.

- Abrimos el archivo creado y buscamos la sección *Optional Properties* > *Configuration properties* > presionamos el botón *Add* y agregamos la siguiente propiedad

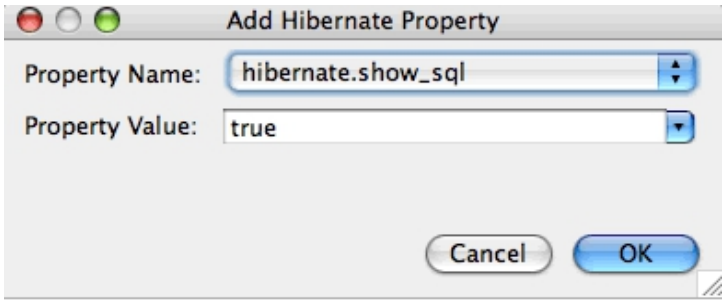


Figura 33. Propiedad “show sql”

Fuente: Elaboración propia.

- Sobre la misma sección *Optional Properties* buscamos la sección *Miscellaneous Properties* y agregamos las siguientes propiedades

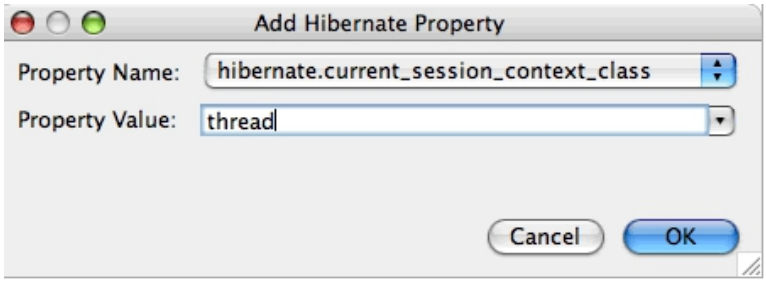


Figura 34. Propiedad “current sesión context class”.

Fuente: Elaboración propia.

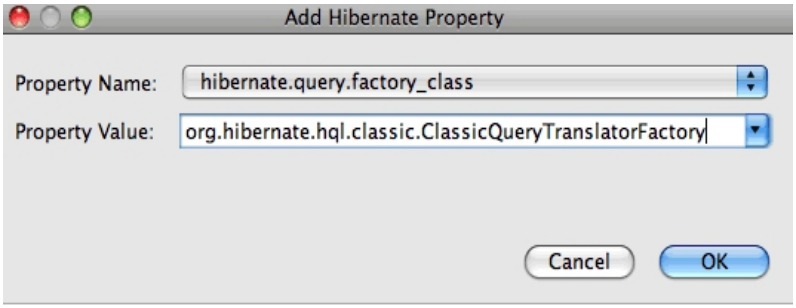


Figura 35. Propiedad *Factory class*

Fuente: Elaboración propia.

5.4.2.4.4. Crear archivo HibernateUtil.java

- Clic derecho en el proyecto > Seleccionamos *New* > Seleccionamos *Other*
- Buscamos el archivo sobre la carpeta *Hibernate* y lo creamos

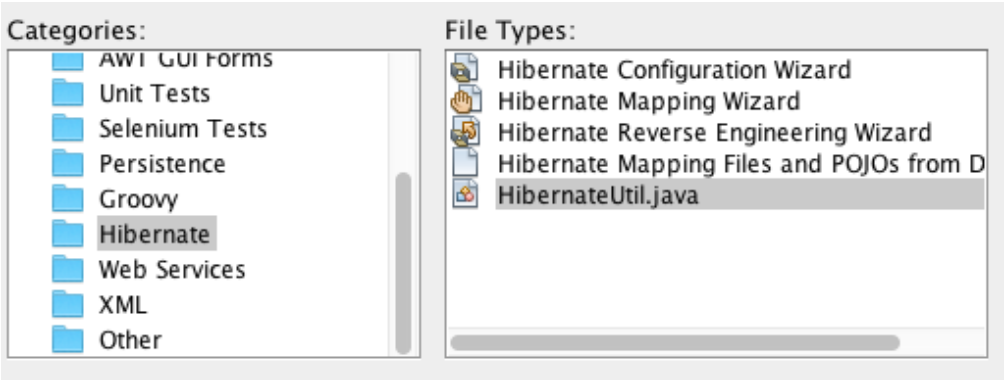


Figura 36. Agregar archivo hibernate.util

Fuente: Elaboración propia.

5.4.2.4.5. Crear archivo hibernate.reveng.xml

- Clic derecho en el proyecto > Seleccionamos *New* > Seleccionamos *Other*
- Buscamos el archivo sobre la carpeta *Hibernate* > Presionamos el botón *Next*

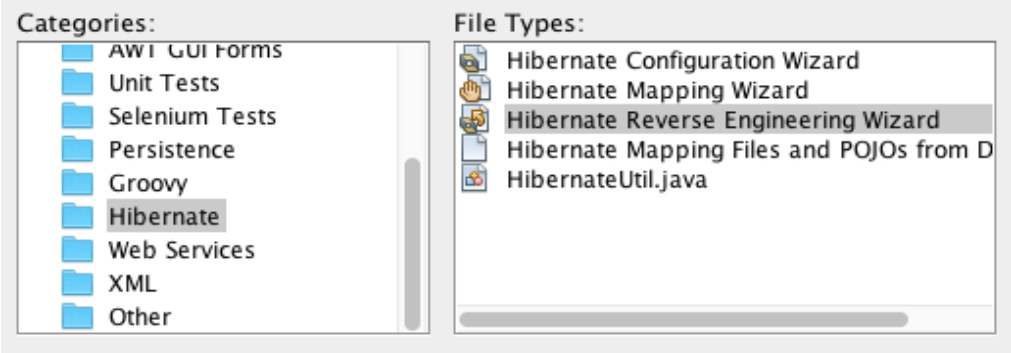


Figura 37. Agregar archivo hibernate.reveng.xml

Fuente: Elaboración propia.

- Verificamos que el archivo de configuración concuerde con el creado anteriormente y seleccionamos las tablas que deseamos generar

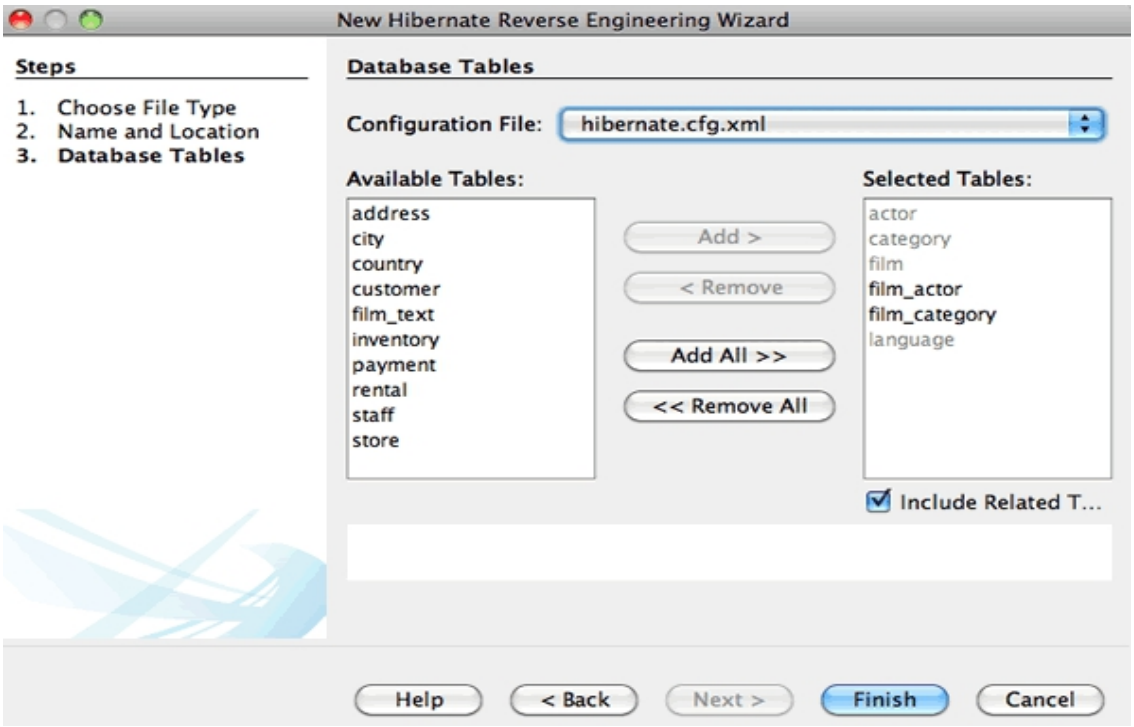


Figura 38. Seleccionar tablas para crear entidades

Fuente: Elaboración propia.

5.4.2.4.6. Creación de las entidades (pojos)

- Clic derecho en el proyecto > Seleccionamos *New* > Seleccionamos *Other*
- Buscamos el archivo sobre la carpeta *Hibernate* > Presionamos el botón *Next*

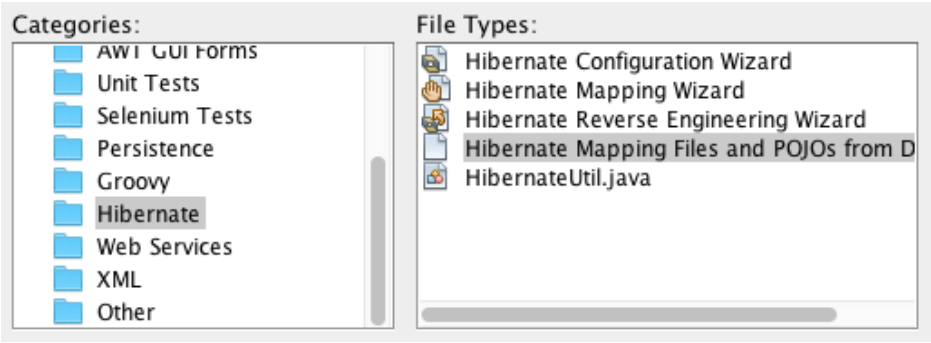


Figura 39. Archivo para generar entidades

Fuente: Elaboración propia.

- Debemos asegurarnos de que ambas opciones de *General Settings* se encuentren marcadas, desmarcar la opción *Hibernate XML Mappings* y finalmente definir un nombre del *Package*, el cual va a contener los archivos.

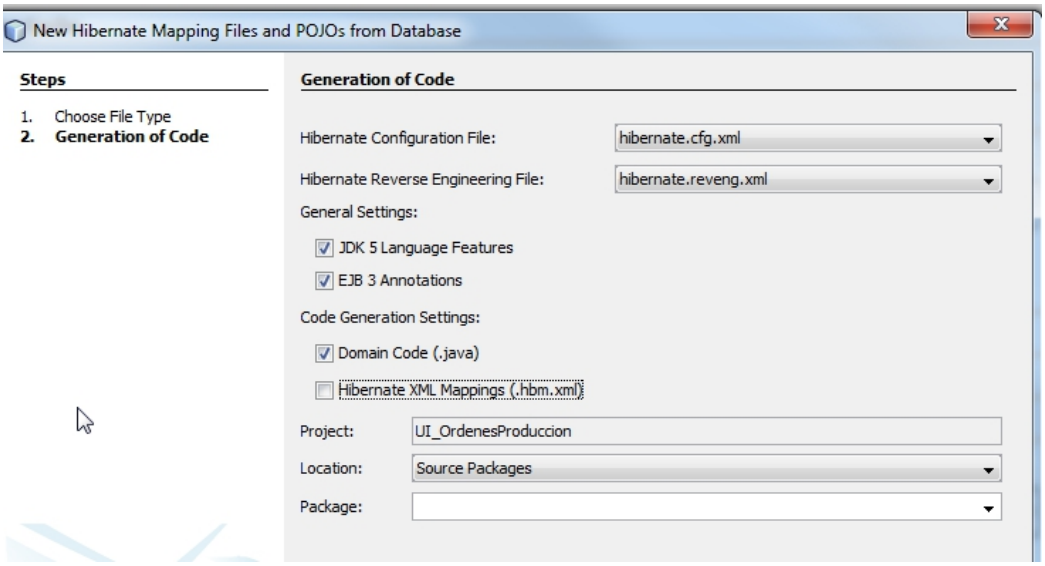


Figura 40. Configuración final para generar entidades

Fuente: Elaboración propia.

- Una vez que ya tenemos generadas todas las entidades, podemos proceder a copiar estos archivos al proyecto que es nuestra capa intermedia.

5.4.3. Capa superior (interfaz)

En esta capa se encuentran todas las pantallas con las cuales tiene interacción el usuario, para esta propuesta se utiliza el *framework Primefaces,* el cual es un componente de *Java Server Faces* (JSF) y cuenta con sus objetos propios.

Cada página web cuenta con su archivo *.xhtml, el cual es el diseño que ve el usuario final; estos archivos deben estar separados por el módulo al que pertenecen y a su tipo (listado o formulario). Además, cada página web posee su archivo *.java, el cual cuenta con todos los métodos y las acciones que realiza la página web.

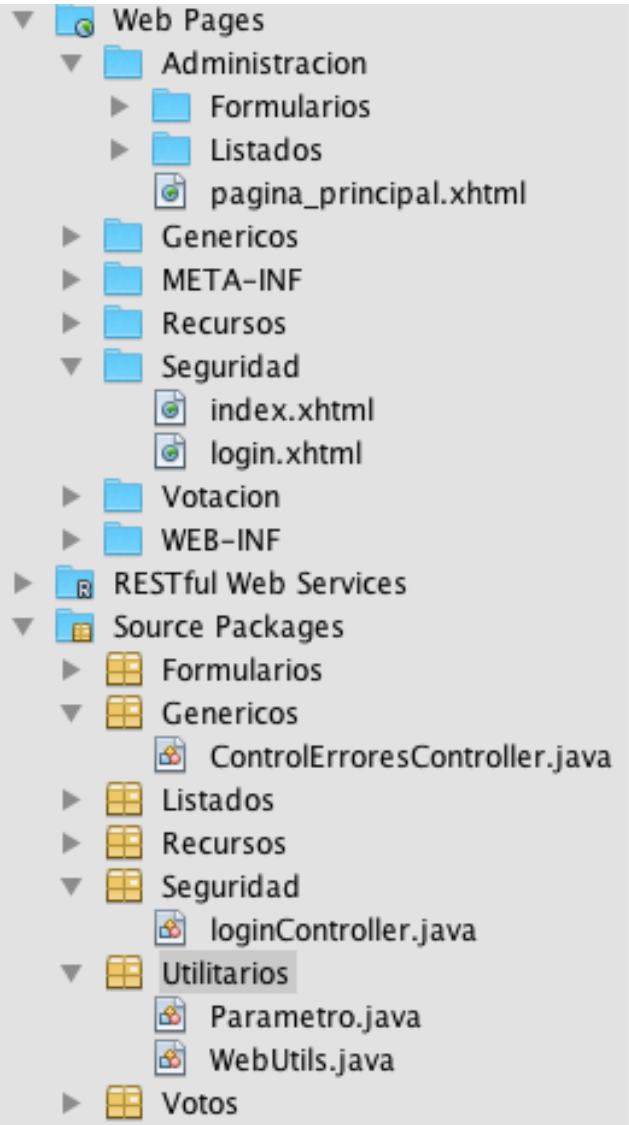


Figura 41. Estructura básica de la capa superior

Fuente: Elaboración propia.

5.4.3.1. Funciones genéricas

El listado de funciones genéricas se encuentran dentro del archivo ¨WebUtils.java¨, pueden ser utilizadas desde cualquier otro archivo *.java, simplemente importándolo. Dichas funciones fueron creadas por el arquitecto de la UCR.

Dentro de estas funciones básicas se encuentran los métodos re direccionar, agregar parámetro de sesión, leer parámetro de sesión y reportar error del sistema.

5.4.3.2. Listados

5.4.3.2.1. Archivo *.xhtml

```
<h:body>

    <h:form id="formulario">
        <p:menubar>
            <p:menuitem value="Pagina principal" url="/Administracion/pagina_principal.xhtml"  />
        </p:menubar>

        <h1><center>Rol</center></h1>
        <p:panelGrid id="pnDatos" columns="1" layout="grid">
            <p:messages id="messages" />
            <p:growl id="mensajes" showDetail="false" life="4000"/>
            <p:commandButton id="btnAgregar" style="width:auto" action="#{lstRolController.agregar()}" title="Agregar" update="messages,pnDatos" >
                <p:tooltip id="toolTipAgregar" for="btnAgregar" value="Agregar" position="top"/>
            </p:commandButton>

            <p:dataTable widgetVar="vlo_WidgetDataTable" var="vlo_EntRol"  value="#{lstRolController.vlo_Lista}" rows="10"
                         paginator="true" paginatorPosition="bottom" currentPageReportTemplate="({currentPage} de {totalPages})"  paginatorTemplate="
                         {CurrentPageReport} {FirstPageLink}
                         {PreviousPageLink} {PageLinks} {NextPageLink}
                         {LastPageLink} {RowsPerPageDropdown}" emptyMessage="No hay registros que cumplan con el criterio de búsqueda">
                <p:column headerText="Descripción" sortable="true" sortBy="#{vlo_EntRol.descripcion}" filterMatchMode="contains" filterBy="
                #{vlo_EntRol.descripcion}" filterStyle="width:40%"   >
                    <h:outputText value="#{vlo_EntRol.descripcion}"/>
                </p:column>
                <p:column  width="6%" id="colModificar"   >
                    <p:commandButton id="btnModificar" icon="ui-icon-pencil" class="btn btn-default" style="border-width: 0;  " action="
                    #{lstRolController.modificar(vlo_EntRol.idTipoPersona)}"  >
                        <p:tooltip id="toolTipModificar" for="btnModificar" value="Modificar" position="top"/>
                    </p:commandButton>
                </p:column>
                <p:column  width="6%" id="colEliminar"   >
                    <p:commandButton id="btnEliminar" icon="ui-icon-trash" class="btn btn-default"
                                     style="border-width: 0;  "
                                     action="#{lstRolController.Borrar(vlo_EntRol.idTipoPersona)}"
                                     update=":formulario:messages,:formulario:mensajes,:formulario:pnDatos"
                                     oncomplete="PF('vlo_WidgetDataTable').filter();" >
                        <p:confirm header="Confirmación del sistema" message="¿Confirma que desea eliminar el registro?" icon="ui-icon-alert" />
                        <p:tooltip id="toolTipEliminar" for="btnEliminar" value="Eliminar" position="top"/>
                    </p:commandButton>
                </p:column>
            </p:dataTable>
            <p:confirmDialog global="true" showEffect="fade" hideEffect="fade" style="align-content: center;">
                <p:commandButton value="Sí" type="button" styleClass="ui-confirmdialog-yes" icon="ui-icon-check" />
                <p:commandButton value="No" type="button" styleClass="ui-confirmdialog-no" icon="ui-icon-close" />
                <p:commandButton value="Cancelar" type="button" styleClass="ui-confirmdialog-no" icon="ui-icon-close" />
            </p:confirmDialog>
        </p:panelGrid>
    </h:form>
</h:body>
```

Figura 42. Ejemplo archivo *.xhtml de un listado

Fuente: Elaboración propia.

En esta imagen podemos ver la estructura básica de un listado:

- Objeto *menubar*, el cual contiene las opciones de menú del sistema
- Título de la página.
- Objeto *messages* y *growl* para el despliegue de mensajes al usuario, *messages* se muestra en la parte superior de la página y *growl* aparece en una esquina, desvaneciéndose después de un tiempo determinado, siendo visible para el usuario sin importar la posición de la página.
- Botón de agregar, el cual nos re direcciona al formulario para ejecutar esta acción.

- Objeto *datatable* mediante el cual mostramos el listado de información, asignándole un objeto de tipo lista.
- Objeto *confirmDialog,* el cual sirve para configurar los botones que se muestran cuando levantamos una alerta mediante el objeto *confirm*.

5.4.3.2.2. Archivo *.java

El archivo *.java de un listado inicia con el encabezado, donde se define las variables por utilizar en la página web, cada una de estas variables debe tener su constructor para poder ser utilizada desde el archivo *.xhtml. En el caso del listado se define un objeto de tipo lista, el cual contiene toda la información por mostrar en pantalla un vez que se llena mediante una consulta a la base de datos.

```
@ManagedBean(name = "lstRolController")
@ViewScoped

public class lstRolController {
    private List<entTipoPersona> vlo_Lista;

    public List<entTipoPersona> getVlo_Lista() {
        return vlo_Lista;
    }

    public void setVlo_Lista(List<entTipoPersona> vlo_Lista) {
        this.vlo_Lista = vlo_Lista;
    }
```

Figura 43. Ejemplo encabezado de archivo *.java en un listado

Fuente: Elaboración propia.

Los métodos de agregar y modificar son muy similares y su acción es re direccionar a un formulario donde se termina de realizar la acción, por lo que solo se definen las variables necesarias para operar en la otra pantalla y se procede a re direccionar.

```
public void agregar() {
    try
    {
        String vlc_Operacion = "AGREGAR";
        WebUtils.agregarParametroDeSesion("pvc_Operacion", vlc_Operacion);
        WebUtils.redireccionar("/Administracion/Formularios/frmRol.xhtml", null, false);
    }catch (Exception ex) {
        WebUtils.reportarErrorDeSistema(ex);
    }
}

public void modificar(int vln_IdTipo) {
    String vlc_Operacion;

    try
    {
        vlc_Operacion = "MODIFICAR";
        WebUtils.agregarParametroDeSesion("pvn_IdTipoPersona", vln_IdTipo);
        WebUtils.agregarParametroDeSesion("pvc_Operacion", vlc_Operacion);
        WebUtils.redireccionar("/Administracion/Formularios/frmRol.xhtml", null, false);
    }catch (Exception ex) {
        WebUtils.reportarErrorDeSistema(ex);
    }
}
```

Figura 44. Ejemplo método agregar y modificar de archivo *.java en un listado

Fuente: Elaboración propia.

El método de borrar se realiza sobre la propia pantalla del listado, por lo que se recibe por parámetro el identificador del registro y se obtiene el registro, posteriormente verificamos si tiene algún archivo asociado, dependiendo de la estructura de la tabla y si no hay registros procedemos a eliminar refrescando la lista, para que el cambio se vea reflejado para el usuario.

```
public void Borrar(int vln_IdTipo) {
    try
    {
        DalTipoPersona vlo_DalTipoPersona = new DalTipoPersona();
        entTipoPersona vlo_TipoPersona = null;

        vlo_TipoPersona = vlo_DalTipoPersona.ObtenerRegistro(String.valueOf(vln_IdTipo));

        FacesContext facesContext = FacesContext.getCurrentInstance();
        facesContext.getExternalContext().getFlash().setKeepMessages(true);

        if (vlo_TipoPersona.getPersonaPartidos().size() > 0)
        {
            facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "No se puede eliminar el registro ya que posee registros
            asociados", ""));
        }
        else
        {
            vlo_DalTipoPersona.BorrarRegistro(vlo_TipoPersona);

            facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "La operación ha finalizado exitosamente", ""));
            vlo_Lista = vlo_DalTipoPersona.ListarTodos();
        }


    }catch (Exception ex) {
        WebUtils.reportarErrorDeSistema(ex);
    }
}
```

Figura 45. Ejemplo método borrar de archivo *.java en un listado

Fuente: Elaboración propia.

El método inicializar se ejecuta de forma automática gracias a su etiqueta “PostConstruct”, por lo que en este método llenamos el archivo de tipo lista ejecutando uno de los métodos de la capa intermedia.

```
@PostConstruct
public void Inicializar()  {

    try {
        DalTipoPersona vlo_DalTipoPersona = new DalTipoPersona();

        vlo_Lista = vlo_DalTipoPersona.ListarTodos();
    }catch (Exception ex) {
        WebUtils.reportarErrorDeSistema(ex);
    }
}
```

Figura 46. Ejemplo método borrar de archivo *.java en un listado

Fuente: Elaboración propia.

5.4.3.3. Formularios

5.4.3.3.1. Archivo *.xhtml

```
<h:body>
    <p:menubar>
        <p:menuitem value="Pagina principal" url="/Administracion/pagina_principal.xhtml"  />
    </p:menubar>

    <h1><center>Rol persona</center></h1>

    <p:panel id="pnDatosUsuario" >
        <p:messages id="messages"  />
        <p:growl id="mensajes" showDetail="false" life="4000"/>
            <h2><p:outputLabel value="#{frmRolController.vlc_Accion}" /></h2>
            <p:separator/>
            <p:panelGrid  columns="2" layout="grid" columnClasses="ui-grid-col-2,ui-grid-col-10" styleClass="ui-panelgrid-blank" >
                <p:outputLabel for="txtDescripcion" value="Descripción" />
                <p:inputText id="txtDescripcion" maxlength="50" value="#{frmRolController.vlo_EntRol.descripcion}"  ></p:inputText>

            </p:panelGrid>

        </p:panel>
        <br/>
        <center>
            <p:commandButton  id="btnAceptar"  value="Aceptar" style="width: auto" actionListener="#{frmRolController.Aceptar()}"  update="
            messages,mensajes" />
            <p:tooltip id="tooltipAceptar" for="btnAceptar" value="Aceptar" position="top"/>
            <p:spacer width="1%"/>
            <p:commandButton  id="btnRegresar"  value="Regresar" style="width: auto" actionListener="#{frmRolController.Regresar()}" immediate="
            true" />
            <p:tooltip id="tooltipRegresar" for="btnRegresar" value="Regresar" position="top"/>

        </center>
</h:body>
```

Figura 47. Ejemplo archivo *.xhtml de un formulario

Fuente: Elaboración propia.

En esta imagen podemos ver la estructura básica de un formulario:

- Objeto *menubar,* el cual contiene las opciones de menú del sistema.
- Título de la página.
- Objeto *messages* y *growl* para el despliegue de mensajes al usuario, *messages* se muestra en la parte superior de la página y *growl* aparece en una esquina, desvaneciéndose después de un tiempo determinado, siendo visible para el usuario sin importar la posición de la página.
- Titulo para mostrar la acción que estamos realizando, dicho mensaje se muestra de forma dinámica mediante la variable “vlc_accion” que se llena desde el archivo *.java.
- Objeto *panelGrid,* el cual contiene todos los campos editables, conformado por el identificador del campo en un objeto *outputLabel* y un objeto *input* con el dato que vamos a almacenar en la base de datos
- Botón regresar para volver al listado sin realizar una acción y botón aceptar para almacenar la información ingresada en la base de datos.

5.4.3.3.2. Archivo *.java

El archivo *.java de un formulario inicia con el encabezado, donde se define las variables por utilizar en la página web, cada una de estas variables debe tener su constructor para poder ser utilizadas desde el archivo *.xhtml. En el caso del formulario se define un objeto de tipo entidad, el cual mapeamos a los campos editables en el archivo *.xhtml para poder modificarlos y posteriormente guardarlos en la base de datos, un objeto para almacenar el identificador del objeto con el que estamos trabajando, un *string* llamado vlc_Operacion, el cual llenamos con el dato que enviamos

desde el listado y un *string* llamado vlc_Accion, el cual mostramos en el archivo *.xhtml para informarle al usuario el tipo de acción que está realizando.

```
@ManagedBean(name = "frmRolController")
@ViewScoped

public class frmRolController {
    private entTipoPersona vlo_EntRol;
    private int vln_IdRol;
    private String vlc_Operacion;
    public String vlc_Accion;

    public int getVln_IdRol() {
        return vln_IdRol;
    }

    public void setVln_IdRol(int vln_IdRol) {
        this.vln_IdRol = vln_IdRol;
    }
```

Figura 48. Ejemplo encabezado de archivo *.java en un formulario

Fuente: Elaboración propia.

Mediante el método regresar le habilitamos al usuario la opción de volver al listado sin realizar ninguna otra acción.

```
public void Regresar() throws Exception {
    WebUtils.redireccionar("/Administracion/Listados/lstRol.xhtml", null, false);
}
```

Figura 49. Ejemplo método regresar de archivo *.java en un formulario

Fuente: Elaboración propia.

Con el método aceptar podemos almacenar la información en la base de datos, diferenciando si vamos a agregar o modificar un registro mediante la variable vlc_Operacion, una vez terminada la acción re direccionamos al usuario a la pantalla del listado.

```
public void Aceptar()  {
    try
    {
        DalTipoPersona vlo_DalTipoPersona = new DalTipoPersona();
        if (vlc_Operacion.equals("AGREGAR"))
        {

            vlo_DalTipoPersona.InsertarRegistro(vlo_EntRol);
        }
        else if (vlc_Operacion.equals("MODIFICAR"))
        {

            vlo_DalTipoPersona.ModificarRegistro(vlo_EntRol);
        }

        FacesContext facesContext = FacesContext.getCurrentInstance();
        facesContext.getExternalContext().getFlash().setKeepMessages(true);
        facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "La operación ha finalizado exitosamente", ""));

        WebUtils.redireccionar("/Administracion/Listados/lstRol.xhtml", null, false);



    }catch (Exception ex) {
        WebUtils.reportarErrorDeSistema(ex);
    }
}
```

Figura 50. Ejemplo método aceptar de archivo *.java en un formulario

Fuente: Elaboración propia.

El método inicializar se ejecuta de forma automática gracias a su etiqueta *PostConstruct,* por lo que en este método recuperamos los valores enviados desde el listado, para poder identificar el tipo de acción que vamos a realizar.

Cuando vamos a agregar un registro inicializamos la variable de la entidad como nueva, en caso de modificar el registro, lo recuperamos de la base de datos.

```
@PostConstruct
public void Inicializar()  {

    try {


        vlc_Operacion=  (String)WebUtils.leerParametroDeSesion("pvc_Operacion");

        if (vlc_Operacion.equals("AGREGAR"))
        {
            vlo_EntRol = new entTipoPersona();
            vlc_Accion = "Agregar nuevo rol";
        }
        else if (vlc_Operacion.equals("MODIFICAR"))
        {
            DalTipoPersona vlo_DalTipoPersona = new DalTipoPersona();
            vln_IdRol =  (int)WebUtils.leerParametroDeSesion("pvn_IdTipoPersona");
            vlo_EntRol = vlo_DalTipoPersona.ObtenerRegistro(String.valueOf(vln_IdRol));
            vlc_Accion = "Modificar rol";

        }


    }catch (Exception ex) {
        WebUtils.reportarErrorDeSistema(ex);
    }
}
```

Figura 51. Ejemplo método borrar de archivo *.java en un listado

Fuente: Elaboración propia.

5.5. IMPLEMENTACIÓN DE PLAN PILOTO

Con el fin de comprobar el funcionamiento de los estándares y la metodología de desarrollo se crea un proyecto base, el cual se fundamenta en una aplicación de votaciones que contiene dos módulos: el módulo administrativo para manejar la información del sistema, tal como periodos, partidos políticos, integrantes de los partidos, entre otros y además lograr ver el resultado de las votaciones para un periodo determinado; por otro lado se cuenta con un módulo para ejercer el voto, mediante el cual podemos seleccionar el partido político de preferencia, votar como nulo o bien registrar el voto en blanco después de no realizar ninguna acción durante un tiempo determinado.

5.5.1. Pantalla de ingreso al sistema

En la pantalla de ingreso al sistema se muestran dos campos, el usuario y clave.

Figura 52. Pantalla de ingreso

Fuente: Elaboración propia.

Debe digitar el usuario y clave, posteriormente presionar el botón “Aceptar” (ver Figura 45), dependiendo del nivel de acceso configurado para el usuario el sistema re direccionará al área administrativa o al área para emitir votos.

5.5.2. Área administrativa

5.5.2.1. Periodo

La opción de periodo permite crear periodos en el sistema para llevar a cabo una nueva elección electoral.

Inicialmente se mostrará una lista de periodos ya registrados en el sistema.

Figura 53. Listado de periodos

Fuente: Elaboración propia.

5.5.2.1.1. Agregar un registro

Debe presionar el botón “Agregar” para poder ingresar un nuevo registro mediante un formulario, el cual solicita la información requerida.

Figura 54. Agregar nuevo registro

Fuente: Elaboración propia.

A continuación se detallan los campos de este formulario:

- Periodo = Corresponde al año del periodo que vamos a crear.
- Descripción = Corresponde a una descripción adicional en caso de ser necesario.

Presione el botón “Aceptar” para almacenar la información o el botón “Regresar” para volver al listado sin realizar ninguna acción.

5.5.2.1.2. Modificar registro

Al dar clic en el botón de Modificar Registro se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

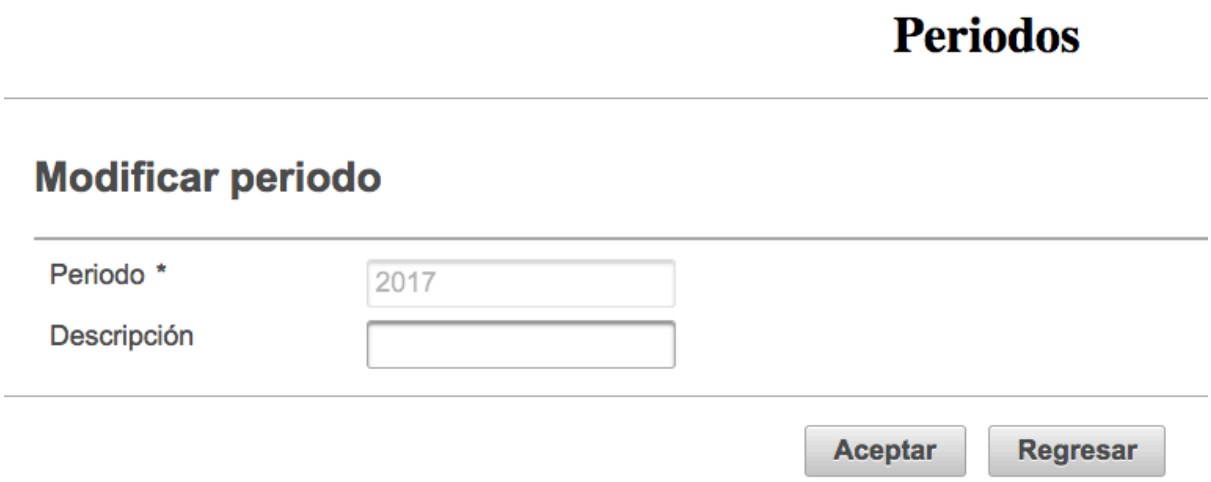


Figura 55. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.5.2.1.3. Eliminar registro

Al presionar el botón de eliminar en el listado, se muestra un mensaje de confirmación, presione Sí para eliminar el registro, No o Cancelar para volver al listado sin eliminar.

Figura 56. Eliminar registro

Fuente: Elaboración propia.

Una vez eliminado el registro se muestra un mensaje de confirmación y este desaparece del listado.

5.5.2.2. Partido político

La opción de partido político permite crear partidos en el sistema asociados a un periodo, para llevar a cabo una nueva elección.

Inicialmente se mostrará una lista de partidos ya registrados en el sistema.

Figura 57. Listado de partidos.

Fuente: Elaboración propia.

5.5.2.2.1. Agregar un registro

Debe presionar el botón “Agregar” para poder ingresar un nuevo registro mediante un formulario, el cual solicita la información requerida.

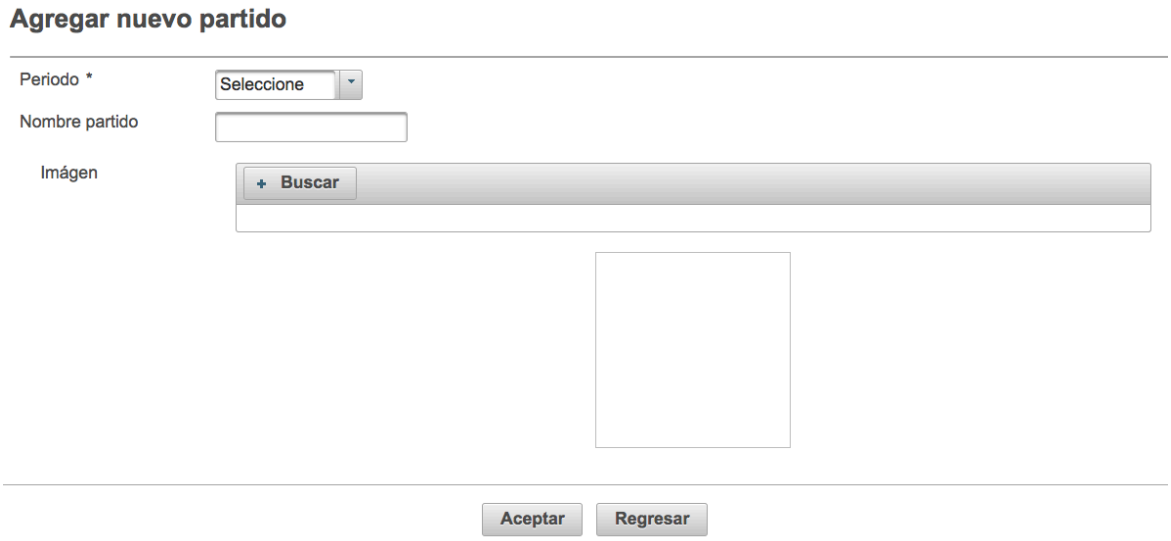


Figura 58. Agregar nuevo registro

Fuente: Elaboración propia.

A continuación se detallan los campos de este formulario:

- Periodo = Corresponde al periodo al cual va a pertenecer el partido que creemos.
- Nombre partido = Corresponde al nombre del partido.

- Imagen = Corresponde a la imagen de la bandera del partido político, la cual se va a mostrar en la pantalla para emitir el voto.

Presione el botón “Aceptar” para almacenar la información o el botón “Regresar” para volver al listado sin realizar ninguna acción.

5.5.2.2.2. Modificar registro

Al dar clic en el botón de “Modificar Registro” se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

Figura 59. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.5.2.2.3. Eliminar registro

Al presionar el botón “Eliminar” en el listado se muestra un mensaje de confirmación, presione Sí para eliminar el registro, No o Cancelar para volver al listado sin eliminar.

Figura 60. Eliminar registro

Fuente: Elaboración propia.

Una vez eliminado el registro se muestra un mensaje de confirmación y este desaparece del listado.

5.5.2.3. Rol integrante

La opción de rol integrante permite crear roles en el sistema para los integrantes de los partidos políticos.

Inicialmente se mostrará una lista de partidos ya registrados en el sistema.

Figura 61. Listado de roles para integrantes

Fuente: Elaboración propia.

5.5.2.3.1. Agregar un registro

Debe presionar el botón “Agregar” para poder ingresar un nuevo registro mediante un formulario, el cual solicita la información requerida.

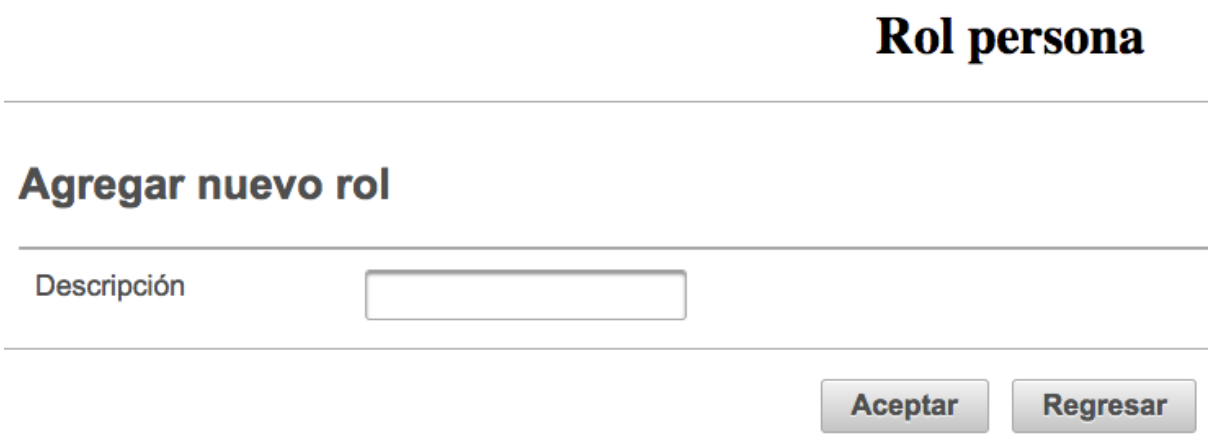


Figura 62. Agregar nuevo registro

Fuente: Elaboración propia.

A continuación se detallan los campos de este formulario:

- Descripción = Corresponde al nombre del rol

Presione el botón “Aceptar” para almacenar la información o el botón “Regresar” para volver al listado sin realizar ninguna acción.

5.5.2.3.2. Modificar registro

Al dar clic en el botón de “Modificar Registro” se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

Figura 63. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.5.2.3.3. Eliminar registro

Al presionar el botón de “Eliminar” en el listado se muestra un mensaje de confirmación, presione Sí para eliminar el registro, No o Cancelar para volver al listado sin eliminar.

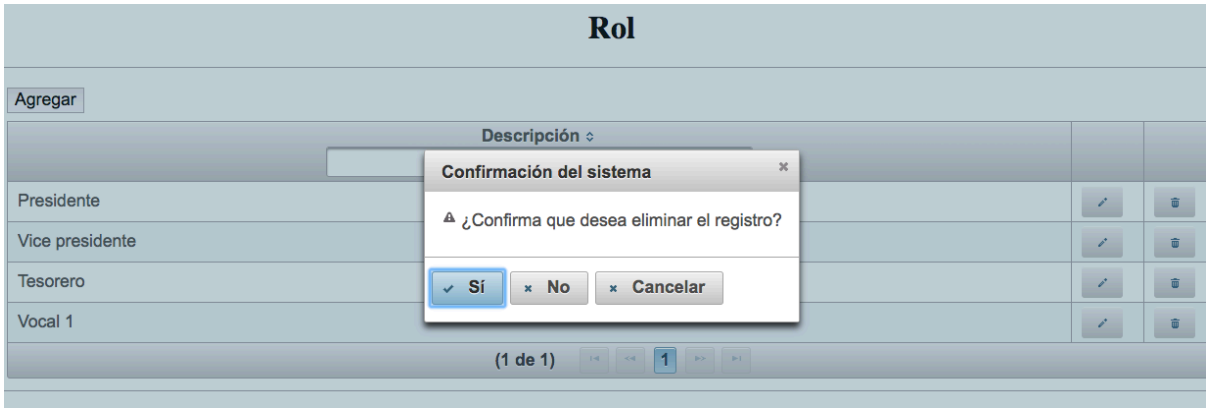


Figura 64. Eliminar registro

Fuente: Elaboración propia.

Una vez eliminado el registro se muestra un mensaje de confirmación y este desaparece del listado.

5.5.2.4. Integrantes partido

La opción de integrantes partido permite asociar los integrantes a un partido político ya creado en la base de datos.

Inicialmente se mostrará una lista de integrantes ya registrados en el sistema.

Figura 65. Listado de integrantes por partido

Fuente: Elaboración propia.

5.5.2.4.1. Agregar un registro

Debe presionar el botón “Agregar” para poder ingresar un nuevo registro mediante un formulario, el cual solicita la información requerida.

Figura 66. Agregar nuevo registro

Fuente: Elaboración propia.

A continuación se detallan los campos de este formulario:

- Partido político = Corresponde al partido político al que va a pertenecer el integrante.
- Rol = Corresponde a la función que va a tener el integrante.
- Identificación = Corresponde a la identificación personal del integrante.
- Nombre = Corresponde al nombre del integrante.
- Primer apellido = Corresponde al primer apellido del integrante.
- Segundo apellido = Corresponde al segundo apellido del integrante.

Presione el botón “Aceptar” para almacenar la información o el botón “Regresar” para volver al listado sin realizar ninguna acción.

5.5.2.4.2. Modificar registro

Al dar clic en el botón de “Modificar Registro” se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

Figura 67. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.5.2.4.3. Eliminar registro

Al presionar el botón “Eliminar” en el listado se muestra un mensaje de confirmación, presione Sí para eliminar el registro, No o Cancelar para volver al listado sin eliminar.

Figura 68. Eliminar registro

Fuente: Elaboración propia.

Una vez eliminado el registro se muestra un mensaje de confirmación y este desaparece del listado.

5.5.2.5. Reporte votaciones

La opción de reporte votaciones permite ver el total de votos registrados para un periodo determinado.

Inicialmente se mostrará un combo con los periodos registrados en el sistema.

Figura 69. Listado de integrantes por partido

Fuente: Elaboración propia.

5.5.2.5.1. Generar reporte

Una vez seleccionado el periodo presione el botón “Aceptar” para generar el reporte del sistema.

Figura 70. Reporte votaciones

Fuente: Elaboración propia.

Una vez presionado el botón “Aceptar” aparece un cuadro de diálogo con la información correspondiente al periodo seleccionado.

5.5.3. Área de votos

5.5.3.1. Pantalla acceso para emitir el voto

En esta pantalla se debe digitar el número de identificación y posteriormente el botón “Aceptar” para emitir el voto.

Figura 71. Pantalla para digitar la identificación

Fuente: Elaboración propia.

5.5.3.2. Pantalla para emitir el voto

En esta pantalla se pueden registrar 3 tipos de voto diferentes, los cuales se detallan a continuación:

- Voto por partido político = Cada partido político registrado para el periodo actual cuenta con un botón de “Seleccionar” debajo de la bandera del mismo, presionando este botón se emite el voto.
- Voto nulo = En la parte inferior de la pantalla está el botón “Voto nulo”, al presionar este botón se registra el voto como nulo.
- Voto en blanco = Si el usuario no realiza ninguna acción durante 30 segundos después de haber sido habilitada la pantalla, se registra el voto como en blanco.

Figura 72. Pantalla para digitar la identificación

Fuente: Elaboración propia.

5.6. PRIMER MÓDULO DE ÓRDENES DE PRODUCCIÓN

El sistema de Órdenes de Producción tiene la finalidad de apoyar la editorial de la Universidad de Costa Rica en el manejo de todas las solicitudes que reciben para realizar un trabajo determinado.

Mediante el primer módulo de este sistema se pretende habilitar a todos los funcionarios de la institución para que mediante su correo institucional puedan ingresar al sistema y gestionar sus solicitudes, desde el ingreso de la solicitud como tal, hasta la alerta para saber cuándo puede retirar su trabajo terminado.

Dicho proyecto fue estimado para 11 meses en su totalidad, a continuación se muestra el detalle de todos los puntos del proyecto.

Tabla 10. Detalle proyecto UCR

Producto	Duración	Porcentaje
Desarrollo de las funcionalidades de gestión de procesos de ingreso y seguridad	4.5	2.5
Desarrollo de las funcionalidades de catálogos del sistema	6	6
Desarrollo de las funcionalidades de gestión de procesos de creación y aprobaciones de las solicitudes de órdenes de producción	12	5.5
Desarrollo de las funcionalidades de gestión de procesos de control presupuestario y aprobaciones	7.5	3.4
Desarrollo de las funcionalidades de gestión de procesos de autorizaciones del jefe administrativo	6	2.7
Desarrollo de las funcionalidades de gestión de procesos de aprobaciones Dirección	2.5	1.5

Desarrollo de las funcionalidades de gestión de procesos de trazabilidad de las órdenes de producción	4	1.8
Desarrollo de las funcionalidades de generación de solicitudes de reclamos	3	1.3
Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción	5	2.3
Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción trabajos adicionales	5	2.3
Desarrollo de las funcionalidades de gestión de órdenes de producción emergencia	3	1.4
Desarrollo de las funcionalidades de gestión de procesos de entidades externas y control de usuarios autorizados	14	6.5
Desarrollo de las funcionalidades de gestión de procesos de gestión de órdenes de producción para entidades externas (control de usuarios autorizados)	15.5	7
Desarrollo de las funcionalidades de gestión de procesos de solicitudes de producción de recepción, reimpresiones y recepción de reclamos	12	5.5
Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con SIAF	20	9
Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con *FileMaker*	20	9
Desarrollo de las funcionalidades de reportes y consultas del sistema	10	4
Creación de guías rápidas y videos tutoriales	10	4
Implementación del sistema y acompañamiento	60	27.3

Totales	220	100

Fuente: Elaboración propia.

A continuación se detallan algunas de las pantallas del sistema, en las cuales se hizo uso de los estándares y metodología de desarrollo propuestos.

5.6.1. Pantalla de ingreso al sistema

En la pantalla de ingreso al sistema se muestran dos campos, el usuario y clave.

Figura 73. Pantalla de ingreso

Fuente: Elaboración propia.

Debe digitar el usuario y clave, posteriormente presionar el botón “Aceptar” (ver Figura 66), dependiendo del nivel de acceso configurado para el usuario el sistema mostrará solo las opciones de menú a las cuales tiene acceso.

5.6.2. Pantalla de bienvenida

Pantalla que se muestra una vez que el usuario ha logrado ingresar al sistema, en la parte superior se muestra un menú con todas las opciones disponibles.

Figura 74. Pantalla de bienvenida

Fuente: Elaboración propia.

5.6.3. Parámetros

La opción de parámetros permite configurar opciones básicas del sistema. Inicialmente se mostrará una lista de parámetros ya registrados en el sistema.

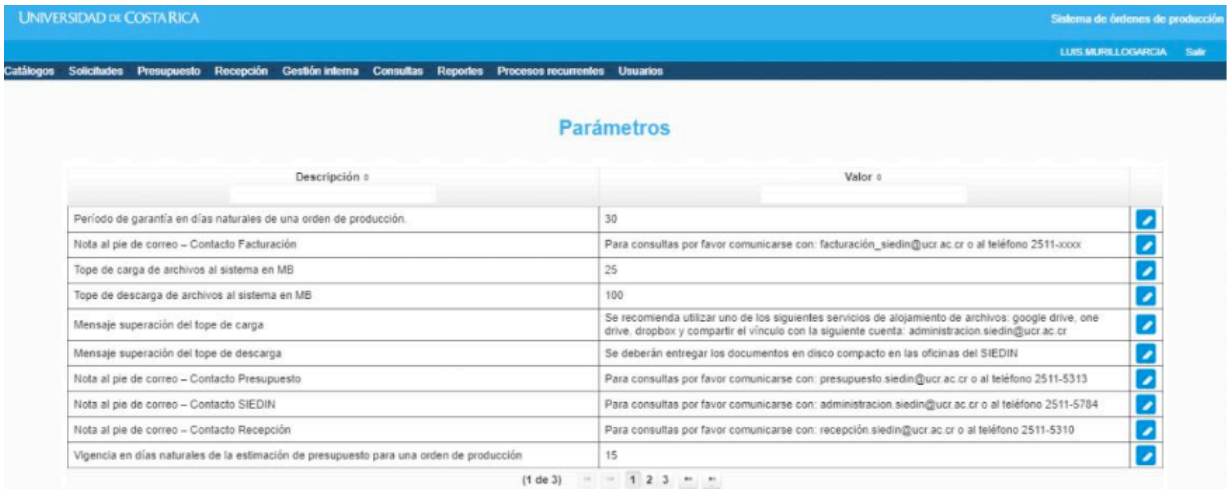


Figura 75. Listado de parámetros

Fuente: Elaboración propia.

5.6.3.1. Modificar registro

Al dar clic en el botón de “Modificar Registro” se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

Figura 76. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.6.4. Tipos de impreso

La opción de tipos de impreso permite definir características a los trabajos que se registren por el sistema.

Inicialmente se mostrará una lista de tipos ya registrados en el sistema.

Figura 77. Listado de tipos de impreso

Fuente: Elaboración propia.

5.6.4.1.1. Agregar un registro

Debe presionar el botón “Agregar” para poder ingresar un nuevo registro mediante un formulario, el cual solicita la información requerida.

Figura 78. Agregar nuevo registro

Fuente: Elaboración propia.

A continuación se detallan los campos de este formulario:

- Descripción = Corresponde al nombre de tipo de impreso que va a registrar.

Presione el botón “Aceptar” para almacenar la información o el botón “Regresar” para volver al listado sin realizar ninguna acción.

5.6.4.1.2. Modificar registro

Al dar clic en el botón de “Modificar Registro” se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

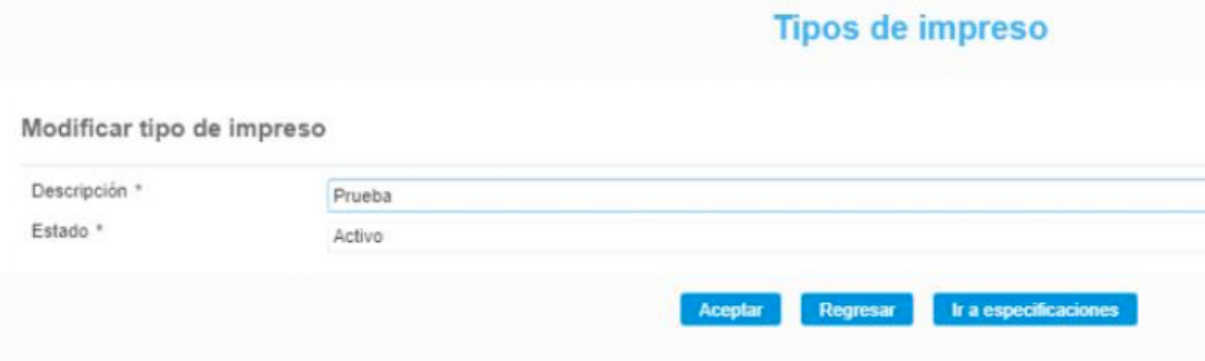


Figura 79. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.6.4.1.3. Eliminar registro

Al presionar el botón de “Eliminar” en el listado se muestra un mensaje de confirmación, presione Sí para eliminar el registro, No o Cancelar para volver al listado sin eliminar.

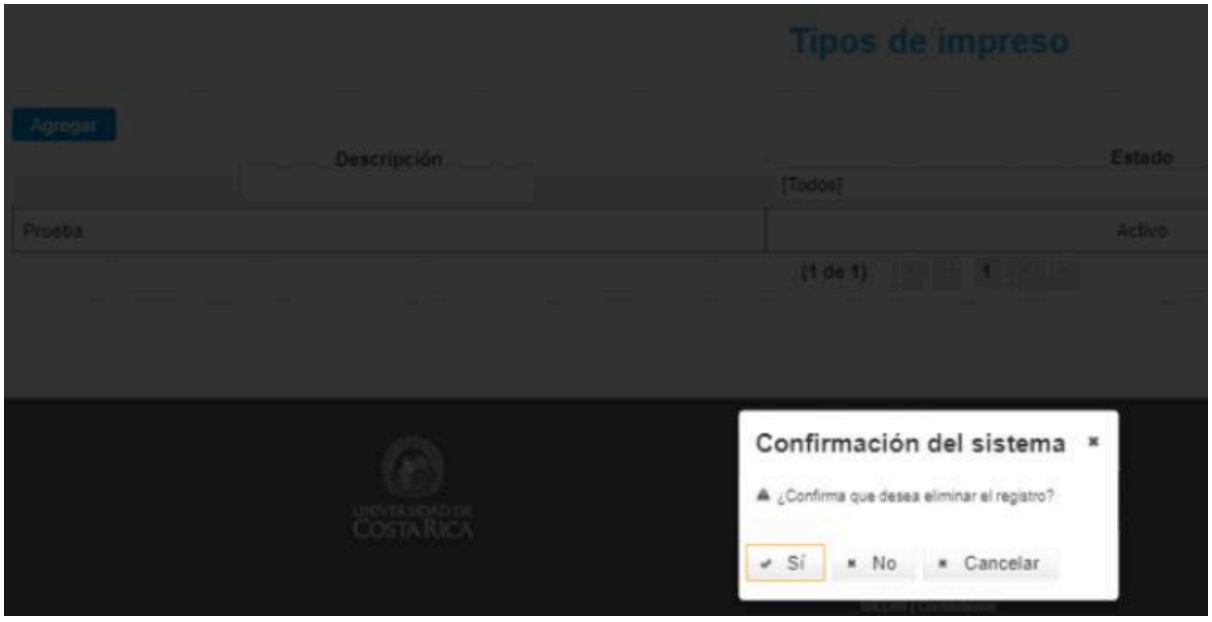


Figura 80. Eliminar registro

Fuente: Elaboración propia.

Una vez eliminado el registro se muestra un mensaje de confirmación y este desaparece del listado.

5.6.5. Formas de pago

La opción de formas de pago permite configurar las opciones de pago con las que cuenta cada usuario que solicite un trabajo.

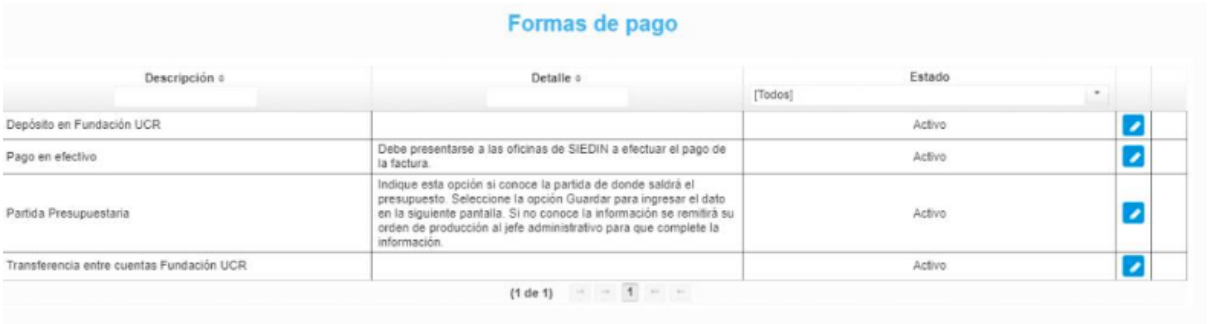


Figura 81. Listado de formas de pago

Fuente: Elaboración propia.

5.6.5.1. Modificar registro

Al dar clic en el botón de “Modificar Registro” se cargará la pantalla de mantenimiento con la información del registro seleccionado, donde usted podrá modificar los datos.

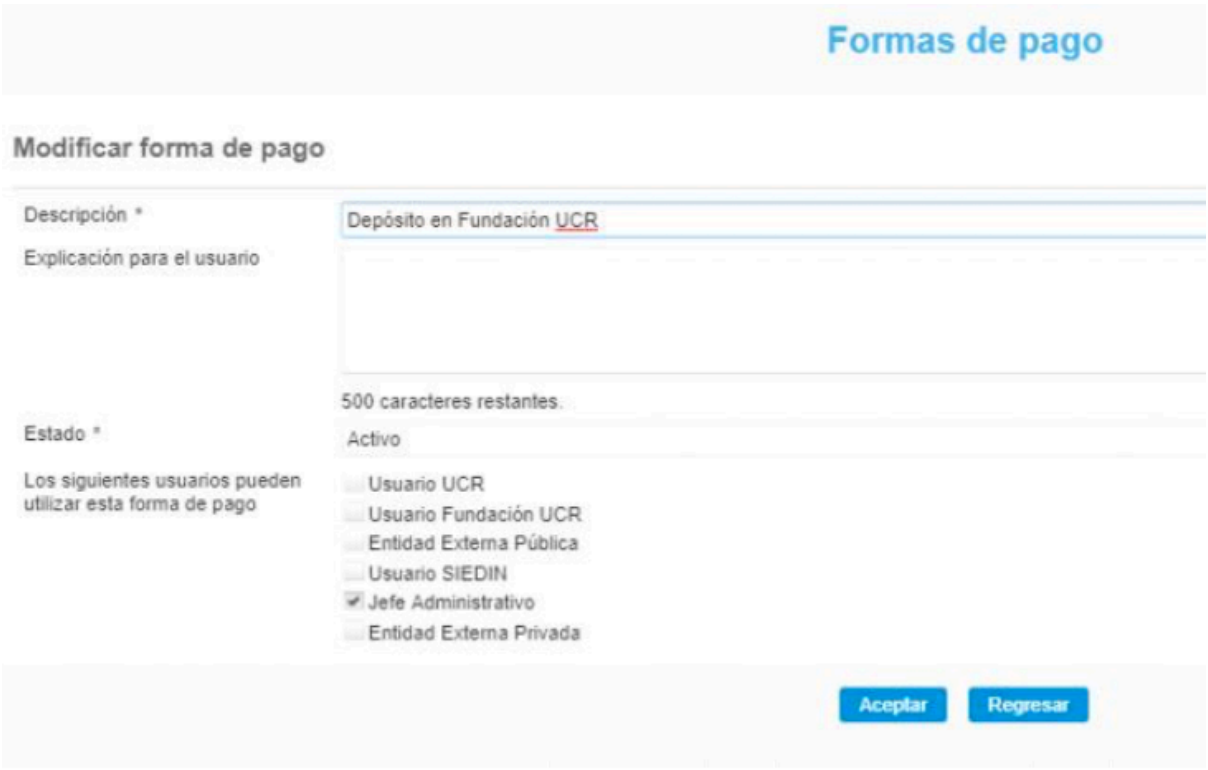


Figura 82. Modificar registro

Fuente: Elaboración propia.

Presione el botón “Aceptar” para modificar la información o el botón “Regresar” para volver al listado sin realizar ningún cambio.

5.6.6. Consulta de notificaciones

La opción de consulta de notificaciones permite consultar todas las notificaciones enviadas por el sistema partiendo de criterios de búsqueda.

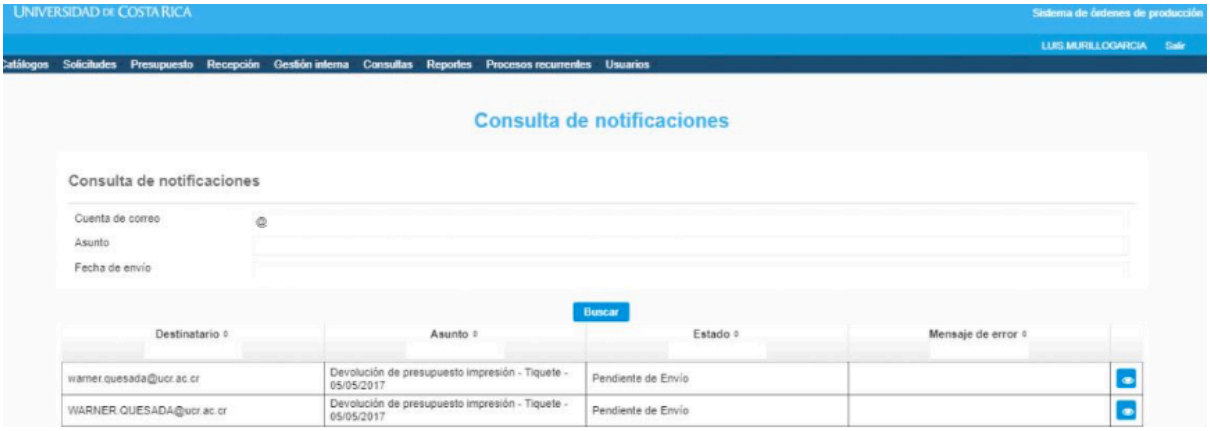


Figura 83. Pantalla de búsqueda de notificaciones

Fuente: Elaboración propia.

5.6.6.1. Ver notificación

Al dar clic en el botón de “Ver Notificación”  se cargará la pantalla de visualización del correo.

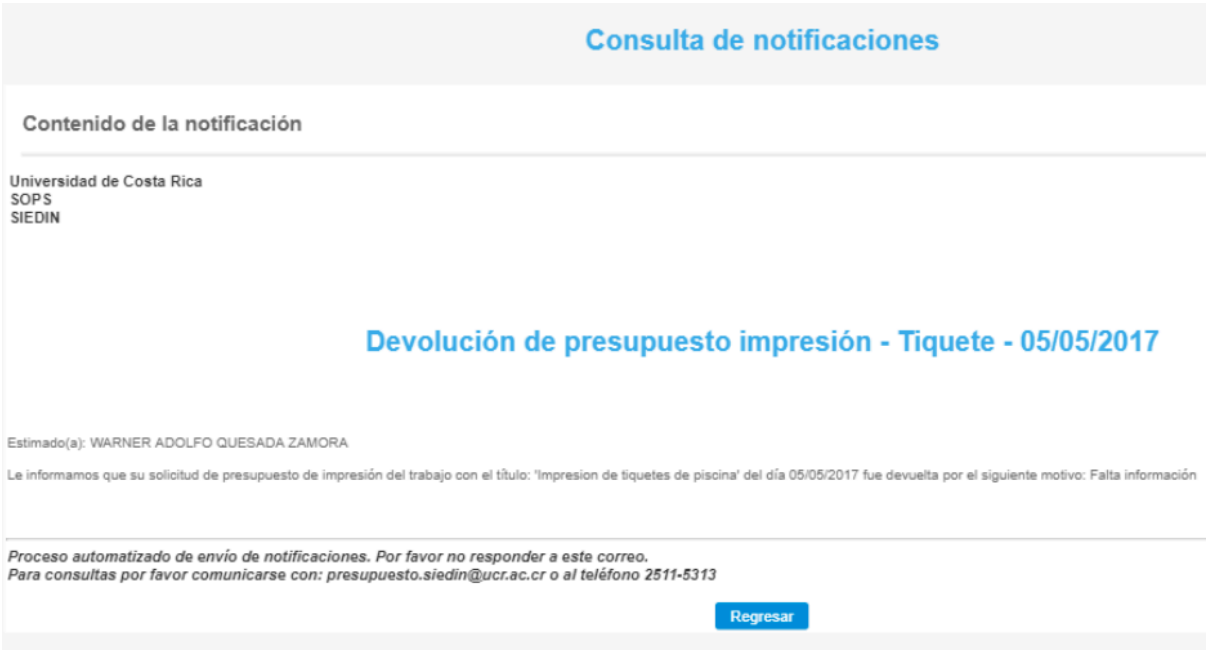


Figura 84. Ver notificación

Fuente: Elaboración propia.

Presione el botón “Regresar” para volver a la pantalla de búsqueda de notificaciones.

5.6.7. Ejecutar procesos recurrentes de forma manual

La opción de procesos recurrentes permite ejecutar cualquiera de los procesos recurrentes establecidos en el sistema de forma manual, en caso de que haya ocurrido algún problema.

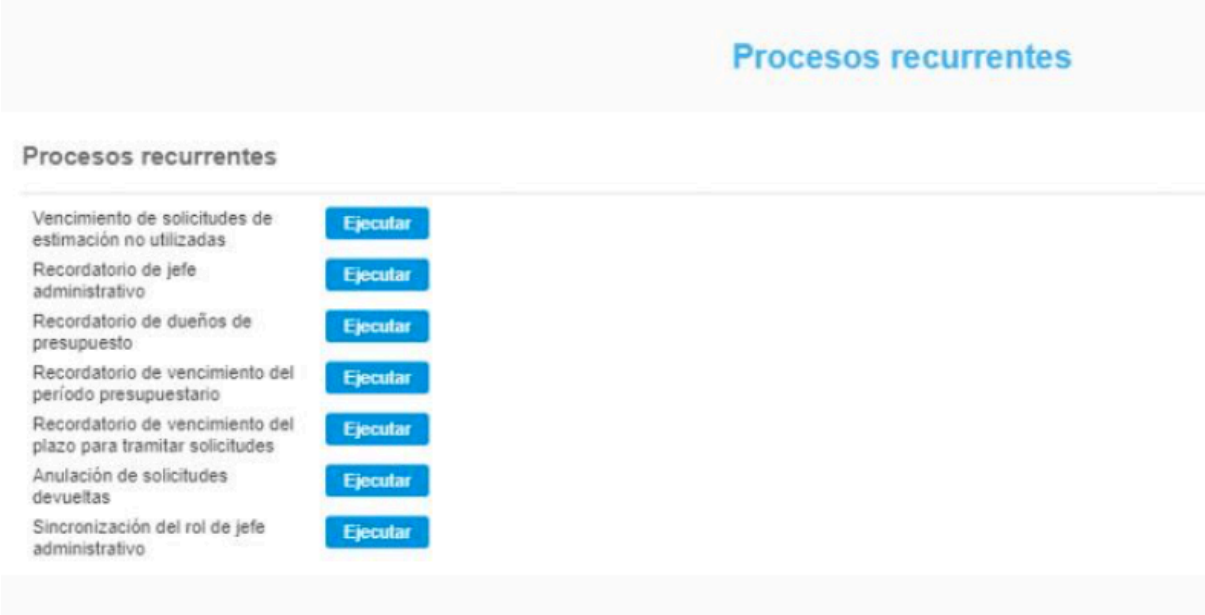


Figura 85. Pantalla de procesos recurrentes

Fuente: Elaboración propia.

5.6.8. Asignación de roles a usuario

La opción de asignación de roles permite configurar a cuáles pantallas tiene acceso cada usuario, ocultando las que no están configuradas para cada rol.

Figura 86. Pantalla de asignación de roles

Fuente: Elaboración propia.

5.6.8.1. Agregar roles

Al dar clic en el botón de “Búsqueda”  se cargará la información del usuario indicado y los roles habilitados para este.

Figura 87. Configuración de roles

Fuente: Elaboración propia.

Presione el botón “Guardar” para asociar los roles que se encuentren marcados con el usuario.

5.6.9. Pantalla de cierre de sesión

En la parte superior derecha de todas las pantallas se encuentra el botón Salir, el cual cierra la sesión para el usuario, mostrando la pantalla de cierre de sesión.

Figura 88. Pantalla de cierre de sesión

Fuente: Elaboración propia.

5.7. RESULTADO DE LA PROPUESTA

Como punto de referencia se toma otro proyecto desarrollado para la Universidad de Costa Rica, el cual tiene el nombre de “Sistema para el examen general básico clínico”, dicho proyecto fue estimado al igual que el “Sistema de órdenes de producción” para una duración total de 11 meses y fue desarrollado en el lenguaje ASP .net.

Este primer proyecto fue iniciado aproximadamente el 01/09/2015 y fue terminado en su totalidad el 20/06/2016, lo cual nos da un tiempo total aproximado de 10 meses.

El primer módulo del “Sistema de órdenes de producción” se comenzó a desarrollar aproximadamente el 01/11/2016 y fue terminado el 27/03/2017, con un tiempo aproximado de 4 meses, siendo este primer módulo la etapa más extensa del proyecto, ya que las demás etapas -4 en su totalidad, tomando en cuenta la ya finalizada- contemplan la integración de otros tipos de usuario al sistema para que también puedan tramitar sus solicitudes y nuevas funcionalidades.

Dicho esto, aún se cuenta con 7 meses para terminar el desarrollo completo del sistema, realizar pruebas integrales en su totalidad, implementar el sistema y capacitar a los usuarios, lo cual nos da un buen panorama de que el proyecto pueda ser terminado según las expectativas, sin que el cambio de lenguaje afecte al usuario final.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- Se concluye que la aplicación de estándares y una metodología de desarrollo a la hora de crear una aplicación web, permite trabajar de una forma más acelerada de parte de todos los miembros del equipo involucrado en este proceso, ya que se cuenta con una base para empezar a trabajar reduciendo los tiempos de entrega y mejorando la calidad del producto.
- Se determina que el lenguaje de programación en el cual se desarrolla una aplicación no es un factor determinante para el éxito del proyecto, más bien hay que buscar la opción que más se adecue a la empresa, al tipo de proyecto y al equipo de desarrollo, partiendo de los beneficios y puntos en contra que conlleva cada uno de los lenguajes.
- Se concluye que el utilizar estándares de programación a nivel de código y base de datos facilita la legibilidad de los objetos y métodos, lo cual permite que cualquier miembro del equipo pueda entender lo que se está haciendo más fácilmente e incluso poder hacer ajustes posteriores de una forma más rápida.
- Se concluye que una metodología de desarrollo permite establecer un esquema de trabajo más marcado, mediante el cual es posible dar una cantidad de tareas específicas en un tiempo determinado a cada integrante, lo cual permite realizar mediciones más concisas sobre lo que está realizando cada uno, e identificar posibles atrasos o puntos de alerta, para poder así realizar los ajustes respectivos y cumplir con la meta.

- Se concluye que gracias a la aplicación de estándares y la metodología de desarrollo propuesta, el primer módulo del “Sistema de órdenes de producción” fue terminado dentro de los tiempos estimados de desarrollo, con una duración aproximada de 4 meses.

6.2. RECOMENDACIONES

- Para el éxito de un proyecto en lenguaje Java es importante tener previamente definidos y configurados los servidores que van a alojar tanto la aplicación como la base de datos, ya que a pesar de que un proyecto ha sido desarrollado en su totalidad, no implica que vaya a poder ser accedido por los usuarios finales.
- Se recomienda utilizar capacitaciones o algún método similar a todo nuevo miembro de un equipo de desarrollo, para que tenga un conocimiento básico del esquema y forma de trabajo.
- Se recomienda tomar en cuenta la base de datos en la cual se va a implementar un proyecto y no solo el lenguaje de programación, ya que cada una de ellas tiene un costo diferente y sus propias ventajas y limitaciones, por ejemplo, en MySql no se pueden crear secuencias para tener un campo de base de datos autoincremental, o vistas materializadas y otras opciones más complejas que ofrece una base de datos Oracle, por lo que se debe valorar según el alcance del proyecto si va a funcionar o es necesario buscar otra alternativa.

BIBLIOGRAFÍA CONSULTADA

Anghel, L. (2016). What is PrimeFaces? Disponible en: http://www.developer.com/java/data/what-is-primefaces.html

Anónimo. (2004). Ventajas y desventajas del asp. Disponible en: http://www.lawebdelprogramador.com/foros/ASP/381741-Ventajas-y-desventajas-del-asp.html

Barba, A. (2015). Eventos de Scrum. Revisión de Sprint. Disponible en: http://www.becominganagilearchitect.com/scrum-eventos-revision-de-sprint

Berzal, F. (2005). El ciclo de vida de un sistema de información. (p.3). Disponible en: http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf

Berzal, F. (2005). Introducción a la programación Java. Disponible en: http://elvex.ugr.es/decsai/java/pdf/2B-Java.pdf

Bien, A. (2009). Java EE or .Net. Disponible en: http://www.adam-bien.com/roller/abien/entry/java_ee_or_net_an

Casanova, S. (2015). Reunión diaria de Scrum, más allá de las 3 preguntas. Disponible en: http://samuelcasanova.com/2015/03/reunion-diaria-de-scrum-mas-alla-de-las-3-preguntas/

Castañeda Bueno, L. (2010). Ingeniería de requerimientos: ciclo de vida y requerimientos de software. Disponible en: https://repository.icesi.edu.co/biblioteca_digital/bitstream/item/4083/1/Presentacion_ciclo_vida_software.pdf

Cohen, Karen D. y Asín Lares, E. (2009). Tecnologías de información en los negocios. 5ª ed. (p. 304). México: McGraw-Hill/Interamericana Editores.

Cohen, Karen, D. y Asín Lares, E. (2009). Tecnologías de información en los negocios. 5ª ed. (p. 2). México: McGraw-Hill/Interamericana Editores

Educaweb. (s.f.). Analista de sistemas informáticos. Disponible en: http://www.educaweb.com/profesion/analista-sistemas-informaticos-362/

Fernández Alarcón, V. (2006). Desarrollo de sistemas de información. 1ª ed. (p. 16). España: Edicions-UPC.

Fernández Alarcón, V. (2006). Desarrollo de sistemas de información. 1ª ed. (p. 17). España: Edicions-UPC.

Garzás, J. (2014). Scrum Master: recopilación de datos y consideraciones importantes. Disponible en: http://www.javiergarzas.com/2014/09/scrum-master-consideraciones-importantes.html

Hibernate community. (s.f.). Chapter 4. Tutorial Using the Java Persistence API (JPA). Disponible en: https://docs.jboss.org/hibernate/orm/3.6/quickstart/en-US/html/hibernate-gsg-tutorial-jpa.html

Instituto Nacional de Educación Tecnológica. (2006). Perfil Profesional Técnico en Programación. Disponible en: http://www.cicomra.org.ar/cicomra2/archivos/notas/Perfil%20Prof%20Tecnico%20Programador%20V6.pdf

Instituto Técnico de Sonora. (s.f.). Introducción a los sistemas de información. Disponible en: http://biblioteca.itson.mx/oa/dip_ago/introduccion_sistemas/p3.htm

Kendall, K. y Kendall, J. (2005). Análisis y diseño de sistemas. 5ª ed. (p. 10). México: Pearson.

LIDER Integrated Technology Consulting. (s.f.). ¿Qué es un ERP? Disponible en: http://www.andece.org/adheridos/images/stories/LIDER_IT/Qu-es-un-ERP.pdf

López, A. (2008). Introducción a Java. Disponible en: https://aureliux.files.wordpress.com/2008/07/curso1-3.pdf

MADO. (2017). Sistemas hechos a la medida. Disponible en: http://www.ecuaportales.com/mado/index.php/noticias-publicidad/189-sistemas-hechos-a-la-medida

Martínez, J. (2014). Java vs .NET: Una discusión bizantina. Disponible en: http://destinodotnet.com/java-vs-net-una-discusion-bizantina/

Mercedes Cáceres, A. (2006). Ciclo de vida de desarrollo de sistemas. Disponible en: http://rcruz0423.galeon.com/docs/clase2.pdf

Microsoft. (2017). Información general acerca de .Net Framework. Disponible en: https://msdn.microsoft.com/es-es/library/zw4w595w(v=vs.110).aspx

Microsoft. (2017). Requisitos de desarrollo en ASP .NET. Disponible en: https://msdn.microsoft.com/es-es/library/ms228041(v=vs.80).aspx

Microsoft. (2017). Suscripción a Visual Studio Enterprise (Nueva). Disponible en: https://www.microsoftstore.com/store/mslatam/es_MX/pdp/Suscripci%C3%B3n-a-Visual-Studio-Enterprise/productID.5094454500

Microsoft. (s.f.). Reunión retrospectiva. Disponible en: https://msdn.microsoft.com/es-es/library/ee191586(v=vs.100).aspx

Moreno Navarro, J. (2014). Arquitecturas software. Disponible en: http://babel.ls.fi.upm.es/~fred/sbc/arquitecturas_sw.pdf

NetBeans Forums. (2010). Hibernate mapping files and POJO from database. Disponible en: https://forums.netbeans.org/topic23146.html

OBS Business School. (2016). Las 5 etapas en los “Sprints” de un desarrollo Scrum. Disponible en: http://www.obs-edu.com/int/blog-investigacion/project-management/las-5-etapas-en-los-sprints-de-un-desarrollo-scrum

Pérez, J. (2015). Principales lenguajes de programación web, ventajas y desventajas. Disponible en: http://www.registrodominiosinternet.es/2013/08/lenguajes-programacion-web-ventajas.html

PMI. (2017). Who are Project Managers? Disponible en: https://www.pmi.org/about/learn-about-pmi/who-are-project-managers

Proyectos Ágiles. (s.f.). Equipo (*team*). Disponible en: https://proyectosagiles.org/equipo-team/

Proyectos Ágiles. (s.f.). Lista de objetivos / requisitos priorizada (*Product Backlog*). Disponible en: https://proyectosagiles.org/lista-requisitos-priorizada-product-backlog/

Quacquarelli Symonds. (2016). University Rankings Latin America. Disponible en: http://www.topuniversities.com/university-rankings/latin-american-university-rankings/2016

RAE. (2017). Diccionario de la lengua Española. Disponible en: http://dle.rae.es/?id=bBsqKPc

Rouse, M. (2015). Sprint (software development). Disponible en: http://searchsoftwarequality.techtarget.com/definition/Scrum-sprint

Schwaber, K. y Sutherland, J (2013). La guía de scrum. (p.4). Disponible en: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Scrum Manager. (2014). Historia de usuario. Disponible en: http://www.scrummanager.net/bok/index.php/Historia_de_usuario

Scrum Manager. (2016). Propietario del producto. Disponible en: http://www.scrummanager.net/bok/index.php?title=Propietario_del_producto

Semanario Universidad. (2011). UCR utilizara “software” libre. Disponible en: http://semanariouniversidad.ucr.cr/universitarias/ucr-utilizar-software-libre/

Significados. (s.f.). Significado de encuesta disponible en: https://www.significados.com/encuesta/

SmarterAsp. (2017). Superior ASP.NET Hosting. Disponible en: http://www.smarterasp.net/index

Sonsale, D. (2010). How to generate POJOs from Database from HBM Annotated java beans. Disponible en: http://sonsale.blogspot.com/2010/06/how-to-generate-pojos-from-database-for.html

StackOverflow. (2011). What is a named query? Disponible en: http://stackoverflow.com/questions/4517069/what-is-a-named-query

Sun Microsystems. (s.f.). Java. Disponible en: https://www.arkaitzgarro.com/java/capitulo-1.html

Tanuki Software. (2017). Java Service Wrapper. Disponible en: https://wrapper.tanukisoftware.com/doc/spanish/product-overview.html

Trigo Aranda, V. (s.f.). Historia y evolución de los lenguajes de programación. Disponible en: https://www.acta.es/medios/articulos/informatica_y_computacion/034083.pdf

Universidad de Alicante. (2014). Características de JSF. Disponible en: http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html

Universidad de Chile. (s.f.). Java virtual machine. Disponible en: http://www.cec.uchile.cl/~luvasque/edo/java/manuales/JVM%20y%20variables%20de%20entorno.pdf

Universidad de Costa Rica (2014). 5 Razones para Migrar a Software Libre [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=SBw9wpO0p30

Universidad de Costa Rica. (2016). Historia de la Universidad de Costa Rica. Disponible en: http://www.ucr.ac.cr/acerca-u/historia-simbolos/historia.html

Universidad de Costa Rica. (s.f.). La migración. Disponible en: http://www.migracion.ucr.ac.cr/lamigracion

Universidad de Costa Rica. (s.f.). Misión de la Universidad de Costa Rica. Disponible en: http://www.so.ucr.ac.cr/mision-y-vision

Universidad de Costa Rica. (s.f.). Visión de la Universidad de Costa Rica. Disponible en: http://www.so.ucr.ac.cr/mision-y-vision

Wong, H. (2012). Java – Configurar SVN con Netbeans. Disponible en: http://www.programandoconcafe.com/2012/03/java-configurar-svn-con-netbeans.html

Zachman, J. “A Framework for Information Systems Architecture”. IBM Systems Journal. 1987. 26(3): 276-292.

APÉNDICES

Encuesta a trabajadores de la UCR

1. Indique el nivel de conocimiento que tiene sobre el lenguaje Java
    - Nulo
    - Poco
    - Mediano
    - Avanzado
2. ¿Ha recibido capacitación sobre el lenguaje Java?
    - Sí
    - No
3. ¿Cree usted que el cambio de lenguaje Visual Basic a Java trae beneficios?
    - Sí
    - No
4. ¿Cree usted que el cambio de lenguaje Visual Basic a Java trae tiempos mayores de entrega en los proyectos?
    - Sí
    - No
5. ¿Ve necesaria la creación de métodos y estándares de programación sobre el lenguaje Java?
    - Si
    - No
6. ¿Cuenta la institución con las herramientas y equipo adecuado para una arquitectura en lenguaje Java
    - Sí
    - No

ANEXOS

CONTRATACIÓN DIRECTA No. 2016CD-000053-CI

"Contratación de Servicios profesionales para el desarrollo y la implementación del Sistema de Ordenes de Producción para el Sistema Editorial y de Difusión de la Investigación (SIEDIN)."

La Unidad de Gestión de Adquisiciones del Centro de Informática de la Universidad de Costa Rica recibirá ofertas por escrito para la compra citada:

Fecha de apertura de ofertas: Martes 9 de Agosto del 2016 a las 9:30 a.m.

Renglón	Ud	Cantidad	Descripción de la Contratación
1	Ud	1	**Contratación de un Servicio Profesional para la programación e implementación del Sistema de Ordenes de Producción para el Sistema Editorial y de Difusión de la Investigación y procesos relacionados.** • Desarrollo de las funcionalidades de gestión de procesos de ingreso y seguridad. • Desarrollo de las funcionalidades de catálogos del sistema. • Desarrollo de las funcionalidades de gestión de procesos de creación y aprobaciones de las solicitudes de órdenes de producción. • Desarrollo de las funcionalidades de gestión de procesos de control presupuestario y aprobaciones. • Desarrollo de las funcionalidades de gestión de procesos de autorizaciones del jefe administrativo. • Desarrollo de las funcionalidades de gestión de procesos de aprobaciones Dirección. • Desarrollo de las funcionalidades de gestión de procesos de trazabilidad de las órdenes de producción. • Desarrollo de las funcionalidades de generación de solicitudes de reclamos. • Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción. • Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción trabajos adicionales. • Desarrollo de las funcionalidades de gestión de órdenes de producción de emergencia. • Desarrollo de las funcionalidades de gestión de procesos de entidades externas y control de usuarios autorizados. • Desarrollo de las funcionalidades de gestión de procesos de gestión de órdenes de producción para entidades externas (control de usuarios autorizados). • Desarrollo de las funcionalidades de gestión de procesos de solicitudes de producción en recepción, reimpresiones y recepción de reclamos. • Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con SIAF. • Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con FileMaker. • Desarrollo de las funcionalidades de reportes y consultas del sistema.

7./

			• Creación de guías rápidas y videos tutoriales. • Implementación del sistema y acompañamiento.

Objetivo General:

Objetivo General: Crear un sistema de información al Sistema Editorial y de Difusión de la Investigación (SIEDIN) para la gestión de las órdenes de producción.

Objetivos Específicos:

Desarrollar los componentes de las aplicaciones según los requerimientos técnicos, que permitan crear la aplicación para el control y seguimiento de las órdenes de producción, así como las aplicaciones administrativas del sistema y/o procesos relacionados.

a) Programar, probar e implementar los catálogos y mantenimientos del sistema y sus procesos relacionados.
b) Programar, probar e implementar los procesos asociados a las reglas de negocios.
c) Programar, probar e implementar los reportes del sistema y sus procesos relacionados.
d) Documentar y ejecutar las pruebas correspondientes a cada módulo de la aplicación y a los procesos relacionados.
e) Preparar las documentaciones técnicas e instructivas de usuario de cada módulo de la aplicación y sus procesos relacionados.

I. Especificaciones Técnicas

Requerimientos técnicos del sistema:

Se detallan las etapas que componen el sistema a desarrollar, con un criterio de agrupamiento de mantenimientos, procesos, consultas, reportes y procesos relacionados.

1: Etapa I: Gestión de solicitudes de órdenes de producción.

Mantenimientos

a. Catálogos del sistema

Procesos:

b. Procesos de ingreso y seguridad.
c. Procesos de creación y aprobaciones de las solicitudes de órdenes de producción.
d. Procesos de control presupuestario y aprobaciones.
e. Procesos de autorizaciones del jefe administrativo.
f. Procesos de aprobaciones Dirección.
g. Procesos de trazabilidad de las órdenes de producción.

Procesos Relacionados:

h. Manuales I Etapa

2: Etapa II: Facturación y reclamos de solicitudes de órdenes de producción.

Mantenimientos

a. Generación de solicitudes de reclamos

Procesos:

b. Procesos de facturación de órdenes de producción.
c. Procesos de facturación de órdenes de producción trabajos adicionales.
d. Gestión de órdenes de producción de emergencia.

Procesos Relacionados:

e. Manuales II Etapa

3: Etapa III: Gestión de entidades externas.

Procesos:

a. Proceso de entidades externas y control de usuarios autorizados.
b. Proceso de gestión de órdenes de producción para entidades externas (control de usuarios autorizados)

Procesos Relacionados:

c. Manuales III Etapa

4: Etapa IV: Gestión de órdenes de producción urgentes y en recepción.

Procesos:

a. Procesos de solicitudes de producción en recepción, reimpresiones y recepción de reclamos.

Procesos Relacionados:

b. Manuales etapa IV

5: Etapa V: Gestión de Interfaces con SIAF y FILEMAKER

Procesos:

a. Proceso de incorporación de interfaz con SIAF.
b. Proceso de incorporación de interfaz con FileMaker.

M.I

Procesos Relacionados:

c. Reportes y Consultas
d. Creación de guías rápidas y videos tutoriales.
e. Implementación del sistema y acompañamiento.

1. Especificaciones o requerimientos técnicos mínimos del renglón 1:

Renglón 1. Contratación de un Servicio Profesional para la programación e implementación del Sistema de Ordenes de Producción para el Sistema Editorial y de Difusión de la Investigación y procesos relacionados.

- Desarrollo de las funcionalidades de gestión de procesos de ingreso y seguridad.
- Desarrollo de las funcionalidades de catálogos del sistema.
- Desarrollo de las funcionalidades de gestión de procesos de creación y aprobaciones de las solicitudes de órdenes de producción.
- Desarrollo de las funcionalidades de gestión de procesos de control presupuestario y aprobaciones.
- Desarrollo de las funcionalidades de gestión de procesos de autorizaciones del jefe administrativo.
- Desarrollo de las funcionalidades de gestión de procesos de aprobaciones Dirección.
- Desarrollo de las funcionalidades de gestión de procesos de trazabilidad de las órdenes de producción.
- Desarrollo de las funcionalidades de generación de solicitudes de reclamos.
- Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción.
- Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción trabajos adicionales.
- Desarrollo de las funcionalidades de gestión de órdenes de producción de emergencia.
- Desarrollo de las funcionalidades de gestión de procesos de entidades externas y control de usuarios autorizados.
- Desarrollo de las funcionalidades de gestión de procesos de gestión de órdenes de producción para entidades externas (control de usuarios autorizados).
- Desarrollo de las funcionalidades de gestión de procesos de solicitudes de producción en recepción, reimpresiones y recepción de reclamos.
- Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con SIAF.
- Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con FileMaker.
- Desarrollo de las funcionalidades de reportes y consultas del sistema.
- Creación de guías rápidas y videos tutoriales.
- Implementación del sistema y acompañamiento.

7.1

3. Cronograma de tareas y actividades:

Cronograma para la ejecución y prestación de servicios del adjudicatario		
Producto	**Duración /Días**	**Porcentaje**
Desarrollo de las funcionalidades de gestión de procesos de ingreso y seguridad	4.5	2.5
Desarrollo de las funcionalidades de catálogos del sistema	6	3
Desarrollo de las funcionalidades de gestión de procesos de creación y aprobaciones de las solicitudes de órdenes de producción	12	5.5
Desarrollo de las funcionalidades de gestión de procesos de control presupuestario y aprobaciones	7.5	3.4
Desarrollo de las funcionalidades de gestión de procesos de autorizaciones del jefe administrativo	6	2.7
Desarrollo de las funcionalidades de gestión de procesos de aprobaciones Dirección	2.5	1.5
Desarrollo de las funcionalidades de gestión de procesos de trazabilidad de las ordenes de producción	4	1.8
Desarrollo de las funcionalidades de generación de solicitudes de reclamos	3	1.3
Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción	5	2.3
Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción trabajos adicionales	5	2.3
Desarrollo de las funcionalidades de gestión de órdenes de producción de emergencia	3	1.4
Desarrollo de las funcionalidades de gestión de procesos de entidades externas y control de usuarios autorizados	14	6.5
Desarrollo de las funcionalidades de gestión de procesos de gestión de órdenes de producción para entidades externas (control de usuarios autorizados)	15.5	7
Desarrollo de las funcionalidades de gestión de procesos de solicitudes de producción en recepción, reimpresiones y recepción de reclamos	12	5.5
Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con SIAF	20	9

Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con FileMaker	20	9
Desarrollo de las funcionalidades de reportes y consultas del sistema	10	4
Creación de guías rápidas y videos tutoriales	10	4
Implementación del sistema y acompañamiento	60	27.3
TOTALES	**220**	**100**

4. Procedimientos de control de calidad

Un funcionario del Centro de Informática, verificará los productos entregados por el programador para que cumplan con los requerimientos especificados tomando en cuenta lo siguiente:

1. El Analista de Sistemas del Centro de Informática que validará los productos entregados, debe ser capaz de evacuar dudas.

2. El adjudicatario deberá utilizar los estándares y metodología utilizadas por la Universidad de Costa Rica, así como el uso de herramientas de programación, Visual Basic .NET, Oracle/PL SQL, entre otras. La Universidad de Costa Rica cuenta con un Arquitecto de Software encargado del desarrollo lógico del sistema de información del Sistema Editorial y de Difusión de la Investigación (SIEDIN) para la gestión de las órdenes de producción.

3. El adjudicatario deberá hacer un plan de pruebas, acorde al proceso de desarrollo del Área de Desarrollo de Sistemas del Centro de Informática de la Universidad de Costa Rica, para la correspondiente entrega de los productos (programas) generados. Se validará las características funcionales y no funcionales de los productos finalizados, de forma tal que se compruebe la correcta funcionalidad del producto realizado, el cual debe ser ejecutado y aprobado por: el analista, y el usuario experto asignado al proyecto.

4. El Arquitecto de Software verificará el cumplimiento de estándares y la validación del buen funcionamiento de los requerimientos indicados, además se ejecutaran los planes de prueba elaborados por el adjudicatario por un funcionario designado y con las competencias adecuadas para esta labor.

5. El producto será aprobado una vez satisfaga los requerimientos indicados por el usuario experto en cumplimiento de cada uno de los objetivos planteados para el sistema.

7.1

II. Condiciones Invariables

a) Aspectos Generales

1. Vigencia de la oferta

Las ofertas deberán tener una vigencia no menor a 30 días hábiles después del día siguiente hábil a la fecha de la apertura de ofertas.

2. Plazo de adjudicación

10 días hábiles.

3. Lugar de Entrega

Centro de Informática de la Universidad de Costa Rica.

4. Plazo de Entrega

Once (11) meses.

b) Requisitos y Condiciones Especiales

1. Calidades Mínimas del Oferente:

1.1 El oferente debe ser un profesional con un grado mínimo de Bachillerato en una Carrera de Computación e Informática o Ingeniería en Sistemas reconocida por el CONESUP, quien ejecutará las labores a desarrollar.

1.2 Experiencia demostrada: Conocimiento en programación con .Net, java script y desarrollo Web.

Nota:
Se considera "Profesional en el área de computación e informática" a aquella persona que tenga el grado académico a nivel universitario (Bachillerato, Licenciatura o superior) referente a la carrera de Computación e Informática o Ingeniería en Sistemas.

1.3 El oferente debe aportar el título universitario (grado mínimo de Bachiller en una carrera de Computación e Informática o Ingeniería en Sistemas de Computación o similar reconocida por el CONESUP).

2. Garantía

2.1 La garantía mínima sobre los productos del servicio deberá ser de **3 meses**, contados a partir del recibido conforme y por escrito del informe final por parte del encargado del proyecto. La Universidad de Costa Rica se reserva el derecho de verificar la información suministrada.

M./

2.2 El oferente deberá especificar los beneficios de la garantía y deberá especificar también las exclusiones. Las exclusiones que no queden explícitas en la oferta no serán válidas.

3. Presentación de Oferta Base y Oferta Alternativa

Deberá presentar las ofertas en forma escrita y digital (tipo de archivo PDF), separando la información legal, técnica y de precios en forma ordenada y clara.
En caso de que el oferente presente varias ofertas, debe indicar claramente cual es su oferta base, cual es su oferta alternativa y se deben garantizar los siguientes puntos:

3.1 La(s) Oferta(s) Base será(n) aquella(s) que cumple(n) con todas las especificaciones técnicas incluidas en los requerimientos mínimos solicitados en el cartel.

3.2 La(s) Oferta(s) Alternativa(s) existe siempre en función y dependencia de la oferta base y si bien puede mejorar en todos los aspectos esta última; la posibilidad de adjudicar una oferta alternativa se dará solamente cuando la misma empresa haya ganado de previo con la oferta base. Se considerará como una oferta base si la misma no ofrece mejora o ventajas mayores a las requeridas en el cartel.

4. Criterios de Evaluación de Ofertas.

Tabla # 1

Factores a Evaluar

Factor	Puntos
Precio	80
Capacidades y experiencia	20
Total	100

Para asignar la puntuación de los rubros anteriores, se hará por línea de acuerdo a la siguiente fórmula:

PRECIO: La oferta que ofrezca el menor precio obtendrá 80 puntos.

Para la determinación de la puntuación a asignar a las demás ofertas económicas se utilizará la siguiente fórmula:

$$PP = \left(\frac{P_{\min}}{P_{oferta}} \right) * PT$$

Donde:
PP = Puntaje precio obtenido por la empresa
Poferta = Precio ofertado por la empresa en la evaluación
Pmin = Precio mínimo ofertado entre las empresas participantes.
PT = Puntaje total del Factor (80)

7.1

Nota: El límite del precio lo constituye el disponible presupuestario para la presente licitación.

4.1 Factores de calificación:

Precio: 80%

Capacidades y experiencia: 20%

El oferente debe indicar en forma expresa (mediante cartas o certificaciones de los lugares, empresas o instituciones donde ha realizado o desarrollado proyectos o practicas con los puntos aquí descritos) la experiencia y capacidad con que cuenta para realizar las labores en su calidad de desarrollador, los cuales serán calificados de la siguiente manera.:

Metodologías ágiles, Scrum	3
Implementación de software Visual Basic .net, c#, Team Fundation	3
Base de datos Oracle PL/SQL	3
Arquitectura Orientada a Servicios (SOA)	3
Manejo de reporting Services	3
Tecnologías HTML5, CSS3, JavaScript, y otras tecnologías web	5

5. Aspectos generales de la evaluación

5.1 Base de calificación.
La máxima cantidad que pueda obtener un oferente es de 100 puntos. La oferta elegible que obtenga el mayor puntaje será la adjudicada.

5.2 Criterios para el redondeo.
Para los cálculos de puntaje se utilizarán dos decimales truncados (nn.dd).

5.3 Porcentaje mínimo de adjudicación.
La oferta válida para ser adjudicada debe obtener como mínimo 80 puntos, en caso contrario la oferta será descartada.

5.4 Criterio de desempate.
En caso de presentarse empate en la calificación de una oferta se utilizará, como criterio para el desempate la siguiente prioridad:

5.4.1 Se utilizará la parte entera mas dos decimales truncados (nn.dd) del resultado obtenido de la aplicación de la fórmula del punto A.

5.4.2 En caso de persistir empate, la Administración convocará por escrito con tres días de antelación a la fecha en que se resolverá el desempate, a los representantes legales de los oferentes que se encuentren en situación de

M./

empate, para efectuar una rifa y así seleccionar el adjudicatario, la cual será efectuada en el centro de Informática. Cada oferente tomará al azar un papel donde en uno de ellos se detallará la palabra "adjudicatario", el resto estarán en blanco; el oferente que tenga el papel con la palabra antes indicada, será el adjudicatario.
La no asistencia de las partes no impedirá la realización de la rifa.
De lo actuado se levantará un acta que se incorporará al expediente.

6 Encargado general del Contrato.

El Centro de Informática designará para la adecuada ejecución del contrato al señor Luis Jiménez C., así mismo, el encargado podría designar a un profesional o técnico, para la fiscalización de tareas, actividades y productos a realizarse por el adjudicatario.

7 Ejecución del contrato de servicios.

- El adjudicatario deberá ejecutar los servicios contratados bajo la filosofía del trabajo en equipo.
- Debe mantener actualizados los conocimientos y destrezas relacionadas con los objetivos y temáticas del contrato.

8 Prestación de servicios técnicos o profesionales.

- La prestación de servicios se realizará bajo la Coordinación del Área de Desarrollo de Sistemas del Centro de Informática.
- El o la coordinador(a) de la Unidad o Área y el adjudicatario en forma conjunta realizarán un plan de trabajo para la correcta ejecución del contrato, estableciendo reuniones para las revisiones de avance del proyecto y fechas de los entregables o productos, durante el período de contratación, acorde al cronograma.
- El adjudicatario informará el desarrollo de los servicios a la Coordinación de la Unidad o Área respectiva del Centro de Informática.
- La prestación de servicios se realizará según el *"cronograma establecido para la ejecución y prestación de servicios"*.
- El adjudicatario se compromete a trasladarse a las Unidades de la Universidad de Costa Rica, cuando se requiera, según lo establecido en las condiciones de la contratación.
- El adjudicatario deberá utilizar los estándares y metodología utilizadas por la Universidad de Costa Rica, así como el uso de herramientas de programación, Visual Basic .NET, Oracle/PL SQL, entre otras. La Universidad de Costa Rica cuenta con un Arquitecto de Software encargado del desarrollo lógico del **Sistema de Ordenes de Producción para el Sistema Editorial y de Difusión de la Investigación (SIEDIN)**

- El adjudicatario deberá hacer un plan de pruebas, acorde al proceso de desarrollo del Área de Desarrollo de Sistemas del Centro de Informática de la Universidad de Costa Rica, para la correspondiente entrega de los productos (programas) generados. Se validará las características funcionales y no funcionales de los productos finalizados, de forma tal que se compruebe la correcta funcionalidad del producto realizado, el cual debe ser ejecutado y aprobado por: el analista, la persona encargada de control de calidad y el usuario experto asignado al proyecto.

M.I

- El Arquitecto de Software verificará el cumplimiento de estándares y la validación del buen funcionamiento de los requerimientos indicados, además se ejecutaran los planes de prueba elaborados por el adjudicatario por un funcionario designado y con las competencias adecuadas para esta labor.
- El producto será aprobado una vez satisfaga los requerimientos indicados por el usuario experto en cumplimiento de cada uno de los objetivos planteados para el sistema.

- **El código fuente de los aplicativos de software y la documentación generada serán propiedad intelectual de la Universidad de Costa Rica. El adjudicatario deberá entregar todos productos en formato digital editable y completo.**

9 Sujeción del contrato.

Con sujeción al acuerdo entre ambas partes, las condiciones de los servicios técnicos o profesionales establecidos, podrán ser modificados según el necesario interés de la Administración (esta modificación se negociara si el trabajo disminuye o aumenta según las condiciones).

10 Derechos y obligaciones del adjudicatario y la Administración.

- El adjudicatario, no será considerado como funcionario de la Administración, dado que su relación se da por la vía de los servicios técnicos o profesionales.
- El adjudicatario, no está exento de pago de impuestos y será su exclusiva responsabilidad el pago de los mismos, cuando aquellos graven las sumas recibidas en virtud del presente contrato y por tanto, las partes convienen en que son de cuenta del adjudicatario los montos que en concepto de cualesquiera impuestos pudieran corresponderle o estaría obligado a satisfacer de conformidad con la legislación tributaria respectiva.

- El adjudicatario, no tendrá derecho a ningún pago por concepto de prestación, subsidio, indemnización, pensión u otra prestación social por parte de la Administración, que no esté expresamente prevista en este Contrato. Por lo tanto, los seguros de enfermedad, invalidez, vejez y muerte, la Póliza de Riesgos del Trabajo con el Instituto Nacional de Seguros y cualquier otra prestación social para los funcionarios y empleados a cargo del adjudicatario, así como a sus dependientes son de cuenta y responsabilidad exclusiva del mismo.

  En estos términos la Administración y el adjudicatario, quedan exentos de cualquier responsabilidad por los conceptos descritos.

  El adjudicatario debe presentar copia del pago de la Póliza de Riesgos del Trabajo con el Instituto Nacional de Seguros al día y un día hábil después de recibir la orden de compra.

- La Administración como encargada general del contrato, podrá establecer períodos de receso al contrato durante los días en que la Universidad se encuentre en períodos de receso, o a solicitud del adjudicatario, sin que esto supere los 30 días naturales consecutivos por año de contrato. Dicho receso debe ser acordado por ambas partes con al menos 15 días naturales de anticipación.

M./

- Toda la infraestructura universitaria, sobre la cual el adjudicatario al realizar los trabajos provoque daños en el equipo, ya sea en su parte funcional u operativa, o en perjuicio a otras propiedades u obras de la Universidad de Costa Rica, deberá reparar el daño en cuestión dentro del plazo y condiciones que fije el encargado general del contrato designado por el Centro de Informática, sin que esto implique un gasto adicional para la Institución, tanto en mano de obra como en repuestos o materiales.
-

11. Cancelación y forma de pago

La forma de pago se regirá bajo las siguientes cláusulas:

- El monto total del proyecto se realizará en pagos, de acuerdo con la entrega de productos establecidos en el cronograma por parte de la Administración y aprobados.
- Para que la Administración autorice un pago de un producto, éste debe ser entregado en el tiempo establecido según el cronograma, cumpliendo con la totalidad de las funcionalidades solicitadas para dicho producto. En caso de que éstas dos condiciones no se cumplan, la administración brindará 3 días hábiles para cumplir con lo solicitado. Luego de cumplido este período de tiempo se multará con un porcentaje de 1% del monto de pago del producto por día de atraso, una vez comunicado oficialmente al adjudicatario. El total de multas aplicadas durante la ejecución del contrato no excederá un 25% del total del Contrato.
- Se cancelarán los montos de pago únicamente a los productos completos.
- Las correcciones solicitadas no deben retrasar la entrega de otros productos establecidos en el cronograma de la contratación. (*).

Para los pagos correspondientes, es requisito:

•La presentación de la factura timbrada por el Ministerio de Hacienda.

•La presentación del informe del adjudicatario que indique: tareas, actividades, productos, conclusiones y/o recomendaciones, del producto entregado.

•Aceptación del producto entregado, por parte de la administración, acorde a los puntos de control y procedimientos definidos, indicando el pago correspondiente en el *"cronograma para la ejecución y prestación de servicios del adjudicatario"*.

•Copia del pago de los seguros de enfermedad, invalidez, vejez y muerte, y/o la Póliza de Riesgos del Trabajo con el Instituto Nacional de Seguros, al día.

M.I

Distribución de Pagos por Producto

•La distribución de pagos, se realizará , según los Productos solicitados en el siguiente cuadro:

Cronograma para la ejecución y prestación de servicios del adjudicatario	
Producto	**Porcentaje**
Desarrollo de las funcionalidades de gestión de procesos de ingreso y seguridad	2.5
Desarrollo de las funcionalidades de catálogos del sistema	3
Desarrollo de las funcionalidades de gestión de procesos de creación y aprobaciones de las solicitudes de órdenes de producción	5.5
Desarrollo de las funcionalidades de gestión de procesos de control presupuestario y aprobaciones	3.4
Desarrollo de las funcionalidades de gestión de procesos de autorizaciones del jefe administrativo	2.7
Desarrollo de las funcionalidades de gestión de procesos de aprobaciones Dirección	1.5
Desarrollo de las funcionalidades de gestión de procesos de trazabilidad de las ordenes de producción	1.8
Desarrollo de las funcionalidades de generación de solicitudes de reclamos	1.3
Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción	2.3
Desarrollo de las funcionalidades de gestión de procesos de facturación de órdenes de producción trabajos adicionales	2.3
Desarrollo de las funcionalidades de gestión de órdenes de producción de emergencia	1.4
Desarrollo de las funcionalidades de gestión de procesos de entidades externas y control de usuarios autorizados	6.5
Desarrollo de las funcionalidades de gestión de procesos de gestión de órdenes de producción para entidades externas (control de usuarios autorizados)	7
Desarrollo de las funcionalidades de gestión de procesos de solicitudes de producción en recepción, reimpresiones y recepción de reclamos	5.5
Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con SIAF	9

M./

Desarrollo de las funcionalidades de gestión de procesos de incorporación de interfaz con File-Maker	9
Desarrollo de las funcionalidades de reportes y consultas del sistema	4
Creación de guías rápidas y videos tutoriales	4
Implementación del sistema y acompañamiento	27.3
TOTAL	**100**

(*) El supervisor del contrato debe comunicar por escrito a la Unidad de Adquisiciones, el incumplimiento del producto por parte del adjudicatario, para que dicha Unidad formalice la entrega o multa correspondiente.

12 Incumplimientos

En caso de incumplimiento de cualquiera de las cláusulas del cartel, la oferta o del contrato, la Universidad de Costa Rica se arroga el derecho de rescindir y/o resolver unilateralmente el contrato, sin responsabilidad alguna para la Institución y sin perjuicio de la aplicación de la normativa correspondiente.

III. CONDICIONES GENERALES

1. Presentación de la oferta:

La recepción de ofertas será en el Centro de Informática de la Universidad de Costa Rica, ubicada entre la Facultad de Ciencias Económicas y la Escuela de Geología.

- Los proveedores interesados en participar deben estar activos en el Sistema de Compras Públicas "*Mer-link*". A los participantes les corresponde aportar los documentos legales y declaraciones juradas que establece la Ley de Contratación Administrativa y su Reglamento (certificaciones sobre la personería jurídica y propiedad de las acciones, copia certificada de la cédula jurídica, declaración jurada de que no le alcanzan las prohibiciones del artículo 22 de la Ley de Contratación Administrativa y los artículos 24.4 y 5.3.2 del Reglamento General de Contratación Administrativa.
- La oferta deberá presentarse por escrito en la fecha y hora que indique la invitación, en sobre cerrado rotulado con el número y el objeto de la Contratación. Toda oferta deberá presentarse en papel corriente, en original, con la firma del adjudicatario o de su representante legal, - *debe aportar la personería jurídica y la certificación emitida por el Registro Público actualizada* - sin tachaduras ni borrones. Cualquier corrección debe ser hecha mediante nota.
- El oferente debe indicar nombre y firma del responsable, así como la descripción completa del servicio. No se aceptarán ofertas hechas a mano.
- La oferta deberá ser firmada por el oferente

M.I

- Toda oferta debe ser cotizada libre de todos los impuestos, indicando el monto y el tipo de impuestos por separado. La Universidad de Costa Rica está exenta de los mismos, según Ley No. 7293, artículo 6, publicada en la "La Gaceta "No. 63 del 31 de marzo de 1992.
- 5. Deberá indicar el monto unitario y total en números y letras.

2. Documentos que deben entregarse.

2.1 El adjudicatario debe presentar certificación emitida por la Caja Costarricense de Seguro Social indicando que se encuentra al día con las obligaciones obrero-patronales.
La inobservancia del *artículo 74 de la Ley Constitutiva de la Caja Costarricense de Seguro Social* se considerará para todos los efectos procesales y legales incumplimiento contractual grave.

2.2 El adjudicatario debe presentar una declaración jurada de que no le alcanzan, las prohibiciones para contratar con la Universidad, a que se refiere el *numeral 22 de la Ley de Contratación Administrativa y en los artículos 19 y siguientes del Reglamento General de la Contratación Administrativa.*

2.3 El adjudicatario debe presentar una declaración jurada de que se encuentra al día en el pago de todo tipo de impuestos nacionales de conformidad con lo dispuesto en el *artículo 65 del Reglamento General de la Contratación Administrativa.*

2.4 Si el adjudicatario es persona física debe presentar fotocopia de la cédula de identidad.

2.5 Cualesquiera otros documentos que se considere oportuno acompañar, según la naturaleza del objeto licitado y el tipo de licitación que se haya promovido.

3. Regulaciones que deben observarse.

Los participantes deberán cumplir con lo que establece la Ley de Contratación Administrativa, el Reglamento General de Contratación Administrativa y otras leyes pertinentes.

4. Formalización del contrato.

En todo lo relacionado con la formalización del contrato deberá cumplirse con lo estipulado en los artículos 188, 189 y 190 numerales del Reglamento General de la Contratación Administrativa.

5. Re adjudicación.

La Administración re adjudicará a la segunda mejor oferta calificada, en caso de incumplimiento por parte del adjudicatario, ya sea en la entrega de los bienes o por no presentar la garantía de cumplimiento en el plazo legal. Esta re adjudicación será aprobada por la Unidad de Compras Especializadas, previo estudio del expediente.

6. En caso de extravío de cualquier documento original, *(recibo de garantía, orden de compra, orden de servicios),* el interesado deberá solicitar su reposición mediante nota dirigida a la Unidad de Compras Especializadas y adjuntará declaración jurada otorgada ante notario público donde manifieste tal situación.

7.1

Estimación Presupuestaria:

Monto Presupuestado: ¢ 11,000,000.00 colones. (Once millones con 00/100)

Para recibo de cotizaciones, utilizar el correo electrónico, deberán presentar la oferta original en un plazo máximo de tres días hábiles.

Notas:
1. En caso de no participar, favor indicar por escrito las razones, para tomarlo en cuenta en futuras contrataciones.
2. Este cartel se rige bajo la Ley de Contratación Administrativa y su Reglamento.

Ing. Gerardo Méndez S.
Encargado de elaboración del Cartel
Unidad de Gestión de Adquisiciones

Lic. Edgardo Baltodano X.
Coordinador A.I.
Unidad de Gestión de Adquisiciones

M.Sc. Luis Jiménez C.
Sub-Director
Centro de Informática

M.Sc. Alonso Castro Mattei
Director
Centro de Informática

GMS.

San José 22 de Agosto de 2017

Universidad Hispanoamericana.

Con respecto al proyecto "Propuesta de métodos y estándares de desarrollo en lenguaje java para la Universidad de Costa Rica, sede Rodrigo Facio en el periodo 2017", elaborado por Mauricio Salas Chaves, número de identificación 2-0706-0790, doy a constar que el mismo satisface las necesidades planteadas por el equipo de desarrollo.

Además el desarrollo del primer módulo del "Sistema de Órdenes de Producción" tuvo una duración aproximada de cuatro meses.

Atentamente,

MAP. Pedro Céspedes Calderón, Director de proyectos
Ced. 1-0825-0470