Sistema de gestión de actividades para la Dirección General de Economía

Trabajo de diploma para optar por el título

de Ingeniero en Ciencias Informáticas

Autor: Grisel Inés Rodríguez Machado

Tutores:

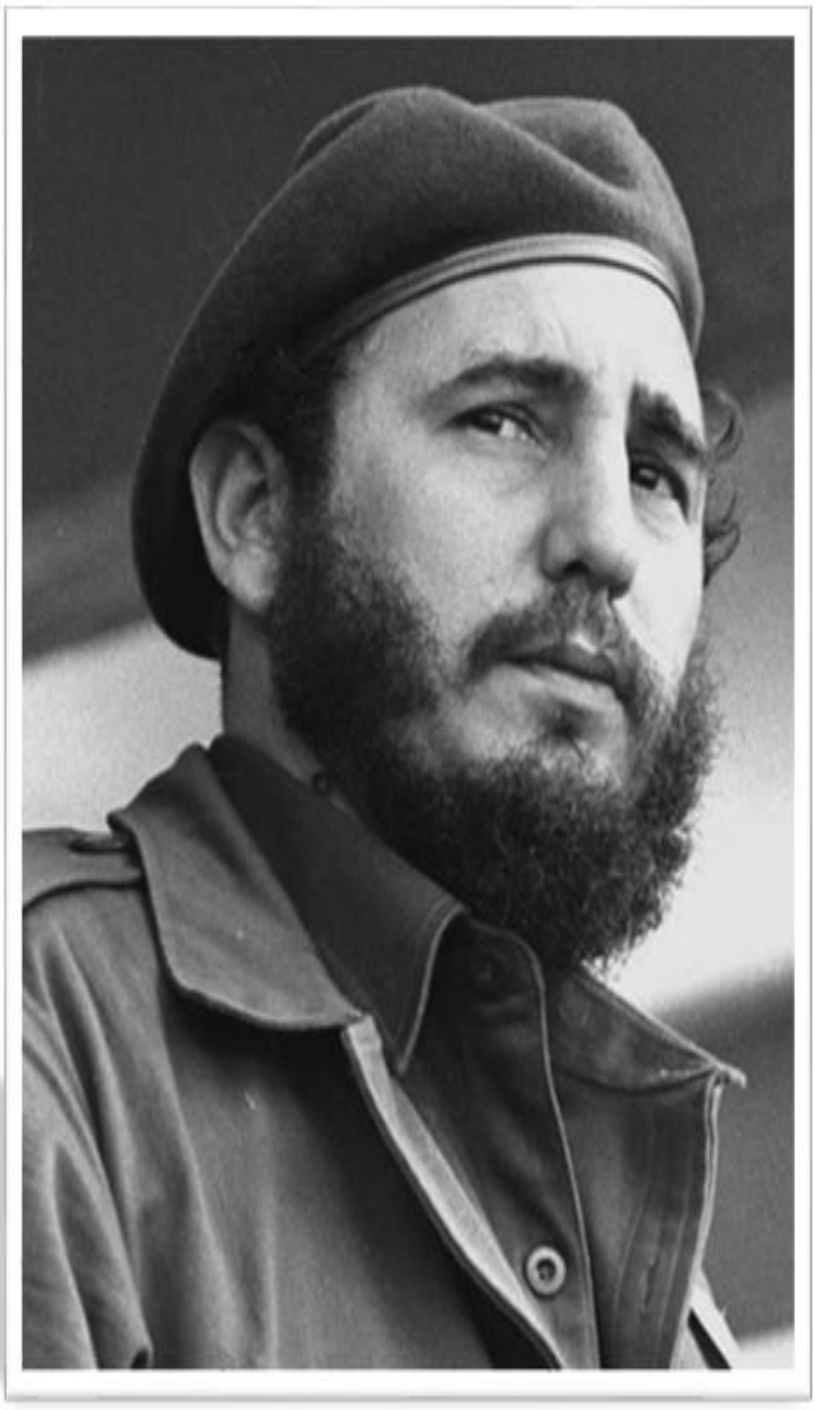
Msc. Radel Calzada Pando

Ing. Isvany Hernández Carmona

La Habana, Junio de 2019

"Nuestra sociedad es la sociedad de la esperanza en el hombre, frente a un mundo que no confía en el hombre".

Fidel Castro Ruz



Agradecimientos

Quisiera agradecer a todas las personas que me ayudaron durante el desarrollo de esta tesis, al igual que a todos los profesores que impartieron los talleres de tesis.

Dedicatoria

A todas las personas que me han apoyado y han hecho que esta labor se realice con éxito en especial a aquellos que me abrieron las puertas y compartieron sus conocimientos.

A Isvany, mi Co-tutor y gran amigo a quien estimo tanto y a quien le debo su apoyo incondicional, por facilitarme los caminos para seguir y no rendirme.

Declaración jurada de autoría

Declaro por este medio que yo Grisel Inés Rodríguez Machado, con carné de identidad 96090422832 soy el autor principal del trabajo titulado “Sistema de Gestión de actividades para la dirección general de economía” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste firmo la presente declaración de autoría en La Habana a los _____ días del mes de _____ del año _____

_____

Firma del autor

Grisel Inés Rodríguez Machado

_____ _____

Firma del tutor Firma del tutor

MSc. Radel Calzada Pando Ing. Isvany Hernández Carmona

Resumen

La Universidad de las Ciencias Informáticas es un centro de estudio que posee como una de sus misiones: producir aplicaciones y servicios informáticos, para servir de apoyo en la informatización del país. Esta institución cuenta con varias áreas, entre estas se encuentra la Dirección General de Economía que posee insuficiencias en su proceso de gestión de actividades, porque se realiza de forma manual el registro de todas las tareas mensuales por plan de trabajo. En aras de darle solución a dicha situación, se desarrolló un sistema de gestión de actividades para la Dirección General de Economía, guiado por las etapas que propone la metodología de desarrollo AUP en la variante para la Universidad de las Ciencias Informáticas. Se definió una estrategia de prueba y con el fin de corroborarla, se creó un entorno adecuado para la utilización del sistema de gestión de actividades que permitió asegurar que el sistema es funcional y cumple con los requisitos identificados con el cliente según las entradas de datos que se probaron y se obtuvieron tiempos de respuesta relativamente bajos en cuanto a la cantidad de peticiones por segundo.

Palabras clave: actividades, gestión de actividades, informe de cumplimiento, planificación, plan de trabajo.

Índice de contenido

Índice

Índice de tablas

Índice de ilustración

Introducción

Un Sistema de Información (SI) es un elemento muy valioso dentro del mundo empresarial, ayuda a conocer la situación de la empresa y a analizar las estrategias que se ponen en marcha para su correcto funcionamiento. En la actualidad, las empresas implementan buenas prácticas para efectuar una correcta coordinación de los diferentes esfuerzos individuales que realizan sus empleados. Un ejemplo, es la implantación de un software de información que permite que las empresas reciban de manera automatizada, información relevante que les ayude a la toma de decisiones, a la vez que mejora la ventaja competitiva de la organización. Una de las tareas fundamentales de la administración empresarial es la planificación, encauzada a diseñar las acciones orientadas a utilizar racionalmente los recursos disponibles. La planificación es un proceso de concertación que, por su carácter dinámico, evoluciona y se adecua a un contexto social, espacial y temporal. Mediante este proceso se obtiene una visión del futuro, donde es posible determinar y lograr los objetivos, a través de la elección de un curso de acción. Una adecuada planificación contribuye al desarrollo profesional y empresarial, reduce riesgos e incrementa el aprovechamiento del tiempo y los recursos (1).

La economía en Cuba esta conducida hacia un desarrollo planificado, con la implantación de una correcta administración empresarial. La necesidad de ubicar la planificación empresarial en el lugar que le corresponde, dentro del contexto de la actualización del modelo económico cubano, constituye una demanda de primer orden, que requiere de la supervisión y la voluntad de funcionarios, empresarios y académicos (2). Las disposiciones de nuevas tecnologías en el país han tenido un papel significativo para alcanzar la expansión y la creación de nuevos tipos de instituciones. La Universidad de las Ciencias Informáticas(UCI) es una institución que con el uso de las tecnologías tiene como uno de sus principales objetivos estrechar la relación universidad-empresa. Funciona igual que otras empresas estableciendo un conjunto de pautas que contribuye al desarrollo de la planificación empresarial.

Dentro de la Universidad una de las áreas vinculada a este proceso es la Dirección General de Economía (DGE), la cual presenta insuficiencias en el proceso de gestión de actividades de sus trabajadores, debido a que esta se realiza de forma manual y esto provoca:

1. Pérdida de tiempo.
2. Documentos duplicados.

3. Alto consumo de papel.
4. Dificultad en la organización de los documentos.
5. Demora en la entrega de la información por el personal.
6. Pérdida de documentos.
7. Poca fiabilidad de la información.
8. Introducción de errores humanos.

Con la implantación de un sistema de gestión de actividades, la entidad mejoraría en varios aspectos como:

1. Control efectivo de las actividades de la organización.
2. Disponibilidad de las actividades para los usuarios en tiempo real.
3. Acceder a la información de forma rápida y oportuna.
4. Disminuir errores, tiempo y el uso de recursos.
5. Permitir la comparación de los resultados alcanzados con los objetivos programados, con fines de evaluación y control.

Por todo lo anteriormente planteado se determina el siguiente **problema de la investigación:** ¿Cómo mejorar el proceso de gestión de actividades en la Dirección General de Economía?

Se define como **objeto de estudio** los procesos de gestión de actividades y el **campo de acción** lo constituye el proceso de gestión de actividades en la Dirección General de Economía de la Universidad de las Ciencias Informáticas.

En aras de darle solución al problema planteado, se define como **objetivo general:** desarrollar un sistema para la gestión de actividades en la Dirección General de Economía de la Universidad de las Ciencias Informáticas.

A partir del objetivo general, se trazan los siguientes **objetivos específicos:**

1. Establecer los fundamentos teóricos relacionados con la gestión de actividades.
2. Fundamentar la selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema de gestión de actividades.

3. Definir los elementos en cuenta para llevar a cabo el análisis y diseño del sistema de gestión de actividades de la Dirección General de Economía.

4. Implementar el sistema para la gestión de las actividades de la Dirección General de Economía.

5. Implementar una estrategia de prueba a fin de validar los resultados en el objetivo anterior.

Para encaminar la investigación en vista a resolver el problema planteado se propone como **idea a defender:** El desarrollo de un sistema para gestionar las actividades de la Dirección General de Economía en la Universidad de las Ciencias Informáticas mejorará este proceso.

Las **tareas de investigación** que se proponen para dar solución a los objetivos son:

1. Sistematización de los referentes teóricos que sustentan la investigación, relacionados con el uso de las herramientas informáticas.

2. Análisis de elementos comunes y necesarios que en cada documento se ajustan a las necesidades de la gestión de la información para el sistema para la gestión de las actividades de la Dirección General de Economía.

3. Análisis y diseño de las funcionalidades del sistema de gestión de actividades.

4. Implementación de las funcionalidades del sistema de gestión de actividades.

5. Validación del sistema de gestión de actividades a través de la estrategia de prueba.

Para el desarrollo de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

- **Histórico-Lógico:** se realiza un estudio del desarrollo de la gestión de actividades de la Dirección General de Economía, para tener una visión global de la problemática planteada.
- **Analítico-Sintético:** se utiliza en el análisis bibliográfico del tema y para realizar una síntesis de los aspectos esenciales relacionados con el desarrollo de la gestión de actividades, además se empleó para analizar las herramientas, metodologías y lenguajes utilizados en el desarrollo de la solución.
- **Modelación:** se utiliza para realizar una abstracción de la realidad mediante los diagramas correspondientes al sistema, para lograr una mejor comprensión de los procesos de la DGE y describir paso a paso como va a funcionar cada proceso según los requerimientos definidos.

- **Inductivo-Deductivo:** se emplea para arribar a razonamientos que puedan ser aplicables al problema a resolver, luego de adquirir una serie de elementos referentes a la gestión de actividades.

Métodos Empíricos:

- **Observación:** Se parte de una serie de observaciones de los problemas relacionados con el proceso referente a la gestión de actividades de la DGE.
- **Entrevista:** realizada al cliente y especialista para determinar sus principales afectaciones con el proceso actual de gestión de actividades, y hacer uso de esta información para el levantamiento de requisitos funcionales. (Ver Anexo1)

El presente documento está estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica del proceso de gestión de actividades: este capítulo contiene la base teórica para entender el problema planteado, se realiza un estudio acerca del estado del arte de los sistemas homólogos y una valoración de la importancia de un sistema de gestión de actividades para la Dirección General de Economía. Se describen las herramientas y tecnologías que permitirán el diseño e implementación de las funcionalidades que darán respuesta al problema planteado, además de la metodología a utilizar para el desarrollo de la solución.

Capítulo 2: Características y diseño del sistema de gestión de actividades: en este capítulo se exponen las características generales de la solución propuesta y se detallan los requisitos funcionales y no funcionales que se tienen en cuenta para su implementación. Se especifican las historias de usuarios y la descripción textual de cada uno de ellos incluyendo los prototipos de interfaz de usuario para lograr un entendimiento claro del sistema a desarrollar, la arquitectura, los patrones de diseño empleados y los artefactos derivados de la metodología de desarrollo seleccionada.

Capítulo 3: Implementación y prueba del sistema de gestión de actividades. Se definen los estándares de codificación a utilizar y se realiza el diagrama de componente y de despliegue de la solución. Además, se realizan pruebas para verificar la calidad del sistema de gestión de actividades, se muestran los resultados y las interfaces principales.

CAPÍTULO 1. Fundamentación teórica del proceso de gestión de actividades

1.1. Introducción

Los sistemas de gestión de actividades son de gran utilidad a la hora de realizar un control efectivo de las actividades en una organización. La capacidad de realizar la gestión de actividades aumenta la disponibilidad de la información en tiempo real. Con el objetivo de lograr una mayor comprensión del alcance de la investigación y esclarecer su objeto de estudio, se realiza un análisis del estado del arte y se establecen los conceptos fundamentales para la comprensión de la investigación. Además, se presenta el entorno de desarrollo a utilizar para dar solución al problema planteado.

1.2. Proceso de gestión de actividades de la Dirección General de Economía.

La DGE, la cual está integrada por varias Direcciones y Departamentos, que realizan de forma independiente las actividades correspondientes a cada una de estas áreas:

- La Dirección de Contabilidad y Finanzas(DCF), la cual tiene subordinado 7 departamentos, donde cada uno de ellos hace tareas específicas ajustándose al manual de normas contables que se emplea en el centro y a nivel nacional. Mensualmente esta dirección crea la nómina de pago de estudiantes y trabajadores, así como la contabilización de todos los procesos del centro que tributan a otras áreas de la Universidad. Otras de las ramas de actividades realizadas en la DCF son las correspondientes a los departamentos de caja, inventario y activos fijos tangibles, que todos tienen tareas propias pero que incluyen a varias áreas del centro.
- La Dirección de Planificación, la misma tiene procesos que generan reportes de informaciones de todas las áreas del centro, pues en su manual de procedimiento incorporar actividades de forma mensual que son aquellas que se orientan por el ministerio.
- La Dirección de Almacenes, permite el control de todos los medios que aparecen en los inventarios de los 13 almacenes que posee el centro, generándose actividades internas y externas al área.
- El Departamento de Información y Estadística, realiza sus actividades rigiéndose por un plan generado por varios organismos a los cuales se tributan informaciones mensuales y estas actividades dependen de un día en específico para la entrega o recepción de información.
- El grupo de Gestión Energético, sus actividades van enfocadas a las labores diarias que se hacen en el centro con los portadores energéticos.

Luego de las explicaciones de cada una de las áreas que conforman la DGE, se puede apreciar que la generación de actividades es muy diversa debido a que cada una de las áreas dependen de entidades externas al centro, así como áreas internas del mismo.

1.2. Conceptos asociados a la investigación

Actividad laboral

Es la intervención del ser humano que opera interactuando entre objeto y medios de trabajo, es decir, la inversión física e intelectual de la trabajadora o el trabajador, que incluye las tareas con su conjunto de operaciones y acciones realizadas, para cumplir con la intención de trabajo, donde existe la interacción dinámica con el objeto que ha de ser transformado y los medios (herramientas, máquinas, equipos, entre otros) que intervienen en dicha transformación (3).

Actividad externa

Son aquellas tareas que están contenidas en los planes de la Universidad que tienen incidencias en los diferentes niveles de la Dirección General de Economía. Son actividades propuestas por otras organizaciones que cumplen con los objetivos de una organización específica y que pertenecen a su plan.

Actividad interna

Son aquellas actividades que están contenidas en los planes de una organización que son propuestas por ella misma. Su función principal es proporcionar respuestas a todos los objetivos trazados por la misma y donde los subordinados estarán directamente dirigiéndola (3).

Plan

Es un modelo sistemático que se desarrolla antes de concretar una cierta acción con la intención de dirigirla. También constituye un instrumento de planificación y gestión que permite llevar a cabo los fines de la organización, mediante una adecuada definición de los objetivos y metas (4).

Plan de trabajo

Es un plan que permite ordenar y sistematizar información relevante para realizar un trabajo, propone una forma de interrelacionar los recursos humanos, financieros, materiales y tecnológicos disponibles (4).

1.3. Análisis de sistemas homólogos

SIPAC

El Sistema para la Planificación de Actividades SIPAC constituye una herramienta para la gestión de las actividades a todos los niveles organizacionales, desarrollado el Centro de Informatización de Entidades (CEIGE), de la UCI en conjunto con el Grupo de Planificación de la Secretaría del Consejo de Ministros. SIPAC cuenta con varios módulos encargados de generar las configuraciones necesarias para el seguimiento de las tareas principales de cada entidad, gestión de los posibles involucrados, nomencladores y niveles de subordinación basados en reglas de la compartimentación de la información. Mediante el sistema se realiza el registro, seguimiento y control de los planes, las puntualizaciones de las actividades, la evaluación de los objetivos y la recuperación de la información, específicamente: Plan de trabajo individual, Plan mensual, Plan anual de actividades y Resumen de cumplimiento (5).

Kronos Workforce

Kronos incluye tanta funcionalidad en su software, que lo hace más complejo de lo que debería ser para una pequeña y mediana empresa. Los usuarios a nivel de administrador del sistema pueden adaptar los flujos de trabajo y las plantillas básicas de *Kronos* para que se ajusten a la forma en que su empresa administra su negocio. El sistema permite la creación personalizada de formularios de evaluación de empleados y puede almacenar dicha documentación de forma segura en línea. Los usuarios a nivel de administrador del sistema pueden configurar procesos y pautas, en la temporada de revisión, un gerente puede agregar o programar una revisión. En el último año, *Kronos* ha agregado la integración de Google *for Work*, que le permite incorporar calendarios, documentos, hojas de cálculo e incluso mapas en su gestión de recursos humanos. Además, la compañía agregó una función llamada *Kronos Workforce Payroll*

Services, que le permite convertir sus datos de nómina en informes en tiempo real para mantenerse al día. La herramienta cuenta con una opción de autoservicio para empleados y administradores, que permite a los usuarios enviar depósitos directos y ver estados de pago, sin necesidad de permisos adicionales (6).

Deputy

Es una solución humana basada en nubes de administración de recursos que cuida negocios de todos los tamaños a través de verticales diversas de la industria y les provee funcionabilidades del empleado de la gerencia y de planificación. *Deputy* posee negocios aplicativos de ayudas para crear horarios que pueden calcular tiempo extra y proveen capacidades para organizar al personal. Si un miembro del equipo cancela su cambio, los gerentes le pueden enviar una petición de cambio a su siguiente miembro de la administración disponible y pueden llenar el espacio del cambio. Además del tiempo, asistencia y la planificación, *Deputy* incorporan otras características que incluyen una plataforma de comunicación que crea una forma para generar anuncios en un espacio solo para los empleados. Soporta integración con varias plataformas de la nómina en el mercado y les ofrece a las funcionabilidades de la gerencia de actuación a los usuarios. Los servicios son ofrecidos en una base anual y mensual de suscripción que incluye soporte por teléfono, el empleado guía y otros recursos en línea (7).

Shiftboard

Shiftboard es una oferta sólida en el campo de planificación de turnos y planificación de empleados. Sin embargo, la vista del calendario de *Shiftboard* es complicada y poco intuitiva. Aun así, *Shiftboard* sigue siendo una de las herramientas de planificación de actividades más fuertes en el mercado hoy en día debido a que Incluye acceso a las aplicaciones móviles basadas en la web. Los planes aumentan a medida que aumenta su personal o aumentan las necesidades de funciones, brinda acceso a usuarios ilimitados, marca personalizada e integración de nómina, entre otras características. En comparación con otras herramientas, tiene un precio asequible en los niveles más bajos, pero más caro a medida que sus necesidades se vuelven más complejas. Los administradores pueden ver quién ha ingresado, quién no está disponible y quién está programado, pueden crear tarjetas de tiempo para exigir a los usuarios que ingresen y salgan. Los datos de estas tarjetas de tiempo se pueden utilizar para enviar informes avanzados a sus herramientas integradas de nómina y otras herramientas de recursos humanos para supervisar los pagos y el rendimiento de los empleados en lo que respecta a la asistencia (8).

Conclusiones del análisis de las soluciones existentes.

La siguiente tabla muestra un resumen de algunas de las características antes descritas.

Tabla 1: Resumen comparativo de soluciones existentes (Elaboración propia).

Sistemas informáticos	Ventajas	Desventajas
DEPUTY	Permite crear horarios que pueden calcular tiempo extra y proveen capacidades para organizar al personal.	Su gestión de actividades mayormente se enfoca en los recursos humanos. Software privativo
SHIFTBOARD	Incluye acceso a las aplicaciones móviles basadas en la web.	No es multiplataforma Software privativo
KRONOS	Permite incorporar calendarios, documentos y hojas de cálculo. Permite convertir sus datos de nómina en informes en tiempo real.	Se necesita capacitación para utilizarlo. Software privativo
SIPAC	Permite generar: Plan de trabajo individual, Plan mensual, Plan anual de actividades y Resumen de cumplimiento	Posee una estructura engorrosa. Presenta problemas técnicos y de soporte.

Pese a que los sistemas analizados se enfocan en gran medida en la gestión de actividades, ninguno cumple con todos los requisitos y los estándares de la gestión de actividades establecidos en la DGE. El proceso de gestión de actividades podría sufrir cambio en un futuro, por lo que el cliente requiere un software que pueda ser modificado y además ser libre costo para cumplir con las políticas abogadas en la UCI y en el país. *KRONOS, DEPUTY y SHIFTBOARD* son *softwares* privativos, lo que impide tener acceso a su código fuente y por tanto no se pueden modificar. Actualmente los servicios de SIPAC no cumplen con todas las expectativas de sus clientes debido a problemas técnicos y de soporte. La UCI se encuentra desarrollando una nueva versión de SIPAC para corregir los problemas ocasionados a los clientes, imposibilitando su utilización inmediata. Los resultados de la investigación manifiestan que aun cuando las características y funciones de estos sistemas son semejantes a lo que se desea lograr, no se pueden usar como solución al

problema existente. Independientemente de lo antes mencionado servirán de base para la identificación de una serie de funcionalidades a incluir en la solución.

1.4. Ambiente de desarrollo a emplear en la propuesta de solución

1.4.1. Metodologías de desarrollo de software

La metodología de desarrollo de software hace referencia al conjunto de procedimientos racionales utilizados para alcanzar el objetivo o la gama de objetivos que rige una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. Con frecuencia puede definirse la metodología como el estudio o elección de un método pertinente o adecuadamente aplicable a determinado objeto. Metodología de Desarrollo de Software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir: Quién debe hacer Qué Cuándo y Cómo debe hacerlo. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende actividades a seguir para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (9).

El Proceso Unificado Ágil fue creado por *Scott Ambler*, es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. Algunas técnicas usadas por AUP incluyen el desarrollo orientado a pruebas, modelado y gestión de cambios ágiles, y refactorización de base de datos para mejorar la productividad (10). La Universidad de las Ciencias Informáticas define como política a seguir en los proyectos productivos la utilización de la metodología AUP (Proceso Unificado Ágil) adaptada al ciclo de vida definido para la actividad productiva de la Universidad, por lo que se decide su utilización para guiar el proceso de desarrollo de la solución.

AUP cuenta con cuatro fases que transcurren de manera consecutiva: Inicio, Elaboración, Construcción y Transición. De las 4 fases que propone AUP se decide para el ciclo de vida de los proyectos de la UCI

mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, llamada Ejecución y se agrega la fase de Cierre (10).

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Escenarios para la disciplina de requisitos

Existen tres formas de encapsular los requisitos: Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP), agrupados en cuatro escenarios condicionados por el Modelado de negocio. A la presente investigación se le ajusta el escenario 4 en específico porque se cuenta con un equipo de desarrollo pequeño y el cliente estará siempre acompañando al equipo de desarrollo con la idea del negocio bien definida para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. No se modelará el negocio, sólo se puede modelar el sistema a partir de las HU. Se recomienda en proyectos no muy extensos, pues una HU no debe poseer demasiada información.

1.4.2. Modelado de software

UML (*Unified Modeling Language*), es un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo, que son la representación simplificada de la realidad; el modelo UML describe lo que debería realizar un sistema, pero no dice cómo implementar dicho sistema (20).

Herramienta para el modelado

Visual Paradigm en su versión 8.0, se selecciona como herramienta *CASE* de modelado profesional, que utiliza *UML* para la completa representación de las etapas por las que transita un producto de software. Dispone de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad. Ayuda el trabajo en grupo, proporcionando herramientas de compartición de trabajo y generara diversos informes a partir de la información introducida en la herramienta. Facilita la reutilización y se dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos (21).

1.4.3. Lenguaje de programación

PHP versión 7.2

PHP (*Hypertext Preprocessor*) es un lenguaje de código abierto muy popular. Es interpretado, se ejecuta en el servidor y se puede incrustar en HTML. Se utiliza principalmente en desarrollo web, para el que es especialmente adecuado. Una de sus características más potentes es el soporte para gran cantidad de bases de datos como MySQL, Oracle e InterBase. PHP también ofrece la posibilidad de integrar distintas bibliotecas externas, lo que le da mucha versatilidad y permite facilitar la tarea al desarrollador. Es que puede ser desplegado en la mayoría de servidores web y en casi todos los sistemas operativos (11).

JavaScript

Javascript es un lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario y la realización de automatismos sencillos. En ese contexto se puede decir que nació como un "lenguaje de scripting" del lado del cliente, sin embargo, hoy Javascript es mucho más. Las necesidades de las aplicaciones web modernas y el HTML5 ha provocado que el uso de Javascript que se encuentra hoy haya llegado a unos niveles de complejidad y prestaciones tan grandes como otros lenguajes de primer nivel (13).

Librería *jQuery*

La librería más conocida de Javascript se llama jQuery y se ha convertido en un complemento en la mayoría de las webs que se usan actualmente, por su potencia. Con jQuery se puede escribir código Javascript que es capaz de ejecutarse sin errores en cualquier navegador, incluso los antiguos y te implementa muchas

funcionalidades que puedes requerir repetidamente en cualquier sitio web. jQuery te permite además programar nuevas funcionalidades por medio de plugins para hacer actividades tan variadas como la validación de formularios, sistemas de plantillas, pases de diapositivas e interfaces de usuario (16).

Lenguaje marcado de hipertexto (HTML 5)

HTML, que significa Lenguaje de Marcado para Hipertextos (HyperText Markup Language) es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad. HTML le da "valor añadido" a un texto estándar en español. Hypertext se refiere a enlaces que conectan una página web con otra, ya sea dentro de una página web o entre diferentes sitios web, por tanto, los vínculos son un aspecto fundamental (12).

Hojas de Estilo en Cascada (CSS 3)

Hojas de Estilo en Cascada (Cascading Style Sheets) es el lenguaje utilizado para describir la presentación de documentos HTML o XML, esto incluye varios lenguajes basados en XML como son XHTML o SVG. La versión 3 de CSS (CSS3) proporciona un estándar modularizar para que cuando los cambios ocurran en un sitio específico, solo un módulo particular dentro de CSS3 debe ser actualizado, en lugar de todo el estándar. Esta flexibilidad permite una actualización más oportuna del estándar. La nueva construcción modular de CSS3 incluye los estándares anteriores de CSS, así como nuevos módulos. Algunos de los módulos clave de CSS3 incluyen: selectores, modelo de caja, fondos y fronteras, efectos de texto, fuentes e interfaz de usuario. CSS3 también presenta algunas capacidades nuevas para los desarrolladores, que incluyen: transformaciones 2D y 3D, transiciones, animaciones, transparencia alpha y superposiciones (14).

Bootstrap versión 4.0

Bootstrap es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como diseño adaptativo (15). Bootstrap en su versión 4 mantiene su escala responsiva a una alta resolución de pantalla y brinda nuevas opciones más específicas para facilitar la aplicación de los estilos a los elementos en el HTML.

1.4.4. Marco de trabajo

Se seleccionó Symfony como framework de desarrollo web del lado del servidor, pues, está diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador (MVC). Su objetivo es, separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos (17).

1.4.5. Sistema gestor de base datos

PostgreSQL en su versión 11.0, es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl y python). Debido a la licencia liberal, PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo gratis de forma gratuita para cualquier fin, ya sea privada, comercial o académico (18).

1.4.6. Entorno integrado de desarrollo

Netbeans es un entorno de desarrollo integrado (IDE) basado en Java. El término también se refiere al marco de la plataforma de aplicaciones subyacente del IDE. El IDE está diseñado para limitar los errores de codificación y facilitar la corrección de errores con herramientas como Netbeans FindBugs para localizar y corregir problemas comunes de codificación Java y depurador para administrar código complejo con relojes de campo, puntos de interrupción y monitoreo de ejecución. Aunque el Netbeans IDE está diseñado específicamente para desarrolladores de Java, también es compatible con C / C ++, PHP, Groovy y HTML5 además de Java, JavaScript y JavaFX. Netbeans IDE puede ejecutarse en cualquier sistema operativo que admita una JVM compatible, incluidos Linux y Windows. La plataforma Netbeans subyacente admite la creación de nuevas aplicaciones y un mayor desarrollo de aplicaciones existentes utilizando componentes

de software modulares. Como una aplicación que se ejecuta en la plataforma Netbeans, el IDE de Netbeans es extensible y puede ampliarse para admitir nuevos lenguajes (19).

1.4.7. Servidor web

Apache HTTP Server

Apache es una aplicación gratuita que convierte un ordenador en un servidor web. Es de código abierto, flexible, rápido y eficiente. *Apache* permite negociar protocolos *HTTP* entre una máquina que haría de servidor web y los otros ordenadores que deseen ver un determinado sitio web. Es multiplataforma y gracias a su popularidad es fácil encontrar documentación para profundizar en las funcionalidades que ofrece. Se caracteriza por su gran escalabilidad, seguridad y rendimiento. Tiene soporte para varios lenguajes como *Perl*, *Python* y *PHP*, lo que permite desarrollar aplicaciones web de gran calidad (38).

1.5. Conclusiones del capítulo

Se realizó un análisis de los principales conceptos relacionados con los sistemas de gestión de actividades, en tal sentido se puede arribar a las siguientes conclusiones:

1. La definición de los principales conceptos asociados al dominio de la presente investigación y las relaciones entre estos, permite alcanzar una mayor comprensión de la propuesta de solución.

2. El análisis de los sistemas homólogos, permite identificar las tendencias en cuanto al desarrollo de sistemas informáticos para la gestión de actividades, e identificar algunas funcionalidades de utilidad para la propuesta de solución.

3. La definición de las herramientas, tecnologías y lenguajes, permite sentar las bases para el desarrollo de la propuesta de solución.

CAPÍTULO 2. Características y diseño del sistema de gestión de actividades

2.1 Introducción

En todo desarrollo de sistemas de software es necesario seguir algunas especificaciones, que permitan a los desarrolladores tener una disciplina que haga que todas las etapas del desarrollo del sistema, sean coherentes y formales. El desarrollo del sistema que se propone, al ser una herramienta que pretende tener aplicación dentro del contexto de un problema real, tiene que seguir un proceso de análisis y diseño que proporcione las bases bajo las cuales se va a desarrollar la aplicación. El objetivo de este capítulo es presentar los resultados que se obtuvieron una vez cumplidas las fases de Análisis y Diseño que propone la metodología AUP-UCI. Se detallan las características del sistema, se especifican los requisitos funcionales y no funcionales del mismo. Se realiza la descripción de la propuesta de solución, para proporcionar un mayor entendimiento.

2.2 Características de la propuesta de solución

Para dar solución a la problemática planteada en la presente investigación se propone la implementación de SADE (Sistema de gestión de actividades para la Dirección Economía en la Universidad de las Ciencias Informáticas), un sistema que servirá de apoyo a al proceso de gestión de actividad laboral de los trabajadores de la DGE. Desarrollado mediante el marco de trabajo Symfony, que permite la especificación de las tareas de cada área, departamento y trabajador de la DGE, además de las tareas nuevas incorporadas como parte de la puntualización y las suspendidas o canceladas. El mismo debe permitir un manejo adecuado de la información referente a la gestión de las actividades tanto individual como colectiva. Cada usuario a partir del rol correspondiente podrá en el sistema registrar todas sus tareas para confeccionar un plan de trabajo, así como informes de cumplimiento de los mismo. Se propone un modelo para el resumen de cumplimiento del Plan mensual de actividades donde se incluyen las tareas externas, las propias y las nuevas incorporadas como parte de la puntualización. Lo que permitirá realizar el análisis de lo que se planificó, lo que se cumplió y no se cumplió, contando también las tareas suspendidas o canceladas, las modificadas y las incorporadas.

2.3 Modelo conceptual

Un modelo de conceptual es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, presentado como uno o más diagramas de clases. Se pueden utilizar para

capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Los objetos del dominio o clases pueden obtenerse a partir de una especificación de requisitos (22).

Componen el modelo conceptual definido para la presente investigación:

Actividad: conjunto de acciones que el trabajador lleva a cabo para cumplir con un plan de trabajo

Informe de cumplimiento: es el documento que muestra los resultados de las actividades realizadas por un trabajador.

Plan de trabajo: es un plan donde se puntualiza la forma y los medios necesarios para llevar a cabo un conjunto de actividades.

Trabajador: persona encargada de realizar un conjunto de actividades y así cumplir con su plan de trabajo**.**

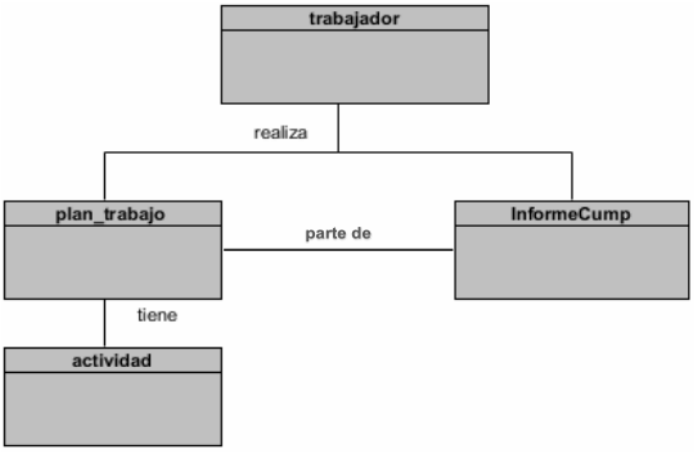


Ilustración 1: Modelo de dominio (Elaboración propia).

2.4 Especificación de requisitos

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas. Estos requisitos dependen del tipo de software

que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requisitos (23).

Tabla 2: Requisitos Funcionales (Elaboración propia).

Requisito funcional	Prioridad	Requisito funcional	Prioridad
RF1: Autenticar usuario	Alta	**RF12:** Insertar plan de trabajo	Alta
RF2: Registrar usuario	Media	**RF13:** Modificar plan de trabajo	Alta
RF3: Modificar usuario	Media	**RF14:** Eliminar plan de trabajo	Alta
RF4: Listar usuario	Media	**RF15:** Mostrar plan de trabajo	Alta
RF5: Eliminar usuario	Media	**RF16:** Listar plan de trabajo	Media
RF6: Mostrar usuario	Media	**RF17:** Insertar cumplimiento de actividad	Alta
RF7: Insertar actividad	Alta	**RF18:** Modificar cumplimiento de actividad	Alta
RF8: Modificar actividad	Alta	**RF19:** Eliminar cumplimiento de actividad	Alta
RF9: Eliminar actividad	Alta	**RF20:** Mostrar cumplimiento de actividad	Alta
RF10: Mostrar actividad	Alta	**RF21:** Listar cumplimiento de actividad	Media
RF11: Listar actividad	Alta	**RF22:** Exportar a PDF	Alta

Requisitos no funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (23).

Requerimientos de software

RNF1: Gestor de base de datos PostgreSQL en la versión 11.0 o superior.

Requerimientos de hardware

RNF2: El servidor de aplicaciones web y de base de datos deben poseer como mínimo un CPU Dual Core a 1.30 GHz.

RNF3: El servidor de aplicaciones web debe poseer una capacidad mínima de 10 GB pues la aplicación no debe crecer en el tiempo.

RNF4: El servidor de base de datos debe poseer una capacidad mínima de 5 GB.

RNF5: Los servidores web y de base de datos deben poseer como mínimo 2 GB de memoria RAM.

Requerimientos de Seguridad

Confidencialidad

RNF6: El sistema debe registrar cualquier fallo de su funcionamiento con el objetivo de permitir su solución por los administradores.

RNF7: En caso de que ocurran fallas, los errores deben mostrarse sin detalles de información que puedan comprometer la integridad y seguridad del mismo. Sólo se mostrarán detalles ampliados del error a usuarios con privilegios de administración.

RNF8: En el sistema se deben establecer mecanismos que garanticen la confiabilidad de la información ante posibles accesos no autorizados tales como: la protección *Cross Site Request Forgery* (CSRF), la autenticación mediante usuario y contraseña y garantizar los niveles de acceso a través de roles.

RNF9: El acceso al sistema será controlado a través de la autenticación por nombres de usuario y contraseñas internos del sistema.

Integridad

RNF10: Se garantizará la integridad de la información mediante mecanismos de control de acceso, utilizando, usuario, contraseña y niveles de accesos mediante el uso de roles para cada usuario, de manera que cada uno pueda tener disponible solamente las opciones que se encuentran en correspondencia con su rol y las funciones que el mismo realiza.

Requerimientos de usabilidad

RNF11: El sistema debe presentar una estructura de navegación semejante en todo el sitio.

RNF12: El sistema debe proveer una iconografía descriptiva y representativa de las funcionalidades del sistema.

RNF13: El sistema debe presentar una letra más grande en los encabezados de sección que posibilite la orientación en el sitio, tanto para los usuarios con conocimientos avanzados en informática como los usuarios más inexpertos.

RNF14: El sistema debe tener buena visibilidad en los navegadores web a partir de las siguientes versiones: Internet Explorer 9, Firefox 4, Chrome 16 y Opera 9.

Requerimientos legales

RNF15: Utilizar sistemas de código abierto para el desarrollo de la aplicación.

2.5 Historias de usuario

Entre los artefactos que define la metodología seleccionada se encuentran las historias de usuario que son utilizadas para especificar las funcionalidades que brindará el sistema. Cada historia de usuario es una representación de un requerimiento de software escrito en una o dos frases. Representan una forma rápida de administrar los requerimientos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos (24).

Tabla 3: Historia de usuario insertar actividad (Elaboración propia).

Historia de Usuario	
Número: **HU_7**	**Usuario:** Usuario
Nombre de historia: **Insertar actividad**	
Prioridad en negocio: **Alta**	**Nivel de complejidad:** Alto
Programador Responsable: **Grisel Inés Rodríguez Machado**	
Descripción: **El sistema debe permitir insertar una actividad verificando que la actividad cumpla con los datos establecidos, siempre especificando los datos requeridos por la aplicación:** • **Fecha** • **Hora** • **Nombre de la actividad**	

- **Lugar**
- **Responsable de la actividad**

Observaciones: **Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación debe mostrar un mensaje correspondiente a cada error.**

Prototipo de interfaz:

2.6 Estilo arquitectónico

La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar el objetivo del sistema por tanto brinda una visión global del sistema. Esto permite entenderlo, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar. Entre los mencionados componentes se encuentran los estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, los cuales se encuentran estrechamente relacionados, conformando la base de la arquitectura de un software (25).

Patrón arquitectónico

Los patrones de software permiten que se reutilice tanto el diseño como la arquitectura, adoptando estructuras estáticas y dinámicas de soluciones exitosas en la solución de nuevos problemas. Los patrones arquitectónicos representan el nivel más alto dentro del sistema de patrones y expresan el esquema de la

estructura fundamental de la organización para sistemas de software. Además, capturan existencia, experiencia comprobada en el desarrollo del software y ayudan a promover buenas prácticas (26). A partir de lo planteado se asumirá el estilo arquitectónico MVC es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. MVC sugiere la separación del software en 3 estratos: Modelo, Vista y Controlador (27).

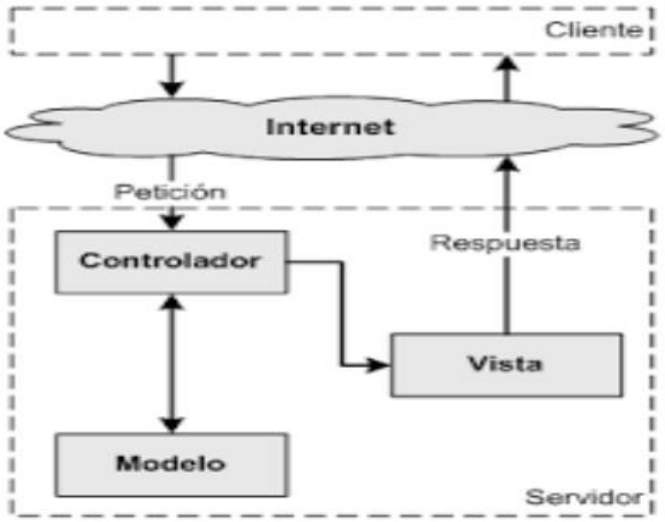


Ilustración 2: Patrón MVC en Symfony3. (28)

Vista: compone la información que se envía al cliente, dentro de ella se ubica todo lo referente a la visualización de la información en las páginas de la aplicación. La lógica de las vistas será manejada por el motor de plantillas Twig, la biblioteca jQuery, en conjunto con los archivos CSS, JavaScript y HTML, que se encargarán de estructurar y aplicar estilos a las interfaces creadas. Dichos archivos se encuentran ubicados en el directorio /Resources/views dentro de cada bundle de la aplicación.

Controlador: actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno. Es la encargada de responder a las acciones del usuario e invocar cambios en el modelo o generar la vista apropiada, dependiendo de las peticiones del usuario. En Symfony 3.4 como marco de trabajo para el desarrollo de la

solución, queda evidenciada esta capa a través del Controlador Frontal, mediante los archivos app.php para entornos de producción y app_dev.php para entornos de desarrollo. Por último, las clases Controller, que controlarán el flujo de información que se recibe y se envía hacia la vista.

Modelo: En la estructura del sistema para gestionar actividades esta capa está dividida en 2, la Capa de Negocio y la Capa de Acceso a datos.

- Capa de Negocio: En la Capa de Negocio se localizan las entidades y los repositorios. Las entidades son la representación de las tablas de la base de datos previamente mapeadas por el ORM Doctrine. Para abstraer la lógica de negocio de los controladores se crean repositorios de entidades personalizados que permiten llevar el manejo de la base de datos a una clase aislada del controlador.

- Capa de acceso a datos: La Capa de Acceso a datos es la encargada de establecer la conexión con la base de datos. Dentro de esta capa se encuentra ubicado el ORM Doctrine, que posibilita la separación en la aplicación del gestor de base de datos mediante su lenguaje propio de consultas DQL.

2.7 Patrones de diseño

El patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. Su aplicación no tiene efectos en la estructura fundamental del sistema, pero sí sobre la de un subsistema, debido a que especifica en mayor nivel de detalle, sin llegar a la implementación, el comportamiento de los componentes del subsistema (29).

Patrones *GRASP*

GRASP (General Responsibility Assignment Software Patterns), son patrones generales de asignación de responsabilidades en software ya que la palabra 'grasp' significa comprender, entender o alcanzar. GRASP son una serie de buenas prácticas enfocadas a la calidad del software, calidad "estructural", ya que no se ha centrado en pruebas sino en la estructura y las responsabilidades de las clases que componen el software que se desarrolla. Hay que considerarlo como una serie de consejos, que hay que seguir si se quiere hacerlo bien: Alta cohesión y bajo acoplamiento, Controlador, Creador, Experto en información, Fabricación pura, Indirección, Polimorfismo, Variaciones protegidas (26).

Experto: Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión del ORM Doctrine para mapear la Base de Datos. Symfony utiliza este ORM para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan. El patrón Experto se encarga de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (29).

Algunas ventajas del patrón Experto son (30):

- Se conserva el encapsulamiento, pues los objetos se valen de su propia información para hacerlo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases “sencillas” y más cohesivas que son más fáciles de comprender y de mantener.

Este patrón se evidencia en el desarrollo del sistema de gestión de actividades siendo las entidades las expertas en la información. En la siguiente ilustración se muestra un ejemplo donde se especifica en la entidad Actividad:

```
/**
 * Actividad
 *
 * @ORM\Table(name="actividad")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\Actividad
 */
class Actividad {

    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="hora", type="time")
     */
    private $hora;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="fecha", type="datetime")
     */
```

Ilustración 3: Uso del patrón Experto en la clase Actividad (Elaboración propia).

Creador: "El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento, constituyendo su principal beneficio, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización" (30).

Este patrón se evidencia en el desarrollo del sistema para la gestión de actividades en las clases controladoras, que son las responsables de la construcción de los formularios que se visualizarán en las vistas de la aplicación.

```
public function newAction(Request $request) {
    $actividad = new Actividad();
    $form = $this->createForm('AppBundle\Form\ActividadType', $actividad);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($actividad);
        $em->flush();

        return $this->redirectToRoute('actividad_show', array('id' => $actividad->getId()));
    }

    return $this->render('actividad/new.html.twig', array(
                'url' => "actividad",
                'actividad' => $actividad,
                'form' => $form->createView(),
    ));
}
```

Ilustración 4: Uso del patrón Creador en la clase ActividadController.php (Elaboración propia).

Alta Cohesión: “En la perspectiva del diseño orientado a objetos, la cohesión (o más exactamente, la cohesión funcional) es una medida de cuanto están relacionadas y enfocadas las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme” (30).

Algunas ventajas del patrón Alta cohesión son según:

- Mejora la claridad y la facilidad con que se entiende el diseño.
- Se simplifican el mantenimiento y las mejoras en funcionalidad.
- A menudo se genera un bajo acoplamiento.
- La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. En el sistema para la gestión de actividades se evidencia mediante la clase Actions.

Bajo acoplamiento: las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja. El acoplamiento "es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras; "muchas otras" depende del contexto. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades." (30). En la implementación del sistema para la gestión de actividades este patrón se evidencia en las entidades que son las clases más reutilizadas y son totalmente independientes, pues no están asociadas ni a la vista ni al controlador.

Controlador:

El patrón Controlador se encarga de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones (30):

Todas las peticiones web son manipuladas por un solo controlador frontal que es el punto de entrada único de toda la aplicación en un entorno determinado. Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad. Este patrón se evidencia a través de las clases controladoras que se localizan en el directorio, *Controller* perteneciente a cada *Bundle* del sistema gestión de actividades y en el uso del controlador frontal ubicado en directorio web de cada aplicación concebida en Symfony 3.4.

```
class ActividadController extends Controller {

    public function indexAction() {
        $em = $this->getDoctrine()->getManager();

        $actividads = $em->getRepository('AppBundle:Actividad')->findAll();

        return $this->render('actividad/index.html.twig', array(
                    'url' => "actividad",
                    'actividads' => $actividads,
        ));
    }
```

Ilustración 5: Uso del patrón Controlador en la clase ActividadController.php (Elaboración propia).

Patrones *GoF* (*Gang of Four*)

Describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. El grupo GoF, se dedicó a analizar los problemas recurrentes en el desarrollo de software y realizaron una clasificación y agrupación a partir de dos criterios, su propósito y alcance, las categorías definidas son: creacionales, estructurales y de comportamiento (29).

Patrones creacionales: Se ocupan del proceso de creación de clases y objetos, son los encargados de "abstraer el proceso de instanciación o creación de objetos, ayudan a que el sistema sea independiente de cómo sus objetos son creados, integrados y representados". Los patrones que hacen parte de esta categoría son cinco: *Factory Method, Abstract Factory, Builder, Prototype y Singleton* (31).

Patrones estructurales: Tratan de la composición de clases y objetos, "se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes". Los patrones en esta categoría se encargan de lograr que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Los patrones que hacen parte de esta categoría son siete: *Adapter, Bridge, Composite, Decorator*, *Facade, Flyweight* y *Proxy* (31).

Patrones de comportamiento: Caracterizan las formas en que las clases o los objetos interactúan y distribuyen la responsabilidad. "Son los encargados de las opciones de comportamiento de la aplicación,

permitiendo que el comportamiento varíe en tiempo de ejecución, sin estos patrones cada comportamiento tendría que diseñarse e implementarse por separado" Los patrones que se agrupan en esta categoría son once: Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor (31).

En la implementación del sistema para la gestión de actividades el patrón Decorador se evidencia en esta potencialidad que posee Twig relacionada con la herencia entre plantillas, es decir, en el uso de una plantilla global que contendrá los elementos comunes del sitio para las vistas que decorarán las demás páginas de la aplicación.

```
{% extends 'base.html.twig' %}
{% block title %}SADE | Iniciar Sesión{% endblock %}

{% block body %}

{% endblock %}
```

Ilustración 6: Empleo del patrón Decorador en la plantilla index.html.twig (Elaboración propia).

Patrón Front controller (Controlador frontal): Posee una estructura bien organizada de controladores, que comienza desde el “index.php” del ambiente y termina en los “Actions”. Cada clase de esta capa tiene su responsabilidad y es única hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, y otros que se encargan de identificar mediante algunos datos las clases que deben realizar determinadas tareas.

2.8 Modelo de datos

Un modelo de base de datos muestra la estructura lógica de la base, incluidas las relaciones y limitaciones que determinan cómo se almacenan los datos y cómo se accede a ellos. Los modelos de bases de datos individuales se diseñan en base a las reglas y los conceptos de cualquier modelo de datos más amplio que los diseñadores adopten (32). Este modelo está compuesto por las entidades (tablas) de la base de datos de PostgreSQL que garantiza la persistencia y manejo de los datos desde las clases controladoras del sistema, que fueron representadas en los epígrafes anteriores. El modelo de datos generado, como se

observa en la Ilustración 7, está compuesto por (5) relaciones y (4) tablas, las cuales se explican a continuación, para un mejor entendimiento del modelo:

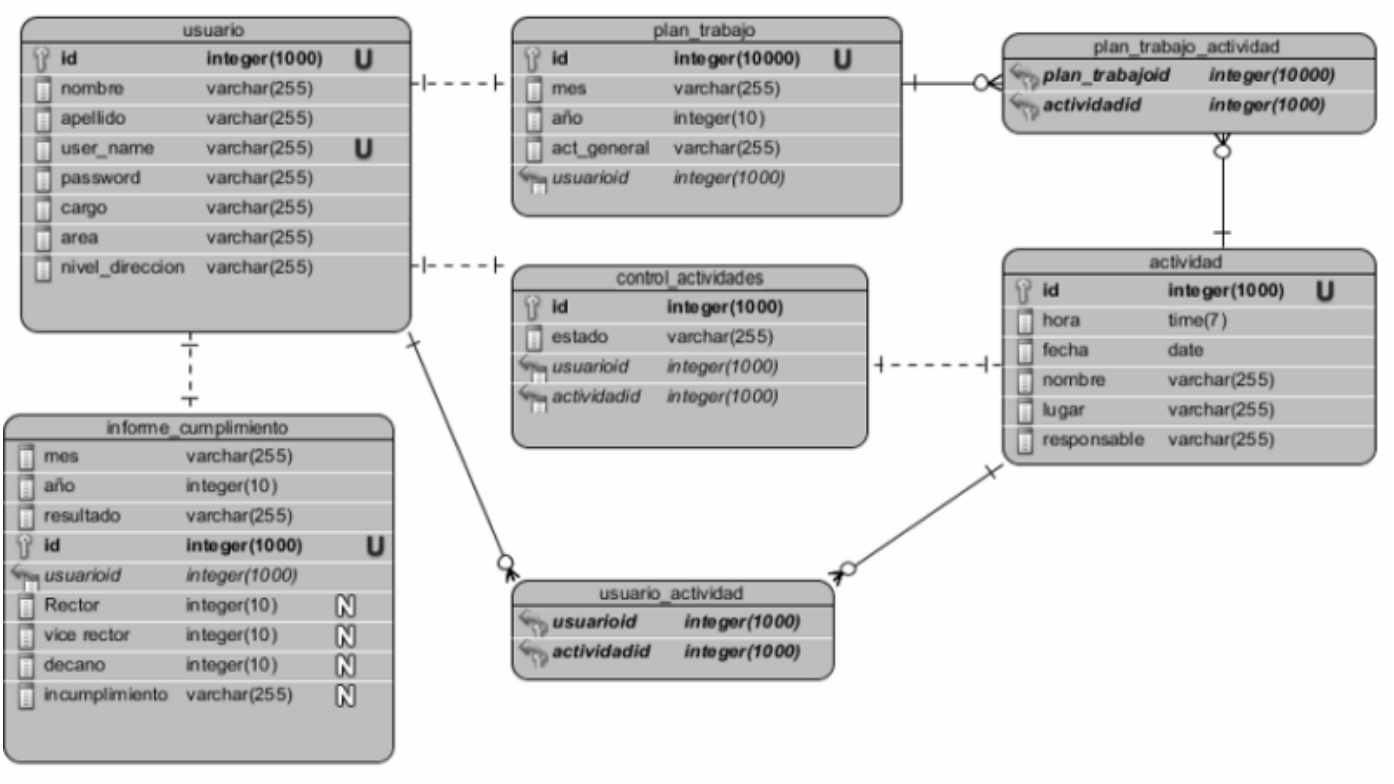


Ilustración 7: Modelo de datos (Elaboración propia).

En la ilustración anterior se muestra la representación de modelo de datos donde se establecen todas las relaciones entre los estudiantes los profesores y las evaluaciones.

2.9 Modelo de despliegue

Un diagrama de despliegue visualiza los elementos de hardware y software utilizados en la implementación del sistema y además las relaciones entre ellos. Es utilizado para modelar la topología de procesadores y dispositivos empleados en el software (29). La ilustración representa la vista de despliegue del sistema.

Ilustración 8: Diagrama de despliegue (Elaboración propia).

<<HTTPS>>: Protocolo para la comunicación a través del puerto 443 la conexión segura entre la PC_Cliente y el servidor de aplicaciones. La conexión es por cable vía modem, LAN o red inalámbrica con una velocidad de más de 64 Kbps.

<<TCP>>: Este protocolo establece la conexión entre el servidor de aplicaciones y el servidor de base de datos. Para el servidor de base de datos de PostgreSQL se define el puerto 5432.

PC_Cliente: Computadora desde donde se realizan las peticiones de información y hacia la que regresan las respuestas de esas peticiones.

Servidor web: Es el encargado de brindar la interfaz de la plataforma para que los usuarios puedan hacer uso de esta, almacena todo el código fuente del sistema y se comunica por medio de los protocolos TCP con el servidor de bases de datos.

SGBD PostgreSQL: "Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia y con su código fuente disponible libremente" (18).

2.10 Conclusiones del capítulo

Luego de realizado el análisis de las funcionalidades con sus respectivas historias de usuario y haber logrado diseñar una propuesta de solución que generase los diferentes artefactos que dispone la metodología AUP, se puede concluir lo siguiente:

1. El análisis de las características del sistema y la modelación del dominio, permiten identificar los

principales requisitos funcionales y no funcionales del SADE, los cuales fueron categorizados por historias de usuarios.

2. La identificación de los patrones de diseño y el estilo arquitectónico del sistema, evidencia que la

solución propuesta cuenta con un alto grado de resistencia ante posibles modificaciones.

3. El diseño de los diagramas, facilita la visión en cuanto a composición física y lógica del sistema.

4. La generación de todos los artefactos requeridos por el modelo de desarrollo, documentan la solución propuesta, lo cual facilita su posterior mantenimiento.

CAPÍTULO 3. Implementación y pruebas del sistema de gestión de actividades

3.1 Introducción

En todo desarrollo de sistemas de software es de suma importancia que antes de empezar a codificar un programa, se tenga una completa y plena comprensión de los requisitos del software. Luego de realizado el análisis y diseño del sistema, es el momento de pasar a la etapa de implementación. En esta fase, se procede a programar los diseños especificados, los cuales son implementados en términos de componentes, ficheros de código fuente y ejecutables.

La verificación y validación es el nombre que se da a los procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes. Inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño y las inspecciones del código hasta la prueba del producto (29). Este capítulo tiene como objetivo, documentar los resultados de las fases de implementación del sistema y de la estrategia de pruebas desarrollada.

3.2 Diagrama de componentes

El diagrama de componentes describe la descomposición física del sistema en componentes, a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos. Los componentes identifican objetos físicos que hay en tiempos de ejecución, de compilación o de desarrollo y tienen identidad propia con una interfaz bien definida. Los componentes incluyen código en cualquiera de sus formatos, DLL, Active X, bases de datos. Cada componente incorpora la implementación de ciertas clases del diseño del sistema (23). Para una mayor especificidad del diagrama mostrado a continuación se explicarán los componentes:

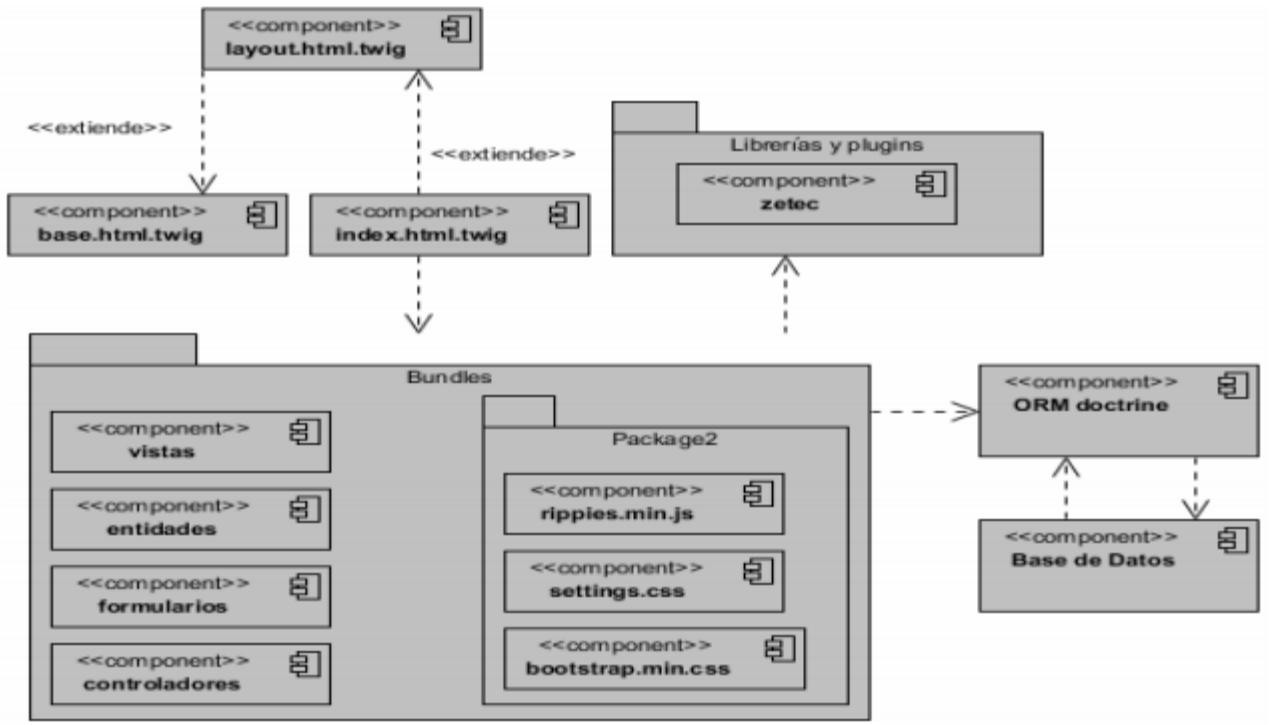


Ilustración 9: Diagrama de componentes (Elaboración propia).

base.html. twig: plantilla en la que se encuentra el diseño principal de la aplicación.

layout.html. twig: plantilla que define la estructura para cada sección del sitio. La misma hereda de base.html. twig.

index.html. twig: página cliente mediante la que se realizan las peticiones al servidor, la misma hereda de layout.html. twig.

Bundle: paquete de funcionalidades en el que se agrupa todo el código de la aplicación estructurado siguiendo el patrón arquitectónico MVC.

Controladores: páginas servidoras de la aplicación mediante las cuales se realizan las funcionalidades. Son los intermediarios entre la vista y el modelo, quienes controlan las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta lo presente al usuario.

Entidades: clases PHP que constituyen una réplica de las entidades de la base de datos en el proyecto.

Bibliotecas y plugins: conjuntos de archivos que permiten agregar nuevas funcionalidades en el proyecto.

ORM Doctrine: conjunto de objetos que permiten acceder a los datos en la base de datos y que contienen en sí mismos el código necesario para hacerlo.

3.3 Estándares de codificación

Cuando se desarrollan aplicaciones web es importante que el código sea fácil de leer y modificar, para esto se deben seguir una serie de normas comunes para todos los desarrolladores. En Symfony3.4 se siguen estándares para facilitar el desarrollo de estas aplicaciones y la contribución de sus comunidades, algunos de estos estándares son:

Indentación: consiste en insertar dos espacios en determinadas líneas de código y no se debe dejar espacios en blanco al final de cada línea para facilitar su comprensión.

Etiquetas de apertura y cierre de PHP: Siempre se debe utilizar la versión de las etiquetas (<?php) y (?>) para abrir y cerrar el código PHP, por lo general se omite la etiqueta de cierre (?>) al final de algunos archivos, evitando que se queden espacios olvidados no deseados al final luego de la etiqueta de cierre (?>), que serían identificados como salida HTML y podrían provocar errores, por lo que la etiqueta de cierre final del archivo (?>) es opcional.

Operadores: Los operadores binarios, utilizados entre dos valores, deben estar separados a ambos lados del operador por un espacio. Esto se aplica a operadores como +, -, *, /, =, ==, !=, >, <, . (Concatenación de cadenas), .=, +=, -=, etc. Los operadores unarios como ++, -- no deben tener separación.

Uso de comillas: pueden ser usadas tanto las comillas simples (‘cadena') como las comillas dobles ("cadena") para delimitar las cadenas de caracteres, si es necesario el uso de variables dentro de la cadena, se deben usar las comillas dobles.

Uso de punto y coma (;) en código PHP: el uso del punto y coma (;) al final del código en las líneas individuales es siempre obligatorio.

Funciones: Los nombres de las funciones se escribirán utilizando el estilo camello (la primera palabra comenzará con minúscula y la letra inicial de las próximas palabras estará en mayúscula), con el objetivo de evitar duplicidad de funciones. Symfony 3.4 establece que el nombre de cada función que se defina debe

terminar con la palabra Action. Después del nombre de la función el paréntesis de los argumentos no debe llevar espacios, pero sí los argumentos entre sí, luego de cada coma (,).

Comentar el código: El uso de los comentarios pueden hacerse utilizando las etiquetas /* */ para comentarios en varias líneas y // para comentarios de una única línea.

Estructuras de control: Las estructuras de control deben cumplir un conjunto de normas para su correcto funcionamiento; las estructuras (if, while, for, etc.) y el primer paréntesis deben tener un espacio intermedio, para no confundirlas con la nomenclatura de las funciones; la llave de apertura ({) estará en la primera línea separada por un espacio y es recomendado usar las dos llaves ({ }) aunque el código lo permita; y las estructuras else y elseif serán escritas en la línea siguiente de la llave de cierre anterior (}).

Arrays: Los valores dentro de un array deben ser separados por un espacio después de la coma. El operador => debe separarse por un espacio a ambos lados. Cuando la línea de declaración del array supera los 80 caracteres, cada elemento se debe escribir en una única línea.

3.4 Pruebas de software

Las pruebas de software permiten determinar si el producto generado satisface las especificaciones establecidas. Permite detectar la presencia de errores que pudieran generar salidas o comportamientos inapropiados durante su ejecución. Las pruebas de software son importantes porque aseguran el correcto cumplimiento de la funcionalidad del producto, ayudan a ganar confianza, confirman la fiabilidad del uso del software y previenen defectos en producción, lo cual tiene un impacto económico positivo en la entidad en cuestión (32). Las pruebas de software son una actividad primordial en el proceso de “aseguramiento de la calidad”. Por tanto, la adecuada aplicación de métodos y herramientas, revisiones técnicas efectivas, gestión y medición sólidas conducen a la calidad del software, que se confirma durante las pruebas (33).

En la evaluación del sistema de gestión de actividades es necesario tener en cuenta varias características dependiendo del tipo de aplicación, además del objetivo que persiguen las pruebas a realizar. Después de analizar las propiedades del sistema web se determinó que es necesario medir su reacción integral frente a diversas acciones que podrán efectuar los usuarios, para ello es conveniente la selección de las siguientes pruebas: pruebas de funcionalidad, pruebas de aceptación, pruebas de carga y pruebas de estrés. A continuación, se describirán las pruebas seleccionadas y los resultados arrojados con su ejecución.

3.4.1. Pruebas funcionales

Las pruebas funcionales están centradas en comprobar que las funcionalidades descritas en el documento de requisitos del sistema se cumplen con la implementación realizada. A este tipo de pruebas también se les denomina pruebas de comportamiento o de caja negra, debido a que los analistas enfocan su atención a las respuestas del sistema de acuerdo a los datos de entrada y sus resultados en los datos de salida, los cuales se definen generalmente en los casos de prueba que se crean antes del inicio de las de las pruebas" (34). El proceso para ejecutar este tipo de pruebas es el siguiente (35):

1. Analizar los requisitos y sus especificaciones.
2. Seleccionar entradas válidas y no válidas de acuerdo con las especificaciones.
3. Determinar las salidas esperadas para cada entrada.
4. Diseñar los casos de pruebas con las entradas seleccionadas.
5. Ejecutar los casos de prueba.
6. Comparar las salidas encontradas con las salidas esperadas.
7. Determinar si el funcionamiento del software en prueba es apropiado.

En el Anexo 3 se muestra el diseño de casos de pruebas correspondiente al requisito 7.

3.4.2. Pruebas de carga

"Las pruebas de carga permiten la simulación del acceso de muchos usuarios a un servidor al mismo tiempo, posibilitando observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas. La carga puede ser el número de usuarios concurrentes que se espera que utilicen la aplicación, y que realizan durante el tiempo en que dura la carga un número específico de transacciones. Este tipo de prueba facilita la monitorización del servidor y la base de datos para obtener el cuello de botella en la aplicación, y puede mostrar los tiempos de respuesta de todas las transacciones importantes" (36).

3.4.3. Pruebas de estrés

“Las pruebas de estrés posibilitan la obtención de datos sobre la carga del sistema. Tiene como objetivo generar cargas en el sistema hasta hacerlo inutilizable, para centrarse en verificar la calidad de los mensajes de error o establecer alertas para anticipar los fallos. Las pruebas de estrés son uno de los últimos tipos de pruebas que se deben efectuar, debido a que tienen un carácter poco realista ya que podría darse el caso que nunca se diera en la vida real la situación de carga simulada” (38).

3.4.4. Pruebas de aceptación

El objetivo en este nivel de prueba es obtener la aprobación del cliente, no se deberían encontrar defectos funcionales graves en el sistema. Es por ello que las pruebas de aceptación son realizadas por el usuario. Se puede decir que las pruebas de aceptación son las pruebas de sistema por parte del cliente. Existen dos tipos de pruebas de aceptación (32):

-Pruebas Alfa: el cliente utiliza el software para hacer el tratamiento de sus procesos de negocio en las dependencias del proveedor.

-Pruebas Beta: se ejecutan en las dependencias del cliente.

Las pruebas de aceptación son consideradas como la fase final del proceso para crear una confianza en que el producto es apropiado para su uso, de esta manera se verificará que el software satisface los requisitos del cliente.

Resultados de las pruebas funcionales

Las pruebas funcionales que fueron aplicadas a la propuesta de solución se realizaron utilizando el documento Diseño de casos de pruebas, recogiendo los distintos escenarios que se correspondían a cada uno de los requisitos funcionales. Para la realización de las pruebas se utilizaron en la ejecución de las historias de usuario datos válidos e inválidos, eligiendo de forma correcta los valores de entrada, con el objetivo de abarcar la mayor cantidad posible de combinaciones, sin hacer que la cantidad de casos de prueba fuera muy elevada.

Para ejecutar las pruebas se realizaron tres iteraciones, en la primera se encontraron 14 no conformidades que fueron solucionadas en su totalidad; en la segunda iteración se detectaron 3 nuevas no conformidades

a las que se les dio solución, para una tercera iteración en la que todas las no conformidades estaban resueltas.

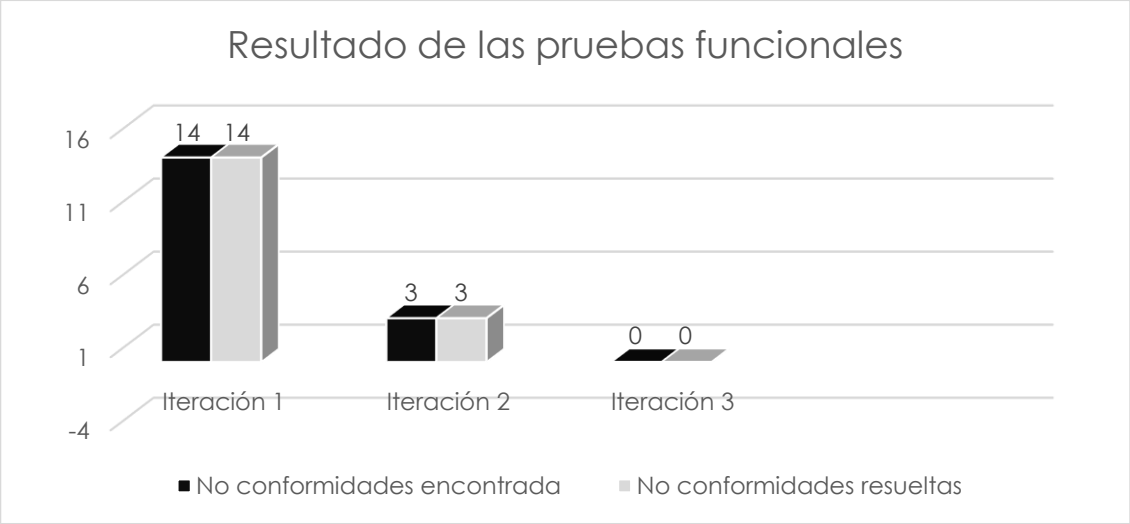


Ilustración 10: Gráfico de resultado de Pruebas funcionales (Elaboración propia).

3.4.5. Pruebas de rendimiento

Resultados de las pruebas de carga y estrés

Para realizar las pruebas de carga y estrés se utilizaron 2 PC, una como cliente y la otra como servidor, creando así un entorno de trabajo en el cual se efectúo un análisis del tráfico de usuarios en el sistema con ayuda de la herramienta Apache JMeter en su versión 4.0. Debido a los datos obtenidos y a que en la DGE se cuenta aproximadamente con 120 usuarios, se realizaron las pruebas de carga y estrés en el caso crítico en que se conecten 100 usuarios concurrentes. Las pruebas de carga y estrés se desarrollaron, en un ambiente utilizando dos ordenadores con las siguientes características:

PC servidor

- Sistema operativo Windows 8.1.
- Microprocesador Intel Dual Core a 1.30 GHz.
- Memoria RAM 3GB.

PC cliente

- Sistema operativo Windows 10.
- Microprocesador Intel Core i3 a 2.60 GHz.
- Memoria RAM 4GB.

Ambas PC

- Características de la red: 100.0 Mbps
- Para poder acceder a esta pantalla debe estar corriendo el servidor web de la aplicación y el servidor de base de datos.

Los resultados obtenidos en las pruebas de carga se consideran satisfactorios. La propuesta de solución generó una buena transferencia de datos para 80 usuarios concurrentes esperados, lo que incurrió en un rendimiento medio de 15,0 segundos. Se demuestra que la propuesta de solución es estable, ya que se mantuvo prestando servicios todo el tiempo sin incurrir en fallos. Además, se probó para un total de 100 usuarios concurrentes, obteniendo los resultados mostrados en la Ilustración 11 y la Tabla 4.

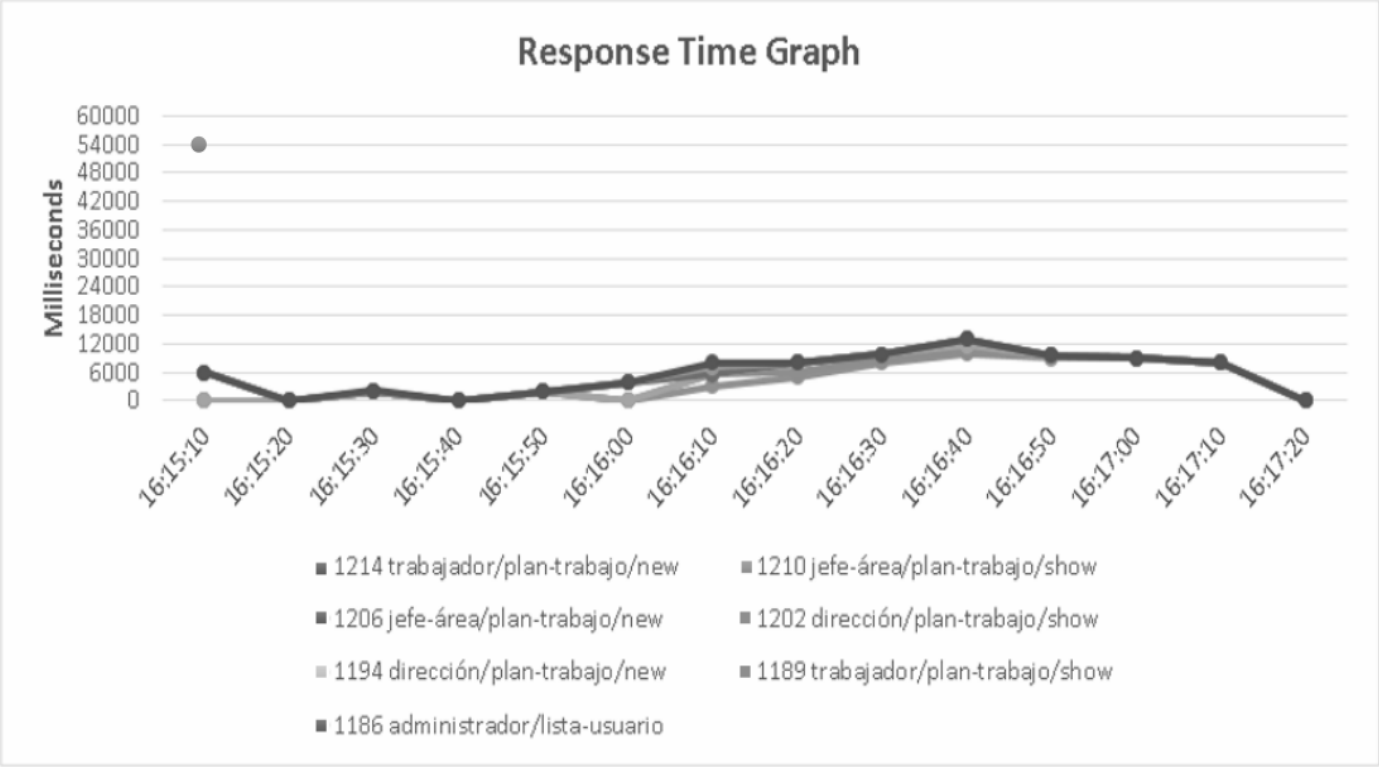


Ilustración 11: Gráfico de resultado de Pruebas funcionales (Elaboración propia)

Tabla 4: Resultados de las pruebas de carga y estrés *(Elaboración propia)*

Etiqueta	#Muestras	Media	Mediana	Línea de 90%	Mín	Máx	%Error	Rendimiento	Kb/sec
1186 administrador/lista-usuario	100	53717	62050	67118	758	131646	0.00%	45.2/sec	4.1
1189 trabajador/plan-trabajo/show	100	6817	7126	12807	666	20631	0.00%	45.0/sec	5.1
1194 dirección/plan-trabajo/new	100	7760	8807	14515	731	20739	0.00%	45.0/sec	5.1
1202 dirección/plan-trabajo/show	100	6734	6852	13283	491	16303	0.00%	45.2/sec	5.1
1206 jefe-área/plan-trabajo/new	100	7183	7854	13857	526	17236	0.00%	45.2/sec	5.1

1210 jefe-área/plan-trabajo/show	100	6678	7290	12327	630	15962	0.00%	45.3/sec	5.1
1214 trabajador/plan-trabajo/new	100	5184	5074	10124	486	14514	0.00%	45.4/sec	5.1
Total	700	13439	8009	42076	486	131646	0.00%	5.27/min	34.7

Resultados de la prueba de aceptación

El resultado de la prueba de aceptación será dado en un documento firmado por el director general de economía, reconociendo el sistema como solución para la gestionar de actividades y confirmando su validez.

3.5 Conclusiones del capítulo

En este capítulo se realizó una descripción de los estándares de codificación utilizados y se aplicaron las pruebas al sistema, concluyendo que:

1. La confección del diagrama de componentes ofreció una vista arquitectónica de alto nivel para ayudar al equipo de desarrollo en la implementación.

2. La definición de los estándares de codificación a tener en cuenta para la implementación del sistema, permitió garantizar que el código posea alta calidad, menos errores y que pueda ser mantenido fácilmente.

3. La realización de las pruebas de software al sistema facilitó la identificación de errores para su rápida solución, posibilitando determinar y asegurar la calidad del sistema de gestión de actividades.

Conclusiones generales

La presente investigación tuvo como base el desarrollo de un sistema de gestión de actividades para la DGE para ello se dio cumplimiento a una serie de objetivos específicos, los cuales fueron cumplidos satisfactoriamente, por lo que se puede arribar a las siguientes conclusiones:

- El estudio realizado de los sistemas homólogos, evidenció la necesidad de implementar un sistema de gestión de actividades para la DGE y facilitar el apoyo a la toma de decisiones.
- La selección de herramientas, lenguajes y tecnologías permitió la implementación del sistema de gestión de actividades.
- La utilización de la estrategia de pruebas garantizó la identificación temprana de las deficiencias en el sistema que se desarrolló; corrigiéndose las mismas logrando un producto más seguro y funcional.
- El desarrollo de SADE renovó el proceso de gestión de actividades en la DGE.

Recomendaciones

Se recomienda utilizar la base de datos de la UCI para autocompletar los datos necesarios en la creación de usuarios.

Referencias bibliográficas

1. Herrera, Luz Marina Carvajal. *Guía para la elaboración del plan de trabajo.* Universidad de Colombia : s.n., 2014.

2. Amador, Javier. *Cubanos.* 2016.

3. Hector. ¿Que es la Actividad en el Trabajo? [En línea] Parra. [Citado el: 13 de 11 de 2018.] http://www.enfoqueocupacional.com/2011/05/que-es-la-actividad-en-el-trabajo.html.

4. María, Bernal Dra. *La Planificación: Conceptos básicos, principios, componentes, característica y desarrollo del proceso.* s.l. : Los Teques, 2012.

5. Medina, Nancy Pérez. SIPAC, solución novedosa para la planificación de actividades | Universidad de las Ciencias Informáticas. [En línea] 31 de Marzo de 2017. [Citado el: 13 de Noviembre de 2018.] http://www.uci.cu/universidad/noticias/sipac-solucion-novedosa-para-la-planificacion-de-actividades.

6. Marvin, By Rob Watts y Rob. Kronos Workforce Ready. [En línea] PCMAG. [Citado el: 13 de noviembre de 2018.] https://www.pcmag.com/ review/341708/kronos-workforce-ready.

7. Rafter, By Rob Watts Michelle V. Deputy. [En línea] PCMAG. [Citado el: 13 de noviembre de 2018.] https://www.pcmag.com/ review/341707/deputy.

8. Martinez, By Juan. Shiftboard. [En línea] PCMAG. [Citado el: 13 de noviembre de 2018.] https://www.pcmag.com/review/351065/shiftboard.

9. Diaz, F. Javier 2009. Las metodologías ágiles como garantía de calidad del software. 2009. págs. 40-43. Vol. 3.

10. Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI v1.2. La Habana, Cuba: s.n., 6 de marzo de 2015.

11. @PHP. PHP. PHP. [En línea] [Citado el: 18 de Enero de 2019.] http://www.php.net/.

12. @MOZDEVNET. *HTML. HTML | MDN.* [En línea] [Citado el: 18 de Enero de 2019.] https://developer.mozilla.org/es/docs/Web/HTML.

13. @MOZDEVNET. *JavaScript. JavaScript | MDN.* [En línea] [Citado el: 18 de Enero de 2019.] https://developer.mozilla.org/es/docs/Web/JavaScript.

14. @MOZDEVNET. *CSS. CSS | MDN.* [En línea] https://developer.mozilla.org/es/docs/Web/CSS.

15. @MDO AND @FAT. Bootstrap. Bootstrap [online]. Available from: http://getbootstrap.com

16. JS FOUNDATION - JS.FOUNDATION. *jQuery. jQuery.* [En línea] [Citado el: 20 de Enero de 2019.] https://jquery.com/jQuery: The Write Less, Do More, JavaScript Library.

17. SYMFONY. What is Symfony. *What is Symfony.* [En línea] [Citado el: 20 de Enero de 2019.] https://symfony.com/what-is-symfony.
18. postgres. PostgreSQL. [En línea] https://www.postgresql.org/about/.
19. NetBeans. NetBeans IDE - Overview . [En línea] https://netbeans.org/features/index.html.
20. FOWLER, MARTIN y SCOTT, KENDALL . UML GOTA A GOTA. S.A. ALHAMBRA MEXICANA, [no date] : ISBN 978-968-444-364-8.
21. Visual Paradigm. Visual Paradigm International. [En línea] [Citado el: 10 de Febrero de 2019.] http://www.visualparadigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html. Herramientas de diseño de software para equipos ágiles con UML,BPMN y más. Soft.
22. IVÁN GARCERANT. Modelo de Dominio. *Tecnología y Synergix | Modelo de Dominio .* [En línea] 10 de Julio de 2008. https://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.
23. ROGER S. PRESSMAN. Ingeniería del Software Un enfoque práctico. Séptima. [no date]. ISBN 978-607-15-0314-5.
24. IEEE - The world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. *IEEE.* [En línea] http://www.ieee.org/.
25. RIOS-PATIÑO., JORGE IVÁN, TELLO-BORJA, WILMAN y JIMENEZ-TORRES, VICTOR HUGO. Lenguajes de Patrones de Arquitectura de Software. *Una Aproximación Al Estado del Arte. Scientia et Technica Año XIX.* 2014. Vol. 19.
26. JUAN, FRANCISCO JAVIER MARTÍNEZ. GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO. *ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA EN INFORMÁTICA DE OVIEDO.*
27. GONZÁLEZ, YENISLEIDY FERNÁNDEZ ROMERO y YANETTE DÍAZ. Patrón Modelo-Vista-Controlador. *Revista Digital de las Tecnología de la Información y las Comunicaciones.* Abril 2012. Vol. 11, p. 47–57.
28. SENSIO LABS. Symfony | MVC. - Symfony . [En línea] https://symfony.com/legacy/doc/gentle-introduction/1_4/en/02-Exploring-Symfony-s-Code.
29. IAN SOMMERVILLE. Ingeniería del Sofware. Séptima. [no date]. ISBN 84'7829'074'5.
30. CRAIG LARMAN. *UML y patrones. .* 2da esdición. s.l. : ALHAMBRA, 2003. ISBN 84-205-3438-2.
31. GAMMA., ERICH. *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995.

32. CINDY CAMPOS CHIU. LAS PRUEBAS EN EL DESARROLLO DE SOFTWARE. UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO. México : s.n., 2015.
33. SANZ, LUIS FERNÁNDEZ. Un sondeo sobre la práctica actual de pruebas de software en España. *Revista Española de Innovación.* 2005. Vol. 1, p. 13–26.
34. ORÉ B., ING. ALEXANDER. pruebas. calidadysoftware.com. [En línea] 2009. http://www.calidadysoftware.com/testing/pruebas_funcionales.php
35. M.SC EDGAR SERNA and DRC. FERNANDO ARANGO. Prueba del software: más que una fase en el ciclo de vida/Software testing: more than a stage in the life cycle. Diciembre 2011. Vol. 35.
36. ARCHIVEDDOCS. *Performance Testing Guidance for Web Applications* | *Microsoft Docs .* [En línea] https://docs.microsoft.com/en-us/previous-versions/msp-n-p/bb924375(v=pandp.10).
37. FRAILE, LUIS. Pruebas de Rendimiento. [En línea] 2011. http://www.globetesting.com/pruebas-derendimiento.
38. *Public Records*. https://www.apache.org/foundation/records/. Accedido 21 de mayo de 2019.

Anexos

Anexo1: Entrevista

Estimado especialista: Se necesita de su cooperación en una investigación para una tesis de pregrado. Para ello, sería de gran ayuda que respondiera las siguientes preguntas:

1. ¿Cómo se realizan actualmente la gestión de actividades en la Dirección General de Economía?

2. ¿Qué características presentan la gestión de actividades?

3. ¿Cómo se manejan los datos de las actividades?

4. ¿Cómo afecta la falta de un sitio para la gestión de actividades en la Dirección General de Economía?

5. ¿Cuáles son las Reglas del negocio?

6. ¿Cuál es la idea que se tiene para sitio?

7. ¿Qué no le puede faltar al sistema?

Estimado cliente: Se necesita de su cooperación para una investigación de pregrado. Para ello sería de gran ayuda que respondiera lo siguiente:

1. ¿Considera usted que el sistema responde con las demandas solicitadas?

2. ¿Con cuáles no cumple? ¿En qué aspectos se podría mejorar?

3. ¿Le aporta algún beneficio el sistema?

Anexo 2: Historias de usuario

RF1. Autenticar usuario

Número: 1	**Nombre del requisito**: Autenticar usuario
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 2días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir autenticar usuario. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para autenticar usuario hay que: - Estar registrado en el sistema. **3- Comportamientos válidos y no válidos (flujo central y alternos):** Los campos usuario y contraseña son obligatorios. Usuario: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres Contraseña: campo de texto que permite cualquier carácter **4- Flujo de la acción a realizar:** - El usuario debe rellenar los siguientes campos - Usuario - Contraseña - Se selecciona el botón Autenticarse - El sistema valida los datos insertados - Si los datos son correctos el usuario se autentica - Concluye el requisito Flujo alternativo. Datos incorrectos - El sistema señala los datos incorrectos y permite corregirlos. - Se corrigen los datos. Flujo alternativo. Campos en blanco - El sistema señala los campos en blanco y permite rellenarlos. - Se validan los datos.	
Observaciones:	
Prototipo de interfaz:	

RF2. Registrar usuario

Número: 2	**Nombre del requisito:** Registrar usuario
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 5días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:
1- Objetivo:
Permitir registrar un nuevo usuario.
2- Acciones para lograr el objetivo (precondiciones y datos):
Para registrar un usuario hay que:
- No estar autenticado en el sistema.

3- Comportamientos válidos y no válidos (flujo central y alternos):
Los campos nombre de usuario, apellido, contraseña, caro, área, son obligatorios.
Nombre de usuario: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres
Contraseña: campo de texto que admite caracteres alfanuméricos y tiene cualquier cantidad de caracteres.
Nombre y apellidos: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 255 caracteres.
Cargo: responsabilidad que ocupa el usuario.
Área: lugar de trabajo al que pertenece el usuario

4- Flujo de la acción a realizar:
- El sistema debe permitir ingresar los datos para incluir un nuevo usuario dando clic en el hipervínculo registrar.
- Se selecciona el hipervínculo registrar en la vista de iniciar sesión.
- Se muestra un formulario a rellenar
 - Nombre de usuario
 - Contraseña
 - Nombre y apellidos
- Cargo
- Área
- Se define la opción registrar
- El sistema valida los datos insertados

- Si los datos son correctos el sistema registra al usuario - Concluye el requisito Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos. -Se corrigen los datos. Flujo alternativo. El usuario cancela la acción -Concluye el requisito Flujo alternativo. Coincidencias con el nombre de usuario -El sistema busca el nombre de usuario ingresado uno igual, lanza una alerta y permite modificar los valores.
Observaciones:
Prototipo de interfaz:

RF3. Modificar usuario

Número: 3	**Nombre del requisito:** Modificar usuario
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir modificar los datos personales de un usuario **2- Acciones para lograr el objetivo (precondiciones y datos):** Para editar los datos personales de un usuario hay que: - Estar autenticado en el sistema con el rol de administrador. - El usuario debe existir en el sistema. **3- Comportamientos válidos y no válidos (flujo central y alternos):** El proceso de editar los datos personales depende del rol Administrador **4- Flujo de la acción a realizar:** - El sistema debe permitir editar los datos personales, esta acción puede realizarse seleccionando el nombre en la barra de menú. - Cuando el usuario edita de forma correcta los datos necesarios y selecciona la opción Actualizar.	

Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos. -Se corrigen los datos.
Observaciones:
Prototipo de interfaz:

RF4. Listar usuario

Número: 4	**Nombre del requisito:** Listar usuario
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: 1- Objetivo: Permitir listar los usuarios en el sistema **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los usuarios en el sistema hay que: - Estar autenticado en el sistema con el rol Administrador. - Debe existir en el sistema al menos un usuario. **3- Flujo de la acción a realizar:** Cuando el usuario despliega la opción Usuario, selecciona la opción Gestionar aparecen todos los usuarios registrados en el sistema. Además, el administrador tiene la posibilidad de ver detalles de un usuario, editarlo y eliminarlo.	
Observaciones:	
Prototipo de interfaz:	

RF5. Eliminar usuario

Número: 5	**Nombre del requisito:** Eliminar usuario
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A

Prioridad: Alta	**Tiempo Estimado:** 2días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir eliminar un usuario existente **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar un usuario hay que: - Estar autenticado en el sistema con el rol Administrador. - Debe existir en el sistema al menos un usuario. **3- Flujo de la acción a realizar:** Se elige el botón Usuario en el menú y luego se despliega la opción Gestionar, donde se selecciona el usuario a eliminar. - El sistema permite eliminar un usuario seleccionando, la opción eliminar de las opciones que muestra el propio elemento. Luego el listado se actualizará.	
Observaciones:	
Prototipo de interfaz:	

RF 7. Insertar actividad

Número: 7	**Nombre del requisito:** Insertar actividad
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 5días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: 1- Objetivo: Permitir insertar una nueva actividad. 2- Acciones para lograr el objetivo (precondiciones y datos): Para registrar una actividad hay que: - Estar autenticado en el sistema. **3- Comportamientos válidos y no válidos (flujo central y alternos):**	

Los campos nombre, hora, fecha, lugar, responsable, son obligatorios. Nombre: campo donde se registra el nombre de la actividad, de tipo texto donde que admite caracteres alfabéticos. Hora: campo de texto que admite caracteres numéricos. Fecha: campo de texto. Lugar: responsabilidad que ocupa el usuario. Responsable: persona encargada de la actividad. **4- Flujo de la acción a realizar:** - El sistema debe permitir ingresar los datos para incluir una nueva actividad dando clic en el hipervínculo insertar. - Se selecciona el hipervínculo insertar en la vista de iniciar sesión. - Se muestra un formulario a rellenar - Nombre - Hora - Fecha - Lugar - Responsable - Se define la opción insertar - El sistema valida los datos insertados - Si los datos son correctos el sistema registra la actividad - Concluye el requisito Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos. -Se corrigen los datos. Flujo alternativo. El usuario cancela la acción -Concluye el requisito
Observaciones:
Prototipo de interfaz:

RF8. Modificar actividad

Número: 8	**Nombre del requisito:** Modificar usuario
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A

Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir modificar la actividad seleccionada **2- Acciones para lograr el objetivo (precondiciones y datos):** Para editar la actividad hay que: - Estar autenticado en el sistema con el rol de Administrador. - La actividad debe existir en el sistema. **3- Comportamientos válidos y no válidos (flujo central y alternos):** El proceso de editar la actividad depende del rol Administrador **4- Flujo de la acción a realizar:** - El sistema debe permitir editar la actividad. - Cuando el usuario edita de forma correcta los datos necesarios y selecciona la opción Editar. Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos. -Se corrigen los datos.	
Observaciones:	
Prototipo de interfaz:	

RF9. Eliminar actividad

Número: 9	**Nombre del requisito:** Eliminar actividad
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 2días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir eliminar una actividad existente **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar una actividad hay que:	

- Estar autenticado en el sistema con el rol Administrador. - Debe existir en el sistema al menos una actividad. **3- Flujo de la acción a realizar:** Se elige el botón actividad en el menú y luego se despliega la lista de actividades, donde cada actividad tiene un botón eliminar. - El sistema permite eliminar la actividad seleccionando, la opción eliminar de las opciones que muestra el propio elemento. Luego el listado se actualizará.
Observaciones:
Prototipo de interfaz:

RF11. Listar actividad

Número: 11	**Nombre del requisito:** Listar actividad
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: 1- Objetivo: Permitir listar las actividades en el sistema **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar las actividades en el sistema hay que: - Estar autenticado en el sistema con el rol Administrador. - Debe existir en el sistema al menos una actividad. **3- Flujo de la acción a realizar:** Cuando el usuario despliega la opción Actividad, aparecen todas las actividades registradas en el sistema.	
Observaciones:	
Prototipo de interfaz:	

RF12. Insertar plan de trabajo

Número: 12	**Nombre del requisito:** Insertar plan de trabajo
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 5días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:
1- Objetivo:
Permitir insertar un nuevo plan de trabajo.
2- Acciones para lograr el objetivo (precondiciones y datos):
Para registrar un plan de trabajo hay que:
- Estar autenticado en el sistema.

3- Comportamientos válidos y no válidos (flujo central y alternos):
Los campos mes, año, actividad general son obligatorios.
mes: campo que contiene una lista con los 12 meses del año.
año: campo de texto que admite caracteres numéricos.
Actividad general: campo de texto que admite caracteres alfabéticos.

4- Flujo de la acción a realizar:
- El sistema debe permitir ingresar los datos para incluir un nuevo plan de trabajo dando clic en el hipervínculo insertar.
- Se selecciona el hipervínculo insertar en la vista.
- Se muestra un formulario a rellenar
- Mes
- Año
- Actividad general
- Se define la opción insertar
- El sistema valida los datos insertados
- Si los datos son correctos el sistema registra el plan
- Concluye el requisito

Flujo alternativo. Datos incorrectos o incompletos
-El sistema señala los datos incorrectos o incompletos y permite corregirlos.
-Se corrigen los datos.
Flujo alternativo. El usuario cancela la acción
-Concluye el requisito

Observaciones:

Prototipo de interfaz:

RF13. Modificar plan de trabajo

Número: 13	**Nombre del requisito:** Modificar plan de trabajo
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir modificar el plan de trabajo seleccionado **2- Acciones para lograr el objetivo (precondiciones y datos):** Para editar el plan de trabajo hay que: - Estar autenticado en el sistema. - El plan de trabajo debe existir en el sistema. **3- Comportamientos válidos y no válidos (flujo central y alternos):** El proceso de editar el plan de trabajo depende del rol usuario **4- Flujo de la acción a realizar:** - El sistema debe permitir editar el plan de trabajo. - Cuando el usuario edita de forma correcta los datos necesarios y selecciona la opción Editar. Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos. -Se corrigen los datos.	
Observaciones:	
Prototipo de interfaz:	

RF14. Eliminar plan de trabajo

Número: 14	**Nombre del requisito:** Eliminar plan de trabajo
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A

Prioridad: Alta	**Tiempo Estimado:** 2días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir eliminar un plan de trabajo existente **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar un plan de trabajo hay que: - Estar autenticado en el sistema. - Debe existir en el sistema al menos un plan de trabajo. **3- Flujo de la acción a realizar:** Se elige el botón Plan de trabajo en el menú y luego se despliega la lista de plan de trabajo, donde cada plan de trabajo tiene un botón eliminar. - El sistema permite eliminar un plan de trabajo seleccionando, la opción eliminar de las opciones que muestra el propio elemento. Luego el listado se actualizará.	
Observaciones:	
Prototipo de interfaz:	

RF16. Listar plan de trabajo

Número: 16	**Nombre del requisito:** Listar plan de trabajo
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: 1- Objetivo: Permitir listar los planes de trabajos en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los planes de trabajos en el sistema hay que: - Estar autenticado en el sistema. - Debe existir en el sistema al menos un plan de trabajo.	

3- Flujo de la acción a realizar: Cuando el usuario despliega la opción Plan de trabajo, aparecen todos los planes de trabajos registrados en el sistema.
Observaciones:
Prototipo de interfaz:

RF17. Insertar informe de cumplimiento

Número: 17	**Nombre del requisito:** Insertar informe de cumplimiento
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 5días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:
1- Objetivo:
Permitir insertar un nuevo informe de cumplimiento.
2- Acciones para lograr el objetivo (precondiciones y datos):
Para registrar un informe de cumplimiento hay que:
- Estar autenticado en el sistema.
- Debe existir en el sistema al menos un plan de trabajo

3- Comportamientos válidos y no válidos (flujo central y alternos):
Los campos mes, año, resultado son obligatorios.
mes: campo que contiene una lista con los 12 meses del año.
año: campo de texto que admite caracteres numéricos.
Resultado: campo de texto que admite caracteres alfabéticos.

4- Flujo de la acción a realizar:
- El sistema debe permitir ingresar los datos para incluir un nuevo informe de cumplimiento dando clic en el hipervínculo insertar.
- Se selecciona el hipervínculo insertar en la vista.
- Se muestra un formulario a rellenar
 - Mes
- Año
 - Resultados

- Se define la opción insertar - El sistema valida los datos insertados - Si los datos son correctos el sistema registra el plan - Concluye el requisito Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos. -Se corrigen los datos. Flujo alternativo. El usuario cancela la acción -Concluye el requisito
Observaciones:
Prototipo de interfaz:

RF18. Modificar informe de cumplimiento

Número: 18	**Nombre del requisito:** Modificar informe de cumplimiento
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir modificar el informe de cumplimiento seleccionado **2- Acciones para lograr el objetivo (precondiciones y datos):** Para editar el informe de cumplimiento hay que: - Estar autenticado en el sistema. - El informe de cumplimiento debe existir en el sistema. **3- Comportamientos válidos y no válidos (flujo central y alternos):** El proceso de editar el plan de trabajo depende del rol usuario **4- Flujo de la acción a realizar:** - El sistema debe permitir editar el informe de cumplimiento. - Cuando el usuario edita de forma correcta los datos necesarios y selecciona la opción Editar. Flujo alternativo. Datos incorrectos o incompletos -El sistema señala los datos incorrectos o incompletos y permite corregirlos.	

-Se corrigen los datos.
Observaciones:
Prototipo de interfaz:

RF19. Eliminar informe de cumplimiento

Número: 19	**Nombre del requisito:** Eliminar informe de cumplimiento
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 2días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir eliminar un informe de cumplimiento existente **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar un informe de cumplimiento hay que: - Estar autenticado en el sistema. - Debe existir en el sistema al menos un informe de cumplimiento. **3- Flujo de la acción a realizar:** Se elige el botón informe de cumplimiento en el menú y luego se despliega la lista de informe de cumplimiento, donde cada informe de cumplimiento tiene un botón eliminar. - El sistema permite eliminar un informe de cumplimiento seleccionando, la opción eliminar de las opciones que muestra el propio elemento. Luego el listado se actualizará.	
Observaciones:	
Prototipo de interfaz:	

RF21. Listar informe de cumplimiento

Número: 21	**Nombre del requisito:** Listar informe de cumplimiento
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A

Prioridad: Alta	**Tiempo Estimado:** 3días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: 1- Objetivo: Permitir listar los informes de cumplimiento en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los informes de cumplimiento en el sistema hay que: - Estar autenticado en el sistema. - Debe existir en el sistema al menos un informe de cumplimiento. **3- Flujo de la acción a realizar:** Cuando el usuario despliega la opción Informe, aparecen todos los informes de cumplimiento registrados en el sistema.	
Observaciones:	
Prototipo de interfaz:	

RF22. Exportar pdf

Número: 22	**Nombre del requisito:** Exportar pdf
Programador: Grisel Inés Rodríguez Machado	**Iteración Asignada:** N/A
Prioridad: Alta	**Tiempo Estimado:** 5días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir exportar pdf en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para exportar pdf hay que: - Estar autenticado en el sistema con el rol Administrador. **4- Flujo de la acción a realizar:**	
Observaciones:	
Prototipo de interfaz:	

Anexo 3: Diseño de Casos de pruebas

A continuación, se muestra el caso de prueba SC1 en correspondencia con el requisito RF7:

Descripción General:

Permitir registrar una nueva actividad.

Condiciones de ejecución:

Estar autenticado en el sistema.

SC1. Registrar actividad

Descripción General:

El sistema debe mostrar los datos de la actividad seleccionado.

Condiciones de ejecución:

- Estar autenticado en el sistema con el rol Administrador.

- Las actividades deben existir en una lista.

SC2. Mostrar los datos de la actividad

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de mostrar los datos de la actividad	Selecciona la opción de ver los datos de una actividad.	Muestra los siguientes datos de la actividad: Nombre de la actividad –Fecha –Hora –Lugar -Responsable Permite además: - Editar los datos de la actividad. - Eliminar los datos la actividad.	Actividad/Gestionar/Mostrar

EC 1.2 Opción de modificar los datos	Selecciona la opción de Editar los datos de la actividad.	El sistema brinda la posibilidad de modificar los datos de la actividad.	Actividad/Gestionar /Mostrar/Editar