Facultad de Ciencias y Tecnologías Computacionales

Soporte al Sistema de Gestión de Alimentación de la UCI en el proceso de gestión de horarios

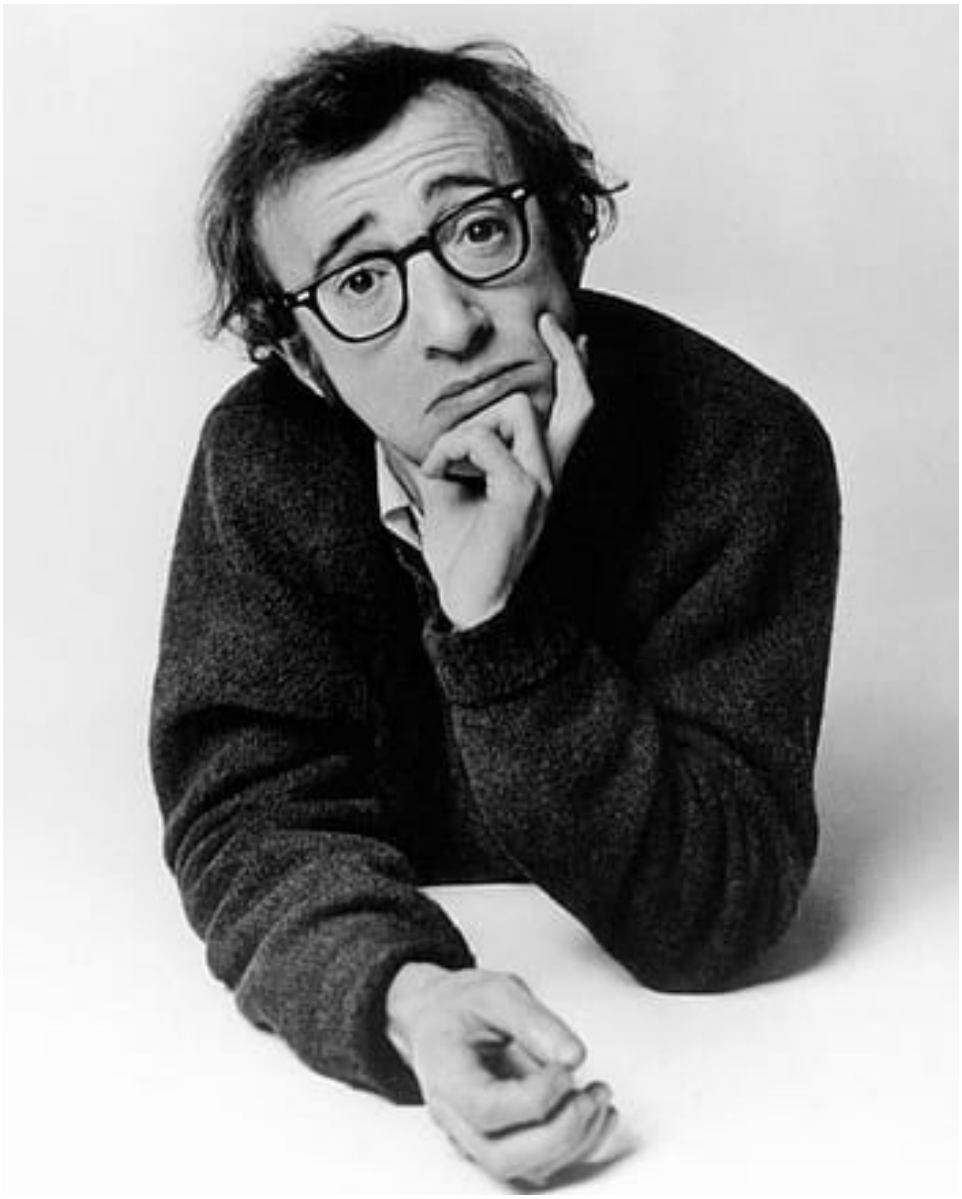
Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: **Zoe Ramírez López**

Tutor: **Ing. Dayron Mederos Ramos**

La Habana, noviembre de 2022

Año 64 de la Revolución



El 80% del éxito se basa simplemente en insistir.

Woody Allen

Declaración de autoría

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título *"***Soporte al Sistema de Gestión de Alimentación de la UCI en el proceso de gestión de horarios** "concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los <día> días del mes de <mes> del año <año>.

Para que así conste firma(n) la presente a los <día> días del mes de <mes> del año <año>.

Zoe Ramírez López

_____
Firma del Autor

Ing. Dayron Alejandro Mederos Ramos

_____
Firma del Tutor

DATOS DE CONTACTO

Tutor:

Ing. Dayron Alejandro Mederos Ramos: Graduado en el año 2019 como Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se encuentra trabajando de adiestrado en el Departamento de Tecnología de la Dirección de Informatización ocupando el rol de programador en el proyecto Sistema Gestión de Alimentación. Correo electrónico: damederos@uci.cu

RESUMEN

La Dirección de Informatización (DIN) se encuentra trabajando en el Sistema de Gestión de Alimentación (SIGA) para la Universidad de las Ciencias Informática (UCI) dicho sistema se realizó con el propósito de que este evolucionara de aplicación de escritorio a sistema web y así facilitar el trabajo de las personas que trabajan con él. La presente investigación tiene como objetivo dar soporte al SIGA ya que su proceso no cuenta con la aceptación, ni con las exigencias de la Dirección General de Alimentos para generar los horarios por individual. Se quiere contribuir a que el sistema tenga un mejor funcionamiento. La investigación estuvo orientada a fundamentar y documentar el proceso de desarrollo del Módulo de Configuración para la gestión de horarios individualmente. El proceso fue guiado por la metodología AUP-UCI escenario 4 y la arquitectura escogida fue Modelo-Vista-Controlador. Para lograr una aplicación con la menor cantidad de errores, se le realizaron pruebas funcionales y unitarias.

ABSTRACT

The Directorate of Informatisation (DIN) is working on the Food Management System (SIGA) for the University of Informatics Sciences (UCI). This system was developed with the purpose of evolving from a desktop application to a web system, thus facilitating the work of the people who work with it. The objective of this research is to support the SIGA, since its process does not have the acceptance or the requirements of the General Directorate of Food to generate individual timetables. The aim is to contribute to a better functioning of the system. The present research was oriented to substantiate and document the development process of the Configuration Module for the management of individual timetables. The process was guided by the AUP-UCI scenario 4 methodology and the architecture chosen was Model-View-Controller. In order to achieve an application with the least amount of errors, functional and unit tests were carried out.

Palabras clave:

Alimentación, horario, módulo, planificación, soporte

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El creciente uso de las Tecnologías de la Información y las Comunicaciones (TIC) en la última década está revolucionando la sociedad. Están presentes en gran parte de las actividades humanas como son la educación, la comunicación, la medicina y el mundo empresarial (González Espinosa, Alfonso Posada et al. 2015).

Las TIC en el mundo representa un conjunto de herramientas y técnicas relacionadas con la transmisión, procesamiento y almacenamiento digitalizado de la información. En la actualidad existen numerosos cambios en los mercados, competencias entre organizaciones, tecnologías y sociedades, razón por la cual se considera poco pertinente seguir maniobrando bajo el mismo enfoque tradicional. Para lograr ser competitivo dentro de este entorno tan cargado de dinamismo y turbulencia, es indispensable buscar ventajas competitivas y por ende un desarrollo económico a largo plazo, así como también desarrollar una capacidad para producir, circular y utilizar correctamente la información, la comunicación y el conocimiento, ellos constituyen la materia prima de esta nueva sociedad (Granda Asencio, Espinoza Freire et al. 2019).

Los sistemas de información son uno de los componentes más relevantes del entorno empresarial actual. Tienen la capacidad de recopilar, procesar, distribuir y compartir datos de manera integrada y oportuna. Permiten que los empleados sean más eficientes, lo que se refleja en una mejora de los procesos, la administración y el manejo de la información, lo que repercute positivamente en la productividad y competitividad de las empresas (Abrego Almazán, Sánchez Tovar et al. 2017).

La gestión de la información es el conjunto de las actividades que se realizan con el propósito de adquirir, procesar, almacenar y finalmente recuperar, de manera adecuada, la información que se produce o se recibe en una organización. (Suárez Alfonso, Cruz Rodríguez et al. 2015).

En los últimos años el gobierno de Cuba ha dado prioridad a la informatización de la sociedad cubana. Con la COVID-19 dicha informatización tomo mayor auge, los medios televisivos y las redes sociales se dedicaron día tras día ha mantener informado a la población de todo lo que estaba aconteciendo en el país, el nauta hogar, los datos móviles, aplicaciones como el toDus y el transfermóvil son ejemplos claros de los pasos que va dando la sociedad. La UCI fue creada con la misión de formar profesionales para la informatización de nuestro país.

Dentro de su estructura cuenta con la DIN, encargada de manejar el proceso de informatización en la misma. Uno de los sistemas desarrollados por la DIN es el SIGA que se encuentra en su versión 1.0, es el sistema que controla toda la Dirección General de Alimento (DGA) de

la UCI, en este se incluyen los 3 complejos comedores y los pantrys de la escuela. SIGA cuenta con 8 módulos (Abastecimiento, reservación, distribución, cajero, facturación, reportes, configuración y administración).

Actualmente existen un problema con el módulo de configuración para gestionar los horarios correctamente, este se encuentra centralizado para los 3 complejos comedores, lo que implica que cualquiera de los 3 pueden extender los eventos por alguna situación que se presente y dicho evento cambiara para los otros 2 complejos restantes. El proceso se vuelve tedioso pues los administradores se tienen que llamar para decir que el horario fue cambiado ya que el sistema no emite ninguna alerta. A raíz de esto se pueden destacar problemas como: ocasionalmente a la hora del doble, algún complejo se queda sin comida y atrasa el evento sin comunicarlo, los cajeros no detectarán que se realizó dicho cambio a no ser que se refresque el sistema y las personas que se encuentren marcando solapín en ese momento no estarán registradas alterando así la información. Se afecta el por ciento de aprovechamiento cuando una persona que no ha comido llega en el evento doble y marca su solapín, el sistema lo toma como evento doble y se requiere que lo tome como accedido de evento sencillo así no habría necesidad de atrasar el horario. La mala gestión del horario también influye en que los administradores tengan que estar continuamente revisando el sistema por si ocurren cambios, se necesita que la gestión sea lo más automática posible, que después de terminado cada evento vuelva a retomar su horario predeterminado, que brinde respuestas rápidas de cualquier cambio que suceda y que sea un sistema más cómodo de utilizar.

Con la situación problemática antes descrita se plantea como ***problema de la investigación:***

¿Cómo contribuir a la gestión y generación de horarios individuales para cada complejo comedor de la UCI?

Presentando como ***objeto de estudio*** *Sistema de Gestión de Alimentación* y como ***campo de acción*** queda enmarcado específicamente en el proceso de *gestión de horarios del módulo de configuración para SIGA en la UCI.*

Planteándose como ***objetivo general:*** Soporte al módulo de configuración para una mejor gestión del sistema y que este a su vez controle por separado los horarios de los 3 complejos comedores.

A partir del objetivo general y en víspera de darle solución se derivan los siguientes ***objetivos específicos:***

- ✓ Sistematizar los principales referentes teóricos metodológicos del tema.
- ✓ Definir los requerimientos de la gestión de horarios del módulo de configuración.
- ✓ Implementar la gestión de horarios del módulo de configuración.
- ✓ Validar la propuesta a partir de la aplicación práctica implementada para la gestión de horario en el módulo de configuración.

Para dar cumplimiento al objetivo planteado se proponen las siguientes ***Tareas a realizar:***

- ✓ Elaboración del marco teórico para la sistematización de sus principales aspectos.
- ✓ Definición de las tecnologías, herramientas y metodología a utilizar para el desarrollo de la gestión de horarios del módulo de configuración.
- ✓ Selección de los requisitos funcionales y no funcionales para la realización de la gestión de horarios del módulo de configuración.
- ✓ Implementación del módulo de configuración para dar solución al problema planteado.

Para realizar esta investigación se utilizaron los *siguientes* ***métodos de investigación científico:***

Teóricos

- **Histórico -Lógico:** se empleó con el objetivo de verificar cómo evolucionan teóricamente los sistemas que gestionan horarios.

- **Inductivo – Deductivo:** Este método se aplicó durante la investigación para obtener una propuesta de solución a partir del análisis de diferentes arquitecturas y herramientas existentes en internet, así obtener una propuesta de solución.

- **Modelación:** se utilizó para modelar las diferentes etapas de la ingeniería de *software* por las que transita el desarrollo del Módulo de configuración para gestionar horarios.

Empíricos:

- **Observación:** Se emplea durante la investigación para observar y estudiar el funcionamiento de los sistemas que gestionan horarios. Permite alcanzar una mayor comprensión del problema, especifica el objeto de estudio y el campo de acción proporcionando un enfoque a la investigación hacia una posible solución.

- **Entrevista:** se realizó a los directores de cada complejo comedor con el objetivo de recopilar toda la información necesaria para así modificar los horarios.

La estructuración del contenido queda divido en tres capítulos:

Capítulo 1. Fundamentación teórica del sistema: En este capítulo se definen aquellos conceptos que son de vital importancia para el desarrollo y entendimiento del problema de la investigación. Se realiza un análisis de las principales tendencias actuales de la gestión de horarios con el objetivo de determinar el enfoque adecuado para la solución. Además, analiza la metodología de desarrollo de software, tecnologías, herramientas y lenguajes que se usarán para la implementación del sistema.

Capítulo 2. Análisis y Diseño del sistema: Se describe el flujo actual del proceso de planificación, así como la propuesta de solución que responde a la problemática planteada, se definen los requisitos funcionales y no funcionales que debe cumplir y se describen a grandes rasgos las características de la solución propuesta mediante los diversos artefactos que especifica la metodología de desarrollo de software utilizada.

Capítulo 3. Implementación y pruebas a la solución propuesta: se detallan los estándares de codificación usados, se definen los tipos de pruebas que se realizaron con el objetivo de validar que la solución funcione de manera correcta y cumpla con las especificaciones planteadas en las descripciones funcionales.

CAPÍTULO I: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS PARA EL DESARROLLO DEL PROCESO DE GESTIÓN DE HORARIO EN EL MÓDULO DE CONFIGURACIÓN

Introducción

En este capítulo se realiza el estudio de soluciones que gestionan horarios en el ámbito nacional e internacional, para ello se tienen en cuenta características que permiten comprender su funcionamiento y cuánto aportan a la investigación en curso. Se abordan conceptos asociados a la gestión del módulo de Configuración. Se describen brevemente la metodología, las herramientas, tecnologías y el lenguaje.

1.1 Conceptos y definiciones

1.1.2 Gestión

“El arte de la gestión ha sido definido, 'como saber exactamente lo que quieres que hagan los hombres, y luego ver que lo hagan de la mejor manera y más económica'(Taylor 1911).

Gestionar es pronosticar y planificar, organizar, mandar, coordinar y controlar. Prever y proporcionar medios para examinar el futuro y elaborar el plan de acción. Organizar significa construir la estructura dual, material y humana, de la empresa. Mandar significa mantener la actividad entre el personal. Coordinar significa unir, unificar y armonizar toda actividad y esfuerzo. Controlar significa velar por que todo ocurra de conformidad con la regla establecida (Robbins and Coulter 2016).

“Administración: El acto de trabajar con y a través de un grupo de personas para lograr una meta u objetivo deseado de manera eficiente y eficaz” (Gulati, Mayo et al. 2016).

Gestión es el proceso mediante el cual el directivo o equipo directivo determinan las acciones a seguir según los objetivos institucionales, necesidades detectadas, cambios deseados, nuevas acciones solicitadas, implementación de cambios de mandos o necesarios y la forman como se realizan estas acciones y los resultados que se lograrán (Kaehler and Grundei 2018).

Después de revisar una serie de conceptos, se entiende que gestión es un conjunto de acciones que se realizan para llevar a cabo un proyecto y lograr un determinado objetivo. Se hace referencia al efecto y la acción de administrar o gestionar asumiendo que gestionar es el proceso que se ocupa del funcionamiento u organización de una entidad.

1.1.3 Gestión de configuración

La Gestión de la Configuración (GC) tiene por objetivo controlar los activos y elementos de configuración que forman parte de la infraestructura TIC, asegurando su contribución al valor de los servicios o productos institucionales (Fernandez 2019).

El propósito de la gestión de configuración (GC) es establecer y mantener la integridad de cada uno de los productos que se van obteniendo del ciclo de desarrollo de software. La GC representa un elemento clave en el proceso de desarrollo de software ya que proporciona estabilidad a la producción de software, controla el cambio continuo y concurrente que viene con la evolución del producto de software y obliga a implementar estrategias de versiona miento (Paredes and Mosquera).

La gestión de la configuración se encarga de indicar al equipo cómo debe trabajar con la información que maneja, ya sea la generada por el propio equipo o la recibida de otro dispositivo (Merelo Hernández 2017) .

Después de revisar una serie de conceptos se entiende que la gestión de la configuración es un proceso que busca mantener los servidores, los sistemas informáticos, y el software en un estado deseado y uniforme, garantiza que funcione como se espera a medida que se realizan cambios y a su vez evita que estos se realicen sin ser documentados por muy importantes o pequeños que sean. Si se necesita algún cambio en el proyecto se trabaja desde la misma configuración por lo que se puede considerar la base del sistema.

1.2 Soluciones existentes que gestionan horarios

Actualmente existen soluciones encargadas de gestionar Horarios. Por tal razón se hace necesario realizar un estudio de las mismas, con el objetivo de definir si cubren las necesidades planteadas en la problemática descrita y si pueden ser adaptadas a la Universidad de las Ciencias Informáticas. En caso contrario, este estudio permitirá analizar todos los aspectos que puedan ser de utilidad para el desarrollo de la propuesta de solución, ajustada a las necesidades de la universidad en cuanto a la gestión de horarios.

Sistema experto basado en minería de datos y programación entera lineal para soporte en la asignación de materias y diseño de horarios en educación superior.

Es un sistema informático creado de una versión extendida y ampliada del trabajo presentado en la Conferencia Internacional de Sistemas de Información yCiencias de la Computación -INCISCOS (*International Conference on Information Systems and Computer Science*).Que presenta una propuesta para el desarrollo de un sistema experto que se fundamenta en minería de datos y programación entera lineal para realizar dos complejas tareas de la gestión universitaria: la asignación de cursos (materias o asignaturas) a docentes (considerando la pertinencia y perfil de los docentes) y la generación automática de horarios. El sistema de recomendación horaria permite ubicar lo mejor posible a un docente en un horario específico de clase según sus variables biográficas y de orden institucional, permite la identificación de espacios de tiempo para otras acciones como la conformación de claustros docentes que interactúan alrededor de procesos de innovación educativa. Con el fin de probar el algoritmo encargado de la generación de horarios, se implementó un script en Python, utilizando la librería *PuLP* y el *SolverGLPK* (Calle-López, Cornejo-Reyes et al. 2018).
No es una solución que cubre los requisitos de la propuesta de solución incorpora otras funcionalidades que no son necesarias. No contempla entre sus procesos de negocio la gestión descentralización de los horarios

Bizneo

Bizneo es un *software* de control horario que optimiza la jornada de trabajo. Un programa en la nube que simplifica la gestión de vacaciones, ausencias y turnos de los empleados, permitiendo registrar la presencia a través de diferentes métodos para fichar y dejar constancia de las horas trabajadas en la empresa. Con Bizneo el fichaje de los trabajadores se lleva a cabo desde cualquier dispositivo (móvil, tablet o PC). Su aplicación para Android cuenta con un sistema de geolocalización que permite al personal registrar fácilmente su jornada desde cualquier lugar. En el módulo denominado “Gestión del tiempo” se presentan diferentes funcionalidades como la emisión de todo tipo de informes, que pueden ser descargados en for-

mato .xml o .pdf. Además, la información de los empleados y sus datos de acceso se almacenan en un entorno seguro, accesible y adaptado a la ley (Perez 2022).

Es un software privativo propuesta diseñada para empresas donde se aplique el perfeccionamiento empresarial. Los procesos que la componen no se describen detalladamente porque depende de las características de cada empresa. La Universidad de las Ciencias Informáticas a pesar de ser una institución creada para la producción de *software* que vincula los procesos de docencia-investigación-producción es considerada una Universidad y no una empresa, por tal razón este modelo no responde a las necesidades de la misma en cuanto a la gestión de la configuración de horarios.

Factorial Control Horario

Factorial *software* de control horario que ofrece una forma fácil de registrar el tiempo exacto de trabajo de los empleados, permitiendo fichar a la entrada y salida desde dispositivos móviles o a través de la aplicación de escritorio. Con Factorial las empresas obtendrán un sistema avanzado que incluye Códigos QR y geolocalización. La empresa puede habilitar el registro desde la oficina o fuera de ella recibiendo una notificación con la ubicación desde la que se ha fichado. Los trabajadores disponen de múltiples opciones. Por ejemplo, pueden acceder a la oficina escaneando el código QR, realizar el registro de la jornada con un simple clic en la aplicación, o rellenar los turnos de manera manual en la web.

Es una aplicación web ya diseñada solo para ese proceso, es privativo y no cumple con ninguna función que necesita el módulo de configuración de alientos para gestionar los horarios.

Aplicación Web para la Gestión y Control de Horarios en las Carreras de la Facultad de Administración, Finanzas e Informática

La aplicación web se basa en brindar un fácil control y automatización de los datos generando eficiencia en la realización de cada actividad de dicha aplicación, la cual almacena toda la información de horarios de la institución facilitando el uso del mismo. Tiene como propósito de controlar y automatizar procesos para la obtención de la información de horarios. Cumple con funcionalidades como: Almacenar información sobre los docentes (tiempo completo, medio tiempo y tiempo parcial), el cual obtenga datos como: Números de cédula, nombres y

apellidos, dirección domiciliaria, número de teléfono, correo electrónico, entre otros y registra información de materias de las carreras, requiriendo datos como: El código de la materia, el nombre de la materia, las horas presenciales y autónomas.

Es una aplicación web ya diseñada solo para ese proceso, no cumple con las funciones que necesita el módulo de configuración para gestionar los horarios.

1.3 Herramientas, tecnologías y metodología a utilizar

Lenguaje Unificado de Modelado (UML) 2.5.1 es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. También está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Leyva Jerez 2018).

Marco de trabajo:
Symfony 2.8

Symfony es un proyecto de *software* libre que publica pequeñas librerías PHP independientes llamadas "componentes". Combinando varios de esos componentes, también ha publicado un *framework* para desarrollar aplicaciones web. Debido al uso de los componentes, la arquitectura interna del *framework* está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un determinado proyecto. Symfony también es el *framework* que más ideas incorpora del resto de *frameworks*, incluso de aquellos que no están programados con PHP (Lubiński, Podborączyńska-Jodko et al. 2020) .

Lenguaje de Programación:

PHP 7.0

PHP es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. Es un acrónimo recurrente que significa "PHP *Hipertexto Pre-processor*". Su interpretación y ejecución se da en el servidor, en el cual se encuentra almacenado el *script*, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, generada por un *script* PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el *script* solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual se encarga de regresárselo al cliente. Además, es posible utilizar PHP para generar archivos PDF, Flash, así como imágenes en diferentes formatos (Arias 2017).

Permite la conexión a diferentes tipos de servidores de bases de datos tales como *MySQL*, *Postgres*, *Oracle*, ODBC, DB2, *Microsoft SQL Server*, *Firebird* y *SQLite*; lo cual permite la creación de Aplicaciones Web muy robustas. PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX (y de ese tipo, como *Linux*), *Windows* y MAC OS X, y puede interactuar con los servidores de Web más populares ya que existe en versión CGI y módulo para Apache (Bandera Rodríguez 2016) .

Bootstrap 4.0:

Es un entorno de trabajo definido con artefactos o módulos de *software* concreto, desarrollado por *Twitter*. Esta tecnología simplifica el proceso de creación de la interfaz de usuario combinando HTML, *JavaScript* y CSS (por sus siglas en inglés *Check Cascading Style Sheets*, Hojas de Estilo en Cascada). Además, se adapta a los distintos navegadores con numerosos componentes webs como: botones, etiquetas, alertas, entre muchos otros definidos en la web. *Bootstrap* fue programado para dar soporte a CSS 3 y HTML 5.

Bootstrap está diseñado para todos los niveles: diseñador, desarrollador y principiante. Este *framework* se utiliza para hacer más fácil y rápida su implementación. Bootstrap está definido por módulos que son reutilizables e independientes en la página web y es una gran comunidad abierta y seguida por millones de personas (Reyes Navarro 2016).

JavaScript

JavaScript, fue diseñado para ser un lenguaje de elaboración script que pudieran incrustarse en archivos HTML. No es complicado, y fue creado para darle más dinamismo a las páginas.

JavaScript es el lenguaje interpretado más utilizado, principalmente en la construcción de páginas Web, con una sintaxis muy semejante a Java y a C. Técnicamente, *JavaScript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Saks 2019).

Herramientas:

Visual Paradigm 8

*Visual Paradigm***,** es una herramienta de diseño, que hace uso del Lenguaje Unificado de Modelado (de sus siglas en inglés UML), este soporta todos los diagramas UML y el diagrama de entidad-relación. Produce documentación del sistema en varios formatos como PDF y HTML. Los desarrolladores pueden diseñar documentación del sistema con una plantilla de diseño. Los analistas de sistemas pueden estimar las consecuencias de los cambios con los diagramas de análisis de impacto, tales como la matriz y el diagrama de análisis (García Morales 2017).

Pgadmin 3

*Pgadmin 3***,** generalmente se usa como la herramienta de administración de interfaz gráfica de usuario (GUI) para *PostgreSQL* (Venkat 2017). Es uno de los administradores de bases de datos más utilizados, por sus características de administración de código abierto y porque posee una plataforma de desarrollo para *PostgreSQL*. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL sencillas hasta el desarrollo de bases de datos complejas. La interfaz gráfica de este administrador es compatible con todas las características de *PostgreSQL*. Es un software libre publicado bajo la licencia de PostgreSQL (Vidal Piña 2017).

PostgreSQL10

PostgreSQL.10, es un sistema que brinda una alta seguridad de la información manejada y posibilita la optimización de consultas; posee una arquitectura cliente-servidor. Este gestor, implementa transacciones, integridad referencial, vistas y multitud de funcionalidades; es, además, una aplicación de código abierto (Fernández Pérez 2018).

PostgreSQL es un potente sistema de base de datos relacionales de objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas (Ordóñez, Ríos et al. 2017).

PostgreSQL se ha ganado una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos,

conjunto de características sólidas, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer constantemente soluciones innovadoras y de alto rendimiento. *PostgreSQL se* ejecuta en todos los sistemas operativos principales, ha sido compatible con ACID desde 2001 (González Sierra 2017) .

Servidor web

Nginx

Nginx es un servidor web y proxy inverso: mecanismo desplegado en una red para proteger a los servidores HTTP de una intranet, realiza funciones de seguridad que protegen a los servidores internos de ataques de usuarios en internet. Gestiona las peticiones por medio de una arquitectura orientada a eventos donde las peticiones son aceptadas mediante *sockets* asíncronos y son procesadas en un único hilo de ejecución, esto a fin de reducir memoria y uso de CPU. La comunidad de *Nginx* no es tan amplia en comparación a la de Apache debido a su corto tiempo de vida en el mercado. Los tiempos de respuesta son menores con *Nginx*, Apache se torna cada vez más lento cuando las peticiones aumentan y además tiende a usar más ancho de banda para servir las mismas peticiones. Utilizado como servidor web para visualizar el proceso de desarrollo de la gestión de horarios en el Módulo de Configuración (Soni 2016).

Entorno integrado de desarrollo (IDE)

PhpStorm

PhpStorm es un IDE de programación desarrollado por *JetBrains.* Es uno de los entornos de programación más completos de la actualidad; permite editar código no sólo del lenguaje de programación PHP como lo indica su nombre. Actualmente es compatible con Sistemas Operativos *Windows, Linux* y *Mac OS X*. Algo que destaca en *PhpStorm* es la ejecución de código en la misma interfaz del IDE, así como también la interpretación y visualización inmediata de código PHP hasta en 5 de los navegadores web más populares. *PhpStorm* proporciona un editor para PHP, HTML y *JavaScript* con análisis de código sobre la marcha, prevención de errores y refactorizaciones automatizadas para código PHP y *JavaScript*. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación PHP

como lo indica su nombre sino también de los mencionados anteriormente. Entre sus características están las siguientes: (JETBRAIN, 2017).

1. Permite la gestión de proyectos fácilmente.
2. Proporciona un fácil autocompletado de código.
3. Sintaxis abreviada

Metodología

La metodología hace referencia al conjunto de procedimientos racionales utilizados para alcanzar un objetivo que requiera habilidades y conocimientos específicos. La metodología es una de las etapas específicas de un trabajo o proyecto que parte de una posición teórica y conlleva a una selección de técnicas concretas o métodos acerca del procedimiento para el cumplimiento de los objetivos. Es el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea (Saeed, Jhanjhi et al. 2019).

Las metodologías para el desarrollo del *software* imponen un proceso disciplinado con el fin de hacerlo más predecible y eficiente. Una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema. El desarrollo de la solución fue guiado por la metodología Proceso Unificado Ágil (AUP por sus siglas en inglés). 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. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (*test driven development* - TDD), Modelado, Gestión de Cambios, y Refactorización de Base de Datos para mejorar la productividad (Ambler 2005).

La misma ha sido designada UCI para la actividad productiva. AUP describe de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles. Ha sido adaptada a las necesidades de desarrollo de *software* en la universidad (Sánchez 2015).

En la variación efectuada se proponen 3 fases para guiar el proceso de desarrollo de los proyectos en la universidad: (Sánchez 2015)

- **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 cliente 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.

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP (Sánchez 2015).

Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Sánchez 2015).

El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el *software* desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes:

- Casos de Uso del Negocio (CUN).

- Descripción de Proceso de Negocio (DPN).
- Modelo Conceptual (MC)

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. 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.

Escenarios para la disciplina de requisitos:

- **Escenario No 1**: Proyectos que modelen el negocio con Casos de Uso del Negocio (CUN), solo pueden modelar el sistema con Casos de Uso del Sistema (CUS).

Figura 1: Escenario No.1 Fuente: [(Pressman 2009)]

- **Escenario No 2:** Proyectos que modelen el negocio con Modelo Conceptual (MC) solo pueden modelar el sistema con CUS.

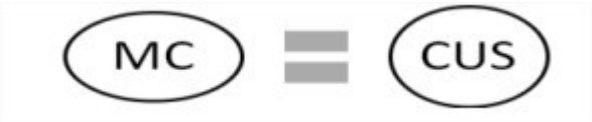


Figura 2: Escenario No.2 Fuente: [(Pressman 2009)]

- **Escenario No 3:** Proyectos que modelen el negocio con Descripción de Proceso de Negocio (DPN), solo pueden modelar el sistema con Descripción de Requisitos por Proceso DRP.

Figura 3: Escenario No.3 Fuente: [(Pressman 2009)]

- **Escenario No 4**: Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de usuario (HU)

Figura 4 Escenario No.4 . Fuente: [(Pressman 2009)]

El sistema SIGA emplea el escenario No 4 para modelar el sistema a partir de las HU. Este escenario es aplicado a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. Además, el cliente siempre acompañará al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, pues una HU no debe poseer demasiada información (Sánchez 2015).

Conclusiones parciales

Una vez finalizado este capítulo, se arribaron a las siguientes conclusiones:

- Se definieron los conceptos relacionados con el dominio del problema contribuyendo a una mayor comprensión del entorno del negocio.
- Después de realizar una valoración acerca de algunas soluciones que gestionan el proceso de configuración de horarios, se decidió no emplear ninguna de ellas, ya que no se ajustan a las necesidades de la DNA en cuanto a la generación de horarios por individual.
- Se definieron las diferentes herramientas y la metodología que se utilizarán en el desarrollo del proceso, las cuales guiarán dicho proceso de desarrollo.

CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

Para desarrollar un buen sistema informático es necesario priorizar aspectos en su diseño para crear una representación o modelo de software que proporcione detalles acerca de las estructuras de los datos, las interfaces, los componentes del software y la arquitectura que es esencial para el éxito de un proyecto. En este capítulo se obtendrán los requisitos del sistema, se definiría el estilo arquitectónico y se mostrarán diagramas como el de clases del diseño y el de despliegue.

2.1 Descripción de la propuesta solución

La propuesta de solución se basa en el desarrollo de los horarios por individual, para que cada complejo comedor funcione conforme a su dinamismo. El sistema a desarrollar, facilitará en buena medida el trabajo del personal del comedor, contará con funcionalidades que posibiliten la agilidad y organización de este proceso.

Con el objetivo de desarrollar el módulo de configuración para la gestión de horarios del SIGA, se requiere la implementación de la gestión de los mismos. Se incorporan nuevos servicios entre los que se encuentran:

- ✓ Posibilitar que cada complejo comedor y pantrys tengan su horario de apertura y cierre propio.
- ✓ Permitirá un mejor control de horarios.
- ✓ Permitirá que el sistema responda rápidamente en caso de que ocurra algún cambio.
- ✓ Que sea un sistema automatizado.

2.2 Especificación de los requisitos de software

Los requisitos son la base para un desarrollo exitoso, así como para una plena conformidad con el entregable final. La especificación de los requisitos de software establece una base sólida para su diseño e implementación. Estos definen las necesidades, describen los escenarios de uso, delinean las funciones y características e identifican las restricciones del proyecto (Westfall 2005). Para llevar a cabo el proceso de captura de requisitos se aplicó la técnica que a continuación se expone:

Entrevista: se realizaron entrevistas a los trabajadores Lester González, Yoandris Pacheco entre otros, sustentado en una serie de preguntas, con el objetivo de obtener la mayor cantidad posible de información y comprender los objetivos generales de la solución buscada.
Durante la disciplina de captura de requisitos se identificaron 6 requisitos funcionales (RF) con lo que debe constar el módulo para satisfacer las necesidades requeridas por el cliente.

2.2.1 Requisitos funcionales

Los requisitos funcionales (RF) son aquellos directamente relacionados con las funciones y las reacciones que el sistema debe proporcionar. Son asignados directamente a elementos o características del sistema de software, capacidades o condiciones que el sistema debe cumplir, de cómo debería reaccionar el mismo a entradas particulares y de cómo debería comportarse en situaciones específicas. Además, surgen de la razón fundamental de la existencia de un producto. A continuación, se muestran los requisitos funcionales:

Requisitos funcionales del Módulo Configuración.

Funcionalidad gestión de horarios

- RF1 Registrar horarios
- RF2 Listar horarios
- RF3 Modificar horarios
- RF4 Eliminar horarios
- RF5 Activar y desactivar horarios
- RF6 Mostrar Alerta

2.2.2 Requisitos no funcionales (RnF)

El sistema debe ser eficiente en el campo de acción en que se utiliza, no solo es importante especificar las funcionalidades que debe desempeñar, también se tienen que definir todas las cualidades que lo hacen más atractivo al cliente, usable, rápido y confiable. Los requisitos no

funcionales se reconocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el *software* (Martínez Cabrera 2019).

La siguiente tabla muestra los requisitos no funcionales identificados en el desarrollo del Módulo de Configuración para la gestión de horarios:

Usabilidad:

RNF7: Facilidad de empleo para usuarios sin experiencia. El módulo debe tener una interfaz de fácil aprendizaje para que el administrador se familiarice rápidamente.

RNF8: Se documenta el módulo con un manual de usuario con el objetivo de explicar su uso.

RNF9: El módulo permitirá el fácil acceso a las funcionalidades del mismo, brindando accesos directos a dichas funcionalidades mediante íconos flotantes e internos del sistema.

Fiabilidad:

RNF9: Disponibilidad, el módulo contribuye a la muestra del horario, por lo cual debe estar disponible para el administrador en el momento en que lo necesite.

Seguridad:

RNF10: El sistema debe garantizar la protección de los datos almacenados. Para ello se establecerán diferentes niveles de acceso, permitiendo que cada usuario acceda solamente a las funcionalidades que necesite. Esto se logra mediante las listas de control de acceso definidas en el fichero de configuración de *Symfony security.yml.*

Mantenibilidad:

RNF11: Se utiliza para la construcción del módulo los lenguajes de programación php, HTML, y las herramientas que se utilizan son de distribución bajo licencias libres, para favorecer el desarrollo de nuevos módulos o subsistemas.

Hardware y software requerido para utilizar la aplicación:

RNF12: El *hardware* donde se instalará el sistema debe poseer al menos una Interfaz de red cuya velocidad de transferencia iguale o supere los 100 Mbps. El Servidor donde se instale el Sistema debe tener como mínimo un Microprocesador Intel-Core i3 a 3.0 Mhz3.

RNF13: El sistema debe integrarse con el Gestor de base de datos *PostgreSQL* 9.5, usando como servidor web *Nginx*.

Interfaz

RNF14: Para acceder al Sistema debe usarse una versión del navegador Mozilla/Firefox v 35.0 o superior. No se garantiza la correcta visualización en otros navegadores. Además, que para hacer uso de todas las funcionalidades el navegador debe tener habilitado el soporte para *Java Script*.

2.3 Historia de Usuario

La metodología AUP-UCI, en su escenario 4 para la disciplina requisitos, genera las Historias de Usuario (HU), estas se utilizan para especificar los requisitos de software en la aplicación. Las HU se descomponen en tareas de programación y describen las características que el sistema debe cumplir, están escritas en un formato legible por el cliente, sin necesidad de sintaxis técnicas (Raharjana, Siahaan et al. 2021).

Se muestra a continuación en las siguientes tablas varias historias de usuarios las cuales fueron generadas a partir de los requisitos funcionales y las mismas se estructuran de la siguiente forma:

Número: A cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Nombre: Nombre descriptivo de la HU.

Prioridad: Grado de prioridad que le asigna el cliente a la HU en dependencia del valor en el negocio. Los valores que puede tomar son (Alta, Media o Baja).

Iteración Asignada: Número de la iteración en la cual será implementada la HU.

Programador: Nombre del programador encargado de desarrollar la HU.

Tiempo estimado: Esfuerzo estimado por el equipo de desarrollo para darle cumplimiento a la HU.

Tiempo real: Plazo real que el equipo de desarrollo tiene para dar cumplimiento a la HU.

Tabla 1: Historia de Usuario Adicionar horarios*. Fuente: [Elaboración Propia]*

Historia de Usuario	
Número:1	**Nombre de Usuario:** Registrar horarios
Programador: Zoe Ramírez López	**Iteración asignada:**1

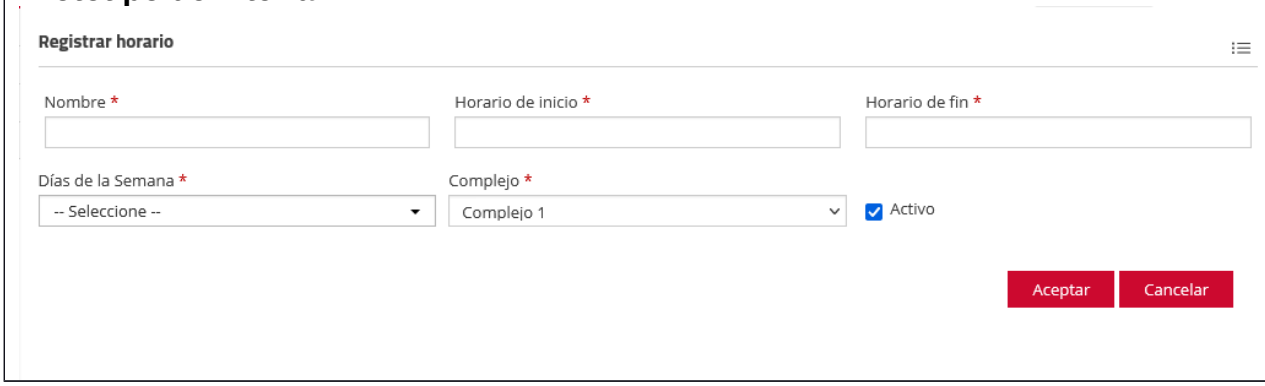
Prioridad: Alta	**Tiempo estimado:** 3h
Riesgo de Desarrollo: Poca experiencia en el lenguaje de programación.	**Tiempo real :**2h
Descripción: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá insertar un nuevo horario llenando los campos correspondientes.	
Prototipo de interfaz : 	

Tabla 2: Historia de Usuario Listar Horarios. Fuente: [Elaboración Propia]

Historia de Usuario	
Número: 2	**Nombre HU:** Listar Horarios
Programador: Zoe Ramírez López	**Iteración asignada:**1
Prioridad : Alta	**Tiempo estimado :**3h
Riesgo de Desarrollo: Poca experiencia en el lenguaje de programación	**Tiempo real:** 2h
Descripción: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá visualizar el listado de horarios que han sido registrados en el sistema.	

Prototipo de interfaz :

Tabla 3: Historia de Usuario Modificar Horarios. Fuente: [Elaboración Propia]

Historia de Usuario	
Número:3	**Nombre de HU:** Modificar Horarios
Programador: Zoe Ramírez López	**Iteración asignada:** 1h
Prioridad: Alta	**Tiempo estimado :** 3h
Riesgo de Desarrollo: Poca experiencia en el lenguaje de programación	**Tiempo real :** 2h
Descripción: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá modificar un horario que ya existe con el objetivo de cambiar sus datos sin crear uno nuevo.	
Prototipo de interfaz :	

Tabla 4: Historia de Usuario Eliminar Horarios. Fuente: [Elaboración Propia]

Historia de Usuario	
Número:4	**Nombre de HU:** Eliminar Horarios
Programador: Zoe Ramírez López	**Iteración asignada:** 1h
Prioridad: Alta	**Tiempo estimado:** 3h
Riesgo de Desarrollo: Poca experiencia en el lenguaje de programación	**Tiempo real :**2h

Descripción: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá eliminar los horarios.

Tabla 5: Historia de Usuario Activar y Desactivar. Fuente: [Elaboración Propia]

Historia de Usuario	
Número:5	**Nombre de HU:** Activar y Desactivar Horarios
Programador: Zoe Ramírez López	**Iteración asignada:**1h
Prioridad : Alta	**Tiempo estimado :**1h
Riesgo de Desarrollo: Poca experiencia en el lenguaje de programación	**Tiempo real:** 2h
Descripción: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá visualizar el estado de los horarios para activar y desactivar que han sido registrados en el sistema.	

Tabla 6: Historia de Usuario Mostar Alerta. Fuente: [Elaboración Propia]

Historia de Usuario	
Número:6	**Nombre de HU:** Mostar Alerta
Programador: Zoe Ramírez López	**Iteración asignada:**1h
Prioridad : Alta	**Tiempo estimado :**1h
Riesgo de Desarrollo: Poca experiencia en el lenguaje de programación	**Tiempo real:** 2h
Descripción: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá visualizar el listado de horarios	

que han sido registrados en el sistema.

2.4 Patrón Arquitectónico

Un patrón arquitectónico se puede considerar como una descripción abstracta estilizada de buena práctica, que se ensayó y puso a prueba en diferentes sistemas y entornos. Debe describir una organización de sistema que ha tenido éxito en sistemas previos. Debe incluir información sobre cuándo es y cuándo no es adecuado usar dicho patrón, así como sobre las fortalezas y debilidades del patrón. El patrón arquitectónico utilizado para el desarrollo del Módulo de Configuración es el Modelo-Vista-Controlador (MVC).

El MVC es un patrón de arquitectura de software que separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí. El componente Modelo maneja los datos del sistema y las operaciones asociadas a esos datos. El componente Vista define y gestiona cómo se presentan los datos al usuario. El Controlador dirige la interacción del usuario (por ejemplo, teclas oprimidas, clics del mouse, etcétera) y pasa estas interacciones a Vista y Modelo (Deacon 2009).

El modelo: Debe representar la parte de la aplicación que implementa la lógica de negocio y el acceso a datos. Esto significa que debe ser responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos. En el desarrollo del Módulo de Configuración para hacer uso del modelo se explica a continuación:

Lógica 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* que se encuentran en *Src/systems/Alimentación/Configuración/DistribucionBundle/Entity*. Para abstraer la lógica de negocio de los controladores se crean repositorios de entidades perso-

nalizados que permiten llevar el manejo de la base de datos a una clase aislada del controlador y se pueden encontrar en la siguiente dirección: *Src/systems/Alimentación/Configuración/ConfiguracionBundle/Repository*.

Acceso a datos: establece la conexión con la base de datos, además, 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 SQL que se encuentra en *app/config/parameters.yml*.

La vista: Debe hacer una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente. En el desarrollo del Módulo de Configuración para la presentación de los datos se utiliza *twig* como motor de plantillas, 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 */Resouces/views* dentro de cada *bundle* de la aplicación *en Src/systems/Alimentación/Configuración/DistribucionBundle/Resouces/Views*.

El controlador: Gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista. Los controladores pueden ser vistos como administradores, espera peticiones de los clientes, comprueba su validez de acuerdo a las normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente. Finalmente delega este proceso de presentación a la capa de la Vista. Para el desarrollo del Módulo de Configuración este patrón se pone de manifiesto en todas las clases *Controller*, que controlarán el flujo de información que se recibe y se envía hacia la vista y estarán ubicadas en *Src/systems/Alimentación/Configuración/DistribucionBundle/Controller* junto a las demás clases que se localizan en el directorio */Controller* de cada *buldle de la aplicación*.

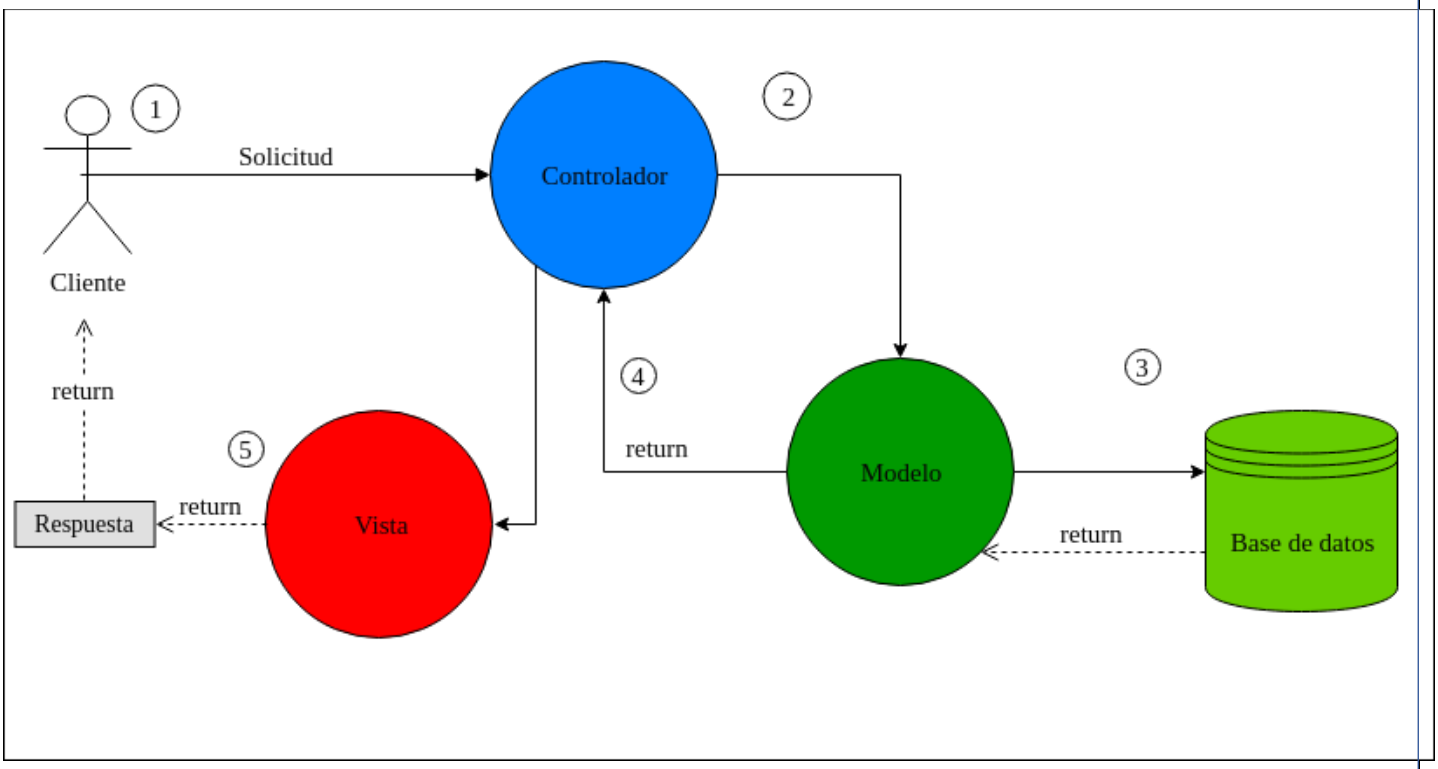


Figura 5: Modelo Vista Controlador Fuente: [(Deacon 2009)]

2.5 *Patrones de Diseño*

Un patrón de diseño se caracteriza como una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución. Para el diseño de software, el contexto permite entender el ambiente del problema y la solución apropiada. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la interpretación del contexto del problema y cómo podría aplicarse con eficacia la solución (Cooper 2000) . Durante la implementación de la propuesta de solución se utilizaron los siguientes patrones de diseño:

2.5.1 Patrones Generales de Asignación de Responsabilidades

Los patrones GRASP (*General Responsabilty Assignment Software Patterns*, “Patrones de *Software* para la Asignación General de Responsabilidad”) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Se pueden destacar 5 patrones principales que son: Bajo Acoplamiento, Alta Cohesión, Experto, Creador y Controlador.

Experto: el objetivo es asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir con la funcionalidad. El patrón Experto posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes (Tabares 2010) .En el Módulo de Configuración este patrón se pone en práctica en la clase *HorarioController* que es la encargada de proporcionar información necesaria para la gestión del menú.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. El uso de este patrón se refleja en las clases controladoras pertenecientes al sistema. En el Módulo de Configuración este patrón se pone en práctica en todas las clases controladoras *HorarioController*.

Creador: este patrón como su nombre lo indica es el que crea, ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos. Es donde se asigna la responsabilidad de que una clase B cree un objeto de la clase A. En el Módulo Configuración este patrón se pone de manifiesto en todos los servicios definidos en los ficheros *service.yml.*

2.5.2 Patrones Gang of Four (GOF)

Los patrones "Banda de los Cuatro" (*Gang-of-Four*) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas.

Mediador (Mediator): define un objeto que coordine la comunicación entre objetos de distintas clases. Se refleja en las librerías que funcionan como mediadoras entre las clases controladoras y las modelos de acceso a datos. Este patrón se refleja en las clases *HorarioController.php*.

2.6 Diagrama de Clase del Diseño

Un Diagrama de Clase del Diseño muestra la especificación de las clases de una aplicación, sus asociaciones, atributos y métodos, interfaces, navegabilidad y dependencias. El diagrama de clases recoge las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos. Con el fin de facilitar la comprensión del diagrama, se pueden incluir paquetes como elementos del mismo, donde cada uno de ellos agrupa un conjunto de clases (Aliaga, Ruiz et al. 2019). A continuación, en la Figura 6, se expone un diagrama de clase del diseño el cual fue generado para la presente investigación.

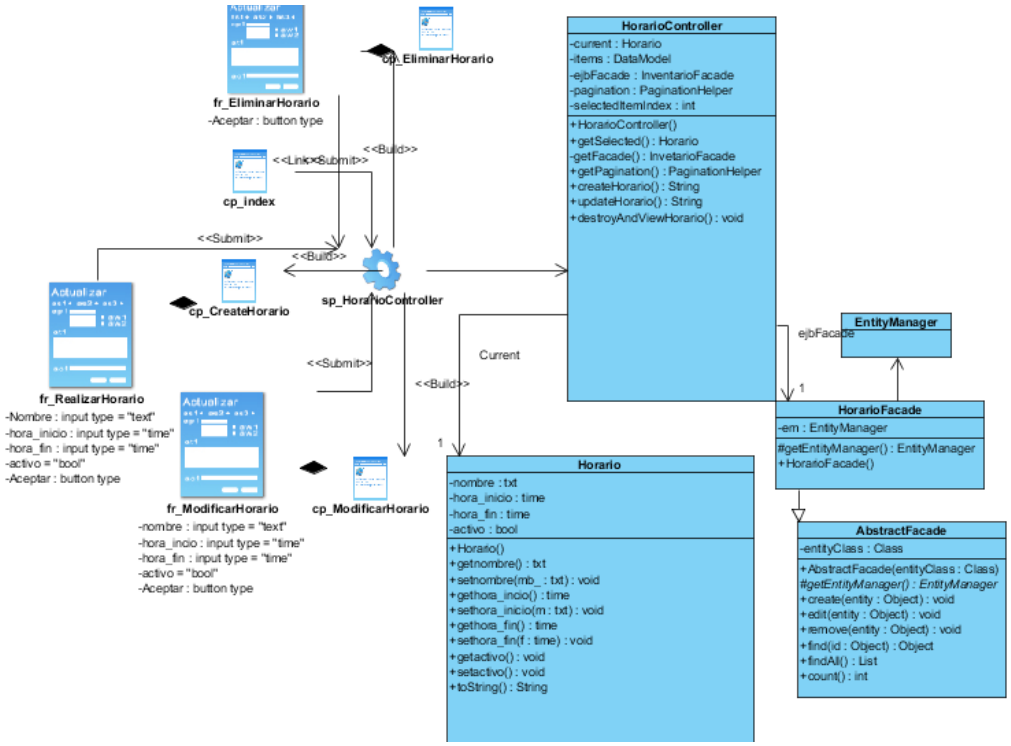


Figura 6: Diagrama de clase del Diseño. *Fuente: [Elaboración Propia]*

2.7 Modelo de datos

Modelo: conjunto de conceptos que permite construir una representación organizacional de una institución.

Universo de discurso: visión del mundo real del diseñador de la Base de Datos. Su definición, constituye el primer paso del diseño para poder conseguir los objetivos del diseñador.

Modelo de Datos: conjunto de conceptos, reglas y convenciones que permiten describir, a distintos niveles de abstracción, los datos del Universo de Discurso o la estructura de una base de datos, la cual es denominada esquema, o, con otras palabras, que permiten construir una representación organizada de un sistema real. Por lo tanto, se considera un modelo de datos como una herramienta que facilita la interpretación del Universo de Discurso y la representación de un sistema de información. Los mismos comprenden aspectos relacionados

con estructuras y tipos de datos, operaciones y restricciones (Capacho and Nieto Bernal 2017).

Descripción del modelado de datos

Sc_distribucion. tb_nhorario: Registra la información de los horarios del sistema. Se almacena en la base de datos los atributos que lo identifican id horario, nombre_horario, hora_inicio, hora_fin, activo, structure.

Sc_distribucion. tb_nevento: Contiene la información de los eventos. Se almacena en la base de datos los atributos que lo identifican id_ evento, nombre evento, activo, descripción_evento, fecha_ registro, id_ clasificación_evento, accesos, orden, evento_ padre, id_ horario, icono, color.

Sc_distribucion. tb_revento_horario: Contiene la información de los eventos con sus respectivos horarios. Se almacena en la base de datos los atributos que lo identifican id_ horario, id_ evento, id_ evento_horario.

Sc_distribucion. tb_nhorario_dia_semana: Contiene la información de los horarios con sus respectivos días de la semana. Se almacena en la base de datos los atributos que lo identifican id_horario, id_dia_semana, id_horario_semana.

Sc_distribucion. tb_ndia_semana: Contiene como información de días de la semana. Se almacena en la base de datos los atributos que lo identifican id_dia_semana, id_nombre_semna, abreviatura_dia_semana.

Sc_distribucion. tb_structure: Contiene la información de la estructura de la DGA de la UCI Se almacena en la base de datos los atributos que lo identifican category_id estructura _parent_ id, name, initials, active, created_ at, updated_at, version, branch, description, capacidad, id_ sub_ director, id _tecnico_ general, centro_ costo, id_ almacen_ pertenece, id_ tecnico_ atm, id _especialista_complejo.

A continuación, se muestra en la Figura 7, un fragmento del modelo de datos del Módulo Configuración, el cual ofrece una descripción abstracta sobre la representación de los datos

en un Sistema Gestor de Base de Datos (SGBD). Los componentes que lo conforman, son entidades existentes en el módulo. Este modelo contiene a su vez, características propias de estos objetos y la forma en que se relacionan entre sí.

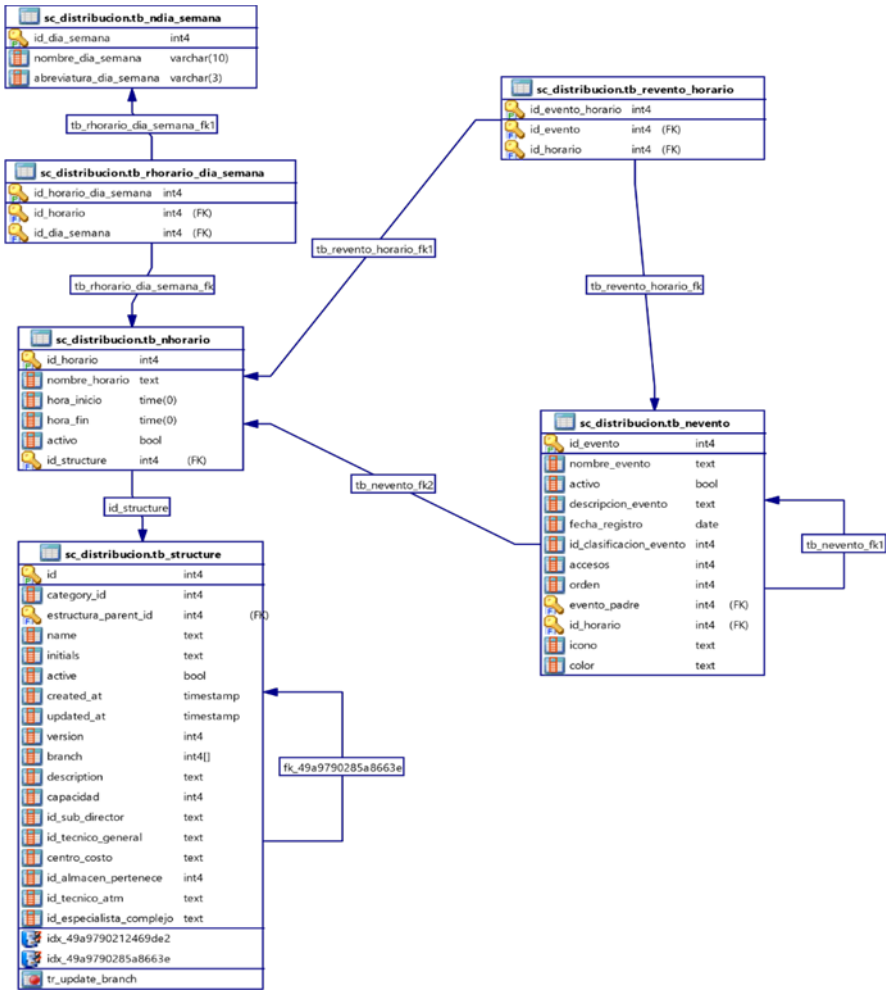


Figura 7:Modelado de Base de Datos

2.8 Diagrama de Componente

Representan las relaciones entre los componentes individuales del sistema mediante una vista de diseño estática. Pueden ilustrar aspectos de modelado lógico y físico. Los componentes se consideran unidades autónomas encapsuladas dentro de un sistema o subsistema que proporcionan una o más interfaces. Estos diagramas son generalmente dirigidos al per-

sonal de aplicación de un sistema, además presenta una comprensión temprana del sistema global que se está construyendo (Pérez Bejerano 2019).

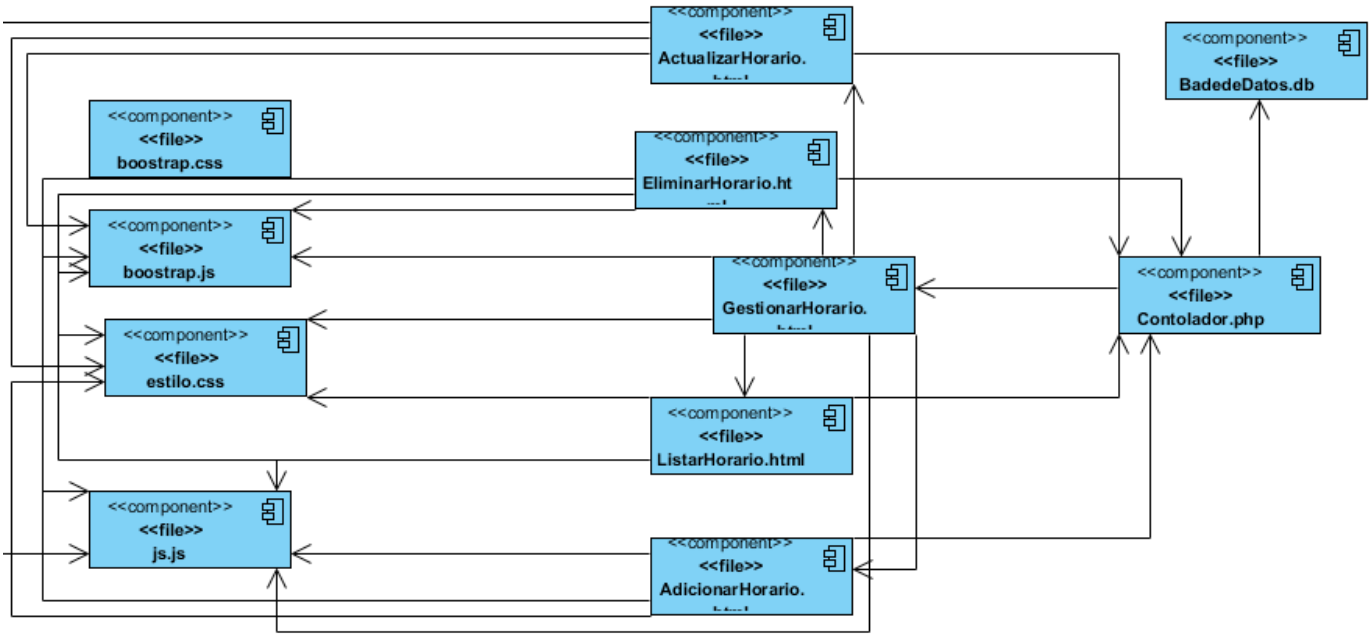


Figura 8: Diagrama de Componentes. Fuente: [Elaboración Propia]

Conclusiones del capítulo

En este capítulo se evidencian las principales características y elementos significativos de la propuesta de solución, así como los diferentes requisitos funcionales y no funcionales, por lo que se puede constatar que:

- ✓ La definición de los requisitos funcionales y no funcionales obtenidos a partir del proceso de identificación de los requisitos, garantizó que la solución responda a las necesidades del cliente, sirviendo de guía para el desarrollo de las funcionalidades del sistema de gestión tecnológica.
- ✓ Los artefactos generados constituyeron una guía fundamental para el desarrollo del sistema de gestión tecnológica.
- ✓ La arquitectura MVC seleccionada permitió definir la estructura del software y su interrelación entre los diferentes componentes

CAPÍTULO III: Validación y Prueba

En el presente capítulo se propone explicar a partir de los resultados del diseño, la implementación del Módulo Configuración para el Sistema de Gestión de Alimentación. Se definen los estándares de codificación y las pruebas de *software* como elemento esencial que garantiza la calidad del módulo y revisión final del cumplimiento de los requisitos, especificaciones de diseño y codificación.

3.1 Estándares de codificación

Los estándares de codificación son reglas de codificación que permiten tener una programación homogénea, comprendiendo todos los aspectos de la generalización del código, pues la aplicación debe de estar implementada como si un único programador escribiera el código de una sola vez. Son un conjunto de convenciones para lograr uniformidad en el código fuente de un software. Permiten una mayor comprensión, modificación, calidad y mantenibilidad del código generado. La usabilidad de estos permite conservar el código fuente entendible y fácil de mantener, además de mejorar la forma en la que se programa (Zandstra 2021) .

Indentación, llaves de apertura y cierre y tamaño de líneas

El código debe usar cuatro espacios para la indentación en vez de usar el tabulado, esto minimiza problemas con otras herramientas de desarrollo.

Las líneas podrían tener 80 caracteres o menos evitando tener más de 120 caracteres.

Los paréntesis en las estructuras de control, no deben usar espacios antes ni después. Se añade un solo espacio después de cada limitador de coma y alrededor de los operadores (==, &&, …).

Las aperturas de llaves en las estructuras de control se colocan en la misma línea.

El cierre de llaves para clases, funciones y estructuras de control deben estar en la próxima línea después del bloque de código.

```
    public function findHorarioByFilters($filters = array(), $order = []) {
        $qb = $this->createQueryBuilder('qb');
        $qb->select(
                'qb.nombre,'
                . 'qb.id,'
                . 'to_char(qb.horaInicio,' . "'HH24:MI'" . ') as horaInicio, '
                . 'to_char(qb.horaFin,' . "'HH24:MI'" . ') as horaFin, '
                . 'qb.activo'
        );
        if (isset($filters['fsearchtext']) && !empty($filters['fsearchtext'])) {
            $exp = $qb->expr()->like("format_ltt_string(qb.nombre)", "format_ltt_string('%" . $filters['fsearchtext'] .
            $qb->andWhere($exp);
            unset($filters['fsearchtext']);
        }
        if ([] != $order) {
            unset($order['paginate_params']);

            foreach ($order as $key => $value) {
                $qb->addOrderBy($key, $value);
            }
        }

        $qb = $this->addFilters($qb, (array) $filters);
        $qb->addOrderBy('qb.id', 'ASC');
        return $qb->getQuery()->getResult();
    }
```

Figura 9: *Llaves de apertura y cierre y tamaño de líneas*.

Convención de nomenclatura

Variables: Se rigen por la nomenclatura *camelCase* para declarar. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Se muestra en el método un ejemplo donde se emplean las variables:

```
ublic function indexAction(Request $request)

    $em = $this->getDoctrine()->getManager();
    $fecha_actual = date_create();
    $Object = new DateTime();
    $numDiaSemanaSistema = (int)date_create()->format('N') + 1;
    $esCajero=false;
    $ipPC=$this->container->get('request')->getClientIp();
```

Figura 10: *Convención de nomenclatura: variable camelCase*

Clases: Se rigen por la nomenclatura *StudlyCaps*. El patrón que sigue para declarar las clases es que siempre comienzan con mayúscula y en caso de nombre compuesto cada palabra comienza en mayúscula, sin espacios o guion bajo, como se muestra a continuación:

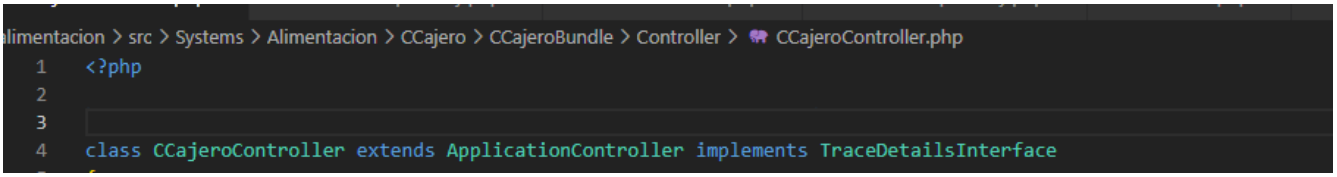


Figura 11: Convención de nomenclatura. Clase nombre compuesto. StudlyCaps

Funciones: Se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa como se muestra a continuación:

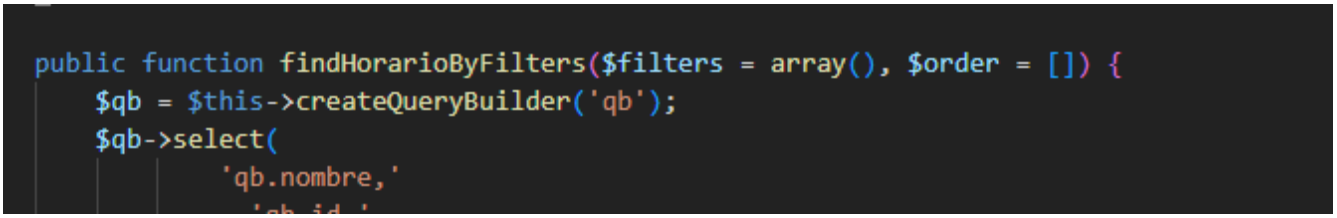


Figura 12: Convención de nomenclatura: función: camelCase

Estructuras de control

Las estructuras de control incluyen *if*, *for*, *for each*, *while*, *switch*, entre estas estructuras y los paréntesis que encierran la condición no existe espacio. Es recomendable utilizar en cualquier caso llaves de apertura y cierre al comienzo de una nueva línea, como se muestra a continuación:

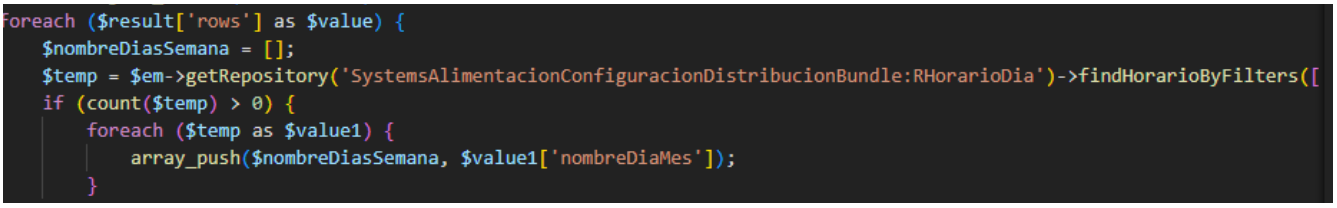


Figura 13: Estructura de control

3.2 Diagrama de despliegue

El diagrama de despliegue se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el hardware y el software que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno. En la

siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta (Irmayani and Susyatih 2017):

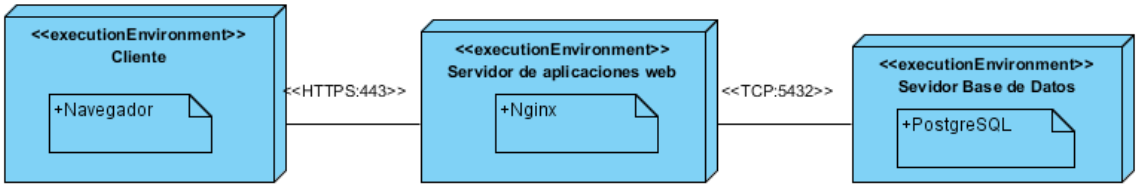


Figura 14: *Diagrama de despliegue. Fuente: [Elaboración Propia]*

El diagrama de despliegue representado muestra la siguiente distribución:

Cliente: dispositivo cliente capaz de conectarse al servidor de aplicaciones mediante el protocolo de comunicaciones HTTPS.

Servidor de aplicaciones web: computadora en que se encuentra el servidor web *NGinx*, este será el lugar en que se gestione todo el contenido de la aplicación. El mismo establecerá comunicación con los ordenadores clientes mediante protocolo HTTPS y con el servidor de base de datos por medio del protocolo TCP.

Servidor de base de datos: ordenador en que se encuentra el gestor de base de datos *PostgreSQL* capaz de mantener persistente la información generada y a utilizar.

HTTPS: protocolo de transferencia de hipertexto seguro, por sus siglas en inglés, *Hypertext Transfer Secure Protocol* (HTTPS), es un protocolo de red basado en HTTP por lo que está orientado a transacciones sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor (Felt, Barnes et al. 2017).

3.3 Criterios para validar los requisitos

Para validar los requisitos del sistema según Pressman, se pueden chequear mediante un cuestionario guiado por un conjunto de interrogantes, con el objetivo de descubrir la mayor cantidad de errores posibles (Pressman 2009).

A continuación, se muestran las preguntas utilizadas.

Interrogantes para la validación de requisitos:

- ¿Es comprensible, puede ser entendido por los participantes?
- ¿Es Consistente?
- ¿Es ambiguo?
- ¿Es Verificable?
- ¿El requisito incumple alguna restricción definida?
- ¿Está el requisito asociado con los rendimientos del sistema o con su comporta miento?
- ¿El requisito está implícitamente definido?

Resultado de aplicar los criterios de validación

Luego de aplicar el conjunto de interrogantes para validar los requisitos definidos para el desarrollo del sistema, se obtuvo el 100 % de aprobación por parte del cliente.

3.3.1Técnicas de validación de requisitos

Con el objetivo de obtener una mayor calidad y demostrar que los requisitos definidos realmente describen el sistema que el cliente necesita; se utilizó la técnica siguiente:

Prototipado de interfaz: permite al cliente entender fácilmente la propuesta de solución, al brindar la representación aproximada de la interfaz de usuario que tendrá el sistema. Existen dos tipos principales de prototipo de interfaz:

- ✓ **Desechables:** se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en *software*.

- ✓ **Evolutivos:** una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final. El tipo utilizado finalmente para la propuesta de solución fue Evolutivos pues de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del sistema.

Resultado de aplicar las técnicas de validación

- ✓ Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como la falta de concordancia entre la complejidad de la especificación y la registrada en el documento de Evaluación de requisitos. La técnica utilizada para la propuesta fue prototipado de interfaz evolutivo pues de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del sistema.

3.4 Pruebas

La fase de validación es el proceso de evaluar la calidad del software, con el objetivo de detectar posibles errores y corregirlos enfocándose en la lógica interna y los requerimientos especificados (Suárez, Rabaz et al. 2017) . A continuación, se muestra la estrategia de prueba diseñada para aplicar en la solución desarrollada:

Tabla 7: Estrategia de Prueba *Fuente: [Elaboración Propia]*

Prueba	**Método**	**Técnica**
Prueba Funcionales	Caja negra	Partición de equivalencia
Prueba Unitarias	Caja blanca	Camino básico

3.4.1 Pruebas Funcionales

Este tipo de prueba se basa en las funcionalidades de un sistema que se describen en la especificación de requisitos. También pueden no estar documentadas, pero se requiere un nivel de experiencia más elevado. Estas pruebas son especificadas por el cliente y se centran en las características y funcionalidades generales del sistema que son visibles y revisables por parte del cliente (Blanco Pérez, Martínez López et al. 2016).

Las características de funcionalidad según las establece la ISO 25010 son idoneidad, exactitud, interoperabilidad y seguridad. En la ISO 25010 indican que “la funcionalidad representa la capacidad del producto de *software* para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones específicas”. La funcionalidad a su vez, son divididas en las siguientes características:

- ✓ Completitud funcional: el grado en el que las funcionalidades cubren todas las tareas y objetivos del usuario especificados.
- ✓ Corrección funcional: capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- ✓ Pertenencia funcional: capacidad del producto de *software* para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados (Sanchez Peño 2015).

A continuación, se muestra un ejemplo de la prueba funcional realizada a una historia de usuario que se mostraron en el capítulo anterior:

Tabla 8: *Caso de Prueba Funcional. Fuente: [Elaboración Propia]*

Caso de Prueba Funcional	
Código Caso de Prueba: **1**HU_1-1	**Nombre Historia de Usuario:** Registrar Horario
Nombre de la persona que realiza la prueba: Zoe Ramírez López	
Descripción de la Prueba: El administrador o cualquier usuario que tenga asignado el rol que le permitirá tener acceso a esta funcionalidad podrá adicionar un horario llenando los campos correspondientes.	
Condiciones de Ejecución: Debe existir un código escrito en la sección correspondiente del sistema.	
Entrada / Pasos de ejecución: Para evaluar el requisito es necesario: - El usuario una vez autenticado en el Sistema de Gestión de Alimentación selecciona el módulo de Configuración y luego el módulo Distribución. El sistema muestra las opciones de menú. El usuario selecciona en la agrupación funcional "Horarios" y luego el icono flotante para re-	

gistrar.
Resultado Esperado: El horario fue adicionado correctamente.
Evaluación de la Prueba: Satisfactoria.

Los **métodos de caja negra**, también denominados pruebas de comportamiento, se centran en los requisitos funcionales del *software*. Permite al ingeniero del *software* obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a base de datos externas, errores de rendimiento y errores de inicialización y de terminación. La **técnica de partición de equivalencia** consiste en ejecutar el flujo básico de las funcionalidades utilizando datos válidos e inválidos, donde se generarán los artefactos “Diseño de casos de pruebas” (Aliaga, Ruiz et al. 2019).

Tabla 9: Registrar Horarios. *Fuente: [Elaboración Propia]*

Escenario	**Descripción**	**V1 Nombre**	**V2 Hora de Inicio y Fin**	**V3 Día de la semana**	**V4 Complejos**	**Respuesta Esperada**
Insertar datos correctamente	Mediante este escenario se inserta en el sistema una nuevo	V Horario Almuerzo	V 11:30am 1:50 pm	V Lunes	V Complejo 1	El sistema crea el nuevo horario y muestra el mensaje: “La acción ha sido realizada satisfactoriamente ”.

	horario.					
Insertar datos incorrectos	Mediante este escenario se introducen datos incorrectos.	I a	6:00pm 7:50 pm	Lunes Martes	Complejo 2	El sistema muestra en rojo el mensaje en rojo : Por favor no escriba menos de dos caracteres
Insertar datos incompletos	Mediante este escenario no se introducen todos los datos para registrar un Horario.	I	I	Miércoles		El sistema muestra en rojo el mensaje: Campo requerido.
Cancelar operación.	Se cancela la creación del nuevo Horario.	V Horario de Desayuno	V 7:30am 8:50 am	V		El sistema muestra el mensaje de confirmación: ¿Está seguro que desea Cancelar la operación? Si el usuario presiona el botón Si el sistema retorna a la página que le dio inicio sin guardar las últimas operaciones Si el usuario presiona el botón No se mantiene en la misma vista, mostrando las últimas operaciones realizadas.

Tabla 10: *Descripción de las variables Adicionar Horario. Fuente: [Elaboración Propia]*

NO	**Nombre del Campo**	**Valor Nulo**	**Descripción**
1	Nombre	No	Es un campo que permite al administrador escribir el nombre del evento este debe escribirse correctamente. Acepta letras, números acepta espacio entre letras.
2	Hora de Inicio y Fin	No	Es un campo que permite al administrador escoger la hora para el evento de inicio y de fin.
3	Día de la semana	No	Es un campo que permite al administrador escoger el día o los días de la semana que así desee.
4	Complejos	No	Es un campo que permite al administrador escoger el complejo comedor .

En total, se planificaron 2 iteraciones de prueba. La figura 15 aporta la información del resultado, total de no conformidades encontradas y las que se resolvieron por cada iteración. Para un total de 6 requisitos funcionales se detectaron 4 no conformidades en la primera iteración y las 4 se solucionaron satisfactoriamente, en la segunda iteración se redujo las no conformidades a 0, dando como resultado que fueran resueltas todas las no conformidades que se encontraron en la aplicación de la prueba.

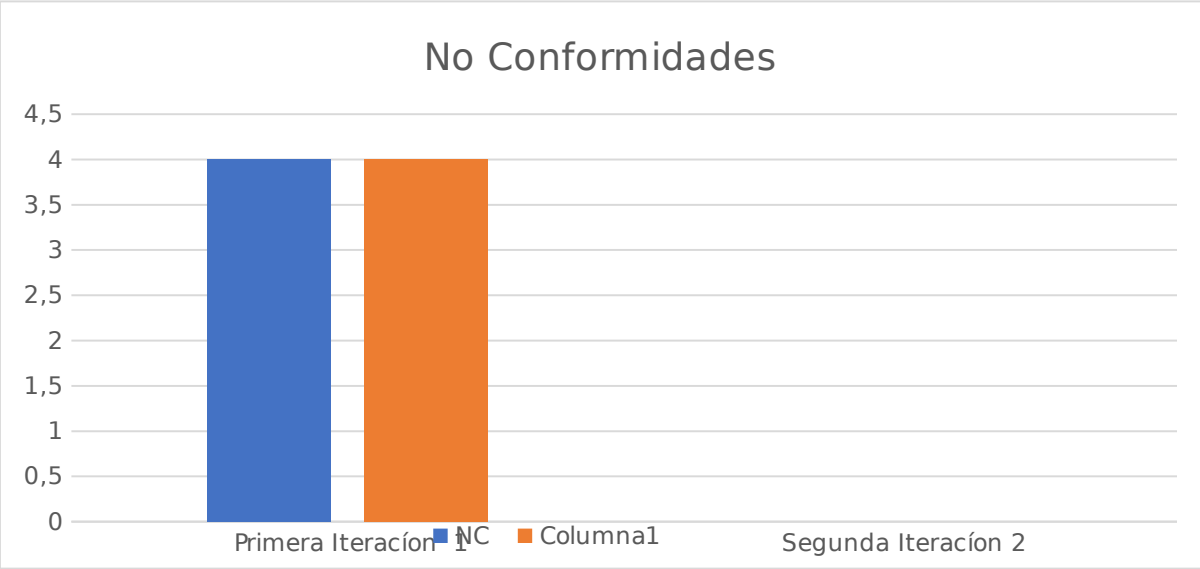
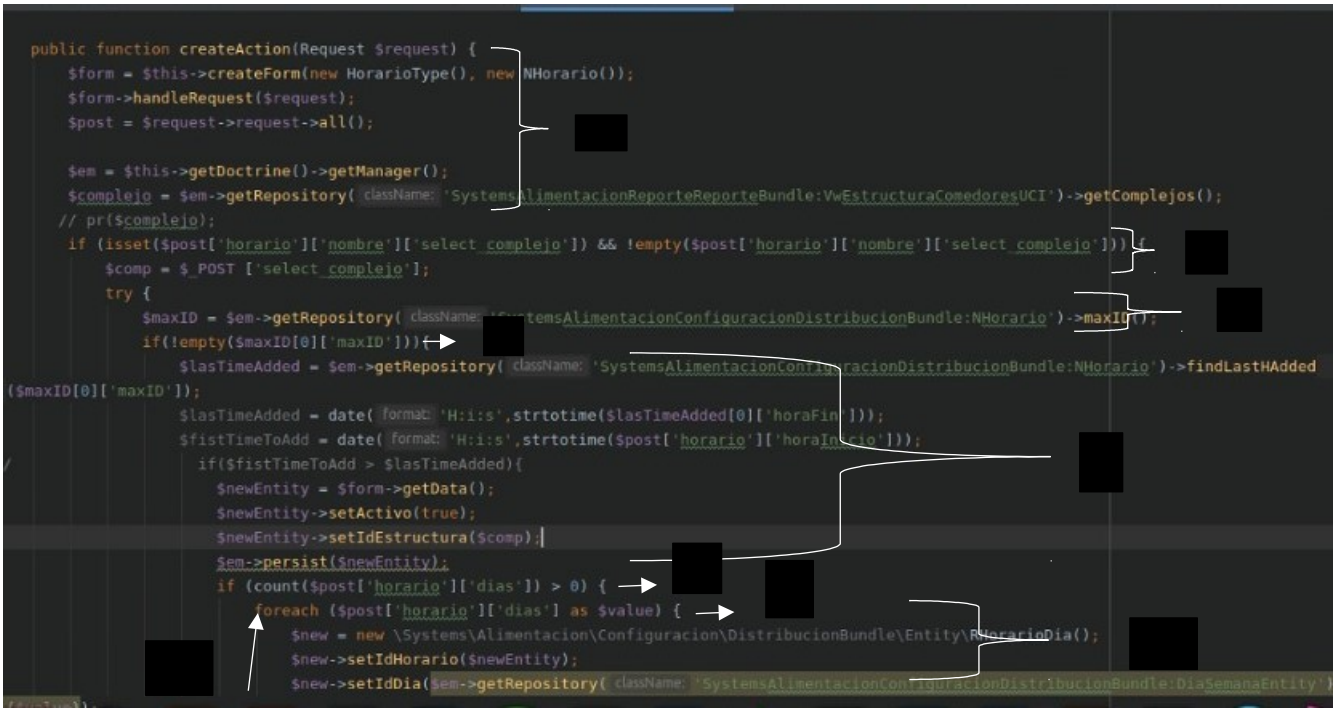


Figura 15: No conformidades. Fuente: [Elaboración Propia]

3.4.2 Pruebas de Unidad

Las pruebas de unidad tienen como objetivo verificar la unidad más pequeña del diseño del *software*. Estas pruebas se concentran en la lógica del procesamiento interno y en las estructuras de datos tales como: código fuente, archivos binarios, archivos de datos, entre otros. Este tipo de prueba se puede aplicar en paralelo a varios componentes (Rosales González and Martínez León 2015).

Las pruebas de unidad se realizan mediante el **método de caja blanca o estructural**, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. La técnica de prueba de **caja blanca** que se sugiere para la investigación es el **camino básico**, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Martínez Cabrera 2019).



```
public function createAction(Request $request) {
    $form = $this->createForm(new HorarioType(), new NHorario());
    $form->handleRequest($request);
    $post = $request->request->all();

    $em = $this->getDoctrine()->getManager();
    $complejo = $em->getRepository( className: 'SystemsAlimentacionReporteReporteBundle:VwEstructuraComedoresUCI')->getComplejos();
   // pr($complejo);
    if (isset($post['horario']['nombre']['select complejo']) && !empty($post['horario']['nombre']['select complejo'])){
        $comp = $_POST ['select complejo'];
        try {
            $maxID = $em->getRepository( className: 'SystemsAlimentacionConfiguracionDistribucionBundle:NHorario')->maxID();
            if(!empty($maxID[0]['maxID'])){
                $lasTimeAdded = $em->getRepository( className: 'SystemsAlimentacionConfiguracionDistribucionBundle:NHorario')->findLastHAdded
($maxID[0]['maxID']);
                $lasTimeAdded = date( format: 'H:i:s',strtotime($lasTimeAdded[0]['horaFin']));
                $fistTimeToAdd = date( format: 'H:i:s',strtotime($post['horario']['horaInicio']));
                if($fistTimeToAdd > $lasTimeAdded){
                    $newEntity = $form->getData();
                    $newEntity->setActivo(true);
                    $newEntity->setIdEstructura($comp);
                    $em->persist($newEntity);
                    if (count($post['horario']['dias']) > 0) {
                        foreach ($post['horario']['dias'] as $value) {
                            $new = new \Systems\Alimentacion\Configuracion\DistribucionBundle\Entity\RHorarioDia();
                            $new->setIdHorario($newEntity);
                            $new->setIdDia($em->getRepository( className: 'SystemsAlimentacionConfiguracionDistribucionBundle:DiaSemanaEntity')
```

```
                $new->setIdHorario($newEntity);
                $new->setIdDia($em->getRepository( className: 'SystemsAlimentacionConfiguracionDistribucionBundle:DiaSemanaEntity')->f

                $em->persist($new);
            }
        }
        $em->flush();
        $message = $this->get('translator')->trans('app.msg.inf_success');
        return self::showMessage( success: true, $message, array(), $this->generateUrl
ms_alimentacion_configuration_distribucion_horario_index'));
    }else{
        $message = $this->get('translator')->trans('No es posible realizar la accion porque el horario coincide con uno de los
strados');
        return self::showMessage(false, $message);
    }
}else{
    $newEntity = $form->getData();
    $newEntity->setActivo(true);
    $em->persist($newEntity);
    if (count($post['horario']['dias']) > 0) {
        foreach ($post['horario']['dias'] as $value) {
            $new = new \Systems\Alimentacion\Configuracion\DistribucionBundle\Entity\RHorarioDia();
            $new->setIdHorario($newEntity);
            $new->setIdDia($em->getRepository( className: 'SystemsAlimentacionConfiguracionDist                SemanaEntity')->find

            $em->persist($new);
```

Figura 16: Código para realizar el método

Una vez estudiado el método mostrado, se prosiguió a realizar el grafo correspondiente y la **Complejidad Ciclomática** que se realiza a todos los métodos o algoritmos de un sistema informático para determinar los caminos que debe tomar el método. A continuación, se muestran cada uno de estos:

Fórmula 1	**Fórmula 2**	**Fórmula 3**
V (G) = (A - N) + 2	V (G) = P+ 1	V (G) = R
V (G) = 23-17 + 2	V (G) = 7+1	V (G)= 8
V (G) = 8	V (G)= 8	

A: es la cantidad de aristas.

N: la cantidad de nodos.

P: es el número de nodos predicado contenidos en el grafo de flujo G.

R: representa la cantidad de regiones en el grafo.

Como se puede observar, después de aplicadas las fórmulas 1, 2 y 3 a la funcionalidad *createHorario* posee una complejidad ciclomática igual a 8, lo cual demuestra que el grafo está bien diseñado y que existen 8 caminos lógicos e independientes.

C#1: 1- 2 -14 - 5- 9

C#2: 1- 2 - 3 - 13 - 9

C#3: 1 - 2 - 3 – 4 – 10 – 11 - 9

C#4: 1 - 2 - 3 - 4 – 10 – 11 -12 -15 - 9

C#5: 1 – 2 – 3 – 4 – 10 – 11 – 12 – 15 -9

C#6: 1- 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

C#7: 1 – 2 – 3 – 4 – 10 – 11 – 12 – 15 – 12’ – 12 -15 -9

C#8: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 -7’- 7 - 9

Grafo resultante

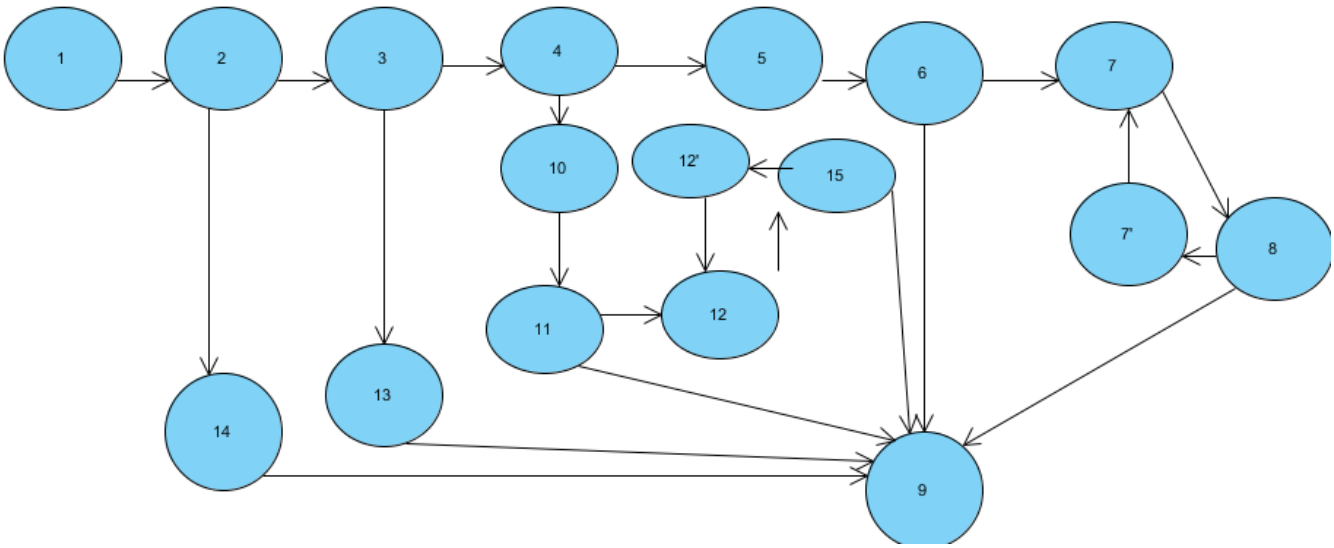


Figura 17: Grafo resultante Fuente:[Elaboración Propia]

Al analizar el grafo de flujo e identificar los caminos a recorrer, se definieron los casos de pruebas por cada camino respectivamente, de manera que se verifique que las condiciones de los nodos predicados estén establecidas correctamente. A continuación, se relacionan

ejemplos de los casos de pruebas realizados, los demás se archivaron en un documento donde se guardan datos de las pruebas realizadas:

Caso de prueba # 1

Tabla 11: Caso de prueba Horario *Fuente: [Elaboración Propia]*

Descripción: En el método *createAction crea un horario.*
Condición de ejecución: Seleccionar el ícono flotante Registrar Horario.
Entrada : N/A
Resultados Esperados : N/A
Resultados: N/A

Caso de prueba # 2

Tabla 12: Caso de prueba Horario Almuerzo *Fuente: [Elaboración Propia]*

Descripción: En el método *createAction crea un horario.*
Condición de ejecución : Seleccionar el ícono flotante Registrar Horario.
Entrada : N/A
Resultados Esperados : N/A
Resultados: N/A

Caso de prueba # 3

Tabla 13: Caso de prueba Horario Comida *Fuente: [Elaboración Propia]*

Descripción: En el método *createAction crea un horario.*
Condición de ejecución: Seleccionar el ícono flotante Registrar Horario, no haya horario registrado.
Entrada: Nombre: Comida. Horario de inicio: 6:30 pm. Horario de fin: 7:50 pm. Días de la semana: martes, miércoles Complejo: complejo comedor 1.
Resultados Esperados: Se añadió un nuevo horario, mostró mensaje de confirmación.
Resultados: Satisfactorio.

Caso de prueba # 4

Tabla 14: Caso de prueba Horario Desayuno *Fuente: [Elaboración Propia]*

Descripción: En el método *createAction crea un horario.*
Condición de ejecución: Seleccionar el ícono flotante Registrar Horario, no haya horario registrado.
Entrada: Nombre: Desayuno Horario de inicio: 7:30 am Horario de fin: 8:50 am Días de la semana: lunes, martes Complejo: complejo comedor 2
Resultados Esperados : Se añadió un nuevo horario, mostró mensaje de confirmación.
Resultados: Satisfactorio.

Caso de prueba # 5

Tabla 15 Caso de prueba Horario Comida Fuente: [Elaboración Propia]

Descripción: En el método *createAction crea un horario.*
Condición de ejecución : Seleccionar el ícono flotante Registrar Horario.
Entrada : N/A
Resultados Esperados : N/A
Resultados: N/A

Caso de prueba # 6

Tabla 16 Caso de prueba Horario Almuerzo Fuente: [Elaboración Propia]

Descripción: En el método *createAction crea un horario.*
Condición de ejecución : Seleccionar el ícono flotante Registrar Horario.
Entrada: Nombre: almuerzo Horario de inicio: 11:50 am Horario de fin: 1:50 pm Días de la semana: jueves, viernes

Complejo: complejo comedor 1
Resultados Esperados : Se añadió un nuevo horario, mostró mensaje de confirmación.
Resultados: Satisfactorio

Caso de prueba # 7

Tabla 17 Caso de prueba Horario Comida Fuente: [Elaboración Propia]

Descripción: En el método *createAction crea un horario.*
Condición de ejecución : Seleccionar el ícono flotante Registrar Horario, no haya horario registrado.
Entrada: Nombre: comida Horario de inicio: 7:00 pm Horario de fin: 9:00 pm Días de la semana: domingo, lunes Complejo: complejo comedor 2
Resultados Esperados : Se añadió un nuevo horario, mostró mensaje de confirmación.
Resultados: Satisfactorio

Caso de prueba # 8

Tabla 18 Caso de prueba Horario Desayuno Fuente: [Elaboración Propia]

Descripción: En el método *createAction crea un horario.*
Condición de ejecución : Seleccionar el ícono flotante Registrar Horario.
Entrada: Nombre: desayuno Horario de inicio: 5:00 am Horario de fin: 6:00 am Días de la semana: lunes, martes Complejo: complejo comedor 3
Resultados Esperados : Se añadió un nuevo horario, mostró mensaje de confirmación.
Resultados: Satisfactorio

Con la realización de los casos de prueba diseñados se probó la ejecución de cada sentencia del código al menos una vez, teniendo en cuenta todas las condiciones lógicas en sus variantes verdaderas y falsas. Los resultados del método de caja blanca fueron satisfactorios.

Figura 18: Resultados de los casos de prueba de caja blanca.

Conclusiones del capítulo

- Se detallaron los estándares de codificación empleados en la implementación del Módulo de Configuración permitiendo una mejor organización y comprensión del código.
- Con los resultados de la validación de los requisitos identificados se logró obtener un listado de requisitos correctamente redactados y descritos.
- La descripción de las pruebas utilizadas para asegurar la calidad del *software*, permitió obtener resultados satisfactorios, asegurando que el sistema implementado no contiene errores y tiene la aceptación requerida.

CONCLUSIONES FINALES

Al terminar la presente investigación se dio solución a la problemática planteada, contribuyendo a dar soporte al módulo de configuración para una mejor gestión y generación de horarios por individual, se obtuvieron resultados que permiten arribar a las siguientes conclusiones:

- Con el estudio de soluciones informáticas homólogas se identificaron las principales tendencias en el desarrollo de sistemas de gestión de configuración. Aunque los sistemas estudiados no brindan una solución directa al problema de la investigación.
- El análisis de los principios teóricos de la investigación, enfocados a los sistemas de gestión de configuración, permitió un mejor entendimiento de los procesos de este tipo en la universidad.
- La elaboración de los artefactos propuestos por la metodología de desarrollo y el levantamiento de requisitos permitieron una mayor comprensión, identificación de los procesos y características del sistema desarrollado, como resultado se obtuvo un sistema que cumple con los requerimientos definidos.
- Con la adopción de un estándar de código se logró la organización, legibilidad y comprensión del código generado en la implementación del sistema propuesto.
- El diseño y ejecución de los casos de prueba, permitieron identificar errores de implementación en la aplicación y darle cumplimiento al objetivo general de esta investigación.

RECOMENDACIONES

Se recomienda implementar un nuevo módulo de configuración q sea capaz de visualizar los horarios del complejo solo para los directivos de dichas áreas y su jefes superiores.

REFERENCIAS BIBLIOGRÁFICAS

1. Abrego Almazán, D., et al. (2017). "Influencia de los sistemas de información en los resultados organizacionales." **62**(2): 303-320.

2. Aliaga, A. M. C., et al. (2019). "Sistema para la gestion de activos multimedia para la Dirección de Extensión Universitaria." **12**(12): 1-10.

3. Ambler, S. J. O. d. A. h. w. a. c. u. a. h. (2005). "The agile unified process (aup)."

4. Arias, M. Á. (2017). Aprende Programacion Web con PHP y MySQL: 2ª Edicion, IT campus Academy.

5. Bandera Rodríguez, Y. (2016). Migración del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón, Universidad de las Ciencias Informáticas. Facultad 3.

6. Blanco Pérez, O., et al. (2016). Herramienta para la estimación de tiempo y esfuerzo de las pruebas de aceptación, liberación y piloto.

7. Calle-López, D., et al. (2018). "Un sistema experto basado en minería de datos y programación entera lineal para soporte en la asignación de materias y diseño de horarios en educación superior." **9**(1): 102-117.

8. Capacho, J. R. and W. Nieto Bernal (2017). Diseño de bases de datos, Universidad del Norte.

9. Cooper, J. W. (2000). "Java design patterns: a tutorial."

10. Deacon, J. J. O. h. w. j. c. u. b. M. p. (2009). "Model-view-controller (mvc) architecture." **28**.

11. Felt, A. P., et al. (2017). Measuring {HTTPS} adoption on the web. 26th USENIX Security Symposium (USENIX Security 17).

12. Fernandez, M. A. J. C., Docencia y Tecnología Suplemento (2019). "Gestión de la configuración: Evaluación de la aplicabilidad en pequeñas y medianas áreas de Informática de la Administración Pública de Entre Ríos." **9**(9).

13. Fernández Pérez, Y. (2018). "Modelo computacional para la evaluación y selección de productos de software."

REFERENCIAS *BIBLIOGRÁFICAS*

14. García Morales, Y. (2017). Aplicación informática para la adquisición de datos en sistemas basados en la telemetría, Universidad de las Ciencias Informáticas. Centro de Investigación y ….

15. González Espinosa, G., et al. (2015). Sistema de Reportes y apoyo a la toma de decisiones para la Dirección de Alimentación de la Universidad de las Ciencias Informáticas.

16. González Sierra, I. (2017). Sistema de gestión del Calendario de tesis de grado en la Universidad de las Ciencias Informáticas, Universidad de las Ciencias Informáticas. Facultad de Ciencias y Tecnologías ….

17. Granda Asencio, L. Y., et al. (2019). "Las TICs como herramientas didácticas del proceso de enseñanza-aprendizaje." **15**(66): 104-110.

18. Gulati, R., et al. (2016). Management: An integrated approach, Cengage Learning.

19. Irmayani, W. and E. J. J. K. I. Susyatih (2017). "Sistem Informasi Anggaran Pendapatan dan Belanja Desa Berorientasi Objek." **5**(1).

20. Kaehler, B. and J. Grundei (2018). HR Governance: A Theoretical Introduction, Springer.

21. Leyva Jerez, G. R. (2018). Modelo para la réplica de datos en sistemas de bases de datos distribuidas, Universidad de las ciencias Informáticas. Centro de Consultoría y Desarrollo ….

22. Lubiński, W., et al. (2020). "Comparison of visual outcomes after implantation of AtLisa tri 839 MP and Symfony intraocular lenses." **40**(10): 2553-2562.

23. Martínez Cabrera, L. (2019). Sistema para la Gestión de la Información Estadística del Festival de Artistas Aficionados, Universidad de las Ciencias Informáticas. Facultad 4.

24. Merelo Hernández, M. (2017). "Estudio de la gestión de configuración a través de NETCONF."

25. Ordóñez, M. P. Z., et al. (2017). Administración de Bases de datos con PostgreSQL, 3Ciencias.

26. Paredes, L. M. and D. A. G. Mosquera "Gestión de confiGuración."

27. Pérez Bejerano, Y. (2019). Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba versión 2.0, Universidad de las Ciencias Informáticas. Facultad 1.

28. Perez, J. S. (2022). "Los 13 mejores Software de Control Horario y Presencia."

29. Pressman, R. (2009). Software Engineering: A Practioner's Approach (New York). Thomas Casson.

30. Raharjana, I. K., et al. (2021). "User stories and natural language processing: A systematic literature review." **9**: 53811-53826.

31. Reyes Navarro, Y. (2016). Migración del módulo Ejecución del Sistema Orbita a la arquitectura de referencia en PHP Bosón, Universidad de las Ciencias Informáticas. Facultad 3.

32. Robbins, S. P. and M. J. E. P. E. Coulter (2016). "Management (13th global edition)."

33. Rosales González, L. I. and Y. Martínez León (2015). Sistema para la Gestión de la Información de los Expedientes Tecnológicos del Centro de Gobierno Electrónico, Universidad de las Ciencias Informáticas. Facultad 3.

34. Saeed, S., et al. (2019). "Analysis of software development methodologies." **8**(5): 446-460.

35. Saks, E. (2019). "JavaScript Frameworks: Angular vs React vs Vue."

36. Sanchez Peño, J. M. (2015). "Pruebas de software. fundamentos y técnicas."

37. Sánchez, T. R. J. L. H. s. (2015). "Metodología de desarrollo para la Actividad productiva de la UCI."

38. Soni, R. (2016). Nginx, Springer.

39. Suárez Alfonso, A., et al. (2015). "La gestión de la información: herramienta esencial para el desarrollo de habilidades en la comunidad estudiantil universitaria." **7**(2): 72-79.

40. Suárez, M. C., et al. (2017). "Diseño y validación de un instrumento de observación para valorar la toma de decisiones en la acción de recepción en voleibol." **12**(34): 67-75.

41. Tabares, R. B. J. E. C. e. I. (2010). "Patrones grasp y anti-patrones: un enfoque orientado a objetos desde logica de programacion." **4**(8): 161-173.

42. Taylor, F. W. (1911). Shop Management (new edition 2008), Nu Vision Publications (first published 1911).

43. Venkat, N. (2017). "RDBMS PostgreSQL-Installation of PostgreSQL."

44. Vidal Piña, L. M. (2017). Modelo de alertas sobre la ocurrencia de eventos extraordinarios en sistemas SCADA, Universidad de las Ciencias Informáticas. Facultad 4. Centro de Informática ….

45. Westfall, L. J. S. Q. P. (2005). "Software requirements engineering: what, why, who, when, and how." **7**(4): 17.

46. Zandstra, M. (2021). PHP Standards. PHP 8 Objects, Patterns, and Practice, Springer**:** 571-593.

ANEXOS

Entrevista realizada para el levantamiento de requisitos.

1. ¿Qué problemas tiene el sistema?
2. ¿Qué es lo que necesita?
3. ¿Qué cree que pueda ser mejorado?
4. ¿Cómo le gustaría que el sistema muestre las nuevas funcionalidades?

Patrón controlador muestra del código:

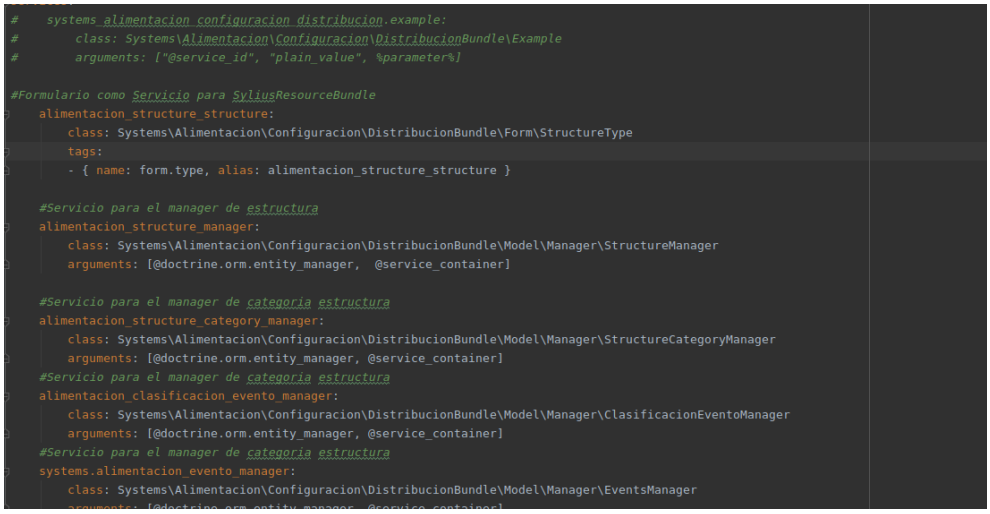
```
public function createAction(Request $request) {
    $form = $this->createForm(new HorarioType(), new NHorario());
    $form->handleRequest($request);
    $post = $request->request->all();

    $em = $this->getDoctrine()->getManager();
    $complejo = $em->getRepository( className: 'SystemsAlimentacionReporteReporteBundle:VwEstructuraComedoresUCI')->getComplejos
    if (isset($post['horario']['nombre']) && !empty($post['horario']['nombre'])) {
        try {
            $maxID = $em->getRepository( className: 'SystemsAlimentacionConfiguracionDistribucionBundle:NHorario')->maxID();
            if(!empty($maxID[0]['maxID'])){
                $lasTimeAdded = $em->getRepository( className: 'SystemsAlimentacionConfiguracionDistribucionBundle:NHorario')->fi
maxID[0]['maxID']);
                $lasTimeAdded = date( format: 'H:i:s',strtotime($lasTimeAdded[0]['horaFin']));
                $fistTimeToAdd = date( format: 'H:i:s',strtotime($post['horario']['horaInicio']));
                  if($fistTimeToAdd > $lasTimeAdded){
                    $newEntity = $form->getData();
                    $newEntity->setActivo(true);
                    $em->persist($newEntity);
                    if (count($post['horario']['dias']) > 0) {
```

Figura 19. Patrón Controlador.

Patrón Creador muestra del código:

Anexos



```
#    systems alimentacion configuracion distribucion.example:
#        class: Systems\Alimentacion\Configuracion\DistribucionBundle\Example
#        arguments: ["@service_id", "plain_value", %parameter%]

#Formulario como Servicio para SyliusResourceBundle
    alimentacion_structure_structure:
        class: Systems\Alimentacion\Configuracion\DistribucionBundle\Form\StructureType
        tags:
        - { name: form.type, alias: alimentacion_structure_structure }

    #Servicio para el manager de estructura
    alimentacion_structure_manager:
        class: Systems\Alimentacion\Configuracion\DistribucionBundle\Model\Manager\StructureManager
        arguments: [@doctrine.orm.entity_manager,  @service_container]

    #Servicio para el manager de categoria estructura
    alimentacion_structure_category_manager:
        class: Systems\Alimentacion\Configuracion\DistribucionBundle\Model\Manager\StructureCategoryManager
        arguments: [@doctrine.orm.entity_manager, @service_container]
    #Servicio para el manager de categoria estructura
    alimentacion_clasificacion_evento_manager:
        class: Systems\Alimentacion\Configuracion\DistribucionBundle\Model\Manager\ClasificacionEventoManager
        arguments: [@doctrine.orm.entity_manager, @service_container]
    #Servicio para el manager de categoria estructura
    systems.alimentacion_evento_manager:
        class: Systems\Alimentacion\Configuracion\DistribucionBundle\Model\Manager\EventsManager
        arguments: [@doctrine.orm.entity_manager, @service_container]
```

Figura 20: Patrón Creador.