Universidad de las Ciencias Informáticas

Facultad 4

“Módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1.”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Heriberto Sarmiento González

Tutor(es): Ing. Alberto Torres Junco

Ing. Lisandra Remedios Revol

Ing. Carlos Manuel Castillo Chacón

La Habana, junio de 2018

"Recuerden que el eslabón más alto que puede alcanzar la especie humana es ser revolucionario"

Ernesto "Che" Guevara

Declaración de Autoría

Declaración de Autoría

Declaro ser el único autor del presente trabajo de diploma y otorgo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor

Heriberto Sarmiento González

_____

Firma del Tutor

Ing. Alberto Torres Junco

_____

Firma del Tutor

Ing. Lisandra Remedios Revol

_____

Firma del Tutor

Carlos Manuel Castillo Chacón

Datos de contacto

Ing. Alberto Torres Junco

Correo electrónico: ajunco@uci.cu

Ingeniero en Ciencias Informáticas, UCI, 2015. Trabaja en el Departamento de Componentes del centro FORTES de la Facultad 4.

Ing. Lisandra Remedios Revol

Correo electrónico: lremedios@uci.cu

Ingeniera en Ciencias Informáticas, UCI, 2016. Trabaja en el Departamento de Aplicaciones del centro FORTES de la Facultad 4.

Ing. Carlos Manuel Castillo Chacón

Correo electrónico: cmcastillo@uci.cu

Ingeniero en Ciencias Informáticas, UCI. Trabaja en el Departamento de Programación de la Facultad 4.

Dedicatoria

A mis padres Heriberto y Diabelkis y mi hermanita Yanet por ser las personas más importantes de mi vida a las cuales amo y dedico todos mis logros.

A mis abuelas Ana y Magalis por todo el cariño, amor y dedicación.

A mi familia por su apoyo incondicional.

Agradecimientos

A mis padres por apoyarme en mis decisiones, por la confianza que han depositado en mí y por darme todos los gustos.

A mi hermanita Yanet por quererme tanto y estar siempre pendiente de mí.

A mis primos Guillermo y Alberto por toda la ayuda brindada

A mi familia porque de una forma u otra me apoyaron en todo momento.

A mis tutores Lisandra Remedio Revol, Alberto Torres Junco y Carlos Manuel Castillo Chacón por guiarme durante el desarrollo de la tesis.

A los que conocí en la universidad y consideré amigos, porque aprendí a confiar en ellos y de los cuales recibí apoyo y cariño: Alexnurín, Ernesto, Eliany, Alex, Rioger, Darian, Raymari, Katherine, Yolanda, Daniela, Roberto, Randy, Alejandro, Yoandy, Yancel, Leonel, Leandro, Michel y Karel.

A mis amigos de toda la vida: Idel, Orlando, Reinaldo, Maibel, Yasmany y Eduardo.

A mis vecinos y amigos por preocuparse por mí.

A todos los profesores que me ayudaron en mi preparación profesional.

Resumen

Con la evolución de las Tecnologías de la Información y las Comunicaciones, han evolucionado además la forma de enseñar y los métodos para medir lo aprendido. Una de las modalidades educativas que han surgido en correspondencia con las TIC es el modelo conocido como *e-learning*; conjunto de tecnologías que permiten transmitir el conocimiento por medios electrónicos. Para dar soporte a esta modalidad surgen los Sistemas de Gestión de Aprendizaje; sistemas que permiten administrar, evaluar y apoyar las actividades previamente diseñadas y programadas dentro de un proceso de formación. En la Universidad de las Ciencias Informáticas, se desarrolla la Plataforma Educativa Xauce ZERA v2.1. Sin embargo, en esta plataforma es difícil seguir de una manera eficiente la programación de las actividades creadas en los cursos de la misma. Hoy el usuario no dispone de recursos que le indiquen de manera resumida y centralizada las actividades que están próximas a comenzar. Para solventar el problema se desarrolla un módulo que permite la planificación y organización de eventos en la plataforma. Su desarrollo estuvo guiado por la metodología AUP versión UCI. Se utilizó el entorno integrado de desarrollo NetBeans, servidor de bases de datos PostgreSQL, el servidor de aplicación Nginx, entre otras herramientas y tecnologías. Se describió la propuesta de solución definiéndose las condiciones y funcionalidades del sistema en los requisitos funcionales y no funcionales. Se obtuvieron los artefactos necesarios para un mejor entendimiento y comprensión de las tareas a realizar, así como toda la documentación correspondiente.

Palabras clave: *e-learning*, Plataforma Educativa y eventos.

Índice de contenido

Índice

Índice de contenido

Índice de contenido

Índice de figuras y tablas

Índice de figuras

Índice de figuras y tablas

Índice de figuras y tablas

Índice de tablas

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) en el mundo, han llevado a una implicación de los gobiernos de todos los países y a todas las escalas, a promover y propiciar la constitución de redes de información que garanticen el acceso efectivo de los ciudadanos a las mismas. Constituyen aquellas herramientas y materiales computacionales e informáticos que procesan, almacenan, resumen, recuperan y presentan información representada de la más variada forma. Ofrecen facilidades para el desarrollo del aprendizaje, menores riesgos en la toma de decisiones de las organizaciones, nuevas formas de trabajo y desarrolla a las personas con el apoyo de redes de intercambio. También brindan un fácil acceso a una gran fuente de información, realizando un proceso rápido y fiable mediante canales de comunicación inmediata (José Huidobro 2018).

Hoy día existen herramientas y aplicaciones enfocadas a la educación tales como: multimedias, bibliotecas virtuales, laboratorios virtuales y libros electrónicos; que mediante Internet o una red entre ordenadores logran que se desarrollen acciones formativas y un satisfactorio proceso enseñanza-aprendizaje. Este proceso sin limitaciones de horarios caracterizado por una separación física entre estudiantes y profesores con un apoyo continuo de tutores especializados toma el nombre de educación a distancia o teleformación (Ortega, Jesús Fernán 2018).

El surgimiento de la educación a distancia dio paso al aprendizaje electrónico o *eLearning*, que permite transmitir el conocimiento por medios electrónicos. El *eLearning* es definido como un: “conjunto de tecnologías, aplicaciones y servicios orientados a facilitar la enseñanza y el aprendizaje a través de *Internet*/*Intranet*, que facilitan el acceso a la información y la comunicación con otros participantes” (Guerra, Yulaine Arias 2010). Para dar soporte a esta modalidad educativa surgen los Sistemas de Gestión de Aprendizaje (LMS por sus siglas en inglés); herramientas que permiten automatizar contenidos de formación y controlar los procesos de enseñanza-aprendizaje (Universidad Internacional de Valencia 2018).

En Cuba desde hace varios años diversas instituciones educativas comenzaron a utilizar estas plataformas *web*, como apoyo a la educación tradicional y para impartir algunos cursos a distancia. También se desarrollaron herramientas para el uso del aprendizaje electrónico en los procesos docentes educativos.

Un ejemplo claro de la aplicación de estas tecnologías en nuestro país lo constituye la Universidad de las Ciencias Informáticas (UCI). Esta universidad cuenta con un gran número de centros productivos, entre estos se encuentra el Centro de Tecnologías para la Formación (FORTES), cuya razón de ser es desarrollar tecnologías que permitan ofrecer servicios y productos encaminados a la Formación aplicando las TIC.

FORTES se encuentra desarrollando la plataforma para la gestión del aprendizaje Xauce ZERA v2.1. Esta integra las principales especificaciones y estándares educativos, desarrollados y utilizados a nivel mundial en plataformas de aprendizaje colaborativo. Sin embargo, en esta plataforma es difícil seguir de una manera eficiente la programación de las actividades creadas en los cursos de la misma. Hoy el usuario no dispone de recursos que le indiquen de manera resumida y centralizada las actividades que están próximas a comenzar o en qué momento va a ocurrir algún evento planificado por el profesor en el sistema. La plataforma no cuenta con un componente que permita el tratamiento de eventos.

En antiguas versiones de la Plataforma Educativa Xauce ZERA v2.1 existían recursos que permitían darle tratamiento a la situación antes descrita. Sin embargo, en dichas versiones se utilizaba una tecnología antigua e incompatible con la que se emplea en la versión actual, lo que dificulta la reutilización del código en la plataforma actual. Otro elemento a tener en cuenta es un cambio completo en la arquitectura[1] que soporta el nuevo sistema, el cual se basa en un núcleo diferente llamado Xalix, cuyo comportamiento y funcionamiento cambia radicalmente respecto a versiones antiguas.

A partir de la situación problemática descrita anteriormente se propone como **problema a resolver** ¿Cómo contribuir a la planificación y organización centralizada de eventos en la Plataforma Educativa Xauce ZERA v2.1?

Con el fin de solucionar el problema, se define como **objeto de estudio** de la presente investigación: la gestión de eventos mediante agendas en sistemas informáticos. Enfocando el **campo de acción**: la gestión de eventos mediante agendas en plataformas educativas.

Teniendo en cuenta el problema a resolver se define como **objetivo general**: desarrollar un módulo Agenda que contribuya a la planificación y organización centralizada de eventos para la Plataforma Educativa Xauce ZERA v2.1.

[1] Estructuras de un sistema compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.

Complementando el objetivo general se definen los siguientes **objetivos específicos:**

- ✓ Elaborar el marco teórico referencial sobre el proceso de gestión de eventos en plataformas educativas.
- ✓ Desarrollar el análisis y diseño de las funcionalidades del módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1.
- ✓ Implementar las funcionalidades del módulo Agenda para la plataforma Educativa Xauce ZERA v2.1 de acuerdo con la estructura de diseño definida.
- ✓ Realizar pruebas a la solución propuesta para comprobar el cumplimiento de los requerimientos de la misma.

Para guiar la investigación se define como **idea a defender**: el desarrollo del módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1 contribuirá a la planificación y organización centralizada de eventos.

Posibles resultados:

Al concluir se tendrán:

- ✓ Módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1 en el cual se podrán gestionar los eventos de forma centralizada.
- ✓ Documentación complementaria que incluya artefactos de análisis, diseño e implementación del módulo que permita la gestión de eventos en la Plataforma Educativa Xauce ZERA v2.1.

Métodos investigativos

Para realizar la investigación se utilizan métodos teóricos y empíricos que a continuación se relacionan:

Métodos Teóricos:

- ✓ Histórico-Lógico: permitió el estudio de la evolución y desarrollo de la gestión de eventos mediante agendas en plataformas educativas.
- ✓ Modelación: se utilizó para reflejar la estructura, relaciones internas y características de la solución a través de la realización de diagramas.

- ✓ Analítico-Sintético: se utilizó con el objetivo de analizar los elementos bibliográficos y definiciones relacionadas con el término de agenda y evento en plataformas educativas. También para identificar las tecnologías a utilizar en el desarrollo de la solución.

Métodos Empíricos

- ✓ Observación: mediante la utilización del presente método se logra conocer la esencia del problema planteado, pues analizando desde varios puntos de vista la propuesta de solución y otras soluciones existentes, se permite identificar qué está hecho y qué falta por hacer.

Estructura capitular

El presente documento está compuesto por tres capítulos que incluyen todo lo relacionado con el trabajo investigativo, así como el análisis y diseño del módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1.

Capítulo 1: Fundamentos de la investigación.

Se hace referencia a los elementos teóricos que constituyen la base de la investigación realizada. Se describen los conceptos esenciales para comprender el problema, así como las tecnologías y herramientas a utilizar en el análisis y diseño del módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1.

Capítulo 2: Análisis y diseño del sistema.

Se describe el proceso de desarrollo de la propuesta de solución especificando sus características principales. Se expone el modelo conceptual y los requisitos funcionales y no funcionales de la aplicación a implementar. Se muestran los principales procesos haciendo uso de las historias de usuario. Y se exponen los artefactos que se generan a partir de la metodología de desarrollo de *software* utilizada.

Capítulo 3: Implementación y validación del sistema.

Se describen los elementos necesarios para la implementación, partiendo del resultado obtenido del diseño. Se muestra la organización de los componentes con sus relaciones lógicas a través del diagrama de componentes, quedando conformado el módulo Agenda. Se realizan las pruebas de caja negra utilizando la técnica de la Partición de Equivalencia y se reflejan los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTOS DE LA INVESTIGACIÓN

Introducción

En este capítulo se realiza la fundamentación teórica de la investigación, abordando temas de relevancia para la misma como los conceptos y términos asociados al dominio del problema. Se realiza un estudio de las diferentes herramientas que son útiles para la implementación del módulo. Se ejecuta la selección de la metodología a utilizar en el ciclo de vida del *software*. De igual manera se hace una revisión y elección de lenguajes de programación propuestos para el desarrollo del sistema y se analizan soluciones similares.

1.1 Conceptos asociados al dominio del problema

A continuación, se realiza una descripción detallada de los conceptos asociados a la investigación.

1.1.1 Evento

Un evento es un hecho programado que reviste gran importancia para sus mentores, por lo que es planificado con suficiente antelación. Algunas organizaciones e instituciones planifican anualmente sus eventos, organizando sus agendas en torno a los principales eventos programados. También en informática se utiliza el término evento para denominar las acciones detectadas por los programas*(Evento 2018)*.

A la hora de crear un evento hay que definir claramente el tipo de evento que se va a utilizar, una vez definido el tipo comienza la fase de preparación, que son las acciones que han de realizarse antes de la creación del evento. Esta fase se encarga en la selección del día y la hora en que se realice el evento, cuantas personas serán notificadas por el evento y cuantos días permanecerá el evento *(Caritol 2009)*.

1.1.2 Agenda

Agenda es el programa que contiene, ordenadamente, un conjunto de temas, tareas o actividades para su realización en un periodo de tiempo determinado. Es una serie de asuntos, compromisos u obligaciones que una persona ha ordenado, dispuesto y planificado para ir tratando en un periodo de tiempo específico, también puede hacer referencia a la lista de temas que serán abordados durante una reunión.

Es aquella donde se asientan, disponen y programan, de manera ordenada, una serie de tareas o actividades relacionadas con el desempeño laboral. Como tal, permite al usuario organizar su tiempo de acuerdo con los objetivos de su gestión para maximizar su rendimiento, eficacia y productividad. En este

sentido, es una herramienta muy útil a la hora de organizar tareas y eventos. Posee la ventaja de encontrar un plan de actividades actualizado y preciso de forma sencilla en todo momento (Agenda 2014).

Las agendas son utilizadas mayormente en las plataformas educativas con el objetivo de que los estudiantes y los profesores agreguen en ella sus actividades personales a realizar en la semana o un evento específico.

1.1.3 LMS

Un LMS o Sistema de Gestión de Aprendizaje, es una aplicación instalada en un servidor, que administra, distribuye y controla las actividades de formación de una institución u organización**.** Su arquitectura y herramientas son apropiadas para clases en línea, así como también para complementar el aprendizaje presencial.

Las principales funciones del LMS son:

- ✓ Gestionar recursos de usuarios.
- ✓ Administrar el acceso, controlar y dar seguimiento del proceso de aprendizaje.
- ✓ Realizar evaluaciones.
- ✓ Generar informes.
- ✓ Gestionar servicios de comunicación como foros de discusión y videoconferencias.

Es una plataforma sólida que incorpora todo el contenido del curso, sus materiales y su administración en un único y mismo sistema en línea de uso sencillo. Permite a los docentes manejar con facilidad las clases y el progreso de sus alumnos, destacando sus puntos fuertes y débiles para posibilitar la continua mejora de su desempeño («LMS» 2018).

1.1.4 Plataforma Educativa

Una plataforma educativa es un punto de encuentro entre los estudiantes y profesores fuera del horario lectivo, y no sólo para comunicarse sino también para compartir material, intercambiar experiencias o prácticas, incluso para descubrir algo nuevo sobre una materia, o un concepto más específico de ella (Mónica María Agudelo B. 2017).

Es un entorno de trabajo que permite la capacidad de interactuar con uno o varios usuarios a través de diversas herramientas con fines pedagógicos. Se considera una herramienta que contribuye a la evolución

de los procesos de enseñanza y aprendizaje. Las Plataformas Educativas o *Learning Management System* (LMS) son de suma importancia en los Entornos Virtuales de Aprendizaje Híbridos (EVAH) que forman un espacio de interacción entre el profesor y el alumno (Arturos Ocampo López 2012).

Las plataformas educativas tienen, normalmente, una estructura modular que hace posible su adaptación a la realidad de los diferentes centros escolares. Cuentan con distintos módulos que permiten responder a las necesidades de gestión de los centros a tres grandes niveles: gestión administrativa y académica, gestión de la comunicación y gestión del proceso de enseñanza-aprendizaje (Díaz Becerro. S. 2009).

1.1.5 Plataforma Educativa Xauce ZERA v2.1

La Plataforma Educativa XAUCE ZERA 2.1 se desarrolla en el Centro de Tecnologías para Formación (FORTES) de la Universidad de las Ciencias Informáticas. Es una plataforma innovadora, flexible, fácil de usar, interactiva y adaptable, capaz de guiar el proceso de enseñanza-aprendizaje en empresas, organizaciones, comunidades e instituciones de cualquier nivel de enseñanza a partir de un conjunto poderoso de herramientas centradas en los aprendices y ambientes de aprendizaje colaborativo, que le dan poder, tanto a la enseñanza como al aprendizaje.

Es accesible desde cualquier dispositivo móvil y soporta contenidos interactivos, así como recursos multimedia a partir de plantillas previamente definidas que se entremezclan con documentos, presentaciones y otros materiales junto a foros de discusión, tareas, cuestionarios y disímiles tipologías de ejercicios que pueden ser evaluados de forma automática, por coevaluación o directamente por el profesor.

Es capaz de proporcionar a los profesores y aprendices un sistema integrado en línea único, robusto, seguro y fácil de usar para crear ambientes de aprendizaje personalizados que puedan soportar las necesidades tanto de clases pequeñas, como de grandes organizaciones debido a su flexibilidad y escalabilidad. También aporta a la enseñanza y el aprendizaje con tecnología educativa innovadora que ayuda a los profesores a adaptarse a las nuevas normas y personalizar el aprendizaje ofreciendo experiencias nuevas y emocionantes de aprendizaje digital y garantizando que todos los estudiantes tengan la oportunidad de desarrollar todo su potencial. Esta plataforma se está desarrollando sobre el marco de trabajo Xalix el cual utiliza como *framework* de desarrollo Symfony en su versión 2.7.16 (XAUCE ZERA 2018).

Capítulo 1: Fundamentos de la investigación

1.2 Análisis de soluciones similares existentes

1.2.1 Soluciones Genéricas

Se realizó un análisis de varios componentes diseñados para *Symfony* que pudieran ser útiles para la creación de una agenda. Entre ellos se encuentran: *Full Calendar Bundle* de Ancarebeca, Agenda *Bundle* de Claroline y *Calendar Bundle* de *Adesigns*. El primero no se puede utilizar puesto que no es compatible con el *framework* sobre el que está desarrollado la plataforma Xauce ZERA v2.1, ya que este está desarrollado sobre la versión 2.7.16 de *Symfony* y el *bundle* está desarrollado sobre la versión 3.0. El segundo se encuentra descontinuado y sin soporte, por lo que utilizar un recurso en estas condiciones es muy arriesgado al no contar con un equipo de desarrollo que le brinde mantenimiento. El tercero sí es compatible con el *framework* de la plataforma, pero no soporta la clasificación de eventos; ya que este es solo para eventos personales y la plataforma actualmente requiere además la gestión de eventos asociados a los cursos y grupos.

1.2.2 Soluciones a nivel internacional

Se han desarrollado diferentes sistemas de gestión de aprendizaje, los cuales cuentan con herramientas aplicadas a la gestión de eventos. Entre ellos se encuentran:

Moodle

Moodle (Entorno de Aprendizaje Dinámico Modular Orientado a Objetos) es una aplicación *web* de tipo Ambiente Educativo Virtual, un sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea. Moodle fue creado por Martin Dougiamas. Basó su diseño en las ideas del constructivismo en pedagogía que afirma que el conocimiento se construye en la mente del estudiante en lugar de ser transmitido sin cambios a partir de libros o enseñanzas y en el aprendizaje cooperativo.

El calendario de Moodle puede mostrar eventos de sitio, curso, grupo, usuario y categoría, además de fechas límites para tareas y exámenes, horas del chat y otros eventos del curso. El administrador puede elegir un tipo de calendario por defecto para todo el sitio, pero el usuario puede anular esto dentro de las configuraciones del mismo o dentro de su propio perfil de usuario y preferir su calendario. Algunas de sus funcionalidades son: añadir un evento, exportar calendario, importar calendario, importar eventos múltiples y mostrar eventos del día actual (Programadores 2017).

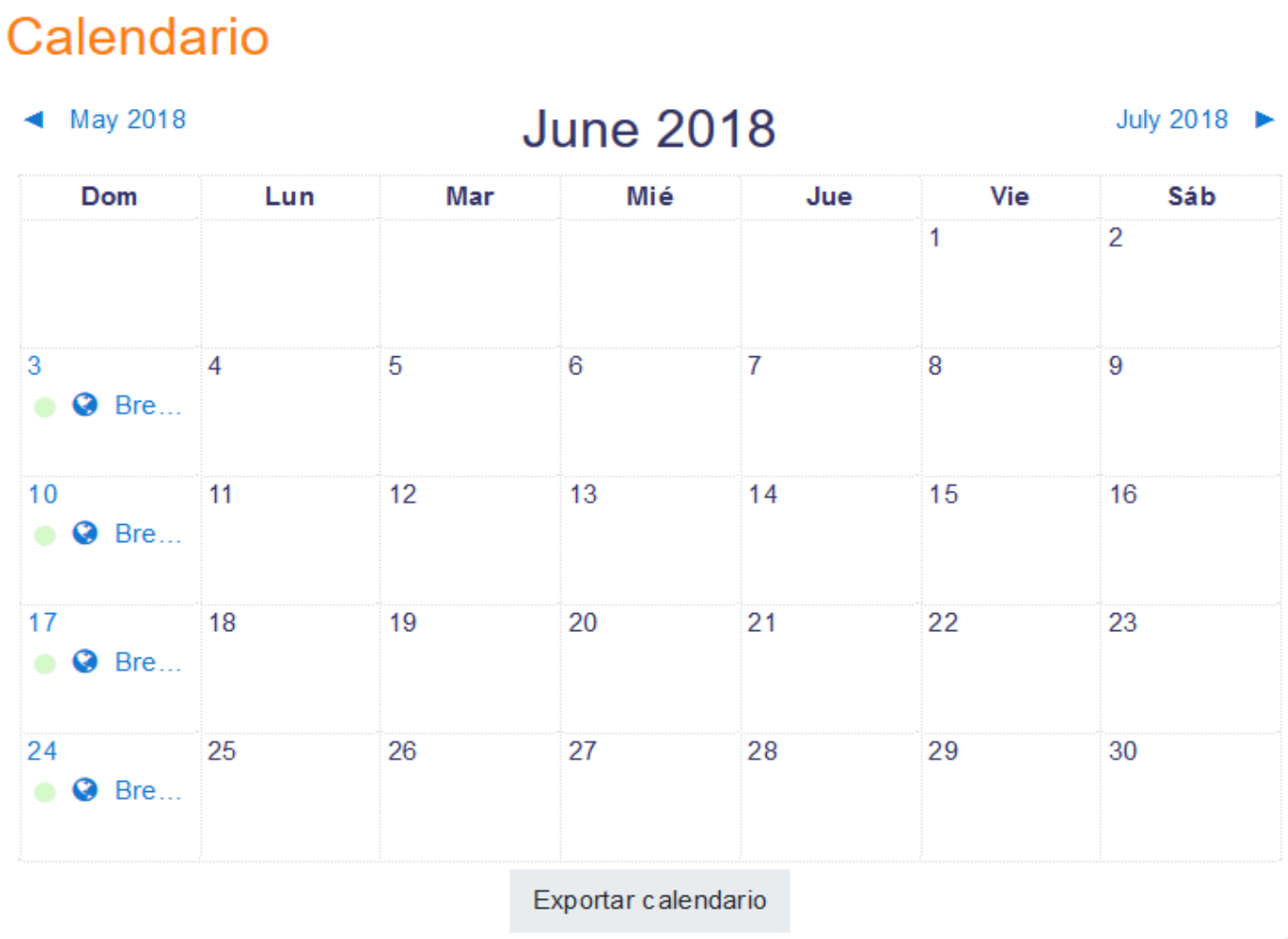


Figura 1 Calendario de Moodle.

Claroline

Es una plataforma de aprendizaje y trabajo virtual creada en el año 2001 en la Universidad Católica de Louvain en Bélgica (UCL). La plataforma Claroline Connect funciona con muchos recursos. Todos ellos tienen un modo de gestión idéntico para sus funciones básicas. También tienen características de tipo social como compartir y la capacidad de comentar. Estos se dividen en tres categorías:

- ✓ Los conceptos básicos.
- ✓ La comunicación.
- ✓ La pedagogía.

Cada espacio de la plataforma proporciona diversas herramientas que permiten crear contenidos de aprendizaje y gestión de actividades de formación. Esta plataforma posibilita: administrar una agenda con tareas y fechas límite, publicar anuncios que pueden ser enviados por correo electrónico, así como administrar foros públicos y privados (CLAROLINE 2017).

Sakai

Representa un enfoque fundamentalmente diferente del sistema de gestión del aprendizaje. A diferencia de otros sistemas "abiertos" disponibles en la actualidad, la dirección y el conjunto de características de Sakai se originan dentro de la educación superior para abordar las necesidades dinámicas de una comunidad académica mundial. La comunidad de código abierto de Sakai valora mucho la participación de sus contribuyentes, con educadores y desarrolladores de diversas instituciones que trabajan juntos para convertir grandes ideas en realidad para toda la comunidad de usuarios de Sakai.

Con un amplio conjunto de funciones, Sakai proporciona las herramientas necesarias para que los instructores, estudiantes, investigadores y líderes de proyectos tengan éxito en sus iniciativas en línea. Cada versión de Sakai se incluye con un conjunto estándar de herramientas básicas, que ofrecen capacidades de discusión, anuncios, mensajería, administración de archivos, entrega de tareas, evaluaciones y más. Contiene un calendario para mantener las actividades y eventos relacionados con el sitio, mantener la agenda actualizada para cada una de las materias y puede mostrar eventos de sitio, curso, grupo y usuario (Plataforma de Sakai 2014).

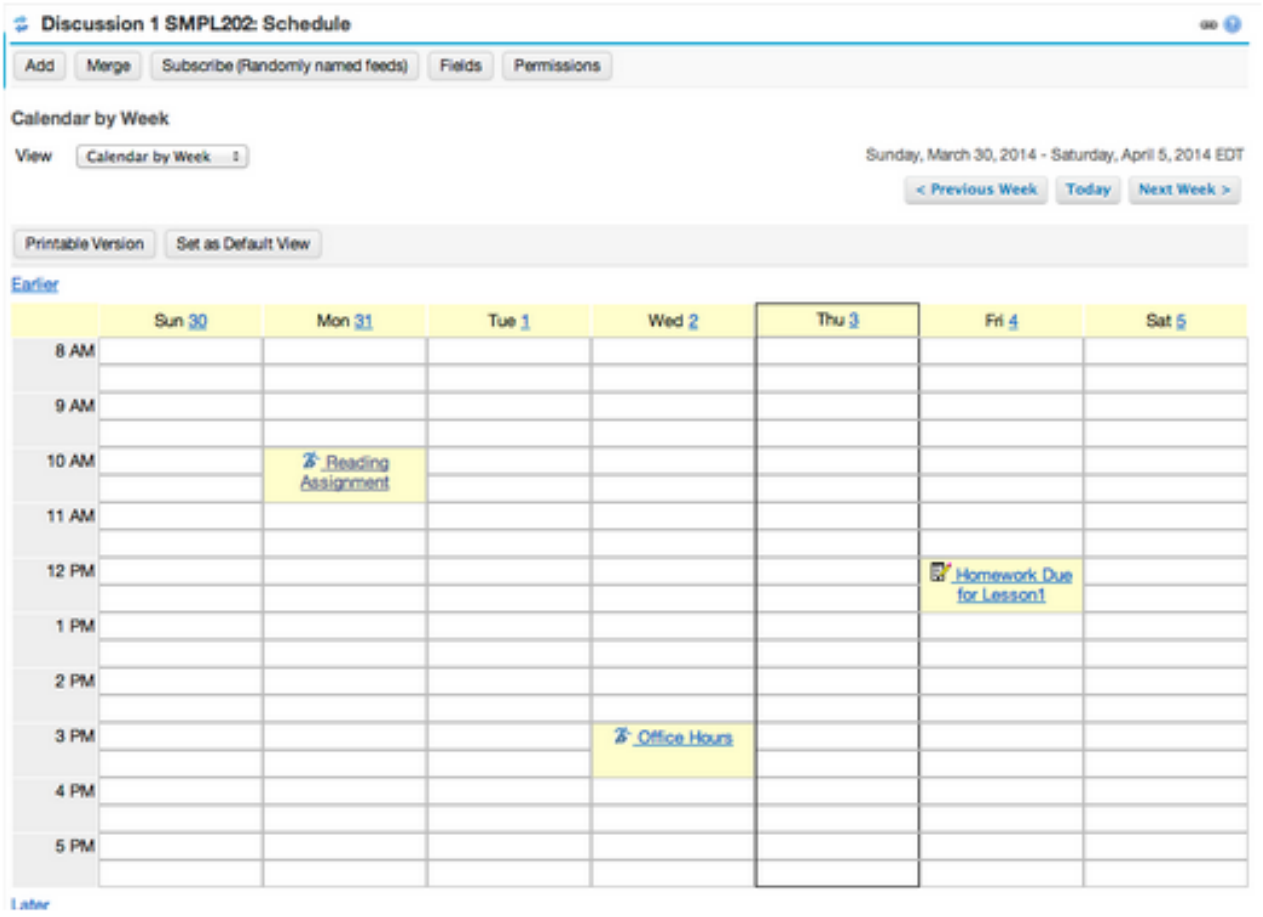


Figura 2 Agenda de Sakai *(Plataforma de Sakai 2014)*

1.2.3 Soluciones a nivel nacional

AprenDIST

Es una plataforma para la educación a distancia, es flexible y ajustable a gran cantidad de cursos y está diseñada para un alumnado con determinadas características. Su diseño es sencillo e intuitivo por lo que los usuarios de este sistema, pueden llegar a dominarlo con solo conocer un poco de navegación *web*. Esta plataforma ha comenzado a revolucionar el proceso de educación existente y a medida que aumenta su explotación, los impactos se van extendiendo hacia diversas entidades. Cuenta con un calendario en el cual los profesores y los estudiantes pueden gestionar sus eventos, algunas de sus funcionalidades son:

añadir un evento, exportar calendario, importar calendario y mostrar eventos del día actual (J.C. Sepulveda Peña 2005).

Al analizar los elementos que se han brindado anteriormente, se concluye que:

- ✓ En el desarrollo de la presente investigación se empleará la librería JavaScript FullCalendar para la visualización del calendario del módulo Agenda.
- ✓ Se tomó parte de la estructura propuesta por Adesigns Bundle como punto de partida para la organización e implementación de la solución.
- ✓ Con el análisis realizado a los calendarios de estas plataformas, se obtuvo una visión más amplia de las funcionalidades a tener presentes en el módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1.

1.3 Metodologías de desarrollo

Actualmente las metodologías de desarrollo de *software* pueden considerarse como una base necesaria para la ejecución de cualquier proyecto de desarrollo de *software*. Las mismas necesitan sustentarse en algo más que la experiencia de las capacidades de sus programadores y equipos de desarrollo. Estas metodologías son necesarias para poder realizar un proyecto profesional, así como poder desarrollar efectiva y eficientemente el *software.* En la actualidad existen una gran cantidad de metodologías para el desarrollo de *software*, separadas en dos grandes grupos; las metodologías tradicionales y las metodologías ágiles (Estaban Gabriel Maida, Julian Pacienzia. 2015).

1.3.1 Metodologías tradicionales

Las metodologías tradicionales se basan en las buenas prácticas dentro de la ingeniería del *software*, siguiendo un marco de disciplina estricto y un riguroso proceso de aplicación. Están basada en normas provenientes de estándares seguidos por el entorno de desarrollo, presenta grupos de desarrollo bastante grandes y posiblemente distribuidos, posee más artefactos y más roles que la metodología ágil. El cliente interactúa con el equipo de desarrollo mediante reuniones (Estaban Gabriel Maida, Julian Pacienzia. 2015).

Proceso Unificado de Desarrollo (RUP)

Es una metodología pesada para el desarrollo de *software* que constituye la metodología estándar más utilizada para el diseño, implementación y documentación de sistemas orientados a objetos. Establece un conjunto de flujos de trabajo adaptables al contexto y necesidades de cada organización.

Utiliza el Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) para preparar todos los esquemas de un sistema de *software*. Está basado en componentes, dirigida por casos de uso, centrada en la arquitectura, además de ser iterativa e incremental. Los casos de uso reflejan lo que los usuarios futuros necesitan y se representan a través de los requerimientos; los cuales guían el proceso de desarrollo a partir de los diferentes flujos de trabajo.

RUP cuenta con varias fases:

- ✓ Inicio: alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto, generar el ámbito de trabajo, el caso de negocio, síntesis de arquitectura posible y el alcance del proyecto.
- ✓ Elaboración: establecimiento de la línea base para la arquitectura del sistema y proporcionar una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, mitigando la mayoría de los riesgos tecnológicos.
- ✓ Construcción: completar el desarrollo del sistema basado en la línea base de la arquitectura. En otras palabras, lograr la funcionalidad operativa del *software*.
- ✓ Transición: garantizar que el *software* esté listo para entregarlo a los usuarios, y lograr la aprobación cuanto antes para liberar el producto al mercado (X Albaladejo 2012).

Es considerada una de las metodologías más importantes, debido a que en cada ciclo de iteración se exige el uso de artefactos. Además, se aplica a un proceso más controlado, con numerosas políticas o normas, lo cual proporciona una mayor estabilidad en el proceso de desarrollo del *software*.

1.3.2 Metodologías ágiles

Las metodologías ágiles representan una solución a los problemas que requieren una respuesta rápida en un ambiente flexible y con cambios constantes, haciendo caso omiso a la documentación rigurosa y los

métodos formales. Está basada en heurísticas[2] provenientes de prácticas de producción de código, presenta grupos de desarrollo bastante pequeños, poseen pocos artefactos y roles, además permite que el cliente sea parte del equipo de desarrollo (Estaban Gabriel Maida, Julian Pacienzia. 2015).

Programación Extrema (XP): es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Presenta como objetivo:

- ✓ Establecer las mejores prácticas de Ingeniería de *Software* en los desarrollos de proyectos.
- ✓ Mejorar la productividad de los proyectos.
- ✓ Garantizar la calidad del *software* desarrollado, haciendo que este supere las expectativas del cliente.

Tanto el análisis como el diseño son tareas muy importantes en XP, pero se realizan con un enfoque más ligero y transparente. El análisis es una parte fundamental, aquí se intentan recoger todas las necesidades del usuario. A partir de estas necesidades surgen las “historias de usuarios” que serán el paso inicial para comenzar a desarrollar el sistema.

Esta metodología suele estar especialmente orientada a proyectos cortos, aquellos en los cuales los equipos de desarrollo son pequeños, con plazos reducidos, requisitos volátiles, y basados en nuevas tecnologías, y en los cuales el cliente forma parte del equipo de desarrollo («XP (Programación Extrema)» 2017).

Proceso Unificado Ágil

El Proceso Unificado Ágil de Scott Ambler (AUP por sus siglas en inglés) es una versión simplificada del Proceso Unificado de Desarrollo (RUP). Este describe de una manera simple y fácil de entender la forma

[2] Es la capacidad que tiene el hombre de crear o inventar algo, con la finalidad de proporcionar estrategias que ayuden a la resolucion de un problema.

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:

- ✓ Modelado ágil.
- ✓ Desarrollo Dirigido por Pruebas (*test driven development* –TDD en inglés).
- ✓ Refactorización de Base de Datos para mejorar la productividad.
- ✓ Gestión de cambios ágil.

Variación de AUP-UCI

No existe una metodología de *software* universal, debido a que toda metodología debe ser adaptada a cada proyecto. En la UCI se decide realizar una variación de la metodología AUP de forma que se adapte al ciclo de vida definido para la actividad productiva. Entre los objetivos de una metodología de *software* está aumentar la calidad del *software* que se produce, de ahí la importancia de aplicar buenas prácticas, para ello la UCI emplea el Modelo CMMI-DEV v1.3. Estas prácticas se centran en el desarrollo de servicios de calidad y productos.

De las 4 fases que presenta AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, modificando el objetivo de la misma, se unen las restantes 3 fases de AUP en una sola, descrita como Ejecución y se agrega la fase de Cierre.

A continuación, se realiza una descripción detallada de los escenarios que presenta la metodología AUP en su variación UCI.

Escenario No 1: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los Caso de Uso del Negocio (CUN) muestran como los procesos son llevados a cabo por personas y los activos de la organización.

Escenario No 2: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

Escenario No 3: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

Escenario No 4: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando 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, ya que una Historias de Usuarios (HU) no deben poseer demasiada información (Tamara Rodríguez Sánchez 2015).

Metodología seleccionada

Dentro de las metodologías de desarrollo de *software* vistas anteriormente, se destacó la adaptación de AUP que se propone para la actividad productiva de la UCI, consiguiendo estandarizar el proceso de desarrollo de *software*, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Para el desarrollo de la solución propuesta, en la disciplina de Requisitos, el autor de la presente investigación decide utilizar el escenario número 4. Esta selección se realiza, teniendo en cuenta, que el proyecto está bien definido, el cliente en todo momento estará junto al equipo de desarrollo para convenir los detalles de los requisitos. Además, a pesar de la complejidad mostrada por el proyecto, este no es extenso, posibilitando el uso de las Historias de Usuarios.

1.4 Lenguajes y técnicas de programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana («Lenguaje de programación» 2018).

Marco de trabajo Xalix

Como parte de la arquitectura de referencia para el desarrollo de aplicaciones en el Centro FORTES y con el propósito de disminuir la diversidad tecnológica de soluciones, se desarrolló un marco de trabajo en las Líneas de Producción de *Software* (LPS) llamado Xalix. Actualmente representa el marco de trabajo sobre el que se está desarrollando la Plataforma Educativa Xauce ZERA v2.1.

Las tecnologías utilizadas por este marco de trabajo son:

- ✓ Gestor de bases de datos: PostgreSQL.
- ✓ Framework de desarrollo: Symfony 2.7.16.
- ✓ Lenguaje de programación para el servidor: php 7.0.
- ✓ Lenguaje de programación para el cliente: HTML 5 (*HyperText Markup Language*) y JavaScript.
- ✓ Librería CSS (*Cascading Style Sheets*): Boostrap 3.0.0 (compilado para el marco de referencia).
- ✓ Componentes nativos de Xalix (en desarrollo): Gestión de autenticación, Gestión de servicios *web*, Temas y Panel de administración.

1.4.1 Lenguajes del lado del cliente

HTML 5: es el Lenguaje para el Formato de Documentos de Hipertexto. Es un sistema para definir tipos de documentos estructurados y lenguajes de marcas para representar dichos documentos. HTML no es un lenguaje de programación como C++, sino un sistema de etiquetas que no se compila por lo que si existe algún error este no lo señalará, sino que representará en el navegador lo que interprete del código erróneo.

Estas etiquetas permiten representar textos con diversos estilos, así como aplicar diferentes efectos a los párrafos que permitan que el entorno del sitio *web* sea agradable para el usuario. Además, se pueden incorporar a las páginas multimedia como sonidos, imágenes y videos. También con HTML se pueden crear vínculos a otras páginas y pueden incluir los *scripts* que permiten más interactividad en la navegación, como los de *JavaScript*. El propio *World Wide Web Consortium* (W3C) define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global".

Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas *online* y banca electrónica (Manual Práctico de HTML. Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, España. 2015).

CSS 3.0: es un lenguaje de hojas de estilo creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML, es decir páginas *web*. CSS es la mejor forma de separar los contenidos y su presentación, es imprescindible para crear páginas *web* complejas. Separar la

definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Cuando se crea una página *web* con HTML se enmarcan los contenidos como son: párrafo, titular, texto destacado, tabla, lista de elementos, entre otros. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, posición de cada elemento dentro de la página, entre otros (J.E. Pérez 2014b).

Ajax: es un acrónimo de *Asynchronous JavaScript* + XML, que se puede traducir como "JavaScript asíncrono + XML". En el artículo publicado por Jesse James Garrett en el 2005 define Ajax como: "Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes".

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y *Document Object Model* (DOM). Las tecnologías que forman Ajax son XHTML (*eXtensible HyperText Markup Language*) y CSS para crear una presentación basada en estándares, DOM para la interacción y manipulación dinámica de la presentación. XML *(eXtensible Markup Language)*, XSLT (*Extensible Stylesheet Lenguaje Transformations*) y JSON (*JavaScript Object Notation*) para el intercambio y la manipulación de información, XMLHttpRequest para el intercambio asíncrono de la información y JavaScript para unir todas las demás tecnologías. Desarrollar aplicaciones AJAX requiere un conocimiento adecuado de todas y cada una de las tecnologías anteriores (J.E. Pérez 2014a).

JavaScript: técnicamente, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Se utiliza principalmente para crear páginas *web* dinámicas, una página *web* dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos (J.E. Pérez 2014c).

1.4.2 Lenguaje del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor *web*, justo antes de que se envíe la página a través de *Internet* al cliente. Las páginas que se ejecutan en el lado del servidor pueden efectuar consultas a la base de datos y otras tareas para crear la página final que es lo que verá el cliente, la cual contiene solamente código HTML, como resultado de la ejecución de la página PHP.

PHP 7.0: es un lenguaje de *script* interpretado, utilizado para la generación de páginas *web* dinámicas. PHP es un acrónimo recursivo que significa *Hypertext Pre-processor*. La mayor parte de su sintaxis ha sido tomada de C++, Java y Perl con algunas características específicas de sí mismo. El objetivo que persigue este lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas *web*. No es un lenguaje de marcas como podría ser HTML o XML. Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas *web* dinámicas.

Entre sus ventajas principales se pueden encontrar:

- ✓ El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.
- ✓ Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL *Server*, entre otras.
- ✓ Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- ✓ Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes (Manuel Torres Remón 2012).

1.4.3 *Frameworks* y librerías para el desarrollo

El concepto *framework* es ampliamente empleado en muchos ámbitos del desarrollo de sistemas de *software*, no solo para aplicaciones *web*. Se pueden encontrar *framework* para el desarrollo de aplicaciones médicas, visualización gráfica, desarrollo de juegos y otros. En general el término *framework*, se refiere a una estructura de *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Los objetivos principales que persigue un *framework* son: acelerar el

proceso de desarrollo, reutilizar el código ya existente y promover buenas prácticas de desarrollo con el uso de patrones (Javier J. Gutiérrez 2010).

Symfony: es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones *web*, simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, es un *framework* que proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Es un *framework* facilitador de la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja.

Symfony está desarrollado completamente con PHP. Ha sido probado en numerosos proyectos reales y se utiliza en sitios *web* de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL *Server* de *Microsoft* (F. Potencier 2010).

Entre sus principales características están:

- ✓ Fácil de instalar y configurar en la mayoría de plataformas.
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sigue la mayoría de las mejores prácticas y patrones de diseño para la *web*.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Bootstrap: es un *framework* basado en HTML y CSS, creado por Twitter y liberado en 2012. Desde entonces ha ganado muchos adeptos, hasta el nivel de contar con la mayor comunidad de *Github* del mundo. Este *framework* ayuda a agilizar la creación de la interfaz de una página *web*. Su uso permite que el sitio *web* sea adaptable a la pantalla del dispositivo con el que se accede, ya sea un ordenador, *tablet*, *smartphone* o televisión (Niska C. 2014).

JQuery: es una librería de JavaScript creado inicialmente por John Resig que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología Ajax a páginas *web*. Al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Con las

funciones propias de esta librería se logran resultados en menos tiempo y espacio (Ortiz Batista Y, López Reinoso Y, Medina León Y, Gonce Fernández S, Batard Lorenzo D, Gulín González 2010).

1.5. Lenguaje de Modelado

Es un conjunto de símbolos estandarizados usados para diseñar un sistema orientado a objetos. Su utilización depende generalmente de la combinación de una metodología de desarrollo de *software*, para obtener una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores *(Jacobson I., Booch G 2018)*.

1.5.1 Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de *software*. UML es utilizado para el modelado completo de sistemas, tanto en el diseño de los sistemas *software* como para la arquitectura *hardware* donde se ejecuten.

Figura 3 Logo de UML *(E. Hernández Orallo 2014)*

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos y otros, hasta la implementación y configuración con los diagramas de despliegue. Este modelado visual está diseñado para que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

El lenguaje UML presenta varios objetivos, a continuación, se exponen algunos de ellos sintetizados en sus funciones:

- ✓ Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- ✓ Especificar: UML posibilita especificar cuáles son las características de un sistema antes de su construcción.
- ✓ Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión (E. Hernández Orallo 2014).

1.6 Herramientas CASE

CASE (*Computer Aided Software Engineering*), su traducción al español es: Ingeniería de *Software* Asistida por Computación. Se puede definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software*. A continuación, se ofrece una breve descripción de la herramienta CASE que se va a utilizar.

1.6.1 Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Soporta todos los diagramas UML y el diagrama Entidad-Relación, ayudando a una rápida construcción de aplicaciones con calidad y a un menor coste. Posee una interfaz amigable y se puede modelar en varios idiomas. Es una herramienta colaborativa, pues soporta múltiples usuarios trabajando sobre el mismo proyecto. La documentación del proyecto puede ser generada automáticamente en varios formatos (*web* o .pdf), y permite control de versiones. Soporta la notación UML 2.1, ingeniería inversa, generador de informes, editor de figuras, integración con MS Visio, *plugin*, integración IDE con Visual Studio, IntelliJ IDEA, Eclipse, NetBeans y otros.

Entre sus ventajas se encuentran:

- ✓ Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requisitos y de especificación de componentes.

- ✓ Tiene disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- ✓ Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo *Visio* y *Rational Rose*.
- ✓ Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir del código.
- ✓ Brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas («Visual Paradigm» 2012).

1.7 Sistema Gestor de Bases Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos (Angel Cabo Yera 2007).

MYSQL: es un gestor de base de datos sencillo de usar e increíblemente rápido. También es uno de los motores de base de datos más usados en *Internet*, la principal razón de esto es que es libre para aplicaciones no comerciales.

Las características principales de MySQL son:

- ✓ Es un sistema de base de datos relacional. Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL (*Structured Query Language*).
- ✓ Es una base de datos muy rápida, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una de las bases de datos más usadas en *Internet* (MySQL-Hispano.org. 2017).

PostgreSQL: es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) *Open Source*. Es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la

herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Principales características:

- ✓ Implementación del estándar SQL92/SQL99.
- ✓ Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc.
- ✓ Incorpora una estructura de datos *array*.
- ✓ Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores.
- ✓ Soporta el uso de índices, reglas y vistas.
- ✓ Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos (Daniel Pecos Martinez 2017).

El sistema de base de datos seleccionado para el desarrollo de la aplicación es PostgreSQL 9.5 debido a que es considerado entre los ORDBMS de licencia libre como uno de los más completos por poseer todas las funcionalidades de un gestor de bases de datos privativo. Entre otras de las ventajas determinantes es su documentación bien detallada y organizada. Además, es el gestor de base de datos utilizado y definido por la Plataforma Educativa Xauce ZERA v2.1.

1.8 Herramientas de Desarrollo

El desarrollo de *software* es un proceso complejo que requiere de herramientas competentes para lograr resultados satisfactorios. En el presente epígrafe se hace referencia a la herramienta Empleada para el desarrollo de la propuesta de solución y las características por la que fue seleccionada. Se determinó el uso del Entorno de Desarrollo Integrado(IDE) NetBeans versión 8.0.

NetBeans IDE (Entorno de Desarrollo Integrado): es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans está formado por un IDE de código

abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones *web*, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++.

Principales características:

- ✓ Lenguajes *Web*: HTML, CSS, JavaScript
  - Reestructuración y búsqueda de usos para CSS y lenguajes parecidos a HTML.
  - Reestructuración de estilos en línea de CSS.
- ✓ PHP o Compatibilidad con PHP Zend Framework.
  - Compatibilidad con PHP *Symfony*.
- ✓ General o Corrector ortográfico en el Editor.
  - Brinda reportes sobre fallos.
  - Compatibilidad para diversos servidores de trabajo en equipo, basados en Kenai (por ejemplo: kenai.com y netbeans.org) (Oracle Corporation 2017).

1.9 Servidor web

Un servidor web es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación del lado del cliente. Mientras que comúnmente se utiliza la palabra servidor para hacer referencia a una computadora con un *software* servidor instalado, en estricto rigor un servidor es el *software* que permite la realización de las funciones antes descritas (Palma Pérez N 2013).

Nginx: es un servidor *web* y proxy inverso de código abierto ligero de alto rendimiento, creado por Igor Sysoev y lanzado en octubre de 2004 bajo la Licencia BSD[3] simplificada, que incluye servicios de correo electrónico con acceso al internet Message Protocol (IMAP)[4] y al servidor Post Office Protocol (POP)[5].

[3] Licencia de software otorgada principalmente para la distribución de software Berkeley.
[4] Protocolo de aplicación que permite el acceso a mensajes almacenados en un servidor de Internet.
[5] Protocolo utilizado en clientes para obtener los mensajes de correo electrónico almacenados en un servidor remoto.

Una de las principales razones para utilizar este servidor web es que se trata de un *software* que es asíncrono, a diferencia de Apache que está basada en procesos. La ventaja principal de ser asíncrono, es su escalabilidad. En un sistema basado en procesos, cada conexión simultánea requiere de un hilo, lo que puede llevar a sobrecargar el servidor, mientras que en un servidor asíncrono se gestionan las peticiones en muy pocos hilos, reduciendo las posibilidades de sobrecarga en el servidor *(Corona S 2008)*.

1.10 Sistema de control de versiones

El control de versiones (VCS) es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

El mismo permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más. Usar un VCS también significa que, si se dañan o pierden archivos, se pueden recuperar fácilmente *(Chacón S 2014)*.

Git: es un sistema de control de versiones distribuido diseñado por Linus Torvalds en el año 2005. Surge como alternativa a BitKeeper[6], un control de versiones privativo que usaba en ese entonces para el kernel.

Es liberado bajo una licencia GNU GPLv2[7] y su última versión estable fue publicada a inicios de abril de 2017. Por sus disímiles ventajas, se ha convertido en uno de los más usados alrededor del mundo, puesto que no depende de acceso a la red o un repositorio central; además, está enfocado en la velocidad, uso práctico y manejo de proyectos grandes *(Chacón S 2014)*.

[6] Es un sistema de control de versiones distribuido para el código fuente de los programas producidos a partir de BitMover Inc.
[7] Licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto que garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software.

Conclusiones parciales

Teniendo en cuenta el estudio realizado previamente se concluye:

- ✓ Los conceptos asociados al dominio del problema enmarcados en este capítulo, enriquecieron el conocimiento sobre la gestión de eventos en plataformas educativas.
- ✓ Las soluciones similares estudiadas no satisfacen las necesidades asociadas a los problemas que presenta la Plataforma Educativa Xauce ZERA con respecto a la gestión de eventos, por lo que se determina el desarrollo de un módulo que cumpla con estas necesidades.
- ✓ Para la elaboración de la propuesta de solución fueron seleccionados los siguientes lenguajes, tecnologías y herramientas: HTML, PHP, CSS, Symfony, JQuery, Bootstrap, NetBeans, Nginx, Git y JavaScript.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se procede a realizar la descripción de la propuesta de solución. Se elabora el modelo de dominio con la correspondiente descripción de cada uno de sus elementos. Se describen las características principales del sistema mediante los requisitos funcionales y no funcionales, los cuales son modelados en las historias de usuario. Por último, se obtienen los artefactos generados durante la fase de análisis y diseño de la propuesta de solución.

2.1 Modelo de dominio

El Modelo de Dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocio al cual el sistema va a servir. Permite comprender y describir las clases más importantes dentro del contexto del sistema. No es una descripción del diseño del *software* es una representación visual de las clases conceptuales del mundo real en un dominio de interés («Visual Paradigm» 2012).

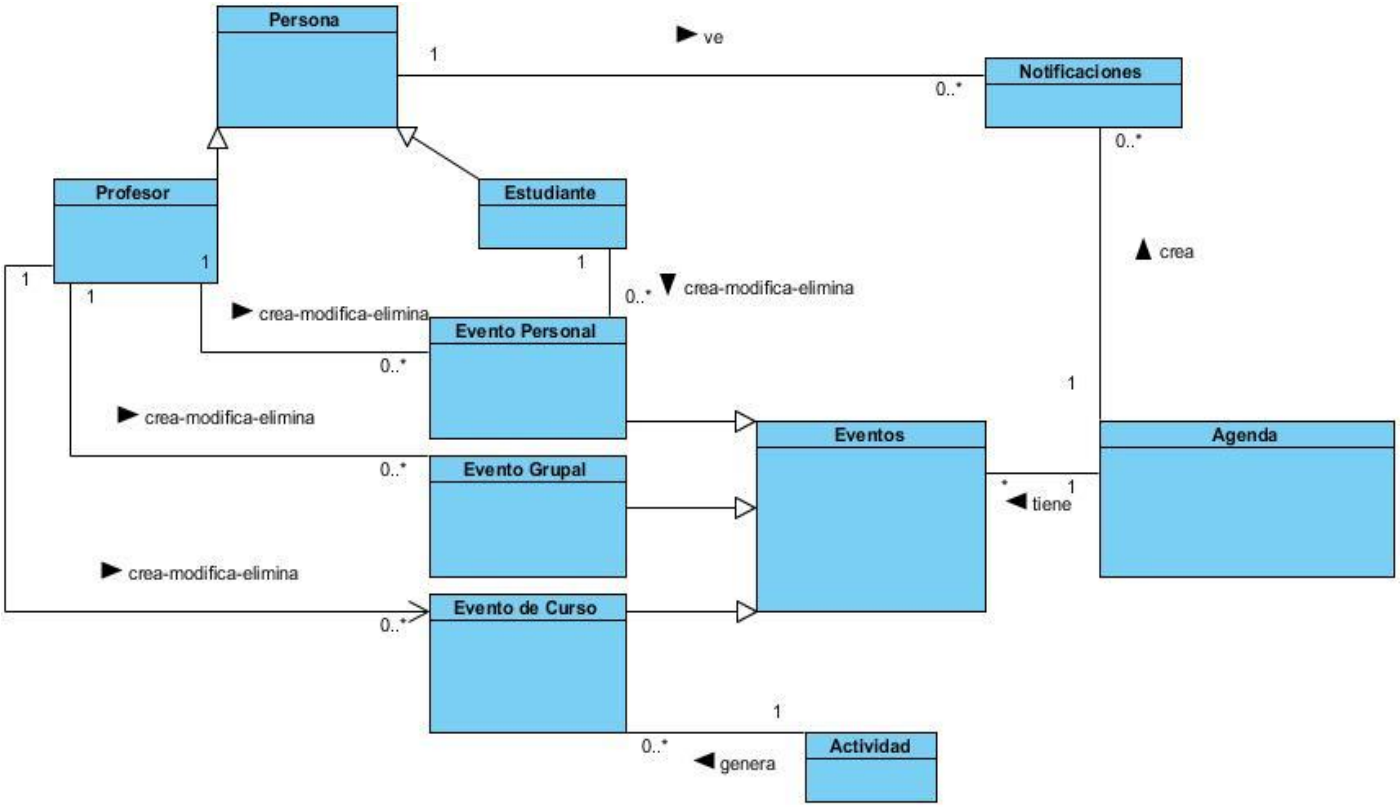


Figura 4 Diagrama de Modelo de Dominio

2.1.1 Glosario de términos del modelo de dominio

Persona: hace referencia a la persona que accede a la agenda y gestiona los eventos dentro de ella.

Profesor: es la persona que gestiona los eventos en la agenda ya sean personales, grupal o de curso.

Estudiante: es la persona que gestiona eventos en la agenda, pero sólo los eventos personales.

Agenda: es el elemento del sistema donde se gestionan y organizan, a través de un calendario, todos los eventos creados ya sean personales, de grupo o de curso.

Notificaciones: son alertas de correo electrónico o mensajes de información generadas por la agenda, a través del sistema de notificaciones de la plataforma, para informar al usuario acerca de la gestión de eventos.

Eventos: son hechos programados gestionados por las personas ya sean profesores o estudiantes y estos pueden ser personales, grupales o de curso.

Evento personal: es un hecho programado gestionado por el profesor y el estudiante, sólo son vistos por el usuario que lo creó.

Evento grupal: es un hecho programado gestionado por el profesor y asociado a un grupo dentro de un curso. Este tipo de evento sólo es mostrado a los miembros del grupo para el que fue creado.

Evento de curso: es un hecho programado gestionado por el profesor y asociado a un curso dentro del sistema. Este tipo de evento solo es mostrado a los miembros del curso para el que fue creado.

Actividad: es una labor gestionada por el profesor en el sistema la cual genera eventos de tipo de curso. Estas se clasifican en Tareas, Cuestionarios y Foros.

2.2 Descripción del sistema propuesto

Para dar solución a los objetivos anteriormente planteados se determina el desarrollo de un módulo para la Plataforma Educativa Xauce ZERA v2.1. El módulo constará de una agenda, la cual permitirá a las personas gestionar sus eventos ya seas personales, de grupo y de curso.

Cada evento estará conformado por: nombre, descripción, fecha de inicio, fecha de fin y tipo de evento. En particular los estudiantes solo podrán gestionar eventos personales y los profesores podrán gestionar eventos personales, de curso y de grupo. Estos dos últimos son creados por el profesor para todos sus estudiantes, una vez creado o modificado el evento se enviará una notificación al correo de los implicados

en el evento. Permitirá exportar la agenda para su uso fuera de la plataforma en el formato ICS[8] y listar todos los eventos que sean creados por una persona. También se podrá buscar los eventos que el usuario desea, además se mostrará la agenda según la selección que se elija, puede ser por días, semanas o meses.

2.3 Especificación de requisitos

El módulo de Agenda tendrá que cumplir con una serie de condiciones y funcionalidades para dar solución a la necesidad de una agenda en la Plataforma Educativa Xauce ZERA v2.1 para la gestión de eventos. A continuación, se definirán dichas condiciones y funcionalidades en los siguientes requisitos clasificados como funcionales y no funcionales.

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) especifican las capacidades o condiciones que el sistema debe ser capaz de cumplir. Son los que definen una función dentro de un sistema de *software*. Describen la interacción entre el sistema y su ambiente independientemente de su implementación (I. Garcerant 2014). A continuación, se muestra el listado de los requisitos funcionales obtenidos:

RF1: Incluir evento. El sistema debe permitir dentro de un curso, al usuario con el rol profesor, crear un evento de tipo personal, de curso y de grupo y con rol de estudiante, sólo crear eventos personales. Fuera del curso el usuario sólo puede crear eventos personales.

RF2: Editar evento. El sistema debe permitir dentro de un curso, al usuario con rol profesor, editar un evento de tipo personal, de curso y de grupo y con rol de estudiante, sólo editar eventos personales. Fuera del curso el usuario sólo puede editar un evento personal.

RF3: Eliminar evento. El sistema debe permitir dentro de un curso, al usuario con rol profesor, eliminar un evento de tipo personal, de curso y de grupo y con rol de estudiante, sólo eliminar eventos personales. Fuera del curso el usuario sólo puede eliminar un evento personal.

RF4: Mostrar evento. El sistema debe permitir dentro de un curso, al usuario ver los datos de un evento de tipo personal, de curso o de grupo. Fuera del curso, sólo puede ver los datos de un evento personal.

RF5: Listar eventos. El sistema debe permitir dentro de un curso, al usuario listar los eventos de tipo personal**,** de curso o de grupo. Fuera del curso, sólo puede listar los eventos personales.

[8] Es un formato universal usado para guardar información relacionado con calendarios.

RF6: Buscar evento. El sistema debe permitir dentro de un curso, al usuario buscar los eventos de tipo personal**,** de curso o de grupo. Fuera del curso, puede buscar sólo los eventos personales.

RF7: Notificar evento. El sistema debe permitir notificar a los miembros de un grupo o un curso que un evento fue creado.

RF8: Exportar agenda. El sistema debe permitir exportar la agenda con los eventos asociados al usuario ya sea de forma directa o indirecta a través de un curso o un grupo.

RF9: Mostrar agenda. El sistema debe permitir que el usuario observe los eventos de su agenda en correspondencia al rol y al contexto[9].

RF10: Cambiar apariencia de agenda. El sistema debe permitir la selección de la apariencia de su calendario según su gusto, la misma puede ser en día, semana o mes.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) se definen como las cualidades o propiedades que el *software* debe tener. Son aquellos que imponen restricciones en el diseño, la implementación, diseño y estándares de Calidad (J. P. Gomes Gallego, J.A. Galves 2010). Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

Requisitos de seguridad y confiabilidad:

RNF1: Garantizar que la información sea editada únicamente por quien está autorizado y posea permisos para ello.

Requisitos de usabilidad

RNF2: El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente *web* en sentido general.

2.4 Historias de usuarios

Las historias de usuario (HU) son tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento estas pueden romperse o reemplazarse por otras más específicas y generales, añadirse nuevas, además de poder ser modificadas. Cada historia de usuario es

[9] Contexto en el que se visualiza el calendario, pudiendo estar este fuera o dentro de un curso.

lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en el tiempo estimado (Patricio Letelier 2006).

Las HU deben estar elaboradas según el modelo que propone AUP en su variación UCI, este modelo se presenta en la siguiente tabla.

Tabla 1 Descripción de la HU

Número: Número de la HU.	**Nombre del requisito:** El nombre de la UH que la identifica.
Programador:	**Iteración Asignada:** Número de iteraciones.
Prioridad: Cuán difícil es para el desarrollador (Alta, Media, Baja).	**Tiempo Estimado:** Tiempo en día que se le asignará.
Riesgo en Desarrollo: Hacer referencia a los riesgos identificados en plan de riesgos.	**Tiempo Real:** Tiempo real dedicado a la realización de la HU en semanas.
Descripción: Descripción de la HU detallando las operaciones del usuario y opcionalmente la respuesta del sistema.	
Observaciones: Información de interés, como glosarios o información de los usuarios .	
Prototipo de interfaz:	

A continuación, se muestran aquellas HU de mayor importancia para el cliente. Las demás HU se muestran en el anexo 1.

Tabla 2 Descripción de la HU incluir evento

Número: 1	**Nombre del requisito:** Incluir evento

Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

El sistema debe permitirle al usuario con el rol profesor crear un evento de tipo personal, de curso y grupo y al usuario con rol de estudiante, sólo crear eventos de tipo personal.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir un evento es necesario:

- Tener en cuenta los siguientes datos: nombre, descripción, fecha de inicio, fecha de fin y tipo de evento.

- Estar autenticado en el sistema.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Nombre: campo de texto de carácter obligatorio y tiene un máximo de hasta 255 caracteres de longitud.

Descripción: campo de texto que permite escribir una breve descripción del evento.

Fecha de inicio: campo de cuadro combinado que puedes elegir la fecha en que se quiere que se publique el evento. Representa un campo de carácter obligatorio.

Fecha de fin: campo de cuadro combinado que permite elegir la fecha en que se quiere que culmine el evento.

Tipo de evento: campo de carácter obligatorio para la selección del tipo de evento.

4- Flujo de la acción a realizar:

- Para incluir un evento el usuario selecciona la opción de Crear evento, introduce los datos imprescindibles y elige el tipo de evento.

- Luego selecciona la opción Guardar y el sistema muestra un mensaje de que se ha realizado la acción satisfactoriamente. Una vez creado un evento de tipo curso o grupo el sistema envía una notificación (ver HU_Notificar evento.docx). Si algún campo está en blanco se señalarán los campos dando la posibilidad al usuario de realizar nuevamente la acción y si la fecha de fin es menor que la fecha de inicio se señalara el campo dando la opción de cambiar la fecha para realizar nuevamente la acción.

- Si selecciona Cancelar regresará a la vista anterior.

Observaciones: Estar autenticado en el sistema.

Prototipo de interfaz:

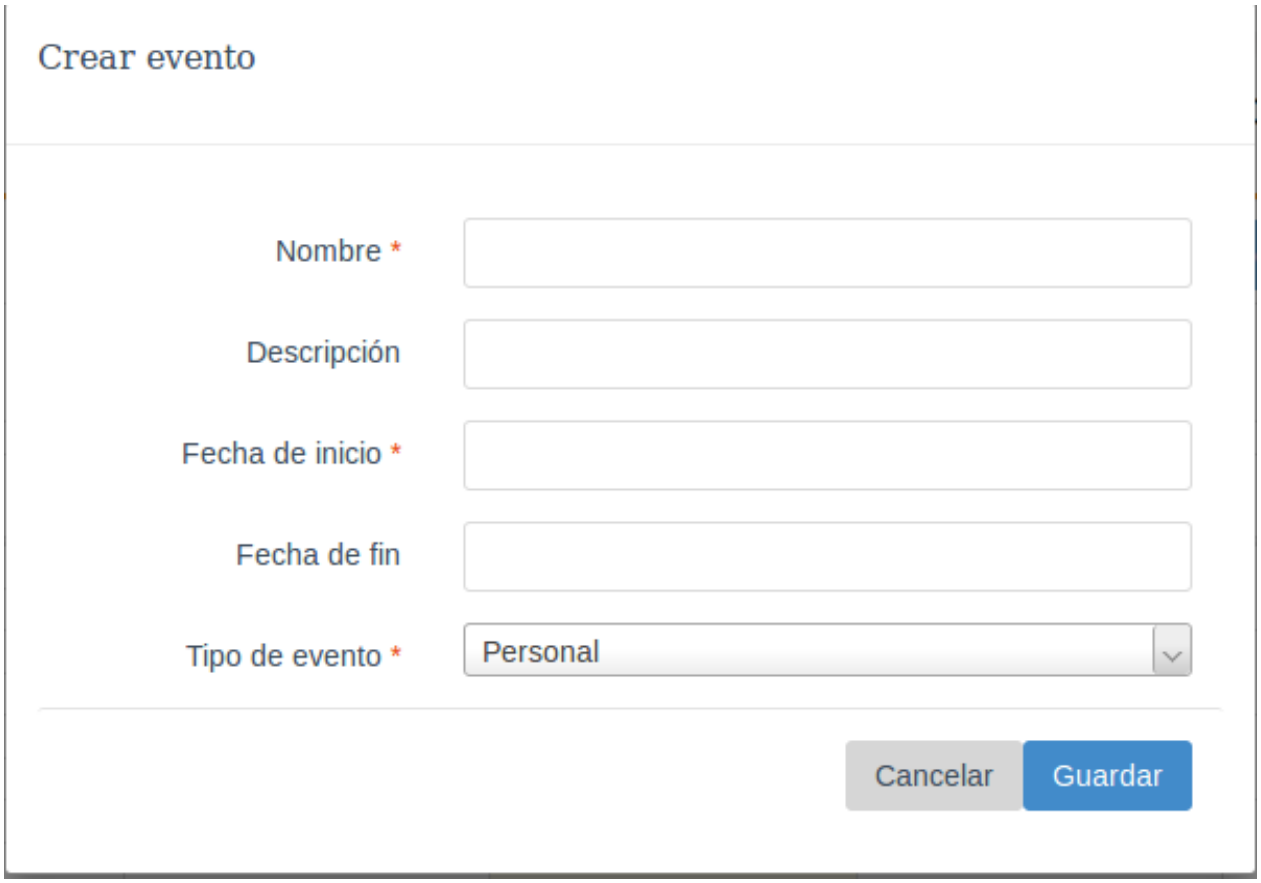


Tabla 3 Descripción de la HU editar evento

Número: 2	**Nombre del requisito:** Editar evento
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

El sistema debe permitirle al usuario con el rol profesor editar un evento de tipo personal, de curso y grupo y al usuario con rol de estudiante, sólo editar eventos de tipo personal.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para modificar los datos de un evento es necesario:

- Tener en cuenta los siguientes datos: nombre, descripción, fecha de inicio, fecha de fin.

- Estar autenticado en el sistema.

- Debe existir en el sistema al menos un evento que haya sido creado por él.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Nombre: campo de texto de carácter obligatorio y tiene un máximo de hasta 255 caracteres de longitud.

Descripción: campo de texto que permite escribir una breve descripción del evento.

Fecha de inicio: campo de cuadro combinado que puedes elegir la fecha en que se quiere que se publique el evento. Representa un campo de carácter obligatorio.

Fecha de fin: campo de cuadro combinado que permite elegir la fecha en que se quiere que culmine el evento.

Tipo de evento: campo de carácter obligatorio para la selección del tipo de evento.

4- Flujo de la acción a realizar:

- El sistema debe permitir modificar un evento seleccionando el evento a modificar en el calendario y posteriormente la opción Editar.

- Luego modifica los campos que desea y selecciona la opción Guardar, mostrándose así un mensaje de que el evento fue modificado de forma correcta. A la hora de editar el evento si selecciona una fecha de fin menor que la fecha de inicio, se señalarán los campos dando la posibilidad al usuario de realizar nuevamente la acción.

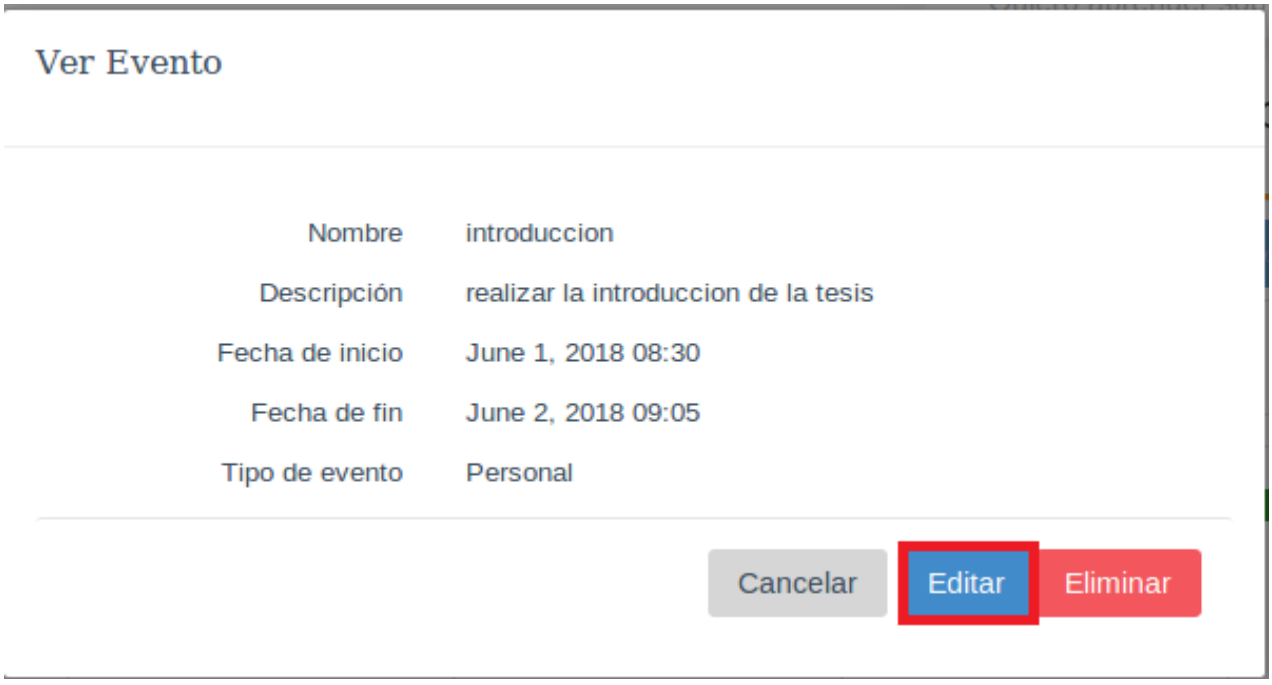
- Si selecciona la opción Cancelar regresará a la vista anterior.

Observaciones:

- Estar autenticado en el sistema.

- El tipo de evento no se puede editar una vez creado.

Prototipo de interfaz:



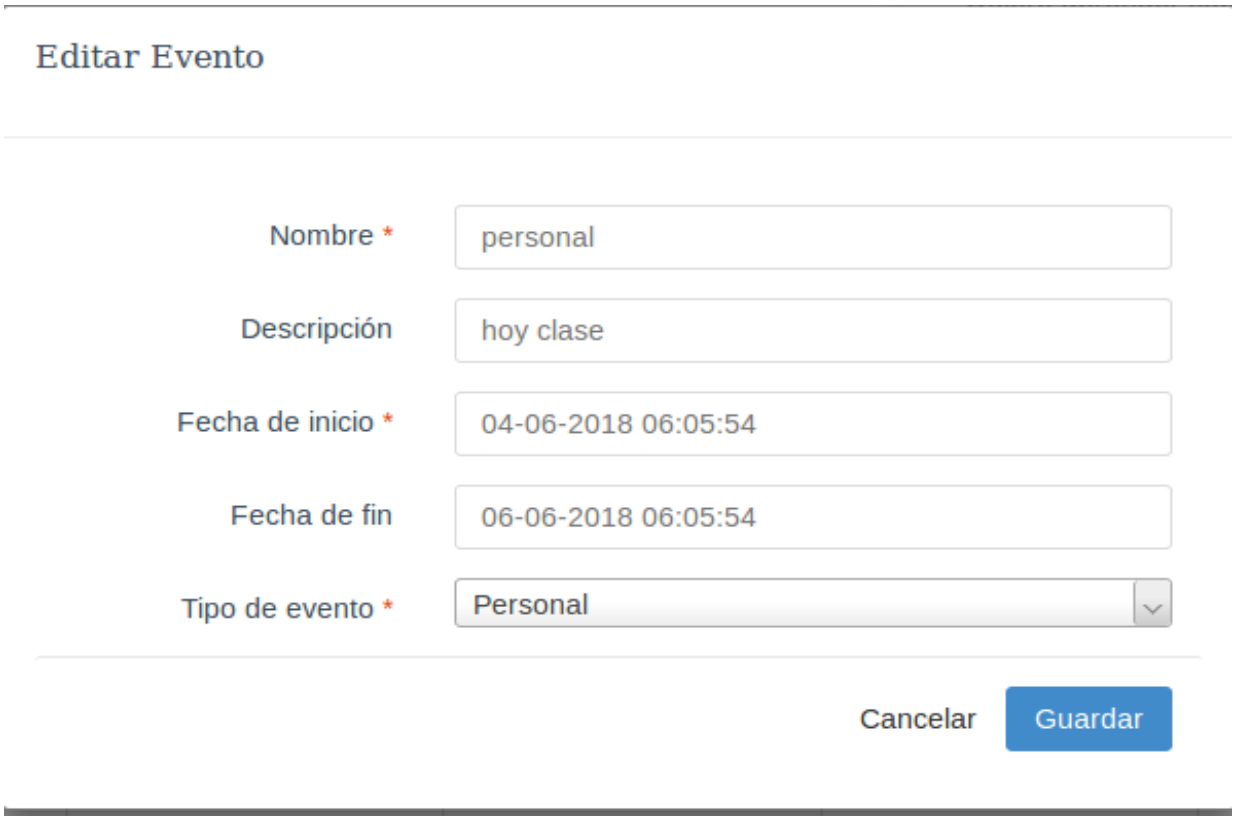


Tabla 4 Descripción de la HU mostrar evento

Número: 4	**Nombre del requisito:** Mostrar datos del evento
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:**	

El sistema debe permitirle al usuario consultar los datos de un evento de tipo personal, de curso y grupo.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para ver detalles de un evento es necesario:

- Estar autenticado en el sistema.

- Debe existir en el sistema al menos un evento que sea accesible por él.

3- Flujo de la acción a realizar:

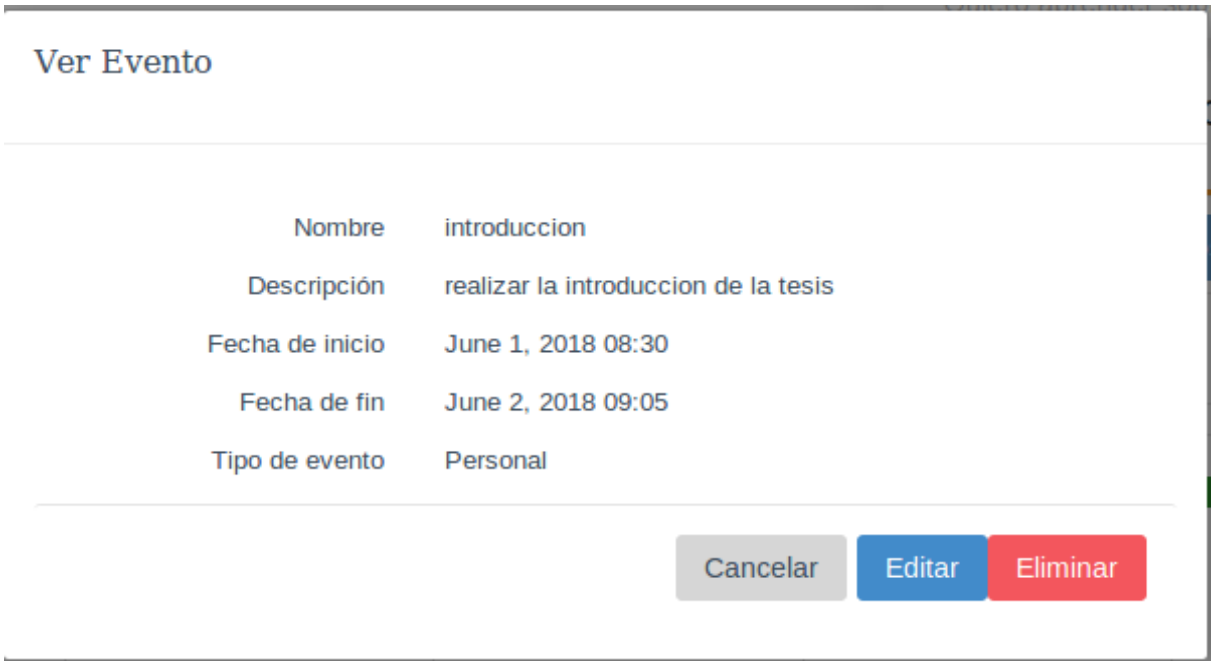
- Inicialmente se muestra el calendario con los eventos que han sido incluidos en el sistema y posteriormente el sistema debe permitir seleccionar el evento que se desee ver.

- Una vez seleccionado un evento, se podrán ver los datos que este contiene.

- Si selecciona la opción Cancelar regresará a la vista anterior.

Observaciones: Estar autenticado en el sistema.

Prototipo de interfaz:



2.5 Patrones Arquitectónicos

Los patrones arquitectónicos están relacionados con la interacción de objetos dentro o entre niveles arquitectónicos. Se utilizan para expresar una estructura de organización base o esquema para un *software*. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos (Mejia Gomez 2013). *Symfony* implementa en su estructura la combinación del patrón arquitectónico Modelo Vista Controlador, a continuación, se describen sus principales características.

- ✓ **Patrón Arquitectónico Modelo Vista Controlador (MVC)**: La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP (*Hypertext Transfer Protocol*), consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Potencier, Fabien; Zaninotto, Francois 2018).

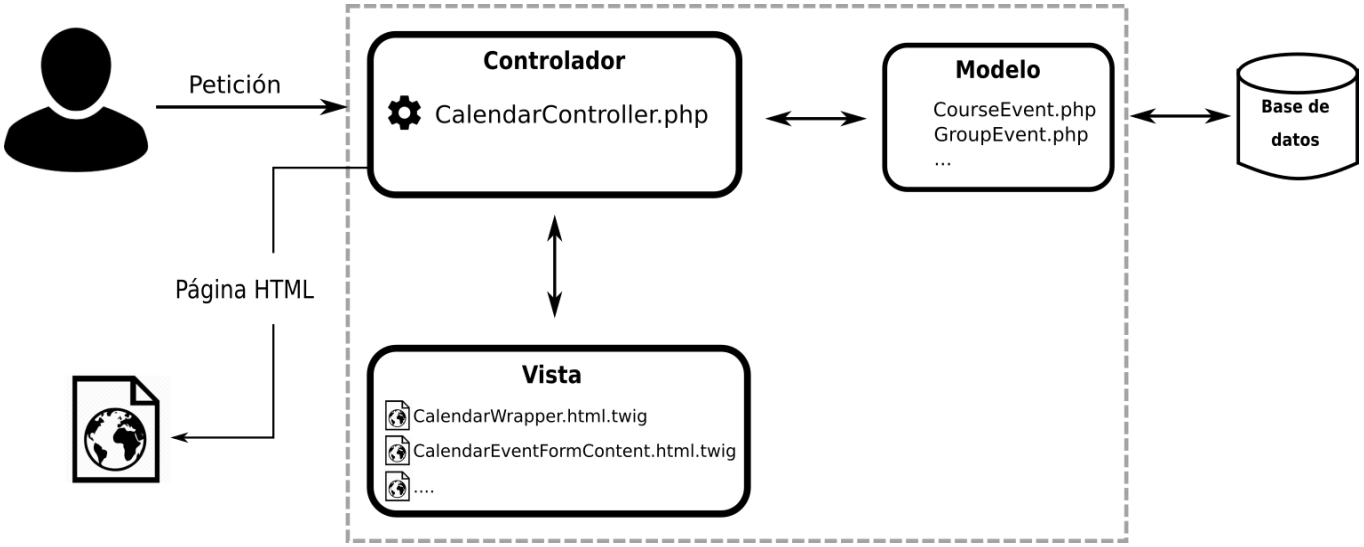


Figura 5 Patrón Modelo-Vista-Controlador

2.6 Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño estructurado común, que lo hace útil para la creación de diseños orientados a objetos reutilizables. Definen una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones («Patrones de diseño» 2018). A continuación, se describen los patrones de diseño utilizados durante el desarrollo del sistema.

2.6.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. Se conocen como una serie de "buenas prácticas" de aplicación, recomendable en el diseño de *software*. Tienen gran importancia, debido a que dan solución a muchos de los problemas que se pueden presentar a la hora de programar (Fernando Alfonso Casas De la Torre [sin fecha]).

Dentro de los patrones GRASP utilizados en el desarrollo del sistema se encuentran los siguientes:

- ✓ **Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. En la solución propuesta este patrón se evidencia en la clase entidad EventBase().

- ✓ **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. En la solución propuesta este patrón se evidencia en la clase controladora CalendarController().

```
public function showCalendarEventAction()
{

    $request = $this->get('request');
    $eventId = is_null($request->get('id')) ? $request->get(
        'eventId'
    ) : $request->get('id');
    $eventEntity = $this->get('fortes_calendar.manager')->getEventById($eventId);

    $course_id = $request->get('course_id');


    return $this->render(
        'CalendarBundle:Default:CalendarEventShowContent.html.twig',
        array(
            'event' => $eventEntity,
            'view' =>'show'
        )
    );
}
```

Figura 6 Patrón controlador

2.6.2 Patrones GoF

Los patrones GoF (*Gang of Four*, en español Pandilla de los Cuatro) se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Joan Sebastián Ramírez Pérez 2016). Dentro de los patrones GoF se emplea en el desarrollo del sistema el que se muestra a continuación:

Creacionales:

- ✓ **Singleton:** Permite el manejo de objetos únicos y que sean accesibles a otros objetos. Además, permite el acceso controlado a una única instancia. Symfony usa por defecto este patrón al crear las instancias de los servicios, un ejemplo específico en la solución propuesta, es el servicio

calendar_event_types.manager, que tiene como función devolver una instancia de la clase CalendarEventManager().

```
calendar_event_types.manager:
        class: FORTES\CalendarBundle\Model\CalendarEventManager
        arguments:  ['@service_container',  '@doctrine.orm.entity_manager']
```

Figura 7 Patrón singleton

2.7 Modelo de diseño

El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y para documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que abarca todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos («Modelo de diseño» 2010).

2.7.1 Diagrama de clase del diseño

Los diagramas de clase de diseño son encargados de mostrar todas las clases participantes, subsistemas y relaciones, así como los atributos y operaciones correspondientes a cada clase. Una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. Las mismas muestran un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones («Universidad internacional abierta y a distancia» 2018).

A continuación, se muestran algunos de los diagramas de clases del diseño que sirven de apoyo en la comprensión de la estructura del sistema. Para el estudio de los demás diagramas de clase del diseño remitirse al Anexo 2.

Capítulo 2: Análisis y diseño del sistema

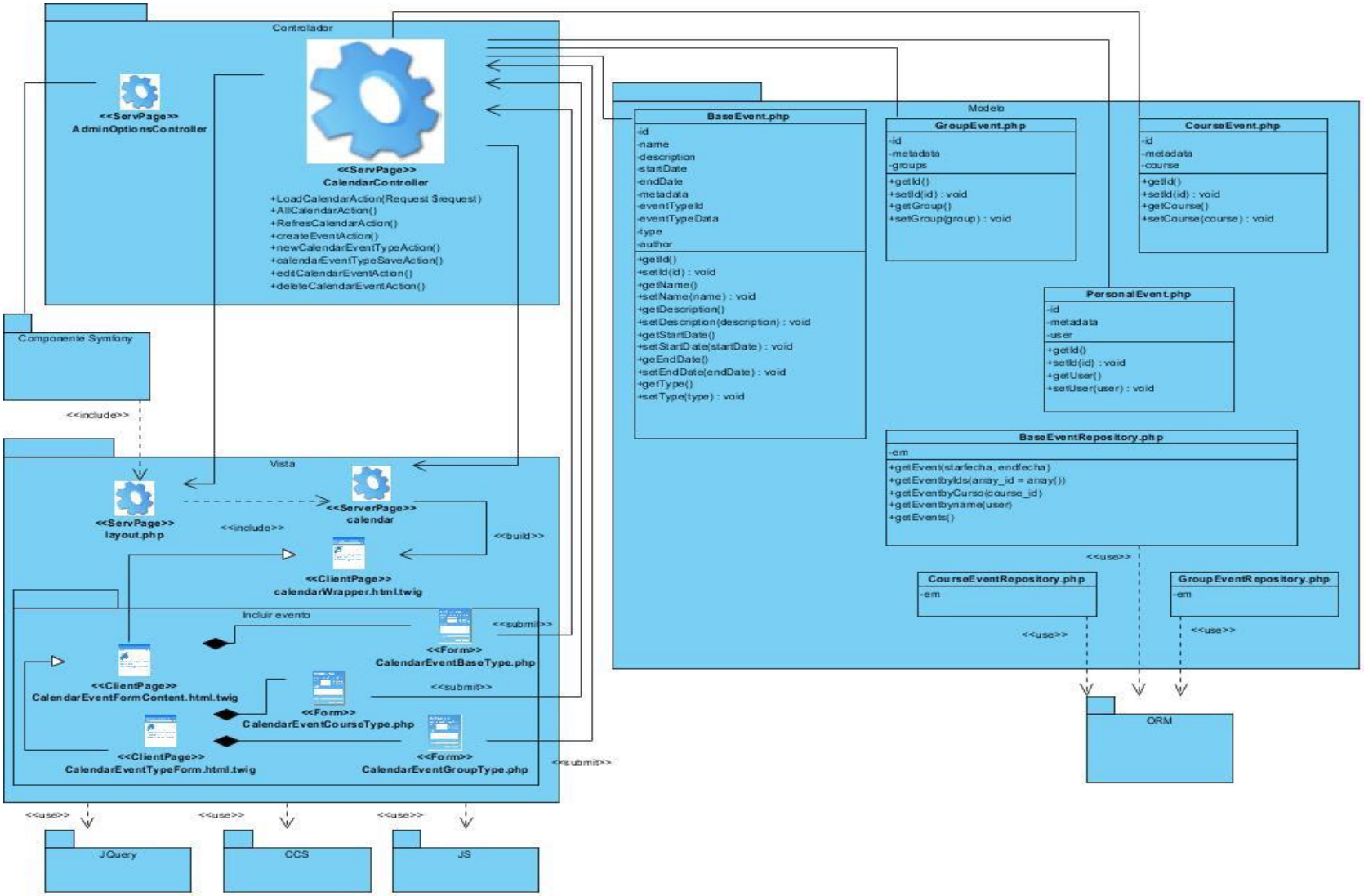


Figura 8 Diagrama de clase de diseño. Incluir evento

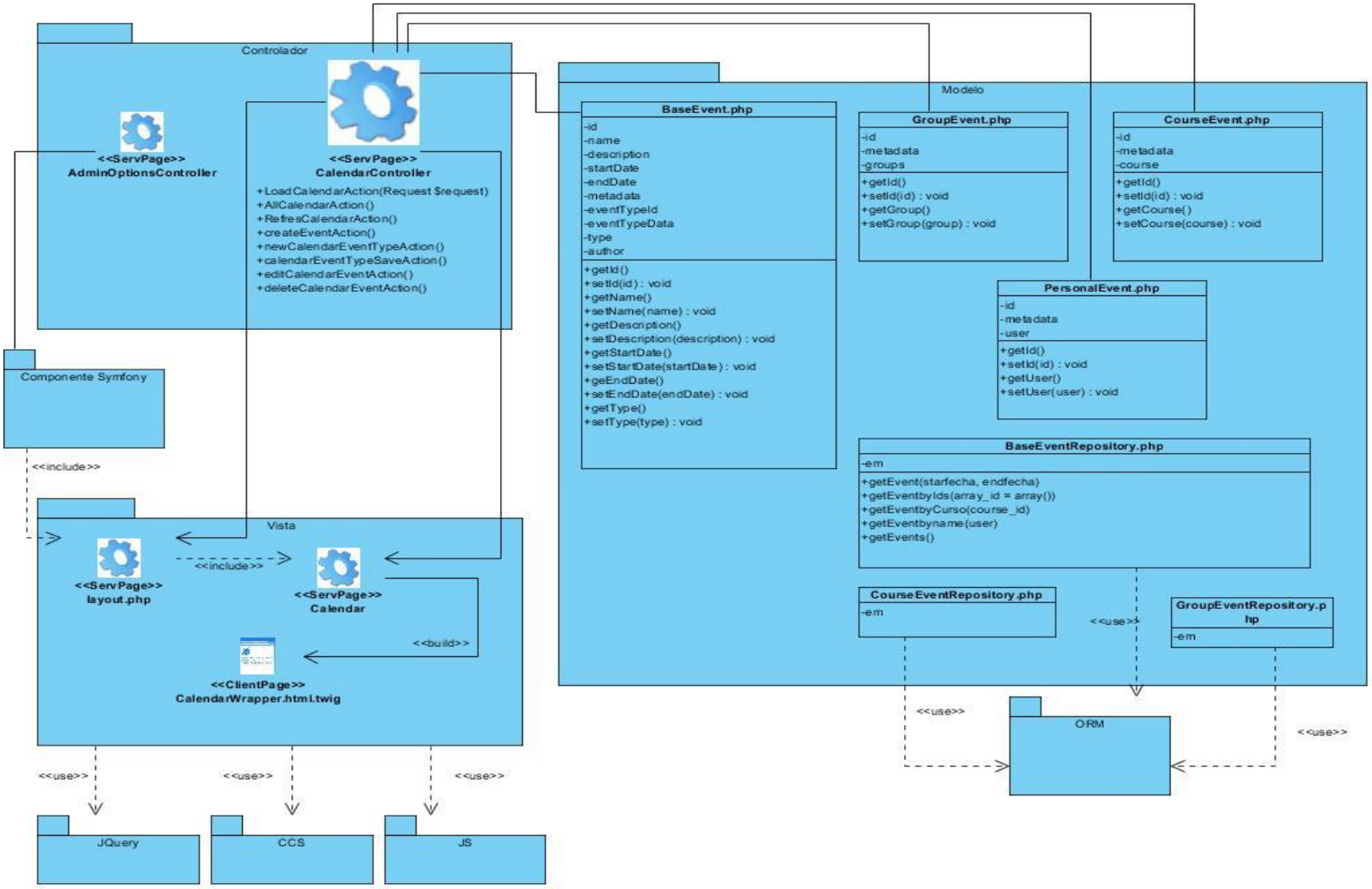


Figura 9 Diagrama de clase de diseño. Mostrar agenda

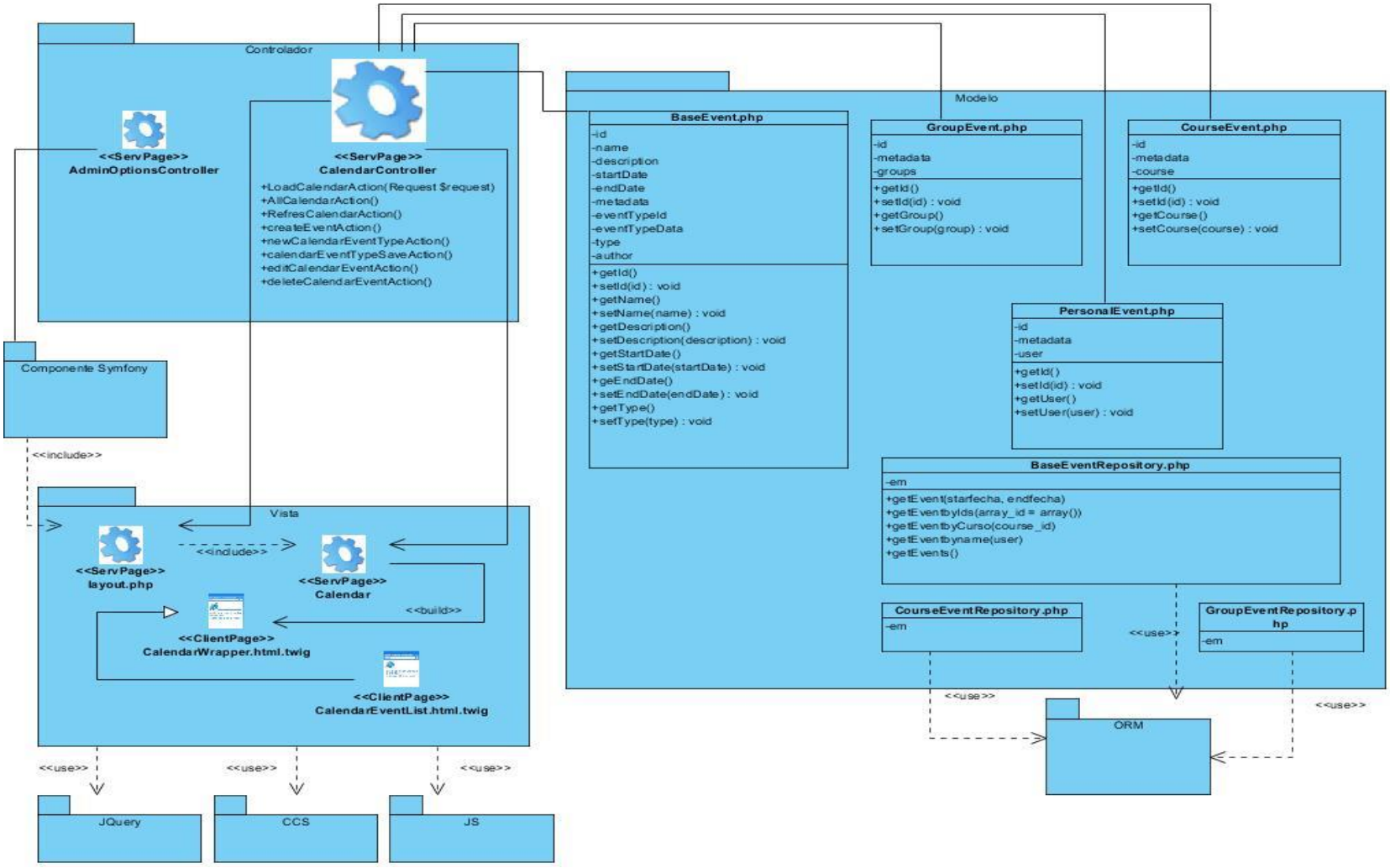


Figura 10 Diagrama de clase de diseño. Listar eventos

2.8 Diseño de la base de datos

Uno de los modelos más utilizados para diseñar base de datos es el modelo entidad-relación, ya que permite una definición clara y concisa de los esquemas conceptuales y de su visión. Este modelo se encuentra basado en dos conceptos: las entidades, que son objetos sobre los cuales se desea guardar información y las relaciones, que constituyen asociaciones entre entidades (Jon Mircha 2014).

A continuación, se presenta el modelo de datos que contiene las entidades del módulo Agenda, que formarán parte de las tablas de la base de datos de ZERA y las relaciones entre ellas.

BaseEvent se encarga de almacenar la información referente a los eventos que se crean en el calendario.

CourseEvent se encarga de almacenar la información referente a los eventos de curso que se crean en el calendario.

GroupEvent se encarga de almacenar la información referente a los eventos de grupo que se crean en el calendario.

User tiene la información asociada a cada usuario que interactúa con el BaseEvent.

Course contiene los datos de los cursos en los que se crean y utilizan CourseEvent.

Rol almacena los roles que por defecto presenta la base de datos de ZERA.

Group contiene los datos de los grupos en los que se crean y utilizan GroupEvent.

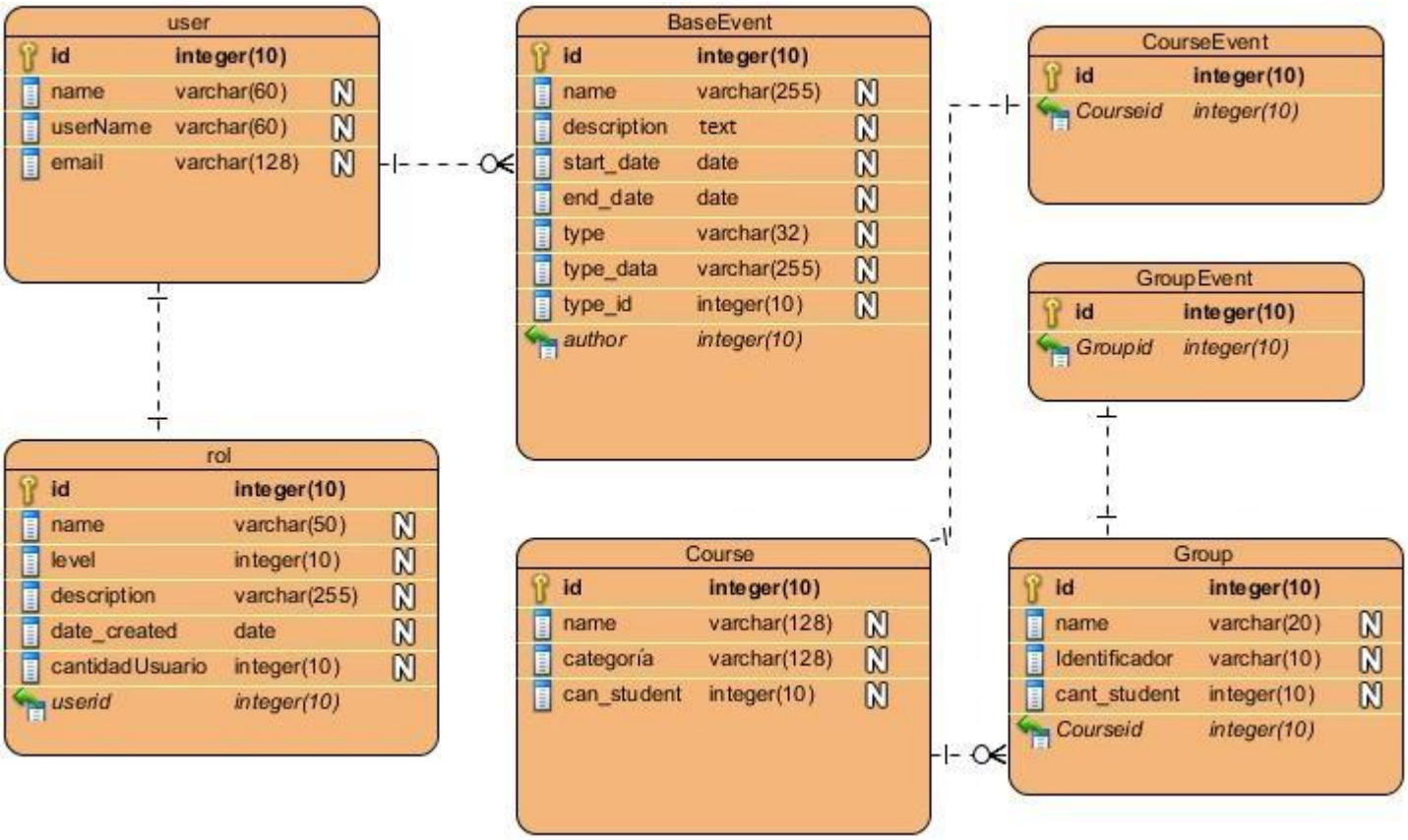


Figura 11 Diagrama Entidad-Relación

Conclusiones parciales

Teniendo en cuenta el estudio realizado previamente se concluye:

- ✓ Se definieron los requisitos funcionales y no funcionales del sistema que sustentan la propuesta de solución.
- ✓ Las definiciones de historias de usuario permitieron tener la base para la implementación de la aplicación.
- ✓ Se aplicaron los patrones necesarios para lograr un diseño flexible y eficiente que sin forzar al programador ayudan a la solución del sistema.
- ✓ Se obtuvo como resultado el diagrama Entidad - Relación que define el diseño de la base de datos y los diagramas de clases del diseño en los que se tuvo en cuenta principalmente las facilidades que el *framework* aporta.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

Introducción

Los artefactos generados en el diseño, constituyen la entrada esencial para el flujo de trabajo de implementación, el cual propone mostrar cómo se implementan las funcionalidades en términos de componentes y cómo lograr la organización de los mismos. Después de realizar este flujo se elabora el de pruebas, donde se efectúan las pruebas necesarias a las funcionalidades implementadas para verificar que estén correctamente desarrolladas, permitiendo comprobar su adecuado funcionamiento antes de ser usado por los usuarios finales.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y los lenguajes de programación utilizados y cómo dependen los componentes uno de otros (Jacobson, Ivar; Booch, Grady; Rumbaugh, James 2018).

Los diagramas de despliegue y componentes, que son artefactos generados en este flujo de trabajo, conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

3.1.1 Diagrama de componentes

Los diagramas de componentes permiten modelar la vista de implementación del sistema a partir del cual se construye la aplicación. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de *software* y organiza los subsistemas de implementación en capas. Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño (Jacobson, Ivar; Booch, Grady; Rumbaugh, James 2018). A continuación, se muestra el diagrama de componente correspondiente al sistema.

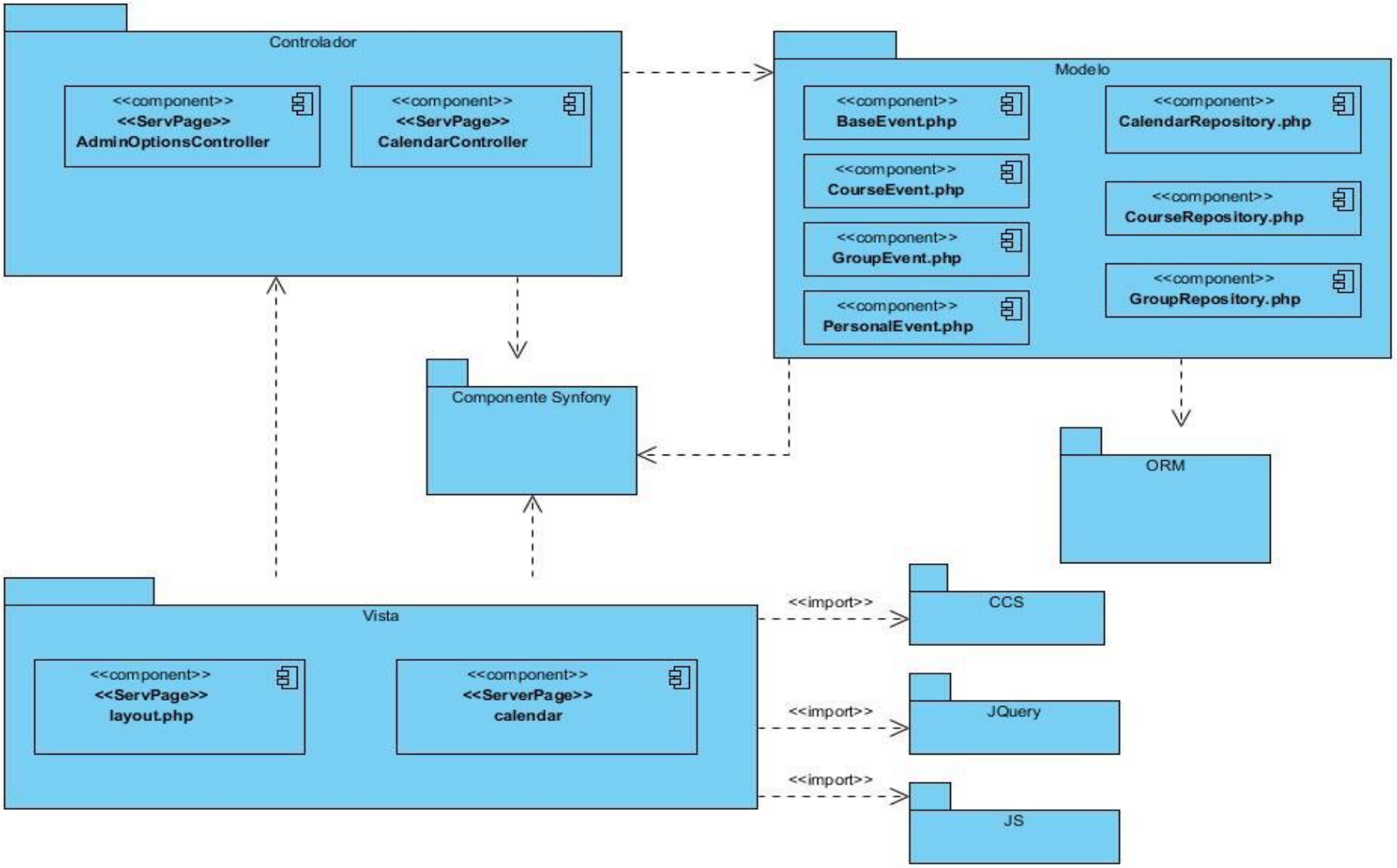


Figura 12 Diagrama de Componentes

3.1.2 Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama UML que muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución *run-time* en las instancias de los nodos de proceso (Larman C 2010).

El diagrama de despliegue que se muestra a continuación representa la distribución física del sistema a través de nodos. Está compuesto por una computadora cliente que se comunica con los servidores de medias y aplicaciones a través de los protocolos HTTP y HTTPS (*Hyper Text Transfer Protocol Secured*) respectivamente. También existe una conexión entre el servidor de media y el de aplicaciones la cual se realiza mediante el protocolo NFS (*Network File System*) y una última conexión vía TCP (Protocolo de Control de Transmisión) entre el Servidor de aplicaciones ZERA 2.1 y el Servidor de Base de Datos ZERA 2.1.

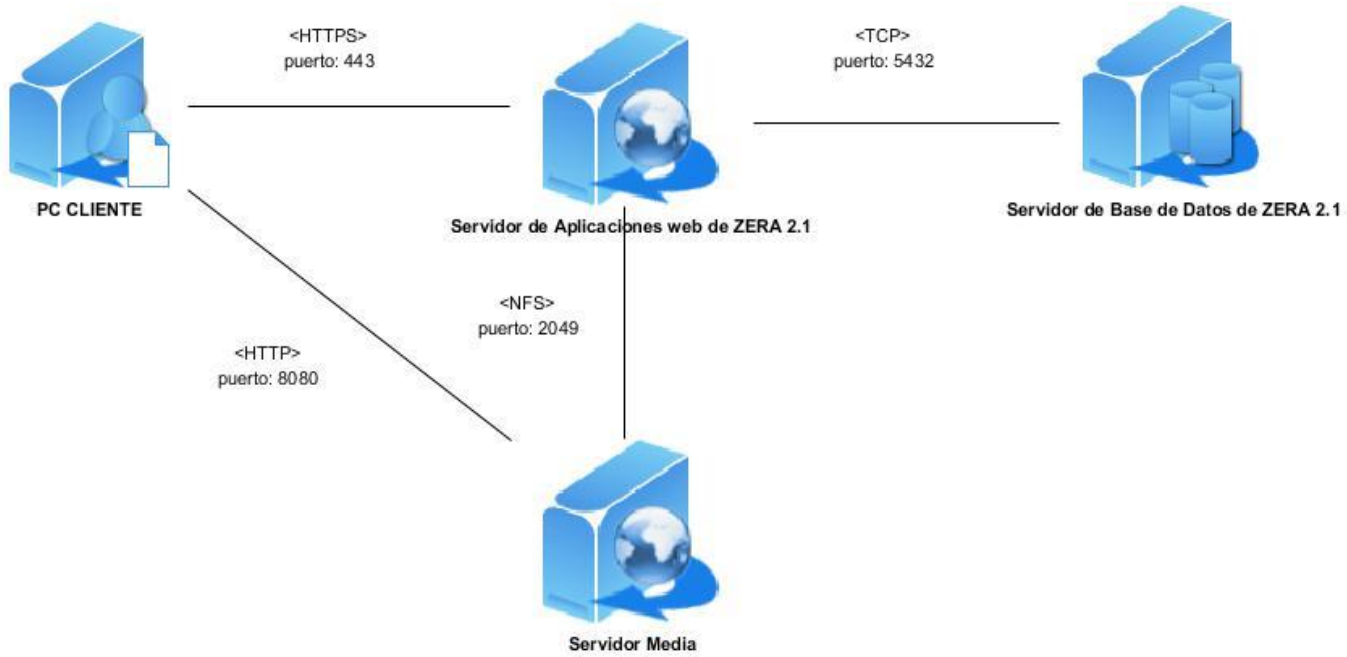


Figura 13 Diagrama de despliegue

3.1.3 Estándares de codificación

Es un estilo de programación homogéneo, en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

JavaScript

Descripción: se utiliza la notación camelCase para los nombres de funciones. Todos los nombres comienzan con una letra y se utiliza la palabra reservada "var" para declarar variables.

Ejemplo:

```
function createEventPopover(event, jsEvent){

    calendar = event.source.calendar.el;

    var popover = $(jsEvent.target);
    var removeEventBtn = $('div', {id:'removeEventBtn', class: 'btn btn-danger'}).append($('i', {class: 'fa fa-trash'}));
    var editEventBtn = $('div', {id:'editEventBtn', class: 'btn btn-primary'}).append($('i', {class: 'fa fa-pencil'}));
    var showEventBtn = $('div', {id:'showEventBtn', class: 'btn btn-primary'}).append($('i', {class: 'fa fa-eye'}));
```

Figura 14 Ejemplo del código de JavaScript

HTML

Descripción: no se debe colocar espacios entre la relación atributo-valor. Se debe utilizar lowercase para el nombre de los atributos de las etiquetas.

Ejemplo:

```
<script type="text/javascript">
    initCalendarForm('form[name="fortes_calendar_event_base_form"]');

    $('form[name="fortes_calendar_event_base_form"]').find('select').select2();
</script>
```

Figura 15 Ejemplo del código de HTML

CSS

Descripción: una declaración CSS siempre termina en punto y coma y los conjuntos de declaraciones se colocan entre llaves. Para hacer un código CSS legible, ponga una declaración en cada línea. Coloque la llave que cierra en una línea nueva.

Ejemplo:

```
.calendar-header-bar {
    margin-bottom: 15px;
}
```

Figura 16 Ejemplo del código de CSS

PHP

Descripción: adicionar un espacio alrededor de cada operador binario (==, &&, ...), exceptuando el operador de concatenación (.).

Ejemplo:

```
if ($this->type == 'calendar_event_course') {
    $event['backgroundColor'] = 'darkorange';
    $event['borderColor'] = 'darkorange';
```

Figura 17 Ejemplo del código de PHP

Descripción: adicionar una línea en blanco antes de cada sentencia return, a menos que se encuentre como única sentencia (como en un if o un else).

Ejemplo:

```php
if ($eventType->hasForm()) {
    $eventType = $this->get('calendar_event_types.manager')->getEventTypeByName($eventTypeName);

    $eventTypeForm = $eventType->getForm();
    $entityName = $eventTypeForm->getEntityName();
    $entity = new $entityName();
    $form2 = $this->createForm($eventTypeForm, $entity);

    return $this->render(
        'CalendarBundle:Default:CalendarEventTypeForm.html.twig',
        array(
            'form' => $form2->createView()
        )
    );
} else {
    return new Response();
}
```

Figura 18 Ejemplo del código de PHP

Descripción: declarar los atributos de las clases antes de los métodos.

Ejemplo:

```php
/**
 * @ORM\Column(name="id", type="integer")
 * @ORM\Id
 * @ORM\GeneratedValue(strategy="AUTO")
 */
private $id;
/**
 * Get id
 *
 * @return integer
 */
public function getId()
{
    return $this->id;
}
```

Figura 19 Ejemplo del código de PHP

Convención de nombres

Descripción: la declaración de funciones o métodos siempre comenzará con letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa camelCase.

Ejemplo:

```
public function newCalendarEventTypeAction(){...}
```

Figura 20 Ejemplo de código de PHP

Descripción: se utiliza namespaces para todas las clases.

Ejemplo:

```
namespace FORTES\CalendarBundle\Entity;
```

Figura 21 Ejemplo de código de PHP

Descripción: para definir cada uno de los servicios hay que tener en presente los siguientes indicadores:

- ✓ Los nombres de los servicios contienen grupos separados por puntos.
- ✓ El alias de Inyección de Dependencias del bundle es el primer grupo.
- ✓ Se utiliza minúsculas para los nombres de servicios y sus parámetros.
- ✓ Un nombre de grupo utiliza la notación guion bajo.

Ejemplo:

```
calendar_event_types.manager:
```

Figura 22 Ejemplo de código de PHP

Comentarios en el código

Descripción: los comentarios que emplean una sola línea se definen de la siguiente manera.

Ejemplo:

```
//Nombre del formulario del Evento Base
```

Figura 23 Ejemplo de código comentado

Descripción: los comentarios que emplean varias líneas utilizan las anotaciones siguientes:

- ✓ @param: esta etiqueta provee el nombre, el tipo y la descripción de los parámetros de una función.
- ✓ @return: esta etiqueta es utilizada para documentar el valor que retorna una función.

Ejemplo:

```
/**
 * Set description
 *
 * @param string $description
 *
 * @return BaseEvent
 */
```

Figura 24 Ejemplo de código comentado

3.2 Pruebas de *software*

Las pruebas del *software* son una técnica dinámica de validación y verificación (V&V), las cuales implican ejecutar una implementación del *software* con datos de prueba, se examinan las salidas del *software* y su entorno operacional para comprobar que funciona tal y como se requiere (Sommerville, Ian 2018).

Con la ejecución de las pruebas de *software* se persigue descubrir defectos en el sistema asociados a comportamientos incorrectos o no deseables y para verificar que cumple con los requerimientos del cliente, con el fin de suplir sus necesidades. Al ejecutarse esta actividad no se obtiene un sistema totalmente libre de errores, pero si apto para ser usado por el usuario final.

3.2.1 Niveles de pruebas

Las pruebas son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Entre los niveles de prueba se encuentran el nivel de pruebas unitarias, nivel de pruebas de integración, nivel de pruebas del sistema y nivel de pruebas de aceptación. Una vez implementado el sistema fue sometido al nivel de prueba que a continuación se detalla (Isi docencia 2018):

Pruebas del Sistema: se realiza durante la construcción del sistema con el objetivo de verificar el ingreso, procesamiento y recuperación apropiado de datos. Este tipo de pruebas se basan en técnicas de caja negra, esto es, verificar el sistema y analizar las salidas o resultados.

Para ejecutar la prueba de sistema se determinó aplicar el siguiente tipo de prueba:

- ✓ **Prueba funcional:** encuentra diferencia entre los requerimientos funcionales y el sistema (Jorge Hernan Abad Londoño 2018).

Pruebas de aceptación: es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción. La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar como el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo. Basado en esta prueba el cliente determina si acepta o rechaza el sistema (Jorge Hernan Abad Londoño 2018).

3.2.2 Métodos de prueba

Existen distintas técnicas de pruebas que proporcionan criterios para generar casos de pruebas que provoquen fallos en los programas (Juristo, Natalia; Moreno, Ana M.; Vegas, Sira 2018). Se agrupan en:

- ✓ **Enfoque estructural o de caja blanca:** las pruebas de caja blanca en ocasiones llamada prueba de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero de *software* podrá derivar casos de prueba que:
  - Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
  - Se ejerciten los lados verdaderos y falsos de todas las decisiones lógicas.
  - Se ejecuten todos los bucles dentro de sus límites operacionales.
  - Se ejerciten estructuras de datos internos para asegurar su validez (Pressman RS 2010).
- ✓ **Enfoque funcional o de caja negra:** las pruebas de caja negra, también llamadas pruebas de comportamiento se concentran en los requisitos funcionales del *software*. Estas permiten al ingeniero de *software* derivar un conjunto de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. La prueba de caja negra no es una opción frente a las técnicas de caja blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de descubrir errores de clases diferentes de los que se descubrirán en los métodos de caja blanca (Pressman RS 2010).

  Con el objetivo de aplicar las pruebas de caja negra o funcional, es necesario apoyarse en el Diseño de Casos de Prueba propuesto por la metodología de desarrollo de *software* seleccionada.

> Un caso de prueba es una forma de comprobar el correcto funcionamiento del sistema, en estos se incluyen las entradas, resultados y condiciones con la que se ha de verificar, constituyendo la guía principal para el probador (Sommerville I 2006).

Para la validación de la propuesta de solución será aplicado el método de caja negra, debido a que este permitirá corregir problemas en la interfaz del usuario y se comprobará que esta propuesta realiza las funciones requeridas por el usuario.

Una de las técnicas usadas en el método de prueba de caja negra es la partición de equivalencia, este permite examinar los valores válidos y no validos de las entradas existentes en el *software*. Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del *software*.

3.2.3 Partición equivalente

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Este se dirige a una definición de casos de pruebas que descubran clases de errores, reduciendo así el número total de casos de pruebas que hay que desarrollar.

El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. Por lo general, esta condición es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (Juristo, Natalia; Moreno, Ana M.; Vegas, Sira 2018).

3.2.4 Diseños de caso de prueba

La intención de los casos de prueba es probar el sistema de una forma detallada, incluyendo las entradas con las que se experimentarán, las condiciones bajo las cuales se realizan y los resultados esperados.

A continuación, se presentan los casos de prueba propuestos para la HU Incluir evento, Editar evento y Ver evento. Para consultar el resto de los casos de prueba remitirse al Anexo 3.

Tabla 5 Caso de prueba incluir evento

CP Incluir evento								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide adicionar un evento. Para ello el usuario selecciona la opción de Crear evento. El sistema muestra un formulario con los campos correspondientes al evento. Una vez llenado el formulario se selecciona la opción Guardar y se guarda el evento en la base de datos. Una vez creado el evento se muestra una notificación de validez. **Condiciones de ejecución** La persona debe estar autenticada en el sistema.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC1.1 Opción Incluir evento	Selecciona en el calendario la opción de "Crear						El sistema debe permitir introducir y/o seleccionar los siguientes datos para	Calendario/Crear evento

	evento”						Incluir un Evento: • Nombre • Descripción • Fecha de inicio • Fecha de fin • Tipo de evento	
EC1.2 Opción Guardar	Introduce y/o selecciona los datos y selecciona ‘‘Guardar”.	V	V	V	V	V	Valida los datos, registra el evento en el sistema, envía una notificación (ver DCP_Notificar evento.ods) y muestra el mensaje de información: Se ha realizado la acción satisfactoriamente.	Calendario/Crear evento/Guardar
EC1.3 Opción Cancelar	Selecciona la opción “Cancelar”						Elimina los datos y regresa a la vista anterior.	Calendario/Crear evento/Cancelar

EC1.4 Longitud máxima	Existen datos que no cumplen con la restricción.	I	N/A	N/A	N/A	N/A	El sistema señala el o los campos que no cumplen con la restricción de longitud máxima, estos pueden ser: • Nombre Muestra un indicador sobre el campo que no cumple con la restricción. Muestra un mensaje de información que dice: Este valor es demasiado largo. Debería tener 255 caracteres o menos. Regresa al EC1.2.	Calendario/Crear evento/Guardar
EC1.5 Datos	Existen datos	N/A	N/A	V	I	N/A	El sistema señala un indicador sobre el	Calendario/Crear evento/Guardar

incorrectos	incorrectos.						campo de fecha de fin y muestra el mensaje de información: La fecha de fin debe ser mayor que la fecha de inicio. Regresa al EC1.2.	
EC1.6 Campos vacíos	Existen campos vacíos.	I	N/A	V	V	N/A	El sistema señala el o los campos obligatorios que no hayan sido introducidos y/o seleccionados. Se muestra un indicador sobre el campo vacío Muestra un mensaje de información que dice: Este valor no debería	Calendario/Crear evento/Guardar
		V	N/A	V	V	N/A		
		V	N/A	I	V	N/A		
		V	N/A	V	I	N/A		
		V	N/A	V	V	N/A		

							estar vacío. Regresa al EC1.2.	

Tabla 6 Caso de prueba editar evento

CP Editar evento								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide editar un evento creado. Para ello el usuario selecciona en el calendario el evento que desea editar que haya sido creado por él, se muestra una vista con los datos correspondientes al evento y selecciona la opción Editar. El sistema muestra un formulario con los datos del evento que desea editar. Una vez llenado el formulario se selecciona la opción Guardar y se guardan los cambios en la base de datos. Una vez editado el evento se muestra una notificación de validez. **Condiciones de ejecución** La persona debe estar autenticada.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**

EC2.1 Opción seleccionar evento	Selecciona en el calendario el evento que desea editar.						El sistema muestra una vista con los siguientes campos y sus datos correspondiente: • Nombre • Descripción • Fecha de inicio • Fecha de fin • Tipo de evento	Calendario/Evento
EC2.2 Opción Editar	Selecciona la opción “Editar”						El sistema debe permitir introducir y/o seleccionar los siguientes datos para modificar los datos del evento seleccionado: • Nombre • Descripción • Fecha de inicio • Fecha de fin • Tipo de evento	Calendario/Evento/ Editar

EC2.3 Opción Guardar	Introduce los datos y selecciona ‘‘Guardar”.	V	V	V	V	V	Valida los datos. Registra los datos nuevos del evento en el sistema y envía una notificación (ver DCP_Notificar evento.ods).	Calendario/Evento/ Editar/Guardar
EC2.4 Opción Cancelar	Selecciona la opción “Cancelar”						Elimina los datos. Regresa a la vista anterior.	Calendario/Evento/ Editar/Cancelar
EC2.5 Longitud máxima	Existen datos que no cumplen con la restricción.	I	N/A	N/A	N/A	N/A	El sistema señala el o los campos que no cumplen con la restricción de longitud máxima, estos pueden ser: • Nombre Muestra un indicador sobre el campo que	Calendario/Evento/ Editar/Guardar

							no cumple con la restricción. Muestra un mensaje de información que dice: Este valor es demasiado largo. Debería tener 255 caracteres o menos. Regresa al EC2.2.	
EC2.6 Datos incorrectos	Existen datos incorrectos.	N/A	N/A	V	I	N/A	El sistema señala un indicador sobre el campo de fecha de fin y muestra el mensaje de información: La fecha de fin debe ser mayor que la fecha de inicio. Regresa al EC2.2.	Calendario/Evento/ Editar/Guardar
EC2.7	Existen	I	N/A	V	V	N/A	El sistema señala el o	Calendario/Evento/

Campos vacíos	campos vacíos.	V	N/A	V	V	N/A	los campos obligatorios que no hayan sido introducidos y/o seleccionados. Se muestra un indicador sobre el campo vacío. Muestra un mensaje de información que dice: Este valor no debería estar vacío. Regresa al 2.2.	Editar/Guardar	
		V	N/A	I	V	N/A			
		V	N/A	V	I	N/A			
		V	N/A	V	V	N/A			

Tabla 7 Caso de prueba mostrar evento

CP Mostrar evento								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide ver un evento creado. Para ello el usuario selecciona en el calendario el evento que desea ver que haya sido creado por él o por el profesor del grupo al que pertenece. El sistema muestra una vista con los campos correspondientes al evento. Una vez revisado el evento se selecciona la opción Cancelar y se regresa a la vista anterior. **Condiciones de ejecución** La persona debe estar autenticada.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC3.1 Opción seleccionar evento	Selecciona en el calendario el evento que desea ver.						Se muestra una vista con los siguientes campos y sus datos correspondiente: • Nombre	Calendario/Evento

							• Descripción • Fecha de inicio • Fecha de fin • Tipo de evento	
EC3.2 Opción Cancelar	Selecciona la opción "Cancelar"						Regresa a la vista anterior.	Calendario/Evento/ Cancelar
EC3.3 Opción Editar	Selecciona la opción "Editar"						El sistema debe permitir introducir y/o seleccionar los siguientes datos para modificar los datos del evento seleccionado: • Nombre • Descripción • Fecha de inicio • Fecha de fin • Tipo de evento Ver DCP _Editar evento.	Calendario/Evento/ Editar

EC3.4 Opción Eliminar	Selecciona la opción “Eliminar”						Se muestra una vista con la interrogante ¿Está seguro que desea eliminar el elemento seleccionado? Ver DCP _Eliminar evento.	Calendario/Evento/ Eliminar	

3.2.5 Resultados obtenidos en las pruebas

En las pruebas realizadas al sistema mediante el método de caja negra, se pudo comprobar el cumplimiento de los requisitos funcionales y no funcionales del *software* obtenido.

A continuación, se presenta el resultado de las pruebas realizadas.

Tabla 8 Resultados de las pruebas realizadas

Iteraciones	Cantidad de casos de prueba	No conformidades detectadas			
		Alta	Media	Baja	Total
1	10	4	11	15	30
2	10	1	9	11	21
3	10	0	2	1	3

A continuación, se muestra un gráfico donde se puntualiza por iteraciones el total de no conformidades detectadas en alta, media y baja.

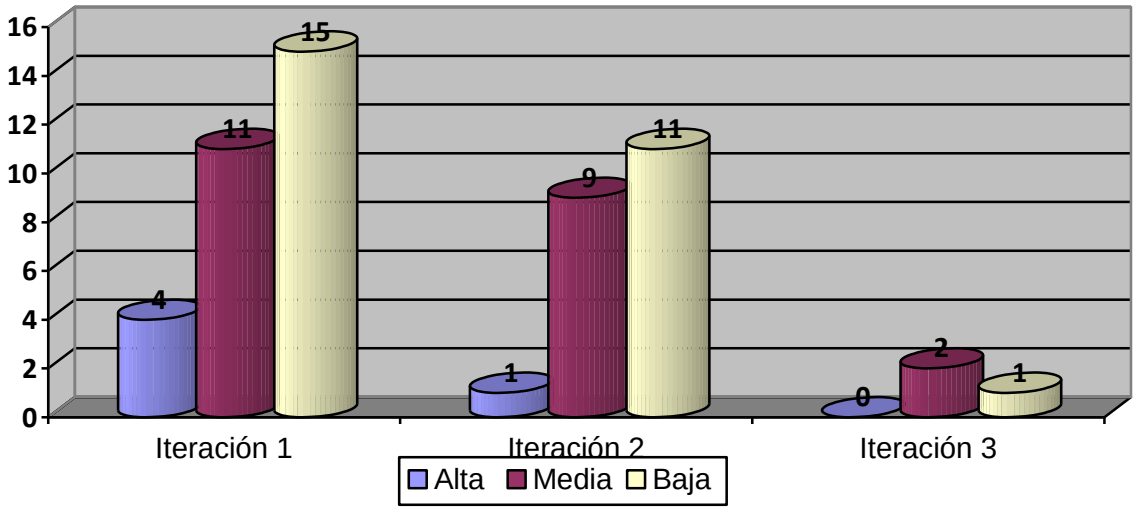


Figura 25 Resultado de las pruebas realizadas

Capítulo 3: Implementación y validación del sistema

Conclusiones Parciales

Teniendo en cuenta el estudio realizado previamente se concluye:

- ✓ El modelo de implementación, conformado por el diagrama de componente permitió implementar y organizar los elementos del modelo del diseño.
- ✓ Las pruebas realizadas al sistema mostraron en la primera iteración treinta no conformidades, en la segunda iteración veintiuna no conformidades y en la tercera iteración tres no conformidades, que fueron validadas posteriormente en las pruebas de regresión quedando totalmente solucionadas las no conformidades.

CONCLUSIONES

Con la culminación del presente trabajo se ha dado cumplimiento a los objetivos trazados en la investigación, obteniéndose como resultado principal, un módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1. A continuación se exponen las conclusiones generales a las que se arribó luego del desarrollo de la solución:

- ✓ Las soluciones analizadas durante la elaboración del marco teórico de la investigación no satisfacen en su totalidad el problema planteado.
- ✓ Las herramientas, tecnologías y lenguajes seleccionados; y los artefactos generados de acuerdo la metodología AUP UCI, facilitaron la implementación del calendario.
- ✓ Se obtuvo el módulo Agenda para la Plataforma Educativa Xauce ZERA v2.1, el cual permite la gestión de eventos, listar todos los eventos creados por el usuario o que sean asociados a él y buscar eventos en el sistema. También admite exportar los eventos de la agenda en el formato ICS, mostrar la agenda y cambiar la apariencia de la misma.
- ✓ Se aplicaron pruebas al sistema permitiendo determinar y corregir las no conformidades encontradas, lo que incidió positivamente en la calidad de la solución.

RECOMENDACIONES

A partir de los resultados obtenidos con la investigación el autor propone las siguientes recomendaciones para futuros trabajos tomando como referencia el actual:

- ✓ Implementar una funcionalidad para importar eventos desde otros calendarios usando el estándar ICS.
- ✓ Incorporar funcionalidades para la repetición y recordatorio de eventos en futuras versiones.

REFERENCIAS BIBLIOGRÁFICAS

1. AGENDA, 2014. Significados. [en línea]. Disponible en: https://www.significados.com/agenda/.

2. ANGEL CABO YERA, 2007. *Diseño y programación de Bases de Datos*. Madrid, España: s.n.

3. ARTUROS OCAMPO LÓPEZ, 2012. relación PE LMS. [en línea]. Disponible en: https://es.slideshare.net/Departamento_Academico/plataformas-educativas-concepto-caractersticas-y-ejemplos.

4. CARITOL, 2009. Organización de Eventos. [en línea]. Disponible en: http://72020organizaciondeeventos-caritol.blogspot.com/2009/09/concepto-agenda-de-un-evento-es-un.html.

5. CHACÓN S, 2014. *Pro Git, el libro oficial de Git. 2da ed*. S.l.: s.n.

6. CLAROLINE, 2017. UCL. [en línea]. Disponible en: http://www.claroline.net.

7. CORONA S, 2008. *Nginx: A Practical Guide to High Performance. Estados Unidos: O'Reilly Media*. 2008. S.l.: s.n.

8. DANIEL PECOS MARTINEZ, 2017. POSTGRESQL. [en línea]. Disponible en: http://danielpecos.com/docs/mysql_postgres/x15.html.

9. DÍAZ BECERRO. S., 2009. *PLATAFORMAS EDUCATIVAS, UN ENTORNO PARA PROFESORES Y ALUMNOS.* 2009. S.l.: s.n.

10. E. HERNÁNDEZ ORALLO, 2014. *El Lenguaje Unificado de Modelado (UML)*. S.l.: s.n.

11. ESTABAN GABRIEL MAIDA, JULIAN PACIENZIA., 2015. Metodología de Desarrollo de Software. [en línea]. Disponible en: http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf.

12. EVENTO, 2018. Que Significado. [en línea]. Disponible en: http://quesignificado.com/evento/.

13. F. POTENCIER, 2010. *Symfony* [en línea]. S.l.: s.n. Disponible en: http://librosWeb.es/.

14. FERNANDO ALFONSO CASAS DE LA TORRE, [sin fecha]. patrones grasp. [en línea]. Disponible en: https://es.slideshare.net/Indiana_1969/patrones-grasp-76195661.

15. GUERRA, YULAINE ARIAS, 2010. *Conceptualización de una red social educativa que integre de forma colaborativa las aplicaciones e-learning de la Universidad de las Ciencias Informática*. 2010. S.l.: s.n.

16. I. GARCERANT, 2014. Tecnología y Synergix. [en línea]. Disponible en: http://synergix.wordpress.com/author/igarcerant/.

Referencias bibliográficas

17. ISI DOCENCIA, 2018. Niveles de pruebas. [en línea]. Disponible en: http://www.lsi.us.es/docencia/get.php?id=361.

18. J. P. GOMES GALLEGO, J.A. GALVES, 2010. Ingeniería de Requerimientos. Universidad Tecnológica de Pereira.

19. JACOBSON I., BOOCH G, 2018. *El Lenguaje Unificado del Modelado. Modelo de Referencia.* S.l.: s.n.

20. JACOBSON, IVAR; BOOCH, GRADY; RUMBAUGH, JAMES, 2018. *El proceso unificado de desarrollo de software*. S.l.: s.n.

21. JAVIER J. GUTIÉRREZ, 2010. *Framework* [en línea]. 2010. S.l.: s.n. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

22. J.C. SEPULVEDA PEÑA, 2005. *aprenDIST*. 2005. S.l.: s.n.

23. J.E. PÉREZ, 2014a. Introducción a Ajax. [en línea]. Disponible en: http://librosWeb.es/.

24. J.E. PÉREZ, 2014b. Introducción a CSS. [en línea]. Disponible en: http://librosWeb.es/.

25. J.E. PÉREZ, 2014c. Introducción a JavaScript. [en línea]. Disponible en: http://librosWeb.es/.

26. JOAN SEBASTIÁN RAMÍREZ PÉREZ, 2016. patrones gof. [en línea]. Disponible en: https://es.slideshare.net/SebastianRamrez2/patrones-gof?qid=68693a66-ba49-4a7e-82e7-7e96b4d308dd&v=&b=&from_search=1.

27. JON MIRCHA, 2014. EDteam. [en línea]. Disponible en: https://ed.team/blog/modelo-entidad-relacion.

28. JORGE HERNAN ABAD LONDOÑO, 2018. TIPOS DE PRUEBAS DE SOFTWARE. [en línea]. Disponible en: http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html.

29. JOSÉ HUIDOBRO, 2018. *Tecnologías de información y comunicación* [en línea]. 15 mayo 2018. S.l.: s.n. Disponible en: http://cmapspublic3.ihmc.us/rid=1H3108YC5-BYQQP-R83/Tecnologias%20de%20Informaci%C3%B3nyComunicacion.pdf.

30. JURISTO, NATALIA; MORENO, ANA M.; VEGAS, SIRA, 2018. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE*. S.l.: s.n.

31. LARMAN C, 2010. *UML y Patrones: Introducción al análisis y diseño orientado a objetos. 2da ed. Vancouver, Canadá: Prentice Hall*. S.l.: s.n.

32. Lenguaje de programación. [en línea], 2018. Disponible en: http://www.juegotangram.com.ar/informatica/prg/lenguajedeprogramacion.html.

33. LMS. [en línea], 2018. Disponible en: http://www.newwweb.com.mx/plataforma-lms.

Referencias bibliográficas

34. MANUAL PRÁCTICO DE HTML. ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN, UNIVERSIDAD POLITÉCNICA DE MADRID, ESPAÑA., 2015. HTML.

35. MANUEL TORRES REMÓN, 2012. *PHP*. 2012. S.l.: s.n.

36. MEJIA GOMEZ, 2013. Ingenio DS. Patrones Arquitectónicos. [en línea]. Disponible en: https://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/.

37. Modelo de diseño. [en línea], 2010. Disponible en: https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/workproducts/rup_design_model_2830034D.html.

38. MÓNICA MARÍA AGUDELO B., 2017. Plataforma educativa. [en línea]. Disponible en: http://aprendeenlinea.udea.edu.co/banco/html/plataformaseducativas/.

39. MYSQL-HISPANO.ORG., 2017. MySGL. [en línea]. Disponible en: http://www.webestilo.com/mysql/intro.phtml.

40. NISKA C., 2014. *Extending Bootstrap. UK: Packt Publishing Ltd*. S.l.: s.n.

41. ORACLE CORPORATION, 2017. NetBeans. [en línea]. Disponible en: http://netbeans.org/community/releases/69/index_es.html.

42. ORTEGA, JESÚS FERNÁN, 2018. ¿Qué es la Teleformación? [en línea]. Disponible en: http://www.adrformacion.com/articulos/formacion/_que_es_la_teleformacion_y_que_ventajas_aporta_/artic ulo28.html.

43. ORTIZ BATISTA Y, LÓPEZ REINOSO Y, MEDINA LEÓN Y, GONCE FERNÁNDEZ S, BATARD LORENZO D, GULÍN GONZÁLEZ, 2010. *Propuesta de solución para la gestión de la información de la actividad de ciencia, tecnología e innovación en la Universidad de las Ciencias Informáticas. RCCi.* S.l.: s.n.

44. PALMA PÉREZ N, 2013. *Módulo para la administración de los servidores web en HMAST. [Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.]. [Cuba]: Universidad de las Ciencias Informáticas*. 2013. S.l.: s.n.

45. PATRICIO LETELIER, 2006. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP*. 2006. S.l.: s.n. Vol. 5. ISSN 1666-1680

46. Patrones de diseño. [en línea], 2018. Disponible en: https://msdn.microsoft.com/es-es/library/bb972240.aspx.

47. PLATAFORMA DE SAKAI, 2014. Sakai. [en línea]. Disponible en: https://www.sakaiproject.org/about.

48. POTENCIER, FABIEN; ZANINOTTO, FRANCOIS, 2018. *Symfony la guía definitiva* [en línea]. S.l.: s.n. Disponible en: http://librosweb.es/libro/symfony_1_2/capitulo_2/el_patron_mvc.html.

Referencias bibliográficas

49. PRESSMAN RS, 2010. *Ingeniería de software: Un enfoque práctico. 6ta ed. Nueva York, EUA: McGraw-Hill*. S.l.: s.n.

50. PROGRAMADORES, 2017. calendario moodle. [en línea]. Disponible en: https://docs.moodle.org/all/es/Calendario.

51. SOMMERVILLE I, 2006. *Software Engineering. 8va ed. Pearson Education Limited*. S.l.: s.n.

52. SOMMERVILLE, IAN, 2018. *Ingeniería del software. Séptima edición*. S.l.: Pearson Addison Wesley.

53. TAMARA RODRÍGUEZ SÁNCHEZ, 2015. *Metodología de desarrollo para la Actividad productiva de la UCI*. 2015. S.l.: s.n.

54. Universidad internacional abierta y a distancia. [en línea], 2018. Disponible en: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html.

55. Universidad Internacional de Valencia. [en línea], 2018. Disponible en: http://www.viu.es/caracteristicas-tipos-y-plataformas-mas-utilizadas-para-estudiar-adistancia/.

56. Visual Paradigm. *Visual Paradigm* [en línea], 2012. Disponible en: http://www.visual-paradigm.com/.

57. X ALBALADEJO, 2012. RUP (Proceso Unificado de Desarrollo). [en línea]. Disponible en: http://www.proyectosagiles.org/que-es-scrum.

58. XAUCE ZERA, 2018. Plataforma educativa. [en línea]. Disponible en: https://eva.uci.cu/es/aboutAs.

59. XP (Programación Extrema). [en línea], 2017. Disponible en: http://ingenieriadesoftware.mex.tl/52753_xp---extreme-programing.html.

ANEXO 1

Tabla 9 Descripción de la HU eliminar evento

Número: 3	**Nombre del requisito:** Eliminar evento
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** El sistema debe permitirle al usuario con el rol profesor eliminar un evento de tipo personal, de curso y grupo y al usuario con rol de estudiante, sólo eliminar un evento de tipo personal. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar un evento es necesario: - Estar autenticado en el sistema. - Debe existir en el sistema al menos un evento que haya sido creado por él. **3- Flujo de la acción a realizar:** - El sistema permite eliminar un solo evento, se puede realizar seleccionando la opción Eliminar de las opciones que muestra la vista previa del evento. Posteriormente se muestra una vista con la interrogante ¿Está seguro que desea eliminar el elemento seleccionado? Si selecciona Aceptar se eliminará el evento y si selecciona Cancelar regresa a la vista anterior. El sistema muestra un mensaje de información.	
Observaciones: Estar autenticado en el sistema.	

Prototipo de interfaz:

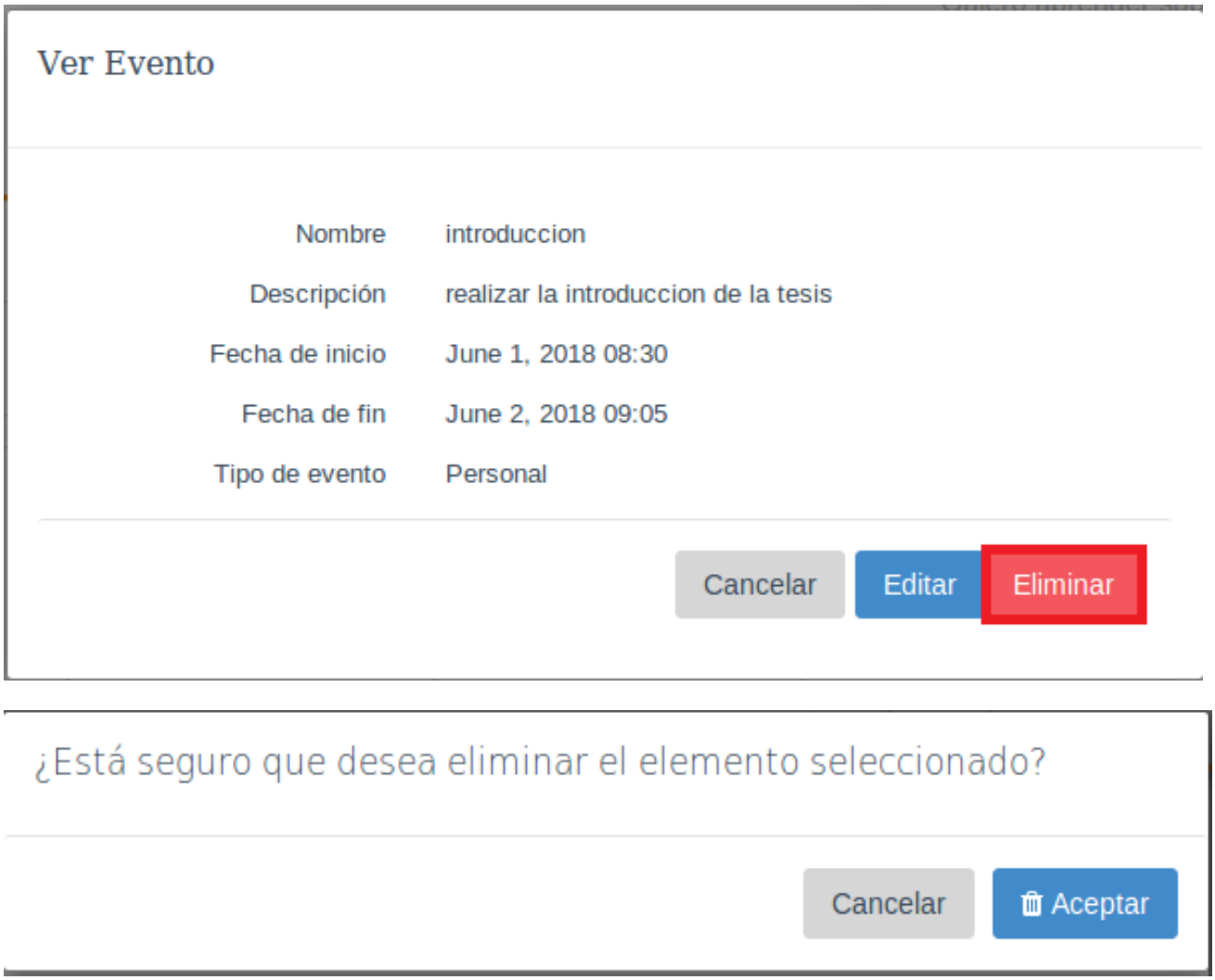


Tabla 10 Descripción de la HU listar eventos

Número: 5	**Nombre del requisito:** Listar eventos
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era

Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

El sistema debe permitirle al usuario listar los eventos de tipo personal, de curso y grupo.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para listar eventos es necesario:

- Estar autenticado en el sistema.

3- Flujo de la acción a realizar:

- Cuando el usuario selecciona la opción Listar eventos aparece una tabla con todos los eventos registrados que sean asociados a él. Además, el usuario tiene la posibilidad de ver los detalles de los eventos. Si el sistema tiene eventos se mostrarán y si no tiene eventos se mostrará la siguiente información con el texto: Ningún dato disponible en esta tabla.

Observaciones: Estar autenticado en el sistema.

Prototipo de interfaz:

Tabla 11 Descripción de la HU buscar evento

Número: 7	**Nombre del requisito:** Buscar evento
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** El sistema debe permitirle al usuario buscar eventos en el sistema de tipo personal, de curso y grupo. **2- Acciones para lograr el objetivo (precondiciones y datos):**	

Para buscar un evento es necesario:

- Estar autenticado en el sistema.

3- Flujo de la acción a realizar:

- Cuando el usuario desea buscar un evento debe seleccionar la opción de Listar eventos El campo de texto de la opción Buscar permitirá escribir el nombre del evento que se desee buscar. Si el evento es encontrado se mostrará y si no es encontrado se mostrará la siguiente información con el texto: No existe ninguna coincidencia.

Observaciones: Estar autenticado en el sistema.

Prototipo de interfaz:

Tabla 12 Descripción de la HU notificar evento

Número: 8	**Nombre del requisito:** Notificar evento
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era

Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

El sistema debe permitir notificar al usuario una vez creado un evento de tipo de curso o grupo.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para notificar al usuario es necesario:

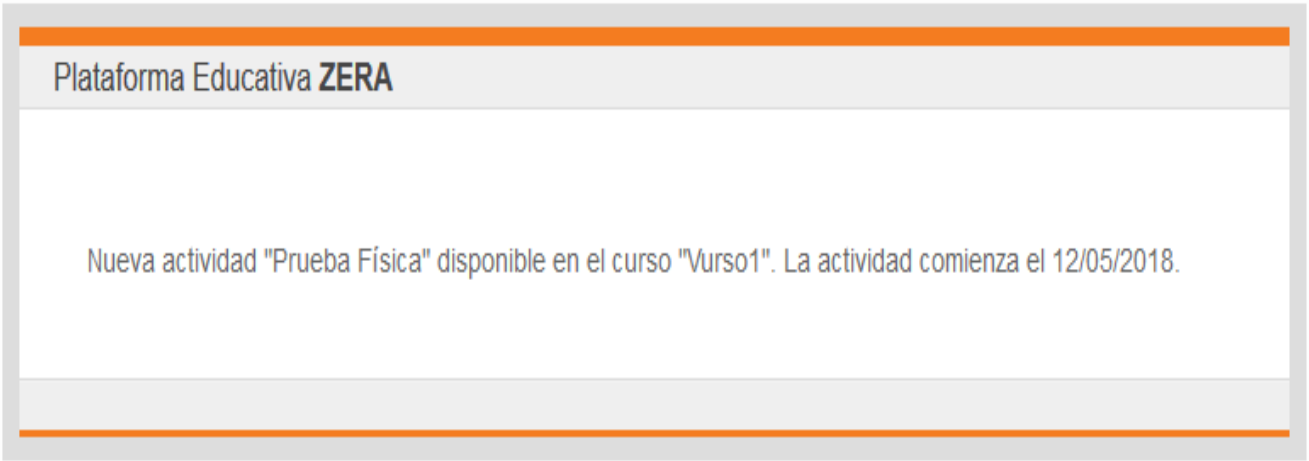
- Estar autenticado en el sistema.

3- Flujo de la acción a realizar:

- El usuario es notificado cuando el profesor crea o modifica un evento de grupo o de curso y él pertenece a ese grupo o curso.

Observaciones: Estar autenticado en el sistema.

Prototipo de interfaz:



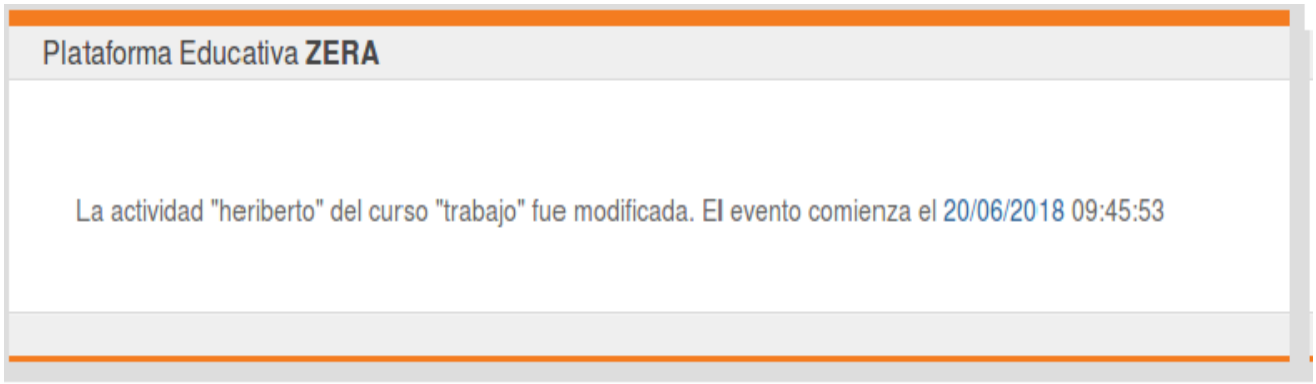


Tabla 13 Descripción de la HU exportar agenda

Número: 9	**Nombre del requisito:** Exportar agenda
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

El sistema debe permitirle al usuario exportar los eventos de tipo personal, de curso y grupo.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para exportar la agenda es necesario:

- Estar autenticado en el sistema.

- Debe existir en el sistema al menos un evento que sea accesible por él.

3- Flujo de la acción a realizar:

- Cuando el usuario selecciona el botón de Exportar agenda se mostrará la ventana de guardar la agenda, en caso de aceptar se guardará en el lugar que desee en el formato ICS y en caso de cancelar regresará a la vista anterior.

Observaciones: Estar autenticado en el sistema.

Prototipo de interfaz:

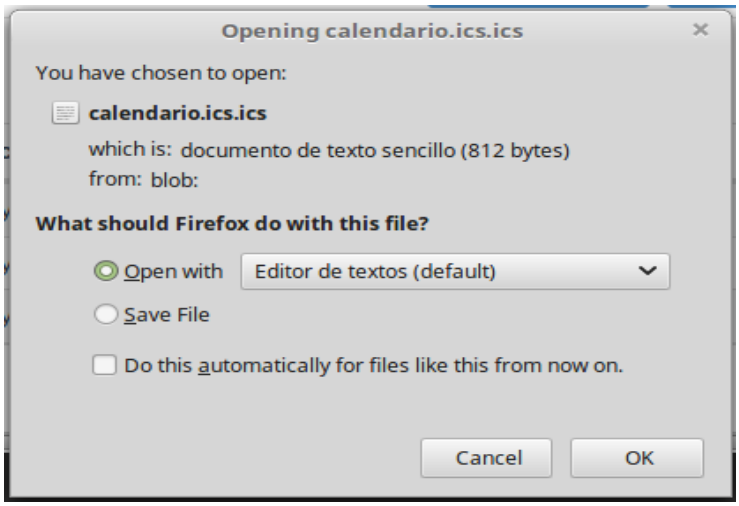
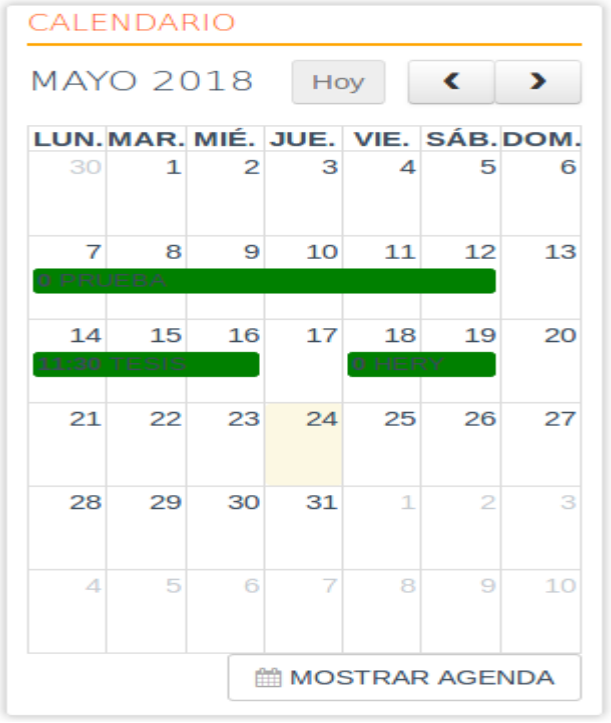


Tabla 14 Descripción de la HU mostrar agenda

Número: 10	**Nombre del requisito:** Mostrar agenda
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** El sistema debe permitir mostrar la agenda al usuario. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para mostrar la agenda es necesario: - Estar autenticado en el sistema. **3- Flujo de la acción a realizar:** - Muestra un calendario en la página inicial del sistema en el lado izquierdo y en la página de curso con la opción de Mostrar agenda. - Cuando el usuario selecciona la opción de Mostrar agenda aparecerá el calendario utilizando toda la pantalla.	
Observaciones: Estar autenticado en el sistema.	
Prototipo de interfaz:	


CALENDARIO
MAYO 2018
Hoy
LUN. MAR. MIÉ. JUE. VIE. SÁB. DOM.
30 1 2 3 4 5 6
7 8 9 10 11 12 13
PRUEBA
14 15 16 17 18 19 20
TESIS
HERY
21 22 23 24 25 26 27
28 29 30 31 1 2 3
4 5 6 7 8 9 10
MOSTRAR AGENDA

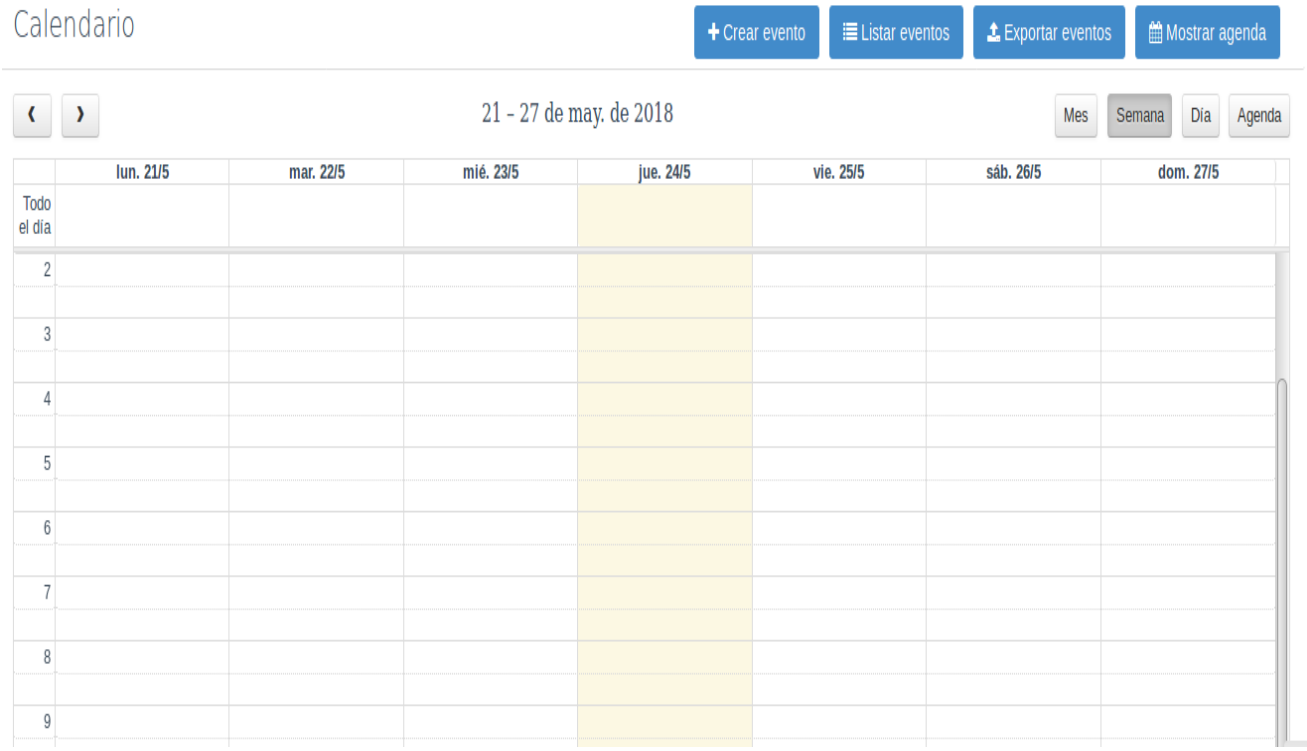
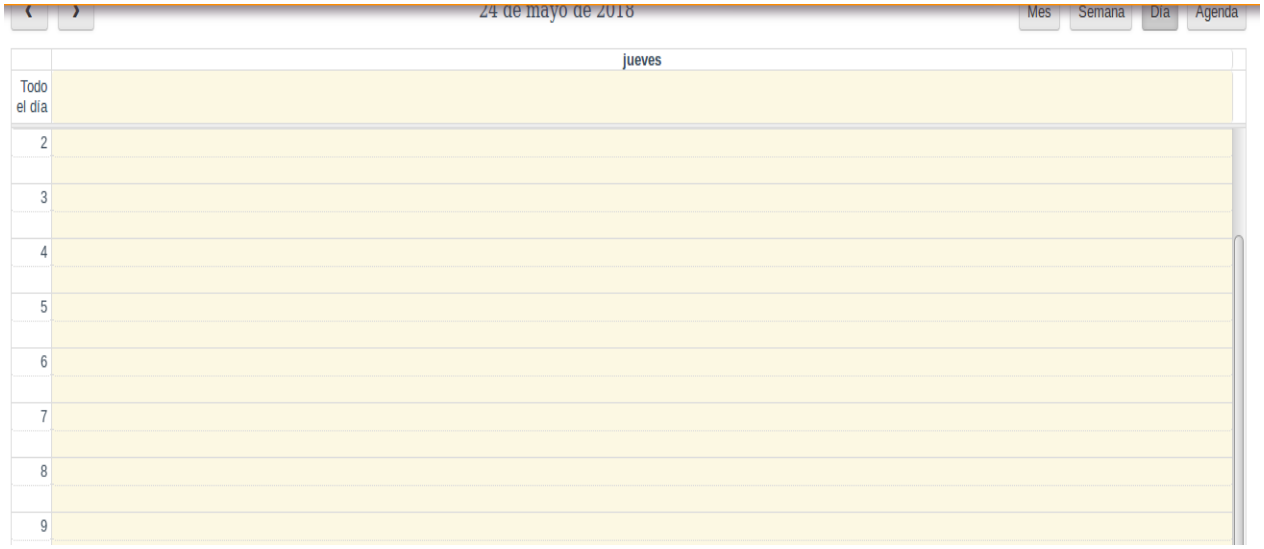
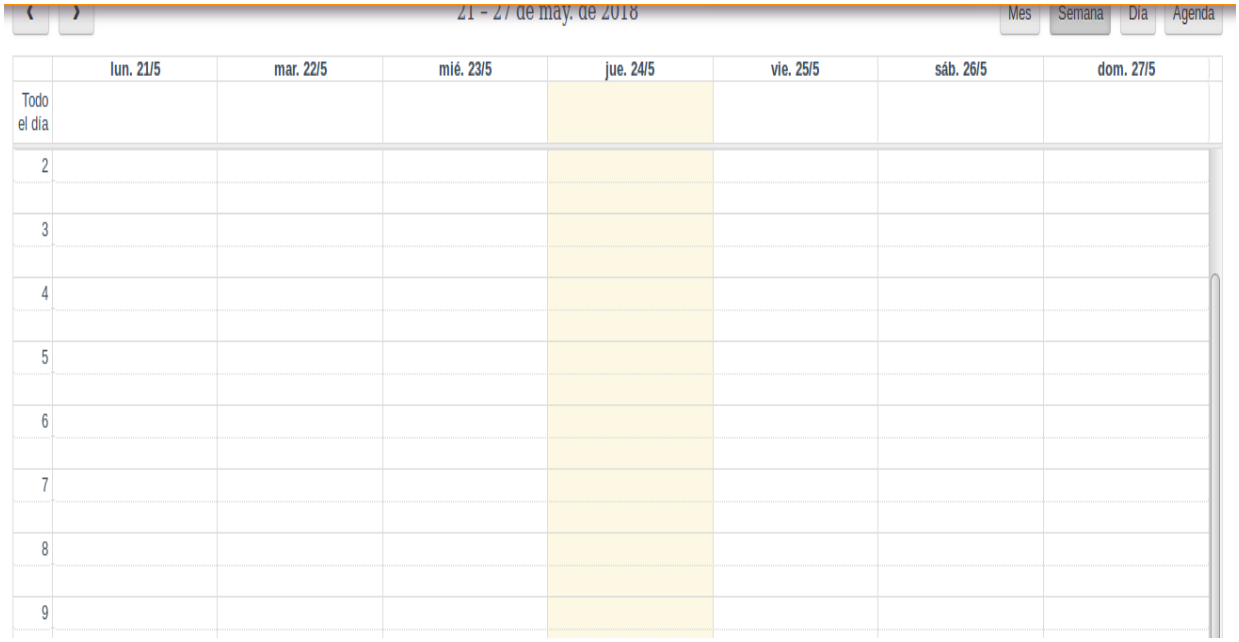
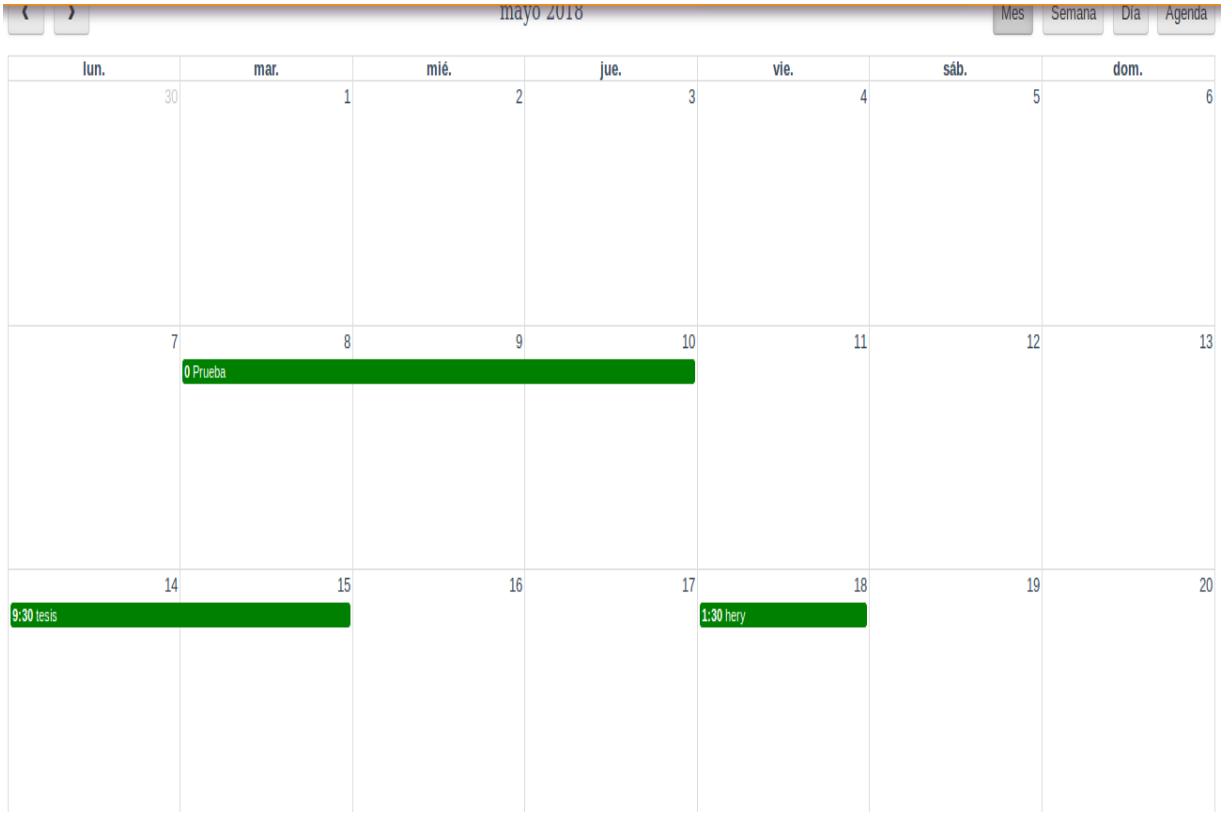

Calendario
Crear evento
Listar eventos
Exportar eventos
Mostrar agenda
21 - 27 de may. de 2018
Mes
Semana
Día
Agenda
lun. 21/5
mar. 22/5
mié. 23/5
jue. 24/5
vie. 25/5
sáb. 26/5
dom. 27/5
Todo el día
2
3
4
5
6
7
8
9

Tabla 15 Descripción de la HU cambiar apariencia de la agenda

Número: 11	**Nombre del requisito:** Cambiar apariencia de agenda
Programador: Heriberto Sarmiento González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** El sistema debe permitir cambiar la apariencia de la agenda. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para cambiar apariencia de agenda hay que: - Estar autenticado en el sistema. - Debe mostrarse la agenda en toda la pantalla. **3- Flujo de la acción a realizar:** - El usuario puede seleccionar según el desee la apariencia de la agenda que puede ser en días, semanas o meses.	
Observaciones: Estar autenticado en el sistema.	
Prototipo de interfaz:	


24 de mayo de 2018
Mes
Semana
Día
Agenda
jueves
Todo el día
2
3
4
5
6
7
8
9


21 - 27 de may. de 2018
Mes
Semana
Día
Agenda
lun. 21/5
mar. 22/5
mié. 23/5
jue. 24/5
vie. 25/5
sáb. 26/5
dom. 27/5
Todo el día
2
3
4
5
6
7
8
9


mayo 2018
Mes
Semana
Día
Agenda
lun.
mar.
mié.
jue.
vie.
sáb.
dom.
30
1
2
3
4
5
6
7
8
9
10
11
12
13
0 Prueba
14
15
16
17
18
19
20
9:30 tesis
1:30 hery

Anexo 2

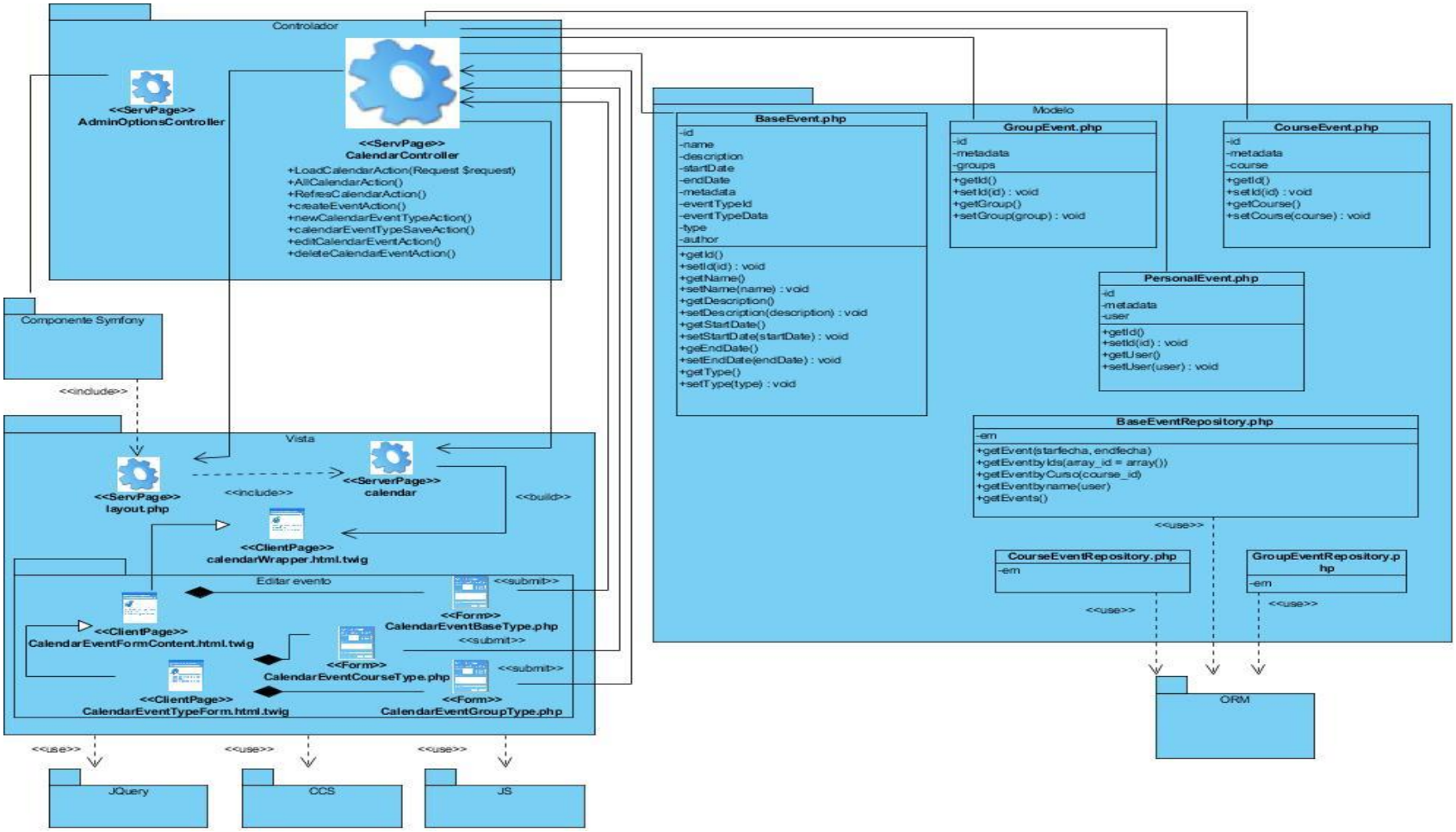


Figura 26 Diagrama de clase de diseño. Editar evento

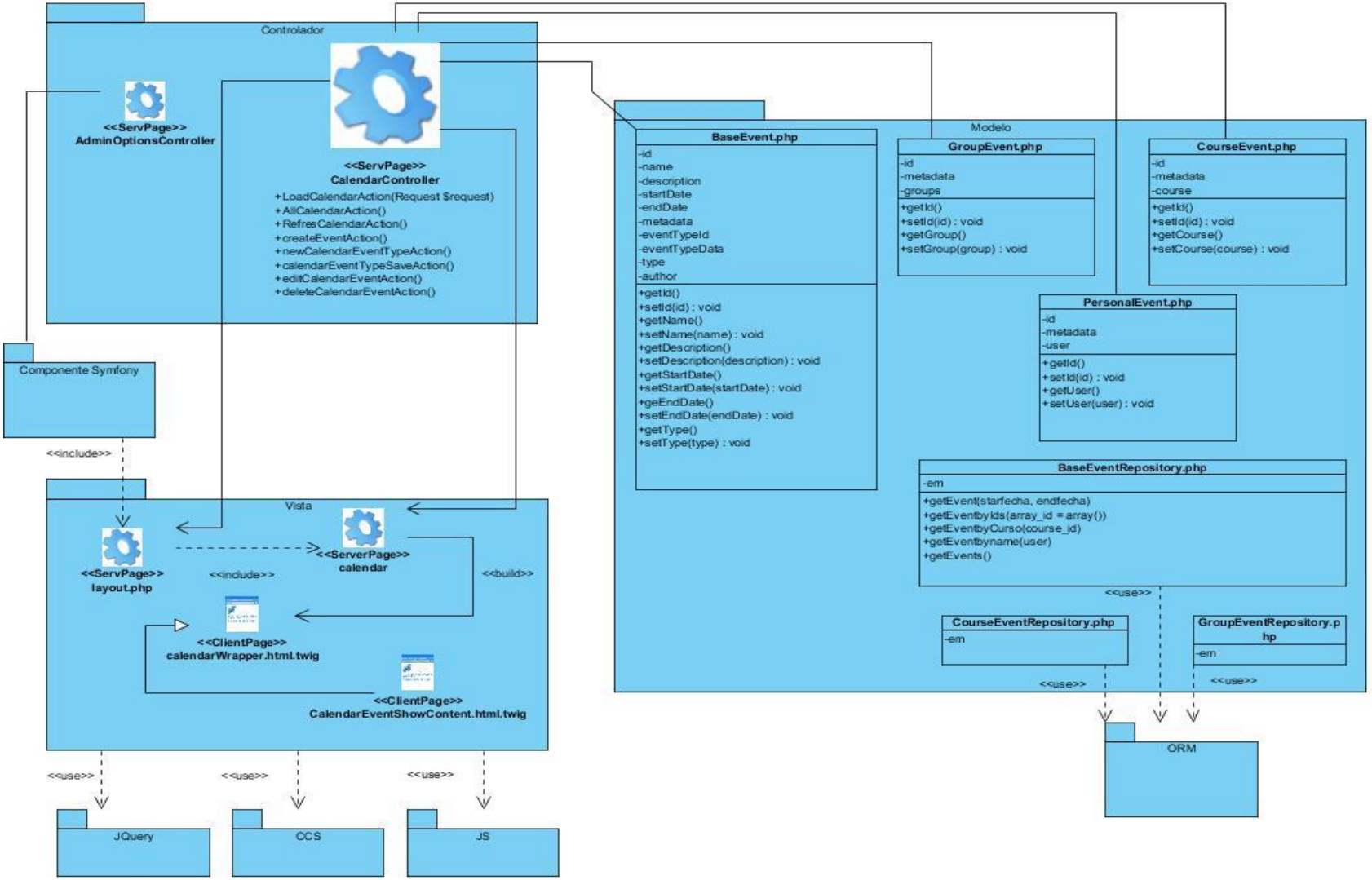


Figura 27 Diagrama de clase de diseño. Mostrar evento

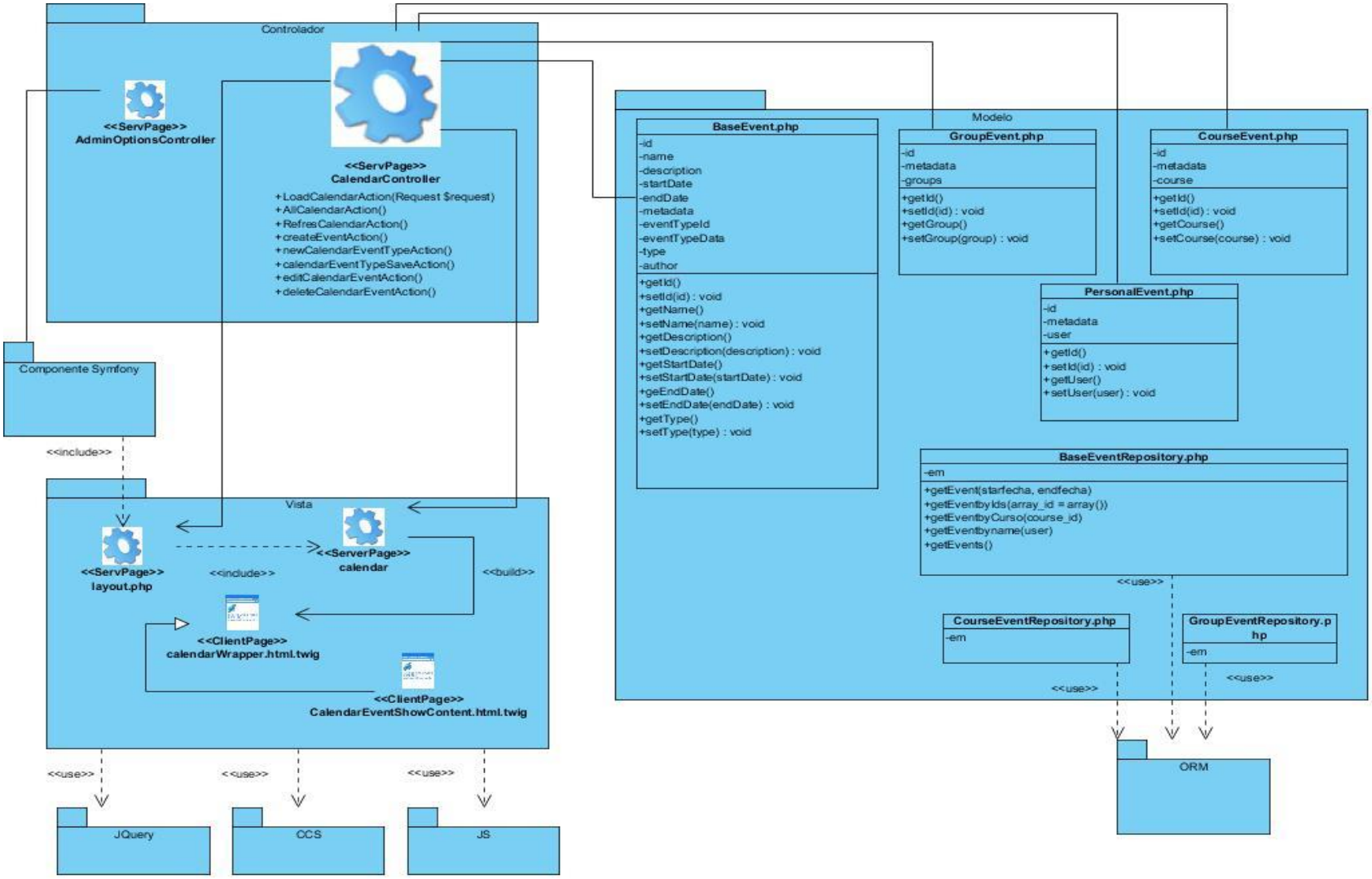


Figura 28 Diagrama de clase de diseño. Eliminar evento

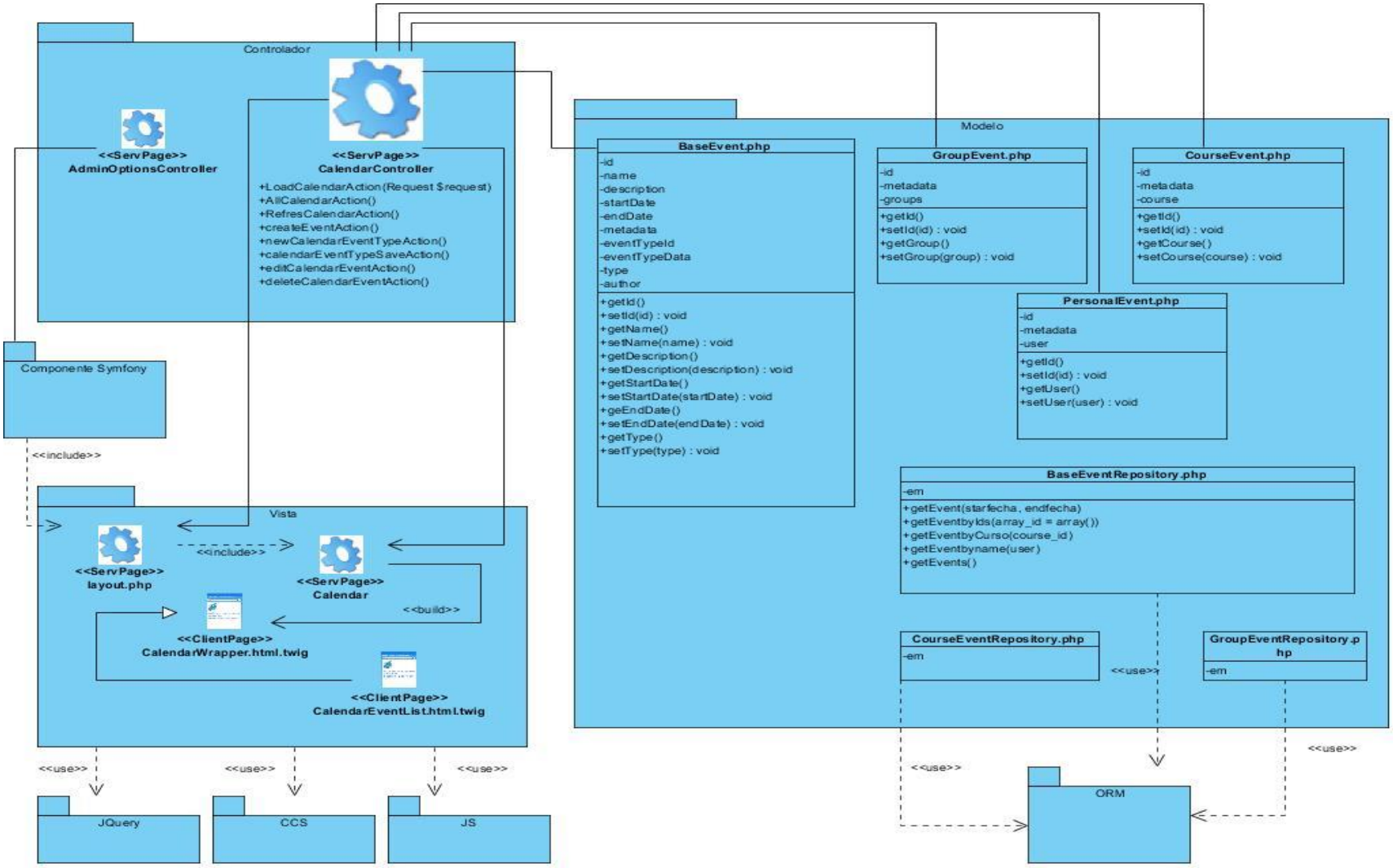


Figura 29 Diagrama de clase de diseño. Buscar evento

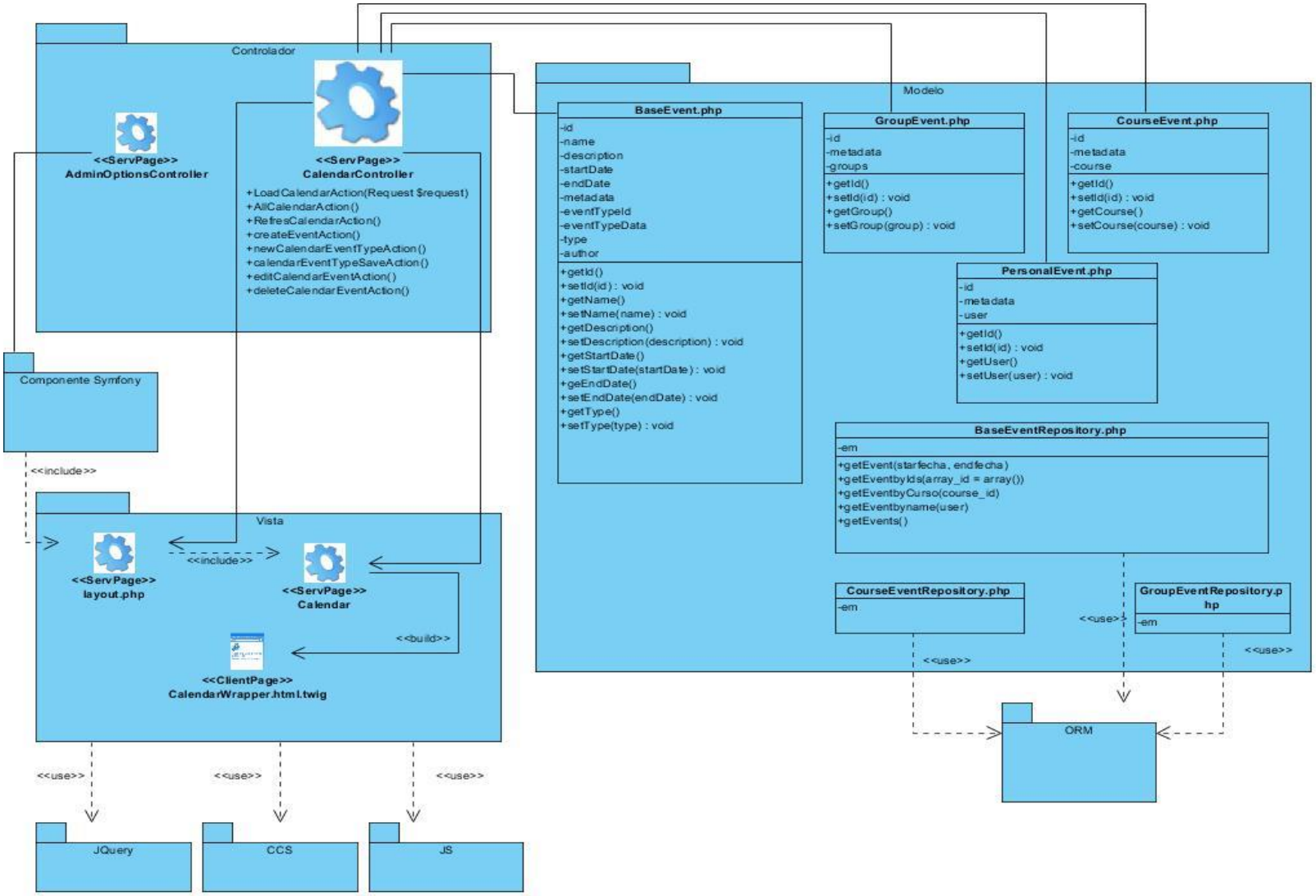


Figura 30 Diagrama de clase de diseño. Exportar evento

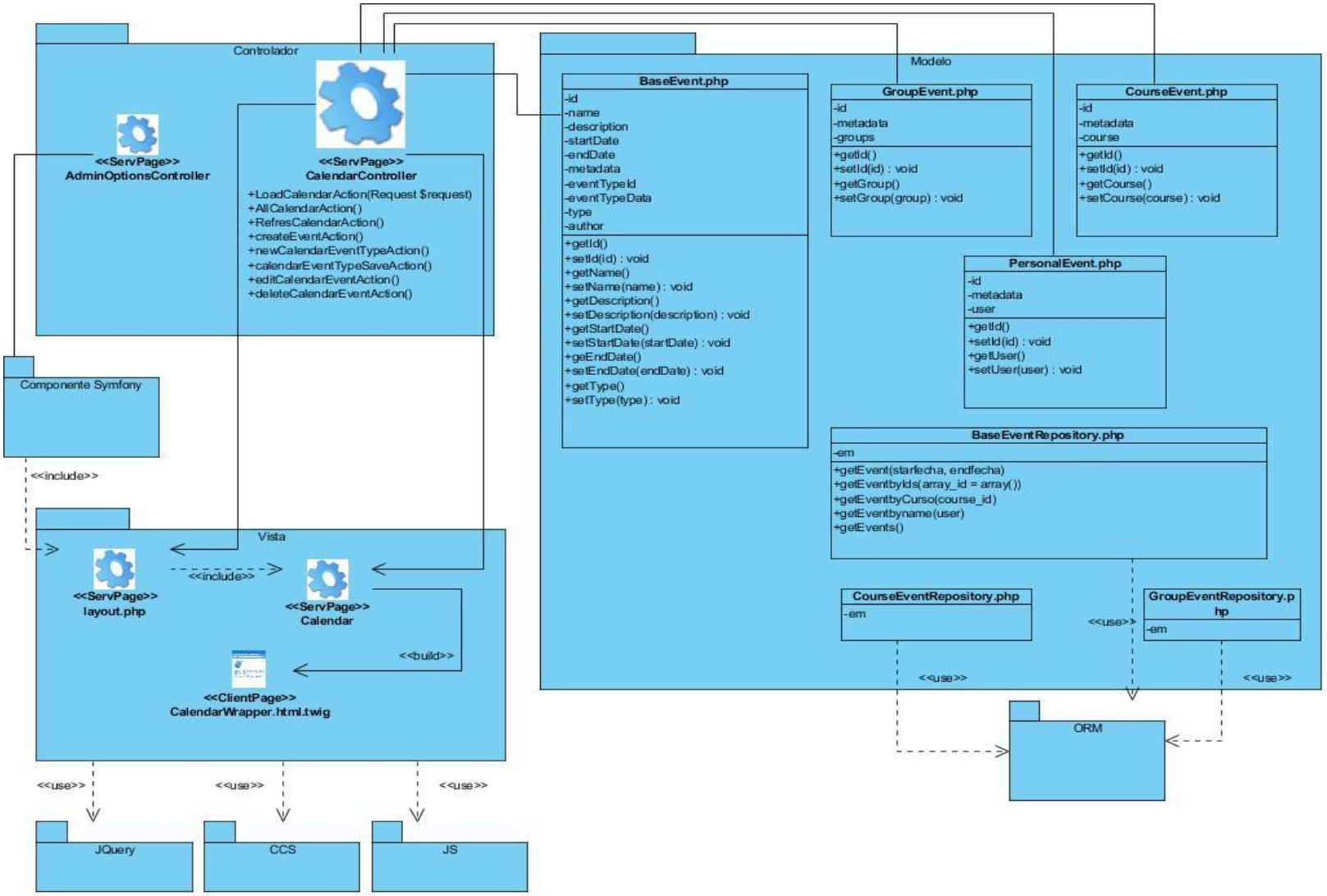


Figura 31 Diagrama de clase de diseño. Cambiar apariencia

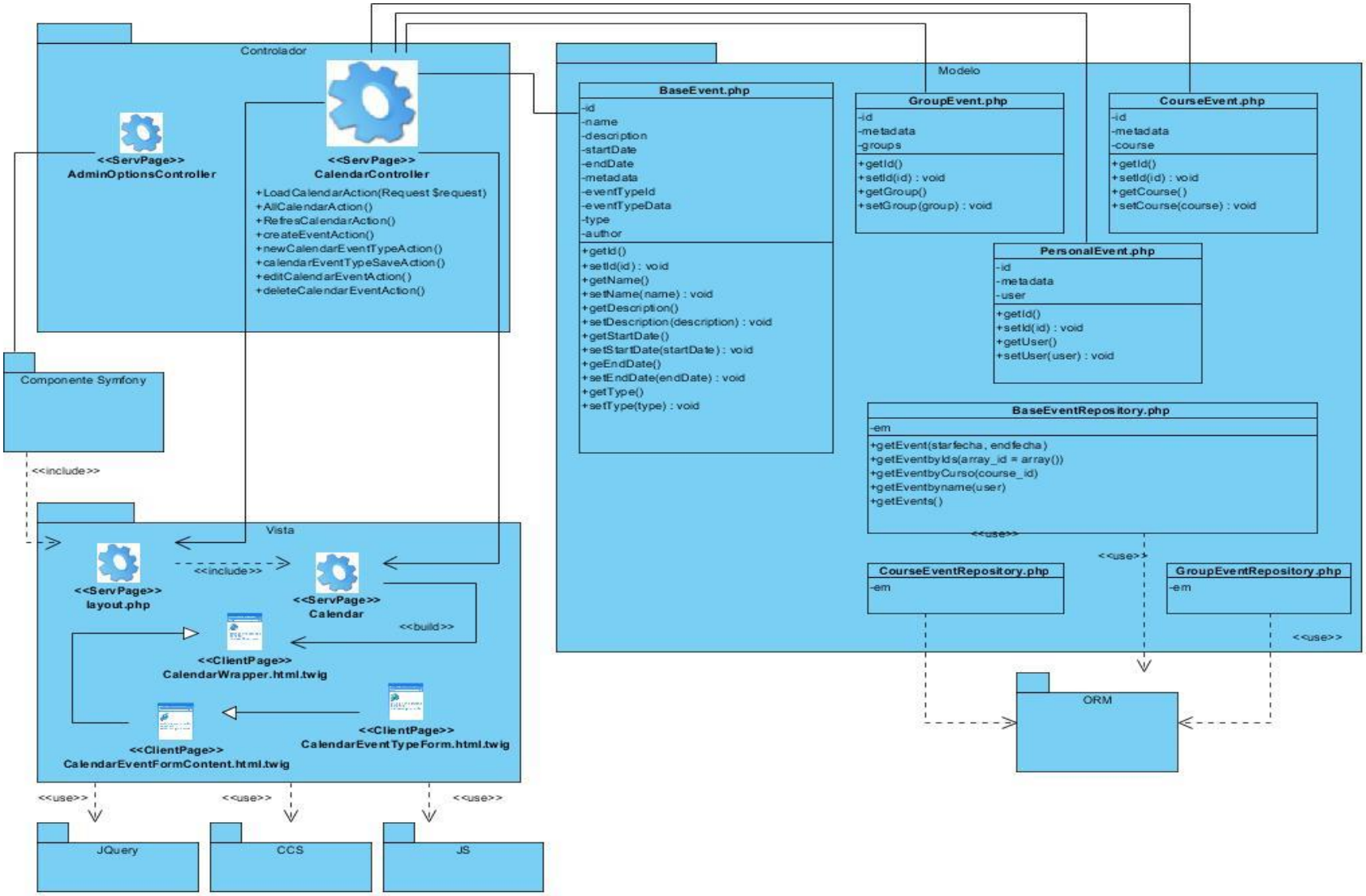


Figura 32 Diagrama de clase de diseño. Notificar evento

Anexo 3

Tabla 16 Caso de prueba eliminar evento

CP Eliminar evento
Descripción general La funcionalidad inicia cuando un usuario del sistema decide eliminar un evento creado. Para ello el usuario selecciona en el calendario el evento que desea eliminar que haya sido creado por él, se muestra una vista con los datos correspondientes al evento y selecciona la opción Eliminar. Posteriormente se muestra una vista con la interrogante ¿Está seguro que desea eliminar el elemento seleccionado? Si selecciona Aceptar se elimina el evento y si selecciona Cancelar regresa a la vista anterior. Una vez eliminado el evento se muestra una notificación de validez. **Condiciones de ejecución** La persona debe estar autenticada.

Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC4.1 Opción seleccionar	Selecciona en el calendario el						Se muestra una vista con los siguientes campos y sus datos	Calendario/Evento

evento	evento que desea eliminar.						correspondiente: • Nombre • Descripción • Fecha de inicio • Fecha de fin • Tipo de evento	
EC4.2 Opción Eliminar	Selecciona la opción “Eliminar”						Se muestra una vista con la interrogante ¿Está seguro que desea eliminar el elemento seleccionado? y las opciones de “Aceptar” y “Cancelar”.	Calendario/Evento /Eliminar
EC4.3 Opción	Selecciona la opción						Elimina el evento y muestra un mensaje	Calendario/Evento/ Eliminar/Aceptar

Aceptar	"Aceptar"						de información: Se ha eliminado el elemento satisfactoriamente.		
EC4.4 Opción Cancelar	Selecciona la opción "Cancelar"						Regresa a la vista anterior.	Calendario/Evento/ Eliminar/Cancelar	

Tabla 17 Caso de prueba listar eventos

CP Listar eventos								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide listar los eventos creados. Para ello el usuario selecciona en el calendario la opción de Listar eventos. El sistema muestra un listado con todos sus eventos. **Condiciones de ejecución** La persona debe estar autenticada.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC5.1 Opción Listar eventos	Selecciona en el calendario la opción de Listar eventos						El sistema muestra la lista de los eventos.	Calendario/Listar eventos

EC5.2 No existen eventos	No hay eventos en el sistema.						Se muestra la lista vacía con el mensaje de información: Ningún dato disponible en esta tabla.	Calendario/Listar eventos
EC5.3 Opción Buscar	El usuario introduce datos.						El sistema permite buscar los datos del evento. Ver DCP _Buscar evento.	Calendario/Listar eventos/Buscar

Tabla 18 Caso de prueba notificar evento

CP Notificar evento
Descripción general La funcionalidad inicia cuando un usuario del sistema decide crear un evento de grupo o de curso. Para ello el usuario crea el evento y el sistema manda un correo de notificación a los estudiantes que pertenecen a ese grupo o curso. **Condiciones de ejecución** La persona debe estar autenticada con el rol de profesor principal del curso o como profesor de un grupo.

Escenario	Descripción	Nombre	Descripción	Fecha de inicio	Fecha de fin	Tipo de evento	Respuesta del sistema	Flujo Central
EC6.1 Enviar notificación	Enviar una notificación al usuario.						El sistema envía un correo de notificación a los estudiantes que pertenecen a ese grupo o curso.	Calendario/Crear evento/Guardar/ Notificar

Tabla 19 Caso de prueba buscar evento

CP Buscar evento
Descripción general La funcionalidad inicia cuando un usuario del sistema decide buscar un evento. Para ello el usuario selecciona en el calendario la opción de Listar eventos. El sistema muestra un listado con todos sus eventos y escribe el nombre del evento que desea buscar en el campo de texto de la opción Buscar. **Condiciones de ejecución** La persona debe estar autenticada.

Escenario	Descripción	Nombre	Descripción	Fecha de inicio	Fecha de fin	Tipo de evento	Respuesta del sistema	Flujo Central
EC7.1 Opción Buscar	El usuario introduce datos.						El sistema permite buscar los datos del evento.	Calendario/Listar eventos/Buscar
EC7.2 No existe coincidencia	No existe el evento asociado a los datos introducidos en el sistema.						El sistema no encuentra coincidencia con los datos introducidos y muestra la lista vacía con el mensaje de información: No existe ninguna coincidencia.	Calendario/Listar eventos/Buscar

Tabla 20 Caso de prueba exportar agenda

CP Exportar agenda								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide exportar la agenda. Para ello el usuario selecciona en el calendario la opción de Exportar agenda. El sistema muestra una vista con la opción de OK para exportar los eventos en el formato ICS y la opción Cancel para regresar a la vista anterior. **Condiciones de ejecución** La persona debe estar autenticada.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC8.1 Opción Exportar eventos	Selecciona en el calendario la opción de Exportar eventos.						El sistema muestra una vista con la opción de OK o Cancel.	Calendario/Exportar eventos

EC8.2 Opción OK	Selecciona la opción de OK.						El sistema exportan los eventos en el formato ISC.	Calendario/Exportar eventos/OK
EC8.3 Opción Cancel	Selecciona la opción de Cancel.						Vuelve a la vista anterior.	Calendario/Exportar eventos/Cancel
EC8.4 No existen eventos	No hay eventos en el sistema.						El sistema no muestra la vista que permite exportar los eventos en el formato ISC.	Calendario/Exportar eventos

Tabla 21 Caso de prueba mostrar agenda

CP Mostrar agenda								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide ver la agenda. Para ello el usuario selecciona en la vista inicial del sistema la opción de Mostrar agenda. El sistema muestra una vista con el calendario. **Condiciones de ejecución** La persona debe estar autenticada.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC9.1 Opción Mostrar agenda	Selecciona la opción de Mostrar agenda						El sistema muestra una vista con el calendario.	Mostrar agenda/ Calendario

Tabla 22 Caso de prueba cambiar apariencia de la agenda

CP Cambiar apariencia de la agenda								
Descripción general La funcionalidad inicia cuando un usuario del sistema decide cambiar la apariencia de la agenda. Para ello el usuario selecciona en el calendario las opciones de ver el calendario que son en Mes, Semanas y Días. **Condiciones de ejecución** La persona debe estar autenticada.								
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de inicio**	**Fecha de fin**	**Tipo de evento**	**Respuesta del sistema**	**Flujo Central**
EC10.1 Opción Mes	Selecciona en el calendario la opción de Mes.						El sistema muestra una vista con el calendario en forma de mes.	Calendario/Mes
EC10.2 Opción	Selecciona en el calendario la						El sistema muestra una	Calendario/Semana

Semana	opción de Semana.						vista con el calendario en forma de semana.		
EC10.3 Opción Días	Selecciona en el calendario la opción de Días.						El sistema muestra una vista con el calendario en forma de días.	Calendario/Días	