Universidad de las Ciencias Informáticas

Facultad 3

Título: Paquete de solicitudes de cambio del procedimiento Ordinario Penal del Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos.

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Rosalia Rodríguez Labrada

Julio Luis Rivera Abreu

Tutor: Ing. Darián González Ochoa

Cotutores: Ing. Yaiset Moreno Aleaga

Ing. Yosviel Domínguez González

La Habana, Cuba

Curso: 2017-2018

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Autores:

Rosalia Rodríguez Labrada

Julio Luis Rivera Abreu

Tutor:

Ing. Darián González Ochoa

Cotutores:

Ing. Yaiset Moreno Aleaga

Ing. Yosviel Domínguez González

Agradecimientos

A la revolución cubana y al Comandante en Jefe Fidel Castro por crear esta hermosa Universidad y permitirme, aun viniendo de una familia humilde, formarme como ingeniero en ciencias informáticas.

A mis padres les pido perdón por cado uno de los momentos difíciles que pasaron por mi causa y les agradezco por confiar en mi aun cuando otros no lo hicieron, por no perder la fe en mí, por su apoyo incondicional y por formarme en los valores que llevaré conmigo toda la vida. Los amo con todo mi corazón.

A mi novio o como otros dicen: a mi “marido”, por ayudarme a llegar a la meta, por sus consejos, por su calma, por hacer que me propusiera objetivos más altos en mi formación, que sepa que mis padres y yo le estaremos eternamente agradecidos por todo lo que hizo por mí, si hoy estoy aquí en gran parte, es gracias a él. Te amo y te amaré siempre.

A mi compañero de tesis, que al final se convirtió en un amigo para toda la vida, que sin importar las peleas y lo obstinada que yo pueda llegar a ser y a pesar de todos mis defectos me quiere tal y como soy. Te quiero mucho.

A mi mejor amiga, Adis Mirta, que más que mi amiga es una segunda madre para mí, que nunca me ha fallado, que peleamos, pero nos queremos muchísimo, por su apoyo y su cariño. Te quiero mucho.

A mi hermano, a mis abuelos y al resto de mi familia que sepan que los tengo presente en este día tan importante de mi vida.

A mis amigas de la Universidad son las mejores, las quiero mucho Anayanci, Yuliet, Yaima, Silvia (la tata), Yamilé, Mairena, Irina y Roxana.

A mis amigos que algunos están a mi lado, pero otros están lejos por su trabajo, y que de una forma u otra siempre están a mi lado: Mario, Nadir, Roberto Carlos, Luis Angel, Roberto, Mark, Ariel y a una nueva adquisición a la que le tengo un cariño especial de algo que nace ahora Yoslenis.

A dos personas especiales sin las cuales no me hubiese podido graduar porque me ayudaron con mis peores pesadillas, el inglés y la tesis: El Pingüino Luis y al Mejor consultor de tesis Reynier, gracias por todo. Los quiero muchísimo.

A mis profesores y todas las personas tan lindas de la regional que tanto me ayudaron durante esa etapa de mi vida especialmente a: Wilfredo, Fidel, Yamila y Danilo.

A mis profesores de la UCI, por ayudarme y guiarme para llegar hasta el final de mi carrera especialmente a: Gaspar, Olga, Lieen, Jorge, Anel, Yadian, Liset y los que no menciono, tengan presente que, aunque no esté su nombre los quiero y les agradezco a todos.

A mis tutores Darian, Yaiset y Yosviel por su apoyo y consideración para la realización de este trabajo.

A mis compañeros de aula que están presentes por compartir este momento tan importante.

A Mailen y Osvaldo, Dailien y Polanco y Eliober y Nairovis, por permitirme formar parte de sus familias, por su cariño, por ayudarme, los quiero mucho.

A Adelina y Gricel que también han sido como mis madres en este tiempo, por sus consejos, por su ayuda, por su amor, muchas gracias.

A las tías del comedor por su cariño y por su constante preocupación por mi especialmente a Chichi.

A mis amigos de la FEU, con los que he compartido tantos buenos y malos momentos: Claudia, Victor, Jean, Guisao, Ale, Haraid, Dorita, Claudia, Juan Gabriel y Alvaro.

Rosalia

A mi mamá, por hacer de mi un hombre de bien, por siempre brindarnos a mí y a mis hermanas todo lo que estuvo a su alcance, por todos los sacrificios que has hecho durante toda tu vida para que no nos faltara nada. Gracias por todo lo que me diste y me sigues dando.

A mi papá por ser mi ejemplo, por su apoyo incondicional, por hacer de mi sueño su sueño, por sus consejos cuando en algún momento intenté abandonar, gracias a ti hoy estoy aquí. Te quiero con todo el corazón.

A mi hermana mayor Yunet por todo el apoyo que me brinda a diario, por traer al mundo al negrito que más quiero en esta vida, por ser mi tata la mejor del mundo. Gracias por ser como eres, por quererme tanto, por estar en los buenos y en los malos momentos. Esta persona que soy hoy, también es gracias a ti.

A mi hermana Mariset por regalarme dos sobrinos bellos, por ser la hermana del carácter, por sus consejos. Gracias por todo lo que me diste y me sigues dando.

A mi hermano Randier por su apoyo. Gracias por tus consejos de ingeniero, ya somos dos en la familia.

A mi segunda familia. Al que siempre fue y será un segundo padre para mí: Rogelio. Gracias por inculcarme siempre los valores revolucionarios, por confiar en mí, por quererme siempre incondicionalmente. Ojalá y Dios te dé mucha salud para que nos sigas brindando a todos los que te queremos todo tu amor. Te quiero inmensamente. A Miriela por su fuerza de voluntad, por mirar siempre al frente y seguir luchando por tu familia. Gracias por apoyarme siempre, por ser mi segunda madre. Gracias por tener un padre tan grande y maravilloso, y por tener a un hijo que es como un hermano para mí.

A la familia Vázquez: Adela y Jaime por ser las personas más buenas que he conocido. Por todo el apoyo que me brindan a diario. Gracias Jaime por llegar a la vida de mi hermana en su momento y hacerla feliz, todavía no pierdo las esperanzas de que todo vuelva a ser como antes. Gracias Adela por convertirte en la mamá de los pollitos. Te admiro mucho como persona y pido para ti mucha salud para que puedas gozar de tus nietos y de los que vendrán porque eres una súper abuela.

A mis abuelos que Dios los tenga en la gloria: Teresa, Lucia, Luis, Felipe y Eumelia. Gracias por todo lo que me brindaron en vida y sé que donde quiera que estén, se sienten orgullosos de mí y les hubiera gustado estar aquí hoy celebrando este triunfo conmigo. Los quiero.

A mis tutores: Yaiset, Yosviel y Darián por su entrega y su paciencia en la realización de este trabajo.

A todos los profesores que han contribuido a mi formación como profesional, en especial a las profesoras(es) Mailen Edith, Dailien Moré, Polanco, Dariela Elvira, Rosalina Ibarra, Mónica Cortina, Eliober Cleger, Yorguy Batista, Yadian Betancourt y Liset Polanco.

A la vieja más loca que tiene el docente 2: Adelina. Gracias por hacer los días más a menos. Por tus buenos consejos y por tus locuras. Mucha salud para ti. También a la tía más buena que tiene el complejo comedor 1: Chichi. Gracias por ser tan cariñosa y por estar siempre atenta a nosotros.

A mi team Discrepo y mis compañeros de batalla de la FEU: Haraid, Alex, Hean Carlos, Claudia, Victor, José Luis, Angel, Oslaine, Eddy y Sael. Gracias por compartir conmigo tantos momentos buenos, gracias por ser mi familia aquí en la UCI y sepan que de cada uno me llevo lo mejor.

A todos mis compañeros del aula y amigos que siempre voy a respetar, recordar y querer, por ayudarme y acompañarme en momentos difíciles y alegres, especialmente a mis cocuyos Ailyn y Milena, a Juan Gabriel, Yulieth, Esmirna, Leodan, Oscar, Bárbaro, Dalilis, Jenniffer, Pang, Luis Ramiro, Lucio, Anayanci, Dianelis e Ingrid.

A mi compañera de tesis, por ayudarme en la realización de este trabajo, y ser mi amiga incondicional. Gracias por todo lo que hiciste por mí, por las broncas diarias, por los jalones de oreja. Poco a poco te convertiste en una de las personitas más importantes para mi dentro de esta Universidad. Te quiero con el alma. Puedes contar conmigo siempre.

A todos los compañeros que por diversas razones no pudieron continuar junto a mí en este camino: Claudia Margarita, Danailys, Hector, Dexter, Abduan, Andy. Gracias por los momentos vividos.

Para finalizar quiero agradecerle a una persona muy especial. Una persona que supo ganarse un lugar en nuestros corazones por su forma de ser. A ti te agradezco por ser como eres, por tu amistad incondicional, por tu entrega como profesional, por tu forma de inculcar conocimiento, por ayudarnos a sacar esta tesis adelante. Este logro es gracias a ti: Reinier Fernández Coello, ya eres para mí más que un profesor, más que un amigo, eres mi hermano.

Julio

Dedicatoria

A toda nuestra familia por su apoyo, en especial a nuestros padres, hermanos y abuelos.

A nuestro comandante en jefe Fidel Castro y a la Revolución.

RESUMEN

Como parte del esfuerzo por impulsar el desarrollo de la informática en Cuba, al Centro de Gobierno Electrónico adscrito a la Universidad de las Ciencias Informáticas se le asigna la tarea de informatizar los procesos que se llevan a cabo en los Tribunales Populares Cubanos. Como resultado de esta colaboración se obtuvo el Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos. Para ayudar al uso de este sistema se impartieron cursos de capacitación a todos los involucrados en el mismo desde los administradores del sistema, hasta los jueces y secretarias. A partir de esta interacción se han propuesto un número considerable de mejoras o aspectos que los especialistas funcionales del sistema no tuvieron en cuenta en el primer levantamiento de información que se hizo para su desarrollo. Por tal motivo la presente investigación tiene como objetivo desarrollar el paquete de solicitudes de cambio del procedimiento Ordinario Penal para satisfacer las necesidades de modificación demandadas por los especialistas funcionales del mismo. Para el desarrollo de la propuesta de solución se analizaron los elementos teóricos que fundamentan la investigación, así como la metodología de desarrollo, las herramientas y tecnologías utilizadas durante el ciclo de vida del sistema. Además, se obtuvo el modelado de sistema, implementación y, por último, se validó dicha propuesta a través de las pruebas de software, corroborando que las solicitudes de cambios conciliadas en los talleres se habían llevado a cabo según lo acordado y que el sistema contaba con la calidad para ser utilizado por los especialistas funcionales.

Palabras claves: tecnologías, metodología, informatización, solicitudes de cambio, Tribunales Populares Cubanos.

ÍNDICE

INTRODUCCIÓN

Actualmente, debido al auge de las Tecnologías de la Información y la Comunicación (TIC), la mayor parte de los procesos que se realizan en el mundo están informatizados, lo cual ha permitido una mayor agilidad en la realización de las tareas y también dar solución a muchos de los problemas que representaba el trabajo manual con la información. (De Jesús Reyes, 2013)

Como parte de ese esfuerzo por impulsar el desarrollo de la informática en Cuba, y en particular de la industria del software, se decide crear la Universidad de las Ciencias Informáticas (UCI). A raíz de esto se le ha encomendado al Centro de Gobierno Electrónico (CEGEL) perteneciente a esta universidad, la tarea de informatizar los procesos que se llevan a cabo en los Tribunales Populares Cubanos (TPC). Como resultado de esta colaboración se obtuvo el Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC), que tiene como principal objetivo guiar la impartición de justicia en las distintas instancias municipales y provinciales, hasta llegar a la instancia suprema.

El proceso de implantación de esta solución trae consigo un conjunto de beneficios para el sistema judicial cubano, por ejemplo, la supervisión de las materias jurisdiccionales, ya que se desarrollará desde instancias superiores, sin tener que mover recursos de una localidad a otra. De igual forma la captación de los datos primarios en el lugar de origen, o sea, que cualquier usuario, teniendo en cuenta los permisos que poseen, podrán introducir los datos ellos mismos sin necesidad de mediadores, por ejemplo: fiscales, secretarias, abogados y los propios jueces.

El turnado y returnado automático de asuntos es otro de los beneficios. Anteriormente se realizaba de forma manual, tarea engorrosa teniendo en cuenta la cantidad de información que se debe tener en cuenta para estas dos actividades. Con el SITPC, estas se llevan de forma automática, haciendo los cálculos necesarios y establecidos por la ley, facilitando el trabajo de las personas encargadas de estas actividades. Por otra parte, la numeración única para los expedientes se realiza colocando un número único en los expedientes que no se repita en todo el país, dando lugar a un número de 14 dígitos.

La generación de los documentos a medida que trascurren los procesos judiciales, resulta un trabajo tedioso, debido al cúmulo de documentos que se generan única y exclusivamente de forma manual. Con el sistema esta actividad se realiza de forma dinámica, centrando el proceso en las actividades judiciales y no en el formato, estructura y estándares de los documentos dictados por la ley. De igual manera, de acuerdo con lo establecido por la ley, se garantizará que cada paso que se vaya a dar en medio del proceso sea el correcto, por lo que no habrá cabida a bifurcaciones que no contemplan la ley, favoreciendo la consecutividad del proceso.

Por último y no menos importante, el aviso sobre el vencimiento de los términos y las notificaciones electrónicas, son dos actividades que se manifiestan con el SITPC, ya que se contará con un sistema de señalización advirtiendo los términos que están vencidos, los que se vencen en el día en cuestión y los próximos a vencerse, razón fundamental ya que de acuerdo a lo que por ley está escrito, las violaciones de los términos acarrean otros problemas. Por su parte las notificaciones se realizarán utilizando el propio sistema y no el correo conforme a lo que establece la Ley del Procedimiento Civil, Administrativo, Laboral y Económico, así como la Ley del Procedimiento Penal. (González Ochoa, y otros, 2018)

El SITPC cuenta con módulos que responden a cada una de las materias que se tramitan en los tribunales: Penal, Civil, Administrativo, Laboral y Económico. Dentro de la materia Penal se encuentra el procedimiento Ordinario Penal, el cual contempla las solicitudes hechas por fiscales que deben ser juzgadas según las leyes del Código Penal. Este proceso se encuentra dividido en varios momentos procesales importantes, comenzando por la presentación del Expediente de Fase Preparatoria (EFP) por parte del fiscal que puede incluir cuatro solicitudes: sobreseimiento libre, extinción de responsabilidad penal, artículos de previo y especial pronunciamiento, así como apertura a juicio oral. Una vez que la secretaria le da entrada y el presidente de sala o sección turna el asunto, el expediente queda listo para que se proceda según el tipo de solicitud.

Para ayudar al uso del sistema se impartieron cursos de capacitación a todos los involucrados en el mismo desde los administradores del sistema, hasta los jueces y secretarias. A partir de la interacción de los usuarios con el sistema en estos encuentros y posteriormente en cada tribunal, se han propuesto un número considerable de mejoras o aspectos que los especialistas funcionales no tuvieron en cuenta en el primer levantamiento de información que se hizo para desarrollar dicho sistema. Estas mejoras y cambios van dirigidas principalmente a los documentos que se generan en cada acto procesal del sistema. Además, se refieren a los concursos y forma de calificar los delitos cometidos violando algún artículo del código de tránsito.

Es por ello que la dirección del equipo de desarrollo y la máxima dirección del Tribunal Supremo Popular (TSP), decidieron realizar un taller conjunto entre la dirección del equipo de desarrollo y los especialistas funcionales que participaron en los concentrados de pruebas y capacitación. El objetivo de este taller era centrar todas las solicitudes de cambio y proceder a presentarlas a la presidencia de la Sala de lo Penal del TSP para su posterior aprobación por parte de la UCI y del TSP. El resultado de este trabajo detallado arrojó un total de 56 solicitudes de cambio, afectando 30 casos de uso de diferentes complejidades y la adición de 4 casos de uso nuevos para desarrollar.

En estos momentos se encuentran listadas todas las solicitudes de cambio que se necesitan acometer; especificando y modificando requisitos que se diseñarán e implementarán para el cumplimiento de las expectativas del cliente.

Teniendo en cuenta estos elementos, surge como **problema a resolver:** ¿Cuál es la especificación de los artefactos necesarios que informaticen los requisitos identificados en el paquete de solicitudes de cambio del módulo Ordinario Penal del SITPC, para satisfacer las necesidades de modificación demandadas por los especialistas funcionales del mismo?

Tomando como **objeto de estudio:** El proceso de desarrollo de software de sistemas informáticos para la gestión jurídica.

Con el fin de solucionar el problema planteado se define como **objetivo general:** Desarrollar el paquete de solicitudes de cambio del módulo Ordinario Penal perteneciente al SITPC, para satisfacer las necesidades demandadas por los especialistas funcionales de este software.

Enmarcándose en el **campo de acción**: Desarrollo del módulo Ordinario Penal perteneciente al SITPC.

Teniendo como referencia el problema a resolver y el objetivo general se plantea la siguiente **idea a defender**: Con el desarrollo del paquete de solicitudes de cambio para el procedimiento Ordinario Penal se obtiene la especificación de los artefactos necesarios de forma tal que se garantice satisfacer las necesidades de modificación demandadas por los especialistas funcionales del mismo.

Posibles resultados en artefactos a entregar:

Modelo de diseño, Modelo de datos, Modelo de implementación y Diseño de casos de prueba.

Para dar cumplimiento al objetivo propuesto se han derivado un conjunto de **Objetivos específicos** orientados a proveer los elementos necesarios para la implementación de la solución, los cuales se listan a continuación:

- Elaborar el marco teórico de la investigación para sustentar los principales referentes teóricos en los que se enmarca el desarrollo de la propuesta de solución.
- Obtener el Modelo de sistema, a partir de las solicitudes de cambios identificadas, para tener un acercamiento de las necesidades a desarrollar.
- Obtener el Modelo de implementación para lograr la especificación técnica o algoritmos como un programa que respondan a las funcionalidades definidas en el Modelo del sistema.
- Validar los resultados obtenidos a través de pruebas de software para evaluar la calidad de la propuesta de solución.

En correspondencia con los objetivos previamente expuestos, se tienen como **tareas de investigación:**

1. Estudio del proceso de desarrollo de software y su metodología.
2. Estudio de la plataforma de desarrollo y de las herramientas utilizadas para diseñar, implementar y probar el sistema.
3. Realización de los Diagramas de clases del diseño.

4. Realización de Casos de uso del diseño (Diagramas de comportamiento).
5. Realización del Diagrama de clases (implementación).
6. Estudio de los Paradigmas de programación.
7. Estudio y selección de los Patrones de diseño más factibles para esta propuesta de solución.
8. Realización de los Diagramas de secuencia.
9. Realización del Modelo de datos.
10. Realización del Diagrama de despliegue.
11. Implementación de la personalización.
12. Diseño de casos de prueba.

Para el desarrollo de las tareas de investigación serán empleados los métodos de Investigación y Desarrollo (I + D) de la Informática: Lógicos y Empíricos, que constituyen los procedimientos que se utilizan para estudiar la realidad, la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia, sus relaciones y llegar al conocimiento científico; por lo que son el instrumento para alcanzar los fines de la investigación. (Hernández Sampieri, y otros, 2010)

Métodos lógicos:

Se basan en la utilización del pensamiento en sus distintas funciones de deducción, análisis y síntesis para llegar al conocimiento.

- **Métodos lógicos formales:** Responden a las formas básicas de razonamiento. Dentro de estos métodos se utilizó:
  1. El **método deductivo:** Facilitó el planteamiento de la idea a defender.
- **Métodos lógicos de soporte:** Contribuyen a la obtención de conocimiento. A continuación, se describen los métodos lógicos de soporte que se utilizan en la presente investigación:
  1. El **método analítico** posibilitará la realización de un estudio teórico de la investigación, y la comprensión de las solicitudes de cambio del proceso ordinario del subsistema Penal del SITPC.
  2. El **método sintético** facilitará la separación lógica de los elementos fundamentales encontrados en el estudio realizado, en aras de relacionarlos con el objeto de estudio.
  3. El uso de la **modelación** es sumamente importante puesto que permitirá construir modelos (diagramas) que explicitarán a través de símbolos y relaciones entre estos, el flujo de actividades y las propiedades del proceso ordinario del subsistema Penal del SITPC.

Métodos empíricos:

Se aproximan al conocimiento del objeto mediante su conocimiento directo y el uso de la experiencia.

- **Métodos cuantitativos**

Tratan de establecer relaciones causales que supongan una explicación del fenómeno. En el caso de la presente investigación se utilizó el método **simulación** clasificado también dentro de los métodos experimentales controlados, permitirá la realización de casos de prueba con datos artificiales, para comprobar la correcta ejecución de los flujos básicos y alternos correspondientes a cada funcionalidad que deba realizar el producto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se abordan los conceptos fundamentales para sustentar los principales referentes teóricos en los que se enmarca el desarrollo de la propuesta de solución. Así como los tipos de soporte que brinda la UCI a los sistemas informáticos con el fin de tener una base para la actualización del sistema a raíz de las solicitudes de cambio. También se fundamenta la selección de la metodología, los paradigmas de programación, patrones de diseño, las herramientas y lenguajes de programación por parte del equipo de arquitectura de proyecto, que son usados en el desarrollo de las solicitudes de cambio aplicadas al SITPC.

1.1 Procedimiento Ordinario Penal

La materia Penal en los TPC se encarga de tramitar todos los asuntos que se originen por la comisión de cualquier tipo de delito. La justicia penal se imparte en nombre del pueblo de Cuba. No puede imponerse sanción o medida de seguridad sino de conformidad con las normas de procedimiento establecidas en la ley y en virtud de resolución dictada por el tribunal competente. Se presume inocente a todo acusado mientras no se dicte fallo condenatorio contra él. Todo delito debe ser probado independientemente del testimonio del acusado, de su cónyuge y de sus familiares hasta el cuarto grado de consanguinidad o segundo de afinidad. En consecuencia, la sola declaración de las personas expresadas no dispensará de la obligación de practicar las pruebas necesarias para la comprobación de los hechos. (García, 2008)

El procedimiento Ordinario Penal comprende los delitos sancionados por multa de 300 a 1 000 cuotas, privación de libertad con un límite de tiempo de 1 a 3 años o incluso ambas. Además de las sanciones subsidiarias de la privación de libertad, las cuales son trabajo correccional con internamiento, trabajo correccional sin internamiento, limitación de libertad y remisión condicional. Este procedimiento está constituido por varios procesos que se encargan de tramitar las causas que se presentan en el tribunal, entre estos procesos se encuentra la presentación del Expediente de Fase Preparatoria (EFP), la apertura a juicio oral, la celebración del juicio oral, la firmeza de la sentencia, entre otros.

1.2 Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC)

Con el objetivo de informatizar la actividad jurisdiccional en el marco de los TPC, surge el SITPC, trayendo consigo impactos positivos para el trabajo en los tribunales y el sector público, tales como: informatización integral de la actividad judicial, control, cumplimiento y alerta del vencimiento de los términos, reportes estadísticos en tiempo real, ayuda a la toma de decisiones de jueces y dirigentes, celeridad en la tramitación de los procesos, estandarización y uniformidad de los actos procesales,

seguridad, restringiendo la ejecución de acciones según niveles de acceso definidos, almacenamiento seguro y organizado de la información, con facilidad de acceso o consulta y el ahorro de recursos materiales y esfuerzo humano. (González Ochoa, y otros, 2018)

La implementación de este sistema se divide en los subsistemas Administración y Gobierno, Común, Penal, Económico, Administrativo, Laboral y Civil. Además, cuenta con varios procedimientos mediante los cuales se hacen cumplir las pautas establecidas dentro del Código Penal cubano; algunos de estos procedimientos son: sumario, abreviado, ordinario, entre otros.

Luego del despliegue de este sistema informático, se dio inicio a la fase de soporte, en aras de garantizar el buen funcionamiento del mismo. A continuación, se explican los tipos de soporte que se brindan a nivel internacional, y principalmente en el caso de la UCI, con el objetivo de identificar el proceder de la actualización de las solicitudes de cambio en SITPC.

1.3 Soporte para sistemas informáticos

Este servicio se brinda a empresas o personas que buscan soluciones a las averías sean físicas (hardware) o lógicas (software) de su computadora, y lo brinda un personal especializado en informática o mantenimiento de las mismas denominados técnicos. El mismo es ofrecido por algunas empresas que dan garantía con respecto al servicio brindado o personas naturales capacitadas, con el motivo de obtener un ingreso económico. (Znet, 2016)

Hoy día existen varios tipos de soporte en el mundo:

- Soporte técnico vía telefónica
- Soporte técnico vía correo electrónico
- Soporte técnico vía chat
- Soporte técnico vía asistencia técnica personal
- Soporte técnico vía asistencia remota

En el Centro de Soporte de la UCI específicamente se brindan los siguientes servicios:

- Soporte estándar: recoge la actividad básica de soporte técnico. En él podrá tener contacto directo con los especialistas mediante la web o por vía telefónica, en un horario limitado de 8 horas, que comprende la jornada laboral. Este tipo de soporte está orientado a entidades que cuenten con una infraestructura tecnológica estable, y colectivo altamente preparado en las funcionalidades del sistema adquirido.
- Soporte operacional: recoge la actividad intermedia de soporte técnico. En el podrá acceder a las oportunidades descritas en el Soporte estándar en un horario de 24 horas. Además, incluye el mantenimiento y actualización del sistema informático adquirido. Contiene todos los servicios del Soporte estándar y otros.

- Soporte profesional: recoge la actividad avanzada de soporte técnico. Mediante este servicio se pueden obtener las oportunidades que brindan el Soporte estándar y el operacional. Además, incluye Soporte in-situ y Consultorías. Este tipo de servicio está orientado a entidades que precisen un seguimiento exhaustivo de la actividad de soporte y la atención especializada al sistema informático adquirido. (Centro de Soporte, 2018)

Para el caso de las solicitudes de cambio de SITPC este centro brinda el Soporte profesional que incluye todos los servicios de los Soportes estándar y operacional. Luego de un proceso realizado entre las partes involucradas (TSP y CEGEL) se determinó que era necesario acometer el desarrollo de esas solicitudes de cambio reportadas en las incidencias del soporte para el correcto funcionamiento del software, para lo cual se decidió utilizar las mismas tecnologías bajo las que se desarrolló el SITPC en su primera versión.

1.4 Metodología de desarrollo de software

Podría llegar a ser complicado establecer una forma estándar para definir el concepto de metodología de desarrollo de software, sin embargo, existen algunas definiciones importantes utilizadas en la actualidad. Por ejemplo, la revista electrónica International Journal of Computer Applications define una metodología de desarrollo como “un proceso mediante el cual un proyecto de software es completado o desarrollado a través de procesos o etapas bien definidas”.

También el concepto brindado por Pressman cuando dice: “Se puede definir metodología de desarrollo de software como un conjunto de procedimientos, técnicas, herramientas y un soporte documental para el desarrollo de productos de software”. (Pressman, 2010)

Como metodología de desarrollo de software se utilizó el Proceso Unificado Ágil (AUP por las siglas en inglés de Agile Unified Process) variación para la UCI por parte de la dirección del equipo de desarrollo, basándose en los lineamientos de la infraestructura productiva de la universidad. Esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. (Sánchez, 2014)

1.4.1 Metodología AUP variación para la UCI

A raíz de la utilización de varias metodologías en los proyectos productivos de los centros de desarrollos de la universidad, surge la metodología AUP variación para la UCI, con el objetivo de estandarizar el proceso que guiará el desarrollo productivo para la misma.

Esta metodología cuenta con las fases de Inicio, Ejecución y Cierre. Además, define las disciplinas para la fase de Ejecución: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. Para la disciplina de Requisitos define 4 escenarios, de los cuales, la presente investigación se guiará por el Escenario No 1, el cual

plantea que los proyectos que modelen el negocio con CUN[1] solo pueden modelar el sistema con CUS[2].

Figura 1: Escenario 1 para la disciplina de Requisitos

Además de seleccionar la metodología de desarrollo para crear un software se debe tener una arquitectura sólida que guíe la implementación de todos los componentes del mismo y se adapte a los requerimientos del sistema.

1.5 Arquitectura de software

Una arquitectura de software define la forma de trabajar en un sistema, implica definir una solución estructurada que satisfaga todos los requisitos técnicos y operacionales y, a la vez, optimizar los atributos comunes de calidad como rendimiento, seguridad y capacidad de administración. Además, implica una serie de decisiones basadas en una amplia gama de factores, y cada una de esas decisiones puede tener un considerable impacto sobre la calidad, rendimiento, mantenimiento y éxito general de ese software. (Microsoft, 2017)

1.5.1 Estilos arquitectónicos

Los Estilos arquitectónicos son un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer en sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo. (POSSO, 2010)
Los estilos se clasifican de la siguiente forma:

Estilo de Flujo de datos

- Tuberías y filtros

Estilos Centrados en datos

- Arquitecturas de pizarra o repositorio

Estilos de Llamada y Retorno

- Modelo-Vista-Controlador (MVC)
- Arquitecturas en capas
- Arquitecturas orientadas a objetos
- Arquitecturas basadas en componentes

Estilos de Código móvil

[1] Caso de Uso del Negocio

[2] Caso de uso del Sistema

- Arquitectura de máquinas virtuales

Estilos heterogéneos

- Sistemas de control de procesos
- Arquitecturas basadas en atributos

Estilos Peer-to-Peer

- Arquitecturas basadas en eventos
- Arquitecturas orientadas a servicios
- Arquitecturas basadas en recursos

Teniendo como referencia la selección del marco de trabajo se asume el estilo arquitectónico Modelo Vista Controlador (MVC), ya que es el que implementa dicho marco de trabajo. Este estilo de arquitectura de software separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste.
- El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno. (Alicante, 2018)

1.5.2 Marco de trabajo (framework)

Un marco de trabajo es una estructura de soporte definida, donde un proyecto de software puede ser organizado y desarrollado. Simplifican el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, facilitan la programación de aplicaciones, encapsulan operaciones complejas en instrucciones sencillas y proporcionan estructura al código fuente, forzando al desarrollador a crear código legible y más fácil de mantener. Además, permiten facilitar el desarrollo del software, suelen incluir soporte de programas, bibliotecas, y software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas. (Hernando Recaman Chaux, 2012)

A continuación, se describe el marco de trabajo que se tienen en cuenta en el desarrollo del SITPC.

Symfony2

Es una versión de Symfony ideada para explotar todas las características de Hypertext Preprocessor o Procesador de Hipertexto (PHP por sus siglas en inglés). Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un proyecto. Symfony2 también es el framework que más ideas incorpora del resto de los framework, incluso de aquellos que no están programados con PHP. (Eguiluz, 2012)

Ventajas de utilizar Symfony2 según (4rsoluciones, 2013):

- Es fácil de instalar y configurar en cualquier plataforma.
- Libera a los desarrolladores de la tarea de crear funcionalidades menores y, en ocasiones, aburridas de implementar.
- Las aplicaciones desarrolladas con Symfony son compatibles con la mayoría de las plataformas, bibliotecas e infraestructuras que existen. Se adaptan a entornos de negocio en cambio permanente, requiriendo menos esfuerzo para su mantenimiento.
- Se integra fácilmente con las Application Programming Interface conocida también como Interfaz de Programación de Aplicaciones (API) de plataformas como Twitter, Yahoo y Google, entre otras.
- Es fácil de extender. Como se trata de un proyecto de código abierto, se le puede agregar nuevas funciones desarrolladas por programadores externos.
- Promueve el uso de buenas prácticas de programación y genera código fácilmente comprensible por el desarrollador.
- Cuenta con una comunidad de programadores con el fin de contar con la seguridad de que cualquier posible defecto será corregido en versiones posteriores.

A raíz del estudio y la opción de actualizar la versión de Symfony, se tiene como resultado que existen problemas de incompatibilidad de paquetes con los subsistemas de terceros que usa el sistema, por ejemplo: el generador dinámico de reportes y el componente de seguridad, además no cuenta con servicio de soporte.

Motor de plantillas Twig

Para la generación de las vistas del SITPC se utiliza el motor de plantillas Twig ya que es el motor de plantillas recomendado cuando se utiliza el framework Symfony2.

Twig, el motor de plantillas flexible, rápido y seguro para PHP. Es una herramienta de código abierto incluida por defecto en Symfony2. (SensioLabsWorld, 2018)

Las características claves son:

- Rápido: Twig compila las plantillas en un código PHP simple y optimizado. La sobrecarga en comparación con el código PHP normal se redujo al mínimo.

- Seguro: Twig tiene un modo de espacio aislado para evaluar el código de la plantilla que no es de confianza. Esto permite que Twig se use como un lenguaje de plantilla para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- Flexible: Twig es alimentado por un lector flexible y analizador sintáctico. Esto le permite al desarrollador definir sus propias etiquetas y filtros personalizados.

1.5.3 Mapeador Objeto-Relacional (ORM)

Un ORM permite convertir los datos de tus objetos en un formato correcto para poder guardar la información en una base de datos (mapeo) creándose una base de datos virtual donde los datos que se encuentran en nuestra aplicación, quedan vinculados a la base de datos (persistencia). (Hernández, 2018)

Un ORM es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador.

Entre las ventajas que ofrecen los ORM se encuentran:

- Facilidad y velocidad de uso.
- Abstracción de la base de datos usada.
- Seguridad de la capa de acceso a datos contra ataques. (Programción.NET, 2017)

El ORM que se utiliza en el desarrollo del SITPC es Doctrine 2, el único incluido en el framework Symfony2.

Doctrine 2

Doctrine 2 es un ORM que proporciona persistencia transparente para objetos PHP. Utiliza el patrón Data Mapper[3] en su núcleo, con el objetivo de lograr una separación completa de su lógica de negocios de la persistencia en un sistema de administración de bases de datos relacionales.

El beneficio de Doctrine para el programador es la capacidad de enfocarse en la lógica de negocios orientada a objetos y preocuparse por la persistencia solo como un problema secundario. (Doctrine-project.org, 2018)

1.5.4 Bootstrap

Bootstrap es un framework desarrollado por Twitter que simplifica el proceso de creación de diseños web combinando Hojas de Estilo en Cascada (CSS) y JavaScript. Su mayor ventaja es que se pueden crear interfaces que se adapten a los distintos navegadores apoyándose en un framework

[3] Data Maper: es una capa de acceso a datos que realiza la transferencia bidireccional de datos entre un almacén de datos persistente (a menudo una base de datos relacional) y una representación de datos en memoria (la capa de dominio).

potente con numerosos componentes webs que ahorrarán mucho esfuerzo y tiempo. Es un proyecto de código abierto con licencia Apache v2.0. (Rodríguez, 2012)

Características principales

Bootstrap ofrece una serie de plantillas CSS y ficheros JavaScript que permiten integrar el framework de forma sencilla y potente en proyectos webs.

- Permite crear interfaces que se adapten a los diferentes navegadores, a distintas escalas y resoluciones.
- Se integra perfectamente con las principales librerías JavaScript, por ejemplo, jQuery.
- Ofrece un diseño sólido y estándares como CSS3/ Lenguaje de Marcas de Hipertexto (HTML5).
- Es un framework ligero que se integra de forma limpia en proyectos web.

Por lo expuesto anteriormente y teniendo en cuenta que SITPC es una aplicación web, el equipo de arquitectura seleccionó este framework para el trabajo en la vista.

1.6 Paradigmas de programación

Un Paradigma de programación es una propuesta tecnológica que es adoptada por una comunidad de programadores, cuyo núcleo central es incuestionable en cuanto a que unívocamente trata de resolver uno o varios problemas claramente delimitados. La resolución de estos problemas debe suponer consecuentemente un avance significativo en al menos un parámetro que afecte a la ingeniería de software. Tiene una estrecha relación con la formalización de determinados lenguajes en su momento de definición. Un paradigma de programación está delimitado en el tiempo en cuanto a aceptación y uso ya que nuevos paradigmas aportan nuevas o mejores soluciones que la sustituyen parcial o totalmente. (Barzanallana, 2011)

Actualmente existe un elevado número de paradigmas, aunque los esenciales se podrían dividir en dos grupos:

Programación Declarativa: Le dice al ordenador qué hacer, pero no cómo hacerlo, o sea, se describe el problema que se quiere solucionar, pero no las instrucciones necesarias para solucionarlo. La solución se logrará mediante mecanismos internos de inferencia de información a partir de la descripción realizada. Los paradigmas Funcional y Lógico corresponden a este grupo.

Programación Imperativa: Describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea. A este grupo corresponden los paradigmas: Orientado a objetos, Visual, Orientado a eventos y Orientado a aspectos.

De acuerdo con las características necesarias para el desarrollo del SITPC se decidió por parte del equipo de arquitectura del proyecto la utilización del paradigma de Programación Orientada a Objetos (POO).

1.6.1 Programación Orientada a Objetos

El paradigma POO se le considera como una técnica, que permite la reutilización y extensibilidad de líneas código simulando un entorno real. Se puede definir reutilización y extensibilidad de la siguiente manera:

- Reutilización: Es la capacidad de un producto software de ser utilizado en la construcción de diferentes aplicaciones, de modo que permite no reinventar soluciones para problemas ya resueltos. Entonces, se escribe menos software y se puede dedicar más tiempo a mejorar otros factores.
- Extensibilidad: Es la facilidad de adaptación de los productos software a los cambios en la especificación que puede dar un usuario que utilizará el sistema. Los cambios son frecuentes, puesto que nuestro entorno es muy cambiante. Los principios esenciales para facilitar la extensibilidad son: Simplicidad de la arquitectura del software y la descentralización, pues crea módulos autónomos. (Urquiza, 2017)

1.7 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. A continuación, se enuncian los beneficios de dichas herramientas:

- Permiten la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta consiguen agilizar el trabajo.
- Facilitan la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplifican el mantenimiento de los programas.
- Mejoran y estandarizar la documentación.
- Aumentan la portabilidad de las aplicaciones.
- Facilitan la reutilización de componentes software.
- Permiten un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos. (Valderrama, 2016)

Algunos ejemplos de herramientas CASE son:

- Enterprise Architect.
- Rational Rose Enterprise Edition 2003.

- ERwin PLATINUM.
- EasyCASE.
- Visual Paradigm for UML[4].

Para la realización del SITPC, se decide utilizar la herramienta Visual Paradigm for UML en su versión 8.0, debido a la fácil integración de esta herramienta con el lenguaje PHP y a la experiencia del equipo de desarrollo en su empleo, lo que posibilita reducir el tiempo en el proceso de desarrollo del software ya que no hay necesidad de capacitar a los desarrolladores.

1.7.1 Visual Paradigm for UML

Visual Paradigm es una herramienta de diseño y administración poderosa, así como multiplataforma. Además, ofrece a los desarrolladores de software una plataforma de desarrollo innovadora para crear aplicaciones de calidad de manera más rápida, mejor y más económica. Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los Entorno de Desarrollo Integrado (IDE) líderes que sobresalen todo su proceso de desarrollo.

Las ventajas que proporciona Visual Paradimg for UML son:

- Dibujo. Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica. Controla que el modelado con UML sea correcto.
- Coherencia entre diagramas. Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Integración con otras aplicaciones. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Trabajo multiusuario. Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- Reutilización. Facilita la reutilización, ya que es una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- Generación de código. Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- Generación de informes. Permite generar diversos informes a partir de la información introducida en la herramienta. (VisualParadigm, 2014)

1.8 Lenguajes de programación

[4] Lenguaje Unificado de Modelado

Un lenguaje de programación debe resultar correcto, es decir, determinar qué debe hacer antes de desarrollarlo y compararlo, una vez acabado, con lo que realmente hace. Debe ser lo más claro, conciso y legible posible, con una estructura sencilla y entendible. Eficaz, que sea capaz de gestionar de la mejor manera posible los recursos que utiliza. Portable, con capacidad para ser ejecutado en una plataforma diferente a aquella en la que se elaboró inicialmente ya sea hardware o software. (SoftDoit, 2018)

1.8.1 Procesador de Hipertexto 5.3 (PHP)

El lenguaje de programación PHP, del inglés Hypertext Preprocessor, fue desarrollado puntualmente para diseñar páginas web dinámicas programando scripts del lado del servidor. El lenguaje PHP siempre va incrustado dentro del HTML y generalmente se le relaciona con el uso de servidores Linux. El lenguaje PHP presenta cuatro grandes características:

1. Velocidad: PHP no solo es rápido al ser ejecutado, sino que no genera retrasos en la máquina, por esto no requiere grandes recursos del sistema. PHP se integra muy bien junto a otras aplicaciones, especialmente bajo ambientes Unix.
2. Estabilidad: PHP utiliza su propio sistema de administración de recursos y posee un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
3. Seguridad: PHP maneja distintos niveles de seguridad, estos pueden ser configurados en los archivos de configuración .ini.
4. Simplicidad: Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente. Además, PHP dispone de una amplia gama de librerías, y permite la posibilidad de agregarle extensiones. Esto le permite su aplicación en múltiples áreas, tales como encriptado, gráficos, Lenguaje de Marcado Extensible (XML) y otras. (Latinoamericana, 2018)

1.8.2 JavaScript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js o Apache CouchDB. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa.

Ventajas y desventajas:

1. El lenguaje de scripting es seguro y fiable porque está en claro y hay que interpretarlo, por lo que puede ser filtrado; para el mismo Javascript, la seguridad es casi total y sólo en su primera versión el CIAC (Computer Incident Advisory Committee) señaló problemas de leve entidad, entre ellos la lectura de la caché y de los sitios visitados, de la dirección e-mail y de

los archivos presentes en el disco. Sin embargo, estos fallos se corrigieron ya en las versiones de Netscape sucesivas a la 2.0.

2. Los scripts tienen capacidades limitadas, por razones de seguridad, por lo cual no es posible hacer todo con Javascript, sino que es necesario usarlo conjuntamente con otros lenguajes evolucionados, posiblemente más seguros, como Java. Dicha limitación es aún más evidente si se quiere operar en el hardware del ordenador, como, por ejemplo, la fijación en automático de la resolución vídeo o la impresión de un documento.
3. Un problema importante es que el código es visible y puede ser leído por cualquiera, incluso si está protegido con las leyes del copyright. Esto, que es una ventaja, representa el precio que tiene que pagar quien quiere utilizar el web: la cuestión de los derechos de autor ha asistido a una verdadera revolución con la llegada de Internet (el ejemplo más representativo, el MP3). La tutela que proporcionan las leyes actuales resulta débil e inadecuada, por lo que la única solución es tomarse las cosas con filosofía.
4. El código Javascript se ejecuta en el cliente por lo que el servidor no es solicitado más de lo debido; un script ejecutado en el servidor, sin embargo, sometería a éste a dura prueba y los servidores de capacidades más limitadas podrían resentir de una continua solicitud por un mayor número de usuarios.
5. El código del script debe descargarse completamente antes de poderse ejecutar y ésta es la otra cara de la moneda que se ha mencionado anteriormente: si los datos que un script utiliza son muchos (por ejemplo, una recopilación de citas que se mostrará de manera casual), el tiempo que tardará en descargarse será muy largo, mientras que la interrogación de la misma base de datos en el servidor sería más rápida. (Next_U, 2018)

JQuery 1.8

JQuery no es un lenguaje, sino una serie de funciones y métodos de Javascript. Por tanto, Javascript es el lenguaje y jQuery es una librería que se puede usar opcionalmente cuando se programa en Javascript. A veces se puede hablar de jQuery como framework o incluso como un API de funciones, útiles en la mayoría de proyectos web. Facilita mucho el desarrollo de aplicaciones enriquecidas del lado del cliente, en Javascript, compatibles con todos los navegadores. (Jqueryui.com, 2018)

1.9 Entorno de desarrollo integrado

Entorno de desarrollo integrado, llamado IDE (siglas en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Ha sido empaquetado como un programa de aplicación, es decir, que consiste

en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). (AprendiendoArduino, 2014)

Existen varios IDE de múltiples lenguajes tales como Eclipse, Oracle JDeveloper, Codenvy, Microsoft Visual Studio, NetBeans, etc. Para el desarrollo del SITPC se define por parte del equipo de desarrollo utilizar NetBeans en su versión 8.1 como IDE, para ello se tuvieron en cuenta las características que se mencionan en el próximo epígrafe.

1.9.1 Netbeans 8.1

Entorno de desarrollo de código abierto integrado con PHP que posee un sistema para hacer un reconocimiento y carga de clases, métodos y objetos, permite la creación de aplicaciones web, propone un esqueleto para organizar el código fuente, es multiplataforma y el editor conjuntamente integra lenguajes como HTML, JavaScript, CSS, y PHP. (Oracle, 2018)

NetBeans IDE proporciona soporte para el cliente de control de versiones de Git este permite realizar tareas de control de versiones directamente desde su proyecto dentro del IDE. (NetBeans.org, 2018)

1.9.2 Control de versiones Git

Git es un sistema de fuente abierta y gratuita, diseñado para manejar los archivos lógicos, desde proyectos pequeños hasta proyectos muy grandes, con velocidad y eficiencia. Cada clon de Git es un repositorio completo con historial completo y capacidad completa de seguimiento de revisiones, que no depende del acceso a la red o de un servidor central. La ramificación y la fusión son rápidas y fáciles de hacer. (Dudler, 2017)

1.10 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información del modo más eficiente posible. (Empresariales, 2016) En el caso del SITPC se decide utilizar por parte del equipo de desarrollo PostgreSQL 9.5.

1.10.1 PostgreSQL 9.2

PostgreSQL es un servidor de base de datos avanzado con un largo historial de desarrollo. Está disponible para una amplia variedad de plataformas y es usado desde el más pequeño de los sistemas integrados hasta enormes sistemas de terabytes. PostgreSQL se ha ganado una excelente reputación debido a sus características innovadoras, integridad, seguridad y fiabilidad.

Además, es bien conocido como una base de datos que puede mantenerse en línea por periodos prolongados y requiere de poco a ningún mantenimiento en muchos casos. (2ndQuadrant+PostgreSQL, 2018)

Características:

- Excelente cumplimiento del estándar SQL (por sus siglas en inglés Structured Query Language), siguiendo el último SQL: 2011.
- Arquitectura cliente-servidor con un amplio rango de drivers y clientes.
- Diseño de alta concurrencia donde los lectores y escritores no se bloquean.
- Altamente configurable y extensible para muchos tipos de aplicación.
- Excelente escalabilidad y rendimiento con características de ajustes extensas.
- Optimizador de consultas sofisticado, adecuado para inteligencia de negocios.
- Soporta totalmente el acceso y procedimientos de base de datos en Java, Python, perl, PHP y muchos más.
- Altamente confiable con características extensivas para durabilidad y alta disponibilidad.
- Tipos de datos avanzados como Información Geográfica, búsqueda completa de texto y más.
- Características de internacionalización, codificaciones multibyte y colaciones.
- Excelente soporte.

1.11 Servidor web

La definición más sencilla de servidor web, que es un programa especialmente diseñado para transferir datos de hipertexto, es decir, páginas web con todos sus elementos (textos, widgets y banners). Estos servidores web utilizan el protocolo HTTP (Hypertext Transfer Protocol).

Los servidores web están alojados en un ordenador que cuenta con conexión a internet. El servidor web, se encuentra a la espera de que algún cliente le haga alguna petición, como, por ejemplo, acceder a una página web y responde a la petición, enviando código HTML mediante una transferencia de datos en red. (Ibrugor, 2014)

Algunos ejemplos de servidores web son Nginx, Internet Information Services (IIS), Cherokee, Tomcat y Apache. En el desarrollo del SITPC se decidió utilizar Apache en su versión 2.2.

Apache 2.2

El servidor Apache HTTP, también llamado Apache, es un servidor web HTTP de código abierto para la creación de páginas y servicios web. Es un servidor multiplataforma, gratuito, muy robusto y que destaca por su seguridad y rendimiento.

Ventajas de Apache:

- Instalación/Configuración. Software de código abierto.
- Coste. El servidor web Apache es completamente gratuito.
- Funcional y Soporte. Alta aceptación en la red y muy popular, esto hace que muchos programadores de todo el mundo contribuyen constantemente con mejoras, que están

disponibles para cualquier persona que use el servidor web y que Apache se actualice constantemente.

- Multi-plataforma. Se puede instalar en muchos sistemas operativos, es compatible con Windows, Linux y MacOS.
- Rendimiento. Capacidad de manejar más de un millón de visitas/día.
- Soporte de seguridad SSL (Secure Sockets Layer) y TLS (Transport Layer Security). (Ibrugor, 2014)

Se definió Apache como software para el servidor de aplicaciones en su versión 2.2, por su flexibilidad, rapidez, porque es gratuito, multiplataforma e interpreta varios lenguajes como PHP.

1.12 Conclusiones parciales

El desarrollo del marco teórico referencial relacionado con el proceso de desarrollo de software de sistemas informáticos para la gestión jurídica, facilitó una mejor comprensión del objeto de estudio de la presente investigación. Por otra parte, el análisis de las características de la metodología AUP variación para la UCI, las distintas herramientas a utilizar, fundamenta la correspondencia de su selección por parte del equipo de arquitectura para el desarrollo de la propuesta de solución.

Capítulo 2: Propuesta de solución

Introducción

En el presente capítulo se realiza un análisis del proceso de las incidencias y una descripción de los cambios en el sistema como consecuencia de las solicitudes de cambio. Además, se hace un análisis de la arquitectura del sistema y de los patrones asociados al diseño de la aplicación. También se presentan los diagramas de clases, el modelo de datos y los de comportamiento, particularmente los diagramas de secuencia y despliegue. Por último, se describen los estándares de codificación que deben ser utilizados para la implementación de las solicitudes.

2.1 Análisis del proceso de solicitud de cambio

Una Solitud de Cambio (SC) es un producto de trabajo enviado formalmente que se utiliza para rastrear todas las solicitudes del interesado, incluidas funciones nuevas, solicitudes de mejora, defectos, requisitos cambiados e información relacionada con el estado a través del ciclo vital del proyecto. El historial de cambios se mantendrá junto con la solicitud de cambio, incluidos todos los cambios de estado, las fechas y los motivos de dicho cambio. Esta información estará disponible para las revisiones repetidas y para el cierre final. (RUP, 2006)

La gestión de las SC es un proceso para garantizar que se utilicen los procedimientos y los métodos estandarizados para el manejo eficaz y rápido de todos los cambios y para minimizar el impacto de los incidentes relacionados con los cambios.

En el caso del Proyecto para la Informatización de la Gestión de los Tribunales Populares Cubanos, la solicitud de cambio llega al equipo de proyecto a través de incidencias reportadas al Centro de Soporte, por encuentros de capacitación o aceptación con los jueces de la Sala de lo Penal del TSP. Estas incidencias son analizadas en el proyecto para determinar si son SC o No Conformidades (NC). Las NC constituyen errores del sistema y se procede de inmediato a darle solución, luego se hacen pruebas y se actualiza el sistema con los cambios realizados.

Por su parte, las SC se consideran como elementos que no se tuvieron en cuenta por parte de los especialistas funcionales inicialmente para el desarrollo del sistema. Estas son analizadas por el equipo de proyecto de conjunto con los especialistas funcionales del software. Cuando quedan finalmente definidas son nuevamente analizadas por parte del proyecto teniendo en cuenta aspectos importantes, tales como: el módulo al que pertenece, el elemento al que hace referencia la solicitud y el nombre de la línea base a modificar. Luego de implementarse estas SC se hacen pruebas al sistema y se actualiza.

A continuación, se muestra la descripción del proceso sobre análisis y solución de las incidencias.

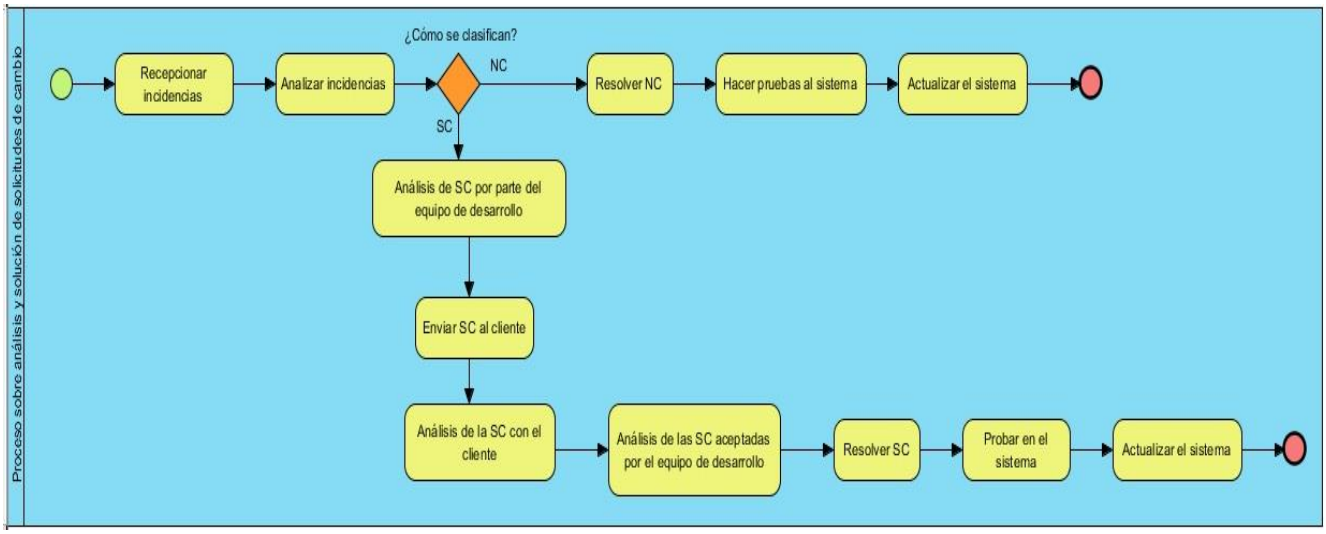

Figura 2: Descripción del proceso sobre análisis y solución de las incidencias

La presente investigación se basa solamente en el proceso que se refiere a la solución de las SC. A continuación, se describen los cambios realizados basados en los casos de uso existentes y la concepción de los casos de uso nuevos, así como su relación con los Requisitos Funcionales (RF).

2.2 Descripción de los cambios a realizar

Tabla 1: Descripción de los cambios en los casos de uso existentes

Casos de uso existentes	Total de RF afectados	Descripción del cambio
Registrar EFP	1	Agregar un campo al registrar el EFP para captar la cantidad de folios útiles que tiene y luego mostrar ese número en el documento de entrega de EFP al abogado.
Gestionar delito	4	Cambiar en el listado de delitos donde dice Lavado de dinero y poner Lavado de activos. Agregar un campo de texto al calificar el delito para que inserten los artículos de otras leyes, como el código de vialidad y tránsito, que puedan estar relacionados con los delitos calificados. Consignar en su forma más simple el nombre de los delitos y no contradecir lo que plantea la Instrucción 208.

		Nomenclar de forma separada los artículos de los delitos del tránsito para que puedan ser calificados en concurso ideal y mostrarlos por separado en los documentos
Registrar sanción principal	1	Incorporar para el caso del fiscal, la posibilidad de poder seleccionar el fallo absolutorio cuando es en legítima defensa, las opciones serían: • El error • El estado de necesidad • La legítima defensa • El cumplimiento de un deber o el ejercicio de derecho, profesión, cargo u oficio • Miedo insuperable En caso del abogado: serían las mismas que para el fiscal, además: • Por no constituir delito • Por insuficiencia de pruebas Habilitar un campo que diga • Absolución
Gestionar sanción accesoria	1	Insertar en la sentencia 2 campos: • en el caso de prohibición de frecuentar por X tiempo un lugar, especificar el nombre del mismo. • si es la prohibición del ejercicio de un cargo o puesto de trabajo, especificar cuál es ese puesto.
Registrar solicitud del fiscal	1	Hacer que el auto de extinción de la responsabilidad penal sea editable y que el sistema capte la fecha de fallecimiento.
Gestionar acusado	1	Hacer que el auto de extinción de la responsabilidad penal sea editable y que el sistema capte la fecha de fallecimiento.

Modificar medidas cautelares	1	Agregar la disposición sobre las medidas cautelares y bienes.
Registrar disposición sobre los bienes	1	Agregar una opción que sea Unir al expediente en la disposición sobre los bienes.
Registrar disposición sobre regulaciones migratorias	1	Permitir que al disponer sobre la regulación migratoria el juez disponga si es solo en cuanto a la sanción principal o comprende la responsabilidad civil.
Registrar asistencia a juicio oral	1	Cambiar el del artículo 332 del Código Penal por el 210.
Crear acta de juicio oral	1	Modificar el acta y hacer las diferentes pantallas antes de llegar al documento donde se capte todo.
Crear acta de votación	1	Agregar una opción para sancionar en parte. Que permite sancionar por un delito y absolver por otros.
Crear sentencia	1	Ajustar el formato del documento para adecuarlo a las sentencias que se generan actualmente en los tribunales.
Registrar entrega de EFP a abogado	1	Mostrar en el documento de entrega de EFP a abogado la cantidad de folios útiles del expediente.
Registrar notificación de abogado/fiscal	1	Al notificar la sentencia a las partes, si el acusado está en prisión se le notifica a él y si no se notificaría al acusado o al abogado.
Registrar notificación de acusado	1	Incorporar el modelo de notificación personal para el acusado.
Tramitar apertura	1	Incluir siempre en el tramitar apertura el Oficio comunicando el número de causa al

		fiscal.
Crear auto disponiendo el sobreseimiento libre	1	Agregar la disposición sobre las medidas cautelares y bienes.
Crear auto de extinción de la responsabilidad penal	1	Agregar la disposición sobre las medidas cautelares y bienes.
Admitir pruebas y señalar	1	Mostrar el nombre de los defensores por las pruebas que proponen cada uno.
Registrar pruebas	1	Quitar las restricciones de formato y duplicidad de ese formulario.
Listar delitos	1	Listar los delitos en la forma simple que les correspondan.
Crear comunicación de número de causa a la prisión	1	Las generales que debe llevar son el nombre y los apellidos del acusado, el número de EFP y el número de causa.
Crear citación para acusado	2	Agregar el apercibimiento para el acusado en la citación.
Crear citación para testigo y perito	2	Agregar la dirección y nombre de la persona citada en la diligencia.
Crear orden de conducción al preso	1	Crear orden de conducción al preso.
Crear orden de conducción de testigo y perito	1	Crear orden de conducción de testigo y perito.
Auto de apertura	1	Agregar un escribir al final en la disposición sobre los bienes.

Crear resolución para disponer sobre la reclamación de documentos	1	Para el caso en que no se soliciten documentos, no debe salir ese trámite. Una vez que se conteste, se dispone directo sobre la cuestión o la vista, en caso de no tener ningún documento.

Tabla 2: Descripción de los casos de uso nuevos

Caso de uso nuevos	Total de RF afectados	Descripción del cambio
Modificar conclusiones de abogado y fiscal	1	Para el caso de que el que modifique sus conclusiones sea el abogado se consignará directamente en el acta del juicio y el documento se mantendrá en formato duro fuera del sistema. Si el que modifica sus conclusiones es el fiscal, se debe registrar esta modificación a través del usuario del fiscal antes de llegar al acta de votación.
Registrar nuevas pruebas en el juicio oral	1	El sistema debe permite que los abogados y fiscales registren nuevas pruebas en el desarrollo del juicio oral.
Refallar	2	Dejar un margen a la posibilidad de refallar, esta será una funcionalidad de oficio para el presidente de sala.
Modificar datos de los litigantes para los jueces	15	Agregar para los jueces profesionales la funcionalidad de Modificar datos de los litigantes.

2.3 Descripción de la arquitectura

Para el desarrollo del SITPC se define una arquitectura en capas basada en el patrón arquitectónico Modelo-Vista-Controlador (MVC) que es uno de los más utilizados en las aplicaciones web y también viene integrado al marco de trabajo Symfony 2 que fue el seleccionado para la implementación de la aplicación. La estructura que define el marco de trabajo Symfony2 para la separación de estas capas deja bien delimitado dónde quedan las clases del modelo, las de la vista y las del controlador. A continuación, se describe cada una de estas capas.

Vista

Es la capa con la que interactúan directamente los usuarios finales, siendo la encargada de representar los datos del modelo, gestionados por el controlador. Para esto se apoyan en la utilización del motor de plantillas Twig y las librerías jQuery y Bootstrap, que facilitan la construcción de interfaces bien acabadas y agradables a la vista del usuario. En el caso de SITPC las vistas se encuentran dentro del paquete “views”, las cuales transforman los datos obtenidos del modelo en vistas que son las que permiten al usuario interactuar con la aplicación.

Controlador

Esta capa es la intermediaria entre la vista y el modelo, ya que recibe las solicitudes o peticiones realizadas por el usuario desde la vista, ejecuta las acciones pertinentes y devuelve una respuesta a la interfaz con los resultados de las operaciones realizadas. Para SITPC, las clases que se encuentran dentro del paquete “Controller” responden a esta capa, y constituyen la parte de la aplicación que se encarga de realizar una funcionalidad completa y específica.

Modelo

La función de esta capa es servir de conector entre el controlador y el gestor de base de datos. En el caso particular del sistema, las clases gestoras que se encuentran en el paquete “Gestor” y las clases del repositorio que se encuentran en el paquete “Repository”, son las que responden a esta capa. También se encuentra el ORM Doctrine como marco de trabajo que permite la independencia de la aplicación respecto al gestor de base de datos, mediante su lenguaje propio de consultas DQL (Doctrine Query Language). Las entidades, por otra parte, también forman parte de esta capa; son la representación de las tablas de la base de datos previamente mapeadas por Doctrine, estas se encuentran en el paquete “vendor”.

2.1. Patrones de diseño

Un patrón de diseño se caracteriza como “una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución”. Para el diseño de software, el contexto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas

que influyen en la manera en la que puede interpretarse el problema en este contexto y en cómo podría aplicarse con eficacia la solución. (Pressman, 2010)

Para el diseño de la propuesta de solución se define por parte del equipo de arquitectura utilizar los Patrones Generales de Software para Asignación de Responsabilidades (GRASP) y los Patrones del grupo de los cuatros: patrones GoF, que se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns—Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides. A continuación, se enumeran los patrones GRASP y Gof utilizados en SITPC.

2.1.1. Patrones GRASP

Patrón Controlador: El objetivo de este patrón es asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas, o sea, delega la responsabilidad en otras clases con las que mantiene un modelo de alta cohesión. Este patrón se evidencia en la solución a través de las clases controladoras que se encuentran en la carpeta Controller perteneciente a cada Bundle de la solución.

Patrón Experto: Este patrón define que la clase experta en información es la que debe llevar la responsabilidad, ya que esta última es la que contiene toda la información para cumplir con la responsabilidad. Este patrón se observa en la solución al crear las vistas para mostrárselas al usuario; así como en las clases Controladoras, que son las expertas en información, son las encargadas de crear los formularios, y a través de los mismos se creará la vista que se le mostrará al usuario.

Patrón Creador: En la implementación del SITPC este patrón se evidencia en las clases Gestoras en el momento de crear nuevas instancias de las clases Entidades. Otro ejemplo son las clases Controladoras, que son las responsables de la creación de los formularios que se mostrarán en las vistas.

Patrón Bajo acoplamiento: Este patrón plantea la baja dependencia que debe existir entre las clases, además está estrechamente relacionado con los patrones Experto o Alta Cohesión. Symfony favorece ampliamente el bajo acoplamiento de las clases en el sistema, esto ocurre porque a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras. Este patrón se observa en el uso del patrón arquitectónico MVC del SITPC ya que las clases de las vistas no tienen relación con las clases gestoras y las entidades, que se refieren al Modelo, y estas últimas no presentan ninguna relación con las clases controladoras que representan

el Controlador. Esto favorece la escalabilidad del sistema, la reutilización y el mantenimiento futuro de la aplicación.

Patrón Alta cohesión: Este patrón tiene como propósito asignar responsabilidades de manera que la cohesión siga siendo alta. El marco de trabajo Symfony favorece la alta cohesión asignando responsabilidades a las clases de tal manera que estas se encuentren estrechamente relacionadas entre sí y no lleguen a realizar un trabajo excesivo. En la implementación del SITPC este patrón se pone de manifiesto en la interrelación que existe entre las clases controladora, las clases gestora y las funcionalidades del sistema, donde cada una de estas presenta una clase controladora y una gestora, la primera encargada de manejar la lógica de presentación y el flujo de los datos provenientes de la vista y la segunda encargada de manejar la lógica del negocio de cada funcionalidad.

2.1.2. Patrones GoF

Patrón Factory Method: Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar; su objetivo es devolver una instancia de múltiples tipos de objetos, que normalmente provienen de una misma clase padre y sólo se diferencian entre ellos por algún aspecto de comportamiento. En la implementación del SITPC este patrón es utilizado cuando se realiza una petición al contenedor de servicios, en la que se le pasa por parámetros el servicio que tiene definido el repositorio que se desea instanciar; internamente el contenedor de servicios determina cuál es la entidad que corresponde a dicho repositorio y realiza una llamada a la clase EntityManager.php, pasándole por parámetro la entidad calculada. Dicha clase es la encargada de implementar el patrón Factory Method, esto se realiza específicamente en el método getRepository(entityName).

Patrón Decorador: El principal objetivo de este patrón es añadir responsabilidades a objetos concretos de manera dinámica y transparente sin afectar a otros objetos. Este patrón brinda más flexibilidad que la herencia estática y evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas. En la implementación del SITPC este patrón se evidencia en el uso de una plantilla global para las vistas que decorará las demás páginas de la aplicación.

Patrón Instancia única: aparece por defecto en la estructura de Symfony2, se evidencia en el uso del controlador frontal que se encuentra en la carpeta web de cada proyecto creado con dicho marco de trabajo.

2.1.3. Otros patrones

Patrón Inyección de dependencias

Este patrón es utilizado en los contenedores de servicios (o contenedores de inyección de dependencias), los cuales son objetos PHP que gestionan la creación de instancias de servicios, es decir, objetos. Esto se evidencia en la relación que se establece entre los gestores y los controladores. Cada controlador tiene asociado un gestor para realizar las acciones de acceso a datos. Estos gestores son publicados como servicios en el archivo "services.yml" y cuando una clase controladora necesita hacer uso de su gestor; en lugar de crear una instancia del mismo, simplemente llama a un método denominado getGestor() el cual accede al contenedor de servicios, que es el encargado de hacer la instancia del gestor solicitado, en tiempo de ejecución y de destruirla posteriormente cuando ya no la necesite.

2.4 Análisis y diseño

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos y una descripción que sea fácil de mantener y que ayude a la estructuración del sistema. Además, se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales, los modelos desarrollados son más formales y específicos que el de análisis. (Sánchez, 2014)

2.4.1 Diagrama de clases de diseño

En el siguiente diagrama de clases se muestran los archivos correspondientes e involucrados en el caso uso Gestionar Delito. Entre las que se encuentran:

- La clase "GestionarDelito.html.twig" que representa la capa de presentación o vista del patrón MVC. Es la interfaz con la que el usuario interactúa, mostrándole los datos necesarios y las acciones que puede llevar a cabo.
- La clase controladora "GestionarDelitoController", encargada de las funcionalidades que debe realizar el caso de uso.
- La clase "GestionarDelitoGtr" encargada de listar, registrar, generar documentos y una serie de acciones necesarias para dar cumplimiento a las funcionalidades del caso de uso.
- Las clases Form son una representación de ventanas mostradas en la aplicación. Encargadas de crear ventanas estándar, ventanas modales y herramientas.
- Las clases Repository son una representación abstracta de la lógica de negocio, permiten llevar el manejo desde la base de datos a una clase aislada del controlador.
- Las clases Entidades son la representación de las tablas de la base de datos.

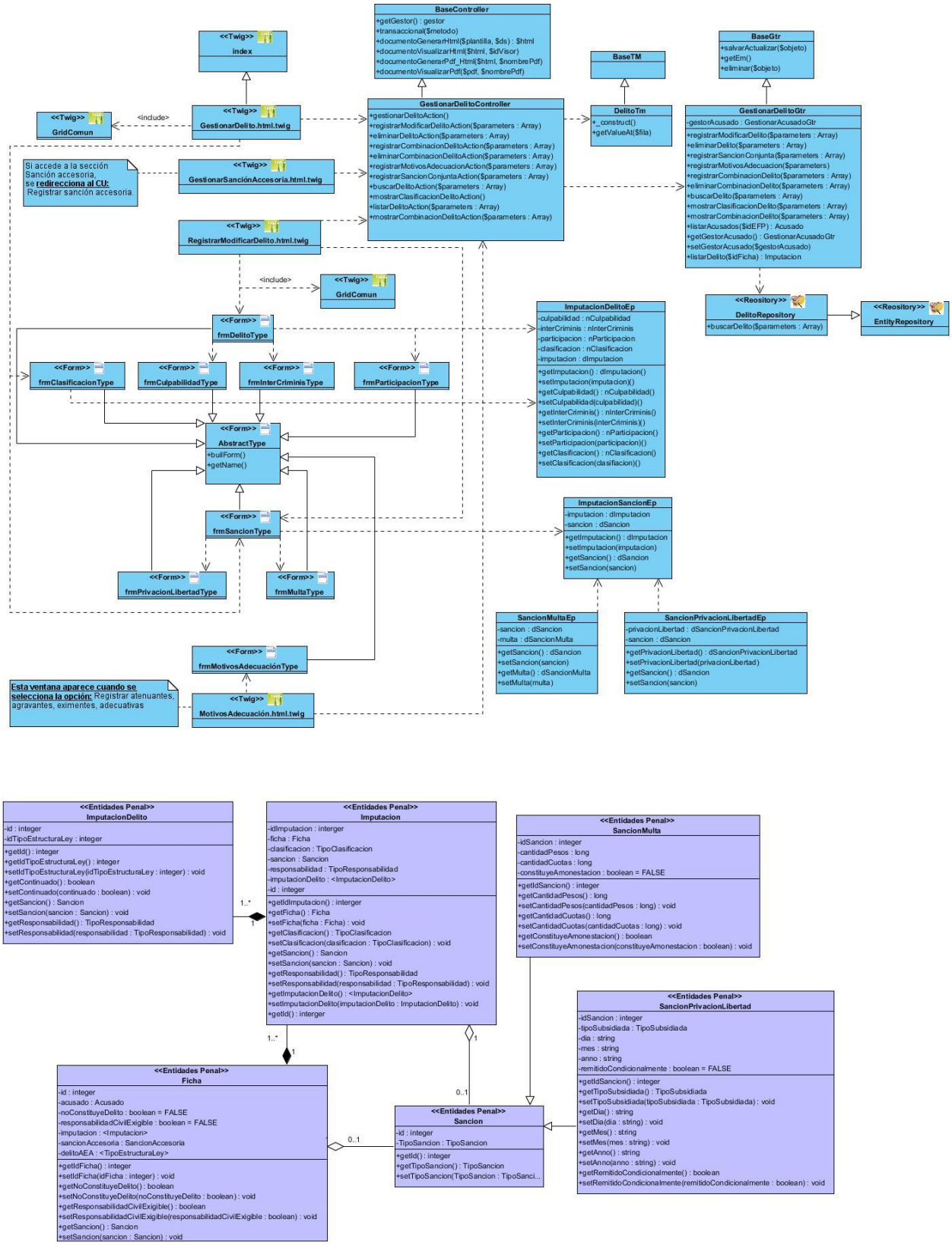

Figura 3: Diagrama de clases de diseño: CU_Gestionar delito

2.4.2 Diagrama de secuencia

Los diagramas de secuencia muestran las interacciones entre objetos, ordenadas en secuencia temporal y durante un escenario concreto. Estos diagramas son importantes para modelar los aspectos dinámicos de un sistema. (Pressman, 2010)

A continuación, se presenta el diagrama de secuencia para el caso de uso Eliminar Delito:

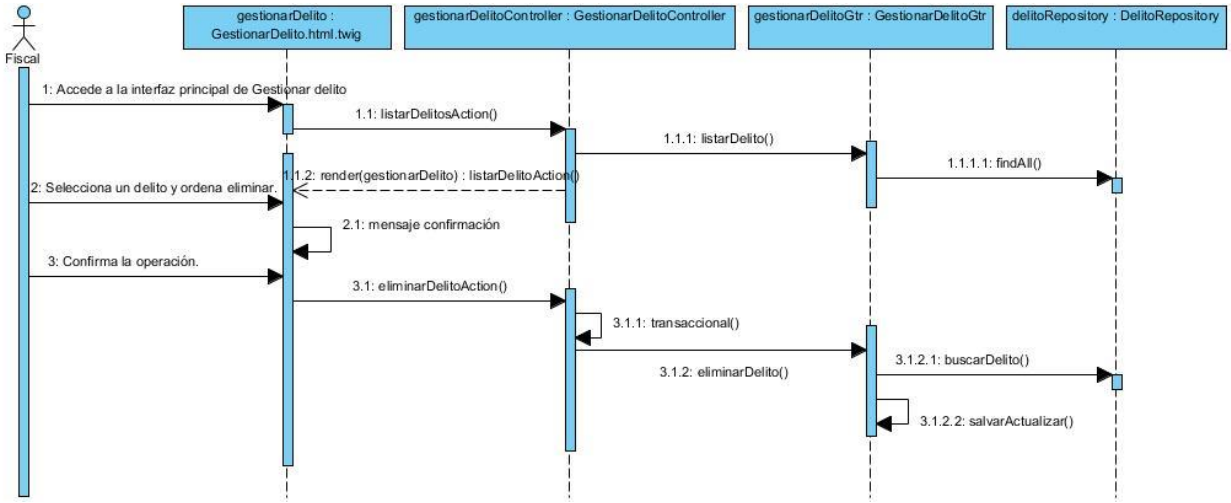

Figura 4: Diagrama de secuencia: CU_Eliminar delito

2.4.3 Modelo de datos

Un modelo de base de datos muestra la estructura lógica del sistema, incluidas las relaciones y limitaciones, que determinan cómo se almacenan los datos y cómo se accede a ellos. Los modelos de bases de datos individuales se diseñan en base a las reglas y los conceptos de cualquier modelo de datos más amplio que los diseñadores adopten. La mayoría de los modelos de datos se pueden representar por medio de un diagrama de base de datos acompañante. (Lucidchart, 2018)

Para evidenciar el impacto de las solicitudes de cambio en la base de datos, se muestran algunas de las tablas que sufrieron cambios. Cabe destacar que la base de datos brinda gran flexibilidad, ya que al implementar la primera versión del sistema se previó que pudiesen ocurrir cambios. En la tabla "*defp*" se agregó un nuevo campo "*FoliosUtiles*" con el objetivo de captar la cantidad de páginas que tiene cada expediente de fase. Además, en la tabla "*dimputacion*" se agregó un campo de texto "*relacionadocon*" para poder agregar los artículos o apartados de otras leyes como el código de vialidad y tránsito que pudiesen estar relacionados con los delitos que se calificaron. Otra de las solicitudes de cambio era que el sistema debía permitir a los jueces absolver a los acusados por algunos de los delitos que le imputaba el fiscal y/o sancionarlos por otros, en caso de absolverlos debía permitir además insertar el motivo por el cual se absuelve; para lo cual fue necesario crear la nueva tabla "*dabsolucion*" y para guardar los motivos de absolución se creó el nomenclador "*nmotivoabsolucion*".

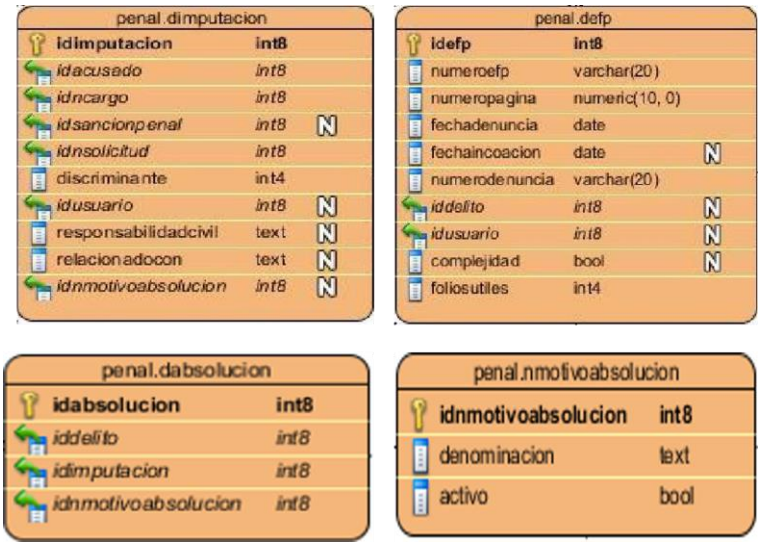

Figura 5: Representación de las clases del modelo de datos afectadas con las SC

2.4.4 Diagrama de despliegue

Un Diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema**.** Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. (SparxSystems, 2018)

A continuación, se presenta el diagrama de despliegue del SITPC:

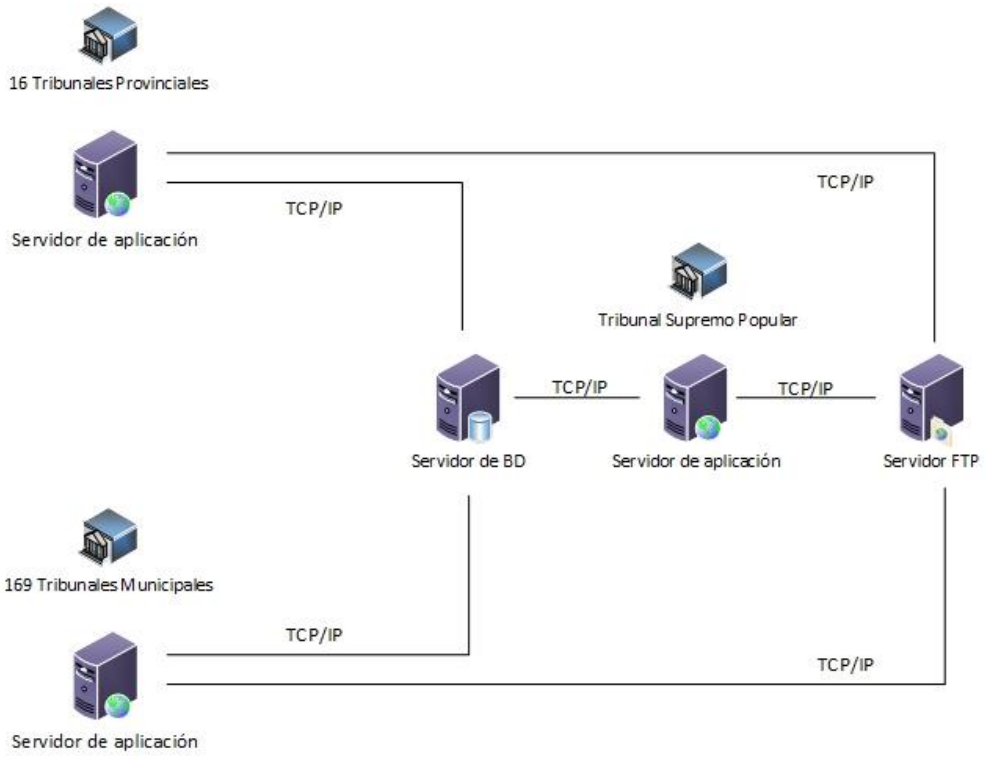

Figura 6: Diagrama de despliegue para el SITPC

2.5 Implementación

En la implementación, a partir de los resultados de la disciplina de Análisis y diseño, se construye el sistema. (Sánchez, 2014)

2.5.1 Estándares de codificación

El estándar de codificación para el desarrollo del SITPC fue seleccionado por el equipo de arquitectura del proyecto. A continuación, se describen cada uno de estos aspectos que se tuvieron en cuenta al implementar la propuesta de solución.

- **Identación**

El contenido siempre se identará con tabs, nunca utilizando espacios en blanco.

- **Cabecera del archivo**

Es importante que todos los archivos *.php inicien con una cabecera específica que indique información de la versión, autor de los últimos cambios, etc. Es decisión de cada equipo decidir si se quieren o no agregar más datos (ver **Figura 7**).

```
/**
 * Clase controladora " CalificacionDelitoController.php"
 *
 * @modificado: 1 de Diciembre del 2017
 *
 * @author rosalia
 */

class CalificarDelitoControllerr {
    //put your code here
}
```

Figura 7: Cabecera del archivo

- **Comentarios en las funciones**

Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben ser suficiente para entender el código.

- **Ubicación y denominación de archivos**

Se ubicarán los archivos según las convenciones establecidas por Symfony2 o según las especificaciones del equipo de arquitectura.

Para la denominación de los archivos se seguirán las convenciones establecidas por Symfony2 o el equipo de arquitectura. A continuación, se muestran los estándares definidos para cada una de las clases:

  - En el caso de las clases de gestión del negocio se usará el sufijo Gtr, por ejemplo: GestionarDelitoGtr.

- En el caso de los table model definidos para los grid se usará el sufijo Tm, por ejemplo: DelitoTm.
- En las entidades de presentación se usará el sufijo Ep, por ejemplo: DelitoEp.
- En las páginas o plantillas html.twig definidas para la vista el nombre del archivo debe seguir el estándar de la denominación de las clases, por ejemplo: GestionarDelito.html.twig.

Para la denominación de los servicios a crear se tendrán en cuenta los siguientes elementos:

- Para los parámetros se utilizará la siguiente nomenclatura: bundle.entidad.categoria.class (todo en minúscula utilizando como separador el carácter punto (.)).
  - Ejemplo:
    ordinario.gestionardelito.gtr.class:
    ordinario.delito.tm.class:
- Para los servicios se utilizará la siguiente nomenclatura: bundle.entidad.categoria (todo en minúscula utilizando como separador el carácter punto (.)).
  - Ejemplo:
    ordinario.gestionardelito.gtr:
    ordinario.delito.tm:
- Para las rutas se utilizará la siguiente nomenclatura: bundle_ruta (todo en minúscula utilizando como separador el carácter guion bajo (_))
  - Ejemplo:
    ordinario_inicio

- **Clases**

Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo del de la clase. Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad. Los nombres de las clases deben iniciar con letra mayúscula. Si un nombre de clase se comprende de más de una palabra, la primera letra de cada nueva palabra debe comenzar con letra mayúscula. No se permiten las letras mayúsculas sucesivas; por ejemplo, una clase "SymfonyPDF" no se permite, mientras " SymfonyPdf" es aceptable.

Siempre utilizar las etiquetas <?php ?> para abrir un bloque de código. No utilizar el método de etiquetas cortas.

- **Estilo y reglas de escritura de código PHP**
  - **Nombres de variables**

Los nombres deben ser descriptivos y concisos. No usar grandes frases ni pequeñas abreviaciones para las variables. Siempre es mejor saber qué hace una variable con solo conocer su nombre. Esto se aplica para los nombres de variables, funciones, argumentos de funciones y clases. Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula. Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.

- **Las definiciones de la función**

Los nombres de las funciones pueden contener solo caracteres alfanuméricos y siempre deben empezar en letras minúsculas. Cuando el nombre de una función conste de más de una palabra, la primera letra de cada nueva palabra debe comenzar con mayúscula. (ver Figura 8)

- **Llamadas a funciones**

Deben llamarse las funciones sin los espacios entre el nombre de la función, el paréntesis de la apertura, y el primer parámetro; los espacios entre las comas y cada parámetro, y ningún espacial entre el último parámetro, el paréntesis del cierre, y el punto y coma (ver **Figura 8**).

```
/**
 * @author Julio luis
 * @param $idtramite
 * @return \Base\ComunBundle\Entity\AG\Tramite|object
 */
public function buscarTramite($idtramite)
{
    return $this->getEm()->getRepository('ComunBundle:AG\Tramite')->find($idtramite);
}
```

Figura 8: Llamadas a funciones

- **Siempre incluir las llaves**

En todo momento a la hora de codificar un bloque de instrucciones, este debe ir encerrado entre llaves, aun cuando conste de una sola línea (ver **Figura 9**).

- **Llaves**

Las llaves de apertura irán al final de la sentencia que delimitan, y las de cierre alineadas con el inicio de la sentencia en una nueva línea (ver **Figura 9**).

```
public function array_differencia($arrGen, $arr)
{

    $arrResult = array();
    $cont1 = count($arrGen);
    $cont2 = count($arr);
    for ($i = 0; $i < $cont1; $i++) {
        $esta = false;
        for ($j = 0; $j < $cont2; $j++) {
            if ($arrGen[$i]->getId() == $arr[$j]->getId()){
                $esta = true;
            }
        }
        if ($esta == false){
           $arrResult[] = $arrGen[$i];
        }
    }
    return $arrResult;
}
```

Figura 9: Estándar de las funciones y las llaves

2.6 Conclusiones parciales

El análisis del proceso de las incidencias facilitó la obtención de la descripción de los cambios en el sistema como consecuencia de las solicitudes de cambio. Además, el análisis de la arquitectura del sistema y de los patrones asociados al diseño, los diagramas de clases, el modelo de datos, así como los diagramas de secuencia y despliegue, sentaron las bases para la disciplina de Implementación. Por último, los estándares de codificación permitieron la creación del código más legible o entendible.

Capítulo 3: Validación

Introducción

En el presente capítulo se muestran los resultados de la aplicación de las métricas para validar el diseño. De igual forma se presentan los resultados de la aplicación las pruebas con el objetivo de evaluar la calidad del sistema una vez concluida la disciplina de implementación.

3.1 Validación del diseño. Métricas de diseño OO[5]

Las métricas de software constituyen los elementos que permiten evaluar la calidad de una determinada característica o artefacto que se genere en un proyecto de software. Para la validación del diseño de la solución propuesta, en el proyecto se estudiaron las diferentes métricas de diseño y sus características.

Con el objetivo de medir el nivel de relaciones entre las clases del sistema y la complejidad de cada clase por separado se seleccionaron 2 métricas que permiten realizar mediciones de este tipo: Relaciones entre clases y Tamaño de clase. A continuación, se muestran los resultados de la aplicación de estas métricas.

3.1.1 Métricas de diseño

Una métrica permite medir de forma cuantitativa la calidad de los atributos internos del software y de esta forma permite al ingeniero evaluar la calidad del diseño durante el desarrollo del sistema. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo. (Pressman, 2010)

- **Relaciones ente clases (RC)**

Esta métrica está dada por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- ✓ Acoplamiento: Un aumento del RC implica un aumento del Acoplamiento de la clase.
- ✓ Reutilización: Un aumento del RC implica una disminución en el grado de reutilización de la clase.

En la siguiente tabla se muestran las medidas utilizadas para evaluar cada uno de estos parámetros de calidad.

Tabla 3: Criterios de evaluación para la métrica RC

Parámetros	Categoría	Criterio

[5] Orientadas a objetos

Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

La siguiente tabla muestra un ejemplo del resultado de la aplicación de esta métrica al diseño de la aplicación.

Tabla 4: Relación entre clases

Clase	RC[6]	Acoplamiento	Reutilización
GestionarDelito.html.twig	5	Alto	Media
GestionarSancionAccesoria.html.twig	1	Bajo	Alta
RegistrarModificarDelito.html.twig	4	Alto	Media
MotivosAdecuación.html.twig	2	Medio	Alta
frmDelitoType	6	Alto	Baja
frmClasificacionType	3	Alto	Media
frmCulpabilidadType	2	Medio	Alta
frmInterCriminisType	2	Medio	Alta
frmParticipacionType	2	Medio	Alta
frmSancionType	6	Alto	Baja
frmPrivacionLibertadType	2	Medio	Alta
frmMultaType	2	Medio	Alta
frmMotivosAdecuacionType	2	Medio	Alta

[6] Cantidad de Relaciones entre Clases

BaseControler	1	Bajo	Alta
GestionarDelitoController	7	Alto	Baja
DelitoTm	2	Medio	Alta
BaseGtr	2	Medio	Alta
GestionarDelitoGtr	4	Alto	Media
ImputacionDelitoEp	2	Medio	Alta
ImputacionSancionEp	3	Alto	Media
SancionPrivacionLibertadEp	1	Bajo	Alta
SancionMultaEp	1	Bajo	Alta

A continuación, se muestran las figuras con los resultados obtenidos:

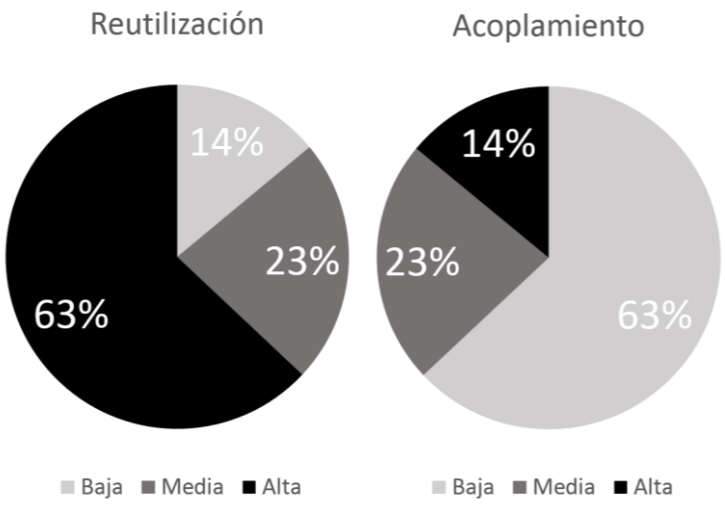

Figura 10: Representación de la incidencia de los resultados de la evaluación de la métrica RC en los atributos Acoplamiento y Reutilización.

Al analizar los resultados obtenidos para cada uno de los atributos de calidad que mide esta métrica se evidencia que el 63 % de las clases posee un acoplamiento Bajo, un 23 % Medio y un 14 % Alto. Por otra parte, el 63% de las clases presenta una alta reutilización, el 23 % media y el 14% baja.

- **Tamaño de clase (TC)**

Consiste en medir el tamaño general de una clase tomando los siguientes valores:

- ✓ El total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- ✓ El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.

En la siguiente tabla se muestran las medidas utilizadas para evaluar cada uno de estos parámetros de calidad.

Tabla 5: Criterios de evaluación para la métrica TC

Parámetros	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Tabla 6: Relaciones de Uso

Clase	RU[7]	Responsabilidad	Complejidad	Reutilización
GestionarDelito.html.twig	5	Baja	Baja	Alta
GestionarSancionAccesoria.html.twig	1	Baja	Baja	Alta
RegistrarModificarDelito.html.twig	4	Baja	Baja	Alta
MotivosAdecuación.html.twig	2	Baja	Baja	Alta
frmDelitoType	6	Baja	Baja	Alta
frmClasificacionType	3	Baja	Baja	Alta
frmCulpabilidadType	2	Baja	Baja	Alta
frmInterCriminisType	2	Media	Media	Media

[7] Cantidad de Relaciones de Uso

frmParticipacionType	2	Media	Media	Media
frmSancionType	6	Media	Media	Media
frmPrivacionLibertadType	2	Media	Media	Media
frmMultaType	2	Media	Media	Media
frmMotivosAdecuacionType	2	Media	Media	Media
BaseControler	1	Alta	Alta	Baja
GestionarDelitoController	7	Alta	Alta	Baja
DelitoTm	2	Baja	Baja	Alta
BaseGtr	2	Alta	Alta	Baja
GestionarDelitoGtr	4	Alta	Alta	Baja
ImputacionDelitoEp	2	Alta	Alta	Baja
ImputacionSancionEp	3	Baja	Baja	Alta
SancionPrivacionLibertadEp	1	Baja	Baja	Alta
SancionMultaEp	1	Baja	Baja	Alta

Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas de dicha clase. Por tanto, mientras menor sea el valor para el TC se hará mucho más fácil la reutilización de dicha clase dentro del sistema. Una vez aplicada dicha métrica a las clases de la solución, la misma arrojó los siguientes resultados:

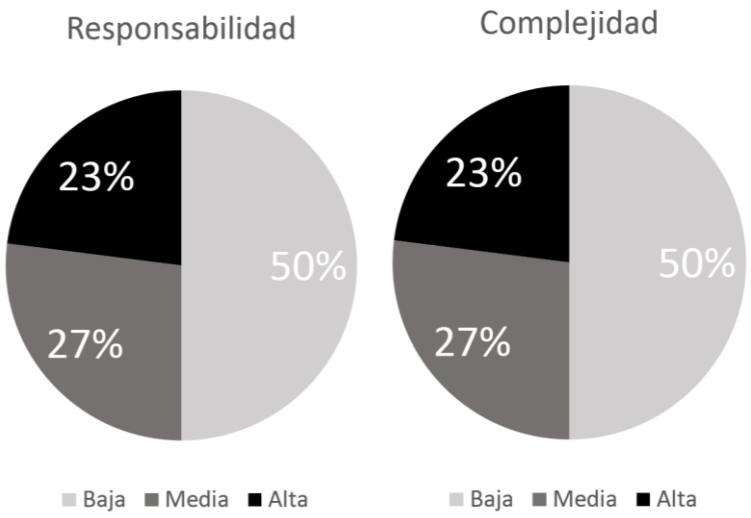

Figura 11: Representación de la incidencia de los resultados de la evaluación de la métrica TC en los atributos Responsabilidad y Complejidad de implementación.

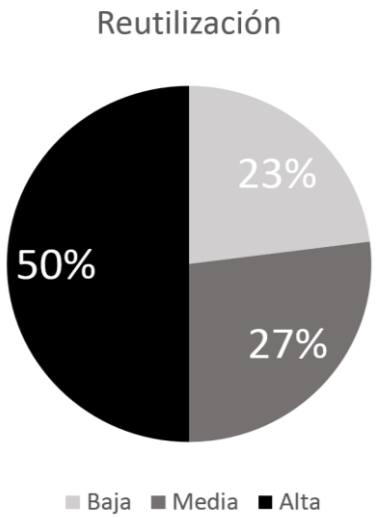

Figura 12: Representación de la incidencia de los resultados de la evaluación de la métrica TC en el atributo Reutilización.

Después de aplicada la métrica se puede concluir que el diseño realizado tiene una buena calidad pues se aprecian resultados satisfactorios en los atributos de calidad medidos, ya que se obtuvieron bajos niveles de responsabilidad (50%) y de complejidad de implementación (50%), evidenciándose la correcta asignación de responsabilidades a las clases involucradas en la solución. Además, se puede asegurar que la solución obtenida es eficiente, con altos niveles de reutilización (50%).

Luego de aplicadas estas métricas, los datos obtenidos demuestran que las clases están concebidas de forma tal que se facilite su implementación, ya que al presentar un acoplamiento relativamente

bajo se obtiene una reutilización alta, lo que implica que al realizar el desarrollo de un sistema con la complejidad del SITPC la mayoría de las clases son reutilizadas en más de un caso de uso.

3.2 Validación de la implementación

Las pruebas y validaciones de software son de vital importancia para corroborar que la aplicación desarrollada no presente problemas de ejecución y se encuentre libre de errores, ya que durante el proceso de desarrollo de software es frecuente que los desarrolladores, al programar las diferentes funcionalidades, obvien algunas posibilidades que pueden variar el resultado esperado durante la ejecución del código; por tanto, estas pruebas y validaciones constituyen la vía a través de la cual se asegura que el producto desarrollado está listo para ser entregado a los usuarios finales. Los errores más frecuentes pueden suceder a causa del mal uso de estructuras de datos, errores lógicos, entre otras. (Pressman, 2010)

Teniendo en cuenta la metodología de desarrollo que guía el presente trabajo, se decide aplicar las disciplinas de Pruebas internas y de aceptación.

3.2.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. (Sánchez, 2014)

Para validar la implementación como producto de la presente investigación, se tuvo en cuenta las pruebas funcionales y las pruebas unitarias:

- **Pruebas funcionales**: son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema. Se realizan para comprobar si el software cumple las funciones esperadas. Deben enfocarse en los requisitos funcionales y pueden estar basadas directamente en los Casos de Uso, Requisitos o Historias de Usuarios. (Bord, 2010)
- **Pruebas unitarias:** Estas pruebas enfocan los esfuerzos de verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Las pruebas de unidad se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes. (Pressman, 2010)

A continuación, se describen como se aplicaron ambas pruebas en la propuesta de solución a través a través de los métodos de Caja negra y Caja blanca.

Método de Caja negra

El método de caja negra, se centra en los requisitos funcionales del software, o sea, permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente

todos los requisitos funcionales de un programa. Evalúa las funcionalidades del software (lo que hace). (Pressman, 2010)

Este método se llevó a cabo por parte del equipo de desarrollo mediante la técnica de Partición equivalente y Análisis de valores límites.

- **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. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de todos los datos de carácter) que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. (Pressman, 2010)

Como complemento a este técnica según (Pressman, 2010), se debe tener en cuenta la técnica de valores límites.

- **Análisis de valores límites**

El análisis de valor de frontera es una técnica de diseño de casos de prueba que complementan la partición de equivalencia. En lugar de seleccionar algún elemento de una clase de equivalencia, el BVA conduce a la selección de casos de prueba en los "bordes" de la clase. En lugar de enfocarse exclusivamente en las condiciones de entrada, el BVA[8] también deriva casos de prueba a partir del dominio de salida. (Pressman, 2010)

En el **Anexo 1** se muestra la descripción de las variables para el requisito "Adicionar delito". Luego de definir el tipo de variables y las descripciones se procede a realizar el diseño de caso de pruebas para este requisito, en el **Anexo 2** se muestra el Diseño de caso de pruebas para dicho requisito, el cual pertenece al Caso de uso "Gestionar delito".

Durante la ejecución de estas pruebas se realizaron tres iteraciones por cada caso de uso, donde los errores detectados fueron solucionados conllevando a que en la 3ra iteración no se detectaron errores. Entre los errores detectados se encuentran: errores ortográficos, de validación, de interfaz. A continuación, se presentan los resultados obtenidos al aplicar esta prueba al producto obtenido.

Tabla 7: Resultado de las iteraciones al aplicar la prueba de caja negra

Iteraciones	No conformidades

[8] El BVA extiende la partición de equivalencia al enfocarse en datos en los "bordes" de una clase de equivalencia.

Iteración 1	20
Iteración 2	6
Iteración 3	0

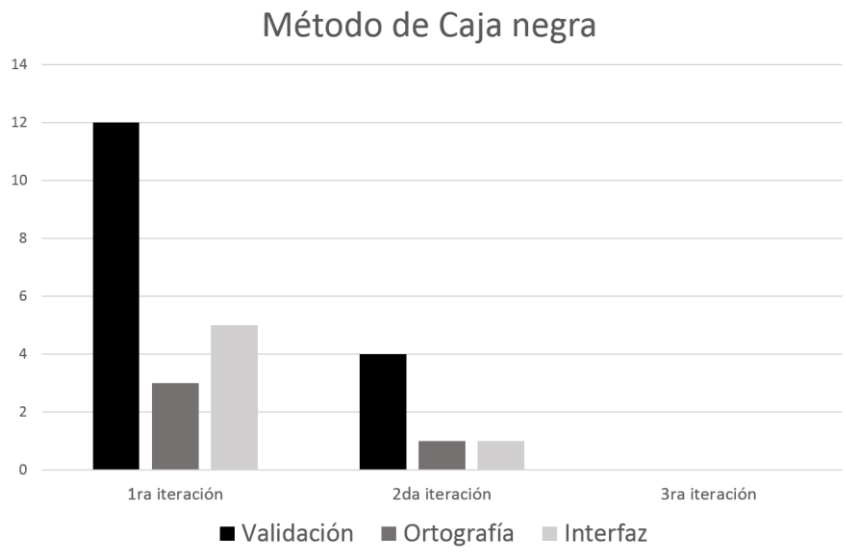

Figura 13: Resultados de la método de Caja negra.

Método de caja blanca

El método de caja blanca del software se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles.

Este método se realiza sobre las funcionalidades internas de un módulo y se encargan de comprobar los caminos lógicos, ciclos (bucles) y condiciones que debe cumplir el programa. Con el empleo de este método es posible desarrollar casos de prueba que garanticen la ejecución, al menos una vez, de los caminos independientes (camino que recorre por lo menos una nueva arista en el grafo de flujo). (Pressman, 2010)

Este método se aplicó teniendo en cuenta la técnica de ruta básica. A continuación, se describe los resultados de esta técnica.

- **Técnica de ruta básica**

La técnica de ruta básica permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico

tienen garantía para ejecutar todo lo enunciado en el programa, al menos una vez durante la prueba. (Pressman, 2010)

La complejidad ciclomática es una medición de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y le brinda una cota superior para el número de pruebas que debe realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez. (Pressman, 2010)

La complejidad ciclomática tiene fundamentos en la teoría de gráficos y proporciona una medición de software extremadamente útil. La complejidad se calcula en una de tres formas: (Pressman, 2010)

1. El número de regiones **(R)** del gráfico de flujo corresponde a la complejidad ciclomática
2. La complejidad ciclomática V(G) para un gráfico de flujo G se define como

   V(G) = E - N +2

   donde **E** es el número de aristas del gráfico de flujo y **N** el número de nodos del gráfico de flujo.
3. La complejidad ciclomática V(G) para un gráfico de flujo G también se define como

   V(G) = P+ 1

   donde **P** es el número de nodos predicado contenidos en el gráfico de flujo G.

El valor para **V(G)** proporciona una cota superior para el número de rutas independientes que forman el conjunto básico y, por implicación, una cota superior sobre el número de pruebas que deben diseñarse y ejecutarse para garantizar cobertura de todos los enunciados del programa. (Pressman, 2010)

En la **Figura 14** se muestra el método EliminarDelito() utilizado en el caso de uso Gestionar delito. Se selecciona el mismo ya que es la acción donde se eliminan las atenuantes del acusado al calificar el delito, y por la importancia que tiene al actualizar una información previamente insertada por el usuario.

```
public function EliminarDelito(TipoEstructuraLey $objeto, $idacusado, $cargo)
{
1   $acusado = $this->BuscarAcusado($idacusado);
    $imputaciones = $acusado->getImputacion();
2   foreach ($imputaciones as $imputacion) {
3       if ($imputacion->getCargo() == $cargo) {
4           $imputacion->removeEstructuraLey($objeto);
5       }
6   }
7   $this->getEm()->flush();
}
```

Figura 14: Método EliminarDelito utilizado en el caso de uso Gestionar delito

A continuación, se procede a realizar el grafo de flujo para el método previamente descrito:

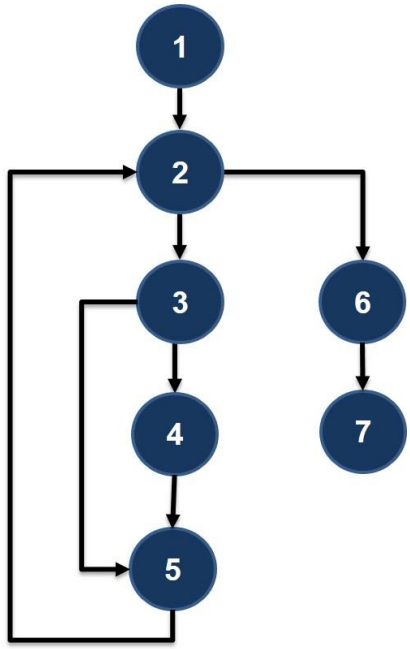

Figura 15: Grafo de flujo para el método EliminarDelito() previamente descrito

A dicho grafo de flujo se le calculó la complejidad ciclomática, calculada por las tres vías conocidas, obteniéndose los siguientes resultados:

1. **V(G)** = R = 3.
2. **V(G)** = (8 – 7) + 2 = 3.
3. **V(G)** =2 + 1 = 3.

Después de calcular la complejidad del grafo, se pudo comprobar que los resultados obtenidos son iguales a 3; por tanto, se deben concebir 3 rutas independientes, y por cada una elaborar un caso de prueba para aplicarlo al método en cuestión. Las rutas resultantes fueron:

1) 1-2-3-4-5-2-6-7

2) 1-2-3-5-2-6-7
3) 1-2-6-7

Cada ruta independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvieron 3 rutas independientes, que dan lugar a la confección de igual número de casos de pruebas, para aplicar las pruebas a este método. A continuación, se muestran el caso de prueba para la ruta independiente #1, el resto se pueden consultar en los **Anexos 3 y 4** respectivamente:

Tabla 8: Caso de prueba. Ruta independiente #1

Caso de prueba: Ruta independiente #1	
Descripción: este camino verifica la existencia del cargo que se introduce por parámetro para luego eliminar los atenuantes al calificar el delito.	
Entrada	Tipo de Estructura ley, el identificador del acusado al cual se le eliminarán los atenuantes y el cargo que se desea eliminar de ese acusado.
Condiciones	El acusado debe poseer el cargo que se introduce por parámetro.
Resultados esperados	Se eliminan las atenuantes del acusado.

Descripción de la ejecución de los casos de prueba

Para ejecutar cada caso de prueba se realizaron pruebas manuales utilizando un modelo ejecutable al gestionar delitos en el sistema. En el caso de prueba # 1 se introdujo un cargo que coincidiera con los que tenían asociado ese acusado, de tal forma que al hacer la comprobación eliminara la estructura ley asociada a la imputación de ese acusado. Una vez ejecutado este caso de prueba el resultado fue el esperado. En el caso de prueba # 2, se introdujo un cargo que no coincidiera con los que tenía asociado ese acusado, de tal forma que al hacer la comprobación no eliminaría la estructura ley asociada a las imputaciones de ese acusado, siendo esta la respuesta del sistema. En el caso de prueba # 3, se introdujo un identificador de acusado de forma tal que no tuviera imputaciones y por lo cual no eliminaría ninguna de las estructuras ley asociada a las imputaciones de ese acusado. El resultado de este caso de prueba fue el esperado.

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica ruta básica, se concluye que los mismos fueron probados satisfactoriamente, demostrando que todas las rutas de este código se ejecutaron al menos una vez.

3.2.2 Pruebas de aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. (Sánchez, 2014)

Estas pruebas se llevaron a cabo por parte del cliente con el objetivo de verificar que las solicitudes de cambio estaban listas para de conjunto con el SITPC ser utilizadas por los usuarios finales. Para estas pruebas se realizaron 2 iteraciones, en la 1ra se detectaron solo 4 NC, 2 de Validación, 1 de Ortografía y 1 de Interfaz. Ya en la 2da iteración los errores fueron resueltos satisfaciendo las necesidades del cliente. Estos resultados quedan reflejados en la siguiente imagen:

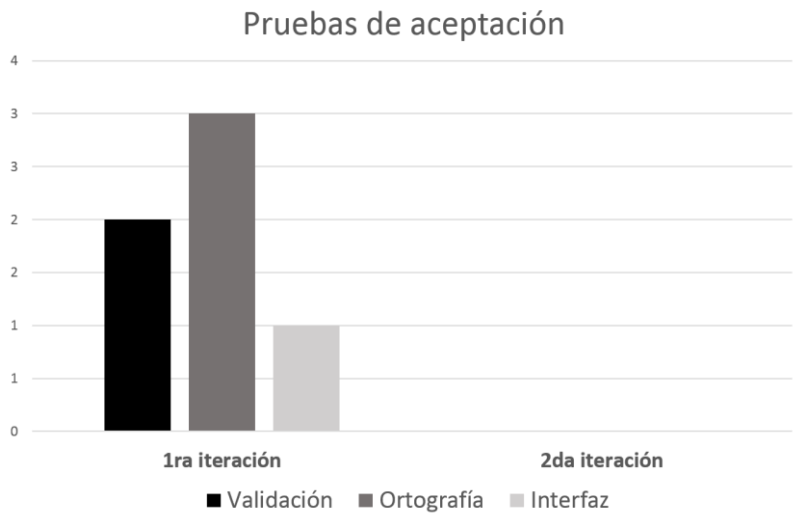

Figura 16: Resultados de la Prueba de aceptación

Una vez realizadas las pruebas de aceptación, se generó el Acta de aceptación del producto como constancia de la conformidad por parte del cliente de la solución propuesta (ver **Anexo 4**).

3.3 Conclusiones parciales

La aplicación de las métricas de validación del diseño a través de los atributos de Responsabilidad, Acoplamiento, Reutilización y Complejidad de implementación, corroboró que el sistema está diseñado de forma tal que se facilite su implementación, ya que al presentar un acoplamiento relativamente bajo se obtiene una reutilización alta. La validación de la implementación del paquete de solicitudes de cambio permitió evaluar la calidad del sistema con el fin de entregarle al cliente un producto que satisface sus necesidades.

Conclusiones generales

Con el desarrollo del paquete de solicitudes de cambio del procedimiento ordinario Penal perteneciente al SITPC se satisfacen las necesidades demandadas por los especialistas funcionales de este software. Es por ello que al finalizar la presente investigación se puede concluir afirmando que:

- La elaboración del marco teórico de la investigación sustentó los principales referentes teóricos en los que se enmarca el desarrollo de la propuesta de solución.
- El Modelo de sistema obtenido, a partir de las solicitudes de cambios identificadas, permitió obtener un acercamiento de las necesidades a desarrollar.
- La obtención del Modelo de implementación logró la especificación técnica o algoritmos como un programa que respondan a las funcionalidades definidas en el Modelo del sistema.
- La validación de los resultados obtenidos a través de las métricas de validación del diseño y las pruebas de software permitió comprobar de forma cuantitativa la calidad de los artefactos obtenidos durante el desarrollo del sistema.

Referencias

- **2ndQuadrant+PostgreSQL. 2018.** 2ndquadrant.com. [En línea] 2018. [Citado el: 12 de Febrero de 2018.] https://www.2ndquadrant.com/es/postgresql/.
- **4rsoluciones. 2013.** 4rsoluciones. *¿Por qué elegir Symfony2?* [En línea] 12 de junio de 2013. [Citado el: 18 de enero de 2018.] http://www.4rsoluciones.com/blog/por-que-elegir-symfony2-2/.
- **Alicante, Universidad de. 2018.** Universidad de Alicante. *www.ua.es.* [En línea] 2018. [Citado el: 15 de 2 de 2018.] https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.
- **AprendiendoArduino. 2014.** Aprendiendo Arduino. [En línea] diciembre de 2014. [Citado el: 22 de febrero de 2018.] https://aprendiendoarduino.wordpress.com/2016/03/29/entorno-de-programacion-de-arduino-ide/.
- **Barzanallana, Rafael. 2011.** Universidad de Murcia. *Ingeniería del software. Metodologías habituales.* [En línea] 13 de Octubre de 2011. [Citado el: 21 de enero de 2018.] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-habituales.html.
- **Bord, International Software Testing Qualifications. 2010.** *Programa de estudio nivel básico.* s.l. : International Software Testing Qualifications Bord. Versión 2010, 2010.
- **Centro de Soporte, UCI. 2018.** Centro de Soporte UCI. *Tipos de soporte.* [En línea] 2018. [Citado el: 4 de diciembre de 2018.] https://soporte.uci.cu/sites/default/files/documentacion/Tipos%20de%20soporte.pdf.
- **De Jesús Reyes, Jenny Esther. 2013.** [En línea] 6 de Febrero de 2013. http://www.eoi.es/blogs/mtelcon/2013/02/06/las-tics-y-la-gestion-empresarial/.
- **Doctrine-project.org. 2018.** Doctrine project. *What is Doctrine?* [En línea] 2018. [Citado el: 20 de enero de 2018.] http://docs.doctrine-project.org/en/latest/tutorials/getting-started.html.
- **Dudler, Roger. 2017.** DISQUS. *Git - la guía sencilla.* [En línea] 2017. http://rogerdudler.github.io/git-guide/index.es.html.
- **Eguiluz, Javier. 2012.** *Desarrollo web ágil con Symfony2.* 2012.
- **Empresariales, Intituto Europeo de Estudios. 2016.** revistadigital.inesem.es. [En línea] Octubre de 2016. [Citado el: 10 de Febrero de 2018.] https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.

- **Gamma, Erich. Helm, Richard. Johnson, Ralph. Vlissides, John. 1995.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995. ISBN: 0-201-63361-2.
- **García, Danilo Rivero. 2008.** *Ley de Procedimiento Penal. Articulo 1.* Ciudad de la Habana : Ediciones ONBC, 2008. ISBN 959-7066-43-9.
- **Garzás, Javier. 2014.** javiergarzas.com. [En línea] 5 de Agosto de 2014. [Citado el: 3 de Febrero de 2018.] http://www.javiergarzas.com/2014/08/los-patrones-grasp.html.
- **González Ochoa, Darián y Domínguez González, Yosviel. 2018.** *SITPC, una herramienta informática cubana para la administración de la justicia.* La Habana : s.n., 2018.
- **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar. 2010.** *Metodología de la investigación.* México : McGRAW-HILL, 2010. ISBN: 978-607-15-0291-9.
- **Hernández, Luis del Valle. 2018.** programarfacil.com. *¿Qué es un ORM?* [En línea] 2018. [Citado el: 19 de enero de 2018.] https://programarfacil.com/blog/que-es-un-orm/.
- **Hernando Recaman Chaux, Carlos Andrés Guerrero Alarcón. 2012.** *Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias.* Bogotá, D.C. – Colombia : Universidad Cooperativa de Colombia, 2012. Vol. 10. 2382-4239.
- **Ibrugor. 2014.** ibrugor. [En línea] 11 de junio de 2014. [Citado el: 20 de febrero de 2018.] http://www.ibrugor.com/blog/apache-http-server-que-es-como-funciona-y-para-que-sirve/.
- **Ingeniería, Academia de Madrid.** Academia Cartagena 99. [En línea] [Citado el: 25 de febrero de 2018.] http://www.cartagena99.com/recursos/alumnos/apuntes/Patrones%20de%20Diseno.pdf.
- **Jqueryui.com. 2018.** Jquery. User Interface. [En línea] 2018. [Citado el: 28 de Abril de 2018.] https://jqueryui.com/.
- **Latinoamericana, Red Gráfica. 2018.** redgrafica.com. [En línea] 2018. [Citado el: 5 de Febrero de 2018.] http://redgrafica.com/El-lenguaje-de-programacion-PHP.
- **Lucidchart. 2018.** Lucidchart. *Lucid Software Inc.* [En línea] 2018. [Citado el: 5 de enero de 2018.] https://www.lucidchart.com/pages/es/qu%C3%A9-es-un-modelo-de-base-de-datos.
- **Microsoft. 2017.** Microsoft Developer Network. *¿Qué es la arquitectura de software?* [En línea] 2017. [Citado el: 18 de enero de 2018.] https://msdn.microsoft.com/es-es/hh144976.aspx.

- **NetBeans.org. 2018.** NetBeans IDE. [En línea] 2018. [Citado el: 25 de febrero de 2018.] https://netbeans.org/kb/docs/ide/git.html.
- **Next_U. 2018.** Next_U. [En línea] 2018. [Citado el: 28 de Abril de 2018.] https://www.nextu.com/blog/conoce-las-ventajas-y-desventajas-de-javascript/.
- **Oracle. 2018.** netbeans.org. *NetBeans IDE Features.* [En línea] 2018. [Citado el: 20 de febrero de 2018.] https://netbeans.org/features/index.html.
- **POSSO, ALCIDES NEPTALÍ RIVERA. 2010.** Repositorio Digital Universidad Técnica del Norte. [En línea] Diciembre de 2010. [Citado el: 1 de Febrero de 2018.] http://repositorio.utn.edu.ec/bitstream/123456789/988/2/04%20ISC%20164%20Informe%20Tecnico.pdf.
- **Pressman, Roger S. 2010.** *La Ingeniería del Software, un enfoque práctico.* México : s.n., 2010. ISBN: 978-607-15-0314-5.
- **Programción.NET. 2017.** Programción.NET. *¿Qué es ORM?* [En línea] 2017. http://www.tuprogramacion.com/glosario/que-es-un-orm/.
- **Rodríguez, Txema. 2012.** GENBETA:dev. *Bootstrap from Twitter.* [En línea] 16 de junio de 2012. [Citado el: 16 de enero de 2018.] http://www.genbetadev.com/frameworks/bootstrap.
- **RUP. 2006.** RUP.es. [En línea] 2006. [Citado el: 5 de febrero de 2018.] https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/guidances/concepts/change_request_management_119F2E47.html.
- **Sánchez, Tamara Rodríguez. 2014.** *PROGRAMA DE MEJORA. Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2014. pág. 6.
- **SensioLabsWorld. 2018.** Twig. *The flexible, fast, and secure template engine for PHP.* [En línea] 2018. [Citado el: 19 de enero de 2018.] https://twig.symfony.com/doc/2.x/intro.html.
- **SoftDoit, S.L. 2018.** SoftDoit. [En línea] 2018. [Citado el: 26 de enero de 2018.] https://www.softwaredoit.es/definicion/definicion-lenguaje-de-programacion.html.
- **SparxSystems. 2018.** Sparx Systems. [En línea] 2018. [Citado el: 10 de abril de 2018.] http://www.sparxsystems.com.ar/.

- **Urquiza, Henry Joe Wong. 2017.** Repositorio Universidad Continental. [En línea] mayo de 2017. [Citado el: 21 de enero de 2018.] http://repositorio.continental.edu.pe/bitstream/continental/4290/1/DC_FIN_103_MAI_UC0688_2018. ISBN electrónico n.° 978-612-4196-.
- **Valderrama, Johan Sebastián. 2016.** MindMeister. *Herramientas CASE.* [En línea] 17 de agosto de 2016. [Citado el: 22 de enero de 2018.] https://www.mindmeister.com/es/742289673/herramientas-case.
- **VisualParadigm. 2014.** www.ie.inf.uc3m.es. [En línea] 2014. [Citado el: 25 de enero de 2018.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.
- **Znet. 2016.** Znet. *¿Qué es el soporte informático?* [En línea] Znet, Noviembre de 2016. [Citado el: 3 de diciembre de 2017.] https://www.znet.com.ar/blog/2016/11/que-es-el-soporte-tecnico-nformatico/. 2.