UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Contaduría y Administración

Facultad de Ciencias Químicas e Ingeniería

Maestría en Tecnologías de la Información y la Comunicación

Desarrollo Backend del Sistema de Gestión Académica de la Facultad de Contaduría y Administración (SIGAF)

TESIS

PARA OBTENER EL GRADO DE

MAESTRÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN

Presenta:

Edgar Iván Avila Garrido

Bajo la dirección de:

Dra. Esperanza Manrique Rojas

Codirección:

Dra. Margarita Ramírez Ramírez

Tijuana, Baja California, México

Octubre 2015

Dedico esta obra a Dios que no ha escatimado

en darme todo lo necesario para

concluir este reto en mi vida.

Agradecimientos

Deseo agradecer la presente obra a todos mis maestros de la MTIC, sin ellos no habría concluido esta redacción, a mis padres por el apoyo, amor y sacrificio brindado desde que abrí los ojos en esta tierra, a mi novia Cynthia Duarte por estar a mi lado y apoyarme en los momentos de dificultad y por último, al sistema SIGAF porque a pesar de ser un programa me dio muchos dolores de cabeza y alegrías al desarrollarlo.

Resumen

La presente obra ilustra el desarrollo de los módulos base de un sistema de gestión académica para la Facultad de Contaduría y Administración o por sus siglas SIGAF, a través de una arquitectura de desarrollo Modelo-Vista-Controlador y obedeciendo las mejores prácticas, metodologías y técnicas del desarrollo de software. Todo envuelto y construido sobre tecnologías web modernas como PHP, Javascript, jQuery, Ajax, HTML5 y CSS3, utilizando marcos de trabajo como Laravel con la finalidad de realizar código sustentable y mantenible para un proyecto de desarrollo sustentable y continúo.

En el primer capítulo se describen las causas y la problemática que dieron origen a la creación del sistema SIGAF, posteriormente se da el fundamento teórico que rodea al desarrollo de las aplicaciones web y tecnologías con las que se construye el sistema. Una vez terminada la explicación del fundamento el siguiente capítulo se centra en la metodología que sirve de guía para el desarrollo del proyecto, posteriormente a la metodología, se documenta la aplicación de la misma en el desarrollo y por último se concluye con las experiencias y recomendaciones del desarrollo del mismo.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo I

Introducción

Capítulo I - Introducción

1.1 Antecedentes

Actualmente en la Facultad de Contaduría y Administración de la Universidad Autónoma de Baja California, campus Tijuana se imparten 4 Licenciaturas, se ofertan cerca de 240 asignaturas, cuenta con un plantel de 300 a 350 docentes, y por lo regular se generan cerca de 120 grupos por cada semestre del plan de estudios en curso. Todos estos datos son organizados y conjugados por el personal administrativo para crear los horarios de cada grupo de la Facultad de Contaduría y Administración (FCA). Los horarios luego son ofertados al alumnado vigente. Este proceso de obtención de horarios es realizado durante los últimos meses de cada semestre y es organizado por un sistema de información local.

El sistema de horarios actual de la Facultad de Contaduría y Administración, en el cual se apoya el personal administrativo para la generación de horarios, fue desarrollado por un estudiante de la Licenciatura de Informática como un proyecto para la FCA en el año 1998, el sistema fue desarrollado en el lenguaje Delphi y se conecta a un fichero de datos InterBase, estos elementos se unieron en una aplicación de escritorio con el que actualmente trabaja el personal administrativo. El sistema posee entre sus virtudes, la capacidad de generar los grupos de horarios en base a la disponibilidad de los docentes, estas decisiones son tomadas al momento por el personal analista mediante el despliegue de la información en diversas ventanas, el personal elige, captura los grupos, el docente, las materias que se imparten, para enseguida generar el horario. Una vez introducida está información se crea un reporte de los horarios de cada grupo generado por el personal. Actualmente se ha detenido el desarrollo y mantenimiento del sistema.

La demanda actual del alumnado y los planes de estudio han crecido respecto a años anteriores hace necesario que el proceso de creación y gestión de horarios sean actividades que se planeen al terminó de cada semestre aproximadamente con una duración de uno a tres meses, para que cuando se inicie un nuevo semestre, los alumnos puedan conocer y tomar decisiones en base a los horarios publicados de cada grupo por carrera de la FCA.

Algunas instituciones educativas han implementado sistemas de horarios a través de aplicaciones informáticas y han logrado de esta manera optimizar la gran mayoría o la totalidad los procesos de generación y gestión de horarios, estos sistemas reducen significativamente largos tiempos de trabajo u horas hombre invertidas en estas actividades. Instituciones educativas tanto regionales como extranjeras, han solventado esta problemática mediante la aplicación de la tecnología con el uso de los sistemas de información.

Una de estas instituciones es el Instituto Tecnológico de Tijuana implementó un sistema de horarios a través de una aplicación web, en el cual, el personal administrativo gestiona los grupos, maestros y personal docente para obtener los horarios que son puestos a disposición del alumnado que a través del mismo portal el usuario en esta caso alumno elije sus materias de acorde al semestre en curso, el plan de estudios vigente y créditos obtenidos hasta el momento, toma como parámetro el record del alumno durante el ciclo escolar actual, unos cuantos minutos después obtiene su horario impreso y queda registrado en una base de datos que retroalimenta al semestre siguiente para seguir este proceso, todo esto integrado en un solo sistema. Otro beneficio de la base de datos es que es utilizada al mismo tiempo por Control Escolar para diferentes fines y se actualiza cada período haciéndola funcional y eficiente para otros procesos de la institución (Tecnólogico Tijuana, 2015).

Otro caso de éxito, es la implementación de un sistema experto mediante la aplicación de algoritmos genéticos en la generación automática de horarios en la Facultad Regional de

Granma en Cuba, los algoritmos eligen la mejor opción en base a la denominada cruza de especies, mediante una combinación de las distintas variables involucradas del proceso, como los salones, cantidad de docentes, evaluaciones, disponibilidad de horarios y todos los aspectos que contribuyen a la realización de los horarios, una vez obtenida la cruza de estas variables se elige al candidato más apto, en este caso selecciona el horario, la materia y el docente más cualificado para impartirla. Aún sigue en uso el sistema y se actualiza constantemente (Karel Rodríguez, 2012).

El proceso de generación y gestión de horarios se remonta a la creación de las instituciones educativas, convirtiéndose en un proceso clave de logística. Todo este proceso de logística en épocas anteriores se llevó de forma manual, mediante análisis exhaustivos de los datos, cotejamiento y validación de las distintas variables presentes, y mediante la observación totalmente de una manera empírica. Con la introducción de los sistemas de información se automatizan los procesos claves en las organizaciones e instituciones al hacer más eficiente las operaciones relativas al giro en el que se desenvuelven, así la generación y gestión de los horarios tiene un área de oportunidad para adoptar este avance tecnológico, como los son los sistemas de información, en la actualidad existen soluciones informáticas y sistemas que agilizan todo este proceso de logística.

El software de generación y gestión de horarios, es un tipo de solución informática que se presenta ante la cantidad de información y variables que manejan los diferentes planteles e instituciones educativas. Empresas dedicadas al desarrollo de software, han incluido dentro de su catálogo de productos sistemas informáticos que permitan agilizar, generar, almacenar y gestionar todo este tipo de información, de manera que con poco esfuerzo por parte de los usuarios puedan realizar esta actividad, el análisis y cotejo de las variables es delegado al

poder de procesamiento de las computadoras mediante algoritmos y secuencias de programación. A continuación, se mencionan ejemplos comerciales de este tipo de software:

- aSc TimeTables: Software realizado por la empresa *aSc Applied Software Consultants,* actualmente es un software de escritorio que tiene vigencia y fue premiado por la facilidad de su interfaz que posee, es usado por cerca de 100,000 instituciones educativas y esta traducido a 7 idiomas diferentes su principal nicho de mercado es escuelas primarias y secundarias su licencia depende del plan y va desde 300 a 2000 USD (Applied Software Consultants, 2016).
- AGH/iX: Desarrollado por *Catalonia Software Solutions,* este sistema de escritorio se encarga de gestionar y generar horarios de una organización, este sistema a diferencia de los otros posee una singularidad, su funcionamiento esta modulado por lo que se puede adquirir módulos adicionales e independientes del sistema principal, también existe un módulo web para hacer el desarrollo por Internet los costos de licencia se piden por teléfono a la empresa (Catalonia Software Solutions, 2016).
- UntisExpress: Este software de escritorio fue desarrollado por la empresa *Untis Gruber & Petters* y posee algoritmos para la generación automática de la estructura de los horarios y da la posibilidad de elegir entre una gran cantidad de resultados propuestos, su licencia va de 250USD y tiene un costo de mantenimiento de 70USD (Untis Grubers and Petters , 2015).

Cabe señalar, que estos sistemas poseen documentación, soporte, manuales de usuario y por lo general son aplicaciones de escritorio, sin embargo, algunos poseen características

interesantes que son de gran utilidad en el desarrollo de esta actividad, existen pocas plataformas web y aplicaciones en dispositivos móviles para este tipo de procesos. Con los adelantos de la tecnología esto cambiará en poco tiempo, se verán soluciones en plataformas web de alta usabilidad y escalabilidad que no solo estarán enfocadas al resultado de los horarios como tal, sino que, también contarán con retroalimentación y experiencia de usuario que harán muy ameno e intuitivo realizar esta tarea de logística, recurrente en las instituciones educativas.

Anteriormente, una sola persona era encargada de desarrollar las denominadas páginas o sitios web, en la actualidad, no solo documentos interrelacionados que contienen información en texto, sino que, han evolucionado hasta convertirse en sistemas web relativamente complejos y robustos que pueden albergar una gran cantidad de contenido multimedia. Estas aplicaciones, han pasado de ser emisores de información a ser receptores para procesar y devolver al usuario un resultado en concreto, por lo cual, la interacción con el usuario se ha hecho indispensable (Kiselev, 2015).

Ahora mismo, gracias a este avance significativo, existen las denominadas redes sociales en el mundo del desarrollo web, ya no solo la interacción humano-máquina importa, sino que, se contempla la interacción entre los propios usuarios de las aplicaciones en tiempo real para socializar y comunicarse (Kiselev, 2015).

Para poder desarrollar este tipo de plataformas, es necesaria el área de la programación web, mediante la aplicación correcta de las técnicas y herramientas de la programación se puede construir la arquitectura adecuada que de soporte y estructura a los sistemas web. Debido a la especialización que requieren las aplicaciones, han surgido los profesionistas en BACKEND. En la actualidad, este tipo de profesión es la encargada de desarrollar la arquitectura de programación de cualquier aplicación web, además, el profesionista Backend necesita

interactuar con un equipo especializado en diferentes áreas del desarrollo web, como el Frontend, Base de Datos, Análisis, Pruebas y Documentadores. Una de las actividades principales de esta área es unir el modelo de negocio (base de datos) de las aplicaciones con las interfaces donde el usuario interactúa en las plataformas y sistemas web (Valdés, 2007).

Los lenguajes de programación han evolucionado y tomado un enfoque diferente, antes no existían dispositivos móviles, ahora las plataformas están encaminadas a los Smartphone y los lenguajes han modificado sus modelos para poderse adaptar a la web moderna. Han surgido nuevos estándares y tecnologías cuyo objetivo, es desarrollar aplicaciones para sistemas móviles y ya no tanto para sistemas de escritorio o inclusive web. También la programación ha cambiado sus paradigmas para realizar sistemas en tiempo real, lo que permite interacciones directas y donde la retroalimentación es vital para su funcionamiento (Valdés, 2007).

Existen lenguajes especializados para el desarrollo web, sin embargo, el solo hecho de programar en estos lenguajes, no asegura un orden o una estructura eficaz para el procesamiento de la información, para esto, existen modelos de programación probados y definidos, uno de los más utilizados en la actualidad es la Arquitectura N-Capas, este estilo de programación tiene como objetivo principal, separar los diferentes aspectos del desarrollo, tales como la presentación, lógica de negocio y mecanismos de almacenamiento. Esto surge, a raíz de la desorganización de código que existía en la programación de los sitios web, comúnmente denominado como “código espagueti”, una analogía a como se presenta el desorden en la pasta como comida.

Los lenguajes especializados para generar aplicaciones web son generalmente muy permisivos, esto crea una fácil integración con otro tipo de lenguajes, este tipo de característica permite separar una aplicación web completa en varios aspectos o puntos de vista del

desarrollo. Desde el punto de vista de la interfaz e interacción existen lenguajes como HTML, CSS, y Javascript, estos forman parte dentro desarrollo en área de denominada desarrollo Frontend que solo es visible del lado del cliente, lo que se denomina como capa de presentación, por otro lado, el área encargada de Bases de Datos permite que la información tenga permanencia en las aplicaciones, comúnmente se denomina como capa de persistencia o datos, y por último, el Backend de una aplicación, precisamente el área encargada de conectar las bases de datos con el Frontend. La aplicación de este modelo, permite tener una arquitectura robusta, separada que ayuda a la corrección y organización del código que se genera y ejecuta (Libros Web, 2009).

1.2 Definición del problema

El personal administrativo de la Facultad de Contaduría y Administración (FCA) genera los horarios de las licenciaturas en un proceso largo, tedioso y con posibles errores de consistencia (como el cruce de aulas y grupos) durante su elaboración, este proceso dura aproximadamente tres meses por semestre. La realización de esta tarea se hace de forma semi-manual mediante un sistema de información que no cumple con la funcionalidad requerida por el personal administrativo, tanto la gestión como la creación de horarios se lleva a cabo durante los últimos meses de cada semestre que contemplan el plan de estudios de la FCA, el tiempo de realización del proceso es muy largo y requiere un sobreesfuerzo por parte del personal administrativo. Este proceso es elaborado por tres personas que interactúan con coordinadores de carrera, docentes y directivos a través de documentos en papel, con el fin, de obtener la fuente de información necesaria para la creación de los horarios.

Una de las principales causas de la problemática, es el actual sistema de horarios de la FCA, contiene limitaciones en la captura, proceso y salidas de información, si bien el sistema ayuda a la automatización de este procedimiento, su funcionalidad no reduce significativamente las horas de trabajo que conlleva la elaboración y gestión de los horarios, el procesamiento es lento, laborioso y no genera toda la información necesaria por los analistas y el personal involucrado.

La cantidad de información que se ingresa en el sistema depende totalmente del llenado de formatos en papel por parte del personal involucrado, estos documentos, pueden mezclarse o perderse, lo que provoca que pueda haber perdida de información, no existe una base de datos optimizada y funcional que permita obtener reportes sobre los horarios que se generan en el actual sistema de horarios y solo existen tres personas encargadas de la recolección, análisis

y procesamiento de la información de toda la facultad, lo cual repercute en una gran cantidad de tiempo en la elaboración de los horarios.

Por último, la FCA no cuenta con el código fuente del sistema actual, no existe documentación, manuales de ningún tipo y no existe un plan de mantenimiento sobre su infraestructura esto provoca que no puedan realizarse modificaciones y hace al sistema actual de horarios obsoleto y cerrado para cualquier mejora.

1.3 Justificación

El proyecto se justifica por la necesidad de agilizar los tiempos al igual que optimizar los procesos en la generación y gestión de los horarios en la Facultad de Contaduría y Administración. A continuación se presentan los puntos generales para la viabilidad del proyecto:

- Se cuenta con la tecnología necesaria para desarrollar la plataforma.
- El costo de las herramientas no serán altos o en su defecto no existirán porque son herramientas de software libre que se pueden obtener gratuitamente desde Internet.
- El tiempo de elaboración es el suficiente para obtener la plataforma tecnológica.
- Hay un equipo de desarrollo completo que cubre cada uno de los aspectos necesarios en la construcción de la plataforma.
- La disponibilidad del personal para la obtención de los requerimientos de los procesos es aceptable para el desarrollo del proyecto.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar la arquitectura de programación y la codificación necesaria para la implementación de cada uno de los tres módulos base que contempla el sistema SIGAF (Sistema de Gestión Académica de la Facultad de Contaduría y Administración).

1.4.2 Objetivos específicos

Elaborar el Backend de los siguientes módulos del sistema SIGAF:

1.-Desarrollar código funcional para el desarrollo de los planes de estudios de la facultad que resulte a partir de la creación y gestión de las unidades de aprendizaje asignadas en un período determinado.

2.-Desarrollar código funcional que permita realizar la carga académica de cada programa educativo del ciclo escolar vigente, basándose en los planes de estudio previamente establecidos por el personal administrativo.

3.-Desarrollar código funcional que permita al personal docente de cada programa educativo capturar y actualizar su disponibilidad docente, formación académica e información relevante a la institución a lo largo de su trayectoria educativa en la facultad, la cual será esencial al momento de la asignación de docentes en la creación de horarios.

4.- Desarrollar código funcional que permita la administración y asignación de privilegios a los diferentes usuarios que podrán utilizar los módulos de la plataforma.

1.5 Alcance del proyecto

1.5.1 Alcances

Los módulos de desarrollo en primera instancia, estarán destinados y disponibles para la Facultad de Contaduría y Administración de la Universidad Autónoma de Baja California Campus Tijuana y las licenciaturas que se ofertan:

- Licenciatura en Informática.
- Licenciatura en Contaduría.
- Licenciatura en Administración de Empresas.
- Licenciatura en Negocios Internacionales.

1.5.2 Limitaciones

El sistema está modulado, por lo cual, se necesita limitar actividades en el desarrollo de cada uno de los módulos que comprende el sistema SIGAF:

- Desarrollo de la lógica de negocio acorde al análisis del sistema.
- Diagramación de flujo de la lógica de cada módulo.
- Codificación de cada módulo.
- Documentación del código de cada módulo.
- Pruebas de cada módulo.
- Verificación de la integración del Frontend con la base de datos a través del Backend.

Capítulo II

Marco teórico

Capítulo II - Marco Teórico

Sin el auge del internet el desarrollo web no hubiera tomado tanta importancia como en la actualidad, en este capítulo se estudia a grandes rasgos la historia y fundamento del desarrollo web moderno, el cuadro conceptual de la figura 1 muestra de modo sintetizado el contenido de este capítulo.

2.1 Marco conceptual del marco teórico

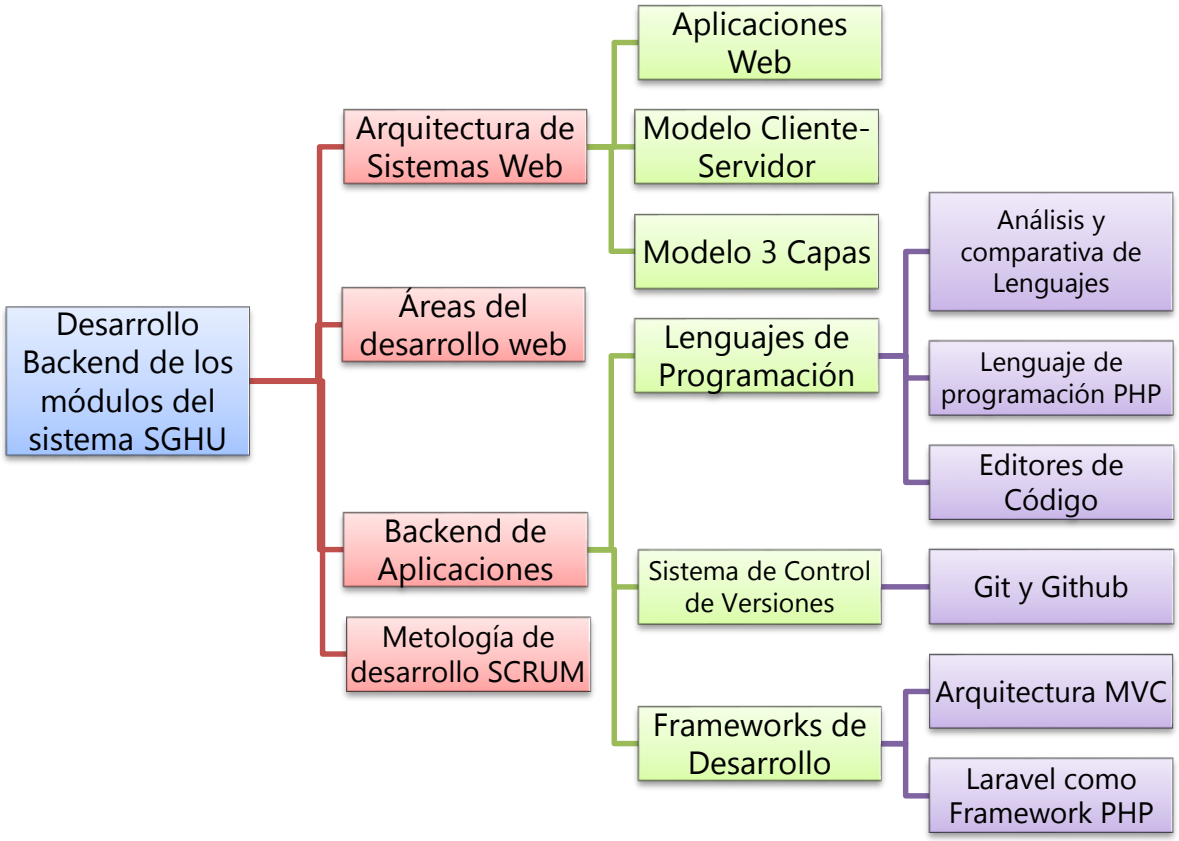


Figura 1. Diagrama marco conceptual. Fuente Propia.

2.2 Arquitectura de sistemas web

2.2.1 Aplicaciones web

Una aplicación web es un programa informático, que es accedido y ejecutado en una red como internet o intranet, mediante una interfaz comúnmente por un navegador web (Alegsa, 2013).

2.2.1.1 Definición de aplicación web

Como se define en el sitio Diccionario de informática (Alegsa, 2013) “En general, el término también se utiliza para designar aquellos programas informáticos que son ejecutados en el entorno del navegador (por ejemplo, un applet de Java) o codificado con algún lenguaje soportado por el navegador (como JavaScript, combinado con HTML); confiándose en el navegador web para que reproduzca (rende-ricé) la aplicación”.

Una de las ventajas de las aplicaciones web cargadas desde internet (u otra red), es la facilidad de mantener y actualizar, sin la necesidad de distribuir e instalar un software en potencialmente miles de clientes. También tiene como ventaja la posibilidad de ser ejecutadas en múltiples plataformas.

Las aplicaciones web son utilizadas para implementar webmail, ventas online, subastas online, wikis, foros de discusión, weblogs, MMORPGs, redes sociales, juegos, portales de gobierno.

Ejemplos de aplicaciones web: Instagram, Google, Facebook, Twitter, Github.

2.2.1.2 Características de las aplicaciones web

Las aplicaciones web a diferencia de otro tipo de programa tienen la parte del procesamiento comúnmente en un solo equipo denominado servidor, la infraestructura que soporta a la

aplicación cambia la forma tradicional en que se desarrolla y mantienen los programas, a continuación se listan las características de este tipo de programa según (Alegsa, 2013):

- El usuario puede acceder fácilmente a estas aplicaciones mediante un navegador web (cliente) o similar.
- Si es por internet, el usuario puede entrar desde cualquier lugar del mundo donde tenga un acceso a internet.
- Pueden existir miles de usuarios pero una única aplicación instalada en un servidor, por lo tanto se puede actualizar y mantener una única aplicación y todos sus usuarios verán los resultados inmediatamente.
- Emplean tecnologías como Java, JavaFX, JavaScript, DHTML, Flash, Node.js, Ajax... que dan gran potencia a la interfaz de usuario.
- Emplean tecnologías que permiten una gran portabilidad entre diferentes plataformas. Por ejemplo, una aplicación web flash podría ejecutarse en un dispositivo móvil, en una computadora con Windows, Linux u otro sistema, en una consola de videojuegos, etc.

2.2.1.3 Interfaz gráfica de las aplicaciones web

La interfaz gráfica de una aplicación web puede ser compleja y funcional, gracias a las variadas tecnologías web que existen: Java, JavaScript, DHTML, Flash, Silverlight, Ajax, entre otras. Prácticamente no hay limitaciones, las aplicaciones web pueden hacer casi todo lo que está disponible para aplicaciones tradicionales: acceder al mouse, al teclado, ejecutar audio o video, mostrar animaciones, soporte para arrastrar y soltar, y otros tipos de tecnologías de interacción usuario-aplicación (Lapuente, 2013).

Ajax es un ejemplo de una tecnología de desarrollo web que le da gran poder de interactividad a las aplicaciones web.

2.2.2 Arquitectura cliente – servidor

Según (Orfali, 1999) “Cliente / Servidor es la palabra más de moda en la industria de la computación, no existe consenso acerca de cuál es, en realidad, su significado. Por tanto, se tiene una gran oportunidad de crear una propia definición. Como lo indica su nombre, clientes y servidores son entidades lógicas autónomas que trabajan juntas en una red para cumplir una tarea.” Bien pero ¿que hace que la tecnología cliente/servidor sea diferente de otros tipos de software distribuido? El siguiente planteamiento es que todo sistema cliente /servidor se distingue por las siguientes características según (Delgado, 2011):

- Servicio: La arquitectura cliente/servidor es, ante todo, una relación entre procesos que se ejecutan en máquinas independientes una de la otra. El proceso servidor es un proveedor de servicios; el cliente los consume. En esencia, la tecnología cliente/servidor provee una clara separación de funciones con base en la idea de servicio.
- Recursos compartidos: Un servidor puede servir a varios clientes al mismo tiempo y regular su acceso a recursos compartidos.
- Protocolos asimétricos: Existe una relación de muchos a uno entre varios clientes y un servidor. Los clientes siempre empiezan el diálogo al solicitar un servicio; los servidores esperan de modo pasivo a que les lleguen solicitudes de los clientes. Obsérvese que, en algunos casos, un cliente podría pasar una referencia de retro llamada (callback) a un objeto cuando solicita un servicio, lo cual permite al servidor devolver la llamada (call back) al cliente. Así, el cliente se convierte en servidor.

- Transparencia de ubicación: Un servidor es un proceso que puede residir en la misma máquina que el cliente, o en otra, en la red. Normalmente, el software cliente/servidor oculta a los clientes la ubicación del servidor re-direccionando las solicitudes de servicio que les son requeridas. Un programa puede ser cliente, servidor, o las dos cosas.
- Mezclar y acoplar: El software cliente/servidor ideal es independiente de plataformas de equipo o de sistemas operativos. Siempre debe ser posible “mezclar y acoplar” plataformas de clientes y servidores.
- Intercambios basados en mensajes: Clientes y servidores son sistemas acoplados sin grandes restricciones que interactúan mediante un mecanismo de intercambio de mensajes; así, éstos se convierten en el mecanismo de entrega para las solicitudes y respuestas de servicio.
- Encapsulado de servicios: El servidor es un “especialista”. A través de un mensaje se le indica cuál es el servicio que se le solicita, y luego depende de él la forma en que se satisface tal solicitud. Los servidores pueden actualizarse sin afectar a los clientes siempre y cuando la interfaz de mensajes publicada no cambie.
- Escalabilidad: Los sistemas cliente/servidor pueden escalarse horizontal o verticalmente. El escalamiento horizontal implica que al agregar o quitar estaciones de trabajo clientes sólo se produce un pequeño efecto en el desempeño. El escalamiento vertical significa migrar (mudar) a una máquina servidor más grande y rápido, o distribuir la carga de procesamiento entre varios servidores.
- Integridad: El código y la información del servidor se administran de manera central, lo que da como resultado un mantenimiento más barato y el resguardo de la integridad de

la información compartida. Al mismo tiempo, los clientes permanecen personales e independientes.

Las características de la tecnología cliente/servidor arriba descritas permiten distribuir sin problemas la inteligencia a lo largo de la red. Asimismo, brindan el marco para diseñar aplicaciones de red acopladas sin grandes restricciones (Delgado, 2011).

2.2.3 Modelo tres capas

Las aplicaciones web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dan servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. Esto ha exigido reflexiones sobre la mejor arquitectura y las técnicas de diseño más adecuadas (Garrido J. S., 2004).

En los últimos años, la rápida expansión de Internet y del uso de intranets corporativas ha supuesto una transformación en las necesidades de información de las organizaciones. En particular esto afecta a la necesidad de que:

1. La información sea accesible desde cualquier lugar dentro de la organización e incluso desde el exterior.
2. Esta información sea compartida entre todas las partes interesadas, de manera que todas tengan acceso a la información completa (o a aquella parte que les corresponda según su función) en cada momento.

Estas necesidades han provocado un movimiento creciente de cambio de las aplicaciones tradicionales de escritorio hacia las aplicaciones web, que por su idiosincrasia, cumplen a la perfección con las necesidades mencionadas anteriormente. Por tanto, los sitios web tradicionales que se limitaban a mostrar información se han convertido en aplicaciones

capaces de una interacción más o menos sofisticada con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar opciones de diseño nuevas que permitan dar con la arquitectura óptima que facilite la construcción de los mismos (Paredes, 2011).

El usuario interacciona con las aplicaciones web a través del navegador. Como consecuencia de la actividad del usuario, se envían peticiones al servidor, donde se aloja la aplicación y que normalmente hace uso de una base de datos que almacena toda la información relacionada con la misma. El servidor procesa la petición y devuelve la respuesta al navegador que la presenta al usuario. Por tanto, el sistema se distribuye en tres componentes: el navegador, que presenta la interfaz al usuario; la aplicación, que se encarga de realizar las operaciones necesarias según las acciones llevadas a cabo por éste y la base de datos, donde la información relacionada con la aplicación se hace persistente. Esta distribución se conoce como el modelo o arquitectura de tres capas (figura 2).

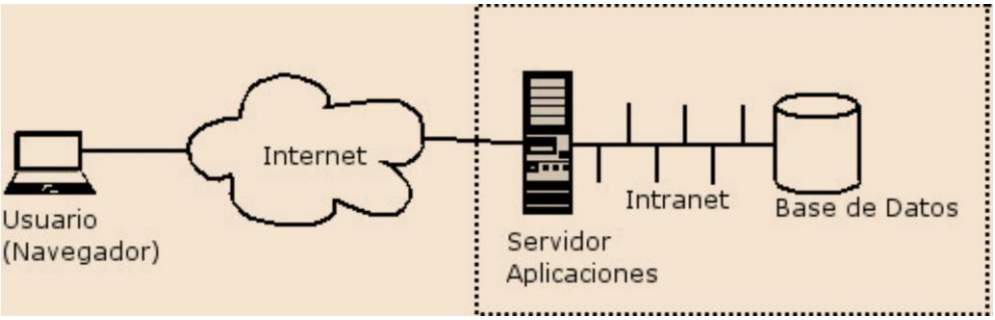


Figura 2. Arquitectura 3 Capas. Fuente (Paredes, 2011).

En la mayoría de los casos, el navegador es un presentador de información (modelo de cliente delgado), y no lleva a cabo ningún procesamiento relacionado con la lógica de negocio. No obstante, con la utilización de applets, código javascript y DHTML la mayoría de los sistemas se sitúan en un punto intermedio entre un modelo de cliente delgado y un modelo de cliente grueso (donde el cliente realiza el procesamiento de la información y el servidor sólo es

responsable de la administración de datos). No obstante el procesamiento realizado en el cliente suele estar relacionado con aspectos de la interfaz (como ocultar o mostrar secciones de la página en función de determinados eventos) y nunca con la lógica de negocio (Abc, 2011).

En todos los sistemas de este tipo y ortogonalmente a cada una de las capas de despliegue comentadas, al realizar esta acción la aplicación se divide en tres áreas o niveles según (Garrido, 2004):

1. Nivel de presentación: es el encargado de generar la interfaz de usuario en función de las acciones llevadas a cabo por el mismo.
2. Nivel de negocio: contiene toda la lógica que modela los procesos de negocio y es donde se realiza todo el procesamiento necesario para atender a las peticiones del usuario.
3. Nivel de administración de datos: encargado de hacer persistente toda la información, suministra y almacena información para el nivel de negocio.

Los dos primeros y una parte del tercero (el código encargado de las actualizaciones y consultas), suelen estar en el servidor, mientras que la parte restante del tercer nivel se sitúa en la base de datos (notar que, debido al uso de procedimientos almacenados en la base de datos, una parte del segundo nivel también puede encontrarse en la misma). A partir de estas características en la arquitectura de los sistemas web, a continuación se listan algunos patrones de diseño de aplicación básica que pueden facilitar un diseño apropiado para este tipo de sistemas (Garrido , 2004).

2.3 Áreas del desarrollo web

En épocas anteriores el profesionista denominado Webmaster era el encargado de diseñar, desarrollar e implementar un sitio web en su totalidad, sin embargo, los sitios web de épocas anteriores, no tenían la complejidad ni la interacción con el usuario que se ha desarrollado en la actualidad, fue necesario la creación de varias áreas para cubrir cada una de las necesidades de las aplicaciones actuales y portales web (Vega, 2013).

Actualmente el desarrollo web puede clasificarse en las siguientes áreas:

- Interfaces de usuario (GUI´s).
- Lógica de Programación.
- Diseño Gráfico.
- UX – User Experience.
- Usabilidad.
- Posicionamiento.
- Estrategia.
- Arquitectura de información.

Estas áreas forman parte en la actualidad en un proyecto de desarrollo web, se engloba fácilmente en dos especializaciones según (Vega, 2011) en su artículo publicado en Internet denominado “¿Qué significa Backend y Frontend en el diseño web?”, “Ahora es imposible crear un producto completo sin por lo menos un diseñador, un frontend y un backend”. Podemos concluir que en la actualidad para afrontar el desarrollo de un proyecto web necesitamos dos áreas un Backend y un Frontend (figura 3).

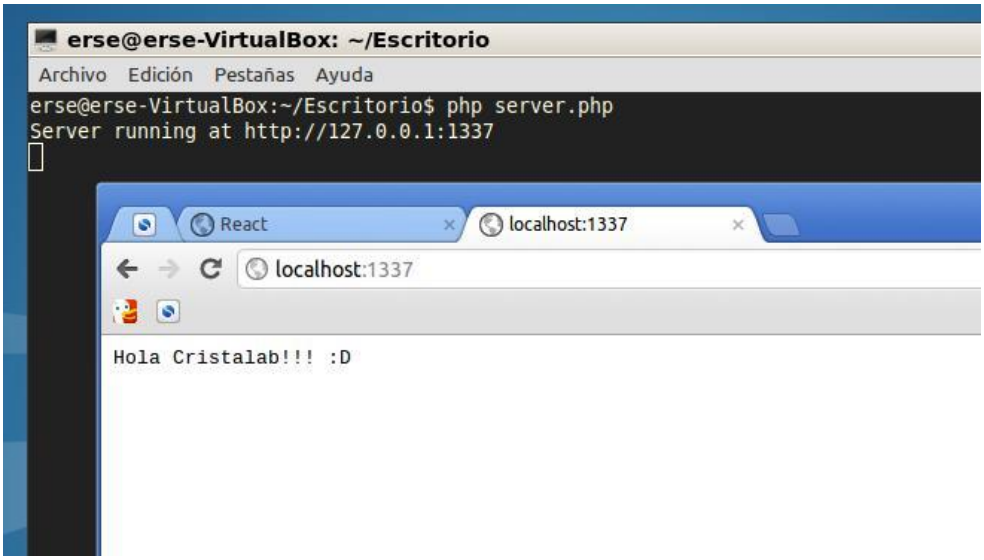
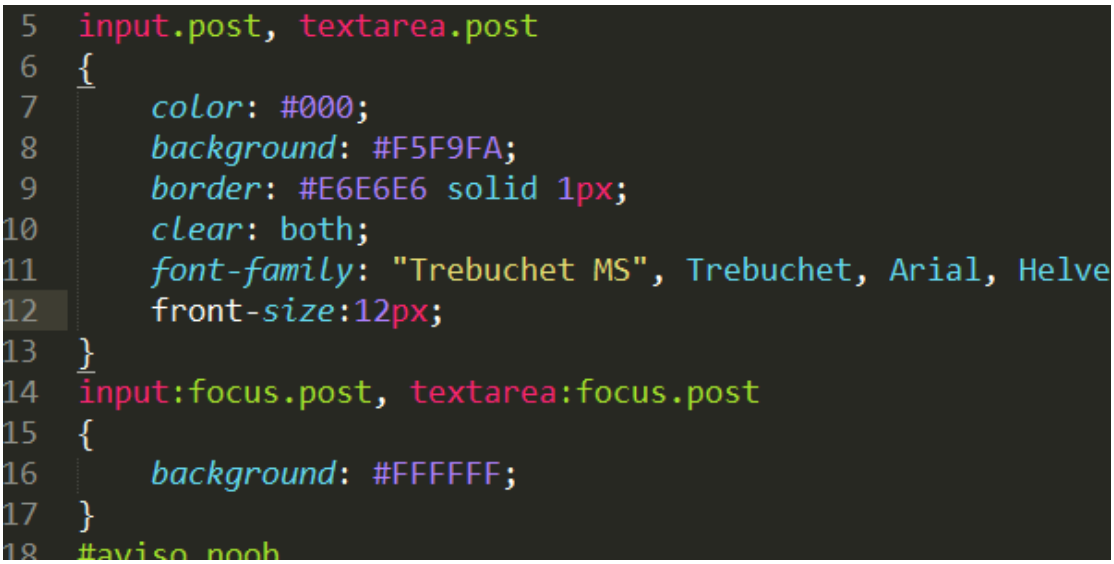


Figura 3. Representación de Backend y Frontend. Fuente (Vega, Cristalab)

2.3.1 ¿Qué es un Backend?

El Backend de una aplicación se encarga de implementar en el desarrollo de aplicaciones web bases de datos como MySQL, PostgreSQL, SQL Server o MongoDB. Se implementan lenguajes de programación como PHP o JSP, o frameworks como RoR, Django, Node.JS o .NET que se conectan a la base de datos. A través de estos lenguajes y frameworks se recibe, procesa y envía información al navegador del usuario (Vega, 2013).

En código HTML (que desarrolla un frontend) o con él envió de datos puros en XML, RSS o JSON, para ser procesados por Javascript. En Facebook, por ejemplo, PHP manda la estructura básica del sitio web, pero son múltiples programas y servidores hechos en C++ o Erlang los que procesan la información en tiempo real (como chat, comentarios, notificaciones) y las envían y reciben a través de Javascript en el navegador (figura 4).



```
5  input.post, textarea.post
6  {
7      color: #000;
8      background: #F5F9FA;
9      border: #E6E6E6 solid 1px;
10     clear: both;
11     font-family: "Trebuchet MS", Trebuchet, Arial, Helve
12     front-size:12px;
13 }
14 input:focus.post, textarea:focus.post
15 {
16     background: #FFFFFF;
17 }
18 #aviso.noob
```

Figura 4. Codificación en Backend. Fuente (Vega, Cristalab)

2.3.2 ¿Qué es Frontend?

El Frontend es el área Web que trabaja con el diseño, visualización y dinamismo del usuario, a través del lado del cliente o navegador. Básicamente, todo lo que el usuario puede ver e interactuar en un sitio o aplicación.

De acuerdo a Nate Koechley, 2009 Senior Frontend de Yahoo! : "Los profesionales Frontend dan a los sitios fuerza y resiliencia, apariencia y forma, funcionalidad e interactividad. Esta área es crítica para el éxito del proyecto web."

Los objetivos Frontend están enfocados sobre:

Implementar tecnologías para desarrollo web como HTML, CSS y Javascript. Maquetar la estructura semántica del contenido, codificar el diseño en hojas de estilo y agregar la interacción con el usuario. El Frontend necesita enfocar su esfuerzo en los siguientes aspectos del desarrollo web (figura 5):

- Crea adaptación y compatibilidad con navegadores y dispositivos.
- Accesibilidad.

- Optimización y Ejecución
- Diseño Visual
- Comunicación y Construcción de Procesos

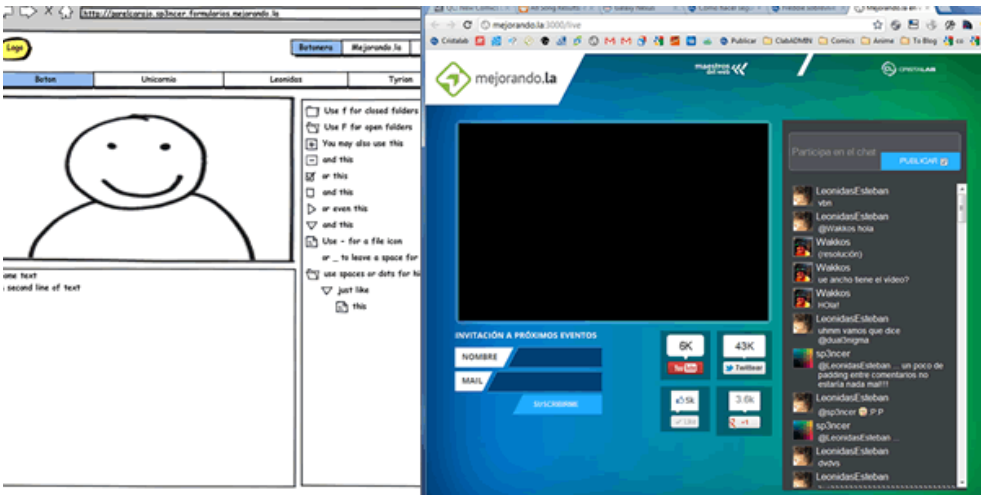


Figura 5. Codificación de Frontend. Fuente (Vega, Cristalab)

2.4 Backend de aplicaciones

2.4.1 Definición de Backend

Una definición puntual del término, es el área que soluciona procesos de información, fluidez de datos, arquitectura interna y usualmente está basado en servidores remotos (Vega, 2013). Se considera invisible a la vista de los usuarios. Hablar de Backend requiere hablar de lenguajes como Python o Ruby porque pueden trabajar Web ágilmente a través de frameworks o entornos de desarrollo como: Ruby on Rails, NodeJS, Django, Wordpress, etc.

Regularmente el trabajo en Backend se divide en tres partes:

- Un servidor.
- Una aplicación.
- Una base de datos.

2.4.2 El profesionista Backend

Según el artículo publicado en Maestros del Web los profesionales Backend “son los autores de la ejecución y lógica de las aplicaciones web” (Herts, 2011).

Los profesionales Backend entienden que hay diferentes formas de resolver un problema y que cada solución puede encontrarse en:

- Patrones de acceso a información.
- Características de velocidad.
- Operaciones sintetizadas.
- Potencial de automatización.
- Diseño de métodos para escalar información más allá de la capacidad del hardware.

Un profesionista de Backend sabe hablar con claridad y explicar los problemas para todo su equipo de manera simple al integrar una aplicación. Finalmente es importante recordar que:

- Escribir código que resuelva un problema es importante.
- Escribir código que resuelva un problema y maneje condiciones para los diferentes errores es más importante.
- Escribir código que resuelva problemas, maneje condiciones para múltiples peticiones simultáneamente sin errores es mucho más importante.
- Escribir código que haga todo lo anterior, al asegurar los segundos de entrega, es primordial.

Los profesionistas Backend son talentos muy necesitados en la Web profesional y es parte fundamental de un equipo de desarrollo del mundo web moderno (Herts, 2011).

2.4.3 Lenguajes de programación web, análisis y comparativa

2.4.3.1 Ranking de lenguajes

Antes de seleccionar un lenguaje de programación, visualizar un ranking general de los lenguajes más populares o usados a nivel mundial es importante, para tener un panorama más amplio sobre el uso y la posible comunidad que encontraremos, que gira entorno a estos y cómo podemos hacer uso de ellos.

El Índice TIOBE recoge el ranking de los lenguajes de programación más usados, en función de los ingenieros informáticos cualificados de todo el mundo que lo utilizan, cursos y proveedores de terceros. Todo ello, a través de motores de búsqueda como Google, Bing, Yahoo, Wikipedia, Amazon, Youtube y Baidu.

El objetivo es ofrecer una referencia para comprobar como de actualizados están nuestros conocimientos en cuanto a lenguajes de programación se refiere o en cuanto a la hora de decidir cuál aprender o adoptar. En las primeras posiciones se puede encontrar los lenguajes más populares y con gran variedad de aplicaciones alrededor del mundo, comunidades, búsquedas, etc. Podemos visualizar de igual manera como se encontraban rankeados el año anterior para hacer una comparativa y de la misma forma si suben o bajan de posición de un mes a otro.

Existe un portal donde se lleva estadísticamente datos relevantes de los lenguajes de programación utilizados para desarrollar (Tiobe Software, 2015) en él se pueden localizar varias gráficas que permiten visualizar las posiciones de los lenguajes respecto a varios factores como, popularidad, uso, estilos de programación y características de los mismos. Si se requiere un análisis más minucioso se realiza un análisis más profundo de una gran cantidad de información de este sitio.

En la tabla 1 se muestra un top de los 20 lenguajes más utilizados según el índice (Tiobe Software, 2015) se observa claramente la posición de los lenguajes utilizados en la actualidad.

Tabla 1. Índice TIOBE Agosto 2015. Fuente: www.tiobe.com.

2015	2014	Cambio	Lenguaje	Rango	Cambio
1	2	^	Java	19.274%	+4.29%
2	1	v	C	14.732%	-1.67%
3	4	^	C++	7.735%	+3.04%
4	6	^	C#	4.837%	+1.43%
5	7	^	Python	4.066%	+0.95%
6	3	v	Objective-C	3.195%	-6.36%
7	8	^	PHP	2.729%	-0.14%
8	12	^^	Visual Basic .NET	2.708%	+1.40%
9	10	^	JavaScript	2.162%	-0.01%
10	9	v	Perl	2.118%	-0.10%
11	11		Visual Basic	1.781%	-0.23%
12	24	^^	Assembly language	1.760%	+1.11%
13	13		Ruby	1.416%	+0.17%
14	18	^^	Delphi/Object Pascal	1.407%	+0.49%
15	21	^^	MATLAB	1.232%	+0.50%
16	14	v	F#	1.232%	+0.14%
17	23	^^	Swift	1.179%	+0.51%
18	15	v	Pascal	1.138%	+0.09%
19	20	^	PL/SQL	1.137%	+0.35%
20	30	^^	R	1.010%	+0.49%

A partir de la tabla 1 se observa que los lenguajes de .NET han ganado gran popularidad posiciona a dos de sus lenguajes en los primeros puestos C#, Visual Basic .NET. De igual manera se demuestra que los lenguajes más usados son C, Java y Objetive-C (Aplicaciones nativas Iphone). También aparecen lenguajes modernos para la web como Python y Ruby,

estos se localizan en buenas posiciones a pesar de que el lenguaje Ruby bajo en el ranking en comparación del año anterior.

Ahora bien, se recoge un histórico de estos mismos lenguajes de 5,10 y 15 años para poder resolver la estabilidad de los mismos en el paso del tiempo.

Tabla 2. Historia de largo plazo. Fuente: www.tiobe.com.

Lenguaje de programación	Posición Nov. 2012	Posición Nov. 2007	Posición Nov. 1997	Posición Nov. 1987
C	1	2	1	1
Java	2	1	4	-
Objective-C	3	38	-	-
C++	4	3	2	6
C#	5	7	-	-
PHP	6	5	-	-
(Visual) Basic	7	4	5	5
Python	8	6	29	-
Transact-SQL	9	45	-	-
JavaScript	10	10	14	-
Lisp	15	21	17	3
COBOL	19	17	3	12
Ada	22	23	18	2

Mediante esta tabla se observa claramente que los primeros lugares a lo largo de la historia ha sido constantes en uso para desarrollo, C, Java y como novedad Objetive-C, sin embargo, la tabla 2 muestra cuanto han escalonado de posición, lenguajes .NET, Python y Javascript, lenguajes propios y nativos de la web, también en la tabla 2 se observa el desarrollo PHP que tiene una gran comunidad y muchas aplicaciones web han sido desarrolladas en el entorno de este lenguaje.

Basados en estas estadísticas de tabla 2, se proponen los siguientes lenguajes que claramente marcan la diferencia respecto a otros en cuanto a desarrollo de aplicaciones web e intranets se refiere (Tiobe Software, 2015):

1. Java: Es un candidato muy fuerte aunque su desarrollo no gira su entorno totalmente a las aplicaciones web, podemos considerarlo un lenguaje muy fuerte por su larga historia.
2. .NET: Claramente los lenguajes ASP.NET (Aplicaciones Web), C#, Visual Basic .NET y Transact-SQL (Manejador de base de datos) tienen muchos recursos y una gran comunidad de desarrollo y han subido de ranking súbitamente en estos años.
3. PHP: A pesar que en noviembre de este año tuvo una baja podemos considerarlo uno de los candidatos más fuertes localizado en el puesto seis de la lista.
4. Python: Un lenguaje muy estable se mantiene, en los primeros puestos durante varios meses sin variar, actualmente se localiza en la posición ocho del ranking.
5. Javascript: En los últimos meses ha tomado por sorpresa a las comunidades de desarrollo ya que podemos en esta época desarrollar cualquier tipo de aplicaciones con este lenguaje a través de la tecnología NODE.js.

Al realizar una síntesis de la tabla general se presenta un claro esquema de los lenguajes a utilizar en la tabla 3 se muestra los lenguajes más populares en la actualidad.

Tabla 3. Lenguajes propuestos según ranking Tiobe. Fuente: www.tiobe.com.

LENGUAJE	POSICIÓN
Java	Primero
C# - ASP.NET	Cuarto
PHP	Septimo
Python	Quinto
Javascript	Noveno

Como conclusión la tabla 3 muestra que el lenguaje Java es el mejor candidato para el desarrollo, sin embargo, existen otros factores a considerar como costos, requerimientos y performance. No solo la popularidad es importante, hay que analizar dentro de estos lenguajes qué tanto rendimiento se puede obtener al ser utilizados en el desarrollo de aplicaciones.

2.4.3.2 Características generales

Cada lenguaje posee especificaciones diferentes y requerimientos tanto de plataforma como de hardware necesario para su implementación, es necesario conciliar todos estos datos en una tabla comparativa que nos permita visualizar la viabilidad del lenguaje respecto a recursos y requerimientos para su implementación.

En la tabla 4 se presenta un análisis más detallado de distintas fuentes que permiten visualizar más detalle de estos lenguajes y que se necesita para poder utilizarlos y desarrollar de la forma adecuada.

Tabla 4. Tabla Comparativa de lenguajes de programación. Fuentes: Diversas.

Concepto	ASP .NET	PHP	Java	Python	Ruby
Costo de servidor	Alto	Gratuito	Gratuito	Gratuito	Gratuito
Sintaxis de lenguaje base	VB y C#	C / C++	C/ C++	C/ C++	Perl, Smalltalk, Eiffel, Ada, y Lisp
Orientado a objetos	Si	No completamente	Si	Si	Si
Sistemas operativos	Windows y Linux (solo C# con Mono)	Linux o Windows	Linux o Windows	Linux o Windows	Linux o Windows
Servidor	IIS o Mono	Apache, compilador propio	Apache, Tomcat y Glassfish	Apache, compilador propio	Apache, compilador propio
Empresa	Microsoft y Xamarin (para Mono)	The PHP Group (open source)	Oracle (open source)	Python software foundation (open source)	Grupo Ruby (open source)
Base de datos (principalmente)	MsSQLServer	Mysql	Oracle, mysql	Mysql y PostgreSQL	Mysql y PostgreSQL

Rapidez de ejecución Generación de página web.	3er lugar	4to lugar	Último lugar	1er lugar	2do lugar
Propósito	Generar dinámicamente páginas web	Generar dinámicamente páginas web	Generar dinámicamente páginas web	Enfatiza la productividad y la lectura fácil del código	Código “divertido” y fácil de modificar por parte del desarrollador.
Apoyo de aprendizaje	Sitio web, foros, documentos proporcionados por Microsoft. En general buen soporte. Muy centralizada	Mucha, pero descentralizada. No hay una entidad que de forma oficial centralice la ayuda	Mucha, pero descentralizada. No hay una entidad que de forma oficial centralice la ayuda	Mucha, pero descentralizada. No hay una entidad que de forma oficial centralice la ayuda	Menos, pero descentralizada. No hay una entidad que de forma oficial centralice la ayuda
Soporte a móviles (todos por medio de un browser)	Native: Windows phone		Native: android		
Ambiente de desarrollo	Ms Visual Studio (costo) Y herramientas open source	Eclipse y otras herramientas open source	Eclipse, netbeans y otras herramientas open source	Eclipse, netbeans y otras herramientas open source	Eclipse, netbeans y otras herramientas open source

Nota: En todos los lenguajes se pueden realizar invocaciones con AJAX y a web services.

La tabla 4 muestra que el lenguaje Javascript no está contemplado en esta comparativa, ya que no es un lenguaje propio para desarrollar Backend, sin embargo, con la introducción de Node.js este paradigma cambia, pero para fines del proyecto en cuestión puede mezclarse con cualquier lenguaje Backend y realizar características en tiempo real. Referencia (Campus MVP, 2015).

El análisis de los resultados anteriores presentan los siguientes resultados:

- .NET: La tabla 4 presenta que los lenguajes .NET requieren un costo tanto en su implementación (servidor) como en su ambiente de desarrollo, los demás son open source (código libre), los costos de servidor de igual manera son gratuitos para todos los otros lenguajes menos .NET, todos son multiplataforma, .NET con el proyecto

MONO de Linux y solo permite usar C#, los motores de base de datos están especificados, SQL Server es muy potente mejor que los otros listados en performance y características, sin embargo, tiene un costo alto de licenciamiento, totalmente el lenguaje .NET está orientado a objetos, un paradigma moderno de los estilos de programación, cuenta con una ayuda centralizada muy potente y útil para los desarrolladores y por último posee el 3er lugar en la generación dinámica de páginas web también posee soporte nativo para desarrollo móvil.

- Java: Multiplataforma, gratis tanto en su implementación (servidor) como ambiente de desarrollo, posee motor de base de datos Oracle muy potente pero con costo de licenciamiento, aunque también puede usar mysql que es open source, totalmente orientado a objetos, posee mucha ayuda, sin embargo, posee el último lugar en la generación dinámicas de páginas web lo que lo hace muy lento para esta tarea.
- PHP: Multiplataforma, gratis tanto en implementación (servidor) como en ambiente de desarrollo, posee compatibilidad con motores de base de datos open source y puede escalar con costos de licenciamiento, sin embargo, las versiones gratuitas son muy estables y potentes, mucha ayuda aunque no centralizada, posee el cuarto lugar en la generación de páginas web, sin embargo, no está totalmente orientado a objetos lo que puede provocar un estilo de programación desordenado.
- Python: Multiplataforma, gratis en sus dos frentes, ocupa el primer puesto en la generación de páginas web y tiene una característica interesante está enfocado a la productividad y legibilidad en el código, lo cual permite que otros desarrolladores lo entiendan muy rápido, posee ayuda pero no esta tan centralizada y no posee soporte nativo para desarrollo móvil.

- Ruby: Por último Ruby, posee características similares a python inclusive su código se enfoca en escribir código divertido y muy ameno, sin embargo, no posee mucha ayuda para los desarrolladores.

2.5 Metodología de desarrollo SCRUM

La metodología ágil SCRUM consiste en un marco de trabajo que define las buenas prácticas necesarias y los roles para el diseño y desarrollo de entregables de un sistema, en períodos no mayores a un mes hasta completar la funcionalidad total del sistema requerido (Scrum Alliance, 2012).

Los roles principales en *Scrum* son el *ScrumMaster*, que mantiene los procesos y trabaja de forma similar al director de proyecto, el *ProductOwner*, que representa a los *stakeholders* (interesados externos o internos), y el *Team* que incluye a los desarrolladores.

El desarrollo de un sistema basado en esta metodología se trabaja mediante *sprint´s*, cada *sprint* define un período de tiempo comprendido entre una y cuatro semanas (el tiempo es definido por el equipo de desarrollo), durante el *sprint* el equipo diseña, prepara y desarrolla un incremento de software potencialmente (entregable). El conjunto de características que forma parte de cada sprint viene descrito en el *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar comúnmente denominadas (historias de usuario).

Los elementos del *Product Backlog* que forman parte del *sprint* a trabajar se determinara durante la reunión del *Sprint Planning*, durante esta reunión, el *Product Owner* identifica los

elementos del *Product Backlog* que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente *sprint.* Durante el *sprint*, nadie puede cambiar el *Sprint Backlog* que definieron el *Product Owner y el Team* de desarrollo, lo que significa que los requisitos están congelados durante el sprint.

La metodología SCRUM genera equipos auto-organizados y fomenta la comunicación continua entre los miembros del equipo y disciplinas involucradas en el proyecto. La figura 6 describe el proceso SCRUM en su totalidad.

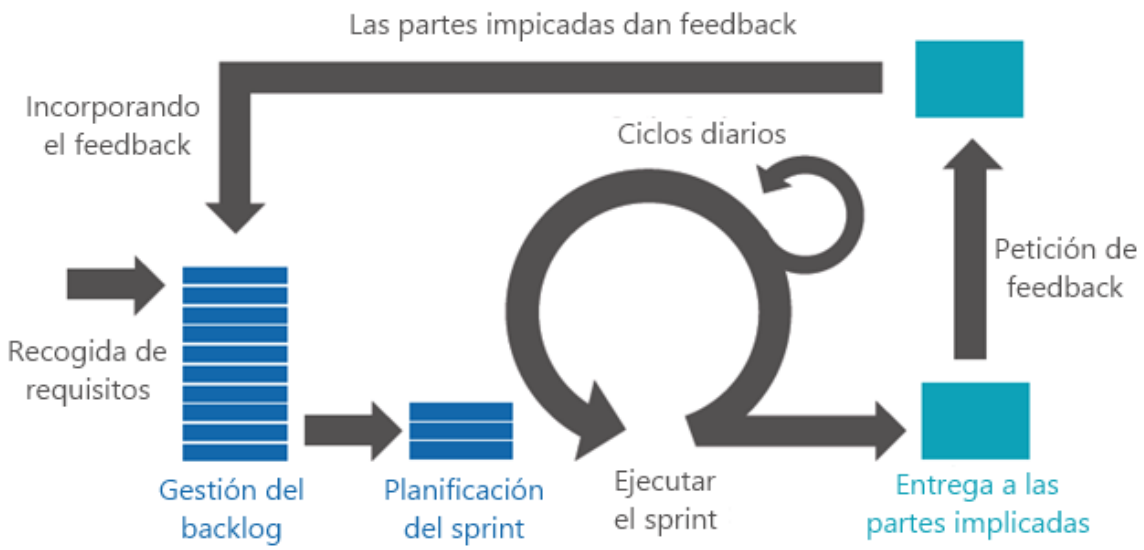


Figura 6. Proceso del desarrollo mediante la metodología SCRUM. Fuente http://programandonet.com/web/scrum-con-tfs/

Las características más notorias de SCRUM se listan a continuación según (Scrum Alliance, 2012).

1. Gestión regular de las expectativas del cliente.
2. Resultados anticipados.
3. Flexibilidad y Adaptación.

4. Retorno de Inversión.
5. Mitigación de riesgos.
6. Productividad y Calidad.
7. Alineamiento entre cliente y equipo.
8. Motivación al equipo.

Estas características hacen que SCRUM sea utilizado de manera regular en un conjunto de buenas prácticas para el equipo de desarrollo a manera de obtener los mejores resultados posibles. Las iteraciones entre las fases de desarrollo no necesariamente siguen un orden, se puede avanzar o retroceder entre ellas, haciendo el proceso más flexible y robusto en vez de llevar una metodología lineal y secuencial donde es necesaria la terminación de una etapa para continuar (Scrum Alliance, 2012).

Capítulo III

Metodología

Capítulo III - Metodología

3.1 Introducción

El desarrollo y gestión del proyecto SIGAF sigue la metodología de SCRUM, esta a su vez se gestiona mediante la herramienta RALLY que esta específicamente diseñada para trabajar sobre SCRUM. En esta herramienta gratuita online (también existen versiones de pago) cada rol del equipo registra su trabajo durante cada sprint, así como el registro de las tareas necesarias para completar cada historia del usuario definida en los sprint´s, Referencia (Cohn, 2014). A continuación se listan las actividades generales del rol de desarrollo Backend de la aplicación (Rally Software, 2015):

1. Investigación CVS (Git, Github).
2. Investigación del framework LARAVEL para programación MVC.
3. Codificación.
4. Integración de interfaz y base de datos.
5. Pruebas unitarias, depuración, verificación de requerimientos (Pruebas de Caja Blanca).

Nota: Estas actividades aplican para cada módulo de desarrollo.

3.2 Metodología de desarrollo Backend

SCRUM es utilizado como línea general para el ciclo de vida del desarrollo de software, dentro de la metodología empleada para desarrollar el Backend de los módulos del sistema SIGAF, se enumeran las actividades generales para el desarrollo de forma secuencial y sistemática para la obtención del código usable, que mantendrá la arquitectura del sistema en cada módulo descrito por el análisis, para el funcionamiento del sistema en general, la metodología está enfocada a un estudio de caso, por lo que, está dirigida a comprender la dinámicas presente

en un contexto singular según (Piedad, 2006). Esta particularidad hace que la metodología del estudio de caso investigado varíe acorde a la disciplina en este caso las ciencias computacionales. Para el desarrollo del sistema SIGAF se seguirá la pauta del modelo propuesto en el fundamento teórico del MVC o Modelo Vista Controlador.

Para optimizar la estructura de la metodología y mejorar la presentación del desarrollo, se divide en las siguientes fases los procedimientos y actividades a realizar en cada módulo del sistema SIGAF.

3.3 Fases de la metodología

3.3.1 Fase de preparación del entorno de desarrollo.

Como primera fase de la metodología se contempla la preparación e instalación del entorno de desarrollo, esto se realiza solo una vez durante toda la vida del ciclo de desarrollo, por lo tanto, se ejecuta esta acción al inicio del proyecto, que consta de las siguientes actividades:

1. Instalar un servidor web de prueba para peticiones http donde se harán las pruebas locales.
2. Instalar el módulo de php para que interprete el código generado.
3. Instalar el servicio de base de datos MYSQL para el almacenamiento y lectura de datos.
4. Instalar el framework de desarrollo LARAVEL el cual tiene las siguientes dependencias: Composer como gestor de paquete, php como lenguaje base y un editor de código para la escritura, configuraciones extra como extensiones openssl para comunicación y la activación del módulo de reescritura de las url del servidor web para permitir direcciones amigables.
5. Instalar un navegador web para pruebas.

6. Probar la página de inicio del LARAVEL para ver que todo esté integrado y funcione correctamente.

3.3.2 Fase de análisis de insumos o entradas necesarias para la codificación.

Para poder codificar y unificar las interfaces de la aplicación con el modelo de negocio (Base de datos del sistema), en el Modelo-Vista-Controlador, es necesario contar con las siguientes fuentes o entradas:

- El análisis de la lógica de negocio o los requerimientos del sistema enunciados en el documento oficial de recopilación de requerimientos, así como los diagramas de proceso y casos de uso que modelan el sistema, recopilados durante la fase de análisis.
- Las interfaces o vistas de la aplicación, como es un sistema web se refiere a los documentos fuentes HTML que contienen la maquetación del diseño gráfico en plantillas funcionales para que los usuarios interactúen con el sistema.
- Los scripts del modelo de negocio (Base de datos), así como también los diagramas entidad relación que describen el comportamiento y las relaciones de las entidades dentro del contexto del sistema.
- Minutas de cambios y acuerdos realizados en las reuniones periódicas que describe la metodología SCRUM.

Una vez obtenidos los insumos se procede al análisis de cada uno, para relacionarlos mediante el modelo descrito con anterioridad, esta actividad de recolección de insumos se realiza por cada módulo durante el proceso de desarrollo.

3.3.3 Fase de emparejamiento y cotejo de los insumos o entrada de proceso.

Las siguientes actividades aseguran que la información de los insumos mantenga una consistencia y coherencia para unificarlos en el modelo de programación:

1. Verificar que los datos descritos en el análisis de requerimientos se cumplan tanto a nivel de modelo (base de datos) como a nivel de vista (interfaces), esto quiere decir, que se verifica si existe un tipo de dato en un requerimiento específico, cumpla con un campo en las entidades de la base de datos y que de igual manera sea provisto un control para el usuario en la interfaz de la vista.
2. Verificar que las validaciones descritas en el análisis estén contempladas, en primer plano, durante la ejecución de acciones en la interfaz del usuario y en segundo plano, durante la integridad y mecanismos de control de la base de datos, para asegurar el almacenamiento de la información correctamente.
3. Si existe alguna discrepancia, comunicar directamente con el encargado del área y resolver las posibles dudas, errores hasta obtener el resultado óptimo antes de empezar con la codificación.

3.3.4 Fase de codificación y desarrollo.

Durante cada módulo se seguirán las siguientes actividades para generar y mantener código sustentable y que tendrá como resultado la funcionalidad de la aplicación:

1. Revisar los diagramas de secuencia y verificar mientras se codifica.
2. Crear la estructura de organización de los archivos dependiendo el modulo.
3. Escribir la parte lógica de la aplicación (controladores) y documentar, los cuales recibirán los datos y serán los encargados de procesar la información.

4. Escribir en la parte del Frontend los resultados que arrojen los controladores.
5. Escribir las funciones y documentar, estas funciones ejecutarán las acciones del lado del Frontend.
6. Hacer pruebas de cada petición y ruta ejecutada por el usuario.
7. Depurar la funcionalidad de la aplicación.
8. Verificar funcionamiento con equipo de desarrollo y retroalimentar.
9. Corregir y verificar de nuevo con equipo de desarrollo hasta que cumpla con el requerimiento aceptado.

3.3.5 Fase de publicación y control de versiones.

Una vez codificadas las funcionalidades de la aplicación se procederá con las publicaciones y fusiones de código con los otros miembros del equipo de desarrollo, esto con el fin de mantener actualizado en todo momento la última versión funcional del sistema, para lograr esto se seguirán las siguientes actividades:

1. Cada vez que se logre la funcionalidad especificada en el análisis, se hace una confirmación al repositorio principal del proyecto a través del sistema de control de versiones GIT-GITHUB, esto con el fin de tener una organización a nivel de código del proyecto.
2. Si el equipo de desarrollo sube actualizaciones en colaboración al repositorio principal del proyecto, se revisa, analiza y se aprueba para hacer una fusión respecto al código, esto provee una mejora, avance y actualización a la aplicación en desarrollo.

3. Revisar periódicamente el repositorio principal para tener actualizada la aplicaciones tanto remotamente en un servidor en la nube como localmente en los equipos de cómputo.

3.3.6 Fase de retroalimentación para la metodología Scrum con usuarios finales y equipo de desarrollo.

Una vez hechas las confirmaciones al repositorio principal del proyecto, se realizan las siguientes actividades para dar seguimiento al proyecto principal:

1. Reportar en la aplicación RALLY el avance de la codificación para retroalimentar al equipo.
2. Revisar en juntas programadas el avance de la aplicación y los problemas que se presentan durante el avance del desarrollo del proyecto con el equipo de desarrollo y los usuarios del sistema (retroalimentación).

Las fases descritas con anterioridad se desarrollaran por módulo hasta concluir el sistema, al final, la documentación de las funcionalidades describirá un API el cual podrá consultar el usuario para su posterior actualización o mejoras del sistema.

Capítulo IV

Desarrollo

Capítulo IV - Desarrollo

A continuación se presenta el desarrollo del proyecto, se aplica la metodología descrita en el apartado anterior. Se documenta el desarrollo modularmente como está dividido el proyecto, se empieza por el módulo de Plan de Estudios y se termina en el módulo de Login y Usuarios, no sin antes programar los catálogos principales de cada módulo en el cual se aplican las mismas fases descritas en la metodología empleando el patrón MVC.

Antes de la codificación se aplica la fase uno o inicial ya que prepara el entorno para realizar las fases posteriores, a partir de la fase dos se realiza la codificación de los catálogos al igual que el desarrollo de módulos principales.

4.1 Preparación del entorno de desarrollo

Esta fase se realiza al inicio del proyecto, esto se debe a que solo es necesario instalar y preparar las herramientas tecnológicas necesarias para la codificación una sola vez, después, durante cada módulo se hace uso de éstas para generar el código funcional de la aplicación.

Para la realización del código es necesario utilizar un entorno de desarrollo adecuado a la arquitectura que se estableció al inicio del proyecto. Mediante el framework Laravel para PHP se puede utilizar el modelo-vista-controlador para centrar los esfuerzos solo en la codificación de la funcionalidad requerida separando apropiadamente las capas del sistema.

El framework Laravel es la médula espinal de la arquitectura MVC utilizando como lenguaje base PHP, su correcta implementación presupone agilidad, desarrollo ágil y código ordenado de acorde a SCRUM, por lo tanto su puesta en marcha y correcto funcionamiento es esencial para continuar con el desarrollo de la codificación.

Como primer actividad se instalan y configuran las herramientas que a continuación se listan:

- Un editor de código en este caso se utilizará Sublime Text 2, con los plugins de PHP, HTML, y herramientas de visualización, para la eficiencia en desarrollo de código, figura 7.

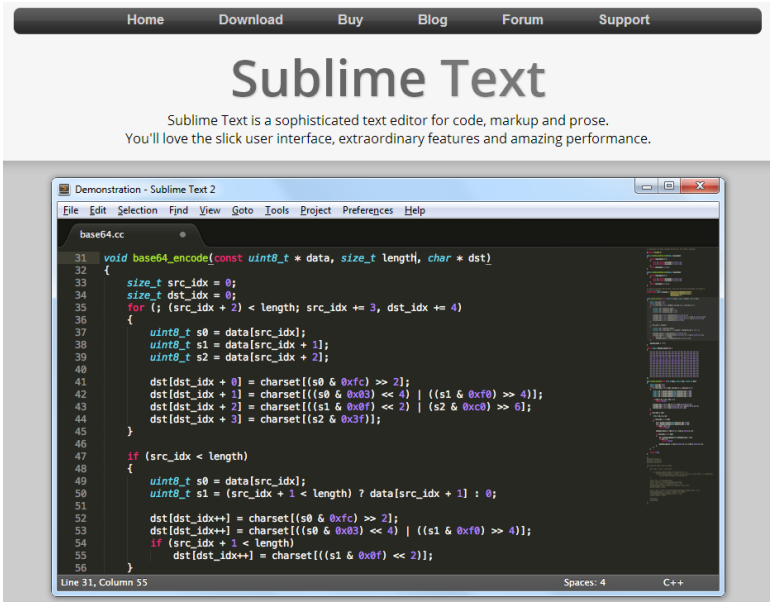


Figura 7. Sublime Text. Fuente http://www.sublimetext.com

- El gestor de paquetes “COMPOSER” para administrar los paquetes que dependen del framework de desarrollo LARAVEL basados en el lenguaje PHP. (Ver figura 8)

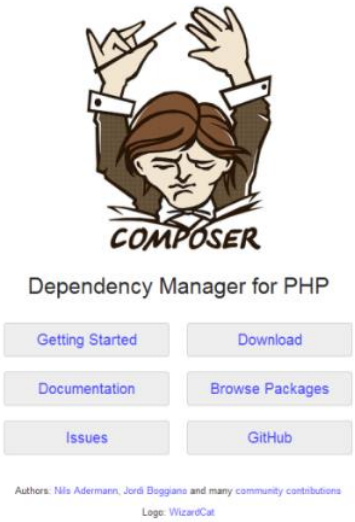


Figura 8. Composer. Fuente https://getcomposer.org

- Un servidor web de prueba en este caso se utiliza el paquete WAMP para la plataforma Windows que es donde se genera el código de la aplicación web. Este paquete contiene un servidor web, el intérprete PHP y el motor de base de datos MYSQL los cuales son necesarios para la generación del código. De la misma manera se activan los protocolos necesarios de seguridad para que permita a nuestra aplicación comunicarse de manera correcta con este servidor (openssl) y por último se activan en el servidor del módulo de reescritura de url's para tener más facilidad en la lectura de las direcciones que contienen las diferentes vistas del sistema (mod_rewrite). Ver figura 9.

Figura 9. WAMP Server. Fuente http://www.wampserver.com/en

- Se instala el framework de desarrollo laravel para utilizar la arquitectura mediante el gestor de paquetes composer. La figura 10 presenta la instalación Laravel.

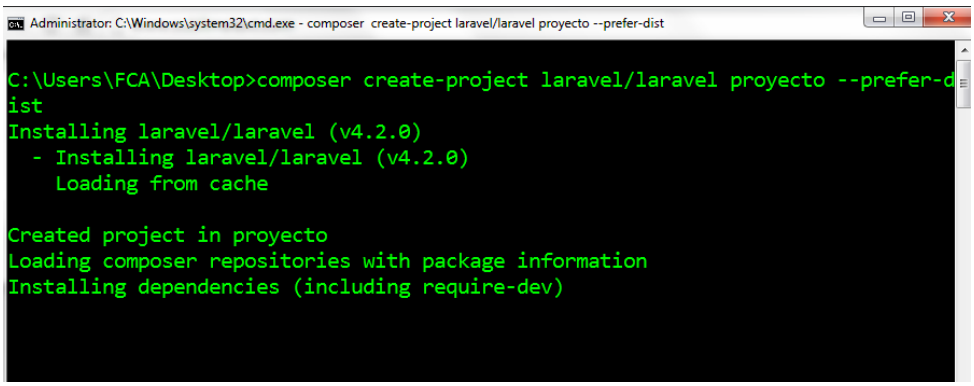


Figura 10. Instalación Laravel. Fuente Propia.

- Se coloca el proyecto en la carpeta local donde se almacenan los documentos públicos del servidor, se prueba que el servidor procese correctamente el framework Laravel, figura 11.

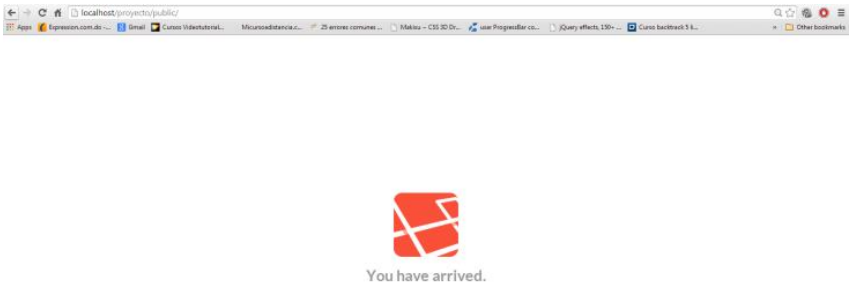


Figura 11. Página inicio Laravel. Fuente Propia.

- Se configura sublime con las rutas del proyecto para empezar con la codificación, se carga al editor las carpetas del framework y se completa la instalación del entorno. Ver figura 12.

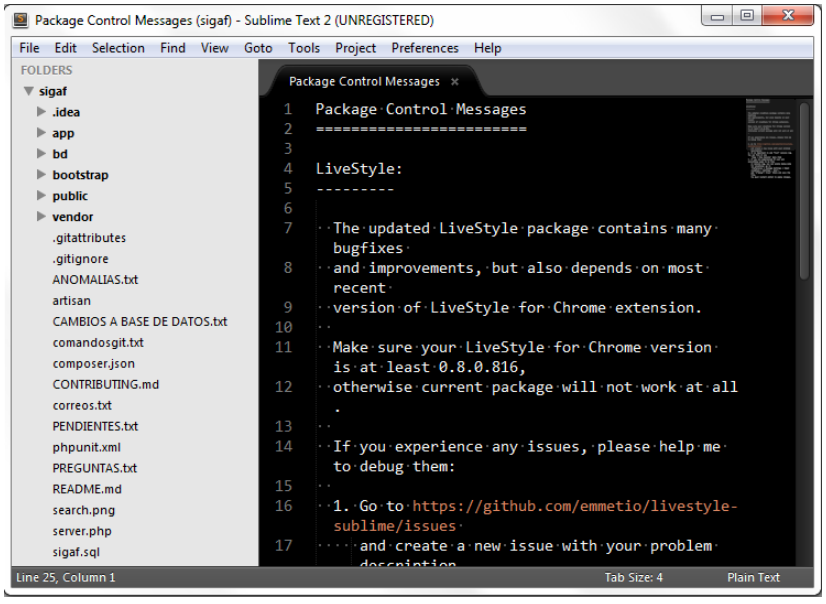


Figura 12. Proyecto configurado en Sublime Text 2. Fuente Propia.

- La figura 13 muestra el programa encargado del control de versiones (GIT-GITHUB), se abre un repositorio que contendrá todo el código generado de cada módulo y supondrá el almacenamiento de todos los avances del proyecto.

Figura 13. Git y Github. Fuente http://www.palermo.edu

- Por último, se crea una cuenta y se configuran el software Rally y Basecamp para dar seguimiento al proyecto tanto en la metodología SCRUM como en la plataforma de administración de proyectos BASECAMP. Ver figuras 14 y 15.

Figura 14. Gestor de proyecto Basecamp. Fuente http://sundogmedia.com

Figura 15. Rally SCRUM. Fuente http://www.forbes.com

4.2 Catálogos – Codificación de los catálogos principales para los módulos del sistema SIGAF

Parte fundamental de cualquier sistema son las fuentes básicas de información como los catálogos que son una colección de tablas propias de una base de datos (Universidad ITAM,

2002), los módulos del sistema SIGAF suplen su proceso de automatización mediante el consumo de los catálogos que se desarrollan a continuación.

4.2.1 Fase de análisis de insumos

Como primer paso se analizan los requerimientos funcionales del módulo **“**Plan de Estudios**”**, en el documento se especifican las funcionalidades que el sistema debe realizar de forma explícita, mientras que en el desarrollo se enfocará en el apartado técnico de la aplicación, básicamente en el cómo realizar la funcionalidad requerida.

Antes de examinar las funcionalidades principales, en las precondiciones del análisis se capturan los catálogos necesarios para realizar el plan de estudios vigente, para esto se procede a realizar el análisis de insumos que permite la captura de los catálogos desde la interfaz del Plan de Estudios.

Los catálogos requeridos para poder generar el código de la funcionalidad principal “Alta del plan de estudios” son:

1. Catálogo de números de plan de estudios.
    a. Catálogo niveles de programa.
    b. Catálogo programas educativos.
2. Catálogo de coordinaciones de área.
3. Catálogo etapas.
4. Catálogo de tipos de unidad.
5. Catálogo de tipos de seriación.

A continuación se describe el proceso para la codificación del primer catálogo, los catálogos restantes manejan la misma mecánica de desarrollo incluso los catálogos subyacentes de los módulos posteriores siguen este procedimiento.

4.2.2 Fase de emparejamiento

Registro de Número de Plan de Estudios (catálogo):

Para completar esta fase del desarrollo es necesario tener consistencia en las entradas que resultarán al final del proceso en el código fuente de la aplicación, para esta fase del desarrollo son necesarios los siguientes archivos:

1. Casos de uso y requerimientos del registro del número de Plan de Estudios.
2. Diagrama de secuencia del registro de un número de Plan de Estudios.
3. Diagrama entidad-relación que representa la entidad del número de Plan de Estudios.
4. Diagrama físico de la base de datos del número de Plan de Estudios.
5. Interfaz codificada para la captura del número del Plan de Estudios.

Una vez obtenido los documentos listados anteriormente se procede a realizar las siguientes actividades:

- Obtiene y analiza el diagrama de secuencia descrito en el análisis como en la figura 16:

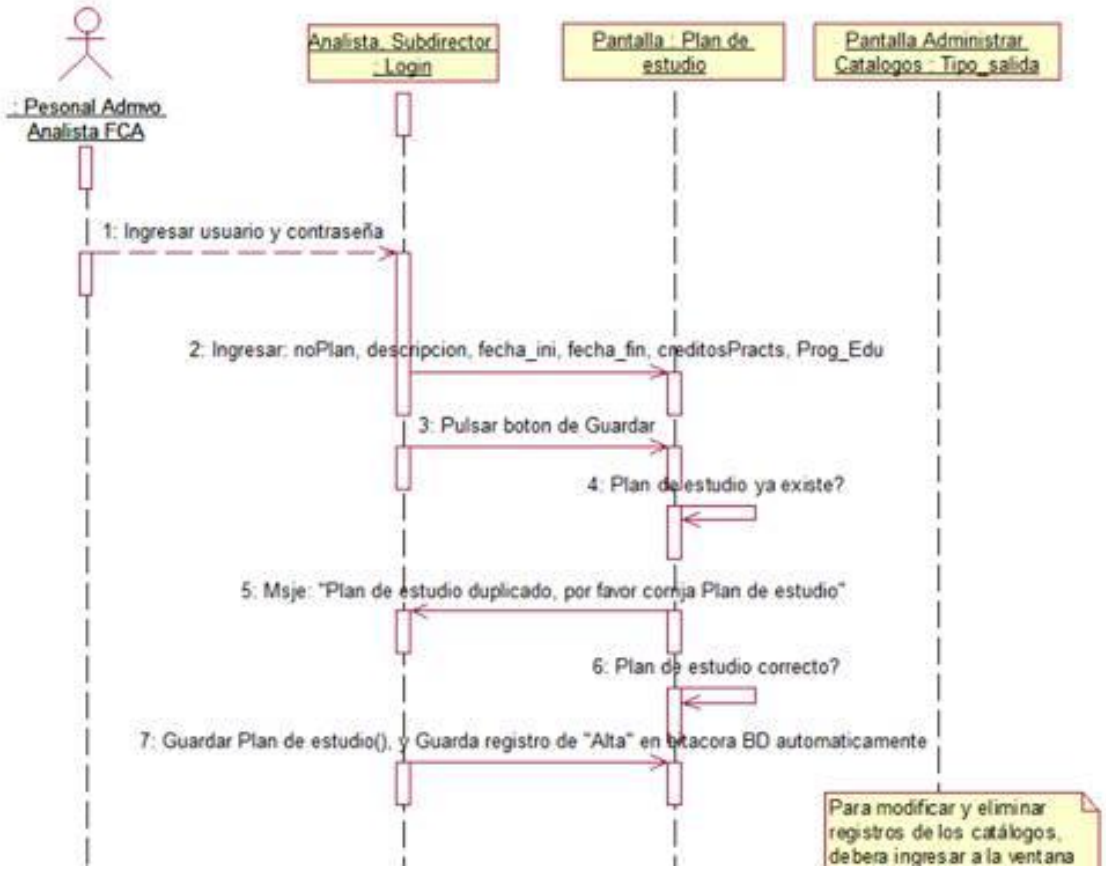


Figura 16. Diagrama de secuencia. Fuente (Duarte, SIGAF)

- Se obtiene y analiza la interfaz de usuario que contemple como realizar las acciones descritas en el diagrama de secuencia como en la figura 17:

Figura 17. Agregar número de Plan de Estudios.
Fuente (Duarte, SIGAF)

En la secuencia se especifica que se introduzca un número de plan, una descripción, se seleccione el nivel del plan a dar de alta, las carreras que serán partes de ese plan de estudios, los créditos totales del plan y la fecha de vigencia del mismo, se observa en la interfaz que se cumple con los controles necesarios para seguir la secuencia, siguen la estructura del diagrama, al final se observan las acciones principales que son guardar y salir.

- Una vez verificado el diseño lógico con el diseño físico incluimos en el análisis de la funcionalidad el diagrama relacional-conceptual, lógico y diccionario de datos de la BD para normalizar la información entre los datos y verificar que todo concuerda con el fin de realizar la funcionalidad requerida, se continúa con el ejemplo y verifica que los datos se puedan guardar en sus correspondientes entidades de la base de datos (Ver figuras 18 y 19).

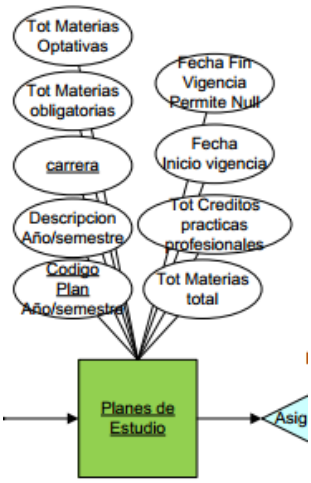


Figura 19. Diagrama físico Plan de Estudios. Fuente (Gambino, SIGAF)

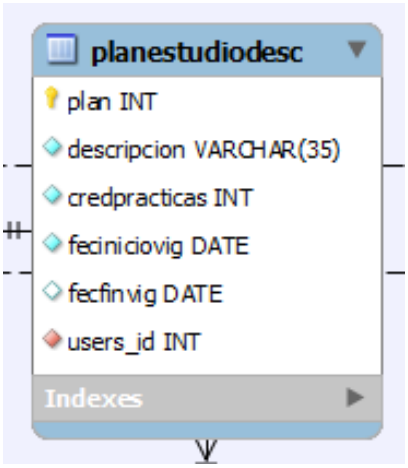


Figura 18. Diagrama conceptual Plan de Estudios. Fuente (Gambino, SIGAF)

Se coteja que existan los campos suficientes para almacenar la información en la interfaz de usuario (figura 19), de la interfaz, la ejecución de la tarea la realizará el código de la aplicación. Si durante el análisis de los insumos se detecta alguna anomalía se procede a corregir en donde exista una inconsistencia y se avisa al encargado para documentar el cambio.

Una vez cotejado que los insumos estén vinculados procedemos a realizar la siguiente fase de la metodología para obtener el código que ejecutara las funcionalidades requeridas por el análisis.

4.2.3 Fase de codificación y desarrollo

Una vez analizado y cotejado los insumos se procede a codificar de la manera en que el patrón de la arquitectura indica, para esto es necesario tener en cuenta que se programan 3 capas en el modelo.

1. Los modelos: entidades del modelo relacional.
2. Los controladores: se encargan de manejar la lógica de programación.
3. Las vistas: La interfaz de usuario con que el usuario interactúa con el sistema.

El framework laravel permite crear la abstracción de las entidades de la base de datos, mediante el ORM (Object-Relational-Mapping) ELOQUENT, este permite convertir datos entre el modelo relacional(tablas, relaciones, tipo de datos, etc.) a un sistema de datos orientado a objetos, transfiere las entidades de la base de datos a clases para utilizarlas en un sistema de objetos omitiendo la parte de las conexiones y el SQL propios del lenguaje de consulta, que son necesarios para manipular las entidades del modelo.

Siguiendo la codificación de la inserción del número del Plan de Estudios se procede a crear los modelos en el framework lo que permitirá la interacción entre la aplicación y la base de datos.

- Se realiza la codificación del modelo que contiene la entidad de la base de datos del número de Plan de Estudios (Ver figura 20).

```php
<?php

class PlanEstudio extends Eloquent
{
    protected $table = "planestudio";
    protected $primaryKey = "plan";
    public $timestamps = false;
    public $incrementing = false;

    public function unidades()
    {
        return $this->hasMany('UnidadAprendizaje','plan');
    }

    public function nivelD()
    {
        return $this->belongsTo('NivelPrograma','nivel','nivel');
    }
}
```

Figura 20. Modelo del número de Plan de Estudios. Fuente Propia.

- Se realiza la codificación del controlador que se hará cargo de conectar el modelo con la vista (Ver figura 21).

Figura 21. Código del controlador Plan de Estudios. Fuente Propia.

- Por último se programa la respuesta y las acciones que tendrá la vista al momento de registrar el número de Plan de Estudios (Ver figura 22).

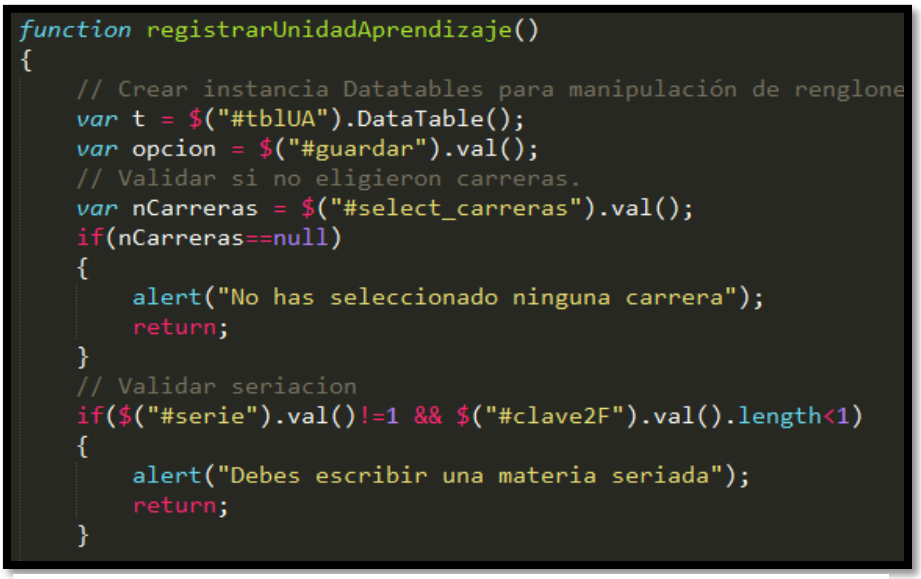


Figura 22. Programación en el Frontend. Fuente Propia.

- Se verifica que la funcionalidad se cumpla y si no se procede a la depuración (Ver figura 23).

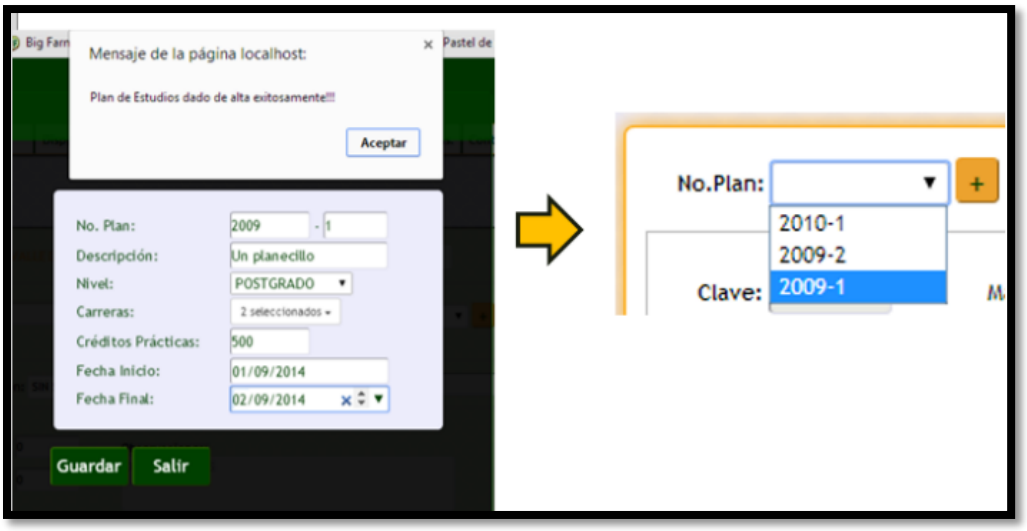


Figura 23. Comprobación de la funcionalidad programada. Fuente (Avila, Duarte, SIGAF)

4.2.4 Fase de publicación

- Una vez desarrollado el código del registro del catálogo, se hace uso de la tecnología de control de versiones para realizar la confirmación de que la funcionalidad ha sido agregada al proyecto principal, esto permitirá regresar en un futuro a versiones posteriores si fuera necesario realizar cambios no previstos desde etapas anteriores, como se muestra a continuación se realiza la confirmación mediante la plataforma git-github (Ver figura 24).

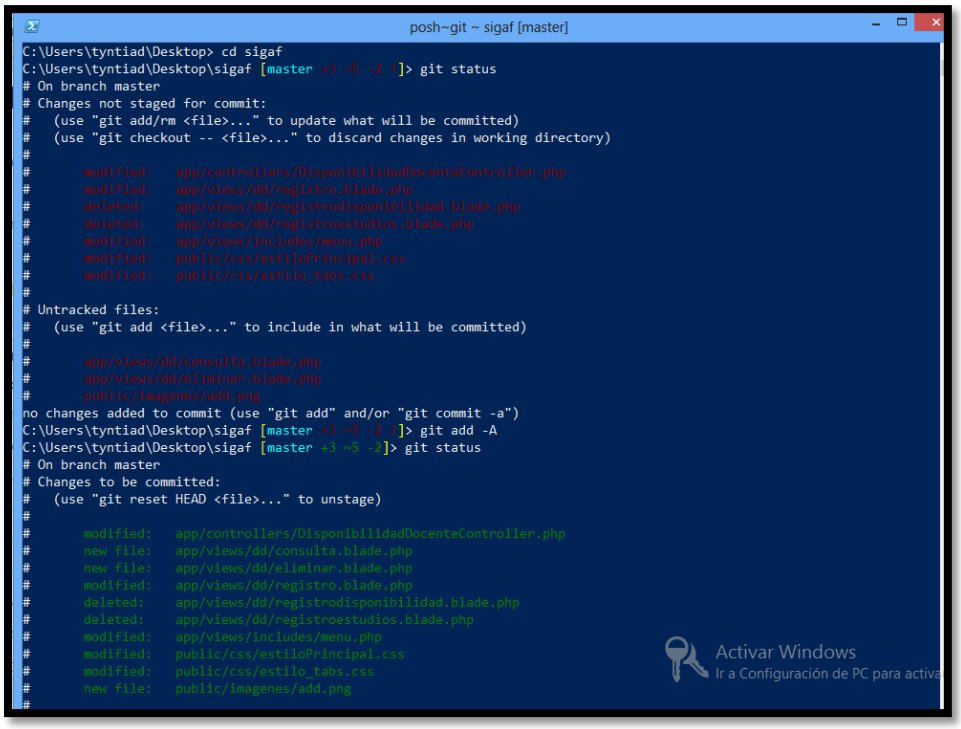


Figura 24. Uso de git para confirmar la funcionalidad.
Fuente Propia.

- Revisar en la plataforma GITHUB que la publicación realizada esta correcta para que todo el equipo de desarrollo pueda descargar la versión más actualizada del sistema (Ver figura 25).

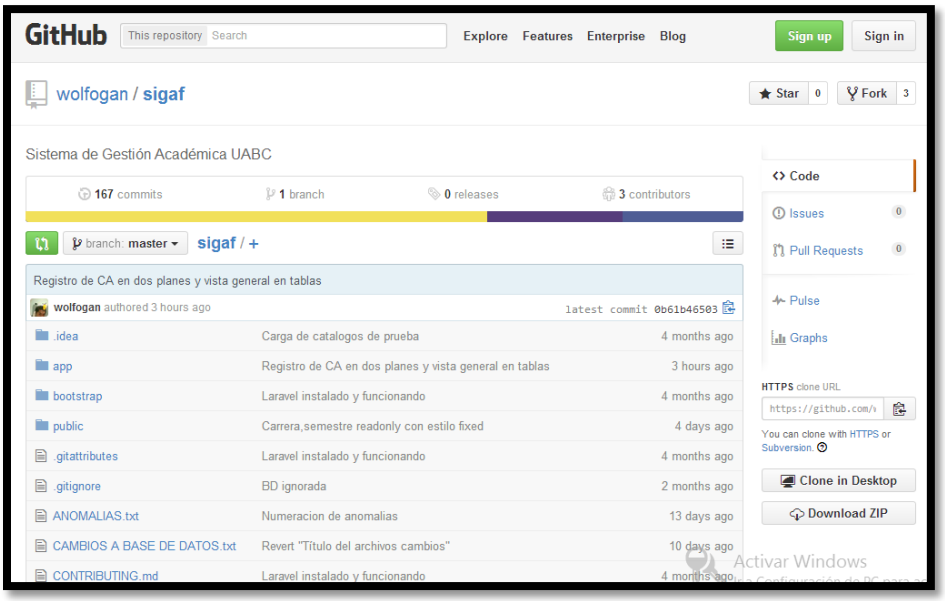


Figura 25. Confirmación de funcionalidad en repositorio remoto.
Fuente http://github.com/wolfogan/sigaf.git

4.2.5 Fase de retroalimentación

- Se procede a analizar junto con el equipo de desarrollo la aplicación y la funcionalidad, enseguida se publican en la plataforma BASECAMP los hallazgos y posibles cambios del desarrollo.
- Por último se actualiza la plataforma Rally para el seguimiento con SCRUM.

Las fases descritas con anterioridad abarcan la realización de los catálogos descritos al inicio este módulo.

4.3 Modulo 1 - Codificación Plan de Estudio

Una vez terminado con la codificación de los catálogos necesarios para el módulo de Plan de estudios, se procede desarrollar la parte funcional del módulo el cual consiste en altas, bajas, modificaciones y consultas de las unidades de aprendizaje dentro de un plan de estudios.

4.3.1 Fase de análisis de insumos

El desarrollo de las funcionalidades principales del módulo de Plan de Estudios comienzan con el análisis a detalle los insumos como es el caso de los casos de uso (tabla 5), diagrama de clases, BPMN, los diagramas de base de datos correspondientes y las interfaces desarrolladas.

Tabla 5. Descripción de casos de uso Plan de Estudios. Fuente (Castillejos, SIGAF)

Caso de uso	Sistema Actual	Sistema Propuesto
Inicio Sesión	(autentifica usuario) Solo firma con un usuario y contraseña común sin privilegios	(identifica y autentifica usuario) Dependiendo del privilegio del usuario habilitara y deshabilitara privilegios.
Borra Unidad de Aprendizaje	Borra de la base de datos el registro de la unidad de aprendizaje	Cambia status a desactivado o activado para tener control de manejo de las unidades de aprendizaje y poder dar seguimiento. El borrado a la base de datos solo se realizara cuando sea un error de captura por parte del usuario.
Registra Unidad de Aprendizaje	Registra todos los atributos mínimos correspondientes a la asignatura	Registra todos los atributos necesarios correspondientes a la asignatura, se integra el registro de especificación de la coordinación a la que pertenece esta unidad de aprendizaje.
Genera Reportes	Solo muestra reportes en pantalla no se puede imprimir la información de este modulo	El sistema ofrecerá la posibilidad de generar reportes predefinidos y también reportes de acuerdo a las necesidades de información del usuario combinando campos.
Gestión de Unidad de aprendizaje	Se realizan consultas y modificaciones de la unidad de aprendizaje	Ofrece consultas con solo acercar el cursor al atributo, para obtener detalle, se puede cambiar el orden de la unidades de aprendizaje con utilidades interactivas para el usuario. Las modificaciones se podrán realizar en la misma interfaz del atributo consultado sin necesidad de cambiar de pantalla.
Confirma alta Programa Unidad Aprendizaje	Se consulta la correcta captura solo por medio verificación visual en el sistema.	Se llevara un registro de confirmación de alta correcta de la unidad de aprendizaje.

Después de tener un panorama resumido de las funcionalidades se revisa a detalle el caso de uso que representa el requerimiento (Ver tabla 6).

Tabla 6. Descripción de caso de uso modificar unidad de aprendizaje.
Fuente (Castillejos, SIGAF)

Identificación del requerimiento:	RF-PE-05
Nombre del Requerimiento:	Modificar Unidad de Aprendizaje
Características:	El sistema permitirá al usuario Administrador y auxiliar administrativo la facilidad de cambiar el plan de estudios de manera fácil y entendible con visualización rápida del cambio con la opción de arrastrar y soltar
Descripción del requerimiento:	• Seleccionar del Menú principal la opción Unidad de Aprendizaje seleccionar submenú de Modificación o actualización Unidad de Aprendizaje . • Por medio de la vista mapa curricular se podrá modificar semestre sugerido, orden de presentación en el mapa, al seleccionar la materia en el mapa curricular se habilitará una pantalla para poder modificar cada uno de los campos capturados, registrar fecha de modificación y usuario que modifico, solo se guardaran los últimos cambios no se considera requerimiento guardar el historial de cambios. • La seriación se cambiara arrastrando y soltando el inicio y fin de la línea. • El único campo que no se podrá modificar es la clave de la unidad de aprendizaje, en caso de que se requiera modificar es necesario borrar ese registro y capturar el correcto como medida de control. • Al finalizar la captura mostrara todos los datos capturado para su revisión visual y mostrara un cuadro de dialogo para confirmar si esta correcto o si requiere alguna modificación, en caso de que este correcto y seleccione guardar mostrara un cuadro de dialogo como "Alta exitosa" y muestra el numero de registro consecutivo de la base de datos con los datos completos de la unidad de aprendizaje.
Requerimiento NO funcional:	• RNF-PE-01 • RNF-PE-02 • RNF-PE-06 • RNF-PE-08
Prioridad del requerimiento: Alta	

Una vez analizado las funcionalidades requeridas por el cliente se analiza el flujo actual del proceso mediante un diagrama BPMN para tener un entendimiento total del proceso, luego se evalúan las mejoras que traerán consigo el automatizar mediante el sistema y se referirá a un nuevo diagrama BPMN en el cual se observa el nuevo proceso (Ver figura 26).

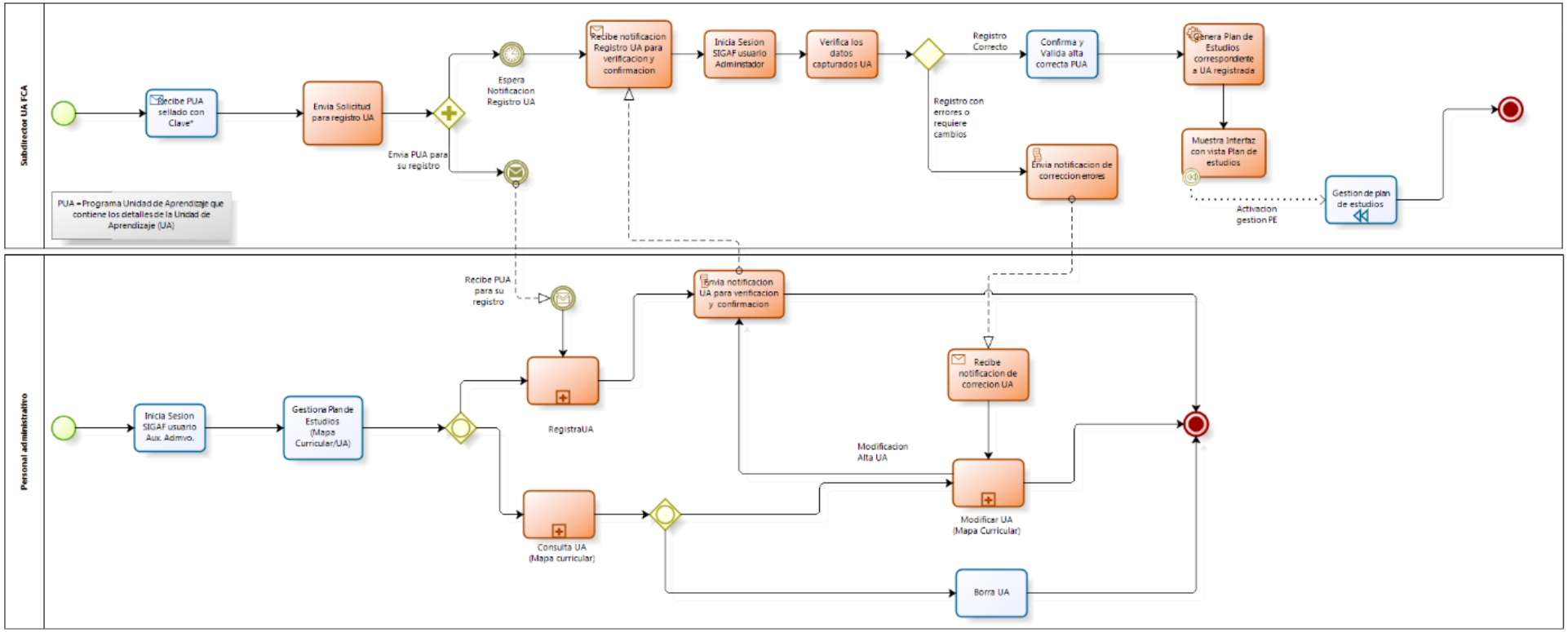


Figura 26. Diagrama BMPN propuesto para Plan de Estudios. Fuente (Castillejos, SIGAF).

Una vez analizado los requerimientos de usuario y los diagramas correspondientes a la creación de un Plan de Estudios continuamos con la siguiente fase que es empatar todas las entradas para comenzar con la codificación.

4.3.2 Fase de emparejamiento

Una vez analizado las acciones para el requerimiento de Alta en el Plan de Estudios se empata la base de datos con las interfaces y el análisis de esta funcionalidad para ello se revisa y corrigen discrepancias que puedan existir entre los diagramas del equipo de desarrollo.

Se verifica que se encuentren las tablas necesarias para almacenar las unidades de aprendizaje con los atributos correspondientes y que existan las relaciones necesarias para los programas educativos que conforman los planes de estudio (Ver figuras 27 y 28).

Se revisa que en la interfaz se registren los datos básicos de las unidades de aprendizaje como el nombre, créditos, horas clase y demás atributos necesarios que se identifican en la entidad (Ver figura 29).

Figura 27. Interfaz para registrar Unidad de Aprendizaje. Fuente (Duarte, SIGAF).

Se verifican que estén los controles para las acciones de registrar, modificar y eliminar las unidades de aprendizaje del Plan de Estudios (Ver figura 30 y 31).

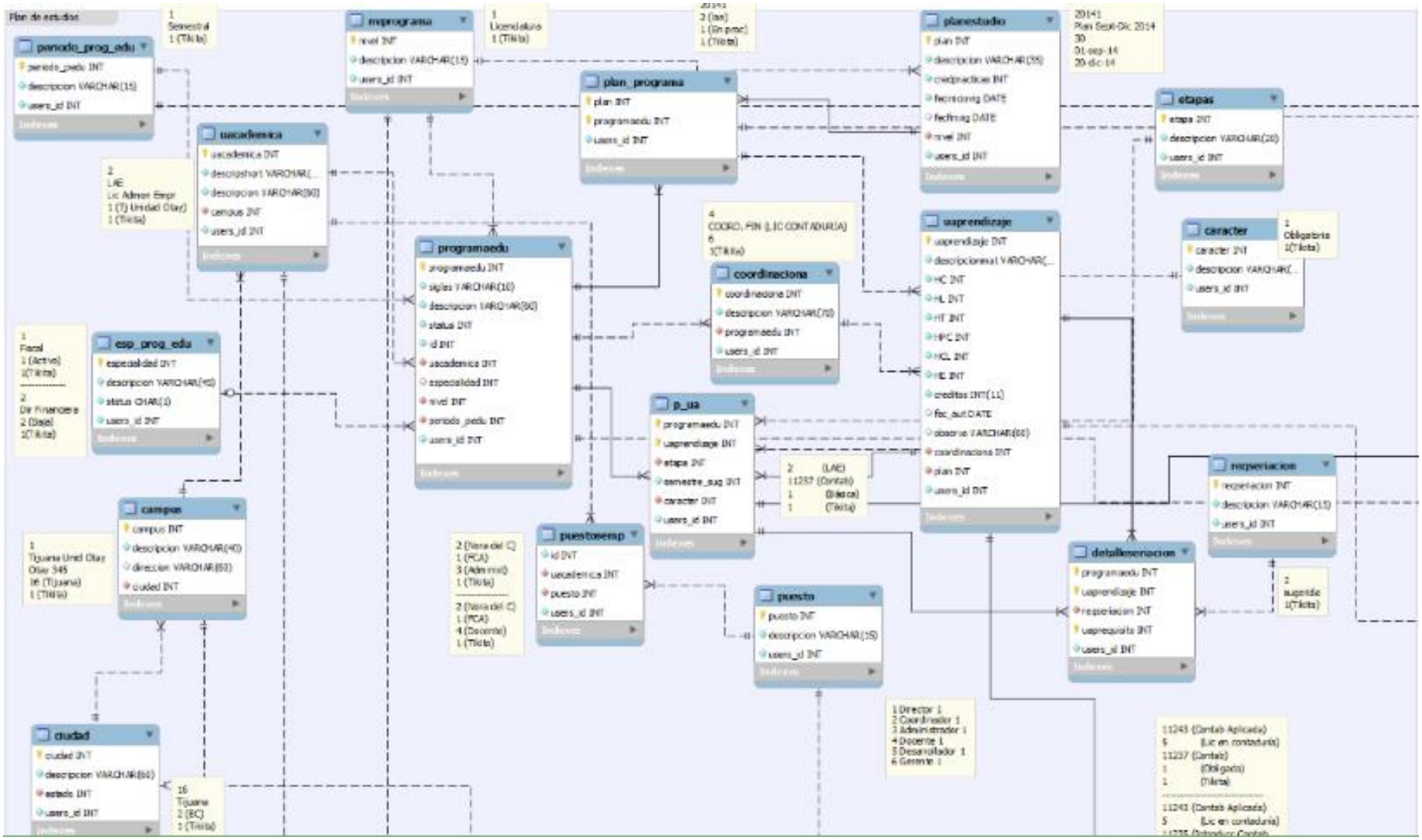


Figura 28. Base de Datos Plan de Estudios. Fuente (Gambino, SIGAF)

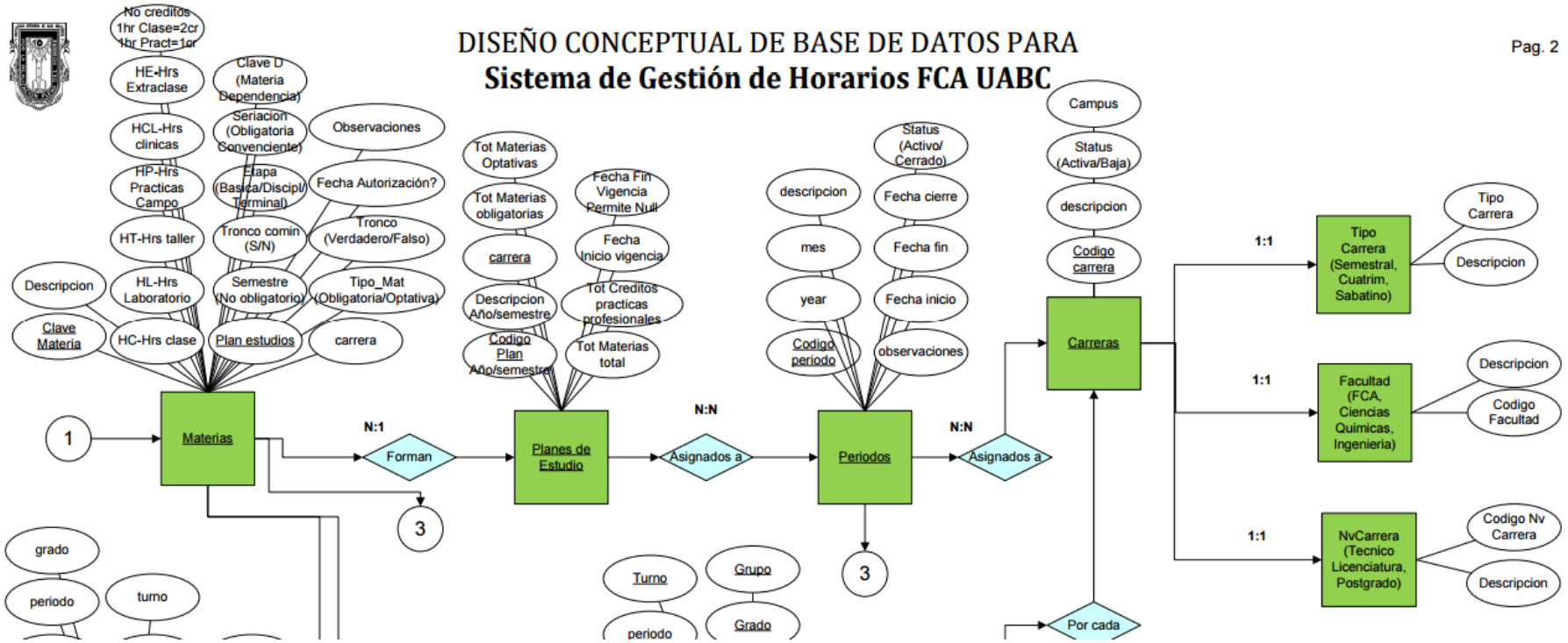


Figura 29. ER Plan de Estudios. Fuente (Gambino, SIGAF).

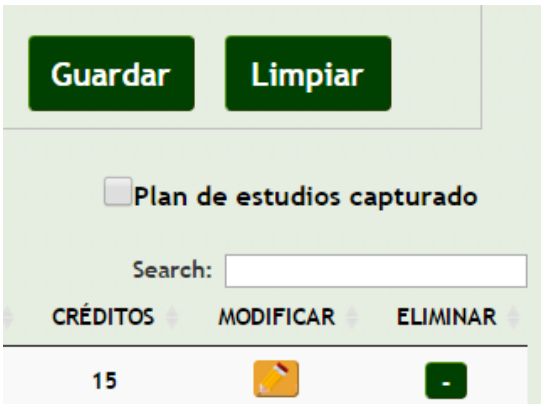


Figura 30. Acciones Unidad Aprendizaje.
Fuente (Duarte, SIGAF).

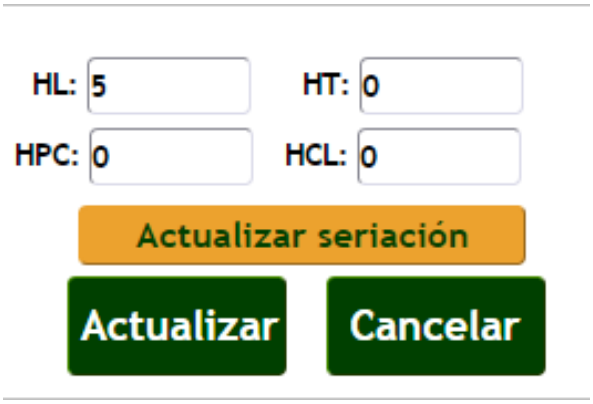


Figura 31. Actualizar Unidad Aprendizaje.
Fuente (Duarte, SIGAF)

También es necesario verificar las interfaces que permitirán las relaciones entre unidad de aprendizaje, carrera y planes de estudio (Ver figura 32).

Figura 32. Relaciones Programa Educativo, Plan Estudios y Unidad de Aprendizaje.
Fuente (Avila, Duarte, SIGAF)

Al existir cualquier discrepancia entre las relaciones de los diagramas, las interfaces y los requerimientos se notifican al equipo para que se realicen las correcciones y se documenta para seguimiento del proyecto.

Al término y aceptación de los insumos cotejados y emparejados se procede a realizar la codificación del módulo de Plan de Estudios.

4.3.3 Fase de codificación y desarrollo

Para codificar el alta del Plan de Estudios se procede a definir los modelos del Plan de Estudio con los cuales los controladores manipulan para mostrar los datos a las vistas. Ver figura 33.

Los modelos necesarios para realizar un alta de Plan de Estudios son los siguientes:

- Campus
- Categoría
- Ciudad
- Carácter
- Coordinación
- Especialidad
- Estado.
- Etapa.
- Level.
- NivelPrograma.
- Pais.
- Periodo.
- PeriodoPrograma.

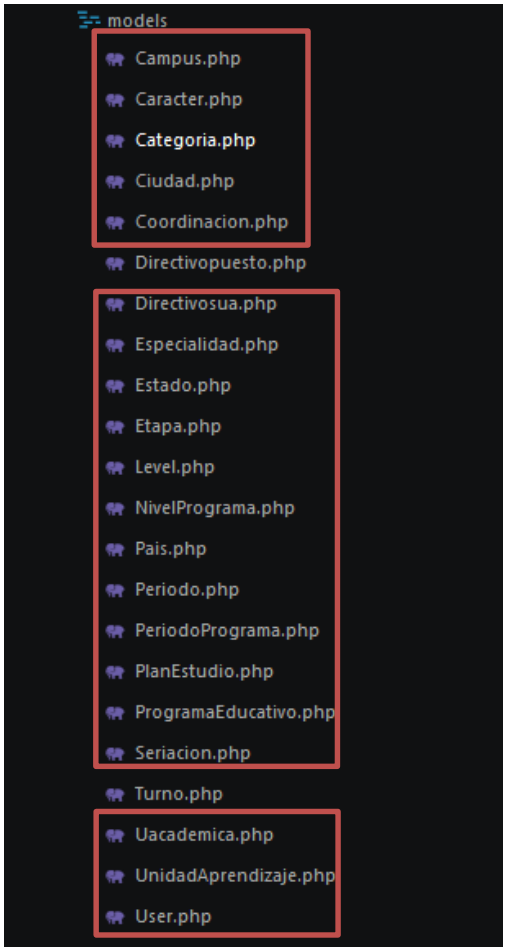


Figura 33. Modelos pertenecientes a Plan de Estudios. Fuente Propia.

- PlanEstudio.
- ProgramaEducativo.
- Seriacion.
- Uacademica.
- Users.

En cada modelo se específica el nombre de la clase debe tener el mismo nombre del archivo, y las propiedades para realizar la abstracción de la base de datos estas propiedades son: tabla, llave primaria y si contiene campos de tipo timestamp (creación y modificación).

Algunos modelos se les programan métodos de extensión para relacionarlos con otros modelos como se muestra en la figura 34:

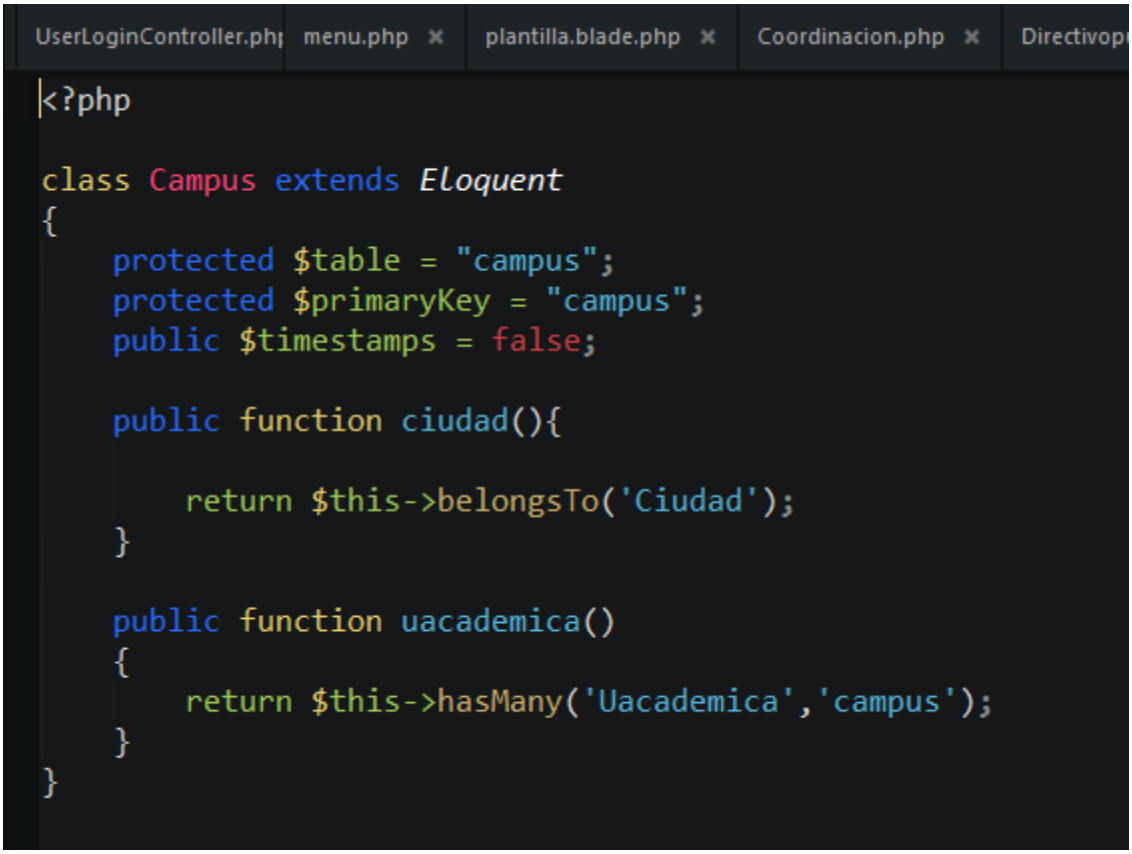


Figura 34. Ejemplo de Modelo Extendido. Fuente Propia.

Enseguida se desarrollan los controladores necesarios que realizarán las altas, bajas, modificaciones y consultas de las unidades de aprendizaje en el directorio destinado para los controladores de este módulo del sistema SIGAF (Ver figura 35).

```php
public function getRegistro()
{
    // Obtener planes de estudio: 2010-1, 2010-2, 2010-3
    //$planestudio = DB::table('planestudio')->distinct()->lists('PE_codigo');
    //$planestudio = DB::table('planestudio')->distinct()->select('PE_codigo')->orderBy('PE_codigo','desc')->get();
    $planestudio = PlanEstudio::select('plan')->orderBy('plan','desc')->get();
    $codigosPE = array();

    /**
    * Función para integrar el guión en el código del plan de estudio 2009-2
     * @param  string $string_add    La cadena a agregar
     * @param  string $string_target La cadena donde se va a agregar el string
     * @param  int $offset        Puntero donde corta la caden
     * @return string                Regresa la cadena concatenada
     */
    function str_insert($string_add,$string_target,$offset)
    {
        $part1 = substr($string_target,0, $offset);
        $part2 = substr($string_target, $offset);

        return $part1.$string_add.$part2;
    }

    for ($i=0; $i < count($planestudio); $i++) {
    $codigosPE[] = ["codigo" => $planestudio[$i]->plan,"formato" => str_insert("-",$planestudio[$i]->plan,4)];
    }
    // Licenciatura, Maestría
    $niveles = NivelPrograma::select('nivel','descripcion')->orderBy('nivel','asc')->get();

    // Cuatrimestral, Semestral
    $periodosPrograma = PeriodoPrograma::select('periodo_pedu','descripcion')->get();

    // Básica, Disciplinaria, Terminal
    $etapas= Etapa::select('etapa','descripcion')->get();

    // Obligatoria, Sugerida, Sin seriación.
    $seriaciones = Seriacion::select('reqseriacion','descripcion')->orderBy('reqseriacion','asc')->get();

    // Oblitatoria, optativa
    $tiposCaracter = Caracter::select('caracter','descripcion')->get();

    // Coordinación de area
    $coordinaciones = Coordinacion::select('coordinaciona','descripcion')->get();
```

Figura 35. Controlador Plan de Estudios interactuando con los Modelos. Fuente Propia.

A continuación se listan los métodos que ejecutan la funcionalidad y muestran los resultados a la vista del Plan de Estudios, el prefijo de cada método muestra el verbo HTTP que ejecuta la acción (“get”, “post”, “put”, “delete”):

- _construct: Valida las credenciales de la sesión.
- getRegistro: Obtiene la vista de registro y la carga de los catálogos iniciales.
- getConsulta: Obtiene la vista de consulta de Plan de Estudios.
- getBitacora: Obtiene la bitácora de las modificaciones al Plan de Estudios.

- getCatalogosadmin: Administra los catálogos básicos del sistema.
- postRegistrarplan: Registra un número de plan de estudios.
- postRegistrarnivel: Registra un nivel del plan (Licenciatura, Posgrado, etc.)
- postRegistrarperiodoprograma: Registra el periodo que puede durar un programa.
- postRegistraretapa: Registra una etapa al que pertenecerá una unidad de aprendizaje.
- postRegistrarseriacion: Registra el tipo de seriación al que puede pertenecer una unidad.
- postRegistrarcaracter: Define un nuevo tipo de carácter para la unidad.
- postRegistrarcoordinacion: Registra una coordinación de programa educativo.
- postRegistarprogramaeducativo: Registra un programa educativo con sus atributos.
- postRegistrarua: Registra una unidad de aprendizaje dentro de un plan de estudios.
- postAsociarprogramas: Asocia una o varias unidades de aprendizaje a diferentes programas educativos pertenecientes a un Plan de Estudios.
- postObtenernivelplan: Devuelve el nivel al que pertenece un Plan de Estudios.
- postObtenermateria: Devuelve una unidad de aprendizaje localizada por su id.
- postObtenerclave: Regresa un nuevo número de unidad de aprendizaje basado en el consecutivo, una vez registrado.
- postObtenerclaveseries: Obtiene las claves de las materias seriadas.
- postObtenerclavescarrera: Obtiene las claves de las unidades contenidas en un programa educativo.
- postObtenerprogramas: Obtiene los programas educativos pertenecientes a un Plan de Estudios.

- postObteneruas: Obtiene las unidades de aprendizaje pertenecientes a un plan de estudios.
- postObteneruascarrera: Obtiene las unidades de aprendizaje pertenecientes a un programa educativo.
- postContaruas: Cuantifica la cantidad de unidades en un programa educativo o plan de estudios.
- postEliminarua: Elimina una unidad de un plan de estudios.
- postElminarpua: Elimina una unidad de un programa educativo.
- postObtenerdetalleseriacion: Obtiene los datos de una unidad seriada dentro de un programa.
- postActualizarua: Actualiza los atributos de una unidad de aprendizaje.
- postActualizaretapa: Cambia de etapa una unidad de aprendizaje.

Una vez terminado los métodos de los controladores se realiza la programación de la respuesta de la vista como se muestra en el ejemplo al registrar una unidad se manda a llamar al método en el controlador mediante una ruta como se muestra en la figura 36.

```
    $.post("<?php echo URL::to('planestudio/registrarua'); ?>",dataUA,function(mensaje){
        var noPlan=$("#noPlan").val();
        var clave1F=$("#clave1F").val();
        var materia=$("#materia").val();

        alert(mensaje);

        //reset_campos();
        // Mostrar la ventan modal para el detalle
        $("#detalle").text(clave1F+ " - " + materia.toUpperCase());
        $("#asignacion").click();

    })
    .fail(function(errorText,textError,errorThrow){
        alert("FALLO EN EL REGISTRO: " + errorText.responseText);
    })
    .always(function(){
        // OCULTAR AJAXLOADER
        $("#ajaxLoad").css("display","none");

    });
}
```

Figura 36. Llamada de la vista al método del controlador. Fuente Propia.

Al final la vista interactúa con el controlador y este a su vez con el modelo realizando el ciclo de funcionalidad completo, una vez realizado este proceso se replica en todas las acciones que permiten que se cumpla el requerimiento, al final la interfaz responde a los métodos como se muestra en la figura 37 completando los casos de uso:

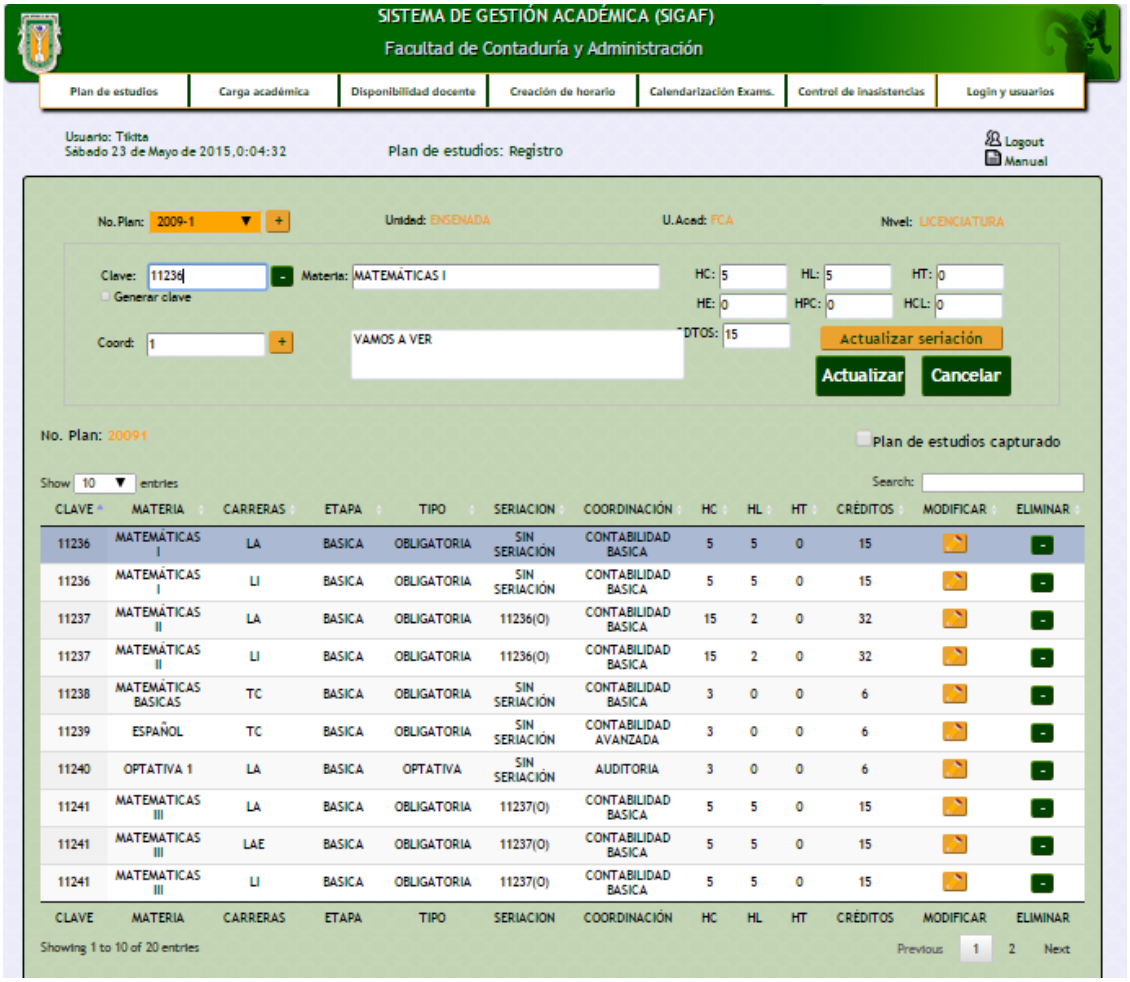


Figura 37. Interfaz principal modulo Plan de Estudios. Fuente (Avila, Duarte, SIGAF)

Después de generar el código se procede con la consulta, el registro de las unidades que presenta la misma funcionalidad solo que a nivel individual permitiendo modificar la unidad de aprendizaje como se muestra en la figura 38.

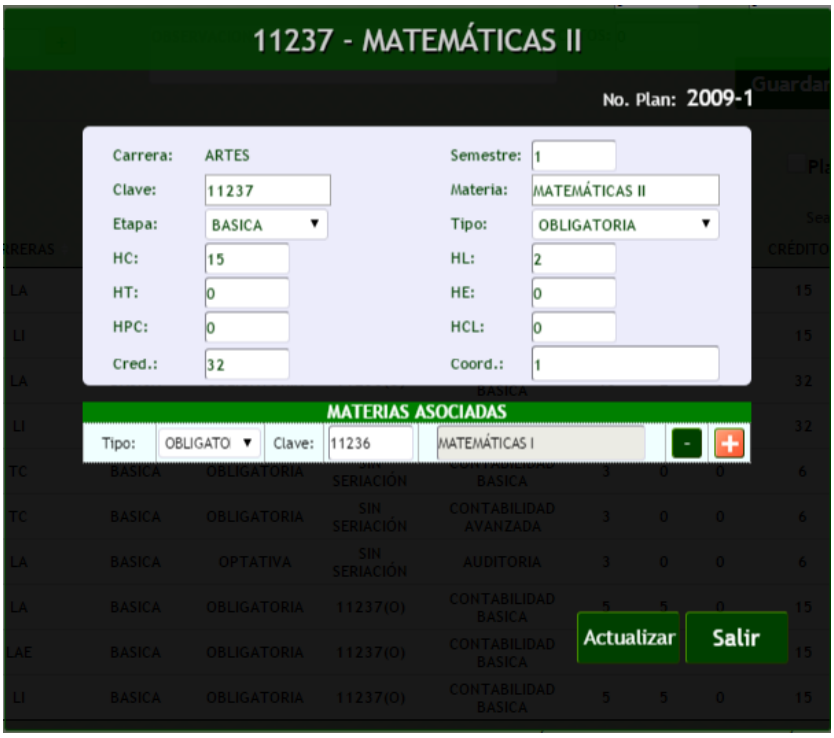


Figura 38. Consulta individual de Unidades de Aprendizaje.
Fuente (Duarte, SIGAF)

Se programan las acciones de búsqueda en la consulta para mostrar en forma de cuadro conceptual las unidades de aprendizaje dadas de alta en un Plan de Estudios, figura 39.

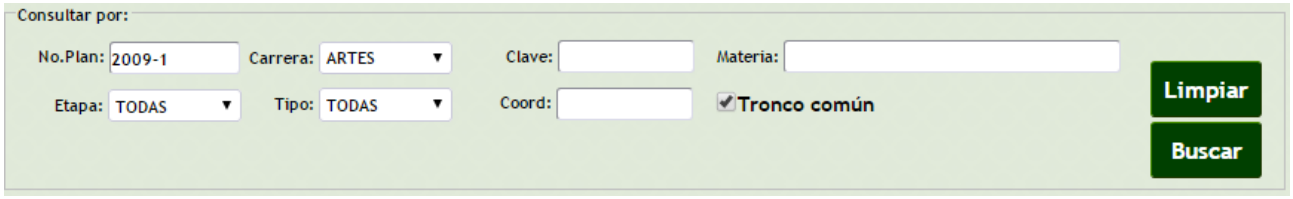


Figura 39. Búsqueda Plan de Estudios. Fuente (Duarte, SIGAF).

Al final se desarrolla el código pertinente para manipular las unidades de aprendizaje mostradas de forma gráfica como permitiendo “drag” a “drop” para facilitar el cambio de etapa, los colores en la interfaz indican el tipo de unidad y mediante programación se definen al realiza la búsqueda, como se visualiza en la figura 40.

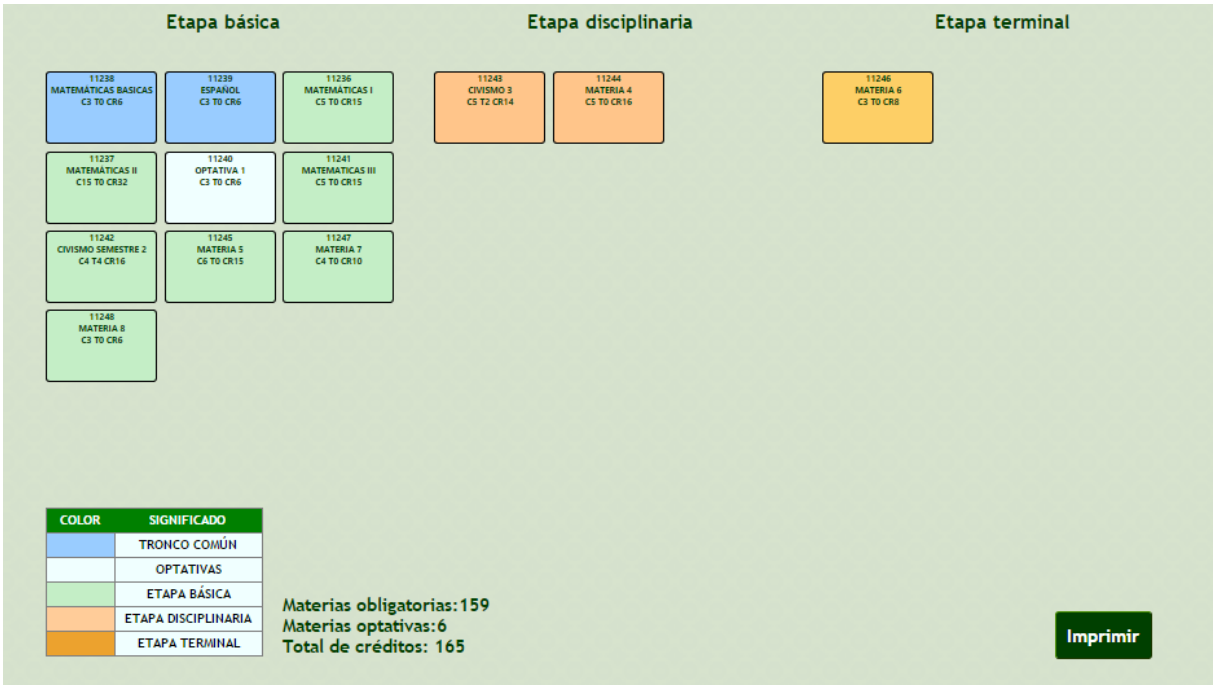


Figura 40. Forma gráfica de Plan de Estudios. Fuente (Duarte, SIGAF).

Con esto se concluye la generación de código para el alta de un Plan de Estudios con sus respectivas unidades de aprendizaje y relaciones con los programas educativos.

4.3.4 Fase de publicación

Cuando se termina el desarrollo se procede a documentar en la plataforma Github y Basecamp para conocimiento de equipo y revisar la retroalimentación de los posibles errores y mejoras en la funcionalidad de la aplicación, si existen errores se documentan y se procede a la depuración.

En Github se maneja un sistema de Milestone para completar los módulos en tiempo, para que el equipo de desarrollo actualice sus versiones locales del proyecto como se muestra en la figura 41:

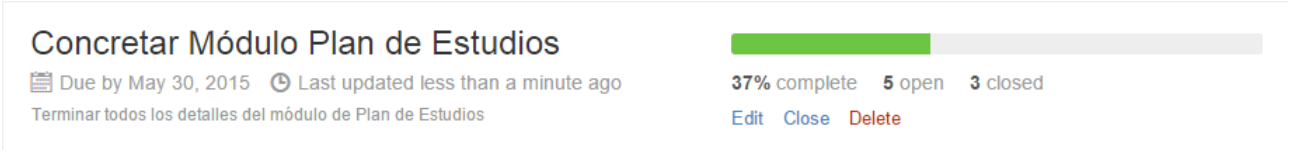


Figura 41. Milestone Plan de Estudios. Fuente Propia.

4.3.5 Fase de retroalimentación

Se recibe por parte del equipo en específico del tester del sistema los errores encontrados durante la ejecución del sistema para corrección, se depura, corrige se actualizar la versión y se publica en las plataformas para seguimiento del mismo esperando que el test libere la versión para producción, lo cual se observa en la figura 42.

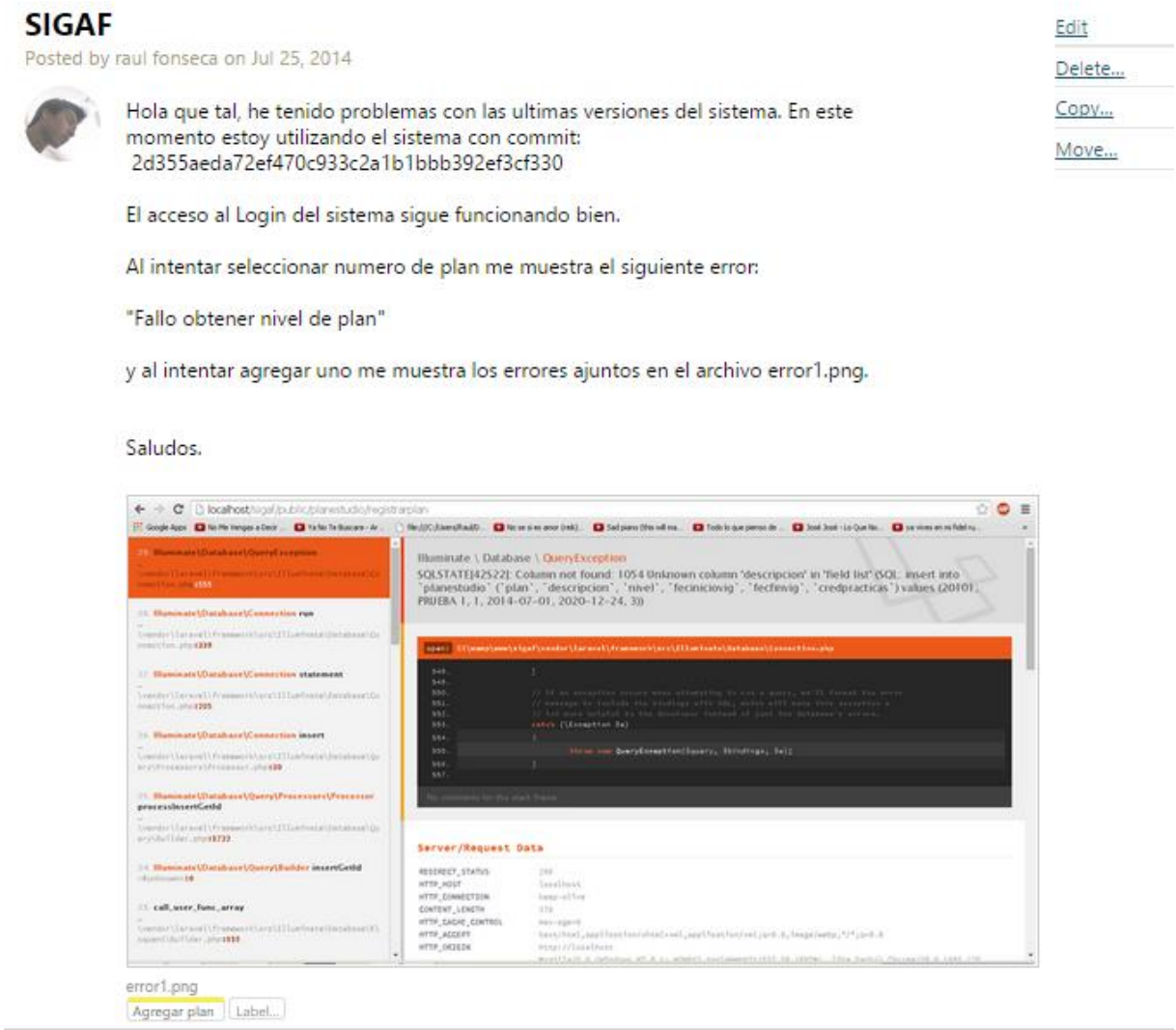


Figura 42 - Ejemplo de Feedback en Basecamp. Fuente Propia.

En Github los Milestones se trabajan mediante eventos denominados issues o cuestiones encontradas, para esto, en el desarrollo se documentan cada vez que se encuentran errores en el código ya sea por el programador o por el equipo.

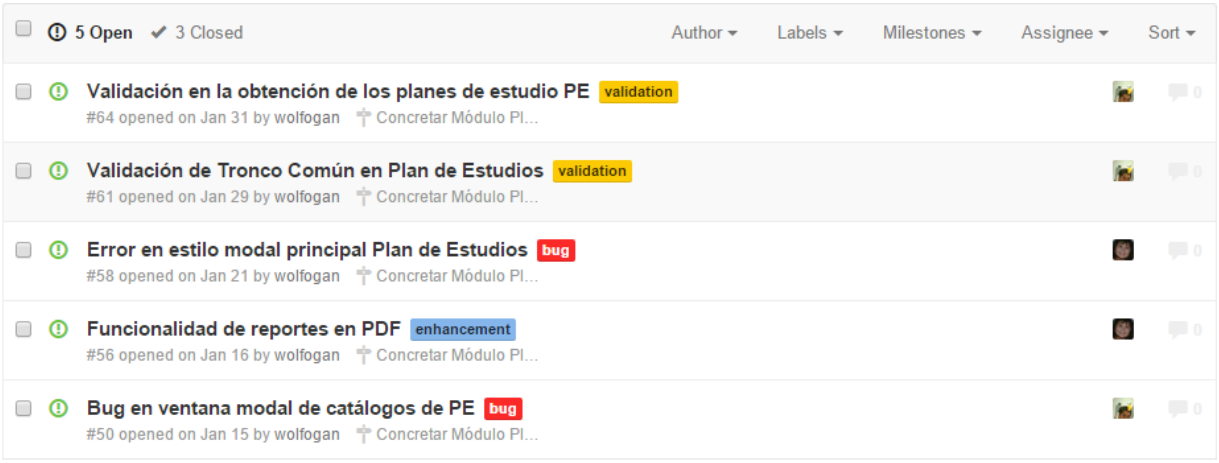


Figura 43. Issues en Plan de Estudios. Fuente Propia.

Al terminar se realizan las actualizaciones pertinentes en el Rally de SCRUM para documentar el proceso (Ver figura 43).

4.4 Módulo 2 - Codificación Carga Académica

El módulo de carga académica contempla tres interfaces que contiene toda la funcionalidad de carga académica estas 3 interfaces son:

1. Registro de Carga Académica.
2. Registro de Carga Subsecuente.
3. Consulta de Carga Académica.

A continuación, se presenta el desarrollo de la funcionalidad de carga académica cuyo objetivo principal es dar de alta una carga académica para un periodo específico perteneciente a un programa educativo siempre obteniendo las unidades de aprendizaje actuales del plan de estudios vigente.

4.4.1 Fase de análisis de insumos

Como en el análisis de las entradas de los primeros módulos se examinan los diagramas correspondientes al módulo de Carga Académica.

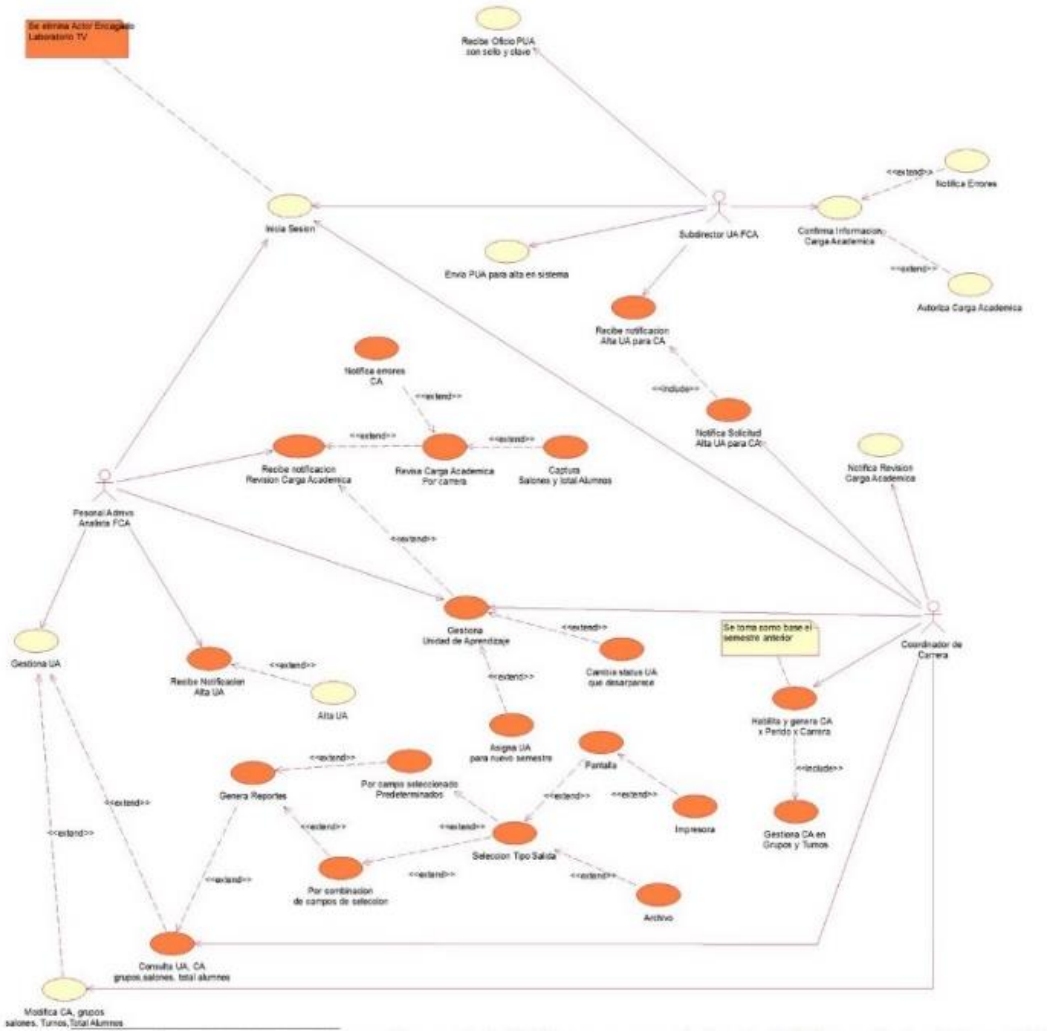


Figura 44. Diagrama casos de uso CA. Fuente (Castillejos, SIGAF).

Se analizan cada uno de los casos de uso (Ver figura 44) y su descripción para abstraer la funcionalidad, se verifica el diagrama BPMN (Ver figura 45) para enfocar las funcionalidades del sistema al flujo de proceso.

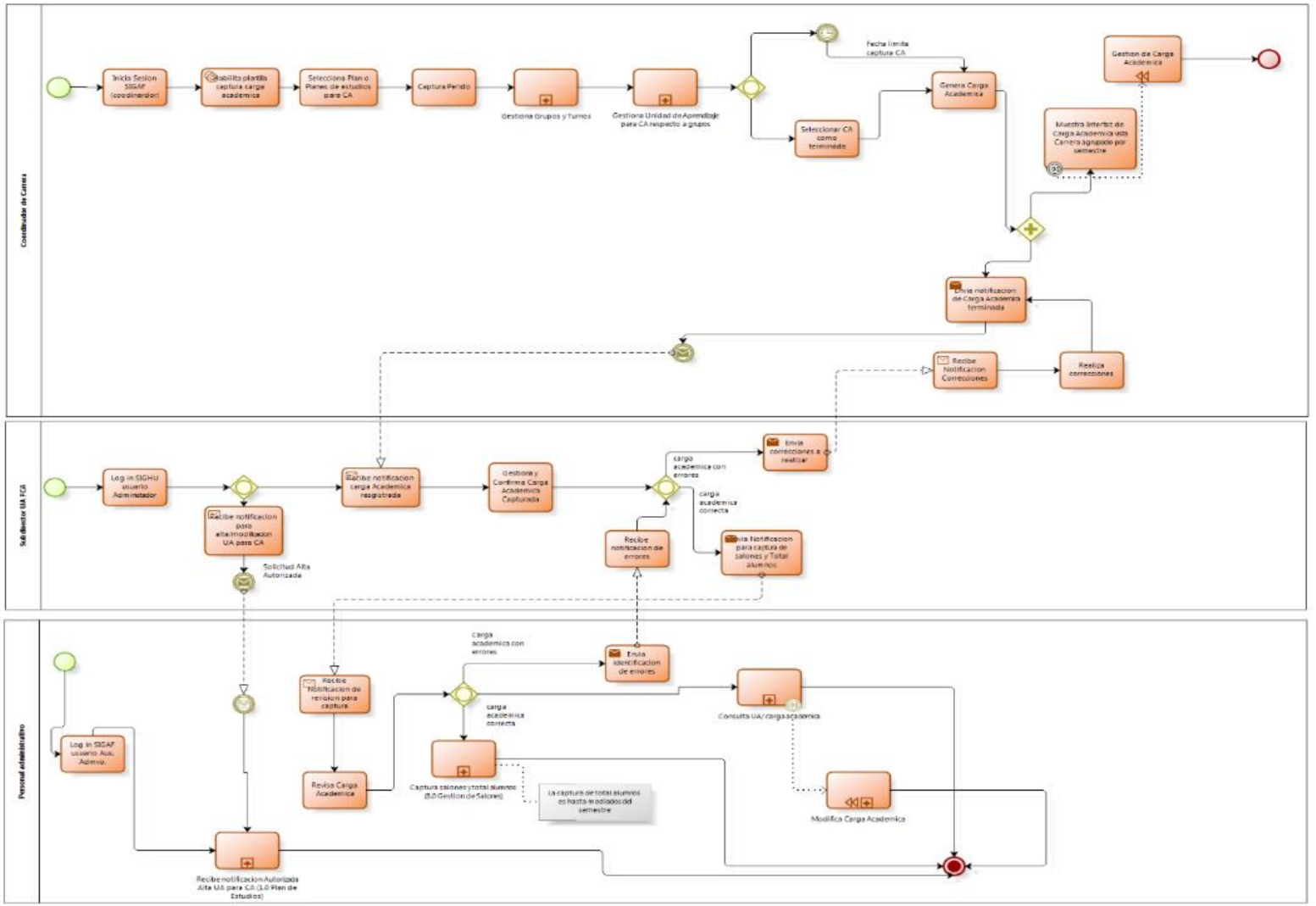


Figura 45. Diagrama BPMN Carga Académica. Fuente (Castillejos, SIGAF).

Una vez verificado el nuevo proceso mediante la funcionalidad de la aplicación, se resume la funcionalidad requerida a las siguientes actividades:

1. Registrar un periodo para dar de alta una carga académica.
2. Registrar las unidades de aprendizaje del plan de estudios vigente o el anterior, dentro de una carga académica.
3. Posibilidad de registrar la carga académica por semestre.
4. Posibilidad de interactuar con las unidades de aprendizaje de 2 planes de estudio en paralelo.
5. Asignar los grupos que pertenecerán a dicha carga académica registrada.
6. ABC de los grupos y semestres que representan la carga académica.
7. Duplicar una carga académica anterior y posibilidad de modificarla.
8. Consultar carga académica por diversos criterios pertenecientes a sus atributos como periodo, grupos, semestres, etc.

4.4.2 Fase de emparejamiento

Para poder emparejar las entradas de este módulo es necesario tomar en cuenta las relaciones que existen con el primer módulo porque las unidades de aprendizaje que se registran en la carga académica pertenecen al plan de estudios. En esta ocasión el diagrama Entidad-Relación debe contener las llaves foráneas necesarias para poder almacenar los datos correctamente. Y las interfaces deben contener los controles necesarios para poder registrar la carga académica.

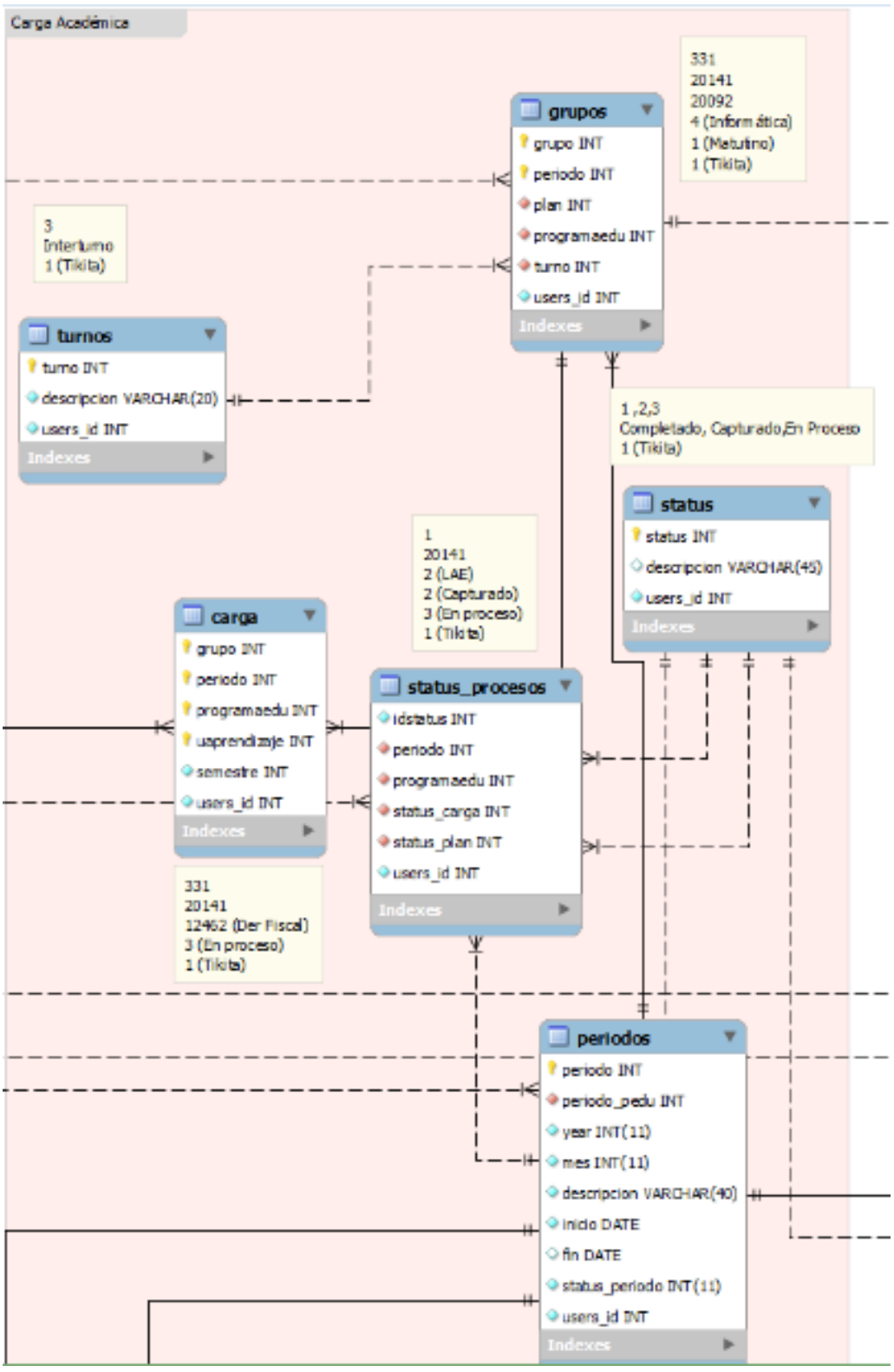


Figura 46. ER Físico Carga Académica. Fuente (Gambino, SIGAF).

Al terminar de verificar el diagrama Entidad-Relación (figura 46) que corresponde a la Base de datos, se procede a evaluar la interfaz y verificar que se contengan los controles necesarios para poder realizar el registro de una carga académica, se inspeccionan los controles y la utilidad que representara para el usuario.

Se examinan las tres interfaces proporcionadas por el equipo:

1.- La interfaz de registro de Carga Académica (Ver figura 47):

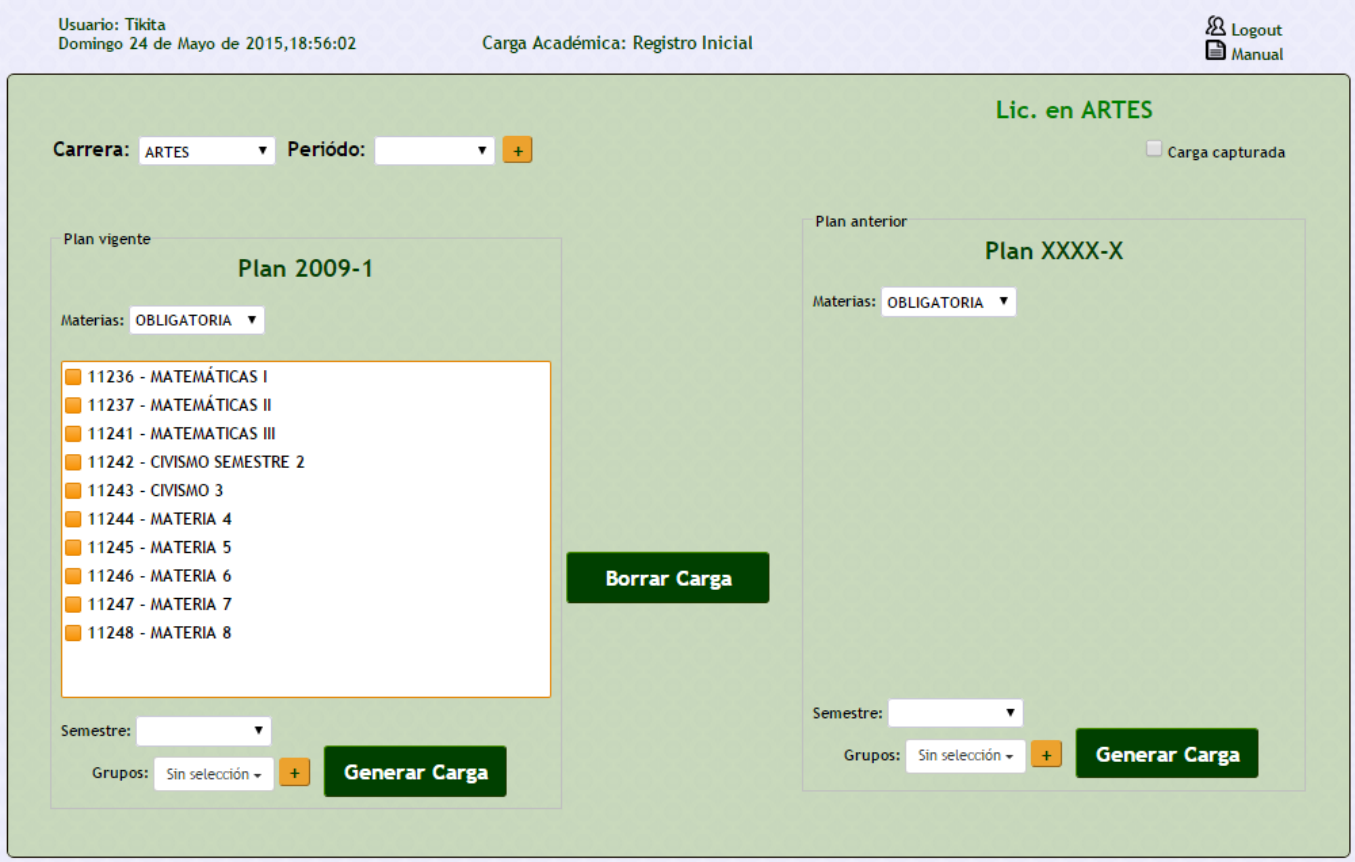


Figura 47. Registro CA. Fuente (Avila, Duarte, SIGAF).

En esta interfaz se contienen los controles necesarios para mostrar las unidades de aprendizaje de los planes de estudio, los semestres al que serán asignados como también los

grupos destinados a los mismos, también se cuenta con las las ventanas modales para dar de alta los periodos y en su defecto los controles para el registro y la modificación de la carga.

2.- La interfaz de Carga Académica Subsecuente (Ver figura 48):

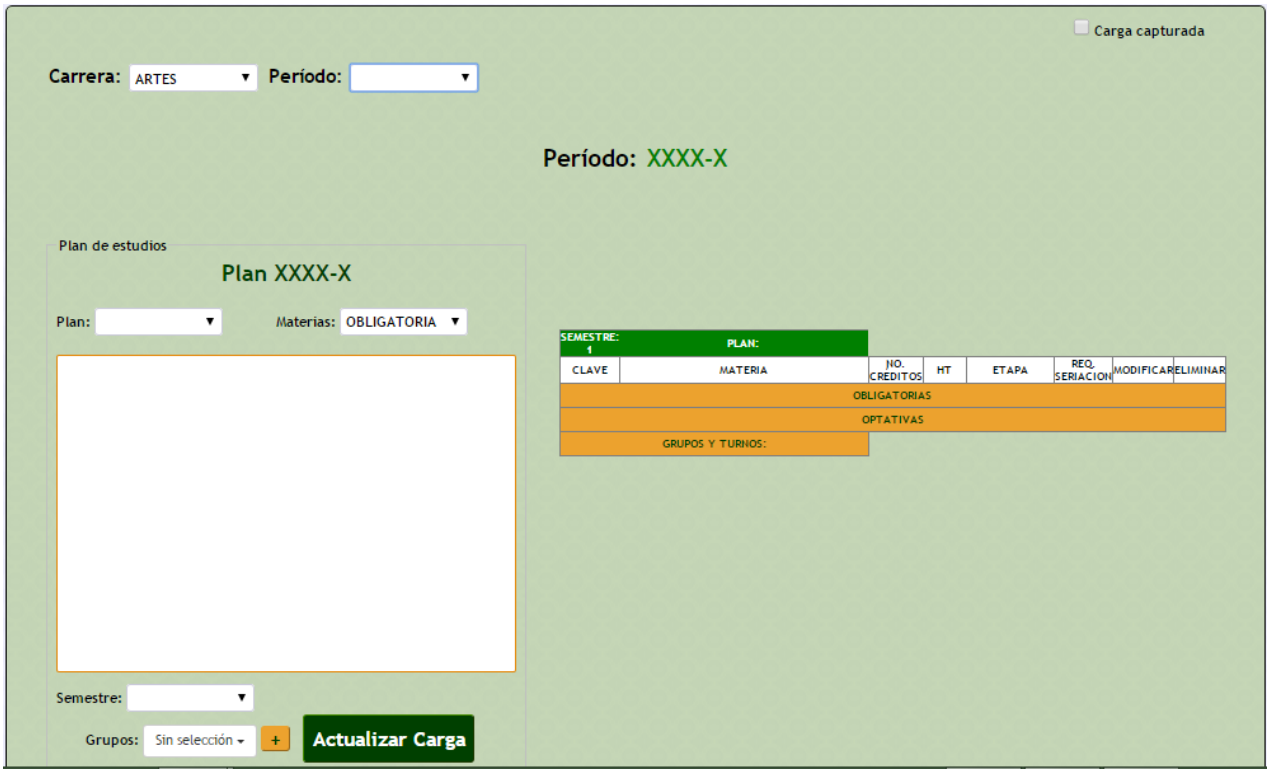


Figura 48. Interfaz de Carga Académica Subsecuente. Fuente (Duarte, SIGAF).

La interfaz de carga subsecuente muestra una interfaz reducida de registro subsecuente, básicamente es una modificación de una carga ya realizada, posee los mismos controles más el añadido de la copia de la carga y un panel de modificación en el lado derecho.

3.- La interfaz de consulta de Carga Académica (Ver figura 49):

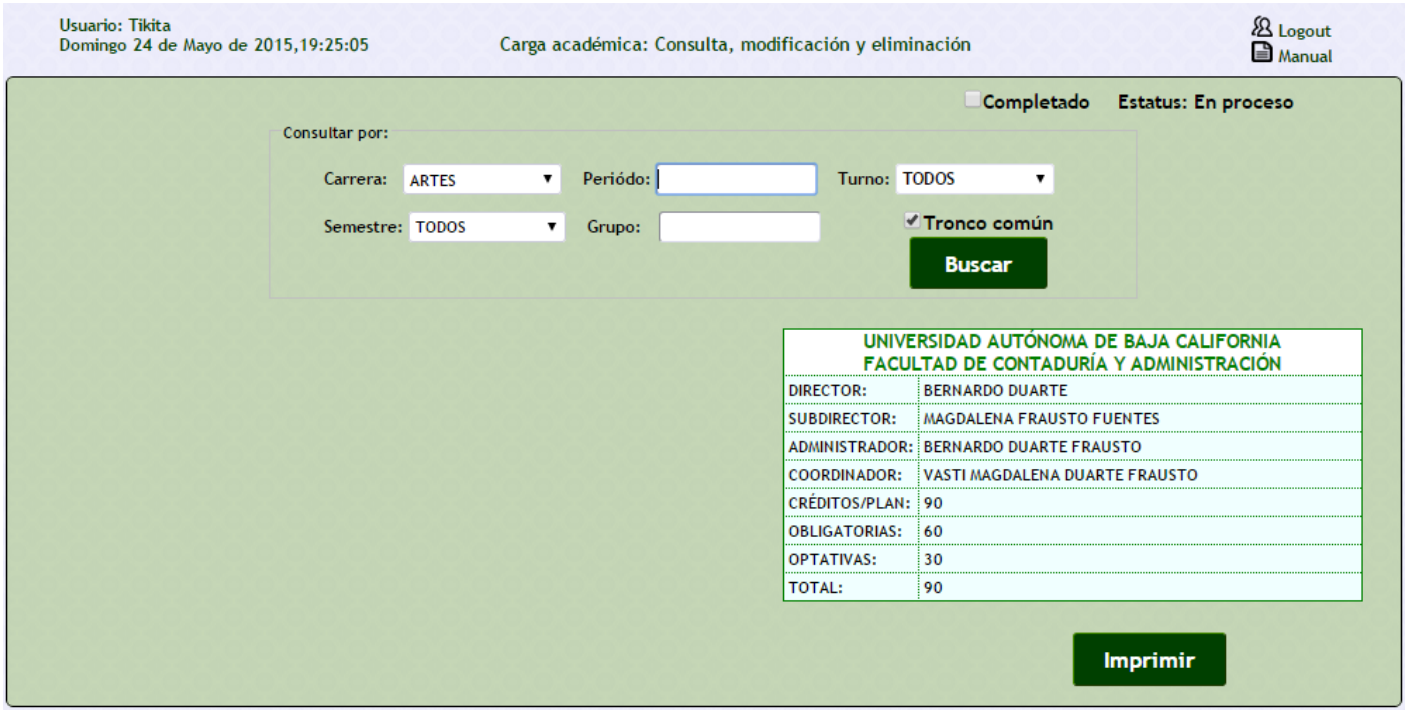


Figura 49. Interfaz Consulta de Carga Académica. Fuente (Duarte, SIGAF).

En consistencia con la consulta de registro de Plan de Estudios se contiene los controles para buscar por diferente criterio las cargas académicas registradas hasta el momento, filtros por carrera, periodo, turno, semestre y grupo y la posibilidad de mostrar las cargas sin Tronco Común.

Una vez verificado los insumos y las relaciones entre ellos procedemos a la codificación del módulo ya como tal.

4.4.3 Fase de codificación y desarrollo

Para desarrollar la codificación se enlistan los modelos creados, los controladores y las vistas que contendrán toda la lógica de programación del sistema de carga subsecuente (figura 50).

1. Modelos de Carga Académica:
    a. Turno.
    b. User.
    c. Level.
    d. Periodo.
2. Métodos del Controlador de Carga Académica
    a. getRegistro: Carga los catálogos básicos en la interfaz de registros de Carga Académica.
    b. getSubsecuente: Carga los catálogos básicos para la interfaz de Carga Subsecuente.
    c. getConsulta: Carga los catálogos básicos para la interfaz de Consulta de Carga Académica.
    d. postUltimoperiodo: Obtiene el último periodo registrado en una Carga Académica.
    e. postRegistrarperiodo: Registra un periodo para dar de alta una Carga Académica.
    f. postRegistrargrupo: Registra un nuevo grupo en un periodo.
    g. postObtenernombreprograma: Obtiene el nombre de un programa educativo.
    h. postObtenerprogramasadmin: Obtiene una lista de todos los programas educativos dados de alta en un programa.
    i. postObtenerplanes: Obtiene los 2 últimos planes de estudio registrados en el sistema.
    j. postObtenergrupos: Obtiene los grupos registrados en un periodo.
    k. postObteneruas: Obtiene las unidades de aprendizaje de un plan de estudio.

l. postRegistrarcarga: Registra y da de alta una carga académica con las unidades de aprendizaje del Plan de Estudios.
m. postObtenergruposua: Obtiene los grupos de una unidad de aprendizaje registrados en una carga académica.
n. postFormateargruposturnos: Formatea un grupo para que muestre el turno al que pertenece.
o. postEliminaruacarga: Elimina una unidad de aprendizaje registrada en una carga académica.
p. postEliminarcarga: Elimina una carga académica registrada.
q. postObtenercarga: Obtiene una carga académica específica por periodo.
r. postActualizargrupos: Actualiza los datos de un grupo registrado.
s. postCopiarcarga: Copia la carga académica anterior a un periodo nuevo.

3. Vistas de Carga Académica:
    a. Registro.
    b. Carga Subsecuente.
    c. Consulta.

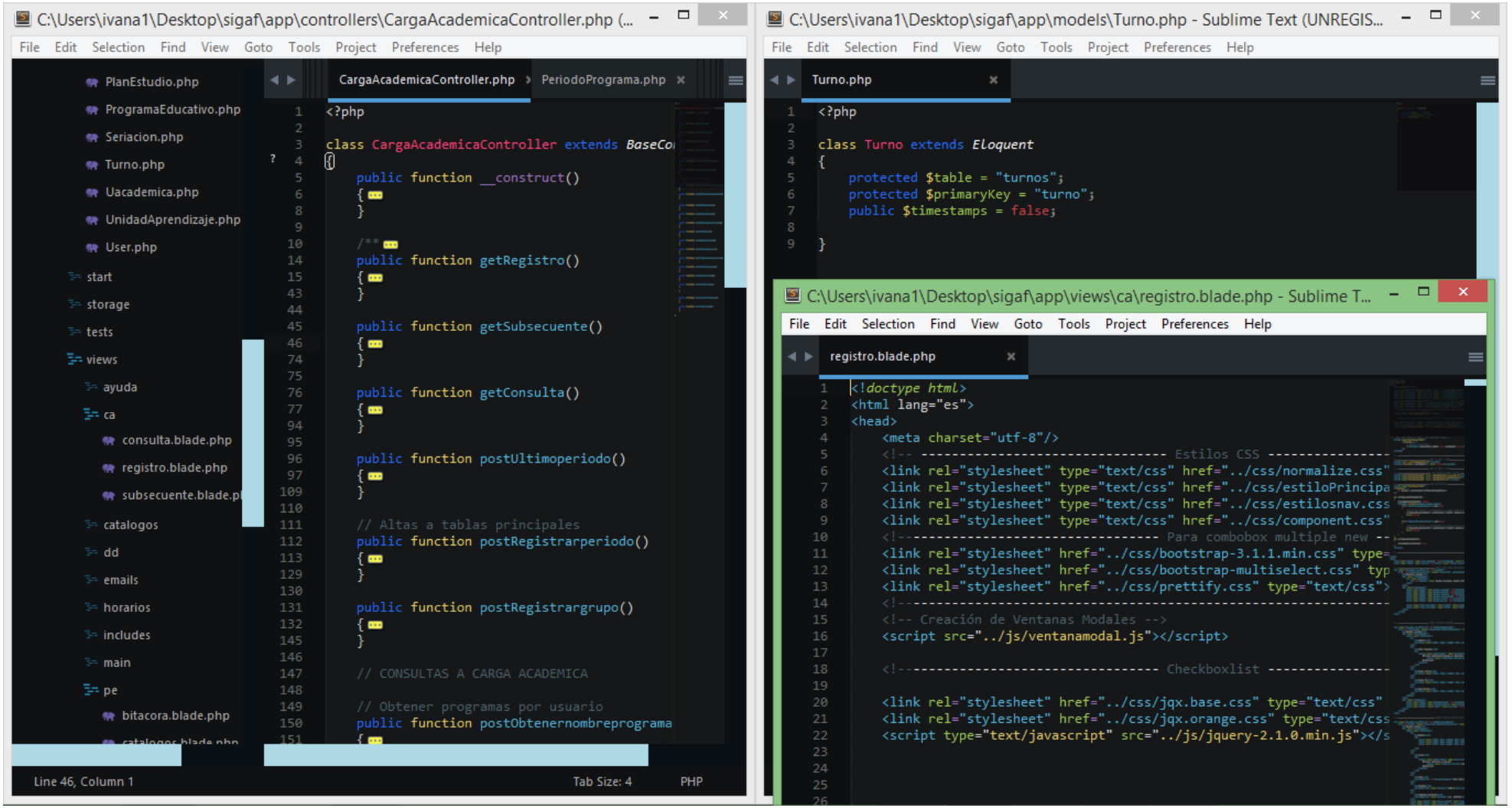


Figura 50. Modelos, Controlador y Vistas de Carga Académica. Fuente (Avila, Duarte, SIGAF)

4.3.4 Fase de publicación

Una vez terminada la codificación se publica el módulo en las plataformas correspondientes de Carga académica, en Basecamp, Github y Rally.

Se hace referencia también en BaseCamp las librerías que se utilizaron y la documentación para que el equipo esté enterado que se hizo uso de componentes de terceros (Ver figura 51).

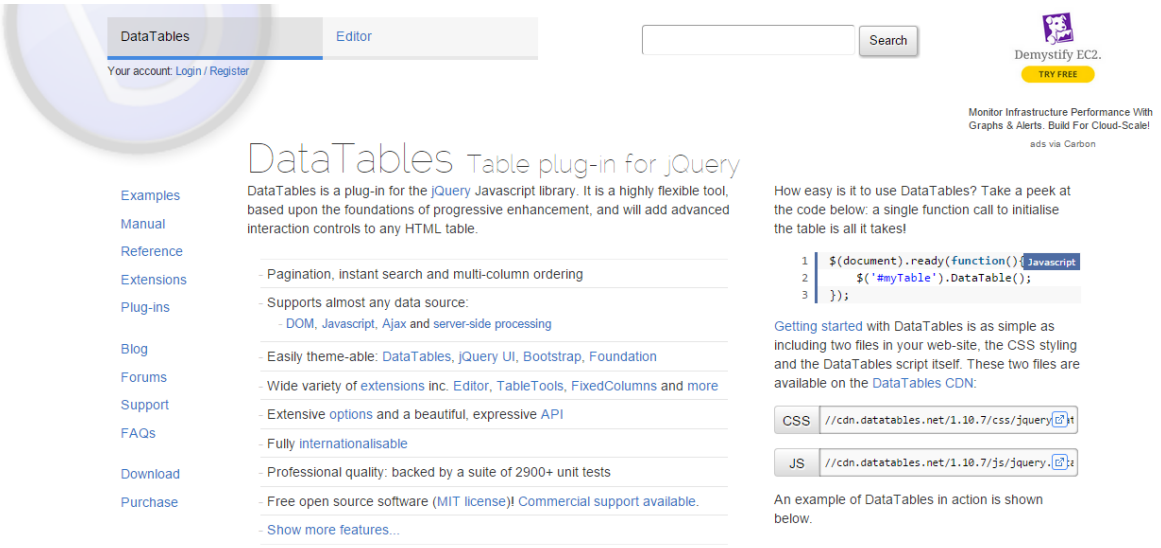


Figura 51. Componentes de Terceros. Fuente https://www.datatables.net

4.4.5 Fase de retroalimentación

Como en los casos anteriores se realiza una junta con el equipo para retroalimentar el avance y se hacen los issues correspondientes en Github para ver el avance de los milestones (Ver figura 52).

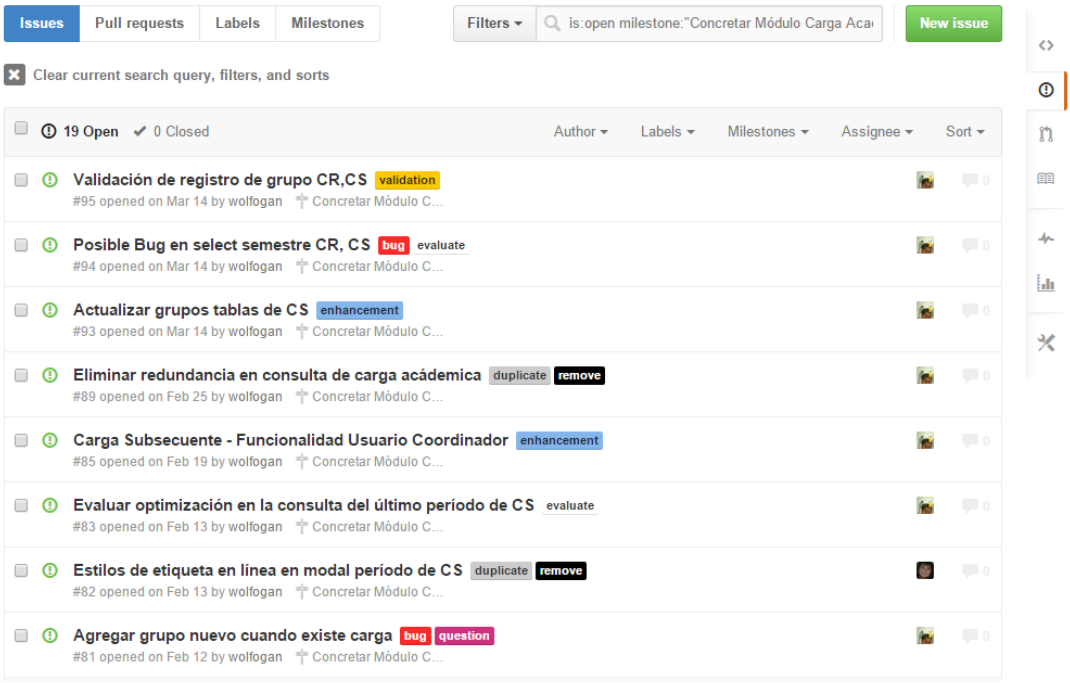


Figura 52. Milestones de Carga Académica. Fuente Propia.

4.5 Módulo 3 – Disponibilidad Docente

El módulo de carga de disponibilidad docente contempla la creación y adición del módulo de generación y administración de usuarios de la aplicación, de este modo los usuarios docentes pueden gestionar su disponibilidad mediante el portal y permite a los administradores, consultar, agregar y gestionar a los docentes para el aprovechamiento de la creación de horarios. Para el registro, consulta y gestión de la disponibilidad docente, se desarrollaron las siguientes interfaces las cuales al término del desarrollo ejecutan el objetivo principal del módulo en su conjunto, las interfaces son las siguientes:

1. Registro, consulta y modificación de usuarios.
2. Registro de Disponibilidad Docente.
3. Consulta de Disponibilidad Docente (Ver tabla 7).

4.5.1 Fase de análisis de insumos

Para iniciar con la codificación del módulo se procede a estudiar los diagramas de caso de uso, diagramas conceptuales, secuencia y de proceso, tanto de usuarios como de disponibilidad docente:

Tabla 7. Descripción de caso de uso (Login de Usuarios).
Fuente (Castillejos, SIGAF).

Identificación del requerimiento:	RF-PE-01
Nombre del Requerimiento:	Autentificación de Usuario.
Caracteristicas:	Los usuarios deberán identificarse para acceder a cualquier parte del sistema.
Descripción del requerimiento:	• El sistema podrá ser consultado por cualquier usuario (con privilegios autorizados) dependiendo del módulo en el cual se encuentre y su nivel de accesibilidad y según su nivel de usuario desplegara las opciones a las que tiene acceso. • Cada usuario solo puede tener una sola sesión abierta, en caso de que abra otra sesión debe enviar una notificación de que existe sesión abierta y se inicie nuevamente en el tiempo para que se cierre la otra sesión que será de por lo menos 15 minutos.
Requerimiento NO funcional:	• RNF-PE-01 • RNF-PE-02 • RNF-PE-05 • RNF-PE-08
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF-PE-02
Nombre del Requerimiento:	Registrar Usuarios.
Caracteristicas:	Si el usuario tipo (docente_coor, docente_asig) no esta dado de alta, debe tener la posibilidad de poder registrarse en el sistema para acceder a las opciones en base a sus lineamientos. Los usuarios Administrador y Auxiliar administrativo los dará de alta el administrador general del sistema.
Descripción del requerimiento:	• El usuario tipo docente_coor, docente_asig debe suministrar como datos mínimos : no_empleado, nombre, apellido paterno, apellido materno, correo valido, Usuario y Password. • Con restricciones y lineamientos de fortaleza de la contraseña con una longitud de 6 a 8 caracteres con uso de minúsculas, mayúsculas y caracteres especiales sin aceptar espacios en blanco • También debe tener la posibilidad de registro con Captcha. •
Requerimiento NO funcional:	• RNF-PE-01 • RNF-PE-02 • RNF-PE-05 • RNF-PE-08
Prioridad del requerimiento: Alta	

Los casos de uso describen las siguientes acciones que el usuario puede realizar en el sistema:

1. Login de usuario por privilegios.
2. Accesos a módulos por privilegios de usuarios.
3. Alta, Consulta, Modificación y Eliminación de usuarios.
4. Administración de privilegios de usuario.
5. Alta de disponibilidad docente.

6. Consulta y modificación por parte de administradores de la disponibilidad docente. (Ver tabla 7).

4.5.2 Fase de emparejamiento

En el caso del módulo de Disponibilidad Docente no se tiene dependencias de otro módulo por lo que el proceso de emparejamiento es relativamente menos complejo que el de los módulos anteriores, por lo contrario, el módulo descrito es entrada para la creación de horarios.

Las acciones para cotejar y emparejar la información se describen a continuación:

1. Verificar que los controles de la interfaz cumplan con los requerimientos de usuario y los casos de uso descritos en el análisis de insumos (Ver figura 53).

Figura 53. Interfaz CRUD Usuarios. Fuente (Duarte, SIGAF).

2. Verificar que el diagrama de base datos describa la persistencia de los datos que se almacenaran en disponibilidad docente (Ver figura 54).

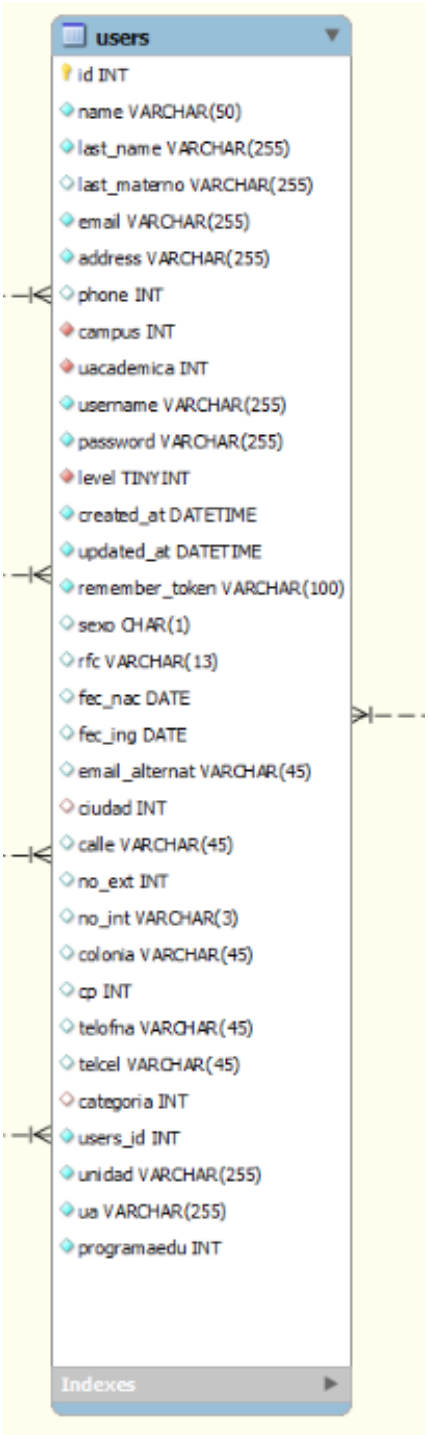


Figura 54. ER Físico Usuarios.
Fuente (Gambino, SIGAF).

3. Se verifica que la interfaz y la BD contengan correlación con los datos que manipulan, en la interfaz que se validen y se visualicen los datos en un orden especifico y en la BD que se pueda almacenar y consultar dicha información (Ver figura 55).

Figura 55. Grilla de visualización de usuarios. Fuente (Avila, Duarte, SIGAF).

Para el caso de disponibilidad docente su dependencia de la información que contiene los usuarios es importante ya que las interfaces deben mostrar información precargada de usuarios con el fin de que los docentes se centren en subir su formación académica y disponibilidad para impartir materias en determinado periodo de tiempo.

4. Se verifica la interfaz de registro de disponibilidad para la pre-carga de los datos (Ver figura 56).

Figura 56. Precarga de usuarios en disponibilidad. Fuente (Avila, Duarte, SIGAF).

5. Una vez verificada la interfaz de disponibilidad, se realiza una corrida de prueba entre BD y la interfaz para correlacionar los datos y asegurar que se cuenta con toda la información necesaria y requerida. Como el ejemplo de la figura 57:

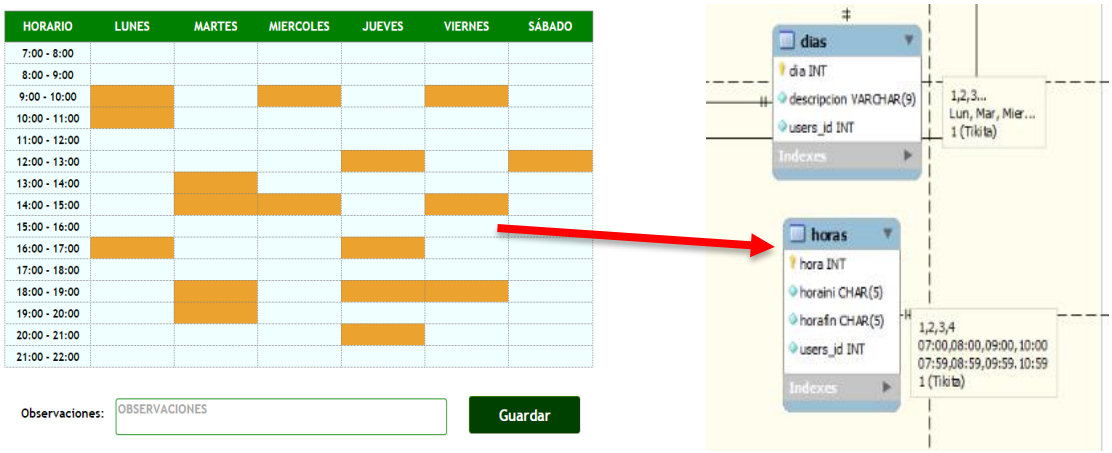


Figura 57. Correlación Interfaz – BD. Fuente (Duarte, Gambino, SIGAF).

4.5.3 Fase de codificación y desarrollo

En el desarrollo de la codificación del módulo de Disponibilidad Docente se desarrolló el módulo de Login y Usuarios, ya que la arquitectura describe codificar los modelos para su uso en los controladores se procede a su desarrollo. La figura 58 presenta el modelo para usuarios.

1. Modelos de Usuarios y Disponibilidad Docente:
    d. User.
    e. Categorias.
    f. Pais.
    g. Estado.
    h. Carreras.
    i. Universidades.

j. TiposCurso.

k. Horas.

l. Dias.

m. Level.

```php
<?php

use Illuminate\Auth\UserInterface;
use Illuminate\Auth\Reminders\RemindableInterface;

class User extends Eloquent implements UserInterface, RemindableInterface {

    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'users';

    /**
     * The attributes excluded from the model's JSON form.
     *
     * @var array
     */
    protected $hidden = array('password');

    /**
     * Get the unique identifier for the user.
     *
     * @return mixed
     */
    public function getAuthIdentifier()
    {
        return $this->getKey();
    }

    /**
     * Get the password for the user.
     *
     * @return string
     */
    public function getAuthPassword()
    {
        return $this->password;
    }
```

Figura 58. Modelo User. Fuente Propia.

2. Métodos del Controlador de Carga Académica:
    a. getRegistro: Obtiene la precarga de los datos de usuario y los muestra en registro de disponibilidad docente.
    b. getConsulta: Consulta y modifica disponibilidad de docentes (usuarios administradores).
    c. postEstados: Obtiene los estados registrados del país.
    d. postCiudades: Obtiene las ciudades de los estados en un país determinado.
    e. postCarrerasEmp: Obtiene las carreras registradas en la BD.
    f. postRegistrardatospersonales: Actualiza los datos personales de un usuario.
    g. postRegistraestudioscursos: Registra los estudios y cursos realizados por un docente.
    h. postRegistrardisponibilidad: Registra la disponibilidad de un docente.
    i. postRegistrarpuesto: Obtiene los grupos registrados en un periodo.
    j. postRegistrarcarreras: Obtiene las unidades de aprendizaje de un plan de estudio.
    k. postRegistraruniversidad: Registra y da de alta una carga académica con las unidades de aprendizaje del Plan de Estudios.
    l. postRegistrarcursos: Obtiene los grupos de una unidad de aprendizaje registrados en una carga académica.
3. Vistas de Disponiblidad Docente:
    a. Registro y Modificación.
    b. Consulta.

Para visualizar la información sin recargas en la página se programa mediante la tecnología AJAX mediante los métodos abreviados de jQuery se obtiene la información necesaria llamando a los métodos antes descritos mediante invocaciones asíncronas, la figura 59 presenta un ejemplo de llama AJAX en disponibilidad docente.

```
$("#dd_pais").on("change",function(){
    var pais = $(this).val();
    //alert(pais);
    $.ajax({
        url : "<?php echo URL::to('disponibilidaddocente/estados') ?>",
        method : "POST",
        data: {pais:pais}

    })
    .done(function(estados){
        //console.log(data);
        var options = "";
        for(x in estados)
        {
            options += "<option value='"+ estados[x].estado +"'>"+ estados[x].descripcion +"</option>"
        }

        $("#dd_estado").html(options);
    })
    .fail(function(error,texterror,errorthrow){
        alert("Error : ", error.responseText);
    });
});
```

Figura 59. Método AJAX que carga los Estados de manera asíncrona en la página. Fuente Propia.

Al término del desarrollo se tiene la estructura MVC en Laravel del módulo con la funcionalidad descrita en el análisis de Disponibilidad Docente y la funcionalidad de la interfaz interactuando con la base de datos.

4.5.4 Fase de publicación

Una vez terminada la codificación se publica el módulo en las plataformas correspondientes de Disponibilidad docente, en Basecamp, Github y Rally (Ver figura 60).

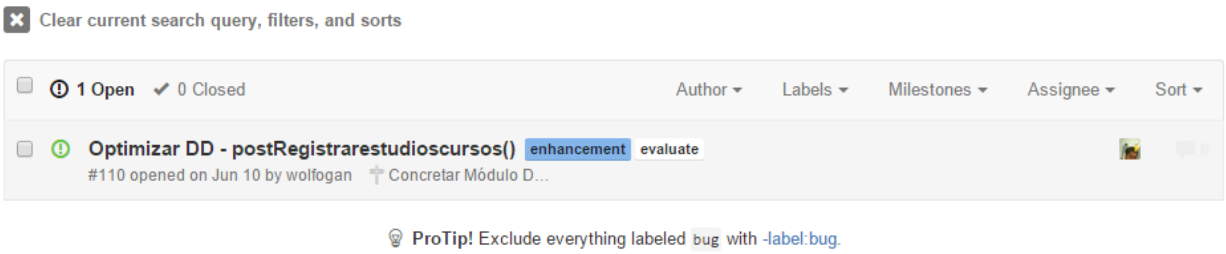


Figura 60. Commits Disponibilidad Docente. Fuente https://github.com/wolfogan/sigaf/commits/master.

4.5.5 Fase de retroalimentación

Se definen los milestones para el módulo de usuarios y disponibilidad, se actualiza Basecamp y se retroalimenta en Rally para seguimiento del proyecto (Ver figura 61).

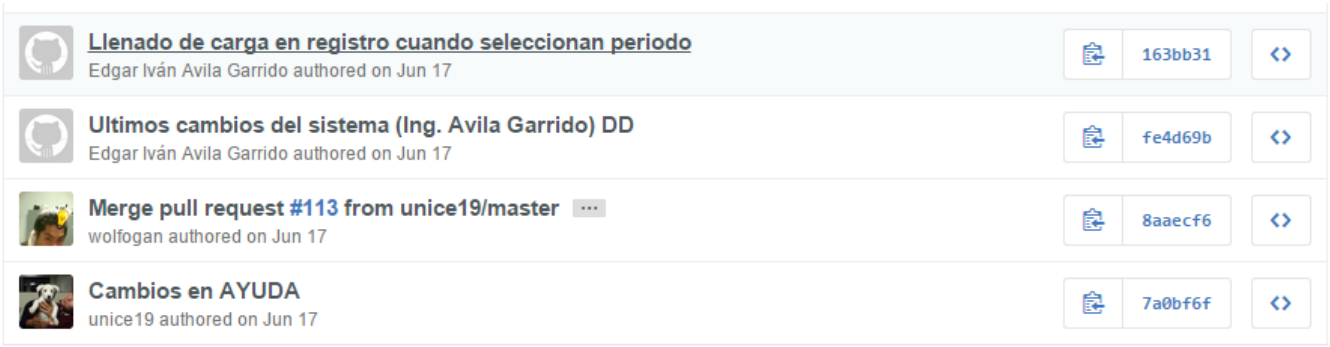


Figura 61. Milestone Disponibilidad Docente. Fuente https://github.com/wolfogan/sigaf/milestones.

Capítulo V

Resultados

Capítulo V – Resultados

En el siguiente apartado comprende los resultados de la codificación de los tres módulos base del sistema SIGAF y el desarrollo de sus dependencias, una vez aplicada la metodología SCRUM el código generado es mantenerle, legible para desarrollos, mantenimientos y actualizaciones futuras.

5.1 Resultados de Plan de Estudios

Una vez desarrollado las fases de la metodología, y teniendo la retroalimentación del equipo de los problemas encontrados en la funcionalidad, el código en ejecución cumple con la funcionalidad requerida descrita en el análisis, y el usuario final puede interactuar con los módulos mediante los controles de la interfaz, se anexa el código para este resultado.

El código generador provoca que a través de la interfaz el usuario pueda:

- Registrar un Plan de Estudios
- Crear unidades aprendizaje en los planes de estudio.
- Consultar los planes de estudio y las unidades aunadas ahí.
- Modificar los atributos de las unidades de aprendizaje.
- Eliminar de los planes de estudios una unidad de aprendizaje.
- Asignar unidades de aprendizaje a diferentes programas educativos.
- Agregar coordinaciones, números de plan de estudios, etapas y demás catálogos relacionados.
- Consultar las unidades de aprendizaje de un plan y visualizar en forma de listado cada una.
- Consultar los diferentes planes de estudios por carrera, etapa, unidad, tronco común y coordinación.

A continuación se visualiza la interfaz con la funcionalidad ejecutada por el código (Ver las figuras 62 y 63).

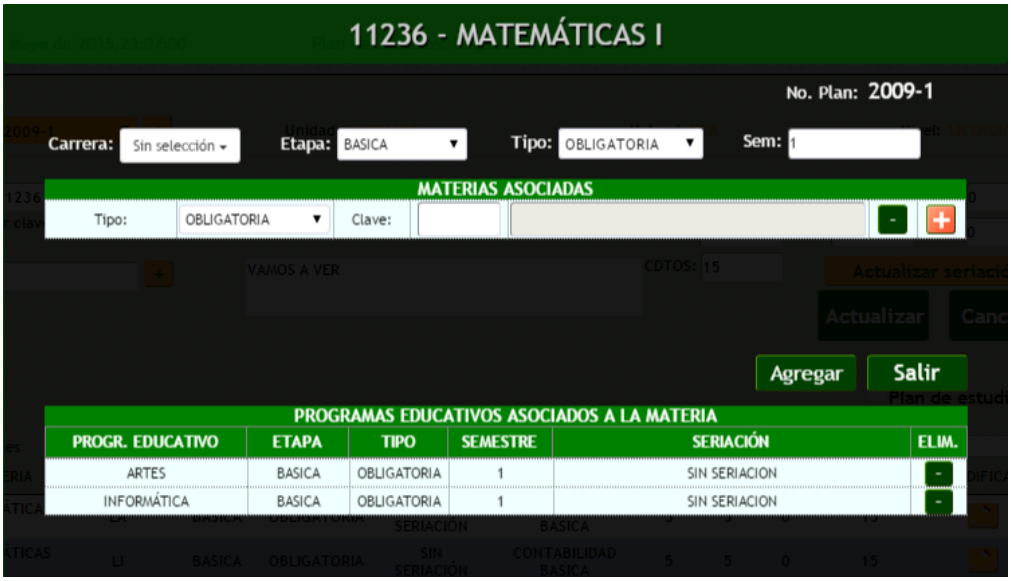


Figura 62. Asociación de las Unidades. Fuente (Avila, Duarte, SIGAF).

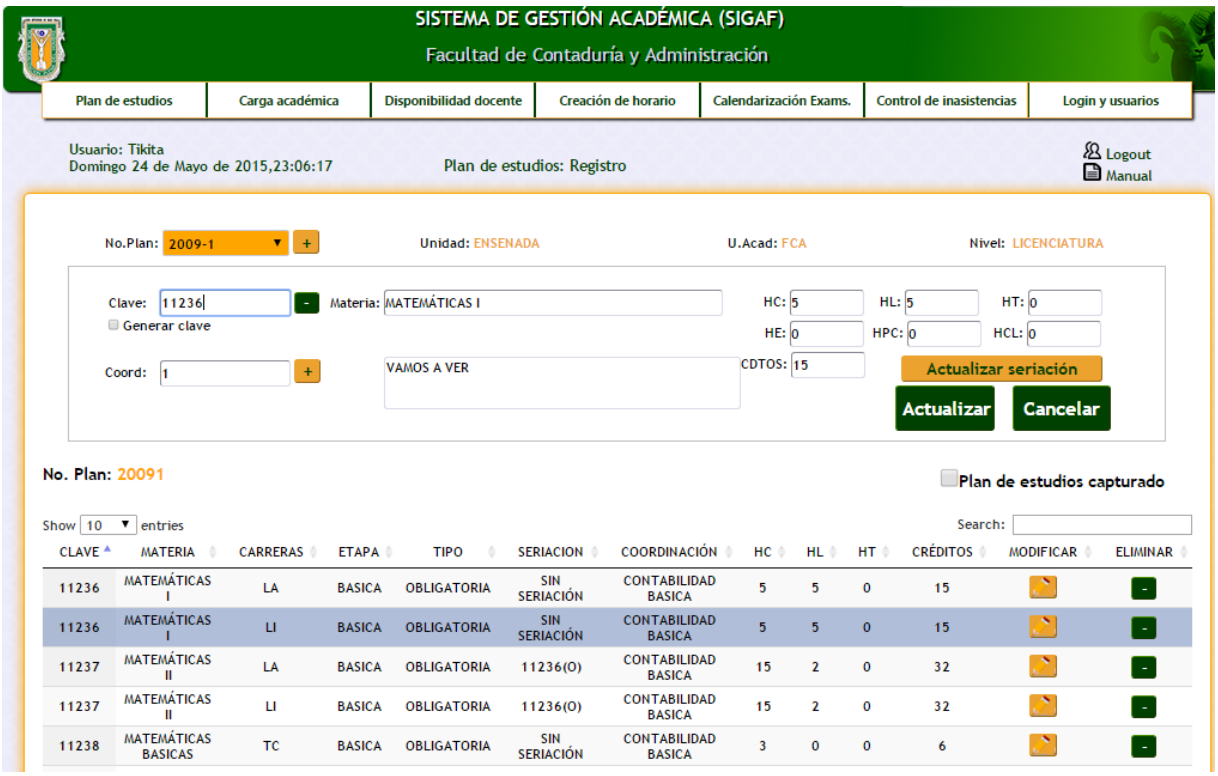


Figura 63. Funcionalidad generada por el código generado. Fuente (Avila, Duarte, SIGAF).

Figura 64. Atributos de Unidad de Aprendizaje. Fuente (Duarte, SIGAF).

De la misma manera la funcionalidad genera que se muestran las unidades de los planes de forma gráfica como un mapa conceptual, y la actualización individual de cada una de ellas (Ver figura 64).

5.2 Resultados de Carga Académica

Al concluir el código generado de Carga Académica el usuario mediante la interfaz puede realizar las siguientes operaciones que responden a los requisitos de módulo de Carga Académica se anexa el código para este resultado.

- Registrar un periodo para dar de alta una carga académica.
- Registrar las unidades de aprendizaje dentro de una carga académica.
- Asignar las unidades al semestre y grupos necesarios dentro de la carga.
- Dar de alta grupos en el periodo y plan de estudios asignado.
- Consultar las cargas registradas por diversos criterios.

- Copiar una carga anterior y generar una nueva a partir de ella.
- Visualizar la carga en semestres, grupos y tipo de unidad de aprendizaje.
- Visualizar reportes de las cargas.
- Separar la funcionalidad en base a los permisos del usuario.

Funcionalidad requerida mediante la interfaz (Ver figura 65):

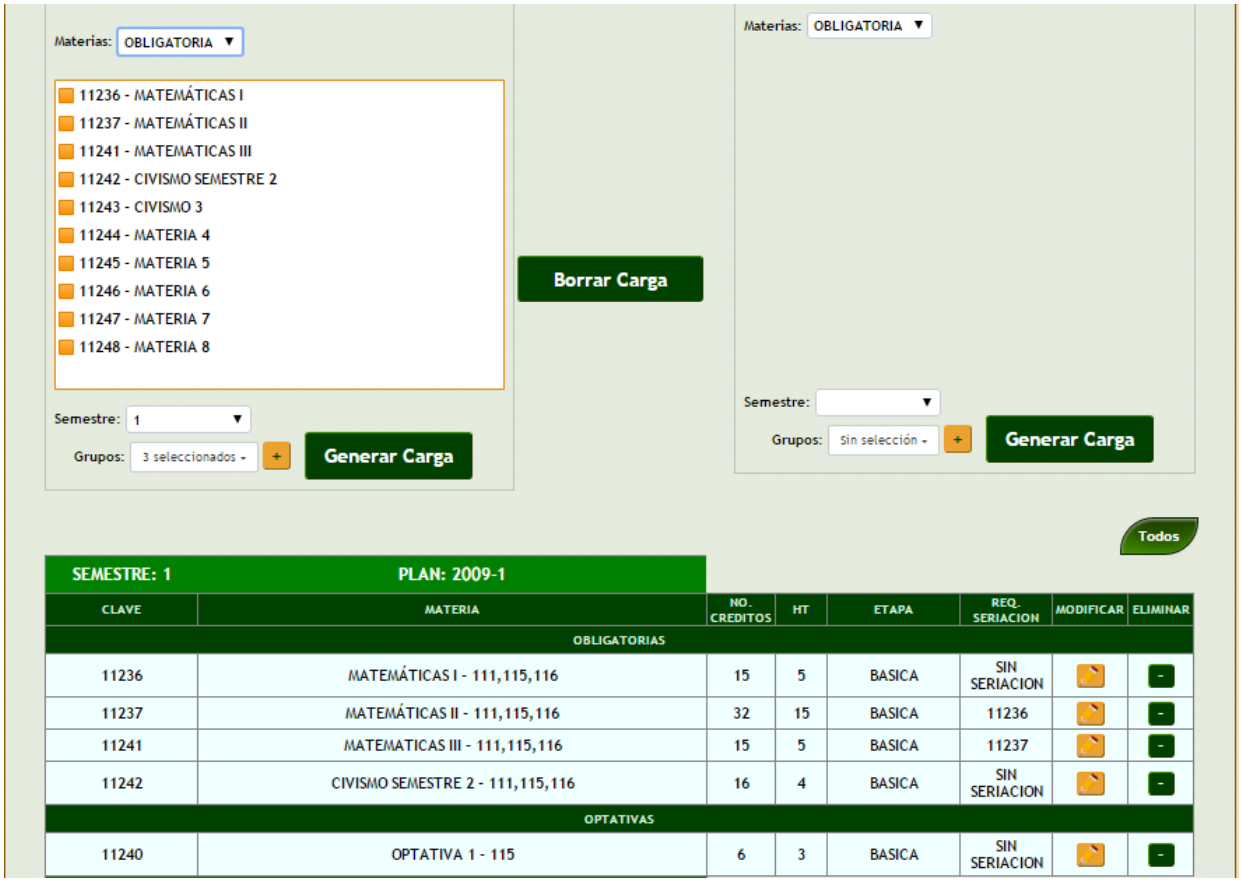


Figura 65. Registro de Carga Académica. Fuente (Avila, Duarte, SIGAF).

5.3 Resultados de Disponibilidad Docente

Al finalizar la codificación de las rutas, modelos, controladores y vistas del módulo de Disponibilidad Docente, resulta en una interfaz funcionalidad con las especificaciones y requerimientos de usuario emitidos para el módulo de Disponibilidad Docente.

El usuario con perfil de docente puede realizar las siguientes funcionalidades dentro del sistema:

1. Registrar y actualizar datos personales, subir y descargar evidencias que acreditan su actividad profesional, el control de la administración de los cursos, congresos y las actividades académicas en las cuales ha tenido participación
2. Registrar y consultar su disponibilidad docente por periodo para registrar y poder impartir algunas de las unidades de aprendizaje disponibles dentro de la creación de horarios.
3. Registrar que unidades de aprendizaje ha impartido dentro de la institución.
4. Puede solicitar nuevos cursos en los cuales desea participar para incrementar la calidad de la enseñanza de alguna unidad de aprendizaje de interés.
5. Los Administradores puede consultar y actualizar de ser necesario las disponibilidades de cada docente.
6. Los administradores pueden agregar o quitar datos personales, historial académico y disponibilidades de los docentes registrado.
7. Los administradores pueden ser docentes y de igual manera registrar su disponibilidad docente en un periodo determinado.

Funcionalidad requerida para el módulo de Disponibilidad Docente (Ver figura 66 y 67):

Figura 66. Datos personales del docente. Fuente (Duarte, SIGAF).

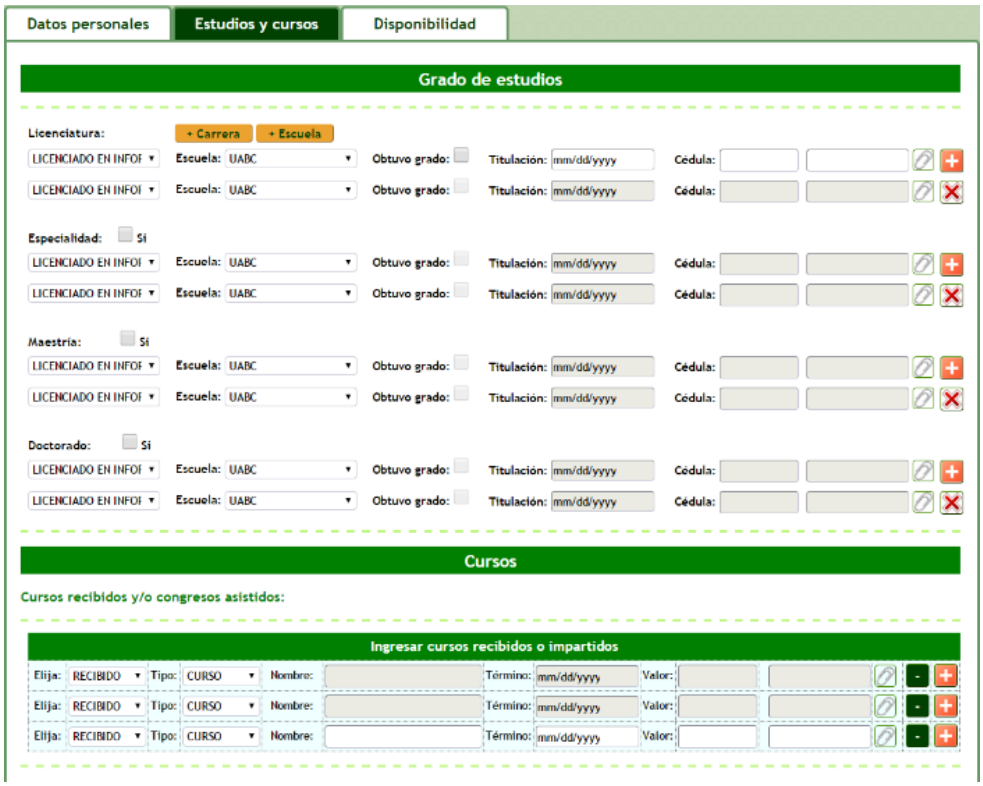


Figura 67. Interfaz Disponibilidad Docente. Fuente Duarte, SIGAF).

5.4 Resultados de Login y Usuarios

Para el desarrollo de disponibilidad docente fue necesario realizar el módulo de usuarios para poder generar los usuarios docentes por parte de los administradores.

El módulo se concreta a las siguientes funcionalidades:

1. Registro, consulta y modificación de usuarios.
2. Asignación de privilegios por rol.
3. Encriptación de contraseñas de usuarios.
4. Envió de e-mail a usuarios nuevos.
5. Búsqueda avanzada de usuarios.

Interfaz funcional de usuarios (Ver figura 68):

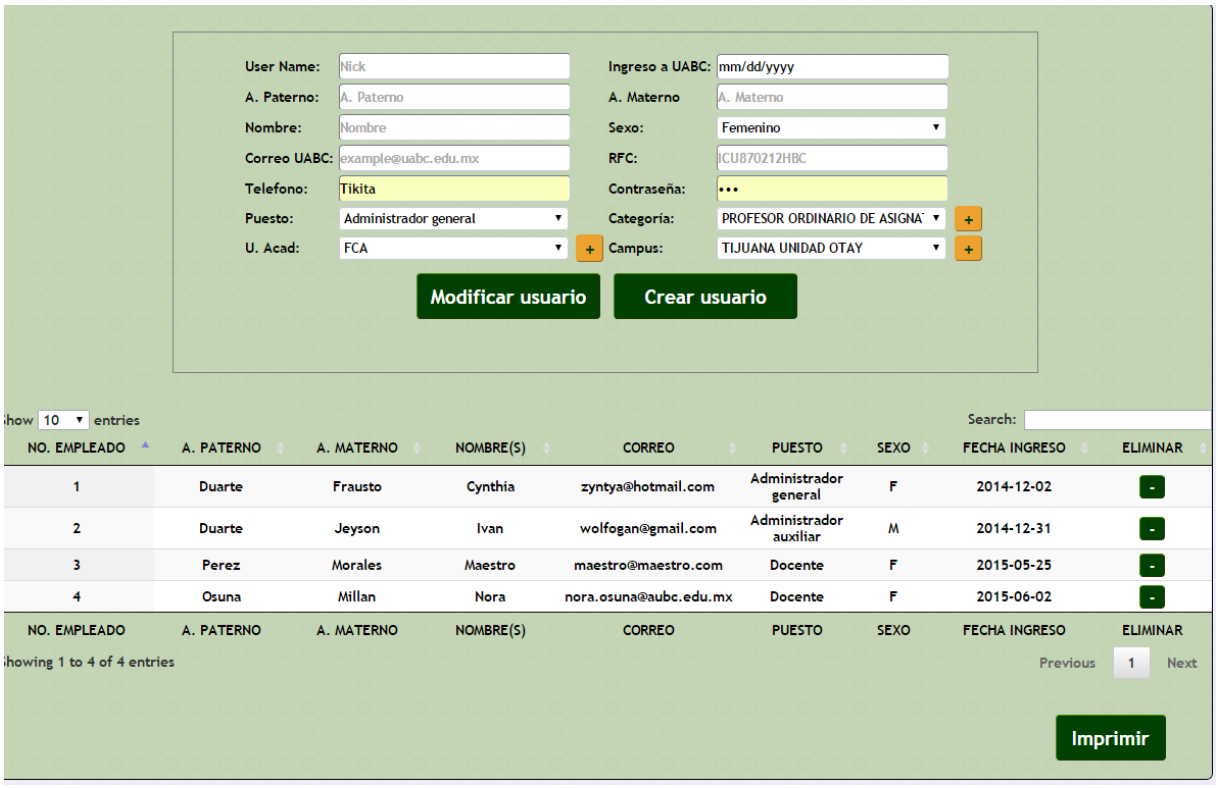


Figura 68. Interfaz Usuarios. Fuente (Avila, Duarte, SIGAF).

5.5 Resultados generales de la arquitectura

Durante el desarrollo de la aplicación se desarrolló un API para retornar información en formato JSON a una petición HTTP, esto con el fin de reutilizar la información en cualquier lenguaje o plataforma que no sea SIGAF.

El framewok Laravel mapea todas la tablas de la BD a clases para tener un entorno de desarrollo más limpio y no mezclar consultas sql en el código de producción. También al termino del desarrollo se concluye con una estructura clara y concisa que no permite visualizar los elementos de programación como fragmentos de código individuales, capaz de sufrir modificaciones sin alterar el flujo de la aplicación.

La interactividad de la aplicación fue desarrollada con Javascript y jQuery para solucionar de forma práctica la dinámica entre usuario-sistema. Al término de los módulos el sistema quedó con la siguiente estructura que permite toda la funcionalidad requerida en el análisis:

Controladores de la aplicación (Ver figura 69):

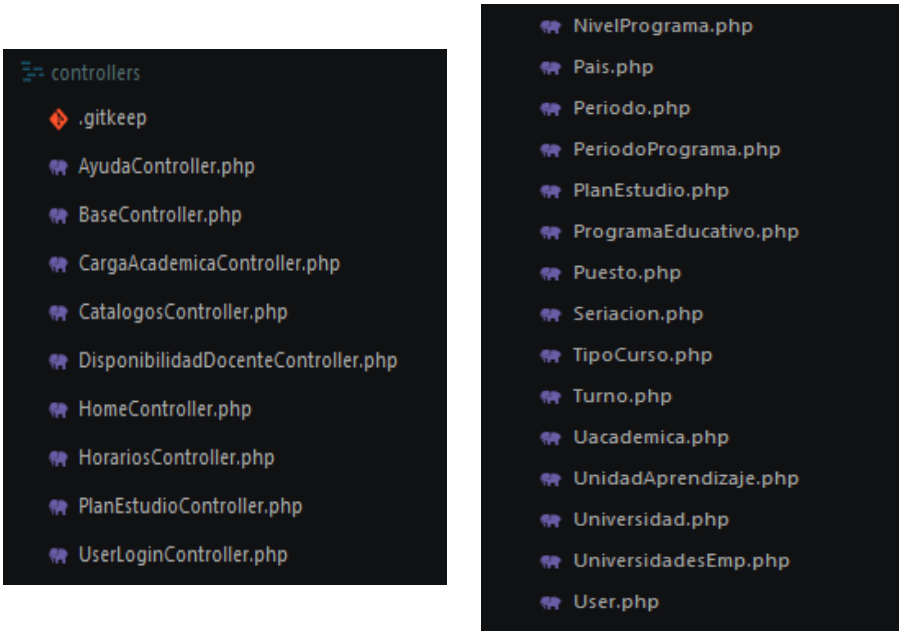


Figura 69. Controladores de SIGAF. Fuente Propia.

Modelos de la aplicación, figura 70:

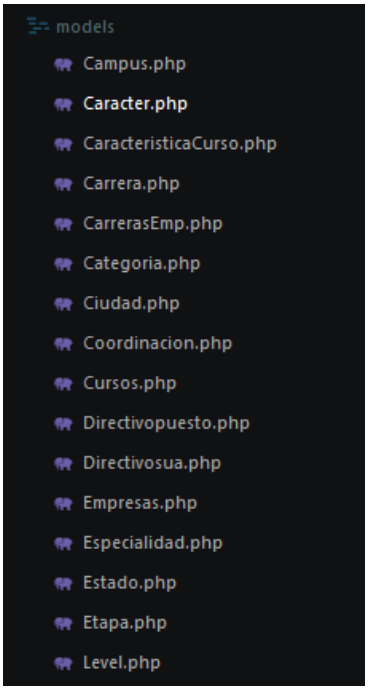


Figura 70. Modelos SIGAF. Fuente Propia.

Vistas de la aplicación, figura 71:

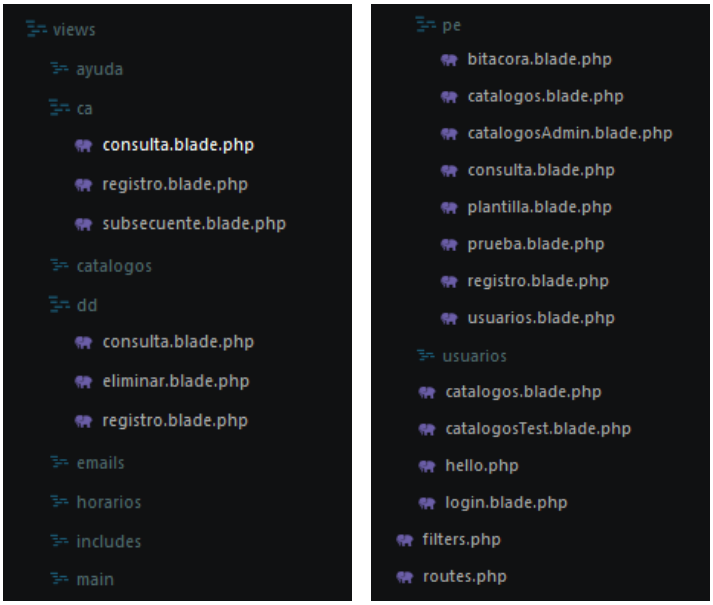


Figura 71. Vistas de la aplicación. Fuente Propia.

Capítulo VI

Conclusiones y Recomendaciones

6.1 Conclusiones

Durante el desarrollo de la aplicación se construyó un API (Interfaz de Programación de Aplicaciones) para retornar información en formato JSON (Notación Objeto de JavaScript) a una petición HTTP mediante el framework Laravel y tecnologías como Javascript, HTML5, CSS3, esto brinda a futuros desarrollos del sistema SIGAF la posibilidad de crear diferentes aplicaciones para diversos fines, con el añadido de utilizar cualquier tecnología Frontend para consumir los datos. Esto es posible gracias al equipo completo de desarrollo y a las habilidades de cada miembro.

Como oportunidad de aprendizaje es necesario recalcar que la integración del equipo de desarrollo fue fundamental, el presente trabajo solo forma una parte del sistema en su conjunto y es el resultado de la construcción de otras áreas como la creación de la base de datos, Frontend, deploy de la aplicación y es un esfuerzo conjunto de diversas áreas del desarrollo, por ello la comunicación entre los miembros del equipo es vital y necesaria, las metodologías describen un área de oportunidad muy grande para facilitar la creación de un sistema, sin embargo, el que ejecuta dicha metodología son personas por lo que la convivencia es necesaria entre los miembros del equipo, durante el desarrollo de los módulos se presentaron cambios radicales por parte del cliente que repercutieron en el avance del mismo, sin embargo, la comunicación hizo posible solventar dicha problemática.

Los módulos realizados cumplen con las expectativas del proyecto y son la base para la creación de horarios del sistema SIGAF.

6.2 Recomendaciones

Un proyecto de desarrollo en manos de un programador es diversión siempre y cuando sepa utilizar las herramientas de manera correcta y es necesario constancia en el trabajo, la convivencia con el equipo es requisito para que el trabajo se vuelva recreativo y ameno.

La opinión de expertos es valorada para el tipo de trabajo donde no existe mucha documentación ya que las tecnologías son nuevas, la experiencia triunfa mientras que se estandariza la tecnología, recomiendo la lectura de tópicos de tecnología de la gente que la crea y por supuesto dar crédito a dichas personas ya que logran que la programación sea una experiencia divertida y entretenida durante los proyectos.

Para el Sistema SIGAF se recomienda seguir construyendo un API más robusta y completa que permita no solo la creación de horarios, sino un sistema de gestión integral para la Facultad de contaduría y administración, se puede automatizar y simplificar la codificación con las nuevas herramientas que están saliendo a la luz y en su defecto aumentar la experiencia de usuario de las aplicaciones que giran en torno al desarrollo del sistema SIGAF.

Bibliografía

Tim Hentenaar's Blog. (s.f.). Obtenido de Benchmark: PHP vs. Python vs. Perl vs. Ruby,: http://hentenaar.com/serendipity/index.php?/archives/27-Benchmark-PHP-vs.-Python-vs.-Perl-vs.-Ruby.html

Abc, D. (10 de Febrero de 2011). *Definición de Navegador*. Recuperado el 10 de Marzo de 2014, de http://www.definicionabc.com/tecnologia/navegador.php

Alegsa. (5 de 12 de 2013). *Alegsa*. Recuperado el 16 de 11 de 2010, de Aplicaciones Web: http://www.alegsa.com.ar/Dic/aplicacion%20web.php

Applied Software Consultants. (2016). Recuperado el 10 de Agosto de 2013, de Asc Time Tables: http://www.asctimetables.com/

Blogspot. (5 de Noviembre de 2013). *De Profesor A Maestro*. Obtenido de 2013: http://deprofesoramaestro.blogspot.mx/2012/11/comparativa-de-lenguajes-web.html

Campus MVP. (22 de Enero de 2015). *Qué lenguajes de programación hay que dominar en 2015.* Recuperado el 20 de Marzo de 2015, de http://www.campusmvp.es/recursos/post/Que-lenguajes-de-programacion-hay-que-dominar-en-2015.aspx

Catalonia Software Solutions. (2016). Recuperado el 13 de Agosto de 2013, de ¿Que es AGH/iX?: http://gestionhorarios.catalonia.es/que-es-agh.html

Cohn, M. (24 de Febrero de 2014). *User Stories & Back-End Systems*. Recuperado el 8 de Julio de 2014, de Scrum Alliance: https://www.scrumalliance.org/community/spotlight/mike-cohn/february-2014/user-stories-back-end-systems?feed=Scrum-Alliance-Spotlight

Debian. (16 de Febrero de 2014). Obtenido de The computer language Benchmarks Game: http://shootout.alioth.debian.org/SharpDevelop.

Delgado, A. (16 de Mayo de 2011). *Anagaldo Blogspot*. Recuperado el 3 de Marzo de 2014, de http://anagaldo.blogspot.mx/2011/05/modelo-cliente-servidor.html

Duque, R. G. (2001). *Python para todos.* España: Creative Commons Reconocimient.

Garrido, J. S. (2004). Arquitectura y diseño de sistemas web modernos. *Revista de Ingeniería Informática del CIIRM*, 1-6.

Garrido, J. S. (10 de Julio de 2004). *Arquitectura y diseño de sistemas web modernos.* Recuperado el 20 de Marzo de 2014, de http://pegaso.ls.fi.upm.es/~sortega/html_css/files/Arquitectura_y_disenyo_de_sistemas_web_modernos.pdf

Genbetadev. (s.f.). *Infografia*. Obtenido de Infografía: Comparativa entre PHP, Ruby y Python : http://www.genbetadev.com/lenguajes-de-programacion/infografia-comparativa-entre-php-ruby-y-phython

Gilmore, W. J. (2015). *EASY LARAVEL 5.* LEANPUB.

Google. (20 de Abril de 2014). Obtenido de Google Developers: https://developers.google.com/speed/

Hearts, C. V. (15 de Marzo de 2012). *Lenguajes de Programación*. Obtenido de Maestros del Web: http://www.maestrosdelweb.com/editorial/los-diferentes-lenguajes-de-programacion-para-la-web-c106224l/

Herts, C. V. (8 de Febrero de 2011). *Maestros del web*. Recuperado el 5 de Abril de 2014, de http://www.maestrosdelweb.com/editorial/frontend-backend-duplicando-programacion/

Instituto Tecnólogico de Tijuana. (15 de Abril de 2015). Obtenido de TECTIJUANA: http://tectijuana.edu.mx/calendario-academico/

Karel Rodríguez Varona. (4 de Octubre de 2012). *Revista Cubana de Ciencias Informáticas.* Recuperado el 17 de Noviembre de 2014, de http://rcci.uci.cu/index.php?journal=rcci&page=article&op=view&path%5B%5D=282&path%5B%5D=179

Kiselev, E. (24 de Junio de 2015). Recuperado el 30 de Agosto de 2013, de Una mirada a la evolución del diseño web y el diseño de interfaz: http://www.workoholics.es/la-evolucion-del-diseno-web-y-el-diseno-de-interfaz/

Koneiform. (10 de Febrero de 2015). Obtenido de Python and AJAX tutorial for beginners with es realtiva.: http://kooneiform.wordpress.com/2010/02/28/python-and-ajax-for-beginners-with-webpy-and-jquery/

Lapuente, M. J. (8 de Diciembre de 2013). *Hipertexto*. Recuperado el 15 de Febrero de 2014, de La interfaz gráfica: http://www.hipertexto.info/documentos/interfaz.htm

Libros Web. (7 de Marzo de 2009). *Javascript Vanilla*. Recuperado el 3 de Octubre de 2013, de http://librosweb.es/libro/jobeet_1_4/capitulo_4/la_arquitectura_mvc.html

McCarthy, P. (s.f.). Obtenido de AJAX for developers: Build dynamic java applications: http://www.ibm.com/developerworks/library/j-ajax1/

Microsoft. (2 de Diciembre de 2013). Obtenido de ASP .net: Enhanced Interactivity and Responsiveness. Microsoft: http://www.asp.net/ajax

Otwell, T. (s.f.). Obtenido de Laravel: http://laravel.com

Paredes, M. A. (10 de Agosto de 2011). *Arquitectura en Capas*. Recuperado el 20 de Febrero de 2014, de http://arquitecturaencapas.blogspot.mx/2011/08/arquitectura-3-capas-programacion-por.html

Piedad Cristina Martínez Carazo. (12 de Marzo de 2006). *El método de estudio de caso.* Recuperado el 15 de Septiembre de 2014, de http://ciruelo.uninorte.edu.co/pdf/pensamiento_gestion/20/5_El_metodo_de_estudio_de_caso.pdf

Platzy. (17 de Diciembre de 2014). Obtenido de Platzi Educacion Online: http://platzi.com

Rally Software. (22 de Enero de 2015). Obtenido de https://www.rallydev.com/

Rissing, A. (s.f.). Obtenido de Comparativa entre J2EE, ASP.NET y PHP. : http://angerrising.com/2010/01/02/comparativa-entre-j2ee-asp-net-y-php/

Robert Orfali, D. H. (1999). Cliente/Servidor y objetos Guía de Supervivencia. En *Cliente/Servidor y objetos Guía de Supervivencia* (págs. 16-17). México: OXXFORD. Recuperado el 5 de Febrero de 2014

Ruby On Rails. (8 de Octubre de 2015). Obtenido de AJAX on Rails. Rails Guides: http://guides.rubyonrails.org/ajax_on_rails.html

Sanderson, S. (s.f.). Obtenido de Blog Steven Sanderson's: http://blog.stevensanderson.com/

Scrum Alliance. (13 de 12 de 2012). *Scrum una descripcion*. Recuperado el 17 de Abril de 2014, de https://www.scrumalliance.org/scrum/media/ScrumAllianceMedia/Files%20and%20PDFs/Why%20Scrum/Core%20Scrum%20Translations/Core-Scrum-Spanish.pdf

Tiobe Software. (1 de Agosto de 2015). *Tiobe Index for August 2015*. Recuperado el 17 de Agosto de 2015, de http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Universidad ITAM. (5 de Febrero de 2002). *El catálogo del sistema.* Obtenido de http://cursos.itam.mx/akuri/2002/S12002/BasesDeDatos/CHAPTER6.PDF

Untis Grubers and Petters . (2015). Recuperado el 20 de Agosto de 2013, de Generador de Horarios: http://www.school-timetabling.com

Valdés, D. P. (2 de Noviembre de 2007). *Los diferentes lenguajes de programación para la web*. Recuperado el 1 de Septiembre de 2013, de MAESTROS DEL WEB: http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web/

Vega, F. (5 de 12 de 2013). *Backend Profesional*. Obtenido de https://cursos.mejorando.la/cursos/backend-online/material/un-backend-profesional/

Vega, F. (5 de 12 de 2013). *Cristalab*. Obtenido de http://www.cristalab.com/blog/que-significa-backend-y-frontend-en-el-diseno-web-c106224l/

Vohra, D. (15 de Noviembre de 2012). Obtenido de Create web services with Ruby on Rails and Action web: http://www.ibm.com/developerworks/opensource/library/os-ws-rubyrails/index.html

W3C. (s.f.). Obtenido de AJAX Tutorial. W3Schools home: : http://www.w3schools.com/ajax/default.asp

WRENSOFT. (s.f.). Obtenido de PHP vs ASP vs ASP.NET vs Javascript vs CGI: : http://www.wrensoft.com/zoom/benchmarks.html