UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

SISTEMA DE GESTIÓN DE ASISTENCIAS DE ASESORES DEL SISTEMA DE UNIVERSIDAD ABIERTA Y EDUCACIÓN A DISTANCIA DE LA FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

DISEÑO DE UN SISTEMA O PROYECTO

CARLOS ALVARADO MARTÍNEZ

CD. MX. 2017

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

SISTEMA DE GESTIÓN DE ASISTENCIAS DE ASESORES DEL SISTEMA DE UNIVERSIDAD ABIERTA Y EDUCACIÓN A DISTANCIA DE LA FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

DISEÑO DE UN SISTEMA O PROYECTO

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMÁTICA

P R E S E N T A:

CARLOS ALVARADO MARTÍNEZ

A S E S O R:

L.A. RAMÓN ARCOS GONZÁLEZ

CD. MX. 2017

Índice general

Dedico el presente trabajo a mis hijos, María Cristina y Carlos, son mi impulso para seguir.
A mi esposa Laura por todo el apoyo que me ha dado, a mi familia y amigos.

CARLOS ALVARADO MARTÍNEZ

Agradecimientos

Quiero agradecer a la Universidad Nacional Autónoma de México por permitirme ser un miembro de su comunidad universitaria, por formarme, por enseñarme que la libertad se conquista, que los valores de la Universidad deben seguir en cada profesional hasta el último de sus días.

A la Facultad de Contaduría y Administración y al Sistema de Universidad Abierta y Educación a Distancia que me proporcionaron una sólida formación académica, a la vez de tener un trabajo de tiempo completo en la Industria de las Tecnologías de la Información.

Mención aparte merece el maestro Ramón Arcos que gustosamente aceptó ser mi asesor para el presente trabajo, a la maestra Marlene Ramírez que me propuso hacer este sistema, el maestro Germán Cervantes que me facilitó la infraestructura del proyecto en el servidor de la Facultad y a todos los asesores que me han ayudado durante mi formación como informático.

Capítulo 1

Introducción

No es tarea de la Universidad ofrecer lo que la sociedad le pide,
sino lo que la sociedad necesita.
— EDSGER WYBE DIJKSTRA

El presente trabajo es la culminación de la implementación del Sistema de gestión de asistencias de asesores en la Coordinación de Estadística y Evaluación del Sistema de Universidad Abierta y Educación a Distancia de la Facultad de Contaduría y Administración de la Universidad Nacional Autónoma de México.

Este sistema surgió como una necesidad de la Coordinación de Estadística y Evaluación del Sistema de Universidad Abierta y Educación a Distancia de la Facultad de Contaduría y Administración, que venía operando con procesos propensos a errores, poco adecuados para la cantidad de información que se tiene que manejar en el día a día, con la idea de simplificar los procesos existentes, mejorarlos y ofrecer mayor fiabilidad de la información que se genera en ésta área de la facultad.

Para este proyecto se hizo un uso intensivo de las herramientas estándar en los sistemas operativos UNIX, como Bash, AWK, sed, grep, entre otros, se usó una base de datos relacional de código abierto (PostgreSQL), se generaron los reportes solicitados usando Jasper Reports (que tiene dependencias como el *Java Runtime*

Environment 1.7) y la parte del *Front-end* se hizo aprovechando las posibilidades de las interfaces web que resultan familiares para los usuarios de los sistemas de cómputo.

Como una muestra de las posibilidades que se alcanzan haciendo uso de bases de datos relacionales para desarrollar sistemas de este tipo, se desarrolló un módulo de envío de correos electrónicos que hace consultas a las bases de datos, obtiene los datos solicitados y se le da formato al cuerpo del mensaje, logrando inclusive tener varios casos posibles con distintas redacciones, dependiendo de qué caso es el que aplica en cada mensaje concreto.

La complejidad de la operación del sistema queda encapsulada desde la perspectiva del usuario, de este modo funciona como una caja negra que es alimentada por dos archivos que contienen los registros a procesar, se generan datos adicionales en tiempo de ejecución y se generan las salidas solicitadas por los requerimientos de los *stakeholders*.

Este desarrollo se planeó haciendo uso de una metodología ágil pero que a la vez fuera lo suficientemente formal para cubrir el rigor de los requerimientos de un proyecto de estas características, se cumplieron las distintas etapas sugeridas por la metodología, se generaron los diagramas necesarios, se tuvo una constante retroalimentación con los *stakeholders* hasta lograr su plena aceptación del producto terminado.

El presente trabajo describe una puesta en producción de un sistema exitoso, que logró cumplir con lo requerido y disminuyó la cantidad de errores operativos que pueden poner en riesgo la integridad de los datos manejados por la Coordinación de manera cotidiana; desde el punto de vista de la implementación de proyectos mediante la ingeniería de software se trató de demostrar que llegar a hacer implementaciones exitosas es posible, siempre y cuando se apegue lo más que se pueda a una metodología que sea suficiente para cubrir las necesidades del

proyecto.

Así pues, desde la parte de investigación, el presente desarrollo intentó conjugar diversas herramientas libres en el contexto de un sistema, disminuyendo la necesidad de desarrollar módulos desde cero con algún lenguaje de programación, en ese sentido este proyecto no tuvo necesidad de compilar ningún binario nuevo, solo usa diversos scripts e inclusive el reporte en Jasper se genera internamente cada que se solicita obtenerlo y está elaborado en XML.

El sistema también se puede migrar fácilmente de un servidor a otro, haciendo factible su puesta en producción de manera relativamente sencilla cuando sea necesario cambiar el servidor en el que está alojado actualmente.

Aún con todo, quedaron algunos pendientes interesantes de implementar, que no fueron parte del alcance acordado para el proyecto, pero que de aprobarse, mejorarían aún más la operación y la seguridad de la información del presente Sistema de Gestión de Asistencias. Algunos de estos pendientes fueron implementar un método de recogida de los datos de los servidores de la plataforma Moodle, parte que no quedó como un alcance de este proyecto, debido a que el área en el que se desarrolló este sistema no tiene competencia suficiente para solicitar estos cambios fácilmente.

La estructura del presente trabajo quedó organizada de la siguiente manera:

-Capítulo 1. Introducción, se trata de la presente sección, se da un esbozo de lo que es este trabajo en términos generales.
-Capítulo 2. En este capítulo se detalla la historia de la organización, en este caso la propia Universidad Nacional Autónoma de México, del Sistema de Universidad Abierta y Educación a Distancia y de la Facultad de Contaduría y Administración.

-Capítulo 3. Se hace el diagnóstico del problema al que se le dio solución, mostrando los fundamentos teóricos de tal diagnóstico en el marco de referencia.
-Capítulo 4. En el cuarto capítulo, se desarrolla la propuesta de solución así como el proceso de implementación de la misma en producción.
-Capítulo 5. Resultados esperados, se incluye una lista con lo que los *stakeholders* del sistema esperan del mismo una vez implementado.
-Capítulo 6. Conclusiones, se describen los logros alcanzados con la implementación de la solución del presente trabajo.
-Capítulo 7. Anexos, se incluyen los scripts relevantes y procedimientos almacenados del sistema.
-Bibliografía. Se detalla la bibliografía consultada para el desarrollo del presente trabajo.

Tabla 1-1: Estructura capitular

Capítulo 2

Breve historia de la organización

2.1. Historia de la Universidad Nacional Autónoma de México

La Universidad Nacional Autónoma de México fue fundada en 1910 por el entonces presidente de México, Porfirio Díaz, como la cristalización de un primer proyecto presentado por Justo Sierra a la Cámara de Diputados en 1881 que, si bien no prosperó, fue la base para la creación de la Universidad. Durante el año de 1907, el presidente Díaz aprobó que se creara la Universidad Nacional de México y para este propósito se envió al pedagogo Ezequiel A. Chávez a Europa para entender cómo funcionaban las universidades de aquel continente.

De acuerdo a la reseña histórica (UNAM, 2015) se mencionan las escuelas que fundaron a la universidad:

La nueva institución estaría constituida por las escuelas Nacional Preparatoria, de Jurisprudencia, de Medicina, de Ingenieros, de Bellas Artes en lo concerniente a la enseñanza de la arquitectura y de Altos Estudios. Por fin, después de aprobado el proyecto, el 22 de Septiembre tuvo lugar la inauguración solemne de la Universidad Nacional de México. Fueron “madrinas” de la nueva universidad mexicana las de Salamanca, París y Berkeley.

Durante el año de 1929, la entonces Universidad Nacional de México ganó su

autonomía, pasando a denominarse como la Universidad Nacional Autónoma de México (UNAM), esto fue el resultado de una huelga estudiantil y del diálogo que sostuvieron con el entonces presidente de México Emilio Portes Gil. En 1945 el Congreso de la Unión aprobó la creación del Campus Central de la UNAM que sería llamado Ciudad Universitaria quedando inaugurado el 20 de noviembre de 1952 y comenzando la mudanza de las Facultades y Escuelas de la UNAM desde el llamado Barrio Universitario en el Centro Histórico hasta las nuevas instalaciones en el sur de la capital del país.

El movimiento de 1968 tuvo una repercusión muy fuerte en la UNAM, el entonces rector Javier Barros Sierra se pronunció a favor de las demandas estudiantiles, en contra de la violencia de cualquiera de las partes y condenó la presencia del ejército en los espacios de educación superior tanto en la propia UNAM como en el Instituto Politécnico Nacional; tras la dura represión y el asesinato de estudiantes ocurrida el 2 de octubre de 1968, la UNAM se volvió un lugar donde las ideas se podían expresar libremente, si bien la represión del entonces partido gobernante siguió desde fuera y desde dentro de la universidad.

En este contexto, el durante el rectorado del doctor Pablo González Casanova se propuso una Reforma Universitaria donde se creó el Sistema de Universidad Abierta (SUA), el Colegio de Ciencias y Humanidades (CCH) y se crearon las Escuelas Nacionales de Estudios Profesionales (ENEP), que fueron los primeros campus fuera de la Ciudad Universitaria para atender la creciente demanda de educación superior en la Ciudad de México; las ENEP evolucionaron con el tiempo y se convirtieron en Facultades de Estudios Superiores (FES).

Después de otras dos huelgas importantes en los años 1987 y 1999, la UNAM se consolidó como la principal institución de educación superior y de investigación humanística y científica de México, su campus central se convirtió en Patrimonio Cultural de la Humanidad en 2007, la UNAM ganó el Premio Príncipe de Asturias

de Comunicación y Humanidades en 2009. La Universidad comenzó su crecimiento a otros estados con las Escuelas Nacionales de Estudios Superiores (ENES), con campus en León Guanajuato, en Morelia Michoacán y otro campus planeado en Mérida Yucatán, aparte de las instalaciones que la universidad tiene de sus diversos Institutos a lo largo del país (UNAM, 2015).

2.2. Historia del Sistema de Educación Abierta y a Distancia de la UNAM

El Sistema de Educación Abierta y a Distancia (SUAyED) de la UNAM surgió como parte del proyecto de Reforma Universitaria impulsado por el entonces rector, el doctor Pablo González Casanova, en donde se intentó hacer más accesible la educación universitaria para quienes por distintas razones no tuvieran la posibilidad de asistir de manera presencial a una licenciatura (SUAYED-UNAM, 2016); el Consejo Universitario aprobó la creación del estatuto del entonces SUA (Sistema de Universidad Abierta) en el año 1972. En 1999 se aprobó un nuevo reglamento para el SUA, dando nacimiento a la modalidad de Educación a Distancia, con lo que el SUA pasó a denominarse Sistema de Universidad Abierta y Educación a Distancia (SUAyED) y se creó la Coordinación de Universidad Abierta y Educación a Distancia (CUAED); el sistema tuvo una reforma adicional en el año 2009 para responder mejor a los cambios de las Tecnologías de la Información a inicios del siglo XXI.

El SUAyED de la UNAM imparte un sistema de bachillerato a distancia, ofrece varias licenciaturas, posgrados, especializaciones, así como cursos de educación continua; para el ingreso al SUAyED, la UNAM exige los mismos requisitos necesarios para ingresar en el sistema escolarizado. La UNAM otorga a los alumnos del SUAyED los mismos títulos y grados que los del sistema escolarizado, al tiempo

que se tienen los mismos derechos y obligaciones que cualquier otro alumno de la Universidad (SUAYED-UNAM, 2016).

2.3. Historia de la Facultad de Contaduría y Administración de la UNAM

La Facultad de Contaduría y Administración de la Universidad Nacional Autónoma de México surgió originalmente como Facultad de Comercio y Administración en 1929, tras haber conseguido la Universidad Nacional de México su autonomía, resultado de una huelga en aquellos años que fue apoyada por la entonces Escuela Superior de Comercio y Administración, dependiente de la Secretaría de Instrucción.

Pero debido a que los estudios que ofrecía la Facultad en aquellos tiempos no exigían los estudios previos de bachiller, perdió su denominación de Facultad y se volvió Escuela Nacional de Comercio y Administración y fue hasta 1960 con la apertura de los posgrados que volvió a ganar el título de Facultad, obteniendo su actual nombre.

El Sistema de Universidad Abierta (SUA) comenzó a ofrecer su modalidad en la Facultad en 1977, la licenciatura en Informática se comenzó a impartir en el año 1985 en el sistema escolarizado, el sistema de Educación a Distancia se fue implantando gradualmente desde 1999; según estadísticas del SUAyED para toda la UNAM (CUAED-UNAM, 2015), la modalidad a Distancia en la Facultad de Contaduría y Administración tiene ya más alumnos inscritos que los alumnos inscritos en las opciones de la modalidad del sistema Abierto, como se puede apreciar en la figura 2-1.

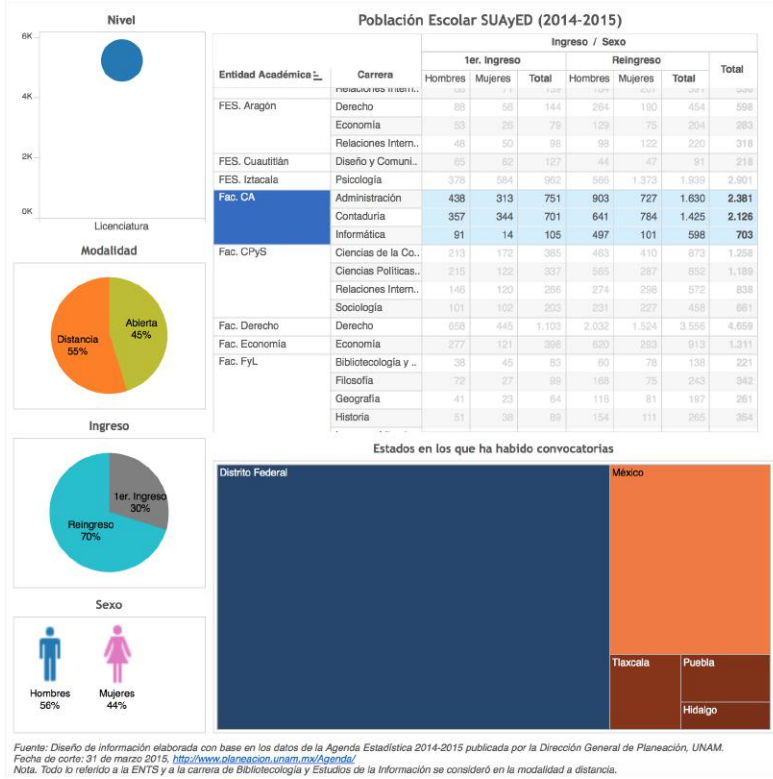


Figura 2-1: Población escolar 2015 del SUAyED en la FCA de la UNAM (población escolar 2016).

La Coordinación de Estadística y Evaluación del Sistema de Universidad Abierta y Educación a Distancia de la FCA, es el área encargada de evaluar el desempeño de los asesores del sistema durante el semestre, llevar el registros de sus asistencias, faltas, reposiciones y justificantes, así como atender sus dudas y reclamaciones; es aquí donde se detectó la necesidad de desarrollar el presente proyecto.

2.4. Objetivos misión, visión y valores de la Facultad de Contaduría y Administración.

La Facultad de Contaduría y Administración de la UNAM tiene como objetivo, misión y visión, de acuerdo a su sitio web, lo siguiente:

2.4.1. Objetivo

La Facultad de Contaduría y Administración, comprometida permanentemente con la misión de formar profesionistas, docentes e investigadores del más alto nivel en las disciplinas financiero-administrativas, busca permanentemente las mejores herramientas que apoyen el desarrollo de su quehacer académico y su desempeño administrativo. (FCA-UNAM, 2016c)

2.4.2. Misión

Formar profesionales, profesores e investigadores de la contaduría, la administración y la informática, que contribuyan al desarrollo económico del país mediante la solución de los problemas prácticos que enfrentan las empresas y las organizaciones, así como realizar investigación orientada a la generación del conocimiento de estas disciplinas; cultivando en su comunidad el espíritu analítico, crítico y reflexivo, y proporcionando las herramientas técnicas que les permitan ser altamente competitivos en los planos nacional e internacional. (FCA-UNAM, 2016b)

2.4.3. Visión

Ser el modelo educativo de mayor influencia en el país y en Iberoamérica, reconocido en los niveles nacional e internacional por la calidad y liderazgo de sus

profesionales y por su contribución al desarrollo de la investigación en sus disciplinas. Así mismo, debe convertirse en el modelo en el que se formen y eduquen verdaderos líderes que propicien el desarrollo económico de nuestro país en las organizaciones nacionales (FCA-UNAM, 2016d).

2.4.4. Valores

La Facultad de Contaduría y Administración tiene los mismos valores de la Universidad Nacional Autónoma de México, cuyos valores son:

La legalidad, la creatividad, el cuidado del ambiente, la lealtad, la innovación, la pasión, la perseverancia, la solidaridad, la integridad académica, la igualdad, la calidad de vida, el compromiso, la amistad, el afán por el saber, la equidad de género, la responsabilidad, la laicidad, el respeto, la autonomía, la libertad de expresión, la honestidad y la tolerancia (UNAM, 2016).

2.5. Organigramas de la Facultad de Contaduría y Administración.

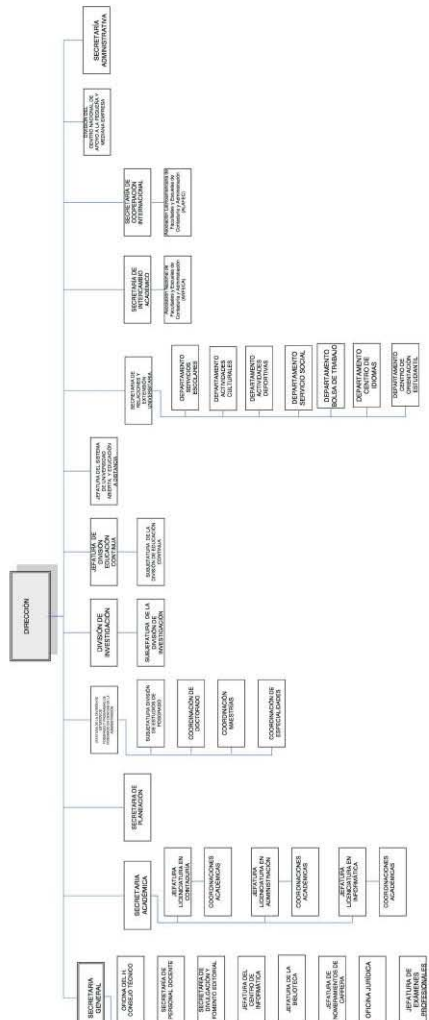


Figura 2-2: Organigrama de la Facultad de Contaduría y Administración de la UNAM.

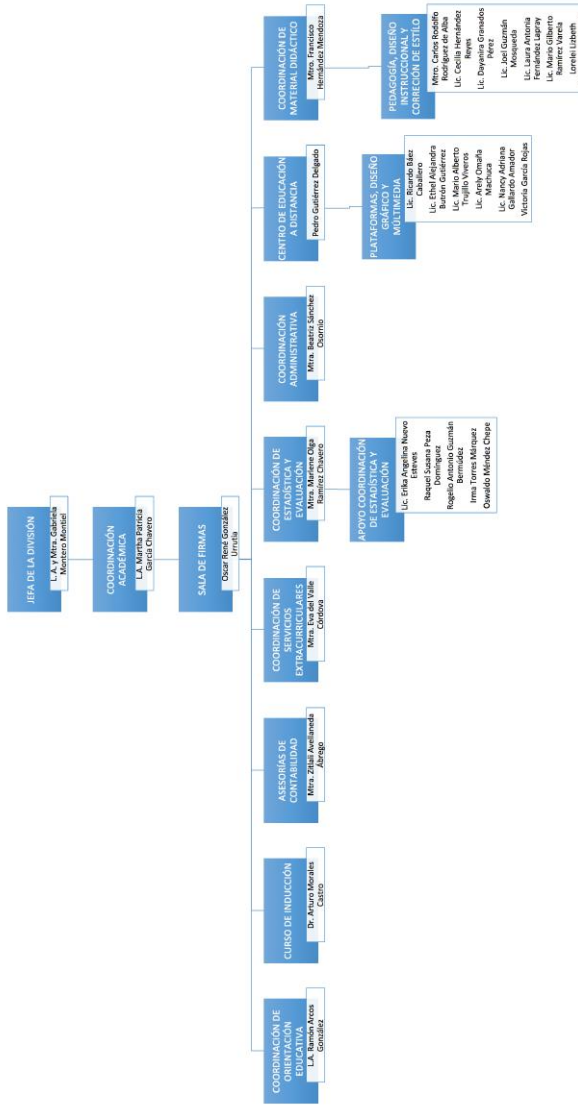


Figura 2-3: Estructura organizacional de la División del Sistema de Universidad Abierta y Educación a Distancia de la Facultad de Contaduría y Administración de la UNAM.

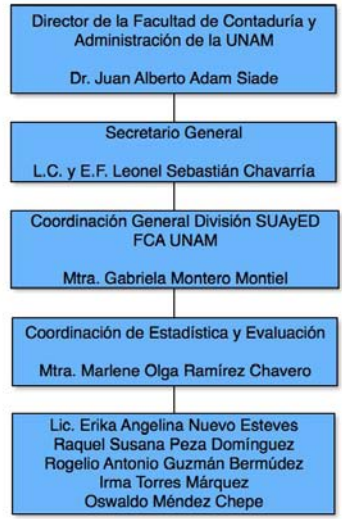


Figura 2-4: Organigrama específico de la Coordinación de Evaluación y Estadística del SUAyED la FCA de la UNAM.

Capítulo 3

Diagnóstico del problema

3.1. Marco Contextual

3.1.1. Situación Previa

El presente desarrollo se implementó como un Sistema de gestión de asistencias para los asesores del SUAyED de en la Coordinación de Evaluación y Estadística de la FCA de la UNAM, la situación previa que se vivía en esta área administrativa era de casi una total operación manual y empírica, cada usuario iba aprendiendo cómo ejecutar sus procesos, utilizando varias veces la misma información en distintos archivos de Excel, lo cual es un ambiente propicio para los errores y la poca garantía de consistencia de la información.

3.1.2. Procesos Previos

Los procesos anteriores a la implementación de la solución propuesta hacían un uso intensivo de la hoja de cálculo Microsoft Excel para generar los reportes y documentos necesarios, en envío de los correos electrónicos a los asesores para informarle el estatus de sus asistencias, faltas y reposiciones era manual y se tenía que replicar la misma información en varios archivos de Excel, de modo que la

información fácilmente podría verse comprometida por errores humanos.

La coordinación disponía de un programa para el envío de correos que llamaba al servidor de correo de la facultad para enviar el correo redactado por la coordinadora del área, los datos del correo se calculaban a mano y la redacción del mismo cambiaba constantemente.

El reporte de cada asesor al finalizar el semestre se iba obteniendo en una hoja de cálculo mediante macros y se tenía imprimir un reporte a la vez, cambiando de asesor y grupo para cada reporte manualmente, cada reporte impreso se iba compilando y se guardaba en el archivo de la Coordinación.

3.1.3. Sistemas Previos

La infraestructura utilizada no incluía ningún servidor, solamente los equipos de cómputo del área, los cuales ejecutaban el sistema operativo Microsoft Windows y tenían instalado los navegadores web Internet Explorer, Mozilla Firefox y Google Chrome.

En cuanto a los sistemas del SUAyED, los servidores de la plataforma están bajo resguardo del CIFCA de la FCA de la UNAM y ejecutan la plataforma Moodle, se tienen dos versiones de la plataforma y a cada plataforma se tiene que entrar e ir sacando los reportes requeridos a mano, quedó fuera del alcance de este proyecto implementar un método automático para recoger esta información.

3.2. Marco Teórico

Se hace una breve descripción de los sistemas operativos, los gestores de bases de datos, los lenguajes de programación y el software de aplicación que se utilizaron en menor o en mayor medida para el desarrollo del presente trabajo. Después, se hace una descripción de algunas de las metodologías principales para el desa-

rrollo de proyectos de software, desde las tradicionales y evolutivas hasta algunas metodologías ágiles.

3.2.1. Sistemas Operativos

UNIX

UNIX es una familia de sistemas operativos, el acrónimo UNIX es comercial y requiere certificación para ostentarse como uno, por lo que los sistemas operativos tipo UNIX que no tienen esta certificación se denominan *Unix-like*. Desarrollado en los Bell Labs por Dennis Ritchie y Ken Thompson, fue el primer sistema operativo escrito en un lenguaje de alto nivel (C) lo que permitió la portabilidad de un componente que se pensaba inherente a la arquitectura de hardware de una computadora como es el sistema operativo. UNIX le brinda al programador un conjunto de herramientas como las *llamadas al sistema* que permiten explotar al máximo los recursos de una computadora (García, 2006).

Hoy en día, las distintas implementaciones de UNIX, certificadas o no, dominan la industria de las TI en muchos segmentos, sus distintas implementaciones han ido evolucionando constantemente, desde la minicomputadora DEC PDP-7 que se usó para su creación hasta las tendencias actuales como el *Cloud Computing* y los sistemas operativos móviles como Android (basado en Linux, un *Unix-like*) e iOS (un derivado de UNIX BSD).

GNU/Linux

Según Richard Petersen en su Manual de referencia, GNU/Linux, también conocido simplemente como Linux, es un sistema *Unix-like* que comenzó su desarrollo en 1991, creado por Linus Torvals quien quería implementar el sistema operativo de propósito académico MINIX en computadoras con arquitectura x86, des-

pués subió su idea a Internet y fue bien recibida por los programadores de todo el mundo (Petersen, 2009); hoy en día millones de dispositivos usan alguna variante de Linux: supercomputadoras, servidores, computadoras de escritorio, laptops, los dispositivos móviles Android cuyo kernel es Linux, además de otros muchos sistemas embebidos.

El éxito de Linux está íntimamente ligado a la Internet, pues usando esta red global, fue que desarrolladores de muchos países pudieron colaborar y fueron creando no solo el kernel de Linux, sino todo un ecosistema de software alrededor de la idea, fue así que surgieron entornos completos de escritorio como KDE y Gnome, también soluciones profesionales de software como LaTeX y GIMP, además de todas las herramientas clásicas heredadas de los sistemas UNIX, sin dejar de mencionar la multitud de *distribuciones* de Linux existentes; esto facilitó a Google para que tomara a Linux como base para el desarrollo de su sistema operativo móvil Android, que domina hoy en día en los dispositivos móviles.

Microsoft Windows

Microsoft Windows es un sistema operativo desarrollado por Microsoft Corporation que surgió en 1985, primero como mejora del sistema operativo DOS y después como un sistema completo, contiene la interfaz gráfica compilada dentro kernel del sistema, por lo que la interfaz gráfica no se puede quitar por completo del sistema, al contrario de lo que suele suceder en los sistemas tipo UNIX (aunque esto no es una regla, OS X ahora llamado MacOS, es un UNIX y también compila la interfaz gráfica dentro del kernel).

El sistema operativo Microsoft Windows dominó ampliamente el mercado de las computadoras personales durante la década de los 90 del siglo XX y la primer década del siglo XXI; para la segunda década del siglo, Microsoft Windows goza

de una posición consolidada, pero está decayendo lentamente en aras de las soluciones móviles (que ya no usan la arquitectura x86, sino la arquitectura ARM) como Android e iOS; no obstante, Windows tiene versiones para móviles compatibles con la arquitectura ARM, llamado Windows Phone y más recientemente, Windows 10 Mobile.

3.2.2. Bases de Datos

Las bases de datos son compendios de información que forman la base de los sistemas informáticos, existen varios tipos de bases de datos, pero las más usadas en el ámbito de las aplicaciones de software son las que siguen el modelo relacional. De acuerdo al curso práctico avanzado de PostgreSQL, el modelo relacional tiene como objetivo la independencia física de la información, de modo que aunque se cambie conceptualmente el modelo de la base de datos, eso resulte irrelevante para el usuario que consulte la información contenida en esa base (Arana, 2015).

El modelo relacional tiene un conjunto de reglas de diseño que son llamadas "formas normales" definidas por Edgar Frank Codd en IBM y tienen como propósito garantizar la integridad de la información, evitar la redundancia de los datos y hacer menores los problemas de actualización de la información de una base de datos; la tercera forma normal es la más empleada para el despliegue de bases de datos relacionales.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales, libre, colaborativo y de código abierto, se originó en 1982 en la Universidad de Berkeley, es uno de los sistemas de gestión de bases de datos preferidos por sus capacidades, cercanas a las que tienen los gestores propietarios como Oracle y DB2, si bien en

los sitios Web tiene menos presencia que MySQL, Postgres como también se le conoce incluyó tempranamente muchas características que gestores comerciales tardaron mucho tiempo en implementar (Arana, 2015).

3.2.3. Lenguajes de Programación

Bash

Bash es un intérprete de comandos cuyo nombre significa *Bourne Again Shell*, sucesor del intérprete *sh*, Bash es un shell de UNIX y es compatible con el estándar POSIX (Sobell, 2010). Permite ejecutar muchas utilerías del sistema operativo UNIX, ejecutarlas en conjunto mediante scripts y también interpreta las estructuras de programación más utilizadas como los ciclos *while*, *for*, *loop*, *case*, *until*, las estructuras de decisión como *if*, *else*, *elif*, entre otras. También se pueden ejecutar otros scripts dentro de un script de Bash con *exec* y se pueden ejecutar otros shells y lenguajes de programación directamente.

AWK

AWK son las siglas de sus creadores (Alfred) Aho, (Peter) Weinberger, y (Brian) Kernighan, es un lenguaje de programación procesamiento de datos de texto, desarrollado originalmente en AT&T, actualmente existen algunas variantes de este lenguaje, como NAWK de la propia AT&T y GAWK la versión GNU, que cuenta con mayores funciones; para este proyecto se usó GAWK que está disponible en la distribución de Debian GNU/Linux.

SQL

SQL (*Structured Query Language*), es un lenguaje de consulta estructurado que permite la interacción con bases de datos relacionales, este lenguaje ha estado en evolución desde su desarrollo inicial que antes se llamaba SEQUEL, basado en los trabajos de Edgar Frank Codd quién definió las bases de datos relacionales. El lenguaje SQL se subdivide en en algunos sublenguajes como DML *(Data Manipulation Language)* y DDL *(Data Definition Language)*, en 1987 fue adoptado por la ISO como estándar para las consultas en las bases de datos relacionales de acuerdo al libro de Bases de Datos de José Reinosa *et al.* (Reinosa, Maldonado, Muñoz, Damiano, y Abrutsky, 2012).

PL/pgSQL

PL/pgSQL (*SQL Procedural Language*) es un lenguaje imperativo de PostgreSQL que permite ejecutar *Stored Procedures*, amplía las funcionalidades del lenguaje SQL, dotándolo de las estructuras comunes de los lenguajes de programación permitiendo ejecutar ciclos, tomar decisiones y reaccionar a eventos en la base de datos (Arana, 2015).

Cada sentencia de tipo SQL tiene que ser ejecutado uno a uno por la base de datos, esto puede implicar que si se tiene una aplicación cliente que use consultas SQL para obtener datos de un servidor, esta tendrá que esperar a que cada consulta sea procesada, se reciban y se procesen los resultados, lo que provoca mucha comunicación interproceso que puede provocar problemas de disponibilidad de la red.

De acuerdo a la información de su sitio web, (PostgreSQL-Group, 2016) para solucionar este problema, PL/pgSQL permite agrupar bloques de consultas que son enviados dentro del servidor de base de datos, logrando tener la potencia de los

lenguajes de procedimientos y la facilidad de las sentencias SQL al mismo tiempo, logrando reducir el tráfico requerido para las aplicaciones que se comuniquen con las bases de datos PostgreSQL, los resultados intermedios no tienen que ser enviados a través de la red, se procesan localmente del lado del servidor, siendo necesario solo enviar la consulta con sus parámetros y obtener la respuesta del procedimiento almacenado.

HTML5

HTML5 es la evolución del estándar Web HTML (*HyperText Markup Language*), añade funcionalidad y versatilidad al antiguo estándar, compitió contra soluciones como Adobe Flash, ganando en el mediano plazo la preponderancia en la Web, interactúa con otros lenguajes como PHP, JQuery, Javascript entre otros.

HTML5 añade etiquetas de tipo semántico al lenguaje HTML como la etiqueta <nav>que abre un bloque de navegación dentro de un sitio web.

PHP

Las siglas de PHP (*PHP: Hypertext Processor*) son un acrónimo recursivo, de acuerdo al sitio del proyecto (PHP-Group, 2016) se trata de un lenguaje orientado al scripting de propósito general, especialmente diseñado para el desarrollo web en HTML, su función principal es simplificar el desarrollo web, así pues, en lugar de usar mucho código HTML para que una página web haga algo en concreto, PHP le dice que haga "algo" dentro de las instrucciones de inicio y fin de PHP (que son <? y ?>), lo que permite entrar y salir del modo PHP al navegador web.

PHP es un lenguaje que se ejecuta del lado del servidor, una de sus características principales es que es multi-paradigma, pues es orientado a objetos, es un lenguaje de procedimientos, es imperativo y también reflexivo. Se puede ejecutar

en casi todos los sistemas operativos y está presente en millones de sitios web en conjunto con HTML.

XML

XML (*eXtensible Markup Language*) es un lenguaje de marcas desarrollado por la W3C, es un derivado de SGML (ISO 8879) (*Standard Generalized Markup Language*) que permite almacenar datos y compartirlos entre distintas plataformas, XML es un lenguaje extensible, de modo que se puede añadir funcionalidad a sus posibilidades lo que lo puede catalogar como un meta-lenguaje que permite definir lenguajes de marcado dependiendo de las necesidades que se tengan.

Se puede decir que el lenguaje *XML es un subconjunto del estándar SGML pero simplificado y adaptado para Internet* (UAdM, 2016).

JavaScript

JavaScript (abreviado comúnmente como JS) es un lenguaje de programación ligero e interpretado, orientado a objetos, se trata de un derivado de ECMAScript, se usa principalmente del lado del cliente, de acuerdo al sitio web de Mozilla Developer Network (Network, 2016), JavaScript es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa. El estándar de JavaScript ECMAScript es soportado por todos los navegadores web existentes, especialmente su versión 5.1.

JavaScript no se debe confundir con el lenguaje de programación Java, que tiene una sintaxis bastante distinta a la de JavaScript, si bien JavaScript se parece mucho al lenguaje C al igual que Java, su nombre se debió meramente a una estrategia comercial de los años 90 del siglo pasado.

AJAX

AJAX (*Asynchronous JavaScript And XML*) permite crear aplicaciones web interactivas de manera asincrónica, está basado en Javascript y en XML, la función principal de AJAX es la de autocompletar búsquedas en interfaces web construidas mediante JavaScript y XML en bases de datos; utiliza el objeto XMLHttpRequest para pasar las consultas y las respuestas de manera asincrónica entre el cliente y el servidor de la aplicación basada en JavaScript, citando al sitio web del proyecto (NetBeans-Group, 2016).

JQuery

JQuery es una biblioteca (que no "librería", pues el término *library* se debe traducir como "biblioteca") de JavaScript, permite ejecutar AJAX en las páginas Web de una manera más simplificada; según la información del sitio web de JQuery (jQuery Foundation, 2016), JQuery provee toda una gama de interacciones, elementos, efectos, widgets y temas para las interfaces gráficas de usuario basadas en JavaScript.

3.2.4. Software de Aplicación

Moodle

Conforme a la información disponible en el sitio web de este proyecto "Moodle es una plataforma de aprendizaje diseñada para proporcionarle a educadores, administradores y estudiantes un sistema integrado único, robusto y seguro para crear ambientes de aprendizaje personalizados" (Moodle.net, 2016).

Moodle es una plataforma libre que se usa por defecto en el SUAyED de la UNAM, cada Facultad recibe el sistema y lo administra desde sus propios servido-

res; los datos de asistencia de los asesores que son utilizados por este proyecto se obtienen de la plataforma Moodle de la Facultad de Contaduría y Administración.

Apache

Apache es un esfuerzo colaborativo para desarrollar y mantener un servidor HTTP para sistemas operativos modernos como Microsoft Windows y la familia de sistemas operativos UNIX; Apache es el estándar mundial de la World Wide Web, compite directamente con otros servidores HTTP como el IIS de Microsoft, pero teniendo una cuota muy superior de presencia con respecto a la solución de servidor HTTP de Microsoft; adicionalmente a Apache, existen derivados del mismo como el IHS (IBM HTTP Server) de IBM que también tienen bastante presencia en la Internet (Apache-Foundation, 2016).

Teampostgresql

Teampostgresql es un gestor web de bases de datos PostgreSQL, se ejecuta en el sistema operativo como un servicio por lo que requiere ser instalado o al menos incluido en los scripts de inicio del sistema en el caso de los sistemas UNIX.

phpPgAdmin

phpPgAdmin es un gestor web de bases de datos PostgreSQL, es una aplicación desarrollada en PHP, por lo que no requiere un servicio aparte para ejecutarse correctamente, solo requiere instalarse en el directorio web del servidor y tener bien configurados los permisos para funcionar adecuadamente.

Jasper Reports

Jaspersoft ha lanzado múltiples productos para elaborar reportes y estadísticas especializadas utilizando distintos gestores de bases de datos, Jasper Reports es un producto derivado de esta empresa, se basa en XML, en Java y permite generar varios tipos de reportes, como reportes HTML y reportes PDF.

Microsoft Excel

Microsoft Excel es la solución de hola de cálculo de Microsoft, forma parte de la suite ofimática Microsoft Office, es la hoja de cálculo más utilizada a nivel mundial, desde su versión 2007 tiene un formato compatible con XML en lugar de generar archivos binarios que eran poco compatibles (que fue su estándar hasta la versión 2003).

Formato CSV

CSV (*Comma Separated Values*) es un formato de hojas de cálculo o de matrices que separa los valores de las columnas con comas, es un formato muy popular en los gestores de bases de datos, no tiene más formateo que las propias comas y que en los valores que tengan una coma como parte del texto, el valor completo se pone con comillas (").

3.2.5. Metodologías de Desarrollo de Proyectos

Los proyectos de desarrollo e implantación de sistemas informáticos, están enmarcados en lo que tanto en la Industria como en la Academia se ha denominado como *Ingeniería de Software*, que si bien ha tenido detractores como Dijkstra que la define como la *Disciplina Condenada* (Dijkstra, 1988), se ha convertido en el

marco estandarizado del desarrollo de proyectos de software en las organizaciones.

De acuerdo con el libro de Guillermo Pantaleo, una *metodología*, es un tipo de trabajo especifico para desarrollar software que indica las tareas por realizar, el tipo de elementos que creará y sus respectivas relaciones (Pantaleo y Rinaudo, 2015).

La llamada Ingeniería de Software propone varias metodologías para abordar los diversos tipos de proyectos que se suelen presentar: unas son las metodologías estructuradas, más viejas, muy orientadas a la generación de documentos formales, apegadas a *deadlines*, donde en muchas ocasiones, con tal de cumplir con la rigidez de sus requisitos, se tiene que mermar la *calidad* del software entregado, generando una mala impresión en los clientes que no ven cumplidas sus expectativas.

Para paliar estas cuestiones, tomando como base la propia experiencia en proyectos de desarrollo de software, fue que surgieron las metodologías *ágiles* que, conscientes de los problemas que surgen con la aplicación de las metodologías estructuradas, que prefieren a los individuos e interacciones sobre procesos y herramientas, al software funcionando sobre documentación extensiva, la colaboración con el cliente sobre negociación contractual, la respuesta ante el cambio sobre seguir un plan (Pantaleo y Rinaudo, 2015).

Es por esto que para el desarrollo del Sistema de gestión de asistencias, se tuvo que elegir alguna metodología de desarrollo de software, a continuación se listan algunas de las candidatas:

3.2.6. Metodologías Tradicionales y Evolutivas

Las metodologías de desarrollo de software tradicionales se rigen por planes de implementación, las más comunes son: la metodología de desarrollo en cascada, la

que se rige por prototipos, la de desarrollo rápido de aplicaciones, la incremental y la metodología en espiral. Las metodologías regidas por planes tienen una planificación muy predictiva, en la práctica existe mucha resistencia al cambio por parte de los usuarios; estas metodologías son fuertemente orientadas a los procesos, en ellas, los contratos y los términos legales tienen la preponderancia. Es en la arena de negociación en la que el cliente plasma sus deseos y el equipo de desarrollo le pone límite a lo que el cliente puede pedir, aunque es frecuente que con tal de no perder un contrato, el equipo de desarrollo acepte hacer cosas de las que no se tiene certeza sobre su viabilidad.

Desarrollo en Cascada

Esta metodología estructurada ha servido como base de otras metodologías, es la más básica de todas, está basada en los documentos que se definen para el proyecto, sirve para los proyectos bien definidos o bien cuando se tienen que hacer mejoras o adaptaciones a sistemas ya existentes que además ya son estables; según Roger S. Pressman, este modelo es sistemático y secuencial; en esta metodología se definen las distintas fases del proceso de desarrollo como se puede apreciar en la figura 3-1, sus fases no se solapan entre sí y son las siguientes (Pressman, 2010):

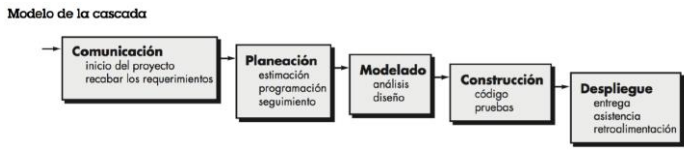


Figura 3-1: Pressman, R. (2010). Fases del Desarrollo en Cascada. [Figura]

-Fase de comunicación, en esta fase se define el proyecto que se desarrollará y se recaban los requerimientos, tanto funcionales como los no funcionales, en-

tre los usuarios y el equipo de desarrollo, de ahí se desprende el documento de especificación de requerimientos.

-Fase de planeación, en esta fase se hace el estimado del proyecto, se programan las actividades a seguir y se indica el modo en el que se le dará seguimiento al avance del mismo, los documentos derivados son los planes de trabajo, los diagramas de Gantt, entre otros.

-Fase de modelado, se analizan los requerimientos y las especificaciones necesarias, se construye el modelo del software que se desarrollará, se toma como base el documento de alcance del proyecto.

-Fase de construcción, aquí se codifica el software propiamente, se crea el programa siguiendo todo lo especificado y posteriormente se realizan las pruebas de lo desarrollado por el equipo, en las pruebas participan los interesados en el sistema.

-Fase de despliegue, se entrega la solución al cliente, se instala y se configura, se le proporciona asistencia técnica y se tiene retroalimentación con el mismo hasta la satisfacción de los requerimientos del proyecto.

El modelo de desarrollo en cascada es el más antiguo de los modelos de desarrollo de software, se ha cuestionado ampliamente su eficacia, dado que los supuestos que debe cumplir un proyecto de software para que el modelo en cascada funcione en raras ocasiones se cumplen, resultando en un modelo que requiere muchas adecuaciones para poder salir adelante en proyectos reales, es por ello que se desarrollaron más modelos de desarrollo dentro de la ingeniería de software.

Desarrollo Incremental

La metodología de desarrollo incremental requiere que los requerimientos iniciales del software estén bien definidos, sin embargo puede que los alcances del mismo no se hayan podido definir con claridad, para esta situación se propone el

modelo de desarrollo basado en incrementos.

Este modelo va ejecutando las fases del modelo en cascada (comunicación, planeación, modelado, construcción y despliegue), el primer incremento se conoce como el *producto fundamental* y al finalizar cada incremento el usuario prueba el producto desarrollado, generando el plan de desarrollo para el incremento siguiente, haciendo que en cada incremento, se esté entregando un producto funcional que puede ser probado por el cliente en producción, lo que lo hace diferente de los modelos de desarrollo evolutivos, es posible revisar el esquema de este modelo en la figura 3-2.

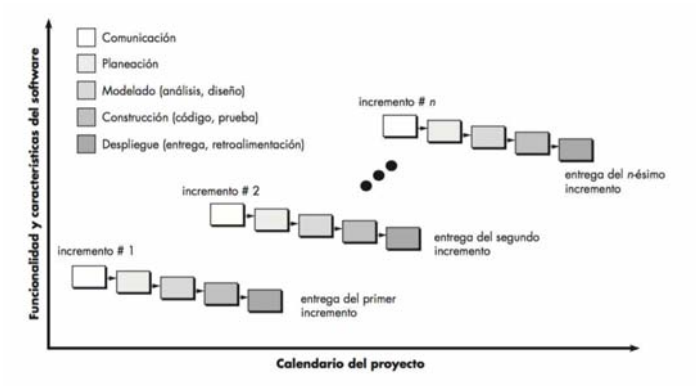


Figura 3-2: Pressman, R. (2010). Fases del Desarrollo Incremental. [Figura]

El desarrollo incremental resulta útil cuando no se puede contar con todo el personal requerido para el desarrollo del producto pero que requiere entregar algo funcional que pueda ser evaluado por el cliente, que, de estar de acuerdo con los avances, posibilitaría incluir en el equipo al personal restante para ir cubriendo las necesidades del proyecto, también resulta útil para gestionar riesgos técnicos asociados al proyecto.

Desarrollo en Espiral

El modelo de desarrollo en espiral es una de las respuestas diseñadas para suplir las carencias que ha demostrado el modelo de desarrollo en cascada, fue propuesto por Barry Boehm y forma parte de los modelos de desarrollo evolutivos, donde las distintas fases del mismo se van ciclando de forma iterativa hasta cumplir con los requerimientos del proyecto encargados por el cliente.

Las fases del modelo de desarrollo en espiral, son las mismas que las del desarrollo en cascada, es decir, la fase de comunicación, de planeación, de modelado, de construcción y de despliegue, pero al finalizar el ciclo, se vuelve a ejecutar cada fase pero en un estado superior, de modo que se pueden ir añadiendo más requerimientos, se puede ir ampliando el alcance de la solución, se pueden realizar varias fases de pruebas y se puede ir añadiendo funcionalidad al proyecto desarrollado, de este modo el desarrollo en espiral supera a su predecesor en cascada, como se ve en la figura 3-3..

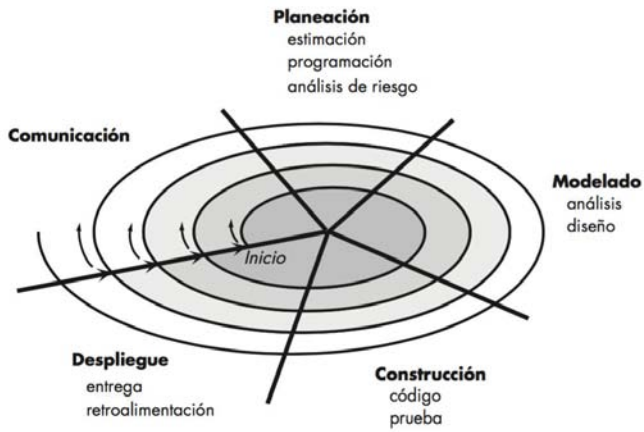


Figura 3-3: Pressman, R. (2010). Fases del Desarrollo en Espiral. [Figura]

Una de las partes negativas de este modelo de desarrollo de software, es que cada vez que el ciclo da vuelta, el costo de ir haciendo modificaciones se eleva bastante, resultando no costeable tras varias vueltas a la espiral, adicionalmente es requerido que el equipo de desarrollo tenga mucha experiencia en la evaluación del riesgo, de modo que si se presenta un riesgo de alto impacto no previsto en ciclos avanzados, el proyecto mismo corre peligro de fracasar; también resulta difícil convencer a los clientes de que más ciclos de desarrollo son necesarios de acuerdo con (Pressman, 2010).

Desarrollo por Prototipos

El desarrollo por prototipos resulta muy útil cuando el proyecto a desarrollar no contiene todos los requisitos, de modo que no se podría garantizar que el producto desarrollado cumpla con las expectativas del cliente del proyecto, sus fases se pueden revisar en la figura 3-4.

Para que los requerimientos vayan conociéndose, es buena idea ir desarrollando prototipos funcionales que con los límites de funcionalidad inherentes que presentan, pueden ir dando claridad tanto al cliente como al equipo de desarrollo de los requisitos que se deberían de ir solventando durante el proyecto.

Esto implica que las fases del modelo de desarrollo en cascada se tengan que cambiar un poco, de modo que las fases de esta forma de desarrollo son:

-Comunicación

-Plan rápido

-Modelado diseño rápido

-Construcción del prototipo

-Despliegue, entrega y retroalimentación

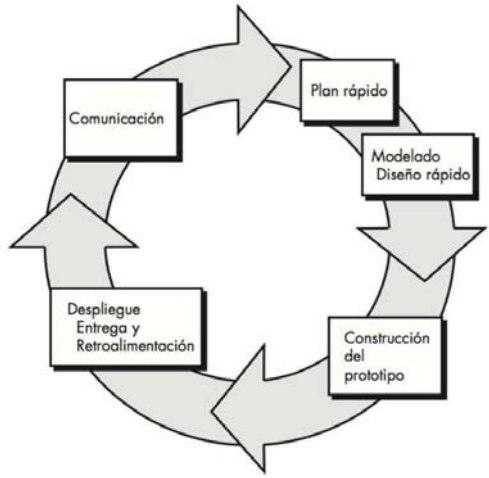


Figura 3-4: Pressman, R. (2010). Fases del Desarrollo por Prototipos. [Figura]

Cada una de las fases se va ejecutando de manera cíclica, hasta que el prototipo generado se pueda considerar como un producto acabado que cumpla con todos los requerimientos encontrados durante el proceso de desarrollo.

Proceso Unificado *(Rational Unified Process)*

En la figura 3-5 se aprecia la matriz del Proceso Unificado, que es un *proceso* de desarrollo de software, donde un proceso es *un conjunto de actividades necesarias para transformar los requisitos del cliente o usuario en un sistema de software funcional*, cuyo nombre comercial es *Rational Unified Process* (RUP) y se trata de una metodología que fue desarrollada por la empresa Rational Software que fue adquirida por IBM en el año 2002.

Una característica fundamental del Proceso Unificado es que se enfoca mucho en los *componentes*, los cuales se pueden ver como objetos, donde dichos componentes se comunican entre sí por medio de *interfaces*; adicionalmente el Proceso

Unificado es un proceso iterativo e incremental, está dirigido por los casos de uso y es un proceso centrado en la arquitectura.

El Proceso Unificado puede verse como una matriz de dos dimensiones, formada por los *flujos de trabajo básicos*, conocidas como etapas o disciplinas en la terminología estándar, en cada una de las etapas se tienen cuatro fases: la fase de concepción, la fase de elaboración, la fase de construcción y la fase de transición; adicionalmente las fases se dividen en iteraciones, las cuales varían en función de las necesidades del proyecto en cada fase en una determinada etapa.

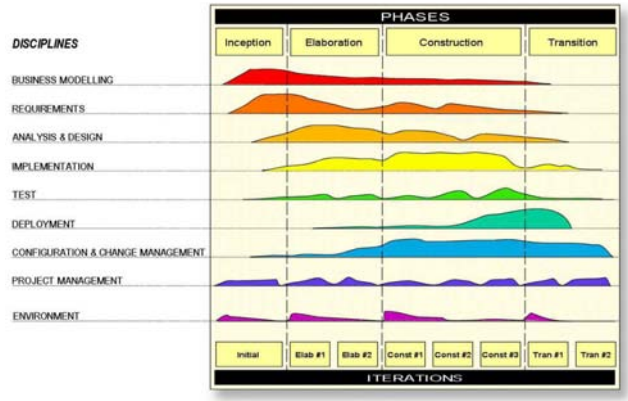


Figura 3-5: Braude, J. (2015). Matriz del Proceso Unificado. [Figura]

Este proceso produce a su vez seis modelos, el modelo de casos de uso que son los modos en los que la aplicación de software se utilizan, el modelo de análisis que describe las clases de la aplicación (recordando que se refiere a las clases en el lenguaje del paradigma orientado a objetos), el modelo de diseño que indica las relaciones entre las clases y los objetos del sistema, el modelo de despliegue que especifica la organización del código desarrollado, el modelo de implementación que ayuda a entender mejor el proceso de implementación del sistema desarrollado y el modelo de pruebas que se refiere a los casos de prueba y sus procedimientos.

De acuerdo con el documento de IBM (IBM, 1998), el Proceso Unificado dentro de las **Disciplinas de Ingeniería** se listan las siguientes etapas:

Modelación del negocio, en esta etapa se elabora el documento de *business object-model* que reflejan el entendimiento de cada uno de los *stakeholders* sobre el negocio y de las necesidades del mismo de cara al proyecto, a esto también se le conoce como Casos de Uso del Negocio.

Requerimientos, el objetivo principal de los requerimientos es saber *qué* es lo que el software debería de hacer y permitirle a los clientes y al equipo de desarrollo de software acordar lo que se tiene que hacer para lograrlo, se crea el documento que plasma la Visión, se identifican los Actores y los Casos de Uso representando el comportamiento del sistema; cada Caso de Uso se describe a detalle en esta etapa.

Análisis y diseño, esta etapa tiene como objetivo principal identificar el *cómo* se *realizará* el sistema en la fase de implementación, durante esta etapa se construye el Modelo de Diseño y a veces también el Modelo de Análisis, el modelo de diseño es una abstracción del código fuente del software, el modelo de análisis representa cómo se estructurará y se escribirá dicho código.

Implementación, durante la misma se tienen varios objetivos, tomando como base lo desarrollado en las disciplinas anteriores, en ella se tiene que definir la organización del código en términos de la implementación de los subsistemas organizados por capas, se tienen que implementar las clases y los objetos en términos de los componentes, se tienen que hacer pruebas de estos componentes y de sus unidades, para finalmente integrar los resultados obtenidos por cada implementador en un sistema ejecutable.

Pruebas, las pruebas tienen que verificar la interacción entre los objetos, que

todos los componentes estén correctamente integrados, que los requerimientos se hayan integrado correctamente en el sistema y tienen que identificar y ayudar a corregir los defectos que surjan tras ejecutarlas.

Despliegue, el propósito de esta etapa es construir una versión de producción del sistema desarrollado para ser entregado a los usuarios, esta etapa cubre varias actividades relacionadas con la entrega formal de la solución como, empaquetado, distribuciones, migraciones, instalaciones y la aceptación final del producto por parte del cliente.

Dentro de las **Disciplinas de Soporte** se listan las etapas siguientes:

Manejo del proyecto, se trata de la manera de lograr el balance entre cumplir los objetivos, manejar los riesgos, satisfacer los entregables y lograr la satisfacción del usuario de la solución implementada.

Configuración y control de cambios, la solución una vez en producción, es susceptible de ser modificada en aras de una mejor funcionalidad, por necesidades del negocio, por obsolescencia entre otras razones, es por ello que se tiene que definir una política de control de cambios que tenga en cuenta las necesidades del proyecto.

Entorno, el objetivo de esta etapa es configurar los procesos existentes dentro del contexto del proyecto, también se enfoca en las actividades necesarias para desarrollar las guías de soporte de la solución.

3.2.7. Metodologías Ágiles

Desde que se dedujo que la Industria del Software se encontraba en una *crisis* a finales de los años 60 hasta el periodo de inicios del siglo XXI, se fueron dando

avances fundamentales en la computación, se crearon los lenguajes de alto nivel, surgieron las bases de datos relacionales, aparecieron nuevas formas de interacción entre varias computadoras, nació la red de Internet, los circuitos digitales se fueron miniaturizando a velocidades sorprendentes (la llamada ley de Moore), pero los proyectos de implementación de software, siguieron siendo muy propensos a fracasar, tal como lo hacían en los años 60, los costos de hacer cambios posteriores a la definición de requerimientos siguieron siendo no lineales, la velocidad con la que se requerían los cambios no era la adecuada y muchos otros problemas asociados fueron creciendo en lugar de paliarse con el paso del tiempo.

Debido a esta situación, muchos consultores, empresas, programadores e ingenieros, usando su experiencia en proyectos de desarrollo de software pensaron en mejorar las metodologías existentes, basándose en el dinamismo, en la efectiva respuesta al cambio, en lugar de cumplir con documentos cerrados y poco modificables de acuerdos entre el cliente y el desarrollador.

Así fue que, de acuerdo con Roger Pressman (Pressman, 2010), en el año 2001, Kent Beck y otros formaron la llamada *Alianza Ágil* que lanzó el siguiente manifiesto:

"Estamos descubriendo formas mejores de desarrollar software, por medio de hacerlo y de dar ayuda a otros para que lo hagan. Ese trabajo nos ha hecho valorar:

-Los individuos y sus interacciones, sobre los procesos y las herramientas.
-El software que funciona, más que la documentación exhaustiva.
-La colaboración con el cliente, y no tanto la negociación del contrato.
-Responder al cambio, mejor que apegarse a un plan.

Es decir, si bien son valiosos los conceptos que aparecen en segundo lugar, va-

loramos más los que aparecen en primer sitio.”

Scrum

Según (Pressman, 2010) la metodología Scrum fue inventada por Jeff Sutherland y su nombre proviene de una cierta jugada realizada en el rugby, cuyos principios son congruentes con el manifiesto ágil y sirven para mejorar las fases de requerimientos, análisis, diseño, evolución y entrega, en esta metodología, las tareas ocurren dentro de un patrón llamado *sprint*, donde el trabajo realizado en un sprint, varía en función de la complejidad del proyecto y de lo que se requiera desarrollar en el mismo, la figura 3-6 muestra el flujo del proceso que sigue Scrum.

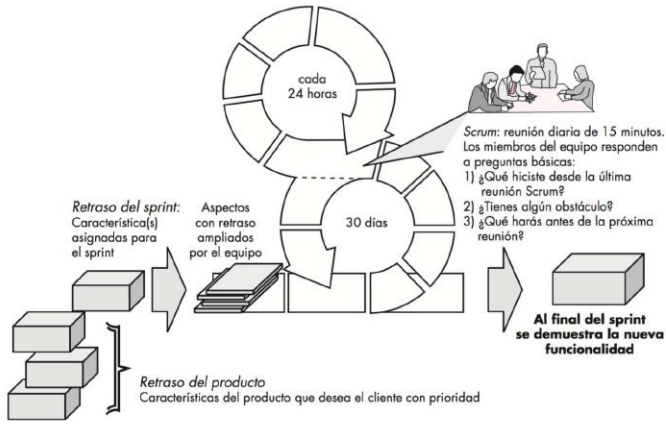


Figura 3-6: Pressman, R. (2010). Flujo principal del proceso Scrum. [Figura]

El retraso es la lista de requerimientos que le dan valor al desarrollo del proyecto desde la perspectiva del cliente, el gerente del proyecto evalúa el retraso y cambia las prioridades de ser necesario.

Los *sprints* son las unidades de trabajo necesarias para alcanzar un requerimiento definido en el retraso, durante la duración de un sprint no se introducen

cambios nuevos a desarrollar, de modo que se tiene un ambiente estable en el corto plazo, que es suficiente para ir cumpliendo con los requerimientos del proyecto.

Las reuniones Scrum (controladas por el *Scrum Master*) son juntas cortas de menos de quince minutos en las que el equipo se hace las siguientes preguntas:

¿Qué se hizo desde la última reunión?
¿Qué obstáculos se han encontrado para el desarrollo del proyecto?
¿Qué se planea hacer hasta la próxima reunión?

Tabla 3-1: Preguntas de SCRUM

El *Scrum Master* se encarga de recoger lo encontrado en estas juntas para poder encontrar problemas potenciales a tiempo y poder tomar acción directa para minimizar las afectaciones al proyecto.

Las demostraciones preliminares son el resultado del trabajo de los sprints, la funcionalidad desarrollada se le muestra el cliente para que éste pueda evaluar la funcionalidad, recordando que se trata de la funcionalidad como unidad, sin integrarse todavía al proyecto final.

eXtreme Programming (XP)

La metodología *eXtreme Programming*, de acuerdo a (Pressman, 2010) también es congruente con el manifiesto ágil, utiliza el paradigma de programación orientada a objetos y se enfoca en las fases de planeación, diseño, codificación y pruebas, se usa también en una forma industrial.

Esta metodología al ser de "programación extrema" (entiéndase esto como un desafío) se usa en proyectos difíciles, que requieren amplios y profundos conocimientos técnicos por parte del equipo de desarrolladores y se suele utilizar en

proyectos donde alguna otra metodología elegida anteriormente se demostró insuficiente.

Se trata de una metodología de tipo incremental, en la que se repiten las fases de planeación, diseño, codificación y pruebas hasta la aceptación del proyecto por parte de los *stakeholders*, como se muestra en la figura 3-7.

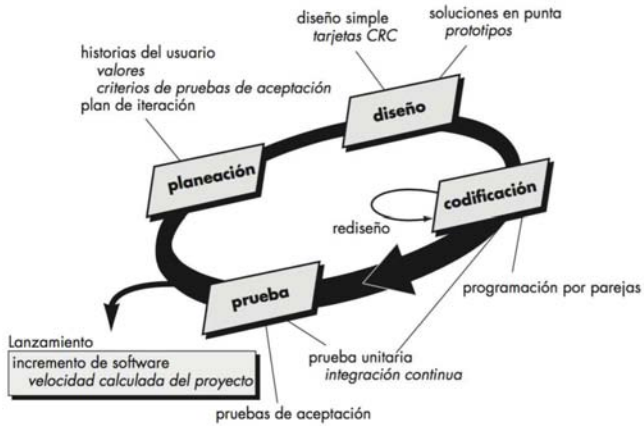


Figura 3-7: Pressman, R. (2010). Flujo principal del proceso eXtreme Programming. [Figura]

Las fases de la metodología *eXtreme Programming* son:

Fase de planeación

En XP, la planeación comienza escuchando, conociendo el contexto del proyecto, esto con el objetivo de sensibilizar al equipo de desarrollo con lo que el cliente necesita y entender mejor lo que se espera del proyecto, este trabajo de escuchar al usuario lleva a la creación de las *historias de usuario* que describen lo que cada interesado consultado durante la planeación necesitan del proyecto, ya sean las salidas necesarias, las características esperadas y la funcionalidad.

Cada historia de usuario es evaluada por el cliente, quien le asigna un número de prioridad y posteriormente el equipo XP le asigna un costo medido en semanas de desarrollo, de modo que, si una historia de usuario ocupa más de tres semanas de desarrollo, se le solicita al cliente que se descomponga en historias más pequeñas con el fin de simplificar el desarrollo y los avances requeridos durante las iteraciones XP.

Dado que se trata de una metodología ágil, en XP las historias de usuario se pueden ir ajustando de acuerdo a la necesidad del cliente, de modo que se pueden descomponer en más historias, se pueden eliminar, reclasificar y añadirse nuevas historias de usuario durante la duración del proyecto, ajustando en consecuencia todo el plan de entrega por parte del equipo XP con el acuerdo permanente y cercano del cliente; en efecto en la metodología XP, el cliente y el equipo de desarrollo trabajan muy de cerca con el fin de tener la mejor retroalimentación posible en el menor tiempo, lo cual no es posible de lograr tan rápido con las metodologías tradicionales.

Fase de diseño

La metodología XP se apega rigurosamente al principio: *keep it simple* (mantenlo sencillo), prescindiendo de mayor complejidad en aras de la simpleza y la claridad. XP usa las tarjetas CRC (clase, responsabilidad, colaborador), recordando que XP se usa en conjunto con un paradigma orientado a objetos, organizando las tarjetas CRC necesarias para el incremento actual del ciclo XP.

XP estimula el rediseño interno, de modo que se va mejorando el software internamente, cuando la funcionalidad básica ya fue alcanzada, esto permite ir aumentando la funcionalidad sin permitir la introducción de nuevos errores en el código del proyecto.

Fase de codificación

Antes de que el equipo de desarrollo comience a codificar, se tienen que hacer pruebas unitarias de cada historia de usuario, de modo que el desarrollador se dedica a cumplir con esta prueba unitaria sin añadir nada más de lo necesario (el principio *keep it simple*).

Una de las características principales de la metodología XP, es que el equipo de desarrollo trabaja en parejas, así se va codificando y asegurando que el código cumple con las especificaciones requeridas por el cliente y por el proyecto, adicionalmente hay un equipo de integración, que se encarga de ir implementando diariamente los avances de las parejas de desarrolladores; esto busca disminuir los tiempos de desarrollo y aumentar al mismo tiempo la calidad del código generado.

Fase de pruebas

Dado que para la metodología XP las pruebas unitarias tienen una gran relevancia, la fase de pruebas requiere que éstas se puedan automatizar para realizarse en múltiples ocasiones, esto tiene como objetivo el desarrollo de pruebas de regresión, para asegurarse de que el cambiante código (consecuencia misma de la metodología XP) siga siendo funcional para los escenarios ya aprobados. De este modo la metodología XP permite ir haciendo pruebas a diario e ir corrigiendo los errores conforme se encuentran, para evitar tener que hacer correcciones mayores cerca de la fecha de entrega del proyecto.

Finalmente en XP se definen las pruebas finales de aceptación del cliente, en el que el software desarrollado es probado, analizado y aceptado por el cliente del desarrollo basándose en las historias de usuario previamente definidas, y una vez que las pruebas se dan por satisfechas, se termina el desarrollo.

Agile Unified Process (AUP)

El *Agile Unified Process* (AUP) fue desarrollado por Scott W. Ambler como una combinación entre el *Rational Unified Process* (RUP) de IBM y la metodología de *eXtreme Programming* (XP), de modo que llega a ser una metodología muy orientada a los casos de uso pero contiene al mismo tiempo mayor rigor que la metodología XP sin llegar a usar toda la gama de herramientas que requiere la metodología RUP.

De acuerdo al sitio web oficial de esta metodología (Ambler, 2006), AUP utiliza un ciclo de vida serial a lo largo e iterativo en lo corto, entregando *releases* incrementales durante el tiempo del desarrollo del proyecto.

Las disciplinas de la metodología AUP, son muy parecidas a las que se utilizan en la metodología RUP (como se puede apreciar en la figura 3-8), las cuales son: la disciplina de modelación del proyecto, la de implementación, la de testing, la disciplina de despliegue, la de configuración de la gestión del proyecto, la gestión del proyecto como tal y el entorno del mismo; éstas disciplinas se usan de manera iterativa y definen las actividades que los miembros del equipo de desarrollo deben de construir y optimizar para cumplir las necesidades de los *stakeholders*. Por otra parte, las fases de la metodología AUP son la de concepción, la de elaboración, la de construcción y la de transición.

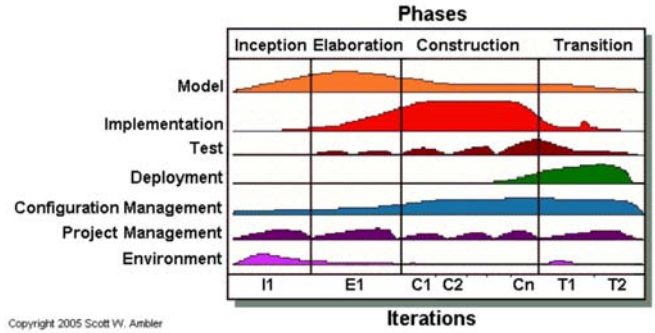


Figura 3-8: Ambler, S. W. (2006). Matriz del *Agile Unified Process*. [Figura]

Esta metodología es muy útil para desarrollos que requieren ser ágiles, donde no se conocen los requerimientos, permite que el equipo de desarrollo vaya adquiriendo las competencias necesarias para el proyecto durante el desarrollo del software y al mismo tiempo genera documentos lo suficientemente formales para cubrir las expectativas de un proyecto de software serio.

Según Scott Ambler en su trabajo sobre la metodología AUP, sigue de manera análoga a la metodología que le dio origen (RUP), las mismas Disciplinas (o Etapas) de Ingeniería, así como las mismas Fases (Ambler, 2006), las cuales son listadas a continuación:

Etapa de Modelado

Fase de concepción. Durante esta fase los *stakeholders* deberían participar activamente en los requerimientos de alto nivel que definen el alcance inicial para el proyecto para poder estimar los costos del mismo, se deben revisar los casos de uso, identificar los procesos del negocio, identificar las entidades más grandes del negocio generando un modelo de dominio, se deben encontrar las reglas del

negocio y los requerimientos técnicos; se debe comenzar con el desarrollo de un glosario para describir los términos técnicos, entender la estructura política de la comunidad de *stakeholders* usando un modelo de la organización, se deben tratar los requerimientos como una cola de prioridades que evoluciona en el tiempo.

Con toda esta información se crea un modelo de arquitectura de alto nivel inicial para identificar una estrategia de arquitectura viable para el proyecto.

Fase de elaboración. En esta fase se tienen que identificar los riesgos técnicos, basándose en los casos de uso y en los requerimientos técnicos que deberían revelar los riesgos técnicos del proyecto, también se modela la arquitectura del proyecto y se hace un prototipo de la interfaz de usuario.

Fase de construcción. Se debe trabajar de manera muy cercana con los *stakeholders* del proyecto para entender sus necesidades para lograr un modelado inclusivo que use herramientas simples y técnicas que sean criticas para el éxito del proyecto. Los casos de uso, las reglas del negocio y los requerimientos técnicos se pueden detallar usando diagramas de flujo o diagramas de actividad UML, se debe hacer un análisis de los sistemas antiguos o *legacy* para identificar la manera en la que se sustituirán dichos sistemas, se debe profundizar en los prototipos del producto y se tiene que mantener actualizado el glosario del proyecto.

Los diagramas que la metodología recomienda durante esta fase de la etapa de Modelado son: diagramas de secuencia UML, modelo de despliegue, diagramas libres que se escriben en pizarrón y se borran cuando ya no se necesitan, diagramas de clases UML, modelo de tratamiento de la seguridad y un modelo de datos físicos.

Fase de transición Se debe hacer trabajo de modelado para ir entendiendo las causas raíz de los posibles defectos que aparezcan durante el desarrollo del proyecto y se debe finalizar la documentación de primera vista del proyecto, es decir lo identificado durante la fase de construcción.

Etapa de Implementación

Fase de concepción. Se debe tener en cuenta cada pequeño detalle para poder hacer un estimado del esfuerzo, los prototipos son pequeños, son piezas de código, se debe hacer el prototipo de Interfaz de Usuario para convencer a los *stakeholders* de que el producto cumple con los requisitos de UI.

Fase de elaboración. Se debe probar la arquitectura, pues la actividad critica durante esta fase es identificar la arquitectura potencial y luego probarla para demostrar que la arquitectura elegida funciona para el sistema que se desarrolla, los prototipos técnicos se pueden pensar como el esqueleto del proyecto y deben tener una calidad similar a la de los sistemas en producción.

Fase de construcción. En esta fase se comienza haciendo pruebas, se tiene que ir construyendo continuamente, tiene que irse implementando la lógica del negocio en las clases de dominio y de negocio, se debe lograr una *usabilidad* suficiente para los usuarios mediante la interfaz, se deben desarrollar los conectores necesarios para la interacción con los sistemas legados y se deben escribir los scripts de conversión de datos entre los sistemas legados y el nuevo desarrollo (Ambler, 2006).

Fase de transición. Se debe tener un enfoque a resolver los defectos encontrados durante las pruebas de la fase anterior.

Etapa de Pruebas

Fase de concepción. Durante esta fase se elabora un plan de pruebas inicial, se tiene que identificar la cantidad de pruebas que se tienen que realizar, quién será el responsable de hacerlo, el nivel de participación de los *stakeholders* en ellas y

las herramientas que se necesitarán para hacerlas. Se hace una revisión inicial de productos funcionales del proyecto, para esto es necesario que se tenga el plan inicial del proyecto y la visión del mismo, se eligen *stakeholders* clave para revisar el producto en esta fase. Finalmente se hace una revisión de los modelos iniciales, es decir, el modelo inicial de requerimientos y el modelo inicial de arquitectura.

Fase de elaboración. Se valida la arquitectura, se muestran los resultados de los esfuerzos hechos en los prototipos a los *stakeholders* del proyecto. El modelo de pruebas debe evolucionar en esta fase, se tiene que hacer una suite de pruebas de regresión, se deben definir las pruebas unitarias, pruebas de aceptación, pruebas del sistema tanto de funcionalidad, de integración, de carga, etc. Se debería tener también el trazado entre los requerimientos, las pruebas y el código fuente para mostrar cómo se validó cada uno de los requerimientos implementados en la solución.

Fase de construcción. Aquí es donde se hacen las pruebas al software desarrollado como tal, se deben probar los scripts de despliegue, se hacen pruebas de estrés, de carga, de funcionalidad y de aceptación del usuario. Se debe poder alcanzar la calidad suficiente para poder migrar el software a un entorno de pre-producción.

Fase de transición. Se valida el sistema, haciendo pruebas del sistema, de integración, de aceptación y pruebas de piloto, el objetivo es probar todo el sistema en un ambiente de pre-producción. La validación de la documentación es otro de los objetivos de esta fase, se trata de los documentos de uso, de soporte, de operación y los materiales de aprendizaje del sistema para capacitar a los usuarios finales del mismo.

Al final de esta fase se tiene que terminar con el modelo de pruebas, actualizando las pruebas regresión tanto como sea necesario hasta que el sistema esté listo para el despliegue en el ambiente de producción, los defectos encontrados se

implementarán posteriormente como fixes (Ambler, 2006).

Etapa de Despliegue

Fase de concepción. Se identifica la ventana potencial de despliegue, se debe decidir la fecha adecuada para lanzar el producto tomando en cuenta las necesidades del negocio y las variables asociadas que no estén bajo el control del equipo de desarrollo y de los *stakeholders* y se hace un plan inicial de alto nivel del despliegue de la solución en producción.

Fase de elaboración. El plan de despliegue tiene que ser actualizado tomando en cuenta todo lo desarrollado en las etapas previas, se deben definir las configuraciones de despliegue del sistema y se debe documentar cada configuración en el modelo de despliegue que también debe definir cómo es que los componentes del software son organizados y desplegados en los componentes de hardware.

Fase de construcción. En esta fase se desarrollan los scripts de instalación del sistema, se hacen las notas de lanzamiento, se hace la documentación inicial, se actualiza el plan de despliegue y se despliega el sistema en los entornos de preproducción para revisiones finales del conjunto de los componentes del sistema desarrollado en el proyecto.

Fase de transición. Durante la fase de transición se finaliza con el paquete de despliegue, se tiene que terminar la documentación, se hace el anuncio de la fecha del despliegue, se entrena a los usuarios finales del sistema para finalmente instalar el sistema en el ambiente de producción, teniendo de forma ideal un ambiente que simule las mismas condiciones de producción para poder atender incidentes emergentes tras la puesta en producción del sistema (Ambler, 2006).

Disciplinas de Soporte

La metodología AUP sugerida por Ambler, también incluye las Disciplinas de Soporte (Manejo del proyecto, Configuración y control de cambios y Entorno) de la metodología RUP, pero dichas etapas no se listan pues son muy similares a las de la metodología RUP y los alcances del proyecto solo llegaron hasta la puesta en producción de la solución documentada en el presente texto.

3.3. Diagnóstico del problema

El Sistema de Gestión de Asistencias de la Facultad de Contaduría y Administración es la respuesta a la problemática que se vive en la Coordinación de Estadística y Evaluación del SUAyED, debido a que actualmente se lleva un proceso a mano, propenso a errores humanos en el que se tienen que realizar muchos pasos intermedios para obtener las salidas de datos necesarias para la operación del área:

-Durante el semestre, los asesores del sistema a distancia se tienen que conectar a la plataforma en línea de la facultad http://fcaenlinea1.unam.mx, en los horarios indicados para su asignatura, desplegando la plataforma en la sala de chat del sistema, una vez identificados, tienen que estar dando clic cada cierto tiempo para mostrar actividad en la sala de chat en espera de alumnos que requieran asesoría; una vez finalizado el plazo de la asesoría (dos horas normalmente), el asesor cierra sesión y se da por concluida la asesoría.

-Estos datos quedan registrados en el servidor del SUAyED de la UNAM, que se obtienen de este servidor en formato xls para cada licenciatura ofertada por la Facultad: a saber Administración, Contaduría e Informática, se bajan estos registros manualmente, se hace un concentrado en un archivo de Excel, que contiene diversas macros que calculan el número y los porcentajes de asistencias, faltas,

reposiciones, justificaciones y demás situaciones que se van presentando durante el semestre.

-Mensualmente se envía un correo a cada asesor indicando su porcentaje de asistencias, faltas y si las tuvo, en qué fecha se tiene registrada la misma, así como las reposiciones a las que tiene derecho para reponer hasta dos asesorías durante el semestre en las que haya tenido falta, no obstante, las faltas que tenga en las reposiciones no se deben de contar como faltas en el apartado destinado para ello. Los porcentajes de asistencias se tienen que redondear, si la parte decimal es mayor o igual que 5, sube al número entero inmediato superior, en caso de no ser así, baja al número entero inmediato inferior.

-Se tiene que realizar un reporte al finalizar el semestre en formato PDF con un estilo ya definido y aprobado, que contenga los datos del asesor, la licenciatura, el grupo, la asignatura, el horario, los porcentajes y el detalle de las faltas y asistencias durante el semestre.

-El principal problema que tiene el modo de operar anterior al sistema es la posibilidad de ir cometiendo errores de captura y que estos se vayan reproduciendo durante la duración del semestre, dado que no existe un mecanismo para garantizar la integridad de la información, adicionalmente todo el trabajo de generar el reporte PDF y enviar los correos electrónicos es manual, cuando se tienen más de 400 grupos en el SUAyED de la Facultad, tomando demasiado tiempo y esfuerzo en una tarea poco productiva para el área.

Tomando estas premisas como base para la propuesta de solución, se decidió crear un sistema que tuviera en su centro a una base de datos normalizada, que permitiera hacer diversas consultas para generar los reportes solicitados, el envío de correos electrónicos y que tuviera capacidad de actualizar los registros necesarios en caso de que hubiera alguna modificación a los registros de la base de datos.

Fue necesario definir con el usuario *qué* datos se podrían modificar directamen-

te y cuales tendrían que cargarse vía archivo CSV para garantizar que el sistema tuviera información correcta y coherente en todo momento, además resultó necesario poder actualizar los registros ya insertados cargando el respectivo archivo CSV sin necesidad de ejecutar el flujo principal del sistema.

Capítulo 4

Propuesta y proceso de desarrollo de la solución

4.1. Propuesta implementada

Se propuso el sistema de Gestión de Asistencias de Asesores que garantiza que la información correcta generará datos correctos, que la información incorrecta no generará datos debido a los filtros que contiene y a la normalización de la base de datos, genera el reporte PDF de una sola vez en un solo archivo, listo para imprimir, envía los correos electrónicos a los asesores de un solo golpe, generando una copia en formato de texto de cada correo enviado para poder tener control de lo que se le mandó a cada asesor, genera la salida requerida en formato CSV que se puede importar directamente en Excel.

El sistema utiliza una interfaz web que simplifica el entendimiento del mismo para el usuario, además ayuda a simplificar el funcionamiento de los distintos componentes del sistema cuyo origen es heterogéneo, además la interfaz se puede ampliar fácilmente al estar basada en HTML5 y en una plantilla web fácil de modificar y de ampliar.

También se incluye en la interfaz del sistema los vídeos de uso solicitados por los usuarios, para tener mayor claridad en los procesos que se tienen que realizar dentro de la operación del sistema una vez en producción.

El desarrollo utilizó dos ambientes, uno de pruebas (en una máquina virtual Debian con las mismas versiones que el servidor del CIFCA) y el ambiente productivo montado en el servidor del CIFCA aprobado para la implementación del proyecto.

4.2. Metodología utilizada

Si se entiende por *metodología* al uso de un método especifico conjugándolo con un ciclo de vida, es necesario definir qué método de desarrollo y qué tipo de ciclo de vida se utilizó para el desarrollo del proyecto; tras lo analizado anteriormente, la metodología seleccionada para el desarrollo e implementación del proyecto fue una de las metodologías ágiles que cuentan al mismo tiempo con un grado de rigor mayor que algunas otras metodologías de esta familia.

Método utilizado

El método utilizado para el desarrollo del sistema fue el de desarrollo de software de aplicación, concretamente el *Proceso Unificado Ágil*.

Según (Martínez y Chávez, 2010) el tipo de ciclo de vida que utiliza el Proceso Unificado:

"emplea un ciclo de vida iterativo e incremental con cuatro fases: iniciación, elaboración, construcción y transición. Dentro de este ciclo se desarrollan actividades asociadas con nueve procesos, también llamados disciplinas. De estos nueve procesos, seis se clasifican como clave y tres como procesos de soporte."

Sin embargo, el Proceso Unificado, aún recortado, representa una mayor carga para un proyecto relativamente pequeño como el que se desarrolló, dejando en claro que este proceso resulta ser muy útil en proyectos más grandes, tanto en recursos humanos, técnicos, financieros y alcances; así pues la solución para el

desarrollo del proyecto se buscó en las metodologías ágiles.

Una cuestión importante que decantó la metodología utilizada para este proyecto fue que durante el proceso de desarrollo, se tuvo que ir aprendiendo a conjugar los distintos elementos que finalmente conformaron el sistema, debido a que faltaba experiencia previa usando todas las herramientas en conjunto, de modo que, al presentar los avances parciales obtenidos a los interesados en el sistema, basándose en las ideas y peticiones que iban surgiendo, se fueron dando más cambios y el alcance del sistema en consecuencia se tuvo que ir ajustando para cubrir las expectativas del usuario.

Se usó el Proceso Unificado Ágil debido a que durante el proceso de desarrollo del sistema no se tuvieron todos los requisitos bien definidos por el usuario y a que se tomaban sobre la marcha decisiones de diseño y de características que debería cumplir el sistema. Según lo comentado en el presente texto, el Proceso Unificado Ágil representa un punto medio entre una metodología ágil como XP y una metodología estructurada como el Proceso Unificado, guardando el nivel de rigor necesario para un desarrollo de seriedad pero permitiendo al mismo tiempo adaptarse rápidamente al cambio y a la curva de aprendizaje de las herramientas de desarrollo.

La retroalimentación entre interesados y desarrollador resultó ser muy importante, permitió afinar el sistema, exceder los alcances originales propuestos y se encontraron tareas de mejora que no le corresponden al área involucrada pero que se pueden ir solicitando con una mejor comprensión de lo que se requiere para simplificar aún más el trabajo diario en el área.

Ciclo de vida utilizado

El desarrollo del sistema fue incremental (es decir, mediante incrementos se le fue añadiendo funcionalidad al sistema hasta la plena aceptación de la Coordinación del mismo), por lo que, de entre todos los ciclos de vida incrementales e iterativos de desarrollo de proyectos, se eligió el ciclo evolutivo debido a que no todos los requisitos eran conocidos por el usuario en el momento de comenzar el proyecto, muchos de ellos se fueron encontrando sobre la marcha, algunos requisitos técnicos fueron descubiertos y definidos en el momento de desarrollar la solución, teniendo que adaptar el sistema para poder cumplir cada uno de los requisitos, también se tuvo que ir añadiendo funcionalidad y de apartados de interfaz para poder integrar los componentes de la solución desarrollada.

Los ciclos de revisión con el usuario se acordaron de forma quincenal, la duración del proyecto fue de 18 semanas, resultando en cuatro iteraciones y algunas reuniones para puntualizar casos de uso y requisitos, la instalación de la primera versión del sistema en el servidor productivo se dio en la tercer iteración, dejando la última iteración en un proceso de desarrollo y testing en el ambiente de pruebas, para posteriormente subir las versiones actualizadas al servidor de producción con el visto bueno de los interesados del sistema.

En la parte de la documentación, se requirió generar un manual técnico con todas las especificaciones necesarias para poder implementar de cero el sistema en un servidor nuevo, un manual de usuario con capturas de pantalla que ilustren la forma de uso completa del sistema en sus casos generales, así como vídeos de uso que deberían estar accesibles desde el propio sistema, se acordó subir estos vídeos a una plataforma de vídeos en streaming de forma restringida para garantizar su disponibilidad y confidencialidad.

De acuerdo con la documentación de AUP (*Agile Unified Process*), no se defi-

nen tantos documentos como en la metodología RUP (*Rational Unified Process*), se basa fuertemente en los casos de uso, heredando esta característica de la metodología XP (*eXtreme Programming*) y las pruebas se hicieron en base a los casos de uso analizados durante cada iteración del sistema.

En los primeros tres ciclos de desarrollo del sistema, se trabajó con una máquina virtual con las mismas características técnicas que el servidor de producción del CIFCA asignado para el sistema, los avances se mostraron y se discutieron en juntas acordadas para tal fin durante los días del periodo de testing; al finalizar dicho periodo se procedió con la implementación en producción de la solución terminada y se realizó otra etapa de pruebas para verificar que todos los componentes funcionaran adecuadamente en el servidor productivo; finalmente la ejecución productiva para el primer semestre en el que el sistema operó (2016-2), se hizo presencialmente con el usuario resolviendo las dudas que surgieron de los flujos de ejecución del sistema.

4.3. Definición de los alcances

Junto con los *stakeholers* se definieron los siguientes alcances para el Sistema de Gestión de Asistencias:

El Sistema de gestión de asistencias se alimentará de los datos obtenidos por los empleados de la Coordinación de Estadística y Evaluación, los cuales son tomados de Excel pues los datos provenientes de la plataforma Moodle contienen muchos datos irrelevantes, de prueba y de semestres anteriores.
Dado el punto anterior, el sistema no tiene como alcance obtener los datos de manera directa y automática de los servidores de la plataforma Moodle, eso quedará para un desarrollo posterior en la Facultad.

La solución desarrollada implementará una interfaz web para su uso por parte de los empleados de la Coordinación.
Se deben generar los reportes especificados en el formato especificado.
El envío de correos electrónicos a los asesores debe poder ser revisado en cada uno de los correos que se generen en el sistema.
El sistema debe permitir hacer respaldos y actualizar los datos existentes directamente.
Se deberá documentar lo necesario, a nivel técnico y a nivel usuario para que el sistema pueda ser adaptado, modificado, migrado y mejorado posteriormente.
Cualquier punto no especificado se pudo discutir durante el desarrollo del proyecto para determinar si se podía implementar o no tras un breve análisis de factibilidad.

Tabla 4-1: Alcances del sistema

Tras definir los alcances que tendría el proyecto, fue que se pudo comenzar su análisis, planeación y ejecución con la seguridad de que no habría disgustos o malos entendidos por parte de alguno de los *stakeholers* del proyecto.

4.4. Requisitos Funcionales

Los Requisitos Funcionales identificados por el programador, por los usuarios y los interesados, fueron:

-Se debe entregar el sistema en una interfaz web, sencilla, intuitiva pero al mismo tiempo segura.

-Los datos deben estar almacenados en una base de datos normalizada hasta la tercera forma normal.
-La propuesta de solución debe poder gestionar más de 400 grupos del SUAyED de la Facultad de Contaduría y Administración.
-La información que alimentará al sistema proviene de dos archivos de Excel, el sistema debe generar de forma automática todo aquello que sea necesario para que los datos sean coherentes y consistentes.
-Entregar el reporte asistencias de cada asesor en un solo archivo PDF.
-El reporte debe de generarse con un formato especifico, debe ser idéntico al formato existente.
-Dicho reporte debe contener algunos datos calculados por el sistema que no están presentes en los archivos de entrada (porcentajes, número de asistencias, faltas y reposiciones).
-Las asistencias y las faltas se suman para calcular el total de sesiones del asesor.
-Las reposiciones se cuentan, si tienen asistencia cuentan como asistencia normal, pero las faltas de las reposiciones solo se muestran, no cuentan para el promedio general del asesor.
-La parte decimal del promedio de asistencias debe redondearse hacia el entero inmediato inferior si es menor a .5 y hacia el entero inmediato superior si es igual o mayor que .5
-Se debe de mostrar el total de reposiciones inscritas por el asesor durante el semestre.

-Se debe contar con la funcionalidad de envío de correos electrónicos automatizados a los asesores de las asignaturas que se imparten en el SUAyED.
-Es necesario que los correos enviados se puedan consultar para tener seguridad de lo que se envió a cada asesor en caso de alguna reclamación.
-Adicionalmente, se debe tener la opción de habilitar y deshabilitar el envío efectivo de los correos electrónicos a los asesores, pero generando los archivos que se enviarían.
-Se debe crear un reporte a final de semestre con estadísticas generales que se entregan a la Coordinación Administrativa del SUAyED de la Facultad.
-Se debe de poder editar desde el sistema los registros de los asesores para hacer correcciones menores (justificación de faltas, por ejemplo).

Tabla 4-2: Requisitos Funcionales

4.5. Requisitos No Funcionales

Los Requisitos No Funcionales identificados por el desarrollador de la solución y por los interesados en el proyecto, fueron:

-Se requiere que la solución se haga a base de Software Libre.
-Se requiere que el sistema quede instalado en un servidor de la Facultad.

-Dada la naturaleza de la información, es necesario que el sistema maneje algún tipo de contraseña de acceso para evitar incidentes de seguridad.
-La interfaz debe ser fácil de usar, de preferencia sin necesidad de instalar software especial.
-Es necesario tener apartados de mantenimiento que de manera sencilla limpien el sistema para un nuevo semestre.
-Se debe de poder generar un respaldo de la información contenida en el sistema.
-El respaldo generado a su vez, se debe poder cargar nuevamente en el sistema (para futuras migraciones de infraestructura).
-Las soluciones de software utilizadas deben ser en su mayoría software estándar fácil de obtener.
-Dado que el origen de los datos proviene de dos fuentes, que el sistema implemente una base de datos normalizada, para garantizar que los datos son consistentes.
-El sistema tiene que poder operar en distintos sistemas operativos (Windows, Linux, OS X) y navegadores web (Internet Explorer, Safari, Firefox, Chrome).
-Migrar el sistema de servidor debe ser una tarea relativamente sencilla, todo el sistema deberá estar localizado dentro de un directorio.
-El rendimiento final de la solución debe ser adecuado pensando en el crecimiento futuro del volumen de la información que se manejará en la Facultad.

-Se debe entregar el primer reporte completo para el final del semestre de implantación (2016-2).
-Generar el manual técnico del sistema y un manual de uso con capturas de pantalla.
-Se deben hacer vídeos de uso que le enseñen al usuario cómo se utiliza el sistema en la mayoría de sus casos.

Tabla 4-3: Requisitos No Funcionales

4.6. *Stakeholders* del sistema

Los *stakeholders* del Sistema de gestión de asistencias son todos aquellos que pueden resultar beneficiados y/o afectados por el sistema, se trata de los usuarios del sistema, la jefatura de la Coordinación donde se implementó el proyecto, las personas a las que se les entregan las salidas, los encargados de la infraestructura del sistema y el propio desarrollador del proyecto.

-Gabriela Montero Montiel, jefa de la División del Sistema de Educación Abierta y a Distancia de la Facultad de Contaduría y Administración de la UNAM.
-Martha Patricia García Chavero, jefa de la Coordinación Académica.
-Marlene Olga Ramírez Chavero, jefa de la Coordinación de Estadística y Evaluación.
-Beatriz Sánchez Osornio, jefa de la Coordinación Administrativa.
-Pedro Gutiérrez Delgado, encarado del Centro de Educación a Distancia.

-Germán Ignacio Cervantes, del Centro de Informática de la Facultad de Contaduría y Administración (CIFCA).
-Erika Angelina Nuevo Esteves, de apoyo a la Coordinación de Estadística y Evaluación.
-Oswaldo Méndez Chepe, de apoyo a la Coordinación de Estadística y Evaluación.
-Rogelio Antonio Guzmán Bermúdez, de apoyo a la Coordinación de Estadística y Evaluación.
-Carlos Alvarado Martínez, Desarrollador del Sistema de Gestión de Asistencias.

Tabla 4-4: Stakeholders del sistema.

Dado que se eligió una metodología ágil, se trabajó muy de cerca con todos los *stakeholders* del sistema, hasta lograr la satisfacción completa respecto a sus inquietudes y necesidades con respecto al proyecto, el soporte técnico quedó asignado al área del CIFCA que tienen los documentos técnicos necesarios para implementar y modificar el sistema posteriormente.

4.7. Análisis de riesgos

Los riesgos principales encontrados durante la planeación y el desarrollo del sistema fueron:

-Problemas de implementación en los servidores de la facultad.
-Incidentes con la red que imposibilitaran la operación adecuada del sistema.

-Ataques externos al sistema por parte de hackers.
-Robo de información sensible para el área por personas ajenas al área.
-Errores en los scripts del sistema que hagan que la información deje de ser coherente.
-Problemas con el hardware del servidor donde está alojado el sistema que comprometan su disponibilidad.
-Fallos con las bases de datos del sistema desarrollado o de sus procedimientos almacenados.
-Incompatibilidad de nuevas versiones Java con respecto a los reportes en Jasper Reports desarrollados en el sistema.
-Virus informáticos, malware, troyanos, spyware, corrupción de datos que impidan un correcto funcionamiento del sistema.
-Errores en la interfaz de usuario que no permitan operar.
-Cambios en la operación del SUAyED de la Facultad que dejen obsoleto al sistema desarrollado.
-La resistencia al cambio de los usuarios que podrían preferir seguir operando como lo venían haciendo.
-La no aprobación de la Coordinación del área para su implementación final.
-Cuestiones no previstas durante el diseño del sistema que se conviertan en un problema grave.

Tabla 4-5: Riesgos del sistema.

Los riesgos asociados a la implementación del sistema resultaron no afectar mucho el desarrollo y el despliegue del mismo, empero los riesgos de operación seguirán durante toda la vida útil del sistema en producción.

4.8. Cálculo del costo de desarrollo del Sistema de gestión de asistencias.

El Sistema de gestión de asistencias se desarrolló durante 18 semanas en 480 horas en total, repartidas de la forma más conveniente para los *stakeholders* del sistema.

Para estimar el costo del desarrollo de este sistema se toma como base el salario promedio para un profesionista de las Tecnologías de la Información en la Ciudad de México durante el año 2015 y según el artículo "Sueldos IT en México 2015: de startups y desarrollo móvil" (::everac99, 2015) fue de $36,764.00 M.N. mensuales brutos, que equivalen a aproximadamente $1225.46 M.N. por día. Si se considera que el día laboral es de 8 horas, el costo por hora quedaría en $153.18 M.N.

Entonces el cálculo del costo total del sistema queda de la siguiente manera:

$153.18 M.N. costo de desarrollo promedio por hora.
x 480 horas acordadas para la duración del proyecto.
= $73,526.40 M.N. sería el costo total del desarrollo del proyecto.

Tabla 4-6: Estimación del costo de desarrollo del sistema.

Adicionalmente a este costo, se tendría que sumar el costo de la adquisición y despliegue de la infraestructura de red, de almacenamiento, de equipo de cómputo, se tendrían que añadir los salarios de los interesados del sistema (que son empleados o eran becarios de la Facultad durante el proyecto), entre otros; estos gastos no se incluyeron porque son gasto corriente de la Facultad. La infraestructura destinada al proyecto es la misma con la que ya se contaba, por lo que se decidió no incluir estos costos en la presente estimación de costos; claro está que al ser un proyecto de titulación, la implementación realizada no representó ningún costo

para la Facultad.

4.9. Gestión de la Seguridad.

El Sistema de gestión de asistencias es un sistema basado en una interfaz web, que cuenta con un password de acceso (se maneja internamente con md5) al cual solo tiene acceso la administración del área de la Coordinación de Estadística y Evaluación, el sistema se aloja en un dominio de la UNAM de acceso interno, por lo que los ataques externos serán difíciles de llevar a cabo, la base de datos también cuenta con su protección por contraseña para entrar en cualquiera de sus interfaces.

Por otra parte, el proyecto se implementó sobre un servidor que ejecuta una distribución Debian Linux, usando Apache como servidor HTTP y una base de datos PostgreSQL, los cuales tienen un manejo de la seguridad bastante elevado, se previenen ataques a la infraestructura de red de la Facultad con firewalls y con un sistema de permisos que usa ACL (*Access Control List*); el usuario asignado en el sistema operativo no tiene permisos suficientes para interferir con los demás usuarios del servidor, de modo que en el peor de los casos la afectación quedaría reducida a los archivos del propio usuario y ninguno más.

Nivel de sistema operativo.	Permisos de Linux e implementación de Listas de Control de Acceso (ACLs).
Nivel de red.	Firewall activo en modo restrictivo y red interna de la UNAM.
Nivel de base de datos.	Base de datos protegida por contraseña encriptada.

Nivel de interfaz del sistema.	Password encriptado con MD5.
Nivel de aplicación.	Aplicaciones actualizadas, con los últimos parches disponibles instalados.

Tabla 4-7: Análisis de la seguridad del proyecto.

4.10. Diccionario de datos

El diccionario de datos del sistema de gestión de Asistencias de asesores, se muestran los datos generados por el mismo sistema, la tabla a la que pertenecen y el archivo del que cada dato se origina cuando así aplique.

Nombre del dato	**Tipo de dato**	**Tabla de origen**	**Archivo origen**	**Generado**
cempleado	Integer	Sesiones	Sesiones.csv	No
cmateria	Integer	Sesiones	Sesiones.csv	No
semestre	Varchar(20)	Sesiones	Sesiones.csv	No
horas	Integer	Sesiones		Sí
pksesiones	Varchar(50)	Sesiones		Sí
carrera	Varchar	Sesiones	Sesiones.csv	No
clave	Integer	Sesiones	Sesiones.csv	No
asesor	Varchar(150)	Sesiones	Sesiones.csv	No
asignatura	Varchar(100)	Sesiones	Sesiones.csv	No
grupo	Integer	Sesiones	Sesiones.csv	No
horario	Varchar(80)	Sesiones	Sesiones.csv	No
dia	Varchar(20)	Sesiones	Sesiones.csv	No

s1fecha	Varchar(50)	Sesiones	Sesiones.csv	No
s1hora	Varchar(50)	Sesiones	Sesiones.csv	No
s1estatus	Varchar(30)	Sesiones	Sesiones.csv	No
s2fecha	Varchar(50)	Sesiones	Sesiones.csv	No
s2hora	Varchar(50)	Sesiones	Sesiones.csv	No
s2estatus	Varchar(30)	Sesiones	Sesiones.csv	No
[...]	[...]	[...]	[...]	[...]
s56fecha	Varchar(50)	Sesiones	Sesiones.csv	No
s56hora	Varchar(50)	Sesiones	Sesiones.csv	No
s56estatus	Varchar(30)	Sesiones	Sesiones.csv	No
r1fecha	Varchar(50)	Sesiones	Sesiones.csv	No
r1hora	Varchar(50)	Sesiones	Sesiones.csv	No
r1estatus	Varchar(30)	Sesiones	Sesiones.csv	No
r2fecha	Varchar(50)	Sesiones	Sesiones.csv	No
r2hora	Varchar(50)	Sesiones	Sesiones.csv	No
r2estatus	Varchar(30)	Sesiones	Sesiones.csv	No
cempleado	Integer	DSesiones	Sesiones.csv	No
clave	Integer	DSesiones	Sesiones.csv	No
tsesiones	Integer	DSesiones		Sí
asistencias	Integer	DSesiones		Sí
faltas	Integer	DSesiones		Sí
treposiciones	Integer	DSesiones		Sí
porcentajes	Integer	DSesiones		Sí
pkdsesiones	Varchar(50)	DSesiones		Sí
semestre	Varchar(20)	Periodo		Manualmente

dsemestre	Varchar(100)	Periodo		Manualmente
numempleado	Integer	Asesor	Planta.csv	No
nombre	Varchar(60)	Asesor	Planta.csv	No
apparterno	Varchar(60)	Asesor	Planta.csv	No
apmaterno	Varchar(60)	Asesor	Planta.csv	No
nombrecompleto	Varchar(183)	Asesor	Planta.csv	No
rfc	Varchar(30)	Asesor	Planta.csv	No
curp	Varchar(30)	Asesor	Planta.csv	No
correo1	Varchar(70)	Asesor	Planta.csv	No
correo2	Varchar(70)	Asesor	Planta.csv	No
cveasignatura	Integer	Asignatura	Planta.csv	No
tipoasignatura	Varchar(50)	Asignatura	Planta.csv	No
asignatura	Varchar(100)	Asignatura	Planta.csv	No
carrera	Varchar(50)	Grupo	Planta.csv	No
plan	Integer	Grupo	Planta.csv	No
semestre	Integer	Grupo	Planta.csv	No
coordinacion	Varchar(50)	Grupo	Planta.csv	No
cveasignatura	Integer	Grupo	Planta.csv	No
grupo	Integer	Grupo	Planta.csv	No
dia	Varchar(20)	Grupo	Planta.csv	No
hora	Varchar(80)	Grupo	Planta.csv	No
salon	Varchar(20)	Grupo	Planta.csv	No
numempleado	Integer	Grupo	Planta.csv	No
pkgrupo	Varchar(50)	Grupo		Sí

Tabla 4-8: Diccionario de datos del sistema.

4.11. Diagramas del Sistema

4.11.1. Diagrama de Gantt de la implementación

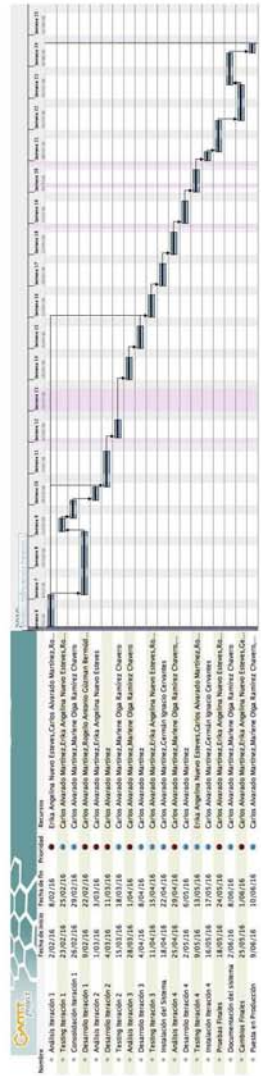


Figura 4-1: Diagrama de Gantt de la implementación

4.11.2. Diagrama de Gantt de los recursos humanos

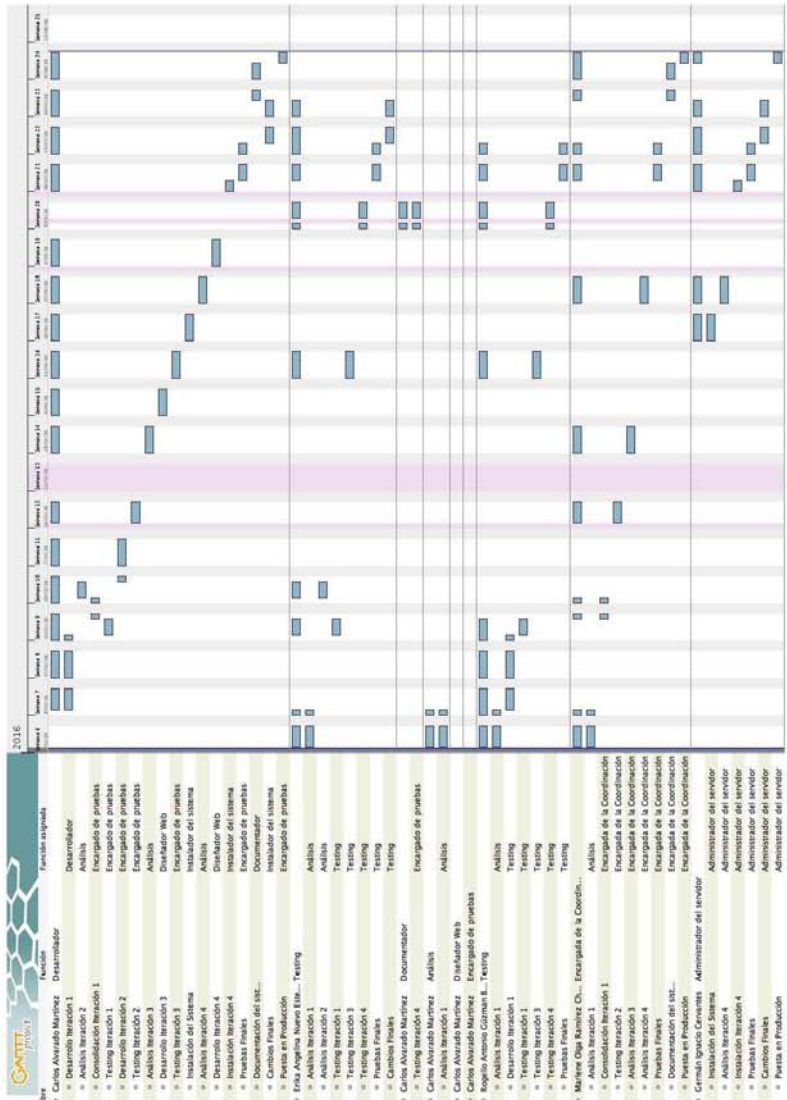


Figura 4-2: Diagrama de Gantt de recursos humanos

4.11.3. Modelo de despliegue

El modelo de despliegue del sistema quedó de la siguiente manera:

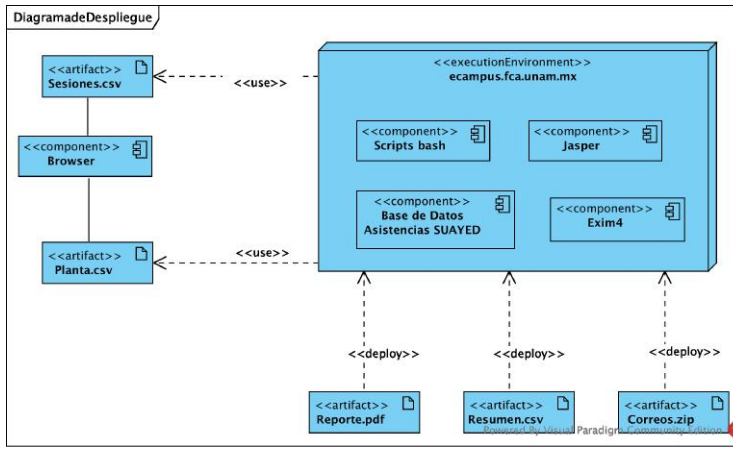


Figura 4-3: Diagrama de Despliegue del proyecto.

4.11.4. Diagrama de Clases UML

Diagrama de Clases UML del Sistema de gestión de asistencias.

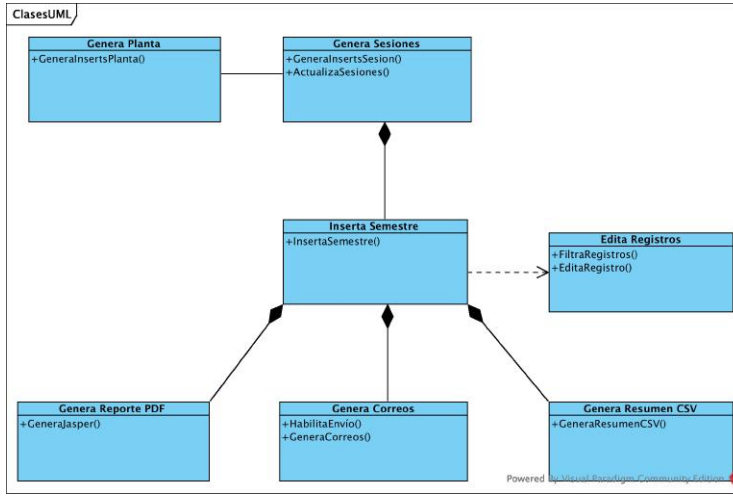


Figura 4-4: Diagrama de clases UML del sistema.

4.11.5. Modelo Físico.

Modelo Físico del Sistema de gestión de asistencias.

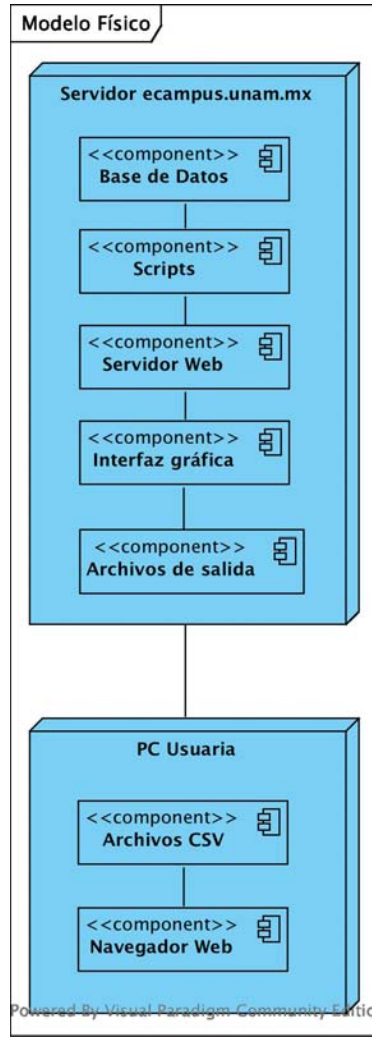


Figura 4-5: Modelo Físico del Sistema de gestión de asistencias.

4.11.6. Diagrama de componentes

El diagrama de componentes del sistema muestra los elementos que lo componen y su forma de interacción entre sí, se muestra cómo los archivos PHP, tanto de interfaz como de funciones, interactúan con los scripts en shell de Bash.

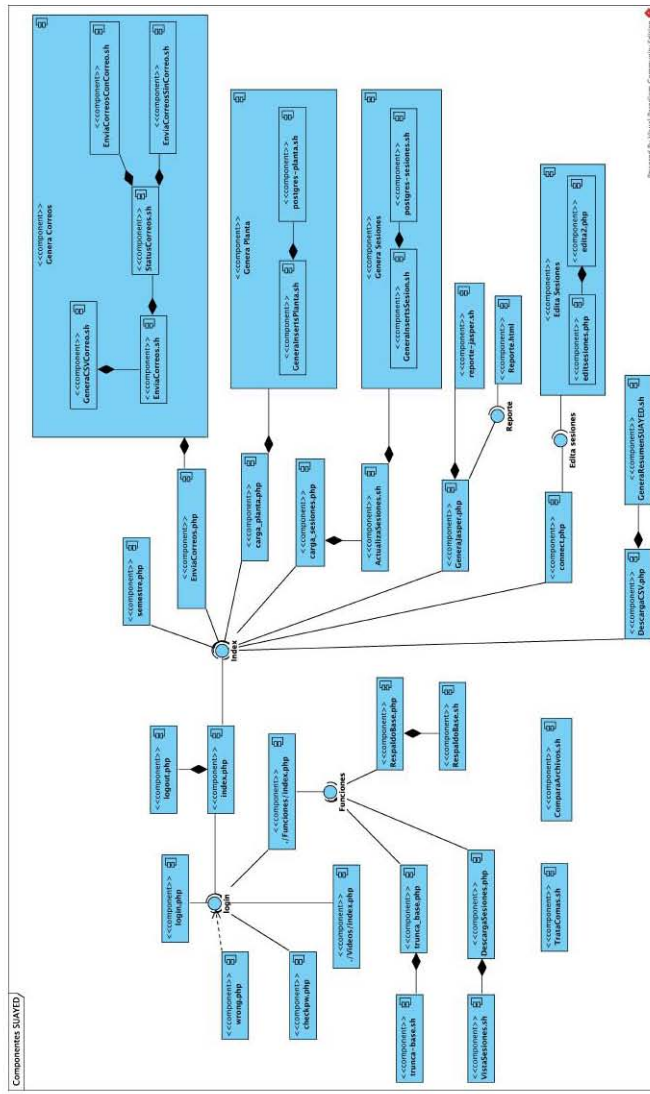


Figura 4-6: Diagrama de componentes del sistema.

4.11.7. Diagramas de secuencias

Diagrama de secuencia para obtener el reporte de asistencias de los asesores para el semestre en curso en formato PDF:

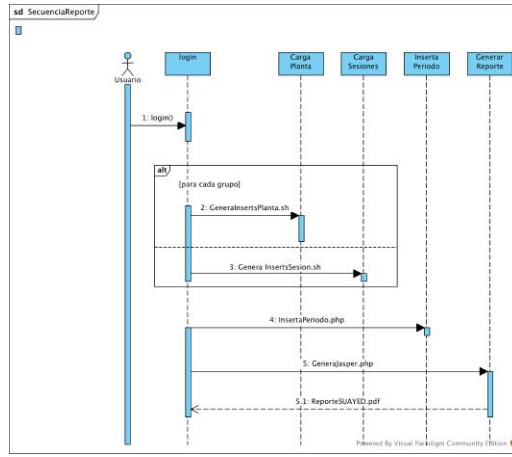


Figura 4-7: Diagrama de secuencia para el generar el reporte PDF.

Diagrama de secuencia para obtener el resumen de asistencias en formato CSV que contiene estadísticas generales de asistencias de los asesores:

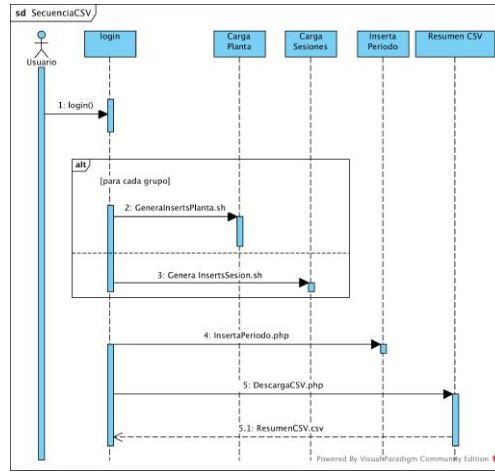


Figura 4-8: Diagrama de secuencia para el resumen CSV.

Diagrama de secuencia para el envío de correos a los asesores y la obtención de los correos en formato de texto en un archivo .zip:

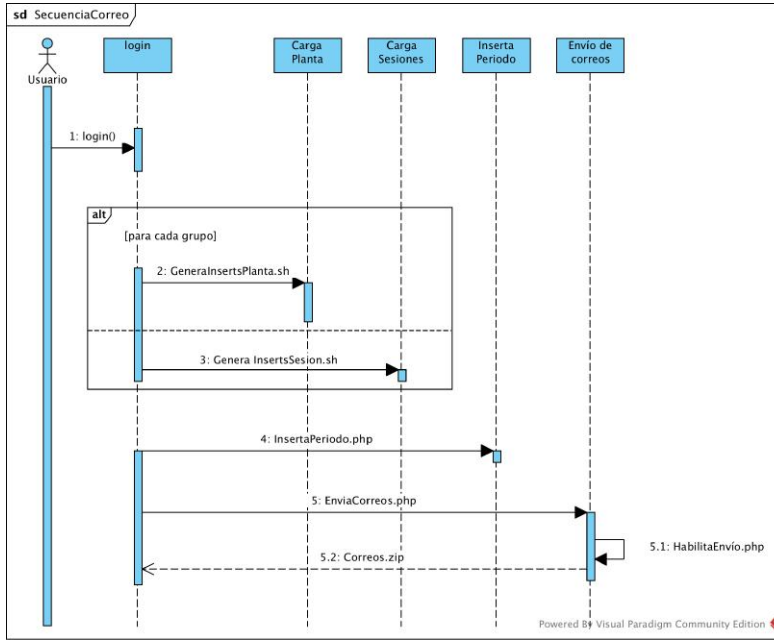


Figura 4-9: Diagrama de secuencia para el envío de correos electrónicos.

4.11.8. Diagramas de casos de uso

Caso de uso inicial

Basando el sistema en los requerimientos de los usuarios, se puede observar el caso de uso inicial de todo el sistema, cargar los dos archivos de Excel que se utilizan en el proceso actual y generar el reporte de los asesores en formato PDF.

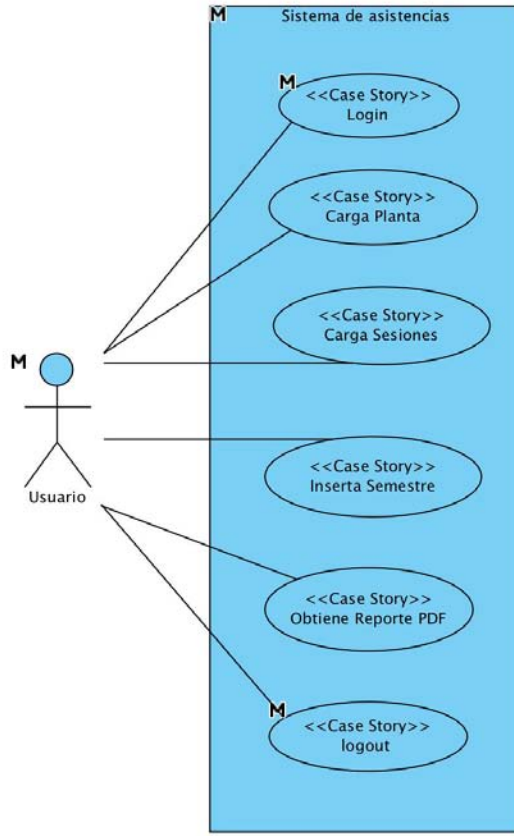


Figura 4-10: Caso de uso inicial del sistema.

Caso de uso, actualizar sesiones

Un caso de uso derivado de la necesidad de alimentar con nuevos datos el sistema, es cargar el archivo de Sesiones de asesores de manera consecutiva al avance del semestre, el diagrama de este caso es el siguiente:

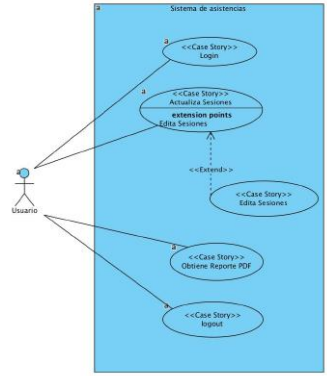


Figura 4-11: Caso de uso de actualización de sesiones.

Caso de uso, envío de correos electrónicos

Para el envío de los correos electrónicos informando de las asistencias, faltas, reposiciones y porcentajes a los asesores de cada grupo de la modalidad a distancia de la Facultad, se genera el siguiente caso de uso:

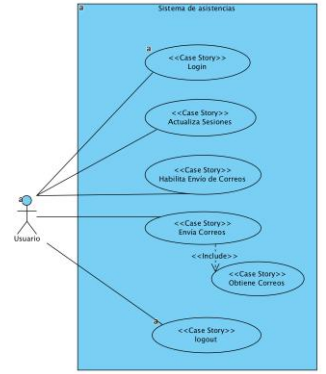


Figura 4-12: Caso de uso, envío de correos electrónicos a los asesores.

4.11.9. Modelo de Entidad-Relación del sistema

El modelo de Entidad-Relación de la base de datos del sistema quedó definido de la siguiente manera:

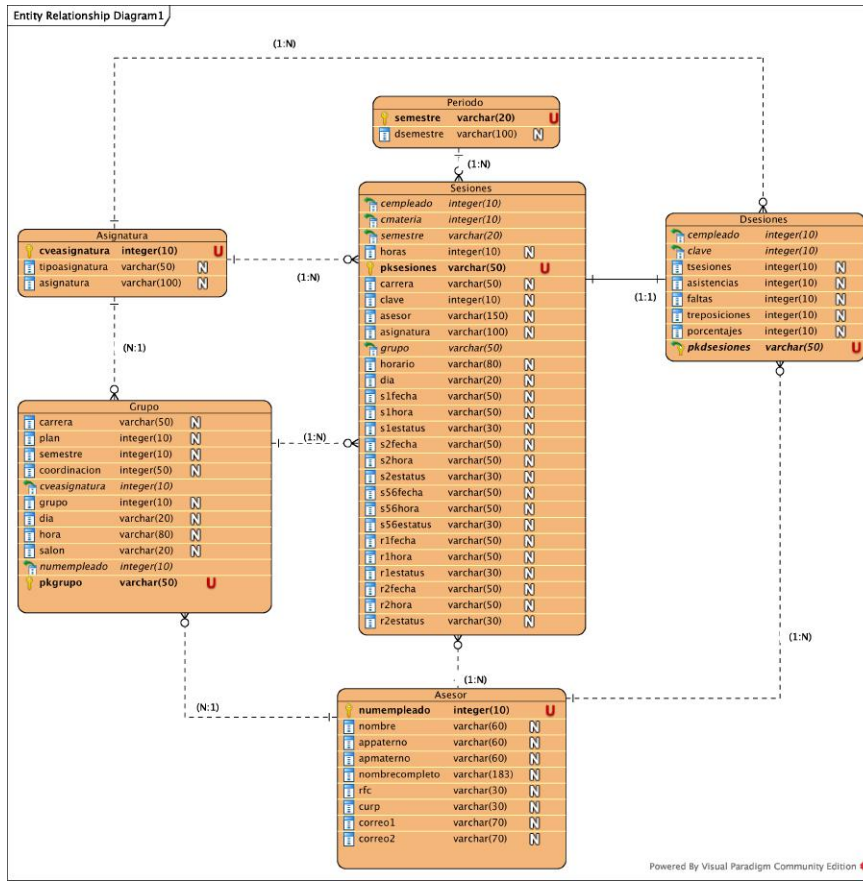


Figura 4-13: Modelo de Entidad-Relación de la base de datos del sistema.

4.11.10. Diagrama de la base de datos del sistema

La base de datos del sistema quedó definida con seis tablas y tres vistas que son de utilidad para los *stored procedures* necesarios para la operación del sistema.

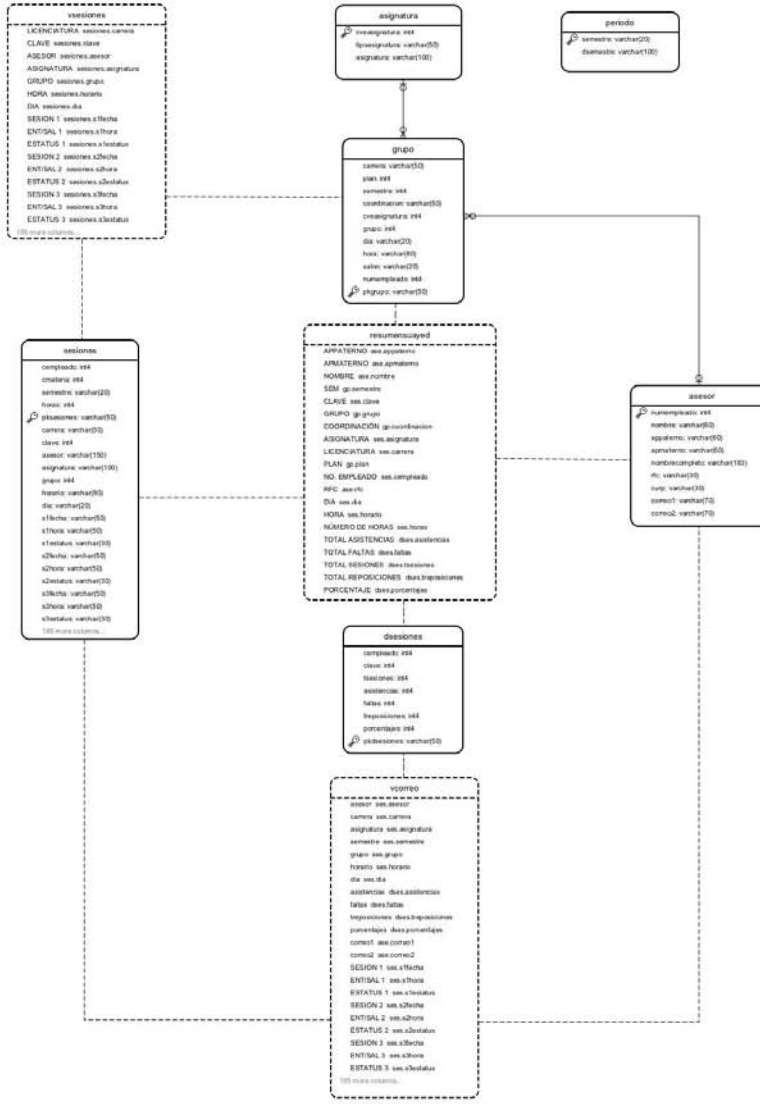


Figura 4-14: Diagrama de la base de datos del sistema (incluye tablas y vistas).

4.12. Interfaz del sistema

El sistema de gestión de asistencias se construyó utilizando una interfaz web de uso intuitivo, pero que contiene las instrucciones mínimas necesarias para poder operar el sistema, adicionalmente se incluyó un apartado con enlaces a los vídeos de uso del sistema.

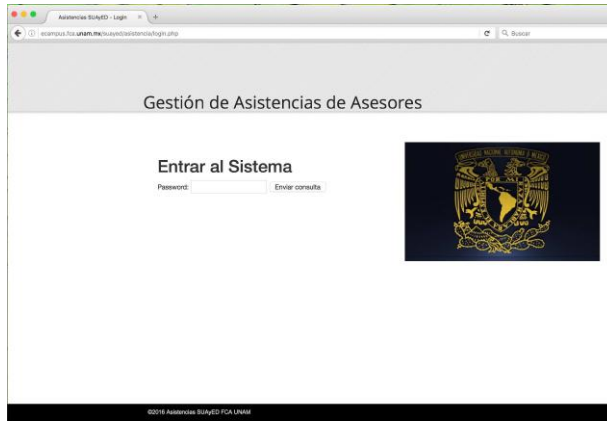


Figura 4-15: Pantalla de inicio del sistema, solicita el password para entrar.

Después de entrar al sistema al proporcionar el password solicitado, la pantalla principal luce así:

Figura 4-16: Pantalla de inicio del sistema

Este apartado muestra las funciones necesarias para operar el sistema, los otros apartados son opcionales, tiene el apartado de edición de las tablas de la base de datos y el enlace al editor externo de la base de datos.

Figura 4-17: Pantalla de inicio del sistema, vista detallada.

El apartado de funciones contiene los apartados de truncar la base de datos, generar un respaldo de la base de datos y generar el archivo Sesiones.csv que es la versión que se encuentra en la base de datos:

Figura 4-18: Pantalla con las funciones del sistema.

La última pantalla del sistema muestra los vídeos de uso del mismo:

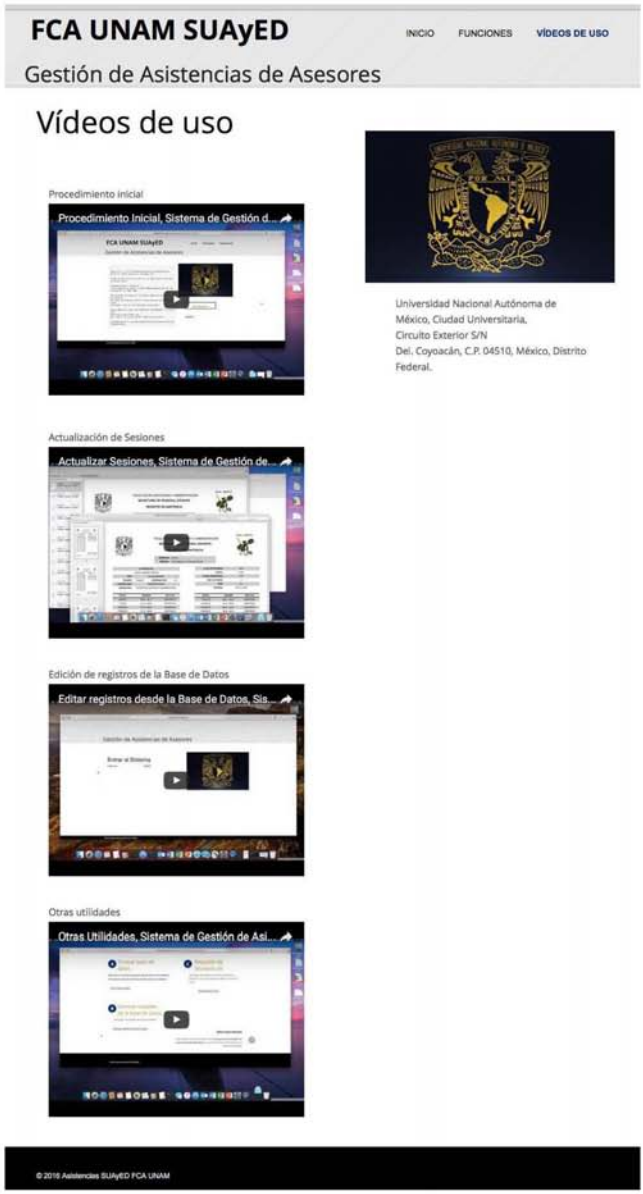


Figura 4-19: Pantalla que enlaza a los vídeos de uso del sistema.

En el apartado 7, se incluyó la posibilidad de editar manualmente las sesiones de los asesores, lo cual se puede hacer directamente en la interfaz (lo cual dispara el trigger asistencias_tri en la base de datos, pero esto es completamente transparente para el usuario del sistema), se hace uso de JQuery y Ajax para el filtro en la tabla que se muestra.

Figura 4-20: Edición manual de sesiones, primer pantalla.

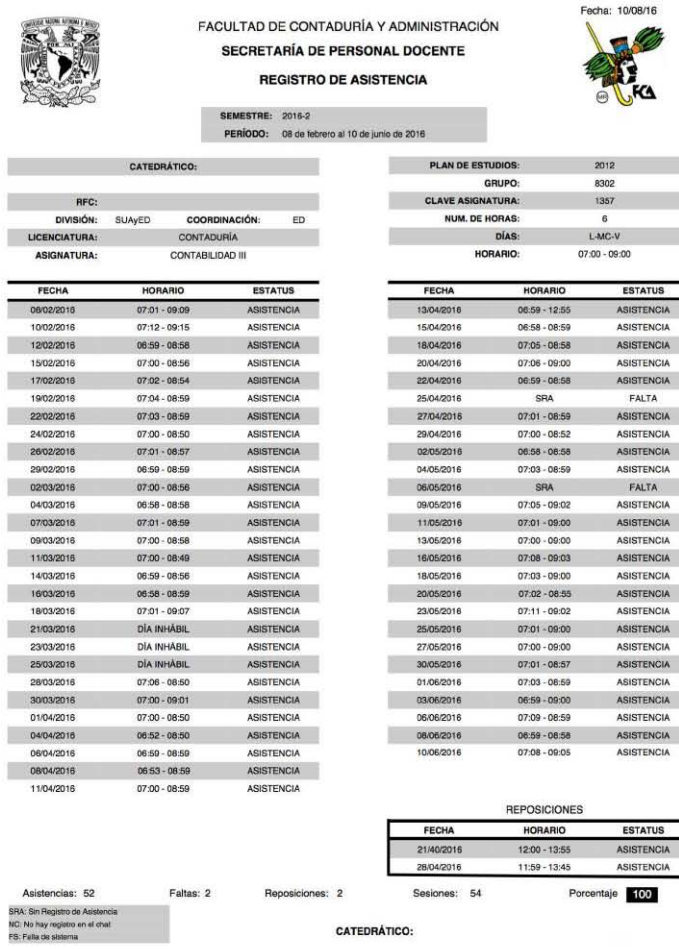
La segunda página recibe la variable del primary key del registro y permite editar directamente las sesiones.

Figura 4-21: Edición manual de sesiones, segunda pantalla.

4.13. Salidas generadas por el sistema

4.13.1. Reporte PDF

Ejemplo de los reportes generados por el sistema en formato PDF, el reporte se desarrolló con Jasper Reports y se conecta a la base de datos PostgreSQL para obtener los datos de cada asignatura impartida en el SUAyED durante el semestre.



FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
SECRETARÍA DE PERSONAL DOCENTE
REGISTRO DE ASISTENCIA

Fecha: 10/08/16

SEMESTRE: 2016-2
PERÍODO: 08 de febrero al 10 de junio de 2016

CATEDRÁTICO:
RFC:
DIVISIÓN: SUAyED COORDINACIÓN: ED
LICENCIATURA: CONTADURÍA
ASIGNATURA: CONTABILIDAD III

PLAN DE ESTUDIOS: 2012
GRUPO: 8302
CLAVE ASIGNATURA: 1357
NUM. DE HORAS: 6
DÍAS: L-MC-V
HORARIO: 07:00 - 09:00

FECHA	HORARIO	ESTATUS
08/02/2016	07:01 - 09:09	ASISTENCIA
10/02/2016	07:12 - 09:15	ASISTENCIA
12/02/2016	06:59 - 08:58	ASISTENCIA
15/02/2016	07:00 - 08:56	ASISTENCIA
17/02/2016	07:02 - 08:54	ASISTENCIA
19/02/2016	07:04 - 08:59	ASISTENCIA
22/02/2016	07:03 - 08:59	ASISTENCIA
24/02/2016	07:00 - 08:50	ASISTENCIA
26/02/2016	07:01 - 08:57	ASISTENCIA
29/02/2016	06:59 - 08:59	ASISTENCIA
02/03/2016	07:00 - 08:56	ASISTENCIA
04/03/2016	06:58 - 08:58	ASISTENCIA
07/03/2016	07:01 - 08:59	ASISTENCIA
09/03/2016	07:00 - 08:58	ASISTENCIA
11/03/2016	07:00 - 08:49	ASISTENCIA
14/03/2016	06:59 - 08:56	ASISTENCIA
16/03/2016	06:58 - 08:59	ASISTENCIA
18/03/2016	07:01 - 09:07	ASISTENCIA
21/03/2016	DÍA INHÁBIL	ASISTENCIA
23/03/2016	DÍA INHÁBIL	ASISTENCIA
25/03/2016	DÍA INHÁBIL	ASISTENCIA
28/03/2016	07:06 - 08:50	ASISTENCIA
30/03/2016	07:00 - 09:01	ASISTENCIA
01/04/2016	07:00 - 08:50	ASISTENCIA
04/04/2016	06:52 - 08:50	ASISTENCIA
06/04/2016	06:59 - 08:59	ASISTENCIA
08/04/2016	06:53 - 08:59	ASISTENCIA
11/04/2016	07:00 - 08:59	ASISTENCIA

FECHA	HORARIO	ESTATUS
13/04/2016	06:59 - 12:55	ASISTENCIA
15/04/2016	06:58 - 08:59	ASISTENCIA
18/04/2016	07:05 - 08:58	ASISTENCIA
20/04/2016	07:06 - 09:00	ASISTENCIA
22/04/2016	06:59 - 08:58	ASISTENCIA
25/04/2016	SRA	FALTA
27/04/2016	07:01 - 08:59	ASISTENCIA
29/04/2016	07:00 - 08:52	ASISTENCIA
02/05/2016	06:58 - 08:58	ASISTENCIA
04/05/2016	07:03 - 08:59	ASISTENCIA
06/05/2016	SRA	FALTA
09/05/2016	07:05 - 09:02	ASISTENCIA
11/05/2016	07:01 - 09:00	ASISTENCIA
13/05/2016	07:00 - 09:00	ASISTENCIA
16/05/2016	07:08 - 09:03	ASISTENCIA
18/05/2016	07:03 - 09:00	ASISTENCIA
20/05/2016	07:02 - 08:55	ASISTENCIA
23/05/2016	07:11 - 09:02	ASISTENCIA
25/05/2016	07:01 - 09:00	ASISTENCIA
27/05/2016	07:00 - 09:00	ASISTENCIA
30/05/2016	07:01 - 08:57	ASISTENCIA
01/06/2016	07:03 - 08:59	ASISTENCIA
03/06/2016	06:59 - 09:00	ASISTENCIA
06/06/2016	07:09 - 08:59	ASISTENCIA
08/06/2016	06:59 - 08:58	ASISTENCIA
10/06/2016	07:08 - 09:05	ASISTENCIA

REPOSICIONES

FECHA	HORARIO	ESTATUS
21/40/2016	12:00 - 13:55	ASISTENCIA
28/04/2016	11:59 - 13:45	ASISTENCIA

Asistencias: 52 Faltas: 2 Reposiciones: 2 Sesiones: 54 Porcentaje 100

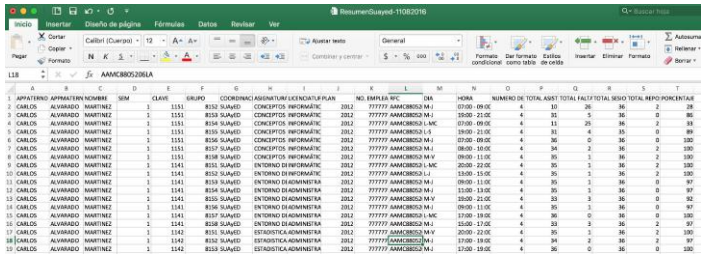
SRA: Sin Registro de Asistencia
NC: No hay registro en el chat
FS: Falla de sistema

CATEDRÁTICO:

Figura 4-22: Ejemplo del reporte de asistencias generado por el sistema, se omiten datos personales.

4.13.2. Resumen CSV

Reporte de resumen en formato CSV generado por el sistema, contiene estadísticas de las asistencias de los asesores durante el periodo.



APPATERNO	APMATERN	NOMBRE	SEM	CLAVE	GRUPO	COORDINACI	ASIGNATURA	LICENCIATU	PLAN	NO. EMPLEA	RFC	DIA	HORA	NUMERO DE	TOTAL ASIST	TOTAL FALT	TOTAL SESIO	TOTAL REPO	PORCENTAJE
CARLOS	ALVARADO	MARTINEZ	1	1151	8152	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	M-J	07:00 - 09:0(	4	10	26	36	2	28
CARLOS	ALVARADO	MARTINEZ	1	1151	8153	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	M-J	19:00 - 21:0(	4	31	5	36	0	86
CARLOS	ALVARADO	MARTINEZ	1	1151	8154	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	L-MC	07:00 - 09:0(	4	11	25	36	2	33
CARLOS	ALVARADO	MARTINEZ	1	1151	8155	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	L-S	19:00 - 21:0(	4	31	4	35	0	89
CARLOS	ALVARADO	MARTINEZ	1	1151	8156	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	M-J	07:00 - 09:0(	4	36	0	36	0	100
CARLOS	ALVARADO	MARTINEZ	1	1151	8157	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	M-J	08:00 - 10:0(	4	34	2	36	2	100
CARLOS	ALVARADO	MARTINEZ	1	1151	8158	SUAyED	CONCEPTOS INFORMÁTIC		2012	777777	AAMC880521	M-V	09:00 - 11:0(	4	35	1	36	2	100
CARLOS	ALVARADO	MARTINEZ	1	1141	8151	SUAyED	ENTORNO DI INFORMÁTIC		2012	777777	AAMC880521	L-MC	20:00 - 22:0(	4	35	1	36	2	100
CARLOS	ALVARADO	MARTINEZ	1	1141	8152	SUAyED	ENTORNO DI INFORMÁTIC		2012	777777	AAMC880521	L-J	13:00 - 15:0(	4	35	1	36	2	100
CARLOS	ALVARADO	MARTINEZ	1	1141	8153	SUAyED	ENTORNO DI ADMINISTRA		2012	777777	AAMC880521	M-J	09:00 - 11:0(	4	35	1	36	0	97
CARLOS	ALVARADO	MARTINEZ	1	1141	8154	SUAyED	ENTORNO DI ADMINISTRA		2012	777777	AAMC880521	M-J	11:00 - 13:0(	4	35	1	36	0	97
CARLOS	ALVARADO	MARTINEZ	1	1141	8155	SUAyED	ENTORNO DI ADMINISTRA		2012	777777	AAMC880521	M-V	19:00 - 21:0(	4	33	3	36	0	92
CARLOS	ALVARADO	MARTINEZ	1	1141	8156	SUAyED	ENTORNO DI ADMINISTRA		2012	777777	AAMC880521	M-J	09:00 - 11:0(	4	35	1	36	0	97
CARLOS	ALVARADO	MARTINEZ	1	1141	8157	SUAyED	ENTORNO DI ADMINISTRA		2012	777777	AAMC880521	L-MC	17:00 - 19:0(	4	36	0	36	0	100
CARLOS	ALVARADO	MARTINEZ	1	1141	8158	SUAyED	ENTORNO DI ADMINISTRA		2012	777777	AAMC880521	M-J	15:00 - 17:0(	4	33	3	36	2	97
CARLOS	ALVARADO	MARTINEZ	1	1142	8151	SUAyED	ESTADISTICA ADMINISTRA		2012	777777	AAMC880521	M-V	20:00 - 22:0(	4	35	1	36	2	100
CARLOS	ALVARADO	MARTINEZ	1	1142	8152	SUAyED	ESTADISTICA ADMINISTRA		2012	777777	AAMC880521	M-J	17:00 - 19:0(	4	34	2	36	2	97
CARLOS	ALVARADO	MARTINEZ	1	1142	8153	SUAyED	ESTADISTICA ADMINISTRA		2012	777777	AAMC880521	M-J	17:00 - 19:0(	4	36	0	36	0	100

Figura 4-23: Ejemplo del resumen CSV, se cambiaron datos personales.

4.13.3. Correos electrónicos generados

Se muestra uno de los correos generados automáticamente por el sistema, se envía cada uno a las dos direcciones de correo especificadas para cada asesor.

```
Apreciable: ALVARADO MARTÍNEZ CARLOS

            Docente de la modalidad a Distancia

            Sirva el presente correo para informarle sobre su estatus de asistencias correspondiente al semestre
2016-2 de la licenciatura en ADMINISTRACIÓN, asignatura: TEORIA DEL CONOCIMIENTO, del grupo 8055; que abarca de la
fecha: 08/02/2016 a la fecha: 10/06/2016.

            ASISTENCIAS: 34

            FALTAS: 2

            Fechas de las faltas: 21/04/2016, 02/06/2016

            REPOSICIONES:

            Reposición 1  29/04/2016  10:58 - 13:05  ASISTENCIA
            Reposición 2  06/05/2016  10:58 - 13:05  ASISTENCIA

            * PORCENTAJE: 100%

            A través de la siguiente dirección electrónica, usted podrá revisar los lineamientos considerados para
la toma de asistencias, así como los puntos necesarios a cubrir para solicitar sus reposiciones:

            http://fcasua.contad.unam.mx/docs/doc_academicos/registro-asistencia-distancia.pdf

            En caso de requerir alguna aclaración, estamos a sus órdenes en un horario de 9:00 a 19:00 horas, al
teléfono 56-22-82-80 al 82, ext. 113 y ext.115.

            Me despido enviando un cordial saludo y quedo a sus órdenes.

            Atentamente:

            Mtra. Marlene Olga Ramírez Chavero
            División SUAyED - FCA
            Coordinación de Evaluación y Estadística
```

Figura 4-24: Ejemplo de los correos generados por el sistema, se cambiaron datos personales.

4.14. Especificaciones técnicas del sistema

El sistema de Gestión de Asistencias se implementó en un servidor productivo Debian y se usó solamente software libre en su desarrollo, a continuación se muestran los detalles técnicos del mismo:

-URL asignada para el sistema:

`http://ecampus.fca.unam.mx/suayed/asistencia`

-Usuario del sistema operativo asignado: SUAyED

-El path asignado en el sistema operativo para la instalación del sistema es:

/var/www/suayed/asistencia/

4.14.1. Software Principal requerido

-Versión oficialmente soportada de Linux: **Debian GNU/Linux Squeeze de 32 bits (i686)**

-URL para descargar la distribución de Linux, solo el primer DVD es necesario, los demás contienen mucho software adicional:

`http://cdimage.debian.org/cdimage`
`/archive/6.0.10/i386/iso-dvd./`
`debian-6.0.10-i386-DVD-1.iso`

`http://cdimage.debian.org/cdimage`
`/archive/6.0.10/i386/iso-dvd/`
`debian-6.0.10-i386-DVD-2.iso`

`http://cdimage.debian.org/cdimage`
`/archive/6.0.10/i386/iso-dvd/`
`debian-6.0.10-i386-DVD-3.iso`

-JasperStarter

Programa para generar vía línea de comandos los reportes de Jaspersoft

`http://sourceforge.net/projects/jasperstarter/files/JasperStarter-3.0/jasperstarter-3.0.0-bin.zip`

-JDBC para JasperStarter

Conexión vía JDBC a la base de datos PostgreSQL de JasperStarter:

`https://jdbc.postgresql.org/download.html`

El JDBC necesario para el sistema es el de la versión 8.1 versiones 2 y 3, para que el JDBC permita generar el reporte PDF se tiene que copiar en la ruta:

/var/www/suayed/asistencia/jasperstarter/jdbc/

-Java Runtime Environment (JRE)

Se incluye el JRE directamente en los archivos del sistema en:

/var/www/suayed/asistencia/jre1.8.0_73

-phpPgAdmin

Se incluye este programa directamente en los archivos del sistema en:

/var/www/SUAyED/asistencia/phpPgAdmin-5.1

`http://ecampus.fca.unam.mx/suayed/asistencia/phpPgAdmin-5.1/`

Este editor no tiene liga de acceso directo porque es muy potente y es peligroso que el usuario final lo utilice sin el debido conocimiento sobre la base de datos.

-Jaspersoft Studio

Este programa no es necesario para que el sistema funcione, pero es el editor del reporte de asistencias, si se quiere modificar el mismo, se tendrá que utilizar este software, compilar el reporte y sustituir la versión que se entrega, la versión que se utilizó para desarrollar el reporte PDF fue: Jaspersoft Studio versión 6.2.1.

`http://community.jaspersoft.com/project/jaspersoft-studio/releases`

El reporte de Jaspersoft quedó localizado en el directorio:

/var/www/suayed/asistencia/Report/

El nombre del reporte generado es: suayed.jrxml, mientras que el archivo SUAyED.jasper es el binario que se obtiene al compilar el reporte con JasperStarter.

4.14.2. Requisitos técnicos adicionales

Se requiere un usuario de PostgreSQL con un puerto libre para la base de datos de AsistenciasSUAyED.

-Permisos:

Se recomienda que el owner de los archivos sea el usuario www-data con el grupo www-data y permisos 755 o 777, los archivo con owner SUAyED son archivos generados durante la ejecución del sistema.

Se tiene que añadir al .profile del usuario SUAyED el PATH de la versión soportada de Java por jasperstarter:

export JAVA_HOME=/var/www/suayed/asistencia/jre1.8.0_73/bin/java
export PATH=/var/www/suayed/asistencia/jre1.8.0_73/bin:$PATH

-Base de Datos

Como usuario postgres se añade el lenguaje de procedimientos almacenados de PostgreSQL a la base de datos:

createlang -U suayed plpgsql asistenciaed

Se tiene que crear el trigger llamado asistencias_tri que ejecuta el procedure

asistencias_tri que se requiere para actualizar los datos de los registros cuando se modifica la tabla sesiones:

CREATE TRIGGER asistencias_tri AFTER INSERT OR UPDATE ON sesiones FOR EACH ROW EXECUTE PROCEDURE asistencias_tri()

4.14.3. Listado de scripts y archivos php principales

-Carga del archivo de Planta.csv:

/var/www/suayed/asistencia/carga_planta.php
/var/www/suayed/asistencia/GeneraInsertsPlanta.sh
/var/www/suayed/asistencia/postgres-planta.sh

-Carga del archivo de Sesiones.csv:

/var/www/suayed/asistencia/carga_sesiones.php
/var/www/suayed/asistencia/GeneraInsertsSesion.sh
/var/www/suayed/asistencia/postgres-sesiones.sh

-Insertar el periodo del semestre manualmente:

/var/www/suayed/asistencia/semestre.php

-Descargar resumen de sesiones en formato CSV:

/var/www/suayed/asistencia/DescargaCSV.php

/var/www/suayed/asistencia/GeneraResumenSUAyED.sh
/var/www/suayed/asistencia/TrataComas.sh

-Generación y envío de correos electrónicos a los asesores:

/var/www/suayed/asistencia/EnviaCorreos.php
/var/www/suayed/asistencia/GeneraCSVCorreo.sh
/var/www/suayed/asistencia/TrataComas.sh
/var/www/suayed/asistencia/EnviaCorreos.sh

-Generar el reporte final del semestre en formato PDF:

/var/www/suayed/asistencia/GeneraJasper.php
/var/www/suayed/asistencia/reporte-jasper.sh

-Editar las sesiones de los asesores manualmente:

/var/www/suayed/asistencia/editsesiones/EditSesiones.php
/var/www/suayed/asistencia/editsesiones/Edita2.php
/var/www/suayed/asistencia/editsesiones/connect.php

-Truncar la base de datos del sistema:

/var/www/suayed/asistencia/trunca_base.php
/var/www/suayed/asistencia/trunca-base.sh

-Generar los respaldos en formato CSV y respaldos de la Base de Datos:

/var/www/suayed/asistencia/RespaldoBase.php

/var/www/suayed/asistencia/VistaSesiones.sh

/var/www/suayed/asistencia/TrataComas.sh

/var/www/suayed/asistencia/RespaldoBase.sh

-Comparar archivos (no disponible en la interfaz):

/var/www/suayed/asistencia/ComparaArchivos.sh

Capítulo 5

Resultados esperados

5.1. Resultados esperados

Los resultados esperados tras la implantación del Sistema de gestión de asistencias en la Coordinación de Evaluación y Estadística son:

Resultados esperados del Sistema de gestión de asistencias.
-Disminuir la carga de trabajo del área con respecto a la gestión de las asistencias de los asesores en un 50 % por lo menos.
-Garantizar la integridad de la información del sistema en todo momento.
-Permitir actualizar los datos del sistema directamente sin que esto represente un esfuerzo doble para el usuario.
-Proveer una interfaz sencilla, no sobrecargada, fácil de manejar y de preferencia mediante alguna interfaz web.

-Permitirle al usuario hacer modificaciones en las sesiones de los asesores, pues esta es una de las tareas principales del área, a saber, la administración de las asistencias, faltas, justificaciones y reposiciones de los asesores del SUAyED de la FCA.
-Tener operativa la solución en un semestre (2016-2).
-La solución no debe hacer uso de software privativo y debe estar bien integrada en la interfaz del usuario.
-El hardware en el que el proyecto tiene que operar no debe requerir velocidad de procesador, cantidad de memoria o espacio de almacenamiento excesivos.
-El reporte PDF generado debe cumplir al 100 % con las especificaciones requeridas y comportarse de acuerdo a los requerimientos del usuario.
-El resumen CSV debe generarse adecuadamente sin inconsistencias en los datos que presente.
-Los correos electrónicos del sistema deben poderse enviar a cada asesor, tanto a su dirección electrónica personal como a su dirección académica de la UNAM.
-La lista de correos generados debe quedar disponible para que el usuario tenga seguridad de lo que se envió a cada asesor por correo electrónico.
-La seguridad del sistema debe ser la adecuada para la infraestructura interna de la UNAM.
-El sistema debe poder migrarse fácilmente de servidor y debe estar documentado el proceso para hacerlo.

-La documentación y los vídeos de uso deben enseñar bien a usar el sistema en varios escenarios.

Tabla 5-1: Resultados esperados

Capítulo 6

Conclusiones

6.1. Objetivos logrados

-El sistema se implementó de manera exitosa en el servidor de la Facultad de Contaduría y Administración que para tal efecto se proporcionó por parte del Centro de Informática de la Facultad (CIFCA), se logró reducir sustancialmente la carga de trabajo derivada de las actividades que dieron pie a la implementación del proyecto, se mejoró la certidumbre respecto a la integridad de la información que se entrega a las distintas instancias de la Facultad y se dotó de una mejor infraestructura que estará preparada para una mayor carga de trabajo sin reducir su efectividad.

-Durante el desarrollo del sistema se fueron descubriendo requerimientos que el usuario no tenía claros o que daban por sentado ya estaban contemplados, esto ayudó a mejorar los propios procesos del área haciéndolos más formales, estandarizados y mejor entendidos por todas las personas que la conforman.

-Se pudo implementar en tiempo y forma el presente sistema, dentro de las 480 horas presupuestadas durante 18 semanas, hoy en día el sistema está en producción y está preparado para varios casos de uso, escenarios no especificados por el usuario pero necesarios de cara a una administración más seria de la base de datos

del sistema (como el módulo de respaldo de la base de datos y la incorporación de gestores web de PostgreSQL).

-Uno de los objetivos más importantes que fueron alcanzados en este proyecto, fue el requerimiento de generar los reportes del semestre 2016-2 con el nuevo sistema en lugar del viejo, lo cual se consiguió al 100 %, con lo que se puede afirmar que el sistema ya está en producción.

-El costo final estimado del sistema desarrollado, fue de $73,526.40 M.N. costo que no erogó ni la Facultad, ni la Universidad, además se usó solamente Software Libre, reduciendo los costos a la infraestructura operativa necesaria en el servidor de la Facultad, que además ya se usaba para otros proyectos.

-El sistema no hace uso intensivo de los recursos del servidor, solamente algunos minutos cuando se cargan los datos al sistema, lo cual permite tenerlo instalado en un servidor que no se dedica en exclusiva al presente proyecto.

-Un objetivo importante cumplido fue el módulo de edición de sesiones mediante la interfaz web, esta parte requirió utilizar JQuery junto con PHP para poderse implementar pues se trata de edición de los registros de la base de datos, que aunque es algo normalmente no recomendado, se implementó de modo que solamente los datos necesarios de modificación para el usuario son editables (las sesiones de los asesores), dejando los demás sin posibilidad de edición, para evitar problemas con los registros en caso de errores al editar campos sensibles.

-El sistema de envío de correos se simplificó bastante con relación al sistema anterior, permitiendo mandar los correos de todos los grupos de una sola vez, generando además el contenido de cada correo para una correcta gestión por parte del usuario, además se puede desactivar el envío de correos pero generar los datos que se enviarían, para analizarlos y enviarlos cuando se tiene certeza de que el contenido es el adecuado.

-Poniéndolo en porcentajes, el sistema logró cumplir a un 95 % con la idea

propuesta, esto debido a que el servidor del CIFCA no tiene el ancho de banda suficiente para garantizar que todos los correos de los asesores se envíen de una sola vez, pues algunos correos que no se alcanzan a enviar se quedan intentando hasta que ocurre el timeout, hay que considerar que se envían de un solo golpe aproximadamente 860 correos electrónicos, los cuales son generados correctamente por el sistema, se trata de una limitación de infraestructura.

6.2. Cuestiones pendientes

El proyecto de software del Sistema de gestión de asistencias tuvo como propósitos principales facilitar la generación del reporte semestral en PDF y el envío mensual de correos a los asesores con su estatus provisionales, objetivos que se lograron casi completamente (salvo la cuestión de los correos comentada anteriormente), simplificando la operación y mejorando la eficiencia de la Coordinación de Estadística y Evaluación; sin embargo, quedaron ventanas de oportunidad que se pueden aprovechar para simplificar aún más el control de asistencias del SUAyED en la Facultad:

-El origen de los datos proviene de dos plataformas, tanto de la plataforma que dejará de funcionar en 2018 y la plataforma nueva proporcionada por el CUAED, cuya versión de la plataforma Moodle es más actual y tiene procedimientos distintos para obtener los datos de las asistencias de cada asignatura impartida.

-El proceso de obtención de los datos de asistencias en ambas plataformas es manual, no fue parte del alcance de este proyecto automatizar la recogida de estos datos, porque de entrada los sistemas de la plataforma Moodle no están bajo resguardo de la Coordinación de Estadística y Evaluación, es más una tarea por hacer por parte de los administradores de la plataforma para generar los registros en formato CSV, que podrían ser exactamente en el orden y con los datos necesarios para

alimentar el presente sistema, inclusive se podrían generar automáticamente y enviarse al servidor del presente sistema, donde se podrían ejecutar automáticamente sus scripts y simplificar aún más la operación del mismo.

-Como prueba del punto anterior, la Coordinación de Estadística y Evaluación tuvo un sistema previo que obtenía los datos de las sesiones automáticamente, pero en una actualización posterior de la plataforma Moodle por parte de los administradores, la solución creada para la Coordinación dejó de funcionar, de modo que, hubiera sido un esfuerzo solamente provisional intentar obtener los datos de parte de la Coordinación, pues forzosamente se debe acordar con los administradores de la plataforma un método eficaz de recogida de la información.

-Los administradores de la plataforma Moodle usan la instancia productiva también para hacer pruebas, lo que genera mucha información basura, otro punto pendiente es acordar con ellos que se debe tener una plataforma productiva que siempre debe estar libre de basura, incluso se podrían crear instancias de la plataforma cada semestre para garantizar la sanidad de la misma.

-Hacer el sistema de recogida de la información necesaria para esta coordinación no sería difícil usando scripts y programación con PHP y Ajax, se puede hacer directamente desde la base de datos o bien, si el sistema envía una cadena JSON como respuesta a la petición, se puede desarrollar una solución basada en algún lenguaje orientado a objetos para rellenar la información y que a su vez genere el archivo de entrada necesario para el sistema desarrollado en este proyecto.

Capítulo 7

Anexos

7.1. Dependencias de Software

Dependencias de Debian GNU/Linux Squeeze para el sistema de Gestión de Asistencias, se instalan desde el medio oficial de la distribución.

Paquete de Software	**Versión instalada**	**Comentario**
postgresql-8.1	8.1.10-1	object-relational SQL database, version 8.1 server
postgresql-client-8.1	8.1.10-1	front-end programs for PostgreSQL 8.1
postgresql-client-common	98	manager for multiple PostgreSQL client versions
postgresql-common	94lenny1	PostgreSQL database-cluster manager
sed	4.2-1	The GNU sed stream editor

sudo	1.7.4p4-2.squeeze.4	Provide limited super user privileges to specific users
apache2	2.2.14-5	Apache HTTP Server meta-package
apache2-mpm-prefork	2.2.14-5	Apache HTTP Server - traditional non-threaded model
apache2-utils	2.2.11-3	utility programs for webservers
apache2.2-bin	2.2.14-5	Apache HTTP Server common binary files
apache2.2-common	2.2.14-5	Apache HTTP Server common files
libapache2-mod-perl2	2.0.4-5	Integration of perl with the Apache2 web server
libapache2-mod-php5	5.2.12.dfsg.1-2	server-side, HTML-embedded scripting language (Apache 2 module)
libapache2-reload-perl	0.10-2	Reload Perl modules when changed on disk
bash	3.2-5	The GNU Bourne Again SHell
bsd-mailx	8.1.2-0.20081101cvs-2	A simple mail user agent
exim4	4.72-2	metapackage to ease Exim MTA (v4) installation

exim4-base	4.72-2	support files for all Exim MTA (v4) packages
exim4-config	4.69-11	configuration for the Exim MTA (v4)
exim4-daemon-heavy	4.69-9	Exim MTA (v4) daemon with extended features, including exiscan-acl
exim4-daemon-light	4.72-2	lightweight Exim MTA (v4) daemon
expect	5.43.0-17	A program that can automate interactive applications
diff	2.8.1-12	File comparison utilities
gawk	1:3.1.6.dfsg-2	GNU awk, a pattern scanning and processing language
gzip	1.3.12-8	GNU compression utilities
openssh-blacklist	0.4.1	list of default blacklisted OpenSSH RSA and DSA keys
openssh-blacklist-extra	0.4.1	list of non-default blacklisted OpenSSH RSA and DSA keys
openssh-client	1:5.1p1-5+b1	secure shell client, an rlogin/rsh/rcp replacement

openssh-server	1:5.1p1-5+b1	secure shell server, an rshd replacement
openssl	0.9.8g-16	Secure Socket Layer (SSL) binary and related crypto-graphic tools
openssl-blacklist	0.5-2	list of blacklisted OpenSSL RSA keys
php-db	1.7.13-2	PHP PEAR Database Abs-traction Layer
php-log	1.10.0-1	Log module for PEAR
php-mail	1.1.14-1	PHP PEAR module for sen-ding email
php-mail-mime	1.5.2-0.1	PHP PEAR module for crea-ting MIME messages
php-mail-mimedecode	1.5.0-3	PHP PEAR module to deco-de MIME messages
php-net-smtp	1.3.1-1	PHP PEAR module imple-menting SMTP protocol
php-net-socket	1.0.9-2	PHP PEAR Network Socket Interface module
php-pear	5.2.9.dfsg.1-4	PEAR - PHP Extension and Application Repository
php5	5.2.9.dfsg.1-4	server-side, HTML-embedded scripting lan-guage (metapackage)

php5-cli	5.2.12.dfsg.1-2	command-line interpreter for the php5 scripting language
php5-common	5.2.12.dfsg.1-2	Common files for packages built from the php5 source
php5-curl	5.2.12.dfsg.1-2	CURL module for php5
php5-gd	5.2.12.dfsg.1-2	GD module for php5
php5-imap	5.2.12.dfsg.1-2	IMAP module for php5
php5-mcrypt	5.2.12.dfsg.1-2	MCrypt module for php5
php5-mysql	5.2.12.dfsg.1-2	MySQL module for php5
php5-pgsql	5.2.12.dfsg.1-2	PostgreSQL module for php5
php5-recode	5.2.12.dfsg.1-2	recode module for php5
php5-suhosin	0.9.29-1	advanced protection module for php5
php5-xmlrpc	5.2.12.dfsg.1-2	XML-RPC module for php5
php5-xsl	5.2.12.dfsg.1-2	XSL module for php5
console-common	0.7.81	basic infrastructure for text console configuration
console-data	2:1.07-11	keymaps, fonts, charset maps, fallback tables for console-tools
console-tools	1:0.2.3dbs-65.1	Linux console and font utilities
coreutils	7.3-1	The GNU core utilities

libc-bin	2.11.2-7	Embedded GNU C Library: Binaries

Tabla 7-1: Dependencias del sistema en Debian GNU/Linux.

7.2. Script en Bash ActualizaSesiones.sh

Este script es el primero que se ejecuta al cargar el archivo de Sesiones.csv, se determina si la base de datos ya tiene registros, de modo que si no tiene registros, se ejecuta el script GeneraInsertsSesion.sh o bien, de encontrarse registros en la tabla de sesiones, se ejecuta un Stored Procedure que se genera desde el mismo script que actualiza la información existente en la base de datos.

```
#!/bin/bash

#Author: Carlos Alvarado

#Script para actualizar las sesiones
#Si no encuentra registros en sesiones, ejecuta el
    script GeneraInsertsSesion.sh

dirsesiones=$(echo "/var/www/suayed/asistencia/
    sesiones")

reporte=$(echo "${dirsesiones}/$(ls -ltr ${dirsesiones
    } | grep .csv | tail -1 | awk '{print $9}')")
```

```
chmod 777 ${reporte}

#Elimina espacios al inicio de cada campo.
cat ${reporte} | sed -i 's/, /,/g' ${reporte}

#Cambiando la codificación de Windows a UTF8.

if [[ $(file -i ${reporte} | awk -F'=' '{print $2}')
    == ''iso-8859-1'' ]]; then

iconv -f ISO-8859-1 -t UTF-8 < ${reporte} > ${reporte
    }.utf8

mv -f ${reporte}.utf8 ${reporte}

fi

#Cambiando los saltos de línea de Windows a UNIX.

dos2unix ${reporte}

#Obtiene el número de registros en la tabla sesiones

expect << EOD

set prompt {[$]}
```

```
spawn ssh -t -o StrictHostKeyChecking=no
    SUAyED@localhost
expect -re "$prompt";
send "psql -d asistenciaed -c \'' SELECT COUNT(*) FROM
    SESIONES \'' > /var/www/SUAyED/asistencia/sexpect.
    txt \r";
expect -re "$prompt";
send "exit \r";
expect eof
EOD

conteo=$(cat /var/www/SUAyED/asistencia/sexpect.txt |
    head -3 | tail -1)

#Si hay registros entonces actualiza

if [ ${conteo} -gt 0 ]; then

echo "ACTUALIZANDO TABLA SESIONES..."
#Generando el UPDATE para la tabla sesiones

echo "CREATE TABLE tmp_sesiones (
carrera character varying(50),
cmateria character varying(50),
asesor character varying(150),
asignatura character varying(100),
grupo character varying(50),
```

```
horario character varying(80),
dia character varying(20),
s1fecha character varying(50),
s1hora character varying(50),
s1estatus character varying(30),
s2fecha character varying(50),
s2hora character varying(50),
s2estatus character varying(30),
s3fecha character varying(50),
s3hora character varying(50),
s3estatus character varying(30),
[...]
s56fecha character varying(50),
s56hora character varying(50),
s56estatus character varying(30),
r1fecha character varying(50),
r1hora character varying(50),
r1estatus character varying(30),
r2fecha character varying(50),
r2hora character varying(50),
r2estatus character varying(30)
);

\\copy tmp_sesiones FROM '${reporte}' WITH CSV HEADER
    NULL AS ' '

UPDATE sesiones
```

```
SET
s1fecha = tmp_sesiones.s1fecha,
s1hora = tmp_sesiones.s1hora,
s1estatus = tmp_sesiones.s1estatus,
s2fecha = tmp_sesiones.s2fecha,
s2hora = tmp_sesiones.s2hora,
s2estatus = tmp_sesiones.s2estatus,
s3fecha = tmp_sesiones.s3fecha,
s3hora = tmp_sesiones.s3hora,
s3estatus = tmp_sesiones.s3estatus,
[...]
s56fecha = tmp_sesiones.s56fecha,
s56hora = tmp_sesiones.s56hora,
s56estatus = tmp_sesiones.s56estatus,
r1fecha = tmp_sesiones.r1fecha,
r1hora = tmp_sesiones.r1hora,
r1estatus = tmp_sesiones.r1estatus,
r2fecha = tmp_sesiones.r2fecha,
r2hora = tmp_sesiones.r2hora,
r2estatus = tmp_sesiones.r2estatus
FROM    tmp_sesiones
WHERE sesiones.cmateria = tmp_sesiones.cmateria
AND sesiones.carrera = tmp_sesiones.carrera
AND sesiones.asesor = tmp_sesiones.asesor
AND sesiones.grupo = tmp_sesiones.grupo;

DROP TABLE tmp_sesiones;"
```

```
> /var/www/suayed/asistencia/UpdateSesiones.sql


# Ejecutando el UPDATE generado en la base de datos

expect << EOD

set prompt {[$]}
spawn ssh -t -o StrictHostKeyChecking=no
    SUAyED@localhost
expect -re "$prompt";
send "psql -d asistenciaed < /var/www/SUAyED/
    asistencia/UpdateSesiones.sql\r";
expect -re "$prompt";
send "exit \r";
expect eof

EOD

#Si no hay registros en sesiones, ejecuta el script
    GeneraInsertsSesion.sh

else

echo "GENERANDO INSERTS DE SESIONES"
```

```
/bin/bash /var/www/suayed/asistencia/
    GeneraInsertsSesion.sh


fi
```

7.3. Stored Procedure PL/pgSQL Asistencias

Stored Procedure *asistencias* que actualiza la tabla dsesiones cuando hay cambios en la tabla sesiones

```
DECLARE

sumasistencias REAL;
sumfaltas REAL;
sumporcentajes REAL;
realporcentaje REAL;
difference REAL;
stsesiones REAL;
sumareposiciones REAL;
reposicionesinscritas REAL;

BEGIN
SELECT SUM(CASE WHEN s1estatus='asistencia' OR
    s1estatus='ASISTENCIA' THEN 1 ELSE 0 END)
+ SUM(CASE WHEN s2estatus='asistencia' OR s2estatus='
    ASISTENCIA' THEN 1 ELSE 0 END)
```

```
+ SUM(CASE WHEN s3estatus='asistencia' OR s3estatus='
    ASISTENCIA' THEN 1 ELSE 0 END)
[...]
+ SUM(CASE WHEN s56estatus='asistencia' OR s56estatus=
    'ASISTENCIA' THEN 1 ELSE 0 END)
INTO sumasistencias
FROM sesiones WHERE pksesiones=NEW.pksesiones;


SELECT SUM(CASE WHEN r1estatus='asistencia' OR
    r1estatus='ASISTENCIA' THEN 1 ELSE 0 END)
+ SUM(CASE WHEN r2estatus='asistencia' OR r2estatus='
    ASISTENCIA' THEN 1 ELSE 0 END)
INTO sumareposiciones
FROM sesiones WHERE pksesiones=NEW.pksesiones;


SELECT SUM(CASE WHEN r1estatus='asistencia' OR
    r1estatus='falta' THEN 1 ELSE 0 END)
+ SUM(CASE WHEN r2estatus='asistencia' OR r2estatus='
    falta' THEN 1 ELSE 0 END)
+ SUM(CASE WHEN r1estatus='ASISTENCIA' OR r1estatus='
    FALTA' THEN 1 ELSE 0 END)
+ SUM(CASE WHEN r2estatus='ASISTENCIA' OR r2estatus='
    FALTA' THEN 1 ELSE 0 END)
INTO reposicionesinscritas
FROM sesiones WHERE pksesiones=NEW.pksesiones;
```

```
SELECT SUM(CASE WHEN s1estatus='falta' OR s1estatus='
    FALTA' THEN 1 ELSE 0 END)
+ SUM(CASE WHEN s2estatus='falta' OR s2estatus='FALTA'
     THEN 1 ELSE 0 END)
+ SUM(CASE WHEN s3estatus='falta' OR s3estatus='FALTA'
     THEN 1 ELSE 0 END)
[...]
+ SUM(CASE WHEN s56estatus='falta' OR s56estatus='
    FALTA' THEN 1 ELSE 0 END)

INTO sumfaltas
FROM sesiones WHERE pksesiones=NEW.pksesiones;


stsesiones := sumasistencias+sumfaltas;


IF sumasistencias > 0 THEN

realporcentaje := ((sumasistencias+sumareposiciones)
    *100)/stsesiones;

sumporcentajes := round(realporcentaje);

difference := realporcentaje-sumporcentajes;
```

```
IF difference >= 0.5 THEN

sumporcentajes := sumporcentajes+1;

END IF;


ELSE

sumporcentajes := 0;

END IF;

IF sumporcentajes > 100 THEN

sumporcentajes := 100;

END IF;

UPDATE dsesiones SET tsesiones = stsesiones WHERE
    pkdsesiones=NEW.pksesiones;

UPDATE dsesiones SET asistencias = sumasistencias
   WHERE pkdsesiones=NEW.pksesiones;
```

```
UPDATE dsesiones SET treposiciones =
    reposicionesinscritas WHERE pkdsesiones=NEW.
    pksesiones;

UPDATE dsesiones SET faltas = sumfaltas WHERE
    pkdsesiones=NEW.pksesiones;

UPDATE dsesiones SET porcentajes = sumporcentajes
    WHERE pkdsesiones=NEW.pksesiones;

RETURN NEW;

END;
```

Bibliografía

Ambler, S. W. (2006). *The Agile Unified Process (AUP).* Consultado el 11 de agosto de 2016. Descargado de http://www.ambysoft.com/unifiedprocess/agileUP.html

Apache-Foundation. (2016). *Apache. The Number One HTTP Server On The Internet.* Consultado el 25 de octubre de 2016. Descargado de https://httpd.apache.org

Arana, B. O. (2015). *Curso Práctico avanzado de PostgreSQL* (1a ed.). México D.F.: Alfaomega.

Braude, E. J. (2003). *Ingeniería de Software, una perspectiva orientada a objetos* (1a ed.). México D.F.: Alfaomega.

Cuadra, D., Castro, E., Iglesias, A. M., Martínez, P., Calle, F. J., de Pablo, C., ... Segura, I. (2013). *Desarrollo de Bases de Datos. Casos prácticos desde el análisis a la implementación* (2a ed.). Madrid: RA-MA Editorial.

CUAED-UNAM. (2015). *Población Escolar SUAyED (2014-2015).* Consultado el 29 de agosto de 2016. Descargado de https://public.tableau.com/profile/coordinaci.n.de.universidad.abierta.y.educaci.n.a.distancia

Dijkstra, E. W. (1988). *Sobre la crueldad de verdaderamente enseñar ciencias de la computación.* Consultado el 26 de julio de 2016. Descargado de

http://www.smaldone.com.ar/documentos/ewd/sobre_la_crueldad.html

::everac99. (2015). *Sueldos IT en México 2015: de startups y desarrollo móvil.* Consultado el 17 de noviembre de 2016. Descargado de https://everac99.wordpress.com/2015/03/20/sueldos-it-en-mexico-2015-de-startups-y-desarrollo-movil/

FCA-UNAM. (2016a). *Facultad de Contaduría y Administración. Estructura.* Consultado el 26 de julio de 2016. Descargado de http://www.fca.unam.mx/estructura.php

FCA-UNAM. (2016b). *Facultad de Contaduría y Administración. Misión.* Consultado el 26 de julio de 2016. Descargado de http://www.fca.unam.mx/mision.php

FCA-UNAM. (2016c). *Facultad de Contaduría y Administración. Objetivos.* Consultado el 26 de julio de 2016. Descargado de http://www.fca.unam.mx/bienvenida.php

FCA-UNAM. (2016d). *Facultad de Contaduría y Administración. Visión.* Consultado el 26 de julio de 2016. Descargado de http://www.fca.unam.mx/vision.php

Fontela, C. (2011). *UML: modelado de software para profesionales* (1a ed.). Buenos Aires: Alfaomega.

García, F. M. M. (2006). *UNIX. Programación Avanzada* (3a ed.). México D.F.: Alfaomega.

IBM. (1998). *Rational Unified Process. Best Practices for Software Development Teams IBM Rational.* Consultado el 16 de agosto de 2016. Descargado de https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251

_bestpractices_TP026B.pdf

jQuery Foundation. (2016). *What's New in jQuery UI 1.12?* Consultado el 19 de octubre de 2016. Descargado de https://jqueryui.com

Martínez, F. R., y Chávez, G. H. (2010). *Administración de Proyectos. Guía para el aprendizaje* (1a ed.). Naucalpan, México: Pearson Educación.

Moodle.net. (2016). *Acerca de Moodle.* Consultado el 24 de octubre de 2016. Descargado de https://docs.moodle.org/all/es/Acerca_de_Moodle

NetBeans-Group. (2016). *Introduction to Ajax for Java Web Applications.* Consultado el 08 de octubre de 2016. Descargado de https://netbeans.org/kb/docs/web/ajax-quickstart.html

Network, M. D. (2016). *JavaScript.* Consultado el 04 de noviembre de 2016. Descargado de https://developer.mozilla.org/es/docs/Web/JavaScript

Pantaleo, G., y Rinaudo, L. (2015). *Ingeniería de Software* (1a ed.). Buenos Aires: Alfaomega.

Petersen, R. (2009). *Linux. Manual de referencia* (6a ed.). México D.F.: McGraw-Hill.

PHP-Group. (2016). *What is PHP?* Consultado el 20 de septiembre de 2016. Descargado de http://php.net/manual/en/intro-whatis.php

PostgreSQL-Group. (2016). *PL/pgSQL - SQL Procedural Language.* Consultado el 06 de septiembre de 2016. Descargado de https://www.postgresql.org/docs/9.2/static/plpgsql-overview.html

Pressman, R. S. (2010). *Ingeniería del Software, un enfoque práctico* (7a ed.). México D.F.: McGraw-Hill.

Reinosa, E. J., Maldonado, C. A., Muñoz, R., Damiano, L. E., y Abrutsky, M. A. (2012). *Bases de Datos* (1a ed.). Buenos Aires: Alfaomega.

Sobell, M. G. (2010). *Manual práctico de Linux* (1a ed.). Madrid: Anaya.

SUAYED-UNAM. (2016). *Acerca del SUAyED.* Consultado el 28 de agosto de 2016. Descargado de `http://suayed.unam.mx/img/Acerca_del_SUAyED_2016.pdf`

UAdM. (2016). *XML ¿Qué es?* Consultado el 14 de septiembre de 2016. Descargado de `http://www.mundolinux.info/que-es-xml.htm`

UNAM. (2015). *Cronología Histórica de la UNAM.* Consultado el 18 de noviembre de 2016. Descargado de `https://www.unam.mx/acerca-de-la-unam/unam-en-el-tiempo/`

UNAM. (2016). *Valores UNAM.* Consultado el 26 de julio de 2016. Descargado de `http://www.dgcs.unam.mx/valor/Valores.html`