



Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

**Desarrollo de un módulo de gestión y planificación  
para los cursos de Extensión de la Escuela de  
Idiomas Modernos de la Facultad de Humanidades  
y Educación de la UCV**

Trabajo Especial de Grado presentado ante la ilustre  
Universidad Central de Venezuela por:

Br. Alexandra Arreaza Zadovsky

Br. Wilson Johan Manyoma Adames

Para optar por el título de  
Licenciado en Computación

Tutores:

Profa. Jossie Zambrano

Prof. Sergio Rivas

Caracas, abril 2012

# Acta

---

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado con el título “Desarrollo de un módulo de gestión y planificación para los cursos de Extensión de la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la UCV”, el cual es presentado por la Br. Alexandra Arreaza, de Cédula de Identidad 19.395.140 y el Br. Wilson Manyoma, de Cédula de Identidad 19.220.025, a los fines de optar al título de Licenciado en Computación, dejamos en constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 17 de abril de 2012, a las 11:00 am en el aula PB-III, para que los autores lo defendieran en forma pública, mediante una presentación oral de su contenido, luego de lo cual se respondió las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de \_\_\_\_ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas a los 17 días del mes de abril del año 2012.

---

Profa. Jossie Zambrano  
(Tutora)

---

Prof. Sergio Rivas  
(Tutor)

---

Profa. Joyce Gutierrez  
(Jurado)

---

Prof. Eugenio Scalise  
(Jurado)

# Resumen

---

La Fundación de la Escuela de Idiomas Modernos (*FUNDEIM*) de la Universidad Central de Venezuela actualmente ofrece cursos de extensión de los idiomas Inglés, Alemán, Italiano, Francés y Portugués. En junio 2011 se desarrolló una *aplicación web* denominada *ACEIM* para apoyar los procesos de inscripción y calificación de la Fundación.

En el presente trabajo, se incorporan nuevos módulos en la *aplicación web* ya existente, los cuales permiten gestionar y planificar los cursos de idiomas. El desarrollo consiste en un sistema que facilita la gestión de cursos, aulas, estudiantes, instructores y cualquier otro tipo de recurso de la Fundación, y permite realizar la planificación trimestral de cada curso de forma sistematizada.

La elaboración del trabajo se realiza usando el método *AgilUS*, el cual considera un diseño centrado en el usuario en cada etapa de desarrollo. La usabilidad es tomada en cuenta desde la etapa inicial de la elaboración del producto hasta su etapa final.

## **Palabras Clave:**

*ACEIM, Aplicación web, Ruby on Rails, AgilUS, Planificación, Gestión, Cursos, Reportes, Idiomas, FUNDEIM.*

# Tabla de Contenido

---

Acta .....	II
Resumen .....	III
Índice de Figuras .....	VI
Índice de Tablas.....	VIII
Introducción.....	9
<b>Capítulo 1 Planteamiento del problema .....</b>	<b>11</b>
1.1. Objetivo General .....	17
1.2. Objetivos Específicos .....	17
1.3. Alcance.....	18
<b>Capítulo 2 Marco conceptual .....</b>	<b>19</b>
2.1. <i>Aplicaciones web</i> .....	20
2.2. <i>Arquitectura cliente-servidor</i> .....	21
2.3. <i>Tecnologías web</i> .....	23
2.3.1. <i>HTML</i> .....	24
2.3.2. <i>CSS</i> .....	26
2.3.3. <i>Javascript</i> .....	27
2.3.4. <i>AJAX</i> .....	29
2.3.5. <i>jQuery</i> .....	30
2.3.6. <i>jQuery UI</i> .....	30
2.3.7. <i>eIRTE</i> .....	32
2.3.8. <i>TableSorter</i> .....	34
2.3.9. <i>Ruby on Rails</i> .....	36
2.3.10. <i>Manejador de versiones de Ruby (RVM)</i> .....	44
2.3.11. <i>Sistema manejador de bases de datos MySQL</i> .....	45
2.3.12. <i>Subversion</i> .....	47

2.4. Método de desarrollo <i>AgilUS</i> .....	48
2.4.1. Ciclo de vida .....	49
2.4.2. Características de <i>AgilUS</i> .....	52
2.4.3. Entregables de <i>AgilUS</i> .....	53
<b>Capítulo 3 Marco Aplicativo .....</b>	<b>58</b>
3.1. Gestión .....	58
3.1.1. Requisitos .....	59
3.1.2. Análisis .....	63
3.1.3. Prototipaje .....	69
3.1.4. Entrega .....	73
3.2. Planificación .....	75
3.2.1. Requisitos .....	75
3.2.2. Análisis .....	79
3.2.3. Prototipaje .....	85
3.2.4. Entrega .....	88
3.3. Reportes .....	90
3.3.1. Requisitos .....	90
3.3.2. Análisis .....	92
3.3.3. Prototipaje .....	95
3.3.4. Entrega .....	96
<b>Conclusiones .....</b>	<b>99</b>
<b>Recomendaciones .....</b>	<b>102</b>
<b>Referencias .....</b>	<b>103</b>

# Índice de Figuras

---

Figura 1.1: Proceso de planificación trimestral de los cursos .....	15
Figura 2.1: Arquitectura <i>cliente-servidor</i> .....	21
Figura 2.2: Arquitectura <i>cliente-servidor</i> . Proceso de petición-respuesta .....	22
Figura 2.3: Tecnologías agrupadas bajo el concepto de AJAX .....	29
Figura 2.4: Ejemplo de un menú con efecto de acordeón.....	31
Figura 2.5: Ejemplo de un calendario implementando el componente <i>Datepicker</i> .....	32
Figura 2.6: Ejemplo de un cuadro de diálogo.....	32
Figura 2.7: Vista preliminar del Editor <i>eRTE</i> .....	34
Figura 2.8: Ejemplo de una <i>Tabla</i> ordenada empleando el <i>plugin tablesorter</i> .....	35
Figura 2.9: Arquitectura <i>cliente-servidor</i> . Ciclo de vida.....	38
Figura 2.10 Etapas de desarrollo y artefactos de <i>AgilUS</i> .....	50
Figura 3.1: Diagrama de casos de uso de gestión .....	64
Figura 3.2: Modelo de objetos del dominio de gestión.....	67
Figura 3.3: Tecnologías empleadas.....	70
Figura 3.4: Interfaz a través de la cual se encuentra la opción de congelar a un estudiante en el período actual.....	71
Figura 3.5: Interfaz para visualizar el listado de alumnos con convenio .....	72
Figura 3.6: Interfaz para la edición del contenido <i>web</i> de <i>ACEIM</i> .....	73
Figura 3.7: Interfaz para configurar los parámetros generales de los cursos.....	74
Figura 3.8: Diagrama de casos de uso de planificación .....	80
Figura 3.9: Modelo de objetos del dominio de planificación .....	82
Figura 3.10: Primer prototipo en papel para la creación de secciones.....	83
Figura 3.11: Prototipo en papel mejorado para la creación de secciones .....	83

Figura 3.12: Primer prototipo en papel para la asignación de horarios a las aulas .....	84
Figura 3.13: Prototipo en papel mejorado para la asignación de horarios a las aulas ..	84
Figura 3.14: Primer prototipo para la asignación de horarios a las aulas .....	85
Figura 3.15: Prototipo final para la asignación de horarios a las aulas .....	85
Figura 3.16: Interfaz para la creación de secciones.....	86
Figura 3.17: Interfaz para asociar uno o más bloques de horario con las aulas disponibles.....	87
Figura 3.18: Interfaz para asignarle un aula a una sección.....	87
Figura 3.19: Interfaz para gestionar los montos de los convenios.....	89
Figura 3.20: Interfaz para añadir un nuevo convenio.....	89
Figura 3.21 Diagrama de Casos de Uso para reportes.....	93
Figura 3.22: Modelo de objetos del dominio de reportes .....	95
Figura 3.23: Reporte: Cantidad de alumnos por idioma.....	95
Figura 3.24: Reporte: Cantidad de alumnos para un idioma en específico .....	96
Figura 3.25: Interfaz para visualizar el reporte de los estudiantes para un convenio en específico.....	97
Figura 3.26: Interfaz para visualizar a los alumnos cuyo cupo se encuentra congelado en el período actual .....	97

# Índice de Tablas

---

Tabla 2.1: Código para inclusión de la librería <i>TableSorter</i> de <i>Javascript</i> en la <i>aplicación</i> .....	34
Tabla 2.2: Código para la invocación del método que ordena la <i>Tabla</i> .....	35
Tabla 2.3: Comparación de un mismo código escrito en <i>ERB</i> y <i>HAML</i> .....	41
Tabla 2.4: Comparación entre líneas de código <i>escritos en CSS, SCSS y SASS</i> .....	42
Tabla 2.5: Comparación entre líneas de código <i>Javascript</i> y líneas de código <i>CoffeeScript</i> .....	43
Tabla 2.6: Código de principales comandos de <i>RVM</i> .....	44
Tabla 3.1: Perfil del usuario Super Administrador de gestión .....	60
Tabla 3.2: Perfil del usuario Administrador de gestión .....	61
Tabla 3.3: Guía de estilo de <i>ACEIM</i> .....	68
Tabla 3.4: Formato de la evaluación heurística aplicada para las funcionalidades de gestión .....	72
Tabla 3.5: Perfil del usuario Super Administrador para el módulo de planificación.....	78
Tabla 3.6: Formato de la evaluación heurística aplicada para planificación .....	87



# Introducción

---

La Fundación de la Escuela de Idiomas Modernos (*FUNDEIM*) forma parte de la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la Universidad Central de Venezuela. Desde hace más de diez (10) años se dedica a impartir cursos de diversos idiomas tales como: inglés, italiano, alemán, francés y portugués. Los idiomas impartidos se subdividen en tres (3) categorías, las cuales clasifican a los estudiantes en grupos; las categorías presentes son adultos, adolescentes (teens) y niños. Todos los idiomas ofrecen una categoría para adultos con una duración de nueve (9) niveles, a excepción del idioma Inglés que ofrece doce (12) niveles para adultos y las categorías adolescentes (teens) con seis (6) niveles y niños con cuatro (4) niveles.

Actualmente los procesos de gestión y planificación de los cursos de idiomas se llevan a cabo por el personal administrativo con poco uso de la tecnología; esto conlleva a que el trabajo realizado sea engorroso y propenso a cometer errores debido a la complejidad, los constantes cambios y la cantidad de variables y factores que dificultan la ejecución de estos procesos.

El presente trabajo forma parte de una alianza entre la Escuela de Computación y la Escuela de Idiomas Modernos, ambas pertenecientes a la Universidad Central de Venezuela. Se propone el desarrollo de un módulo para la *aplicación web ACEIM* que permita la administración y planificación de los cursos de extensión, que apoye la toma de decisiones, los procesos de gestión, planificación y generación de reportes de la fundación.

El presente trabajo se estructura en tres (3) capítulos descritos a continuación:

**Capítulo 1:** planteamiento del problema. En este capítulo se expone la problemática de la situación actual, la importancia y justificación del problema, el objetivo general y los objetivos específicos contemplados y el alcance definido.

**Capítulo 2:** marco conceptual. Se describen los temas más relevantes que se encuentran relacionados con el presente trabajo especial de grado, como lo son las tecnologías *web* y se especifican las bases y las distintas etapas que componen el método *AgilUS*, seleccionado para el desarrollo del presente trabajo.

**Capítulo 3:** marco aplicativo. En este capítulo se exponen y describen las iteraciones llevadas a cabo durante el desarrollo del trabajo especial de grado, especificando las etapas contempladas en el método de desarrollo *AgilUS* seleccionado para el desarrollo del proyecto, detallando las actividades realizadas, los artefactos generados y las entregas efectuadas.

Finalmente, se presentan las conclusiones y recomendaciones del presente trabajo, así como también las referencias.

---

# Capítulo 1

## Planteamiento del problema

---

La Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la Universidad Central de Venezuela cuenta con una Fundación denominada *FUNDEIM*, la cual es una organización que tiene como objetivo “instrumentar proyectos para la generación de recursos financieros necesarios en defensa y promoción de las actividades y objetivos que se propone alcanzar la Escuela de Idiomas Modernos” [FUNDEIM, 2005].

Durante años esta institución ha brindado un conjunto de actividades académicas entre las cuales se encuentra un programa de cursos de extensión de diversos idiomas: alemán, francés, inglés, italiano y portugués. Se encuentra dirigido no sólo a estudiantes de la Universidad Central de Venezuela, sino al público en general. Se ejecutan cuatro (4) períodos al año y tienen una duración de nueve (9) semanas cada período. Los idiomas alemán, francés, italiano y portugués se ofrecen en la categoría adultos para personas mayores de catorce (14) años y está comprendido por nueve (9) niveles. El idioma inglés posee diversas particularidades a diferencia de los demás idiomas:

- Se ofrece a tres (3) tipos de categorías distintas de público:
  - Niños: entre los nueve (9) y once (11) años.
  - Adolescentes: entre los doce (12) y catorce (14) años.
  - Adultos: mayores de catorce (14) años.

- Cada categoría ofrece diferentes niveles:
  - Niños: categoría compuesta por cuatro (4) niveles.
  - Adolescentes (teens): categoría compuesta por seis (6) niveles.
  - Adultos: categoría compuesta por doce (12) niveles.

Para matricularse en un curso, se debe cancelar un monto determinado el cual varía de precio cada período. Anteriormente la Fundación controlaba esta particularidad de manera no automatizada, es decir, colocando manualmente el precio de los cursos en las planillas de inscripción. En la actualidad, el monto de la matrícula es modificado por el grupo de desarrollo, ya que no existe una manera de que la misma Fundación controle esta particularidad.

Existen convenios que posee la Fundación con algunas facultades de la Universidad Central de Venezuela. Cada convenio consiste en proporcionar diversas aulas de una determinada facultad de la universidad, a cambio de un descuento en el costo de la matrícula de los cursos para aquellos estudiantes, docentes y personal administrativo pertenecientes a dicha facultad que realizaron la solicitud del convenio y fueron seleccionados para la *aplicación* del mismo. Cada período se otorgan cuarenta (40) matrículas por cada convenio estructuradas de la siguiente manera: treinta (30) matrículas para estudiantes y diez (10) matrículas para el personal administrativo de cada facultad que posee convenio con *FUNDEIM*. Una consideración que actualmente no controla la Fundación y es de vital importancia, se trata de los convenios otorgados a los estudiantes de las diferentes facultades. No existe manera de validar que los alumnos que afirman tener un convenio, en verdad lo posean.

Existen diversos procesos que se llevan a cabo de manera cíclica durante los cuatro (4) períodos de los cursos de idiomas. Los diferentes procesos son los siguientes:

- Planificación: consiste en programar los diferentes cursos que se llevarán a cabo en el nuevo período. Contempla la creación de secciones en los diferentes idiomas ofertados y la asignación de horarios, aulas e instructores para cada sección. En

este paso es necesario la realización de proyecciones sobre la cantidad aproximada de secciones que deberían crearse por cada curso. Una sección se encuentra asociada a un número determinado de cupos, que, por lo general, incluye de diez (10) a veinte (20) participantes por cada sección.

- Gestión: existen diversos recursos presentes en la Fundación, tales como: instructores, aulas, estudiantes, convenios, entre otros. El proceso de gestión consiste en la administración dichos recursos.
- Inscripción: contempla el ingreso de estudiantes nuevos, los cuales se inscriben por primera vez en los cursos ofrecidos y el ingreso de estudiantes regulares o reincorporados, los cuales se inscriben en el curso asociado al nivel que les corresponde. Primero se ejecuta la inscripción de alumnos regulares y posteriormente se realiza la inscripción de nuevos alumnos.
- Calificación: en este proceso, los instructores califican a sus alumnos.

Hasta el período académico abril-julio de 2011, los datos que maneja la Fundación se almacenaban en una *base de datos* de *Microsoft Office Access*, del cual no se dispone acceso al código fuente por tratarse de un *software* pago. Algunos comprobantes de inscripción y listados requeridos por la Fundación eran creados en *Microsoft Office Excel* y *Microsoft Office Word*.

Actualmente existe una *aplicación web* denominada *ACEIM*, la cual se encuentra en producción desde el penúltimo período del año 2011 y cuenta con su propia *base de datos* en la que se almacenan todos los datos de la Fundación. *ACEIM* incluye varias funcionalidades que brindan apoyo a los procesos de inscripción y calificación; sin embargo, los procesos de gestión y planificación de los cursos todavía no se encuentran desarrollados sobre la *aplicación web*.

Durante la ejecución del proceso de planificación, el coordinador de los cursos debe seguir una serie de pasos descritos a continuación:

1. Consulta de alumnos aprobados y reprobados y consulta de aulas disponibles:
  - Para la consulta de alumnos, el grupo de desarrollo genera un listado con la cantidad de alumnos aprobados y no aprobados por cada curso, tomando sólo la información relacionada al período que acaba de culminar. Para los cursos de

alemán, francés, italiano, portugués e inglés en la categoría de adultos, un alumno aprueba si obtiene una nota mayor o igual a quince (15) puntos. En el caso del idioma inglés para las categorías niños y teens, los alumnos aprueban con una nota mayor o igual a diez (10) puntos.

- Los listados de alumnos aprobados y reprobados se agrupan por sección y son facilitados al coordinador de los cursos para apoyar la toma de decisiones en cuanto al estimado de la cantidad de secciones a crear para el nuevo período.
- Para consultar las aulas disponibles se consignan diversos listados con el conjunto de aulas y horarios asociados que se tienen a disposición, las cuales son suministradas por las diferentes facultades con las que se tiene un convenio. El coordinador de los cursos evalúa la cantidad de aulas disponibles.
- El grupo de desarrollo debe cargar los listados en la *base de datos* de *ACEIM* para que las aulas queden almacenadas en el sistema.

## 2. Creación de secciones y asignación de aulas:

- A través de un documento en *Microsoft Office Excel*, el coordinador de los cursos crea las secciones que fueron estimadas. Las aulas que fueron cargadas en la base de datos de *ACEIM*, son asignadas a las diferentes secciones. Una vez que las secciones son creadas y las aulas son asignadas, el grupo de desarrollo debe cargar el archivo en la base de datos de *ACEIM*.
- Este paso presenta una complejidad durante la asignación de aulas, ya que puede ocurrir un solapamiento de las mismas, es decir, puede que se asigne una misma aula para diferentes secciones en un mismo horario, lo cual no debería suceder.
- Existen cursos que son dictados dos (2) días a la semana, lo cual puede traer como consecuencia que dichos cursos se ubiquen en dos (2) aulas distintas, variable que aumenta la complejidad del problema.

## 3. Asignación de instructores:

- Los instructores poseen una disponibilidad limitada en cuanto a los horarios dentro de los cuales pueden impartir los cursos para los que se encuentran capacitados; por este motivo, el coordinador de los cursos debe comunicarse

con cada instructor y consultar la disponibilidad de su tiempo para asignarlo a una determinada sección en ese momento. No existe la alternativa de poder tener a disposición el horario de cada instructor para una posterior asignación del mismo.

- Debido a la posibilidad de que un instructor dicte más de un curso, puede ocurrir un solapamiento, es decir, un instructor puede quedar asignado a más de un curso en un mismo horario, un error que hasta ahora no puede prevenirse a simple vista.

A continuación se presenta un resumen esquematizado de los pasos del proceso de planificación previamente descritas (ver Figura 1.1).



**Figura 1.1: Proceso de planificación trimestral de los cursos**

Algunos de los pasos se establecen previamente al proceso de inscripción; sin embargo, durante la jornada de inscripción pueden surgir imprevistos que requieren ajustes en la planificación, como en el caso de incorporar nuevas secciones debido a la demanda de más estudiantes de lo estipulado para un curso en específico.

El proceso de planificación se lleva a cabo por cada idioma, categoría y horario y tiene lugar dos (2) veces durante la inscripción: antes de la inscripción de los alumnos regulares y antes de la inscripción de los nuevos alumnos. Al iniciar el nuevo período se planifican los cursos de los alumnos regulares previo a la inscripción de los mismos, y al culminar esta inscripción, la planificación tiene lugar nuevamente para determinar la cantidad de secciones a ofertar para los cursos básicos, es decir, para los alumnos nuevos. En este punto, la planificación de los cursos regulares sufre ajustes para hacer un mejor aprovechamiento de las aulas sobrantes.

Posterior al proceso de inscripción, el coordinador realiza un análisis sobre las secciones que se encuentran abiertas y sobre la cantidad de alumnos asociada a cada una. Al encontrar secciones que no poseen una cantidad suficiente de alumnos, las mismas deben cerrarse con el fin de liberar aulas para ser asignadas a otras secciones. Esta acción implica una comunicación directa con los alumnos inscritos en dichas secciones para notificar la situación y llegar a un acuerdo que puede implicar su traslado a otra sección, un reembolso del pago o congelar el curso al cual se inscribió, en caso de no llegar a una negociación sobre la situación. La acción de congelar a un estudiante implica desincorporar al mismo temporalmente para dicho curso.

*FUNDEIM* requiere llevar a cabo acciones como modificar el precio de la matrícula de los cursos, crear nuevos convenios, abrir o cerrar secciones, modificar las configuraciones generales como establecer el nuevo período, habilitar la inscripción de alumnos, permitir calificar alumnos, entre otras. No existe un soporte tecnológico que apoye dichas acciones. Tampoco se maneja la cantidad de alumnos inscritos por curso ni otros tipos de reportes que pueden apoyar la toma de decisiones de manera oportuna.

Otra necesidad es la difusión de información referente a los cursos, como la fecha de inscripción para nuevos ingresos o para alumnos regulares, el costo de la matrícula, entre otros. Dicha información se puede encontrar en la *página* inicial de *ACEIM*, pero la Fundación no cuenta con un mecanismo para la edición de dicho contenido.

La administración de los diferentes recursos de *FUNDEIM*, al igual que la disposición de diferentes reportes que apoyen la toma de decisiones, no puede llevarse a cabo si no es a través del equipo de desarrollo de *ACEIM*. Esto trae como consecuencia que la información requerida por la Fundación no pueda ser accedida apenas se requiere, desde cualquier ubicación y por cualquier miembro del personal administrativo de los cursos.

Para contrarrestar estos inconvenientes, se desea incorporar un soporte tecnológico que apoye los procesos de gestión, planificación y generación de reportes, añadiendo las funcionalidades necesarias en la *aplicación web ACEIM*. Sería apropiado



que la Fundación tenga control sobre la cantidad de secciones creadas, los estudiantes, instructores, aulas disponibles, entre otros, y otorgar un apoyo sobre el proceso de planificación y generación de reportes y estadísticas. A través de los reportes y las estadísticas, el personal administrativo y coordinador de los cursos podrán obtener información referente a la cantidad de alumnos por curso, cantidad de alumnos por convenio, cantidad de secciones por curso, listado de alumnos congelados y estadísticas generales por cada idioma (necesarias para la planificación), entre otros reportes.

## **1.1. Objetivo General**

Desarrollar un módulo de administración que contemple diversas funcionalidades que permitan generar reportes, gestionar y planificar los cursos de extensión de la Fundación de la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la UCV.

## **1.2. Objetivos Específicos**

- Identificar los pasos que son llevados a cabo al ejecutar procesos como la planificación y gestión de los cursos y la generación de reportes.
- Desarrollar un conjunto de funcionalidades que permitan la gestión de aulas, secciones, períodos, horarios, convenios, cursos, instructores y estudiantes, permitiendo realizar operaciones como visualización, inserción, modificación y eliminación para cada uno de los recursos mencionados anteriormente.
- Desarrollar un conjunto de funcionalidades para la planificación, que permitan proyectar el estimado de cursos que serán ofertados tanto para los alumnos regulares como para los nuevos alumnos.
- Desarrollar un conjunto de funcionalidades que permitan la generación de reportes utilizados en los diferentes procesos de los cursos de extensión.

- Desarrollar una funcionalidad que permita la edición del contenido de la *página* inicial de la *aplicación web*.
- Aplicar el método de desarrollo de software *AgilUS* durante la elaboración del proyecto.

### **1.3. Alcance**

El presente trabajo de investigación consiste en el desarrollo de funcionalidades para los cursos de extensión de la Fundación de la Escuela de Idiomas Modernos, específicamente para la sede ubicada en la Ciudad Universitaria de la Universidad Central de Venezuela.

En base a las observaciones realizadas durante la participación activa en los procesos internos involucrados en los cursos de *FUNDEIM*, se persigue cubrir las necesidades requeridas cumpliendo con los objetivos expuestos en el apartado anterior, los cuales indican de manera explícita la dirección que tiene el trabajo especial de grado, teniendo como exigencia principal la consistencia entre los objetivos propuestos y los resultados obtenidos.

---

# Capítulo 2

## Marco conceptual

---

El marco conceptual de un trabajo de investigación ayuda a explicar por qué se está llevando a cabo un proyecto de una manera determinada. También ayuda a comprender y a utilizar las ideas de otras personas que han hecho trabajos similares [Lazo, 2003].

A lo largo de este capítulo se presentan una serie de conceptos organizados de tal manera que sean fáciles de comunicar al lector. Se da una visión en conjunto de las ideas y las prácticas que conforman el modo en que se llevó a cabo el trabajo especial de grado.

En primera instancia se identifican y enumeran los elementos y tecnologías que fueron utilizadas en la incorporación de las nuevas funcionalidades en la *aplicación web*. Dichos elementos se consideraron idóneos para la integración, desarrollo y uso en conjunto dentro del panorama actual.

Una vez identificados y expuestos los elementos técnicos que intervinieron en el desarrollo del proyecto, se presentan las características y aspectos relevantes del método de desarrollo *AgilUS*, el cual fue empleado para la a lo largo del desarrollo de las funcionalidades que apoyan los procesos involucrados en la planificación y gestión de los cursos de extensión. La finalidad de este método es proporcionar un conjunto de actividades organizadas para construir la usabilidad en el diseño de interfaces de usuario durante el desarrollo de un producto de *software* a lo largo de las etapas de requisitos, análisis, prototipaje y entrega [Acosta, E., 2010].

## 2.1. *Aplicaciones web*

Una *aplicación web* consiste en una *aplicación* “... en la cual un usuario, por medio de un *navegador*, realiza peticiones a una *aplicación* remota accesible a través de Internet (o a través de una intranet) y que recibe una respuesta que se muestra en el propio *navegador*” [Mora, L., 2002].

Actualmente se ha incrementado el desarrollo de *aplicaciones* para el entorno *web* en función de diversos lenguajes de programación, utilizando las distintas alternativas propuestas y tecnologías presentes. Los cambios y las mejoras realizadas en una *aplicación web* son incorporados de manera casi instantánea (a diferencia de las aplicaciones de *escritorio*, las cuales deben instalarse y actualizarse en cada computadora de manera individual). Entre las ventajas que incorporan las *aplicaciones web* se encuentran:

- Fácil de distribuir.
- Fácil de desarrollar.
- Fácil de mantener.
- Plataforma independiente.
- Accesible por cualquier usuario.

Las *aplicaciones web* se desarrollan bajo una arquitectura *cliente-servidor*, donde el *cliente*, es el equipo que solicita los recursos, equipado con una *interfaz de usuario* (generalmente un *navegador web*) para la presentación. El *servidor* (también denominado *software* intermedio), es el encargado de proporcionar los recursos solicitados.

## 2.2. Arquitectura *cliente-servidor*

Desde el punto de vista funcional, se puede definir a la arquitectura *cliente-servidor* como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma [Márquez, B., 2004]. Su funcionamiento consiste en que se tiene una máquina *cliente*, la cual requiere un servicio de una máquina *servidor*, y ésta realiza la función para la que está programada.

Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del *servidor* y la segunda parte consta de un conjunto de *clientes*. Ambas partes deben estar conectadas entre sí mediante una *red*.

Una representación gráfica de este tipo de arquitectura se corresponde con la Figura 2.1:

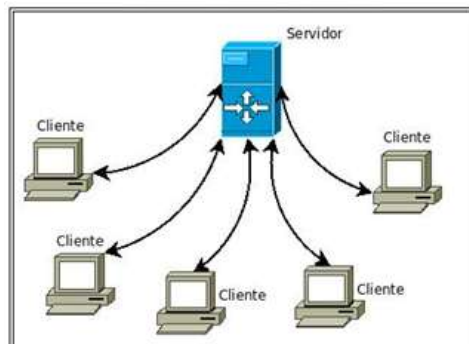


Figura 2.1: Arquitectura *cliente-servidor*

Un *servidor* es una *aplicación* que ofrece un servicio a sus usuarios; un *cliente* es aquel que solicita dicho servicio, es decir, realiza una petición. El *servidor* recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de una respuesta [NEO, 2008].

En la Figura 2.2 se muestra una representación de la interacción entre el *cliente* y el *servidor*.

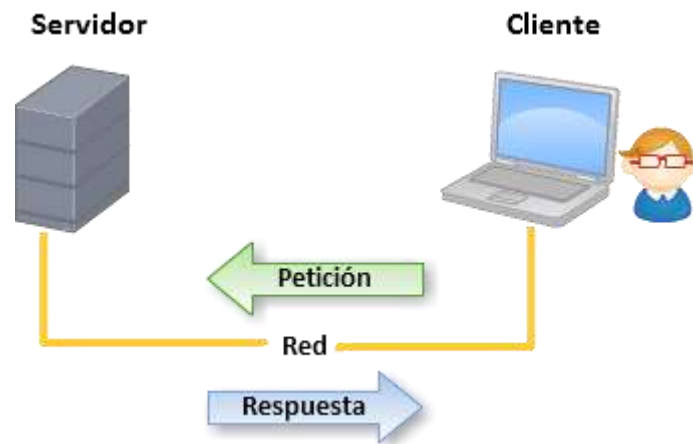


Figura 2.2: Arquitectura *cliente-servidor*. Proceso de petición-respuesta

Generalmente un *servidor* puede tratar múltiples peticiones (múltiples *clientes*) al mismo tiempo. Si se aplica el principio de distribución, cada máquina puede cumplir el rol de *servidor* para algunas tareas y el rol de *cliente* para otras. Los *clientes* y los *servidores* pueden estar conectados a una *red local* o una *red amplia*, como la que se puede implementar en una empresa o a una *red mundial* como lo es Internet.

Para la construcción de una *aplicación web* bajo una *arquitectura cliente-servidor* es recomendable la búsqueda de una solución genérica a problemas de *software* que se asemejen al problema a resolver. A continuación se describe el esquema bajo el cual fue implementado el trabajo especial de grado.

En los apartados anteriores, describieron las dos (2) bases fundamentales sobre las cuales se apoyó el desarrollo del trabajo especial de grado; se trata de una *aplicación web*, construida bajo una *arquitectura cliente-servidor*. En el siguiente apartado se describen el conjunto de herramientas tecnológicas utilizadas en el desarrollo del trabajo especial de grado.

## 2.3. Tecnologías *web*

Actualmente son muchas las organizaciones que se apoyan en el ámbito tecnológico para la ejecución de sus procesos. La tecnología evoluciona en cortos períodos de tiempo y surgen nuevas técnicas y herramientas que permiten ampliar las alternativas ante situaciones que requieran opciones tecnológicas.

Para el desarrollo de *aplicaciones web* se hace necesario el uso de un conjunto de tecnologías de *software* del lado del *servidor* y del *cliente* que involucran una combinación de procesos de *base de datos* con el uso de un *navegador* en Internet a fin de realizar determinadas tareas o mostrar información.

Como se trata de un desarrollo para la Universidad Central de Venezuela, (la cual es una organización pública) se investiga a fin de usar herramientas *open source*<sup>1</sup>, lo cual se sugiere en el Decreto N° 3.390, publicado en la Gaceta Oficial N° 38.095 de fecha 28/ 12/ 2004 sobre el uso obligatorio del software libre en el país para todas las dependencias públicas de carácter oficial. De esta forma, el Ejecutivo nacional establece que es prioridad del Estado incentivar y fomentar la producción de bienes y servicios para satisfacer las necesidades de la población, mediante el uso de estas herramientas desarrolladas con estándares abiertos para robustecer la industria nacional, aumentando y aprovechando sus capacidades y fortaleciendo nuestra soberanía. La ventaja principal de lo anteriormente expresado se evidencia desde el punto de vista monetario, ya que se eliminan los costos de licencia para el producto en sí mismo. Otra ventaja importante es que se puede disponer del *código fuente*, lo cual brinda independencia de proveedores.

---

<sup>1</sup> *Open Source*, (Código Abierto en español) es el término con el que se conoce al *software* distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones éticas y morales las cuales destacan en el llamado *software* libre [Wikipedia, 2012].

### 2.3.1. HTML

*HTML* es el acrónimo de *Lenguaje de Marcado de Hipertexto* (HyperText Markup Language, por sus siglas en inglés) y es el lenguaje que se utiliza para crear las *páginas web*. Es uno de los principales componentes de la plataforma *web* abierta [W3C, 2011].

Es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada, con enlaces que conducen a otros documentos o fuentes de información relacionadas. Además de texto, puede contener inserciones multimedia tales como imágenes, gráficos, sonido etc. y así presentar un producto más dinámico. La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc) así como los diferentes efectos que se quieren dar sobre el mismo, para luego finalmente dejar que la presentación final de dicho hipertexto se realice por un *navegador web*.

El lenguaje *HTML* indica a los *navegadores* cómo deben mostrar el contenido de una *página web*. Los *navegadores* se encargan de interpretar el código *HTML* de los documentos, y de mostrar a los usuarios las *páginas web* resultantes del código interpretado.

*HTML* se creó en un principio con objetivos meramente divulgativos. No se pensó que la *web* llegaría a ser un área de ocio con carácter multimedia, de modo que, el *HTML* se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, se han ido incorporando modificaciones poco a poco con el tiempo, y ahora todo eso encapsulado se transformó en estándares del *HTML*.

*HTML5* es la última versión de *HTML*. Con esta versión se introducen nuevas características en una variedad de áreas. Se trata de una tecnología creada para modernizar la *web* y el desarrollo de *aplicaciones web, online* y *offline* que nació para dar cabida elementos como vídeo, audio, geolocalización, nuevos componentes y



etiquetas, que constituyen la base del gran cambio de Internet en la época actual [Gil, V., 2010].

*HTML5* persigue proporcionar una plataforma que permita desarrollar *aplicaciones web* más semejantes a las *aplicaciones de escritorio*, donde su ejecución dentro de un navegador no implique falta de recursos o facilidades para resolver las necesidades reales de los desarrolladores. Para ello se están creando unas *APIs*<sup>2</sup> que permitan trabajar con cualquiera de los elementos de la página y realizar acciones que hasta hoy era necesario realizar por medio de tecnologías accesorias. Estas *APIs*, que tendrán que ser implementadas por los distintos *navegadores* del mercado, se están documentando con minuciosidad, para que todos aquellos *navegadores* creados por cualquier compañía, les provean soporte tal cual se han diseñado. Este hecho está relacionado con la intención de que no ocurra lo que ha ocurrido en el pasado, en relación a que cada *navegador* se presenta de manera competitiva con el resto de los *navegadores*, y los que sufren las consecuencias son los desarrolladores y usuarios al presentar la posibilidad de acceder a *webs* que no son compatibles con su *navegador* preferido.

Al tratarse de una tecnología completamente nueva, progresivamente se han ido desarrollando las respectivas adaptaciones en los *navegadores web* para hacerlos compatibles con el lenguaje. Actualmente los *navegadores Safari* y *Google Chrome* son los que poseen mayor soporte del lenguaje, seguido por *Mozilla Firefox* y *Opera*. La compatibilidad con *Internet Explorer* en este momento es casi nula, pero la versión 9, que es la más reciente, posee soporte de algunas características.

Hoy en día, pocos *sitios web* se basan únicamente en *HTML*. En cualquier desarrollo *web* actual, se hace uso de *CSS* (hojas de estilo) para definir el aspecto visual de la página, y de algún tipo de *script* del lado del *cliente*, en lenguaje *Javascript*, tales como videos en diversos formatos o *Flash* (para realizar alguna animación o interacción con el usuario).

---

<sup>2</sup> Interfaz de programación de aplicaciones o *API* (del inglés *Application Programming Interface*) es el conjunto de funciones, procedimientos y métodos que ofrece cierta *biblioteca* o *librería* para ser utilizado por otro *software* como una capa de abstracción [Wikipedia, 2012].

### 2.3.2. CSS

“Las *Hojas de Estilo en Cascada* (Cascading Style Sheets, comúnmente abreviadas *CSS* por sus siglas en inglés) son un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de cualquier dispositivo. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos” [W3C, 2008].

*CSS* es un lenguaje utilizado para especificar el aspecto de una *página web* para diferentes dispositivos. Este aspecto contrasta con la concepción de *HTML*, el cual es un lenguaje que define la estructura de un documento para su distribución por la *web*. A través de *CSS* se consigue separar los contenidos de la *página web* de su presentación; además resulta útil al momento de elaborar *páginas web* complejas en cuanto a código.

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

El objetivo inicial de *CSS* era separar el contenido de la forma de visualización y se cumplió con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a las *webs*, pero los desarrolladores aún continúan usando diversos trucos para conseguir efectos que cada vez son más requeridos por los usuarios finales.

*CSS3* es la última versión de *CSS* y se presenta como una versión con mayor control sobre el estilo de los elementos de *páginas web*, ya que añade nuevas capacidades a la especificación anterior. Aunque aún está en proceso de

estandarización, supone uno de los mayores adelantos en el *diseño web* actual [Campos, O., 2011].

La novedad más importante que aporta *CSS3* consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las *páginas*, sin tener que recurrir a trucos, que a menudo complicaban el código de las *páginas web*.

La ventaja principal en esta nueva versión se trata de la inclusión de nuevas propiedades, especialmente en cuanto al aspecto gráfico, aunque se esperan mejoras sustanciales en otros medios como el del sonido.

Debido a la incorporación de propiedades novedosas, la carga de la página se realiza rápidamente porque muchos de los efectos se encuentran bajo control del navegador, por lo cual, los recursos visuales e imágenes que se emplean, ya no tienen razón de seguir siendo utilizados. Esta nueva tecnología se encuentra en pleno crecimiento; por este motivo, es imprescindible conocer el soporte de *CSS* en cada uno de los *navegadores* más utilizados del *mercado*.

### **2.3.3. Javascript**

*Javascript* es un poderoso lenguaje de *programación* interpretado que puede ser embebido directamente en código *HTML*. Permite crear *aplicaciones web* dinámicas e interactivas ejecutadas por completo dentro de un *navegador web*. No es necesario ningún tipo de *programación* del lado del servidor [Flanagan, 1998].

Todos los *navegadores* modernos interpretan el código *Javascript* integrado en las *páginas web*. Para interactuar con una *página web* se provee al lenguaje *Javascript* de una implementación *DOM*<sup>3</sup>.

---

<sup>3</sup> *Document Object Model* (*DOM*, Modelo de Objetos del Documento en español), Es una *API* de lenguaje neutro, que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, estructura y estilo de los documentos *web*. El documento puede ser procesado y los resultados del procesamiento se pueden incorporar de nuevo en la *página* presentada [W3C,2011].

Tradicionalmente se viene utilizando en el desarrollo de *páginas web* escritas en *HTML* para realizar operaciones únicamente en el marco de la *aplicación cliente*, sin acceso a funciones del *servidor*. *Javascript* se interpreta en el agente de usuario (*cliente*), al mismo tiempo que las sentencias van descargándose junto con el código *HTML*. De manera técnica es un lenguaje de programación interpretado, esto quiere decir que no es necesario compilar los programas para ejecutarlos. Una *aplicación* que utiliza *Javascript* se puede probar directamente en cualquier *navegador* sin necesidad de procesos intermedios.

La integración de *Javascript* y *HTML* es muy flexible, ya que existen al menos tres formas para incluir código *Javascript* en las *páginas web*. Una de ellas consiste en incluir *Javascript* en el mismo documento *HTML*. El código *Javascript* se encierra entre etiquetas `<script>` y se incluye en cualquier parte del documento *HTML*. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código *Javascript* dentro de la cabecera del documento.

Las instrucciones *Javascript* también se pueden incluir en un archivo externo de tipo *Javascript* que los documentos *HTML* se encargan de enlazar. Los archivos de tipo *Javascript* son documentos normales de texto con la extensión “.js”. La principal ventaja de enlazar un archivo *Javascript* externo es que se simplifica el código *HTML* de la página, que se puede reutilizar el mismo código *Javascript* en todas las *páginas* del *sitio web* y que cualquier modificación realizada en el archivo *Javascript* se ve reflejada inmediatamente en todas las *páginas HTML* que lo enlazan.

*Javascript* es soportado por la mayoría de los navegadores como *Internet Explorer*, *Opera*, *Mozilla Firefox*, *Google Chrome*, entre otros; sin embargo, se debe tomar en cuenta que aunque *Javascript* sea soportado en gran cantidad de *navegadores*, los usuarios pueden elegir la opción de activar/desactivar el *Javascript* en los mismos.

### 2.3.4. AJAX

*JavaScript* Asíncrono y *XML*, *AJAX* (acrónimo de *Asynchronous JavaScript And XML*, por sus siglas en inglés) se trata de una técnica de desarrollo *web* para crear *aplicaciones* interactivas, las cuales se ejecutan del lado del *cliente* y mantienen una comunicación asíncrona con el *servidor* en segundo plano, consiguiendo realizar cambios sobre la misma página sin necesidad de recargarla. Esto implica un aumento de la interactividad, velocidad y usabilidad en la misma. “*Ajax* no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes” [Garrent, 2005].

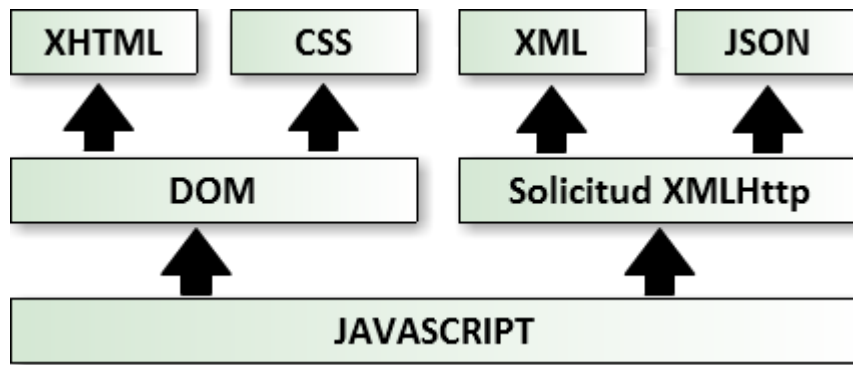


Figura 2.3: Tecnologías agrupadas bajo el concepto de AJAX

En la Figura 2.3 se exponen las diferentes tecnologías que forman *AJAX*. En el nivel superior se encuentra *XHTML* y *CSS* para crear una presentación basada en estándares, y *XML* y *JSON* para proveer un formato para el intercambio y manipulación de la información. En el siguiente nivel se encuentra *DOM* para la interacción y manipulación dinámica de la presentación, así como *XMLHttpRequest* para el intercambio asíncrono de información. Por último, en el nivel base se encuentra *Javascript* para unir todas las demás tecnologías. Desarrollar aplicaciones *AJAX* requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores.

En las *aplicaciones web* tradicionales, cualquier acción del usuario en la *página* desencadena llamadas al *servidor*. Una vez procesada la petición del usuario, el *servidor* devuelve una nueva *página HTML* al *navegador* del usuario. Esta técnica

funciona correctamente, pero no le otorga al usuario la mejor visualización al recargarse la *página*, ya que debe esperar que la misma se actualice completamente. Esto sugiere un inconveniente si la *aplicación* realiza continuas peticiones al *servidor*.

*AJAX* permite mejorar la interacción del usuario con la *aplicación*, evitando recargar la *página* por completo y constantemente mediante la creación de una capa adicional que funciona como intermediario entre el usuario y el *servidor*. Dicha capa mejora el tiempo de respuesta de la *aplicación*, ya que el usuario no debe esperar la respuesta del *servidor* con la ventana del *navegador* vacía.

### **2.3.5. jQuery**

Se trata de una librería escrita en lenguaje *Javascript* que simplifica el código *HTML* y ofrece una infraestructura que facilita la programación de aplicaciones complejas, permitiendo el manejo de eventos, animaciones e interacciones con *AJAX* para un rápido desarrollo *web* en cualquier tipo de plataforma, ya sea personal o comercial [jQuery, 2010].

Cuando un desarrollador programa *aplicaciones web* utilizando el lenguaje *Javascript*, es común que deba preocuparse por la compatibilidad de los *scripts* con respecto a los *navegadores*, incluyendo código que permita detectar el *navegador* utilizado por el usuario para realizar una u otra acción dependiendo del mismo. Con *jQuery* no hay necesidad de tomar en cuenta este problema, ya que esta librería funciona de igual forma en cualquier *navegador*, esto gracias a la incorporación de una serie de clases que permiten al desarrollador programar sin tomar en cuenta el *navegador* en el cual se desplegará la *aplicación*, además el uso de esta librería es no intrusivo ya que está separado de la estructura *HTML* de la *aplicación*.

### **2.3.6. jQuery UI**

Se trata de un *plugin* basado en *jQuery* que extiende las funcionalidades de la librería subyacente de *jQuery* para proveer un conjunto de componentes ricos e interactivos, con la finalidad de ahorrar código y mejorar las interfaces de usuario en

las *páginas* o *aplicaciones web* mediante el uso de los *helpers* [Wellman, D. 2011]. Se encuentra compuesto por cuatro módulos:

- **Núcleo:** funciones básicas para los otros módulos.
- **Componentes de interacción:** *drag & drop* (arrastrar y soltar), ordenamiento, redimensionamiento, entre otros.
- **Efectos:** mostrar, ocultar, color, animación, entre otros.
- **Widgets:** un *widget* se trata de un componente reutilizable de la *interfaz de usuario*, visualmente atractivo y configurable. Entre *widgets* más comunes se encuentran:
  - **Menú con efecto acordeón:** permite mostrar diversos contenidos clasificados en secciones. Como se observa en la Figura 2.4, cada sección posee un título que se utiliza para abrir el contenedor y mostrar su contenido. Al presionar el encabezado de una de las secciones, su contenido se visualiza justo debajo del mismo. Se puede visualizar el contenido de una sección determinada hasta que se cierra automáticamente luego de abrir otra sección. La ventaja de este componente de *jQuery* es que permite ahorrar espacio en la *página web* al permitir visualizar una sola sección a la vez.

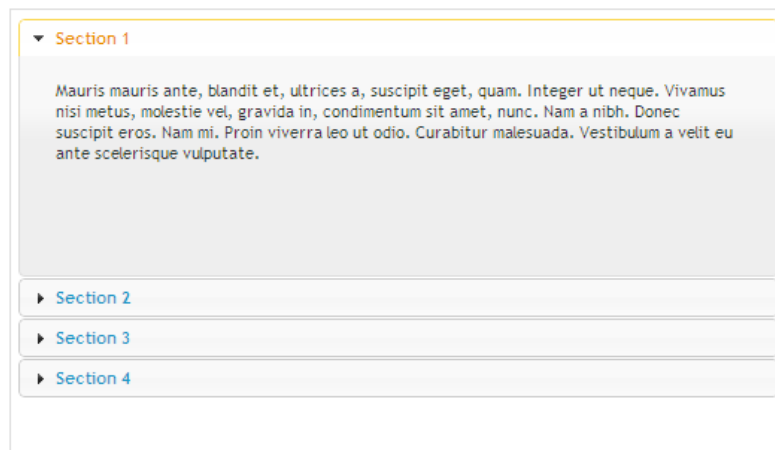


Figura 2.4: Ejemplo de un menú con efecto de acordeón

- **Calendario gráfico:** conocido como *datepicker*, es un *plugin* configurable que permite incorporar la funcionalidad de un calendario a una *página web*. Se puede personalizar el formato e idioma de la fecha, restringir los rangos de

fechas seleccionables y añadir *botones* y otras opciones para *navegar* fácilmente entre las fechas (ver Figura 2.5).

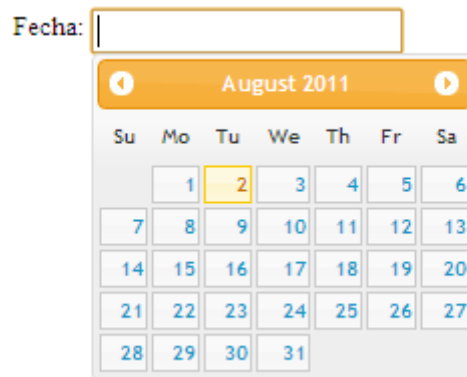


Figura 2.5: Ejemplo de un calendario implementando el componente *Datepicker*

- **Diálogos:** un diálogo es una ventana flotante que contiene una barra de título y un área de contenido como se observa en la Figura 2.6. Una ventana de diálogo puede ser desplazada, cambiada de tamaño y cerrada. Permite mostrar un mensaje específico, incluir imágenes o inclusive, contenido interactivo como un *formulario*.

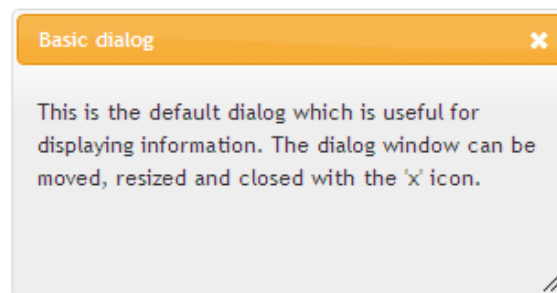


Figura 2.6: Ejemplo de un cuadro de diálogo

### 2.3.7. eIRTE

*eIRTE* es un editor *gratuito WYSIWYG* (es el acrónimo de *What You See Is What You Get* en inglés, "lo que ves es lo que obtienes") para los *sitios web* y sistemas de administración de contenidos (*CMS - Content Management System*, Sistema Manejador de Contenidos, en español) escrito en *JavaScript* con la utilización de



*jQuery UI*. Al ser basado en *JavaScript*, *eIRTE* es independiente de la *plataforma* y se ejecuta en cualquier *navegador* de Internet; además puede ser utilizado libremente en cualquier proyecto, tanto comercial como no comercial [ELRTE, 2011].

El término *WYSIWYG* aplica a los procesadores de texto y otros editores de texto con formato (como los editores de *HTML*) que permiten escribir un documento visualizando directamente el resultado final. En el caso de editores de *HTML*, este concepto se aplica a aquellos editores que permiten escribir la *página* sobre una vista preliminar similar a la de un procesador de textos, ocupándose el programa de generar el *código fuente* en *HTML*.

El objetivo principal del editor es simplificar el trabajo con el texto y la maquetación (*HTML*) en *sitios web*, blogs, foros y otros servicios *online*.

Algunas de sus características destacables son:

- Edición de texto, modificación de su apariencia y estilo.
- Inserción y cambio de propiedades de distintos elementos (imágenes, *tablas*, listas, etc.)
- Visualización y edición de *HTML*.
- Pequeño tamaño, sencillo de integrar en el *sitio web* y alta velocidad de carga
- La apariencia del editor se configura mediante un archivo *CSS*.
- El editor puede ser traducido a cualquier idioma.
- Las funcionalidades pueden ser fácilmente ampliadas por el desarrollador.

A continuación se muestra una vista previa de *eIRTE* (ver Figura 2.7).

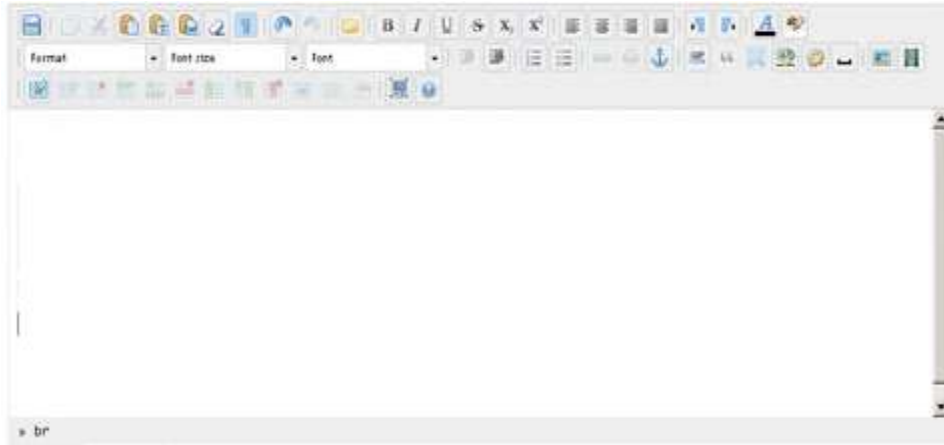


Figura 2.7: Vista preliminar del Editor *eIRTE*

### 2.3.8. *TableSorter*

Se trata de un *plugin* de *jQuery* que permite ordenar las columnas de una *Tabla HTML* sin tener que refrescar la *página*. La *tabla* puede ser ordenada por diversos tipos de datos tales como texto, numérico, fecha, formato personalizado, entre otros, los cuales son detectados automáticamente por el *plugin*. Una de las ventajas es que se permite de la ordenación de la *Tabla* en base a múltiples columnas y es compatible con todos los *navegadores web* [TableSorter, 2010].

Para su utilización sólo es necesario incluir la *librería* de *jQuery* y el *plugin* en el documento *HTML* de la manera como se muestra en el siguiente código:

Tabla 2.1: Código para inclusión de la librería *TableSorter* de *Javascript* en la *aplicación*

```
<script src="js/jquery.js" type="text/javascript"></script>
<script src="js/jquery.tablesorter.js" type="text/javascript"></script>
```

Como se observa en la Tabla 2.1, dentro de las dos etiquetas (*<script>* y *</script>*) se incluye código de tipo *JavaScript*. También se utilizan para enlazar un archivo externo de tipo *JavaScript* (especificado a través del atributo *type*) que en este caso se trata de las *librerías jQuery* y *TableSorter*. Además del atributo *type*, es necesario definir el atributo *src* que indica la *URL* correspondiente al archivo que se

quiere enlazar. Cada etiqueta `<script>` puede enlazar un único archivo, pero en una misma *página* se pueden incluir tantas etiquetas `<script>` como sean necesarias.

Para su funcionamiento, basta con incluir las siguientes líneas de código que se muestran en la Tabla 2.2, las cuales permiten llevar a cabo el ordenamiento.

**Tabla 2.2: Código para la invocación del método que ordena la *Tabla***

```
$(document).ready(function() {
    $("#NombreTabla").tablesorter();
});
```

Siguiendo el ejemplo de la Tabla 2.2, a través de la sentencia `$(document)` se obtiene una referencia del documento (la *página web*) que se está cargando. Luego, con el método `ready()` se define un evento, el cual se activa al quedar listo el documento para realizar acciones sobre el *DOM* de la *página*. Finalmente basta con indicar la *Tabla* que desea ordenarse e invocar al método `tablesorter()`.

A continuación se muestra un ejemplo de cómo luce una *Tabla* luego de asociarla al método de ordenamiento:

ID	Nombre	Teléfono	Correo Electrónico	Código Postal	Fecha de Nacimiento	Último Acceso
5	Porter Thomas	(666) 569-9894	<a href="mailto:non@Proin.ca">non@Proin.ca</a>	59953	09/27/1986	December 05, 2007
27	Patience Battle	(294) 644-5306	<a href="mailto:tempus.mauris@elemurus.com">tempus.mauris@elemurus.com</a>	43578	09/16/1988	October 19, 2003
31	Levi Britt	(272) 171-5731	<a href="mailto:felis@Donecfeugiat.ca">felis@Donecfeugiat.ca</a>	56521	12/10/1988	August 11, 2008
16	Iliana Ballard	(806) 835-7035	<a href="mailto:vel.sapien@mi.ca">vel.sapien@mi.ca</a>	84718	02/09/1989	March 27, 1996
17	Alisa Monroe	(859) 974-4442	<a href="mailto:adipiscing.ligula@aretraNam.edu">adipiscing.ligula@aretraNam.edu</a>	89057	02/14/1990	April 30, 2003

**Figura 2.8: Ejemplo de una *Tabla* ordenada empleando el *plugin tablesorter***

Como se aprecia en la Figura 2.8, los datos se muestran ordenados de manera ascendente según el campo “fecha de nacimiento”.

Existen otros métodos y opciones de configuración del *plugin* que están disponibles y especificados en su documentación oficial.

### 2.3.9. *Ruby on Rails*

“*Ruby on Rails* o *Rails* es un *framework* para *aplicaciones web* desarrollado por David Heinemeier Hansson, basado en el lenguaje de programación *Ruby*” [Ruby on Rails, 2010]. La primera versión fue liberada en el año 2004 y actualmente se encuentra en desarrollo de nuevas versiones. Resulta un *framework* bastante práctico porque ayuda a construir *sitios web* de manera rápida, con código ordenado y que es fácil de mantener.

“*Ruby* es un lenguaje de programación interpretado, dinámico, reflexivo y orientado a objetos” [Ruby, 2010]. Su creador, Yukihiro “Matz” Matsumoto, mezcló partes de sus lenguajes favoritos (*Perl*, *Python*, *Smalltalk*, *Eiffel*, *Ada*, y *Lisp*) para formar un nuevo lenguaje que incorporara tanto la programación funcional como la programación imperativa, y una sintaxis inspirada en los lenguajes mencionados. Su implementación oficial es distribuida bajo una licencia de *software* libre; esto quiere decir que no solamente es gratis, sino que también es libre para usarse, copiarse, modificarse y distribuirse. Es considerado un lenguaje flexible, ya que permite a sus usuarios alterarlo libremente; un ejemplo ello es que las partes esenciales de *Ruby* pueden ser quitadas o redefinidas.

*Ruby on Rails* es de código abierto y multiplataforma. Proporciona una capa de abstracción de *bases de datos* de gran alcance llamada *Active Record*, la cual trabaja con los sistemas manejadores de *bases de datos* más empleados. Provee un sistema multicapa para organizar los diferentes archivos que conforman la *aplicación*.

*Rails* tiene convenciones específicas sobre la estructura de directorios, los nombres de archivos, las estructuras de datos, los argumentos de un método, etc. Cuando se escribe una *aplicación* utilizando *Rails*, deben seguirse dichas convenciones estipuladas por el *framework*.

El *framework* emplea un enfoque bajo el *patrón de diseño MVC* el cual será descrito a continuación.

### **Patrón Modelo-Vista-Controlador (MVC)**

Un *patrón de diseño* se trata de una solución a un problema en un contexto determinado, proporcionando una estructura, efectiva y reutilizable, bien definida para la construcción y/o desarrollo de *software*. De manera formal:

No hay ninguna definición generalmente aceptada de patrón, pero tal vez el mejor lugar para empezar es Christopher Alexander, una inspiración para muchos entusiastas de los patrones: “Cada patrón describe un problema que se produce una y otra vez en nuestro entorno y, a continuación describe el núcleo de la solución a ese problema, de tal forma que puede utilizar esta solución millones de veces, sin hacerlo de la misma manera dos veces”.

Los *patrones* no son ideas originales, sino que son muchas observaciones de lo que sucede en el campo. Como resultado, los autores de *patrones* no decimos que “inventamos” un patrón, sino que “descubrimos” uno. Nuestro papel es observar la solución común, buscar su esencia, y luego escribir el patrón resultante [Fowler, M., 2002].

El *patrón MVC* separa los datos de una *aplicación*, la *interfaz de usuario*, y la lógica de control en tres componentes distintos: *modelo*, *vista* y *controlador*, los cuales se tratan como entidades separadas y son capaces de diseñarse y probarse de manera aislada [Wikipedia, 2012].

El *modelo* consiste en la representación de los datos y las reglas del negocio presentes en la *aplicación*, los cuales, una vez colocados en contexto del sistema, proveen de información a la *aplicación* o al usuario. La *vista* representa el *modelo* en forma gráfica, es decir, la *interfaz de usuario*. El *controlador* se encarga de dirigir y responder todas las solicitudes realizadas por el usuario [Teixidor, S., 2010].

Como resultado de esta organización en base a tres componentes, el *patrón de diseño MVC* consigue separar la lógica de negocio (el *modelo*) y la presentación (la *vista*), aportando como ventaja un mantenimiento más sencillo de las *aplicaciones*.

Otra ventaja presente es que al momento de incorporar un cambio en el *modelo*, no tiene por qué afectar a las *vistas*, al igual que un cambio en una *vista* no tiene por qué tener efecto en el *modelo*. “... un cambio en una aplicación *MVC* tiende a ser localizado y de bajo impacto, facilitando considerablemente el mantenimiento mientras aumenta el nivel de reutilización entre los componentes” [Carneiro, C. y Al Barazi R., 2010].

A continuación, en la Figura 2.9, se expone gráficamente el proceso de solicitud-respuesta en un enfoque *cliente-servidor*.

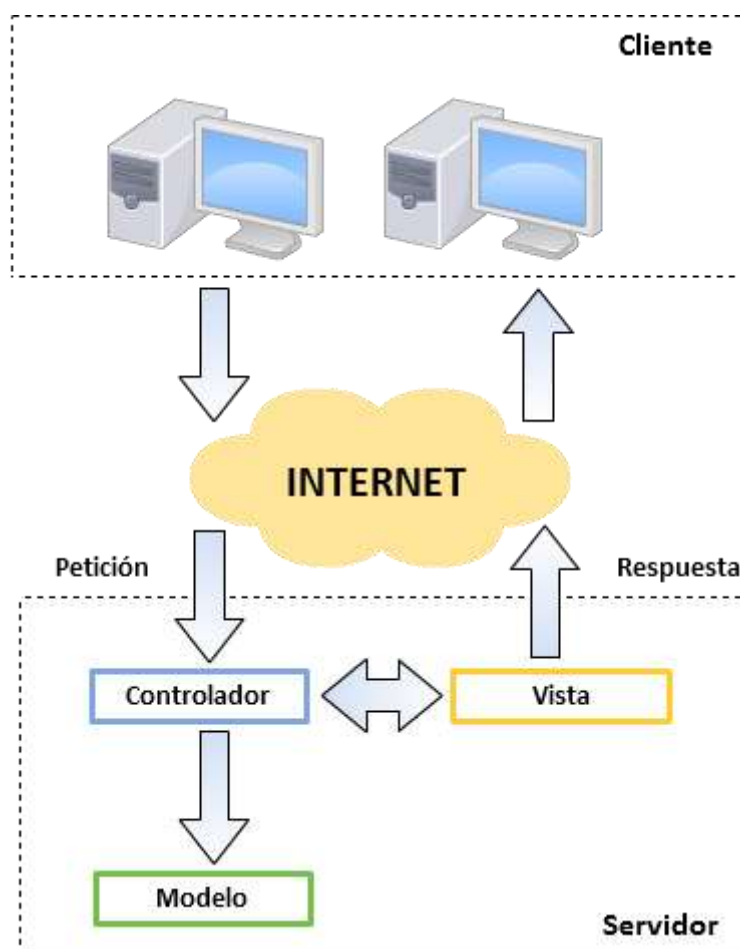


Figura 2.9: Arquitectura *cliente-servidor*. Ciclo de vida.

Como se aprecia en la Figura 2.9, el ciclo inicia cuando un usuario se conecta a Internet, a través del *protocolo HTTP*<sup>4</sup>, y realiza una solicitud al *controlador*. Es tarea

<sup>4</sup> *HTTP* es un protocolo que permite la transferencia de archivos (principalmente en formato *HTML*) entre un *cliente* y un *servidor*. Se encarga de procesar y dar respuestas a las peticiones para visualizar una *aplicación web* [Diccionario de Informática, 2011].

del *controlador* atender dicha solicitud y delegar la tarea al *modelo*, el cual es el ente encargado de realizar operaciones sobre la información que maneja para cumplir con lo que le delegó el *controlador*. Una vez que el *modelo* finaliza su labor, emite una respuesta al *controlador* con la información obtenida luego de realizar las operaciones necesarias. El *controlador* debe redirigir dicha respuesta a la *vista*, quien se encargará de transformar la respuesta en información entendible por el usuario. Una vez realizado esto, la *vista* transmite la representación gráfica de los datos al *controlador*, quien se encargará de transmitírsela al usuario que emitió la solicitud. Este ciclo puede llevarse a cabo cuantas veces lo requiera el usuario.

Cada capa de *MVC* se corresponde con un módulo definido en *Rails*:

- **Modelo (*ActiveRecord*):** se encarga de mantener la relación entre un *objeto* y la *base de datos*. Es implementado a través de la *librería ActiveRecord*, la cual provee una *interfaz* entre las *Tablas* en una *base de datos* relacional y el *código* de la *aplicación* desarrollada en *Ruby*, para manipular los registros de la *base de datos*. Los nombres de los métodos en *Ruby* son generados automáticamente de los nombres de los campos en las *Tablas* de de la *base de datos*.
- **Vista (*ActionView*):** consiste en la representación de los datos en un formato particular. Se implementa mediante la *librería ActiveSupport* que se encuentra *embebida* en *Ruby*, basada en un sistema para definir plantillas de presentación para mostrar la información.
- **Controlador (*ActionController*):** provee una facilidad dentro de la *aplicación* para dirigir el tráfico de las solicitudes, por un lado accediendo a los datos a través del *modelo*, y por otro lado, organizando los datos en una plantilla suministrada por la *vista* para una presentación de los mismos. Implementa un *ActionController*, el cual se sitúa entre el *ActiveRecord* (la *interfaz* de la *base de datos*) y el *ActionView* (el *motor* de presentación).

*Ruby on Rails* continúa siendo una de las herramientas más utilizadas en el área de desarrollo de *aplicaciones web* a pesar de tener 7 años en el mercado; cada versión nueva del *framework* incluye características que hacen más poderosas y productivas a

las aplicaciones hechas en *Rails*. La última versión que ha sido desarrollada es la versión 3.1, la misma fue la usada en el desarrollo de la *aplicación web* que se propuso.

Al hacer uso de la versión 3.1 de *Rails* se decidió hacer la integración de tres (3) nuevas tecnologías de desarrollo que se complementan con el *framework*: *HAML*, *SASS* y *Coffeescript*. Las mismas serán descritas a continuación.

### **HAML**

*HAML* (en inglés, *HTML Abstraction Markup Language*) es un lenguaje de marcado que se utiliza para describir de manera limpia y simple el código *HTML* de cualquier documento *web* sin el uso de código *embebido* tradicional.

Funciona como un reemplazo para los sistemas de plantillas de *páginas* tales como *ERB*<sup>5</sup>, el lenguaje de plantillas usado en la mayoría de *Ruby on Rails*. Sin embargo, *HAML* evita la necesidad de codificar explícitamente *HTML* en la plantilla, ya que es en sí misma una descripción del código *HTML*, con algo de código para generar contenido dinámico lo que encaja mejor con la filosofía de rails [HAML, 2011].

*HAML* es otra forma de crear *HTML*, diseñado para ser un lenguaje de marcado tan elegante como sea posible, usando la indentación del código no como la buena práctica que suele ser, sino como una obligación que tiene significado propio, logrando así una manera más condensada y gráfica de crear plantillas *HTML*.

---

<sup>5</sup> Desde *Ruby on Rails* en su versión 2, las extensiones estándares para las vistas son las *.erb*, (*ERB*, *HTML with embedded Ruby*, *HTML con código integrado Ruby*) [Ruby, 2011].



Una comparación entre un mismo trozo de código escrito en *ERB* y otro en *HAML* se encuentra representada en la Tabla 2.3.

**Tabla 2.3: Comparación de un mismo código escrito en *ERB* y *HAML***

<i>ERB</i>	<i>HAML</i>
<pre> &lt;div id="profile"&gt;   &lt;div class="left column"&gt;     &lt;div id="date"&gt;       &lt;%= print_date%&gt;     &lt;/div&gt;     &lt;div id="address"&gt;       &lt;%=current_user.address %&gt;     &lt;/div&gt;   &lt;/div&gt;   &lt;div class="right column"&gt;     &lt;div id="email"&gt;       &lt;%= current_user.email %&gt;     &lt;/div&gt;     &lt;div id="bio"&gt;       &lt;%= current_user.bio %&gt;     &lt;/div&gt;   &lt;/div&gt; &lt;/div&gt; </pre>	<pre> #profile   .left.column     #date= print_date     #address= current_user.address   .right.column     #email= current_user.email     #bio= current_user.bio </pre>

## SASS

*SASS* (en inglés, *Syntactically Awesome Stylesheets*) es un metalenguaje de las *hojas de estilo en cascada* (CSS). Se trata de un lenguaje de *scripting* que se interpreta al CSS y se utiliza para describir el estilo de un documento de manera limpia estructuralmente. *SASS* proporciona una sintaxis más simple, más elegante para CSS e implementa varias características que son útiles para la creación de hojas de estilo más manejables [SASS, 2012].

*SASS* se compone de dos sintaxis; la sintaxis original, llamado “la sintaxis de indentación” la cual utiliza una sintaxis similar a *HAML*. La nueva sintaxis “*SCSS*” utiliza un formato de bloques como el de la CSS. Para cada una de las sintaxis se han dado nombres de extensiones, *.sass* y *.scss* respectivamente.

Este metalenguaje se extiende CSS mediante varios mecanismos disponibles en los lenguajes de programación más tradicionales, en particular, lenguajes orientados a objetos. Cuando el *script* de SASS es interpretado, crea bloques de reglas CSS para los diferentes elementos *HTML* según se definen en el archivo SASS. Alternativamente, SASS puede monitorear los archivos .sass o .scss. y los traduce a un archivo de salida .css cada vez que los mismos son guardados. La implementación oficial de SASS es de código abierto y codificada en *Ruby*, sin embargo, existen otras implementaciones.

En la Tabla 2.4 se muestra una comparación de código escrito en CSS, SCSS y SASS.

**Tabla 2.4: Comparación entre líneas de código escritas en CSS, SCSS y SASS**

CSS	SCSS	SASS
<pre>.content-navigation {   border-color: #3bbfce;   color: #2b9eab; }  .border {   padding: 8px;   margin: 8px;   border-color: #3bbfce; }</pre>	<pre>\$blue: #3bbfce; \$margin: 16px;  .content-navigation {   border-color: \$blue;   color:     darken(\$blue, 9%); }  .border {   padding: \$margin / 2;   margin: \$margin / 2;   border-color: \$blue; }</pre>	<pre>\$blue: #3bbfce \$margin: 16px  .content-navigation   border-color: \$blue   color: darken(\$blue, 9%)  .border   padding: \$margin / 2   margin: \$margin / 2   border-color: \$blue</pre>

### **CoffeeScript**

“Se trata de un lenguaje de programación que traduce a *Javascript* y está diseñado para hacer más legible el código” [McCaw, A., 2012]. Esta característica permite que el lenguaje sea intuitivo para los desarrolladores: el código se puede expresar con mayor claridad, mayor concisión, o en un estilo alternativo que algunos programadores pueden preferir. *CoffeeScript* es un intento de exponer las partes

buenas de *Javascript* de una manera sencilla. Se trata de una alternativa para expresar y optimizar lo que ya existe.

El lenguaje está inspirado en *Ruby*, *Python* y *Haskell* para mejorar la brevedad de *Javascript* pues los programas pueden ser escritos con menos código (ver Tabla 2.5) sin afectar su desempeño, así como también incluye la adición de características más sofisticadas, como la comprensión de arreglos y reconocimiento de patrones.

**Tabla 2.5: Comparación entre líneas de código *Javascript* y líneas de código *CoffeeScript***

<i>Javascript</i>	<i>CoffeeScript</i>
<pre>var cubo, cuadrado; cuadrado = function(x) {   return x * x; }; cubo = function(x) {   return cuadrado (x) * x; };</pre>	<pre>cuadrado = (x) -&gt; x * x cubo    = (x) -&gt; square(x) * x</pre>

Según la *página* oficial de *CoffeeScript*, la regla de oro consiste en que: “es sólo *Javascript*”. El código es compilado línea por línea en su equivalente en *Javascript* y no existe una interpretación en tiempo de ejecución. Adicionalmente, puede hacer uso de cualquier biblioteca *Javascript* sin inconvenientes. El resultado compilado es legible y funciona en cualquier implementación que requiera código *Javascript*. Una ventaja es que dicho código puede llegar a ejecutarse más rápidamente que el código *Javascript* escrito de forma manual.

Uno de los aspectos relevantes de *CoffeeScript* es que la versión 3.1 de *Ruby on Rails*, incluye el mencionado lenguaje por defecto; esto supone un importante avance para el desarrollo de proyectos, ya que, antes había que *compilar* cada uno de los archivos en *CoffeeScript* que se iban a utilizar para poder incluirlos como *Javascript* dentro de un proyecto. Ahora, al estar integrado en el *framework*, programar en lenguaje *CoffeeScript* resulta tan directo como programar en *Javascript* nativo.

### 2.3.10. Manejador de versiones de *Ruby* (*RVM*)

El manejador de versiones de Ruby (*RVM* por sus siglas en inglés, Ruby Version Manager), es una herramienta que permite instalar, manejar y trabajar con múltiples entornos de trabajo o conjunto de gemas (*RubyGems*) de *Ruby* en cada una de las versiones instaladas [RVM, 2010].

La característica principal y más importante de *RVM* es el poder cambiar fácilmente entre las distintas versiones de *Ruby* instaladas; esto provee una herramienta útil al momento de trabajar con diferentes versiones de *Ruby* en un mismo entorno sin necesidad de configurar el ambiente de desarrollo de manera reiterada (ya sea instalándolo o desinstalándolo).

La posibilidad anteriormente mencionada acerca de la creación de grupos independientes de *gemas*, se denomina *gemsets*. Los *gemsets* son ideales para agrupar las gemas que se van a utilizar en cada proyecto o desarrollo, independizándolas del resto de los proyectos. Esto resulta útil si se necesitan dos (2) versiones de alguna(s) *gema(s)*, pero se debe conservar la misma versión de *Ruby*. Al usar *gemsets*, las modificaciones y actualizaciones de las *gemas* afectan sólo al *gemset* en uso, dejando los restantes intactos.

Otro aspecto relevante acerca de *RVM* se trata de su manejo, el cual es realizado a nivel usuario, por este motivo no es necesario ser un usuario privilegiado o un administrador del sistema para acceder a ciertos comandos que, en una instalación de *Ruby* estándar, requerirían serlo; por ejemplo, el comando para instalar una *gema*.

En la Tabla 2.6 se muestra un listado de los principales comandos provistos por *RVM* ejecutados en una terminal:

**Tabla 2.6: Código de principales comandos de *RVM***

```
$ rvm install 1.8.7 #Instala la versión 1.8.7 de ruby
$ rvm use system #
$ rvm use 1.8.7 #Selecciona la versión previamente instalada de ruby
$ rvm gemset create ejemplo #Crea un gemset llamado "ejemplo"
```

```
$ rvm gemset list #Indica sobre cuál gemset se está situado
$ rvm use 1.8.7@ejemplo #Selecciona el gemset "ejemplo" que será utilizado
#con la versión 1.8.7 de ruby
```

### 2.3.11. Sistema manejador de *bases de datos MySQL*

*MySQL* es un sistema de administración de *bases de datos* relacional que es usado para agregar, eliminar, modificar y consultar los datos del repositorio [MySQL, 2007]. Se encarga de almacenar y distribuir gran cantidad de datos para satisfacer las necesidades de cualquier organización. “*MySQL* incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de un usuario, administrar el sistema y proteger y hacer volcados de datos” [Gilfillan, I., 2003].

Una de las características de *MySQL* es que su arquitectura se encuentra basada en un esquema *cliente-servidor*, es decir, se puede utilizar del lado del cliente o del lado del *servidor*. Existe una *base de datos* del *servidor* (*MySQL*) y varios *clientes* (aplicaciones de programas), los cuales se comunican con el *servidor*. Los *clientes* pueden ejecutarse en la misma computadora.

*MySQL* utiliza *SQL* como lenguaje para realizar *consultas*. Un aspecto importante que debe destacarse es el hecho de que *MySQL* puede almacenar *Tablas* en una variedad de formatos. El formato por defecto es denominado *MyISAM*; este tipo de *Tabla* es estable y simple de manejar. El otro formato importante se llama *InnoDB*, el cual incluye ciertas características que no se encuentran presentes en el formato *MyISAM*; tales como ejecución de transacciones, bloqueo a nivel de fila (durante una transacción, la *Tabla* no se bloquea por completo), reconocimiento y detección de interbloqueos, entre otras. Tal y como recomienda Michael Kofler, las *Tablas* con formato *MyISAM* se pueden utilizar cuando se desee administrar las *Tablas* de la manera más eficiente posible en cuanto a tiempo y espacio; mientras que las *Tablas* con formato *InnoDB* se pueden utilizar cuando se desee implementar transacciones, se requiera de mayor seguridad o se provea el acceso a múltiples usuarios simultáneamente para realizar cambios.

### **Ventajas y desventajas de MySQL**

A continuación se presentan una serie de ventajas y desventajas. Previamente se han mencionado algunas de las características del *manejador*, las cuales pueden incluirse entre sus principales ventajas y desventajas.

#### **Ventajas:**

- Compatibilidad con el *lenguaje SQL*.
- Con el manejador *MySQL* se pueden crear *vistas*, *procedimientos*, *disparadores*, definir restricciones de integridad.
- Posee la alternativa de *replicación* de los datos para incrementar la protección de los mismos al ocurrir una falla del sistema y para proveer mayor eficiencia y rapidez al momento de realizar consultas sobre la *base de datos*.
- *Manejo de transacciones*. En el contexto de un sistema de *bases de datos*, una *transacción* significa la ejecución de varias operaciones de *bases de datos* como un bloque. El sistema de *base de datos* asegura que, o bien todas las operaciones se ejecutan correctamente o ninguna de ellas se ejecuta. Esto es válido incluso si en medio de una *transacción* ocurre una falla de electricidad, la computadora se bloquea, o algún otro desastre se produce. Las *transacciones* permiten al programador interrumpir comandos ejecutados previamente, lo cual simplifica muchas veces el proceso de programación [Kofler, 2005].
- Se pueden utilizar diversos lenguajes de programación para desarrollar aplicaciones de *MySQL*, tales como *C*, *C++*, *Java*, *RubyPerl*, *PHP*, *Python*, *Tcl*, etc.
- Es *multiplataforma*, es decir, puede ejecutarse en diversos sistemas operativos, tales como *Linux*, *Microsoft Windows*, *Apple Macintosh OS X* y algunas variantes de *Unix*.

**Desventajas:**

- Cuando se utilizan *Tablas* con formato tipo *MyISAM*, al momento de consultar o modificar la información de la *base de datos*, toda la *Tabla* donde se encuentra dicha información es bloqueada.
- Al utilizar *Tablas MyISAM*, *MySQL* no puede llevar a cabo copias de seguridad “*en caliente*” (las cuales son copias de la información durante la operación sin bloquear las *Tablas*).
- Muchos *manejadores de bases de datos* permiten definir tipos de datos por el usuario. *MySQL* no provee tal funcionalidad.

**2.3.12. Subversion**

*Subversion* (a menudo abreviado *SVN*) es un *software* de control de versiones y un sistema de control de revisiones distribuido bajo una licencia libre. Los desarrolladores usan *Subversion* para mantener las versiones actuales e históricas de los archivos como el *código fuente*, *páginas web*, y la documentación. Su objetivo consiste en ser un sucesor en su mayoría compatible con el sistema utilizado versiones concurrentes [Wikipedia, 2012].

*SVN* es usado para que varios desarrolladores puedan trabajar en un mismo proyecto en forma más ordenada. Posee una *arquitectura cliente-servidor* con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo y funciona de la siguiente manera: en un *servidor* se monta un repositorio *SVN*; en este lugar se van a registrar los cambios (revisiones) y los *logs* que se vayan generando. El *cliente* de *SVN* se baja una copia local de alguna revisión (generalmente la última), el desarrollador hace los cambios y los sube al *servidor* para que estén disponibles para los otros desarrolladores (además de generar un log con un comentario de archivo que modificó, para qué se modificó, etc.).

A lo largo del presente capítulo se han descrito el conjunto de herramientas y tecnologías que usadas de manera integrada para iniciar con la solución al problema

descrito en el Capítulo 1. A continuación se describirán las bases metodológicas que se acoplaron de forma ideal a la solución de la problemática planteada.

## 2.4. Método de desarrollo *AgilUS*

Para el desarrollo de la *aplicación web* se decidió usar el método de desarrollo *AgilUS*, el cual centra sus procesos en el usuario, brinda flexibilidad en la definición de los requerimientos del sistema y reduce los tiempos de desarrollo. Es por esta razón que *AgilUS* es el método que mejor se adecua al contexto en el cual se enmarca el presente trabajo.

Los cambios y adaptaciones del sistema se determinan en función de los requerimientos. Las funcionalidades del sistema se van añadiendo progresivamente, es por esta razón que el usuario no se tiene que adaptar al sistema, sino el sistema se va adaptando acorde a las necesidades del usuario.

Bajo un enfoque ágil, la usabilidad debe considerarse en cada etapa de desarrollo. "Son muchos los métodos propuestos para el desarrollo de *software* y son muchas las herramientas y técnicas con que cuentan los ingenieros de *software* para realizar su trabajo" [Acosta, E., 2010].

Los aspectos de la interfaz en una *aplicación* resultan un punto importante que debe ser tomado en consideración con la finalidad de lograr una experiencia agradable entre el usuario final y la *aplicación*. También deben considerarse los tiempos de desarrollo, buscando reducirlos para obtener satisfacción por parte del usuario y una mayor productividad por parte del desarrollador.

Con el método de desarrollo *AgilUS* se persigue abarcar aspectos de usabilidad incorporando una visión que complementa el área de Interacción Humano-Computador con la Ingeniería de *Software*, dando lugar a un conjunto de buenas prácticas y evaluaciones aplicadas al *software*. "Se plantea el desarrollo de un sistema partiendo de un prototipo de la *interfaz de usuario*, y que incorpora la *aplicación* de



diversas técnicas de evaluación de usabilidad desde el inicio del ciclo de vida de la aplicación" [Acosta, E., 2010].

Para alcanzar la usabilidad, en cada etapa de desarrollo se considera como eje principal las características y actividades comúnmente realizadas por el usuario, aplicando técnicas de diseño centradas en el mismo; de esta manera se determinan las metas que contemplan cada una de las etapas, así como los objetivos que debe satisfacer el producto final.

La alternativa de *AgilUS* como método de desarrollo se trata de una propuesta llevada a cabo por el Centro de Ingeniería de Software y Sistemas (ISYS) de la Escuela de Computación, Universidad Central de Venezuela, producto de una investigación basada en el análisis de aspectos de usabilidad en un sistema. El objetivo que persigue dicho método es el de "... evolucionar el software, a fin de que éste alcance el mayor grado de usabilidad una vez culminado su desarrollo" [Acosta, E., 2010].

*AgilUS* se trata de un método de desarrollo basado en iteraciones, en donde cada iteración abarca diferentes actividades que se ejecutan de manera incremental, otorgando prioridad a los aspectos de diseño y desarrollo en términos de usabilidad. El aspecto estético de la interfaz no se deja como tarea final, sino se toma en cuenta a lo largo del desarrollo de las etapas. La evaluación y análisis de los requisitos y la elaboración de prototipos, son actividades contempladas en este método.

El hecho de considerar la usabilidad como uno de los principios de *AgilUS* se debe a que un *software* usable es considerado útil, ya que la usabilidad y la utilidad de un *software* van de la mano. Un usuario que interactúa con el sistema, debe hacerlo de forma intuitiva y satisfactoria, sólo así se determina la utilidad y usabilidad del sistema.

### **2.4.1. Ciclo de vida**

Como se ha mencionado anteriormente, en *AgilUS*, la usabilidad es un aspecto que debe incluirse en cada etapa de desarrollo; para ello se toma en consideración la evaluación que realiza el usuario luego de interactuar con el sistema. En cada etapa, se

desarrollan diferentes prototipos que pueden mejorarse en base a las observaciones realizadas.

Cada etapa del ciclo de desarrollo se lleva a cabo de manera incremental e iterativa mediante la construcción de prototipos hasta obtener un producto final para la entrega, sin dejar de lado la usabilidad en cada una de las actividades desarrolladas.

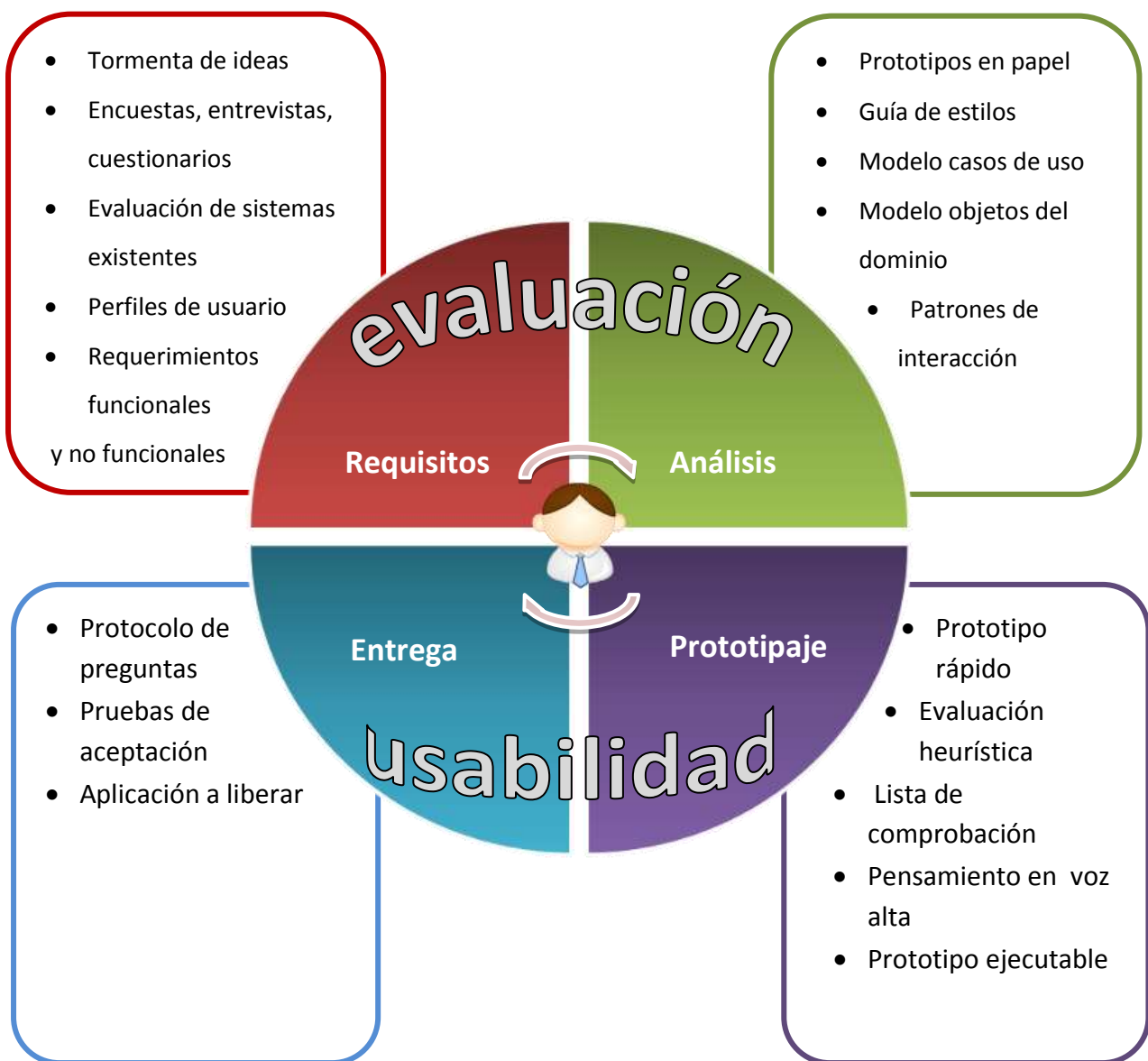


Figura 2.10 Etapas de desarrollo y artefactos de *AgilUS*

Como se aprecia en la Figura 2.10, el ciclo de vida de *AgilUS* comprende la especificación de requisitos, el análisis de dichos requisitos, la elaboración de prototipos y la entrega del producto final. Cada una de las etapas incluye la construcción de diversos artefactos, que pueden ser sujetos a modificaciones hasta obtener el producto final. A continuación se realiza una descripción de cada etapa:

- *Requisitos*: se lleva a cabo el *levantamiento* de requerimientos, generando un perfil de usuario. Cada requerimiento definido se debe desarrollar e implementar para cumplir con las exigencias del usuario.
- *Análisis*: una vez estipulados los requisitos que persigue el sistema, se debe realizar un análisis sobre los mismos, para determinar de qué manera serán implementados, para lo cual se toma como apoyo los diferentes diagramas realizados bajo la notación estándar del *Lenguaje de Modelado Unificado* ó *UML* (por sus siglas en inglés: *Unified Modeling Language*).
- *Prototipaje*: son elaborados diversos prototipos, incluyendo un prototipo de *Interfaz Gráfica de Usuario* ó *GUI* (por sus siglas en inglés: *Graphical User Interface*), facilitado por los diagramas de interacción. En esta etapa, el usuario lleva a cabo la evaluación de los prototipos a fin de conocer si los mismos cumplen con los principios de usabilidad. Para realizar estas inspecciones de usabilidad, se lleva a cabo la elaboración de listas de comprobación, las cuales se utilizan para juzgar los atributos y los métodos de interacción de la *interfaz de usuario*; también se elaboran guías de estilo, con la cuales se determinan las normativas en relación al aspecto de la *interfaz*; por último se lleva a cabo una *evaluación heurística*, que consiste en un análisis de los aspectos de usabilidad para determinar si se encuentran incluidos en cada elemento de la *interfaz de usuario*.
- *Entrega*: consiste en la *aplicación* de pruebas finales para confirmar que la *aplicación* cumple con los principios de usabilidad. Posteriormente, la *aplicación* es pasada a producción.

Al considerar los principios de *AgilUS* y efectuar rigurosamente cada fase del ciclo de vida, se podrá obtener como resultado un producto de calidad cuyo

componente principal será la usabilidad. Es importante tener presente durante todo el proceso de desarrollo, que es el usuario y no el cliente o el equipo desarrollador, el protagonista y juez que dictamina si los aspectos de usabilidad son cubiertos por el sistema.

Debido a la mínima cantidad de generación de artefactos, los tiempos de desarrollo disminuyen y resulta más fácil incorporar cambios que son considerados necesarios y que van surgiendo a lo largo del ciclo de vida. Mediante el diseño de prototipos que evolucionan en función de las necesidades del usuario, el sistema va estructurándose hasta llegar a convertirse en el producto final.

### **2.4.2. Características de *AgilUS***

El método de desarrollo *AgilUS* está orientado al cumplimiento de los requerimientos básicos estipulados por el usuario, avanzando incrementalmente en cada una de las etapas y tomando en cuenta los aspectos de usabilidad. Entre las características, bondades y buenas prácticas estipuladas en el método, se encuentran:

- Diseño centrado en el usuario (*DCU*), basado en las preferencias e inclinaciones que presenta el usuario frente al sistema que desea. “... en el *DCU* se procura construir el sistema para adaptarse, a través de su interfaz, a cómo el usuario desea trabajar, en lugar de forzar al usuario a cambiar su modo de trabajar para adaptarse a lo que los desarrolladores consideraron apropiado” [Acosta, E., 2010]. Para lograr este diseño, el desarrollador debe realizar varias pruebas con el sistema para determinar si poco a poco se van satisfaciendo las necesidades del usuario.
- Diseño basado en prototipos, donde cada uno es revisado y probado por usuarios finales y especialistas en el área de usabilidad. La construcción de cada prototipo viene dada en función de la respuesta que emiten los usuarios y especialistas ante dicho prototipo, permitiendo de esta manera poder seguir avanzando con la construcción del siguiente prototipo. Este proceso se lleva a cabo de manera cíclica hasta culminar la elaboración del producto final.

- Simplicidad en el desarrollo del sistema, previendo posibles cambios en el futuro.
- Desarrollo incremental, de manera de establecer y cumplir requisitos a medida que se desenvuelve el proyecto.
- Desarrollo iterativo, permitiendo el rediseño e implementación de actividades llevadas a cabo en cada una de las etapas. El objetivo al diseñar e implementar en cada iteración consiste en realizarlo de manera simple para no dejar de un lado la posibilidad de rediseñar. Para determinar si debe llevarse a cabo un rediseño, el usuario evalúa el producto y expone su opinión al desarrollador, quien toma en cuenta dicha opinión conjuntamente con el análisis de las funcionalidades del sistema.
- Usabilidad como requisito esencial para considerar un *software* de calidad.
- Interacción entre el equipo desarrollador y el usuario, un aspecto vital si se considera la usabilidad en cada etapa, ya que es el usuario quien determina la misma.

### 2.4.3. Entregables de *AgilUS*

A continuación se describen los artefactos utilizados durante el desarrollo del trabajo especial de grado. Los mismos involucran las actividades en la utilización del método de desarrollo *AgilUS* para el desarrollo de las funcionalidades de gestión, planificación y reportes incorporados en la *aplicación web*.

- **Tormenta de ideas**

La tormenta de ideas fue el primer método creativo propuesto para los grupos. Fue formulada por Alex Osborn y se trata de la práctica de una técnica de conferencia en la que un grupo de personas busca la solución a un problema específico, juntando todas las ideas aportadas en forma espontánea por sus integrantes.

Es útil para atacar problemas específicos, recolectar buenas ideas y donde no hace falta analizar profundamente las decisiones que se toman.

- **Entrevista**

Se trata de una técnica para recopilar información, la cual permite establecer un diálogo a través de un esquema dado, persiguiendo un propósito profesional.

Ocurre entre dos o más personas, donde uno es el entrevistador y otro u otros son los entrevistados. Es uno de los procedimientos más usados en la investigación social, aunque como técnica profesional se usa en otras tareas. Cada uno lo usa de acuerdo a los fines que persiga con su actividad [Egg A, 1979].

- **Perfil de usuario**

El perfil de usuario define un conjunto de rasgos característicos que sirven como fundamento para la planificación y/o realización de cualquier servicio, sistema de información o módulo.

Los perfiles de usuarios constituyen un elemento básico para el desarrollo de sistemas de información y pueden ser obtenidos siguiendo un proceso denominado *Estudios de Usuario*, el cual consiste en realizar análisis tanto en el comportamiento de los usuarios como en las tareas que los mismos desempeñan y sus responsabilidades [Hernández, P].

- **Requerimientos funcionales y no funcionales**

Los requerimientos son una descripción de las necesidades o deseos de un producto. La meta principal al aplicar esta actividad es identificar y documentar lo que en realidad se necesita, en una forma en que pueda fácilmente ser transmitido al usuario y al equipo de desarrollo.

Los requerimientos funcionales corresponden a las peticiones que tienen los usuarios para cumplir con sus necesidades. Expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento). Además describen la funcionalidad o los servicios que se espera que el sistema provea [Soto, L].

Los requerimientos no funcionales corresponden a los atributos de calidad del *software*, los cuales involucran aspectos como la seguridad, eficiencia, usabilidad, portabilidad, interoperabilidad, escalabilidad, entre otros. “... no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento” [Soto, L]. No son solicitados explícitamente por los clientes o usuarios del sistema.

- **Casos de uso**

“Un modelo de caso de uso describe la secuencia de las interacciones que se desarrollarán entre los actores y el sistema, en respuesta a un evento que inicia un actor” [Ingeniero de Software, 2008].

Los Casos de Uso son la base para el trabajo del equipo de desarrollo y permiten definir:

- Arquitectura
- Diseño
- Pruebas funcionales
- Diseño de *interfaz de usuario*

Los casos de uso vienen acompañados de un diagrama empleado para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. Hace uso del *Lenguaje de Modelado Unificado* (Unified Modeling Language, en inglés) utilizado para especificar, visualizar, construir y documentar los artefactos de sistemas de *software*, así como modelar

lógica de negocios. Permite comunicar claramente requerimientos, arquitecturas y diseños de manera estándar de la industria [Ingeniero de software, 2008].

- **Modelo de objetos del dominio**

Un modelo de objetos del dominio describe los objetos que se identifican en el dominio de la *aplicación* y las relaciones con su respectiva cardinalidad. Este modelo se expresa mediante un diagrama de clases del *Lenguaje de Modelado Unificado*. Un diagrama de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras [Ingeniero de software, 2008].

- **Guía de estilo**

Se define como guía de estilo al documento que recoge normativas y patrones básicos relacionados con el aspecto de una *interfaz* para su *aplicación* en el desarrollo de nuevas pantallas dentro de un entorno concreto.

- **Evaluación heurística**

La evaluación heurística es un método de inspección cuyo objetivo es encontrar problemas de usabilidad en el diseño de la *interfaz de usuario*, tal que puedan ser atendidos como parte de un proceso de diseño iterativo. Puede ser realizada por personal no especializado e incluso por usuarios tipo.

Se pueden tomar como criterios de evaluación las siguientes heurísticas [Nielsen, 1994]:

- H1: Diálogo natural y simple
- H2: Hablar el lenguaje del usuario
- H3: Minimizar la carga cognitiva
- H4: Consistencia
- H5: *Feedback* (retroalimentación)



- H6: Proveer claramente las salidas
- H7: Proveer *Shortcuts* (atajos de teclado)
- H8: Mensajes de error descriptivos
- H9: Prevención de errores
- H10: Asistencia al usuario

Es necesario determinar una escala para valorar los problemas detectados al realizar la evaluación heurística, tomando en cuenta la frecuencia con la que ocurre el problema, el impacto que conlleva y la persistencia.

---

## Capítulo 3

### Marco Aplicativo

---

Una vez que se describen las diferentes tecnologías a emplear y se realizan las especificaciones sobre *AgilUS*, el cual es un método de desarrollo iterativo e incremental, se planifican y se llevan a cabo una serie de iteraciones englobadas en tres (3) partes: gestión, planificación y reportes.

Durante las iteraciones mencionadas, se realizan diversas actividades contempladas en las etapas de *AgilUS*, que sirven como aporte para la incorporación de nuevas funcionalidades en *ACEIM*.

A continuación se describen cada una de las iteraciones realizadas y se especifican las diferentes actividades generadas en las etapas contempladas en el método de desarrollo *AgilUS*.

#### 3.1. Gestión

En la presente iteración se llevó a cabo el desarrollo de las funcionalidades de gestión, en donde se definieron funcionalidades para la administración de los diversos recursos que conforman la realidad de los cursos de extensión, tales como: estudiantes, instructores, secciones, aulas, entre otros.

Las funcionalidades de gestión incluyen la edición del *contenido web* de la *página* de inicio de *ACEIM*, la modificación de diversas configuraciones generales como

la fecha de las inscripciones o la modificación del período actual en el que se encuentran los cursos, la modificación del costo de la matrícula para el ingreso general, entre otras acciones.

### **3.1.1. Requisitos**

Constituye la primera etapa del método, en la cual se establecen los requisitos en base a las necesidades del usuario. En esta etapa se aplicaron diversas técnicas de indagación para obtener la información necesaria para un posterior análisis de requerimientos planteados por el personal administrativo de los cursos para el desarrollo de las funcionalidades de gestión. Las técnicas empleadas fueron: tormenta de ideas, perfil de usuario y listado de requerimientos funcionales y no funcionales.

#### **3.1.1.1. Tormenta de ideas**

Esta actividad se llevó a cabo en reuniones con el grupo de desarrollo, el personal administrativo y el coordinador de los cursos de *FUNDEIM*, así como en la asistencia a las jornadas de inscripción, en donde se recolectaron conocimientos acerca de la problemática existente. Entre las ideas que surgieron en estos encuentros se destacan las siguientes:

- Se desea contar con una funcionalidad para crear, modificar y eliminar una sección para un curso específico.
- Incorporar una funcionalidad para congelar el cupo de los estudiantes, que se refiere a la activación o desactivación de un estudiante para un curso dado. Un estudiante puede presentar diversas razones para congelar un período, es decir, para no cursarlo durante actualmente sino reincorporarse el siguiente.
- Permitir establecer los precios de la matrícula y el porcentaje de descuento de los convenios.
- Permitir la creación de un nuevo convenio.

### 3.1.1.2. Perfil de usuario

Dentro de la iteración de gestión existen diversos tipos de usuarios, los cuales asumen un rol dependiendo de las responsabilidades que desempeñan dentro de la Fundación. Los involucrados en el proceso de gestión son los empleados administrativos de los cursos y la persona coordinadora de los mismos.

A continuación se describen cada uno de los roles que pueden ser apreciados en la Tabla 3.1 y Tabla 3.2.

**Tabla 3.1: Perfil del usuario super administrador de gestión**

Representante	Súper Administrador
<b>Habilidades</b>	Este tipo de usuario posee destrezas desarrolladas en el uso del computador, con un nivel intermedio en el uso de la <i>aplicación web</i> ya existente. Es el encargado de coordinar los cursos de extensión y es un profesor de la Escuela de Idiomas Modernos de la Universidad Central de Venezuela
<b>Responsabilidades</b>	<ul style="list-style-type: none"> <li>• Gestionar estudiantes (nuevos, regulares), instructores, aulas, convenios, cursos</li> <li>• Definir el período actual, el período de calificación, el período de inscripción, apertura de la inscripción para alumnos regulares, apertura de inscripción para alumnos nuevos, establecer si se permite el cambio de horario en la inscripción, si la calificación se encuentra abierta, si los listados se encuentran disponibles</li> <li>• Configurar los convenios: definir el precio de la planilla y el precio base de los convenios, crear convenios, cambiar el porcentaje de descuento de cada convenio</li> <li>• Modificar el contenido <i>web</i> de la <i>página</i> principal de los cursos, en el cual se exponen las fechas y datos importantes sobre inscripciones, entre otros</li> </ul>
<b>Grado de participación</b>	Alto

**Tabla 3.2: Perfil del usuario Administrador de gestión**

Representante	Administrador
Habilidades	Este tipo de usuario posee un conocimiento básico sobre el funcionamiento de un computador y sobre el uso de la <i>aplicación web</i> ya existente. Se trata del personal administrativo de los cursos de extensión
Responsabilidades	Gestionar estudiantes (nuevos, regulares), instructores, aulas, cursos, entre otros recursos
Grado de participación	Alto

### 3.1.1.3. Requerimientos funcionales y no funcionales

En base a la observación realizada durante las jornadas de inscripción y las reuniones con el personal administrativo y el encargado de gestionar los cursos de extensión, se recopilaron diversos requisitos considerados al momento del desarrollo de las funcionalidades de gestión en la *aplicación web*. A continuación se detallan cada uno de estos requisitos.

- **Requerimientos funcionales:**

- Manejo de operaciones de creación, consulta, actualización y eliminación sobre todos los recursos de *FUNDEIM*: se debe proveer una *interfaz de usuario* que permita hacer uso de las operaciones *CRUD* (Create, Read, Update, Delete, por sus siglas en inglés). El acceso a algunos de los recursos del sistema como los alumnos, instructores y precios de matrícula puede resultar limitado dependiendo de las permisologías otorgadas a los diferentes roles de usuario.

- Edición y creación de noticias e información: se requiere una funcionalidad que permita la creación y edición de noticias e información en la *página* de inicio de la *aplicación web*. A través de esta funcionalidad, los administradores pueden colocar y editar información de interés, noticias y fechas que deben ser de conocimiento general para los usuarios.
- Congelar cupo del estudiante: se requiere implementar una funcionalidad que permita asignar a un estudiante un status que indique si se encuentra congelado para un curso en un período determinado.
- Modificar el precio de la matrícula para el ingreso general y el porcentaje de descuento de los convenios: se debe proveer la posibilidad de modificar el precio base que cancela un estudiante trimestralmente al ingresar a un curso. También se debe incorporar una funcionalidad para cambiar el porcentaje de descuento de un convenio existente.
- Modificar las configuraciones generales de los cursos: es importante permitirle al usuario administrador definir el período actual, el período de calificación, el período de inscripción, si se encuentra abierta la inscripción para alumnos regulares y la inscripción para alumnos nuevos, si se permite el cambio de horario en la inscripción, si la calificación se encuentra abierta y si los listados de estudiantes se encuentran disponibles.

- **Requerimientos no funcionales:**

A continuación se exponen cada uno de dichos requerimientos.

- *Usabilidad y navegabilidad*: el sistema debe ser de fácil uso para cualquier usuario que visite la *página*, ofreciendo funcionalidades que requieran un bajo conocimiento cognitivo. La solución propuesta debe tener interfaces gráficas de administración y de operación en idioma español y en ambiente *web*, para

permitir su utilización a través de cualquiera de los *exploradores* o *navegadores* de Internet.

- Portabilidad: el desarrollo bajo un enfoque *web* garantiza que las funcionalidades puedan ser accedidas desde diferentes plataformas, ya que sólo se requiere de la disposición de un *navegador web* o *browser* independientemente del sistema operativo.
- Disponibilidad: al tratarse de una *aplicación web*, la característica de disponibilidad se encuentra implícita; las funcionalidades deben estar disponibles y ser accesibles desde cualquier ubicación geográfica con conexión a Internet sin restricciones de horarios.
- Escalabilidad: las funcionalidades son implementadas previendo futuras mejoras sobre las mismas, de manera tal que pueden ser incorporados afectando el código existente de la menor manera posible. El desarrollo bajo el *framework* de aplicaciones web *Ruby on Rails* respalda el cumplimiento de este requerimiento, ya que sigue el paradigma de la arquitectura *MVC* explicado en el Capítulo 2.

Los requerimientos no funcionales mencionados anteriormente, también se aplicaron para las siguientes dos (2) iteraciones descritas en el presente capítulo.

### **3.1.2. Análisis**

Constituye la segunda etapa del método empleado. En esta etapa se definieron los casos de uso de gestión, el modelo de objetos del dominio, prototipos en papel y guías de estilo.

### 3.1.2.1. Casos de uso

A continuación se muestra el diagrama de casos de uso de gestión (ver Figura 3.1).

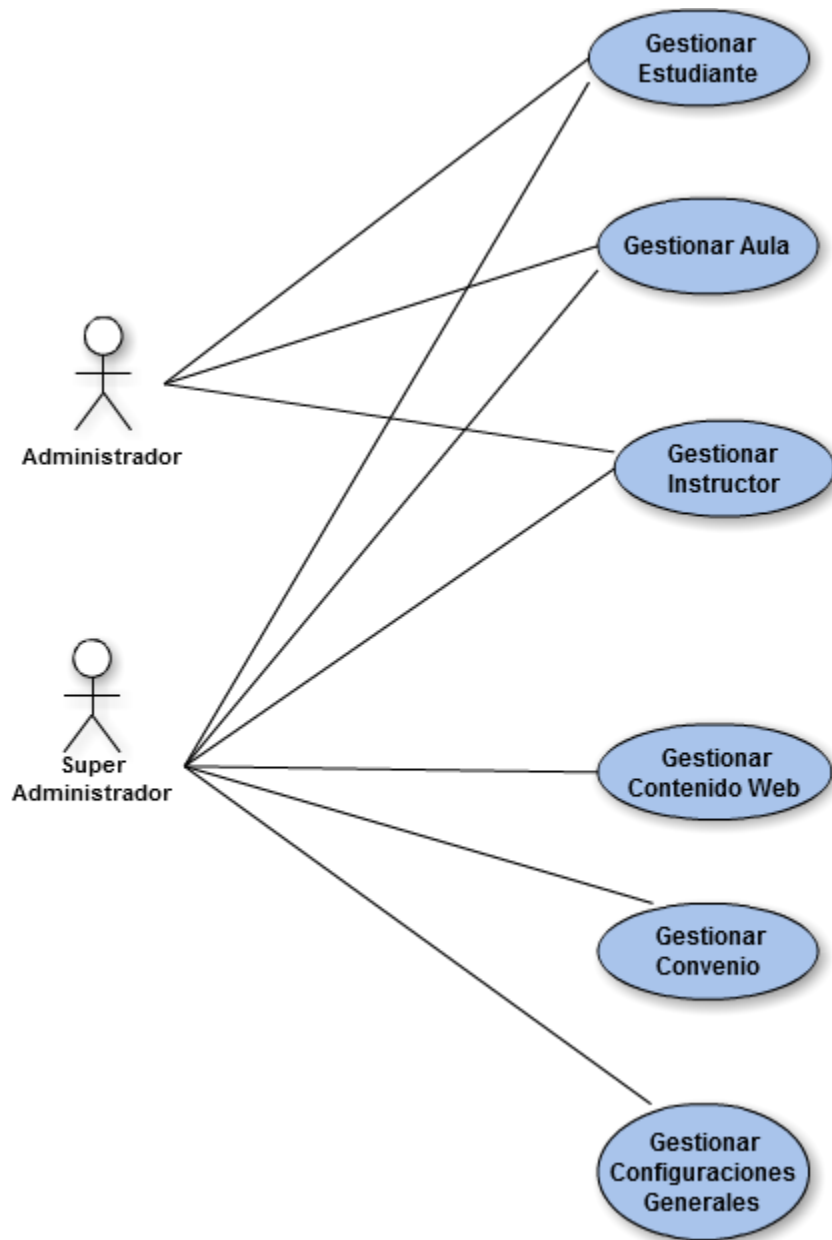


Figura 3.1: Diagrama de casos de uso de gestión



- **Especificación de los Casos de Uso:**

A continuación se expone una breve descripción sobre los casos de uso de la Figura 3.1:

- **Gestionar Estudiante:** permite al usuario personal administrativo agregar un nuevo estudiante, asignar o eliminar un curso asociado, cambiar la sección, congelar al estudiante por ese periodo, asignar o revocar un convenio de dicho estudiante, modificar sus datos personales, modificar su nota, entre las funcionalidades más relevantes.
- **Gestionar Aula:** permite al usuario personal administrativo agregar una nueva aula o modificar una ya existente.
- **Gestionar Instructor:** permite al usuario personal administrativo crear un nuevo instructor o modificar uno ya existente.
- **Gestionar Contenido Web:** permite al usuario super administrador visualizar o modificar el contenido de la *página principal* de la *aplicación web*.
- **Gestionar Convenio:** permite al usuario super administrador modificar el monto de la matrícula para el ingreso en los cursos, modificar el porcentaje de descuento de los convenios existentes o añadir un nuevo convenio.
- **Gestionar Configuraciones Generales:** permite al usuario super administrador definir el período actual, el período de calificación, el período de inscripción, activar la inscripción para alumnos regulares y para alumnos nuevos, permitir o denegar el cambio de horario en la inscripción, establecer si la calificación se encuentra abierta y si los listados de estudiantes se encuentran disponibles.

### 3.1.2.2. Modelo de objetos del dominio

A través de este modelo se puede definir el dominio que compone la gestión, a partir de los objetos que lo conforman y sus relaciones. Los objetos presentes en el modelo de objetos de dominio de gestión son los siguientes:

- **Usuario Administrador:** representa los usuarios del sistema, los cuales pueden ser administradores o super administradores.

- **Instructor:** modelo que representa los educadores que imparten las clases en los cursos.
- **Estudiante:** identifica a los usuarios que asisten a los cursos de idiomas.
- **Curso:** modela la asociación entre una sección, un idioma, una categoría y un horario.
- **Sección:** modelo que representa las distintas secciones existentes.
- **Aula:** representa las aulas donde se pueden dictar los cursos.
- **Horario:** contempla los distintos horarios en los que se dictan los cursos.
- **Idioma:** representa los distintos idiomas ofertados; los mismos pueden ser: inglés, alemán, italiano, francés y portugués.
- **Categoría:** modela las distintas categorías que se manejan en los cursos; las mismas pueden: ser niños, adolescentes y adultos.
- **Nivel:** constituyen los distintos niveles de los cursos.
- **Convenio:** representa los diferentes convenios y/o acuerdos que otorga la Fundación.
- **Contenido Web:** constituye el contenido de la *página* inicial de la *aplicación web ACEIM*.
- **Configuración General:** representa los parámetros de configuración general del sistema.

Entre estos objetos, se establecen diversas relaciones:

Los usuarios administradores gestionan instructores, estudiantes, cursos, secciones y aulas. Los instructores califican a los estudiantes y dictan cursos. Los estudiantes poseen un convenio e inscriben uno o varios cursos. Estos cursos tienen contenido acorde a una categoría e incluyen un idioma. Un curso pertenece a varias secciones. Los idiomas tienen cuatro (4) o más niveles. Las secciones están asociadas a un horario y disponen de una o dos aulas. Un aula está disponible en varios horarios. Un super administrador gestiona convenios, el contenido *web* y las configuraciones generales presentes en los cursos.

En la Figura 3.2 se muestra el modelo de objeto del dominio de gestión, con los objetos y las relaciones entre ellos.

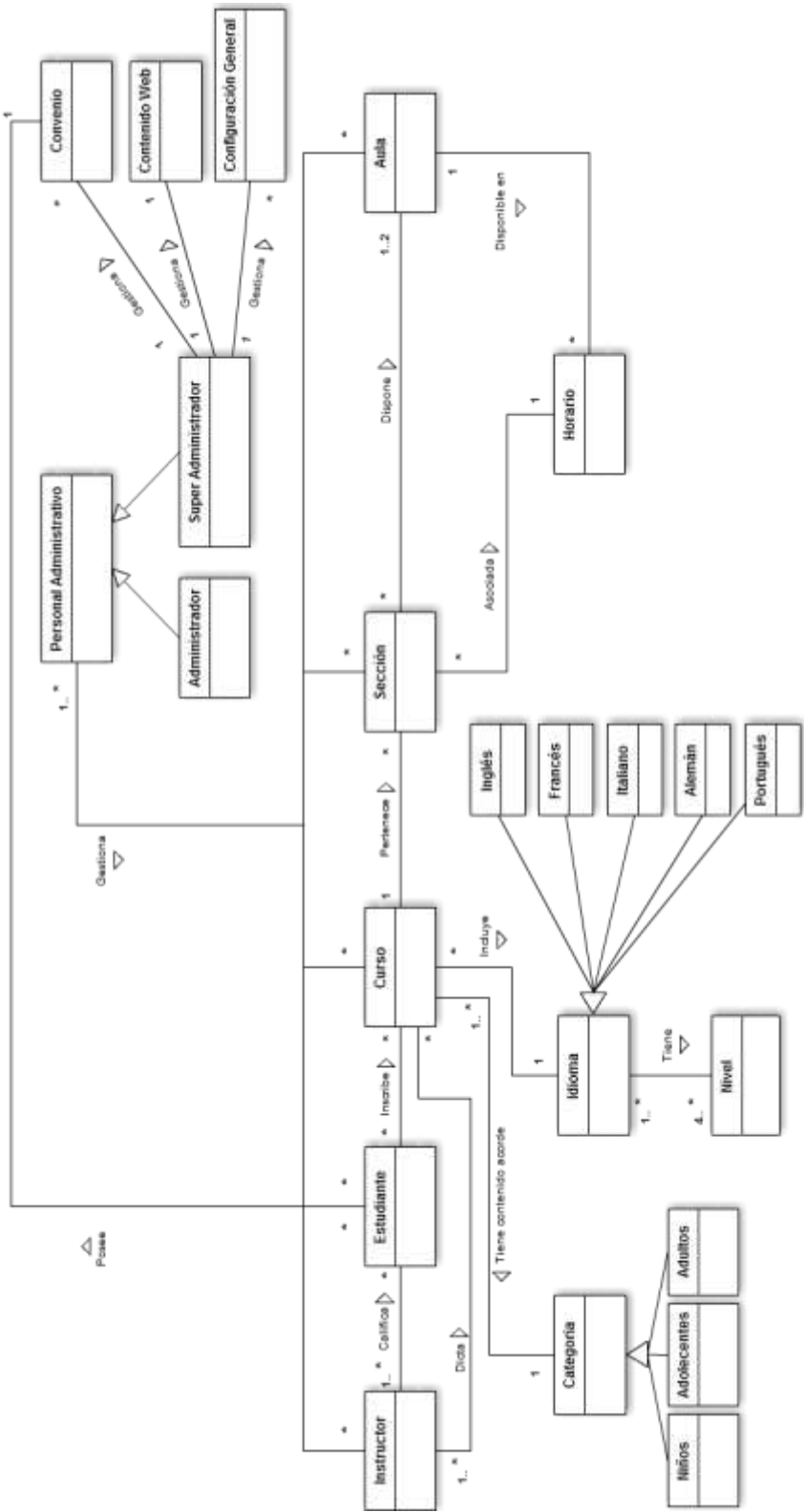


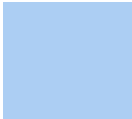

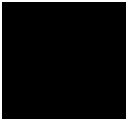

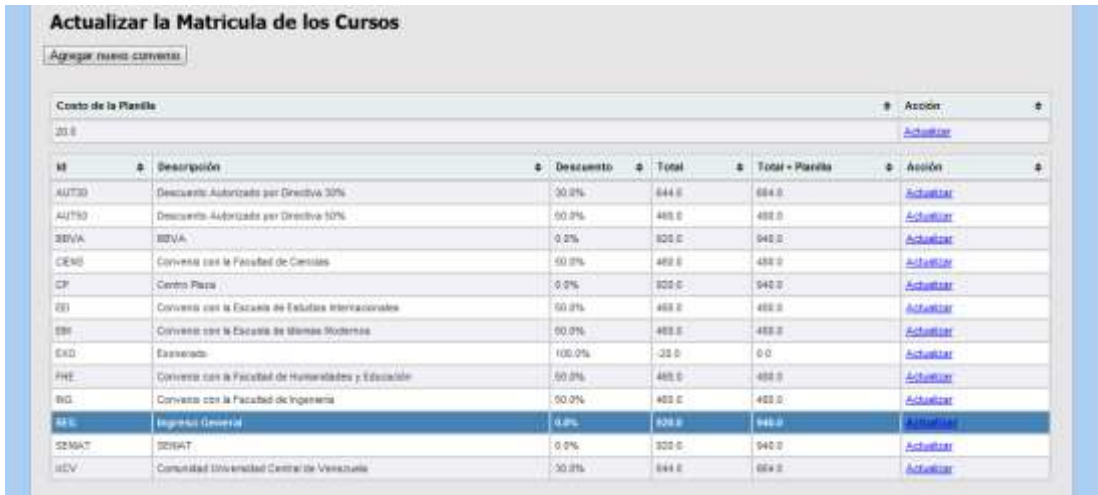


Figura 3.2: Modelo de objetos del dominio de gestión

### 3.1.2.3. Guía de estilos

Como una actividad contemplada en *AgilUS* se propone la guía de estilos. A continuación se muestra la utilizada en la *aplicación web* (ver Tabla 3.3).

Tabla 3.3: Guía de estilo de *ACEIM*

<b>Logo</b>				
				
<b>Colores Principales</b>				
				
RGB(0, 0, 238)	RGB(172, 206, 243)	RGB(255, 228, 92)	RGB(0, 0, 0)	RGB(187, 187, 187)
#0000EE	#ACCEF3	#FFE45C	#000000	#BBBBBB
<b>Tipografía</b>				
Tahoma 12 px - Tahoma 12 px				
<b>Navegación</b>				
				
<b>Estilo de la interfaz para funcionalidades de gestión</b>				

### 3.1.3. Prototipaje

Las nuevas funcionalidades incorporadas en la *aplicación web ACEIM* fueron desarrolladas bajo un enfoque *web cliente-servidor*, y bajo la plataforma del *Sistema Operativo Linux Ubuntu*. El Sistema Manejador de Bases de Datos usado fue *Mysql*, y se utilizó el sistema de control de versiones *Subversion*. El lenguaje de desarrollo de la *aplicación* fue *Ruby* y se usó bajo su *framework Rails*, en su última versión; otro detalle a destacar es que se usó un gestor de versiones del lenguaje *Ruby* denominado *RVM*.

En cuanto a la programación se utilizaron los lenguajes *HAML*, *CoffeeScript*, y *SASS*, además de algunas bibliotecas de desarrollo como *jQuery*. Asimismo se hizo uso de algunos *plugins* para la generación de documentos en PDF y en *Excel*, el *plugin tablesorter* para la visualización ordenada de *tablas* con información y *eIRTE* para la edición del contenido *web* principal de la aplicación.

En la Figura 3.3 se puede visualizar conjunto de herramientas y tecnologías usadas en el proyecto.

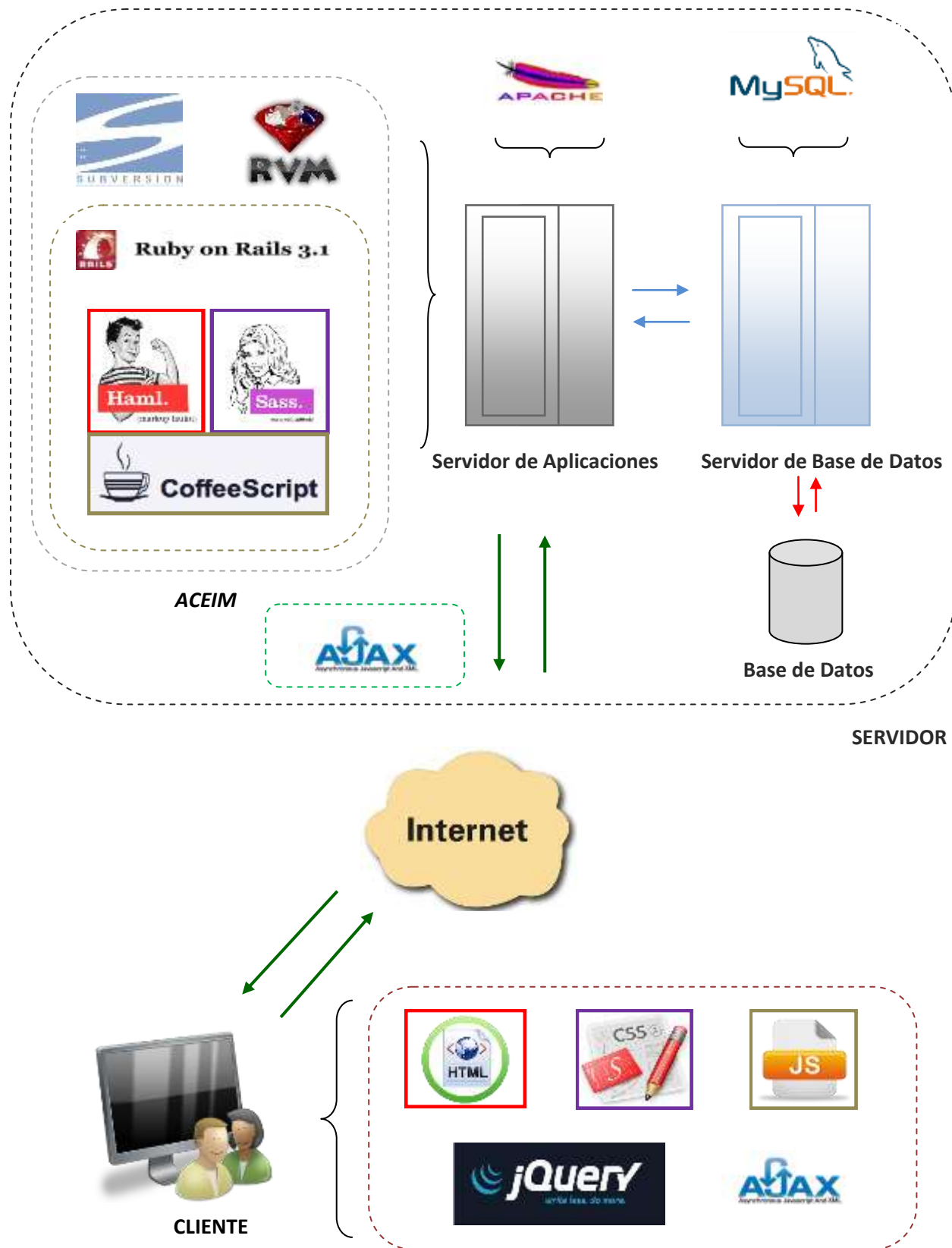


Figura 3.3: Tecnologías empleadas

### 3.1.3.1. Evaluación heurística

La evaluación heurística fue realizada por un experto del dominio en compañía de un experto en *Interacción Humano-Computador* y juntos formularon los diferentes problemas de usabilidad presentes en las interfaces especificadas más adelante.

Fueron utilizadas las heurísticas propuestas por Nielsen (ver Capítulo 2). La escala de evaluación está comprendida entre 0-4 y contempla los siguientes valores:

- 0: No es un problema de usabilidad
- 1: Problema cosmético
- 2: Problema menor
- 3: Problema grave de usabilidad, importante fijar solución
- 4: Usabilidad catastrófica, imperativo fijar solución

El primer escenario para la realización de la evaluación heurística consistió en asignar al experto en el dominio la tarea de congelar un estudiante en específico. La interfaz mostrada para la evaluación se encuentra a continuación (ver Figura 3.4):

**Historial Académico**

Idioma	Nivel	Horario	Período	Convenio	Calificación
Alemán (Adultos)	Intermedio I	Sábado (08:30AM - 01:15PM)	A-2012	Convenio con la Facultad de Ciencias	00 CALIFICAR
Alemán (Adultos)	Conversación Básica	Sábado (08:30AM - 01:15PM)	D-2011	Convenio con la Facultad de Ciencias	17 <a href="#">Modificar</a>
Alemán (Adultos)	Básico I	Sábado (08:30AM - 01:15PM)	C-2011	Convenio con la Facultad de Ciencias	16 <a href="#">Modificar</a>
Alemán (Adultos)	Básico I	Sábado (08:30AM - 01:15PM)	B-2011	Ingreso General	17 <a href="#">Modificar</a>

**Estado de los Cursos en el Período Actual**

Idioma	Nivel	Sección	Aula	Horarios	Instructor	Convenio	Estado	Opciones
Alemán (Adultos)	Intermedio I	01	Sábado Aula 54 - Ciencias	Sábado (08:30AM - 01:15PM)	Miguel Andrés (041009707)	Convenio con la Facultad de Ciencias	Inscritos Retirados	<a href="#">Congelar</a> <a href="#">Modificar_seccion</a> <a href="#">Eliminar</a> <a href="#">Reactivar</a> <a href="#">Generar_asistencia_estado</a> <a href="#">Cancelar</a>

**Figura 3.4: Interfaz a través de la cual se encuentra la opción de congelar a un estudiante en el período actual**

Seguidamente tuvo lugar la segunda evaluación realizada, que consistió en colocar al experto del dominio frente al listado de los alumnos que poseen algún convenio. La interfaz visualizada para realizar la evaluación heurística se aprecia en la Figura 3.5.

Convenio con la Facultad de Ciencias

Generar PDF | Generar Excel

#	Nombres	Cédula	Correo Electrónico	Teléfono Móvil	Idioma	Nivel
1	Fernandez Pereira Vanessa Sofia	20084400	vansofiaedc@gmail.com	04126764904	Alemán (Adultos)	Intermedio I
2	Carroste Garcia Rafael Adolfo	25127003	rocaroste@gmail.com	04142474600	Francés (Adultos)	Básico I
3	Desarlagin Gonzalez Anais Maria	17964185	anais.desarlagin@gmail.com	04160278435	Francés (Adultos)	Básico I
4	Monares Mesa Erick Alexander	18267826	erick.monares79@gmail.com	04196206790	Francés (Adultos)	Básico I
5	Mullaz Pastan Maria Fernanda	18736814	mullaz_julien@hotmail.com	04142674344	Francés (Adultos)	Básico I
6	Berrotarán Carlos	17312984	carlos.berrotaran@gmail.com	04122878366	Francés (Adultos)	Básico II
7	Costara Manuel	4366981	manuel.costara@ucv.ve	04142475836	Francés (Adultos)	Básico II
8	Concepcion Perez Maria Jose	16496190	marjosep_95@hotmail.com	04263950821	Francés (Adultos)	Básico II
9	Oliz Sonia	16761047	so_oliz@hotmail.com	04164157121	Francés (Adultos)	Básico II
10	Issaeri Adriana	14667501	adriana@gmail.com	04141719688	Francés (Adultos)	Básico II
11	Martinez Belloni Gustavo Adolfo	19886790	martinezbelloni@gmail.com	04125677543	Francés (Adultos)	Básico II

Figura 3.5: Interfaz para visualizar el listado de alumnos con convenio

El formato utilizado para la planilla de evaluación heurística para las funcionalidades de gestión es el siguiente (ver Tabla 3.4):

Tabla 3.4: Formato de la evaluación heurística aplicada para las funcionalidades de gestión

#	Problema	Heurística	Valoración	Solución
1	Al crear aulas y elegir una Facultad, varias se repiten y no se sabe cuál elegir	H1	1	Añadir una descripción que especifique más detalles acerca de la Facultad para crear una diferenciación
2	Mucha información en la pantalla con poca claridad visual	H1 y H3	3	Añadir filtros para clasificar la información que interesa encontrar
3	Al congelar el cupo de un estudiante, no se provee algún mensaje de alerta para confirmar la acción	H5 y H9	4	Añadir una <i>ventana</i> en donde confirme si desea congelar al estudiante

Para resolver el problema 1, se incluyó en las descripciones de las ubicaciones de las aulas, una descripción corta de la ubicación, permitiendo diferenciar que no se trataba de las mismas ubicaciones.

Para la resolución del problema 2, fue necesario incorporar filtros en algunas de las interfaces, ya que otras interfaces sí poseían los mismos.



Para resolver el problema 3 se añadió una *ventana* que permite confirmar si se desea congelar al estudiante, previniendo así la alteración de los datos de los estudiantes con la activación errónea de esta opción.

### 3.1.4. Entrega

Una vez realizados los ajustes que surgieron en la evaluación heurística, se hizo la entrega de las funcionalidades de gestión a través de un repositorio presente en el *servidor*. Para dichas funcionalidades, se aplicaron pruebas de aceptación a la persona coordinadora de los cursos, la cual posee un perfil de usuario super administrador y dispone de las permisologías para acceder a todas las funcionalidades presentes en *ACEIM*. También se realizaron pruebas de aceptación por parte del personal administrativo de los cursos, las cuales consistieron en enseñarle al personal administrativo y experto del dominio, cómo utilizar las diversas funcionalidades para la gestión de los recursos de la Fundación, incorporando las nuevas solicitudes y mejoras sugeridas por parte del personal.

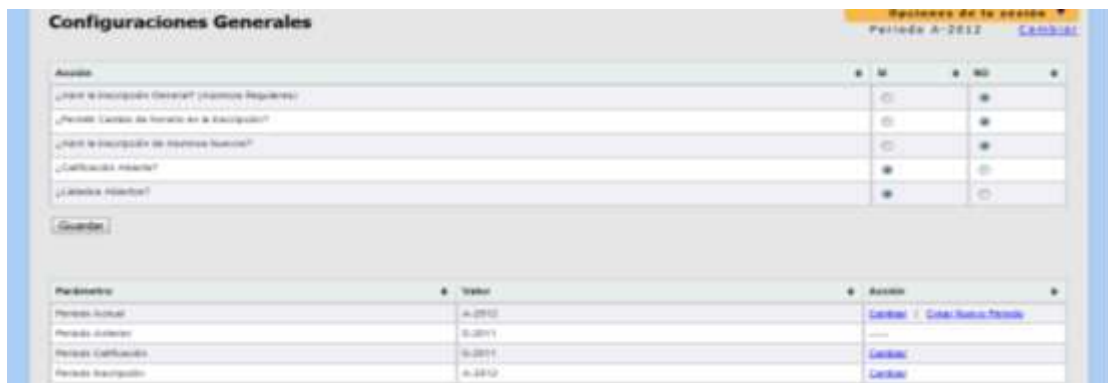
La primera prueba de aceptación consistió en modificar el contenido *web* de la *página* inicial de *ACEIM*. La interfaz empleada para dicha prueba fue la siguiente (ver Figura 3.6):



Figura 3.6: Interfaz para la edición del contenido *web* de *ACEIM*

Luego se realizó otra prueba de aceptación, aplicada al experto del dominio, al cual se le designó la actividad de modificar las configuraciones generales de los cursos

en un período determinado. A continuación se muestra la interfaz utilizada para la ejecución de dicha prueba de aceptación (ver Figura 3.7):



**Figura 3.7: Interfaz para configurar los parámetros generales de los cursos**

Una vez finalizadas las pruebas de aceptación, se formularon un conjunto de preguntas para las dos (2) tareas descritas anteriormente. Las mismas fueron aplicadas a un total de tres (3) personas.

- Pregunta 1: ¿Fue posible finalizar la tarea de manera exitosa?
- Pregunta 2: ¿El modo de realización de la tarea fue sencillo?
- Pregunta 3: ¿Los pasos llevados a cabo para completar la tarea son coherentes?

Para la primera prueba aplicada, los porcentajes de aceptación obtenidos fueron:

- Pregunta 1: 100% de aceptación.
- Pregunta 2: 80% de aceptación.
- Pregunta 3: 100% de aceptación.

Para la siguiente prueba aplicada, los porcentajes de aceptación obtenidos fueron los siguientes:

- Pregunta 1: 100% de aceptación.
- Pregunta 2: 100% de aceptación.
- Pregunta 3: 100% de aceptación.

Como se obtuvo un alto porcentaje de aceptación, se considera que los usuarios quedaron satisfechos con las funcionalidades desarrolladas.

## **3.2. Planificación**

La planificación se encuentra compuesta por varias funcionalidades que permiten realizar paso a paso la planificación trimestral de los cursos. Se presenta como un incremento del software y se describen las actividades realizadas en cada etapa del método de desarrollo de *software AgilUs*.

### **3.2.1. Requisitos**

En esta primera etapa del método se recolectó información en relación a las funcionalidades de planificación, aplicando diversas actividades como la tormenta de ideas, la entrevista, la elaboración del perfil de usuario y la definición de los requerimientos funcionales y no funcionales.

#### **3.2.1.1. Tormenta de ideas**

Luego de realizar reuniones con la persona encargada de coordinar los cursos, se recopilaron diversas ideas y opiniones para el desarrollo de las funcionalidades de planificación. Entre las ideas importantes se tienen:

- Se desea contar con una funcionalidad que permita marcar las aulas disponibles para el período actual.
- Incorporar una acción que permita liberar cupos, la cual consiste en eliminar de una sección a aquellos estudiantes preinscritos en la misma, cuya inscripción nunca fue formalizada en las oficinas de *FUNDEIM*.
- Se debe permitir asignar los bloques de horarios a las aulas disponibles, con la posibilidad de emparejarlas como se desee, es decir, asociar libremente un aula con otra aula que coincida en un mismo bloque horario.

- Como mejora en el proceso de creación de secciones, se debería permitir la creación de varias secciones a la vez y poder visualizarlas a medida que se van creando.
- Al asignar un instructor a una sección, validar que no se encuentre ya asignado a otra sección en el mismo horario.
- Considerar que un instructor puede dictar más de un curso (con un idioma diferente). Al asignar al instructor los horarios en los que se encuentra disponible, se debe poder especificar el horario por cada idioma que dicte.
- Al asignar un aula a una sección, validar que no se encuentre ya asignada a otra sección en el mismo horario.

### 3.2.1.2. Entrevista

Como otra actividad aplicada para la recolección de información en esta segunda iteración, se llevó a cabo una entrevista enfocada en las necesidades del usuario al momento de crear las secciones, marcar las aulas disponibles y elegir los bloques de horarios asociados.

Se realizaron una serie de preguntas a la persona coordinadora de los cursos, con el fin de conocer su opinión con respecto al proceso de crear las secciones. A continuación se encuentra la entrevista realizada.

- ¿Considera que para la creación de secciones, la asignación manual de cada aula que usted realiza, suma una carga adicional a esta parte del proceso?  
*“Sí, asignar las aulas a cada sección me resta tiempo que podría invertir en otra cosa. Me gustaría que las aulas no las tuviese que asignar yo sino que el sistema lo hiciera automáticamente y yo sólo tuviese que indicarle en donde quiero que ubique a la sección. Además sería bueno si pudiese indicar cuántas secciones quiero crear de una vez”.*

- ¿Le sería de utilidad tener una noción sobre las secciones creadas?

*“Sí, me parece algo útil porque así tengo un control de lo que llevo y no se me pasa abrir alguna sección por equivocación”.*

Con respecto al apartado para marcar las aulas disponibles para el nuevo trimestre, las ideas generales recolectadas en la entrevista fueron las siguientes:

- ¿Normalmente le asignan las mismas aulas trimestralmente?

*“Sí, normalmente me dan las mismas aulas, incluso en los mismos horarios, por el asunto de los convenios”.*

- ¿Qué facilitaría el proceso de marcar las aulas disponibles?

*“Si me ponen un listado de todas las aulas, me confundo. Serían útiles los filtros por ejemplo por ubicación, y así voy marcándolas por facultad”.*

En relación a la asignación de bloques de horarios para las aulas disponibles, se realizó la siguiente pregunta: ¿Considera que deben colocarse todos los bloques de horarios para ser asignados a cada aula, o encuentra este proceso muy largo y/o engorroso?

La respuesta por parte del coordinador de los cursos fue la siguiente: *“Si colocan todos los bloques puede resultar confuso porque es mucha información en la pantalla. Sería bueno si pueden agrupar los bloques y añadir filtros”.*

### **3.2.1.3. Perfil de usuario**

Para las funcionalidades de planificación, sólo interviene el rol de usuario super administrador, el cual es el encargado de coordinar y llevar a cabo los pasos que involucran la planificación trimestral de los cursos.

En la Tabla 3.5 se describe el perfil de usuario y sus características.

**Tabla 3.5: Perfil del usuario super administrador para el módulo de planificación**

Representante	Súper Administrador
<b>Habilidades</b>	Este tipo de usuario posee destrezas desarrolladas en el uso del computador, con un nivel intermedio en el uso de la <i>aplicación web</i> ya existente. Es el encargado de coordinar los cursos de extensión y es un profesor de la Escuela de Idiomas Modernos de la Universidad Central de Venezuela
<b>Responsabilidades</b>	<ul style="list-style-type: none"> <li>• Gestionar secciones, añadiendo nuevas, modificando y eliminando las existentes.</li> <li>• Liberar cupos de una determinada sección.</li> <li>• De las aulas que se encuentran en el sistema, marcar las que se encuentran disponibles para el trimestre.</li> <li>• Ingresar los horarios para los cuales están disponibles las aulas.</li> <li>• Ingresar el horario en el que se encuentran disponibles los instructores.</li> </ul>
<b>Grado de participación</b>	Alto

#### 3.2.1.4. Requerimientos funcionales y no funcionales

Luego de sucesivas reuniones con la persona coordinadora de los cursos y tras observar su forma habitual de llevar a cabo los pasos concernientes a la planificación, se definieron los requerimientos que se describen a continuación.

- **Requerimientos funcionales:**

- Gestión de secciones: se debe permitir crear nuevas secciones, incorporando la posibilidad de la creación por lote, modificar las existentes, ya sea cambiando

algún atributo de la misma, cerrando o eliminando la misma. También se debe incorporar la posibilidad de liberar cupos de una sección determinada.

- Asignación de horarios a las aulas: debe permitirse la posibilidad de relacionar uno o más bloques de horarios con cada aula disponible.
- Asignación de horarios a los instructores: se debe proveer una funcionalidad para que el coordinador de los cursos, luego de comunicarse con un instructor, pueda almacenar el horario para el cual se encuentra disponible dicho instructor, tomando en consideración que un instructor puede dictar cursos en más de un idioma en diferentes horarios.
- Asignación automática de aulas: se debe proveer de un procedimiento que, según unos parámetros específicos, asigne aulas disponibles a las secciones creadas, tomando en cuenta la cantidad de aulas disponibles y la ubicación de las mismas.

- **Requerimientos no funcionales:**

Para la iteración de planificación, se tomaron en cuenta los mismos requerimientos no funcionales considerados en la iteración de gestión y se demostraron de la misma manera.

### **3.2.2. Análisis**

En esta segunda etapa del método *AgilUS* se presenta el diagrama de casos de uso con su especificación, el modelo de objetos del dominio y diversos prototipos en papel.

#### **3.2.2.1. Casos de uso**

A continuación se presenta el diagrama de casos de uso utilizado para la planificación, en donde interviene el actor super administrador (ver Figura 3.8).

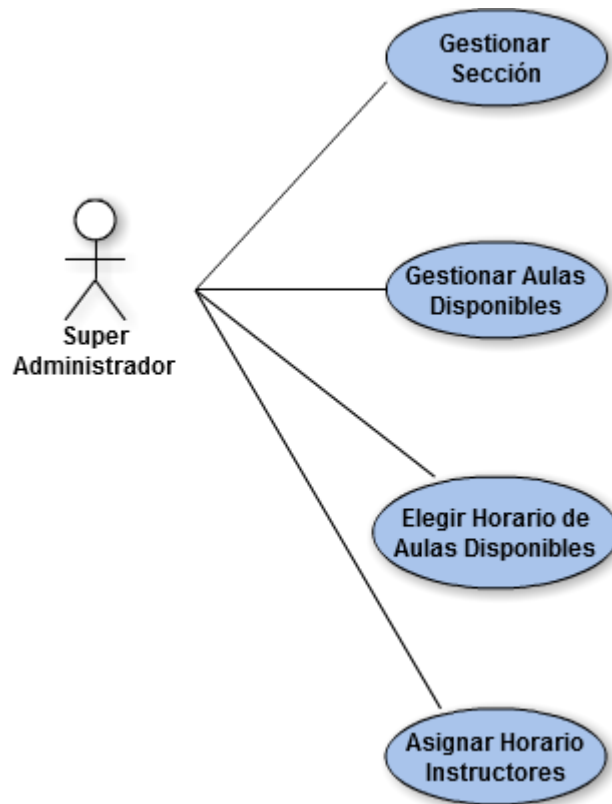


Figura 3.8: Diagrama de casos de uso de planificación

- **Especificación de los Casos de Uso:**

A continuación se describen los casos de uso presentes en el diagrama de la Figura 3.8.

- **Gestionar Sección:** permite al super administrador la creación de nuevas secciones para un curso, modificar una sección (cambiando algún atributo de la misma), el cierre o su eliminación (si no hay estudiantes en dicha sección) y la liberación de cupos.
- **Elegir Aulas Disponibles:** permite consultar el conjunto de aulas existentes y marcar aquellas que estarán disponibles para el nuevo período.
- **Elegir Horario de Aulas Disponibles:** funcionalidad que permite asociar un bloque de horario a cada aula del universo de aulas que se encuentran



disponibles. Es obligatorio que marque al menos un bloque de horario para un aula específica.

- **Asignar Horario Instructores:** otorga la posibilidad de registrar la disponibilidad de un instructor para un idioma dado, tomando en cuenta que un instructor puede dictar más de un curso en diferentes idiomas. Es necesario indicar el idioma y el horario para el cual se encuentra disponible el instructor.

### 3.2.2.2. Modelo de objetos del dominio

A través de este modelo se puede definir el dominio de la iteración de planificación, a partir de los objetos que lo conforman y sus relaciones. Los objetos presentes en el modelo de objetos de dominio de planificación son los siguientes:

- **Super administrador:** representa al usuario del sistema que tiene acceso a todas las funcionalidades referentes a la planificación de los cursos.
- **Nivel:** constituye los distintos niveles presentes en los cursos.
- **Idioma:** Representa los distintos idiomas ofertados; los mismos pueden ser: inglés, alemán, italiano, francés y portugués.
- **Sección:** modelo que representa las distintas secciones existentes para cada curso.
- **Aula:** representa los espacios en donde se dictan los cursos.
- **Horario:** contempla los distintos horarios en los que se dictan los cursos.
- **Instructor:** modelo que representa los educadores que imparten las clases en los cursos.
- **Categoría:** modela las distintas categorías que se manejan en los cursos; las mismas pueden ser niños, teens y adultos.

Entre los objetos mencionados anteriormente, se manejan las siguientes relaciones:

Un super administrador gestiona instructores, secciones y aulas. Los idiomas tienen asociados cuatro (4) o más niveles. Las secciones están asociadas a un horario y tienen asignada de una a dos (2) aulas. Un aula puede estar disponible en varios horarios. Un

instructor imparte clases a una o más secciones y se encuentran disponibles varios horarios. En la Figura 3.9 se encuentra el diagrama de modelo de objetos del dominio.

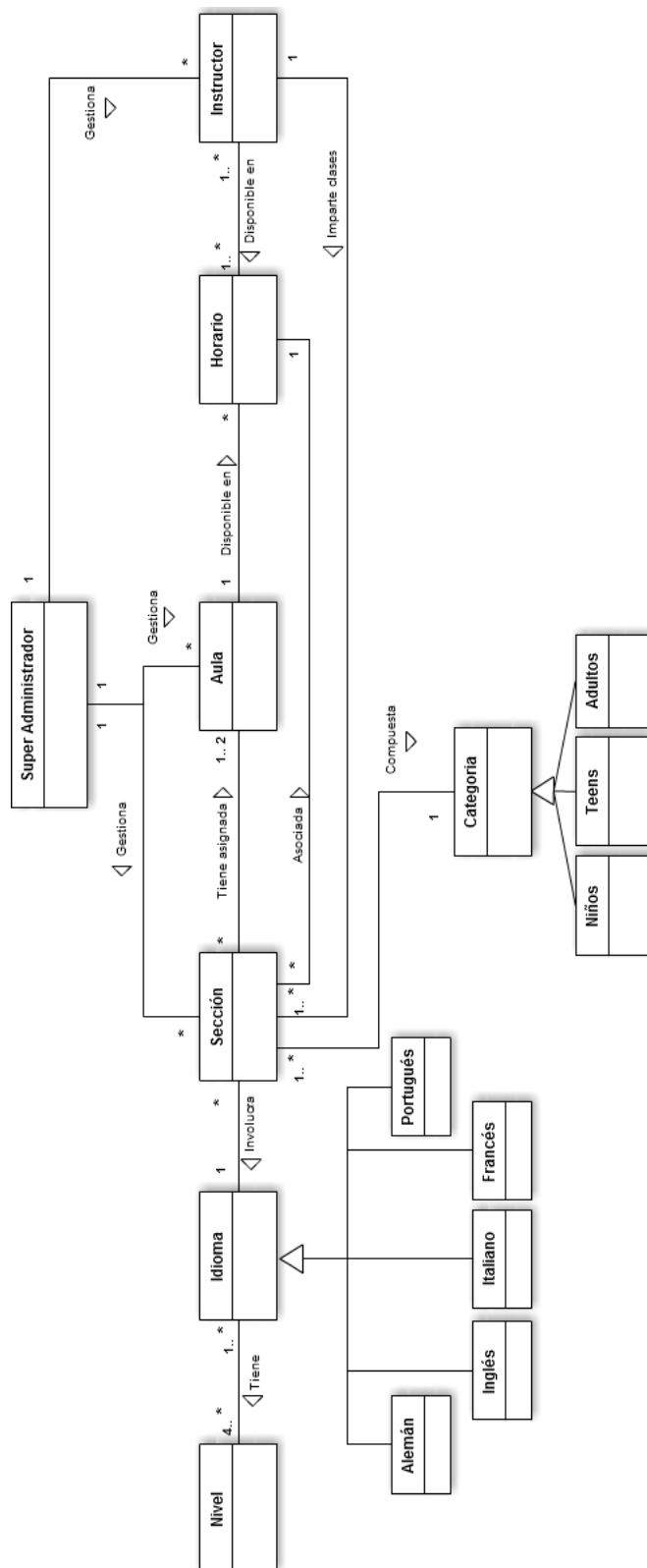


Figura 3.9: Modelo de objetos del dominio de planificación

### 3.2.2.3. Prototipo en papel

Una de las actividades realizadas durante el análisis de los requisitos para el desarrollo de la iteración de planificación consistió en la elaboración de varios prototipos en papel. Uno de ellos representa la interfaz que se le ofrece al usuario cuando desea crear una nueva sección. El primer prototipo se muestra en la Figura 3.10.

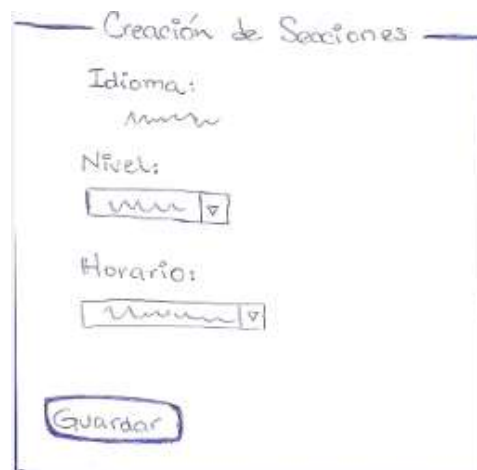


Figura 3.10: Primer prototipo en papel para la creación de secciones

Posteriormente, dicho prototipo fue sujeto a cambios, dando origen a un nuevo prototipo en el que se incorporó la funcionalidad de elegir el número de secciones que desea crearse automáticamente para un curso en específico (ver Figura 3.11).

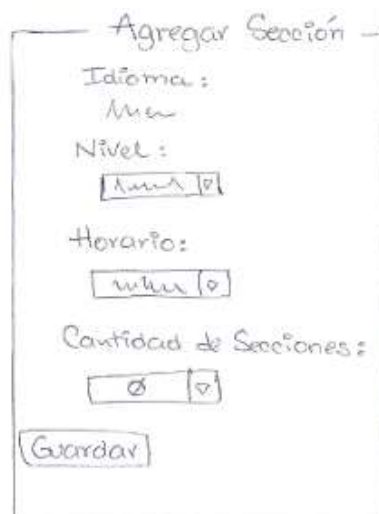


Figura 3.11: Prototipo en papel mejorado para la creación de secciones

Como idea inicial para el despliegue de la información sobre la asignación de un horario a un aula, se realizó un prototipo en papel que representa la interfaz en la que se muestran todas las aulas disponibles con cinco (5) bloques de horario a elegir (ver Figura 3.12).



Figura 3.12: Primer prototipo en papel para la asignación de horarios a las aulas

Luego del planteamiento del primer prototipo observado en la Figura 3.12, surge un inconveniente cuando se presenta el caso en el que un aula no se encuentra disponible para un bloque de horario completo. Es por esto que se rediseñó el prototipo, adaptándolo a las exigencias que a continuación se muestra en la Figura 3.13.



Figura 3.13: Prototipo en papel mejorado para la asignación de horarios a las aulas

### 3.2.3. Prototipaje

En esta fase se llevó a cabo el desarrollo de las funcionalidades estipuladas para la planificación, tomando en cuenta aspectos de usabilidad y los aportes ofrecidos por la persona coordinadora de los cursos durante la etapa de recolección de requisitos.

El prototipo que se presenta a continuación se trata de la interfaz utilizada al momento de asignar los bloques de horarios a cada una de las aulas disponibles para el trimestre (ver Figura 3.14).



Figura 3.14: Primer prototipo para la asignación de horarios a las aulas

Luego de varias discusiones realizadas con la persona coordinadora de los cursos, se definió un nuevo prototipo de interfaz que contempla mejoras en aspectos de usabilidad, tales como la incorporación de colores que permiten diferenciar los bloques de horarios, así como la agrupación de los mismos con una correspondencia de las aulas disponibles, logrando reducir la cantidad excesiva de información presente ante el usuario (ver Figura 3.15).



Figura 3.15: Prototipo final para la asignación de horarios a las aulas

### 3.2.3.1. Evaluación heurística

La evaluación heurística fue realizada por un experto del dominio y un experto en el área de *Interacción Humano-Computador*.

Fueron utilizadas las heurísticas propuestas por Nielsen (ver Capítulo 2). La escala de evaluación está comprendida entre 0-4 y contempla los siguientes valores:

- 0: No es un problema de usabilidad
- 1: Problema cosmético
- 2: Problema menor
- 3: Problema grave de usabilidad, importante fijar solución
- 4: Usabilidad catastrófica, imperativo fijar solución

Para la evaluación heurística, la persona experta en el dominio realizó la tarea de crear una nueva sección para un curso dado a través de la interfaz mostrada en la Figura 3.16.

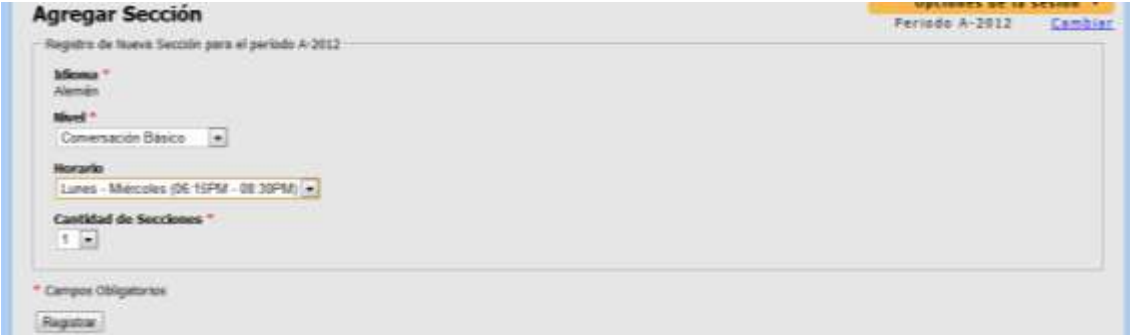


Figura 3.16: Interfaz para la creación de secciones

Luego se llevó a cabo otra actividad, la cual consistió en la asignación de un bloque de horario a cualquier aula disponible. Para ello se hizo uso de la siguiente interfaz (ver Figura 3.17):



Figura 3.17: Interfaz para asociar uno o más bloques de horario con las aulas disponibles

Por último se realizó la tarea de asignar un aula a la sección creada en la primera tarea. Para ello se hizo uso de la siguiente interfaz (ver Figura 3.18):

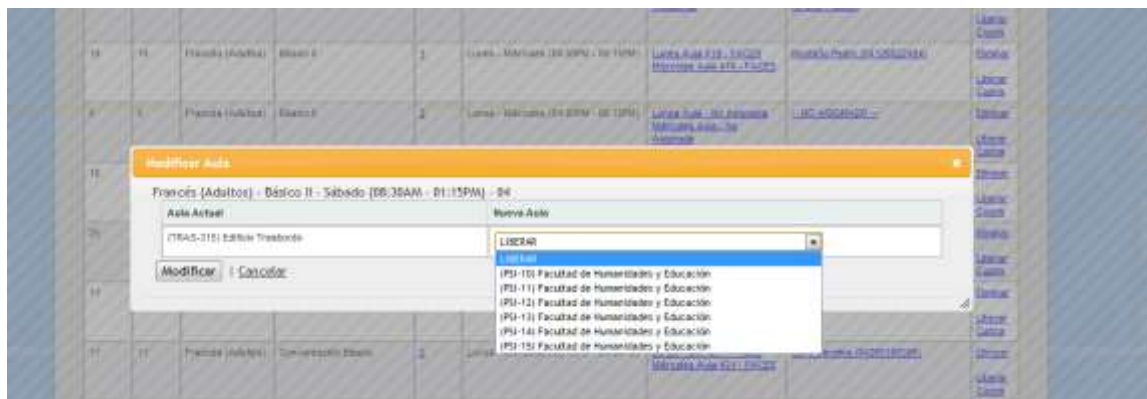


Figura 3.18: Interfaz para asignarle un aula a una sección

El formato utilizado para la planilla de evaluación heurística para planificación fue el siguiente (ver Tabla 3.6):

Tabla 3.6: Formato de la evaluación heurística aplicada para planificación

#	Problema	Heurística	Valoración	Solución
1	No existe un <i>feedback</i> al realizar la acción de crear una sección en la <i>aplicación</i>	H5	2	Al crear una sección, se debe informar que fue creada exitosamente
2	Mucha información en la pantalla con poca claridad visual	H1 y H3	3	Añadir filtros para clasificar la información que interesa encontrar

3	Al desplegar un <i>combo box</i> , se muestra todo el universo de opciones	H1	2	Añadir condiciones antes de mostrar todo el conjunto de aulas e instructores para filtrarlos
4	Mucha información desplegada en forma de columnas y de manera desordenada que entorpece la asignación de horarios a las aulas	H1 y H3	4	Agrupar los bloques de horarios, disminuyendo así la cantidad de columnas. Añadir colores que separen los bloques

Para resolver el problema 1, se incorporó un mensaje en la parte superior de la pantalla que parpadea durante unos segundos para captar la atención del usuario e informarle si la sección fue creada exitosamente o no.

Para el problema 2, se añadieron diversos filtros que permiten una mejor búsqueda dentro del contenido visualizado en la *página*.

El problema 3 requirió de una reformulación del procedimiento invocado para desplegar al conjunto de instructores y aulas al ser asignados, para mostrar solamente aquellas aulas e instructores que tienen el mismo bloque de horario asociado a la sección con la cual se está trabajando.

Para el último problema, se rediseñó la interfaz de asignación de bloques con aulas para una mayor facilidad y mejora visual durante este proceso.

### 3.2.4. Entrega

Una vez realizados los ajustes de la interfaz, se llevaron a cabo diversas pruebas de aceptación, las cuales proveen al equipo de desarrollo una opinión general de los usuarios acerca de las nuevas funcionalidades incorporadas en la *aplicación*.

Para los fines de la planificación, se aplicó la prueba de aceptación liberando el prototipo ejecutable de esta iteración y presentándolo ante la persona encargada de coordinar los cursos.



La primera prueba aplicada consistió en modificar el monto de la matrícula para el ingreso en los cursos. La siguiente interfaz fue la provista para dicha prueba (ver Figura 3.19):

ID	Descripción	Descuento	Total	Total + Pasaje	Acción
00700	Descuento Autorizado por Dirección 10%	10.0%	844.0	844.0	Actualizar
00700	Descuento Autorizado por Dirección 10%	10.0%	465.0	465.0	Actualizar
00700	00%	0.0%	844.0	844.0	Actualizar
00700	Convenio con la Facultad de Ciencias	00.0%	465.0	465.0	Actualizar
EP	Centro Plus	0.0%	820.0	840.0	Actualizar
001	Convenio con la Escuela de Estudios Internacionales	00.0%	465.0	465.0	Actualizar
001	Convenio con la Escuela de Idiomas Modernos	00.0%	465.0	465.0	Actualizar
000	Exonerado	100.0%	0.0	20.0	Actualizar
040	Convenio con la Facultad de Humanidades y Educación	00.0%	465.0	465.0	Actualizar
000	Convenio con la Facultad de Ingeniería	00.0%	465.0	465.0	Actualizar
000	Registro General	0.0%	700.0	940.0	Actualizar

Figura 3.19: Interfaz para gestionar los montos de los convenios

Luego se aplicó otra prueba de aceptación que consistió en la creación de un nuevo convenio, tal y como se muestra en la Figura 3.20:

**Registrar de Nuevo Tipo de Convenio:**

Identificador \*  Ejemplo: EM

Descripción \*  Ejemplo: Convenio con la Escuela de Idiomas Modernos

Descuento \*  Ejemplo: 20.0

\* Campos Obligatorios

Figura 3.20: Interfaz para añadir un nuevo convenio

Para ambas actividades, se efectuaron las siguientes preguntas:

- ¿El proceso resulta engorroso?
- ¿Cree que se puede mejorar?

Los resultados obtenidos para las actividades fueron los mismos: un 100% de aceptación, por lo cual se demuestra que el usuario quedó satisfecho.

## 3.3. Reportes

Dado que la gestión de los cursos involucra diversos recursos de *FUNDEIM*, se necesita una alternativa para proveer información al personal administrativo para la toma de algunas decisiones. Es por esto que se llevó a cabo la elaboración de un plan de iteración sobre reportes, el cual consiste en la generación de información haciendo uso de los datos existentes, dispuesta de una forma sencilla y comprensible para facilitar el análisis y la toma de decisiones por parte del grupo administrativo de los cursos de extensión.

### 3.3.1. Requisitos

En esta primera etapa de *AgilUS* se recolectó información concerniente a las necesidades del usuario, utilizando diversas actividades planteadas por *AgilUS* como la tormenta de ideas, la elaboración del perfil de usuario y la definición de los requerimientos funcionales y no funcionales.

#### 3.3.1.1. Tormenta de ideas

Luego de varios encuentros con el personal administrativo, incluyendo al coordinador de los cursos, surgieron diversas ideas relacionadas a los reportes; entre las más destacadas se encuentran:

- Visualizar a los alumnos que poseen convenios, con niveles de detalle como la cantidad por convenio y la especificación de quiénes son dichos alumnos.
- Visualizar un listado con la cantidad de alumnos aprobados y reprobados y en base a dichos parámetros, un reporte con estadísticas sobre cuántas secciones de recomienda abrir por curso.
- Consultar un listado de los alumnos que se encuentran congelados.
- Generar un listado con información referente a los depósitos bancarios, para tener un estimado de cuántos alumnos depositan en la cuenta de la Escuela de Idiomas Modernos y cuántos en la cuenta de *FUNDEIM*.

- Permitir incorporar una tabla control que pueda ser accedida en cualquier momento, para visualizar las secciones que han sido creadas y alertar sobre aquellos cursos que aún no posean una sección asignada como mínimo.
- Generar los reportes en formato PDF.

### **3.3.1.2. Perfil de usuario**

Para la iteración de reportes, participan aquellos usuarios con rol de administrador y aquellos con rol de super administrador, los cuales fueron descritos en la iteración de gestión. Cualquiera de los usuarios puede generar los reportes que considere necesarios y para los cuales se encuentra autorizado. Estos usuarios poseen conocimientos básicos sobre el uso del computador.

### **3.3.1.3. Requerimientos funcionales y no funcionales**

A continuación se presenta el listado de requerimientos obtenidos para el desarrollo e implementación de las funcionalidades para reportes.

- **Requerimientos funcionales:**

- Generación de reportes en formato PDF y en *Excel*: debe existir la posibilidad de generar cada uno de los reportes en formato PDF y en *Excel*.
- Visualizar alumnos con convenio: se debe proveer un listado de los convenios existentes en la Fundación, cantidad de alumnos por cada convenio y ofrecer un nivel de detalle para saber quiénes son.
- Visualizar alumnos por idioma: se debe poder generar un reporte que permita visualizar cuántos alumnos hay en cada idioma y en cada nivel.

- Visualizar secciones por idioma y por ubicación: para un mayor control de los cursos, se debe proveer una interfaz para visualizar cuántas secciones se han creado para cada uno de los cursos, colocando un identificativo especial a aquellos cursos que aún no poseen sección. Además otro dato importante que se debe proveer es el detalle o conjunto de secciones abiertas para una ubicación dada.
- Visualizar distribución de los depósitos bancarios: se desea conocer la cantidad de alumnos que depositan en la cuenta de la Escuela de idiomas y cuántos en la cuenta de *FUNDEIM*, tomando en cuenta la cantidad de alumnos con convenio, ya que el monto no es el mismo que el precio del ingreso general.
- Estadísticas generales por cada idioma: se debe poder visualizar la cantidad de alumnos aprobados, reprobados, y un estimado de cuántas secciones se deberían abrir para un curso en específico.

- **Requerimientos no funcionales:**

Para la iteración de reportes, se tomaron en cuenta los mismos requerimientos no funcionales que las iteraciones de planificación y gestión mencionadas con anterioridad.

### **3.3.2. Análisis**

Constituye la segunda etapa del método de *software* en la cual se lleva a cabo el análisis de la solución a desarrollar. A continuación se presentan los diagramas de casos de uso para reportes, con su especificación y el modelo de objetos del dominio (siguiendo la notación *UML*) para definir las funcionalidades que tendrá el software.

### 3.3.2.1. Casos de uso

A continuación se muestra el diagrama de casos de uso generado para reportes, junto con una breve especificación de las funcionalidades más relevantes (ver Figura 3.21).

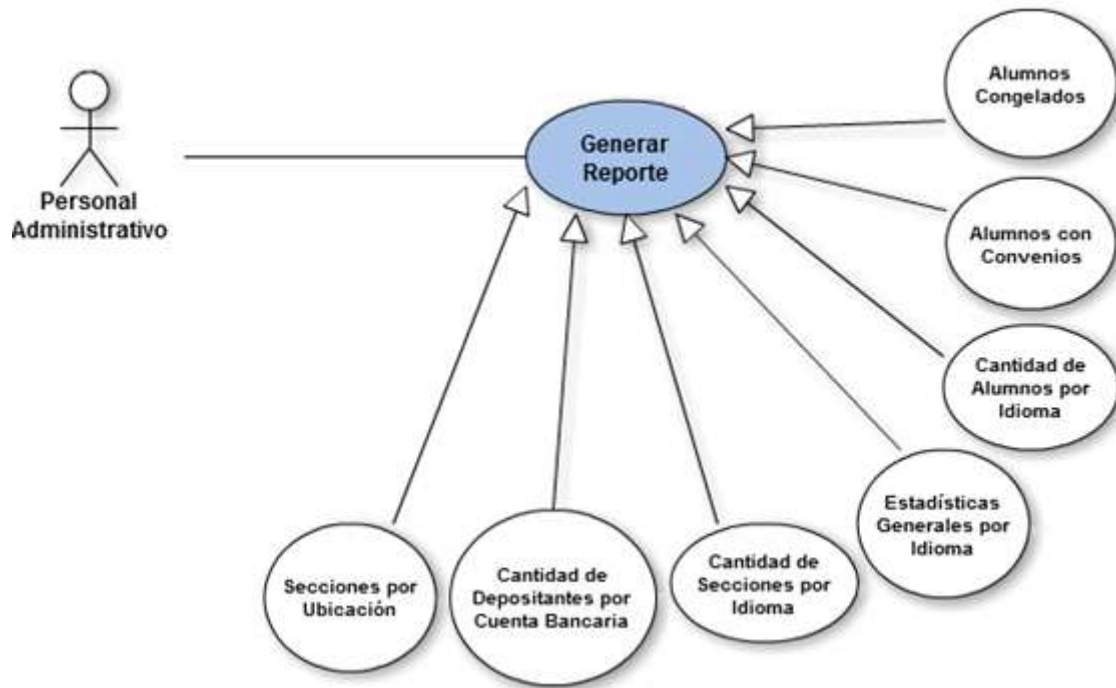


Figura 3.21 Diagrama de Casos de Uso para reportes

- **Especificación de los Casos de Uso:**

A continuación se encuentra una breve descripción del caso de uso genérico presentado en la Figura 3.21.

- **Generar Reporte:** esta funcionalidad permite la generación automática de reportes informativos los cuales son de vital importancia para la gestión, toma de decisiones y correcto funcionamiento de los cursos de idiomas. Algunos de los reportes poseen distintos niveles de detalle a fin de refinar y clasificar la información de manera granular. Algunos reportes están disponibles sólo para el usuario de tipo super administrador, ya que contienen información sensible que no debe ser accedida por los usuarios con rol de administrador.

### 3.3.2.2. Modelo de objetos del dominio

A través de este modelo se puede definir el dominio que componen los reportes, a partir de los objetos que lo conforman y sus relaciones. Los objetos presentes en el modelo de objetos de dominio de reportes son los siguientes:

- **Usuario administrador:** representa los usuarios del sistema, los cuales pueden ser administradores o super administradores.
- **Reporte:** representa el conjunto de reportes a generar.
- **Estudiante:** identifica a los usuarios que asisten a los cursos de idiomas.
- **Sección:** modelo que representa las distintas secciones existentes.
- **Aula:** representa las aulas donde se pueden dictar los cursos.
- **Idioma:** representa los distintos idiomas ofertados; los mismos pueden ser: inglés, alemán, italiano, francés y portugués.
- **Convenio:** representa los diferentes convenios y/o acuerdos que otorga la Fundación.

Las relaciones establecidas entre los objetos son las siguientes:

Los usuarios administradores generan reportes, los cuales incluyen estudiantes y/o secciones. Los estudiantes pueden o no poseer un convenio. Las secciones incluyen diversos estudiantes y están asociadas a un idioma y a una o dos aulas, y además incluyen estudiantes.

A continuación se muestra el diagrama de modelo de objetos del dominio generado para reportes (ver Figura 3.22).

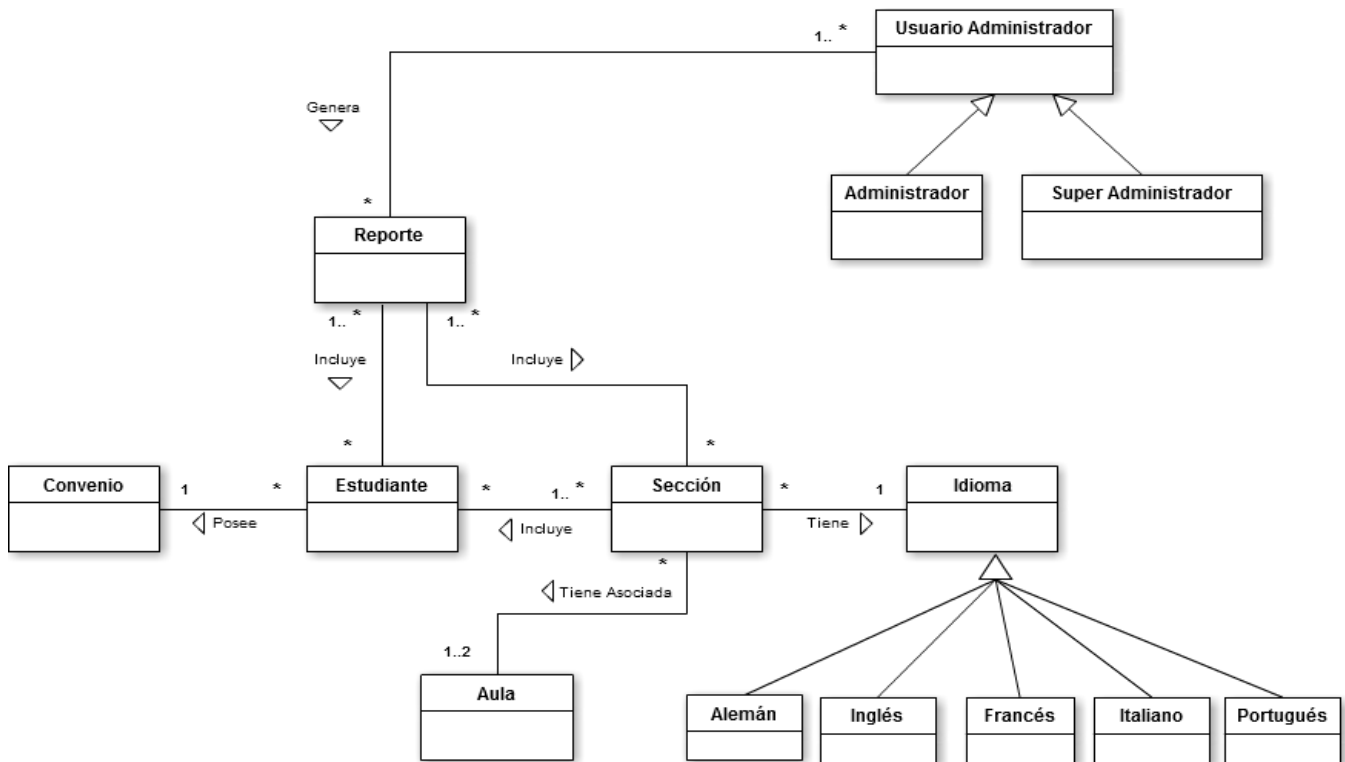


Figura 3.22: Modelo de objetos del dominio de reportes

### 3.3.3. Prototipaje

En esta etapa se realizaron prototipos funcionales que incluyeron la implementación de algunos reportes solicitados durante la recolección de los requerimientos en la primera etapa de la iteración de reportes. A continuación se muestra un prototipo de la interfaz utilizada al momento de generar el reporte para visualizar los alumnos por idioma (ver Figura 3.23).

Idioma	Cantidad de Alumnos	Detalle
Alemán (Adultos)	45	<a href="#">Ver Detalle</a>
Francés (Adultos)	204	<a href="#">Ver Detalle</a>
Inglés (Adultos)	1710	<a href="#">Ver Detalle</a>
Inglés (Teens)	148	<a href="#">Ver Detalle</a>
Italiano (Adultos)	121	<a href="#">Ver Detalle</a>
Portugués (Adultos)	34	<a href="#">Ver Detalle</a>

Figura 3.23: Reporte: Cantidad de alumnos por idioma

Si se desea obtener un nivel superior de detalle, se muestran los niveles conformados por el idioma elegido y la cantidad de alumnos por cada nivel (ver Figura 3.24).

Nivel	Cantidad de Alumnos	Detalle
Básico I	407	<a href="#">Ver Detalle</a>
Básico II	340	<a href="#">Ver Detalle</a>
Básico III	175	<a href="#">Ver Detalle</a>
Conversación Básico	100	<a href="#">Ver Detalle</a>
Intermedio I	82	<a href="#">Ver Detalle</a>
Intermedio II	120	<a href="#">Ver Detalle</a>
Intermedio III	77	<a href="#">Ver Detalle</a>
Conversación Intermedio	70	<a href="#">Ver Detalle</a>
Avanzado I	45	<a href="#">Ver Detalle</a>
Avanzado II	32	<a href="#">Ver Detalle</a>

**Figura 3.24: Reporte: Cantidad de alumnos para un idioma en específico**

En general, todos los reportes poseen el mismo formato de visualización: las columnas identificativas de los atributos a mostrar, y como última columna la posibilidad de visualizar el reporte con la adición de más detalles. Además, incorporan la opción de generarlos en formato PDF o en *Excel*.

### 3.3.4. Entrega

Una vez realizados los prototipos funcionales en la etapa anterior, se liberó un prototipo funcional y se presentó al personal administrativo de los cursos, incluyendo a la persona coordinadora de los cursos, quienes realizaron pruebas de aceptación a través de las cuales se tomaron en cuenta las observaciones emitidas.

Para la primera prueba de aceptación, se definió una pauta sobre la cual los usuarios debían acceder al reporte de la cantidad de alumnos por convenio y visualizar un nivel de detalle adicional que consiste en la cantidad de alumnos por convenio, dado uno en específico, por ejemplo, el convenio con la Facultad de Humanidades y Educación.



La interfaz para realizar esta actividad fue la siguiente (ver Figura 3.25):

**Detalle de Alumnos por Convenio**  
 Convenio con la Facultad de Humanidades y Educación

[Generar PDF](#) | [Generar Excel](#)

Filtrar Contenido:

#	Nombres	Cédula	Correo Electrónico	Teléfono Móvil	Idioma	Nivel
1	Bermudez Graul	8873258	graulbermudez@gmail.com	0988271254	Inglés (Adultos)	Básico I
2	Fernández Finestri Alaner Anny	21013503	maritzapiretti@gmail.com	04997259032	Inglés (Adultos)	Básico I
3	Peñabaz Rodríguez Angélica María José	18903251	angelicaviretti@hotmail.com	04987188800	Inglés (Adultos)	Básico I
4	Di Landeola Ariana Alejandra	26645345	riponza21@yahoo.com	04125827540	Inglés (Teens)	Básico I

[Generar PDF](#) | [Generar Excel](#)

**Figura 3.25: Interfaz para visualizar el reporte de los estudiantes para un convenio en específico**

Luego se definió otra pauta que consistió en visualizar a los alumnos congelados y generar el listado en PDF o en *Excel*. La interfaz utilizada para esta segunda prueba de aceptación es la presentada en la Figura 3.26.

**Alumnos Congelados en el Periodo A-2012**

[Generar PDF](#)

Filtrar Contenido:

#	Nombres	Cédula	Correo Electrónico	Teléfono Móvil	Idioma	Nivel
1	Guerra Oscar	13210287	osidiaspenzako@hotmail.com	04142750718	Alemán (Adultos)	Conversación Básica
2	Zambano Jossie	13672341	jossie.zambano@gmail.com	04265116105	Inglés (Adultos)	Intermedio I
3	Paradiso Vito	14058930	vpecorador@yahoo.com	04123663679	Italiano (Adultos)	Intermedio B
4	Ramos Hahn Kevin Alexander	19139204	kam_kevin@hotmail.com	04269181006	Italiano (Adultos)	Intermedio I
5	Sánchez Vargas Mariavita Milena	26290379	mlena_katy@hotmail.com	04142683019	Inglés (Adultos)	Avanzado B
6	Calabrava Jennifer	21376202	jennifer_calabrava@hotmail.com	04133702405	Italiano (Adultos)	Intermedio I
7	Falcón Dayana	23620900	daya_falon@hotmail.com	04148881986	Inglés (Adultos)	Básico I

[Generar PDF](#)

**Figura 3.26: Interfaz para visualizar a los alumnos cuyo cupo se encuentra congelado en el período actual**

Para cada pauta, se plantearon tres (3) preguntas:

- ¿Considera usted que el proceso para lograr la pauta establecida es sencillo?
- ¿Existe coherencia entre los diferentes pasos ejecutados para lograr la pauta planteada?
- ¿La apariencia del reporte generado en formato PDF le resulta agradable?

Las preguntas fueron planteadas a un total de cinco (5) personas. Los porcentajes de aceptación para cada una de las preguntas presentadas fueron los siguientes:

- Para la primera y segunda pregunta, se obtuvo un 100% de aceptación por parte de los participantes.
- Para la tercera pregunta se obtuvo un 80% de aceptación.

Luego de aplicadas las pruebas de aceptación, los miembros de la Fundación se mostraron satisfechos ante los reportes emitidos.

# Conclusiones

---

A lo largo del proyecto se plantearon las bases teóricas y prácticas para el desarrollo de un módulo de administración para llevar el control de los procesos críticos de los cursos extensión de la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la Universidad Central de Venezuela, con la finalidad de ampliar y mejorar las funcionalidades de la aplicación ya existente *ACEIM*. A continuación se describen las conclusiones obtenidas en base a los objetivos planteados.

Con los aportes brindados por parte del personal administrativo de los cursos y a través de los artefactos empleados a lo largo de cada una de las iteraciones realizadas, se desarrollaron diversas funcionalidades como soporte para los procesos de gestión, planificación y generación de reportes, logrando facilitar la ejecución de los mismos y reducir el tiempo invertido por parte del personal administrativo.

Hubo una aceptación positiva por parte de la Fundación ante la incorporación de las nuevas funcionalidades y la adaptación de procesos que ahora se realizan de manera automatizada.

A través de la participación activa y la observación simple aplicada al personal administrativo durante varios procesos pertinentes al funcionamiento de los cursos, se estudiaron con detenimiento los pasos a seguir para la planificación, gestión y generación de reportes. La importancia de este hecho permitió entender más de cerca la dinámica empleada por el personal administrativo y los detalles de las actividades y procesos realizados. Involucrarse directamente con el problema, hizo más fácil su entendimiento y la búsqueda de soluciones para el mismo.

La aplicación del método *AgilUS* en el desarrollo del proyecto, permitió implementar funcionalidades cercanas a los requerimientos del usuario debido a la constante retroalimentación existente. Los diferentes artefactos de *AgilUS* permitieron el diseño de una *interfaz* usable, tomando en cuenta determinados lineamientos y

estándares. Por tratarse de un método iterativo e incremental, se elaboraron las funcionalidades de manera progresiva, considerando al usuario en cada una de las etapas para el desarrollo de las mismas.

Las funcionalidades para la generación de reportes otorgan a la Fundación la posibilidad de cuantificar la información que maneja, así como el control de sus procesos más relevantes, y a través de las funcionalidades que soportan el proceso de gestión, se controlan los diferentes recursos de la Fundación.

El desarrollo de la funcionalidad para la edición del contenido *web* de ACEIM ofrece una independencia de la Fundación con el grupo de desarrollo, ya que el personal administrativo de los cursos puede añadir el contenido que desea a través de una *interfaz usable* e intuitiva. Existe la posibilidad de una difusión masiva de la información dirigida a cualquier tipo de usuario que navegue por Internet y acceda al *sitio web*.

La planificación inicialmente se trataba de un proceso complejo. Con las funciones automatizadas que actualmente se tienen a disposición, resulta más sencillo llevar a cabo la planificación o realizar cambios sobre la misma. La información se encuentra clasificada y ordenada, lo cual permite acceder directamente a los elementos necesarios para que se lleve a cabo el proceso de manera satisfactoria. El personal administrativo no depende del grupo de desarrollo porque tiene a disposición las herramientas necesarias. A través de la funcionalidad que provee estimaciones y proyecciones con los datos almacenados, se obtiene un aproximado de la cantidad de secciones que deben abrirse, ayudando de esta manera a la optimización de recursos. Existe un control de errores, los cuales son validados por la *aplicación*. Las funcionalidades de planificación pueden ser accedidas desde cualquier lugar y en cualquier momento, permitiendo realizar algún cambio sobre la planificación en caso de surgir algún imprevisto de último momento.

La aplicación de estándares de programación ayudó a la obtención de un código legible y ordenado, el cual facilita el mantenimiento y evolución de la aplicación a lo largo del tiempo. Mediante la implementación del patrón *MVC*, se facilitó la incorporación de las nuevas funcionalidades a la *aplicación* ya existente.

Se emplearon herramientas de *software libre*, lo cual contribuyó a un desarrollo económico y respaldado debido a la gran cantidad de fuentes de información disponible acerca de estas tecnologías.

Para culminar, gracias a las pruebas realizadas en cada iteración se verificaron los errores y fueron corregidos inmediatamente, de igual forma, gracias a los aportes del personal administrativo de los cursos, el objetivo general y los objetivos específicos planteados al comienzo del trabajo especial de grado fueron alcanzados.

El presente trabajo especial de grado no es sólo una propuesta o prototipo de sistema, sino que forma parte de una solución a un problema de la vida real; el mismo fue puesto en práctica y posteriormente en producción a través de las nuevas tecnologías de información.

# Recomendaciones

---

A continuación se presenta una lista de recomendaciones y/o sugerencias que están fuera del alcance del trabajo especial de grado y que pudieran ser aplicadas a las bases del sistema existente:

- Ampliación de las funcionalidades del módulo de instructores que les permita tener un mayor control sobre sus recursos; como por ejemplo el ingreso de sus horarios disponibles, la comunicación vía correo electrónico con sus alumnos, etc.
- Incorporación de nuevos reportes según las nuevas necesidades de la Fundación.
- Mejoramiento de la interfaz de usuario en cuanto a los accesos directos, menú, enlaces, etc. en función de ampliar más los lineamientos de usabilidad ya existentes.
- Para futuros aportes, se recomienda seguir los estándares que se han empleado para el desarrollo de la *aplicación*.

# Referencias

---

Acosta, E. (2010). *AgilUs: Construcción Ágil de la Usabilidad*. Universidad Central de Venezuela. Caracas, Venezuela. [Versión electrónica]. Consultado el 14 de enero de 2012. Disponible en: <http://dircompucv.ciens.ucv.ve/generador/sites/interaccion-humano-comp/archivos/Modelo Articulo.pdf>

Alvarez, R. (2010) *Introducción al HTLM*. Desarrollosweb.com. [En línea]. Consultado el 17 de julio de 2011. Disponible en: <http://www.desarrolloweb.com/articulos/533.php>

Ander, E. (1982). *Técnicas de investigación social*. Humanitas Alicante, España.

Carneiro, C. y Al Barazi, R. (2010). *Beginning Rails 3*. Editorial APress. Estados Unidos.

CoffeeScript (2011). Página Oficial. [En línea]. Consultado el 23 de septiembre de 2011. Disponible en: <http://jashkenas.github.com/coffee-script/>

Diccionario de Informática, (2011). Página Oficial. [En línea]. Consultado el 15 de febrero de 2012. Disponible en: <http://www.alegsa.com.ar/Dic/http.php>

Eguílu, J. (2009). *Introducción al CSS*. Autoedición. España.

Elrte.org (2011). Página Oficial. [En línea]. *Editor WYSIWYG para la Web de código abierto*. Consultado el 16 de marzo de 2012. Disponible en: <http://elrte.org/es>

Flanagan, D. (1998). *Javascript: The Definitive Guide, 3rd Edition*. Editorial O'Reilly Media. Estados Unidos.

Fowler, M. (2009). *Patterns of Enterprise Application Architecture*. Editorial Addison-Wesley. Estados Unidos.

FUNDEIM (2005). *FUNDEIM*. [En línea]. Consultado el 31 de julio de 2011. Disponible en: <http://www.fundeim.org/>

Garro, M. (2009). *Metodología de la Investigación Científica*. Chimbote, Perú. [Versión electrónica]. Consultado el 15 de agosto de 2011. Disponible en: <http://www.slideshare.net/maxgarro/metodologia-de-la-investigacion-presentation-954512>

Gil, V. (2010). ¿Qué es HTML5?. Codigobig.[En línea]. Consultado el 15 de febrero de 2012. Disponible en: <http://www.codigobit.info/2010/02/que-es-html5.html/>

Gilfillan, I. (2003). *La biblia de MySQL*. Editorial Anaya Multimedia. Madrid, España.

Gracia, J. (2003). *UML: Casos de uso*. Ingenierossoftware.com. [En línea]. Consultado el 17 de enero de 2012. Disponible en: <http://www.ingenierossoftware.com/analisisydiseno/casosdeuso.php>

HAML (2011). Página Oficial. [En línea]. #haml.about. Consultado el 15 de febrero de 2012. Disponible en: <http://haml-lang.com/>

Hernández, P. *El perfil del usuario de información*. Ejournal.unam.mx. [Versión electrónica]. Consultado el 20 de enero de 2012. Disponible en: <http://www.ejournal.unam.mx/ibi/vol07-15/IBI000701502.pdf>

Ingeniero de Software (2008). [Artículo en línea]. *Prácticas y métodos para mejorar el desarrollo de Proyectos de Software*. Consultado el 18 de diciembre de 2011. Disponible en: [www.ingenierossoftware.com](http://www.ingenierossoftware.com)

Jquery (2010). Página Oficial. [En línea]. Consultado el 28 de agosto de 2011. Disponible en: <http://jquery.com/>



Jquery UI (2010). Página Oficial. [En línea]. Consultado el 31 de agosto de 2011.  
Disponible en: [jqueryui.com/home](http://jqueryui.com/home)

Juan, J. (2011). *HTML5 Series: ¿Qué es HTML5?*. Baluart.net. [En línea]. Consultado el 15 de julio de 2011. Disponible en: <http://www.baluart.net/articulo/html5-series-que-es-html5-un-poco-de-historia>

Koontz, H. (1991). *Elementos de Administración*. Editorial Mc Graw Hill. México.

Lazo (2003). Marco Conceptual. [Artículo en línea]. Consultado el 15 de febrero de 2012. Disponible en: <http://www.lazogroup.ca/msproject/framework1-s.php>

Lemus, J. (2011). *CSS3 Más Social que Nunca*. Maestros del Web. [Artículo En línea]. Consultado el 17 de julio de 2011. Disponible en:  
<http://www.maestrosdelweb.com/editorial/css-3-mas-social-que-nunca/>

Márquez, B. (2004). *Implementación de un reconocedor de voz gratuito al sistema de ayuda a invidentes Dos-Vox en español*. [En línea]. Consultado el 16 de agosto de 2011. Disponible en:  
[http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lis/marquez\\_a\\_bm/capitulo5.pdf](http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf)

McCaw, A. (2012). *The Little Book on CoffeeScript*. Editorial O' Reilly. Estados Unidos.

Mora, L. (2002). *Programación de aplicaciones web: historia, principios básicos y cliente web*. Editorial Club Universitario. Alicante, España.

Moros, D. (2011). *Tópicos para el desarrollo de un modulo de planificación y administración del proceso de gestión académica de los cursos impartidos por la Escuela de Idiomas Modernos de la Universidad Central de Venezuela*. Caracas, Venezuela. T.E.G.

MySQL (2007). *MySQL*. [Artículo en línea]. Consultado el 17 de agosto de 2011. Disponible en: <http://www.techterms.com/definition/mysql>

Nahuel, H. (2011). *Tutorial RVM (Ruby Version Manager) Actualizado*. [Artículo en línea]. Consultado el 26 de julio 2011. Disponible en: <http://showmehowtocode.blogspot.com/2011/06/tutorial-rvm-ruby-version-manager.html>

NEO, Networking and Emerging Optimization, (2008). El Modelo Cliente Servidor. Universidad de Málaga, España. [Artículo en línea]. Consultado el 15 de febrero de 2012. Disponible en: <http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/cliente-servidor.html>

Nielsen, J. (1994). *Evaluación heurística*. Nueva York, Estados Unidos.

Priolo, S. (2008). *Ruby, Manual del Programador*. USERS. España.

Seguin, W. (2011). *Ruby Version Manager (RVM)*. Página Oficial. Consultado el 20 de Julio de 2011. Disponible en: <https://rvm.beginrescueend.com/>

TableSorter (2010). Página Oficial. [En línea]. Consultado el 28 de agosto de 2011. Disponible en: <http://tablesorter.com/docs/>

Ruby (2010). Página Oficial. [En línea]. *Ruby: a Programmer's Best Friend*. Consultado el 20 de julio de 2011. Disponible en: <http://www.ruby-lang.org/es/about/>

Ruby on Rails (2010). Página Oficial. [En línea]. *Ruby on Rails Guides*. Consultado el 20 de julio de 2011. Disponible en: <http://guides.rubyonrails.org/>

RVM (2010). Página Oficial. [En línea]. *RVM*. Consultado el 25 de septiembre de 2011. Disponible en: <http://beginrescueend.com/rvm/about>

SASS (2012). Página Oficial. [En línea]. Sass makes CSS fun again. Consultado el 15 de febrero de 2012. Disponible en: <http://sass-lang.com/>

Soto, L. *Requerimientos funcionales y no funcionales*. [Artículo en línea]. Consultado el 19 de enero de 2012. Disponible en:  
<http://www.mitecnologico.com/Main/RequerimientosFuncionalesYNoFuncionales>

Teixidor, S. (2010). Modelo Vista Controlador y algunas variantes. Neleste 2.0.[En línea]. Consultado el 15 de febrero de 2012. Disponible en:  
<http://www.neleste.com/modelo-vista-controlador-y-algunas-variantes/>

W3C (World Wide Web Consortium), (2011). Página Oficial. [En línea]. Consultado el 24 de agosto de 2011. Disponible en: <http://www.w3.org/>

Wellman, D. (2011). *jQuery UI 1.8, The User Interface Library for jQuery*. Editorial Packt. Estados Unidos.

Wikipedia, (2012). Página Oficial. [En línea]. *La enciclopedia libre*. Consultado el 15 de febrero de 2012. Disponible en: <http://es.wikipedia.org/>