Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Desarrollo de una nueva versión del portal generador de sitios web de asignaturas (*Portalasig*) para la Facultad de Ciencias de la Universidad Central de Venezuela

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela por el
Br. Juan Luis Sánchez Téllez
para optar al título de Licenciado en Computación

Tutor
Walter Hernández

Caracas, Febrero del 2016

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por el Bachiller Juan Luis Sánchez Téllez de C.I.: 20.870.886, con el título *Desarrollo de una nueva versión del portal generador de sitios web de asignaturas (Portalasig) para la Facultad de Ciencias de la Universidad Central de Venezuela*, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 23 de Febrero de 2016, a las 2:00 PM, para que su autor lo defendiera en forma pública, en el *Centro de Computación Gráfica de la Escuela de Computación*, lo cual se realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas a los veintitrés (23) días del mes de Febrero del año dos mil dieciséis (2016), dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Walter Hernández.

Prof. Walter Hernández
(Tutor)

Prof. Esmitt Ramírez
(Jurado Principal)

Prof. Eugenio Scalise
(Jurado Principal)

Resumen

El objetivo general del siguiente Trabajo Especial de Grado (TEG) fue desarrollar una nueva versión del portal generador de sitios web de asignaturas (*Portalasig*) de la Facultad de Ciencias de la Universidad Central de Venezuela. El desarrollo fue hecho desde el principio ya que se empleó un tipo distinto de arquitectura a la versión anterior del portal, que separa la lógica de negocio (*backend*) de las interfaces de usuario (*frontend*), por lo que fue necesario utilizar tecnologías que se especializan en este nuevo esquema. Todas las funcionalidades de la versión anterior fueron replicadas, como también se añadieron nuevas características con la finalidad de aumentar las posibilidades que tenían los usuarios del sistema. Finalmente fueron realizadas pruebas que compararon ambas versiones. Una prueba de preguntas a los usuarios que pertenecen a la comunidad académica y pruebas de desempeño que mesuraron la velocidad de carga. Ambas pruebas dieron como resultado que la nueva versión es bien aceptada por la comunidad de la Facultad de Ciencias y que también presenta mejorías en cuanto al desempeño y la usabilidad.

Palabras claves: *Portalasig*, portal web, frameworks de desarrollo web, web api, *rest*, *single page application*.

A mis Padres: María y Luis,
Quienes me dieron todo
Y son mi fuente de motivación
Para ser un gran hombre.

Agradecimientos

A Dios por darme la fuerza para llegar hasta el final de mi carrera.

A mis padres María Elena y Luis Alberto, por ser mi ejemplo de vida.

A mi hermano Luis Alejandro, por siempre estar pendiente del desarrollo de mi tesis y brindarme su apoyo.

A mis familiares, por siempre estar presentes.

A mi tutor Walter, gracias por el apoyo durante el desarrollo del trabajo.

A Sergio Rivas, por la ayuda y las ideas para la implementación de este proyecto.

A mis amigos del alma Andrés P, Antony, Francesca, Pedro, Deanne y Andrés A, por estar siempre pendientes de mí.

A mi novia Luanda, por todo el apoyo y estar pendiente del desarrollo del proyecto.

A Hillary por ayudarme con correcciones tanto del documento como de la aplicación.

A las sras. Linda Gutiérrez y María Guerriero, por haberme trasladado incontables veces a la Universidad y permitir que llegara a tiempo a mis actividades.

A todas las personas que de alguna manera me ayudaron para formarme profesionalmente.

Índice general

Índice de figuras

Índice de cuadros

Índice de códigos fuente

Introducción

En los últimos años con el auge de Internet y el rápido acceso a la información que le proporciona a cualquier usuario, la mayoría de las empresas, instituciones, universidades, comercios y otros entes, lo han tomado como una herramienta para facilitar diversos procesos que antes eran complicados y tediosos. Agregado a esto, actualmente, nos encontramos en una época donde las aplicaciones móviles y las aplicaciones web interactivas son el común denominador en soluciones de software y las predilectas por la mayoría de los usuarios. Estas soluciones pueden abarcar varios tipos de comunidades, como por ejemplo, comunidades en ámbito educativo y académico donde existen sistemas que permiten compartir información entre docentes y estudiantes.

La comunidad académica de la Facultad de Ciencias de la Universidad Central de Venezuela cuenta con un portal generador de sitios web, conocido como *Portalasig*, puesto en producción en el año 2013. El mismo sirve de conexión entre docentes y estudiantes a través de sitios web asociados a asignaturas. Dicho portal web ha funcionando desde entonces presentando ciertas limitaciones. Agregado a esto, la arquitectura de dicho portal no puede integrarse con servicios web o aplicaciones móviles nativas sin la necesidad de cambiar el código fuente.

Es por esto, que se ha propuesto realizar una nueva versión, empleando un nuevo tipo de arquitectura, solventar las limitaciones de la versión anterior e incluir nuevas funcionalidades, buscando la satisfacción de la comunidad académica de la Facultad de Ciencias de la Universidad Central de Venezuela.

El presente Trabajo Especial de Grado (TEG) está dividido en los siguientes cinco (5) capítulos.

El primer capítulo, titulado "Planteamiento del problema", describe la versión anterior de *Portalasig*, mostrando sus funcionalidades como también sus limitaciones. Este capítulo concluye explicando la justificación de realizar una nueva versión y los objetivos (general y específicos) del TEG.

El segundo capítulo es el "Marco teórico" en el cual se define lo que es un Portal web y las arquitecturas, patrones y tecnologías que son utilizadas para el desarrollo de una aplicación web.

En el tercer capítulo, titulado “Marco aplciativo”, se explica la metodología de desarrollo de software *SCRUM* y cuales componentes se tomaron de dicha metodología para la implementación de la nueva versión. Luego, son descritas las iteraciones de desarrollo con porciones del código fuente e imágenes con sus respectivas descripciones.

En el cuarto capítulo se muestran las “Pruebas” que se realizaron para la nueva versión del portal y luego se explican los “Resultados” obtenidos con una comparación con la versión anterior del sistema.

Finalmente, el quinto capítulo se exponen las “Conclusiones” del presente TEG, como también las “Recomendaciones” que consisten en posibles trabajos futuros que nacen con el desarrollo de esta nueva versión de *Portalasig*.

1 Planteamiento del problema

1.1. Antecedentes

1.1.1. Portalasig

Portalasig (Portal generador de sitios web de la Facultad de Ciencias de la Universidad Central de Venezuela) es un portal web destinado a la creación de sitios web para la Facultad de Ciencias de la Universidad Central de Venezuela, que innove la comunicación entre docentes y estudiantes a través de un sitio virtual [1]. Está en producción desde el año 2013 en el Centro de Computación de dicha facultad. Este portal contiene entre sus funcionalidades, la creación de asignaturas, docentes, estudiantes y nuevos sitios web de asignaturas. Cada sitio generado por el portal, cuenta con funcionalidades para manejar el contenido de la asignatura clasificado en las siguientes secciones: información general, objetivos, bibliografía, contenido temático, grupo docente, horarios, estudiantes, evaluaciones, entregas, planificación, calificaciones, noticias, foros, descargas, enviar correo, semestres anteriores y registro de actividades.

La metodología de desarrollo utilizada para la realización de *Portalasig* fue *AgilUs*, el cual es un método ágil de desarrollo por etapas, que se enfoca en la usabilidad del producto final.

Como posibles modificaciones futuras, el equipo de desarrollo de *Portalasig* concluyó [1]:

- Un módulo para la conexión al sistema de gestión académica Conest, lo cual reduciría el trabajo del administrador para la carga de las asignaturas, docentes y estudiantes.
- Un servicio web que permita la carga de las calificaciones de manera automática desde el sitio web generado hacia el sistema de gestión académica Conest, lo que optimizaría la carga de notas.
- Una interfaz de programación de aplicaciones (API, por sus siglas en inglés) para teléfonos inteligentes, lo cual facilitaría el acceso al sitio web.
- Funcionalidades de accesibilidad que permitan la inclusión de los usuarios con discapacidades, como aumentar el tamaño de las fuentes, cambiar el contraste de los colores, entre otros.
- Un módulo que permita la personalización del diseño de los sitios web.

1.1.1.1. Funcionalidades generales

Entre las características y funcionalidades que posee actualmente *Portalasig* se separan por los roles de los usuarios, los cuales son: Administrador, Docente, Preparador, Estudiante y Público General.

1.1.1.2. Administrador

- Agregar, modificar o eliminar asignaturas.
- Agregar, modificar o eliminar estudiantes.
- Agregar, modificar o eliminar docentes.

1.1.1.3. Docente

- Crear un sitio web de asignatura nuevo.
- Agregar y eliminar a docentes, preparadores y auxiliares docentes que pertenezcan al grupo docente de un sitio web de una asignatura.

1.1.1.4. Docente/Preparador

- Administrar información de la asignatura.
- Administrar secciones de estudiantes.
- Completar información pertinente a la asignatura, ya sea: información general, objetivos, bibliografía, contenido temático, grupo docente, horarios, estudiantes, evaluación, entregas, planificación, calificaciones, noticias, foros y descargas.
- Agregar evaluaciones que se realizan a lo largo del semestre.
- Agregar asignaciones a través de un modulo de entregas.
- Insertar los eventos que se vayan a realizar a lo largo del semestre en curso.
- Modificar calificaciones de distintas evaluaciones.
- Compartir contenido de noticias y foros a través de las redes sociales: Twitter, Facebook, Google+.
- Agregar archivos a la sección de descargas.
- Enviar un correo a todos los estudiantes cursantes de un sitio web de una asignatura.

1.1.1.5. Docente/Preparador/Estudiante

- Crear y participar en foros de una asignatura.
- Notificar vía correo electrónico cualquier cambio que se realice al sitio web en las secciones más relevantes, las cuales son: entregas, planificación, calificación, noticias, foros y descargas.

1.1.1.6. Todos los usuarios

- Permitir descargar de archivos agregados por el grupo docente.
- Visualizar la información de un sitio web creado.

1.1.1.7. Estado actual

Actualmente *Portalasig* se encuentra en normal funcionamiento proveyendo las funcionalidades que fueron anteriormente descritas. Sin embargo, es pertinente que

se mencione, que posee una serie de limitaciones y dificultades en algunos aspectos, los cuales son:

- En el panel administrativo no se implementa paginación y a la hora de listar a todos los estudiantes o docentes, la navegación es lenta y la interfaz se torna incómoda.
- La hora máxima de una entrega fue prevista que fuera a las 11:59PM del día seleccionado. Actualmente eso no es cumplido, las descargas se cierran mucho antes, generando inconvenientes con los estudiantes a la hora de entregar sus asignaciones a través del portal.
- La funcionalidad de entregas consta de un sistema de subida de archivos para que posteriormente sean revisados por el grupo docente de una materia en particular. Actualmente, la subida de archivos con un peso mayor aproximado a 2 MB no es posible ya que la petición es interrumpida después de unos segundos y el servidor retorna un código de error. Eso ocurre también con la funcionalidad de subir archivos para la sección de "Descargas" de un sitio web.
- El diseño actual del portal web no se adapta a dispositivos con pantallas de diferentes tamaños. Esto es un problema ya que sería deseable poder navegar con un diseño usable sin importar el dispositivo que se use para acceder al portal web.
- El esquema de la base de datos presenta algunos detalles de estándares. Existe tablas con nombres en inglés y también en español.

1.1.1.8. Tecnologías utilizadas

Para la realización del ambiente en producción de *Portalasig* se utilizaron las siguientes herramientas y frameworks:

- El lenguaje de programación *Ruby*, en su versión 1.9.3.
- El framework de desarrollo web *Ruby on Rails*, en su versión 3.2.13.
- El *Object-Relational mapping* (ORM) incluido en *Ruby on Rails*, *ActiveRecord*.
- *MySQL* como sistema manejador de base de datos.
- *Apache*, el servidor web.
- *Debian*, sistema operativo.

1.2. Justificación

Haciendo uso del Internet, se han creado diversas soluciones de software que hacen el día a día de las personas que lo emplean más fácil y llevadero, como lo es *Portalasig*, que es un portal que permite la comunicación entre docentes y estudiantes a través de sitios web de asignaturas. Este portal, puesto en producción en el año 2013, presenta diversas limitaciones en algunas de sus funcionalidades claves como lo son las entregas de asignaciones de los estudiantes y la subida de archivos en la sección de descarga, lo cual presenta directamente un problema a las personas que lo utilizan.

El desarrollo de la nueva versión se realiza con la finalidad de replicar todas las funcionalidades de la versión anterior, solventando los problemas mencionados anteriormente. Otra finalidad del desarrollo es cambiar el tipo de arquitectura del sistema, que consiste en separar la lógica de negocio de las interfaces de usuarios propias del portal. Todo esto, con la finalidad de poder proveer e integrar los diversos servicios del sistema con otros componentes o aplicaciones, como por ejemplo, aplicaciones móviles nativas. Esta separación (lógica de negocio e interfaces) no puede ser realizada sin la necesidad de cambiar el código fuente de la versión anterior, por lo cual la nueva versión será desarrollada desde el principio usando tecnologías que favorezcan el nuevo tipo de arquitectura para el portal.

1.3. Objetivo general

Desarrollar una nueva versión del portal generador de sitios web (*Portalasig*) para la Facultad de Ciencias de la Universidad Central de Venezuela.

1.4. Objetivos específicos

- Implementar una arquitectura que conste de una aplicación de *backend* como web API *Rest* y una aplicación de *frontend* de una única página (*SPA*).
- Cambiar la permisología de los roles Docente y Preparador.
- Agregar el rol Coordinador, el cual sea el Docente con mayores privilegios en un sitio web.
- Mantener los datos de los usuarios, para que puedan acceder sin tener que ser creados nuevamente.

- Mantener los datos de las asignaturas ya utilizadas, con su respectiva clasificación entre los semestres.
- Diseñar interfaces *responsive*, para que el portal web sea correctamente usable al momento de ser accedido a través de dispositivos móviles

2

Marco teórico

2.1. Portal web

El término portal comúnmente es utilizado para referirse a alguna puerta o entrada a algún sitio [2]. Hablando en el contexto de aplicaciones web, no difiere mucho, ya que un portal web es una puerta a diferentes sitios web, ya sea indexándolos a través de enlaces o creándolos en la misma aplicación.

Los portales web son a menudo la primera página que cargan los usuarios a conectarse en la web. Son muy utilizados para realizar páginas web que requieran modificar sus contenidos a menudo, ya que funcionan como un manejador de contenidos, haciendo que, cualquier modificación en la información se haga de manera rápida y sencilla. El éxito de un portal depende de su habilidad de proveer un sitio base en donde los usuarios puedan regresar una vez finalizada la navegación a los demás sitios que lo componen.

2.1.1. Clasificación

En la actualidad no existe algún estándar para clasificar los portales web, recae directamente en el autor que los defina [2]. De una manera muy general se pueden clasificar en dos (2) grandes grupos los cuales son:

2.1.1.1. Horizontal

Los portales horizontales se enfocan en obtener la atención de toda la comunidad. Estos sitios, son llamados comúnmente "megaportales", usualmente contienen motores de búsquedas y le proveen al usuario la habilidad de personalizar la página ofreciendo varios canales (como por ejemplo, información al clima regional, bolsa de valores, noticias).

2.1.1.2. Vertical

Los portales web verticales difieren sólo en la información que quieren reflejar, la cual es mucho más especifica sobre algún contexto. Son sitios que sirven como puerta a información relacionada a un negocio en particular, como pueden ser empresas, e-commerce, académicos entre muchos otros tópicos.

Figura 2.1: Diferencia entre portales *horizontales* y *verticales* en base a los usuarios que los utilizan.

En síntesis, la clasificación de los portales resalta el tipo de información que provee y la comunidad a la cual está dirigido. En la *Figura* 2.1, se puede observar que, los portales *horizontales* intentan llamar la atención de un *público general*, ya que la información que se encuentra dentro de estos es muy variada y no siguen un contexto

específico. Un ejemplo clásico lo representa el portal web de *Yahoo!*.

Después, es apreciable, como los portales *verticales* buscan la atención de usuarios con necesidades e intenciones particulares. Un ejemplo muy cercano de este tipo de portales es el Portal de asignaturas de la Facultad de Ciencias de la Universidad Central de Venezuela (*Portalasig*) ya que la comunidad a la cual va orientada las funciones de dicho portal web es a una comunidad académica, específicamente, la comunidad académica de la Facultad de Ciencias.

2.1.2. Elementos

Los portales web pueden ser muy diversos en cuanto al público al cual quieren obtener su atención o simplemente el contenido, pero todos los portales deben contener una serie de elementos en común [3]. No todos los elementos a continuación deben estar presentes en cada uno de los portales existentes y algunos elementos pueden ser más importantes que otros dependiendo de la audiencia o la lógica del negocio de cada portal. Por ejemplo, un portal dirigido a usuarios anónimos, sin autenticación, podría ofrecer sin ningún problema un soporte limitado de transacciones seguras, integridad de los datos o alguna cualidad de colaboración entre usuarios. En cambio, un portal basado en el manejo de información y conocimientos puede enfocarse únicamente en motores de búsquedas e integración de los datos.

Los elementos claves de los portales web son los siguientes:

- Aspecto y apariencia.
- Seguridad.
- Perfil de usuario.
- Personalización.
- Taxonomía y navegación dinámica.
- Integración.
- Base de datos.
- Herramientas de colaboración.
- Motor de búsqueda.

2.1.2.1. Aspecto y apariencia

Como cualquier programa que interactúa con ser humano, los portales web tienen una apariencia o una interfaz gráfica de usuario (*GUI*). Muchos desarrolladores pueden no darle mucha importancia al aspecto o simplemente no preocuparse en lo absoluto, lo cual puede ser un error muy grave ya que, en el mundo de aplicaciones de software hechas para un usuario final, lo que realmente le importa al consumidor es que la aplicación se vea bien, sea usable e intuitiva, aparte de que logre las funcionalidades para las cuales fue creada. En el caso de los portales web, la página de inicio debe indicarle al usuario que está efectivamente , en un portal, ya sea por colocar en el titulo de *Portal* en el sitio o no, o si no, indicarles en todo momento en que sección o sitio web, alojado en el portal, están en un momento determinado y también, ofrecer una manera fácil e intuitiva de navegar a otra sección o simplemente regresar a la página de inicio.

2.1.2.2. Seguridad

Todo sistema en el cual se manejen cuentas de usuarios y el contenido que éstos pueden crear, colaborar o integrar, es importante establecer método de seguridad, ya sea, para poder identificar de manera univoca a cada uno de los usuarios (autenticación) o evitar que un usuario particular pueda violentar los datos de otro usuario como también entrar a secciones prohibidas las cuales no tiene el privilegio (autorización y roles). El concepto primordial de un portal web es alojar múltiples sitios web en éste, es importante que se maneje un sistema de autenticación que permita a los usuarios identificarse a lo largo de todo el contenido que el portal web provee.

2.1.2.3. Perfil de usuario

Como se mencionó anteriormente, es importante proveer la identificación única a cada usuario, pero no sólo basta con eso. Es importante también que cada usuario tenga la posibilidad de poder ver o modificar en cualquier momento sus datos personales a través de un panel o sitio web que normalmente recibe el nombre de *perfil de usuario*.

2.1.2.4. Personalización

Un portal web debe ser capaz de proveer criterios donde los usuarios decidan como quieren que algún tipo de contenido le sea mostrado o la manera de los mensajes sean entregados a ellos. Es importante decir que cada usuario es único, y no sólo eso,

cada usuario, dependiendo del tipo de portal, puede pertenecer a una cantidad mayor de sitios web que otro usuario. Por ejemplo, en el caso del portal de asignaturas de la Facultad de Ciencias de la Universidad Central de Venezuela (*Portalasig*), un estudiante puede pertenecer a una de las carreras que la facultad provee, por lo tanto es importante que, una vez que un estudiante es identificado en el sistema, el portal sea capaz de resaltarle la información que en verdad le puede interesar a un usuario y que, en este caso en particular, sería las materias que cursa actualmente.

2.1.2.5. Taxonomía

La taxonomía es una jerarquía de categorías usada para simplificar la navegación y la búsqueda. Las categorías están estrictamente relacionadas al tipo de contenido que provee un dicho portal. Por ejemplo, en *Portalasig*, siendo un portal de asignaturas de las carreras dictadas en la Facultad de Ciencias, es importante tener una jerarquía Carrera/Materia/Sitios por semestre, ya que de esta manera se le da un correcto significado y navegación a las funcionalidades que puede realizar la comunidad de dicha facultad.

2.1.2.6. Integración

El límite de un portal web no debe ser su propio dominio de información. En el mundo de las aplicaciones de la *Web 2.0*, un concepto clave es la integración, que da vida a los llamados servicios web que nacen con la necesidad de querer realizar una comunicación máquina-máquina.

2.1.2.7. Base de datos

Detrás de la mayoría de las aplicaciones web, como los portales web, reside una base de datos, usualmente relacional. En un portal web donde se requiere tener un sistema de autenticación, manejar roles, diferentes sitios web, es necesario una base de datos para poseer unos datos íntegros y disponibles en todo momento para el correcto funcionamiento del portal web.

2.1.2.8. Herramientas de colaboración

Los portales de hoy en día deberían contener herramientas que sirvan para la colaboración mutua entre usuario como, dar a conocer la presencia o no de un usuario, mensajería instantánea, comunidades virtuales, un sitio dedicado a discusiones y

perfiles de usuario. Actualmente *Portalasig* posee la funcionalidad de que los usuarios, registrados a los sitios web de alguna materia, puedan participar en foros, que normalmente, son usados para dudas en proyectos o asignaturas.

2.1.2.9. Motores de búsqueda

Esta característica es sumamente importante, ya que recordando que un portal web es la *puerta* a diferentes sitios web, debe existir la posibilidad de que los usuarios puedan buscar algún contenido o material en alguno de estos sitios de manera rápida, sin tener que visitar cada uno de los sitios uno por uno hasta lograr encontrar lo deseado. Para esto nacen los motores de búsqueda que pueden ser tan simples como un campo de texto que tenga una lógica de indexación y una función de búsqueda en todos los sitios web pertenecientes.

2.2. Arquitecturas y patrones

Las nuevas tendencias no se detienen a nivel de plataformas o *APIs* que la *Web* provee actualmente, sino también a nivel de patrones y arquitecturas. Uno de los patrones más usado a la hora de realizar una aplicación web es *MVC* (Modelo Vista Controlador) que básicamente su función es separar los datos de la lógica de negocio.

2.2.1. Modelo-Vista-Controlador

Como fue anteriormente mencionado, el patrón Modelo Vista Controlador (*MVC*) [4] es uno de los patrones más usados a la hora de realizar una aplicación web. Como su nombre lo indica, consiste básicamente en tres (3) componentes y en como éstos interactúan uno con el otro para poder cumplir correctamente la lógica de negocio y las funcionalidades predefinidas de la aplicación.

2.2.1.1. Modelo

Es el ente encargado de representar todos los datos e información con la cual la aplicación opera, por lo tanto gestiona todos los accesos a dicha información, como las consultas o nuevas actualizaciones, implementando también los privilegios de accesos que estén previstos en una aplicación en específico (lógica de negocio).

La información que está preparada para ser mostrada (comúnmente a un usuario) es enviada a la *Vista*. Las peticiones de acceso o futura manipulación de los datos llegan al *Modelo* a través del *Controlador*.

2.2.1.2. Vista

Es la parte con la cual el usuario interactúa directamente. Se encarga de presentar el *Modelo* (información y lógica de negocio) en un formato adecuado para que un usuario pueda recibir la información correctamente.

La vista normalmente, es el resultado de renderización de archivos *HTML*, *CSS* y *Javascript*, pero también pueden ser simples lenguajes de comunicación como XML o JSON.

2.2.1.3. Controlador

Es el encargado de escuchar y responder los eventos (usualmente un usuario) e invoca peticiones al *Modelo* cuando se se hace alguna solicitud sobre la información, como editar algún documento o registro en una base de datos. También puede enviar cambios de presentación a la *Vista* asociada si se presenta un cambio en el *Modelo*, por lo tanto se puede considerar que el *Controlador* es un intermediario entre la *Vista* y el *Modelo*.

Finalmente, se posee un patrón de diseño donde la clave fundamental es la manera de cómo sus componentes colaboran entre sí y el usuario, como es posible observarlo en la *Figura* 2.2:

- El usuario a través de alguna acción, ya sea: ir a una ruta, mandar datos de un formulario, entre otras, *usa* algún procedimiento previsto en el *Controlador*.
- En el procedimiento ejecutado, según la lógica, puede o no haber alguna *manipulación* de los datos que dan vida a la lógica de negocio que se encuentran en el *Modelo*.
- El *Modelo* una vez que tenga los datos solicitados o simplemente haya terminado sus operaciones con los datos persistentes, es necesario dar una respuesta al usuario, por lo tanto se *actualiza* la *Vista*.
- Finalmente el usuario *observa* los cambios y notificaciones, para así poder realizar una futura acción en la aplicación web.

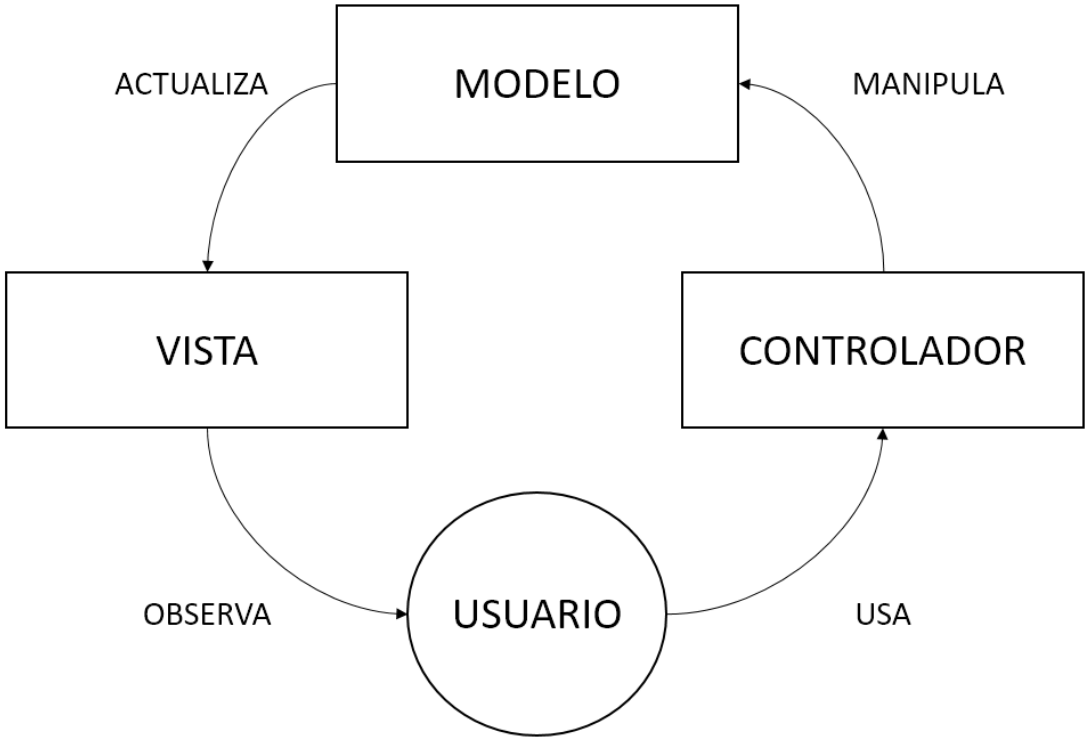


Figura 2.2: Colaboración de los componentes del patrón *MVC*.

2.2.2. Mapeo Objeto-Relación

El Mapeo Objeto-Relación (*Object-Relational mapping*, en su siglas en inglés) es una técnica de programación muy utilizada que sirve para conectar objetos propios de una aplicación, con tablas de un sistema manejador de base de datos. Usando un ORM, las propiedades y relaciones de los objetos de una aplicación puede ser fácilmente registrado y consultados de una base de datos sin tener que escribir sentencias *SQL* directamente, abstrayendo al desarrollador en su mayoría.

Esta técnica es comúnmente usada junto al patrón Modelo Vista Controlador (*MVC*), donde se obtiene que el *ORM* es parte fundamental del *Modelo*, ya que es el encargado de entablar una comunicación con un repositorio de datos, como por ejemplo, con un sistema manejador de base de datos, *Figura* 2.3.

Un ORM bien formado debería ser capaz de:

- Representar los modelos y sus datos.

- Representar asociaciones entre dichos modelos.
- Representar herencias jerárquicas a través de los modelos.
- Validar los modelos antes que estos sean persistidos en una base de datos.
- Permitir operaciones de base de datos de una manera orientada a objetos.

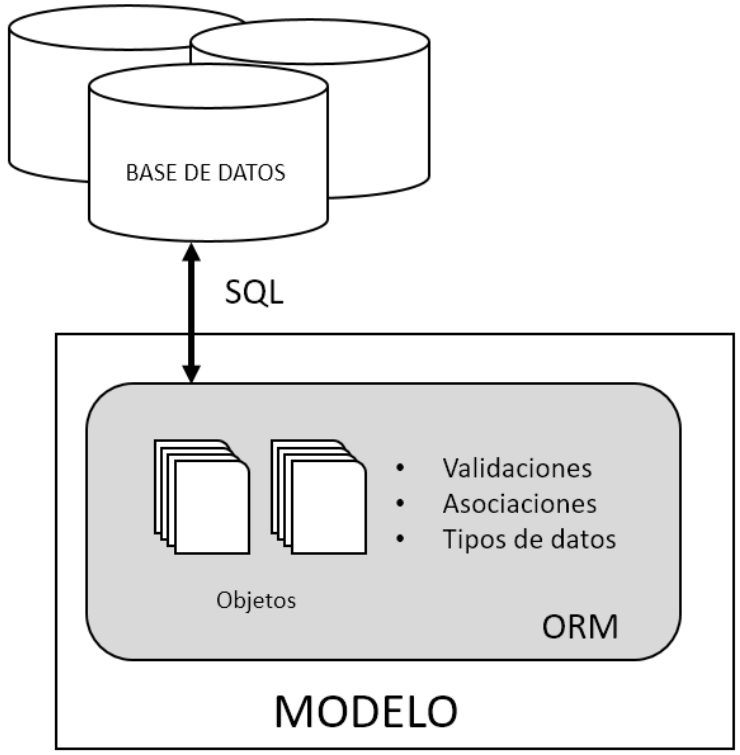


Figura 2.3: Integración de la técnica *ORM* con el *Modelo* del patrón *MVC*.

2.2.3. *Web APIs* y *Single Page Applicaction*

Unas de las tendencias con un auge considerado actualmente, es la realización de aplicaciones web usando una arquitectura que consta en dos (2) partes:

- La primera parte basada en recursos (*Rest*) del lado del servidor (o *backend*).
- Y la segunda parte que se ejecutaría en el lado del cliente (*frontend*), generalmente se usan páginas web enriquecidas o se usan un conjunto de técnicas para dar vida a una *Aplicación de una Sola página* (o *Single-page Application* en inglés).

2.2.3.1. *Web API* Rest

Rest es una arquitectura de desarrollo en *servicios web*, o mejor llamados *Web APIs*. El principio de Rest es utilizar el protocolo HTTP con "significado" en el sentido de [5]:

- Usar correctamente los métodos HTTP; *GET* para consultas, *POST* para enviar datos, *PUT/PATCH* para modificar datos y *DELETE* para eliminar datos. Estas operaciones en conjunto llevan el nombre de *CRUD* (Crear, Leer, Actualizar, Eliminar, según sus siglas en inglés).
- Usar correctamente los códigos de estado de respuesta HTTP.
- Dar un buen uso a las cabeceras HTTP y a su significado. Una de las cabeceras más usadas es *Accept* la cual indica el formato de datos que espera o "acepta" el cliente que realizo una petición.

2.2.3.2. *Single Page Applicaction*

Una *Single Page Applicaction* (*SPA*) [6], es una aplicación web que encaja en una sola página web dando una experiencia al usuario similar a la de una aplicación de escritorio. Al igual que cualquier página web, es necesario código *HTML*, *Javascript* y *CSS*, sólo que este es obtenido en su mayoría en la primera carga de una aplicación *SPA*. Esta primera carga usualmente recibe el nombre de *Bootstraping*. Una vez que la primera carga es completada y el cliente web renderiza la página, no hay necesidad de recargar nuevamente y, los futuros cambios de contenidos se realizarán generalmente con *AJAX* o *WebSockets*.

Al final la interacción de un *Web API* con una aplicación en el lado del cliente *SPA*, sería de la siguiente manera, *Figura* 2.4:

- Se realiza la primera carga de archivos *HTML*, *CSS* y *Javascript*, que usualmente se le conoce como *Bootstraping*.
- La lógica de la *SPA* estará en los archivos *Javascript*, ya alojados en el cliente web.
- La actualización posterior de componentes de la aplicación web se realizaran mediante peticiones *HTTP* en segundo plano, que pueden ser realizadas con la técnica *AJAX*.
- El *Web API* al recibir peticiones *AJAX*, responderá con lenguajes de comunicación como *XML* o *JSON*.
- Una vez que el cliente web reciba las respuestas en *JSON* o *XML*, la lógica de la *SPA* se deberá de encargar de actualizar de manera dinámica la *Vista* al usuario.

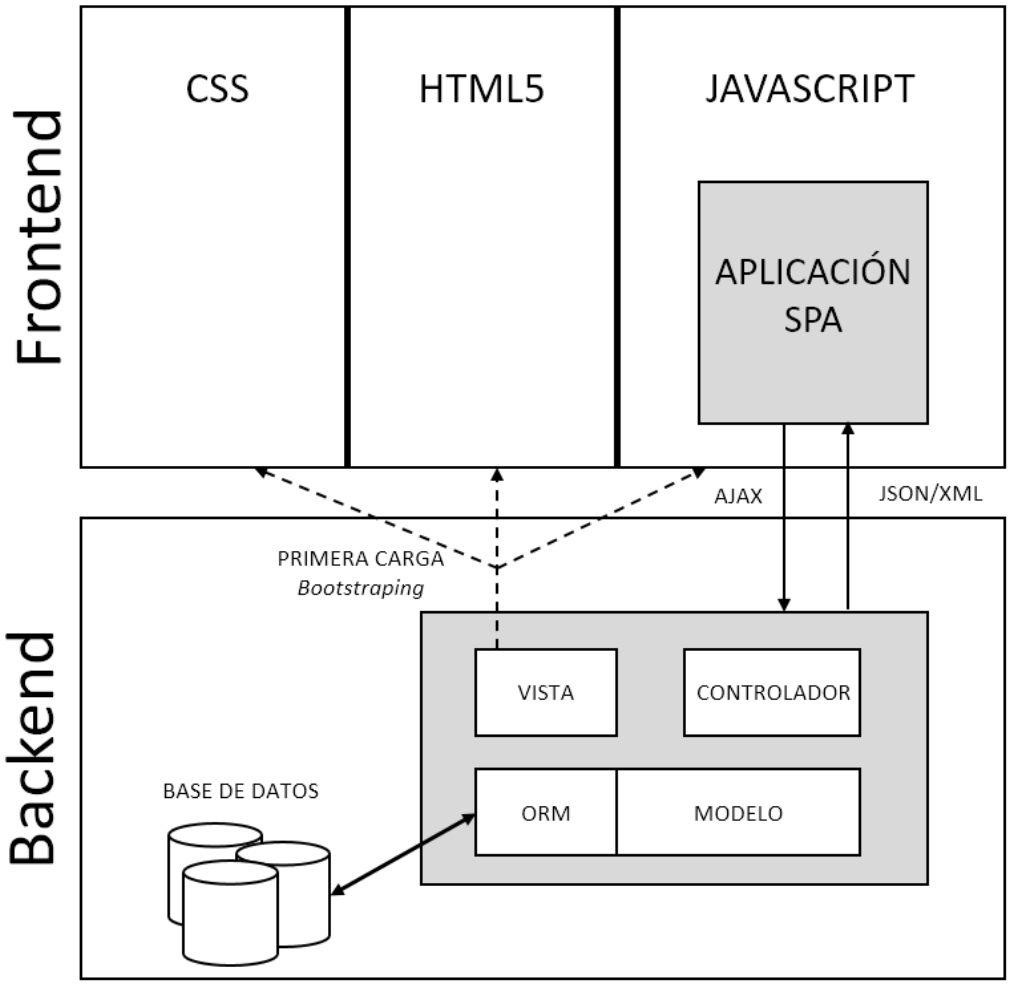


Figura 2.4: Arquitectura de una aplicación web que consta de un *Web API* y una *SPA*.

2.3. Tecnologías

2.3.1. Lenguajes y Frameworks

Un lenguaje de programación es un vocabulario con un conjunto de reglas gramaticales que le indican a un computador realizar una tarea en específico. También podríamos definirlo como aquel lenguaje que permite especificar de manera precisa sobre que datos debe operar una computadora, como deben ser almacenados o transmitidos y que acciones debe tomar bajo una variada gama de circunstancias.

La descripción de un lenguaje de programación usualmente se divide en dos com-

ponentes: la sintaxis (la forma en que se escribe) y la semántica (lo que significa).

Suelen clasificarse entre interpretados y compilados, donde los interpretados tienen un intérprete específico que obtiene como entrada un programa y ejecuta las acciones escritas a medida que las va procesando, mientras que los compilados tienen un compilador específico que obtiene como entrada un programa y traduce las instrucciones las cuales pueden servir de entrada para otro intérprete o compilador.

Una de las ventajas a la hora de desarrollar una aplicación web es la gran cantidad de lenguajes que se pueden utilizar. Muchos de estos lenguajes son de propósito general, lo cual significa que están diseñados para escribir una gran cantidad de tipos de aplicaciones, que pueden ser aplicaciones de escritorio, sistemas manejadores de base de datos, cálculos matemáticos y estadísticos, diseño de imágenes, aplicaciones web y hasta un mismo sistema operativo.

Actualmente los lenguajes más populares para la realización de nuevas aplicaciones web recae en el conjunto de los lenguajes que son interpretados y usualmente usados para crear *scripts*. Esto se debe a su simplicidad sintáctica, son débilmente tipeados y orientado a objetos, lo cual hace que los desarrolladores no tengan que conocer muy bien el lenguaje y se encarguen únicamente de atacar la lógica de negocio de una aplicación en particular.

Es muy común que, en un universo de aplicaciones donde la lógica de negocio de cada una de ellas no tienen ninguna relación, se encuentren operaciones similares que se repiten una y otra vez. Por lo cual, nacen diversa cantidad de bibliotecas que solventan necesidades específicas y se encapsulan en un sólo ambiente de desarrollo. Ese conjunto de bibliotecas, reciben el nombre de *frameworks*.

2.3.1.1. Ruby

Ruby es en lenguaje de programación interpretado, reflexivo y orientado a objetos. Posee una gestor de paquetes y bibliotecas llamado RubyGems, donde cada uno de estos paquetes es denominado *Gema* (o *gem* en inglés).

Algunas de las características de este lenguaje son:

- Orientado a objetos.
- Cuatro (4) niveles de ámbito de variable: global, clase, instancia y local.
- Manejo de excepciones.

- Iteradores.
- Expresiones regulares nativas.
- Sobrecarga de operadores.
- Altamente portable.

2.3.1.2. Ruby on Rails

Ruby on Rails es un framework de desarrollo de aplicaciones web [4], código abierto escrito en el lenguaje de programación Ruby. Sigue el patrón Modelo Vista Controlador (*MVC*). Ruby on Rails se distribuye a través del manejador de paquetes RubyGems.

Es considerado un framework altamente escalable por la gran cantidad de soluciones que posee. Viene incluido con un ORM llamado ActiveRecord, que se basa en el patrón de diseño con el mismo nombre. Otro de sus módulos más importantes son: Action Mailer, que sirve para enviar correos electrónicos y Active Resource, el cual proporciona una infraestructura necesaria para crear de manera rápida y sencilla recursos *Rest*.

2.3.1.3. Sinatra.rb

Sinatra.rb es otro de los frameworks más usados en el lenguaje Ruby. Su diferencia más notable con respecto a Ruby on Rails, es que Sinatra no sigue el patrón *MVC*, en su lugar, Sinatra se enfoca de proveer una mínimas abstracción al desarrollador de las partes fundamentes de una comunicación *HTTP*, la *Petición* y la *Respuesta*, de esta manera se pueden crear aplicaciones web en Ruby con el mínimo esfuerzo.

2.3.1.4. Python

Python es un lenguaje de programación dinámico e interpretado [7].

- Su código fuente no declara los tipos de datos (variables, parámetros o métodos).
- Python obtiene el tipo de dato en tiempo de ejecución.
- Se elimina el tiempo de compilación.

Es un lenguaje multiparadigma, es decir, Python se adapta al programador:

- Paradigma funcional

- Paradigma imperativo.
- Paradigma orientado a objetos.

2.3.1.5. Django

Django es un framework de desarrollo web de código abierto, escrito en el lenguaje Python, que, al igual que Ruby on Rails, sigue el patrón de diseño *MVC*. Django pone énfasis en el reuso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio *No te repitas* (DRY, del inglés *Don't Repeat Yourself*).

2.3.1.6. Flask

Flask es un framework web minimalista escrito en Python y basado en *Werkzeug*. Se puede relacionar perfectamente con Sinatra.rb ya que ambos son frameworks que no obligan al usuario a usar una herramienta o biblioteca particular, sólo provee abstracciones usando objetos que representan la *Petición* y *Respuesta HTTP*.

2.3.1.7. Javascript

Javascript es un lenguaje de programación interpretado, basado en el estándar ECMAScript [8]. Javascript se define como:

- Orientado a objetos.
- Basado en prototipos.
- Imperativo.
- Débilmente tipeado.
- Dińamico.

Se utiliza principalmente en su forma del *lado del cliente*, implementado en la mayoría de los navegadores web modernos permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe también una forma de poder ejecutar Javascript del *lado del servidor*.

2.3.1.8. Node.js

Node.js es una plataforma construida sobre el motor V8 de Google [9] que sirve para desarrollar aplicaciones de lado del servidor, aunque no se limita a eso [10].

Las aplicaciones en Node.js son escritas en el lenguaje de programación Javascript, y pueden ser ejecutadas sobre el ambiente de ejecución de Node.js en distintas plataformas como Mac OS X, Microsoft Windows, Linux y SunOS.

Node.js funciona como un contenedor del motor V8 de Google, que lo optimiza para trabajar mejor en contextos distintos a un navegador web, y además provee una serie de APIs optimizados para ciertos casos de uso. A su vez, Node.js funciona en un ambiente totalmente asíncrono basado en eventos y no bloqueante, lo cual permite realizar muchas operaciones de entrada/salida con un costo mínimo de tiempo, comportamiento usual en aplicaciones web.

Es importante mencionar que Node.js no es un framework de desarrollo web. Node.js es una plataforma que permite la ejecución de código Javascript fuera de un navegador que está optimizado para realizar y recibir peticiones *HTTP*.

Node.js incorpora varios "módulos básicos" compilados en el propio binario, como por ejemplo el módulo de red, que proporciona una capa para programación de red asíncrona y otros módulos fundamentales, como por ejemplo *Path*, *FileSystem*, *Buffer*, *Timers* y el de propósito más general *Stream* [11]. Es posible utilizar módulos desarrollados por terceros, ya sea como archivos ".node" precompilados, o como archivos en Javascript plano. Los módulos Javascript se implementan siguiendo la especificación *CommonJS* para módulos, utilizando una variable de exportación para dar a estos scripts acceso a funciones y variables implementadas por los módulos.

Los módulos de terceros pueden extender Node.js o añadir un nivel de abstracción, implementando varias utilidades *middleware* para utilizar en aplicaciones web, como por ejemplo los frameworks Connect y Express. Pese a que los módulos pueden instalarse como archivos simples, normalmente se instalan utilizando el Node Package Manager (*npm*) que facilita al desarrollador, la compilación, instalación y actualización de módulos así como la gestión de las dependencias.

2.3.1.9. Express.js

Express.js es un framework de desarrollo para aplicaciones web usando Node.js. Está basado en un framework más viejo llamado Connect.js. Es importante destacar que Express.js más que un framework es un microframework muy ligero. "Micro" no significa que la aplicación web tiene que encajar en un único archivo de Javascript, aunque ciertamente puede. Tampoco significa que Express.js carece de funcionalidad. El "micro" en microframework significa que Express.js tiene como objetivo mantener el núcleo simple pero extensible. Este no toma muchas decisiones por el

desarrollador, tales como la base de datos a utilizar. Esas decisiones que si hace por el desarrollador, son fáciles de editar. Todo lo demás depende de que necesite el desarrollador, de modo que Express.js puede ser todo lo que necesites y nada que no desees.

2.3.1.10. Sails.js

Sails.js es un framework web que trabaja sobre Node.js, más específicamente, sobre Express.js, usando las primitivas y abstracciones elementales que este provee.

Es un framework actualmente muy nuevo y en constante desarrollo. Unas de las características más fundamentales son que implementa el patrón *MVC* y viene incluido con un *ORM* llamado *Waterline*, el cual ha ganado su popularidad por la gran semejanza a ActiveRecord de Ruby on Rails.

2.3.1.11. Angular.js

Angular.js es un framework de desarrollo de aplicaciones web que usa tecnologías del lado del cliente (HTML, CSS, Javascript). Es un framework desarrollado y mantenido por Google. Es importante destacar de acuerdo a Brad Green y Shyam Seshadri (2013) que Angular.js se basa en el esquema de una aplicación de una sola página, donde básicamente nunca se hace una carga completa de la página sino de secciones específicas usando técnicas como *AJAX*. Para el desarrollo se basa en el patrón Modelo-Vista-Controlador.

2.3.2. Base de datos

Una base de datos es un repositorio de información que contiene datos que se relacionan bajo alguna temática o lógica. Es una colección de datos organizados, que pueden ser esquemas, tablas, consultas, reportes, vistas y otro tipos de objetos. Toda base de datos necesita un sistema gestor que sirva de puente entre la base de datos en sí y algún programa de alto nivel o a un usuario que normalmente se le llama "administrador de base de datos". Este sistema recibe el nombre de *Sistema Manejador de Base de Datos*. Los más conocidos son *MySQL*, *PostgreSQL*, *Oracle*, *MariaDB* y *MongoDB*.

2.3.3. Pilas de desarrollo web

Una pila de desarrollo es un conjunto de componentes de software que son necesarios para crear una plataforma completa. Luego se tiene que, las aplicaciones "corren en" o "corren sobre" dichas plataformas.

Por ejemplo, en el caso de una aplicación web, el desarrollador arquitecto del proyecto define una pila de desarrollo que puede consistir en:

- Sistema operativo.
- Servidor web.
- Base de datos.
- Lenguaje de programación.

2.3.3.1. *Linux Apache MySQL PHP*

Linux Apache MySQL PHP (*LAMP*) es una de las pilas de desarrollo más usadas en el mundo web a lo largo de los años. Su nombre es acrónimo de los nombres de sus componentes que son:

- Linux: El sistema operativo.
- Apache: El servidor web.
- MySQL: Sistema manejador de base de datos relacional.
- PHP: Lenguaje de programación.

Pueden existir ligeros variantes de esta pila, por ejemplo en el caso de *WAMP*, donde el sistema operativo es *Microsoft Windows*. También pueden haber cambios en el lenguaje de programación utilizado, como es el uso de *Perl* o *Python* y también cambios a nivel de base de datos, pueden ser usados *MariaDB* o *MongoDB*.

2.3.3.2. *MongoDB Express.js Angular.js Node.js*

Mongo Express.js Angular.js Node.js (*MEAN*) es otra de las pilas de desarrollo web con mucho interés actualmente, es más novedosa que *LAMP* debido a el poco tiempo que tienen sus componentes en el mercado. Una de las características que más resalta de *MEAN* es que todos los componentes comparten el lenguaje de programación Javascript y el lenguaje de comunicación *JSON* (*Javascript Notation Object*).

Sus componentes son los siguientes:

- MongoDB: Una base de datos NoSQL.
- Express.js: Framework minimalista para aplicaciones web.
- Angular.js: Framework MVC para el desarrollo de *SPAs*.
- Node.js: Plataforma para realizar aplicaciones en lado del servidor.

3

Marco aplicativo

3.1. Metodología de desarrollo de software: SCRUM

SCRUM es una metodología de desarrollo de software ágil el cual se caracteriza por ser una metodología de adaptación, iterativa, rápida, flexible y eficaz, diseñada para ofrecer un valor significativo de forma rápida en todo el proyecto [12].

Una fortaleza clave de *SCRUM* radica en el uso de equipos multi-funcionales, auto-organizados, y con poder que dividen su trabajo en ciclos de trabajo cortos y concentrados llamados *Sprints*. Un *Sprint* suele durar entre una y seis semanas. Durante un *Sprint* se llevan a cabo reuniones cortas y muy concretas donde los miembros del equipo de desarrollo discuten progresos diarios.

En líneas generales SCRUM se puede conceptualizar en cuatro distintas partes bien diferenciadas. Ellas son:

- Roles
- Reuniones
- Artefactos
- *Sprints*

Básicamente en un equipo de desarrollo *SCRUM* sus miembros asumen roles, que determinan sus funciones, derechos y responsabilidades. Los miembros a su vez asisten a reuniones constantes donde se generan artefactos para planificar una iteración quincenal o mensual (*Sprint*), cuyo principal resultado es un prototipo de software funcional. Las iteraciones se repiten sucesivamente hasta lograr desarrollar un producto con las características deseadas por el cliente.

3.1.1. Roles

La cantidad de personas de un equipo organizado bajo la metodología de desarrollo de software *SCRUM* depende netamente de la complejidad del proyecto a desarrollar o específicamente de los *Sprints* que se definan de una determinada iteración. Sin embargo el tamaño óptimo del equipo de desarrollo es lo suficientemente grande para asegurar habilidades adecuadas, pero lo suficientemente pequeño como para colaborar fácilmente. En el Equipo de Desarrollo no se asignan los cargos típicos contemplados tradicionalmente por la ingeniería de software, tales como programador, diseñador, arquitecto de software, *tester*, etc. En su lugar el Equipo de Desarrollo actúa colectivamente para ejecutar el volumen de trabajo que se han comprometido voluntariamente a completar durante una iteración. La idea es crear en el equipo la sensación de que todos sus miembros se encuentran juntos en el proyecto, en lugar de que cada individuo trabaje de forma aislada rindiendo cuentas a una autoridad superior. Los roles definidos por *SCRUM* son:

3.1.1.1. El Dueño del Producto

El Dueño del Producto es responsable de asegurar una comunicación clara sobre el producto y los requisitos de funcionalidad de servicios con el Equipo de Desarrollo, al igual que definir el criterio de aceptación, y de asegurar que se cumpla. En otras palabras, el El Dueño del Producto es responsable de asegurar que el Equipo de Desarrollo ofrezca valor. El Dueño del Producto siempre debe mantener una visión dual. Debe entender y apoyar las necesidades e intereses de todos los clientes, mientras que comprenden las necesidades y el funcionamiento del Equipo de Desarrollo.

3.1.1.2. El Maestro SCRUM

Es un facilitador que asegura que el Equipo de Desarrollo esté dotado de un ambiente propicio para completar el proyecto con éxito. Para lograr este propósito el Maestro *SCRUM*, ayudará al equipo a seguir la metodología *SCRUM* realizando actividades como organizar las reuniones requeridas, asegurar el buen estado de

los artefactos generados, servir de intermediario entre el Dueño del Producto y los miembros del equipo, entre otras actividades. El Maestro *SCRUM* puede ser un gerente de proyectos, pero también puede seDesarrollor un miembro experimentado de algún Equipo de Desarrollo anterior.

3.1.1.3. Equipo de Desarrollo

Es el grupo o equipo de personas responsables de la comprensión de los requisitos especificados por el Dueño del Producto y de la creación de los entregables del proyecto. Los miembros del equipo suelen ser programadores, administradores de sistemas, diseñadores, *testers*, etc. Cada miembro del equipo se compromete con sus compañeros a asumir la carga de trabajo que el mismo considere capaz de realizar. Los miembros del equipo son los que mejor conocen sus propias habilidades, por tanto son ellos directamente los que se asignan las cargas de trabajo voluntariamente. Es fundamental que el equipo de trabajo conozca muy bien sus capacidades, o que el Maestro *SCRUM* los ayude a determinarlas.

3.1.1.4. Terceras Partes

Son agentes que aunque no intervienen directamente en el proceso de desarrollo del producto, se ven afectados por el éxito del desarrollo del mismo. Estos agentes pueden ser gerentes, auditores, representantes de otras organizaciones, etc. Básicamente la idea es que las terceras partes al no estar comprometidas en el desarrollo, ellas solamente están asociadas, no tienen derecho alguno sobre el proceso de desarrollo. Sin embargo las terceras partes pueden asistir como oyentes a las reuniones *SCRUM* planificadas, y pueden emitir sugerencias pero no pueden tomar decisiones.

3.1.2. Reuniones

3.1.2.1. Reunión de Planificación *Sprint*

Es una reunión donde asisten el Dueño del Producto, el Maestro SCRUM, y el equipo de Desarrollo completo. Las terceras partes también pueden asistir, pero no es obligatorio. La Reunión de Planificación se realiza antes del inicio de cada *Sprint*. Durante la reunión de la planificación el Dueño del Producto describe las características generales que desea que el producto posea en orden de prioridad (Registro del Producto). Los miembros del equipo deben hacer la suficiente cantidad de preguntas como para que una lista detallada de tareas (Registro del *Sprint*) pueda ser elaborada a partir del Registro del Producto. No es requerido que el Dueño del Producto especifique exhaustivamente todas las características deseadas en el producto

a partir de el primer *Sprint*, ya que en los *Sprint* siguientes se puede modificar el Registro del Producto . Una vez establecido el Registro del Producto los miembros del equipo de Desarrollo seleccionan aquellas características que se implementarán en el *Sprint* por comenzar. El equipo de Desarrollo también se debe poner de acuerdo con el Dueño del Producto para establecer una meta general para el Sprint. La meta general del *Sprint* debe ser breve y auto-descriptiva, por ejemplo: Implementar un carrito de compras persistente entre sesiones con su respectivo *CRUD* de artículos. Los artefactos generados por la Reunión de Planificación son: la meta general del *Sprint* y el Registro del *Sprint*. El Registro del *Sprint* estará compuesto a su vez por un conjunto de Historias de Usuario.

3.1.2.2. Reunión Diaria

La reunión diaria Sprint es llevada a cabo por todos los miembros del equipo, el Maestro *SCRUM* y el Dueño del Producto. También pueden asistir terceros, pero estos al no estar comprometidos sino solamente involucrados con el desarrollo usualmente sólo pueden asistir como oyentes, de esta forma también se disemina la información de forma rápida y poco costosa. Usualmente la reunión diaria es llevada a cabo en la mañana antes de cada jornada de trabajo y no debe durar más de 15 minutos. Su carácter es informal y sus asistentes deben ser breves y concisos.

3.1.2.3. Reunión de Revisión del *Sprint*

La reunión de Revisión del Sprint, es el resultado natural del mismo. Una vez concluido el Sprint los miembros del equipo preparan una presentación de sus resultados. Los resultados deben reflejar que la meta general del Sprint fue alcanzada. De no ser este el caso es responsabilidad del Maestro *SCRUM* analizar las razones por las cuales no se cumplió la meta al concluir el Sprint. Esta reunión es de carácter informal, no debe emplearse más de dos horas para su elaboración y no se deben utilizar láminas u otro soporte.

3.1.2.4. Reunión Retrospectiva

La Reunión Retrospectiva es llevada a cabo por los miembros del equipo, el Dueño del Producto y el Maestro *SCRUM*. Esta reunión puede tener carácter privado si así se desea. La Reunión Retrospectiva es llevada a cabo usualmente inmediatamente después de concluida una Reunión de Revisión *Sprint*. Se recomienda que no se emplee más de una hora para su realización. Sin embargo su duración puede extenderse si así se cree necesario. Durante la Reunión Retrospectiva el equipo reflexiona acerca de los resultados obtenidos en el Sprint concluido con la finalidad de proponer

mejores formas de mejorar su rendimiento y motivación. La dinámica de la reunión se puede llevar de varias maneras pero usualmente el Maestro *SCRUM* le pregunta a cada miembro del equipo qué se debe empezar a hacer, qué se de debe dejar de hacer, y qué se debe continuar haciendo. Es una buena práctica someter a votación las mejoras propuestas por todos los miembros del equipo de Desarrollo para determinar cuales se llevarán a cabo durante el próximo *Sprint*. Luego de concluido el *Sprint* se verificarán los resultados arrojados por las mejoras tomadas en la Reunión Retrospectiva anterior.

3.1.3. Artefactos

3.1.3.1. Registro del Producto

El Registro del Producto es una lista de características ordenadas por prioridad. Cada entrada de la lista contiene un ítem con una descripción de las funcionalidades deseadas en el producto. El Registro del Producto se elabora mediante la realización de una tormenta de ideas entre el Dueño del Producto y el Equipo de Trabajo. Ambas partes escriben lo que desean del producto de forma flexible y natural. Esto permite que la primera iteración pueda ser iniciada sin tener que invertir gran cantidad de tiempo y esfuerzo en documentación previa. Los ítems representados en las entradas de la lista pueden ser: características, errores, trabajo técnico (actualización de plataforma de trabajo, o de versión de bibliotecas, por ejemplo) y adquisición de conocimiento (aprender a manejar sistemas de control de versiones distribuidos, manejar determinado framework de desarrollo web, etc).

3.1.3.2. Meta *Sprint*

Una Meta *Sprint* es una descripción breve que representa lo que el equipo de desarrollo desea lograr durante una iteración. La Meta *Sprint* es redactada de forma colaborativa entre el Dueño del Producto y el equipo de desarrollo. La Meta *Sprint* puede ser utilizada como una descripción resumida de las actividades que está realizando el equipo de desarrollo. Terceras partes pueden ser informadas rápidamente de las actividades del grupo de Desarrollo mediante la Meta *Sprint*, sin tener que describirse todos los detalles. El éxito de la iteración se evaluará contra la Meta *Sprint*, en lugar de contra cada uno de los ítems específicos seleccionados en el Registro del Producto.

3.1.3.3. Registro del *Sprint*

El Registro del Sprint consiste en una lista de tareas identificadas por el el Equipo de Trabajo durante la planificación del Sprint. Durante la planificación del Sprint el equipo selecciona una cantidad de características a desarrollar del Registro del Producto, expresadas en la forma de Historias de Usuario, e identifica las tareas necesarias para completarlas. El registro del Sprint puede ser actualizado con una frecuencia diaria por los miembros del equipo a medida que se vayan desarrollando las características, pero se debe tratar de minimizar las veces que se actualiza el registro durante un día.

3.1.3.4. Lista de Impedimentos

La Lista de Impedimentos es la lista donde el Maestro *SCRUM* lleva un seguimiento de todos los bloqueos, impedimentos, inconvenientes, y decisiones pendientes que pudieran representar una amenaza al buen desarrollo del proyecto. El Maestro *SCRUM* deberá actualizar esta lista diariamente y es su responsabilidad tratar de identificar y resolver todos los problemas que surjan durante el desarrollo.

3.1.3.5. Incremento

El incremento es el prototipo funcional o demo que se planteó desarrollar en la Meta Sprint. El mismo puede ser una nueva versión del producto o una funcionalidad específica lista para ser entregada, totalmente probada y documentada.

3.1.3.6. Retroalimentación Visual

Se trata de todos los artefactos generados durante el proceso de desarrollo que son necesarios para implementar el producto. En esta categoría se agrupan los diagramas entidad-relación, los diagramas físicos de Bases de Datos, flujo-gramas, diagramas arquitectónicos, diagrama de objetos de dominio, de clases, Casos de Uso, etc.

3.1.4. *Sprints*

Los procesos Sprint consisten en la realización de un conjunto de actividades (reuniones, tareas, resolución de impedimentos, etc) llevadas a cabo por agentes que asumen roles. Dichas actividades generan artefactos necesarios para desarrollar el incremento del Sprint.

3.1.5. Modificación de la Metodología de Desarrollo de Software

Debido a que la Metodología de Desarrollo de Software *SCRUM* no especifica con precisión que artefactos deben ser generados en cada iteración, y a que la naturaleza del desarrollo está enmarcada dentro de un contexto completamente académico se hace necesario definir una Metodología de Desarrollo de Software Ágil que permita establecer la planificación de las actividades y artefactos necesarios para lograr el desarrollo de la nueva versión de *Portalasig*. Para ello se toman en cuenta aspectos considerados por *SCRUM*, pero también se simplifican otros aspectos y se incluyen nuevos. Los principales cambios en la Metodología empleado se describen a continuación.

3.1.5.1. Roles

Los roles del Dueño del Producto y el Matestro *SCRUM* para el desarrollo del presente TEG. son ejercidos por el tutor. El Equipo de Desarrollo está compuesto por un solo miembro, el estudiante desarrollando el TEG. No se consideran terceras partes involucradas.

3.1.5.2. Artefactos

El principal artefacto que se tomará de *SCRUM* es el Registro del Producto ya que el mismo plantea los requerimientos funcionales a desarrollar necesarios para completar los objetivos específicos planteados en la siguiente sección (3.2) . También se fijará una meta *Sprint* por cada iteración la cual consistirá en una descripción breve de la meta a lograr en cada iteración.

Número de *Sprint*	Tareas	Duración (semanas)
1	Migrar datos de usuarios	1
2	Realizar Web API	4
3	Documentar Web API	1
4	Realizar SPA	4

Cuadro 3.1: Tabla de Control de Iteración

3.1.5.3. Reuniones

De las reuniones planteadas por *SCRUM* sólo se llevaran a cabo la Reunión de Planificación *Sprint* y la Reunión de Revisión *Sprint* (en la misma se puede realizar

la evaluación retrospectiva). Debido a que el equipo de trabajo está compuesto solamente por un integrante se descartan las reuniones diarias. Además, la Reunión de Planificación *Sprint* de la siguiente iteración, puede realizarse inmediatamente después de concluida la Reunión de Revisión *Sprint* de la iteración anterior.

3.2. Descripción del proyecto

A continuación se describen las cualidades que hacen que esta versión de *Portalasig* se diferencie con la versión actual en producción. Estas diferencias se separan en dos (2) secciones las cuales son: Innovaciones y Tecnologías.

3.2.1. Innovaciones

Las Innovaciones describen todos las funcionalidades nuevas y cambiadas con respecto a la versión anterior de *Portalasig*.

3.2.1.1. Nuevas

- Diseño *responsive*: El nuevo portal tendrá un diseño *responsive*, el cual se basa en que el sitio web se vea de manera adecuada en dispositivos con pantallas de diferentes tamaños, ya sea desktops, laptops, tablets o smartphones. Como paradigma de diseño se usará Mobile first (Móvil primero) que consiste en implementar las interfaces desde los dispositivos con pantallas más pequeñas primero para luego ir diseñando los dispositivos con el resto de dimensiones de pantallas.
- Panel de materias de interés al usuario: A la hora de que un usuario se autentique de manera exitosa, la interfaz de la página principal cambiará ligeramente mostrando un panel donde se encuentren las materias asociadas a ese usuario en particular en el semestre actual en curso. En el caso de ser docente, se mostraran las materias el cual él mismo dicta y en el caso de un estudiante, las materias el cual él está cursando como también materias esté ejerciendo funciones como preparador, si es el caso.
- Vídeos de *Youtube* en la sección de Descargas: En la sección de Descargas de un sitio web, ahora se podrán subir los *URLs* asociados a un vídeo de *Youtube*, para así fomentar el uso de esta funcionalidad y evitar la subida de vídeos al sistema que pueden agotar rápidamente la capacidad de almacenamiento del mismo.

- Noticias a través de *Twitter*: Será posible asociar las Noticias de un sitio web con el *WidgetId* de una cuenta de *Twitter*.
- Preferencias del usuario: El usuario ahora contará con preferencias personales con respecto al sistema de envío de correos, donde será posible seleccionar que servicios quiere que el sistema le notifique a través de un correo electrónico, como por ejemplo una nueva noticia, un foro nuevo creado, entre otras. Las notificaciones de modificación de una calificación para un estudiante no podrán ser deshabilitadas, ya que se considera muy importante que el estudiante se entere de manera correcta y rápida que un docente o preparador modificó alguna de sus notas.
- Nuevo rol en el sistema: Coordinador: Actualmente, cualquier docente agregado a un sitio web puede realizar cualquier tipo de cambio (Contenido temático, bibliografías, horarios, etc). Esto será cambiado añadiendo un nuevo rol al sistema: Coordinador. Cualquier docente que creé un nuevo sitio web de alguna asignatura, automáticamente será el Coordinador de dicho sitio, el cual tendrá funcionalidades únicas sobre los demás docentes agregados al sitio web.

3.2.1.2. Cambiadas

- Sección de carga usando HTML5 Multipart-Upload: El sistema de carga de archivos actual presenta problemas al momento de querer subir archivos con un peso aproximado mayor de 2 MB, presentando una respuesta de error por parte del servidor. La solución propuesta, será utilizar la tecnología Multiplart-upload, que consiste en separar un archivos en pequeños trozos, de esta forma subir cada uno de los trozos por separado. Se pretende usar la biblioteca Flow.js ajustada como módulo de Angular.js y de Node.js. El tamaño máximo de cada archivo individual será de 50 MB.
- Esquema de la base de datos: La nueva base de datos se basó en la ya existente, pero con algunos cambios relevantes. La ya existente posee una mezcla de inglés y español, esto se modificó para que se use como estándar sólo el inglés, se agregaron tablas necesarias para poder manejar sesiones y administrar diversas funcionalidades como las preferencias del usuario. También se descartaron algunas relaciones cíclicas encontradas (por ejemplo, entre la tabla evento y planificación) y la tabla mención, ya que es un portal de asignaturas para toda la facultad y hay carreras que no presentan esta característica.
- Paginación en el módulo de administración: El rol de Administrador es el único capaz de listar y editar asignaturas, docentes y estudiantes en el sistema. En la versión anterior de *Portalasig*, a la hora de listar alguno de los recursos anteriormente mencionados, son traídos del servidor en una sola petición lo cual torna una interacción lenta con la interfaz de usuario, ya que actualmente

se cuenta con aproximadamente dos mil quinientos (2500) estudiantes y ciento cincuenta (150) docentes registrados en el sistema.

- Ingreso y modificación de calificaciones con una apariencia parecida a *Microsoft Excel*: Se utilizará un módulo de *AngularJS* llamado *UI-Grid* [13], el cual simula fácilmente el estilo de una hoja de calculo, parecidas a las encontradas en programas como *Microsoft Excel*.

3.2.2. Tecnologías

Para la realización de este Trabajo Especial de Grado se planean utilizar la siguiente pila de desarrollo web, la cual es una ligera modificación de la pila *MEAN* descrita anteriormente. *Figura* 3.1:

- Node.js: La plataforma asincrónica que permite la ejecución de Javascript fuera del navegador.
- Express.js: Módulo y framework que corre sobre Node.js. Provee soluciones y abstracciones con objetos basados en la Petición y Respuesta HTTP, como también la creación de rutas para la implementación de un Web API sencillo.
- Bookshelf.js: Módulo y ORM de Node.js. Se integra muy fácilmente a Express.js. Provee compatibilidad con sistemas manejadores de base de datos como MySQL y PostgreSQL.
- PostgreSQL: Sistema de manejador de base de datos relacional.
- Angular.js: Framework de desarrollo de *frontend*, escrito en Javascript y es usado para crear aplicaciones de una sola página (*SPA*).
- Bootstrap: Framework de desarrollo de *frontend*, es una combinación de archivos en Javascript y CSS, y es usado para realizar diseños y estructuras de una página web de manera rápida y sencilla.

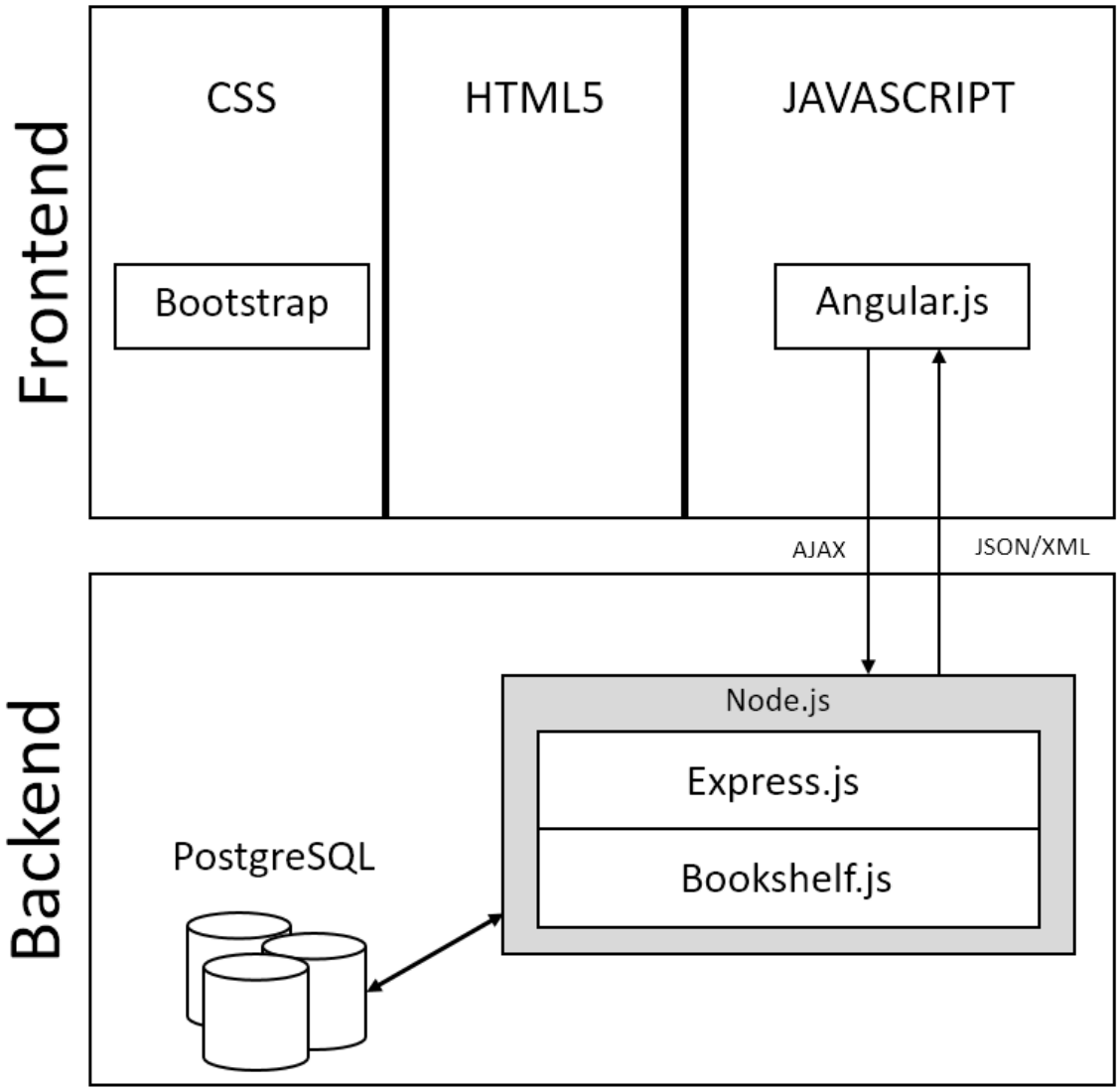


Figura 3.1: Arquitectura propuesta para la nueva versión de *Portalasig*.

3.3. Sprint 1: Migración

Meta *Sprint*: Realizar la migración de información de la base de datos de la versión anterior de *Portalasig*.

El objetivo de este *Sprint* del proyecto se basó en restaurar cierta información de la versión anterior de *Portalasig*. Específicamente se restauraron las siguientes entidades y sus relaciones (*figura*3.2):

- Usuario
- Administrador
- Estudiante
- Docente
- Asignatura
- Carrera
- Clasificación

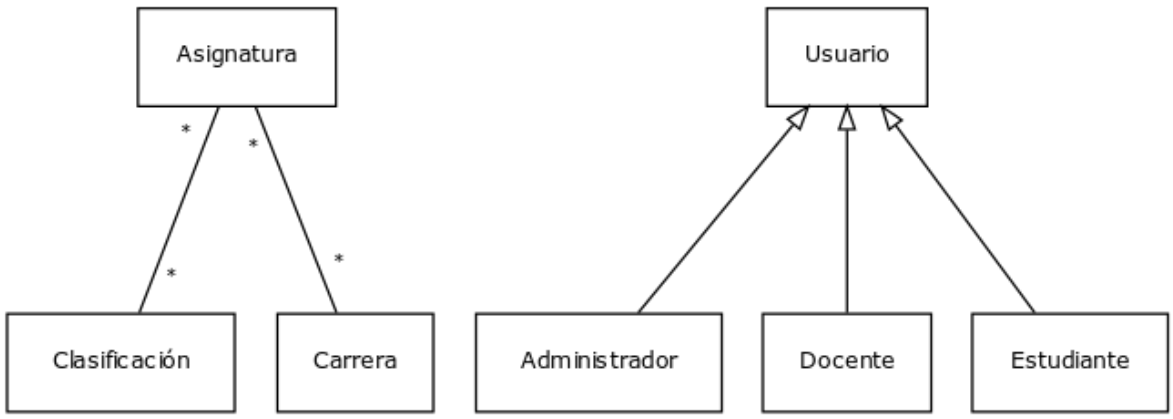


Figura 3.2: Diagrama enditad-relación: Entidades restauradas de la base de datos de la versión anterior de *Portalasig*.

Dado que la base de datos utilizada en la versión anterior del portal es *MySQL*[1] y la versión realizada bajo este TEG se utilizó *PostgreSQL*, la migración no se pudo lograr de manera directa (realizar un *dump* en las tablas de *MySQL* para luego restaurarlas en *PostgreSQL*). Esto se debe a que ambos sistemas manejadores de base de datos utilizan pequeños cambios en la sintaxis de SQL, particularmente en el uso de *comillas*, lo cual hace imposible realizar dicho proceso de manera directa.

Para solventar esto de una manera eficiente, se desarrolló un *script* escrito en JavaScript que utiliza el ORM anteriormente mencionado, llamado *Waterline* [14]. *Waterline* es un ORM muy nuevo, el cual presenta ciertas limitaciones con respecto a otros ORM, como *Bookshelf.js*, por ejemplo. Pero en este caso, *Waterline* fue utilizado únicamente para, de manera programática, conectarse con *MySQL* y *PostgreSQL*, y luego realizar un proceso migratorio.

Las instancias, tanto en la base de datos de la versión anterior como la de la nueva versión de *Portalasig*, fueron descritos como modelos del ORM *Waterline*:

```
var Waterline = require('waterline');

var Usuario = Waterline.Collection.extend({

  identity: 'usuario',
  connection: 'OldDB', //MySQL
  tableName: 'usuario',

  attributes: {
    id: {
      type: 'integer',
      primaryKey: true
    },
    cedula: 'integer',
    clave: 'string',
    nombre: 'string',
    apellido: 'string',
    correo: 'string',
    activo: 'integer',
    created_at: 'datetime',
    updated_at: 'datetime'
  }
});

module.exports = Usuario;
```

Código 3.1: Definición del modelo Usuario

```
var Waterline = require('waterline');

var User = Waterline.Collection.extend({

  identity: 'user',
  connection: 'NewDB', //PostgreSQL
  tableName: 'user',

  attributes: {
    id: {
      type: 'integer',
      primaryKey: true
    },
    identity_card: 'integer',
    password: 'string',
    name: 'string',
    lastname: 'string',
    email: 'string',
    active: 'boolean',
    created_at: 'datetime',
    updated_at: 'datetime'
  }
});

```

```
25    module.exports = User;
```

Código 3.2: Definición del modelo *User*

Luego, todos los modelos fueron cargados en el *script* principal.

```
1     var orm = new Waterline();
2
3     //Loading old db collections
4     orm.loadCollection(Usuario);
5     orm.loadCollection(Asignatura);
6     orm.loadCollection(Docente);
7     orm.loadCollection(Estudiante);
8     orm.loadCollection(Carrera);
9     orm.loadCollection(AsignaturaCarrera);
10    orm.loadCollection(Clasificacion);
11    orm.loadCollection(AsignaturaClasificacion);
12    orm.loadCollection(Administrador);
13    //Loading new db collections
14    orm.loadCollection(User);
15    orm.loadCollection(Course);
16    orm.loadCollection(Professor);
17    orm.loadCollection(Student);
18    orm.loadCollection(Career);
19    orm.loadCollection(CourseCareer);
20    orm.loadCollection(Classification);
21    orm.loadCollection(CourseClassification);
22    orm.loadCollection(Administrator);
```

Código 3.3: Carga de todos los modelos en el ORM *Waterline*

Para que finalmente, por cada par de modelos, correspondientes entre ambas base de datos, se realizará la migración de *MySQL* a *PostgreSQL*.

```
1     //Migrating users
2     models.collections.usuario.find().exec(function(err, usuarios) {
3
4       async.each(usuarios, function(usuario, callback) {
5         models.collections.user.create({
6           identity_card: usuario.cedula,
7           password: usuario.clave,
8           name: usuario.nombre,
9           lastname: usuario.apellido,
10          email: usuario.correo,
11          active: usuario.activo == 0 ? false : true,
12          created_at: usuario.created_at,
13          updated_at: usuario.updated_at
14        }, function(err, user) {
15          if(err)
16            console.log("Users ERR: " + err);
17          models.collections.profile.create({
18            id: user.id
19          }, function(err, profile) {
```

```
20              if(err)
21                console.log("Profiles ERR: " + err);
22
23              callback();
24            });
25          });
26
27        });
28
29        callback(null, 'Users done');
30
31      });
```

Código 3.4: Migración del modelo Usuario

3.4. Sprint 2: Desarrollo del backend

Meta *Sprint*: Desarrollar el Web API (*backend*) de *Portalasig*.

Después de realizar la migración de ciertas entidades de la versión anterior de *Portalasig*, ya era posible comenzar a desarrollar toda la lógica de *backend* en un Web API *Restful*. Como fue descrito en la propuesta de este TEG, las plataforma web utilizada fue Node.js, la cual consiste en un ambiente de ejecución asíncrono, donde se utiliza JavaScript como lenguaje de programación y se pueden integrar módulos que ayudan a resolver necesidades de manera más rápida para el desarrollador.

3.4.1. Módulos

Para el desarrollo del *backend*, se utilizaron varios módulos que resolvían desde necesidades muy puntuales como módulos que daban una forma particular de cómo escribir el código del Web API, por ejemplo, *node-xlsx* y *express* respectivamente.

Como fue mencionado anteriormente, la manera de integrar los módulos y utilizarlos en Node.js es a través de su manejador de paquetes llamado *npm*. Este manejador consta de un repositorio en linea donde los desarrolladores suben módulos, públicos o privados, donde otros desarrolladores pueden descargarlos y utilizarlos para sus fines. Al final del desarrollo del Web API se contabilizaron más de quince (15) módulos, pero los más utilizados fueron los siguientes:

- Express.js
- Knex.js

- Bookshelf
- Bluebird
- Nodemailer
- Cron
- Connect-Multiparty
- Excel-xlsx y excel4node

A continuación de describirán brevemente los módulos anteriormente mencionados y explicará cómo y porqué fueron utilizados para la realización del Web API de la nueva versión de *Portalasig*

3.4.1.1. Express.js

Como fue mencionado anteriormente, *Express.js* es un framework de desarrollo para aplicaciones web. En el caso particular del desarrollo del Web API de *Portalasig*, es el módulo más importante, ya que es el encargado de anclar la aplicación en la red, a través de HTTP. Una cualidad muy importante también, es que permite al desarrollador definir las rutas (*end-points*) las cuales el API escuchará para dar función de servicio de algún requerimiento.

3.4.1.2. Knex.js

Knex.js [15] es un módulo que se encarga básicamente de construir *queries* SQL a través de código JavaScript. También permite la creación de esquemas (migraciones), transacciones y *seeds* (datos de prueba) y es compatible con manejadores de base de datos relacionales, como *MySQL*, *PostgreSQL*, *MariaDB* y *Oracle*.

En el caso del desarrollo del *backend* de la nueva versión *Portalasig*, *Knex.js* se utilizo para la creación del esquema relacional de la base de datos utilizada. Antes de poder realizar cualquier operación, primero se debe especificar como *Knex.js* se va a conectar con la base de datos a utilizar, para eso se debe ejecutar el siguiente comando:

```
$ knex init
```

El cual creará un archivo llamado *knexfile.js* donde se especifican las configuraciones a utilizar con la base de datos. En el caso de *Portalasig*, que se utilizo *PostgreSQL*, el archivo tiene la siguiente apariencia:

```
//knexfile.js

module.exports = {

  development: {
    client: 'postgresql',
    connection: {
      database: 'portalasig',
      user:     'portalasig',
      password: 'password',
      host: 'localhost',
      charset: 'utf8'
    },
    migrations: {
      tableName: 'portalasig_migrations'
    }
  },

  production: {
    client: 'postgresql',
    connection: {
      database: 'portalasig',
      user:     'portalasig',
      password: 'password',
      host: 'localhost',
      charset: 'utf8'
    },
    pool: {
      min: 2,
      max: 10
    },
    migrations: {
      tableName: 'portalasig_migrations'
    }
  }

};
```

Código 3.5: Archivo de configuración de *Knex.js*

Este archivo, principalmente se encarga de definir la base de datos a utilizar y las credenciales a utilizar para poder conectarse a dicha base de datos, como: el nombre de la base de datos y el usuario y clave para conectarse.

3.4.1.3. Bookshelf

Bookshelf.js [16] es un ORM escrito totalmente en JavaScript que utiliza a *Knex.js* para construir *queries* y permite realizar asociaciones uno-a-uno, uno-a-muchos y

muchos-a-muchos como tambien operaciones *CRUD* (Crear, Leer, Actualizar y Eliminar) de una manera muy sencilla.

Bookshelf.js fue esencial a la hora de definir todos los modelos que conforman el portal. Estos modelos se relacionan directamente con tablas definidas, por migraciones usando *Knex.js*, en la base de datos. Por ejemplo, la definición del modelo *User* (usuario) fue de la siguiente manera:

```
1   // ../models/user.js
2
3   var Professor = null,
4       Student = null,
5       Administrator = null,
6       Comment = null,
7       Forum = null,
8       Log = null,
9       LogSite = null,
10      Session = null,
11      Profile = null,
12      Site = null;
13
14  module.exports = function(bookshelf) {
15    var User = bookshelf.Model.extend({
16      tableName: 'user',
17      hasTimestamps: ['created_at', 'updated_at'],
18      constructor: function() {
19        bookshelf.Model.apply(this, arguments);
20        this.on('destroying',
21        function(model, options) {
22
23          return model.load([
24            'comment',
25            'forum',
26            'log',
27            'logSite',
28            'sessions'
29          ], options)
30          .then(function(model) {
31            var promises = [];
32
33            promises.push(model.related('comment')
34            .invokeThen('destroy', options));
35            promises.push(model.related('forum')
36            .invokeThen('destroy', options));
37            promises.push(model.related('log')
38            .invokeThen('destroy', options));
39            promises.push(model.related('logSite')
40            .invokeThen('destroy', options));
41            promises.push(model.related('sessions')
42            .invokeThen('destroy', options));
```

```
43
44              return Promise.all(promises).then(function() {
45              })
46              .catch(options.transacting.rollback);
47            })
48            .catch(options.transacting.rollback);
49          });
50        },
51        validation: {
52          'name': [
53            { validator: 'notEmpty',
54            message: 'name is required' }
55          ],
56          'lastname': [
57            { validator: 'notEmpty',
58            message: 'lastname is required' }
59          ],
60          'email': [
61            { validator: 'notEmpty',
62            message: 'email is required' },
63            { validator: 'isEmail',
64            message: 'not an email' }
65          ],
66          'password': [
67            { validator: 'notEmpty',
68            message: 'password is required' }
69          ],
70          'identity_card': [
71            { validator: 'notEmpty',
72            message: 'identity_card is required' },
73            { validator: 'isInt',
74            message: 'identity_card must be integer' }
75          ]
76        },
77        professor: function() {
78          return this.hasOne(Professor, 'id');
79        },
80        student: function() {
81          return this.hasOne(Student, 'id');
82        },
83        administrator: function() {
84          return this.hasOne(Administrator, 'id');
85        },
86        comment: function() {
87          return this.hasMany(Comment);
88        },
89        forum: function() {
90          return this.hasMany(Forum);
91        },
92        log: function() {
93          return this.hasMany(Log);
94        },
```

```
95          logSite: function() {
96            return this.hasMany(LogSite);
97          },
98          sessions: function() {
99            return this.hasMany(Session);
100         },
101         profile: function() {
102           return this.hasOne(Profile, 'id');
103         }
104       },
105       {
106         associate: function(models) {
107           Professor = models.Professor;
108           Student = models.Student;
109           Administrator = models.Administrator;
110           Comment = models.Comment;
111           Forum = models.Forum;
112           Log = models.Log;
113           LogSite = models.LogSite;
114           Session = models.Session;
115           Profile = models.Profile;
116           Site = models.Site;
117         }
118       });
119
120       return User;
121     };
```

Código 3.6: Archivo de especificación del modelo *User*

A continuación se describirá el significado de ciertas porciones del código anterior.

- Líneas 3-12: Se especifican los modelos los cuales el modelo actual (*User*) se asocia.
- Línea 16: Se especifica cuál tabla de la base de datos el modelo actual representa.
- Líneas 18-50: Se especifica las acciones a tomar cuando el modelo está siendo destruido, en este caso particular se están eliminando las relaciones del modelo a eliminar. Esta tarea es comúnmente llamada *cascade delete* en un contexto de base de datos.
- Líneas 51-76: Se describen las validaciones a realizar al momento de crear o actualizar una instancia y su respectivo mensaje de error en caso de que la velicación falle. Estas validaciones se realizaron gracias a un módulo adicional llamado *Bookshelf-Validator*.
- Líneas 77-118: Se especifica qué tipo de asociación presenta el modelo actual con los modelos descritos en las líneas 3-12.

3.4.1.4. Nodemailer

La versión anterior de *Portalasig* contiene un módulo de enviar correos a los usuarios dependiendo de ciertas eventualidades que ocurran en la misma. Por ejemplo, cuando un usuario era agregado a un sitio web en particular, éste era notificado a través de su correo electrónico asociado con un mensaje cordial de bienvenida.

Para poder realizar esta funcionalidad en la nueva versión del portal, se utilizó el módulo *Nodemailer* [17] el cual permite de una manera muy sencilla enviar correos electrónicos en la plataforma Node.js.

Este módulo se configura de una muy similar a *Knex.js*, simplemente un archivo JavaScript con una serie de credenciales a utilizar:

```
1   //mailer.js
2
3   module.exports = {
4     development: {
5       service: 'Gmail',
6       auth: {
7         user: 'portalasig.prueba@gmail.com',
8         pass: 'password'
9       }
10    },
11    production: {
12      host: 'strix.ciens.ucv.ve',
13      port: 465,
14      secure: true,
15      authMethod: 'LOGIN',
16      auth: {
17        user: 'usuario@ciens.ucv.ve',
18        pass: 'password'
19      },
20      tls: {
21        rejectUnauthorized: false
22      }
23    }
24  };
```

Código 3.7: Archivo de configuración de *Nodemailer*

3.4.1.5. Cron

Cron es un módulo que permite definir funciones en el *backend* que se ejecutaran dado un intervalo de tiempo.

En *Portalasig* se modelan Archivos (*Files*) y Entregables (*AssignmentFiles*). Los primeros, son los que representan los archivos descargables que un integrante del Grupo Docente de un sitio web puede subir al sistema, y luego los entragables son los archivos que los estudiantes, asociados a un entregable de un sitio, suben al portal para que posteriormente un Docente o un Preparador lo descargue para poder evaluarlos. Estos modelos representan directamente a un archivo binario que es guardado en el sistema. Cuando una instancia de estos modelos es eliminada por alguna razón (ya sea que el Estudiante haya sido eliminado del sistema, o el Coordinador haya eliminado un sitio web en particular) se eliminará también el archivo binario al cual ésta representa. Si por algún motivo se elimina la instancia del modelo pero no el archivo binario, este se quedaría utilizando espacio del sistema innecesariamente. Para esto, se creo una función que se ejecutará cinco (5) minutos después de cada media noche que se encargará de listar todos los archivos binarios guardados en el sistema y chequear cuales están referenciados en la base de datos (ya sea por ser un Descargable o un Entregable). Aquellos archivos que no estén referenciados serán eliminados.

La función que es ejecutada por el módulo *Cron* posee la siguiente apariencia:

```
1   new CronJob('5 0 * * *', () => {
2     var uploader = new flow();
3     fs.readdir('./public/uploads',(err, data) => {
4         if (err) console.log('Error: ' + err);
5         bluebird.map(data, function(url) {
6           return app.locals.models.File
7           .forge({
8             url: url,
9           })
10          .fetch()
11          .then(function(file) {
12            if(!file)
13              return app.locals.models.AssignmentFile
14              .forge({
15                url: url,
16              })
17              .fetch()
18              .then(function(assignment) {
19                if(!assignment)
20                  return uploader.delete(url).then(function(err) {
21                    if(err)
22                      console.log('error deleting file');
23                  });
24                return;
25              });
26            return;
27          });
28        }).then(function() {
29          console.log('done');
```

```
30            });
31
32      }); }, null, true, 'America/Caracas');
```

Código 3.8: Definición de un *CronJob* que elimina archivos no referenciados en la base de datos

3.4.1.6. Connect-Multiparty y Flow.js

Este módulo hace una tarea muy simple pero muy útil, se encarga de analizar y traducir (*parse*) las peticiones HTTP con *Content-type*: *multipart/form-data*, que también son conocidas como *file uploads* (subida de archivos).

Cabe destacar que los archivos llegaban separados en pequeños trozos o *chunks*, donde cada *chunk* llegaba al API por una petición distinta. Esto se debe a que en el *frontend* se utilizó un módulo, que será explicado más adelante, llamado *Flow.js* [18]. Dado que estos archivos llegaban en *chunks*, una vez que se alcanzaba el último trozo, se tenían unir nuevamente para formar el archivo original. Para esto se creó una clase en JavaScript que se encargaba, en parte, de recibir todos los *chunks* y mezclarlos.

3.4.1.7. Excel-xlsx y excel4node

En *Portalasig*, un Docente puede agregar estudiantes a través de una hoja de cálculo que es generada por el sistema de *Conest*. En esta hoja de cálculo se especifica la sección de una asignatura y los estudiantes que la conforman, dando sus cédulas, nombres, apellidos y correos electrónicos. Para *parsear* esta hoja de calculo se utilizo el módulo *excel-xlsx* el cual básicamente transforma un archivo valido de *Microsoft Excel* en una matriz (arreglo de arreglos en JavaScript representando filas y columnas de la hoja de cálculo) el cual el desarrollador fácilmente puede recorrer.

3.4.2. Roles

Como en la versión anterior de *Portalasig*, en esta versión se definieron roles y funciones que pueden ejercer estos roles. Se añadió un nuevo rol llamado Coordinador, el cual es un estado especial que puede tener un Docente en un sitio determinado. A continuación de describen los roles del sistema y sus funcionalidades.

3.4.2.1. Público general

Se define Público general a todos aquellos usuarios que ingresen en el sistema y no posean una sesión en el mismo. Las funcionalidades que estos individuos pueden ejercer son las siguientes:

- Observar la información pública de un sitio web, la cual es: Información General, Contenido Temático, Grupo Docente, Horarios, Evaluaciones, Planificación, Noticias y Descargas.
- Poder descargar los archivos y ver los vídeos en la sección de Descargas.

3.4.2.2. Estudiante

Es el tipo de usuario que representa a un Estudiante de la Facultad de Ciencias de la Universidad Central de Venezuela. Un Estudiante podrá:

- Realizar todas las acciones de Público General.
- Cambiar su correo electrónico asociado.
- Cambiar su contraseña.
- Cambiar las notificaciones de correo electrónico que desea recibir cuando esté asociado a un sitio web.
- Crear y participar en foros de un sitio web.

3.4.2.3. Preparador

Es un rol especial que podrá tener un Estudiante en un sitio web determinado. Un Preparador podrá:

- Realizar todas las acciones de Estudiante.
- Modificar calificaciones de Estudiantes agregados a un sitio web.
- Agregar archivos de descargas o vídeos en la sección de Descargas.
- Enviar correos electrónicos a los integrantes de un sitio web, ya sean Estudiantes, Preparadores y/o Docentes.

3.4.2.4. Docente

El Docente, al igual que el Estudiante, representa a un Docente de la Facultad de Ciencias. Los Docentes serán capaces de:

- Realizar todas las acciones de un Preparador.
- Crear una nueva asignatura en el sistema.
- Crear un nuevo sitio web.

3.4.2.5. Coordinador

El Coordinador es el rol nuevo en el sistema. Puede ser considerado como el ´´administrador" de un sitio web en particular, el cual podrá:

- Realizar todas las acciones de un Docente.
- Cambiar el Coordinador de un sitio web.
- Eliminar un sitio web.
- Cambiar la visualización de noticias a través de una cuenta de *Twitter*.
- Completar información pertinente a la asignatura, ya sea: información general,objetivos, bibliografía, contenido temático, grupo docente, horarios, estudiantes, evaluaciones y entregas.
- Agregar y eliminar a docentes, preparadores que pertenezcan al grupo docente de un sitio web de una asignatura.
- Poder observar el Registro de Actividades. El Registro de Actividades es donde se podrán ver los cambios realizados a un sitio web, específicamente: Cambios de calificaciones, cambio de entregas y cambios de evaluaciones.

3.4.2.6. Administrador

- Agregar, modificar o eliminar asignaturas.
- Agregar, modificar o eliminar estudiantes.
- Agregar, modificar o eliminar docentes.

3.5. Sprint 3: Documentación del backend

Meta *Sprint*: Realizar la documentación del Web API (*backend*) de *Portalasig*.

Al desarrollar un Web API, es importante también documentarlo, específicamente documentar los *endpoints* del mismo. Las documentación de los *endpoints* está relacionada directamente a la peticiones HTTP que el Web API puede recibir y cómo

éstas son respondidas. Esto ayuda a que en un posible futuro se puedan desarrollar diferentes aplicaciones *frontend* que utilicen como servicio al Web API de *Portalasig*, por ejemplo una aplicación nativa de *Android* o incluso una nueva versión de la aplicación web SPA.

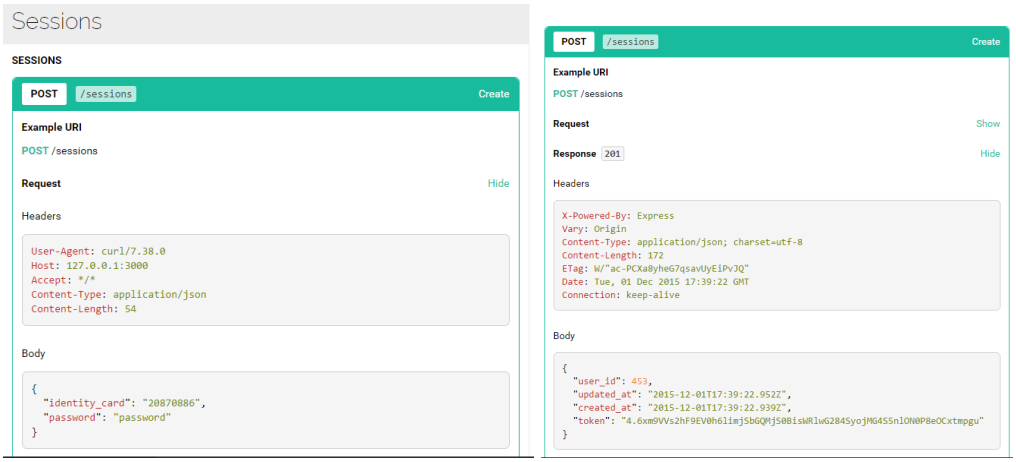
La documentación del *backend* de *Portalasig* se realizó usando el lenguaje de descripción de API´s llamado *Api Blueprint* [19], cuyo propósito es permitir a los desarrolladores documentar las subrutinas o funciones que un API determinado provee. El lenguaje de descripción de esta herramienta posee la extinción *.apib* y presenta la siguiente apariencia:

```
## sessions [/sessions]
### Create [POST]
+ Request
    + Headers

            User-Agent:curl/7.38.0
            Host:127.0.0.1:3000
            Content-Type:application/json
            Content-Length:54

    + Body

            { "identity_card": "20870886", "password": "password" }

+ Response 201
    + Headers

            X-Powered-By:Express
            Vary:Origin
            Content-Type:application/json; charset=utf-8
            Content-Length:172
            ETag:W/"ac-PCXa8yheG7qsavUyEiPvJQ"
            Date:Tue, 01 Dec 2015 17:39:22 GMT
            Connection:keep-alive

    + Body

            {"user_id":453,"updated_at":"2015-12-01T17:39:22.952Z",
            "created_at":"2015-12-01T17:39:22.939Z",
            "token":"4.6xm9VVs2hF9E"}
```

Código 3.9: Ejemplo de un archivo *.apib* de *Api Blueprint* para documentar Web APIs

Es importante destacar que los archivos *.apib* no son fácilmente entendibles para personas que no conozcan muy bien la herramienta *Api Blueprint*, es por eso que se utilizó una segunda herramienta llamada *Aglio*. La función principal de *Aglio* [20] es transformar archivos *.apib* a un archivo *.html* con estilos donde se describen de una manera más entendible los diferentes *endpoints* con sus respectivas peticiones

y respuestas. Entonces al final se puede llevar un archivo como el observado en el código fuente 3.9 a una página HTML como la siguiente:



(a) Petición (b) Respuesta

Figura 3.3: Ejemplo de petición/respuesta HTTP documentada usando *Aglio*

3.6. Sprint 4: Realización del frontend

Meta *Sprint*: Desarrollar la Aplicación Web SPA (*frontend*) de *Portalasig*.

Finalmente, el último componente a realizar fue la Aplicación Web SPA. Como fue descrito en la propuesta, fue utilizado el framework *Angular.js* para generar el comportamiento de una SPA (*Single Page Application*) y el framework de diseño web *Bootstrap* para realizar la interfaces de una manera rápida y sencilla.

Gracias a *Angular.js*, *Portalasig* pasó de ser un sitio web en donde cada página diferente consistía en un archivo HTML diferente, a un sitio web donde sólo una primera petición (*bootstraping*) era necesaria para cargar en el cliente (navegador web) todos los componentes de interfaz, permitiendo así que la navegación posterior se actualizara las interfaces sin necesidad de recargar completamente la página web.

Para el desarrollo de interfaces se usó un método llamado *Mobile First*, el cual consiste en diseñar las interfaces vistas desde dispositivos móviles primero y luego ir diseñándolas para dispositivos de pantallas de mayor tamaño.

3.6.1. Interfaces

A continuación se mostraran las diferentes interfaces de las secciones del nuevo portal y con sus respectivas funcionalidades. Estas interfaces se dividirán en dos (2) modalidades, la versión móvil y la versión para dispositivos de pantallas de mayor tamaño, por ejemplo *laptops* o computadoras personales.

3.6.1.1. Barra de navegación

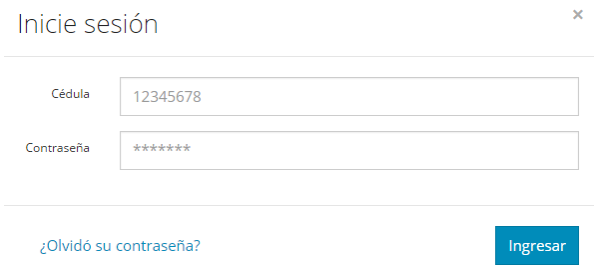
La barra de navegación es una secciones de la aplicación web que nunca cambian. Su principal funcionalidad, como su nombre lo indica, es poder ´´navegar" por distintas páginas que un sitio web provee de una manera sencilla. En el caso de *Portalasig*, el diseño de la barra de navegación es minimalista. En el lado izquierdo se mostrará un *link* para viajar a la página principal del portal y en la parte derecha, un campo de texto que servirá para buscar cualquier asignatura registrada en el sistema y un botón para poder iniciar sesión, en el caso de que no exista una 3.4(a). En el caso de la versión móvil *responsive* no se mostrará la barra de búsqueda de asignaturas y el botón de inicio de sesión podrá se accedido a través de un menú desplegable mostrado al lado derecho de la barra de navegación 3.4(b).

(a) Versión *desktop*

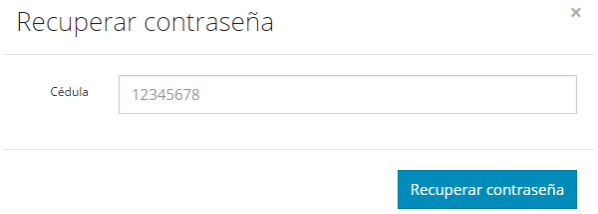
(b) Versión móvil

Figura 3.4: Barra de navegación sin sesión activa

Si el usuario hace *click* en el botón de ´´Iniciar sesión", aparecerá una modal solicitando la cédula y la contraseña 3.5(a). También será notable un *link* para poder recuperar la contraseña, en caso de que al usuario se le haya olvidado. En este caso, aparecerá otra modal solicitando únicamente la cédula, para que posteriormente el sistema le envíe un correo electrónico de recuperación de contraseña al usuario 3.5(b).



(a) Inicio de sesión



(b) Recuperación de contraseña

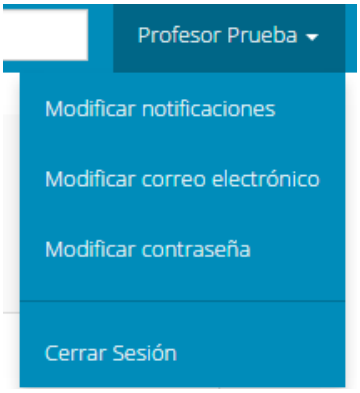
Figura 3.5: Inicio de sesión y recuperación de contraseña

Cuando existe una sesión activa la barra de navegación cambiará ligeramente. En la versión *desktop* de mostrará el nombre de la persona dueña de la sesión al lado derecho 3.6(a), el cual si es presionado aparecerán cuatro (4) opciones 3.6(d): Modificar notificaciones 3.7(a), modificar correo electrónico 3.7(b), modificar contraseña 3.7(c) y cerrar sesión. El lado izquierdo se mantendrá igual si y sólo si la persona no es Administrador del portal 3.6(b) o es Coordinador 3.6(c) de un sitio web activo. En el caso de la versión móvil, las opciones anteriormente mencionadas serán accedidas al presionar el menú ubicado al lado derecho de la barra de navegación.

(a) Nombre del dueño de la sesión

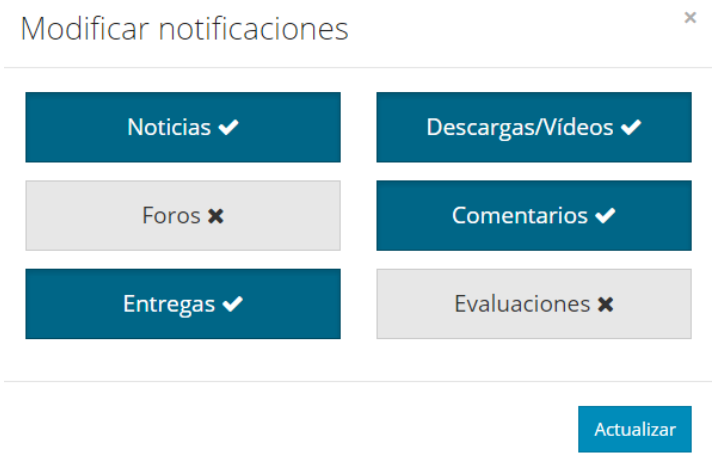
(b) Sesión activa de un Administrador

(c) Sesión activa de un Coordinador de un sitio web



(d) Opciones de sesión

Figura 3.6: Barra de navegación con sesión activa



(a) Cambio de notificaciones

(b) Cambio de correo electrónico (c) Cambio de contraseña

Figura 3.7: Opciones de la barra de navegación con sesión activa

3.6.1.2. Inicio

El Inicio consiste en la sección principal de *Portalasig*, aquella que es cargada cuando se accede desde el URL *http://www.ciens.ucv.ve/portalasig2*. Los componentes de interfaz presentes en esta sección se mantuvieron iguales a la versión anterior del portal de asignaturas. Se encuentran dos (2) bloques principales. El izquierdo presentando las licenciaturas presentes en la Facultad de Ciencias de la Universidad Central de Venezuela, y uno derecho donde se muestran las asignaturas dictadas en la licenciatura seleccionada del bloque anterior. Estas asignaturas son agrupadas por Obligatorias y Electivas 3.8.

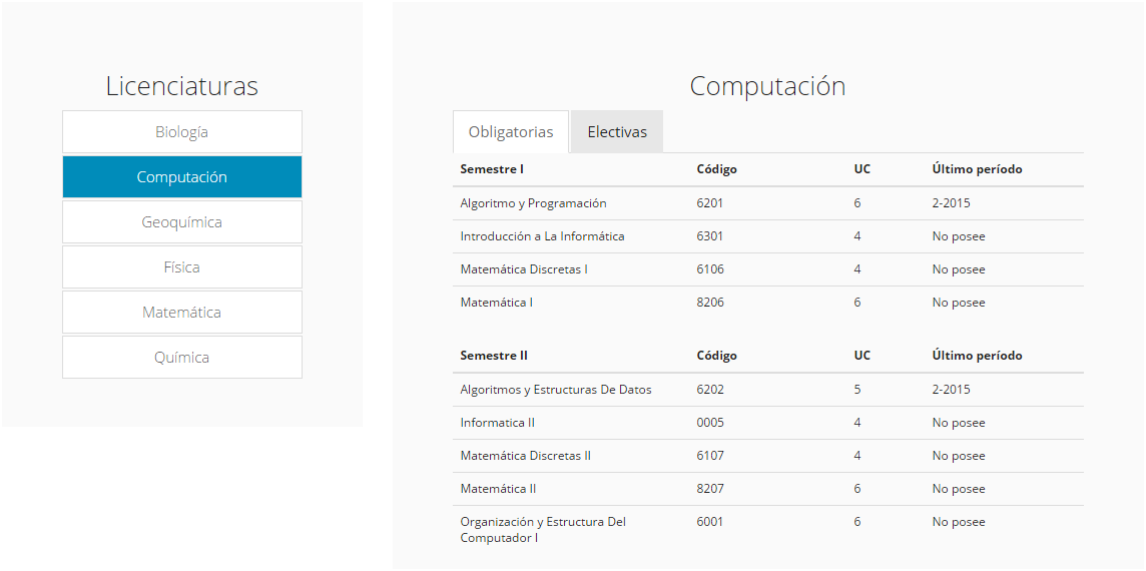
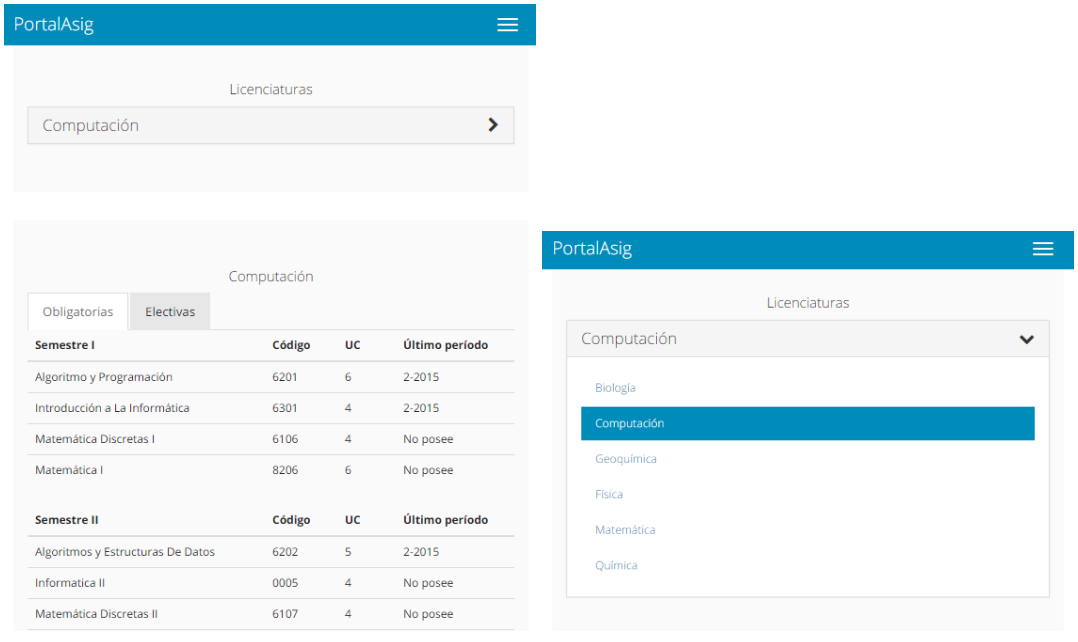


Figura 3.8: Inicio de *Portalasig*.

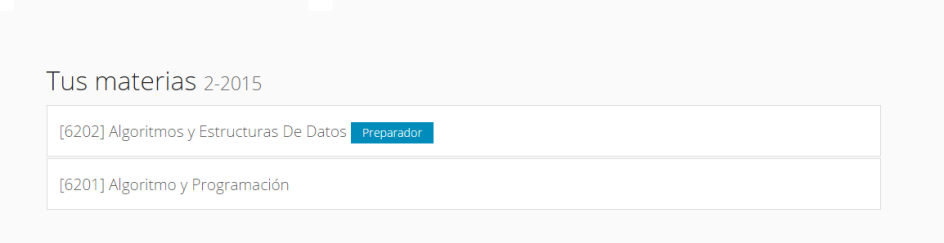
Los bloques anteriores (Licenciaturas y Asignaturas) cambiaran en la versión móvil. Ahora estarán uno encima de otro 3.9(a), siendo el bloque de Licenciaturas de primero, mostrado como una lista desplegable 3.9(b).



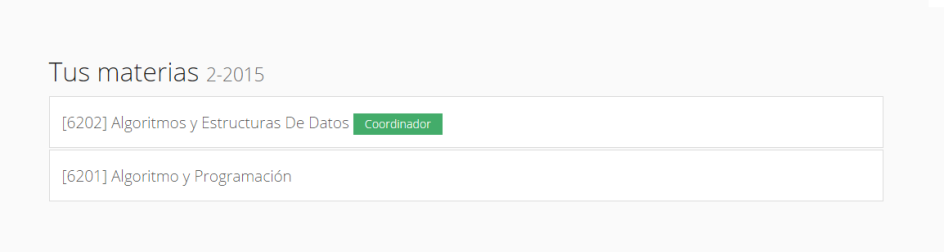
(a) Licenciaturas y Asignaturas (b) Lista de Licenciaturas

Figura 3.9: Inicio de *Portalasig* versión móvil

La interfaz de la sección Inicio cambia cuando existe una sesión activa. Aparecerá un bloque nuevo donde se verán las asignaturas las cuales estén relacionadas con el dueño de la sesión en el semestre en curso 3.10(a) y 3.10(b).



(a) Estudiante

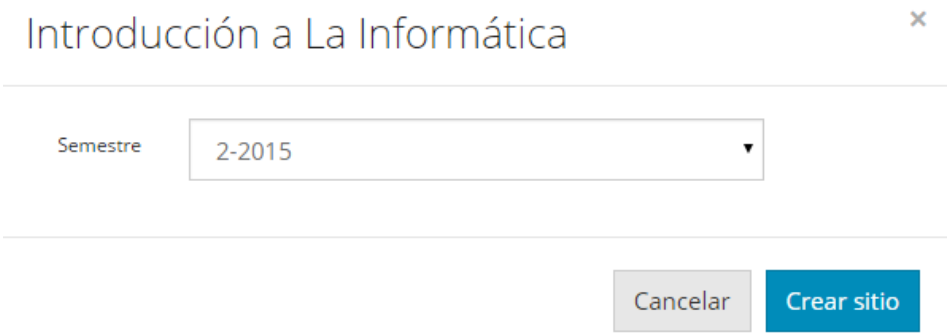


(b) Docente

Figura 3.10: Asignaturas relacionadas con el dueño de la sesión

También habrá un cambio de la lista de asignaturas del bloque derecho, aparecerá una nueva columna con la opción de poder crear un nuevo sitio web de una asignatura determinada 3.11(a). Al hacer *click* en "Crear nuevo sitio", aparecerá una ventana modal donde se podrá elegir el semestre deseado del sitio web a crear 3.11(b).

(a) Opción de "Crear nuevo sitio" presente en la lista de asignaturas



(b) Modal de creación de un nuevo sitio web

Figura 3.11: Creación de un nuevo sitio web

3.6.1.3. Asignatura

La sección de Asignaturas consiste en aquellas interfaces relacionadas a un sitio web seleccionado, o a que es lo mismo, a la instancia de una asignatura dado un semestre. La estructura del URL de un sitio web será:

http://www.ciens.ucv.ve/portalasig2/#/asignatura
/<codigo_asignatura>/<semestre>

Por ejemplo, el URL del sitio web de la asignatura Introducción a la Informática en el semestre *2-2015* sería:

`http://www.ciens.ucv.ve/portalasig2/#``/asignatura/6301/2-2015`

Cuando un Docente decide crear un sitio web, se le enviará a la primera interfaz del mismo y automáticamente se convierte en el Coordinador. Dicha interfaz tendrá la siguiente apariencia en la versión *desktop* 3.12.

Figura 3.12: Interfaz principal de un sitio web.

Y en la versión móvil presentará la siguiente interfaz 3.13

Figura 3.13: Interfaz principal de un sitio web versión móvil.

Como es notable en la figura 3.12, un sitio web de una asignatura posee múltiples secciones, las cuales se mostraran a continuación, especificando cuales roles pueden interactuar con las mismas. Los roles presentes en sistema (Administrador, Coordinador, Docente, Estudiante y Público General) se agruparan de la siguiente manera:

Roles	Grupos
Administrador, Coordinador, Docente, Preparador, Estudiante y Público General	Todos
Coordinador, Docente, Preparador y Estudiante	Miembros
Coordinador, Docente y Preparador	Grupo Docente
Coordinador y Docente	Docentes

Cuadro 3.2: Agrupación de roles del sistema.

- Información General: La Información General de un sitio web se encuentra divida en tres (3) secciones: Información General, Objetivos y Bibliografía. Por defecto la Información General, mostrará la información cargada por el usuario que cree la asignatura. Permisología cuadro: 3.3.

Roles que pueden acceder	Roles que pueden modificar
Todos	Coordinador

Cuadro 3.3: Roles que interactúan con Información General de un sitio web.

- Contenido Temático: Existirá también una sección de Contenidos Temáticos, donde se mostraran los contenidos a ser dictados en la asignatura. Permisología cuadro: 3.4.

Roles que pueden acceder	Roles que pueden modificar
Todos	Coordinador

Cuadro 3.4: Roles que interactúan con Contenido Temático de un sitio web.

- Grupo Docente: La sección de Grupo Docente mostrará todos los Docentes y Estudiantes con rol de Preparador (figura 3.14) encargados de dictar las clases. Permisología cuadro: 3.5.

Roles que pueden acceder	Roles que pueden modificar
Todos	Coordinador

Cuadro 3.5: Roles que interactúan con Grupo Docente de un sitio web.

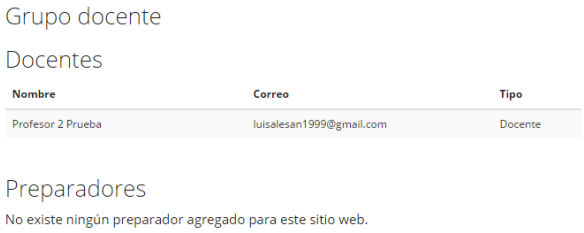


Figura 3.14: Grupo Docente de un sitio web.

- Horarios: En Horarios se mostrarán los diferentes horarios que provee la asignatura, asociando a los integrantes del grupo docente que dictaran clases en los mismos (figura 3.15). Permisología cuadro: 3.6.

Roles que pueden acceder	Roles que pueden modificar
Todos	Coordinador

Cuadro 3.6: Roles que interactúan con Horarios de un sitio web.

Figura 3.15: Horarios de un sitio web.

- Estudiantes: Todos los estudiantes asociados a un sitio web serán mostrados en esta sección. También se podrán agregar nuevos estudiantes, ya sea ingresando uno por uno o agregando una sección completa a través de una lista de estudiantes generada por el sistema *Conest* (figura 3.16). Permisología cuadro: 3.7.

Roles que pueden acceder	Roles que pueden modificar
Grupo Docente	Coordinador

Cuadro 3.7: Roles que interactúan con Estudiantes de un sitio web.

Figura 3.16: Estudiantes asociados a un sitio web.

- Evaluaciones: Se mostrarán todas las evaluaciones que los Docentes asociados hayan agregado al sistema (figura 3.17). Permisología cuadro: 3.8.

Roles que pueden acceder	Roles que pueden modificar
Todos	Docentes

Cuadro 3.8: Roles que interactúan con Evaluaciones de un sitio web.

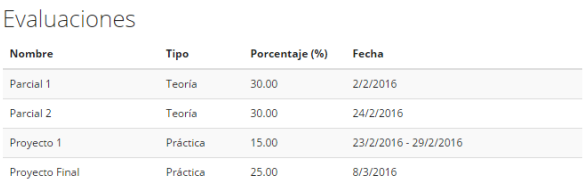


Figura 3.17: Evaluaciones de un sitio web.

- Entregas: En la sección de Entregas se listaran todas las entregas o asignaciones que hayan sido agregadas por el Grupo Docente de un sitio (figura 3.18). Las fechas de entrega que sean de color *verde* aun estarán abiertas para que los Estudiantes puedan subir archivos, en caso contrario la fecha se tornará de color *rojo*. Al hacer *click* en alguna Entrega, se desplegará una modal. Esta modal mostrará una lista de todos los entregables de los Estudiantes, en caso de que el dueño de la sesión sea parte del Grupo Docente (figura 3.19). En caso de ser un Estudiante, la modal dará la opción de subir el entregable asociado (figura 3.20). Permisología cuadro: 3.9.

Roles que pueden acceder	Roles que pueden modificar
Miembros	Grupo Docente

Cuadro 3.9: Roles que interactúan con Entregas de un sitio web.

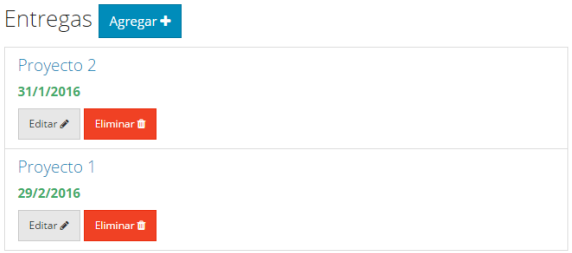


Figura 3.18: Entregas de un sitio web.

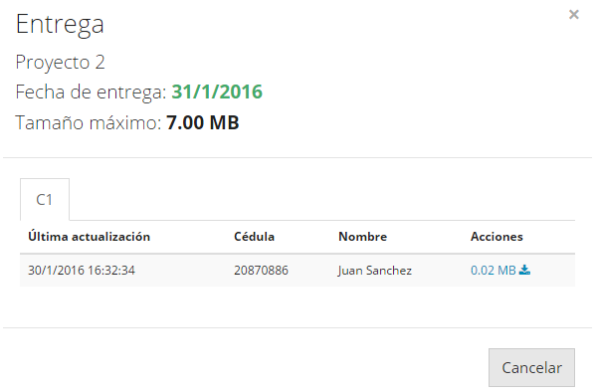


Figura 3.19: Lista de Entregas vista por el Grupo Docente.

Figura 3.20: Modal de subida de una nueva entrega (Estudiante).

- Planificación: Esta sección consistirá en un calendario donde se mostraran todas las Entregas, Evaluaciones y Eventos creados en el sitio web (figura 3.21). Permisología cuadro: 3.10.

Roles que pueden acceder	Roles que pueden modificar
Todos	Grupo Docente

Cuadro 3.10: Roles que interactúan con Planificación de un sitio web.

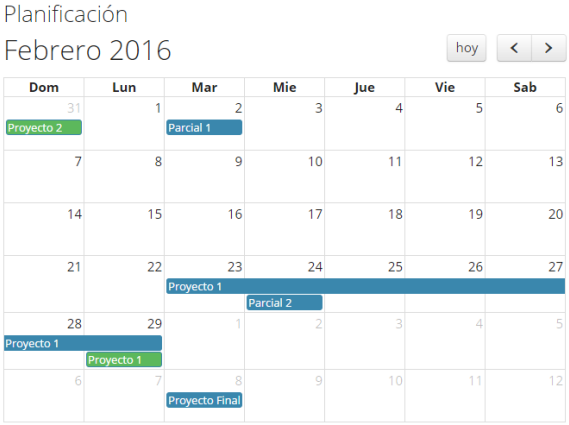


Figura 3.21: Planificación de un sitio web.

- Calificaciones: La apariencia de esta sección dependerá de la persona dueña de la sesión. Si es parte del Grupo Docente, de mostraran una lista de todas las

evaluaciones. Al hacer *click* en alguna Evaluación, de desplegará la lista de todos lo Estudiantes donde se podrá editar su calificación para dicha evaluación (figura 3.22). En caso de ser un Estudiante el dueño de la sesión, se mostrarán las evaluaciones con su calificación respectiva para dicho Estudiante (figura 3.23). Permisología cuadro: 3.11.

Roles que pueden acceder	Roles que pueden modificar
Miembros	Grupo Docente

Cuadro 3.11: Roles que interactúan con Calificaciones de un sitio web.

Figura 3.22: Calificaciones de todos los estudiantes asociados a un sitio web (Grupo Docente).

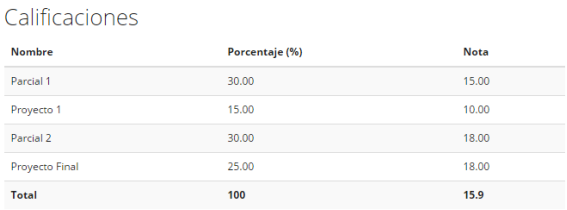


Figura 3.23: Calificaciones de un estudiante en un sitio web.

- Noticias: Aquí se podrán observar todas las noticias o informaciones relevantes agregadas por el Grupo Docente de la asignatura (figura 3.24). Permisología cuadro: 3.12.

Roles que pueden acceder	Roles que pueden modificar
Todos	Grupo Docente

Cuadro 3.12: Roles que interactúan con Noticias de un sitio web.

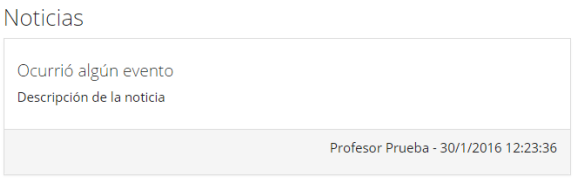


Figura 3.24: Noticias de un sitio web.

- Foros: En la sección de Foros se podrán crear salas de discusiones donde los integrantes del sitio web podrán comunicarse (figuras 3.25 y 3.26). Permisología cuadro: 3.13.

Roles que pueden acceder	Roles que pueden modificar
Miembros	Grupo Docente

Cuadro 3.13: Roles que interactúan con Foros de un sitio web.

Figura 3.25: Foros de un sitio web.

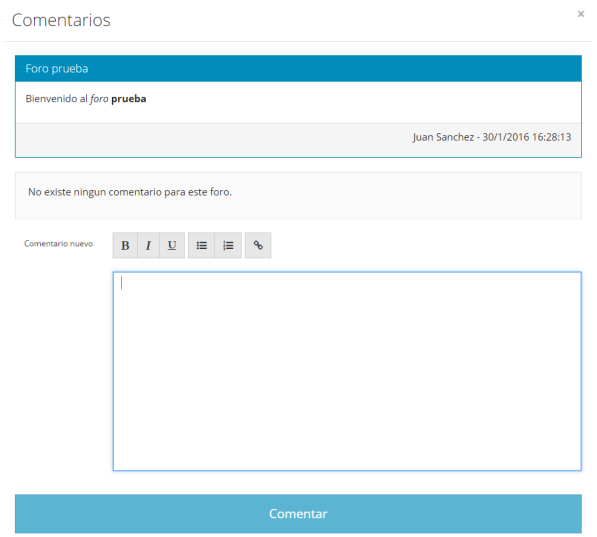


Figura 3.26: Comentarios de un foro.

- Descargas: Aquí se listaran todas las descargas o vídeos que el Grupo Docente del sitio hayan agregado (figura 3.27). Permisología cuadro: 3.14.

Roles que pueden acceder	Roles que pueden modificar
Todos	Grupo Docente

Cuadro 3.14: Roles que interactúan con Descargas de un sitio web.

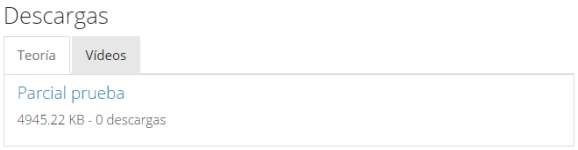


Figura 3.27: Descargas de un sitio web.

- Enviar correo: En esta sección el Grupo Docente podrá enviar un correo a las personas relacionadas con el sitio web. Se podrá elegir entre los Estudiantes de cada sección, como también Docentes y Preparadores (figura 3.28). Permisología cuadro: 3.15.

Roles que pueden acceder	Roles que pueden interactuar
Grupo Docente	Grupo Docente

Cuadro 3.15: Roles que interactúan con Enviar correo de un sitio web.

Figura 3.28: Enviar correo a través de un sitio web.

- Registro de actividades: En esta sección, se mostrarán la fecha la cual se realizaron cambios al sitio web, específicamente cambios a: Coordinador de la asignatura, creación y edición de entregas y evaluaciones y, edición de calificaciones de estudiantes (figura 3.29). Permisología cuadro: 3.16.

Roles que pueden acceder	Roles que pueden modificar
Coordinador	

Cuadro 3.16: Roles que interactúan con Registro de actividades de un sitio web.

Figura 3.29: Registro de actividades de un sitio web.

- Semestres anteriores: Finalmente, esta sección sólo aparecerá si existe más de un (1) sitio web de una asignatura y mostrará una lista de *links* (figura 3.30) de sitios web pertenecientes a la misma asignatura, indicando el semestre y el coordinador asociado a dichos sitios. Permisología cuadro: 3.17.

Roles que pueden acceder	Roles que pueden modificar
Todos	

Cuadro 3.17: Roles que interactúan con Semestres anteriores de un sitio web.

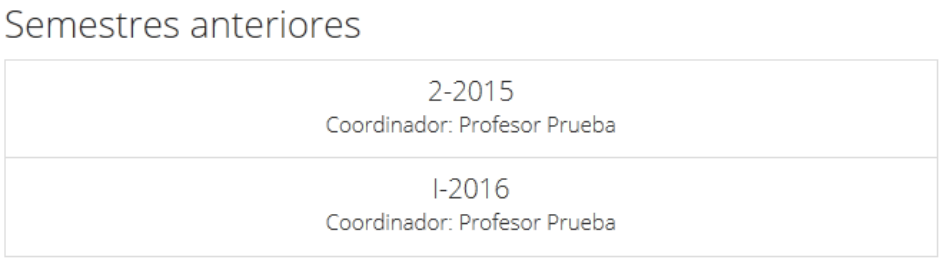
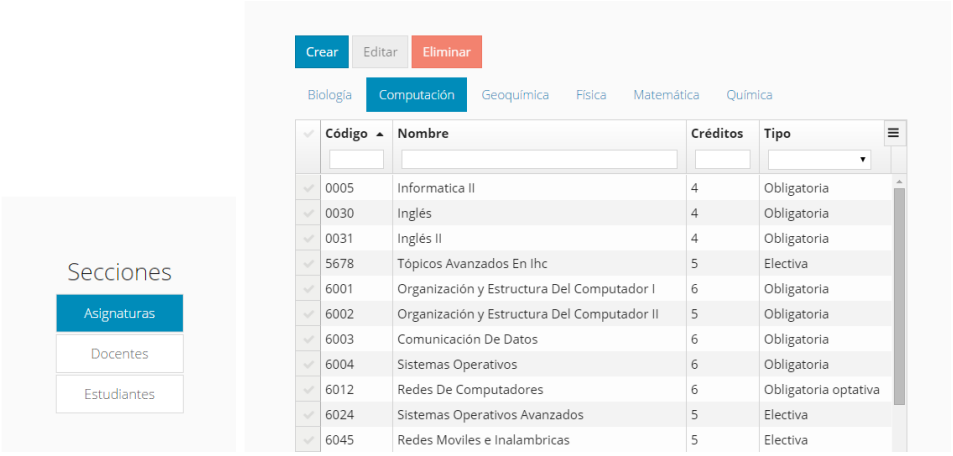


Figura 3.30: Semestres anteriores de un sitio web.

3.6.1.4. Administrador

Al final está el módulo de Administrador. Como fue descrito en la sección (3.4.2.6), un Administrado de *Portalasig* podrá realizar un completo *CRUD* en las Asignaturas, Docentes y Estudiantes guardados en el sistema.

Inmediatamente después de que un Administrador inicie sesión, será redirigido al URL *www.ciens.ucv.ve/portalasig2/#/admin* donde se podrán ver dos bloques principales. El primer bloque mostrará las diferentes secciones a escoger 3.31(a) y el segundo se podrá observar la lista de elementos dependiendo de la sección anteriormente escogida 3.31(b).



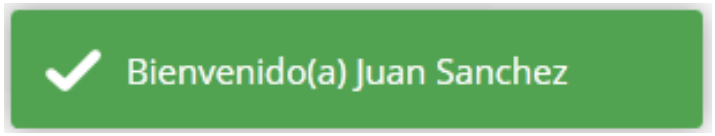
(a) Secciones (b) Lista de elementos

Figura 3.31: Módulo de Administrador.

3.6.1.5. Complementarias

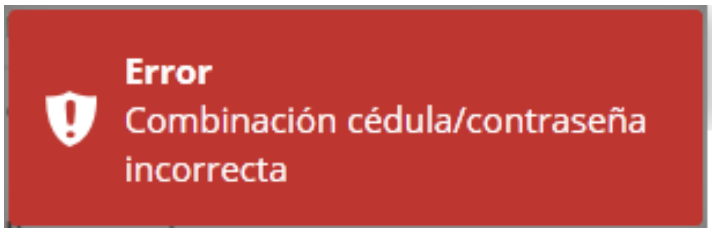
Por último se encuentran aquellas interfaces que no pertenecen a una sección específica del portal web pero que se encuentran a lo largo de este y ayudan a complementar su correcto uso.

- Mensajes de notificación: Son aquellos mensajes que sirve de *feedback* al usuario en cualquiera de las operaciones que realice con el sistema. Se mostraran en la esquina inferior derecha de la pantalla y aparecerán en cuatro (4) colores diferentes. Verdes (figura 3.32(a)), asociados a la creación exitosa de recursos, por ejemplo: Creación de la sesión, creación de un objetivo de un sitio, etc. Azules (figura 3.32(b)), asociados a la edición y eliminación exitosa de recursos, como la edición de algún estudiante en el módulo de administración. Luego están los mensajes en color Rojo (figura 3.32(c)), que estarán asociados a errores que hayan ocurrido tanto por el usuario o por algún error interno en el *backend*. Y finalmente, aquellos mensajes en color Amarillo, que reflejaran advertencias que el sistema le da a conocer al usuario (figura 3.32(d)).

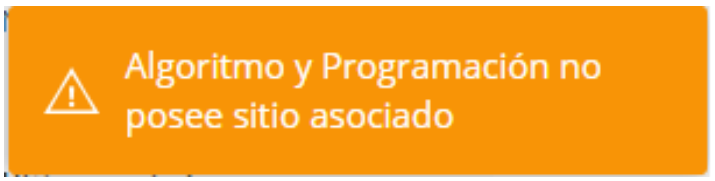


(a) Creación de recursos

(b) Edición de recursos



(c) Error por el usuario o sistema



(d) Advertencia al usuario

Figura 3.32: Mensajes de notificaciones del sistema.

- Mensaje de estado de pruebas: Al finalizar el desarrollo, la nueva versión del portal fue instalado en el servidor de producción, ubicado en el Centro de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, para empezar a realizar diferentes pruebas en el mismo. Para ello, fue colocado un breve mensaje en la parte superior de la página (figura 3.33), después de la Barra de Navegación, indicando al público que la versión nueva se encontraba en "estado de pruebas" lo cual implicaba que la disponibilidad y apariencia pudiera cambiar repentinamente.

Figura 3.33: Mensaje de estado de pruebas.

- *Footer*

  El *Footer* es una sección, que al igual que la Barra de Navegación, está presente en todas las interfaces del sistema (figura 3.34). Éste se encuentra en la parte más inferior y simplemente muestra los logos tanto de la Universidad Central de Venezuela como el de la Facultad de Ciencias, con sus respectivos *links* de páginas web principales.

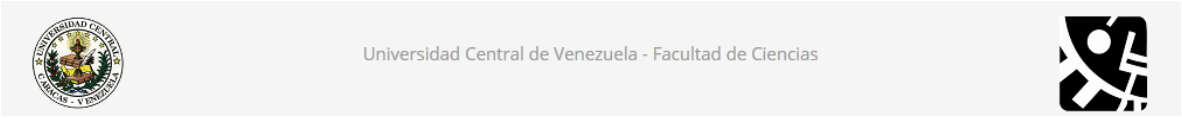


Figura 3.34: *Footer* del portal.

- Correos electrónicos

  Una de las funcionalidades de *Portalasig* es enviar correos electrónicos a sus miembros dependiendo de ciertas eventualidades. El aspecto de los correos electrónicos (figura 3.35) consiste en un título mostrando la razón del mensaje, una sección central describiendo en detalle la razón y finalmente un *footer* con la dirección de la Facultad de Ciencias de la Universidad Central de Venezuela.

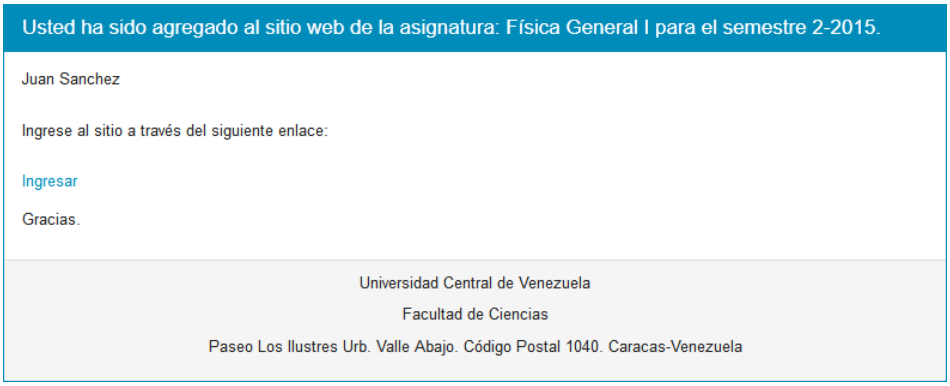


Figura 3.35: Aspecto de los correos electrónicos del portal.

- Recuperar contraseña: Y por último, se encuentra la interfaz de recuperación de contraseña (figura 3.36), el cual se puede acceder a través del *link* del correo electrónico enviado por el sistema. Este correo electrónico es generado si el usuario ingresa su cédula correctamente en la modal de la figura 3.5(b), de la sección 3.6.1.1.

Figura 3.36: Recuperación de contraseñas del portal.

4

Pruebas y resultados

Al final de todo el desarrollo de la nueva versión de *Portalasig* se realizaron una serie de pruebas sobre el sistema para medir su usabilidad y desempeño. Dado que el presente Trabajo Especial de Grado consiste en la realización de una nueva versión de una aplicación web, es importante realizar estas pruebas para demostrar que efectivamente dicha versión es igual o mejor en diferentes aspectos. Estos aspectos, no sólo se encuentran en el ámbito de las interfaces las cuales la componen sino también en la velocidad de respuesta y el uso de recursos del servidor.

Para la prueba de usabilidad se realizaron una serie de preguntas basadas en las Heurísticas de Usabilidad propuestas por Nielsen (1995) [21], las cuales son:

- Visibilidad del estatus del sistema: El sistema debe mantener siempre informados a los usuarios de lo que está ocurriendo, a través de un *feedback* apropiado en un tiempo razonable.
- Consistencia entre el sistema y el mundo real: El sistema debería presentarse en el idioma de los usuarios, con palabras, frases y conceptos familiares, en vez de terminologías técnicas. Como también seguir convenciones usadas en el mundo real, haciendo ver la información en un orden natural y lógico.
- El usuario es libre y tiene el control: Los usuarios usualmente seleccionan funcionalidades del sistema por error, por lo cual es necesario la presencia de "Salidas de emergencias" para abandonar un estado no querido sin tener que pasar por diálogos o procesos extendidos.
- Consistencia y estándares: Los usuarios no deberían tener que preguntarse si

palabras distintas, situaciones o acciones significan lo mismo.

- Prevención de errores: Mejor que el uso de un mensaje de error es la presencia de un diseño el cual prevenga el problema en primer lugar.
- Mejor reconocer que memorizar: Hacer objetos, acciones y opciones visibles. El usuario no debería tener que recordar como seguir usando la aplicación en una determinada interacción.
- Flexibilidad y eficiencia de uso: Proveer al usuario experto el uso de atajos que aceleren una interacción dada. Interfaz flexible que se comporte "a la medida" de usuarios novatos y usuarios expertos.
- Diseño estético y minimalista: Los diálogos presentados en el sistema no deberían contener información la cual sea irrelevante o no necesaria.
- Ayuda al usuario a reconocer, diagnosticar y recuperarse de los errores: Los mensajes de errores deberían ser expresados en lenguaje plano (no códigos), indicar el problema preciso y sugerir una solución inmediata.
- Ayuda y documentación: Es preferible que el sistema pueda ser usado sin ninguna documentación de manera intuitiva. Sin embargo, puede ser necesario proveer ayudar al usuario. Es preciso que esta información sea fácil de encontrar, y sobre todo que esté orientada a las tareas concretas que realiza el usuario, antes que a cuestiones teóricas o demasiado genéricas.

Las heurísticas son utilizadas para identificar cualquier problema asociado con el diseño de interfaces de usuarios. La actividad realizada contó con una serie de preguntas directas acerca el sistema y la capacidad del usuario para contestar. Estas preguntas sirven de ayuda para detectar qué partes de la interfaz resultan mas obvias (usables) y cuáles no.

Para dicha actividad, fueron seleccionados cinco (5) docentes y cinco (5) estudiantes de la Facultad de Ciencias que utilizan la versión anterior de *Portalasig* y se les invitó a utilizar la nueva versión. Las preguntas fueron analizadas entre uno (1) y cinco (5), donde uno es muy malo y cinco excelente.

#	Pregunta	1	2	3	4	5
1	Esta nueva versión, ¿posee las características y funcionalidades de la versión anterior de PortalAsig?	0 %	0 %	0 %	0 %	100 %
2	Los URL de los sitios web, ¿son fáciles de entender de la forma <código>/<semestre>?	0 %	0 %	0 %	20 %	80 %
3	¿Se puede regresar a la página principal de una manera rápida y sencilla?	0 %	0 %	10 %	20 %	70 %
4	¿Posee un correcto uso de mensajes de *feedback*?	0 %	0 %	0 %	0 %	100 %
5	¿Es predecible la respuesta antes de hacer *click* a enlaces y/o botones?	0 %	0 %	0 %	10 %	90 %
6	¿Posee un buen contraste de colores?	0 %	0 %	0 %	20 %	80 %
7	¿Considera útil el uso del panel de materias relacionadas con el dueño de la sesión?	0 %	0 %	0 %	0 %	100 %
8	¿La información se encuentra libre de errores gramaticales y ortográficos?	0 %	0 %	0 %	0 %	100 %
9	¿Es adecuado el tamaño de la tipografía utilizada? Para las versiones *desktop* y móvil.	0 %	0 %	0 %	10 %	90 %
10	El tipo de lenguaje y vocabulario utilizado, ¿son adecuados a la comunidad académica?	0 %	0 %	0 %	10 %	90 %
11	¿Considera correcta la distribución de elementos en la versión móvil?	0 %	0 %	10 %	0 %	90 %

Cuadro 4.1: Preguntas de Usabilidad realizada a usuarios que usan la versión anterior de *Portalasig*.

Bajo estos resultados, se puede apreciar que la nueva versión de *Portalasig* tuvo una buena aceptación entre los usuarios, dando a conocer que en general el sistema satisface las necesidades y cumple con los requerimientos funcionales descritos al principio de este Trabajo Especial de Grado.

A continuación serán presentadas las pruebas de velocidad de respuesta de ambas versiones de *Portalasig*, para luego realizar un cuadro comparativo con los resultados. Para realizar dichas pruebas, fue utilizado el inspector de elementos del navegador Firefox en su versión 43.0.1, el cual tiene una sección denominada “Red”, que permite medir cuanto tiempo es transcurrido desde que se solicita una página web en el campo de direcciones del navegador hasta que es cargada completamente.

Descripción de la prueba	Versión anterior	Versión nueva
Visitar el URL principal del portal	**3.42s**	8.25s
Dirigirse a una licenciatura diferente	1.28s	**0.0s**
Dirigirse a un sitio web	0.84s	**0.42s**
Dirigirse a secciones distintas de un sitio web	0.57s	**0.0s**
Crear un objetivo nuevo en un sitio web	0.22s	**0.08s**
Ver estudiantes de un sitio web	0.24s	**0.05s**
Entrar en el módulo de administración	1.06s	**0.35s**
Ver docentes en el módulo de administración	3.68s	**0.0s**
Ver estudiantes en el módulo de administración	8.54s	**0.0s**

Cuadro 4.2: Pruebas de desempeño: velocidad de carga realizadas en ambas versiones de *Portalasig*.

El cuadro 4.2 muestra que la versión nueva de *Portalasig* responde más rápido en todas las pruebas excepto en la primera. Esto se debe a que esta versión, al estar compuesta por un framework *SPA*, posee gran parte de toda su lógica en el *frontend*, es decir en los archivos JavaScript que ejecuta el navegador. Cabe destacar que esto ocurre la primera vez que se visita la página y es comúnmente llamado el *bootstraping* de la aplicación, ya que es la carga más pesada, aproximadamente 2 MB. Después de esta pri mera carga, la aplicación no requiere más archivos HTML, CSS o JavaScript para su correcto funcionamiento. A partir de este momento, la nueva versión del portal, sólo realizará peticiones HTTP AJAX (si son necesarias) para poder actualizar las diferentes vistas e interfaces, sin la necesidad de realizar una carga completa en el navegador, cualidad que es totalmente distinta en la versión anterior y que puede verse reflejado en los resultados dados en el cuadro mencionado al principio.

5

Conclusiones y recomendaciones

5.1. Conclusiones

Al finalizar el presente Trabajo Especial de Grado, los objetivos propuestos al inicio de la investigación fuero alcanzados con éxito. Esta nueva versión de *Portalasig* se encuentra en un servidor de producción en el Centro de Computación de la Facultad de Ciencias funcionando a la normalidad y puede ser accedido a través del URL: *http://www.ciens.ucv.ve/portalasig2*.

La versión anterior del portal web ha estado funcionando desde que fue puesto en producción, en el año 2013, sin ninguna eventualidad mayor. Sin embargo, se detectaron limitaciones e inconvenientes en algunas de sus funcionalidades, como por ejemplo: No poder subir archivos con peso mayor a 2 MB al sistema y problemas de usabilidad a la hora de acceder desde un dispositivo móvil. Estas incidencias se cuantificaron y se estudiaron para resolverlas en esta nueva versión, las cuales todas fueron solventadas.

Una ligera modificación de la metodología ágil *SCRUM* fue utilizada para realizar los diferentes componentes que comprenden esta nueva versión del portal web. Primeramente fue realizada una migración de los datos de los usuarios y asignaturas de la versión anterior a la nueva, para evitar la creación de todos los usuarios y asignaturas ya existentes, permitiendo que los Docentes y Estudiantes ingresen al nuevo

sistema de manera rápida y sencilla.

En los Trabajos Futuros del TEG asociado a la versión anterior de *Portalasig* fue propuesto el desarrollo de un Web API, el cual sea el *backend* de toda la lógica de negocio del portal web y así poder integrar con distintos entes, como aplicaciones web, móviles o incluso otro *backend*. Esto fue realizado, como segundo punto de la metodología, con éxito.

Seguido a esto, fue realizada la documentación del Web API, utilizando el módulo *Aglio*, que consiste en una página HTML estática con estilos la cual indexa todo los *endpoints* que pueden ser llamados y utilizados del sistema. La documentación es importante, ya que agiliza el posible desarrollo de trabajos futuros que puedan ser integrados con el sistema. Dicho Web API puede ser accedido a través del URL: *http://www.ciens.ucv.ve/portapi*.

Finalmente, el último punto del desarrollo consistió en la realización de la aplicación web SPA (*frontend*). La idea principal de las interfaces se mantuvo; Una lista de de asignaturas, agrupadas por licenciaturas, las cuales se asocian a distintos sitios web dependiendo del semestre. Una de las innovaciones más resaltantes en este punto, fue realizar el *frontend* como SPA. Otra innovación interesante y utilizada en la actualidad, es proveer interfaces *responsive*, las cuales se adaptan a distintos dispositivos de pantallas de diferentes tamaños.

En una perspectiva general, la nueva versión de *Portalasig* logró proveer todas las funcionalidades de la versión anterior, mejorando algunas de ellas, como también agregando nuevas características con el objetivo principal de que el sistema sea usado por la comunidad académica de la Facultad de Ciencias.

5.2. Recomendaciones

- Desarrollo de un módulo de integración con el sistema *Conest*, el cual permita sincronizar las asignaturas, docentes, estudiantes y calificaciones. Esto requeriría aumentar la seguridad del Web API como también implementar conexiones seguras como *SSL*.
- Desarrollo de un módulo generador de notas informativas de las asignaturas dado los objetivos, bibliografías, contenidos temáticos, evaluaciones y eventos de un sitio web particular.

- Desarrollo de un módulo que permita la personalización del portal web en general como también cada sitio web por separado.
- Desarrollo de una aplicación móvil que se integre con el Web API de *Portalasig*.
- Desarrollar un instalador general de *Portalasig*, el cual permita crear portales web destinados a cualquier tipo de comunidad académica.

Anexos

Diagrama de Casos de Uso

Actor Caso de Uso	Roles representados en el sistema
Usuario	Todos los roles
Usuario registrado	Estudiante, Docente y Administrador
Estudiante	Estudiante
Grupo Docente	Estudiante(Preparador), Docente y Coordinador
Docente	Docente
Coordinador	Docente(Coordinador)
Administrador	Administrador

Cuadro 5.1: Leyenda de relación de actores de casos de uso con respecto a roles del sistema.

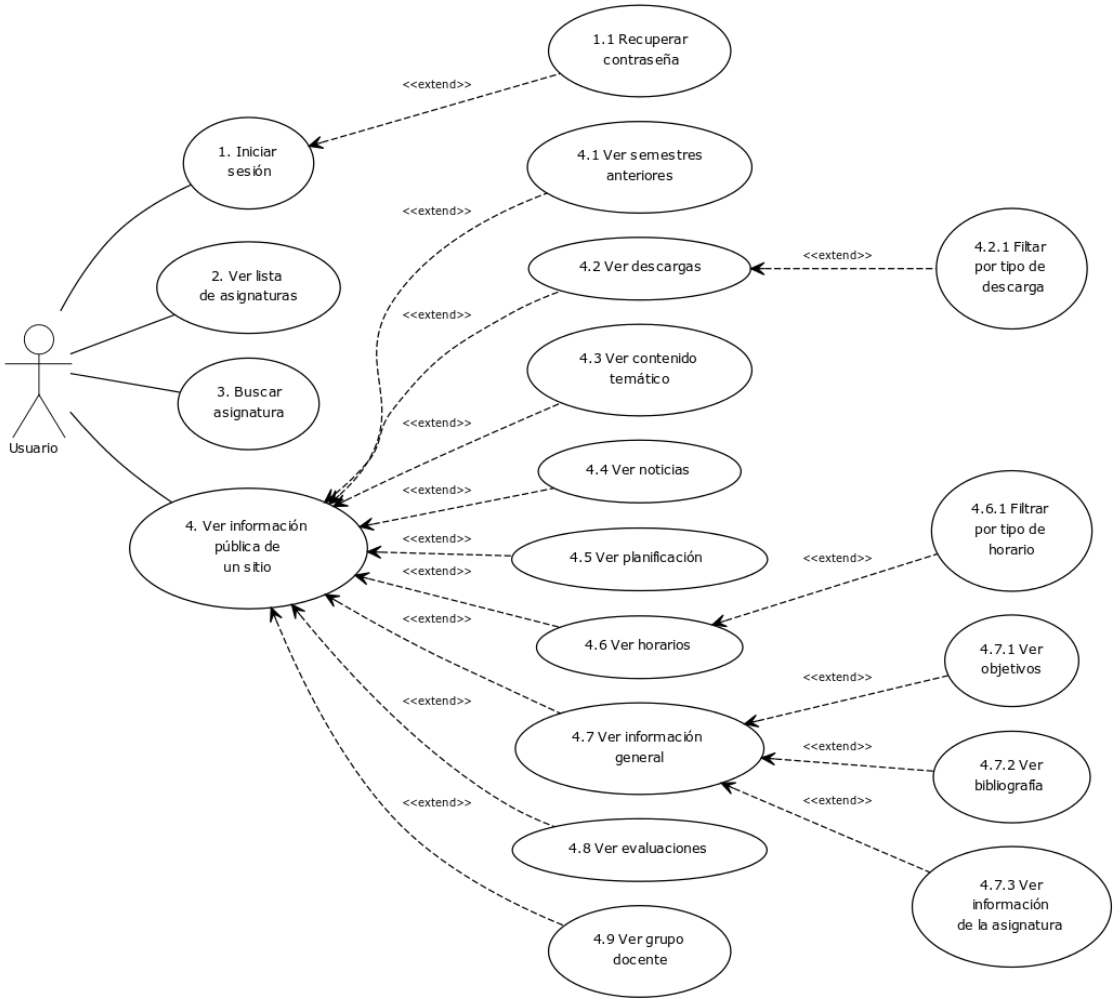


Figura 5.1: Casos de Uso - Usuario.

Caso de Uso 1	**Iniciar sesión**
Actor principal:	Usuario
Descripción	El actor inicia sesión en el sistema
Precondiciones:	▪ El actor desea iniciar sesión en el sistema ▪ El actor no debe poseer una sesión en el sistema
Postcondición:	El actor obtiene una sesión válida en el sistema
Flujo Principal:	

1. El actor selecciona la opción de Iniciar Sesión
2. El actor introduce su cédula y contraseña
3. El actor selecciona el botón de Iniciar Sesión

Flujo alternativo:

a Datos inválidos:
   1. El sistema muestra un mensaje de error
   2. El actor regresa al paso 2

Caso de Uso 2	**Ver lista de asignaturas**
Actor principal:	Usuario
Descripción	El actor observa la lista de asignaturas del sistema
Precondición:	El actor desea ver la lista de asignaturas del sistema
Postcondición:	El actor logra observar la lista de asignaturas del portal

Flujo Principal:

1. El actor selecciona el logo de *Portalasig* o se dirige al URL principal del sistema

Caso de Uso 3	**Buscar asignatura**
Actor principal:	Usuario
Descripción	El actor busca una asignatura del sistema
Precondición:	El actor desea buscar una asignatura
Postcondición:	El actor consigue la asignatura deseada

Flujo Principal:

1. El actor selecciona el campo de texto de Buscar Asignatura
2. El actor escribe nombre o código de la asignatura que desea buscar

Caso de Uso 4	**Ver información pública de un sitio**
Actor principal:	Usuario
Descripción	El actor observa la información pública de un sitio web del sistema, la cual consiste en: ▪ Semestres anteriores ▪ Descargas ▪ Contenido Temático ▪ Noticias ▪ Planificación ▪ Horarios ▪ Información General ▪ Evaluaciones ▪ Grupo Docente
Precondición:	El actor desea ver la información pública de un sitio del sistema
Postcondición:	El actor observa la información pública de un sitio web
Flujo Principal: 1. El actor selecciona una asignatura de la Lista de Asignaturas	
Flujo alternativo: a Asignatura no posee sitio web asociado: 1. El sistema muestra un mensaje de advertencia 2. El actor regresa al paso 1	

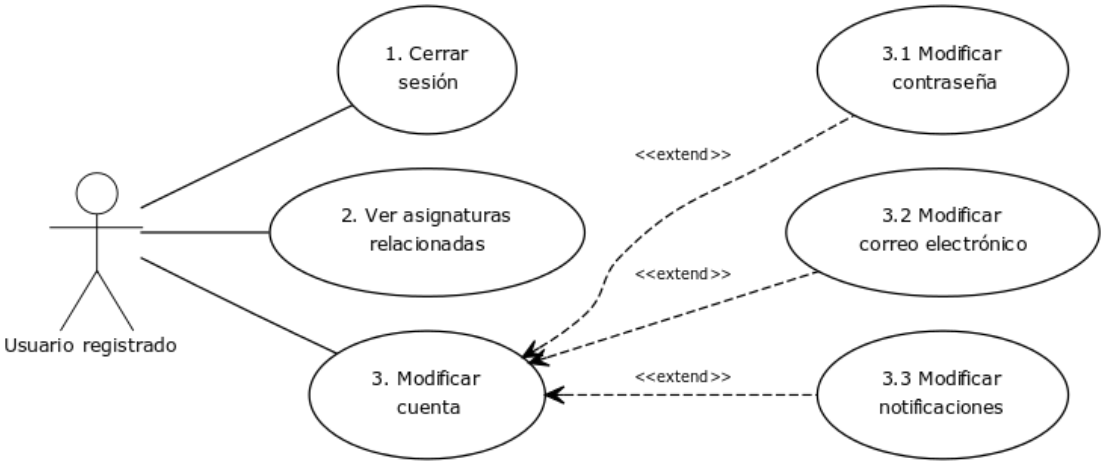


Figura 5.2: Casos de Uso - Usuario Registrado.

Caso de Uso 1	**Cerrar sesión**
Actor principal:	Usuario registrado
Descripción	El actor cierra sesión en el sistema
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor desea cerrar sesión en el sistema
Postcondición:	El actor no posee sesión asociada al sistema
Flujo Principal: 1. El actor selecciona su nombre en la Barra de Navegación, desplegándose el menú de opciones 2. El actor selecciona Cerrar Sesión	

Caso de Uso 2	**Ver asignaturas relacionadas**
Actor principal:	Usuario registrado
Descripción	El actor observa la lista de asignaturas relacionadas con el semestre en curso

Precondiciones:	▪ El actor desea ver la lista de asignaturas relacionadas con el semestre en curso ▪ El actor posee asignaturas relacionadas con su cuenta en el semestre en curso
Postcondición:	▪ El actor posee una sesión válida ▪ El actor observa sus asignaturas relacionadas con el semestre en curso

Flujo Principal:

1. El actor selecciona el logo de *Portalasig* o se dirige al URL principal del sistema

Caso de Uso 3	**Modificar cuenta**
Actor principal:	Usuario registrado
Descripción	El actor modifica su cuenta en alguno de los siguientes aspectos: ▪ Contraseña ▪ Correo electrónico ▪ Notificaciones
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor desea modificar su cuenta
Postcondición:	El actor logra modificar su cuenta

Flujo Principal:

1. El actor selecciona su nombre en la Barra de Navegación, desplegándose el menú de opciones
2. El actor selecciona una de las opciones para modificar su cuenta

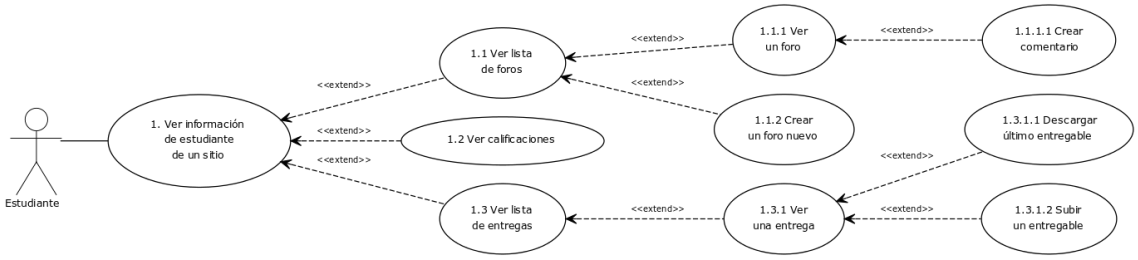


Figura 5.3: Casos de Uso - Estudiante.

Caso de Uso 1	**Ver información de estudiante de un sitio**
Actor principal:	Estudiante
Descripción	El actor observa la información de estudiante de un sitio web del sistema, la cual consiste en: ▪ Foros ▪ Calificaciones del estudiante dueño de la sesión ▪ Entregas
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es estudiante del sitio web seleccionado ▪ El actor desea ver la información de estudiante de un sitio del sistema
Postcondición:	El actor observa la información de estudiante de un sitio web

Flujo Principal:

1. El actor selecciona una asignatura de la Lista de Asignaturas
2. El actor selecciona una de las secciones de estudiante del sitio web

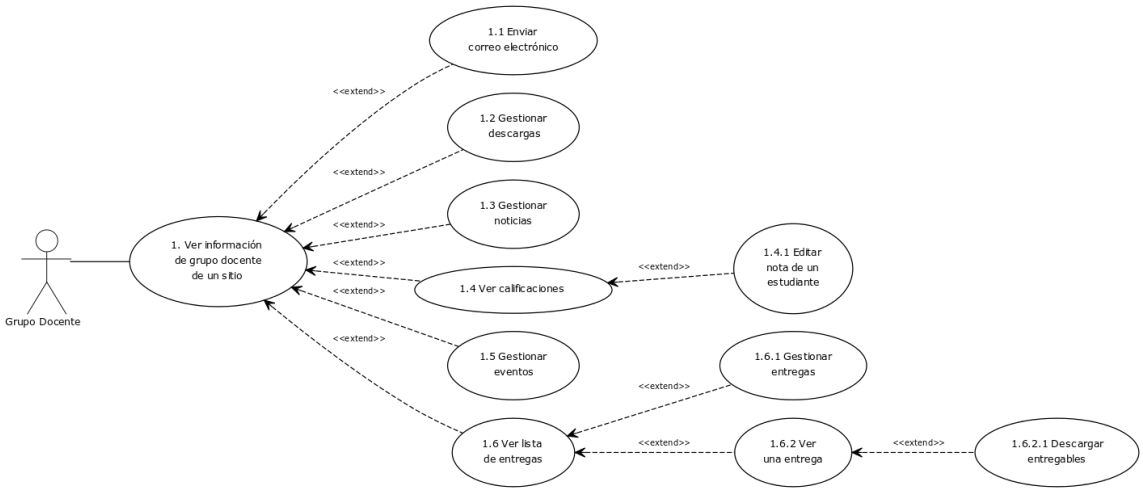


Figura 5.4: Casos de Uso - Grupo Docente.

Caso de Uso 1	**Ver información de grupo docente de un sitio**
Actor principal:	Grupo Docente
Descripción	El actor observa la información de grupo docente de un sitio web del sistema, la cual consiste en: ▪ Enviar correos electrónicos ▪ Gestión de Descargas ▪ Gestión de Noticias ▪ Ver y gestionar calificaciones de los estudiantes asociados al sitio web ▪ Gestión de eventos ▪ Ver Entregas y descargar entregables de los estudiantes asociados al sitio web
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es parte del grupo docente del sitio web seleccionado ▪ El actor desea ver la información de grupo docente de un sitio del sistema
Postcondición:	El actor observa la información de grupo docente de un sitio web

Flujo Principal:

1. El actor selecciona una asignatura de la Lista de Asignaturas
2. El actor selecciona una de las secciones de grupo docente del sitio web

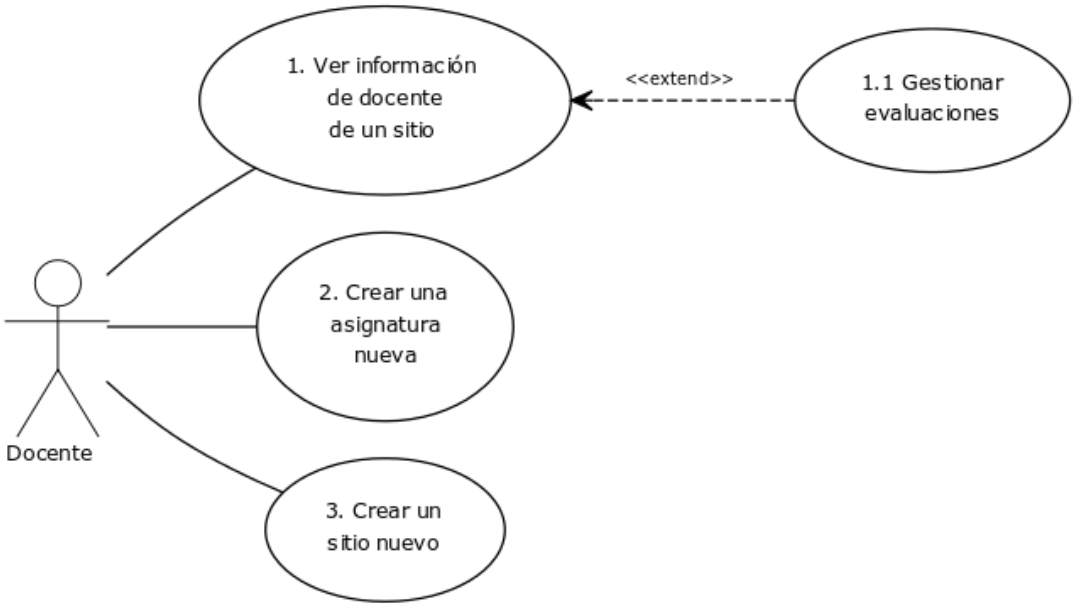


Figura 5.5: Casos de Uso - Docente.

Caso de Uso 1	**Ver información de docente de un sitio**
Actor principal:	Docente
Descripción	El actor observa la información de docente de un sitio web del sistema, la cual consiste en: ▪ Gestión de evaluaciones
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es un docente del sitio web seleccionado ▪ El actor desea ver la información de docente de un sitio del sistema

Postcondición:	El actor observa la información de docente de un sitio web

Flujo Principal:

1. El actor selecciona una asignatura de la Lista de Asignaturas
2. El actor selecciona una de las secciones de docente del sitio web

Caso de Uso 2	**Crear una asignatura nueva**
Actor principal:	Docente
Descripción	El actor crea una nueva asignatura en el sistema
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es un docente del sistema ▪ El actor desea crear una asignatura nueva en el sistema
Postcondición:	El actor efectivamente crea una asignatura nueva en el portal web

Flujo Principal:

1. El actor se dirige a la Lista de Asignaturas
2. El actor selecciona el botón Agregar asignatura
3. El actor ingresa los datos de la asignatura a agregar
4. El actor selecciona el botón de Guardar

Flujo alternativo:

a Código ya utilizado:
   1. El sistema muestra un mensaje de error
   2. El actor regresa al paso 3

Caso de Uso 3	**Crear un sitio nuevo**
Actor principal:	Docente
Descripción	El actor crea un nuevo sitio web en el portal

Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es un docente del sistema ▪ El actor desea crear un sitio web nuevo en el sistema
Postcondiciones:	▪ El actor efectivamente crea un sitio web nuevo ▪ El actor es el Coordinador del sitio web creado

Flujo Principal:

1. El actor se dirige a la Lista de Asignaturas
2. El actor selecciona el botón en la columna de Crear nuevo sitio
3. El actor selecciona el semestre el cual quiere asociar el nuevo sitio web
4. El actor selecciona el botón de Crear Sitio

Figura 5.6: Casos de Uso - Coordinador.

Caso de Uso 1	**Ver información de coordinador de un sitio**
Actor principal:	Coordinador

Descripción	El actor observa la información de coordinador de un sitio web del sistema, la cual consiste en: ▪ Registro de actividades ▪ Gestión de Estudiantes ▪ Gestión de Horarios ▪ Gestión de Grupo Docente ▪ Gestión de Contenido Temático ▪ Gestión Información General ▪ Eliminar el sitio web ▪ Cambiar Coordinador del sitio ▪ Configurar noticias del sitio
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es el Coordinador del sitio web seleccionado ▪ El actor desea ver la información de coordinador de un sitio del sistema
Postcondición:	El actor observa la información de coordinador de un sitio web

Flujo Principal:

1. El actor selecciona una asignatura de la Lista de Asignaturas
2. El actor selecciona una de las secciones de coordinador del sitio web

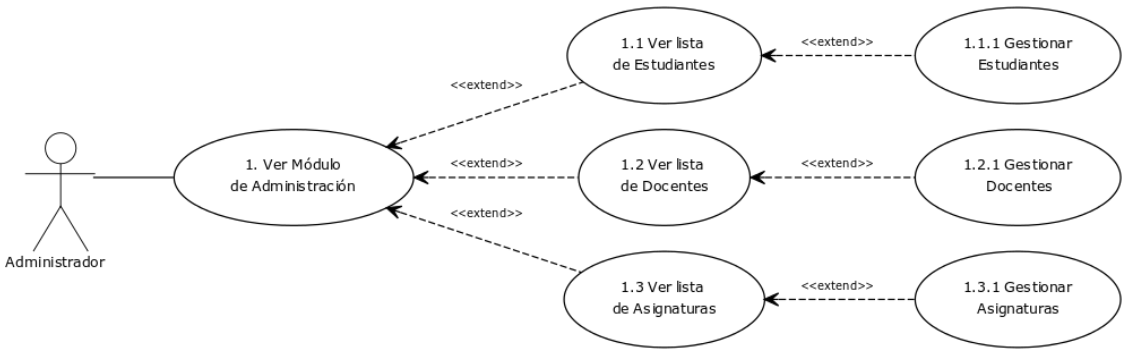


Figura 5.7: Casos de Uso - Administrador.

Caso de Uso 1	**Ver módulo de administración**
Actor principal:	Administrador
Descripción	El actor entra en el módulo de administración del sistema, el cual consiste en: ▪ Ver y gestionar estudiantes del sistema ▪ Ver y gestionar docentes del sistema ▪ Ver y gestionar asignaturas del sistema
Precondiciones:	▪ El actor posee una sesión válida ▪ El actor es Administrador del sistema
Postcondición:	El actor entra en el módulo de administración del portal web
Flujo Principal: 1. El actor selecciona el botón de Administrador en la Barra de Navegación	

Diagrama de despliegue

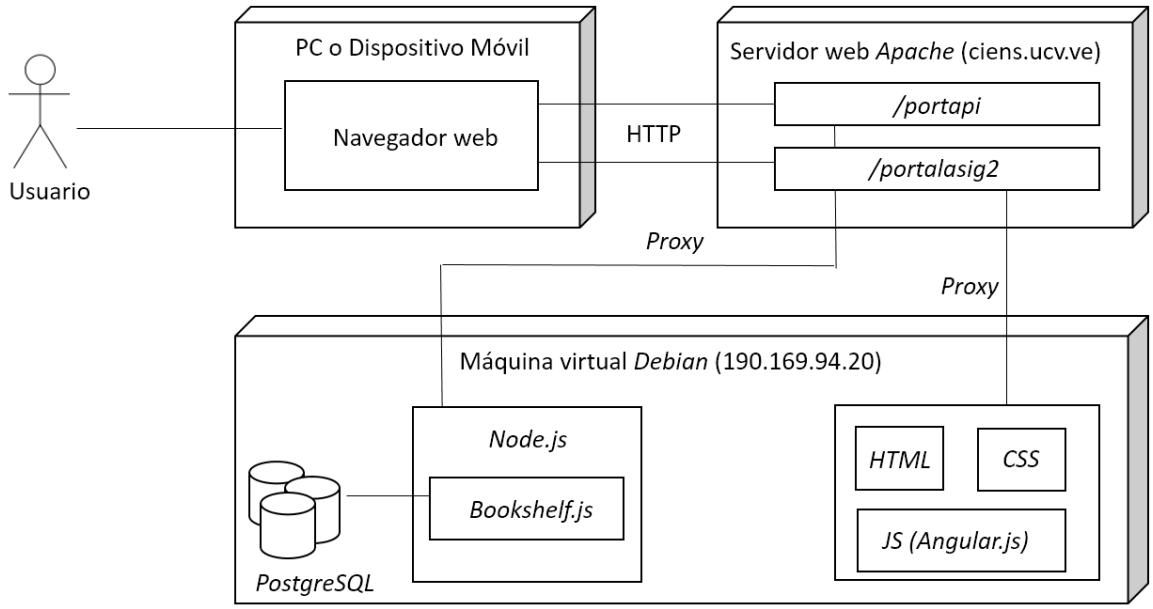


Figura 5.8: Diagrama de despliegue de *Portalasig*.

Referencias

[1] O. Díaz y M. Villavicencio, "Portal generador de sitios web de asignaturas para la facultad de ciencias de la universidad central de venezuela," Caracas, Octubre 2013.

[2] A. Tatnall, Web Portals, the new gateways to internet information and services, 2005, https://books.google.co.ve/books?id=5IeT8JhZSmcC&printsec=frontcover#v=onepage&q&f=false.

[3] J. Townsend, Building Portals, Intranets, and Corporate Web Sites Using Microsoft Servers, 2004, https://books.google.co.ve/books?id=bCXnK31_d_0C&printsec=frontcover#v=onepage&q&f=false.

[4] S. Ruby, *Agile Web Development with Rails 4*. The Pragmatic Programmers, 2013.

[5] B. Mulloy, *Web API Design, Crafting Interfaces that Developers Love*. Apigee.

[6] S. Seshadri, *AngularJS Up & Running*. O'Rielly, 2014.

[7] M. Pilgrim, *Dive into Python 3*. Apress, 2011.

[8] M. Haver, Eloquent Javascript, 2015, http://eloquentjavascript.net/.

[9] A. Mardan, Node.js FUNdamentals: A Concise Overview of The Main Concepts, 2016, http://webapplog.com/node-js-fundamentals-a-concise-overview-of-the-main-concepts/.

[10] NW.js, http://nwjs.io/.

[11] Node.js v4.2.6 Documentation, https://nodejs.org/dist/latest-v4.x/docs/api/.

[12] T. Satpathy, *Una guía para el CONOCIMIENTO DE SCRUM (GUÍA SBOK)*, 2013.

[13] Angular UI Grid, http://ui-grid.info/.

[14] Waterline, An adapter-based ORM for Node.js with support for mysql, mongo, postgres, redis, and more, https://github.com/balderdashy/waterline.

[15] Knex.js, http://knexjs.org/.

[16] Bookshelf.js, http://bookshelfjs.org/.

[17] Nodemailer, http://nodemailer.com/.

[18] Flow.js and NG-Flow, https://flowjs.github.io/ng-flow/.

[19] API Blueprint, https://apiblueprint.org/.

[20] Aglio, an API Blueprint renderer with theme support that outputs static HTML, https://github.com/danielgtaylor/aglio.

[21] 10 Usability Heuristics for User Interface Design, https://www.nngroup.com/articles/ten-usability-heuristics/.

[22] Express.js. Fast, unopinionated, minimalist web framework for Node.js, http://expressjs.com/.

[23] Sails.js. The web framework of your dreams., http://sailsjs.org/.

[24] Angular.js By Google. HTML enhanced for web apps!, https://angularjs.org/.