# UNIVERSIDAD POLITÉCNINA SALESIANA SEDE QUITO

**CARRERA**:

INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:

**INGENIERO DE SISTEMAS**

**TEMA**:

Desarrollo de una aplicación web para la gestión del seguimiento del cumplimiento de tareas de los docentes de la CARRERA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD POLITÉCNICA SALESIANA

**AUTOR:**

PABLO EFRÉN GUERRA MORA

**TUTOR**:

FRANKLIN EDMUNDO HURTADO LARREA

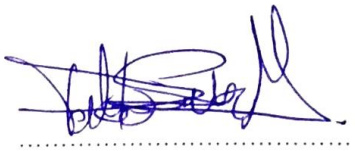
Quito, Septiembre 2017

# Cesión de derechos de autor

Yo Pablo Efrén Guerra Mora con documento de identificación No. 1723676688 manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación Desarrollo de una aplicación web para la gestión del seguimiento del cumplimiento de tareas de los docentes de la CARRERA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD POLITÉCNICA SALESIANA, mismo que ha sido desarrollado para optar por el título de INGENIERO DE SISTEMAS, en la Universidad Politécnica Salesiana, queda la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo el derecho moral de la obra antes citada. En concordancia suscribo este documento en el momento de hacer la entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Politécnica Salesiana.

Quito, septiembre del 2017



Pablo Efrén Guerra Mora

1723676688

## Declaratoria de coautoría del docente tutor

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación Desarrollo de una aplicación web para la gestión del seguimiento del cumplimiento de tareas de los docentes de la CARRERA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD POLITÉCNICA SALESIANA realizado por Pablo Efrén Guerra Mora, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, septiembre del 2017

Franklin Edmundo Hurtado Larrea

1713382016

# Índice

## Índice de tablas

## Índice de figuras

**Resumen**

Con el crecimiento de la Facultad de Ingeniería de Sistemas de la Universidad Politécnica Salesiana han surgido nuevas necesidades que no han podido ser ignoradas. El Consejo de Carrera de la facultad ha venido realizando tareas de evaluación a los docentes y su compromiso con el proceso de enseñanza, también con las responsabilidades fuera del aula incluyendo la entrega (a tiempo) de documentos y otras tareas que los docentes tienen con la universidad y la carrera.

Luego de que el actual sistema de evaluación dé resultados insuficientes y de un análisis por parte del Consejo de carrera de la Facultad, se llegó a la conclusión que el proceso de evaluación usado no es eficiente y que debe existir una nueva forma para evaluar a los docentes que esté alineada a la actual situación de la carrera y todas las personas que la conforman. El proceso actual de evaluación consiste en subir archivos a una carpeta compartida que puede estar en un servidor en la nube o en una computadora local; posteriormente el jefe de área que asignó la tarea a un grupo de docentes descarga el archivo y registra las notas. El principal problema con esto es que la carpeta no fue creada como una plataforma de evaluación y aunque no esté a la vista la evaluación va más allá que subir un archivo y registrar la nota.

Es por eso que, después del análisis del proceso y la forma de mejorarlo, se ha propuesto un sistema que esté específicamente diseñado para cubrir los requerimientos de evaluación de la carrera. Este sistema permitirá la creación, asignación y evaluación de tareas y se evaluarán de forma no sólo cuantitativa sino cualitativa. De esta forma el director de carrera tendrá una perspectiva más amplia de cómo los docentes se están desempeñando y se encontrarán falencias mejoras que renovarán la carrera cada año.

**Abstract**

With the fast growing pace of the Engineering Faculty of Universidad Politecnica Salesiana new needs have aroused and attention to them has not been disregarded. The faculty council has been evaluating teachers regarding their compromise with the teaching process but also with their responsibilities off the classroom including delivery (on time) of paperwork and other duties teachers have with the university and the career.

After the poor output received from the current evaluation system and of an analysis from the faculty council the conclusion met was that the current evaluation process is not efficient and there must be a new way for the council to evaluate teachers and be up to date with the current situation of the career and everyone who is involved in it. The actual process of evaluation consist of uploading files to a shared folder which may be on a cloud server or on a local pc; after that the area chief who assigned the task to a group of teachers downloads the files and registers the grades on a spreadsheet, the main problem with this is that the folder was not meant to serve as an evaluation platform and as invisible as it can be evaluation goes deeper than uploading a file and registering a grade.

This is why, after the analysis of the process and the way to improve it, the development of a system, specifically designed for meeting the requirements of the career has been exposed and been proposed to be implemented soon in the career.

This system allows the creation, assignment, and evaluation of tasks and what will measure will not only be quantity but quality of tasks. This way the career director will have a wide view of how teachers are performing and will find faults and improvements that will make the career better each year.

# Introducción e información general

## Justificación del tema

Como carrera de Ingeniería de Sistemas se puede realizar un sistema que permita automatizar el proceso de evaluación, de modo que los docentes suban los documentos requeridos por los jefes de área y puedan evaluar y obtener reportes además de tener la información disponible cuando se requiera.

Existe una discordancia en cuanto a la carpeta compartida usada actualmente, muchas veces los docentes informan que han subido sus tareas pero el jefe de área no puede visualizarlas y califican como tarea no entregada aun cuando el docente que subió la tarea puede visualizarla.

## Objetivos

### *Objetivo General*

Mejorar el proceso de evaluación de las tareas de los docentes de la carrera de Ingeniería de Sistemas de la Universidad Politécnica Salesiana.

### *Objetivos Específicos*

- Investigar y definir el modelo de gestión de seguimiento de tareas utilizado actualmente.
- Analizar y diseñar la solución.

- Desarrollar un sistema Web que facilite el seguimiento y evaluación de las tareas de los docentes.
- Incrementar el número de tareas entregadas en comparación con el sistema actual.

# Marco metodológico

## *Desarrollo ágil*

Los proyectos que utilizan metodologías de desarrollo ágil difieren de las metodologías tradicionales (que están basadas en un desarrollo secuencial) en que su desarrollo es incremental e iterativo, es decir, el proyecto avanza en iteraciones y las funcionalidades pueden ir creciendo poco a poco según los requerimientos y el tiempo disponible. Las metodologías no ágiles o tradicionales tienen un plan estático que se debe seguir y se obtiene un producto completo al final del plan. El objetivo de comparar metodologías ágiles con tradicionales no tiene como propósito concluir que el desarrollo ágil es mejor que el secuencial, ambas son herramientas de desarrolladores y existen circunstancias apropiadas para cada una (Kenneth, S. Rubin, 2012, pág. 29).

El uso de una metodología ágil en este proyecto fue resuelto debido a que el equipo de trabajo es pequeño (1 persona) y el tiempo de desarrollo es corto (menos de 1 año), además la metodología escogida es muy útil para organizar el tiempo y tener un producto presentable en un período de tiempo corto, así se puede tener retroalimentación de las funcionalidades del software y realizar mejoras para al final tener un producto de calidad y que cumpla con los objetivos establecidos.

# SCRUM

Una de las ventajas de esta metodología de desarrollo ágil es que es modificable y adaptable a las necesidades de cada proyecto, por lo que para el desarrollo de este proyecto se realizaron algunos ajustes que difieren de cómo se trabaja normalmente con SCRUM sin romper ninguna regla de la metodología. Existió un único desarrollador en el desarrollo del sistema. SCRUM no especifica cómo se debe realizar el desarrollo así que si lo realiza un solo desarrollador no está saliendo del margen de la metodología.

Los roles de SCRUM no fueron los tradicionales, algunos roles fueron realizados por la misma persona:

- **Scrum Master:** El desarrollador del proyecto de titulación.
- **Product Owner:** El tutor del proyecto de titulación.
- **Scrum Team:** El desarrollador del proyecto.
- **Stakeholders:** Director y Consejo de la Carrera de Ingeniería de Sistemas de la Universidad Politécnica Salesiana y tutor del proyecto de titulación.

Se realizaron 9 sprints, a los cuales se pudo dar seguimiento con la ayuda de la herramienta Taiga así como a las historias de usuario, tareas, puntos de historia.

El product backlog nos muestra el porcentaje de avance del proyecto, los puntos del proyecto, los puntos definidos y los puntos de sprint. Para el proyecto del sistema de

evaluación se han completado todos los sprints y debido a ajustes en el sistema se tiene un avance del 102%.

Product Backlog.

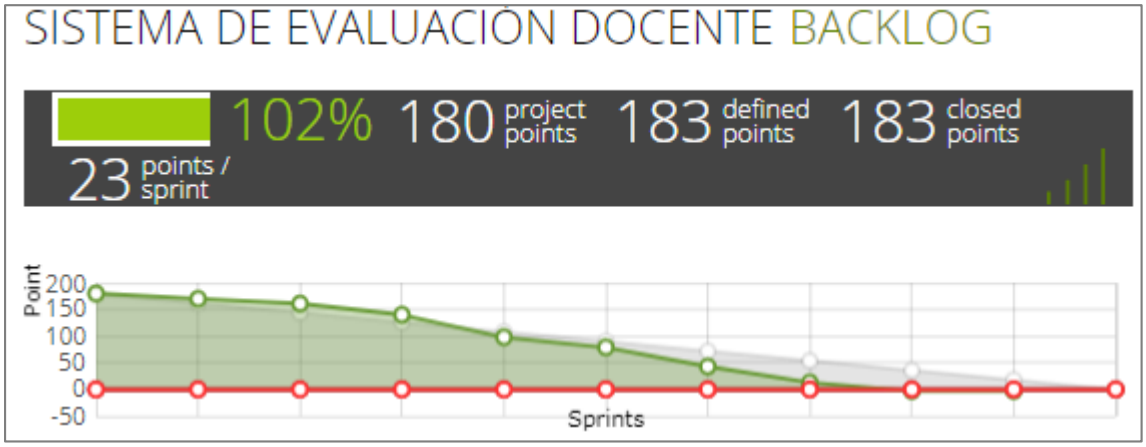



Figura 1. Progreso de todas las iteraciones al final del proyecto.

Elaborado por: Pablo Guerra.

Para cada sprint se creó un sprint backlog que contiene las tareas de cada sprint con los puntos respectivos además del estado de la tarea (en progreso, listo para pruebas, listo).

Sprint backlog.

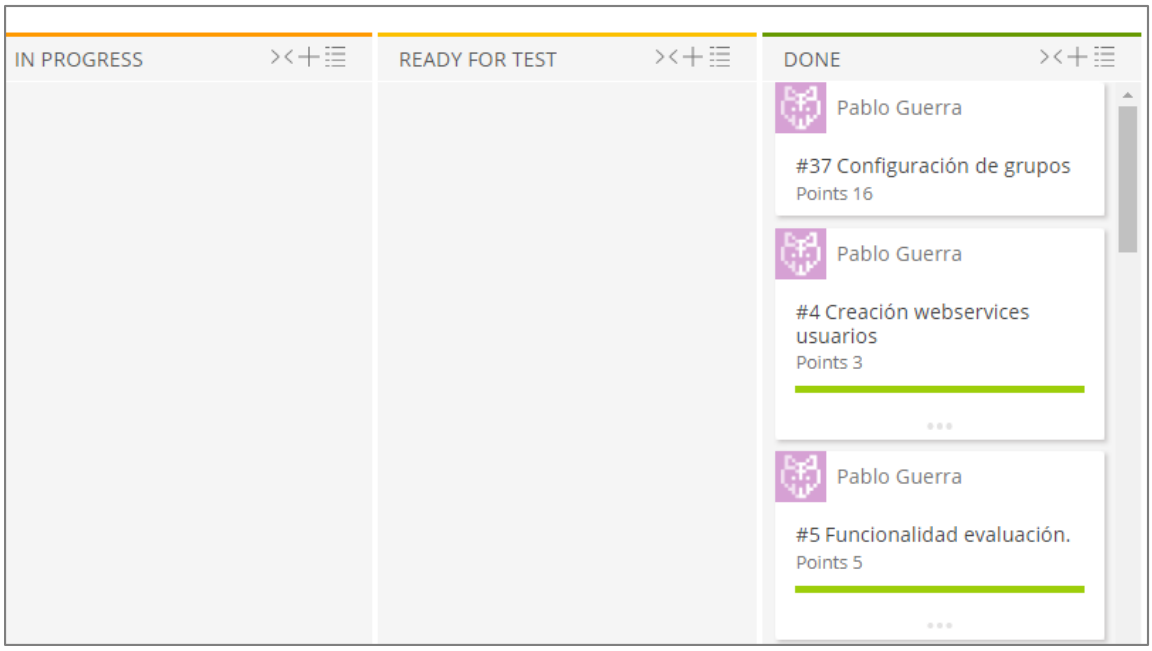



Figura 2. Todos los sprints (período de entregables) del proyecto.

Elaborado por: Pablo Guerra.

## *Taiga*

Taiga es una herramienta de gestión de proyectos SCRUM que soporta desde proyectos simples a proyectos grandes con granes equipos de trabajo, tiene un enfoque de código abierto y es una alternativa a desarrolladores que están utilizando una herramienta de desarrollo por primera vez. Aunque para uso personal es una herramienta gratuita, Taiga

posee varias funcionalidades que lo hacen una herramienta poderosa y que puede ser muy útil para gestionar proyectos.

## UML

Existen varios métodos para desarrollar sistemas de información orientados a objetos y debido a la naturaleza del proyecto se utilizaron diagramas del Lenguaje Unificado de Modelamiento UML para representar los componentes del y los casos de uso, estos diagramas ayudaron al análisis de la solución que se debía dar para que el proceso planteado fuera eficiente y simple a la vez.

Para el desarrollo de este sistema existieron 2 escenarios principales que son el núcleo del sistema.

En el análisis y diseño se evaluó el flujo del sistema y se modeló un caso de uso general en el que los actores completaban sus acciones y se obtenía el resultado esperado explicando el flujo del negocio. Esto sirvió como herramienta para exponer a las partes involucradas exactamente qué haría el sistema y cómo aportaría a la mejora del proceso vigente.

# Capítulo 1

# Marco teórico

## Evaluación de desempeño

Históricamente, desde tiempos bíblicos se ha dado mucha importancia a la evaluación, desde diferentes contextos se han utilizado los resultados de evaluaciones para la toma de decisiones con respecto a los evaluados y el trabajo que realizan, el concepto general de la evaluación de desempeño es comprobar el grado de cumplimiento de objetivos planteados a una persona. Se evalúa con el fin de promocionar, castigar o simplemente obtener información sobre la actividad realizada y los resultados esperados.

"La evaluación es el proceso de identificar, obtener y proporcionar información útil y descriptiva acerca del valor y mérito de las metas, la planificación, la realización y el impacto de un objeto determinado, con el fin de servir de guía para la toma de decisiones, solucionar los problemas de responsabilidad y promover comprensión de los fenómenos implicados" (Stufflebeam, 1971).

Con la industrialización, se dio un enfoque más científico a la evaluación, habilitando a las personas a la realización de ciertos trabajos únicamente si demostraban que poseían las habilidades para hacerlo, con el tiempo no solo la habilidad fue suficiente para definir la competencia de una persona para la ejecución de un trabajo, también se comenzó a medir la eficiencia y así la evaluación fue más rigurosa, con más elementos evaluadores

fue un factor crítico en el proceso de definición de las competencias de las personas para realizar trabajos.

## *Evaluación de desempeño docente*

En las últimas décadas los sistemas educativos, especialmente latinoamericanos, "han identificado el desempeño profesional del maestro como un factor determinante del logro de la calidad de la educación. Diversos estudios concluyen que podrán perfeccionarse los planes de estudio, programas de textos escolares, construirse magníficas instalaciones; obtener excelentes medios de enseñanza, pero sin docentes eficientes, no podrá tener lugar el perfeccionamiento de la educación" (Vélez 2003). Y en los últimos años en Ecuador, se ha dado especial énfasis a la calidad educativa, realizando evaluaciones de habilitación a los docentes para que puedan dictar clases en unidades educativas de todos los niveles, categorizando a las instituciones educativas y proponiendo la forma de llegar a la excelencia educativa mejorando la instrucción de sus docentes así como los recursos y el ambiente en el que se formarán los estudiantes.

Existen indicadores de desempeño que reflejan cómo un docente está llevando a cabo su actividad, generalmente los indicadores son:

- A través de los logros de sus estudiantes.
- A través de instrumentos que midan la habilidad del docente.
- En base a la opinión de los docentes (autoevaluación).
- En base a la opinión de los supervisores, directores y otras autoridades.

## Modelo a implementar

Las características del nuevo modelo evaluación de docentes son:

- Establece las tareas.
- Explica y detalla las tareas.
- Clasifica las tareas en categorías (categorías de tareas).
- Establece las condiciones de evaluación.
- Recopila los datos de la tarea.
- Explica la tarea entregada.
- Comprueba y valida la tarea.
- Evalúa la tarea.
- Recopila datos del objetivo.

El modelo tiene un flujo con varias entradas pero sólo una salida que es el resultado de la evaluación de la tarea.

Flujo de modelo de evaluación.

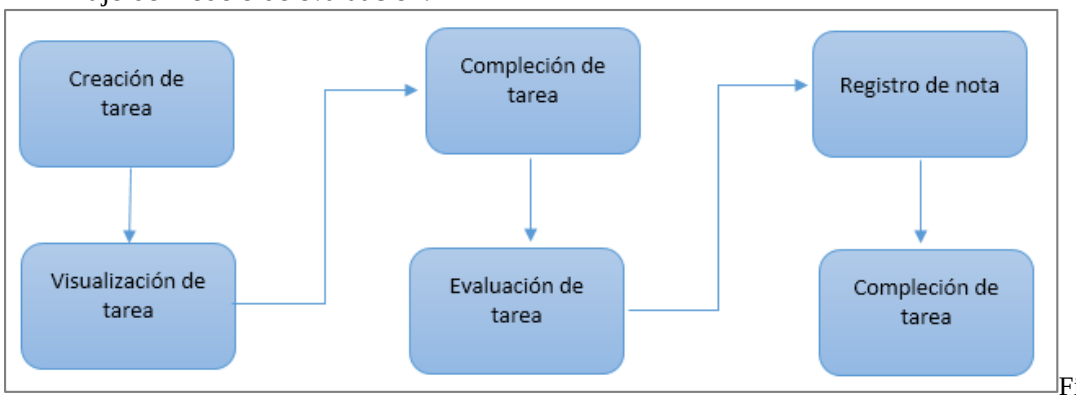



Figura 3. Flujo de evaluación de tareas desde la creación a la compleción.
Elaborado por: Pablo Guerra.

## SCRUM

Es una metodología ágil, utilizada generalmente en proyectos cortos (menos de un año) conformados por equipos pequeños o medianos. Se caracteriza por organizar y manejar el trabajo de los integrantes del proyecto así como el tiempo en el que se desarrollarán las iteraciones. Las prácticas que definen a SCRUM lo hacen flexible y permiten que cada proyecto que lo utilice tenga un enfoque diferente de la metodología. Está basado en los valores de la honestidad, franqueza, coraje, respeto, enfoque, confianza, y colaboración (Kenneth, S. Rubin, 2012, pág. 13).

## Modelo vista controlador

El modelo vista controlador organiza una aplicación de manera que los datos, la lógica y la interfaz de usuario están separados y conectados por un módulo principal, en la aplicación se utilizó debido a la necesidad de dividir en capas las partes de la aplicación para una mejor mantenibilidad y perspectiva a futuro.

Aunque Spring (el framework utilizado para el backend) tiene una arquitectura MVC no fue necesaria la capa de vista y lo que dio funcionalidad a la aplicación sólo fue el modelo y el controlador.

## Base de datos

### *MySQL*

MySQL es un sistema de gestión bases de datos relacionales basado en el lenguaje de consulta estructurado SQL (Structured Query Language). Es uno de los motores de bases de datos más populares del mundo en parte porque tiene una naturaleza open source por lo que cualquiera puede usarlo gratuitamente, además puede ser ejecutado en la mayoría de plataformas incluyendo Linux UNIX y Windows, aunque se usa en una amplia gama de aplicaciones es común encontrarlo en aplicaciones web.

La elección de MySQL sobre otros motores de base de datos se dio debido a que por su gran popularidad tienen una gran comunidad que continuamente está dando solución a

problemas y bugs que presenta la plataforma, además que MySQL posee ciertas ventajas sobre otras plataformas (Tabla 1).

## Persistencia

La definición de persistencia según el diccionario es: *Duración permanente de una cosa* por lo que casi todas las aplicaciones requieren persistencia de datos, es decir, cuando el usuario cierre o apague la aplicación que está usando los datos que ha ingresado o modificado deben ser mantenidos de alguna manera, caso contrario la aplicación no sería muy práctica (Bauer, King, 2005, pág. 27). En programación cuando se habla de persistencia normalmente se habla de guardar datos en una base de datos relacional utilizando el lenguaje SQL y si se utiliza el lenguaje de programación Java existen varias formas de persistir los datos, ya sea nativamente con las librerías que tiene el lenguaje o con frameworks que facilitan la tarea y hacen que la aplicación sea más mantenible.

## Aplicaciones del lado del cliente vs lado del servidor

Una aplicación web completa se compone de dos partes: una del lado del cliente y otra del lado ser servidor. Existen diferencias importantes entre los dos lados:

Tabla 1. Aplicaciones del lado del cliente vs lado servidor.

| Criterio | Cliente | Servidor |
|---|---|---|
| Ejecución | Es renderizada por el navegador | Es ejecutada por un sistema operativo en un servidor |
| Disponibilidad | Tiene interacción directa con el usuario | No puede ser vista por el usuario |
| Datos | No puede leer datos directamente del servidor, debe comunicarse vía HTTP | Guarda información directamente en la base de datos |
| Desarrollo | Utiliza los lenguajes HTML, CSS y Javascript | La posibilidad de lenguajes a utilizar es muy amplia (Java, Javascript, Python, PHP) |

# Frameworks

## *Spring*

En el desarrollo de aplicaciones de software la complejidad es un problema que causa que una aplicación se vuelva inmantenible con el tiempo, Spring es un framework que permite que el desarrollo de aplicaciones complejas sea manejable, escalable, mantenible y estructurado, todo esto es posible debido a que incluye características como:

- Implementa el modelo vista controlador

- Inyección de dependencias
- Soporte nativo de JDBC, JPA, JMS
- Incluye librerías como hibernate, maven.
- Soporte para proyectos web

Spring establece un entorno de desarrollo dentro del cual tareas comunes que llevarían tiempo en implementar como compilación, pruebas, inyección de dependencias, conexión con bases de datos ya se encuentran incluidas en el núcleo por lo que lo único que debe hacer el desarrollador es configurar y comenzar a tomar ventaja de esas características.

### *Angular JS*

Angular JS es un framework estructural para aplicaciones web dinámicas. Utiliza HTML como lenguaje de interfaz y permite extender la sintaxis de HTML para expresar los componentes de la aplicación clara y sucintamente (What is Angular? 2010-2017). Tiene características que lo hacen un framework potente y liviano como inyección de dependencias, que con el uso de Javascript permite añadir y usar librerías ajenas a Angular fácilmente. Angular presenta una solución frontend completa, ya que puede manipular del el DOM y manejar todas las operaciones CRUD (Crear, leer, actualizar y eliminar) sin la necesidad de otras librerías. Una de las características que hacen que Angular JS sea un framework "superheroico" es la posibilidad de crear aplicaciones SPA (single page applications) que no siempre recargan ni cambian de página, en cambio obtiene secciones HTML las añade y las renderiza sin tener que recargar la página.

### *Hibernate*

El mapeo de bases de datos mediante el patrón objeto/relación en aplicaciones Java se lo realiza mediante archivos de configuración y clases. Hibernate es un framework que facilita el mapeo de las tablas representándolas como clases y por lo tanto objetos Java, estas clases son llamadas persistentes ya que en un proceso de la aplicación los datos obtenidos de la base son creados como objetos los cuales pueden ser manipulados en el proceso y devueltos a la base. Hibernate es el encargado de estas tareas, y presenta una solución confiable, eficiente, flexible y escalable.

### *Angular-material*

Material es un framework de interfaz de usuario creado por Google que comprende componentes basados en objetos materiales como luces, sombras y similares. Su interacción con el usuario ofrece una experiencia limpia, agradable a la vista y simple. Es una alternativa poderosa que no solo ofrece interfaz de usuario sino que contribuye a la lógica de la aplicación como tal ofreciendo características como:

- Ruteo
- Testing
- Conexión entre el modelo y la vista en el patrón MVC
- Servicios y Factories que encapsulan funcionalidad

Además trabaja en conjunto con Angular JS utilizando directivas y dotando a la aplicación de la posibilidad de contener lógica dentro del DOM.

## Ambientes de desarrollo

### *JBoss*

Es un servidor de aplicaciones parte de la colección de productos Red Hat que permite ejecutar archivos compilados Java, esto mediante la creación de servicios que en un punto determinado serán requeridos por la aplicación como conexión a base de datos, manejador de transacciones, servicios de seguridad, todos estos servicios son transparentes para el usuario que solo necesitará de archivos XML de configuración para atar los servicios a los componentes de la aplicación sin necesidad de programar. Esto hace de JBoss una eficaz herramienta ahorrando recursos y reduciendo el tiempo de integración y configuración que se requeriría para levantar una aplicación Java desde cero.

### *Node.js*

Es un software que ejecuta código escrito en lenguaje Javascript sin la necesidad de un navegador, está diseñado para el desarrollo de aplicaciones escalables, fue necesario en la construcción de ésta aplicación ya que para la ejecución del sistema frontend en Angular.js se necesitaba un servidor de aplicaciones Javascript y Node.js es el sistema que cumple las necesidades de la aplicación y se acopla muy bien con Angular.js

## Herramientas Javascript

### *Gulp*

En el desarrollo de aplicaciones web existen tareas que son comunes y que son requeridas muchas veces en el proceso de desarrollo, compilación y ejecución de la aplicación como la minificación, compilación y validación de código, levantamiento y recarga del servidor de aplicaciones y otras que pueden ser requeridas según las circunstancias del desarrollador. Babel es un conjunto de herramientas que automatiza tareas y reduce el tiempo de desarrollo de una aplicación. Se configura desde archivo Javascript que contiene todas las tareas y el momento en que se van a ejecutar.

### *Babel*

Con el avance de los lenguajes de programación, ofreciendo cada vez más versatilidad y específicamente la estandarización en un futuro de EcmaScript6 Babel es un compilador de código Javascript, permite adelantarse a la estandarización y escribir código en EcmaScript 6 y compilarlo a la versión estándar actual de EcmaScript (5) para que pueda ser usado en cualquiera de los navegadores modernos que aún no incluyen soporte de todas las características de EcmaScript 6.

### *BrowserSync*

Es una herramienta que ayuda en el proceso de pruebas de una aplicación, esto mediante la recarga automática de páginas cuando detecta que ha existido algún cambio en el

código; así el desarrollador no desperdicia tiempo y puede realizar pruebas a su software de una manera más eficiente.

## UML

El lenguaje unificado de modelamiento (Unified Modeling Language) es un lenguaje de modelamiento utilizado para especificar, visualizar, construir y documentar los artefactos de un sistema de software. (The Unified Modeling Language, 2005). Sirve para representar conceptos intangibles como decisiones, configuraciones, flujos de sistemas de software que ya existen o van a ser construidos. Se utiliza principalmente para planificar y entender cómo va a funcionar un sistema, contiene varios modelos que pueden representar diferentes conceptos como diagramas de flujo, diagramas de casos de uso, diagramas de clases, diagramas de bases de datos.

UML es un recurso muy valioso para la construcción de software ya que los programadores y arquitectos de software pueden expresar sus ideas y cada participante de la construcción puede entender claramente lo que el diseñador quiere y desea obtener.

# Buenas prácticas

## *Git*

Es una herramienta de versionamiento creada por Linus Torvalds en 2005, se usa principalmente para que en los equipos de trabajo no existan problemas ni confusión cuando varios desarrolladores suban o modifiquen partes del mismo código.

En este proyecto se decidió utilizar Git debido a su potente motor de versionamiento desde el cual se pueden subir cambios al repositorio, clonar proyectos, hacer comparaciones de código, revertir cambios y respaldar el código usando únicamente comandos.

# Capítulo 2

# Análisis y diseño

## Análisis del problema

### *Recolección de información*

Se tuvo una reunión con los docentes miembros del consejo de carrera de sistemas y se obtuvo detalles de cuál es el proceso actual de evaluación de docentes además se envió una encuesta a los docentes de la carrera para obtener más información. (Anexo 1).

Con los datos recolectados se obtuvo un panorama general y específico del proceso con lo que se pudo hacer un análisis de las fortalezas y debilidades del sistema actual y cómo se podría mejorar el proceso.

### *Proceso actual*

El proceso actual de evaluación de tareas a docentes consiste en que los jefes de área notifican a los docentes que deben subir documentos o cumplir ciertas actividades.

Para el caso de los documentos los docentes suben el archivo solicitado a una carpeta compartida, luego de la fecha de entrega notificada el jefe de área revisa los documentos de la carpeta y registra la valoración de la tarea en un documento.

## *Requisitos*

### Requisitos de usuarios

Verbalmente, en una reunión con el directorio de la carrera se definieron los siguientes requisitos del sistema:

- El sistema permitirá la evaluación de tareas de docentes por parte de los jefes de área.
- Las tareas serán guardadas en una base de datos.
- Existirán tareas comunes para todos los semestres.
- Se podrán asignar varios docentes a una tarea.
- Existirán dos tipos de tareas:
    - Tareas: deberán contener un archivo adjunto y un comentario.
    - Reuniones: contendrán los detalles de la reunión y podrán contener o no un archivo adjunto.
- Los criterios de evaluación de las tareas serán:
    - Cumplimiento de la tarea.
    - Calidad de la tarea.
    - Tiempo de entrega.

### Requisitos del sistema

Los requisitos para que el sistema esté funcionando en un ambiente de producción son:

- Un servidor java.
- Un servidor Javascript (*Node.js*).

- Una base de datos MySQL.

**Requisitos funcionales**

Las funcionalidades principales que deberá cumplir el sistema para cumplir sus objetivos son:

- Los usuarios se pueden dividir según sus roles.
- Debe dar acceso a diferentes partes del sistema según los roles.
- Los usuarios podrán ingresar al sistema con credenciales creadas por el administrador del sistema.
- El jefe de área debe ser capaz de crear tareas y asignar las tareas a los docentes.
- El jefe de área puede crear reuniones y cancelarlas.
- En la creación y envío de tareas se pueden subir archivos.

**Requisitos no funcionales**

Los requisitos del sistema no funcionales tratan de cumplir las buenas prácticas de desarrollo de software en temas de usabilidad y facilidad de uso, así mismo, tienen como objetivo presentar un sistema de la mejor calidad posible que cumpla con las expectativas de los docentes, jefes de área y la carrera en general para mejorar el proceso de evaluación:

- Flexibilidad.
- Escalabilidad.
- Seguridad.
- Mantenibilidad.

## Diagrama de clases conceptual

La estructura MVC establece una capa para cada parte de la aplicación (modelo, vista y controlador) pero debido a que la tecnología utilizada para el frontend no es la misma del backend sólo se utilizaron las partes de modelo y controlador.

Diagrama de clases conceptual.

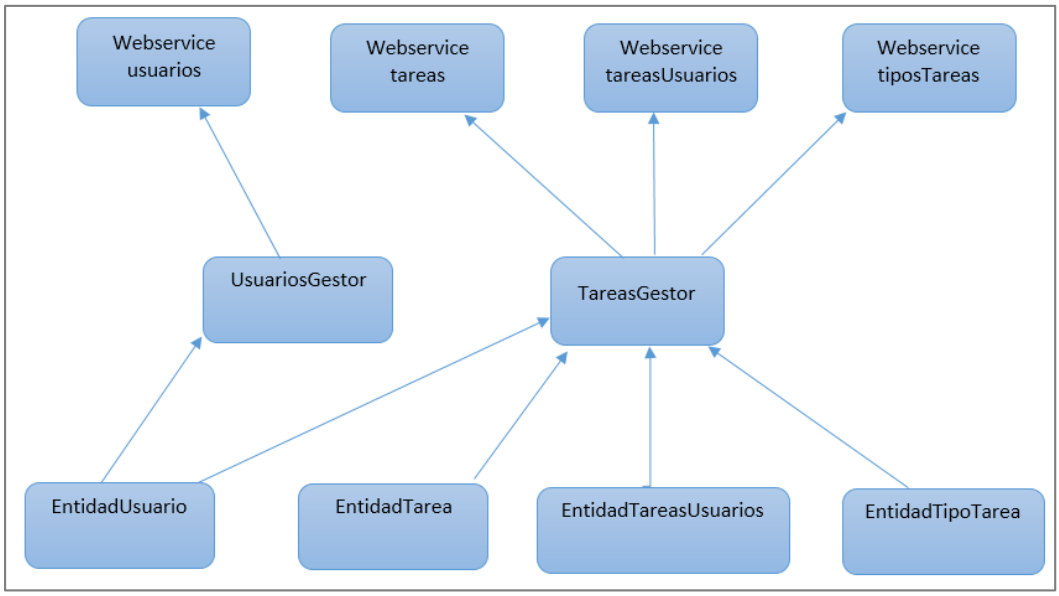



Figura 4. Clases principales usadas en el flujo de la aplicación.

Elaborado por: Pablo Guerra.

## Diagrama de base de datos conceptual

En el modelo entidad relación la tabla principal TareasUsuarios contiene claves foráneas que conectan las tareas, tipos de tareas y usuarios para que así en la consulta de tareas se puedan obtener todos los detalles según el requerimiento.

Diagrama de base de datos conceptual.

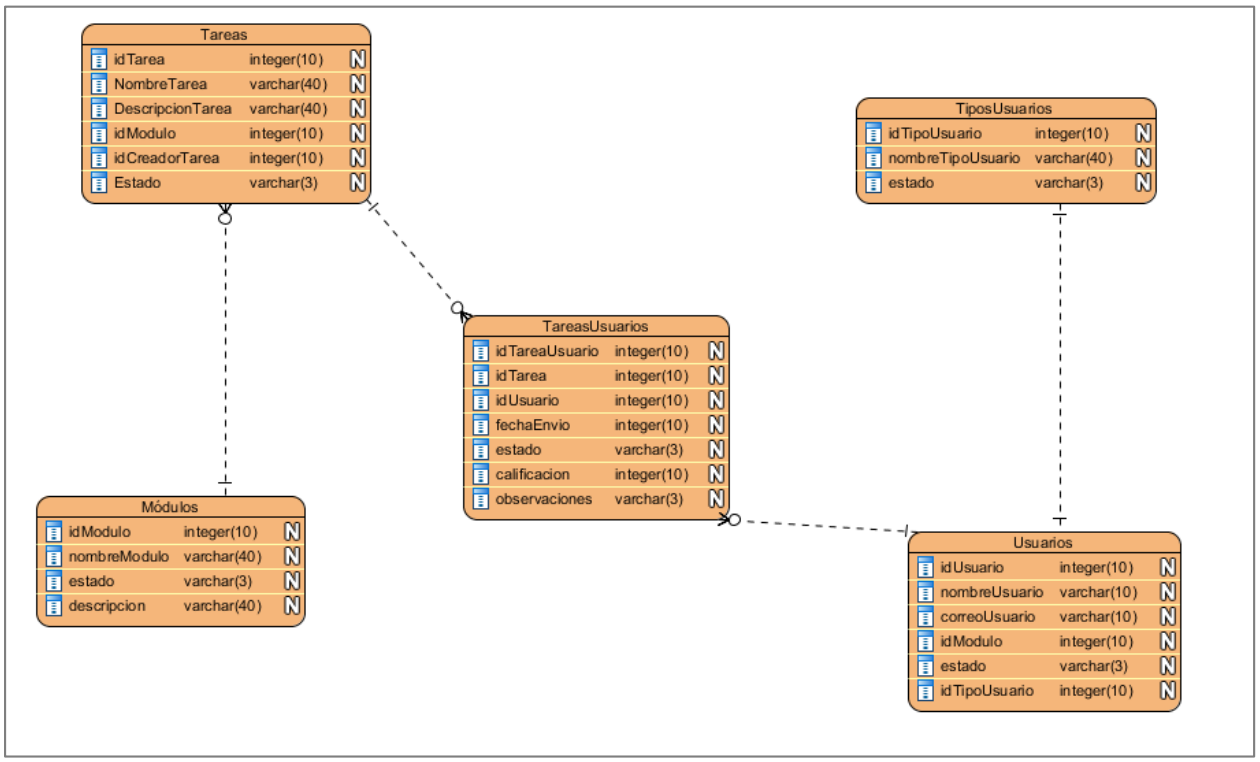



Figura 5. Entidades de la base de datos.

Elaborado por: Pablo Guerra.

## Interfaz de usuario

Luego del análisis del flujo del sistema se realizó una maqueta del módulo principal y más importante del sistema (Tareas).

Maqueta del módulo tareas.

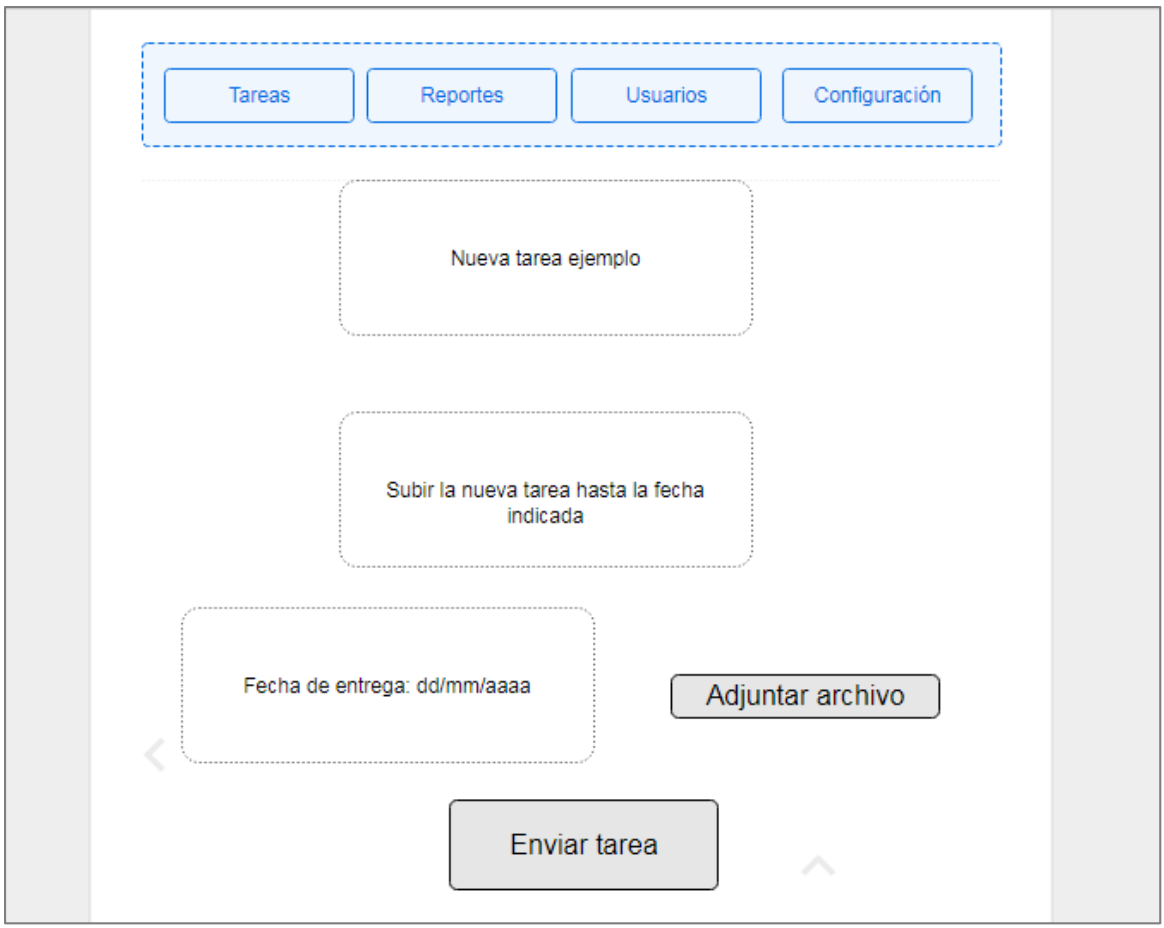



Figura 6. Maqueta de la pantalla del módulo de tareas (enviar tarea).

Elaborado por: Pablo Guerra.

Maqueta del módulo tareas (Crear tarea).

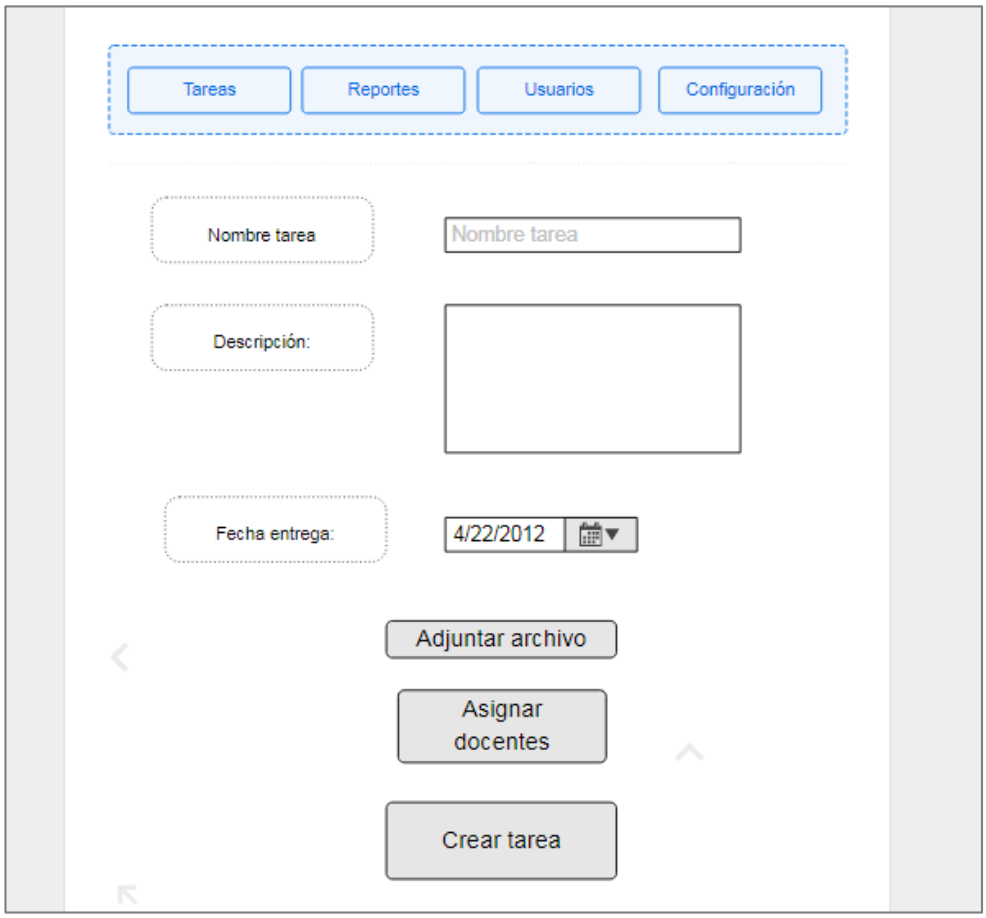



Figura 7. Maqueta de la estructura del módulo de tareas (crear tarea).

Elaborado por: Pablo Guerra.

## Casos de uso

### *Caso de uso principal*

Caso de uso principal

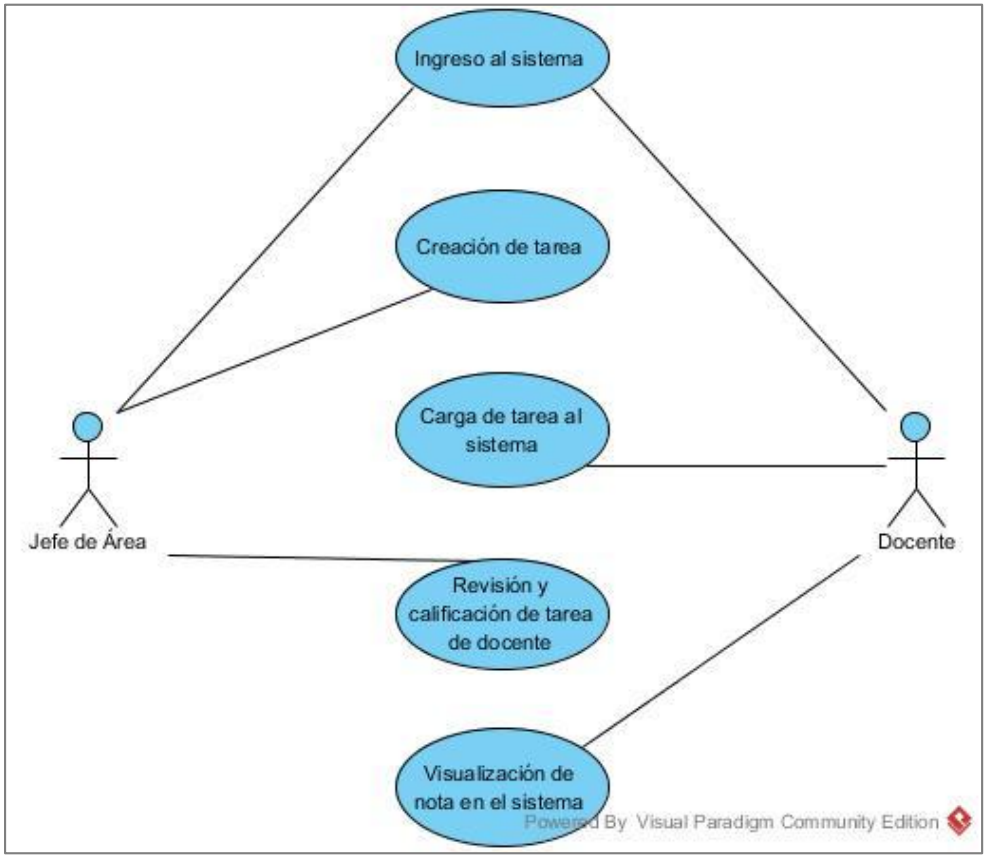



Figura 8. Comportamiento general del sistema.

Elaborado por: Pablo Guerra.

En la figura de caso de uso el jefe de área crea las tareas que estarán disponibles al docente, luego de creadas el docente carga la tarea al sistema para que el jefe de área pueda

visualizar y calificar, finalmente el docente tiene disponible la nota de su tarea en el sistema.

### *Administración de tareas*

Caso de uso administración de tareas.

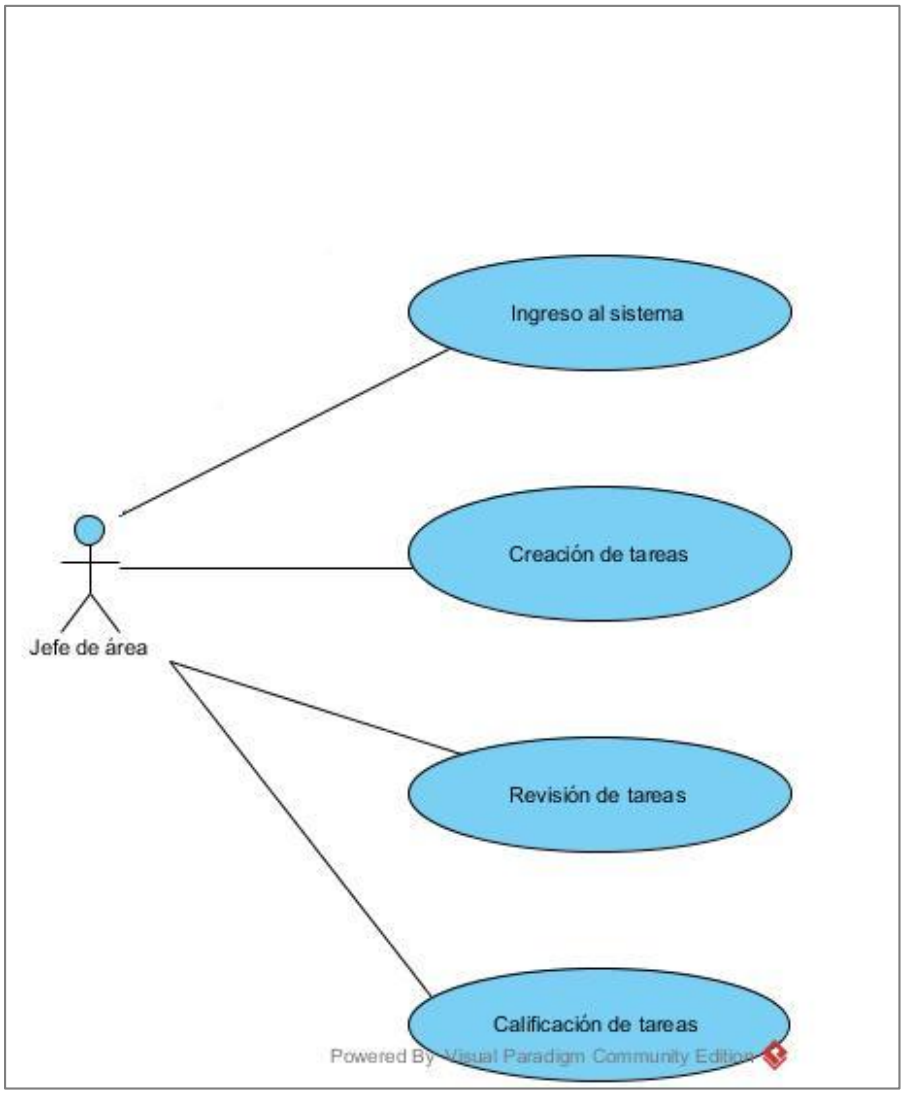



Figura 9. Comportamiento del sistema al administrar tareas.

Elaborado por: Pablo Guerra

Tabla 2. Ingreso al sistema

| | |
|---|---|
| **Caso de uso** | Ingreso al sistema |
| **Descripción** | Ingreso y verificación de nombre de usuario y contraseña |
| **Actores** | Jefe de área |
| **Precondiciones** | Tener registrado un usuario y contraseña en el sistema |
| **Postcondiciones** | Ingreso al sistema exitoso |
| **Escenario principal** | • El jefe de área ingresa su nombre de usuario y contraseña<br>• El sistema verifica las credenciales e ingresa al sistema |
| **Flujo alternativo** | • Las credenciales no son correctas<br>• Se muestra un mensaje de credenciales incorrectas |

Tabla 3. Creación de tarea

| | |
|---|---|
| **Caso de uso** | Creación de tarea |
| **Descripción** | El jefe de área registra una tarea en el sistema |
| **Actores** | Jefe de área |
| **Precondiciones** | Haber ingresado al sistema |
| **Postcondiciones** | Tarea registrada y asignada a docentes |
| **Escenario principal** | • Ingreso al módulo de tareas<br>• Selecciona opción nueva tarea<br>• Selecciona el tipo de tarea (Tarea)<br>• Ingresa los detalles de la tarea |

| | |
|---|---|
| | • Presiona el botón guardar |
| **Flujo alternativo** | • Ingreso al módulo de tareas<br>• Ingreso al módulo de tareas<br>• Selecciona opción nueva tarea<br>• Selecciona el tipo de tarea (Tarea)<br>• Ingresa los detalles de la tarea incompletos<br>• Mensaje de datos de tarea incompletos |

Tabla 4. Revisión de tareas

| | |
|---|---|
| **Caso de uso** | Revisión de tareas |
| **Descripción** | Ingreso y revisión de tarea de docente |
| **Actores** | Jefe de área |
| **Precondiciones** | Seleccionar una tarea enviada |
| **Postcondiciones** | Tarea abierta y marcada como leída |
| **Escenario principal** | • Ingreso al módulo de tareas<br>• Selecciona tarea en lista de tareas enviadas<br>• Observa los detalles de la tarea |
| **Flujo alternativo** | |

Tabla 5. Calificación de tareas

| | |
|---|---|
| **Caso de uso** | Calificación de tareas |
| **Descripción** | El jefe de área selecciona y registra calificación de tarea |
| **Actores** | Jefe de área |
| **Precondiciones** | Haber ingresado a los detalles de la tarea |
| **Postcondiciones** | Tarea registrada como calificada |
| **Escenario principal** | • Selecciona una tarea de la lista<br>• Observa los detalles y selecciona calificar<br>• Selecciona los detalles de calificación<br>• Envía la calificación |
| **Flujo alternativo** | • Selecciona una tarea de la lista<br>• Observa los detalles y selecciona calificar<br>• Selecciona los parámetros de calificación incompletos<br>• Se muestra un mensaje que no ha seleccionado todos los parámetros de calificación |

## *Registro de Reuniones*

Registro de reuniones

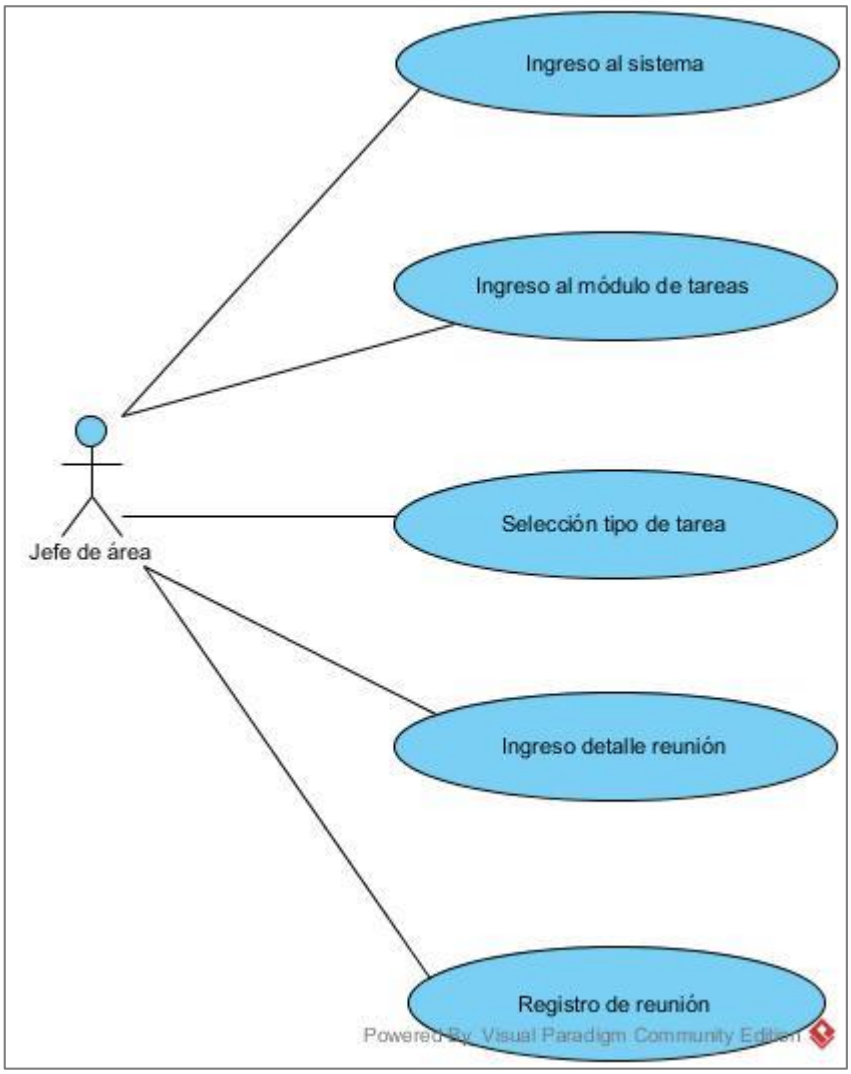



Figura 10. Comportamiento del sistema al registrar reuniones.

Elaborado por: Pablo Guerra.

Tabla 6. Ingreso al sistema

| | |
|---|---|
| **Caso de uso** | Ingreso al sistema |
| **Descripción** | Autenticación en el sistema |
| **Actores** | Jefe de área |
| **Precondiciones** | Tener una cuenta registrada en el sistema |
| **Postcondiciones** | Ingreso exitoso al sistema |
| **Escenario principal** | • Ingresa nombre de usuario<br>• Ingresa contraseña<br>• Selecciona ingresar<br>• El sistema verifica los datos de autenticación<br>• Ingresa a la pantalla principal del sistema |
| **Flujo alternativo** | • Ingresa nombre de usuario<br>• Ingresa contraseña<br>• Selecciona ingresar<br>• El sistema verifica los datos de autenticación<br>• Se muestra un mensaje de credenciales inválidas |

Tabla 7. Ingreso al módulo de tareas

| | |
|---|---|
| **Caso de uso** | Ingreso al módulo de tareas |
| **Descripción** | Ingresa al módulo de administración de tareas |
| **Actores** | Jefe de área |
| **Precondiciones** | Haberse autenticado en el sistema |

| | |
|---|---|
| **Postcondiciones** | Se encuentra en la pantalla principal del módulo de tareas |
| **Escenario principal** | • Se autentica en el sistema<br>• Selecciona tareas en el menú principal<br>• Ingresa al módulo de tareas |
| **Flujo alternativo** | |

Tabla 8. Selección tipo de tarea

| | |
|---|---|
| **Caso de uso** | Selección tipo de tarea |
| **Descripción** | Selecciona reunión como tipo de tarea |
| **Actores** | Jefe de área |
| **Precondiciones** | Haber ingresado en el módulo de tareas |
| **Postcondiciones** | La tarea requerirá los detalles específicos de tipo reunión |
| **Escenario principal** | • Ingreso al módulo de tareas<br>• Selecciona nueva tarea<br>• En tipo de tarea selecciona Reunión |
| **Flujo alternativo** | • Ingreso al módulo de tareas<br>• Selecciona nueva tarea<br>• En tipo de tarea selecciona Tarea |

Tabla 9. Ingreso detalle reunión

| | |
|---|---|
| **Caso de uso** | Ingreso detalle reunión |
| **Descripción** | Ingresa los detalles y el archivo (si existe) de la tarea tipo reunión |
| **Actores** | Jefe de área |
| **Precondiciones** | Haber seleccionado Reunión como tipo de tarea |
| **Postcondiciones** | El formulario de reunión lleno y listo para el registro |
| **Escenario principal** | • Selecciona tarea tipo reunión<br>• Llena los campos de los detalles de reunión<br>• Selecciona archivo adjunto |
| **Flujo alternativo** | • Selecciona tarea tipo reunión<br>• Llena los campos de los detalles de reunión |

Tabla 10. Registro de reunión

| | |
|---|---|
| **Caso de uso** | Registro de reunión |
| **Descripción** | Registra la reunión en la base de datos |
| **Actores** | Jefe de área |
| **Precondiciones** | Haber llenado los campos obligatorios para el tipo de tarea Reunión |
| **Postcondiciones** | Reunión registrada en la base de datos |

| | |
|---|---|
| **Escenario principal** | • Ingresa al módulo de tareas<br>• Selecciona nueva tarea<br>• Selecciona tipo de tarea Reunión<br>• Llena los campos del detalle de la reunión<br>• Selecciona los docentes asignados a la reunión<br>• Presiona “Guardar”<br>• La reunión se registra en la base de datos |
| **Flujo alternativo** | • Ingresa al módulo de tareas<br>• Selecciona nueva tarea<br>• Selecciona tipo de tarea Reunión<br>• No llena los campos obligatorios del detalle de la reunión<br>• Presiona “Guardar”<br>• Se muestra un mensaje que no se han llenado todos los campos obligatorios de los detalles de la reunión |

## *Envío de tareas*

Caso de uso envío de tareas

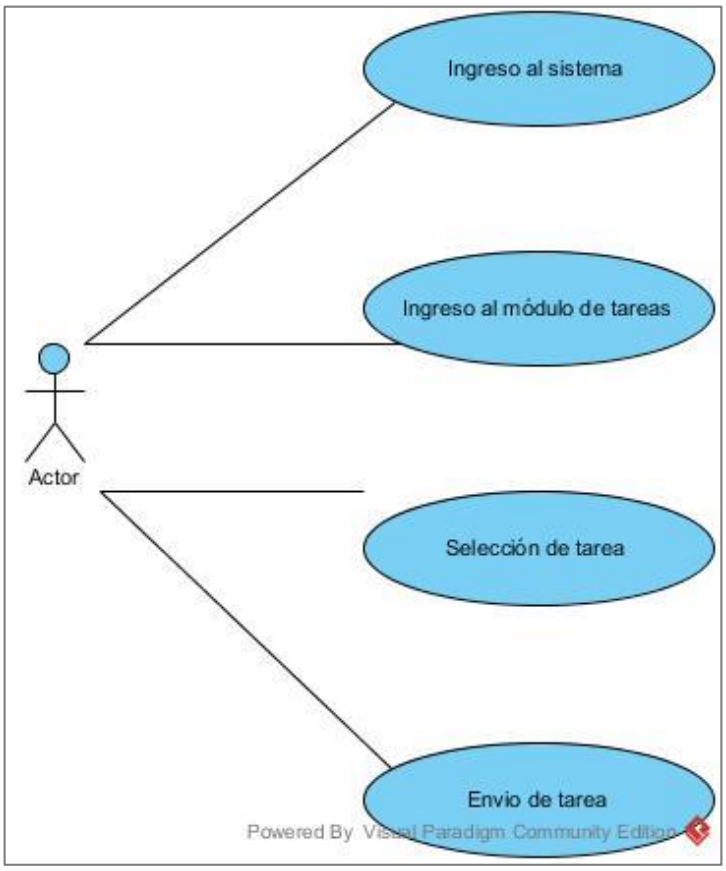



Figura 11. Interacción del usuario al enviar tareas.

Elaborado por: Pablo Guerra.

Tabla 11. Ingreso al módulo de tareas

| | |
|---|---|
| **Caso de uso** | Ingreso al módulo de tareas |
| **Descripción** | Ingresa a la pantalla principal del módulo de tareas |
| **Actores** | Docente |
| **Precondiciones** | Haberse autenticado en el sistema |
| **Postcondiciones** | Ingresar al módulo principal de tareas |
| **Escenario principal** | • Se autentica en el sistema<br>• Selecciona tareas desde el menú principal<br>• Ingresa al módulo de tareas |
| **Flujo alternativo** | |

Tabla 12. Selección de tarea

| | |
|---|---|
| **Caso de uso** | Selección de tarea |
| **Descripción** | Selecciona una tarea registrada por el jefe de área |
| **Actores** | Docente |
| **Precondiciones** | Haber ingresado al módulo de tareas |
| **Postcondiciones** | Entrar a los detalles de la tarea |
| **Escenario principal** | • Ingresa al módulo de tareas<br>• Selecciona de la lista de tareas registradas la tarea a enviar<br>• Ingresa a los detalles de la tarea |
| **Flujo alternativo** | |

Tabla 13. Envío de tarea

| | |
|---|---|
| **Caso de uso** | Envío de tarea |
| **Descripción** | Envía la tarea y se registra como enviada en la base de datos |
| **Actores** | Docente |
| **Precondiciones** | Haber ingresado a los detalles de la tarea |
| **Postcondiciones** | La tarea se registra como enviada y no aparece en la lista de tareas pendientes del docente |
| **Escenario principal** | • Selecciona una tarea de la lista de tareas pendientes<br>• Ingresa a los detalles de la tarea<br>• Llena el campo comentarios<br>• Selecciona y adjunta un archivo<br>• Presiona “Enviar”<br>• La tarea se encuentra registrada como enviada |
| **Flujo alternativo** | • Selecciona una tarea de la lista de tareas pendientes<br>• Ingresa a los detalles de la tarea<br>• Llena el campo comentarios<br>• No selecciona un archivo<br>• Presiona “Enviar”<br>• Se muestra un mensaje de alerta que debe seleccionar un archivo adjunto |

## Diagramas de clases

### *Controladores*

Los controladores son las clases que mapean los servicios web para que puedan ser consumidos por la aplicación del cliente.

Diagrama de clases controladores usuarios.

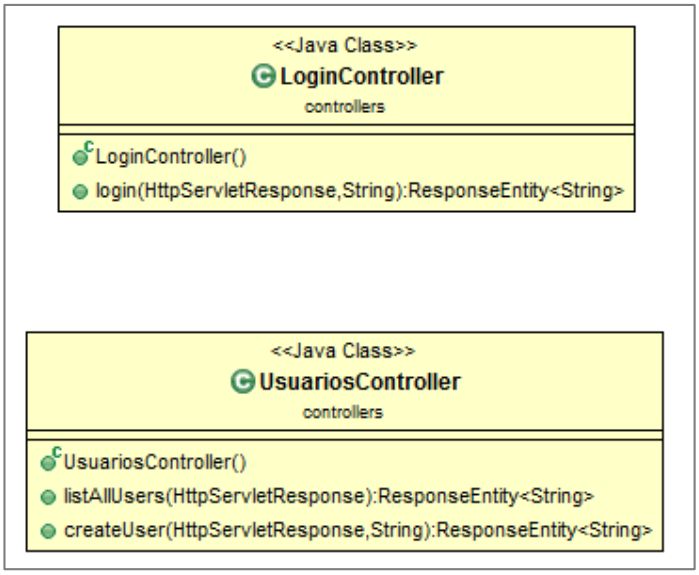



Figura 12. Clases de administración de usuarios.

Elaborado por: Pablo Guerra.

Diagrama de clases controladores Tareas

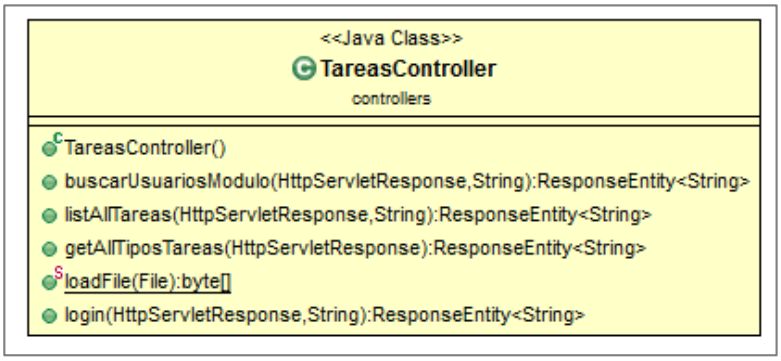



Figura 13. Clases de administración de tareas.

Elaborado por: Pablo Guerra.

## *Utilitarios*

Son clases que cumplen tareas específicas como la inserción de datos, consultas, procesamiento de datos.

Diagrama de clases Utilitarios.

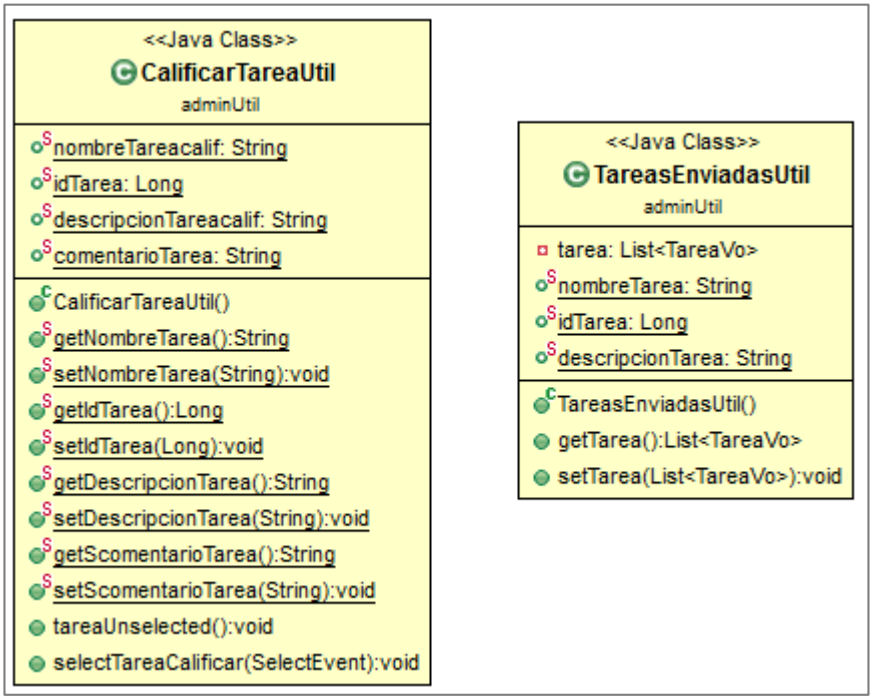



Figura 14. Clases utilitarias suplementarias.

Elaborado por: Pablo Guerra.

## *Entidades*

Son clases que representan tablas de la base de datos a los que se las puede modificar realizando operaciones SQL desde programación.

Diagrama de clases entidades.

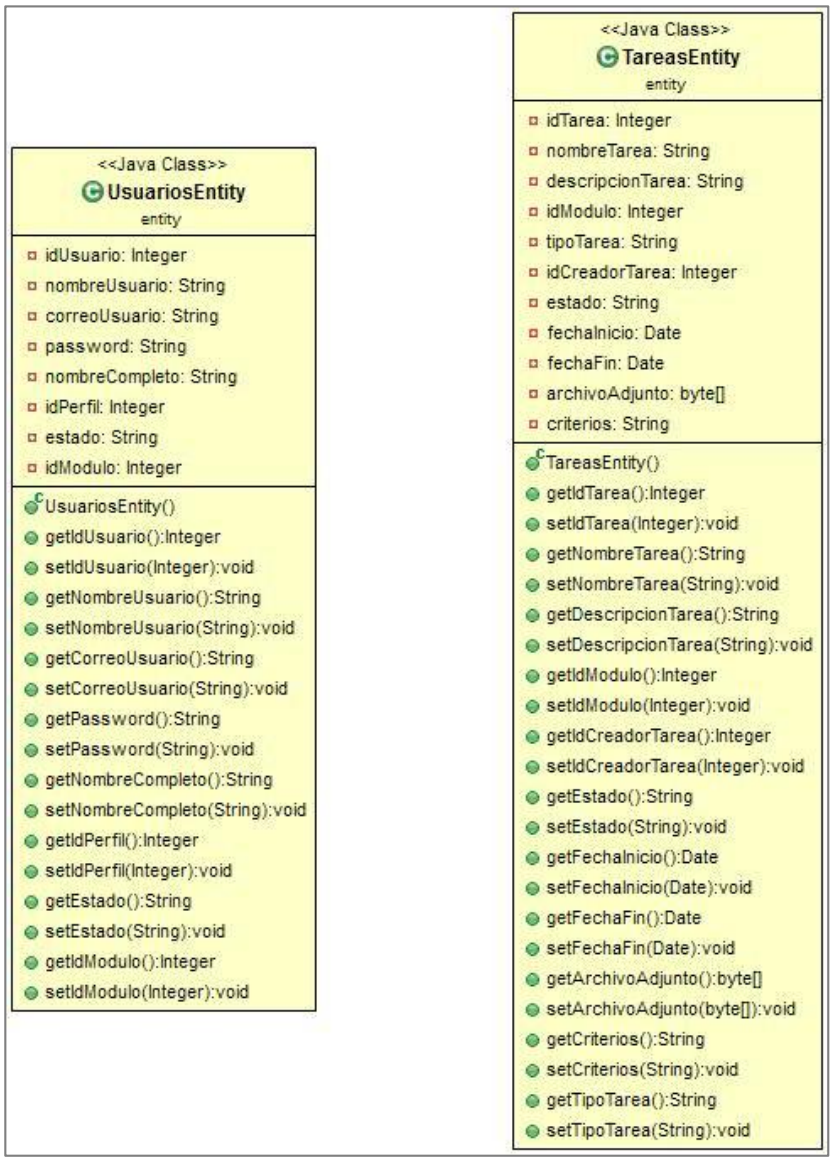



Figura 15. Entidades que representan tablas de la base de datos.

Elaborado por: Pablo Guerra.

## *View objects*

Los view objects representan una estructura de petición o respuesta que viene en formato JSON y es lo que se devuelve y se requiere como parámetro en los servicios web.

Diagrama de clases entidades.

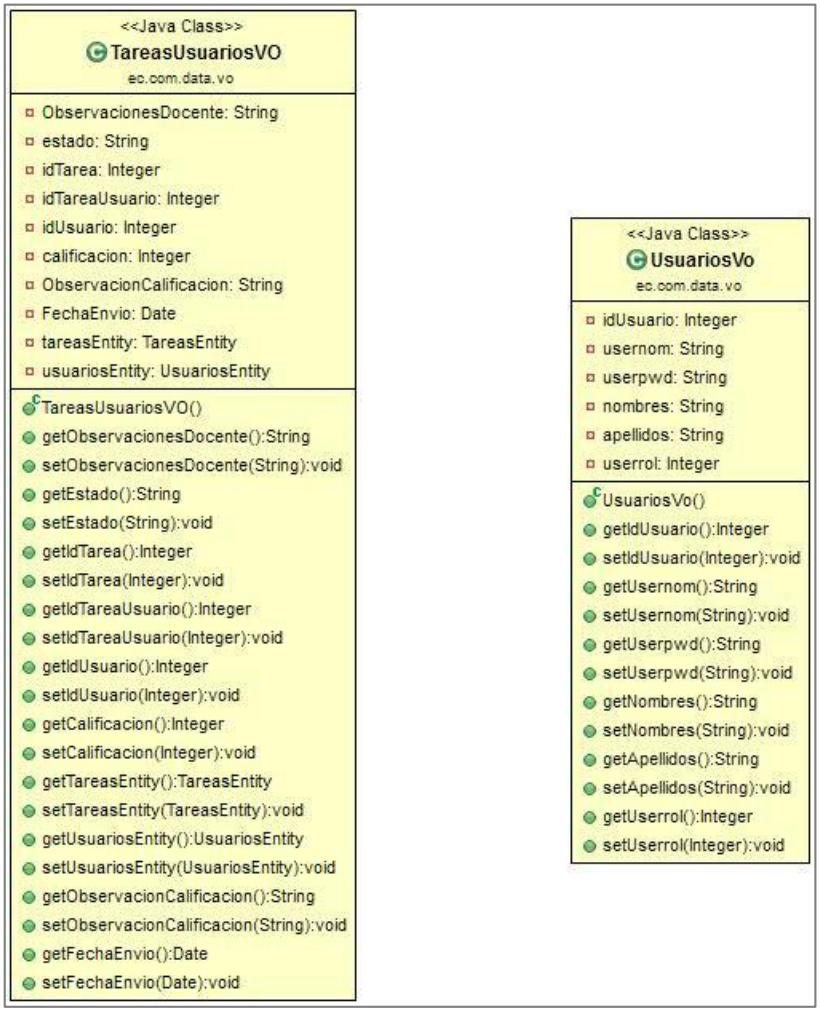



Figura 16. Entidades que representan tablas de la base de datos.

Elaborado por: Pablo Guerra.

## Interfaz de usuario

### *Angular Material*

En la fase de análisis y diseño se decidió usar el framework Angular.JS por lo que un framework de interfaz de usuario que se acopla perfectamente a AngularJS es Angular Material, un framework creado por Google que posee propiedades y directivas propias de AngularJS por lo que su integración no tuvo mayor impacto en el desarrollo del sistema.

# Capítulo 3

# Construcción y pruebas

## Modelo vista controlador

El modelo vista controlador organiza una aplicación de manera que los datos, la lógica y la interfaz de usuario están separados y conectados por un módulo principal, en la aplicación se utiliza este modelo en el backend con el framework Spring y en el frontend con AngularJS, el objetivo principal de este patrón de diseño de software es la mantenibilidad, añadiendo modularidad y organizando el código de forma tal que si existen cambios en la lógica, datos o interfaz éstos sean realizados en el módulo correspondiente.

Los objetos que conforman el patrón de diseño son:

### Modelo

Los objetos modelo representan la información de la aplicación y la forma cómo se van a manipular y procesar esta información, si existen cambios en la lógica del negocio, el modelo no va  a cambiar ya que el controlador utiliza únicamente los métodos accesores a los objetos del modelo.

### Vista

Los objetos que representan la vista son los que el usuario final puede ver, interactuar y con los que se manipulará los datos del modelo. La vista se comunica con el modelo mediante el controlador.

### Controlador

Los objetos controlador son los encargados de conectar la lógica del negocio con los datos (modelo) y la interfaz (vista), el usuario final interactúa con la aplicación en la vista, sus acciones son interpretadas en el controlador y modificadas en el modelo.

## Base de datos

La base de datos utiliza un modelo entidad relación donde las tareas son creadas como un registro único y se unen mediante clave foránea con la tabla TareasUsuarios que contiene un id de tarea y un registro por cada docente que se le asignó la tarea. Los usuarios (docentes o jefes de área) tienen asignados perfiles y cada uno tiene acceso o no a módulos específicos del sistema.

La tabla de criterios contiene los parámetros de evaluación de tareas con lo que cada uno mostrará diferentes componentes al momento de evaluar una tarea que ha sido enviada.

## Catálogo de clases

Tabla 14. Catálogo de clases.

| Nombre | Ubicación | Función |
|---|---|---|
| LoginController | ec.com.controllers | Controlador que contiene servicios y operaciones para el logueo del usuario. |
| NotasController | ec.com.controllers | Controlador que gestiona las peticiones de notas. |
| ReportesController | ec.com.controllers | Controlador que contiene servicios para la creación de reportes. |
| TareasController | ec.com.controllers | Controlador que contiene servicios y operaciones para la administración de tareas |
| UsuariosController | ec.com.controllers | Controlador de servicios para la gestión de usuarios |
| UsuariosVO | ec.com.data.vo | Clase que permite el transporte de datos de usuario desde el cliente. |
| TareasVO | ec.com.data.vo | Clase para la transformación de una cadena de texto con propiedades en un objeto Java. |
| ReporteVO | ec.com.data.vo | Clase para la petición de reportes con diferentes parámetros. |
| NotasGestor | ec.com.data.vo | Clase que gestiona operaciones relevantes a las notas de tareas de usuarios. |

## Catálogo de servicios

Tabla 15. Catálogo de servicios.

| Nombre | Ubicación | Función |
|---|---|---|
| summary | LoginController | Obtiene un resumen de las tareas del usuario. |
| login | LoginController | Recibe el nombre y contraseña del usuario y verifica si los datos coinciden. |
| periodos | LoginController | Obtiene la lista de períodos activos e inactivos registrados. |
| crearPeriodo | LoginController | Crea un nuevo período con estado inactivo. |
| actualizarPeriodoActivo | LoginController | Actualiza el período enviado y cambia su estado a activo. |

| | | |
|---|---|---|
| modulos | UsuariosController | Obtiene todos los módulos activos. |
| usuariosModulo | UsuariosController | Obtiene los usuarios de un módulo. |
| usuarios | UsuariosController | Obtiene los usuarios activos. |
| crearUsuario | UsuariosController | Crea un registro de usuario en la base de datos. |
| actualizarUsuario | UsuariosController | Actualiza los datos de un usuario por id. |
| tareas | TareasController | Obtiene la lista de tareas de un usuario. |
| crearTarea | TareasController | Crea un registro de una tarea. |
| enviarTarea | TareasController | Actualiza la tarea al momento de enviar y calificar. |
| crearReporte | ReportesController | Recibe los parámetros y devuelve los datos de las notas de docentes para crear un reporte. |

## Arquitectura servicios modelo vista controlador

La comunicación entre el cliente y el servidor es mediante los controladores y éstos a su vez tienen acceso a las otras capas de la aplicación por lo que el controlador de cada lado es el puente de conexión entre la interfaz de usuario y los datos.

Arquitectura servicios modelo vista controlador.

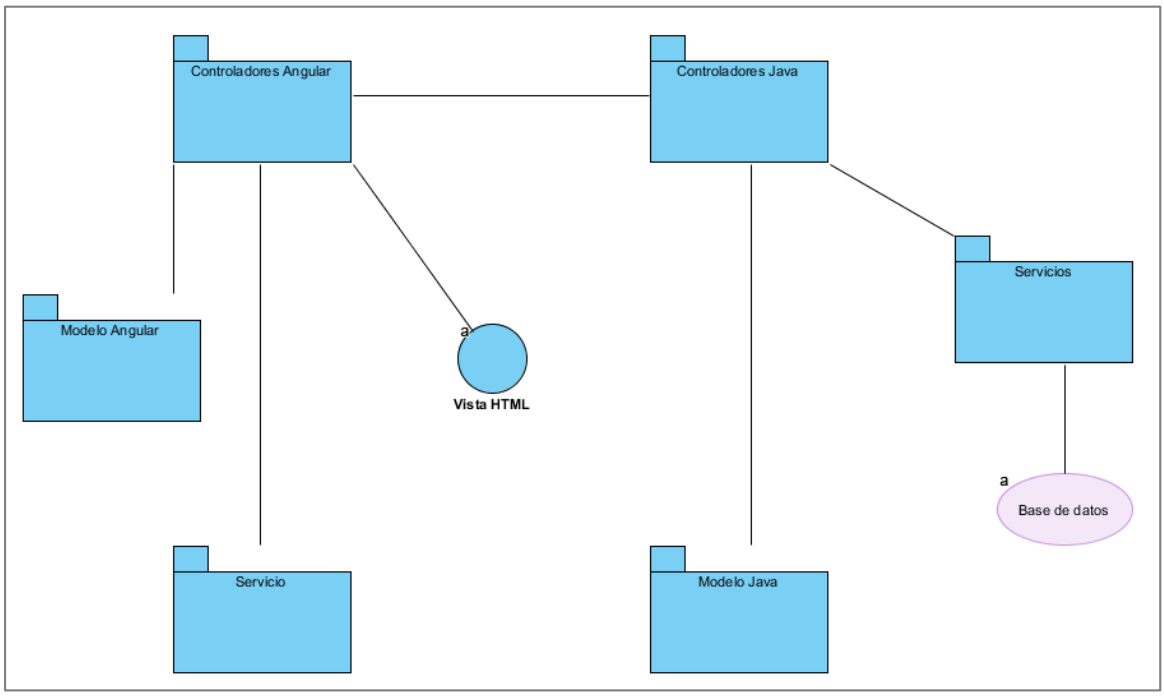



Figura 17. Comunicación entre los controladores, el modelo y la presentación del sistema.
Elaborado por: Pablo Guerra.

## *Diagrama de base de datos físico*

En el proceso de construcción surgieron nuevos requerimientos y nuevas tablas fueron agregadas a las existentes en el modelo conceptual.

Diagrama de base de datos.

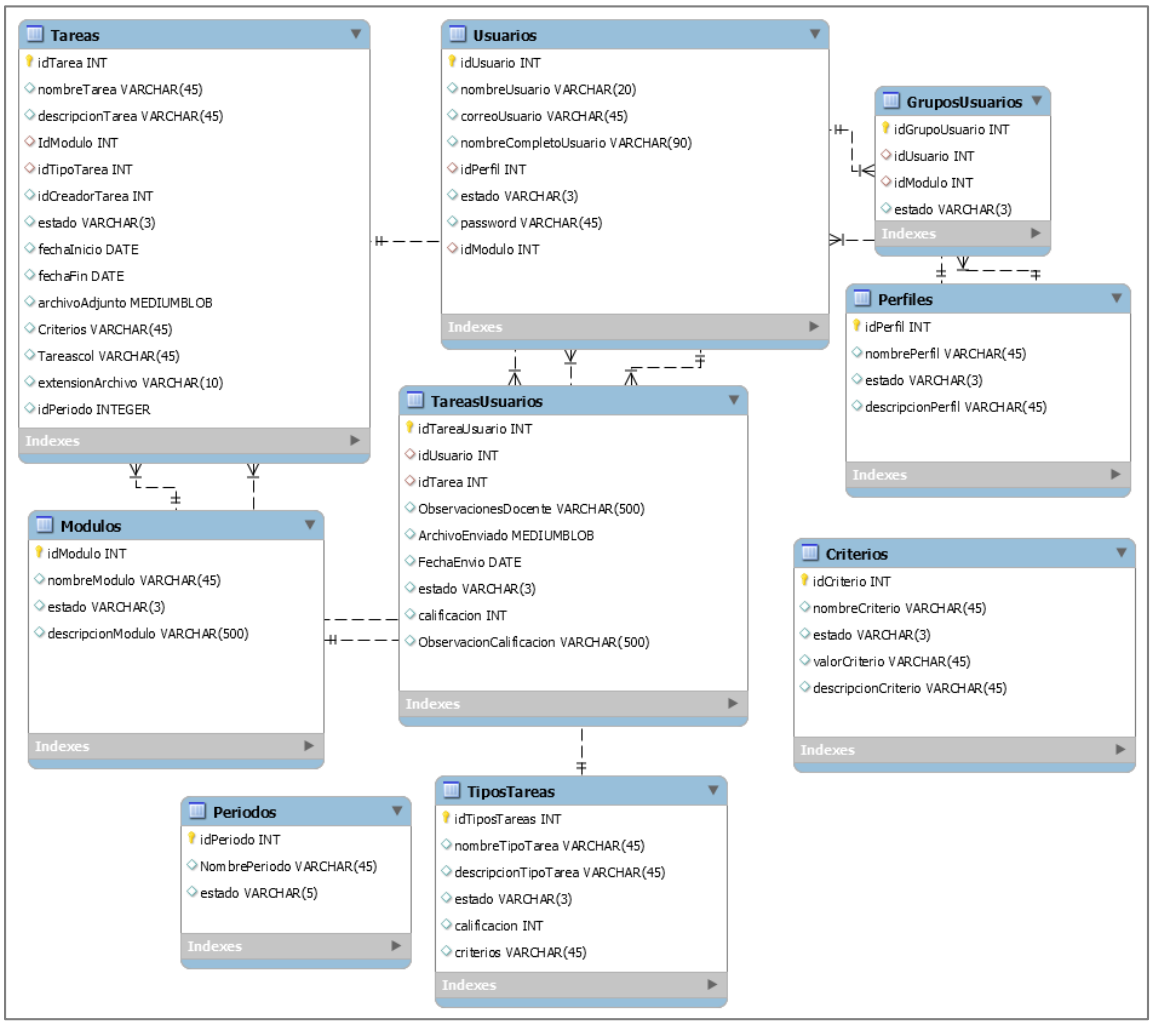



Figura 18. Diagrama general de la base de datos.

Elaborado por: Pablo Guerra.

## Arquitectura de servicios Web

El usuario final interactúa con los datos mediante el servidor de aplicaciones Javascript (Node.js) que llama a los servicios web publicados de la aplicación java. La forma en que se envía y recibe respuestas desde y hacia los servicios web es en texto en formato JSON en el que se puede representar estructuras de datos básicas como objetos y arreglos.

Arquitectura de servicios web.

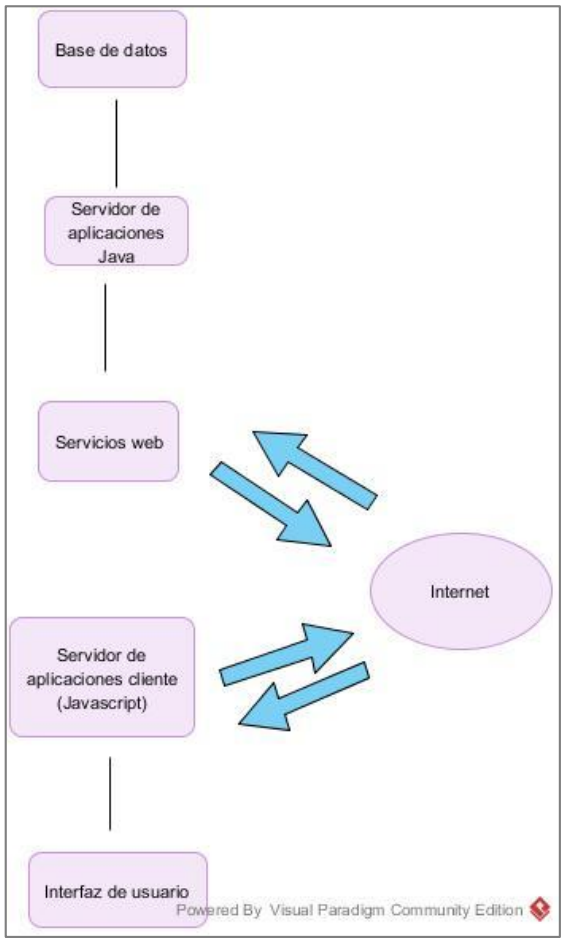



Figura 19. Comunicación de los servicios del sistema.

Elaborado por: Pablo Guerra.

# Diagrama de arquitectura

Diagrama de arquitectura.

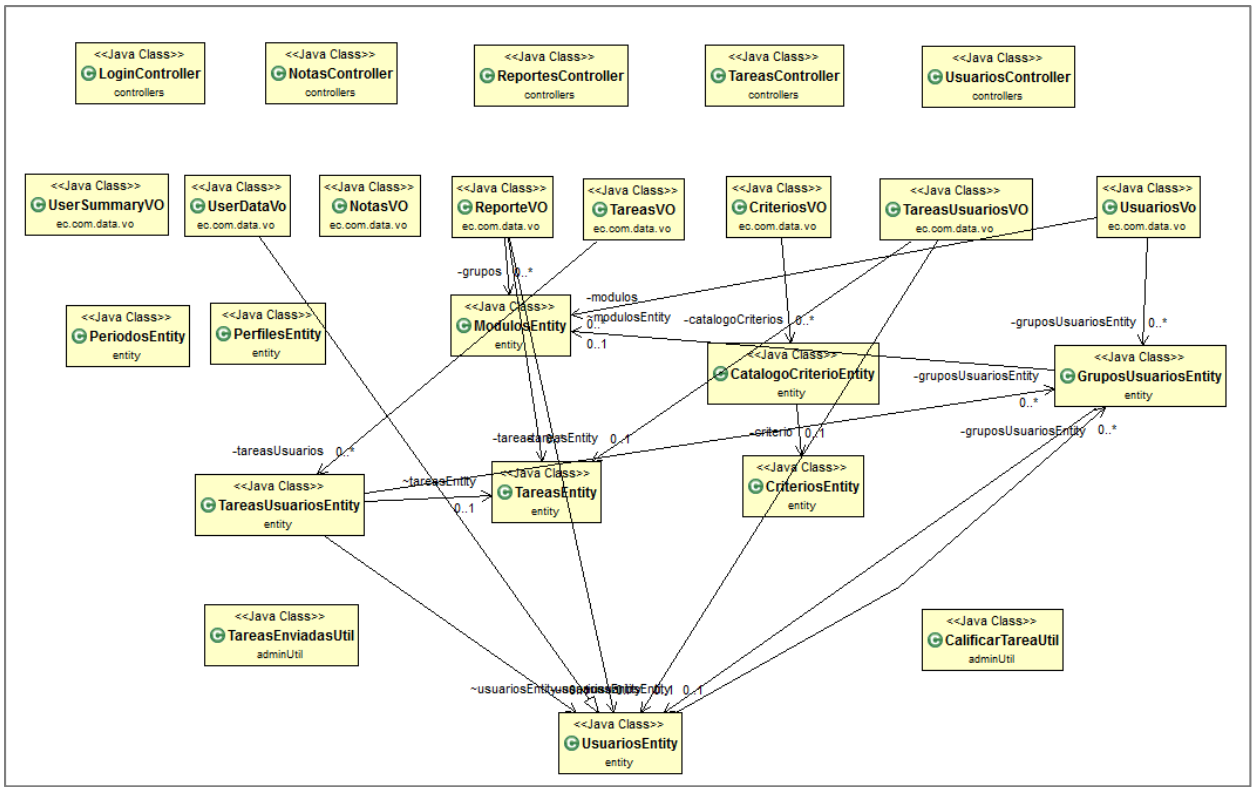



Figura 20. Arquitectura de clases de servicios web del sistema.

Elaborado por: Pablo Guerra.

# Diagramas de clases

## *Controladores*

Diagrama de clases controladores.

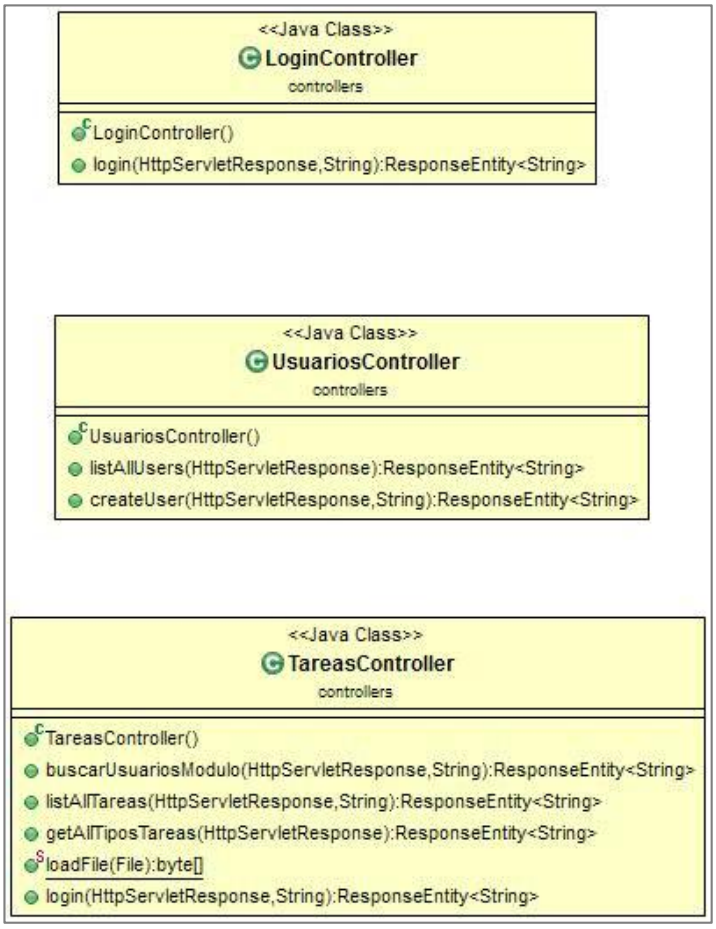



Figura 21. Controladores que publican servicios web.

Elaborado por: Pablo Guerra.

## *Servicios*

Diagrama de clases Servicios.


Figura 22. Clases que se comunican directamente con los servidores.

Elaborado por: Pablo Guerra.

## *Gestores*

Diagrama de clases gestores.

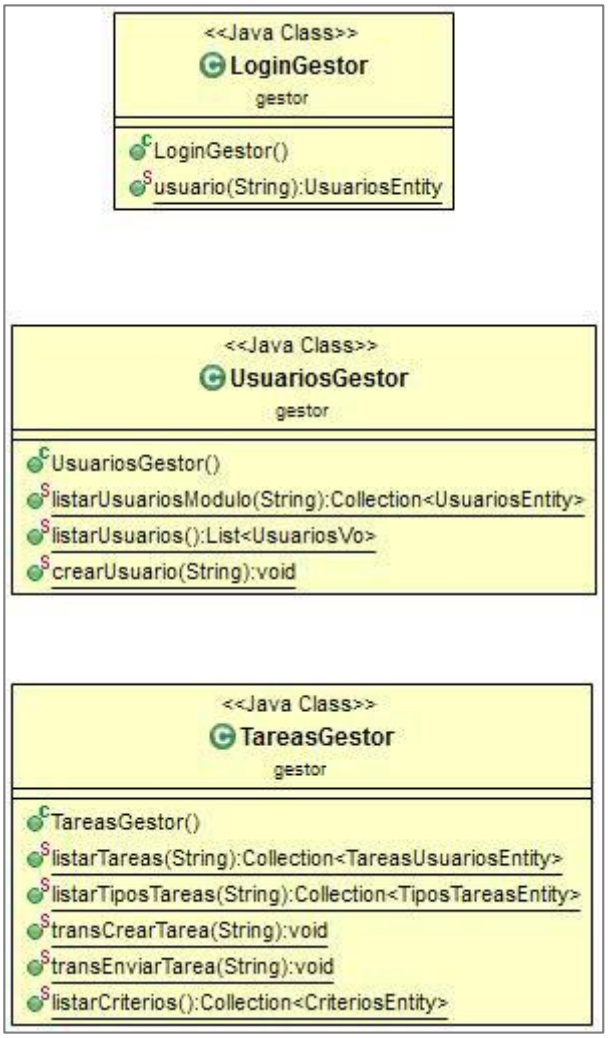



Figura 23. Clases con lógica de procesos.

Elaborado por: Pablo Guerra.

## *Módulos*

Diagrama de clases módulos.

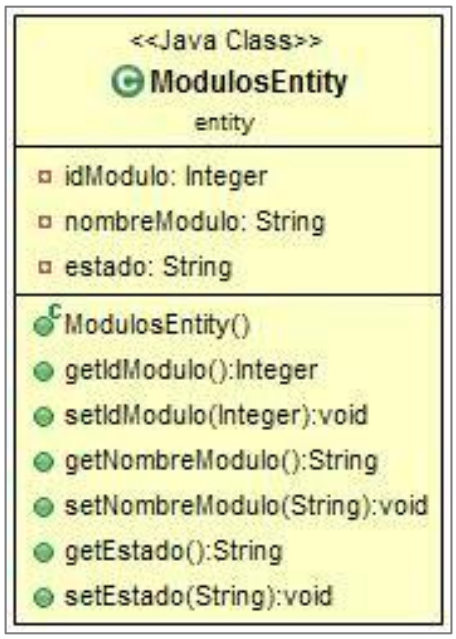



Figura 24. Clase que representa la tabla módulos.

Elaborado por: Pablo Guerra.

## *Criterios*

Diagrama de clases criterios.

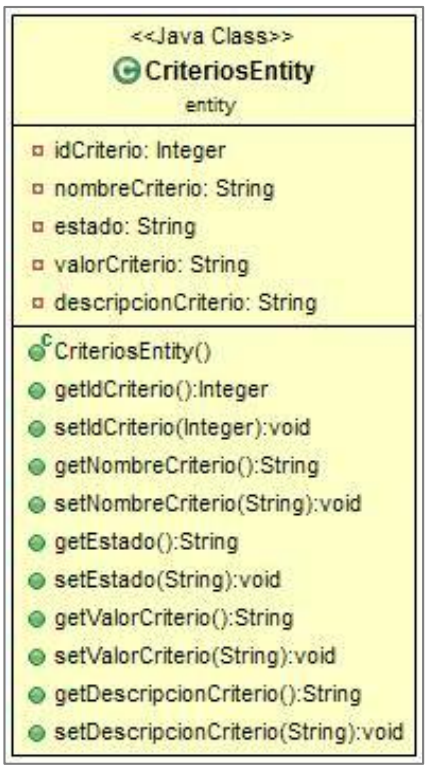



Figura 25. Clase que representa la tabla criterios de la base de datos.

Elaborado por: Pablo Guerra.

## *Tareas*

Diagrama de clases tareas.

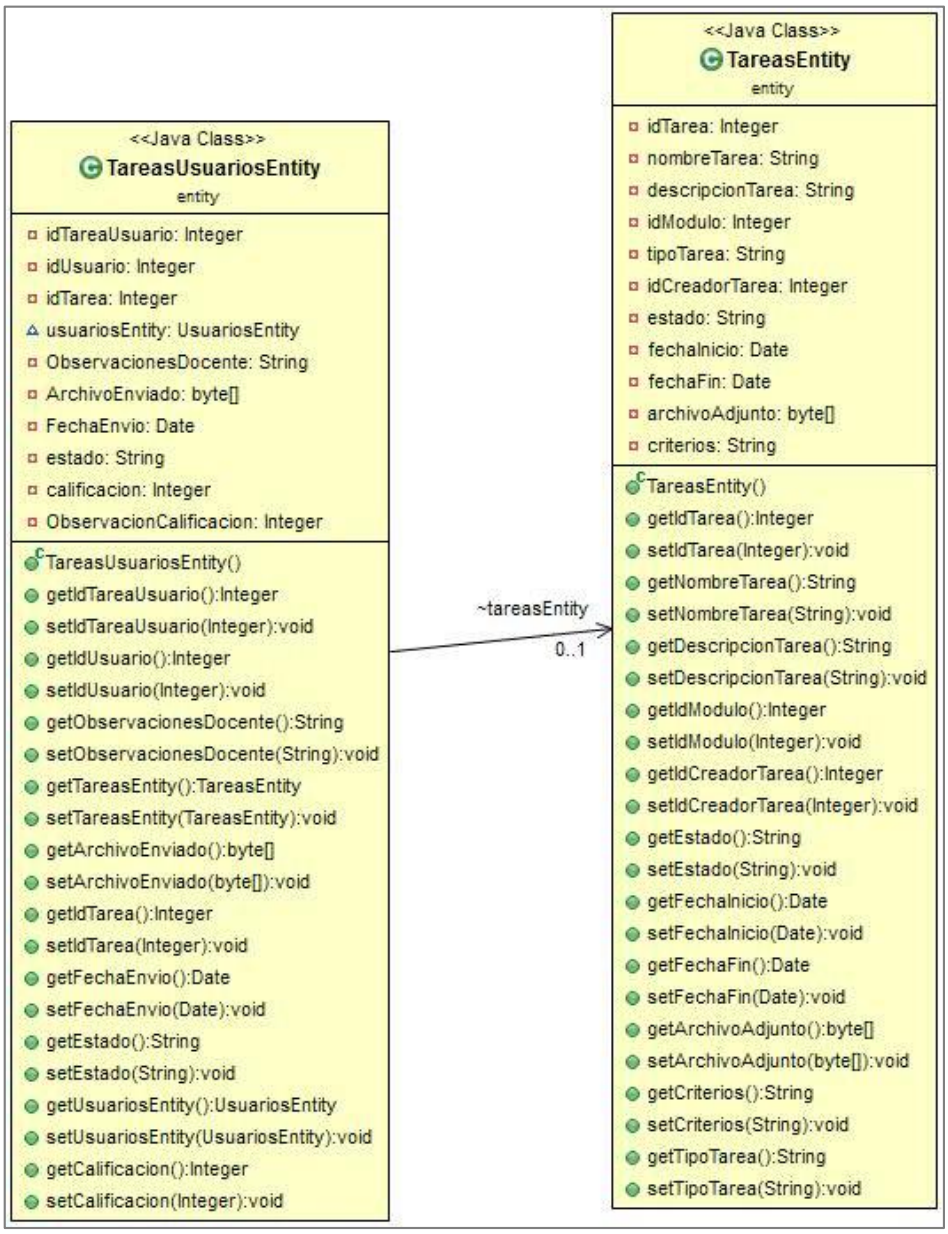



Figura 26. Clases que representan a las tareas y los detalles de tareas.

Elaborado por: Pablo Guerra.

## *Usuarios*

Diagrama de clases usuarios.


Figura 27. Entidad que representa la tabla de usuarios en la base de datos.

Elaborado por: Pablo Guerra.

## *Utilitarios*

Diagrama de clases utilitarios.

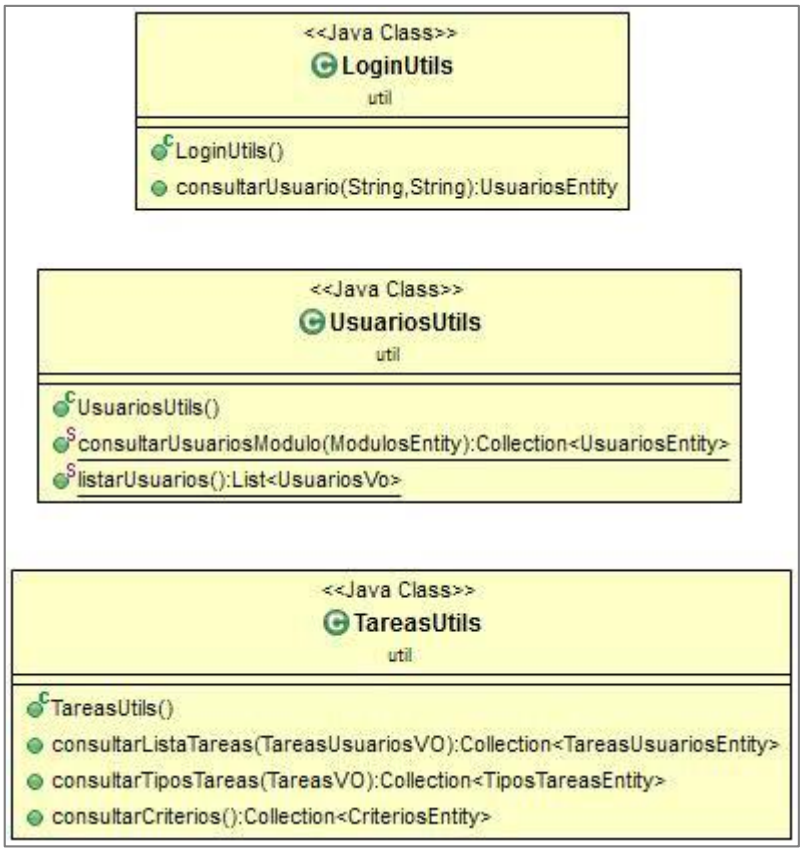



Figura 28. Clases de utilitarios para la gestión de lógica del sistema.

Elaborado por: Pablo Guerra.

## Interfaces de usuario

A continuación se muestran las pantallas de las principales interfaces de usuario del sistema:

### *Login*

El usuario ingresa su nombre de usuario y contraseña para la autenticación.

Interfaz de usuario – login.


Figura 29. Interfaz de usuario para el ingreso al sistema.

Elaborado por: Pablo Guerra.

## *Pantalla de inicio*

Atajos que dirigen a las funciones más utilizadas del sistema.

Pantalla de inicio.

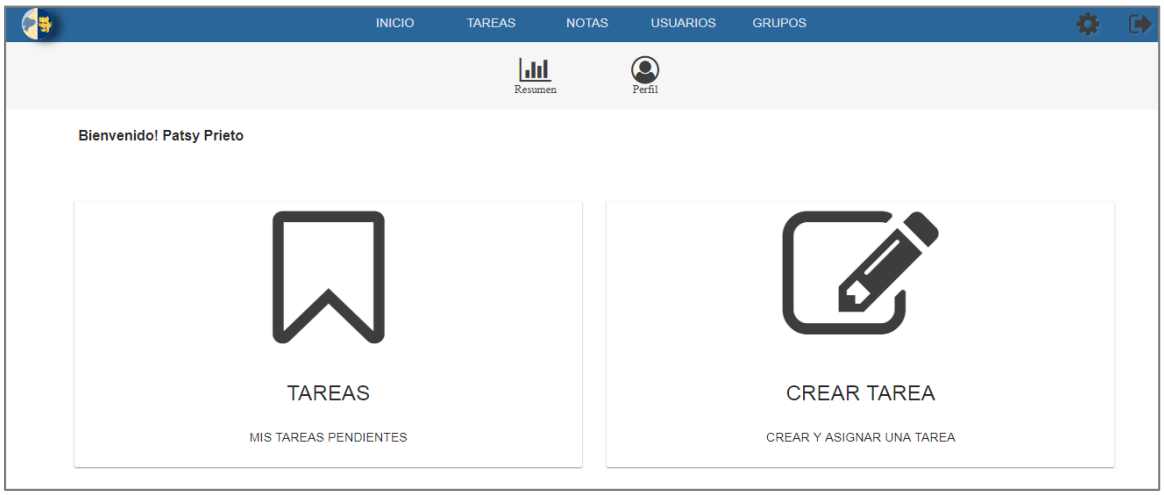



Figura 30. Interfaz de usuario al ingresar al sistema.

Elaborado por: Pablo Guerra.

### *Categorías de tareas*

Pantalla principal de tareas con el número de tareas de cada categoría asignadas al usuario.

Categorías de tareas

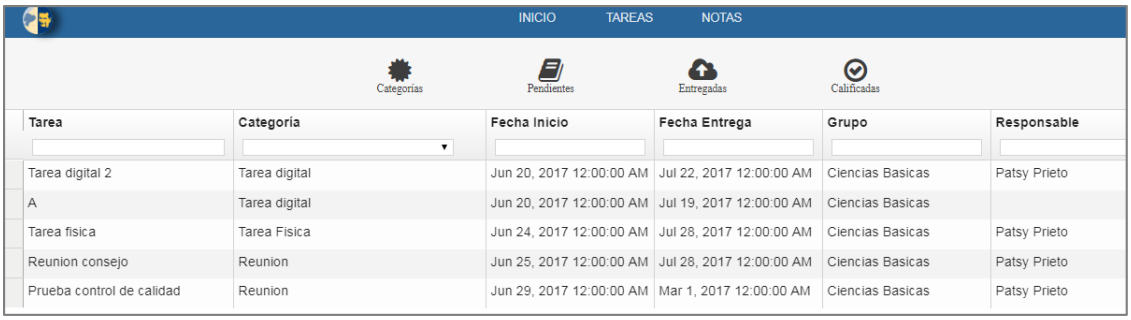

Figura 31. Interfaz de usuario con el resumen de tareas.

Elaborado por: Pablo Guerra.

### *Lista de tareas*

Detalle de las tareas asignadas al usuario, cuando selecciona cualquier tarea pasa a la pantalla de detalle de tarea.

Lista de tareas.




| Tarea | Categoría | Fecha Inicio | Fecha Entrega | Grupo | Responsable |
|---|---|---|---|---|---|
| Tarea digital 2 | Tarea digital | Jun 20, 2017 12:00:00 AM | Jul 22, 2017 12:00:00 AM | Ciencias Basicas | Patsy Prieto |
| A | Tarea digital | Jun 20, 2017 12:00:00 AM | Jul 19, 2017 12:00:00 AM | Ciencias Basicas | |
| Tarea fisica | Tarea Fisica | Jun 24, 2017 12:00:00 AM | Jul 28, 2017 12:00:00 AM | Ciencias Basicas | Patsy Prieto |
| Reunion consejo | Reunion | Jun 25, 2017 12:00:00 AM | Jul 28, 2017 12:00:00 AM | Ciencias Basicas | Patsy Prieto |
| Prueba control de calidad | Reunion | Jun 29, 2017 12:00:00 AM | Mar 1, 2017 12:00:00 AM | Ciencias Basicas | Patsy Prieto |

Figura 32. Interfaz de usuario con la lista de tareas.

Elaborado por: Pablo Guerra.

## *Enviar tareas*

Detalle de la tarea asignada con las opciones de subir el archivo solicitado.

Enviar tarea.


Figura 33. Interfaz de usuario con el detalle de una tarea.

Elaborado por: Pablo Guerra.

### *Calificar tarea*

Pantalla a la que tienen acceso únicamente jefes de área y director de carrera, cuando un docente envía la tarea se selecciona y se obtienen los datos enviados y la opción de ingresar una calificación.

Calificar tarea.

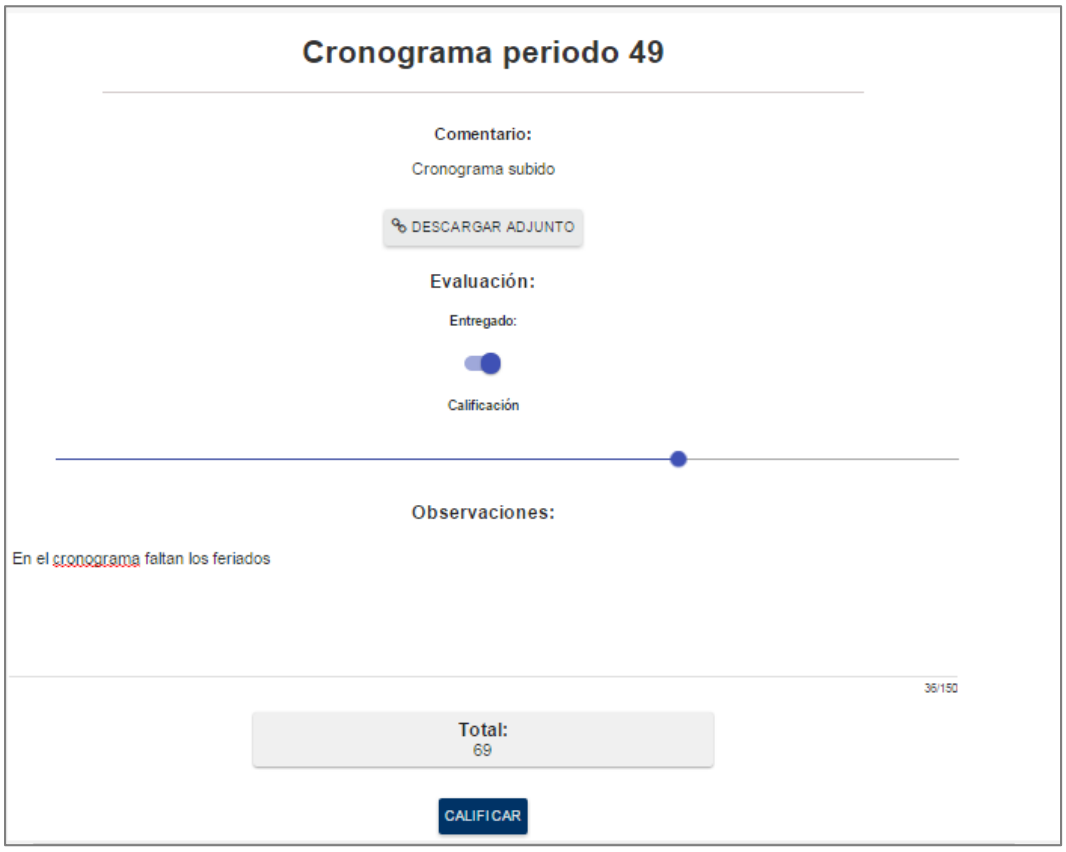



Figura 34. Interfaz de usuario para la evaluación de tarea.

Elaborado por: Pablo Guerra.

## *Crear tarea*

Se ingresan los datos de la tarea y los docentes asignados.

Crear tarea.

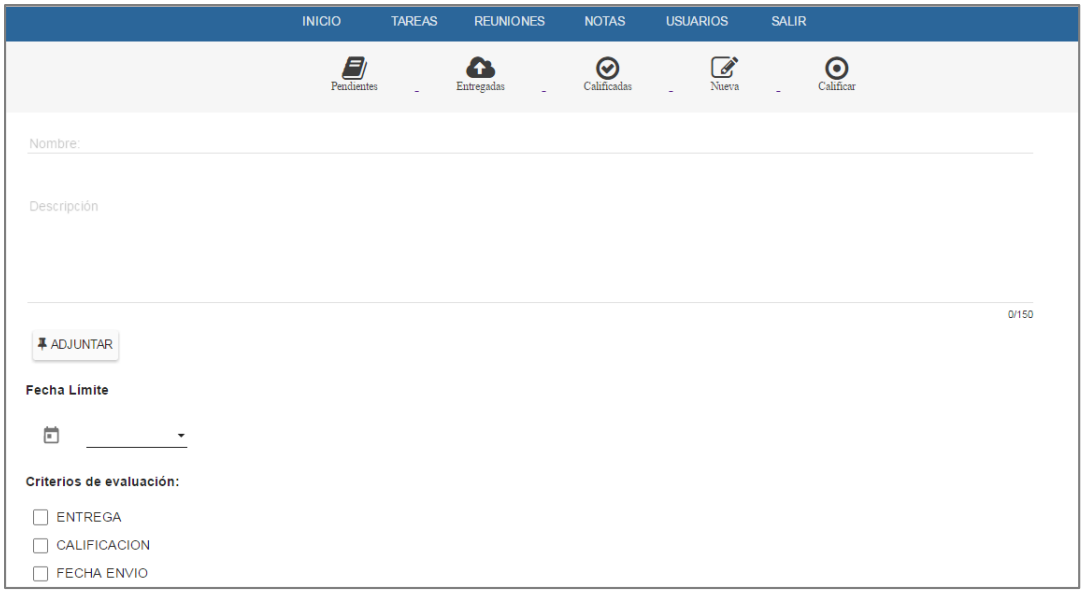



Figura 35. Interfaz de usuario para la creación de tarea.

Elaborado por: Pablo Guerra.

### *Reporte de notas*

Detalle de las tareas asignadas.

Reporte de notas.


Figura 36. Interfaz de usuario con el resumen de calificaciones.

Elaborado por: Pablo Guerra.

## *Reportes de docentes*

Opciones para obtener detalles de notas por docente, grupo, categoría o una tarea específica.

Reporte de docentes.

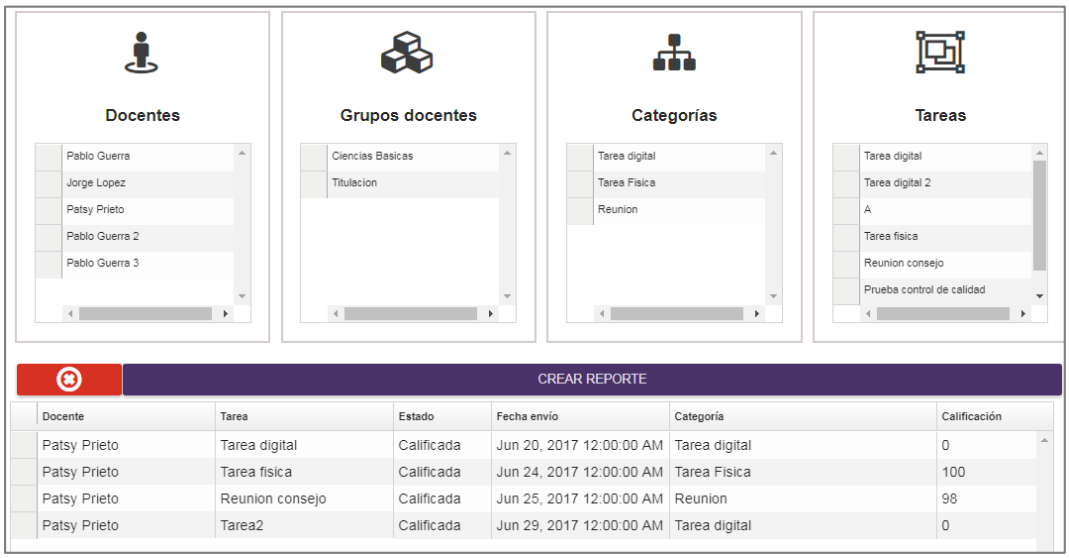



Figura 37. Interfaz de usuario con las calificaciones de los docentes .

Elaborado por: Pablo Guerra.

## Requisitos no funcionales

Se lograron cumplir los objetivos no funcionales con la implementación de los diferentes frameworks propuestos.

### *Flexibilidad*

Si se debe aumentar un módulo al sistema, únicamente se agrega un paquete con archivos de configuración del módulo y ya se pueden crear vistas y controladores que agreguen funcionalidad a la aplicación.

### *Mantenibilidad*

Con el uso de Spring y las herramientas Javascript, si existe algún cambio o corrección al sistema los paquetes y clases involucradas en las funcionalidades llevan el mismo nombre y los cambios se pueden realizar sin ninguna consecuencia en otras partes del sistema.

### *Seguridad*

AngularJS posee características que permiten que elementos de la vista estén restringidos mediante una condición, en el sistema el acceso a los diferentes módulos y características están restringidos a los diferentes perfiles de usuarios y AngularJS garantiza que no accederá a los recursos sin las condiciones requeridas.

# Código relevante

## *Grupos de usuarios*

Debido a que la metodología usada era incremental e iterativa en el desarrollo podían suscitarse cambios de lógica del sistema y uno de los cambios con más impacto fue que los docentes puedan pertenecer a varios grupos (ej. titulación, ciencias básicas) por lo que para este cambio se tuvo que crear una nueva tabla en la base de datos que guarde los diferentes grupos al que pertenece el usuario. En la parte de programación el objeto usuario contiene una colección de objetos gruposUsuario entonces cuando se obtiene la lista de usuarios también se consulta la tabla gruposUsuario y así es posible editar los grupos a los que pertenece un usuario y el funcionamiento de la aplicación no sufrió un gran impacto.

*Actualización de usuarios.*

```
/*** Actualiza Usuario ****/
UsuariosEntity usuarioEntity = new UsuariosEntity();
usuarioEntity.setIdUsuario(usuarioVO.getIdUsuario());
usuarioEntity.setCorreoUsuario(usuarioVO.getCorreoUsuario());
usuarioEntity.setNombreCompleto(usuarioVO.getNombreCompleto());
usuarioEntity.setPassword(usuarioVO.getPassword());
usuarioEntity.setEstado("ACT");
usuarioEntity.setIdPerfil(usuarioVO.getIdPerfil());
session.update(usuarioEntity);
/*** Inactiva Grupos ****/
String hql = "SELECT grupos FROM entity.GruposUsuariosEntity "
        + "grupos where estado = 'ACT' AND grupos.idUsuario = "
        + usuarioEntity.getIdUsuario();
Query query = session.createQuery(hql);
Collection<GruposUsuariosEntity> gruposUsuarios;
gruposUsuarios = query.list();
for(GruposUsuariosEntity gruposUsuario : gruposUsuarios){
    gruposUsuario.setEstado("INA");
    session.update(gruposUsuario);
}
/*** Crea nuevos grupos ****/
for (ModulosEntity grupo : usuarioVO.getGrupos()) {
    GruposUsuariosEntity grupoUsuario = new GruposUsuariosEntity();
    grupoUsuario.setIdUsuario(usuarioEntity.getIdUsuario());
    grupoUsuario.setIdModulo(grupo.getIdModulo());
    grupoUsuario.setEstado("ACT");
    session.save(grupoUsuario);
}
session.getTransaction().commit();
```

Figura 38. Código relevante para la actualización de usuarios.

Elaborado por: Pablo Guerra.

## *Subida de archivos*

Debido a que no se utilizó ningún tipo de componente ni plugin adicional para la subida de archivos, el envío directamente desde el input HTML fue una complicación debido al tipo de datos que se envían y se aceptan en los servicios del servidor. Las cabeceras aceptadas por el servicio web son de tipo texto y los archivos subidos tienen otro formato (byte). Para solucionar este inconveniente se procedió a obtener el objeto del archivo subido y transformarlo a Base64 que es una codificación que puede representar datos binarios en una cadena de texto y así permitir el transporte de archivos a través de cadenas

de caracteres JSON que es el formato más común en la comunicación entre cliente y servidor.

*Código subida de archivos.*

```
//Obtiene el elemento file-upload input
let file = document.getElementById('file-inputDetalle');
if (file.files && file.files != null && file.files != undefined &&
    file.files.length > 0 && file.files['length'] > 0) {
    filename = file.files[0].name.split('.').pop();

    var reader = new FileReader();
    reader.readAsDataURL(file.files[0]);
    //Convierte el archivo a un String Base64
    reader.onload = function () {
        let archivoBase64 = reader.result;
        var arrayBuffer = this.result,

            array = new Uint8Array(arrayBuffer),
            binaryString = btoa(String.fromCharCode.apply(null, array));
        this.archivoAdjuntoDetalle = binaryString;
        tareasModel.tarea.archivoAdjunto = archivoBase64;
```

Figura 39. Código relevante para la subida de archivos en el frontend.

Elaborado por: Pablo Guerra.

En el servidor se debía transformar la cadena de caracteres a un objeto de tipo *byte*. Para esto se utilizó una librería perteneciente a java que transforma la cadena de caracteres codificados en Base64 a un arreglo de tipo byte.

Conversión de archivos.

```
if (tareaVO.getArchivoAdjunto() != null) {
    String base64Image = tareaVO.getArchivoAdjunto().split(",")[1];
    byte[] imageBytes = javax.xml.bind.DatatypeConverter.parseBase64Binary(base64Image);
    tareasEntity.setArchivoAdjunto(imageBytes);
}
```

Figura 40. Código relevante para la conversión de archivos.

Elaborado por: Pablo Guerra.

# Plan de pruebas

## *Objetivo*

Especificar qué partes del sistema van a ser sometidas a pruebas, señalando la razón por la cual estas partes son cruciales para el flujo normal del proceso.

## *Alcance de las pruebas*

Los módulos del sistema que se van a probar incluyen:

- Módulo Tareas.
- Módulo Usuarios.
- Módulo Reportes.

## *Enfoque de las pruebas*

Las pruebas se enfocarán principalmente en el flujo del sistema, debido a que no existirá una afluencia considerable del sistema el enfoque principal no es al rendimiento sino al uso que le darán los docentes y jefes de área.

## *Criterios de aceptación*

- Existe una fluidez normal en el uso del sistema.
- No existen demoras considerables en las llamadas a servicios del sistema.
- El usuario sabe qué hacer para completar una tarea.

## *Entregables*

Para la realización de pruebas se utilizarán cuadros con casos de pruebas que contienen métricas, resultados y criterios de las pruebas a partir de un promedio de los usuarios que realizaron las pruebas.

## *Recursos*

### Herramientas de pruebas

Para las pruebas de rendimiento se utilizara la herramienta JMeter que sirve para probar el comportamiento funcional y medir el rendimiento de aplicaciones web con servicios Java.

### Personal

El personal necesario para la realización de las pruebas fue:

- Administrador del sistema para pruebas de rendimiento.
- Un equipo con conocimientos de sistemas y experiencia en control de calidad de software.

# Ejecución de pruebas

## *Pruebas de usabilidad*

### Módulo Tareas

Tabla 16. Pruebas de usabilidad módulo tareas.

| Tarea | Valor a medir | Valor actual | Peor valor posible | Valor objetivo | Mejor valor posible |
|---|---|---|---|---|---|
| Creación de tarea | Tiempo para creación de tarea exitosa | 3 minutos | 6 minutos | 2 minuto | 1 minuto |
| Envío de tarea | Envío de tarea exitoso | 2 minutos | 5 minutos | 1 minuto | 1 minuto |
| Calificación de tarea | Número de errores | 1 error | 3 errores | 0 errores | 0 errores |

**Módulo Usuarios**

Tabla 17. Pruebas de usabilidad módulo usuarios.

| Tarea | Valor a medir | Valor actual | Peor valor posible | Valor objetivo | Mejor valor posible |
|---|---|---|---|---|---|
| Creación de usuario | Número de errores | 1 error | 3 errores | 1 error | 0 errores |
| Edición de usuario | Tiempo del primer intento | 1 minuto | 2 minutos | 30 segundos | 1 minuto |

**Módulo Reportes**

Tabla 18. Pruebas de usabilidad módulo reportes.

| Tarea | Valor a medir | Valor actual | Peor valor posible | Valor objetivo | Mejor valor posible |
|---|---|---|---|---|---|
| Ingreso al módulo reportes | Tiempo para ingreso al módulo de reportes desde la pantalla de inicio | 30 segundos | 120 segundos | 10 segundos | 20 segundos |

| | | | | | |
|---|---|---|---|---|---|
| Creación de reporte sin filtros | Tasa de completación | 2 intentos | 5 intentos | 1 intento | 1 intento |
| Creación de reporte con filtros | Número de errores | 1 error | 3 errores | 0 errores | 0 errores |

**Conclusión pruebas usabilidad**

Luego de una introducción y capacitación al sistema los usuarios de prueba tuvieron pocos problemas con el uso del sistema, algunos tuvieron problemas con los datos necesarios para la completación de ciertas tareas por lo que se encontraron validaciones faltantes y fueron corregidas después de las pruebas.

## *Pruebas funcionales*

En las pruebas funcionales se seleccionaron casos de uso críticos y se realizó un flujo del proceso principal del sistema que es la gestión de tareas con varios usuarios con los diferentes perfiles que existen en el sistema (docente, jefe de área y director de carrera) y los resultados fueron:

Tabla 19. Prueba funcional creación de tarea.

| **Caso de uso** | | Creación de tarea | | |
|---|---|---|---|---|
| Perfil | **No ejecutado** | **Pasado** | **Fallado** | **Bloqueado** |
| **Docente** | 1 | 0 | 0 | 0 |
| **Jefe de área** | 0 | 1 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| **Director de carrera** | 0 | 1 | 0 | 0 |
| **Total** | 1 | 2 | 0 | 0 |

Tabla 20. Prueba funcional envío de tarea.

| **Caso de uso** | | Envío de tarea | | |
|---|---|---|---|---|
| Perfil | **No ejecutado** | **Pasado** | **Fallado** | **Bloqueado** |
| **Docente** | 0 | 1 | 1 | 0 |
| **Jefe de área** | 1 | 0 | 0 | 0 |
| **Director de carrera** | 1 | 0 | 0 | 0 |
| **Total** | 2 | 1 | 1 | 0 |

Tabla 21. Prueba funcional calificación de tarea.

| **Caso de uso** | | Calificación de tarea | | |
|---|---|---|---|---|
| Perfil | **No ejecutado** | **Pasado** | **Fallado** | **Bloqueado** |
| **Docente** | 1 | 0 | 0 | 0 |
| **Jefe de área** | 0 | 1 | 0 | 0 |
| **Director de carrera** | 0 | 1 | 0 | 0 |
| **Total** | 1 | 2 | 0 | 0 |

Tabla 22. Prueba funcional creación reportes.

| **Caso de uso** | | Creación de reportes | | |
|---|---|---|---|---|
| Perfil | **No ejecutado** | **Pasado** | **Fallado** | **Bloqueado** |
| **Docente** | 1 | 0 | 0 | 0 |
| **Jefe de área** | 1 | 0 | 0 | 0 |
| **Director de carrera** | 0 | 1 | 0 | 0 |
| **Total** | 1 | 1 | 0 | 0 |

Luego de varias pruebas de funcionalidad se encontró una validación faltante en el envío de tareas que hacía que el sistema falle, la solución fue modificar la subida de archivos para cuando se trate de una tarea que no necesita archivo no intente enviar el parámetro *archivoAdjunto* en el objeto que se envía al servidor.

Luego de las correcciones se realizaron nuevamente las pruebas de la funcionalidad para el caso y se obtuvieron los siguientes resultados:

Tabla 23. Prueba funcional envío de tarea (corrección).

| **Caso de uso** | | Envío de tarea | | |
|---|---|---|---|---|
| Perfil | **No ejecutado** | **Pasado** | **Fallado** | **Bloqueado** |
| **Docente** | 0 | 1 | 0 | 0 |
| **Jefe de área** | 1 | 0 | 0 | 0 |
| **Director de carrera** | 1 | 0 | 0 | 0 |
| **Total** | 2 | 1 | 0 | 0 |

## *Pruebas de rendimiento*

Para las pruebas de rendimiento se utilizó el escenario de la consulta de tareas que es una de las que más carga lleva al servidor.

**Tiempo de respuesta**

**1 Usuario**

Pruebas de rendimiento consulta de tareas 1 usuario.

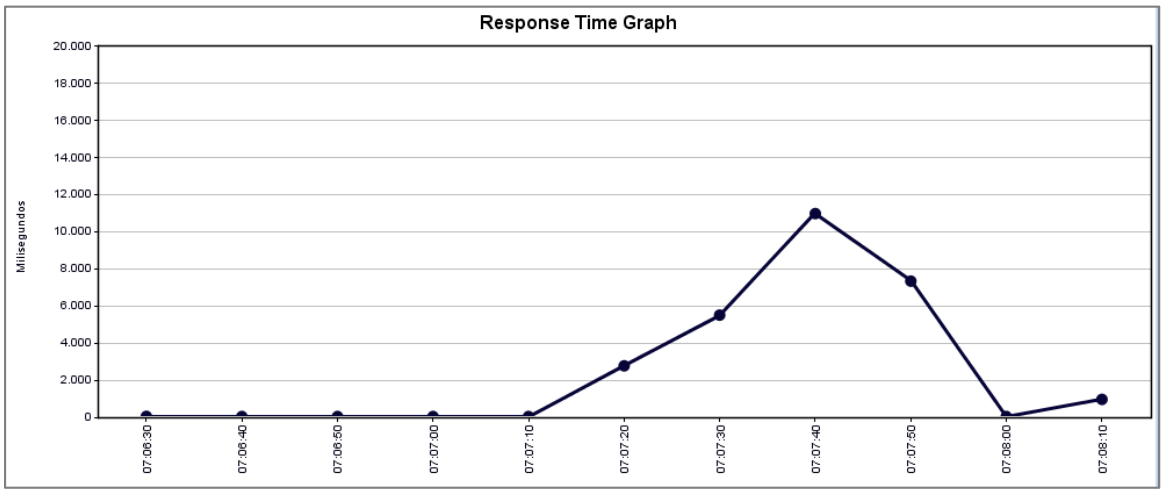



Figura 41. Rendimiento de la aplicación con 1 usuario.

Elaborado por: Pablo Guerra.

Con 1 usuario el tiempo de respuesta es de 12ms por lo que el rendimiento para 1 usuario es aceptable.

### 5 Usuarios

Pruebas de rendimiento con 5 usuarios.

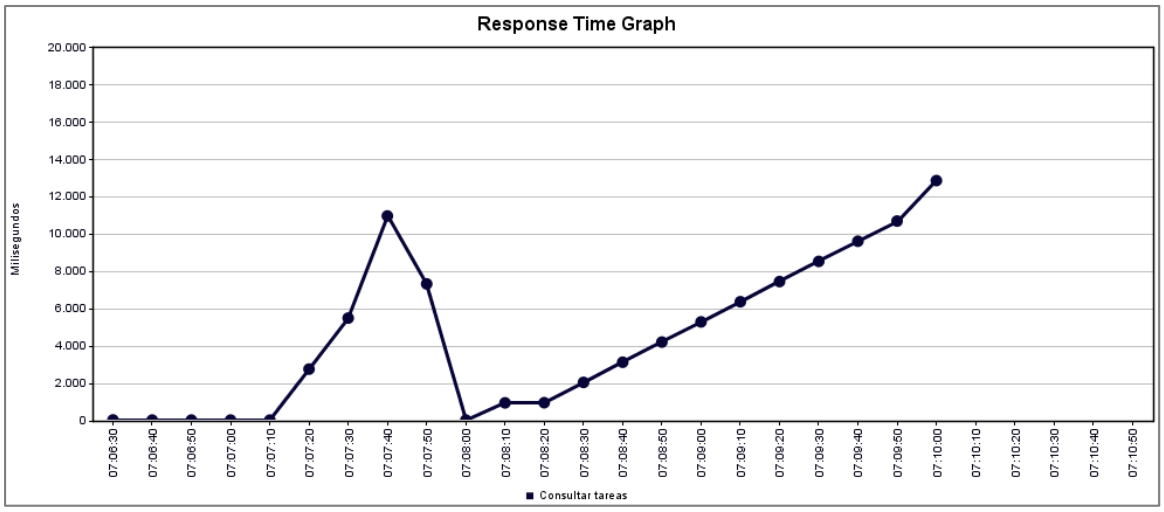



Figura 42. Rendimiento de la aplicación con 5 usuarios.

Elaborado por: Pablo Guerra.

Con 5 usuarios al mismo tiempo la repuesta sube a aproximadamente 14ms lo cual es aceptable para una aplicación java.

### 100 Usuarios

Pruebas de rendimiento con 100 usuarios - resultado.

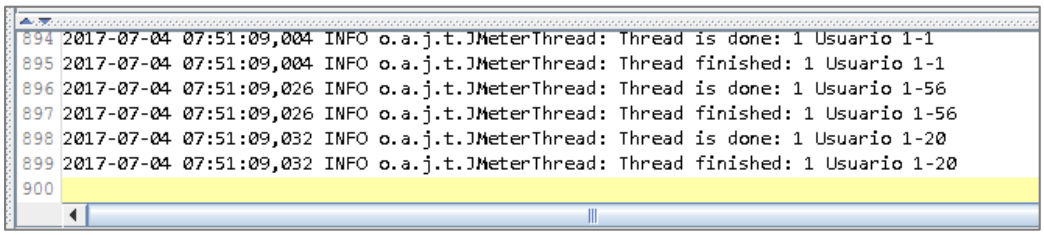



Figura 43. Resultado del servidor de la aplicación con 100 usuarios.

Elaborado por: Pablo Guerra.

Pruebas de rendimiento con 100 usuarios - servidor.

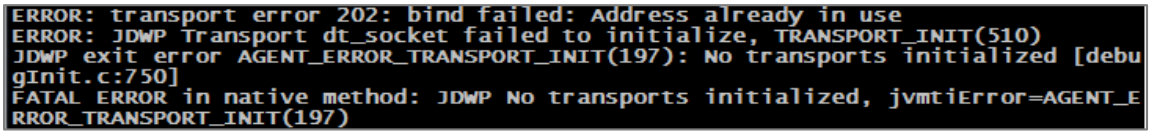



Figura 44. Log del servidor con 100 usuarios.

Elaborado por: Pablo Guerra.

Con 100 usuarios la máquina virtual de Java alcanzó su capacidad máxima de memoria y dejó de responder.

# Capítulo 4

# Conclusiones y Recomendaciones

- Se desarrolló un sistema que permite la creación de tareas, administración de usuarios, configuración de parámetros y generación de reportes utilizando tecnologías que actualmente están a la vanguardia en el campo de la tecnología de información.
- Combinando varias tecnologías para el desarrollo de software se pueden optimizar los recursos y el tiempo utilizado en tareas paralelas al desarrollo puede ser disminuido.
- El sistema va a reemplazar al proceso actual de seguimiento del cumplimiento de tareas que no es un proceso controlado por un sistema de software por lo que no se va a cambiar un software por otro sino se va a implementar uno nuevo.
- Desde un punto de vista analítico y funcional el sistema va a mejorar el proceso de evaluación de los docentes de la carrera y cuando esté en funcionamiento se van a optimizar los procesos, se van a emitir mejores juicios y se podrá observar un mejor panorama sobre el rendimiento de los docentes en cuanto a las responsabilidades extracurriculares.

- En proyectos como este que utilizan tecnologías que no han lanzado su versión final se recomienda, para una mejor estabilidad de los componentes y para la corrección de errores de los mismos analizar la posibilidad de actualizar las versiones si se concluye que no afectarán las funcionalidades del sistema. Esto excluye al framework AngularJS ya que la versión utilizada (v1) es incompatible con la nueva versión (Angular 2) y actualizar implicaría cambiar la sintaxis de todas las clases y vistas del sistema.
- Con el tiempo y el uso del software en un ambiente de producción devolviendo resultados reales se deberá considerar si los parámetros establecidos inicialmente están logrando los resultados esperados, ya que sólo con un uso considerable del sistema se reflejará el cumplimiento de los objetivos. Si se debiere realizar cambios en los parámetros la sección de configuración del sistema (accesible únicamente por la directora de carrera) permite realizar cambios en el sistema sin cambiar líneas de código.

# Lista de referencias


Kenneth, S. Rubin, (2012). Essential SCRUM: A practical guide to the most popular agile process

Jamae Javid, Johnson Peter (2009). JBoss in Action: Installation, configuration and deployment

Fisher Mark, Partner Jonas, Bogoevici Marius, Fuld Iwein (2012) Spring integration in action

Teixeira Pedro (2013) Professional Node.js: Building Javascript based scalable software

Rauch Guillermo (2012) Smashing Node.js: Javascript Everywhere

Bauer Christian, King Gavin () Hibernate in action: A guide to the concepts and practice of object/relational mapping

Kotaru V. Keerti (2016) Material Design implementation with AngularJS

Bell Charles (2016) MySQL for the internet of things: Data management for sensors and connected devices

Seddighi Reza Ahmad (2009) Spring Persistence with Hibernate

Naik Kshirasagar, Tripathy Priyadarshi (2008) Software testing and quality assurance: Theory and practice.

McCullough Matthew, Loeliger Jon (2012) Version control with Git: Powerful tools and techniques for collaborative software development.

James Rumbaugh, Ivar Jacobson, Grady Booch (2005) The Unified Modeling Language Reference Manual

Eckel Bruce (2006) Thinking in Java: The definitive introduction to objetct oriented programming in the language of the world wide web.

Saavedra R. Manuel S. (2001) Evaluación del aprendizaje

Vélez Cecilia (2003) Manual de la evaluación de desempeño

Stufflebeam (1971) The use of experimental design in educational evaluation

Arana, J. & Parreño, T. (2016). *CREACIÓN DE UN FRONT END USANDO HTML5, PARA EXPLOTAR LAS FUNCIONALIDADES DE LA PLATAFORMA MOODLE, ADEMÁS DE UNA APLICACIÓN MÓVIL ANDROID PARA EL MANEJO OPORTUNO DE NOTIFICACIONES CON SINCRONIZACIÓN A GOOGLE CALENDAR* (Ingeniería). Universidad Politécnica Salesiana.

Apple Inc. (2015). Model-View-Controller. 2017, de Apple Inc. Sitio web: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Google. (2010). What is Angular. 2017, de Google Sitio web: https://docs.angularjs.org/guide/introduction

Babel. (2017). Get started. 2017, de BabelJS Sitio web: https://babeljs.io/

Gulp. (2017). GulpJS. 2017, de GulpJS Sitio web: http://gulpjs.com/