Estudio, diseño e implementación de una web de corrección automática de prácticas y trabajos la asignatura Compiladores

Christian García Miñano

Resumen — El proyecto emerge de la necesidad de crear un sistema para que los alumnos que cursan la asignatura de Compiladores de la Universidad Autónoma de Barcelona puedan corregir de forma automática sus prácticas y trabajos. Además de ayudar al profesor a gestionar esta labor que se hace muy compleja al ojo humano puesto que los factores a analizar son variados y complejos de detectar. Actualmente en la asignatura el problema mencionado ya está resuelto mediante unos Scripts en lenguaje Cosel, pero la gestión de los mismos con el tiempo se está volviendo demasiado pesada de administrar y además necesita del trabajo del profesor para funcionar. Todo ese trabajo se puede automatizar y al mismo tiempo funcionar de manera más eficiente.

Palabras clave—Cosel, Corrección automática, web de corrección automática, JSP, JEE, Compiladores, Corrección automática de prácticas en compiladores, Spring, ORM, Hibernate.

Abstract— The project stems from the need to create a system for students that are taking the Compilers subject at the Autonomous University of Barcelona (UAB) so they can automatically correct their practices and assignments. Furthermore, it aims at helping the teachers to manage this task, because, sometimes it can turn out to be very tricky due to the factors that the human eye has to analyze are often changing and hard to detect. Currently, the problem described is being handled by Cosel language scripts, but the amount of data to manage is getting too big for those scripts to do it and they also require of some work from the teacher. Work that, has been considered, can be optimized and automated.

Index Terms— Cosel, automatically correct, web of automatically correct, JSP, JEE, Compilers, Automatically correct in Compilers, Spring, ORM, Hibernate.

1 Introducción

La motivación del trabajo, emerge del deseo del profesor Francisco Javier Sánchez Pujadas de poder ofrecer a los alumnos que cursan asignaturas en las que él ejerce docencia una aplicación para poder corregir de forma automática e instantánea las diferentes prácticas y ejercicios que deben entregar sus alumnos para poder aprobar la asignatura. Asimismo también libera al profesor de una tarea que es muy complicada al ojo humano puesto que la complejidad de detectar errores en este tipo de trabajos es bastante elevada debido al gran número de variables que se analizan.

Para poder entender mejor la complejidad del problema el presente artículo se ha desglosado en los siguientes apartados:

- Introducción: Explica la motivación del trabajo y los diferentes puntos del artículo.
- Objetivos: Definición de los diferentes objetivos y subobjetivos para poder llevar a cabo la primera versión de la aplicación.
- Estado del Arte: Análisis de la situación actual y las herramientas existentes para poder solucionar el problema.
- Metodología: Se realiza una explicación acerca de la metodología utilizada durante la elaboración del proyecto así como la técnica de programación utilizada para realizar la codificación.
- Requisitos: Explicación de los diferentes requisitos funcionales y no funcionales extraídos a través de los objetivos y subobjetivos.
- Diseño: Explicación y análisis de los diferentes diseños (clases, base de datos e interfaces)
- Test: En este apartado se realizará un análisis de tipo de test utilizado para asegurar un nivel aceptable de calidad en la aplicación.
- Resultados: Exposición de los resultados obtenidos de la aplicación.
- Líneas de trabajo futuras: Análisis de las líneas de mejora para posteriores versiones de la aplicación.
- Conclusiones: Exposición de las conclusiones extraídas del trabajo realizado.
- Agradecimientos: Se realizará una mención a to-

das las personas que han hecho posible la realización del proyecto.
- Bibliografía: En este último apartado se detalla la bibliografía utilizada para la realización del proyecto.

2 ESTADO DEL ARTE

Actualmente el problema de la corrección automática de prácticas está solucionado mediante la utilización de scripts en lenguaje Cosel especializados en cada entrega. La gestión de estos scripts con el tiempo se vuelve tediosa puesto que se trata de una gestión manual por parte del profesor, tanto para gestionar los scripts como las entregas se requiere la ejecución y supervisión de un profesional porque pueden generar una parada en el sistema como la que se muestra en la figura 1:

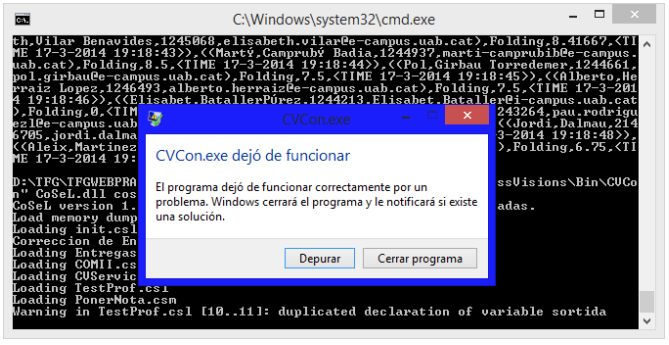


Figura 1: Bloqueo del sistema causado por una práctica.

Por otra parte, la Universidad Autónoma de Barcelona (UAB) posé diferentes portales para la gestión de las asignaturas como puede ser el campus virtual, Cerbero, Neptu,etcétera...

Pero la Universidad Autónoma de Barcelona no posé la conexión de ninguna de sus herramientas con los scripts de corrección destacando que el desarrollo de esta nueva funcionalidad no es competencia de la Universidad. De manera que se optó por el desarrollo de una nueva aplicación exclusiva para la ejecución de los scripts desde cero en el lenguaje de programación Java [1] ya que es un lenguaje compilado y otorga más velocidad frente a otros lenguajes interpretados como puede ser PHP [2].

Por último, se descartó la utilización de gestores de contenidos como pueden ser moodle [3] o Joomla [4] ya que aportan demasiadas funciones que no son necesarias para la elaboración del proyecto y que además estarían repetidas con las otras herramientas de la UAB.

3 OBJETIVOS

3.1 Objetivos de la aplicación

Para un correcto desarrollo de la aplicación se han marcado los siguientes objetivos principales con sus subobjetivos:

1. Conectividad con las diferentes plataformas de la Universidad Autónoma de Barcelona:
   - Comunicación con cerbero.
   - Comunicación con el servidor de autentificación de la UAB.
   - Comunicación en un tiempo inferior a 3 segundos desde red local.
2. Seguridad en el aplicativo:
   - Garantizar que la información de los alumnos, solo podrá ser accesible por parte del profesor de la asignatura.
   - Cifrar en formato Bcrypt toda la información sensible almacenada.
   - Acceso al aplicativo mediante autorización previa.
   - Acceso al aplicativo mediante autentificación previa en un tiempo inferior a 2 segundos.
   - Asegurar que la información de los alumnos, tanto personal como académica será guardada en almacenamiento persistente y duplicada un mínimo de dos veces.
   - El aplicativo guardará en la base de datos todos los registros acerca de su uso.
3. El aplicativo tiene que estar disponible el 99 % del tiempo los 365 días del año:
   - Separar el aplicativo de las distintas plataformas.
   - Duplicación de los distintos servidores.
4. El aplicativo tiene que poder ejecutar scripts y retornar resultados:
   - Ejecutar scripts en Cosel y retornar algún resultado en un tiempo inferior a 10 segundos.
   - Retornar resultados en formato TXT mediante la aplicación.
   - Detener la ejecución del script si este supera los diez segundos de ejecución.
   - Asignar una calificación en función del número de test superados.
   - Permitir entregar la misma práctica y aplicar una penalización, está será definida por parte del profesor.
5. El aplicativo tiene que disponer de una web propia para el profesor con tal de que este pueda subir nuevos scripts para la corrección de prácticas.
6. Los profesores deben poder registrase en el aplicativo.
7. Los alumnos deben poder registrarse en el aplicativo.
   - El profesor subirá un archivo en formato TXT que introducirá los alumnos que tienen acceso a la aplicación y a la asignatura a la que pertenecen.
   - Los alumnos podrán solicitar su contraseña una vez sean dados de alta por el profesor y el sistema les facilitará una contraseña de forma automática.
8. El profesor debe de poder acceder de forma gráfica y en cualquier lugar los resultados de las diferentes prácticas así como la creación de nuevas prácticas.

9. El alumno debe de poder acceder al portal para subir sus prácticas, ver calificaciones e informes.
10. El aplicativo debe adaptarse a cualquier tamaño de pantalla.
    - Adaptar el aplicativo al tamaño de pantalla utilizado en PC.
    - Adaptar el aplicativo al tamaño de pantalla utilizado en *Tablet*.
    - Adaptar el aplicativo al tamaño de pantalla utilizado en móvil.

3.2 Categorización de objetivos

La categorización de los objetivos se ha realizado en función de los criterios de prioridad y criticidad, obteniendo un resultado en función de su impacto en la implementación:

Objetivo	Prioridad	Criticidad	Resultado
1	Baja	Baja	Secundario
2	Alta	Alta	Principal
3	Normal	Alta	Secundario
4	Alta	Alta	Principal
5	Normal	Normal	Secundario
6	Alta	Alta	Principal
7	Normal	Alta	Principal
8	Alta	Alta	Principal
9	Normal	Alta	Principal
10	Baja	Baja	Secundario

Figura 2: Categorización de los objetivos.

4 METODOLOGÍA

Para el desarrollo del proyecto se ha utilizado una versión adaptada a la metodología de desarrollo ágil *Scrum*. Esta metodología se define como un método iterativo e incremental. En la apertura del proyecto se definen los requerimientos que se deberán satisfacer para construir el sistema. Los requerimientos son definidos mediante reuniones de planificación con el *stakeholder*. A partir de ese momento se estipulan las iteraciones, conocidas como *Sprint*, en las que la aplicación irá evolucionando. La duración recomendada del *Sprint* es de un mes.

La adaptación a esta metodología ha consistido en que el mismo sujeto realiza los roles de desarrollador y Scrum master y la figura del Product Owner se representa por el tutor. Los Sprints han tenido una duración aproximada de un mes, al principio y al final de cada sprint se han programado reuniones con el tutor con el objetivo de revisar el trabajo realizado, adaptar modificaciones y planificar las acciones a seguir, así como exponer los obstáculos encontrados, siempre intentando seguir la planificación inicial, pero introduciendo los cambios necesarios para que el desarrollo del proyecto se ajustase al máximo a las necesidades reales del cliente.

Este tipo de metodología proporciona la ventaja de maximizar la retroalimentación entre el cliente y el desarrollador logrando corregir problemas, mitigar riesgos tempranamente.

Cabe enfatizar que Scrum es un marco de trabajo de gestión de proyectos, no una metodología de desarrollo de software, de manera que no define prácticas concretas de desarrollo o pruebas de software. De manera que se ha combinado la sistemática de Scrum con el marco de desarrollo de software conducido por pruebas (TDD) "Test Driven Development". Es una técnica que permite cambiar el orden marcado en cuanto a primero desarrollar y luego realizar pruebas. De manera que a medida que añadimos nuevas funcionalidades, se desarrolla el software necesario para aprobar el caso de prueba, hasta que todo tenga éxito. Esta técnica permite asegurar un nivel de calidad en la codificación y una menor tasa de fallos potenciales.

5 REQUISITOS

Los requisitos del sistema han sido obtenidos a partir de los diferentes objetivos que debe cumplir la aplicación en su primera versión, para una mayor compresión este apartado se ha dividido en dos subapartados, los requisitos funcionales y no funcionales:

5.1 Requisitos funcionales

RF. 1 El alumno debe de poder solicitar su contraseña una vez ha sido dado de alta en la base de datos por el profesor.
RF. 2 El alumno debe de poder acceder al portal una vez ha recibido su contraseña.
RF. 3 El alumno debe de poder visualizar cuales son las prácticas pendientes de entrega.
RF. 4 El alumno debe de poder subir las entregas.
RF. 5 El alumno tiene que poder ver sus calificaciones.
RF. 6 Un profesor debe de poder registrar otro profesor en la aplicación.
RF. 7 El profesor debe de poder iniciar sesión en el aplicativo.
RF. 8 El profesor debe de poder crear asignaturas.
RF. 9 El profesor debe de poder crear entregas que asocia a las asignaturas.
RF. 10 El profesor debe de poder subir nuevos scripts.
RF. 11 El profesor debe de poder subir un fichero con los nuevos alumnos que tienen permiso al aplicativo.

5.2 Requisitos no funcionales

RNF. 1 La aplicación debe de poder cifrar las contraseñas.
RNF. 2 La aplicación debe de garantizar la autentificación previa para acceder a ella.
RNF. 3 La aplicación no ha de permitir que un usuario alumno acceda a las áreas privadas del profesor.
RNF. 4 La aplicación debe de estar disponible el 99 % del tiempo en un año natural.
RNF. 5 La aplicación debe de funcionar en sistemas Windows.
RNF. 6 La aplicación debe de poder ejecutar scripts en Cosel.

RNF. 7 La aplicación debe de poder enviar correos electrónicos.
RNF. 8 La aplicación debe de poder descomprimir archivos.
RNF. 9 La aplicación debe de poder adaptarse al tamaño de la pantalla.
RNF. 10 La aplicación debe dar respuesta en menos de 10 segundos para todos sus apartados.
RNF. 11 La aplicación debe de poder comunicarse con otras herramientas de la Universidad Autónoma de Barcelona.
RNF. 12 La aplicación debe de almacenar en almacenamiento estable toda la información del sistema.
RNF. 13 El sistema deberá estar preparado para soportar la carga de datos que supondrá pasar de trabajar con datos para las pruebas en desarrollo a datos reales de los usuarios.

6 DISEÑO

El diseño general de la aplicación se basa en el paradigma de programación modelo-vista-controlador, se optó por utilizar este paradigma para poder desacoplar el apartado de almacenamiento persistente, la gestión de las diferentes peticiones y las vistas. Además de poder dotar al entorno con las diferentes herramientas para poder cumplir con los requerimientos detallados en el apartado de requisitos. Para realizar una implementación más eficiente del paradigma se optó por utilizar un framework, en este caso Spring [5] ya que posee una gran comunidad de usuarios que lo utilizan y además es utilizado por grandes compañías para sus proyectos de software.

Inicialmente durante los primeros pasos del proyecto se barajó la posibilidad de no implantar una web desde cero como se explica en el apartado de estado del arte, pero si se ha dejado un diseño acorde para poder implementar esta funcionalidad a través de llamadas Soap [6].

En el ecosistema presentado en la figura 3 cada servidor realiza una de las tareas que reflejan el modelo-vista-controlador. El servidor apache [7] sería el encargado de almacenar los elementos visuales de la aplicación como pueden ser las vistas en HTML5 [8], el servidor de aplicaciones Tomcat [9] a través de un dispatcher servelet es el que gestiona las peticiones entrantes del servidor apache, no así con cerbero ya que este necesitaría realizar llamadas soap para poder comunicarse con el servidor de aplicaciones. Para poder realizar la comunicación entre el servidor apache y Tomcat es necesario configurar un conector llamado mod_jk [10] propiedad de la compañía "the apache software fundation", con este tipo de comunicación también conseguimos una independencia a nivel de sistema operativo ya que el servidor apache puede trabajar bajo un sistema de software libre como Linux y el servidor Tomcat bajo sistema Windows, especificado en los requisitos no funcionales de la aplicación para poder ejecutar Scripts en lenguaje Cosel.

A pesar de todo el ecosistema necesario no ha sido uno de los aspectos críticos a la hora de programar el aplicativo, pero si ha influenciado para determinar algunas decisiones a la hora de seleccionar las herramientas.

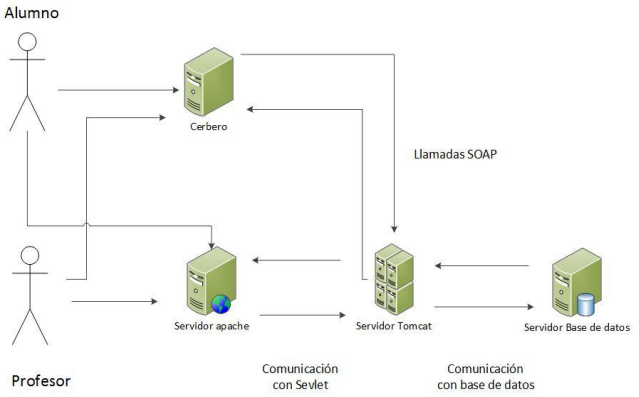


Figura 3: Comunicación global aplicación

A continuación se profundizará en los aspectos más puritas del diseño, diseño de base de datos, clases y las interfaces.

6.1 Diseño base de datos

Para realizar el diseño de la base de datos se ha preferido por seguir un modelo de desarrollo estándar como el modelo entidad-relación. Se ha descartado la utilización de otras tecnologías más actuales como base de datos no relacionales como puede ser MongoDB [11] principalmente porque el tipo de información a tratar es estructurada y no es necesario analizar información que provenga de fuentes no estructuradas.

Entrando más en detalle, la base de datos ha sido implementada bajo un sistema MySQL [12] con una conexión del tipo JDBC [13] para evitar el posible robo de datos a través de la inyección de consultas mediante formularios. Inicialmente se quiso utilizar el sistema de base de datos del Microsoft el SQL Server 2014, pero debido a varios problemas para instanciar la conexión mediante el driver de Hibernate [14] se descartó.

En primeras versiones de la base de datos se optó por diseñar unas tablas centrales y empezar con la codificación de las clases para realizar una familiarización con las herramientas a utilizar, obteniendo al final el diseño en entidad-relación que puede visualizarse en la figura 4:

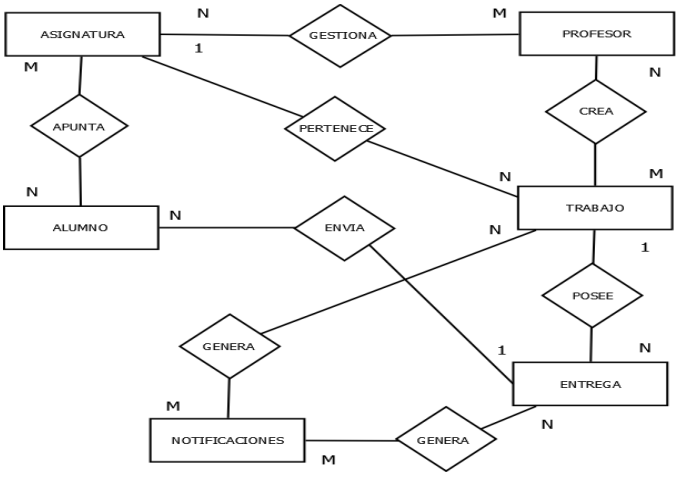


Figura 4: diagrama entidad-relación.

Tras pasar el modelo mediante la herramienta para la gestión de base de datos MySQL Workbench genera automáticamente la formalización del modelo, generando el siguiente diagrama y tablas anexo A1.

6.2 Estructura de la aplicación y diseño clases

Para realizar un correcto diseño de la aplicación primero es necesario realizar una correcta implementación inicial de *Spring* y una jerarquía inicial de paquetes Java.
Al iniciar el trabajo con *Spring* lo primero que se observa es que se genera una estructura estándar con los siguientes puntos:

- Subdirectorio src/main/resources
- Subdirectorio src/main/webapp
- El directorio target
- Fichero Pom.xml

El primer subdirectorio contendrá todos los archivos que son utilizados por la aplicación, en el caso de este aplicativo contiene el fichero persitance.xml con la unidad para la persistencia de la aplicación. En la figura 5 se puede visualizar la estructura del documento:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0">
    <persistence-unit name="correctorPU" transaction-type="RESOURCE_LOCAL">
    </persistence-unit>
</persistence>
```

Figura 5: persistencia del aplicativo.

El subdirectorio 'src/main/webapp' contiene todos los archivos que no son código java, como archivos de configuración o documentos web JSP [15]. En el caso del corrector se puede observar en la figura 6 el árbol de directorios existente tras la finalización de la primera versión del aplicativo:

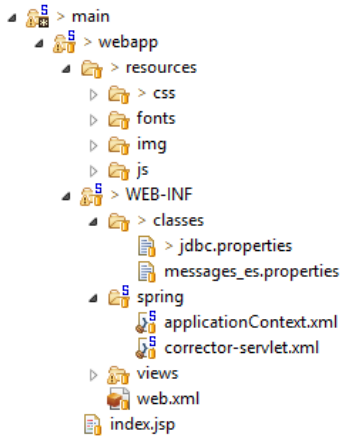


Figura 6: Árbol de directorios de webapp.

Como se observa existen dos directorios principales, el directorio 'resources' contiene todos los archivos que dan estilo a la web, como pueden ser documentos CSS [16], fuentes, imágenes, etcétera… Sin embargo el segundo directorio 'WEB-INF' contiene la configuración propia del aplicativo, dejando a un lado la carpeta 'views' que como su nombre indica contiene las diferentes vistas de la aplicación. Efectuando un análisis más profundo de la carpeta WEB-INF se observan que existen subdirectorios, el primero únicamente contiene dos archivos de configuración del tipo *properties*. El primero es utilizado para almacenar toda la información necesaria como; nombre de usuario, contraseña y driver para establecer la conexión con la base de datos. En la figura 7 se puede observar cual es la estructura del fichero.

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/compiladores
jdbc.username=root
jdbc.password=
hibernate.dialect=org.hibernate.dialect.MySQLDialect
jpa.database = MYSQL
hibernate.generate_statistics = true
hibernate.show_sql = true
jpa.showSql = true
jpa.generateDdl = true
```

Figura 7: Estructura jdbc.properties

Estos parámetros son utilizados por el contexto de la aplicación definido en el archivo applicationContext.xml.
El segundo fichero únicamente contiene etiquetas con mensajes para poder definir varios idiomas en futuras versiones del aplicativo.

Para finalizar con la explicación del subdirectorio 'WEB-INF' se tiene que destacar la importancia de los archivos web.xml y corrector-servlet. En el archivo web.xml se definen todos los parámetros que tienen relación con el contexto de la aplicación, servlet, listeners, sesiones etcétera… En el caso del aplicativo la configuración más importante que se ha realizado es la definición del servlet, concretamente el servlet del tipo utilizado un dispatcher servlet. Este tipo de servlet permiten mapear las URI entrantes y retornar una vista asociada a esa URI. En la figura 8 se puede ver un esquema de su funcionamiento.

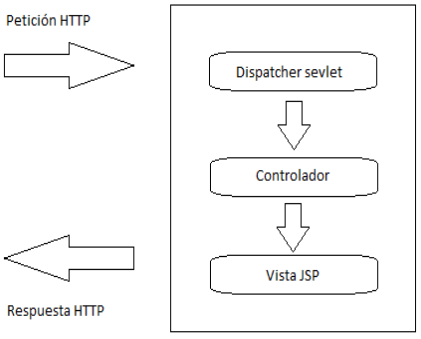


Figura 8: Funcionamiento dispatcher servlet.

En el caso del corrector las URI mapeadas son HTML. Por último el archivo de configuración corrector-servlet, define varios beans [17], en el proyecto el más importante ha sido la generación del fijo y el sufijo, de esta manera se consigue tener una independencia del controlador y la vista. En la figura 9 se puede observar cual ha sido la configuración del bean.

```
<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"></property>
  <property name="prefix" value="/WEB-INF/views/"></property>
  <property name="suffix" value=".jsp"></property>
</bean>
```

Figura 9: Configuración prefijo y sufijo.

Continuando con el directorio target y el archivo pom.xml que faltaban por analizar, el primero únicamente contiene los archivos del tipo war [18] generados para realizar *deploys* de la aplicación de forma más ágil, el archivo pom.xml contiene todas las librerías que están dentro del proyecto, es una herramienta que ha resultado de gran utilidad puesto que posee un cuadro de mando que permite realizar las búsquedas de forma automática, en la figura 10, se puede observar el conjunto de librerías utilizadas en el proyecto

```
junit : 3.8.1 [test]
spring-core : ${org.springframework.version}
spring-webmvc : ${org.springframework.version}
servlet-api : 2.5 [provided]
jstl : 1.2
standard : 1.1.2
jpa-api : 2.0-cr-1
hibernate-entitymanager : 4.2.0.Final
spring-orm : ${org.springframework.version}
mysql-connector-java : 5.1.24
validation-api : 1.1.0.Final
hibernate-validator : 5.2.0.Beta1
spring-security-core : 4.0.1.RELEASE
spring-security-web : 4.0.1.RELEASE
spring-security-config : 4.0.1.RELEASE
mail : 1.4
commons-fileupload : 1.3.1
commons-io : 2.4
javax.el-api : 3.0.1-b04
```

Figura 10: Librerías utilizadas durante el proyecto.

En este punto ya queda expuesta cual es la configuración que se ha realizado en los directorios y archivos que se generan por defecto al crear un nuevo proyecto. A partir de aquí se han generado dos nuevos *source packadges*, uno para la aplicación y otro para test. Ambos contienen los mismos paquetes en su interior:

- domain
- repository
- services
- validator
- web

Esta jerarquía de paquetes, permite modular la aplicación en capas siendo el paquete *domain* el encargado de almacenar el núcleo de la aplicación, *repository* la capa que realiza las peticiones a la base de datos, web donde están los controladores, *service* una capa de enlace entre la capa de controlador y la de persistencia, por último *validator* utilizada para la validación de formularios.

En el paquete de domain como se ha explicado contiene el núcleo de la aplicación. Este núcleo está formado por el diagrama de clases de la figura 11. Está compuesto por las clases del tipo POJO[19] Teacher, Student, Work, Delivery, Notification y Subject. Todas las clases mencionadas hacen referencia a las tablas principales de la base de datos y están mapeadas a través de JPA, para que Hibernate pueda hacer uso de la propiedad "reflecion" de java y acceda a los objetos. En si mismo las únicas obligaciones que impone este ORM y que comparten todas las clases mencionadas son las siguientes:

- Uso de clases POJO
- Implementar un constructor sin argumentos
- Usar nomenclatura JavaBean y una visión privada de las propiedades
- dotar de un identificador único a las clases

Las restantes clases MailSender, CommandExecute, HttpSessionControl y CreateStudent son utilizadas para dar soporte a las diferentes capas como pueden ser la web o services. En el caso de HttpSessionControl es utilizada por la capa web para llevar un control de si la sesión y si es del profesor o alumno, además esta clase utiliza el patrón de diseño *Sigletone* evitando así que exista más de una instancia de esta clase. El resto de clases son utilizadas por los diferentes clases que pertenecen a la categoría de servicios, por ejemplo MailSender es utilizada para enviar mails cuando un alumno solicita una contraseña, CommandExecute es una clase muy versátil que se utiliza tanto para ejecutar Scripts en Cosel como para ejecutar cualquier tipo de comando en entorno Windows y por último CreateStudent es una clase de apoyo que permite subir un fichero del tipo texto y crear una lista de nuevos estudiantes que tendrán acceso a la aplicación.

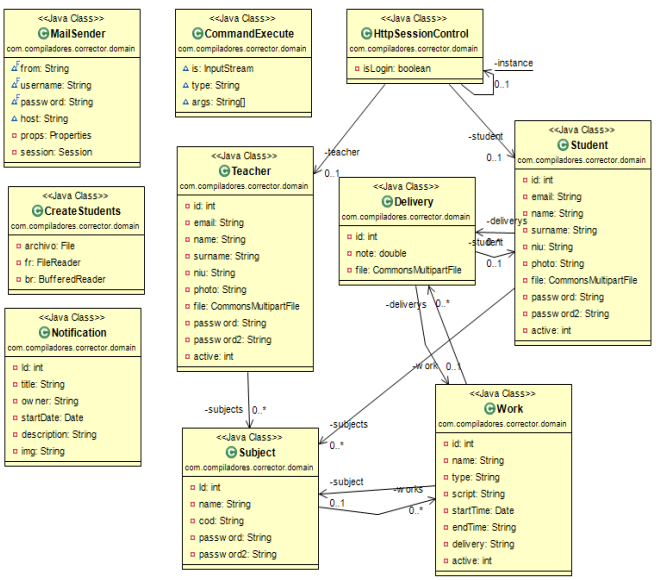


Figura 11: Diagrama de clases.

Para concluir con este apartado cabe destacar que es necesario indicarle a *Spring* de que elemento es cada clase, por ejemplo para crear una clase del tipo controlador y Spring lo reconozca como tal se debe de añadir la etiqueta @Controller como se muestra en figura 12.

```
@Controller
@RequestMapping("student")
public class StudentController
```

Figura 12: Mapeo de la case StudentController como controlador.

Para el mapeo de un servicio se debe añadir la etiqueta @Component, a los cuales se puede instanciar un atributo mediante la etiqueta @Autowired sin necesidad de crear un nuevo objeto y por último los clases que hacer referencia a repositorios se les debe añadir la etiqueta @Repository y darles un nuevo nombre.

Uno de los apartados más útiles a la hora de realizar el acceso a base de datos mediante los @Repository es a través de Entity Manager, se pueden generar transacciones con bloqueo de tablas añadiendo la etiqueta @Transactional (readOnly = false) a un método. Un readOnly igual false indica que la tabla a la que hace referencia va a almacenar un nuevo valor, con lo cual esta tabla no puede ser accesible por otras peticiones para no generar una inconsistencia en la base de datos, mientras que si readOnly es igual a true no bloqueará la tabla y dejará que se realicen otras operaciones de lectura de forma simultánea.

6.3 Diseño de interfaces

El diseño de las interfaces se ha realizado de mediante las tecnologías, HTML5, JSP, CSS y Javascript. Para una mayor facilidad en la implementación se ha decidido utilizar el framework Boostrap 3 [20] para su implementación además de un plantilla estándar en forma de blog que permite hacer que la aplicación sea responsive [21].
Principalmente el aplicativo define una pantalla inicial donde los alumnos y profesores deben introducir sus iniciales para iniciar sesión en el sistema como puede verse en la figura 13

Figura 13: ventana de Login.

Y un segundo layout [22] diferenciado en la cabecera, donde los alumnos únicamente tienen acceso a la lista de entregas, el profesor en cambio tiene más funciones como añadir asignaturas, entregas, profesores, y alumnos. En figura 14 se muestra cual es el esqueleto estándar de la aplicación

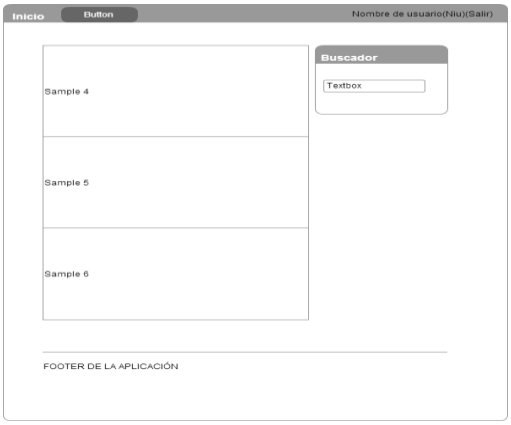


Figura 14: Layout de la aplicación.

Para finalizar con el apartado del diseño cabe remarcar que el layout utilizado para la vista móvil varia debido a que el espacio en estas plataformas es más reducido, y no puede solucionarse el problema con una simple reducción del tamaño de los elementos, es necesario reordenar la información que se muestra en pantalla, en la figura 15 se puede observar la interfaz desarrollada

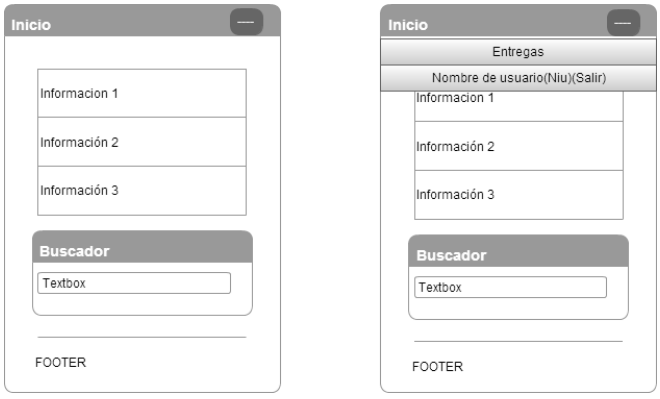


Figura 15: Vista móvil

7 TEST

La fase de test del proyecto como se explica en el aparatado de metodología del proyecto, se realizó mediante TDD, este tipo de metodología de codificación ha asegurado un nivel de calidad aceptable en la aplicación aunque también ha hecho que la codificación de la misma sea más lenta. Para poder implementar este tipo de test ha sido necesario añadir la librería de JUNIT al proyecto, y realizar los test unitarios de cada clase.

Añadir que para la búsqueda de nuevos errores se ha utilizado la técnica de Exploratory Testing [23] donde cada error encontrado en la aplicación ha sido anotado en la siguiente plantilla:

ID	TC1
Caso de uso probado	CU1
Descripción del test	El sistema no retorna la contrase-

	ña al usuario
Tipo de error	Implementación
Impacto	Muy crítico
Clases involucradas	StudentController, StudenManager, StudentDao
Controladores involucrados	LoginController
Vistas involucradas	login.jsp
Tablas de la BD involucradas	Alumnos
Estado	Finalizado
Fecha de la prueba	07-05-2015
Fecha de reparación	07-05-2015
Fecha de la última revisión	07-05-2015
Comentarios	Se procede a corregir el mapeo de una columna Id_asignatura

Figura 17: Creación de alumnos.

Si la aplicación no devuelve ningún mensaje de error es que los alumnos se han registrado correctamente.

Llegados a este punto se puede iniciar la fase donde el alumno puede solicitar su contraseña a través de la vista de previa de acceso al sistema como se puede observar en la figura 18.

8 Resultados

Los resultados obtenidos del desarrollo de la aplicación han sido satisfactorios a pesar de que algunos objetivos no han podido ser desarrollados debido a que la curva de aprendizaje de todas las tecnologías implicadas en el proyecto ha sido muy costosa, especialmente en la primera parte del desarrollo, donde se realizó la primeras configuraciones con Spring y además requerían de la participación de nuevos stakeholders ajenos al proyecto.

Para poder realizar una valoración real del estado de esta primera versión del aplicativo, en este apartado se centrará en la realización de un caso real de uso de la aplicación, desglosado en los siguientes sub apartados.

Además en el anexo A2 se puede visualizar las diferentes visiones de la aplicación en función del tamaño de pantalla.

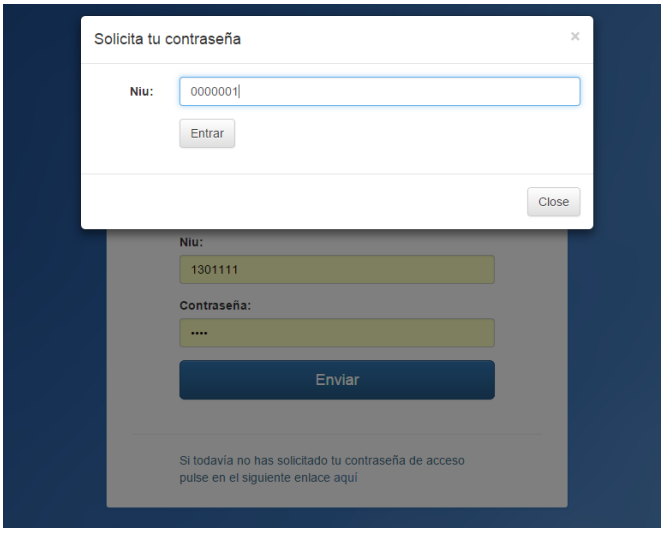


Figura 18: Solicitud de contraseña

8.1 Creación de nuevos alumnos

Como se ha explicado durante el desarrollo del artículo, la creación de un nuevos alumnos, se establece en dos fases:

- El profesor añade una nueva lista de alumnos.
- El alumno solicita su contraseña.

la realizción del primer apartado conlleva tener que genrar un documento con fotmato txt como el que se muestra en la figura 16:

```
Alumno1;0000001;christian.garciamin@e-campus.uab.cat;741852
Alumno2;0000002;christian.garciamin@e-campus.uab.cat;741852
Alumno3;0000003;christian.garciamin@e-campus.uab.cat;741852
```

Figura 16: Formato del archivo a subir al sistema.

Como puede observarse existen dos campos numéricos, el primero hace referencia al NIU del estudiante y el último a la asignatura que tendrá acceso. A partir de este punto, un profesor previamente autentificado en el sistema podrá realizar la subida del fichero como muestra la figura 17

Como resultado de esta operación, el alumno recibe un mail a como puede observarse en la figura 19 y ya podrá acceder al sistema

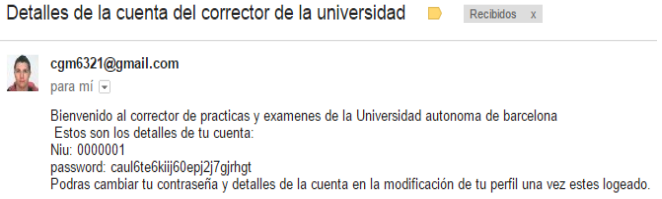


Figura 19: Correo solicitud de contraseña.

Añadir que la contraseña que recibe el usuario se genera de forma aleatoria y que además queda encriptada en la base de datos como se observa en la figura 20:

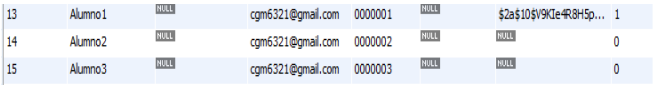


Figura 20: encriptación de contraseña.

Como se ha podido observar, los resultados obtenidos en este apartado son consistentes tanto con los objetivos establecidos como con diversos requisitos de la aplicación.

8.1 Corrección automática

En este subapartado se evaluará la corrección automática de una práctica y un trabajo para resaltar las diferencias que hay entre ambos. Igual que en el punto anterior para realizar una corrección automática requiere de varios pasos:

- El profesor crea un nuevo trabajo y lo asigna a una asignatura.
- El alumno realiza la entrega

El primer apartado igual que para crear una asignatura, requiere de la autentificación previa por parte del profesor. Una vez iniciada su sesión se dirige al apartado Entrega/añadir como en la figura 21.

Figura 21: Creación de una nueva practica.

La creación de un nuevo trabajo en lugar de una práctica, únicamente requiere cambiar el segundo dato del formulario por un trabajo. Tras la creación de la práctica y el trabajo el estudiante ya podrá realizar la entrega de ambos desde el menú correspondiente, en la figura 22 puede observarse como es la interfaz para realizar una entrega.

Figura 22: subida de trabajo

Si el trabajo se ha entregado de forma correcta el alumno recibirá una alerta de que se ha entregado correctamente, en caso contrario recibirá un mensaje de error. Si la entrega es correcta podrá descargar el archivo enviado.

La nota en el caso de entregar una práctica, no la podrá visualizar hasta que se cierre el plazo de entrega mientras que en el caso de la entrega de un trabajo si podrá hacerlo como puede verse en la figura 23:

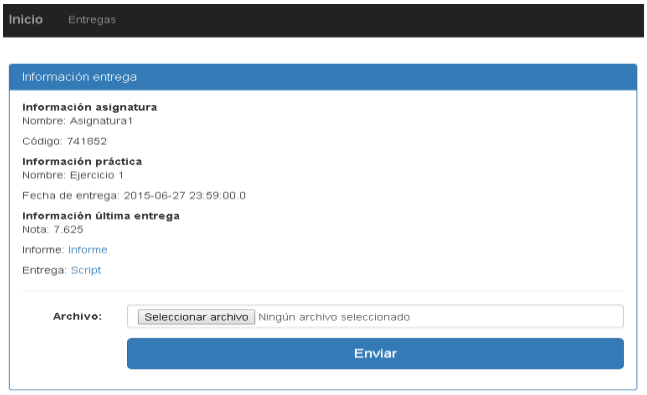


Figura 23: Ejercicio entregado

Por último el profesor en su sección de entregas puede visualizar cuantos alumnos ya la han entregado y visualizar su nota, y descargar tanto los archivos generados como el fichero de entrega del alumno. En la figura 24 se puede observar como es la interfaz para ver la información mencionada:

Figura 24: Entregas realizadas práctica

9 LÍNEAS DE TRABAJO FUTURAS

Las líneas de desarrollo futuras de la aplicación se establecerán en función de los nuevos errores que se detecten al ejecutar la aplicación en un entorno productivo o beta principalmente para realizar pruebas de conexiones simultáneas al sistema y ver cómo se comporta.

En una primera valoración realizada tras la finalización de proyecto, se detecta la necesidad de realizar una mejora de la interfaz principal que permita realizar una paginación de las diferentes notificaciones.

Centrándonos en el apartado de nuevas implementaciones se podría llegar a implementar el servicio de autentificación centralizada de usuarios de la universidad en lugar de la página de login que se ha desarrollado en la primera versión, añadir nuevas funcionalidades como la modificación de entregas y establecer una base para determinar de que se encargan los Scripts u qué la aplicación.

Por último es necesario implementar el sistema de llamadas Soap para poder establecer la conexión con la plataforma cerbero tal y como se explica en el apartado de diseño.

10 Conclusiones

En general el trabajo que se he realizado ha sido satisfactorio acorde a los requerimientos establecidos en los primeros compases del proyecto. En un principio se subestimó la complejidad de programar en un entorno tan rígido como java, y no fue hasta la finalización de la primera parte del proyecto cuando se realizó una planificación más acorde con el tiempo real destinado al desarrollo.

Se podría decir que todo el trabajo que se ha realizado posee una calidad mínima gracias al desarrollo mediante la técnica de codificación TDD y al Exploratory Testing, donde se han detectado ciertos problemas sobre todo con la interfaz que no son posibles de detectar con un test de unidad clásico.

Añadiendo un punto crítico a mí trabajo he de decir que me habría gustado implementar algunos de los objetivos que se han propuesto en las líneas de trabajo futuras y haber podido por lo menos realizar una prueba en fase alpha de la aplicación, pero se opto por realizar mejoras en los en las funcionalidades ya implementadas.

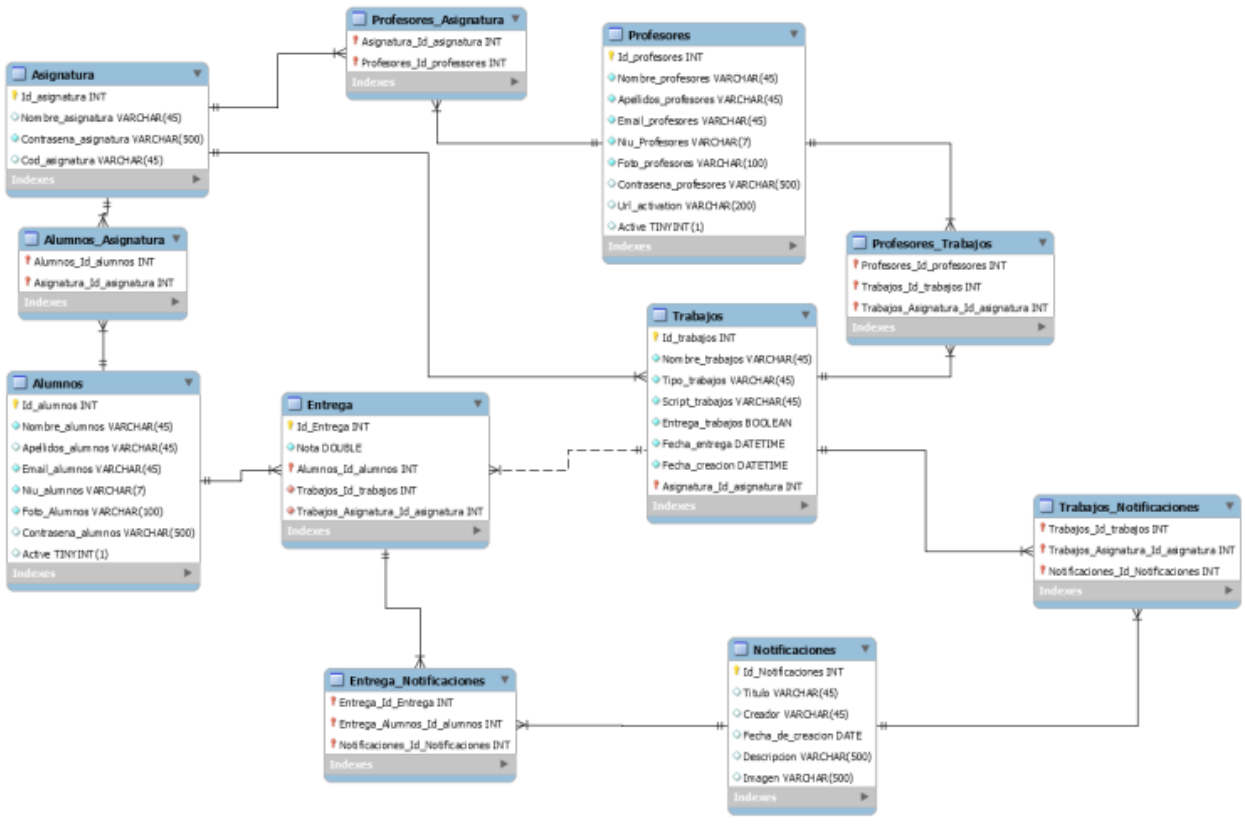
Agradecimientos

Para concluir con este articulo, quiero dar las gracias a mí pareja Sonia Martínez, por realizar una lectura de todos los documentos realizados a lo largo del desarrollo del proyecto y dame su visión crítica para poderlos mejorar. También agradecer al tutor de proyecto Javier Sánchez Pujadas, la ayuda y la libertad que me ha ofrecido durante el desarrollo de la aplicación, no poniendo ninguna restricción y dejándome tomar la mayor parte de las decisiones.

Bibliografía

[1] Bruce Eckel, "Piensa en java" (Edición Española), 2002
[2] ¿Qué es Php?
http://goo.gl/FqnFmT [Fecha de consulta: 15-06-2015]
[3] Acerca de moodle
https://goo.gl/s8EAxj [Fecha de consulta: 15-06-2015]
[4] ¿Qué es joomla?
http://goo.gl/0YGJ7V [Fecha de consulta: 15-06-2015]
[5] Walls Craig, "Spring" Tercera edición, 2011
[6] Soap Introduction
http://goo.gl/8tcUzB [Fecha de consulta: 15-06-2015]
[7] [6] Versión 2.4 de la documentación del servidor http
http://goo.gl/zUPV7P[Fecha de consulta: 05-03-2015]
[8] HTML 5 Introduction
http://goo.gl/4xpbJB [Fecha de consulta: 20-06-2015]
[9] Apache Tomcat
http://goo.gl/hVOhzh [Fecha de consulta: 5-03-2015]
[10] The Apache Tomcat Connector
http://goo.gl/pO8mGR [Fecha de consulta: 05-03-2015]
[11] Mongo DB:¿Qué es, como funciona y cuando debemos usarlo?
http://goo.gl/Z8C9w9 [Fecha de consulta: 20-06-2015]
[12] MySql ¿Qué es y para qué sirve?
http://goo.gl/WefU03 [Fecha de consulta: 07-03-2015]
[13] JDBC conepts
http://goo.gl/s31CB1 [Fecha de consulta: 14-04-2015]
[14] Christian Bauer, Gavin King, Gary Gregory, "java pesistence with hibernate", 2007
[15] JavaServer Pages Technology
http://goo.gl/ZYP9Bo [Fecha de consulta: 12-03-2015]
[16] ¿Qué es css?
http://goo.gl/OmposZ [Fecha de consulta: 20-06-2015]
[17] Introducción a los JavaBeans
http://goo.gl/kPiuuz [Fecha de consulta: 16-04-2015]
[18] Archivos War
http://goo.gl/oVoe0O[Fecha de consulta: 20-06-2015]
[19] Clases persistentes
https://goo.gl/ZrHS2U[Fecha de consulta: 15-04-2015]
[20] Bootstrap 3 tutorial
http://goo.gl/4vKkNR [Fecha de consulta: 07-03-2015]
[21] HTML Responsive Web Design
http://goo.gl/uN3E4z [Fecha de consulta: 07-03-2015]
[22] Definición de layout
http://goo.gl/qaFjge [Fecha de consulta: 20-03-2015]
[23] Exploratory Software Testing
https://goo.gl/NFyccT [Fecha de consulta: 08-03-2015]

Anexo
A1. Diagrama de clases y tablas



Alumnos
Id_alumnos INT NOT NULL AUTO_INCREMENT, PK
Nombre_alumnos VARCHAR(45) NOT NULL
Apellidos_alumnos VARCHAR(45) NULL
Email_alumnos VARCHAR(45) NOT NULL
Niu_alumnos VARCHAR(7) NOT NULL
Foto_Alumnos VARCHAR(100) NOT NULL
Contrasena_alumnos VARCHAR(500) NULL
Active` TINYINT(1) NULL
Indice
Id_alumnos
Profesores
Id_professores INT NOT NULL AUTO_INCREMENT, PK
Nombre_professores VARCHAR(45) NOT NULL
Apellidos_profesores VARCHAR(45) NOT NULL
Niu_Profesores VARCHAR(7) NOT NULL
Email_profesores VARCHAR(45) NOT NULL
Foto_profesores VARCHAR(100) NOT NULL
Contrasena_profesores VARCHAR(500) NULL
Active TINYINT(1) NULL
Indice
Id_profesores
Asignatura
Id_asignatura int(11) NOT NULL AUTO_INCREMENT,

PK
Nombre_asignatura varchar(45) DEFAULT NULL
Cod_asignatura varchar(45) NOT NULL
Contrasena_asignatura varchar(500) NOT NULL
Indices
Id_asginatura
Nombre_asignatura
Trabajo
Id_trabajos int(11) NOT NULL AUTO_INCREMENT, PK
Nombre_trabajos varchar(45) NOT NULL
Tipo_trabajos varchar(45) NOT NULL
Script_trabajos varchar(45) NOT NULL
Entrega_trabajos tinyint(1) NOT NULL
Fecha_entrega datetime NOT NULL
Fecha_creacion datetime NOT NULL
Asignatura_Id_asignatura int(11) NOT NULL, PK
Indices
Id_trabajos
Asignatura_Id_asignatura
Entrega
Id_Entrega int(11) NOT NULL AUTO_INCREMENT, PK
Nota double NOT NULL
student_Id_alumnos int(11) NOT NULL ,PK

work_Id_trabajos int(11) NOT NULL, PK
Trabajos_Asignatura_Id_asignatura int(11) DEFAULT NULL
Indices
Id_Entrega
Student_Id_Alumnos
Work_Id_trabajos

Notificaciones
Id_Notificaciones int(11) NOT NULL AUTO_INCREMENT
Titulo varchar(45) DEFAULT NULL
Creador varchar(45) DEFAULT NULL
Fecha_de_creacion date DEFAULT NULL
Descripcion varchar(500) DEFAULT NULL
Imagen varchar(500) DEFAULT NULL
Indices
Id_notificaciones

A2. VISTAS PÁGINAS