UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

Centro de Tecnologías para la Formación

# Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Jorge Eduardo Pérez Batista

Tutores:

Ing. William Simón Ramírez

Ing. Noralys Almeida Milanés

Ing. Lisset Salazar Gómez

La Habana, 21 de junio de 2017

Año 59 de la Revolución

## DECLARACION DE AUTORIA

Declaro ser el autor de la tesis “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

**Jorge Eduardo Pérez Batista**

**Autor**

_____ _____

**Ing. William Simón Ramírez** **Ing. Noralys Almeida Milanés**

**Tutor** **Tutor**

_____

**Ing. Lisset Salazar Gómez**

**Tutor**

## DEDICATORIA

*A los que necesiten fuerzas alguna vez para terminar lo que empiecen, la universidad, al igual que todo son solo etapas de la vida, saber darle a cada cosa la importancia que merece es lo que nos hace crecer.*

**AGRADECIMIENTOS**

*A mi familia por el apoyo incondicional y el increíble soporte a errores que han tenido y a mi tutor William Simón por nunca dejar que me sintiera solo a lo largo de esta etapa de mi vida.*

## RESUMEN

En Cuba el uso de las aulas tecnológicas es un tema abordado desde el año 2014, cuando se tomó en cuenta variantes de diversos fabricantes para incorporarlos en los primeros niveles de la educación. La Universidad de las Ciencias Informáticas lleva a cabo el proyecto ATcnea desde el año 2016 con el objetivo de crear un software de aula tecnológica mediante el uso de tecnologías libres. Actualmente, los recursos educativos a utilizar a través de ATcnea deben ser descargados localmente por el profesor previo a la clase y suministrados a los estudiantes en el salón. Esto limita la disponibilidad de recursos y consume tiempo innecesariamente, además de dificultar la gestión de los recursos, haciendo que su almacenamiento sea solo local, y su preservación a largo plazo nula.

La solución propuesta permite utilizar recursos almacenados en repositorios digitales como parte de los módulos del software ATcnea, brindando a los docentes recursos de alta calidad y valor científico. El proceso de desarrollo fue guiado por la metodología AUP-UCI, la cual en el escenario escogido encapsula los requisitos funcionales en historias de usuarios. Concluida la fase de implementación se realiza la etapa de pruebas, esto garantiza la entrega de un producto de alta calidad.

Como resultado se obtuvo un módulo que permite al software ATcnea consumir recursos almacenados en repositorios digitales, esto aumenta la disponibilidad de recursos que apoyan el proceso de enseñanza-aprendizaje.



**Palabras claves:** ATcnea, Aula tecnológica, recurso, repositorio

## ÍNDICE

**ÍNDICE DE TABLAS E ILUSTRACIONES**

## INTRODUCCIÓN

El uso de las Tecnologías de la Información y las Comunicaciones (TIC) en la educación es ampliamente difundido. La utilización de herramientas digitales educativas ha permitido que el proceso de enseñanza y aprendizaje se enriquezca a través de mejoras en la motivación hacia el estudio al aplicar métodos más interactivos. Su constante desarrollo en el mundo actual se evidencia en la variedad de recursos existentes y el surgimiento de otros en correspondencia a la evolución de las tecnologías.

Las aulas tecnológicas forman parte de la gama de recursos que estudiantes y profesores utilizan con el fin de aumentar la calidad de su preparación. En la actualidad software de este tipo son usados en varios centros de enseñanza, y grandes compañías como Microsoft y Samsung han desarrollado sus propios softwares de gestión de aula.

En Cuba, con el objetivo de lograr la soberanía tecnológica, la seguridad de los procesos informatizados y el continuo desarrollo y modificación de los mismos, se lleva a cabo la creación de software respondiendo a las necesidades crecientes de cada sector. En la educación el uso de las aulas tecnológicas es un tema abordado desde el año 2014, cuando se comenzó a tomar en cuenta variantes de diversos fabricantes para incorporarlos en los primeros niveles de la educación en el país.

La Universidad de las Ciencias Informáticas lleva a cabo el desarrollo del producto ATcnea desde el año 2016 como una iniciativa del Grupo para la Electrónica (GELECT), de la empresa GEDEME, y con el equipamiento aportado por la empresa china Haier. Esta tarea fue asignada al Centro de Tecnologías para la Formación (FORTES) quien se hizo cargo del proyecto en su totalidad.

El software se desarrolla con el objetivo de gestionar los procesos que se llevan a cabo en un aula tecnológica con uso tecnologías libres. ATcnea consiste en dos aplicaciones, una de escritorio (aplicación del profesor) y una en sistema operativo Android (aplicación del estudiante). En junio del 2016 se obtiene la versión beta del producto ATcnea y es presentada en la Feria CUBAINDUSTRIA, donde tuvo gran aceptación y se identificaron nuevas necesidades. El software ATcnea presenta módulos que gestionan el

flujo de la clase, algunos como el módulo video, presentación, o asistencia facilitan la interacción entre profesores y estudiantes y agilizan el proceso enseñanza-aprendizaje.

Actualmente, cualquier recurso educativo a utilizar a través de ATcnea debe ser descargado localmente por el profesor previo a la clase y suministrado a los estudiantes en el salón. Este modo de gestión limita la disponibilidad de recursos y consume tiempo innecesariamente. Además, dificulta la gestión de los recursos, por lo que su almacenamiento es solo local, y su preservación a largo plazo nula.

La utilización de repositorios digitales como fuentes de recursos ofrece una serie de ventajas que agilizan el acceso, calidad y eficiencia en la utilización de recursos. Algunas de las características más relevantes de los repositorios son:

- **Unidad:** Todos los recursos están almacenados en una misma base de datos, es más fácil su recuperación.
- **Libre acceso a los contenidos.**
- **Preservación a largo plazo:** La garantía de preservación que supone el depósito en un repositorio es mucho mayor que la que ofrecen las páginas web personales, los servidores, etc.
- **Interoperabilidad:** Utilizan el protocolo OAI-PMH (Open Access Initiative-Protocol for Metadata Harvesting) que se basa en la utilización del modelo de metadatos Dublin Core para la descripción del contenido de las publicaciones, lo que facilita la recuperación de los contenidos de los repositorios a través de buscadores y recolectores de información.
- **Comunicación:** Al dotar a los usuarios de esta herramienta de autoarchivo, se facilita la comunicación e intercambio de información científica entre ellos. Contribuye a la más amplia difusión de su trabajo. (UvaDoc: Repositorio Documental de la Universidad de Valladolid, 2013)

Partiendo de lo anteriormente expuesto se llega al siguiente **Problema de la investigación**: ¿Cómo facilitar el acceso del software ATcnea a recursos almacenados en repositorios digitales?
Se definió para la investigación como **objeto de estudio** los “repositorios digitales”, enmarcado en el **campo de acción:** “el acceso del software ATcnea a recursos almacenados en repositorios digitales”.

Debido a la necesidad de agilizar la utilización de recursos, así como aumentar la disponibilidad de los mismos en el software ATcnea, se define como **objetivo general:** “Desarrollar un módulo del software ATcnea para el acceso a recursos almacenados en repositorios digitales que facilite al profesor su utilización en clases”. Para lograr el cumplimiento del objetivo general se definen los siguientes **objetivos específicos:**

1. Definir marco teórico conceptual de la investigación.
2. Seleccionar la metodología, herramientas y lenguajes de programación a usar en la implementación del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales".
3. Diseñar las funcionalidades del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales".
4. Implementar las funcionalidades del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales".
5. Validar el funcionamiento del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales".

Con el objetivo de cumplir los objetivos específicos planteados se elaboraron las siguientes **tareas a cumplir**:

1. Análisis de soluciones similares existentes.
2. Estudio y selección de las técnicas, herramientas y metodologías a emplear en el desarrollo de la solución.
3. Redacción del diseño teórico metodológico de la investigación.
4. Estudio sobre buenas prácticas a utilizar en el desarrollo de software definido en la metodología seleccionada.
5. Descripción de la propuesta de solución, los actores que interactuarán con el sistema, así como de los requerimientos funcionales y no funcionales a implementar.
6. Análisis y diseño de la propuesta de solución mediante la confección de los artefactos definidos en la metodología de desarrollo seleccionada.
7. Definir una arquitectura para la gestión de repositorios en el software ATcnea.
8. Implementar el "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales".
9. Validación funcional de la propuesta solución, mediante las pruebas diseñadas y documentación de los resultados.

Se esperan como **resultados**:

1. Estructura base para el acceso del software ATcnea a recursos almacenados en repositorios.
2. Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales.
3. Investigación sobre interoperabilidad con repositorios digitales.

Los **métodos científicos teóricos** empleados durante el desarrollo del proceso de investigación fueron:

- **Histórico-lógico** al analizar módulos de comportamiento similar al planteado como objetivo y determinar las buenas prácticas de estos.
- **Analítico-sintético** al descomponer el problema de la investigación en varios elementos, estudiar cada uno de ellos y luego fundirlos en la solución propuesta.
- **Modelación** al realizar los diagramas de clases, secuencia y componentes.

Los **métodos científicos empíricos** empleados durante el desarrollo del proceso de investigación fueron:

- **Observación:** Al analizar el comportamiento de aplicaciones homólogas y describir conductas o situaciones en contextos similares a los esperados del software en desarrollo.
- **Entrevista:** La aplicación de este método posibilitará entender el funcionamiento del software ATcnea, además permitirá identificar los requisitos funcionales y no funcionales que debe cumplir el sistema según lo especificado por el usuario. También se utilizó este método para la elección del repositorio a usar como ejemplo de correcto funcionamiento del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

Este documento se encuentra dividido en tres capítulos:

- El primer capítulo es la **“Fundamentación Teórica”** de la investigación. En este capítulo se exponen los principales conceptos que se utilizarán (repositorios, aulas tecnológicas, interconexión) y se analizan sistemas con funcionalidades similares al objeto de estudio con un punto de vista crítico y valorativo.

- El capítulo dos, denominado **“Análisis y diseño”** comprende los requisitos funcionales y no funcionales del módulo a desarrollar. En este capítulo se especifica además el diagnóstico del campo de acción y sus particularidades con el objetivo de realizar una propuesta de solución a partir de lo analizado en el capítulo uno.

- El tercer capítulo: **“Implementación y pruebas de la solución”**, contempla el desarrollo de la propuesta de solución teniendo en cuenta el alcance especificado para el trabajo de diploma, así como la validación de los resultados obtenidos durante el mismo.

# CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el siguiente capítulo se realiza la fundamentación teórica de la investigación, así como la selección de herramientas, metodologías y arquitectura a usar en el proceso de desarrollo de software.

## 1.1. Conceptos y definiciones

### 1.1.1. Aula tecnológica

La educación actual afronta múltiples retos y uno de ellos es dar respuesta a los profundos cambios sociales, económicos y culturales que se prevén. El concepto de aula tecnológica o virtual es un término desarrollado en el siglo XXI. Una de las definiciones más usadas para describir las aulas tecnológicas es: “Nueva forma viable de enseñanza que viene a suplir necesidades, precariedades propias de la educación y la tecnología educativa”. (Rosario, 2007)

Otra de las definiciones sobre este término plantea: Un aula tecnológica es una solución educativa que brinda una experiencia única de aprendizaje. Tiene como objetivo la creación de un ambiente colaborativo, donde la tecnología enriquece el contenido académico de cada asignatura y permite al profesor-estudiante establecer una amplia comunicación. (Segovia Olmo, 2003)

### 1.1.2. Repositorios

El término “repositorio” es ampliamente usado cuando se hace referencia a Tecnologías de la Informática y las Comunicaciones. El profesor Jorge Polanco-Cortés de la Universidad de Costa Rica lo define en sus conferencias como: “Medio para gestionar, almacenar, preservar, difundir y facilitar el acceso a los objetos digitales que alberga”. (Repositorios digitales. Definición y pautas para su creación, 2014)

En este documento se usará el concepto publicado por el sitio web PoliScience donde se define repositorio como: “archivos donde se almacenan recursos digitales de manera que estos pueden ser accesibles a través de internet. Existen tres tipos principales de repositorios:

- **Repositorios institucionales**: son los creados por las propias organizaciones para depositar, usar y preservar la producción científica y académica que generan. Supone un compromiso de la

institución con el acceso abierto al considerar el conocimiento generado por la institución como un bien que debe estar disponible para toda la sociedad.

- **Repositorios temáticos**: son los creados por un grupo de investigadores, una institución, etc. que reúnen documentos relacionados con un área temática específica.
- **Repositorios de datos**: repositorios que almacenan, conservan y comparten los datos de las investigaciones." (PoliScience)

## 1.2. Protocolos de interconexión

### 1.2.1. SWORD

Sword es un protocolo usado en repositorios para poder realizar envíos de contenidos desde otras aplicaciones. Sus siglas corresponden a Simple Web-service Offering Repository Deposit, es decir un Servicio Web simple que ofrece servicios de depósito en un repositorio.

Activar el protocolo Sword permite acceder, mediante un servicio web, a realizar envíos directos al repositorio. Se puede configurar el servicio de envío para poder simplificar u omitir pasos del proceso de envíos y que cualquier usuario registrado o no registrado en el repositorio pueda insertar sus contenidos de forma simple.

Existen librerías para implementar clientes SWORD en los principales lenguajes de programación, como es PHP (SWORD client library), Ruby (SWORD client library), Java (SWORD client and server library) y Python (server library and SWORD 2.0 reference implementation). (Lorenzo, 2012)

### 1.2.2. REST

La Transferencia de Estado Representacional es un estilo de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

El módulo REST API proporciona una interfaz programable para las comunidades, colecciones y elementos. La clave de REST es que un servicio REST no tiene estado, lo que quiere decir que, entre dos llamadas

cualesquiera, el servicio pierde todos sus datos. El estado lo mantiene el cliente y por lo tanto es quien debe pasarlo en cada llamada. Si desea que un servicio REST recuerde algún dato, se le debe pasar toda la información en cada llamada. Eso puede ser un usuario y una contraseña, un *token*, cualquier tipo de credenciales o incluso la propia información.

El no tener estado es una desventaja clara: tener que pasar el estado en cada llamada es, como mínimo, tedioso, pero la contrapartida es clara: escalabilidad. Para mantener un estado se requiere algún sitio (generalmente memoria) donde guardar todos los estados de todos los clientes. A más clientes, más memoria, hasta que al final podemos llegar a no poder admitir más clientes, no por falta de CPU, sino de memoria.

Una diferencia fundamental entre un servicio web clásico y un servicio REST es que el primero está orientado a invocar métodos sobre un servicio remoto, mientras que el segundo está orientado a recursos. En una API REST la idea de “servicio” como tal desaparece. Lo que presenta son recursos, accesibles por identificadores. Sobre esos recursos se pueden realizar acciones, generalmente diferenciadas a través de verbos HTTP distintos. (Tomás, 2014)

### 1.2.3. **OAI-PMH** (Open Archives Initiative - Protocol for Metadata Harvesting)

El Protocolo OAI para la Recolección de Metadatos, define un mecanismo para la recolección de registros que contienen los metadatos de los repositorios. El OAI-PMH ofrece a los Proveedores de Datos una opción técnica sencilla para poner sus metadatos a disposición de servicios basados en los estándares abiertos HTTP (Hypertext Transport Protocol) y XML (Extensible Markup Language). Los metadatos que son recolectados pueden estar en cualquier formato establecido por una comunidad (o por cualquier conjunto específico de Proveedores de Datos y Proveedores de Servicios), con independencia de que hayan establecido el Dublin Core no cualificado para proporcionar un nivel básico de interoperabilidad. Por lo tanto, los metadatos de distintas fuentes pueden ser reunidos en una base de datos, y se pueden ofrecer servicios sobre la base de esta recolección centralizada, o "agregación" de datos.

El vínculo entre los metadatos y el contenido no está definido por el protocolo OAI. Es importante darse cuenta de que OAI-PMH no proporciona una búsqueda a través de estos datos, simplemente permite reunir

los datos en un sitio. Para ofrecer un servicio, el método de la recolección debe combinarse con otros mecanismos. (Carpenter, 2003)

**Características de OAI-PMH**

OAI-PMH tiene características que la rigen, contribuyendo a la estandarización de la interacción con los contenidos de los repositorios: (CLARISE, 2013)

1. Su funcionamiento se basa en una arquitectura cliente-servidor en la que un servicio recolector de metadatos pide información a un proveedor de datos.
2. Las peticiones se expresan en HTTP, al usar únicamente los métodos GET o POST.
3. Todas las respuestas deben ser documentos XML bien formados codificados en UTF-8.
4. Fechas y tiempo se codifican mediante la ISO 8601 y se expresan en UTC.
5. Soporta la difusión de registros en diversos formatos de metadatos.
6. Tiene control de flujo.
7. Cuando hay un error o una excepción los repositorios deben indicarlos distinguiéndolos de los códigos de estado HTTP por incluir uno o más elementos de error en la respuesta.

**Protocolo de peticiones de OAI-PMH**

- GetRecord. Utilizado para recuperar un registro concreto. Necesita dos argumentos: identificador del registro pedido y especificación del formato bibliográfico en que se debe devolver.
- Identify. Utilizado para recuperar información sobre el servidor: nombre, versión del protocolo que utiliza, dirección del administrador, etc.
- ListIdentifiers. Recupera los encabezamientos de los registros, en lugar de los registros completos. Permite argumentos como el rango de fechas entre los que se deseen recuperar los datos.
- ListRecords. Igual que el anterior, pero recupera los registros completos.
- ListSets. Recupera un conjunto de registros. Estos conjuntos son creados opcionalmente por el servidor para facilitar una recuperación selectiva de los registros. Sería una clasificación de los contenidos según diferentes entradas. Un cliente puede pedir que se recuperen solo los registros pertenecientes a una determinada clase. Los conjuntos pueden ser simples listas o estructuras jerárquicas.

- ListMetadataFormats. Devuelve la lista de formatos bibliográficos que utiliza el servidor. (Barrueco)

### 1.3. Softwares de gestión de aula tecnológica e interconexión con repositorios

Los softwares de gestión de aula tecnológica poseen funcionalidades que permiten al profesor impartir la clase de forma más interactiva que en un aula convencional. Las aulas tecnológicas son usadas a nivel mundial como medios para mejorar el proceso de enseñanza basándose en los avances de la tecnología.

#### 1.3.1. Net Control 2 v7.4.0.385

Net Control 2 ofrece amplias capacidades de control remoto que simplifican el trabajo en un salón de clases. De esta manera, no será necesario dejar el ordenador del profesor para comandar el resto de los dispositivos, ya sea para apagarlos, reiniciarlos, ejecutar algún recurso o realizar alguna configuración en particular. Este software permite desarrollar una misma actividad sobre múltiples equipos simultáneamente, consiguiendo así reducir los tiempos de acción y agilizar arduas operaciones. (Int)

La aplicación ofrece funcionalidades para el trabajo en conjunto en un salón de clases, incluye más de 150 comandos y características de control, enseñanza y vigilancia. El profesor, puede compartir información (textos, presentaciones, videos, audios, y cualquier otro tipo de archivos) con los estudiantes a través del envío de un dispositivo a otro. No incluye entre sus funcionalidades la interoperabilidad con repositorios. El producto opera bajo licencia privativa.

#### 1.3.2. Mythware Classroom Management

Mythware Classroom Management maneja una amplia gama de características que permite un soporte educativo al momento de enseñar y aprender, promueve un ambiente interactivo y colaborativo entre los estudiantes, al brindar todas las herramientas necesarias para una clase organizada y para evaluar el rendimiento de los alumnos. El software facilita la participación del alumno durante la clase y está diseñado para mantenerlos concentrados y atentos en aprender.

Manejo de envío de archivos:

- Distribución de archivos (envía archivos multimedia, fotos, documentos, o cualquier otro formato a los estudiantes).

- Colecta de archivos (colecta archivos enviados al profesor desde el dispositivo de los estudiantes).
- Envío de archivos (permite a los alumnos enviar archivos al profesor por su cuenta). (Pla17)

El software no contempla entre sus funcionalidades la conexión con repositorios, ni realizar ninguna función (descargar, subir, buscar) sobre ellos.

### 1.3.3. iTalc

"iTalc es un software de control de aula, ideal para la gestión de aulas TIC. Se trata de un software de uso libre y de muy sencilla instalación que permite controlar los equipos de los alumnos a distancia, bloquear la pantalla, etc.

Esta aplicación permite hacer un seguimiento de la actividad de cada alumno en su ordenador, generar presentaciones o guiar a un alumno con dificultades sin necesidad de abandonar el puesto de trabajo.

También puede servirnos para visualizar las actividades realizadas por un alumno en concreto y que el resto de la clase pueda participar de ellas. (Martínes Esparza, 2011)

iTalc ha sido diseñado para ser utilizado en escuelas, por lo tanto, ofrece amplias posibilidades para profesores, entre ellas:

- Ver lo que sucede en las computadoras del laboratorio usando el modo de supervisión.
- Controlar remotamente las computadoras para ayudar a otras personas en sus tareas.
- Enseñar una demostración (se muestra la pantalla del profesor a todos los estudiantes en tiempo real).
- Bloquear las estaciones de trabajo para evitar que la atención al profesor se vea afectada.
- Enviar mensajes de texto a estudiantes.
- Encender, apagar y reiniciar computadoras a través del acceso remoto. (iTalc)

## 1.4. Aplicaciones con interconexión con repositorios

### 1.4.1. Moodle

Moodle incluye módulos de interoperabilidad con varios tipos de repositorios, el estudio de todos, así como sus ventajas y desventajas sería extenso en demasía, por lo que solo se analizará la interoperabilidad con el repositorio DSpace.

El módulo de Integración Moodle permite que el repositorio Dspace se convierta en el depósito del material educativo o académico de la institución, extendiendo a los sistemas Moodle las funcionalidades de Búsqueda, Recuperación, Descripción y Depósito en el repositorio. Así, Dspace puede usarse como la herramienta integrada para descubrir, contribuir e importar contenido a los cursos existentes en Moodle. (Arvo Consultores)

En el sistema Moodle se implementan las siguientes funcionalidades para la interoperabilidad con repositorios:

- Búsqueda de contenidos.
- Búsqueda de contenidos en comunidades específicas o evitando comunidades privadas.
- Descarga de contenidos y agregación a Moodle.
- Depósito de contenidos desde Moodle.

En el sistema Dspace, el elemento central de la solución es un interfaz REST (evolución de la existente en DSpace v3 y v4, pues permite depósitos) que posibilita las funciones básicas de un repositorio, búsqueda, descargar y depósito de contenido desde sistemas externos, como Moodle.

Moodle presenta un sistema de comportamiento similar al especificado para el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”. La incapacidad para incorporar estos sistemas a ATcnea, evidencia la necesidad de crear uno propio. Aun así, la investigación y el análisis del funcionamiento de Moodle, permitieron comprender y ejemplificar el resultado final del módulo a desarrollar.

### 1.4.2. Blackboard Academic Suite

Blackboard es un LMS (Learning Management System) que provee a más de 17000 instituciones de un software capaz de agilizar los procesos educativos. La compañía tiene la mayor participación en el mercado norteamericano, al ser utilizada en el 75% de los centros universitarios. (Blackboard Inc, 2017)

Blackboard es el socio de la educación en el cambio, ayudando a los estudiantes, educadores, instituciones y empresas a prosperar en un entorno complejo y cambiante. Su visión se fija en el futuro. Al innovar juntos, ayudan a sus clientes a ver nuevas posibilidades.

La forma en que la gente aprende es dinámica y el panorama educativo está en constante evolución. La misión de Blackboard es asociarse con la comunidad global de la educación para permitir el éxito estudiantil e institucional, aprovechando tecnologías y servicios innovadores.

Con una comprensión inigualable del mundo del alumno, las soluciones de éxito estudiantil más completas y la mayor capacidad de innovación, Blackboard es el socio de la educación en el cambio.

Cada alumno es único y ejerce la elección buscando el mejor valor y utilizando los datos para informar sus decisiones. Sus expectativas están evolucionando a medida que el mundo y las oportunidades de aprendizaje dentro de él continúan cambiando rápidamente.

Debido a que Blackboard apoya a los estudiantes de K-12(educación primaria y secundaria) a través del desarrollo profesional, son capaces de liderar la industria en la investigación de estudiantes. Poseen información inigualable sobre flujos de trabajo comunes, comportamientos estudiantiles y arquetipos de cursos. En sociedad con sus clientes, anticipan y resuelven las necesidades, las expectativas, y la demografía del estudiante.

A través de asociaciones con sus clientes, abrazan nuevas oportunidades con agilidad e impulsan el éxito de los estudiantes durante este tiempo de transformación, aprovechando el conjunto más amplio y profundo de soluciones orientadas al éxito del estudiante en el mercado.

Al igual que muchas industrias antes, la educación está experimentando nuevas dinámicas de mercado. Los estudiantes globales ya no están limitados por los límites geográficos y el avance rápido de la tecnología está abriendo nuevos caminos. Blackboard se mantiene como una solución de avanzada.

Blackboard gestiona recursos de repositorios comerciales como Equella, Intralibary y Hive, e incorpora en la versión 9.1 de Blackboard Learn lanzado en 2013 incluye un repositorio llamado Xplor, propio de la compañía. (López Sandino, 2013)

### 1.4.3. Dialnet

Dialnet tiene su origen en la Biblioteca y el Servicio Informático de la Universidad de La Rioja. El objetivo inicial fue emitir alertas informativas a partir de contenidos de revistas científicas. Desde 1999 se apostó por conseguir crear un sistema que permitiera establecer un servicio de alertas por correo electrónico para los usuarios.

Las primeras Bibliotecas que se incorporan al proyecto, en el 2003, fueron las de las Universidades de Cantabria y Pública de Navarra. Poco después se unirían las de Castilla-La Mancha, Zaragoza, Extremadura, Oviedo y País Vasco. Desde entonces han venido uniéndose otras Bibliotecas Universitarias, tanto del territorio nacional, como del internacional, así como Bibliotecas Públicas y Bibliotecas especializadas. Se trata por lo tanto de una cooperación que no gira en torno a un territorio ni a una temática específica, siendo el principal objetivo el unir esfuerzos para ofrecer una serie de recursos y servicios de calidad para usuarios, bibliotecas, autores y editores de revistas.

Dialnet es actualmente uno de los mayores portales bibliográficos del mundo, cuyo principal cometido es dar mayor **visibilidad a la literatura científica hispana.**

Dialnet es un proyecto de cooperación que integra distintos **recursos y servicios documentales**:

- **Base de datos** de contenidos científicos hispanos. En la actualidad se puede encontrar en Dialnet artículos de revista, libros y artículos de libros colectivos, actas de congresos, tesis doctorales, reseñas de otras publicaciones en Dialnet.
- **Servicio de alertas bibliográficas** que difunde, de una manera actualizada, los contenidos de las revistas científicas hispanas.
- **Hemeroteca virtual** hispana de carácter interdisciplinar, aunque con un predominio de las revistas de Ciencias humanas, jurídicas y sociales.

- Depósito o **repositorio** de acceso a la literatura científica hispana a texto completo, con una clara apuesta por el acceso libre y gratuito a la misma, sumándose al movimiento Open Access.

Dialnet es un sitio web que gratuitamente ofrece servicio de búsqueda en varios formatos de datos digitales y agrupa grandes cantidades de información científica. Dialnet apoya la iniciativa OAI (Open Archives Initiative), y aunque tiene una versión “Plus” que no es gratuita, dispone también de una versión totalmente libre. (Dia17)

### 1.4.4. Americanae

En el VII Encuentro de Centros Españoles de REDIAL (Red Europea de Documentación e Información sobre América Latina, creada en 1989) surgió la propuesta de creación de un portal europeo que reuniera la documentación sobre América Latina existente en diversas instituciones europeas y del resto del mundo. Este portal, llamado **Americanae**, está compuesto por diferentes aplicaciones:

- Recolector de metadatos mediante el protocolo OAI-PMH.
- Directorio de colecciones digitales.
- Repositorio OAI-PMH.

El mecanismo de funcionamiento consiste en recolectar de forma selectiva el contenido americanista de distintos repositorios OAI que estén configurados en sets temáticos con dicho contenido:

- La recopilación del patrimonio cultural americano conservado en instituciones culturales (archivos, bibliotecas y museos).
- La recopilación del patrimonio cultural europeo de temática americanista conservado en instituciones culturales.
- La recopilación de estudios e investigaciones sobre América Latina.
- Aportación de los contenidos a Europeana (sitio web de funcionamiento parecido a Americanae).

El estudio de las bibliotecas digitales evidenció el modo de funcionamiento de los protocolos de interconexión, así como la gran variedad de sitios que los implementan y las diferentes funcionalidades que aportan a sus consumidores. El hecho de que todas sean en formato web imposibilita su inclusión como

componente del software ATcnea, pero aun así visibilizan el correcto funcionamiento y utilización de la interoperabilidad.

## 1.5. Repositorios

### 1.5.1. Dspace

DSpace es un software de código abierto diseñado por el Massachusetts Institute of Technology (MIT) y los laboratorios de HP para gestionar repositorios de ficheros (textuales, audio, vídeo, etc.), facilitando su depósito, organizándolos en comunidades, asignándoles metadatos y permitiendo su difusión a recolectores o agregadores. Estas características han hecho que sea uno de los programas preferidos por las instituciones académicas para gestionar el repositorio dónde los investigadores depositan sus publicaciones y materiales de búsqueda con objeto de darles una mayor visibilidad.

DSpace se una aplicación cliente/servidor que se gestiona vía web, es decir, que la mayor parte de procesos pueden llevarse a cabo con un navegador estándar como Internet Explorer, Firefox u Opera. Desde el punto de vista del servidor, DSpace puede instalarse tanto en entornos Linux como en un servidor Windows. A diferencia de otras aplicaciones basadas en AMP (Apache-MySQL-PHP), la verdad es que la instalación de DSpace no es una tarea sencilla y requiere unos ciertos conocimientos de informática para configurar la base de datos y el entorno de desarrollo.

Por defecto, DSpace está configurado con el esquema de metadatos Dublin Core, pero quizás, según los documentos que queramos depositar, nos será muy útil disponer de otros esquemas para definirlos mejor (PRISM, MODS, METS, etc.). Así, por ejemplo, en el caso de artículos de revista o actas de congresos, puede ser útil registrar los datos bibliográficos (como por ejemplo el volumen, número, páginas, ISSN o DOI) que encontramos en PRISM, pero no en DC. (Rodríguez Gairín, y otros, 2008)

DSpace soporta Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) como proveedor de datos. De este modo los registros están disponibles para que los metadatos asociados a los objetos puedan ser recopilados (*harvesting*) por todo tipo de recolectores. También se ha incorporado en su evolución la interoperabilidad con: (Arvo Consultores)

- **REST-API** Existe una REST API que permite la integración basada en REST (Representational State Transfer). El código se puede descargar a partir de la versión 1.8 de DSpace y previsiblemente está incluido a partir del DSpace 3.0, pero ya existe una serie de proyectos interesantes sobre esta interfaz, incluido una integración Moodle.
- **SWORD** (Simple Web-service Offering Repository Deposit) es un protocolo, basado en atompub, Atom Publish Protocol, que define el depósito remoto de ítems en un repositorio por otras aplicaciones. La disponibilidad de librerías Sword en diversos lenguajes, (PHP, java…) promueve el uso de este tipo de integración.
- **LNI (Lightweight Network Interface)** Esta interfaz permite la integración de un sistema con DSpace a través del protocolo WebDAV. A pesar de estar implementado desde las primeras versiones, estas mostraron un rendimiento pobre y hubo problemas en su utilización.
- **JavaAPI** Cualquier aplicación que sea capaz de llamar al JavaAPI de DSpace, podrá usarse para este tipo de integración. Así es como surgió XMLUI y están surgiendo continuamente nuevos proyectos. La JavaAPI no proporciona una separación completa o nítida y normalmente se requerirá re-escribir parte de la lógica de negocio de DSPACE en la nueva aplicación. Aparte de eso, ese el camino para el uso de Frameworks de Aplicaciones Web o de desarrollo rápido, como el framework: “Play!”, la nueva interface anunciada en agosto de este año para Dspace, Freemarker WebMVC o incluso el uso de frameworks no-Java como Ruby on Rails.

DSpace, con sus múltiples posibilidades de interconexión tienta a todo desarrollador a innovar y a ser la primera opción en cuanto a temas de interoperabilidad. Debido a esto, y la existencia de un repositorio de este tipo en la institución (Universidad de las Ciencias Informáticas) se desarrolla como valor agregado al “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” la conexión a repositorios de tipo DSpace.

### 1.5.2. Rhoda

XAUCE RHODA es un repositorio de objetos de aprendizaje, que pone a disposición de las comunidades académicas una colección de objetos de aprendizaje (OA) bajo la figura de recurso de acceso abierto. Permite el almacenamiento de OA, así como su rápida recuperación y accesibilidad desde cualquier plataforma de gestión de aprendizaje estandarizada. Ofrece mayor organización, acceso y reutilización de

los recursos educativos, cuya calidad se garantiza a través de un sistema de revisión. Promueve la formación personalizada y el trabajo colaborativo y reduce los costos y tiempos por concepto de creación de recursos educativos.

Rhoda ofrece una gama bastante amplia de métodos de interconexión, entre los que se encuentran: (Puentes Daniel, y otros, 2014)

- **Interfaz de Consultas Simples (SQI)** es una Interfaz de Programación de Aplicaciones (API) para resolver las problemáticas relacionadas con las búsquedas de contenidos digitales en entornos heterogéneos, el establecimiento de sesión y la realización de consultas síncronas y asíncronas; definiendo los servicios que un repositorio puede tener disponibles para recibir y responder consultas de otros repositorios. La mejor práctica para publicar este estándar es usando servicios web bajo el protocolo SOAP (Simple Objet Access Protocol).
- **Interfaz de Publicaciones Simples (SPI)** es un protocolo que permite publicar recursos educativos y metadatos en un repositorio. Para ella se define una API formada por conjuntos de métodos independientes de la tecnología, lo que facilita la interoperabilidad semántica de las implementaciones de los mismos. Todos los métodos que ofrece la API requieren la creación de una sesión en el destino donde se va a publicar el recurso educativo.
- **OAI-PMH** es otro de los estándares implementados en RHODA, con la finalidad de recolectar metadatos en otros repositorios para posteriormente realizar búsquedas a partir de ellos. Este protocolo genera y promueve estándares de interoperabilidad que facilitan la difusión, intercambio y accesibilidad a documentos de diferente naturaleza. Igualmente, OAI-PMH permite almacenar en un solo lugar los metadatos y es allí en donde se realizan las diferentes consultas, el protocolo no define la creación de los metadatos, únicamente se ocupa de la gestión de la información. Utiliza transacciones del Protocolo de Transferencia de Hipertexto (HTTP) en la transferencia de información de contenido web, donde está definida la sintaxis y la semántica que utilizan los clientes y servicios para comunicarse entre sí. El protocolo OAI está basado en un modelo cliente/servidor que transmite preguntas y respuestas entre un Proveedor de Datos y un Proveedor de Servicios.

Rhoda presenta gran variedad de métodos para la interoperabilidad, sin embargo, se encuentra actualmente en desuso por la institución (Universidad de las Ciencias Informáticas), por lo que su uso como valor

agregado del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” se descarta.

### 1.5.3. Fedora

Fedora es un robusto, modular, sistema repositorio de código abierto para el manejo y propagación de contenido digital. Esta especialmente orientado a bibliotecas digitales y archivos, para su acceso y preservación. Además, provee acceso especializado a una amplia y compleja colección de materiales históricos y culturales, así como datos científicos.

Fedora tiene una base mundial de usuarios instalada que incluye organizaciones académicas y de legado histórico, universidades, instituciones investigativas, bibliotecas y agencias gubernamentales. El proyecto Fedora es liderado por “Fedora Leadership Group” y se encuentra bajo los estándares de la organización sin fines de lucro DuraSpace que le provee liderazgo e innovación en los proyectos de tecnología de código abierto y en soluciones enfocadas en la durabilidad y persistencia del acceso a información digital. (Fedora, 2015)

Fedora como parte de su estructura (hasta su versión 3.5) soporta dos tipos de accesos: mediante REST o utilizando el protocolo SOAP. Debido a su aparentemente inexistente presencia en el sistema educativo cubano, no se considera incluir a Fedora como valor agregado del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

## 1.6. Metodología de desarrollo, herramientas y lenguajes.

### 1.6.1. Metodología de desarrollo de software (metodologías ágiles)

La metodología de desarrollo de software es el conjunto de procedimientos usados en la confección y documentación de un software. Una metodología está compuesta por etapas, tareas, restricciones, técnicas y herramientas que se deben emplear en su realización. La metodología define los pasos a seguir durante el planeamiento, desarrollo y mantenimiento de un proyecto.

El desarrollo ágil de software comprende a los métodos de Ingeniería de Software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de todos los involucrados en el proyecto. Las metodologías ágiles son un conjunto heterogéneo de métodos con más o menos reglas, principios, recomendaciones, buenas prácticas. Cada uno de estos métodos engloba una filosofía que se puede caracterizar en estos cuatro términos: (Pro17)

- Incremental: Las versiones de software son pequeñas con ciclos de desarrollo rápido.
- Cooperación: Existe una estrecha interacción entre cliente y equipo desarrollador.
- Sencillo: El método en sí es fácil de aprender, modificar y está suficientemente documentado.
- Adaptación: Gran capacidad para reaccionar ante cambios en todo momento.

#### 1.6.1.1. AUP-UCI (Proceso Unificado de Desarrollo para la UCI)

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.” AUP-UCI es una variación del AUP de Scott Ambler, que ha logrado unificar el proceso de desarrollo en la Universidad de las Ciencias Informáticas bajo una misma directiva. (Rodríguez Sánchez, 2015)

AUP-UCI es la metodología usada por el proyecto ATcnea, por lo tanto, será la usada en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales.

##### 1.6.1.1.1. Fases de AUP-UCI

1. Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
2. Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante

el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

3. Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto. (Rodríguez Sánchez, 2015)

#### 1.6.1.1.2. Disciplinas AUP-UCI

AUP define 7 disciplinas (4 ingenieriles y 3 de gestión de proyectos), las disciplinas son:

1. Modelado de negocio: El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes:
    - ❖ Casos de Uso del Negocio (CUN).
    - ❖ Descripción de Proceso de Negocio (DPN).
    - ❖ Modelo Conceptual (MC).

   A partir de las variantes anteriores se condicionan cuatro escenarios para modelar el sistema. El objetivo de esta disciplina es transformar el modelo en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.

2. Requisitos. El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de Usuario (HU) y Descripción de Requisitos por Proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio.

3. Análisis y diseño. En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los

requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.

4. Implementación. En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

5. Pruebas internas. En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.

6. Pruebas de liberación. Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

7. Pruebas de Aceptación. Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. (Rodríguez Sánchez, 2015)

#### 1.6.1.1.3. Roles de AUP-UCI

1. Jefe de proyecto.
2. Planificador.
3. Analista.
3. Desarrollador.
4. Modelador ágil.
5. Administrador de la configuración.
6. Stakeholder.
7. Administrador de calidad.
8. Probador.
9. Arquitecto de software.

10. Administrador de bases de datos.

### 1.6.2. Herramientas

#### 1.6.2.1. Herramienta de diseño de interfaz

Las herramientas de diseño de interfaz facilitan el desarrollo de software permitiendo la realización de aplicaciones amigables para los usuarios de manera sencilla y eficaz.

##### 1.6.2.1.1. JavaFX Scene Builder 2.0

JavaFX Scene Builder es una herramienta de diseño visual que permite a los usuarios diseñar rápidamente interfaces de usuario de aplicaciones JavaFX, sin codificación. Los usuarios pueden arrastrar y soltar componentes de interfaz de usuario en un área de trabajo, modificar sus propiedades, aplicar hojas de estilo y el código FXML del diseño que crean se genera automáticamente en segundo plano. El resultado es un archivo FXML que se puede combinar con un proyecto Java vinculando la interfaz de usuario a la lógica de la aplicación. (Oracle)

JavaFX Scene Builder es la herramienta de diseño de interfaz de usuario usada por el proyecto ATcnea, por lo tanto, será la usada en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

Se selecciona esta herramienta de diseño de interfaz para el desarrollo del módulo, debido a las especificidades propias del software ATcnea.

#### 1.6.2.2. Herramienta de modelado

Las herramientas de modelado permiten crear modelos para el sistema que se va a desarrollar. Estos modelos se utilizan en simulacros del sistema para pruebas, además que funcionan como guía durante el proceso de desarrollo.

##### 1.6.2.2.1. Visual Paradigm 8.0

Visual Paradigm for UML 8.0 es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Visual Paradigm).

Visual Paradigm 8.0 es la herramienta de modelado usada por el proyecto ATcnea, por lo tanto, será la usada en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

#### 1.6.2.3. IDE de desarrollo

##### 1.6.2.3.1. NetBeans 8.0

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe, además, un número importante de módulos para extender el NetBeans IDE, el cual es un producto libre y gratuito sin restricciones de uso.

Este IDE dispone de: soporte para crear interfaces gráficas de forma visual, control de versiones, colaboración entre varias personas y resaltados de sintaxis. Además, sus funcionalidades son ampliables mediante la instalación de packs. (software.com.ar, 2016)

NetBeans 8.0 es el IDE de desarrollo usado por el proyecto ATcnea, por lo tanto, será el usado en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

### 1.6.3. Lenguajes

#### 1.6.3.1. Lenguajes de programación

#### 1.6.3.1.1. XML

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Originalmente fue diseñado para lidiar con los grandes formatos de las publicaciones electrónicas, XML juega además un importante rol en el intercambio de gran variedad de información a través de la Web. Es un lenguaje muy similar a HTML, pero su función principal es describir datos, no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. (W3C, 2016)

XML es uno de los lenguajes usados por la tecnología de software JavaFX que utiliza el proyecto ATcnea, por lo tanto, se incluye en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

#### 1.6.3.1.2. CSS

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

El éxito de CSS procede de los innumerables beneficios que ha reportado a los diseñadores. El primer beneficio son las características enriquecidas. Utilizando un estilo declarativo simple, los diseñadores pueden establecer el posicionamiento, márgenes y alineación, colores, estilo de texto, listas numeradas y mucho más. Así como la dirección de escritura, los tipos de letra y otros aspectos que pueden diferir de un lenguaje a otro. CSS es compatible con un número cada vez mayor de tradiciones de tipografía diferentes y ha supuesto un progreso significativo en la representación de documentos multilingües. (Balo Pereira)

CSS es el lenguaje usada por el proyecto ATcnea para el diseño de interfaces, por lo tanto, será el usada en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

#### 1.6.3.1.3. Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados. (Ictea, 2013)

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

#### 1.6.3.1.4. JavaFX

JavaFX es una tecnología de software que, combinada con Java, permite crear y desplegar aplicaciones con un aspecto vanguardista y contenidos avanzados, audio y vídeo. JavaFX amplía la potencia de Java permitiendo a los desarrolladores utilizar cualquier biblioteca de Java en aplicaciones JavaFX. De esta forma, los desarrolladores pueden ampliar sus capacidades en Java y utilizar la tecnología de presentación que JavaFX proporciona para crear atractivo visual. Como usuario, podrá ejecutar aplicaciones de este tipo en un explorador o arrastrarlos y soltarlos en el escritorio. (Oracle)

Características principales de JavaFX:

- Permite a los desarrolladores integrar gráficos vectoriales, animación, sonido y activos web de vídeo en una aplicación interactiva, completa y atractiva.
- Amplía la tecnología Java permitiendo el uso de cualquier biblioteca de Java en una aplicación JavaFX.
- Permite mantener un eficaz flujo de trabajo entre diseñador y desarrollador en el que los diseñadores pueden trabajar en las herramientas que deseen mientras colaboran con los desarrolladores.

JavaFX es la tecnología de software usada por ATcnea, por lo tanto, será la utilizada en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

#### 1.6.3.2. Lenguaje de modelado

##### 1.6.3.2.1. UML (Lenguaje Unificado de Modelado)

UML (Unified Modeling Language) fue adoptado como estándar del Object Management Group (Grupo Gestor de Objetos) en 1997 debido a que representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas. Es un lenguaje para la especificación, visualización, construcción y documentación de sistemas, no solo de software. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos informáticos, de arquitectura o de cualquier otra rama. (Object Management Group)

Por ser el lenguaje usado por la herramienta de modelado seleccionada y ser el designado por el proyecto ATcnea, constituye UML el lenguaje de modelado escogido para la propuesta de solución.

### 1.6.4. Gestor de dependencias

#### 1.6.4.1. Maven

Maven se encarga de gestionar y construir proyectos Java. Maven hace más que únicamente actuar de "Gestor de dependencias". Algunas de sus funcionalidades son:

- Construye el código del proyecto (build).
- Ejecuta las pruebas (testing).
- Trabaja de gestor de dependencias: importa automáticamente las bibliotecas desde un repositorio remoto.
- Almacena (cachea) estas bibliotecas en un repositorio local para agilizar las importaciones posteriores de las bibliotecas que ya tiene.

### 1.6.5. Base de Datos

#### 1.6.5.1. HSQLDB (*Hyperthreaded Structured Query Language Database*)

HSQLDB (HyperSQL DataBase) es el principal software de base de datos relacional SQL escrito en Java. Ofrece un motor de base de datos pequeño, rápido, multihilo y transaccional, con tablas en memoria y basadas en disco y soporta modos incorporados y de servidor. Incluye una poderosa herramienta de SQL de línea de comandos y una herramienta de interfaz gráfica de usuario de consulta sencilla.

HSQLDB admite la más amplia gama de características de estándar SQL que se ven en cualquier motor de base de datos de código abierto: SQL: 2011. Soporta casi completo Advanced ANSI-92 SQL (formato BNF). También se admiten muchas extensiones del estándar, incluyendo modos de compatibilidad de sintaxis y características de otros motores de base de datos populares. (hsqldb.org, 2017)

#### 1.6.5.2. Hibernate v5.1.0

El desarrollo de software orientado a objetos que se ocupa de los datos de bases de datos relacionales puede ser engorroso y consumir recursos. Los costos de desarrollo son significativamente más altos debido a un desajuste de paradigma entre cómo los datos se representan en los objetos versus las bases de datos relacionales. Hibernate es una solución Object / Relational Mapping (ORM) para entornos Java. ORM se refiere a la técnica de mapeo de datos entre una representación de modelo de objeto a una representación de modelo de datos relacional.

Hibernate no solo se ocupa del mapeo de las clases Java a las tablas de base de datos (y de los tipos de datos Java a los tipos de datos SQL), sino que también proporciona servicios de consulta y recuperación de datos. Se puede reducir significativamente el tiempo de desarrollo que de otra manera se gasta con el manejo manual de datos en SQL y JDBC. El objetivo de diseño de Hibernate es liberar al desarrollador del 95% de las tareas comunes de programación relacionadas con la persistencia de los datos, eliminando la necesidad de un procesamiento de datos manual y hecho a mano utilizando SQL y JDBC. Sin embargo, a diferencia de muchas otras soluciones de persistencia, Hibernate no oculta el poder de SQL y garantiza que su inversión en tecnología y conocimiento relacional sea tan válida como siempre. (Ebersole, y otros, 2016)

### 1.6.6. Herramienta para el control de versión

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. Un sistema de control de versiones (VCS por sus siglas en inglés) permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más. Usar un VCS también significa generalmente que, si fastidias o pierdes archivos, puedes recuperarlos fácilmente.

#### 1.6.6.1. Git

Git permite y anima a tener múltiples sucursales locales que pueden ser totalmente independientes entre sí. La creación, fusión y supresión de estas líneas de desarrollo tardan solo unos segundos.

Git realiza casi todas las operaciones localmente, lo que le da una gran ventaja de velocidad en los sistemas centralizados que constantemente tienen que comunicarse con un servidor en alguna parte. Además, fue construido para trabajar en el kernel de Linux, lo que significa que ha tenido que manejar de forma efectiva grandes repositorios desde el primer día. Git se escribe en C, reduciendo la sobrecarga de tiempos de ejecución asociados con lenguajes de nivel superior. La velocidad y el rendimiento han sido una de las principales metas de diseño del Git desde el principio.

El modelo de datos que utiliza Git garantiza la integridad criptográfica de cada bit de su proyecto. Cada archivo y confirmación se suma y recupera por su comprobación al salir. Es imposible sacar algo de Git aparte de los bits exactos que introduces.

A diferencia de los otros sistemas, Git tiene algo llamado el "área de ensayo" o "índice". Este es un área intermedia donde los commits pueden ser formateados y revisados antes de completarlos.

Una cosa que diferencia a Git de otras herramientas es que es posible colocar rápidamente algunos de sus archivos y confirmarlos sin comprometer todos los otros archivos modificados en su directorio de trabajo o tener que listarlos en la línea de comandos durante el commit.

Git se publica bajo la licencia GNU General Public License versión 2.0, que es una licencia de código abierto. El proyecto Git optó por utilizar GPLv2 para garantizar su libertad de compartir y cambiar software libre para asegurarse de que el software es gratuito para todos sus usuarios. (git)

### 1.7. Conclusiones parciales

A partir de las características propias del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” como parte del software ATcnea y lo expuesto en el capítulo número uno, se concluye que:

1. Con la investigación realizada a los protocolos de interconexión usados en repositorios con el fin de visibilizar los contenidos de estos, se ha demostrado que son cada vez más las variantes que existen en este tipo de interoperabilidad.
2. Se evidenció la utilidad de los protocolos de interoperabilidad en los llamados “proveedores de servicios” y en los LMS (Sistemas de gestión de aprendizaje).
3. A pesar de existir aplicaciones con funcionamiento parecido al deseado en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”, todas son en formato web y ninguna se puede incluir como parte del software ATcnea, así se evidencia la necesidad de desarrollar un módulo que funcione como parte del software ATcnea y permita consumir en el salón de clases recursos almacenados en repositorios sin la necesidad de acceder mediante un navegador a la internet.
4. Se determina utilizar en la interconexión del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” con los repositorios cualquier protocolo de interconexión que el repositorio implemente, brindando mayor prioridad a OAI-PMH ya que es parte de una iniciativa de acceso libre y de código abierto (ver anexo#4 Relación de repositorios y protocolos investigados).
5. El estudio de las metodologías, herramientas y lenguajes permitió definir los componentes base para el desarrollo de la solución, donde se define a AUP-UCI como metodología de desarrollo y como herramienta CASE Visual Paradigm 8.0 para el modelado UML de los artefactos del análisis y diseño de la solución, se elige como IDE de desarrollo IntelliJ IDEA, así como GIT para el control de versiones y Maven como gestor de dependencias. Se definen como lenguajes a usar: UML para el modelado, Java para la implementación, CSS y XML en el diseño de interfaces.

# CAPÍTULO 2. DESCRIPCIÓN DE LA PROPUESTA

## 2.1. Introducción

En el desarrollo de un software se debe comenzar definiendo requisitos y funcionalidades que permitan orientarse hacia un objetivo final. Para aclarar el método a seguir para la realización del software, así como la meta del mismo, en este capítulo se presenta la propuesta de desarrollo del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales". Se identifican los requisitos funcionales y no funcionales. Se generan los artefactos según los requisitos determinados y se modelan los diagramas necesarios.

## 2.2. Descripción de la propuesta de solución

El "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales" contará con:

- Estructura definida para la adición de repositorios.
- Componente de configuración.
- Componente de búsqueda.

### 2.2.1. Estructura para la adición de nuevos repositorios

A través del uso de interfaces, el "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales" permite incluir nuevos repositorios de forma estandarizada y simple. Solo se deben identificar los protocolos definidos por ellos y desarrollar el modo de comunicación para agregar cualquier repositorio.

La clase "RepositoryPicker" es a través la cual se accede a las funcionalidades del módulo desde cualquier submódulo de ATcnea. Al realizar la llamada a esta clase principal, se debe especificar:

- ReturnType (forma en que se accederá a los recursos en repositorios, referencial o local).
- MimeTypes (extensión de los archivos que se admitirán: ".pdf", ".mp3").
- MimeTypesGroup (tipo de archivos que se admitirán, ejemplo: audio, video).

En la raíz del módulo(repository/src/main) se encuentra el paquete “repositories”, es allí donde se definen los repositorios para su posterior uso como parte del software ATcnea. Cada repositorio deberá tener un paquete con su nombre (ejemplo: Rhoda, DSpace), dentro de este, se encuentra una carpeta “resources” con las vistas propias del repositorio:

- “<nombre_del_repositorioInstanceForm.fxml>” (vista que contiene el formulario con los datos requeridos para agregar una instancia de repositorio. Ejemplo: DSpaceIntanceForm.fxml).
- “<nombre_del_repositorioSearchView.fxml>” (vista que contiene los filtros y criterios admitidos para realizar la búsqueda. Ejemplo: DSpaceSearchView.fxml).

Dentro del paquete también está la carpeta “icon” que contiene el ícono con que se identifica a dicho repositorio, la imagen deberá estar en formato “png” y deberá tener como nombre el mismo otorgado a la carpeta del repositorio en minúsculas (ejemplo: dspace.png).

Cada repositorio deberá tener un archivo de nombre “<nombre_del_repositorioController.java>” (ejemplo: DSpaceController.java) en la raíz, esta clase será la controladora general de las acciones sobre este repositorio al implementar la interfaz “RepositoryController.java”.

La interfaz “RepositoryController.java” define 4 métodos que todo repositorio a incluir debe implementar:

- getInstanceForm (devuelve el formulario para crear instancias de repositorio específicos para cada uno).
- getInstanceContent (devuelve lo que se muestra en el componente de búsqueda, los criterios y filtros implementados para cada tipo de repositorio).
- getMimeType (devuelve los tipos de archivo que este repositorio contiene, en caso de contener todos los tipos de archivos, o de no conocer específicamente, se devuelve “null”).
- getMimeTypeGroup (devuelve los grupos de archivos que contiene el repositorio, ejemplo: audio, video).

Además de las funciones propias del repositorio también se debe especificar la forma en la que interactúa el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” con el resto de los módulos de ATcnea. Esta interacción será implementada en el método “load” que presentan los módulos

de ATcnea que tienen entre sus funcionalidades el uso de archivos. Un ejemplo de esta implementación se encuentra el anexo#2 de este documento: “Implementación de la interacción del Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales y el Módulo presentación”.

Por último, para que el nuevo repositorio sea visible se debe agregar a la base de datos, a la tabla “TB_REPOSITORY”. La adición del repositorio a la base de datos se debe hacer en el archivo “load.sql”, incluyendo una sentencia de tipo INSERT, ejemplo de añadir el repositorio DSpace:

```
/*MODULE REPOSITORY*/
INSERT INTO TB_REPOSITORY(created_at,updated_at,package_name) VALUES (now(),now(),'dspace')
```

*Ilustración 1 Añadir repositorio DSpace a la base de datos*

### 2.2.2. Componente de configuración

El componente de **configuración** estará disponible en los ajustes de ATcnea junto al resto de las opciones de configuración del software. Una vez se acceda a este componente se podrán hacer cambios en el módulo, algunas de las funcionalidades de dicho componente son:

- Agregar nueva instancia (repositorio alojado en un servidor comúnmente pertenecientes a organismos) de repositorio.
- Eliminar instancia de repositorio.
- Modificar instancia de repositorio.

Estas funcionalidades son explicadas en detalle en las historias de usuarios incluidas en el Anexo#1 de este documento.

### 2.2.3. Componente de búsqueda

El componente de **búsqueda** está disponible para los módulos de ATcnea que consuman recursos. Este componente permitirá mediante la conexión con repositorios digitales disponer de recursos al instante a través de una búsqueda (básica o avanzada), los parámetros permitidos por esta búsqueda serán predefinidos en el desarrollo del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios

digitales", y limitados por el protocolo de interconexión que se utilice en la comunicación. En este componente solo se mostrarán instancias de repositorios previamente añadidas por los usuarios en el componente de **configuración**. Algunas de las funcionalidades de dicho componente son:

- Buscar recursos en instancias de repositorios.
- Buscar recursos localmente.
- Visualizar recursos.

Estas funcionalidades son explicadas en detalle en las historias de usuarios incluidas en el Anexo#1 de este documento.

### 2.3. Modelo del dominio

Un modelo del dominio, también llamado modelo conceptual, es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Es importante resaltar que un modelo del dominio no representa componentes de software. No se trata de un conjunto de diagramas que describen clases de software, u objetos software con responsabilidades (LARMAN, 2004).

#### 2.3.1. Diagrama de clases del modelo de dominio

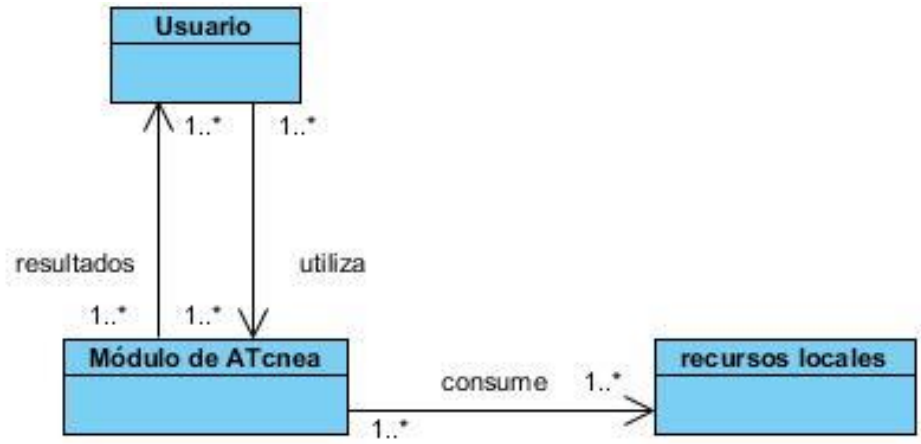



*Ilustración 2 Diagrama de clases del modelo del dominio*

#### 2.3.1.1. Descripción de Clases del Modelo del Dominio

| Concepto | Descripción |
|---|---|
| Usuario | Entidad que solicita los recursos a través de la interacción con los módulos de ATcnea, no todos los módulos consumen recursos, por lo que las interacciones estarán restringidas a módulos que sí contemplen en sus funcionalidades abrir archivos, compartir u otras acciones similares. El usuario deberá estar autenticado como profesor. |
| Módulo de ATcnea | ATcnea gestiona las clases a través de módulos, algunos de estos consumen a su vez recursos, permitiendo usarlos en el salón de clases. |
| Recursos locales | Recursos que se encuentran disponibles en el ordenador del profesor, gestionados por los módulos de ATcnea. |

*Tabla 1 Descripción de clases del modelo del dominio*

### 2.4. Especificación de requisitos de software

#### 2.4.1. Requisitos funcionales

Los requerimientos funcionales de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones. (PMOinformatica, 2017)

| No | Nombre | Descripción |
|---|---|---|
| **1** | Listar repositorios. | El módulo debe permitir a los usuarios visualizar una lista de repositorios disponibles. |
| **2** | Añadir instancia de repositorio. | El módulo debe permitir añadir instancias de repositorios nuevas. Para añadir una nueva instancia de repositorio el usuario debe seleccionar previamente el tipo de repositorio al cual va a pertenecer la instancia. |
| **3** | Buscar instancia. | El módulo debe permitir a los usuarios buscar una instancia de la lista, entrando el siguiente dato como criterio de búsqueda:<br>- Nombre de la instancia |

| | | |
|---|---|---|
| **4** | Buscar repositorio | El módulo debe permitir buscar un repositorio de la lista de repositorios previamente agregados entrando el siguiente dato como criterio de búsqueda:<br>- Nombre del repositorio |
| **5** | Buscar recursos local | El módulo debe permitir buscar recursos localmente en una dirección específica. Se podrá entrar directamente en un campo de texto el URL donde se encuentra el recurso. |
| **6** | Buscar recursos en una instancia usando el protocolo OAI-PMH | El módulo debe permitir realizar una búsqueda en una instancia previamente añadida y cuyo modo de conexión se haya definido mediante OAI-PMH, el criterio de búsqueda permitido será:<br>- Título o palabras claves |
| **7** | Buscar recursos en una instancia usando el protocolo OAI-PMH de manera avanzada | El módulo debe permitir realizar una búsqueda en una instancia previamente añadida y cuyo modo de conexión se haya definido mediante OAI-PMH de manera avanzada. Se podrán elegir varios criterios de búsqueda:<br>- Título y palabras claves<br>- Autores<br>- Fecha<br>- Archivo |
| **8** | Buscar recursos en una instancia usando REST | El módulo debe permitir realizar una búsqueda en una instancia previamente añadida y cuyo modo de conexión se haya definido mediante REST. El criterio de búsqueda será:<br>- Nombre del archivo |
| **9** | Buscar recursos en una instancia usando REST de manera avanzada | El módulo debe permitir realizar una búsqueda en una instancia previamente añadida y cuyo modo de conexión se haya definido mediante REST y de manera avanzada. Los criterios de búsqueda serás:<br>- Nombre del archivo<br>- Tipo de archivo<br>- Descripción |
| **10** | Configurar repositorio | El módulo debe permitir su configuración. Las opciones que se soportarán serán:<br>- Añadir nueva instancia<br>- Eliminar instancia<br>- Editar instancia<br>- Número máximo de resultados por páginas |
| **11** | Editar instancia de repositorio | El módulo debe permitir modificar las características de una instancia previamente añadida. Para modificar |

| | | |
|---|---|---|
| | | la instancia el usuario debe seleccionarla previamente. |
| **12** | Eliminar Instancia de repositorio | El módulo debe permitir eliminar una instancia de repositorio de la lista de disponibles. |
| **13** | Listar recursos vista detalles | El módulo debe permitir visualizar los recursos buscados en las instancias en forma de lista detallada, mostrando la información de cada recurso resultante. De cada recurso se mostrará información como: fecha de creación, última modificación, tamaño y autores. |
| **14** | Listar recursos vista miniatura | El módulo debe permitir visualizar los recursos buscados en las instancias en forma de miniaturas, mostrando el nombre de cada recurso acompañado de un ícono que lo identifique. |
| **15** | Mostrar listado de instancias | El módulo debe permitir visualizar las instancias añadidas previamente. |
| **16** | Probar conexión | El módulo debe permitir probar la conexión con una instancia antes de añadirla a la lista de disponibles. |
| **17** | Seleccionar recurso | El módulo debe permitir usar los recursos resultantes de búsquedas como parte de los otros módulos de ATcnea |

*Tabla 2 Requisitos funcionales del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales"*

### 2.4.2. Requisitos no funcionales

#### 2.4.2.1. Requerimientos de usabilidad

1) El sistema debe cumplir las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad.
2) El sistema debe mostrar mensajes asociados a los fallos de conexión con repositorios del sistema.

#### 2.4.2.2. Requerimientos de software

3) Permitir la instalación de la herramienta en PC que cuenten con Nova 2015 o superior.

#### 2.4.2.3. Requisitos de hardware

4) Se recomienda que tenga 4GB RAM o superior, CPU Core 2 E6300 o superior, 1GB Tarjeta de video o superior. Red cableada de 10Mbytes/100Mbytes/1000Mbytes compartida o conmutada o inalámbrica 802.11b/g/n.

## 2.5. Historias de usuario

| | | |
|---|---|---|
| **Número:** 1 | **Nombre del requisito:** Listar repositorios | |
| **Programador:** Jorge Eduardo Pérez Batista | | **Iteración Asignada:** 1 |
| **Prioridad:** alta | | **Tiempo Estimado:** 12 horas |
| **Riesgo en Desarrollo:** Falta de experiencia con el desarrollo de las herramientas y tecnologías. | | **Tiempo Real:** 18 horas |

**Descripción:**

1- Objetivo:

Permitir al profesor visualizar un listado con los repositorios disponibles.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para visualizar el listado de repositorios:

- El usuario debe estar autenticado.
- El usuario debe acceder al componente de configuración del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”
- Debe haber sido añadido con anterioridad los repositorios a mostrar por un desarrollador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

4- Flujo de la acción a realizar:

El sistema debe permitir al profesor desde su ordenador visualizar un listado de los repositorios previamente añadidos al módulo. Para acceder al listado de repositorios el profesor debe acceder a su menú de configuración, en la pantalla principal del software, una vez dentro del

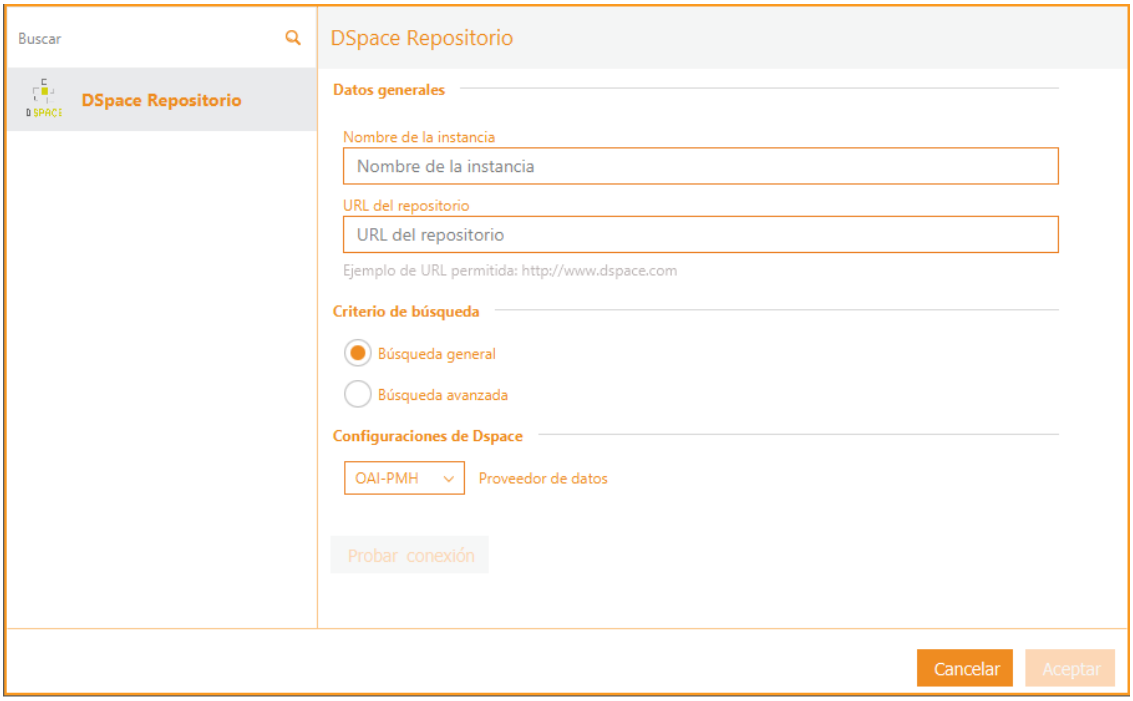
panel de configuración, deberá acceder al submenú Repositorios y “añadir nueva instancia”. El listado debe mostrar los siguientes datos de los repositorios:

- Ícono que lo identifica
- Nombre

**Observaciones:**

Esta acción puede ser realizada solo por profesores autenticados.

**Prototipo de interfaz:**




*HU 1 Listar repositorios*

## 2.6. Arquitectura del sistema

En el desarrollo del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” se ha utilizado el marco de trabajo JavaFX el cual usa en su implementación el patrón arquitectónico modelo-vista-controlador. Este patrón arquitectónico divide la lógica de la solución en tres componentes fundamentales: (Hernandez, 2015)

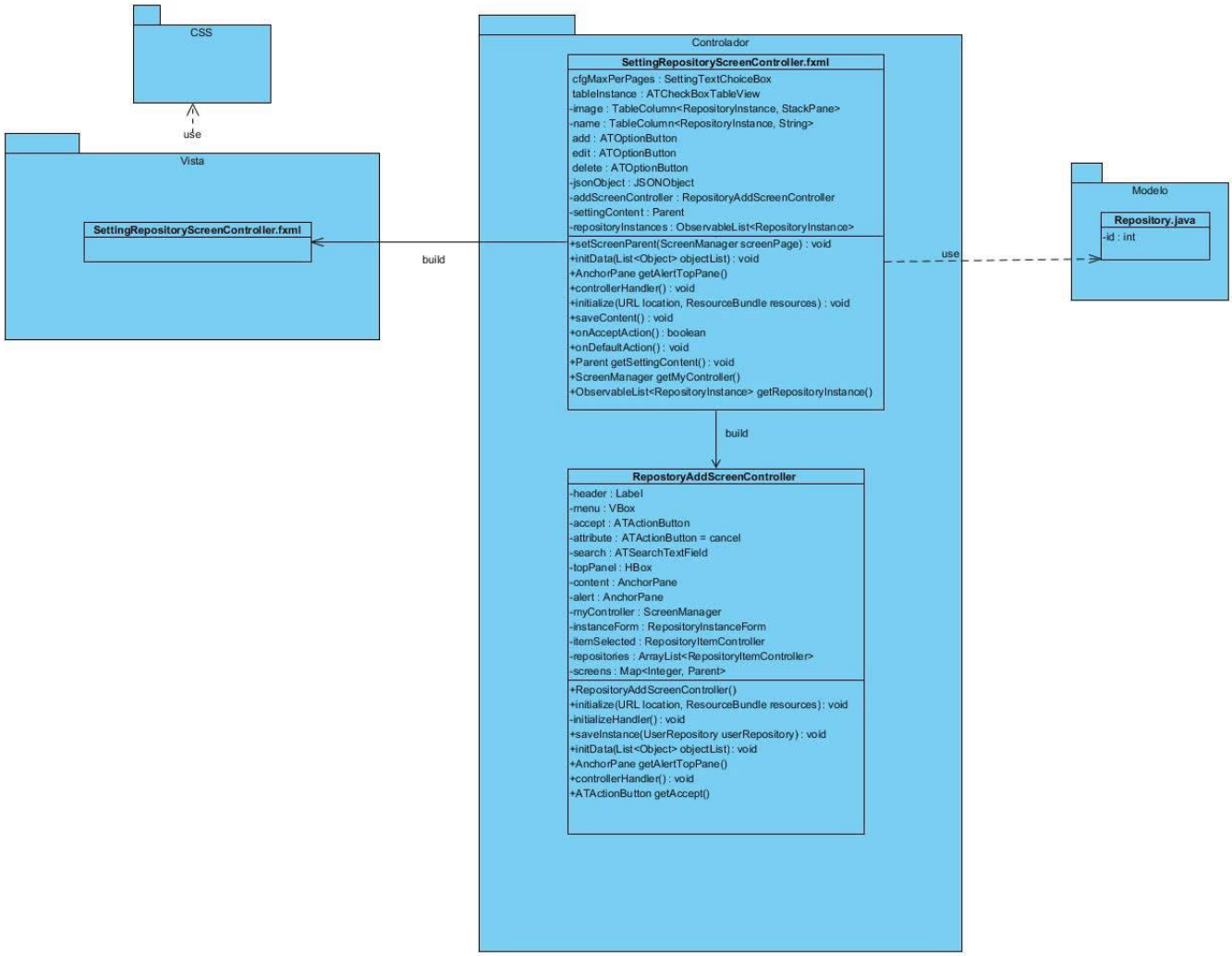
1. Modelo: Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.

2. Controlador: Recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.
3. Vista: Es la representación visual de los datos, contiene todo lo que tenga que ver con la interfaz gráfica. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.

Algunas ventajas de modelo-vista-controlador son: (García, y otros)

- Facilita la agregación de múltiples representaciones de los mismos datos.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento de errores.
- Alta escalabilidad pues se puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware.

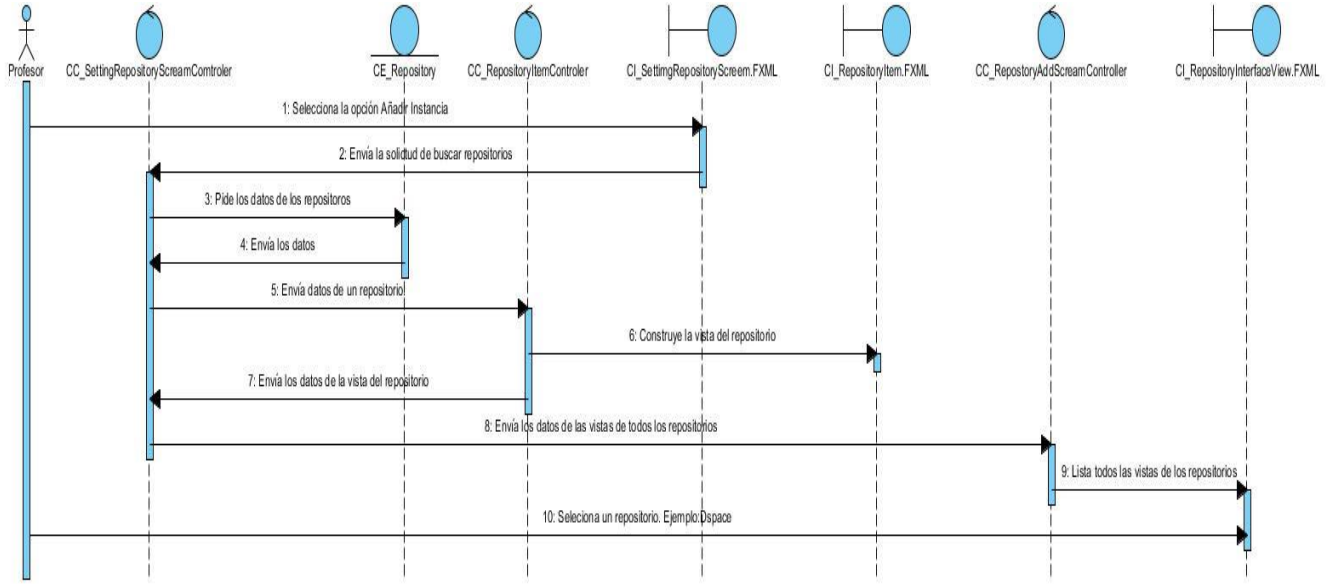
## 2.7. Diagrama de clase de diseño




*DCD 1 Listar repositorio*

## 2.8. Diagrama de secuencia del sistema

Un diagrama de secuencia del sistema (DSS) es un dibujo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas. Todos los sistemas se tratan como cajas negras; los diagramas destacan los eventos que cruzan los límites del sistema desde los actores a los sistemas. (Larman, 2003)




*DS 1 Listar repositorios*

## 2.9. Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que se pueden encontrar a la hora de desarrollar una aplicación. Aunque nuestra aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos, operaciones entre sistemas etc. En lugar de reinventar la rueda, podemos solucionar problemas utilizando algún patrón, ya que son soluciones probadas y documentadas por multitud de programadores. (2014)

**Singlenton:** El singleton es un tipo de patrón de diseño presente en Objective-C o Java, por ejemplo, basado en la mecánica propia de la orientación a objetos. Lo que hace, básicamente, es mantener un objeto de una clase vivo durante todo el transcurso del ciclo de vida de la ejecución desde la primera que se instancia la misma. El propósito es tener métodos de uso general y, sobre todo, tener propiedades persistentes más allá de la vida de la clase donde estamos instanciando un objeto. (Fernández, 2016)

```
    //Actualizar el header de la vista
    if (header != null) {
        header.setText(ATcneaSinglenton.getInstance().getI18N().getString("repository." + packageName + ".name"));
    }

    //Trabajo con el estilo del item
    if (itemSelected != null) {
        itemSelected.clearSelected();
    }
    itemSelected = this;
} catch (Exception e) {
```

*Ilustración 3 Ejemplo de uso del patrón singleton, para mantener el encabezado de la vista actual igual a la anterior.*

Los patrones **GRASP (patrones generales de software para asignar responsabilidades)** describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

**Controlador:** El patrón controlador es intermediario entre una interfaz y el algoritmo que la implementa, de forma que recibe los datos del usuario y es a la vez quien los envía a las distintas clases. Este patrón trata la lógica de negocio separada de la capa de presentación, lo que aumenta la reutilización de código y permite a la vez tener un mayor control. (LARMAN, 2004)

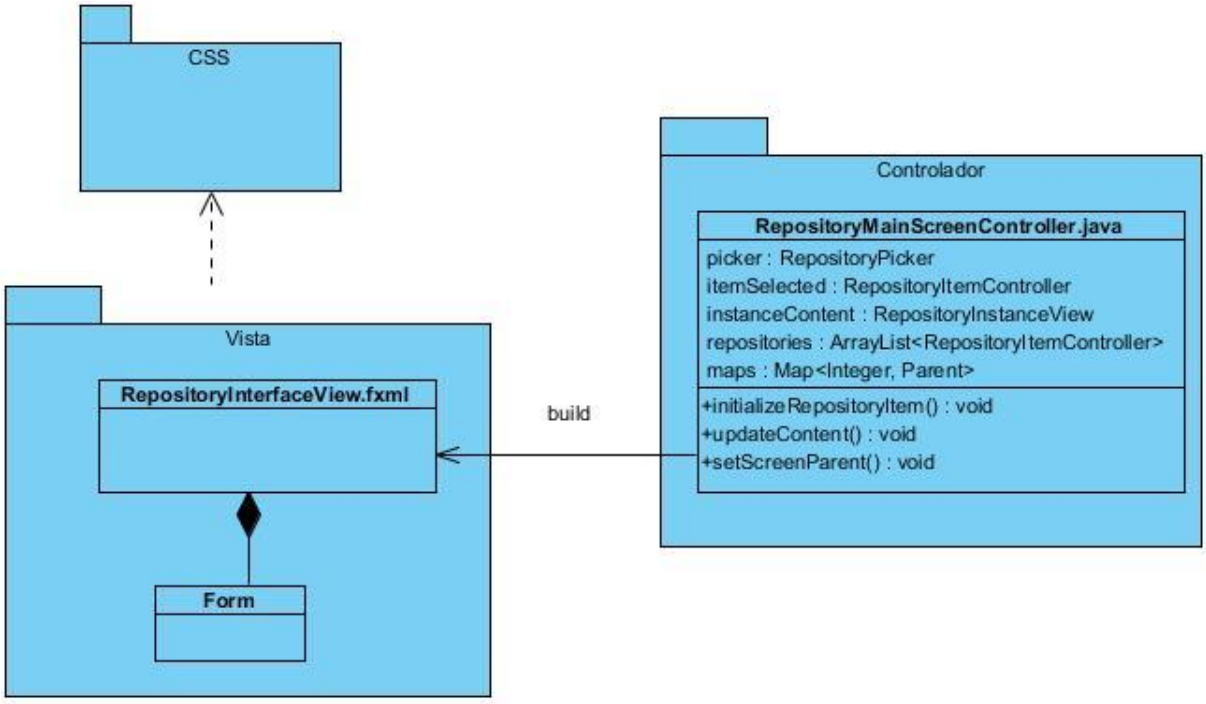



*Ilustración 4 Uso del patrón Controlador en la clase RepositoryMainScreenController.java al manejar la vista RepositoryInterfaceView*

**Alta cohesión:** En el diseño orientado a objetos, la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento (clase o subsistema). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo (Larman, 2004).

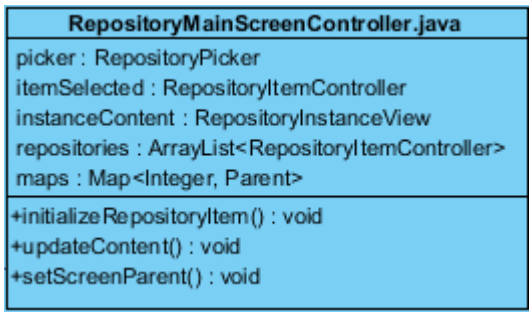



*Ilustración 5 la clase RepositoryMainScreenController solo tiene asignadas responsabilidades con la vista que controla y los items de repositorios cuya gestion se incluyen en esta.*

**Bajo Acoplamiento:** Cuando se habla de acoplamiento entre objetos, se hace referencia a la "fuerza" con la que ciertos objetos están relacionados, o dependen unos de otros. Mientras más dependencias tenga un objeto de otros para llevar a cabo sus tareas, más fuerte será el acoplamiento. Cuando una clase de objetos puede realizar sus tareas, sin depender de ninguna otra clase de objetos (o de un número muy reducido de ellas) se dice que hay bajo acoplamiento. El grado de acoplamiento es importante debido a que está relacionado íntimamente con la reutilización de clases de objetos. Una clase que depende de muchas otras es menos reutilizable (en el sentido de que, si se la quiere reutilizar, hay que reutilizar también todas las clases de las cuales depende). (Larman, 2003)

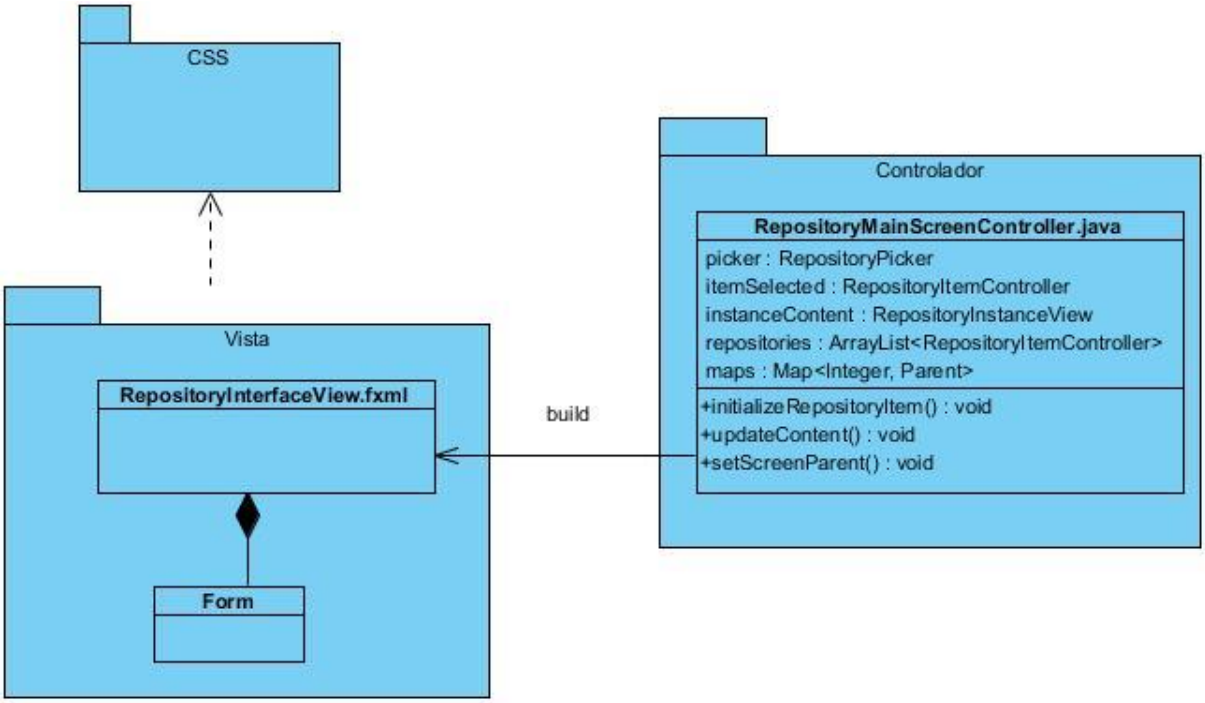



*Ilustración 6 La clase RepositoryMainScreenController se relaciona con la clase RepositoryInterfaceView, la que a su vez se relaciona con la clase Form.*

**Creador:** La creación de instancias es una de las actividades más comunes en la programación orientada a objetos. En consecuencia, es útil contar con un principio general para la asignación de las

responsabilidades de creación. Si se asignan bien, el diseño puede soportar bajo acoplamiento, mayor claridad, encapsulación y reutilización. (Larman, 2003)

```
//Eliminar el comportamiento de deshabilitado de otras pestañas
accept.disableProperty().unbind();
accept.setDisable(false);

//Obtiene la clase controladora del submodulo
String className = StringUtils.capitalize(packageName) + "Controller";
RepositoryController repositoryController = (RepositoryController) UtilRepository.instantiate(new ArrayList<>(), "main.repositories
//Crea el formulario para crear las instancias
RepositoryScreen repositoryScreen=repositoryController.getInstanceForm(null,new ScreenManager(RepositoryAddScreenController.this));
//Obtener el controlador de la vista
instanceForm=(RepositoryInstanceForm) repositoryScreen.getController();
//Obtener la vista
Parent parent= repositoryScreen.getParent();
```

*Ilustración 7 Uso del patrón creador por la clase RepositoryAddScreenController al crear un objeto de tipo RepositoryController.*

**Experto**: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras se considere los mismos aspectos del sistema: (Larman, 1999)

- Lógica de negocio.
- Persistencia a la base de datos.
- Interfaz de usuario.

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    i18n = resources;
    screens = new HashMap<>();
    //Adiciona los item al listado de repositorios
    menu.getChildren().clear();
    for (RepositoryItemController item : repositories) {
        Parent screen = ATcneaSinglenton.getInstance().getScreenManager().bindingFXML(item,
        screens.put(item.getId(), screen);
        menu.getChildren().add(screen);
    }
```

*Ilustración 8 La clase RepositoryAddScreenController le asigna a la clase ATcneaSinglenton la responsabilidad de crear parte de la pantalla a mostrar.*

## 2.10. Modelo de datos

Un modelo de datos está orientado a representar los elementos que va a procesar el sistema, la composición y atributos de los mismos. (Universidad de las Americas Puebla)

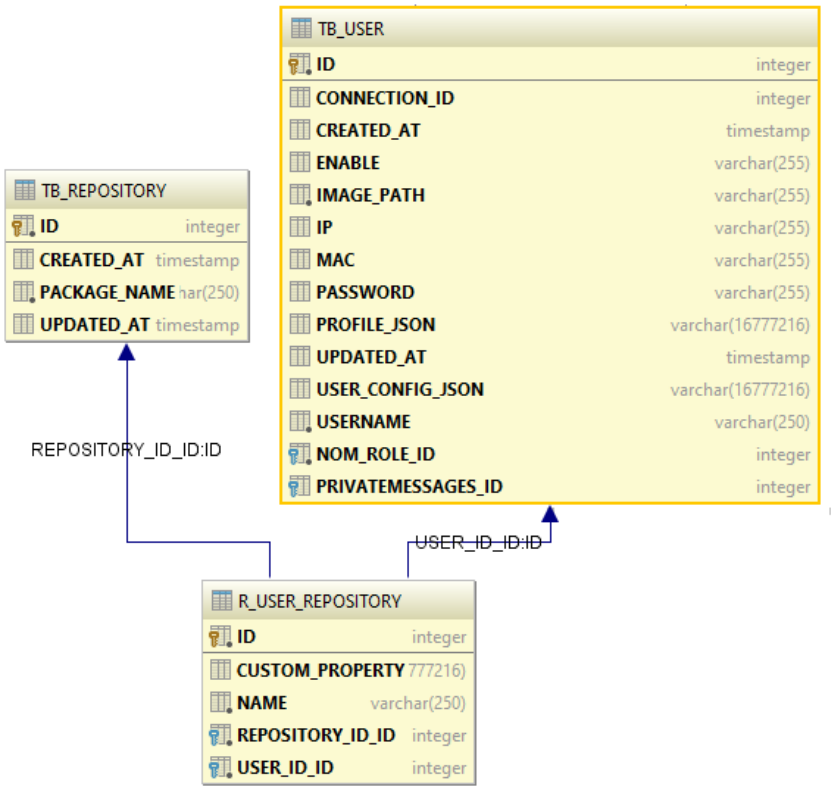



*Ilustración 9 Modelo de datos usado por el "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales"*

## 2.11. Conclusiones parciales

Durante la realización del capítulo número 2 se arribaron a las siguientes conclusiones:

1. A partir de la especificación de los requisitos (funcionales y no funcionales) del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” se obtiene la información necesaria para la realización de las historias de usuario.
2. Al definir la arquitectura y los patrones de diseño se organiza el desarrollo y se facilita la comprensión del funcionamiento del módulo para futuras modificaciones en el código fuente.

3. La descripción de la propuesta muestra el funcionamiento del módulo y describe la estructura para la futura inclusión de repositorios de cualquier tipo como parte del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.
4. Los patrones de diseño de software analizados ofrecieron una estandarización para el desarrollo del módulo.
5. El modelo de datos evidenció los cambios realizados en la base de datos del software ATcnea, agregando la tabla “TD_REPOSITORY” y la relación entre esta y la tabla “TB_USER”.

# CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

El capítulo actual presenta los componentes y estándares de codificación usados en la implementación del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales" y se describe el proceso de validación de la solución implementada, mediante la utilización de los casos de pruebas.

## 3.1. Diagrama de componentes

Un componente representa una parte del software reutilizable que provee alguna funcionalidad significativa. Cada clase en el sistema debe persistir como un componente independiente o en el más alto nivel del mismo. Un componente puede además contener otros componentes.

Los componentes son un tipo estructurado de clasificadores, cuyas colaboraciones y estructura interna pueden ser mostrados en un diagrama de componentes. Un componente, colaborando con otros a través de interfaces bien definidas para proveer funcionalidades a un sistema, puede tener componentes que colaboren para incluir sus propias funcionalidades. Estos componentes pueden ser usados para jerárquicamente descomponer un sistema y representar su arquitectura lógica. (Booch, y otros, 2007)

Existen tres formas de representar el diagrama de componentes: (Pressman, 2007)

- Convencional: Un componente es un elemento funcional de un programa que incorpora la lógica del procesamiento, las estructuras internas de los datos necesarios para implementar dicha lógica, y una interfaz que permita la invocación del componente y el paso de los datos.
- Orientado a Objetos**:** Un componente contiene un conjunto de clases que colaboran entre sí. Cada clase se ha elaborado completamente para incluir todos los atributos y operaciones relevantes de su implementación. También deben de definirse todas las interfaces (mensajes) que permiten que las clases se comuniquen y colaboren con otras clases del diseño.
- Proceso: Construir software con componentes existentes. Existen catálogos de componentes, a medida que se desarrolla la arquitectura del SW se eligen del catálogo los componentes o patrones de diseño y se usan para poblar la arquitectura.

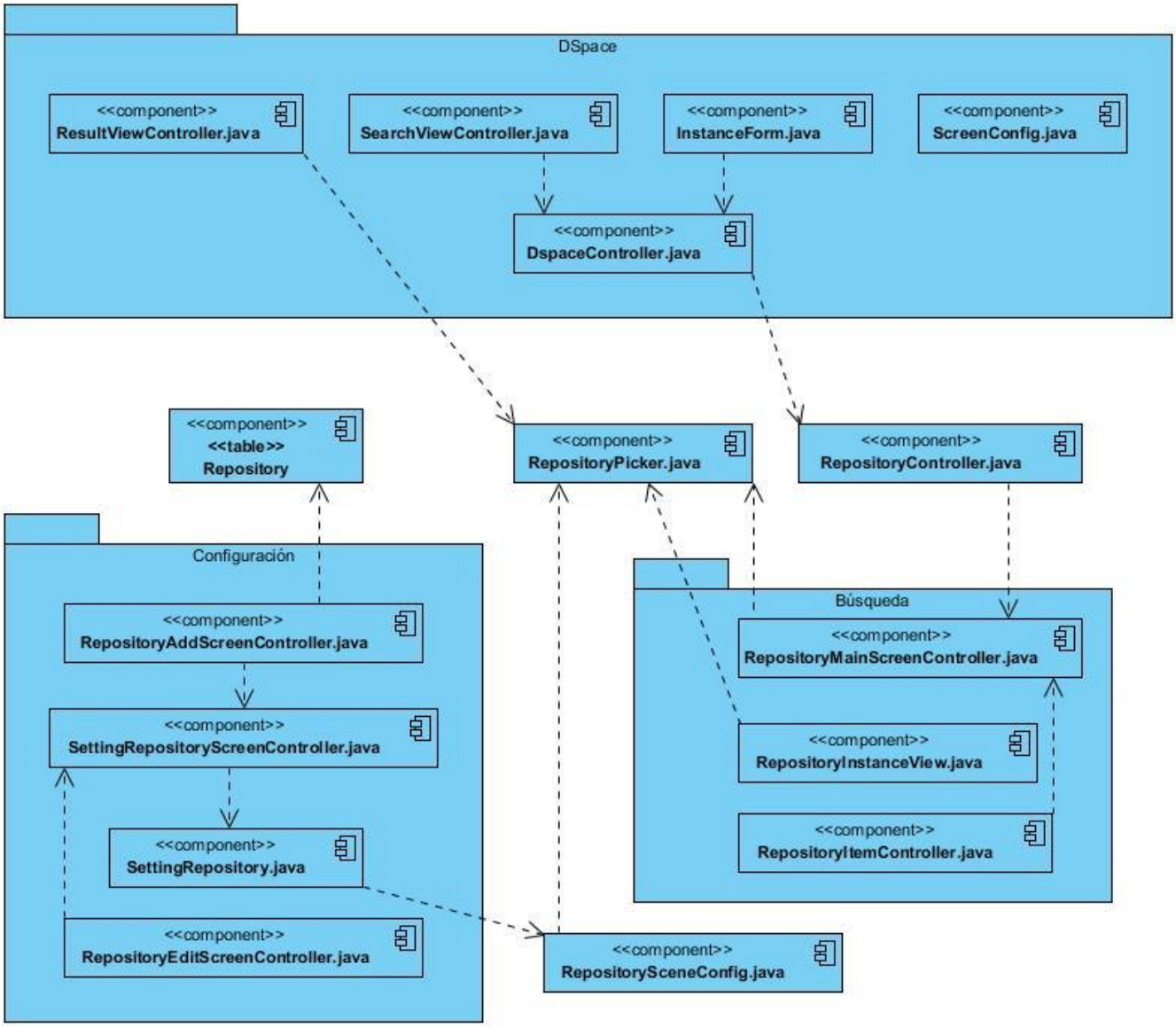



*Ilustración 10 Diagrama de componentes orientado a objetos del "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales"*

## 3.2. Estándares de codificación

### 3.2.1. Sentencia de paquete

La primera línea no comentada de un fichero fuente debe ser la sentencia de paquete, que indica el paquete al que pertenecen las clases incluidas en el fichero fuente. Por ejemplo: (AMAP, 2014)

package main;

### 3.2.2. Sentencias de importación

Tras la declaración del paquete se incluirán las sentencias de importación de los paquetes necesarios. Esta importación de paquetes obligatorios seguirá el siguiente orden: (AMAP, 2014)

- Paquetes del JDK de java.
- Paquetes de utilidades no pertenecientes al JDK de Java, de frameworks de desarrollo o de proyectos opensource tales como apache, hibernate, springframework, etc.
- Paquetes de la aplicación.

### 3.2.3. Comentarios de implementación

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma. Se pueden realizar comentarios de tres tipos: (AMAP, 2014)

- En bloque (describen ficheros, algoritmos, estructuras de datos, clases y bloques).
- De línea (comentarios cortos, ubicados en una sola línea, que describen el fragmento de código del cual están a continuación).
- A final de línea (comentario al final de una sentencia de código y en la misma línea).

### 3.2.4. Declaraciones de clases e interfaces

La siguiente lista describe las partes de la declaración de una clase o interface, en el orden en que deberían aparecer: (Simón Ramírez, 2016)

- Comentario de documentación de la clase o interface (/**...*/).
- Sentencia class o interface.

- Comentario de implementación de la clase o interface si fuera necesario (/*...*/): Este comentario debe contener cualquier información aplicable a toda la clase o interface que no era apropiada para estar en los comentarios de documentación de la clase o interface.
- Variables de clase (static): Primero las variables de clase public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
- Variables de instancia: Primero las public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
- Constructores.
- Métodos: Estos métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.

### 3.2.5. Colocación de declaraciones

Poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}"). No esperar al primer uso para declararlas; puede confundir a programadores no preavisados y limitar la portabilidad del código dentro de su ámbito de visibilidad. (Simón Ramírez, 2016)

```
void myMethod() {
int int1 = 0; // comienzo del bloque del método
if (condition) {
int int2 = 0; // comienzo del bloque del "if"
... }
}
```

La excepción de la regla son los índices de bucles for, que en Java se pueden declarar en la sentencia for: for (int i = 0; i < maximoVueltas; i++) { ... }

Evitar las declaraciones locales que ocultan declaraciones de niveles superiores. por ejemplo, no declarar la misma variable en un bloque interno:

```
int cuenta;
...
miMetodo() {
if (condición) {
int cuenta = 0; // EVITAR!
... }
... }
```

### 3.3. Pruebas

Las pruebas de software (Software Testing) comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo.

Los sistemas informáticos, programas y aplicaciones han crecido a niveles inimaginables en complejidad e interoperabilidad, con lo cual también se han incrementado las posibilidades de defectos (bugs), a imple vista insignificantes, pero que pudieran adquirir proporciones catastróficas. (PMOinformatica)

#### 3.3.1. Diseño de casos de pruebas

Los casos de prueba (CP) describen el comportamiento esperado por el software en escenarios específicos. A partir de la elaboración de los CP se comprueba el correcto funcionamiento del software acorde a los requisitos especificados en etapas anteriores del desarrollo del mismo. A continuación, se presenta el CP propuesto para la historia de usuario “Listar repositorios”.

| **Descripción general:** Permitir al profesor visualizar un listado con los repositorios disponibles. |
|---|
| **Condiciones de ejecución:**<br>- El usuario debe estar autenticado.<br>- El usuario debe acceder al componente de configuración del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”. |

| | | | |
|---|---|---|---|
| - Debe haber sido añadido con anterioridad los repositorios a mostrar por un desarrollador.<br>- El usuario debe seleccionar la opción “añadir instancia” del componente configuración del módulo. | | | |
| **Escenario:** Configuración del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo Central |
| **EC 1.1** Visualizar repositorios | La vista “añadir instancia” muestra los repositorios cuya interacción con el software ATcnea se haya implementado previamente. | El sistema muestra una lista con repositorios previamente añadidos por el desarrollador. | 1. Seleccionar “ajustes” en la pantalla principal del software ATcnea.<br>2. Seleccionar “repositorio” en los ajustes del sistema.<br>3. Añadir instancia en la vista de configuración de repositorio instancia. |

*CP 1 Listar repositorios*

### 3.3.2. Pruebas funcionales

Se denominan pruebas funcionales a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción. (Oré B)

#### 3.3.2.1. Resultados de las pruebas funcionales

Para garantizar que el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” funcione correctamente se realizaron tres iteraciones donde se encontraron un total de 18 no conformidades, 11 en la primera, 7 en la segunda y ninguna en la tercera. Al concluir las iteraciones se corrigieron las no conformidades detectadas en su totalidad.

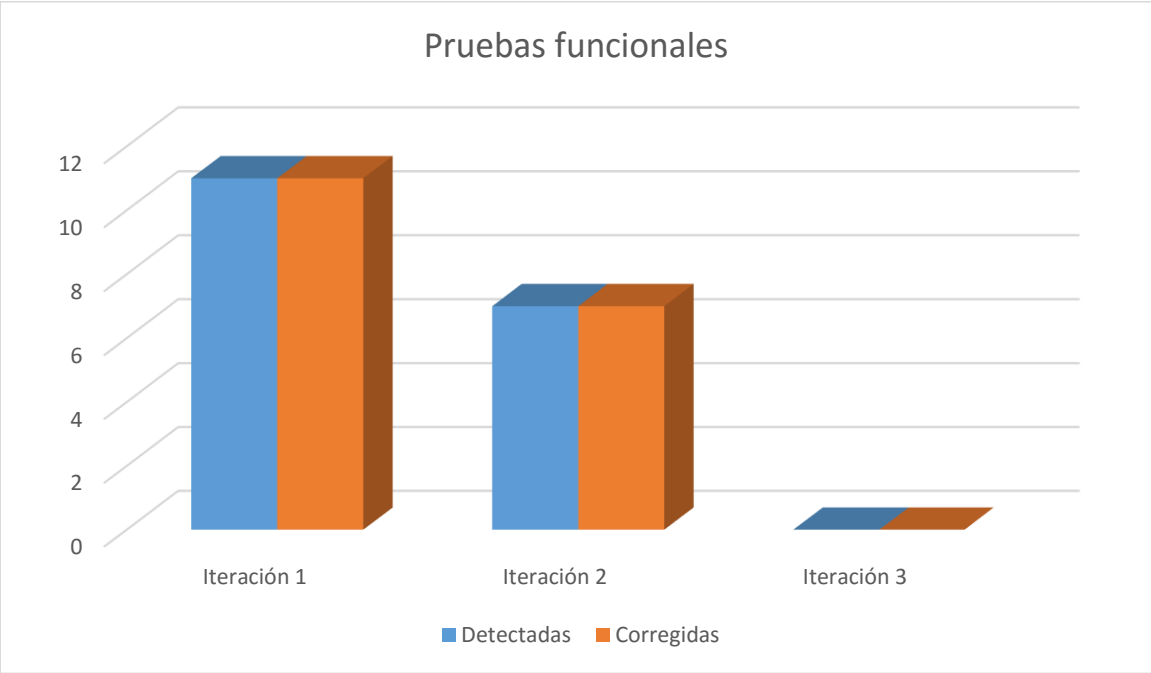



*Ilustración 11 Pruebas funcionales al "Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales"*

Entre las no conformidades detectadas en las pruebas funcionales se encuentran:

- Validación incorrecta sobre datos de entrada correctos.
- Opciones que no funcionan correctamente.
- Errores en la comunicación con los datos mostrados como resultados de las consultas a repositorios.

.

### 3.4. Pruebas de aceptación

Las pruebas de aceptación son aquellas que realiza el cliente al producto una vez concluido. Estas pruebas se consideran de caja negra y se les atribuye gran importancia en la etapa de pruebas, debido a que representan el criterio final del cliente. Cuando se finalizan las pruebas de aceptación, el cliente emite una carta de aceptación avalando que el producto cumple con los requisitos que se especificaron.

## 3.5. Conclusiones parciales

En el presente capítulo se expuso la validación de la solución propuesta por el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” dando paso a las siguientes conclusiones parciales:

- La confección del diagrama de componentes permitió observar la integración de los componentes del módulo separados por paquetes (búsqueda, configuración, ejemplo DSpace).
- Aplicar los estándares de codificación permitió obtener en el sistema un código legible, estándar y fácil de comprender lo que asegura la calidad y facilita un futuro mantenimiento.
- El proceso de validación permitió mediante las pruebas funcionales y de aceptación obtener un producto de mayor calidad.

## CONCLUSIONES GENERALES

Con la investigación y el desarrollo del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” se concluye que:

- Al definir el marco teórico conceptual de la investigación se evidenció la no existencia de un módulo integrable al software ATcnea que cumpliera con los requisitos especificados por el cliente.
- El diseño de la propuesta de solución permitió generar los artefactos acordes a la metodología de desarrollo de software AUP-UCI.
- La implementación del sistema a través de las herramientas y lenguajes seleccionados permitió obtener una aplicación capaz de consumir recursos almacenados en repositorios y su uso en los módulos de ATcnea. Además, se implementó una estructura para la adición de repositorios en una futura etapa de desarrollo del módulo.
- Se incluyó en la solución la implementación de la interacción con el repositorio DSpace, entregando una comunicación funcional plena según los requisitos especificados.
- La etapa de prueba del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales” permitió la detección de no conformidades y su posterior corrección, además, demostraron que el módulo constituye una solución funcional.

## RECOMENDACIONES

Se recomienda para futuras mejoras del “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”:

- Crear una guía paso a paso para añadir nuevos repositorios al módulo.
- Inclusión de nuevos tipos de repositorio (por ejemplo: Rhoda y Fedora) en el “Módulo de ATcnea para el acceso a recursos almacenados en repositorios digitales”.

## REFERENCIAS BIBLIOGRÁFICAS


*Intercambios virtuales.* [Online] http://www.intercambiosvirtuales.org/software/net-control-2-v740385-administre-sus-redes-lan-facilmente.

*Planet Tech Ed Educational Software.* [Online] [Citado em: 18 de enero de 2017.] http://planetteched.com/es/mythware-software-de-gestion-de-aula/.

*Dialnet.* [Online] [Citado em: 18 de enero de 2017.] https://dialnet.unirioja.es/info/ayuda/qe.

*Proyectos Ágiles.* [Online] [Citado em: 18 de enero de 2017.] https://proyectosagiles.org/desarrollo-iterativo-incremental/.

**2014.** *Genbeta.* [Online] 14 de julio de 2014. [Citado em: 14 de marzo de 2017.] https://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos.

**Alegsa, Leandro. 2009.** *Alegsa.* [En línea] 2009. [Citado el: 18 de enero de 2017.] http://www.alegsa.com.ar/Dic/modulo.php.

**Álvarez, Miguel Ángel. 2014.** *desarrolloweb.* [Online] 2 de enero de 2014. [Citado em: 10 de marzo de 2017.] https://desarrolloweb.com/articulos/que-es-mvc.html.

**AMAP. 2014.** *Arquitectura Marco para el desarrollo de software en la administración de la Comunidad Autónoma de Cantabria.* [Online] 14 de enero de 2014. [Citado em: 14 de mayo de 2017.] https://amap.cantabria.es/amap/bin/view/AMAP/CodificacionJava.

**Arvo Consultores.** *Arvo.* [Online] [Citado em: 27 de noviembre de 2016.] http://arvo.es/.

**Balo Pereira, Albert.** Ventajas de las CSS. [Online] [Citado em: 17 de mayo de 2017.] http://cv.uoc.edu/web/~abalo/Practica1/ventajas.html.

**Barrueco, José Manuel.** *OAI-PMH: Protocolo para la transmisión de contenidos en Internet.* Universidad de Valencia. Valencia : s.n.

**Biblioteca AECID.** Americanae. Sistema de Difusión y Recolección de colecciones americanistas. [Online] [Citado em: 15 de enero de 2016.] http://www.americanae.es/.

**Blackboard Inc. 2017.** *Blackboard.* [Online] 2017. [Citado em: 2 de junio de 2017.] http://www.blackboard.com.

**Booch, Grady, et al. 2007.** *Object-oriented analysis and design with applications third edition.* Boston : Pearson Education, 2007. Vol. III.

**CakePHP.** *CakePHP.* [Online] [Citado em: 14 de marzo de 2017.] https://book.cakephp.org/1.2/es/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html.

**Carpenter, Leona. 2003.** *Open Archives Forum.* [Online] 2003. [Citado em: 10 de enero de 2017.] http://travesia.mcu.es/portalnb/.

**Cervantes, Humberto. 2015.** *Software Guru.* [Online] 2015. [Citado em: 10 de marzo de 2017.] https://sg.com.mx/revista/27/arquitectura-software.

**CLARISE. 2013.** *Clarise: Comunidad Latinoamerica Abierta Regional de Investigación Social y Educativa.* [Online] 2013. [Citado em: 11 de junio de 2017.] https://sites.google.com/site/redclarise/.

**Constantino, Griselda. 2009.** *El acontecer en el Aula Tecnológica y el aprendizaje de los alumnos.* Universidad ORT. Uruguay : s.n., 2009.

**Diccionario de la lengua española © 2005 Espasa-Calpe. 2005.** [Online] 2005. [Citado em: 18 de enero de 2017.] http://www.wordreference.com/definicion/m%C3%B3dulo.

*Diseño de un repositorio de objetos de aprendizaje implementado con servicios web.* **Montilva, Jonas, Orjuela, Ailin e Mauricio, Rojas C. 2010.** 2, Mérida : s.n., 2010, Avances en Sistemas e Informática, Vol. 7.

**Ebersole, Steve, et al. 2016.** *Hibernate.* [Online] febrero de 2016. [Citado em: 31 de mayo de 2017.] https://docs.jboss.org/hibernate/orm/5.1/userguide/html_single/Hibernate_User_Guide.html.

**Espinosa, Marcelo, et al. 2015.** *Vinculando el repositorio institucional Dspace con la plataforma virtual Moodle.* Facultad Regional Resistencia – UTN. 2015.

**Fedora. 2015.** *Fedora.* [Online] 2015. [Citado em: 09 de junio de 2017.] http://fedorarepository.org.

**Fernández, Julio César. 2016.** *applecoding academy.* [Online] 2016. [Citado em: 31 de mayo de 2017.]

**Fielding, Roy T.** *Architectural Styles and the Design of Network-based Software Architectures.* [Online] [Citado em: 6 de marzo de 2017.] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

**García Oterino, Ana M del Carmen. 2014.** *javiergarzas.com.* [Online] 4 de julio de 2014. [Citado em: 14 de mayo de 2017.] http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html.

**García, J e Rodeíguez, J.** *Un estudio comparativo de dos herramientas MDA: OptimalJ y ArcStyler.* Departamento de Informática y Sistemas, Universidad de Murcia. Murcia : s.n.

**git.** *git.* [Online] [Citado em: 31 de mayo de 2017.] https://git-scm.com/about.

**Hernandez, Uriel. 2015.** [Online] octubre de 2015. [Citado em: 10 de marzo de 2017.] https://codigofacilito.com/articulos/mvc-model-view-controller-explicado.

**hsqldb.org. 2017.** *HyperSQL; HSQLDB - 100% Java Database.* [Online] 9 de abril de 2017. [Citado em: 31 de mayo de 2017.] http://hsqldb.org/.

**Ictea. 2013.** Ictea. [Online] 2013. [Citado em: 14 de mayo de 2017.] http://www.ictea.com/cs/knowledgebase.php?action=displayarticle&id=8790.

**iTalc.** *iTALC - Intelligent Teaching And Learning with Computers.* [Online] [Citado em: 18 de enero de 2017.] http://italc.sourceforge.net/.

**JetBrains.** *Jetbrains.* [Online] [Citado em: 2 de mayo de 2017.] https://www.jetbrains.com/idea/.

**LARMAN. 2004.** C. UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2004.

**Larman, Craig. 1999.** *UML y Patrones: introducción al análisis y diseño orientado a objetos.* Montevideo : s.n., 1999.

**—. 2003.** *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2da Edición.* Madrid : Pearson Education S.A., 2003. p. 624.

**Leiva, Antonio. 2016.** devexperto. [Online] marzo de 2016. [Citado em: 10 de marzo de 2017.] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwidgNXvgMzSAhXKKiYKHRK0BtcQFggzMAQ&url=https%3A%2F%2Fdevexperto.com%2Fpatrones-de-diseno-software%2F&usg=AFQjCNEZfGYFOCYJ8eaRKuZ2OgPoYqp2Fg.

**López Sandino, Jorge. 2013.** *OjuLearning.* [Online] 24 de junio de 2013. [Citado em: 9 de junio de 2017.] http://ojulearning.es/2013/06/bblearn9-1_sp13/.

**Lorenzo, Emilio. 2012.** *arvo.es.* [Online] 21 de septiembre de 2012. [Citado em: 6 de marzo de 2017.] http://www.arvo.es/dspace/tag/sword/.

**Lucidchart.** *Lucidchart.* [Online] [Citado em: 14 de marzo de 2017.] https://www.lucidchart.com/pages/es/diagrama-de-despliegue.

**Martínes Esparza, Miguel. 2011.** *Observatorio Tecnológico.* [Online] 10 de julio de 2011. [Citado em: 2017 de enero de 2018.] http://recursostic.educacion.es/observatorio/web/ca/software/software-educativo/1001-italc.

**Microsoft. 2017.** *Microsoft.* [Online] 2017. [Citado em: 14 de mayo de 2017.] https://msdn.microsoft.com/es-es/library/dd409390.aspx.

*Modelo vista controlador (MVC).* Universidad de Alicante. Alicante : s.n.

**Object Management Group.** Welcome To UML Web Site! [Online] [Citado em: 15 de enero de 2017.] http://www.uml.org/.

**Oracle.** *Oracle.* [Online] [Citado em: 14 de mayo de 2017.] http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html.

**—.** Java. [Online] [Citado em: 15 de enero de 2017.] https://www.java.com/.

**Oré B, Alexander.** *CalidadySoftware.com.* [Online] [Citado em: 14 de mayo de 2017.] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

**PMOinformatica.** *PMOinformatica.* [Online] [Citado em: 14 de mayo de 2017.] http://www.pmoinformatica.com/p/pruebas-de-software.html.

**—. 2017.** *PMOinformatica: La oficina de proyectos de informática.* [Online] 6 de febrero de 2017. [Citado em: 16 de mayo de 2017.] http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html.

**PoliScience.** *PoliScience.* [Online] [Citado em: 18 de enero de 2017.] http://poliscience.blogs.upv.es/open-access/repositorios/definicion-y-tipos/.

**Pressman, R. 2007.** *Ingeniería de software, un enfoque práctico.* Sexta. 2007.

**Puentes Daniel, Analiet e Jimenez Betancourt, Anyel. 2014.** *Implementación de estándares de interoperabilidad en el repositorio de recursos educativos RHODA.* Xauce Rhoda, Universidad de las Ciencias Informáticas. La Habana : s.n., 2014. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

*Repositorios digitales. Definición y pautas para su creación.* **Polanco-Cortés, Jorge. 2014.** San José : s.n., 2014.

*Rest vs Web Services.* **Navarro Marset, Rafael. 2007.** 2007.

**Rodríguez Gairín, Josep Manuel e Sulé Duesa, Andrew. 2008.** *bid.* [Online] junio de 2008. [Citado em: 9 de junio de 2017.] http://bid.ub.edu/.

**Rodríguez Sánchez, Tamara. 2015.** *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

**Rosario, Jimmy. 2007.** *"Las aulas virtuales como modelo de gestión del conocimiento".* 2007.

**Schmidt, Birgit e Müller, Katharina. 2011.** *El caso de Interoperabilidad para Repositorios de Acceso Abierto.* Confederation of Open Access Repositories. 2011.

**Segovia Olmo, Felipe. 2003.** *El aula inteligente.* 2003.

**Simón Ramírez, William. 2016.** *Fortes_DPA_Estandares de codificacion para Java.* Universidad de las Ciencias Informáticas. La habana : s.n., 2016. p. 30, Programa de mejora.

**software.com.ar. 2016.** *software.com.ar.* [Online] 2016. http://software.com.ar.

**Tomás, Eduard. 2014.** *desarrolloweb.com.* [Online] 25 de abril de 2014. [Citado em: 10 de enero de 2017.] http://www.desarrolloweb.com/manuales/15/.

**—. 2014.** *DesarrolloWeb.* [Online] 25 de abril de 2014. [Citado em: 12 de junio de 2017.] http://www.desarrolloweb.com/.

*Transforming Scholarly Publishing Through Open Access: A Bibliography.* **Bailey, Charles W. 2010.** Houston : s.n., 2010, Texas: Digital Scholarship.

**Universidad de las Americas Puebla.** [Online] [Citado em: 14 de marzo de 2017.] http://ict.udlap.mx/people/carlos/is341/bases02.html.

**UvaDoc: Repositorio Documental de la Universidad de Valladolid. 2013.** *UvaDoc.* [Online] 18 de octubre de 2013. [Citado em: 3 de 12 de 2016.] http://uvadoc.blogs.uva.es/category/repositorios/.

**Viñas, Meritxell. 2011.** *Qué es y cómo usar un software de gestión y control de aula TIC.* Facultad de Economía de la Universidad de Valencia. Valencia : s.n., 2011.

**Visual Paradigm.** [Online] [Citado em: 15 de enero de 2017.] http://www.visual-paradigm.com/.

**W3C. 2016.** *W3C: Information and knowledge domain.* [Online] 11 de octubre de 2016. [Citado em: 16 de mayo de 2017.] https://www.w3.org/XML/.