Sophói

Plataforma web interactiva para capacitación en testing de software

Andrea Estefan
Gerardo Fanjul
Rodrigo Quinta
Tutora: Mónica Wodzislawski

Informe de Proyecto de Grado de la carrera Ingeniería en Computación presentado al Tribunal Evaluador.

Montevideo, República Oriental del Uruguay. Año 2016.

Resumen

Los cursos en modalidad e-learning se han potenciado en los últimos años asociados a un aumento en la conectividad a internet y el deseo de las personas de acceder a cursos a distancia que además no tengan un horario estipulado, permitiendo así a los estudiantes ajustar sus horarios. Simplemente con un pc y acceso a internet, una persona puede participar de cursos online, lo que conlleva que más gente acceda a capacitaciones en distintas áreas de conocimiento.

Este proyecto presenta el desarrollo de una plataforma web interactiva, denominada *Sophói*, del griego antiguo σοφοί (sophói, “sabio”, adj), implementada con el objetivo de capacitar a los testers de software. *Sophói* permite a los usuarios aprender, practicar y combinar el uso de las diferentes técnicas de testing de software, utilizadas habitualmente en los problemas que se presentan en las pruebas de sistemas informáticos. La plataforma enfrenta al alumno a lecciones y ejercicios prácticos, que cuentan con corrección automática y son previamente diseñados por docentes desde la misma plataforma, mediante el uso de un generador de ejercicios. El generador permite combinar un conjunto de componentes predefinidos para diseñar la interfaz de cada ejercicio y organizar éstos dentro de las lecciones.

Contenido

Resumen 2
1.Introducción 5
2. Capacitación en testing de software 7
2.1 Introducción 7
2.2 Evolución del testing 7
2.3 Capacitación en testing 8
2.4 Herramientas educativas 9
3. Descripción del proyecto 12
3.1 Descripción del proyecto 12
3.2 Planificación del proyecto 12
4. Especificación de requerimientos 14
4.1 Definiciones preliminares 14
4.2 Solución plateada 14
4.3 Características de los usuarios 16
4.4 Requerimientos funcionales 17
4.5 Requerimientos no funcionales 20
5. Diseño y arquitectura del sistema 21
5.1 Arquitectura del sistema 21
5.2 Modelado del dominio 23
5.3 Casos de uso relevantes 24
5.4 Base de datos 25
6. Implementación 27
6.1 Elección de tecnologías 27
6.2 Diagrama de tecnologías 36
6.3 Estructura de los proyectos 36
6.4 Implementación de la solución 37
7. Testing 45
8. Gestión del proyecto 48
8.1 Gestión del proyecto 48
8.2 Gestión del riesgo 49
8.3 Gestión de cambios 51
9. Conclusiones y Trabajo Futuro. 52
9.3 Conclusiones 52

9.2 Trabajo a futuro 52
10. Referencias 54
11. Anexos 55

1.Introducción

Con el crecimiento de la producción de software desde Uruguay para el resto del mundo, es importante que el software que es exportado sea de buena calidad. Debido a esto en los últimos años ha aumentado el testing en los proyectos informáticos, lo que conlleva la necesidad de más y mejores testers de software que conforme avanzan las tecnologías, mejoren su base de conocimiento en el área.

En el ámbito del testing funcional, se aplican técnicas, que combinadas permiten testear una gran variedad de sistemas. Un tester con buen conocimiento de las técnicas, puede generar casos de pruebas más eficientes. La práctica de estas técnicas frente a diferentes problemas ayuda al tester a desarrollar su audacia y abre su mente, lo que promueve ideas más interesantes.

Dada la escasez de aplicaciones que complementen el aprendizaje en testing y en particular que permitan practicar el uso de las distintas técnicas en variadas realidades, que a su vez sean didácticos, atractivos y lúdicos, surgió la propuesta de desarrollar una plataforma interactiva para entrenar testers de software.

El proyecto que se describe en este informe surge de la necesidad del Centro de Ensayos de Software de contar con una herramienta que permita potenciar los cursos de testing dictados por el Centro; facilitando a los docentes la generación y corrección de ejercicios prácticos.

Las principales motivaciones que propulsaron el desarrollo del proyecto incluyen facilitar la generación de ejercicios interactivos, sin necesidad de que el docente desarrolle código. Que la corrección de estos ejercicios pueda hacerse de forma automática y los resultados ser presentados de forma detallada a los docentes. Para nosotros, particularmente interesados en el testing de aplicaciones de software y habiendo cursado electivas en dicha área de la carrera, tomamos el gran desafío de generar un producto de software de calidad que sirve como herramienta interactiva de apoyo en la enseñanza de testing.

Por tanto, nuestra motivación es la de aportar valor agregado a los cursos de testing mediante el uso de Sophói, dejando abierta la posibilidad de expandir el proyecto y generalizarlo, no sólo para el aprendizaje de testing sino para cualquier disciplina que involucre lecciones, ejercicios y correcciones automáticas.

En el contexto del proyecto se implementa una aplicación web, escalable, extensible y altamente configurable, que permite a los docentes del Centro de Ensayos de Software generar ejercicios prácticos para ser utilizados en las distintas actividades de capacitación impartidas por el centro y hacer un seguimiento del avance de los estudiantes por medio de estadísticas. La plataforma ofrece diferentes herramientas para la generación de nuevos ejercicios sin requerir conocimientos de desarrollo de nuevo código y a su vez permite que los ejercicios tengan la posibilidad de ser corregidos de forma automática y manual. Sophói permite el acceso *just-in-time* a la herramienta, o sea que los estudiantes pueden experimentar

con casos de estudio y aplicar técnicas de testing con la disponibilidad que ofrece una aplicación web, desde cualquier dispositivo con conexión a internet, sin depender de la presencia física de un docente o de la coordinación en tiempo y espacio con el resto de los alumnos de un curso. El Proyecto es desarrollado conjuntamente con la Lic. Mónica Wodzislawski en el rol de tutor y satisface las necesidades del Centro de Ensayos de Software y su personal docente.

El resto del informe se organiza de la siguiente manera: en el capítulo 2 se presenta el estudio del estado del arte sobre enseñanza del testing y plataformas de aprendizaje. En el capítulo 3 se presenta una introducción al problema que aborda el proyecto y la planificación general. En el capítulo 4 se especifican los requerimientos funcionales y no funcionales. En el 5 se describe la arquitectura y diseño del sistema. El capítulo 6 describe los *frameworks* y bibliotecas utilizados/as para la implementación del proyecto y presenta la lógica de las principales funcionalidades.En el capítulo 7 se exponen las pruebas ejecutadas y los resultados obtenidos. En el capítulo 8 se detalla la planificación del proyecto, las herramientas de gestión de cambio utilizadas y se analizan riesgos. Finalmente, el capítulo 9 contiene las conclusiones del proyecto y el trabajo a futuro.

2. Capacitación en testing de software

En este capítulo se detalla la investigación realizada acerca de metodologías de enseñanza en general y en particular asociadas al área del testing. Por otra parte, se recaba información de plataformas de aprendizaje existentes evaluando sus pros y contras.

2.1 Introducción

La investigación comenzó con el estudio de WebGoat[1], herramienta sugerida por nuestra tutora para tomar como modelo de lo que el cliente necesitaba. También se evaluaron otras herramientas educativas en testing recomendadas por el cliente. Este estudio fue acompañado por lecturas sobre la educación en testing e investigaciones sobre temas tratados en conferencias internacionales y eventos sobre el área de trabajo.

Para esto recurrimos a buscar información en Timbó[2], haciendo búsquedas por palabras claves en Google y recurriendo a los sitios web de los pioneros del testing.

2.2 Evolución del testing

El testing se constituye como disciplina formal de la ingeniería de software en los años 70, pero una década antes ya aparecen publicaciones referentes al testing[3].

A finales de los años 90 comienzan a realizarse conferencias dedicadas al testing y surgen las primeras certificaciones en el área aumentando la necesidad de formar profesionales capaces de llevar a cabo el testing. Se fundan organizaciones internacionales dedicadas a atacar la problemática de la formación de testers, se destacan entre ellas la *International Software Testing Qualifications Board* (ISTQB [4]) fundada en el año 2002 y la *Association for Software Testing* (AST[5]) creada en el año 2003. En ese mismo momento en la facultad de Ingeniería de la Universidad de la República también comienza a considerarse el testing como parte de la formación del ingeniero en informática. Se agrega a la currícula de la carrera el curso optativo Taller de Verificación de Software (TVS) en el año 2004, posteriormente el *Centro de Ensayos de Software* (CES[6]) crea la carrera de "Tester de Software", la cual surge por la necesidad de concentrar las distintas instancias de capacitación que el centro brindaba en la industria y en cursos de posgrado de Facultad de Ingeniería.

Antonia Bertolino, en su presentación "*Recent trends in software testing research: an unsystematic (road)mapping study"*[7], realizada en la 17th CREST *Open Workshop Software Testing and Verification* durante el año 2012, presenta un roadmap con los campos de estudio y los desafíos de y para el testing en ese entonces. Toma como punto de partida el análogo presentado por *Future of Software Engineering* (FOSE) en el año 2007. En esta presentación Bertolino destaca a la educación en testing como uno de los nuevos desafíos que surgen en ese periodo de tiempo.

El testing es la forma de verificación más utilizada por la industria, no es una tarea sencilla, ni repetitiva. Cem Kaner[8], por su parte, dice que se necesitan testers con

imaginación, percepción, que entiendan y se adapten al contexto en el que es realizado el testing.

La diferencia entre el testing y otro tipo de análisis o método de verificación se basa en la habilidad del tester de observar y analizar los resultados, comparándolos con versiones previas. Éste debe desarrollar dichas habilidades para acompañar la evolución del software y del testing de software.

2.3 Capacitación en testing

La educación en testing actualmente continúa siendo un desafío. En muchas universidades a nivel mundial no es considerado en la formación de profesionales informáticos o no tiene la importancia que debería tener. Por ejemplo, la ya mencionada asignatura TVS dictada por el INCO en Facultad de Ingeniería (UDELAR) no es una asignatura obligatoria en la carrera de Ingeniería en Computación, por tal motivo pocos ingenieros recibidos cuentan con la formación de testing sugerida. Fuera de las aulas universitarias existen otras alternativas para aprender testing: cursos online, seminarios y cursos dictados por organizaciones dedicadas a la formación de testers. Muchos de estos cursos están destinados al aprendizaje de herramientas de testing por parte del encargado de cumplir con esta tarea y no orientados a desarrollar sus habilidades.

Entre los principales referentes en lo que a educación e investigación en testing se refiere encontramos a James Bach[9], Michael Bolton[10], Cem Kaner y Scott Barber[11]. Conocidos por ser los autores de importantes artículos y cursos enfocados en “Entrenar testers”, dictados como talleres intensivos presenciales y/o online.

A partir del año 2007 Cem Kaner lanza el curso Black Box Software Testing (BBST), un conjunto de 3 cursos (foundations, advocacy y test design) en línea, para entrenar testers desde el enfoque de “Caja negra”, que incluyen videos, cuestionarios y trabajos que los estudiantes deben realizar.

Michael Bolton y James Bach, con el apoyo de miembros de la Context-Driven School of software testing desarrollaron el Rapid Software Testing[12], un taller de tres días de duración, el cual tiene como principal objetivo mejorar las aptitudes de testing para cualquier tipo de software, en cualquier período de tiempo y bajo cualquier condición. La filosofía que se presenta no es como los enfoques tradicionales de testing, que ignoran la parte pensante de la prueba y en cambio defienden la excesiva documentación (según los autores). El curso no sólo se enfoca en un sentido de urgencia de las pruebas, sino que su máximo objetivo es la eliminación de trabajo innecesario, ayudado por una constante mejora del proceso.

Por su parte James Bach es un apasionado de testing como disciplina y de formar nuevos testers. Es un consultor independiente, quien ha participado activamente en las conferencias de la AST y ha realizado un importante conjunto de publicaciones. Se especializa en el testing exploratorio y junto con su hermano Jonathan, es el creador del testing basado en sesiones. Además de su participación en el curso *Rapid Software Testing*, Bach brinda pequeñas lecciones a través de Skype. En estas lecciones propone preguntas a los participantes sobre cómo encarar distintos problemas de testing y observa cómo ellos los resuelven siguiendo una metodología un tanto peculiar, presionando de manera excesiva al alumno. Su objetivo además de educar según los principios de la *Context-Driven School of software testing*, es mejorar sus habilidades para entrenar testers.

Dentro de lo que es la enseñanza en testing se puede apreciar el trabajo *"I am a bug"*[13], del autor Robert Sabourin, quien definió lecciones para enseñar desde conceptos básicos, por ejemplo lo que es un bug, hasta lo importante: qué es el testing, cómo se deben priorizar los bugs, cómo llegar a ellos, etc. La metodología es un tanto extraña para personas adultas pero fácil de comprender, la comunicación es a través de dibujos donde se explica lección a lección conceptos relacionados con el testing mediante la interacción con el lector. Se hace hincapié en el daño que puede causar cierto tipo de bugs haciendo analogías con la vida real para una mejor comprensión por parte de público no informático.

Por otro lado, el crecimiento del área trae aparejada la necesidad de expandir dichos conocimientos a través de conferencias como la recientemente creada en Europa (precisamente en Suecia), la conferencia Let´s Test[14]. Esta organización pretende con estas conferencias lograr una experiencia valiosa para todos los participantes bajo el lema de "para los testers, por los testers".

Estas conferencias son llevadas a cabo por parte de un equipo formado por testers europeos apasionados y profesionales que ya en 2011 decidieron que era el momento de impulsar las conferencias en Europa luego de presenciar las conferencias de AST en Estados Unidos. Desde la primer conferencia Let`s test en 2012 hasta ahora los miembros del equipo han evolucionado y las conferencias se han expandido y organizado en Australia y Sudáfrica.

2.4 Herramientas educativas

En esta sección se presentan herramientas educativas investigadas tanto en el área de testing como en otras áreas de estudio que sirven de puntapié inicial para definir Sophói.

2.4.1 WebGoat

WebGoat es una aplicación web de código abierto mantenida por OWASP[15] (Open Web Application Security Project) y diseñada con agujeros de seguridad. El propósito de WebGoat es ilustrar las típicas fallas de seguridad que pueden encontrarse en una aplicación web y enseñar formas de atacar estas vulnerabilidades.

La aplicación cuenta con un conjunto de lecciones que permite a los usuarios entender las vulnerabilidades de seguridad y enfrentar estos problemas en la práctica. El usuario prueba la aplicación con la misma información que tendría en un proyecto real, los ataques son remotos y las pruebas son de caja negra. Las lecciones además de presentar el problema que el usuario debe resolver proveen al usuario de conceptos teóricos, ejemplos y pistas. Las lecciones tienen precedencia entre sí, para poder resolver una lección se debe haber aprobado la anterior. La aplicación permite seguir el progreso de un usuario a través de las lecciones.

WebGoat está desarrollado en Java por lo que es independiente de cualquier plataforma, solamente se necesita contar con un entorno virtual de Java. En versiones previas era necesario contar con un servidor Tomcat instalado para poder ejecutar la aplicación, actualmente el servidor viene incluido con la aplicación lo que simplifica su instalación y ejecución.

Dado que WebGoat es de código abierto y apunta a ser desarrollada por la comunidad, si bien es mantenida por OWASP, los usuarios pueden desarrollar nuevas lecciones y agregarlas a la aplicación. Para esto WebGoat cuenta con una clase abstracta que presenta un

conjunto de métodos que el usuario debe implementar para que su lección se integre a la aplicación. El proyecto utiliza el *Element Construction Set* del proyecto Jakarta[16] para la creación de las interfaces gráficas de las lecciones.

Actualmente el personal dedicado a este proyecto está trabajando en cambios arquitectónicos para separar el servidor de lecciones del resto de los componentes y así facilitar el desarrollo de nuevas lecciones.

2.4.2 Bug-Hunt: Making Early Software Testing Lessons Engaging and Affordable[17]

Es una aplicación web para involucrar a los estudiantes en el aprendizaje de técnicas de testing de software, utilizando un enfoque lúdico. La herramienta desarrollada por Sebastián Elbaum, Suzette Person y Jon Dokulil del departamento de Computer Science and Engineering de la Universidad de Nebraska-Lincoln en Estados Unidos, busca incorporar el testing a los cursos iniciales de Computer Science. Con el argumento de que los desarrolladores deben aprender la importancia del testing a una temprana etapa, ya que muchas veces cuando se alcanzan los cursos de testing en etapas avanzadas de la carrera, los desarrolladores han adquirido malos hábitos que después son más difíciles de erradicar.

Bug Hunt consta de un conjunto de lecciones y desafíos pre-cargados que el estudiante debe resolver. Cada desafío busca ejercitar un conjunto de técnicas de testing y provee al alumno de un feedback en tiempo real. Además de las lecciones previamente configuradas ofrece la posibilidad de configurar las lecciones. Mediante el análisis y reporte automático del trabajo de los alumnos evita que el profesor tenga que corregir cada y trabajo. Y permite al docente obtener reportes sobre los resultados obtenidos en las lecciones.

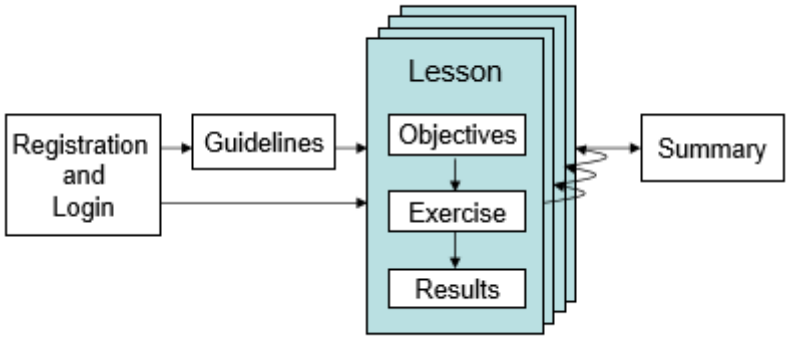


Figura 1 - Estructura Bug Hunt

Cada lección presenta un conjunto de objetivos que el estudiante debe intentar cumplir, para lo cual el estudiante diseña casos de prueba y los carga en la herramienta. La herramienta ejecuta los casos de pruebas y registra los incidentes detectados. Para la evaluación compara los incidentes detectados por el alumno con los incidentes detectados por el resto de la clase. A las lecciones se les pueden agregar nuevos desafíos o modificar los ya existentes mediante una interfaz online. También es posible agregar y modificar los objetivos y las nuevas pistas de las lecciones y desafíos.

2.4.3 Black box puzzles[18] - Otros

Además de los dos casos de estudio previos, que son los que por diferentes razones se relacionan más con nuestro producto, se investigaron otras herramientas educativas.

Black box es una aplicación web con formato de puzle usada para el desarrollo de habilidades de testing, el concepto es entregarle al usuario pequeñas aplicaciones (puzzles) para que pruebe. Fue desarrollado por James Lyndsay para ser utilizado en sus talleres de capacitación en Testing. En el sitio web se pueden observar algunos puzzles de ejemplo. Cada puzle se centra en la metodología de caja negra. Consiste en una serie de botones y luces, el usuario tiene que descubrir el funcionamiento y patrones de comportamiento. El funcionamiento no siempre es fácil de descubrir ni trivial a primera vista, la idea es que el tester encuentre varias combinaciones y deduzca el correcto funcionamiento.

Otra herramienta estudiada, en este caso no relacionada al testing, es la popular Duolingo. Duolingo[19] es una aplicación multiplataforma que ayuda al usuario a aprender diferentes idiomas a través de ejercicios con estilo lúdico. Los usuarios generan un perfil y participan en cursos de diferentes idiomas, la aplicación proporciona premios por el avance en los cursos y quita vidas cuando se comenten errores. También incentiva al usuario cuando comienza a quedarse para atrás o participar menos de los cursos. Tienen una variedad de cursos que permiten practicar el habla, la escritura y la escucha de idiomas.

Finalmente se investigaron los cursos web de W3Schools[20], un sitio web dedicado a ayudar a desarrolladores a aprender los distintos lenguajes y herramientas para el desarrollo de sitios web. W3Schools ofrece cursos en los que al usuario se le presentan conceptos teóricos acompañados de ejemplos y luego de la lectura, se le desafía a implementar algo similar, mediante el concepto “Try it yourself”. Dependiendo del curso el usuario escribe código, lo ejecuta y el sitio le muestra cómo se vería lo que escribió.

3. Descripción del proyecto

Este capítulo cuenta con dos secciones. En la primera se presenta una introducción al problema que propone abordar el proyecto. En la segunda sección se especifica la planificación del proyecto.

3.1 Descripción del proyecto

El proyecto consiste en el diseño e implementación de una plataforma web interactiva que sirva de apoyo para la capacitación de testers de software. La herramienta debe brindar retroalimentación al estudiante mientras éste resuelve los ejercicios, permitiéndole avanzar en la resolución de problemas sin esperar por la participación de un docente. Además, debe ofrecerle al docente una funcionalidad que permita la creación de nuevos ejercicios desde la herramienta, a la vez que hacer seguimiento del avance de los estudiantes.

Para construir una herramienta de enseñanza acorde a las necesidades del cliente se investiga la forma de trabajo de los docentes de la carrera de testing impartida por el CES, incluyendo la manera en que se evalúa a los alumnos. A los docentes se les asignan grupos de alumnos a los que evalúa y hace un seguimiento durante el transcurso del curso. Por otra parte, el CES cuenta con un grupo de tutores que hacen un seguimiento de los alumnos a través de los diferentes cursos a los que asiste el alumno a lo largo de la carrera. La carrera de testing es dictada totalmente en línea y en ella participan alumnos de distintos países de Latinoamérica.

Se propone un alcance que permita cumplir con los requerimientos funcionales descritos anteriormente y cumplir con requerimientos no funcionales deseables en cualquier aplicación web moderna. Como solución se propuso la plataforma web interactiva Sophói.

Los usuarios cuentan con un perfil en la plataforma que les permite interactuar con la misma. Este perfil debe permitir a los estudiantes ver su avance a través de las lecciones y ejercicios propuestos. A su vez el docente cuenta con una interfaz que le permite crear nuevas lecciones que contengan ejercicios y consultar el avance de los estudiantes en la plataforma. Uno de los principales objetivos de la plataforma es permitir la corrección automática de los ejercicios, para lograrlo, se categorizan los ejercicios según las técnicas de testing que ejercitan y se definen algoritmos para la corrección automática. Otro objetivo importante es que la herramienta sea extensible, para lo cual se define una arquitectura que facilita la incorporación de nuevas categorías de ejercicios y algoritmos de corrección.

A su vez la solución planteada es altamente configurable, provee las herramientas necesarias para personalizar los cursos, las escalas de notas de aprobación y todas las funcionalidades para administrar alumnos, profesores y grupos.

3.2 Planificación del proyecto

Al principio del proyecto se acuerda con el cliente la investigación de herramientas ya existentes y junto con las necesidades planteadas por docentes que imparten la carrera de testing se realiza una propuesta de plataforma de enseñanza. Se acuerda implantar la herramienta con una base de ejercicios que abarque la mayoría de las categorías de testing enseñadas.

El proyecto se divide en las siguientes etapas:

3.2.1 Etapa de investigación

En esta primera etapa se lleva a cabo un estudio del testing en general, de las metodologías de enseñanza que se aplican y las herramientas de software utilizadas como soporte. Se realiza un relevamiento de las técnicas de testing y los ejercicios propuestos en los cursos de la carrera de testing. En base a todos los datos recabados y teniendo en cuenta la necesidad de digitalización de la información referente a alumnos y docentes de la carrera se hace una propuesta inicial de alcance junto con la especificación de requerimientos del sistema.

3.2.2 Etapa de diseño

En la segunda etapa del proyecto y luego de haber definido las necesidades del cliente se comienza con la arquitectura del sistema. También en esta etapa se crean algunos prototipos y se investigan *frameworks* de desarrollo que posean la flexibilidad necesaria para poder construir interfaz gráfica en *runtime.* Se toma como condición previa que la herramienta debe ser una aplicación web.

3.2.3 Etapa de implementación

Luego de haber realizado el diseño y definido los requerimientos, en esta tercera etapa se construirá el sistema. Se ajusta el diseño de los ejercicios para que encajen en la plataforma de aprendizaje

3.2.4 Etapa final

En la última etapa del proyecto se realizan pruebas de usabilidad de la herramienta, se implanta el producto en los servidores del cliente y se genera un juego de datos básico para comenzar a utilizar la herramienta en los cursos de testing.

4. Especificación de requerimientos

En este capítulo se describe el problema planteado, especificando los requerimientos funcionales y no funcionales formulados a partir de la información recabada en reuniones con el cliente.

4.1 Definiciones preliminares

En este punto se definen los principales conceptos utilizados a lo largo de los próximos capítulos.

Lección
La lección se define como una unidad de aprendizaje compuesta por un marco teórico y un conjunto ordenado de ejercicios que tienen un hilo conceptual en común, puede ser por ejemplo una realidad concreta planteada para testear.

Ejercicio
Un ejercicio es una unidad práctica que permite desarrollar las habilidades del tester y a su vez evaluar sus conocimientos. El ejercicio se caracteriza por tener una categoría y ser parte de una lección, también tiene definida una potencial solución propuesta por el docente.

Categoría de un ejercicio
La herramienta define categorías de ejercicios para poder clasificarlos según su algoritmo de evaluación, algunas de las categorías coinciden con las técnicas de testing enseñadas en los cursos de las carreras dictadas.

Grupo
Se define un grupo como un docente y un conjunto de alumnos que están bajo su supervisión.

4.2 Solución plateada

La parte central de la plataforma gira en torno a las lecciones y los ejercicios, estos permiten resolver la problemática planteada. Las lecciones se componen de un conjunto de ejercicios ordenados y una descripción. Los ejercicios a los que se enfrenta el alumno, cuentan con una descripción del problema a resolver y campos para que el usuario ingrese los resultados.

El estudiante comienza la lección intentando resolver el primer ejercicio de ésta, si lo aprueba se desbloquea el siguiente ejercicio y así sucesivamente hasta llegar al último. Los ejercicios son presentados al estudiante como una página web en la que se agregan las respuestas y luego se indica que se finalizó el ingreso para que el ejercicio sea corregido. El sistema corrige las respuestas y devuelve al estudiante el puntaje obtenido, indicando además si el ejercicio fue aprobado o no. En caso de no ser aprobado pero se alcance un puntaje mínimo, se ofrece al usuario la posibilidad de consultar pistas que lo ayuden a contestar lo que no respondió, y/o respondió incorrectamente. En caso de utilizarlas, estas pistas le restan puntos del resultado final. El usuario puede aceptar una o varias pistas y responder nuevamente o quedarse con el puntaje obtenido. Al finalizar este proceso si el puntaje alcanza

el mínimo de aprobación, el ejercicio es marcado como aprobado y se habilita, en caso de existir, el siguiente ejercicio de la lección.

Además de las pistas que se le ofrecen al usuario cuando envía a corregir sus respuestas, los ejercicios pueden contar con un conjunto de ayudas para guiar a los estudiantes en la resolución del mismo. Estas ayudas no afectan el puntaje final.

El puntaje de aprobación, el puntaje mínimo para que muestre pistas y los puntos que resta cada pista son configurables desde las preferencias del sistema.

La *Figura 2 - Estados de resolución de un ejercicio*, muestra los estados por los que pasa un ejercicio.

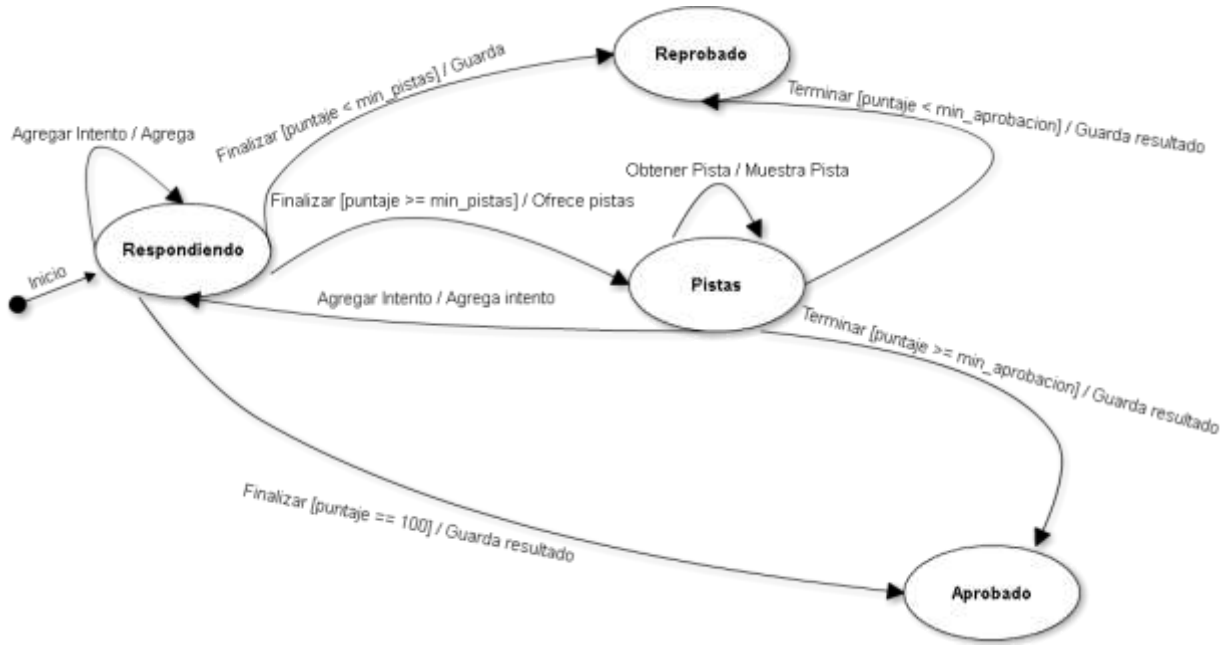


Figura 2 - Estados de resolución de un ejercicio

Se definen categorías para los ejercicios según el problema que permitan resolver. Cada categoría tiene un algoritmo de corrección automática asociado, esto se detalla en la sección 4.2.1. A nivel de interfaz gráfica cada categoría ofrece distintos componentes como se explica en la sección 6.4.2. La *Tabla 1 - Categorías*, muestra los componentes de cada categoría.

Categoría	Componentes[1]
Árbol de decisión	Árbol y Texto
Clase de equivalencia	Texto, Imagen, Lista de Valores, Variable a ingresar y Clase de equivalencia.
Identificación de variables	Lista de Valores y Variable a ingresar
Representante	Texto, Imagen, Lista de Valores, Variable a ingresar y Representante
Revisión de Código	Imagen y Variable a ingresar
Máquina de estados	Grafo y Texto
Valores límites	Lista de Valores y Variable a ingresar

Tabla 1 - Categorías

4.2.1 Corrección de ejercicios

La plataforma permite la corrección de ejercicios en dos modalidades, manual y automática. Al momento de ser creado el ejercicio se debe indicar cuál de las dos modalidades utilizará para la corrección.

La corrección manual refiere a la modalidad en la que el docente debe intervenir. Para esto el docente cuenta con una interfaz en la que puede consultar las respuestas de los alumnos y asignarle un puntaje entre 1 y 100 a cada ejercicio. A partir del puntaje el sistema determina la nota y si corresponde marca el ejercicio como aprobado, desbloqueando el próximo ejercicio de la lección. Los ejercicios que son de corrección manual no permiten que el estudiante avance al siguiente ejercicio hasta que el docente no corrija el ejercicio actual. La idea de ir mostrando los ejercicios de a uno surgió de las reuniones con el cliente. Ya que esto permite generar dependencias entre los ejercicios. Por ejemplo el primer ejercicio de una lección puede ser identificar las variables que afectan el problema y el segundo ejercicio puede ser dar las clases de equivalencia para esas variables

Por otro lado, para la corrección automática la plataforma utiliza algoritmos definidos en archivos JavaScript que se ejecutan en el lado servidor de GWT. Se decidió utilizar JavaScript para la corrección porque permite que se agreguen algoritmos sin tener que compilar y desplegar la aplicación nuevamente.

4.3 Características de los usuarios

El cliente muestra interés en que las distintas funcionalidades del aplicativo puedan ser configurables por perfiles contando con la posibilidad de asignar o quitar privilegios. Se detectan dos roles: alumno y docente. El rol del docente puede tener privilegios de administrador de la herramienta.

[1] Los componentes se describen en la *Tabla 5 - Componentes* de la sección 6.4.2.

Los usuarios sin autenticación pueden entrar a la aplicación como invitados y acceder a lecciones de muestra, predefinidas previamente por los docentes, sin embargo, para guardar el avance en las lecciones es necesario estar logueado al sistema.

Alumno
Usuarios con funcionalidades restringidas a la resolución de los ejercicios planteados dentro de cada lección de la plataforma educativa. El sistema está enfocado en este tipo de usuarios.
Funciones:

- Acceder y resolver las lecciones.
- Obtener reportes de su progreso.

4.4 Requerimientos funcionales

En esta sección se listan y enumeran los requerimientos recabados, especificando el rol del usuario que puede ejecutar la acción. Se priorizan utilizando la siguiente escala.

- **Alta**: La aplicación debe soportar estos requerimientos que además guían el diseño de la arquitectura.
- **Media**: Requerimientos que deben ser implementados en alguna versión del proyecto.
- **Baja**: Son requerimientos complementarios.

Docente
Es el encargado de diseñar los ejercicios y lecciones para los alumnos. El docente además gestiona los alumnos y grupos de alumnos.
Funciones:

- Gestionar usuarios.
- Crear y administrar lecciones.
- Diseñar y administrar ejercicios.
- Generar reportes.

Administrador
El administrador posee todos los privilegios, tiene control absoluto sobre la aplicación, puede cambiar los parámetros de configuración como ser la escala de notas, la nota mínima de aprobación de un ejercicio, etc.

Nombre	Descripción	Rol	Prioridad
Usuarios			
Inicio de sesión	El sistema debe permitir al usuario autenticarse en el sistema.	Alumno Docente Administrador	Alta
Fin de sesión	El sistema debe permitir al usuario cerrar sesión.	Alumno Docente Administrador	Alta
Ingreso como invitado	El usuario si no tiene acceso a la aplicación debe poder ingresar al sistema como invitado.	Alumno Docente Administrador	Media
Modificación de contraseña	El usuario debe poder modificar su contraseña de autenticación al sistema	Alumno Docente Administrador	Alta
Alta de alumno	El sistema debe permitir dar de alta un alumno	Docente Administrador	Alta
Baja de alumno	El sistema debe permitir dar de baja un alumno	Docente Administrador	Alta
Modificación de alumno	El sistema debe permitir modificar los datos de un alumno ya existente.	Docente Administrador	Alta
Alta de docente	El sistema debe permitir dar de alta un docente	Docente Administrador	Alta
Baja de docente	El sistema debe permitir dar de baja un docente	Docente Administrador	Alta
Alta de grupo	El sistema debe permitir crear un grupo de trabajo y asignar un docente	Docente Administrador	Alta
Baja de grupo	El sistema debe permitir eliminar grupos existentes	Docente Administrador	Media
Lecciones y ejercicios			
Crear nueva lección	El usuario debe poder crear una nueva lección	Docente Administrador	Alta
Editar lección	El usuario debe poder editar una lección	Docente Administrador	Media
Inactivar lección	El usuario debe poder ocultar una lección	Docente Administrador	Baja
Eliminar lección	El usuario debe poder eliminar	Docente	Media

	una lección	Administrador	
Listar lecciones existentes	El sistema debe listar las lecciones existentes	Docente Administrador	Alta
Crear nuevo ejercicio	El usuario debe poder crear un nuevo ejercicio vinculado a una lección.	Docente Administrador	Alta
Editar ejercicio	El usuario debe poder editar un ejercicio existente.	Docente Administrador	Alta
Inactivar ejercicio	El usuario debe poder ocultar un ejercicio.	Docente Administrador	Media
Copiar ejercicio	El sistema debe permitir copiar un ejercicio	Docente Administrador	Baja
Agregar pistas al ejercicio	El sistema debe permitir agregar pistas asociadas a un ejercicio.	Docente Administrador	Media
Listar ejercicios	El sistema debe listar los ejercicios según la lección a la que pertenezca.	Docente Administrador	Alta
Hacer ejercicio	El usuario debe poder hacer los ejercicios planteados en las lecciones.	Alumno Docente Administrador	Alta
Reportes			
Ver datos de mis ejercicios	El usuario debe poder acceder a datos de la puntuación obtenida en sus ejercicios resueltos	Alumno	Media
Ver estadísticas de mis ejercicios	El usuario debe poder acceder a datos estadísticos como ser la cantidad de intentos o pistas necesitadas.	Alumno	Media
Reportes por lección	El sistema debe brindar un conjunto de reportes de los alumnos: puntuación, tiempo, intentos, pistas necesitadas.	Docente Administrador	Alta
Configuraciones			
Configurar valores	El sistema debe permitir configurar valores como por ejemplo, puntajes según nota, porcentaje mínimo de aprobación, porcentaje mínimo para mostrar pistas	Docente Administrador	Media

Tabla 2 - Especificación de requerimientos

4.5 Requerimientos no funcionales

A continuación, se listan los requerimientos no funcionales recabados de las entrevistas.

Requerimientos de software no funcionales

- La aplicación debe ser un sistema web que soporte múltiples navegadores (Chrome, Firefox, Edge) y que tenga un diseño responsivo.
- Debe cumplir con los estándares básicos de usabilidad, las 10 heurísticas de Nielsen[22].
- El producto final debe contener un prototipo que contemple las diferentes categorías de ejercicios, realizando una carga inicial de lecciones de prueba y clientes con ambos perfiles (docentes y estudiantes).

Requerimientos de diseño

- El cliente plantea la necesidad de que el desarrollo de la aplicación se haga en un lenguaje multiplataforma, robusto y maduro.
- Utilizar una base de datos que sea libre.

Documentación

- El cliente solicita un manual de usuario del uso del sistema, tanto desde el punto de vista del alumno como del docente.
- Creación de una guía de instalación del producto y configuraciones básicas del sistema.

5. Diseño y arquitectura del sistema

En este capítulo se describe la arquitectura del sistema mediante los diagramas de despliegue y de componentes, el modelo de dominio y el modelo de base de datos diseñados.

5.1 Arquitectura del sistema

La arquitectura aplicada es cliente-servidor y el modelo es de aplicación distribuida. El servidor provee los recursos o servicios que son pedidos por el cliente. Está construida en tres capas bien definidas: cliente, servidor de aplicación que procesa los pedidos del cliente y el servidor de base de datos que almacena los datos necesarios.

5.1.1 Diagrama de despliegue

En este punto se muestra un diseño de la disposición física de los componentes para el despliegue de la aplicación.

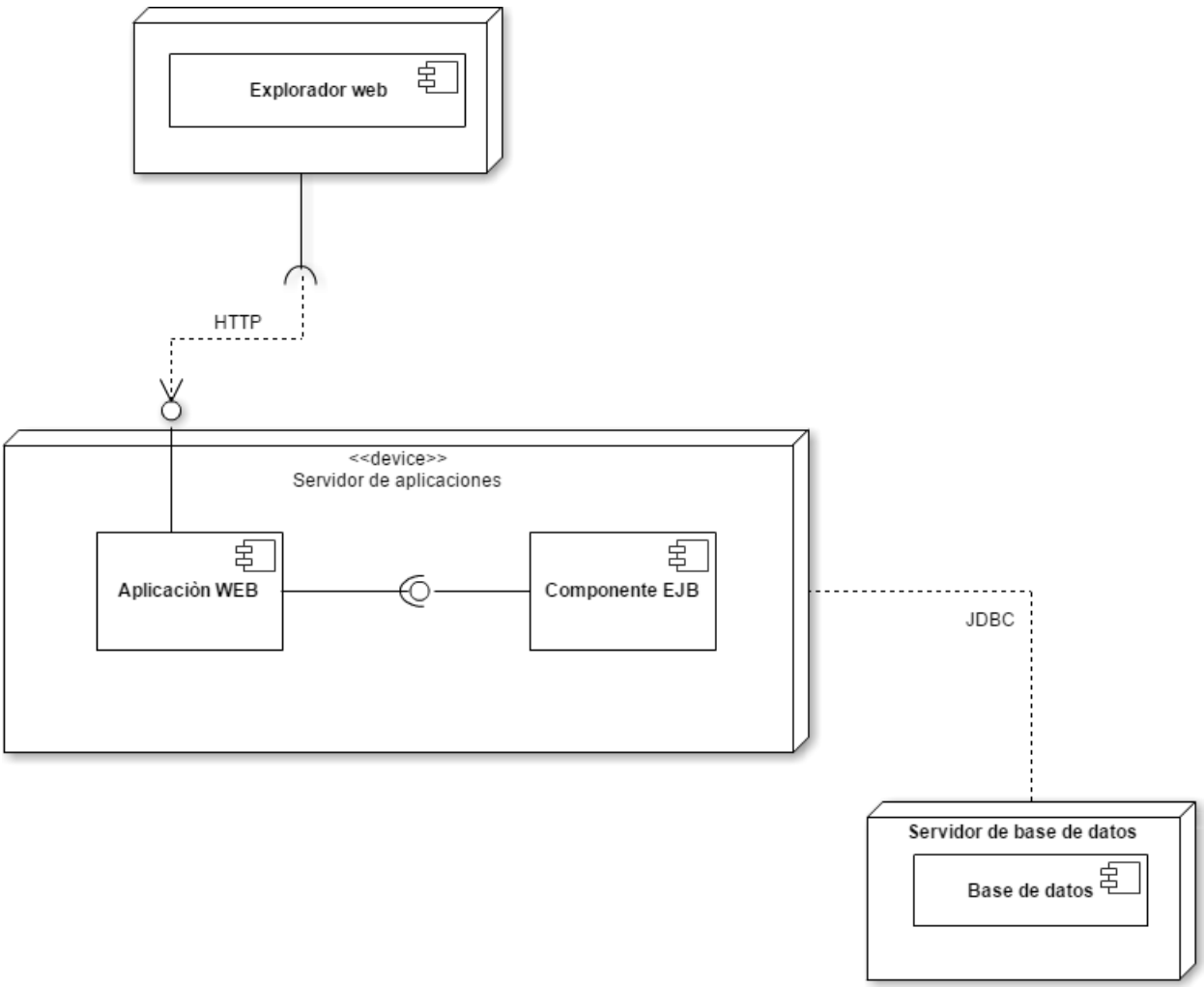


Figura 3 – Diagrama de despliegue

El explorador web es el componente que proporciona al usuario el acceso a la aplicación. Se comunican mediante el protocolo HTTP. El servidor de aplicaciones aloja el componente de software web que implementa el *front-end* del sistema. En tanto el componente EJB es quien se encarga de la comunicación con el servidor de base de datos siguiendo el protocolo JDBC.
El servidor de base de datos almacena la información necesaria.

5.1.2 Diagrama de componentes

Con el diagrama de componentes se presenta el modo de interacción entre los componentes de la aplicación

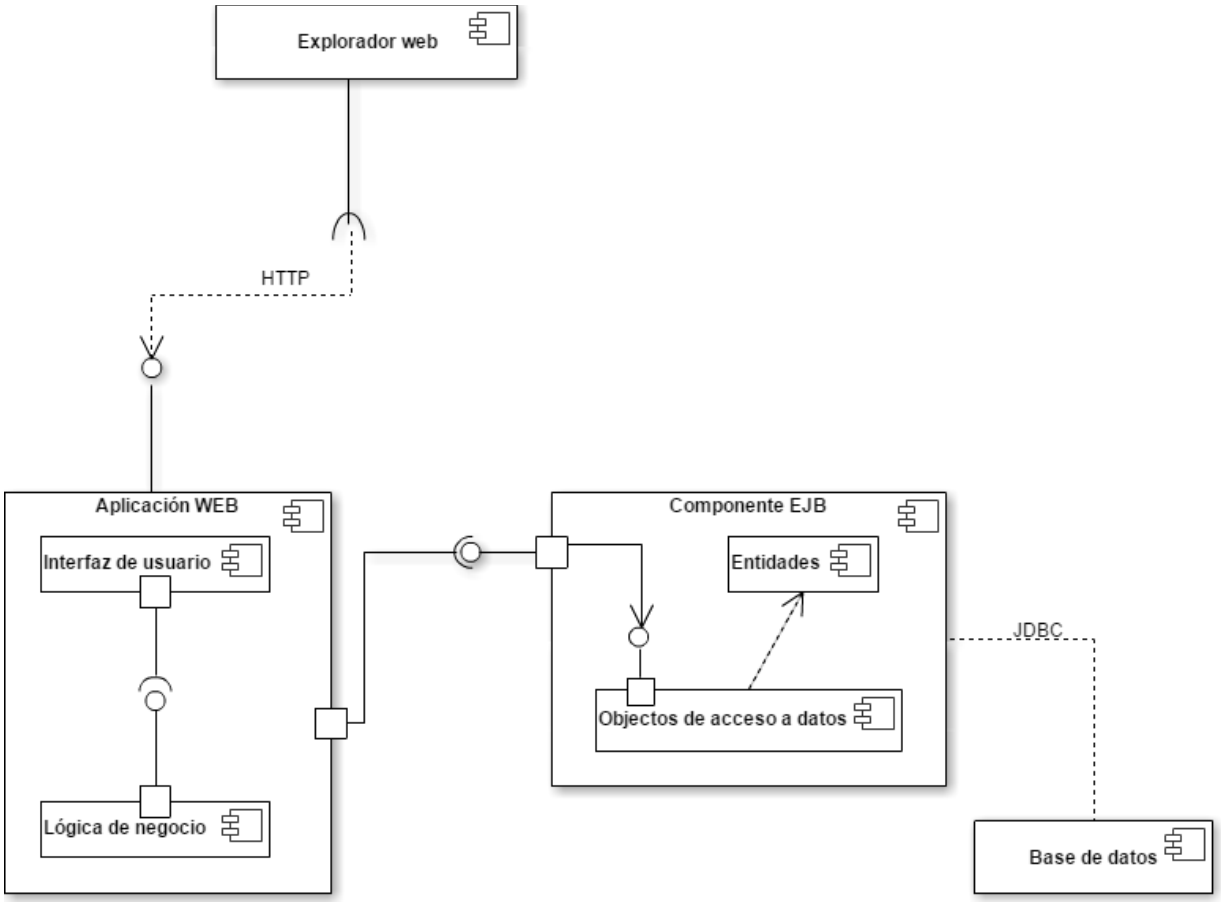


Figura 4 – Diagrama de componentes

5.1.3 Patrones arquitectónicos

Además del patrón de arquitectura ya mencionado anteriormente, programación por capas de la arquitectura cliente/servidor, se aplican otros patrones de diseño.

El patrón Modelo-Vista-Controlador (MVC) se aplica para tener un diseño que permita la reutilización de los componentes en futuros desarrollos y facilitar el mantenimiento de la

aplicación separando los conceptos. MVC consiste en separar los datos y la lógica de negocio de la interfaz de usuario y el módulo que gestiona los eventos y la comunicación.

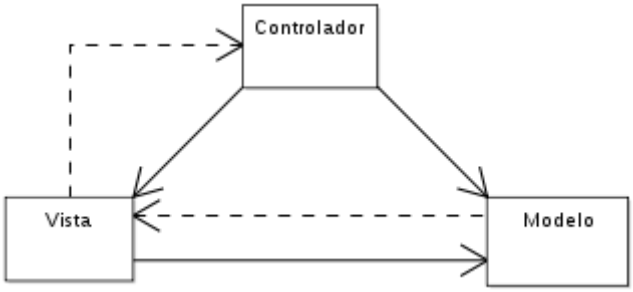


Figura 5 – MVC

El patrón Objeto de Acceso a Datos (DAO) es utilizado para encapsular la comunicación entre la aplicación y los dispositivos de almacenamiento de datos.

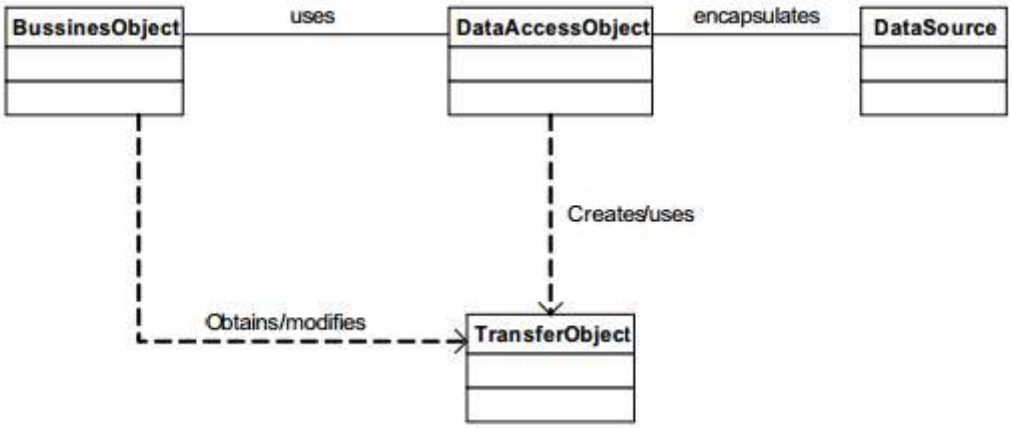


Figura 6 – DAO

5.2 Modelado del dominio

En esta sección se presenta el modelo de dominio parcial con los principales conceptos que representan la realidad planteada anteriormente. En el diagrama se observan dos partes principales, por un lado los conceptos relacionados con los usuarios y por otro todos los conceptos relativos a los ejercicios.

- Los usuarios pueden ser alumnos o docentes y son divididos por grupos. Un grupo está compuesto por un docente del grupo y un conjunto de alumnos. El docente es quien asigna las notas cuando el ejercicio es manual y es el responsable del grupo. Todos los usuarios tienen un rol asignado. El rol se define como un conjunto de permisos asociados. El alumno resuelve ejercicios propuestos por el docente y obtiene un puntaje.
- El ejercicio es parte de una lección y se le asigna una única categoría. Los elementos de la interfaz que posee el ejercicio son determinados por la categoría que tiene

asignada. Además, el ejercicio puede contener pistas que ayuden al alumno a resolver el ejercicio y un conjunto de soluciones.

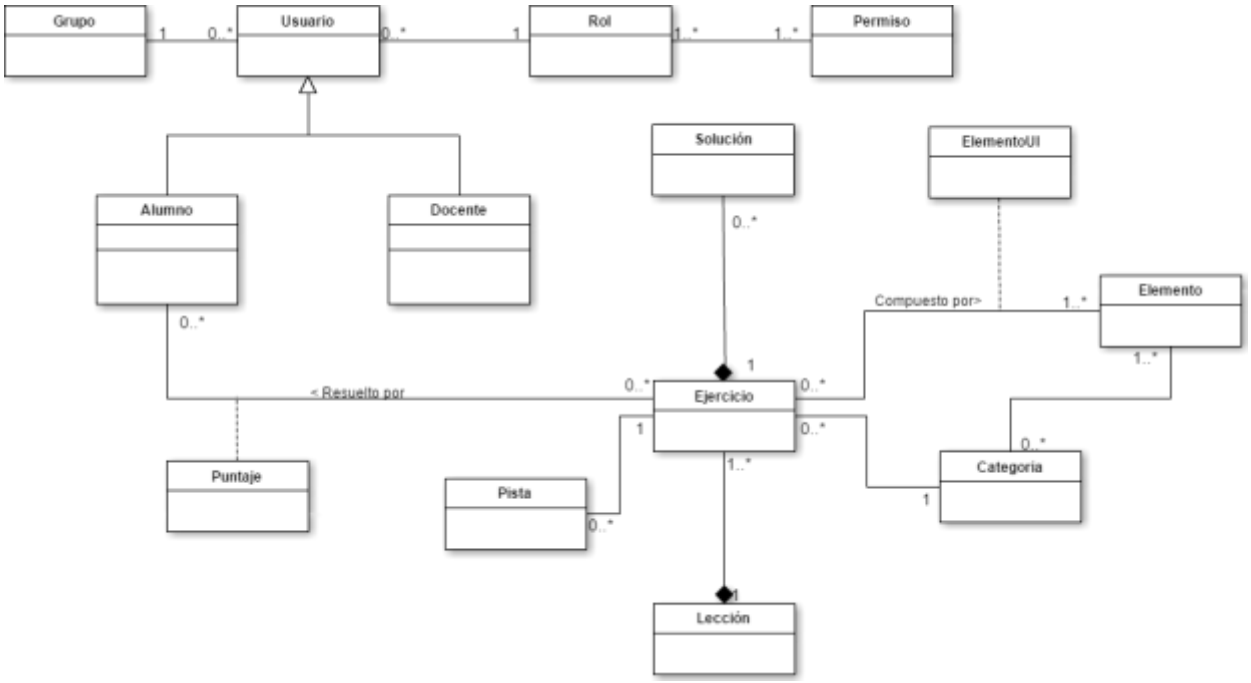


Figura 7 – Modelo de dominio

Restricciones del modelo:

- Los elementos (componente de interfaz) asociados al ejercicio tienen que estar incluidos dentro del conjunto de elementos asociados a la categoría a la que pertenece el ejercicio.
- Los grupos de usuarios están formados por un solo profesor y cero o más alumnos. El docente es el tutor del grupo.

5.3 Casos de uso relevantes

Se presenta a continuación un diagrama de casos de uso para ilustrar la relación entre los casos de uso más relevantes de la aplicación, seleccionados además por su alta complejidad y criticidad.

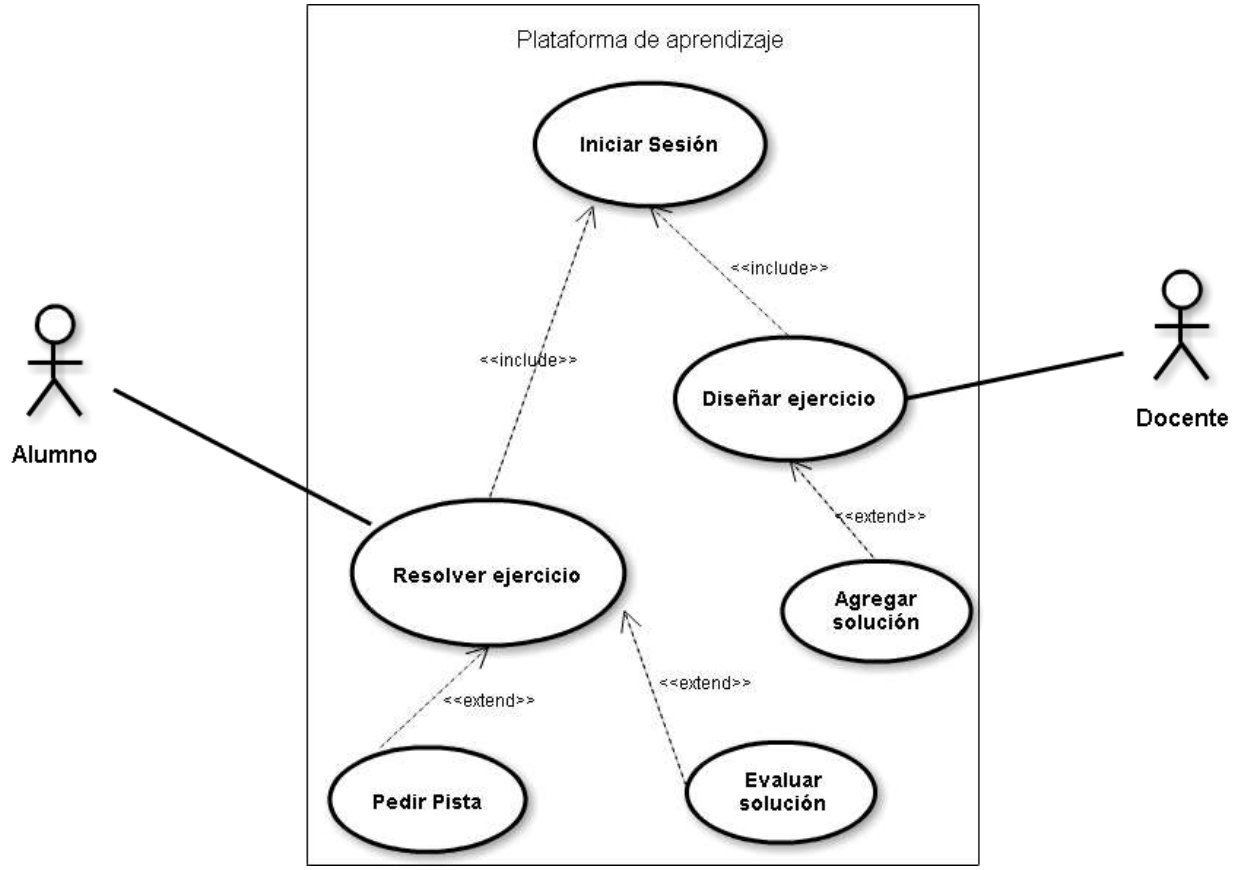


Figura 8 – Diagrama de casos de uso

5.4 Base de datos

Se presenta el diseño de la base de datos (se excluyen algunos campos de las tablas para simplificar el diagrama).

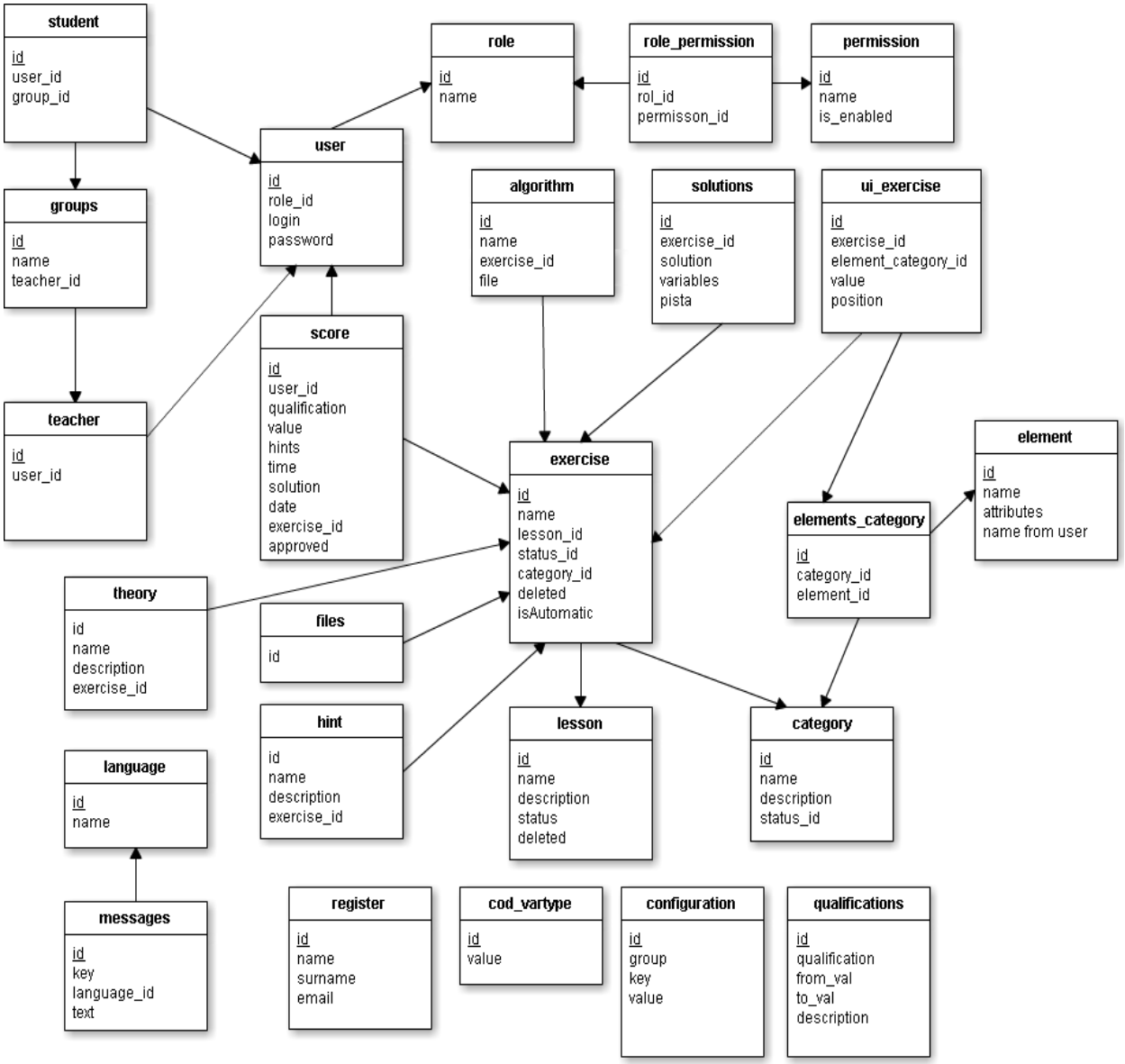

student
id
user_id
group_id
role
id
name
role_permission
id
rol_id
permisson_id
permission
id
name
is_enabled
user
id
role_id
login
password
groups
id
name
teacher_id
algorithm
id
name
exercise_id
file
solutions
id
exercise_id
solution
variables
pista
ui_exercise
id
exercise_id
element_category_id
value
position
score
id
user_id
qualification
value
hints
time
solution
date
exercise_id
approved
teacher
id
user_id
exercise
id
name
lesson_id
status_id
category_id
deleted
isAutomatic
element
id
name
attributes
name from user
elements_category
id
category_id
element_id
theory
id
name
description
exercise_id
files
id
hint
id
name
description
exercise_id
lesson
id
name
description
status
deleted
category
id
name
description
status_id
language
id
name
messages
id
key
language_id
text
register
id
name
surname
email
cod_vartype
id
value
configuration
id
group
key
value
qualifications
id
qualification
from_val
to_val
description

Figura 9 – MER

6. Implementación

En este capítulo se describen los *frameworks* y librerías utilizados/as para la implementación del proyecto, los complementos de terceros incorporados y se presenta la lógica de las principales funcionalidades.

6.1 Elección de tecnologías

Teniendo en cuenta las características del proyecto y los requerimientos funcionales y no funcionales descritos en el capítulo anterior se llevó a cabo una investigación con el fin de relevar las tecnologías actualmente utilizadas para el desarrollo de aplicaciones web.

6.1.1 Almacenamiento de datos

Para almacenar los datos de la aplicación se utiliza el motor de base de datos MySQL por ser un requerimiento del cliente que la base fuera libre. MySQL es un motor de base de datos relacional open source, en nuestro proyecto se utiliza la versión 5.6.26 – MySQL Community Server (GPL). Como manejador de la base de datos se utiliza XAMPP versión 3.2.1. XAMPP es un servidor multiplataforma que contienen el sistema de gestión de base de datos MySQL, un servidor web Apache y los intérpretes para los lenguajes PHP y Perl. Se selecciona XAMPP por ser una herramienta de código abierto y gratis. Además, es fácil de instalar y usar. El equipo tiene conocimiento en el uso de la herramienta.

6.1.2 Lenguaje de programación

El lenguaje de programación utilizado en el proyecto es Java EE 8. El mismo fue seleccionado para cumplir con los requerimientos del cliente, en particular que la herramienta sea portable y mantenible. Además, tiene una apariencia moderna y desplegada en un servidor web. Java cumple con estos requisitos, además es un lenguaje orientado a objetos que facilita la construcción de sistemas de información en capas. Otras características que nos llevaron a elegir Java son la simpleza, robustez y madurez del lenguaje, acompañado de la gran cantidad de *frameworks* disponibles y la existencia de una gran comunidad trabajando con este lenguaje.

6.1.3 Servidor de aplicación

El servidor de aplicaciones a utilizar debe ser una implementación de la especificación J2EE (Java Enterprise Edition).

Algunos de los servidores más utilizados son GlassFish, Jboss y Apache, existen otras implementaciones de servidores basados en los anteriormente mencionados.

Para el despliegue de nuestro proyecto se usa el servidor Wildfly de RedHat (en versiones anteriores conocido como Jboss), por ser una herramienta de código abierto robusta, muy utilizada en el mercado y por contar con gran soporte en la comunidad. Los integrantes del equipo cuentan con conocimientos previos y experiencia en cuanto al despliegue de aplicaciones y configuración del servidor, por todos estos motivos se decide su utilización para desplegar la aplicación.

La versión del servidor Wildfly usada es la 8.2.0.Final, última versión estable al comienzo del proyecto liberada el 20 de noviembre de 2014. Se presenta una breve descripción de la estructura del servidor Wildfly para introducir conceptos necesarios al momento de hacer el despliegue de la aplicación.

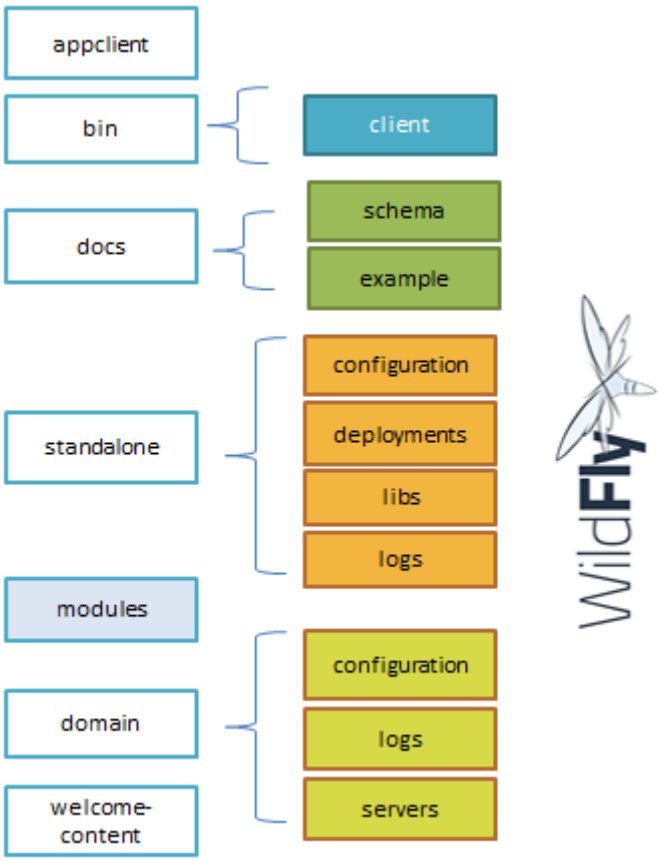


Figura 10 – Estructura WildFly

El despliegue de la aplicación se hace dentro de la carpeta standalone/deployments. La configuración del acceso a base de datos se hace en el archivo *standalone.xml* que se encuentra en la carpeta standalone/configurations. Además, el servidor guarda logs de acciones por día y los almacena en la carpeta standalone/logs

Es de utilidad la configuración de un usuario para poder acceder a la consola de administración del servidor, la carpeta bin contiene los scripts necesarios para hacer todas las configuraciones iniciales del Wildfly.

6.1.4 Entorno de desarrollo integrado

El entorno de desarrollo integrado (IDE) usado para el desarrollo de la aplicación es Eclipse, versión Mars (4.5). Eclipse es uno de los entornos de desarrollo de aplicación más usados, de código abierto y multiplataforma. La flexibilidad de la arquitectura de plugins permite equipar una instalación particular de eclipse con una combinación de elementos como ser el soporte de Maven y el control de versiones. Para facilitar nuestro desarrollo se usa Jboss Tools para el manejo del servidor de aplicaciones Wildfly, Subclipse para el versionado del

código y plugins de GWT (GWT Designer y Google Web Toolkit Tooling) que facilita el despliegue del proyecto GWT.

6.1.5 Herramienta de control de cambios.

Para el control y versionado del código se usa la herramienta Apache Subversion (SVN) de código abierto y versión libre. SVN se basa en el concepto de revisiones por cada vez que se guardan cambios. Siguiendo las políticas de buenas prácticas de gestión de la configuración se crean tres componentes en el primer nivel de directorios del repositorio: Trunk, Tags y Branches

Trunk
Es la línea principal del desarrollo, contiene la versión actual del trabajo sobre el que se van realizando las modificaciones.

Tags
Directorio del repositorio donde se guardan fotos de las revisiones del proyecto, por ejemplo del cierre de una versión. Se respetan las buenas prácticas que sugiere no modificar nunca un tag creado.

Branches
Son las ramas de evolución paralelas al trunk, se utilizan en caso de que sea necesario bifurcar el código por ejemplo para realizar pruebas.

El cliente SVN usado a lo largo del proyecto es el proporcionado por Assembla. Assembla es una empresa que proporciona herramientas de colaboración en la nube para organizar y administrar proyectos. Se crea un espacio de trabajo SVN para alojar Sophói con visibilidad restringida a los miembros del proyecto. El seguimiento de horas trabajadas y tareas cumplidas también es administrado por la herramienta.

Para manejar las versiones desde el IDE Eclipse se usa el complemento Subclipse ya incorporado en la distribución de Eclipse Mars.

6.1.6 GWT

Para cumplir con el requerimiento de que el docente pueda diseñar todo el ejercicio, incluyendo la interfaz, agregar imágenes, campos de texto, combos para seleccionar valores, grafos y otros componentes, sin necesidad de recompilar la aplicación, es necesario desarrollar la interfaz con un *framework* que permita generar los elementos del diseño de forma dinámica. Además, es de gran valor que las mismas se adapten de la mejor manera posible a los diferentes dispositivos y resoluciones de pantallas sin generar un aumento significativo de dificultad en la implementación o un importante incremento en las horas de desarrollo.

Se hace una investigación para seleccionar el *framework* que nos permitiera cumplir con los objetivos establecidos. De la investigación inicial se concluye que se podría utilizar Vaadin o GWT. Consultando manuales y sitios de internet para desarrolladores se hace una comparación entre ambos para seleccionar el más adecuado para nuestro proyecto. En

principio el *framework* seleccionado fue Vaadin. Se implementó un prototipo con dicho *framework* pero los problemas encontrados llevaron a descartarlo y elegir GWT.
A continuación, se explican las características principales de ambos *frameworks* y se explican los motivos que llevaron a elegir GWT.

Vaadin

Vaadin es un *framework* Java de código abierto para la creación de aplicaciones web. Esta herramienta fue seleccionada para estudio por soportar dos modelos de programación diferentes: del lado del servidor y del lado del cliente. El modelo del lado del servidor es el que permite desarrollar la interfaz por código como si fuera una aplicación de escritorio. La lógica para el control de la interfaz se ejecuta en el servidor web junto con la lógica de negocio. Por otro lado, el cliente se ejecuta como JavaScript en el navegador e incluye Google Web Toolkit (GWT) que proporciona un compilador de Java a JavaScript que ejecuta en el navegador.

Se implementa un prototipo para evaluar la herramienta que consiste en la creación de una pantalla que se comunique con todas las capas de la arquitectura planteada, obtenga y despliegue algún dato de la base de datos. Como puntos a favor se destacan que el diseño de la interfaz es muy amigable, respetando los conceptos de usabilidad, la incorporación de elementos a la interfaz de usuario es sencillo, siguiendo la metodología de las aplicaciones de escritorio (swing por ejemplo) y la documentación en cuanto a éste punto es muy útil.

Las dificultades surgidas durante el desarrollo del prototipo nos llevaron a descartar el uso de este *framework*. A grandes rasgos, no fue posible separar y comunicar la lógica de negocio de la capa de presentación en dos proyectos diferentes. También se encontraron problemas para acceder a un proyecto EJB desde el proyecto Vaadin. La documentación existente en este punto no fue de ayuda debido a que “El libro de Vaadin”, como se denomina el manual, se encontraba desactualizado al momento de implementar el prototipo. Por lo que los ejemplos allí presentados no coinciden con la versión del producto usado para prototipar. Se discute la posibilidad de desarrollar un solo proyecto sin separar el *front-end* del *back-end*, se decide seguir adelante con el diseño de la arquitectura prevista e investigar otros *frameworks*.

A pesar de todas las facilidades que ofrece Vaadin en cuanto a la programación de las pantallas, los problemas de configuración y comunicación con otros componentes nos llevaron a construir un prototipo con la herramienta GWT.

GWT – Características y arquitectura

Google Web Toolkit (GWT) es un *framework* Java lanzado en 2006 que tiene como objetivo utilizar las herramientas que provee java para construir aplicaciones AJAX independientes del navegador. GWT transforma el código Java a JavaScript.
GWT provee tres estrategias de implementación dependiendo del modo de comunicación con el servidor:

- GWT-RPC
- Recuperación de datos JSON vía HTTP
- Peticiones de dominio cruzado para JSONP.

Se detalla la estrategia GWT-RPC por ser la que mejor se adecúa a nuestro sistema. GWT-RPC tiene la característica de que todas las llamadas hechas desde una página HTML al servidor son asincrónicas, es decir que el cliente continúa con sus tareas mientras se procesan

los llamados, cuando la llamada asíncrona es completada se ejecuta el código especificado para el fin de la llamada. Esta metodología es la característica principal de los llamados AJAX.

El *framework* GWT-RPC facilita el intercambio de objetos java entre cliente y servidor a través de HTTP. Se trata como un servicio al código del lado del servidor que es invocado desde el cliente, la implementación de estos servicios se basa en la arquitectura Servlet de Java. El cliente usa una clase proxy generada automáticamente para hacer llamadas al servicio. GWT se encarga de que los objetos que se intercambian sean serializables, ya sean los argumentos del llamado como el valor de retorno.

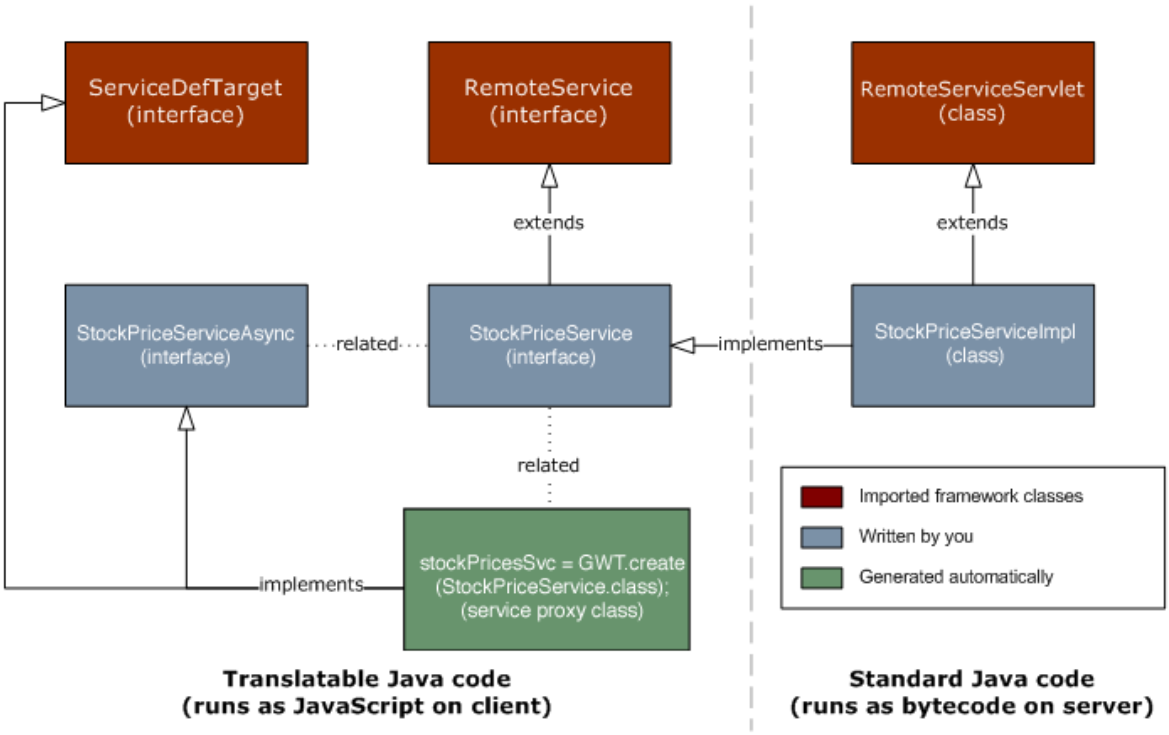


Figura 11 – Arquitectura GWT

Como se observa en la *Figura 11 – Arquitectura GWT*, del lado del servidor se crea la clase que extiende *RemoteServiceServlet* y que implementa la interfaz del servicio (*StockPriceService* en el ejemplo) que extiende *RemoteService*. Además, se define la interfaz asincrónica *StockPriceServiceAsync* del servicio, para poder ser llamado del lado del cliente.

En nuestra implementación definimos siete servicios:

Servicio	Descripción
LessonServiceImpl	Servicio donde se definen operaciones para el manejo de las lecciones y ejercicios. Implementa las principales funciones para el procesamiento de los ejercicios *excutePistas*, *executeEnd* y *executeNewResult* y las operaciones de alta, modificación y baja.
LoginServiceImpl	Servicio que implementa las funcionalidades básicas de login y logout de los usuarios.
MessagesServiceImpl	Creado para manejar los mensajes informativos desplegados al usuario en la aplicación.
ReportServiceImpl	En este servicio se manejan las operaciones relacionadas a estadísticas, datos de usuarios, y resoluciones de ejercicios.
SessionServiceImpl	Servicio que maneja los elementos guardados en la sesión provista por Shiro (se define en la sección 6.1.8). Se implementa para poder acceder a datos en la sesión del usuario en el servidor por parte del cliente.
UserServiceImpl	Implementa las operaciones relativas al manejo de los usuarios: alta, baja, modificación y listado.
UtilitiesServiceImpl	Servicio que contiene operaciones complementarias, por ejemplo todos los convertidores de entidades a objetos de transferencia de datos (DTO) y viceversa.

Tabla 3 – Servicios

Despliegue de la aplicación GWT

Para publicar la aplicación se empaqueta en un WAR pudiendo ser desplegada en cualquier contenedor de servlets, en nuestro proyecto usamos Wildfly como contenedor. Dentro de la SDK (Software Development Kit) de GWT se encuentra el componente que traduce código desarrollado en java a código JavaScript optimizado.GWT genera un archivo *noncache.js* cuyo propósito es decidir que permutación generada debe ser descargada. Cada permutación individual consiste en la implementación de la aplicación específica para un navegador, idioma, etc. Éste es mucho más código que el código bootstrapping simple, y por lo tanto necesita ser almacenado en caché para que la aplicación responda rápidamente.Estos son los archivos *cache.html* generados por el compilador GWT.Cuando se recompile la aplicación, los usuarios descargarán el archivo *nocache.js* como de costumbre, pero este le indicará a sus navegadores que descarguen un nuevo archivo cache.html con las nuevas características de la aplicación. Esto también se almacenará en caché también para la próxima vez que cargan su aplicación.

6.1.7 Otros frameworks

Hibernate

El *framework* de software libre Hibernate es una herramienta de mapeo objeto-relación que facilita el mapeo entre elementos de una base de datos y el modelo de objetos de la aplicación Java. En nuestra implementación el mapeo se hace mediante las anotaciones en los beans de las entidades. Hibernate implementa como parte de su código la especificación JPA.

Utilizando Hibernate se puede reducir el tiempo de desarrollo que se hubiera gastado en el manejo de SQL y JDBC puro, es por este motivo y dado la experiencia del equipo con Hibernate que se decide usar dicho *framework*. La versión utilizada es la 4.3.8 Final

GWT-JSNI

Para integrar JavaScript con el código Java, GWT proporciona JavaScript Native Interface (JSNI) que es una combinación de Java nativo y JavaScript. JSNI permite llamar y crear funciones JavaScript dentro del código Java y viceversa. Con esta metodología se carga en nuestro proyecto los componentes JS de terceros Ckeditor y JointJS que se describen más adelante en este documento.

Gquery

Gquery, también conocida como GwtQuery, es una librería similar a Jquery desarrollada específicamente para integrarse con GWT. Al igual que Jquery es una biblioteca JavaScript rápida, pequeña. Sus principales funcionalidades son el manejo de código HTML dinámico, el manejo de eventos, animación, y Ajax.

6.1.8 Componentes externos

Para el desarrollo del proyecto fue necesario utilizar componentes implementados por terceros, que ayudaron a potenciar la herramienta. Como es común al trabajar con tecnologías Java, existe una amplia gama de frameworks dedicados a resolver problemas particulares. Por ejemplo, manejar la sesión de un usuario o crear pantallas amigables y atractivas. La elección se centró principalmente, por requisito del cliente, en que fueran frameworks open source o que no requieran pago de licencia. En esta sección se detallan los frameworks utilizados.

GWT-Bootstrap

GWT estándar brinda pocos componentes atractivos para el diseño de la interfaz, por este motivo se investiga el uso de algún complemento que mejore dicha falencia, identificándose como tal GWT-Bootstrap.

GWT-Bootstrap es una herramienta de código abierto que combina el uso de GWT con Twitter Bootstrap creado por Mark Otto y Jacob Thornton. Para el desarrollo del proyecto se usa la versión 2.3.2. Bootstrap es un framework CSS y JavaScript creado por Twitter con la finalidad de diseñar interfaces adaptativas, es decir, que se adapten al dispositivo y a la resolución de la pantalla de forma automática.

CKeditor

Es un editor de texto implementado en JavaScript que brinda las funcionalidades de un editor de texto clásico generando código HTML. Entre sus funcionalidades se destaca la

posibilidad de agregar imágenes, dar formato básico a un texto, agregar numeración, sangrías y viñetas.

CKeditor es un producto de código abierto creado por CKSource, la versión utilizada es la 4.5.11. En nuestra solución es incorporado el editor para poder ingresar la descripción de las lecciones y ejercicios al momento de crearlos.

Apache Shiro

Complemento de código abierto que provee funcionalidades para gestionar la autenticación, autorización, encriptado y manejo de sesiones de usuario. Shiro está diseñado para ser intuitivo y fácil de usar pero robusto en cuanto a seguridad. En nuestra implementación es usada la versión 1.2.4 configurarlo mediante un archivo XML.

Está configurado para chequear la autenticación en la base de datos del proyecto y guardar una sesión del lado del servidor, que contiene datos del estudiante de utilidad para la plataforma.

JointJS

Para el diseño de grafos se utiliza el complemento HTML5 JointJS que provee funcionalidades para visualizar e interactuar con diagramas, gráficos y grafos, entre otros elementos. Está altamente orientado a eventos, puede reaccionar con cualquier evento que sucede en el interior del panel donde se agregan los elementos. Está desarrollado con JavaScript por Client.io y es de código abierto. En nuestro desarrollo es usado para crear los nodos y transiciones de los grafos incorporados en los ejercicios.

Jaro

Cuando un estudiante ingresa texto en los campos de respuesta puede cometer errores de tipeo u ortográficos que llevan a que la corrección automática indique que la respuesta no es correcta, cuando en realidad lo es. Para que la aplicación pueda detectar estos tipos de error y dar la respuesta como correcta se investigó el uso de algoritmos para determinar distancias entre cadenas de caracteres. La distancia entre una cadena y otra permite determinar cuán diferente son. Los algoritmos investigados fueron Jaro–Winkler Distance[22] y Levenshtein Distance[23]. Tras la investigación de estos algoritmos se decidió que Jaro era el adecuado para este problema en particular, porque es un algoritmo recomendado para trabajar con cadenas de caracteres cortas, detecta permutaciones y omisión de caracteres.

La distancia en Jaro se define como:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

donde *m* es la cantidad de caracteres que coinciden y t la cantidad de transposiciones.

William Cohen junto a Pradeep Ravikumar y Stephen E. Fienberg definieron un conjunto de herramientas para Java que aglomeran varios algoritmos de medición de distancia entre cadenas de caracteres, el cual es descrito en el artículo "*A Comparison of String Metrics*

for Matching Names and Records". Para nuestro proyecto se utilizó una adaptación de la implementación del algoritmo Jaro–Winkler propuesta por ellos.

El algoritmo Jaro se implementó en el método *compare* de la clase *Algoritmos*. Es un método público por lo que es posible accederlo desde otras clases y archivos JavaScript. El método recibe dos cadenas de caracteres y retorna un número entre 0 y 1 que indica cuán similares son las cadenas. Si el valor obtenido es 1 significa que las cadenas son iguales y si es 0, que son totalmente distintas.

En los JavaScript incluidos para la corrección automática de ejercicios se considera un valor de retorno del algoritmo Jaro mayor o igual a 0.85 como indicador de cadenas coincidentes.

Cadena 1	Cadena 2	Valor Jaro
bariavle	variable	0.83
dias	dia	0.91
Setiembre	sePtIemVre	0.89
distinto	diferente	0.64
incorrecto	correcto	0.93
traspuesta	tarspuesta	0.96

Tabla 4 - Comparativa Jaro

El uso de este algoritmo, si bien ayuda a evitar las correcciones erróneas por errores ortográficos también puede dar resultados como correctos cuando no lo son (falsos positivos).

6.2 Diagrama de tecnologías

Luego de haber presentado el conjunto de herramientas seleccionadas para llevar a cabo la implementación de la aplicación se actualiza el diagrama de despliegue.

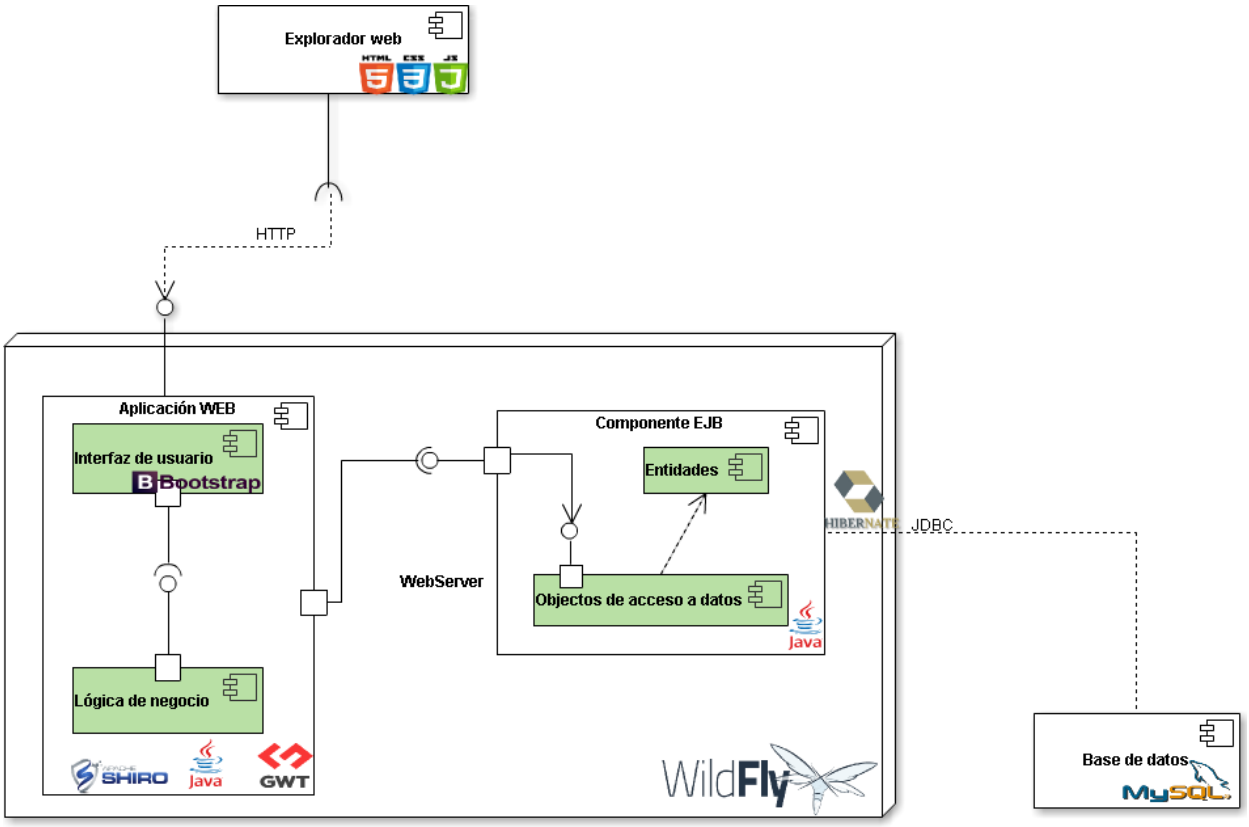


Figura 12 - Tecnologías

6.3 Estructura de los proyectos

El proyecto es empaquetado dentro de un Enterprise Application Archive (EAR) que está compuesto por un módulo Enterprise JavaBeans (EJB) y un módulo Web Application Archive (WAR).

El módulo EJB llamado appEJB es el encargado de la comunicación con la base de datos, está compuesto por los controladores y sus correspondientes interfaces además de las entidades del modelo. Incluye las librerías de hibernate-jpa 2.1, el conector mysql-connector y las librerías necesarias para el uso de Apache Shiro.

El módulo WAR llamado gwtWEB contiene la lógica de la aplicación necesaria para construir la interfaz usando GWT, es quien reacciona ante los eventos del navegador, los procesa y si corresponde, se comunica con el componente EJB para obtener o actualizar datos de la base de datos.

Estructura de paquetes del componente WEB GWT

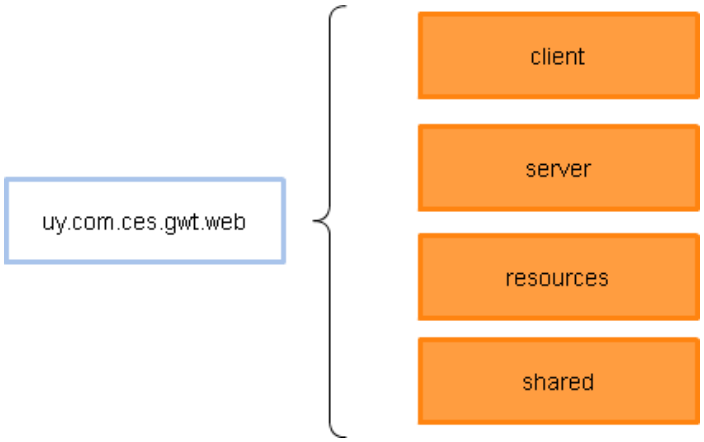


Figura 13 - Estructura gwtWEB

En la carpeta *shared* se alojan los elementos DTO a ser compartidos entre el lado del cliente y el lado del servidor de GWT. En *server* se encuentran los servicios ya especificado en la sección 6.1.6, además se definen clases auxiliares que implementan lógica de negocio necesaria del lado del servidor de GWT. Por otro parte, en el paquete *client* se encuentran los archivos fuentes necesarios del lado del cliente, concretamente en nuestro caso se alojan los archivos java UIBinder utilizados para la construcción de la interfaz. Además, como se especificó en la arquitectura GWT-RPC la definición de las interfaces e interfaces remotas son creadas en el paquete del cliente. Por último, en la carpeta resources se encuentran los archivos JavaScript con los algoritmos de resolución de los ejercicios predefinidos.

Dentro de la carpeta uy.com.ces.gwt.web se encuentra el archivo gwtWEB.gwt.xml que es el módulo punto de entrada de la aplicación. En este archivo se especifican las configuraciones, por ejemplo para qué navegadores se va a compilar el código, cuál es el módulo java punto de entrada de la aplicación y cuáles son los paquetes que contienen archivos fuentes a compilar y traducir a JavaScript (shared y client)

6.4 Implementación de la solución

6.4.1 Extensibilidad

Como parte de la extensibilidad de la plataforma es posible agregar nuevas categorías. En este caso sí es necesaria la intervención de un desarrollador con conocimientos de base de datos, ya que primero hay que insertar una nueva tupla en la tabla *Category* especificando el nombre y la descripción, luego en la tabla *elements_category* asociar los elementos que pueden contener los ejercicios de esa categoría. Como los cambios solamente implican modificaciones en la base de datos no es necesario reiniciar el servidor ni volver a hacer el

despliegue de la aplicación. Cerrando sesión y volviendo entrar quedan reflejados los cambios. Si se desea contar con corrección automática para esta nueva categoría se debe modificar la clase *LessonServiceImpl*, para incorporar la lógica de corrección.

6.4.2 Interfaz

Uno de los mayores potenciales de la herramienta es la capacidad de generar la pantalla de los ejercicios en tiempo de ejecución, sin depender de un programador para que incorpore nuevos ejercicios.

Por este motivo y como ya se explicó anteriormente la implementación de la interfaz se diseñó con la herramienta GWT. Sus componentes UIBinder brindan la flexibilidad necesaria para la construcción de interfaces.

En la *Figura 14 - Mockup ejercicio* se muestra un *mockup* de la pantalla donde se dibujan los ejercicios.

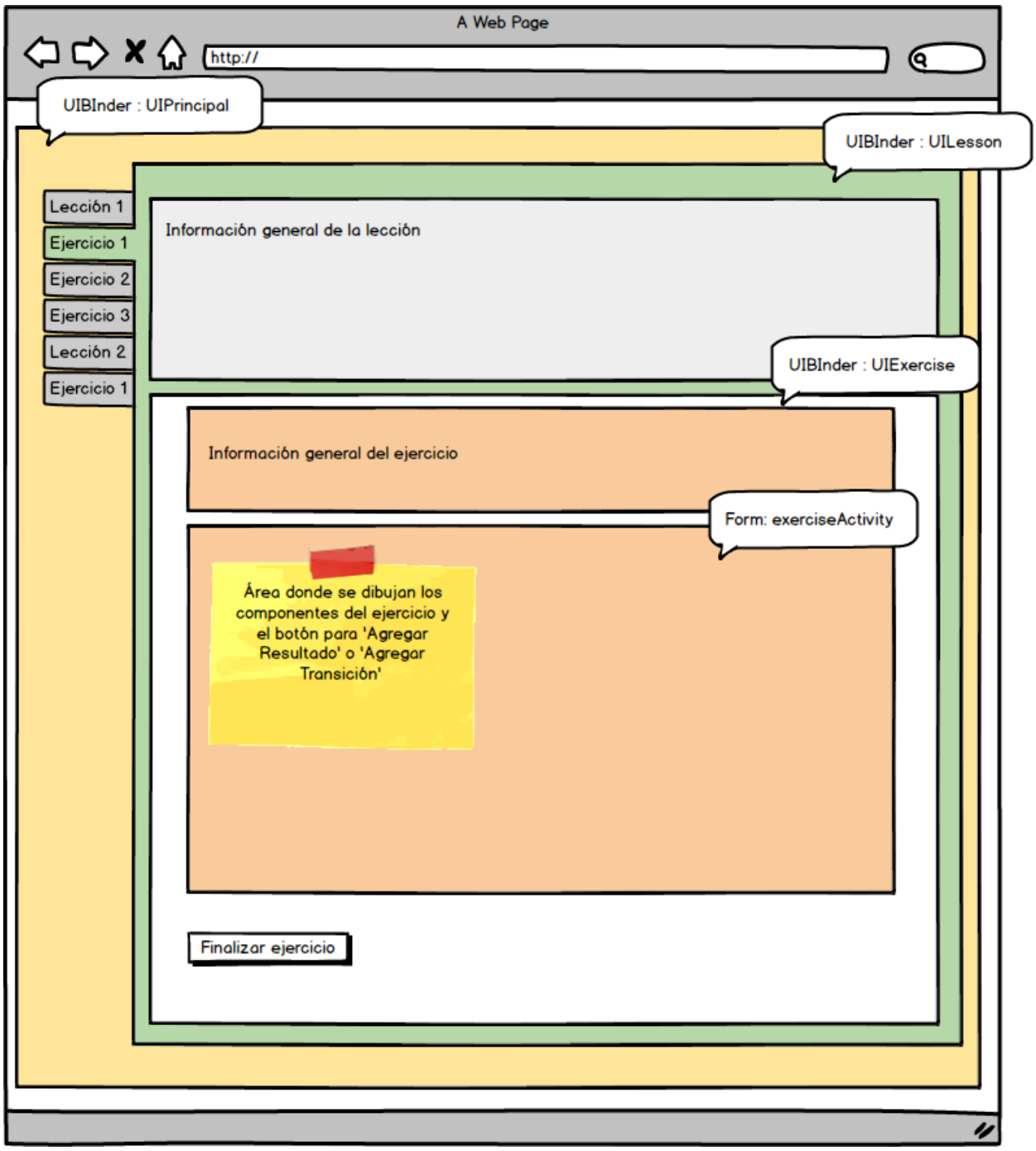


Figura 14 - Mockup

El diseño de la actividad del ejercicio es definido por un conjunto de componentes que el usuario, profesor o administrador, agrega para construir la página del ejercicio. Los componentes de los que se dispone para agregar en el área de “exerciseActivity” dependen de la categoría a la que pertenece el ejercicio.

Los componentes predefinidos están almacenados en la tabla *Elements* de la base de datos. Las características del componente que pueden ser configuradas se almacenan como una lista de atributos. Como la herramienta está diseñada de tal manera que se pueda extender, un programador puede agregar nuevos componentes a la herramienta, para lo cual

tiene que definir el componente en la tabla *Elements* y modificar la clase *AbstractActivity,* que es la encargada de dibujar el formulario *exerciseActivity* del ejercicio.

Los componentes pueden separarse en dos categorías, los que son informativos y los receptores. Los informativos son elementos que muestran texto o imágenes, mientras que los receptores reciben los datos a ser ingresados por el alumno y posteriormente corregidos por el docente. Los receptores dependiendo de la categoría del ejercicio pueden soportar distintos tipos de datos.

Componente	Descripción	Atributos configurables
Texto	Representa un componente HTML label.	• text: texto del componente label
Imagen	Componente que permite agregar una imagen a la actividad, la imagen puede ser cargada desde una carpeta local.[2] La imagen tiene que tener formato png	• url: dirección de la imagen
Variable a ingresar	Es desplegado en la actividad como un componente HTML input acompañado de un label.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Lista de valores (combo box)	Representa un elemento HTML select. Los elementos se ingresan separados por coma.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Clase de equivalencia	Se dibuja en la actividad como un elemento input HTML y la etiqueta que lo describe. Se diferencia del componente *variable a ingresar* en la forma que se procesa internamente el dato. Se ingresan los rangos de la clase de equivalencia separados por coma.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Representante de Clase de eq.	Se dibuja en la actividad como un elemento input HTML y la etiqueta que lo describe. Se diferencia del componente *variable a ingresar* en la forma que se procesa internamente el dato.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable

[2] A pesar de que el componente de descripción del ejercicio (CKEditor) permite agregar imágenes, no es posible agregarlas desde una ruta local, por este motivo se decidió incorporar un componente de imagen propio.

Grafo	Es dibujado en la actividad como campos para ingresar el nodo inicial, final, la acción, evento y guarda de una transición del grafo. Al agregar las transiciones se va armando el grafo resultado. Los cinco campos pueden ser dibujados como un comboBox con valores sugeridos o un input HTML.	• init: posibles valores del nodo inicial de una transición. • end: posibles valores del nodo final de una transición. • action: posibles valores que representan la acción • event: evento que produce el pasaje de un nodo a otro. • guard: condición que se debe cumplir para pasar del nodo inicial al final.
Árbol	Similar al grafo, se diferencia por no poseer evento y acción.	• init: posibles valores del nodo inicial de una transición. • end: posibles valores del nodo final de una transición. • guard: condición que se debe cumplir para pasar del nodo inicial al final.

Tabla 5 - Componentes

Tipo de dato	Descripción
Numérico	Valida que el campo sea numérico.
Enumerado	Lista de valores separados por coma.
Texto Libre	Campo que puede contener cualquier valor.
Tipo de la variable	Se define una lista de posibles tipos de datos para una variable. Los tipos son cargados de la codiguera de la base.
Expresión regular	Utilizado por el docente para cargar una solución como una expresión regular.

Tabla 6 - Tipos de datos

6.4.2.1 Componente AbstractActivity

En la vista del alumno se dibuja dinámicamente la interfaz de la actividad diseñada por el profesor. El proceso es el siguiente: el alumno selecciona del menú lateral el ejercicio de una lección que va a resolver. La aplicación obtiene de la base los datos del ejercicio seleccionado. Se invoca el método *loadActivity* de la clase *AbstractActivity* que retorna el elemento HTML *form* que será cargado en el UIBinder UIExercise.

La clase AbstractActivity es la encargada de procesar la definición de un ejercicio y mapearla a un conjunto de componentes HTML a ser incorporados en el formulario de la actividad. Para cada componente existe un método encargado de dibujarlo en la interfaz.

Otra de las funcionalidades de la clase es la creación y manejo de eventos del botón de *agregar resultados*. Cada vez que se agrega un resultado se guardan los valores en la lista de resultados. La misma está compuesta por un Map que contiene los valores de cada componente del ejercicio, la clave es el identificador del ejercicio.

```
lista de resultados =  List<a>
a = lista de variables
variables = new HashMap<String key,Object value>();
      Key = nombre de la variable
      value = Object con el valor de la variable
            Value es un array donde
                  Value[0]= Valor/es de la variable.
                  Value[1] = Type de la variable
```

Figura 15 - Estructura de resultado

En el caso de que el ejercicio posea un árbol o grafo el botón de agregar resultado se convierte en agregar transición, tomándose como solución única del ejercicio todas las transiciones agregadas.

6.4.3 Corrección de ejercicios

La plataforma cuenta con un conjunto de algoritmos predefinidos en JavaScript y también permite al docente agregar sus propios archivos con las soluciones. En particular esta última es una mejor opción porque permite hacer correcciones más específicas, pero tiene como contra que se debe implementar el JavaScript, por lo que se requieren conocimientos básicos de programación por parte del usuario que crea el ejercicio.

Al momento de crear un ejercicio, si se elige corrección automática, el docente debe ingresar la solución del ejercicio. En la interfaz de creación de ejercicios se ingresan para cada componente receptor los posibles valores que puede tomar. Esta interfaz es similar a la que usa el estudiante para ingresar sus respuestas. Se asigna a cada componente una lista de valores que representan la solución en la base de datos. Si se desea, se puede agregar una pista asociada a cada solución, esta pista se mostrará cuando esta solución no esté marcada como correcta en el transcurso de un ejercicio.

La creación de ejercicios puede se encuentra detallada en el *Anexo I – Manual de Usuario*. Como ejemplo se muestra la creación de un ejercicio de la categoría “Identificación de Variables”.

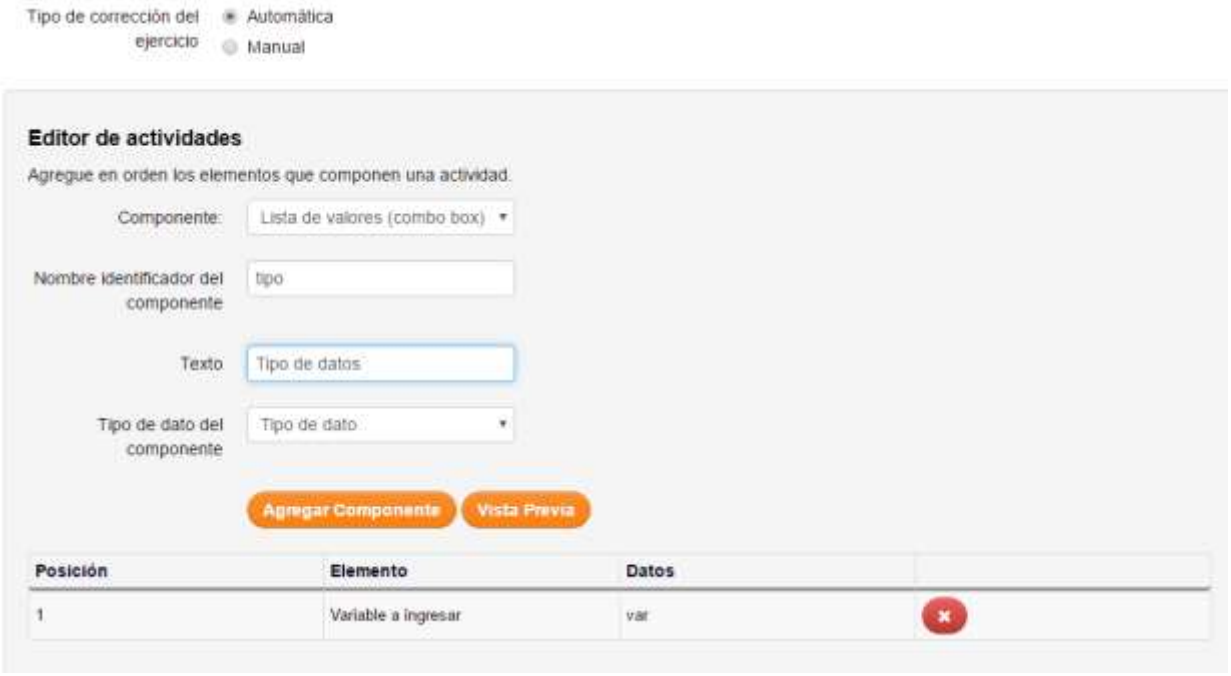


Figura 16 - Selección de componentes de la actividad

Figura 17 - Ingreso de soluciones

En la plataforma se dispone de un algoritmo de corrección automática por categoría de ejercicio. Estos algoritmos son genéricos y se basan en el tipo de componentes receptores que brinda la categoría, haciendo uso de su identificador para comparar con la solución. Lo que permite desvincular la solución del orden de los componentes en la pantalla y a su vez que los ejercicios tengan distinta cantidad de componentes de igual o distinto tipo, exceptuando las categorías que utilizan grafos o árboles.

Como se mencionó en la sección 6.1.8, estos algoritmos hacen uso del algoritmo Jaro para evitar errores de tipeo por parte del alumno. La implementación de Jaro proporcionada también puede ser utilizada en los JavaScript que implementa el docente.

Muchas veces los algoritmos utilizan caracteres especiales para dar un significado a la solución por lo que existe un conjunto de caracteres reservados que no pueden ser utilizados como parte de la solución. Estos son punto y coma (;), coma (,), llaves ({ }) y paréntesis curvos (( )) y rectos ([ ]).

7. Testing

El testing sobre la aplicación se hizo ejecutando pruebas de humo y misiones de testing exploratorio. Se ejecutaron pruebas de humo de todas las funcionalidades. Estas pruebas son llevadas a cabo por el desarrollador de cada funcionalidad. Se centran principalmente en verificar que se ejecuten correctamente los flujos principales de los casos de uso asociados a dicha funcionalidad.

Para profundizar en las pruebas se ejecutaron misiones de testing exploratorio sobre las funcionalidades centrales de la aplicación. Estas son la creación de lecciones y ejercicios desde el lado del docente y las funcionalidades relacionadas a la ejecución de ejercicios por parte del alumno. Como técnica para las pruebas se eligió el testing exploratorio[24] por ser más ágil que el testing planificado y ser más práctica para la ejecución de pruebas ya que en una sola sesión se alcanza un cubrimiento amplio de la funcionalidad. Además, por las características de nuestro grupo de trabajo, donde los testers son los desarrolladores, no fue necesario contar con una documentación extensa de las pruebas ni de los incidentes. Por eso no se usó una herramienta para gestión de incidentes, sino que se utilizó un documento de texto en que se registraron los incidentes con la evidencia correspondiente y se les asignó un estado. Se definieron 3 estados, abierto, corregido y cerrado y se clasificó los incidentes según su prioridad.

Se hizo un inventario con las funcionalidades a testear y se definieron misiones para cubrir estas funcionalidades. Las misiones definidas junto al total de incidentes encontrados en cada una, la cantidad de incidentes cerrados y la cantidad de incidentes abiertos pueden verse en la *Tabla 7 - Misiones*.

Los objetivos de las misiones son diseñar y ejecutar un ejercicio de cada categoría, esto llevó a su vez a probar la creación y modificación de lecciones y también a verificar que al aprobar un ejercicio se desbloquee el siguiente y se pueda navegar entre ellos. Para cada misión se ejecutaron varias sesiones, la cantidad varía según la misión. En algunos casos las sesiones fueron ejecutadas por distintos testers para tener un cubrimiento más amplio.

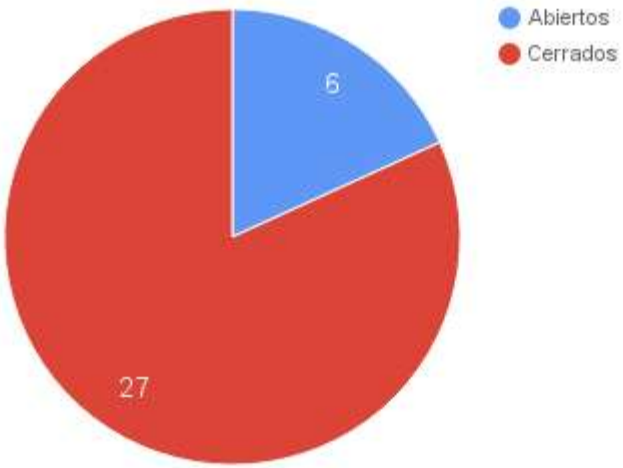
Misión	Objetivo	Incidentes	Cerrados
M1_CrearLeccion	Crear una lección	0	0
M1_IdentVars	Crear un ejercicio de la categoría Identificación de variables	0	0
M1_EqClass	Crear un ejercicio de la categoría Clases de equivalencia.	2	2
M1_Árbol	Crear un ejercicio de la categoría Árbol de decisión	2	2
M1_Máquina	Crear un ejercicio de la categoría Máquina de estados.	0	0
M1_Revision	Crear un ejercicio de la categoría Revisión de código.	1	1
M1_Representante	Crear un ejercicio de la categoría	1	1

	Representante.		
M2_IdentVars	Resolver un ejercicio de la categoría Identificación de variables	1	1
M2_EqClass	Resolver un ejercicio de la categoría Clases de equivalencia.	1	1
M2_Árbol	Resolver un ejercicio de la categoría Árbol de decisión	5	4
M2_Máquina	Resolver un ejercicio de la categoría Máquina de estados.	2	2
M2_Revision	Resolver un ejercicio de la categoría Revisión de código.	0	0
M2_Representante	Resolver un ejercicio de la categoría Representante.	1	1
M1_AltaDoc	Alta de docente	3	0
M1_CorManual	Corrección Manual	3	2
M1_Login	Login/Logout	1	1
M1_Pistas	Obtención y uso de pistas y ayudas	2	2
M1_Menú	Navegar por el menú de lecciones y ejercicios	4	4
M1_CrearEjer	Copiar ejercicio	1	0
M1_Registro	Registrar un usuario	3	3

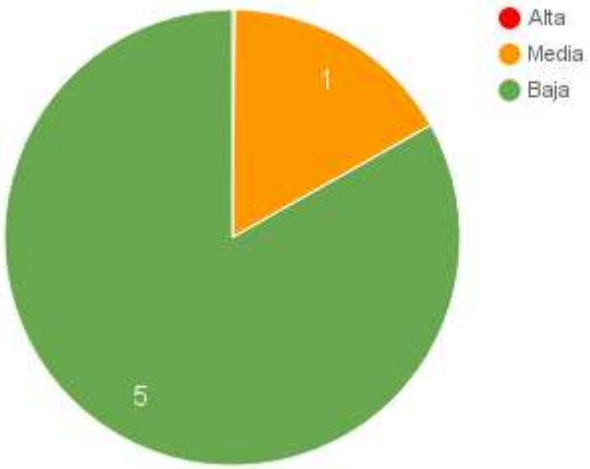
Tabla 7 - Misiones

Durante las pruebas se detectaron 33 incidentes de los cuales restan sin corregir 6. Se ejecutaron dos ciclos de pruebas, en el segundo ciclo se hizo regresión de los incidentes encontrados en el primer ciclo. Los incidentes encontrados en las pruebas de humo no fueron gestionados porque fueron corregidos en el momento.

Incidentes



Severidad



8. Gestión del proyecto

En la primera parte de este capítulo se describe la planificación del proyecto, la estimación inicial y el tiempo real insumido en cada etapa, mientras que en la segunda parte se analizan los riesgos. Como cierre del capítulo se presenta la estrategia y las herramientas de gestión de cambio utilizadas.

8.1 Gestión del proyecto

El proyecto se divide en cuatro etapas bien marcadas. En la primera se relevan y definen los requerimientos por parte del cliente (CES). En la segunda etapa se hace un estudio del estado del arte de las herramientas y talleres de capacitación en testing y otras áreas de enseñanza en la que se aplica la capacitación a distancia.

Luego de estas dos etapas se comienza con el desarrollo de software diseñando la arquitectura y evaluando las tecnologías a utilizar. Dentro de la etapa de implementación se planifican dos grandes hitos. El primero consiste en el desarrollo de prototipos en los que se prueban las posibles herramientas de desarrollo a usar. Esta etapa se caracteriza por poseer una gran carga horaria usada para investigación, pruebas, estudio y aprendizaje de diferentes tecnologías. Se investiga más en profundidad librerías que brinden la potencialidad necesaria para el desarrollo dinámico de interfaces. Luego el segundo ciclo, se trabaja en el producto final que nos insume diez meses de desarrollo.

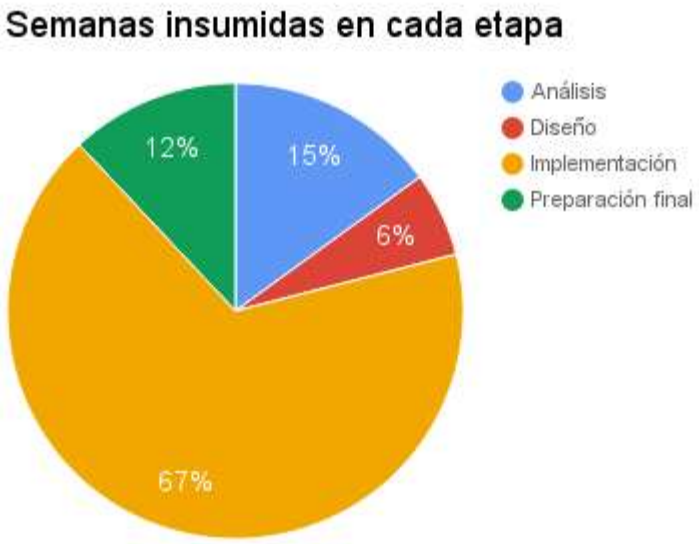
Para planificar el tiempo y ordenar las tareas nos concentramos en el alcance propuesto por el cliente (CES) que sugiere la implementación de ejercicios para un conjunto de técnicas de testing necesarias para el aprendizaje. Ya con el proyecto a punto de finalizar nos presentamos a *Ingeniería de Muestra*.

En la última etapa del proyecto se hace testing al sistema y se corrigen los incidentes encontrados. Finalmente se elabora el informe del proyecto y se prepara la presentación final.

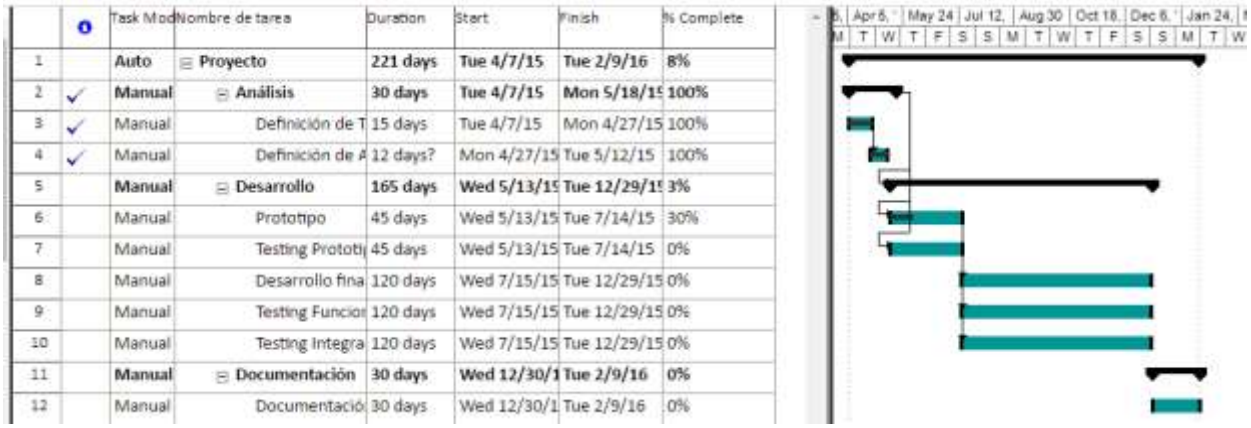
8.1.1 Distribución de horas trabajadas

El proyecto abarca un total de 86 semanas de trabajo distribuidas de la siguiente manera:

Figura 18 - Gantt real



En un principio se había estimado que el proyecto debía finalizar en diciembre de 2015 pero, fundamentalmente por los problemas con la elección de las herramientas a utilizar y sus dimensiones, no fue posible cumplir con los plazos.



		Task Mod	Nombre de tarea	Duration	Start	Finish	% Complete
1		Auto	Proyecto	221 days	Tue 4/7/15	Tue 2/9/16	8%
2	✓	Manual	Análisis	30 days	Tue 4/7/15	Mon 5/18/15	100%
3	✓	Manual	Definición de T	15 days	Tue 4/7/15	Mon 4/27/15	100%
4	✓	Manual	Definición de A	12 days?	Mon 4/27/15	Tue 5/12/15	100%
5		Manual	Desarrollo	165 days	Wed 5/13/15	Tue 12/29/15	3%
6		Manual	Prototipo	45 days	Wed 5/13/15	Tue 7/14/15	30%
7		Manual	Testing Prototi	45 days	Wed 5/13/15	Tue 7/14/15	0%
8		Manual	Desarrollo fina	120 days	Wed 7/15/15	Tue 12/29/15	0%
9		Manual	Testing Funcior	120 days	Wed 7/15/15	Tue 12/29/15	0%
10		Manual	Testing Integra	120 days	Wed 7/15/15	Tue 12/29/15	0%
11		Manual	Documentación	30 days	Wed 12/30/1	Tue 2/9/16	0%
12		Manual	Documentació	30 days	Wed 12/30/1	Tue 2/9/16	0%

Figura 19 - Gantt estimado

8.2 Gestión del riesgo

8.2.1 Riesgos de Requerimientos

Luego de varias reuniones con el cliente se definen las funcionalidades que debe poseer la herramienta y se formulan los requerimientos. En el proceso de desarrollo surgen nuevos requerimientos los cuales debieron ser abordados en posteriores reuniones con el CES.

Se entendió que las nuevas solicitudes aportarían funcionalidades realmente necesarias, aunque ello implicaría un riesgo para el correcto funcionamiento del proyecto pues se debía replantear cada iteración y definir en qué momento iban a ser llevadas a cabo. Por ejemplo, mejorar la interfaz de los ejercicios que definen grafos, esto nos lleva a investigar e incorporar herramientas externas. Otra de las funcionalidades planteadas con posterioridad es la tolerancia a fallos en el ingreso de las respuestas por parte de los alumnos.

8.2.2 Versionado del software

Surgieron inconvenientes con el uso del repositorio de software GIT, fue costoso el aprendizaje consumiendo la adaptación demasiado tiempo extra. Finalmente se decidió cambiar de repositorio a SVN debido a inconsistencias sufridas y problemas para hacer merge de los cambios, esto nos llevó a un retraso en la primera etapa de desarrollo.

8.2.3 Framework principal

Uno de los principales problemas fue el *framework* utilizado en primera instancia Vaadin. Luego de tres meses de desarrollo del prototipo se debió abandonar el software ya que no nos permitía llevar a cabo la arquitectura planteada. Luego de esto se investiga nuevamente un *framework* que se adaptara a nuestros propósitos encontrándonos con GWT (*framework* base de Vaadin).

8.2.4 Registro de horas.

Al inicio del proyecto se registran las horas en una planilla Excel, luego en la etapa de implementación se comienza a usar la herramienta de control de cambio Assembla que cuenta con un complemento para registro de tareas y horas trabajadas por semana.

Figura 20 - Registro de horas

8.2.5 Estimación de esfuerzo

La sub-estimación en cuanto al esfuerzo requerido para las iteraciones y el correcto funcionamiento del proyecto llevaron a negociar nuevamente el alcance del proyecto. Al pasar a las últimas etapas del proyecto trabajamos con una lista de tareas pendientes tanto a nivel de implementación de funcionalidades nuevas y corrección de incidentes como para la documentación. Se crea un sistema de prioridades alta, media-alta, media y baja para planificar el desarrollo.

8.3 Gestión de cambios

8.3.1 Versionado

Para el versionado del código se utiliza un servidor provisto por la herramienta Assembla con la estructura SVN. Se crean cuentas de usuarios para todos los integrantes y se mantiene el repositorio como privado para que solamente sea visible para los miembros del equipo. Assembla es un servidor estable y muy seguro el cual facilita el trabajo remoto y trabajo en equipo.

8.3.2 Documentación

Para la documentación se utiliza una estructura de directorios ordenados y alojados en Google Drive, donde la información se comparte con el equipo completo. Dicha estructura se mantiene y actualiza durante el transcurso del proyecto.

8.3.3 Base de datos

A lo largo del desarrollo y diseño de la aplicación, se fue redefiniendo la base de datos, se modifican los datos y actualiza la estructura en base a las necesidades que van surgiendo. Para llevar una traza de los cambios se crean respaldos semanales de la base de datos que son versionados conjuntamente con el código.

9. Conclusiones y Trabajo Futuro.

9.3 Conclusiones

Este proyecto de grado permitió diseñar y construir una plataforma interactiva que alcanza los objetivos planteados por el cliente. El resultado es un producto que puede y será puesto en producción, además ayudará al CES en el dictado de la carrera de testing.

El planteamiento original fue desarrollar una herramienta que presentara ejercicios que permitieran practicar las distintas técnicas de testing. Se propuso un conjunto inicial de técnicas que durante el transcurso del proyecto sufrió cambios. En particular la técnica “Tabla de decisión” fue excluida, por las dificultades tecnológicas en la generación de ejercicios de la categoría mencionada utilizando las herramientas disponibles.

Por otro lado, se agregaron funcionalidades que entendimos contribuyen a mejorar la experiencia con la plataforma como ser las funcionalidades administrativas, la corrección manual de los ejercicios y la incorporación de una bibliotecas que permitiera la visualización de grafos.

Durante el proyecto nos encontramos con problemas tecnológicos y de gestión. En particular estimamos de manera incorrecta el tiempo de algunas tareas y fue necesario descartar una tecnología elegida para la solución generando que el proyecto se extendiera más de lo esperado.

Por la capacidad de la herramienta para construir distintos tipos de ejercicios y agregar scripts para la corrección automática, concluimos que puede ser utilizada para capacitar en otras áreas que no estén relacionadas con el testing. Por ejemplo, se puede utilizar para enseñar a niños el uso de los operadores aritméticos y lógicos, así como también las tablas de multiplicar o la enseñanza de idiomas.

Por todo lo mencionado anteriormente podemos concluir que los resultados alcanzados son buenos, no obstante entendemos que tiene puntos que se pueden mejorar como se detalla en la sección 9.2.

El proyecto nos aportó a los integrantes del equipo, la experiencia de trabajar con herramientas que no conocíamos. Nos permitió aplicar conocimientos de gestión de proyectos adquiridos en cursos anteriores, así como también de análisis de una problemática y diseño de la solución. Nos enfrentamos a diferentes problemas que pudimos sortear durante este periodo lo que hace la experiencia muy enriquecedora.

9.2 Trabajo a futuro

Además de la corrección de los incidentes detectados que se encuentran abiertos, identificamos un conjunto de actividades que ayudarían a mejorar el producto obtenido. Por motivos de tiempo no se hicieron y se presentan como trabajo a futuro. A continuación, se enumeran las actividades que se pueden hacer para continuar con el desarrollo de la plataforma:

- *Mejorar el manejo de lenguaje natural*: Mejorar la tolerancia a errores en las respuestas de texto libre, sería de gran valor que el sistema pudiera reconocer palabras con el mismo significado o con variantes de tiempos verbales. Mejora el trabajo hecho por el algoritmo Jaro.
- *Autenticación*: Los alumnos de la carrera de testing del CES también usan la plataforma de aprendizaje Moodle. Sería de utilidad que el alumno pudiera autenticarse a Sophói con las credenciales de su cuenta del Moodle.
- *Validaciones de datos*: Mejorar la validación de los datos de entrada tanto en el proceso de diseño del ejercicio como en la resolución del mismo. Permitir que haya campos obligatorios u opcionales. Restringir la cantidad de componentes que se pueden agregar dentro de una actividad.
- *Expandir el conjunto de componentes*: Crear nuevos componentes para enriquecer las actividades de los ejercicios. En el proceso de construcción de la solución se pensó en la posibilidad de que la actividad a resolver dentro de un ejercicio tuviera varios pasos o pantallas, a futuro se podría implementar dicha mejora.
- *Testing*: Someter la aplicación a pruebas de performance.
- *Seguridad*: Ejecutar pruebas de seguridad, como ser las recomendadas por OWASP.

10. Referencias

1. Web Goat. https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project . Consultado Abril 2015.
2. Timbó, http://www.timbo.org.uy/udelar . Consultado Mayo 2015.
3. Juliana Herbert. *La evolución del testing a lo largo del tiempo*. TestingUy Edición 2015. Montevideo, Uruguay. http://testing.uy/charlas/12-Evolucion-del-testing.pdf. Consultado Mayo 2015.
4. About us - ISTQB® International Software Testing Qualifications Board. http://www.istqb.org/about-as.html . Consultado Mayo 2015
5. Association for Software Testing. https://www.associationforsoftwaretesting.org/ . Consultado Mayo 2015.
6. Centro de Ensayos de Software (CES). http://www.ces.com.uy/ . Consultado Abril 2015
7. Centre for Research in Evolution, Search & Testing. http://crest.cs.ucl.ac.uk/cow/17/slides/COW17_Bertolino.pdf . Consultado Mayo 2015.
8. Cem Kaner, J.D., Ph.D. http://kaner.com/ . Consultado Mayo 2015.
9. James Bach's Blog | Creator of Rapid Software Testing™. http://www.satisfice.com/blog/ . Consultado Mayo 2015.
10. Michael Bolton. http://www.developsense.com/blog . Consultado Mayo 2015.
11. Scott Barber. http://www.perftestplus.com/tech_leadership.htm . Consultado Mayo 2015.
12. Rapid Testing - James Bach - Satisfice, Inc. 2002. http://www.satisfice.com/info_rst.shtml . Consultado Mayo 2015.
13. I am a Bug. Robert Sabourin. http://www.amibug.com/iamabug/p01.html . Consultado Mayo 2015.
14. Let's Test Conferences - For Testers, By Testers 2011.http://lets-test.com/ . Consultado Mayo 2015.
15. OWASP - Open Web Application Security Project. https://www.owasp.org/index.php/Main_Page . Consultado Mayo 2015.
16. Jakarta Project. http://jakarta.apache.org/ . Consultado Mayo 2015.
17. Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable. Lincoln, USA, 2011. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.6086&rep=rep1&type=pdf . Consultado Junio 2015.
18. Black Box Puzzles. 2014. http://blackboxpuzzles.workroomprds.com/ . Consultado Junio 2015.
19. Duolingo. https://es.duolingo.com/ .Consultado Junio 2015.
20. W3Schools. http://www.w3schools.com/ . Consultado Junio 2015.
21. 10 Heurísticas de Nielsen. 1995. https://www.nngroup.com/articles/ten-usability-heuristics/ . Consultado Octubre 2015.
22. Cohen, W. W.; Ravikumar, P.; Fienberg.2003.A comparison of string distance metrics for name-matching tasks. KDD Workshop on Data Cleaning and Object Consolidation. https://www.cs.cmu.edu/afs/cs/Web/People/wcohen/postscript/kdd-2003-match-ws.pdf . Consultado Agosto 2016.
23. Levenshtein Distance. http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm . Consultado Agosto 2016.
24. Testing Exploratorio. http://www.ces.com.uy/documentacion/imasd/CES-Presentacion_Testing_Exploratorio_en_la_Practica.pdf . Consultado Noviembre 2016.

11. Anexos

Anexo I - Manual de Usuario.
Anexo II - Manual de Instalación.

Manual de instalación

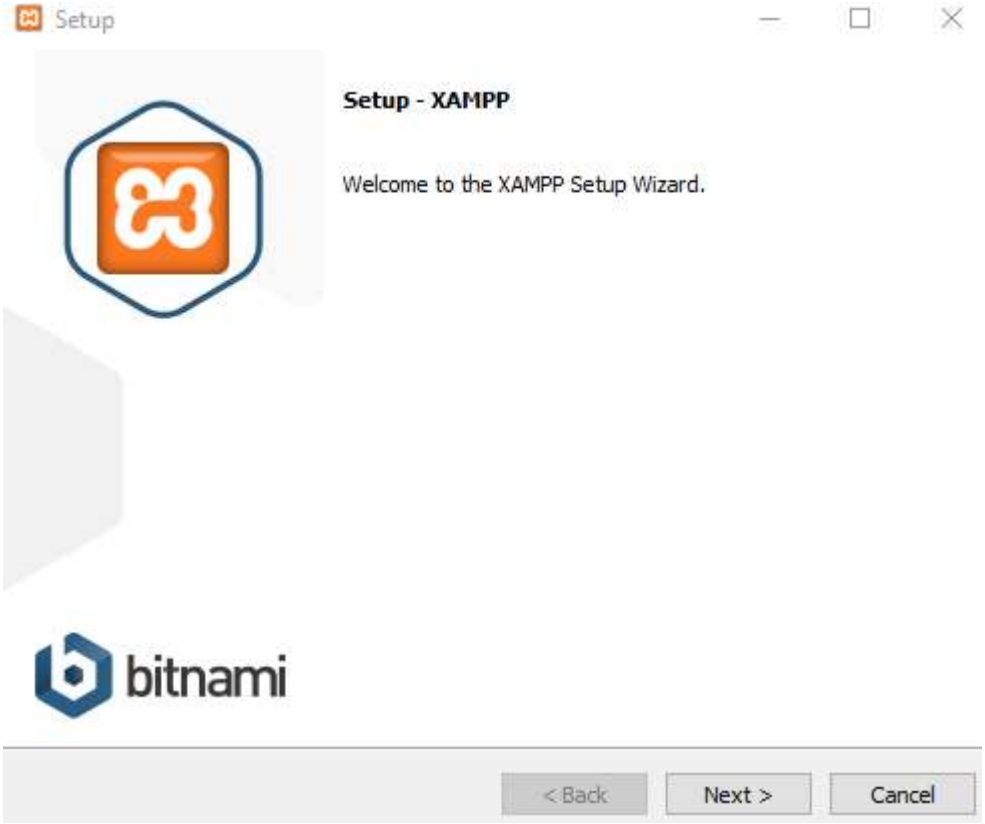
Contenido

Servidor de base de datos 3
Instalación de xampp 3
Creación de la base de datos 5
Servidor de aplicaciones 8
Instalación del servidor 8
Instalación del driver 8
Configuración conexión 9
Máquina virtual java 10
Instalación 10
Sophói 12
Despliegue 12

Servidor de base de datos

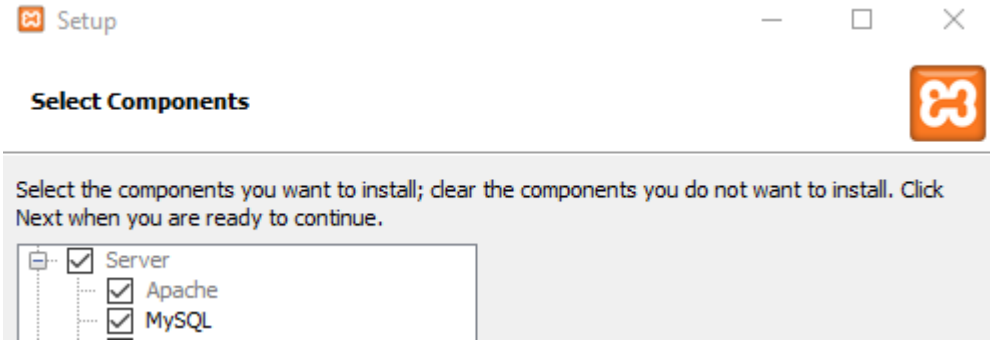
Instalación de xampp

- Para alojar la base de datos del sistema se utilizó la aplicación **XAMPP 5.6.28** disponible en: https://www.apachefriends.org/es/index.html.

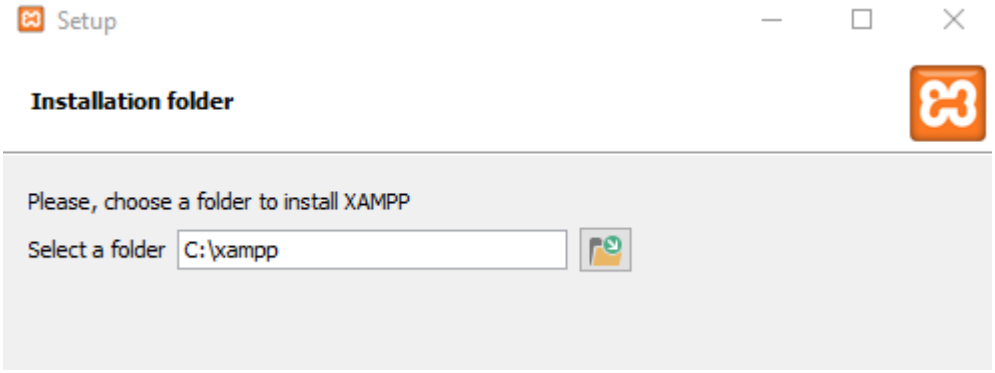


- Al comenzar la aplicación se debe hacer click en *Next*.

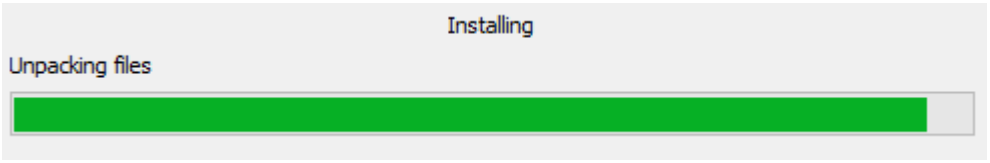
- A continuación se elige el servidor de MySQL y luego se debe hacer click en *Next*.



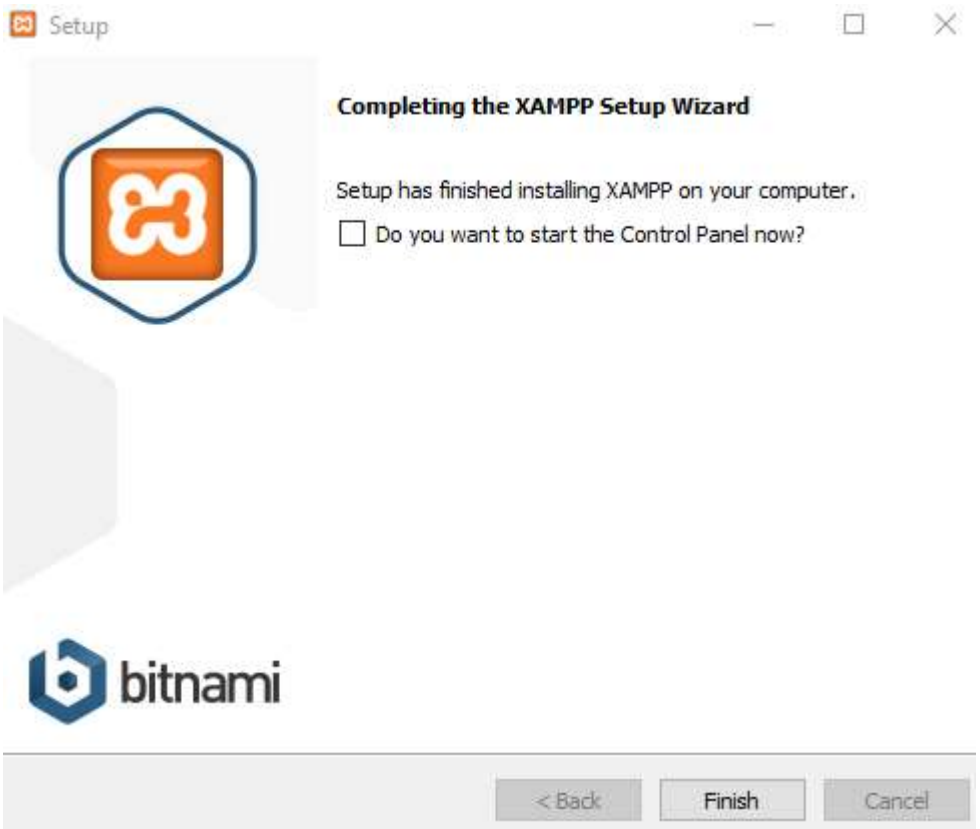
- A continuación se elige el directorio de instalación del servidor Xampp y luego se debe hacer click en *Next* en dos oportunidades.



- A continuación se ve el progreso de la instalación.

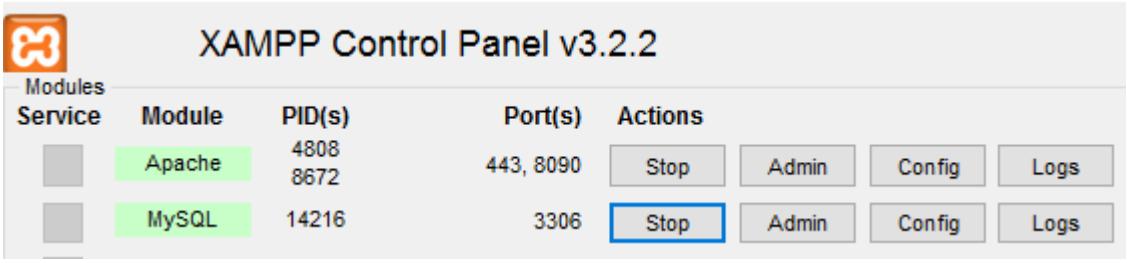


- Al finalizar vemos una pantalla donde damos click en *Finalizar*.



Creación de la base de datos

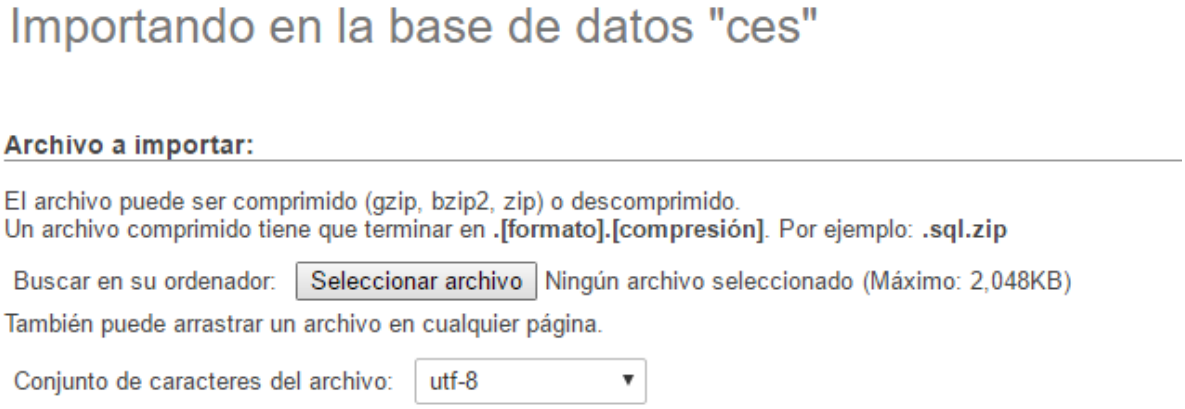
- Luego de instalar el servidor de base de datos, debemos iniciarlo.



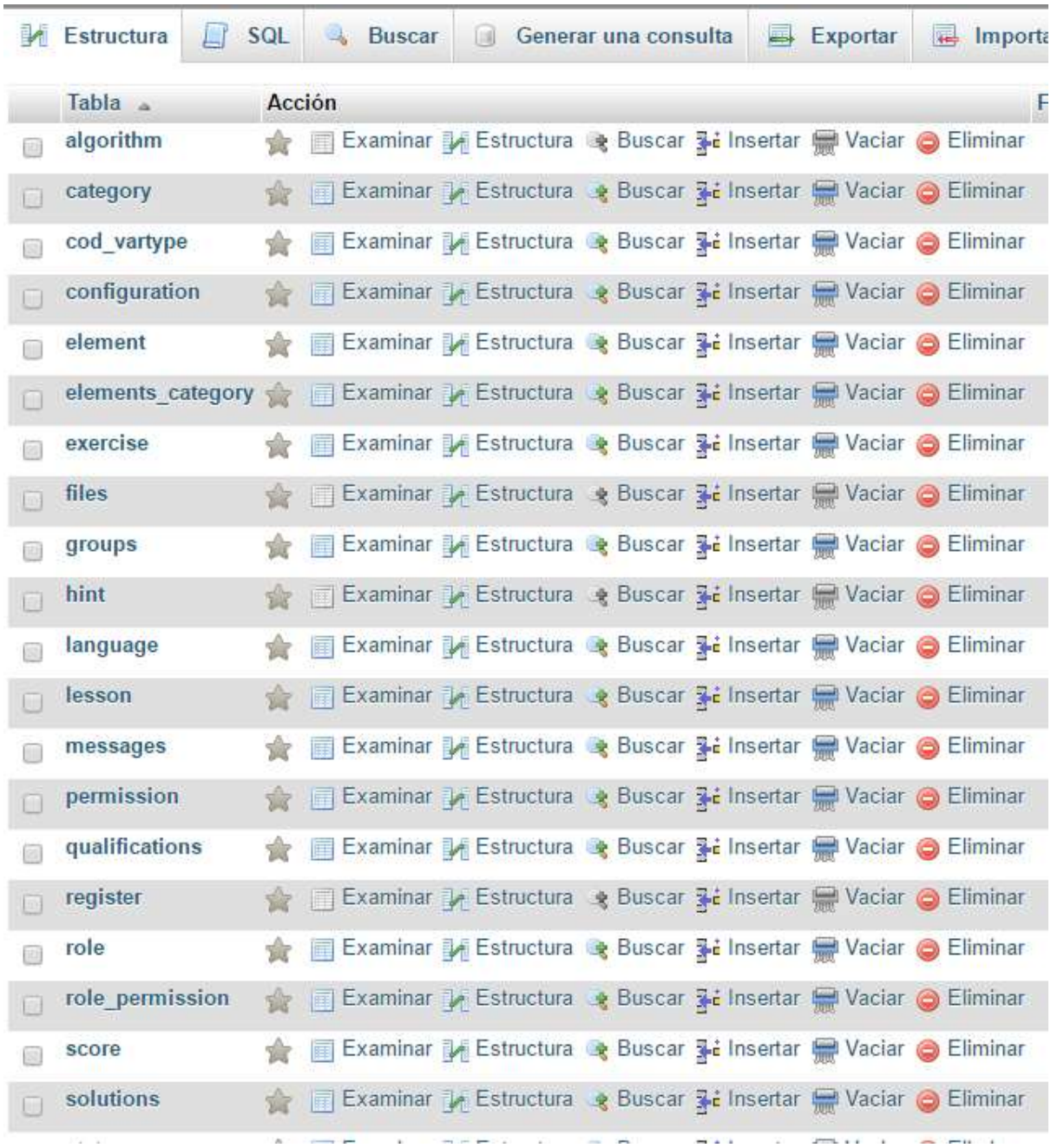
- Como cliente de base de datos podemos utilizar phpmyadmin, accesible desde: http://localhost/phpmyadmin.

- A continuación vamos al ícono de nueva (base de datos) y se crea el esquema ***ces***

- Para crear las tablas y datos de configuración, se debe hacer click en *Importar*

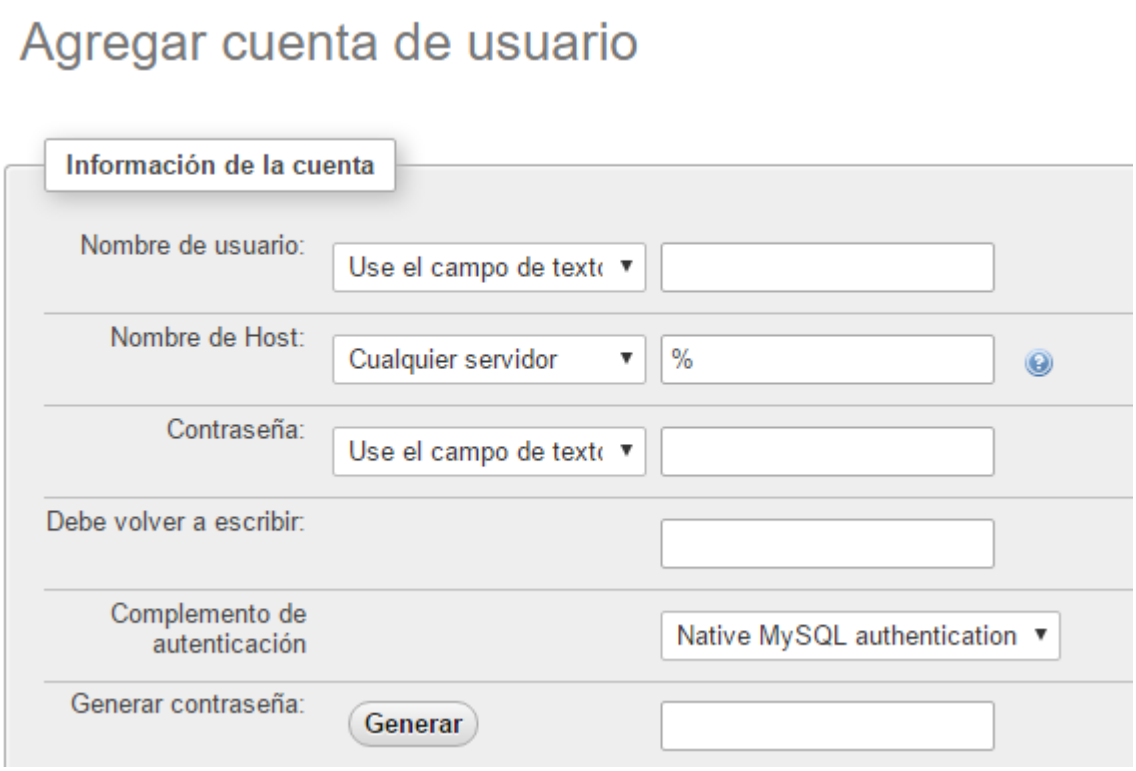


- Seleccionamos un archivo que contenga la estructura de la base de datos que además contiene los datos de configuración generales de la aplicación.
- Al culminar podemos ver la estructura de la base de datos.



- Para acceder desde la aplicación, se debe crear un usuario para el acceso.

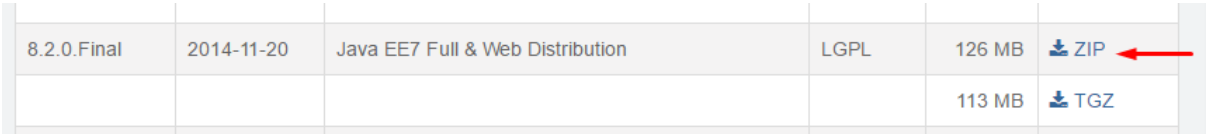
- El nombre del usuario deberá ser *ces*.


Agregar cuenta de usuario
Información de la cuenta
Nombre de usuario:
Use el campo de text
Nombre de Host:
Cualquier servidor
%
Contraseña:
Use el campo de text
Debe volver a escribir:
Complemento de autenticación
Native MySQL authentication
Generar contraseña:
Generar

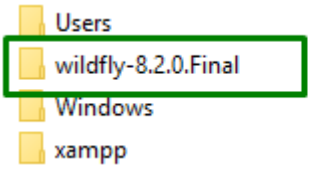
Servidor de aplicaciones

Instalación del servidor

- Descargar el release del servidor Wildfly 8.2 disponible en: http://wildfly.org/downloads/.



- Descomprimir el archivo descargado en C://



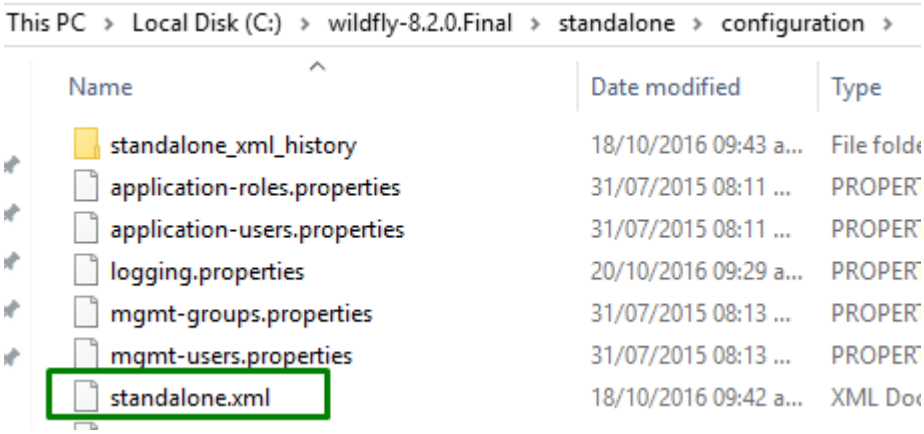
Instalación del driver

- Descargar el driver mysql de: https://downloads.mysql.com/.

- Ir al directorio C:\wildfly-8.2.0.Final\modules\system\layers\base\com\sql\mysql\main
- Copiar el driver: mysql-connector-java-5.1.6.jar dentro del mismo.

Configuración conexión

- Ir al directorio C:\wildfly-8.2.0.Final\standalone\configuration.



- Editar el archivo standalone.xml
- En él configurar el datasource para la conexión a la base de datos.

```
<datasource jndi-name="java:/appEJB" pool-name="appEJB" enabled="true" use-ccm="true">
    <connection-url>jdbc:mysql://localhost:3306/ces</connection-url>
    <driver>mysql</driver>
    <security>
        <user-name>ces</user-name>
        <password>ces</password>
    </security>
```

- Luego configurar el driver para la conexión exitosa a MySQL.

```
            </driver>
            <driver name="mysql" module="com.sql.mysql">
                <driver-class>com.mysql.jdbc.Driver</driver-class>
            </driver>
        </drivers>
    </datasources>
```

Máquina virtual java

Instalación

- Descargar el release de Java Development Kit disponible en: www.oracle.com.

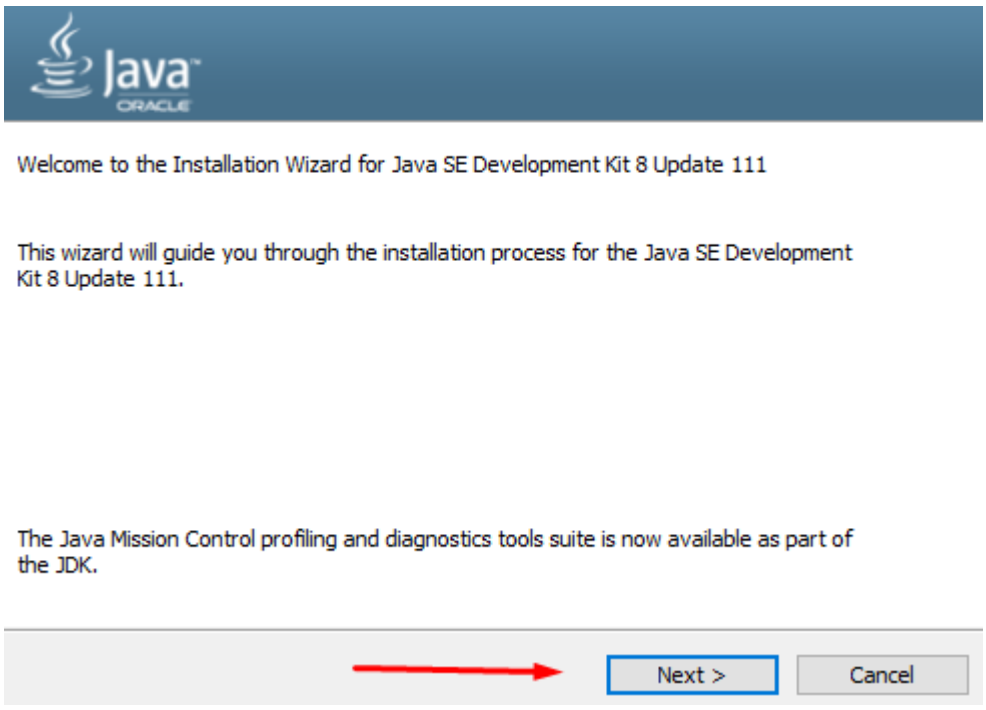
Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

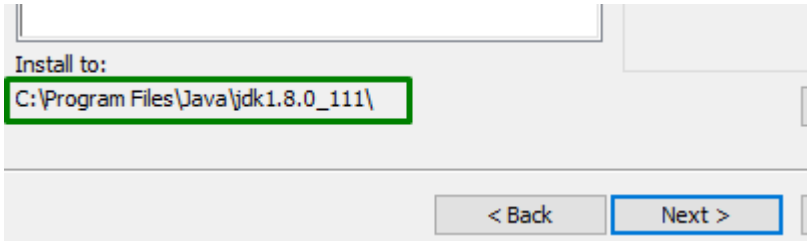
The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

- Seleccionamos la versión de 64 bits.

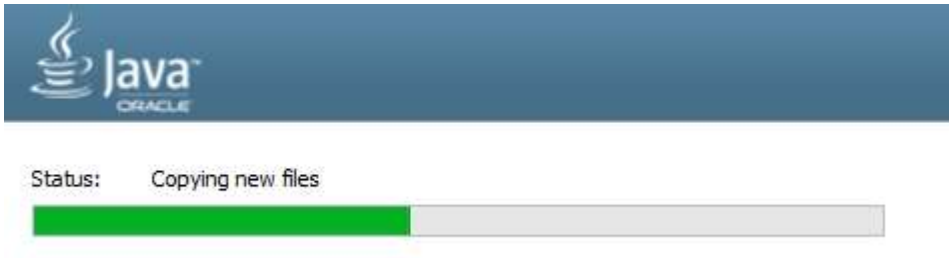
Solaris x64	96.82 MB	jdk-8u111-solaris-x64.tar.gz
Windows x86	189.22 MB	jdk-8u111-windows-i586.exe
Windows x64	194.64 MB	jdk-8u111-windows-x64.exe



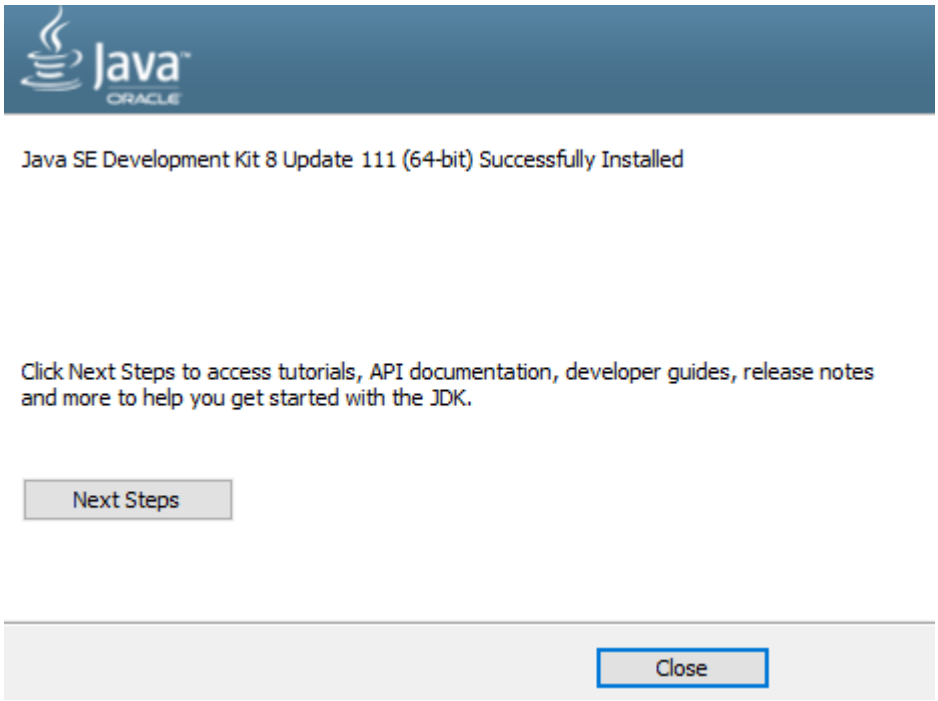
- Al ejecutar el release se ve la pantalla de bienvenida y para proceder, hay que hacer click en *Next*.
- Luego nos presenta la elección del directorio de instalación.
- Se continúa con la instalación haciendo click nuevamente en Next.



- Se visualiza el progreso de la instalación.



- Luego se observa la finalización de la instalación.

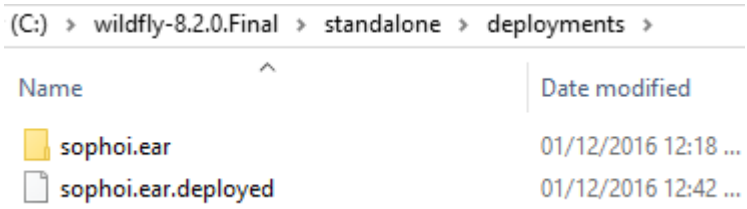


- Damos click en *Close* para finalizar la instalación.

Sophói

Despliegue

- En la carpeta *standalone* del servidor se debe guardar el archivo sophoi.ear. La capeta standalone se encuntra en C:\wildfly-8.2.0.Final\standalone\deployments



Manual de usuario

CES
CENTRO DE ENSAYOS DE SOFTWARE

Contenido

Manual para el alumno 3
Ingreso al sistema 3
Registrarse 3
¿Cómo resolver ejercicios? 4
Preferencias 9
Preferencias: Mis Datos 9
Puntuación 10
Estadísticas 10
Manual para el docente 11
Ingreso al sistema 11
Preferencias 11
Preferencias: Usuarios 12
Usuarios: Docentes 12
Usuarios: Alumnos 13
Usuarios: Registros 13
Preferencias: Calificar 14
Preferencias: Grupos 15
Preferencias: Lecciones 15
Lecciones 15
Ejercicios 16
Preferencias: Reportes 25
Preferencias: Configuraciones 26

Manual para el alumno

Ingreso al sistema

Para comenzar a usar la aplicación es necesario loguearse en el sistema ingresando usuario y contraseña.

- Si no está registrado puede pedir acceso desde el link ***Registrarse*** (consultar la sección Registrarse).

- Puede ingresar como invitado a la aplicación mediante el link ***Ingresar como Invitado***. En ese caso las funcionalidades estarán restringidas y no se guarda el progreso en lecciones.

Registrarse

Para pedir acceso al sistema ingresar al link ***Registrarse*** y completar los datos requeridos. El pedido será chequeado por un administrador para confirmar el registro.

¿Cómo resolver ejercicios?

Luego de iniciar sesión en la aplicación aparecen las lecciones para resolver.

Una lección se compone de varios ejercicios que se irán activando para resolver a medida que se aprueben los ejercicios anteriores.

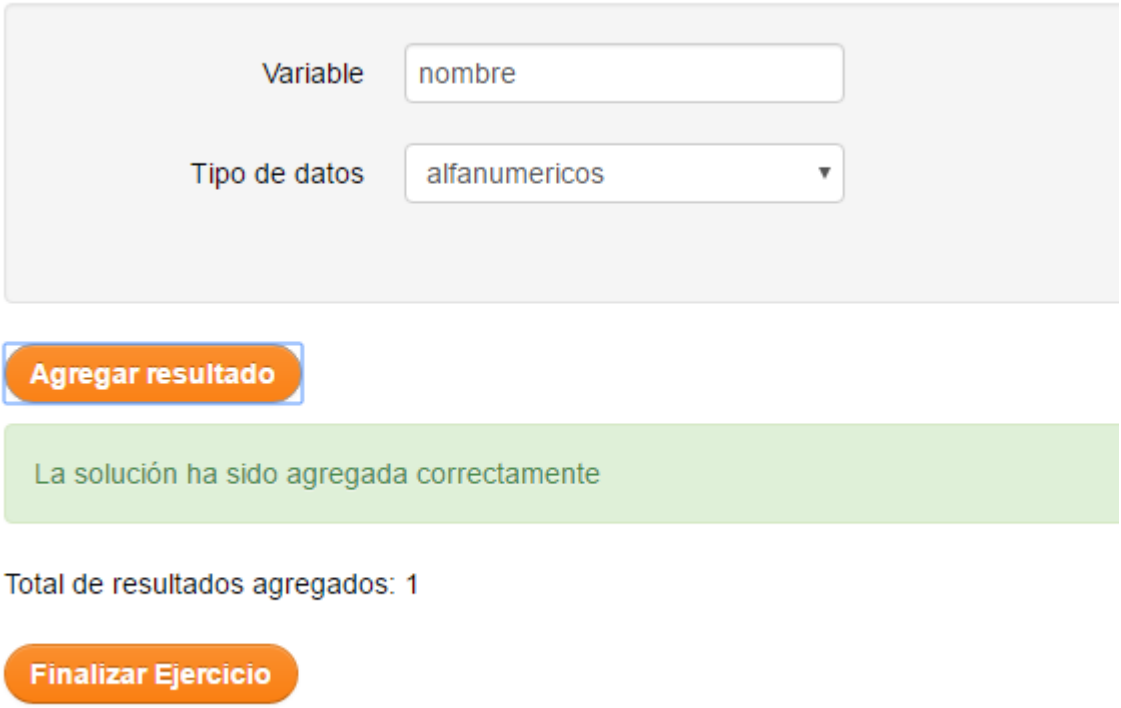
Por ejemplo haciendo click en el item *Variables* del menú de la lección *Clases* se muestra en parte principal de la pantalla ese ejercicio.

Observación: Las lecciones y ejercicios presentados en este manual pueden diferir con los existentes en versión.

➔ Completar los campos.

➔ Presionar el botón **Agregar resultados** para agregar una solución del ejercicio. Aparecerá un mensaje indicando que la respuesta ha sido agregada correctamente.

Observación: si el ejercicio implica la construcción de un grafo el botón Agregar resultado se llama **Agregar transición** y como dice su nombre agrega una transición al grafo.



➔ Agregar todas las respuestas que crea conveniente.

> ⚠ Las respuestas incorrectas restan puntos.

➔ Para finalizar el ejercicio presionar el botón **Finalizar ejercicio**

- ◆ Si el ejercicio está resuelto correctamente se habilita el siguiente ejercicio (si existe). Se muestra el puntaje obtenido.

- ◆ Si el resultado no es del todo correcto y existen pistas, haciendo click en **Obtener Pistas** se muestra una pista relacionada a una de las soluciones faltante.
  - Agregar los nuevos resultados que se desee y volver a finalizar el ejercicio.
  - Repetir el proceso anterior hasta alcanzar el 100% de las soluciones o presionar el link **Terminar** para finalizar el ejercicio.

> ⚠ Las respuestas pistas restan puntos.

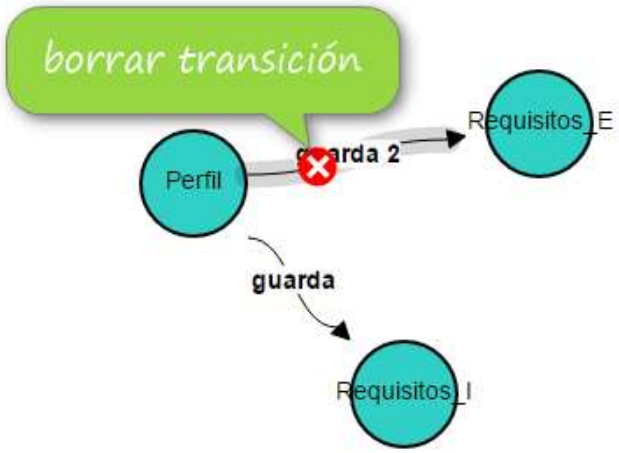
- Si el resultado no alcanza el mínimo establecido por el docente para mostrar pistas, estas no se mostrarán. El ejercicio se debe comenzar nuevamente.

Ejercicios con grafos

El grafo se construye agregando transiciones con el botón **Agregar Transición**. El sistema irá dibujando las transiciones de manera automática. El dibujo es interactivo: las aristas se pueden mover de lugar y/o eliminar.

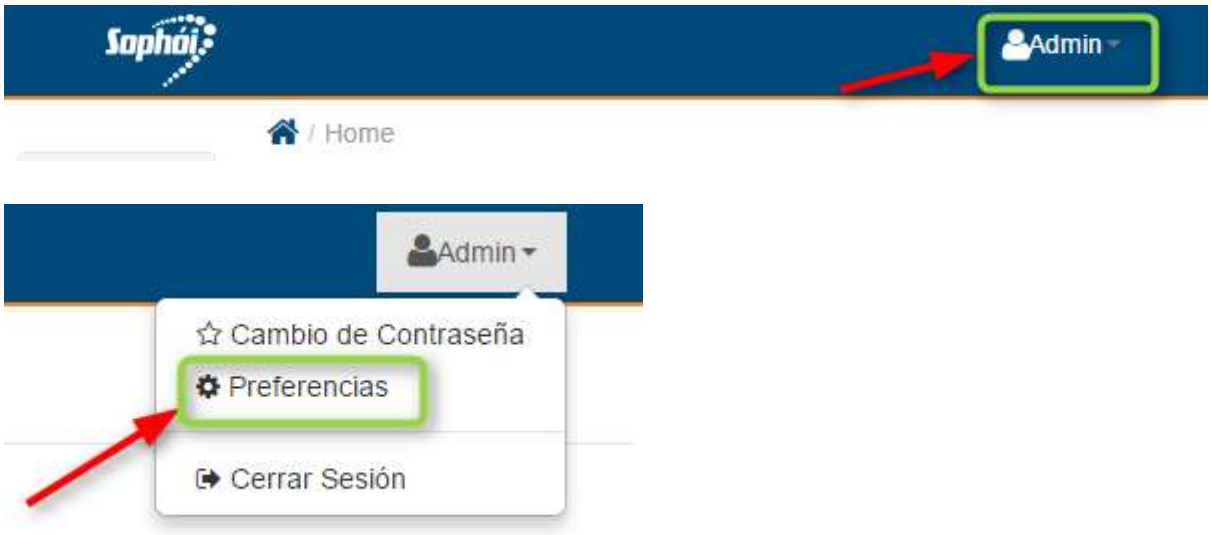
Al eliminar una transición se elimina del grafo resultado.


Nodo incial
Perfil
Nodo Final
Requisitos_E
Guarda
guarda 2
nueva transición
Agregar transición


borrar transición
Perfil
guarda 2
Requisitos_E
guarda
Requisitos_I

Preferencias

En el cabezal de la aplicación se encuentra un botón con el nombre del usuario logueado, desplegarlo y hacer click sobre **Preferencias** para ver los resultados obtenidos en los ejercicios.



Preferencias: Mis Datos

Ingresando a **mis datos** luego de haber entrado a la sección de preferencias (paso anterior del manual) se accede a la puntuación obtenida en los ejercicios resueltos y a información relativa a la resolución, como ser cantidad de pistas usadas, puntos restados, etc.

Puntuación

En la sección de **Mis Datos -> puntuación** se muestran los ejercicios resueltos, aprobados y no aprobados. La puntuación es la obtenida al finalizar el ejercicio que va de 0 a 100. La nota es una transformación de la puntuación en una escala de 1 a 12.
Presionando la lupa se accede al filtro por lecciones.

Estadísticas

Muestra datos estadísticos de las resoluciones de ejercicios agrupado por lecciones, por ejemplo la cantidad de intentos que se hicieron en todos los ejercicios de una lección, cuántos ejercicios se aprobaron, la cantidad de pistas pedidas y otros indicadores.

Manual para el docente

Ingreso al sistema

Para comenzar a usar la aplicación es necesario loguearse en el sistema ingresando usuario y contraseña.

- Si no está registrado debe solicitar al administrador del sistema permisos para ingresar a la aplicación.

  Observación: el link **Registrarse** es solamente para alumnos.

- Puede ingresar como invitado a la aplicación mediante el link ***Ingresar como Invitado***. En ese caso las funcionalidades estarán restringidas y no se guarda el progreso en lecciones.

Preferencias

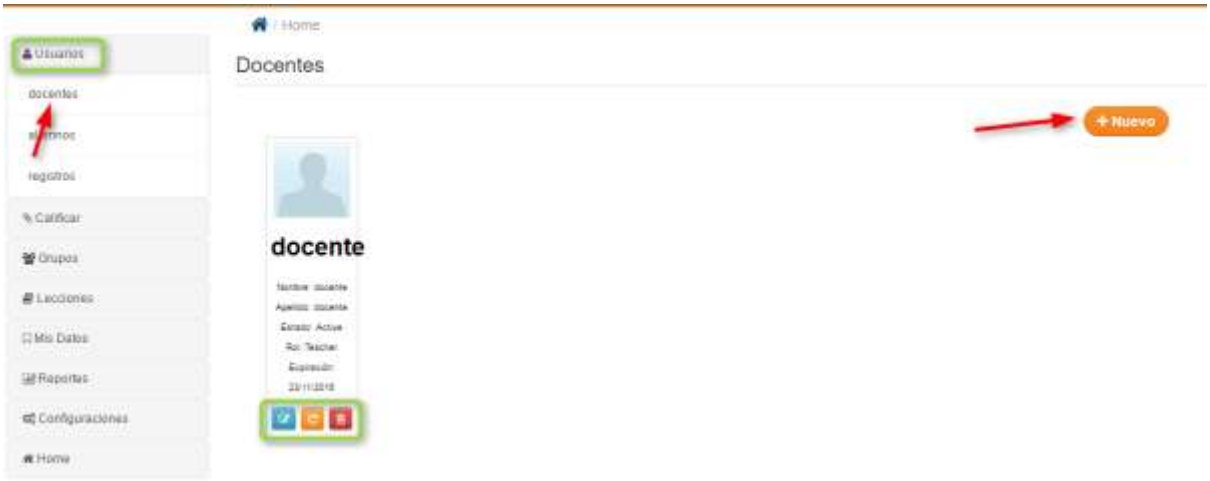
En el cabezal de la aplicación se encuentra un botón con el nombre del usuario logueado, al desplegarlo los usuarios pueden acceder a las preferencias del sistema.

Preferencias: Usuarios

En esta sección se crean docentes, alumnos y se chequea la lista de usuarios que han pedido acceso a la aplicación.

Usuarios: Docentes

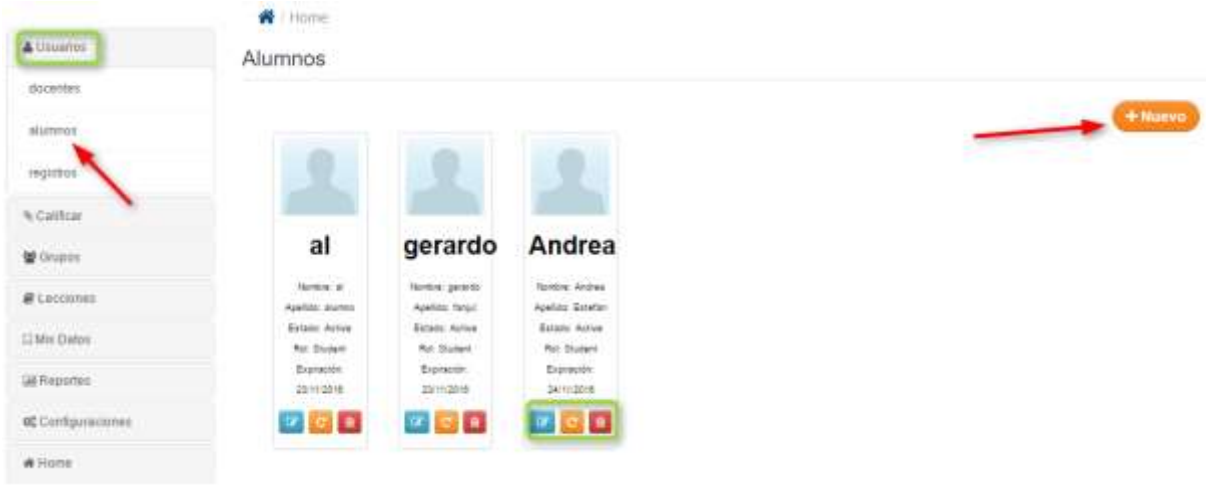
Ingresando a la sección ***Usuarios*** → ***docentes*** se puede ver la lista de docentes existentes, agregar un nuevo docente, modificar los datos de los docentes ya creados, resetear la contraseña y borrarlos.



Los datos solicitados para dar de alta o modificar un docente son el nombre de usuario, contraseña, nombre, apellido, estado (activo o bloqueado) y el rol asignado.

Usuarios: Alumnos

Ingresando a la sección **usuarios** → **alumnos** se puede ver la lista de alumnos existentes, agregar un nuevo alumno, modificar los datos de los alumnos ya creados, resetear la contraseña y borrarlos.



Los datos solicitados para dar de alta o modificar un alumno son el nombre de usuario, contraseña, nombre, apellido, estado (activo o bloqueado), rol asignado y grupo al que pertenece.

Usuarios: Registros

En la sección **usuarios** → **registros** se listan los usuarios que han solicitado registrarse en la aplicación. Al presionar el tic verde se agrega el usuario con el rol de estudiante quedando activo para ingresar al sistema, sino se puede borrar el registro presionando el icono de la papelera.

Preferencias: Calificar

Los estudiantes que tienen ejercicios que están pendientes de corrección por parte del docente (ejercicios con corrección manuales) se encuentran en esta sección.

- ➔ Para calificarlo hacer click en el icono dentro de la columna calificar.
- ➔ A continuación, se despliega un popup donde se muestran los resultados agregados por el alumno, ingresar el puntaje que corresponda y confirmar la corrección manual. El rango del puntaje permitido es de 0 a 100.

Preferencias: Grupos

Un grupo es definido como un conjunto de alumnos monitoreados por un docente que será el tutor del grupo. En esta sección se pueden ver los grupos existentes y dar de alta nuevos grupos que son definidos por un nombre y un docente asignado.

➔ Para poder agregar usuarios es necesario tener al menos un grupo creado. Todo usuario debe pertenecer a un grupo.

Preferencias: Lecciones

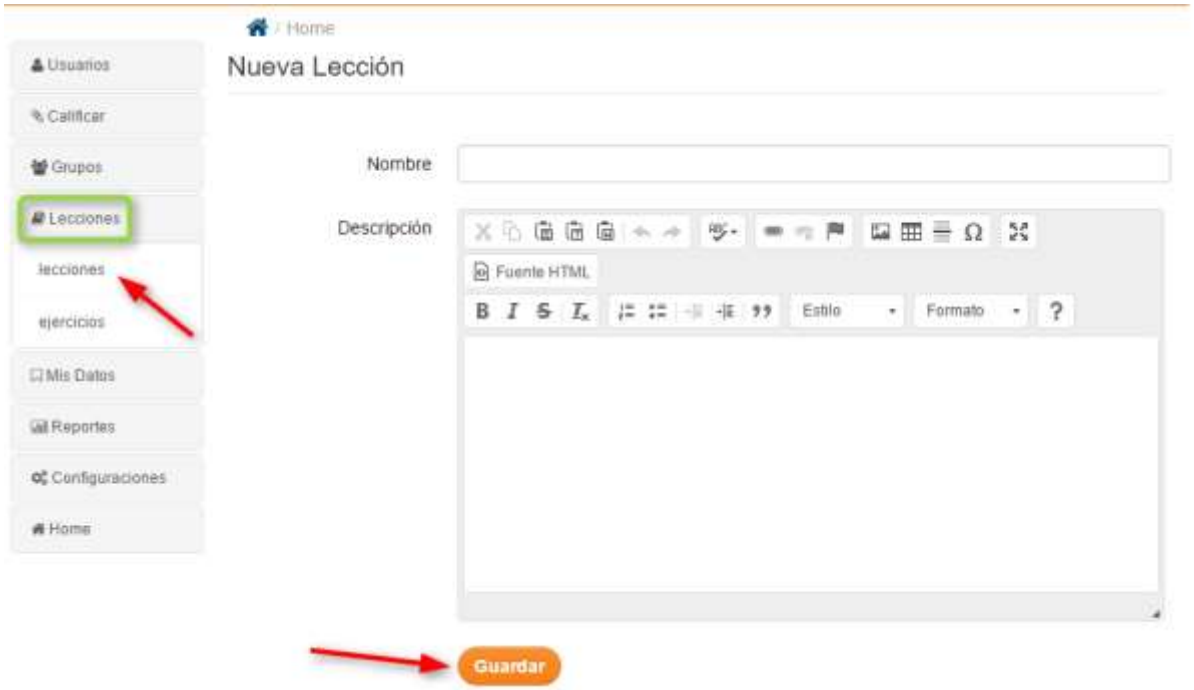
En esta sección se explica cómo configurar las lecciones y ejercicios de la aplicación.

Lecciones

Una lección se conforma de ejercicios (uno o más), el alumno debe ir aprobando en orden los ejercicios de una lección para ir activando nuevos ejercicios.
En la sección de lecciones se listan las lecciones existentes para poder editarlas, inactivarlas o borrarlas. Además se pueden crear nuevas lecciones ingresando al botón **Nueva Lección**.

Nueva lección

- ➔ Ingresar el nombre
- ➔ Ingresar descripción de la lección. Para dar formato al texto que se va a mostrar al alumno como descripción de la lección se proporciona un editor de texto clásico. Éste cuenta con botones para dar formato al texto (negrita, subrayado, cursiva, etc), agregar imagenes desde una url, crear tablas, citar texto, ingresar viñetas entre otras funcionalidades.
- ➔ Para finalizar presionar el botón guardar y quedará creada la lección, a continuación se debe agregar al menos un ejercicio a la misma.



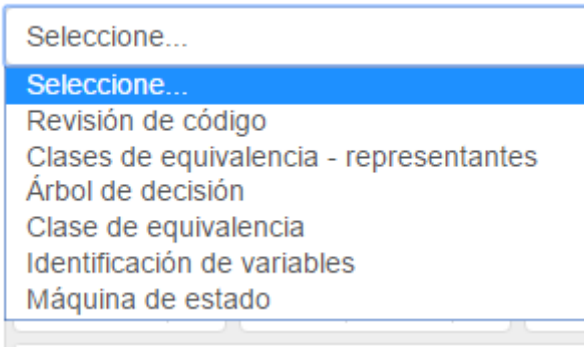
Ejercicios

Un ejercicio es una parte de una lección. En la sección de ejercicios seleccionar una lección para poder ver los ejercicios que posee y para crear nuevos ejercicios. En la tabla de ejercicios se encuentran botones para editar, copiar, agregar pistas, inactivar y borrar un ejercicio.

Nuevo ejercicio

Para crear un nuevo ejercicio, luego de seleccionar la lección a la que se va a incorporar el ejercicio nuevo hacer clic en **Nuevo ejercicio**.

➔ Seleccionar la categoría a la que va a pertenecer el nuevo ejercicio. Las posibles categorías son

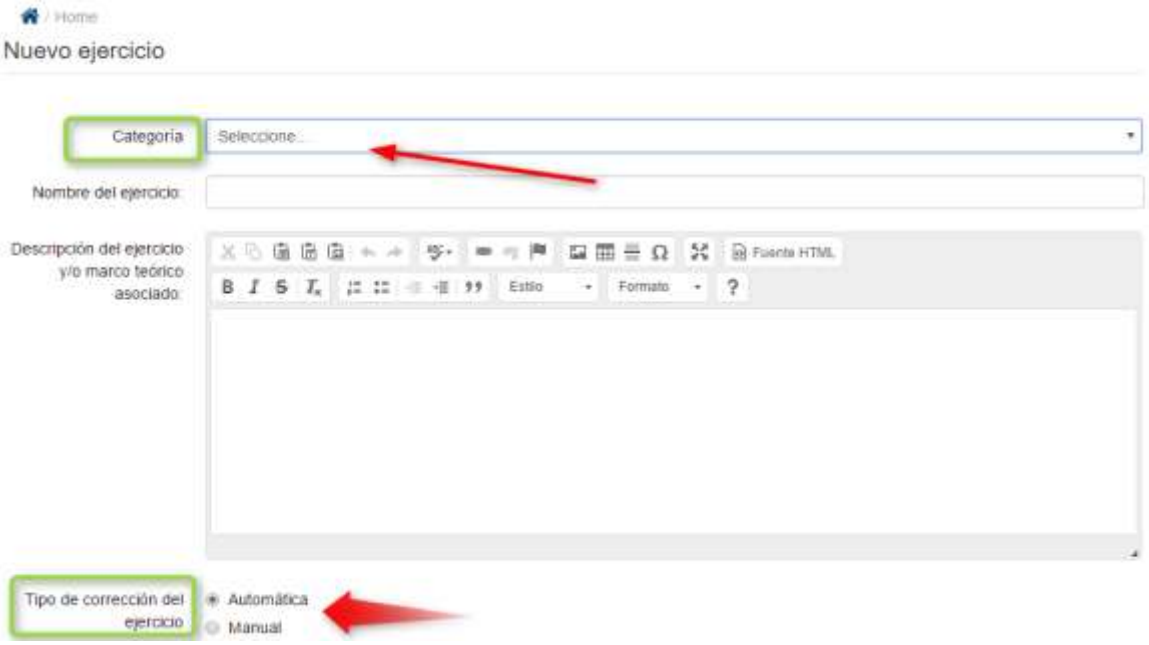


En las siguientes tablas se listan las categorías y los elementos que la componen. También se describe cada elemento. En el *Apéndice de Categorías* se muestra un ejemplo de cada un de ellas.

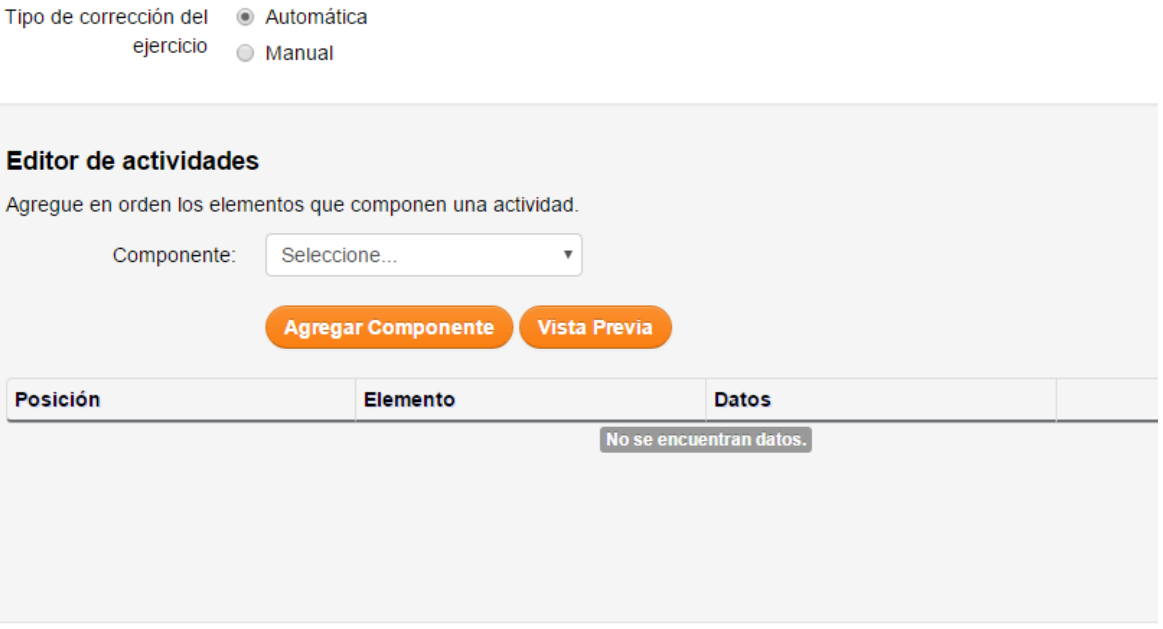
Categoría	Componentes
Árbol de decisión	Árbol y Texto
Clase de equivalencia	Texto, Imagen, Lista de Valores, Variable a ingresar y Clase de equivalencia.
Identificación de variables	Lista de Valores y Variable a ingresar
Representante	Texto, Imagen, Lista de Valores, Variable a ingresar y Representante
Revisión de Código	Imagen y Variable a ingresar
Máquina de estados	Grafo y Texto
Valores límites	Lista de Valores y Variable a ingresar

Componente	Descripción	Atributos configurables
Texto	Representa un componente HTML label.	• text: texto del componente label
Imagen	Componente que permite agregar una imagen a la actividad, la imagen puede ser cargada desde una carpeta local. La imagen tiene que tener formato png	• url: dirección de la imagen
Variable a ingresar	Es desplegado en la actividad como un componente HTML input acompañado de un label.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Lista de valores (combo box)	Representa un elemento HTML select. Los elementos se ingresan separados por coma.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Clase de equivalencia	Se como un elemento input HTML y la etiqueta que lo describe. Se diferencia del componente *variable a ingresar* en la forma que se procesa internamente el dato. Se ingresan los rangos de la clase de equivalencia separados por coma.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Representante de Clase de eq.	Se dibuja como un elemento input HTML y la etiqueta que lo describe. Se diferencia del componente *variable a ingresar* en la forma que se procesa internamente el dato.	• varName: identificador del componente • forText: etiqueta del input a dibujar • type: tipo de la variable
Grafo	Es dibujado en la actividad como campos para ingresar el nodo inicial, final, la acción, evento y guarda de una transición del grafo. Al agregar las transiciones se va armando el grafo resultado. Los cinco campos pueden ser dibujados como un comboBox con valores sugeridos o un input HTML.	• init: posibles valores del nodo inicial de una transición. • end: posibles valores del nodo final de una transición. • action: posibles valores que representan la acción • event: evento que produce el pasaje de un nodo a otro. • guard: condición que se debe cumplir para pasar del nodo inicial al final.
Árbol	Similar al grafo, se diferencia por no poseer evento y acción.	• nit: posibles valores del nodo inicial de una transición. • end: posibles valores del nodo final de una transición. • guard: condición que se debe cumplir para pasar del nodo inicial al final.

- ➔ Agregar nombre del ejercicio y descripción.
- ➔ Seleccionar el tipo de corrección del ejercicio.
    a. Si es automático será corregido automáticamente por el sistema utilizando un algoritmo de corrección escrito en JavaScript.
    b. Si es manual significa que el nuevo ejercicio será corregido por un docente



- ➔ Siguiendo con la creación del ejercicio y en la misma pantalla continuar con el editor de actividades, sección donde se va a diseñar la interfaz del nuevo ejercicio.



➔ Ir agregando los componentes en el orden en el que queremos que se dibujen en la pantalla. Cuando se selecciona un componente de la lista de componentes aparecen nuevos campos para completar, estos campos permiten identificar el elemento y agregarle funcionalidad.

Por ejemplo si se selecciona el componente *Variable a ingresar* el sistema pide ingresar un nombre para identificar el componente, el texto que queremos que le aparezca al alumno y el tipo de componente que va a a ser, en este caso texto libre.

➔ Presionar ***Agregar Componente*** y si desea ir viendo cómo va quedando diseñado el ejercicio presionar ***Vista previa***

➔ Luego de terminar con el diseño de la actividad del ejercicio, si es automático ir a la página siguiente donde se van a agregar las soluciones. Si es manual presionar ***Guardar*** y quedará creado el ejercicio.

Si es un ejercicio automático se pasa a la página de agregar soluciones.

➔ Seleccionar el algoritmo de corrección que se va a usar para el ejercicio. Si se va a utilizar el algoritmo de corrección por defecto o si se va a utilizar un algoritmo propio. En el último caso se debe agregar el archivo que contiene el código JavaScript de corrección.

El JavaScript debe estar correctamente implementado, de lo contrario indicará un error cuando se intente resolver un ejercicio.

➔ Para todas las variables definidas en la pantalla anterior agregarle los valores solución del problema y si se desea también se puede agregar una pista asociada a cada solución. Ésta pista será mostrada al alumno si no contesto correctamente esa solución.

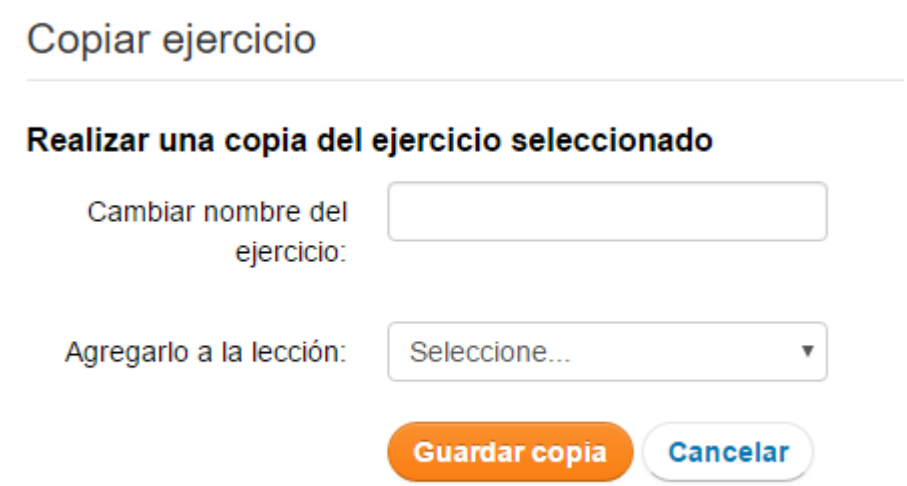
Se pueden agregar varias soluciones si el ejercicio lo amerita. Las soluciones agregadas irán apareciendo en una tabla, pudiendo eliminar entradas si se desea.

➔ Presionar ***Finalizar*** para terminar de crear el ejercicio.

Copiar ejercicio

Para copiar un ejercicio hacer click en el link de copiar que se encuentra en la columna de acciones

➔ Agregar el nuevo nombre de la copia del ejercicio y seleccionar a qué lección va a pertenecer. Para confirmar la acción hacer click en **Guardar copia**



Modificar ejercicio

Para modificar un ejercicio existente hacer click en el ícono de modificación de la columna de acciones.

- ➔ El nombre y la categoría de un ejercicio no son editables porque son atributos que identifican a un ejercicio.
- ➔ Es posible editar, agregar o eliminar componentes de la actividad

➔ Para finalizar editar las soluciones y **guardar** los cambios.

Agregar ayudas del ejercicio

En ayudas se pueden agregar conceptos teóricos para orientar al estudiante a completar el ejercicio.

➔ Para agregar ayudas a un nuevo ejercicio seleccionar la lamparita de la columna de acciones

Inactivar/activar un ejercicio

Si es necesario ocultar un ejercicio para que los alumnos no lo tengan disponibles para resolver, hacer click en el candadito de la columna acciones. Para revertir esta acción volver a hacer click en el candadito.

Preferencias: Reportes

En esta sección se encuentran reportes de utilidad para el docente.

Puntuación

Muestra el puntaje y la nota obtenida por los estudiantes, se puede filtrar por alumno y/o lección.

Estadísticas

Muestra datos estadísticos de las resoluciones de ejercicios agrupado por lecciones, por ejemplo la cantidad de intentos que se hicieron en todos los ejercicios de una lección, cuántos ejercicios se aprobaron, la cantidad de pistas pedidas y otros indicadores.

Grupos

Por último, en el reporte por grupos se agrupa la información de los estudiantes según el grupo al que pertenecen.

Preferencias: Configuraciones

En esta sección de las preferencias se pueden editar algunos datos de configuración como ser el puntaje de penalización por pistas obtenidas, el porcentaje mínimo de aprobación del ejercicio y el porcentaje mínimo que se debe sacar un alumno para poder acceder a las pistas. Esta funcionalidad está habilitada solamente para los usuarios con rol de administrador.