UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Diseño y Desarrollo del sistema ASys con Spring y Vue.js

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Armando Maya Gomis

Tutor: Josep Silva Galiana

Curso 2018-2019

Dedicado a la persona

que, en su último aliento,

me animó a seguir.

Gracias Tito.

Resumen

La plataforma ASys tiene como principal objetivo mejorar las capacidades de programación de los alumnos y facilitar las tareas de corrección del profesorado. Se trata de un sistema con ejercicios autoevaluables en el que, por una parte, los alumnos podrán recibir estadísticas y recomendaciones de mejora; y por otra parte, los profesores tendrán herramientas de corrección automática y semiautomática a su disposición, que en muchos casos ahorrarán ingentes cantidades de trabajo.

Actualmente, ASys cuenta con dos versiones: una versión de escritorio y una versión web. La necesidad que ha impulsado este trabajo es la de crear una nueva versión que unifique ambas aproximaciones, y mejore el rendimiento de estas para soportar muchos usuarios—al menos varios cientos—al mismo tiempo. Además, esta nueva versión debe ser escalable y fácilmente ampliable por los nuevos desarrolladores que puedan incorporarse al proyecto en un futuro.

El principal objetivo de este trabajo es el de crear un proyecto nuevo, con una estructura base, para que pueda ser fácilmente ampliable y cumpla las necesidades de rendimiento descritas anteriormente.

Palabras clave: ASys, corrección automática, corrección semiautomática, ejercicios, programación, Spring, Vue, Web, PWA, Java, Javascript.

Abstract

The main goal of the ASYS platform is to improve the programming abilities of the students and to facilitate the assessment tasks of the teaching staff. This system manages self-assessing exercises in which, on the one hand, students can receive statistics and recommendations for improvement and, on the other hand, teachers will have automatic and semi-automatic correction tools at their disposal that, in most cases, will save huge amounts of work.

Currently, ASys has two versions, a desktop version and a web version. The need that has driven this work is to create a new single version that unifies both approaches, and improves their performance giving support for a large number of users (at least several hundreds) at the same time. In addition, this new version must be scalable and easily expandable by new developers who may take part in the project in the future.

The main objective of this work is to create a new project, with a base structure, so that it can be easily expanded and meet the performance needs described above.

Keywords: ASys, automatic correction, semi-automatic correction, exercises, programming, Spring, Vue, Web, PWA, Java, Javascript.

Tabla de contenidos

1. Introducción

1.1 Motivación

Antes de abordar este proyecto, mi plan era hacer un TFG propio. Pensaba desarrollar un proyecto que, en mi opinión, era novedoso y se le podía sacar cierta rentabilidad económica. Pero mis planes se torcieron cuando me ofrecieron desarrollar ASys. Un proyecto fruto de una larga investigación y con varias publicaciones científicas a sus espaldas [1, 2, 3, 4].

ASys es un proyecto destinado a revolucionar la formación de los estudiantes en el entorno académico actual y, por tanto, un proyecto con un futuro prometedor y una gran trascendencia social.

Evidentemente, cuando me propusieron unirme al proyecto acepté de inmediato.

1.2 Objetivos

El principal objetivo de este TFG es producir una nueva versión de ASys que estará formada por un esqueleto creado desde cero y que servirá como base estructural y núcleo de futuras versiones. Esta nueva versión se desarrollará en un entorno web. Además, deberá contar con una alta capacidad para soportar cambios por parte de los desarrolladores, y con un alto rendimiento para soportar una gran cantidad de usuarios al mismo tiempo.

1.3 Impacto esperado

Se espera que la versión final de ASys sea un gran avance en el modelo de docencia actual. En particular, se espera observar este avance en la Escuela Técnica Superior de Ingeniería Informática, donde ASys se implantará inicialmente.

Por un lado, los alumnos contarán con una plataforma llena de ejercicios para realizar y así poder mejorar sus aptitudes como programadores, contando con estadísticas y recomendaciones en base a sus calificaciones. Esto debería de mejorar el rendimiento académico de los alumnos y verse reflejado en mejores calificaciones en las asignaturas relacionadas con ASys.

Por otro lado, se espera reducir considerablemente el tiempo que invierte el profesorado en la corrección de ejercicios y exámenes, automatizando el proceso en gran medida; además de reducir el factor humano en las correcciones para que los alumnos puedan obtener calificaciones más exactas y objetivas.

Todo esto deberá de estar implementado en un entorno web, que sea fácilmente accesible por todo el mundo y que no requiera instalaciones de ningún tipo por parte de los usuarios. De manera que cualquier usuario con acceso a internet pueda disfrutar de la plataforma en cuestión de segundos.

1.4 Metodología

La metodología seguida a lo largo de este trabajo ha constado de diferentes fases. La primera fase constó a su vez de una serie de reuniones semanales, en las que íbamos definiendo y concretando el trabajo específico a desarrollar y el cómo desarrollarlo.

Después de esto, siguió una segunda fase de formación en la cual iba enviando informes semanales en los que detallaba mis avances. Tras un periodo de formación

personal, vino un periodo de formación grupal en el cual realizamos una serie de reuniones/seminarios para transmitir los conocimientos adquiridos al resto del equipo.

La fase tres se realizó en paralelo en gran parte con la fase dos, y constaba del desarrollo de la plataforma. Por un lado, desarrollaba la plataforma, y por otro explicaba cómo usarla y ampliarla al resto del equipo.

Finalmente, llego una fase de validación del trabajo realizado, y de búsqueda de flecos o posibles deficiencias en el proyecto para apuntarlos y solventarlos.

Este proceso se repitió de forma iterativa en los dos proyectos realizados, tanto en el servidor (*backend*) como en el cliente (*frontend*). A continuación, se puede observar un diagrama de Gantt que ilustra el tiempo empleado en las diferentes fases del trabajo.

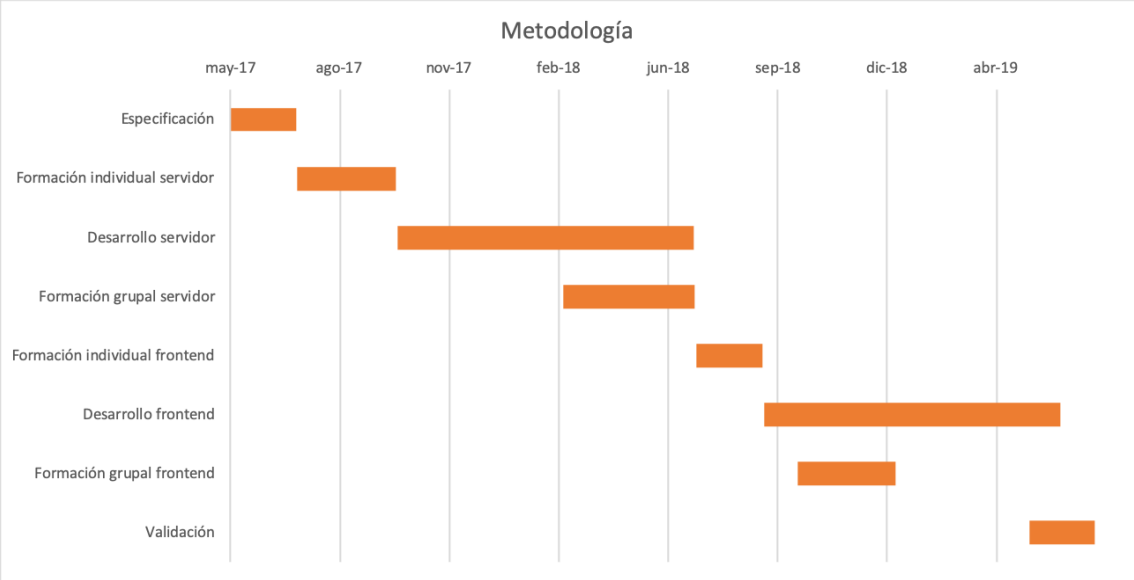


Ilustración 1: Diagrama de gantt tiempo invertido

1.5 Estructura

El trabajo se va a dividir en diferentes secciones, las cuales en su mayoría representan una fase del proceso de desarrollo en un proyecto software. A continuación, se va a realizar una breve explicación de cada punto.

Estado del arte: Se realizará una explicación de la situación actual del producto y los posibles competidores o productos que se asemejen.

Especificación de requisitos: Se detallarán los objetivos a cumplir por el sistema o diferentes sistemas a desarrollar, así como se definirán las características que debe cumplir por los diferentes actores.

Análisis del problema: Se realizará un estudio de cómo alcanzar los requisitos definidos previamente.

Diseño de la solución: Se formalizará el trabajo a desarrollar tras realizar un estudio.

Desarrollo de la solución: Se explicarán los detalles de implementación del sistema.

Implantación: Se detallarán las tareas a realizar para que los usuarios puedan hacer uso del sistema.

Pruebas: Se evaluará el sistema con baterías exhaustivas de pruebas de funcionalidad y de rendimiento.

Conclusiones: Se hará una retrospectiva del trabajo realizado y se escribirá una pequeña valoración personal.

Trabajos futuros: Se detallarán posibles flecos o deficiencias que hayan quedado por solventar en el sistema, así como posibles mejoras que se puedan implantar.

Referencias: Conjunto de enlaces a recursos utilizados para el desarrollo del trabajo.

Glosario: Definición de conjunto de términos técnicos empleados a lo largo del trabajo.

Anexo – Manual de usuario: Se realizará un pequeño manual de como emplear la plataforma.

1.6 Colaboraciones

El trabajo se ha realizado prácticamente en su totalidad por mi parte, con la ayuda y guía de mi tutor Josep Silva Galiana. Menos la parte final en la cual se realiza un pequeño proyecto de corrección automática de ejercicios, con labores de investigación, el cual se ha desarrollado conjuntamente con el alumno Saul Blanco Villorejo.

2 Estado del arte

Actualmente, existen numerosas plataformas que ofrecen facilidades para todas aquellas personas que deseen aprender programación o perfeccionar sus habilidades a través de la web.

Entre el numeroso abanico de plataformas de docencia web, las hay con un planteamiento más manual y personalizado, en las cuales los alumnos se descargan tareas, las realizan en sus ordenadores y acto seguido las envían a la plataforma para que una persona física les corrija de forma manual dicha tarea, otorgándoles una nota como resultado. Este es el enfoque más común, utilizado en numerosas universidades o escuelas para dar soporte a las asignaturas impartidas. Algunos ejemplos podrían ser, la propia plataforma de la UPV poliformaT, la plataforma edX impulsada por grandes universidades de renombre como las universidades de Massachusetts, Harvard, Berkeley entre otras, o plataformas de cursos online como Udemy y Udacity [5, 6, 7, 8].

En contraposición a este planteamiento, existe uno en el cual la corrección de los ejercicios se realiza de forma automática. Este planteamiento es mucho más complejo tecnológicamente hablando debido a la dificultad que supone realizar la corrección por parte de un ordenador. Este enfoque sin embargo tiene la ventaja de ahorrar mucho tiempo al profesorado en la corrección de las tareas, ya que únicamente tienen que supervisarlas, además da un feedback inmediato al alumno puesto que la corrección es instantánea en muchos casos. Algunas plataformas que realizan una corrección automática del código son CodeAcademy y la antigua CodeSchool, ahora unificada en la plataforma PluralSight [9, 10].

El campo de la corrección automática es un campo en el que se están realizando numerosas investigaciones y con ello se están logrando grandes avances. La mayoría de las técnicas para la corrección automática del código se basan en la comparación de la salida (es decir, caja negra) ver, por ejemplo [11, 12, 13, 14, 15], y los estudios [16, 17, 18, 19]. La estrategia común es compilar el código del estudiante, ejecutar este código compilado con una colección de entradas previamente preparadas, y comparar el estándar de oro con los resultados obtenidos. La mayoría de los sistemas novedosos [20, 21, 22, 14], utilizan algún modelo (generalmente el algoritmo original) y la generación aleatoria de casos de prueba para comparar ambos resultados. Algunos sistemas utilizan el compilador como parámetro y solo comparan los resultados, lo que los hace independientes del lenguaje. Una de las herramientas que es más similar al enfoque de ASys para la corrección automática es Web-CAT [23], que también ofrece un sistema de calificación automático e informes para estudiantes y profesores. Sin embargo, hay diferencias importantes. Primero, Web-CAT no es una herramienta independiente. Es un plugin de Eclipse. En contraste, el sistema de corrección de ASys es independiente de cualquier IDE. En segundo lugar, la arquitectura interna de los sistemas de evaluación también es diferente. Además, si bien el motor de prueba es similar en ambos sistemas, los análisis estáticos del código son diferentes. Web-CAT integra herramientas para el análisis estático, y para evaluar la documentación y el estilo de codificación. La herramienta ASys va más allá de esto y también permite utilizar análisis estáticos para verificar las propiedades estructurales del código fuente y corregirlo automáticamente.

También existen enfoques que analizan el código fuente (es decir, enfoques de caja blanca). Dos ejemplos son [24] y [25], donde la idea principal es comparar el código del estudiante con un conjunto de soluciones conocidas, midiendo así la similitud del código del estudiante con respecto a un conjunto de soluciones. El nivel de similitud se calcula con un grafo que representa el código. Esta idea ha sido explotada en otros enfoques. Por ejemplo, la técnica propuesta en [24] utiliza la similitud de grafos para cuantificar la similitud estructural entre las presentaciones de estudiantes ya calificadas y las no

calificadas. De manera similar, en [26], los autores complementan las pruebas con dos enfoques diferentes: verificación y similitud de CFG. La verificación se utiliza para detectar errores con la herramienta LAV. Esto se hace localizando errores como división por cero, errores de puntero y desbordamientos de búfer; y comprobando estáticamente las aserciones del programa. La similitud de CFG se utiliza para calificar. Otro enfoque es [27], donde los envíos se transforman en grafos de dependencias extendidos que combinan el control y los flujos de datos. Luego, aprovechan las técnicas de comparación de subgrafos para calcular los comentarios personalizados adecuados. Además, las restricciones que relacionan los patrones permiten realizar evaluaciones detalladas. Otra herramienta basada en análisis estáticos es Algo+ [15]. Todas estas técnicas, especialmente aquellas que utilizan algún tipo de similitud, han demostrado ser útiles y podrían combinarse con nuestra técnica.

Esta área de investigación se está volviendo muy popular debido a la creciente necesidad de sistemas de corrección automática en MOOC (Cursos masivos en línea) [28, 27, 29, 15]. Una discusión de los desafíos en esto de la evaluación dentro de los MOOC aparece en [30]. Las herramientas de corrección automática más importantes se han analizado en una revisión sistemática. Las revisiones anteriores se pueden encontrar en [16, 18, 17, 19].

El método utilizado en ASys para la corrección automática de código, también utiliza la generación automática de pruebas y la comparación de resultados. Sin embargo, en lugar de utilizar un grafo, un modelo o un conjunto de soluciones, puede inferir todas las propiedades que deben validarse a partir de la solución proporcionada por el profesor. La solución del profesor es compilada y considerada como un oráculo para la generación de casos de prueba. Además, los casos de prueba no se generan de forma aleatoria, se generan teniendo en cuenta las condiciones de la ruta del programa, maximizando de este modo la cobertura de su código.

El módulo de verificación de propiedades en ASys es novedoso. Es la diferencia más notable entre nuestro enfoque y otros enfoques. Según nuestro conocimiento, no existe ninguna otra técnica capaz de especificar propiedades por medio de un DSL que pueda validarse, corregirse y calificarse automáticamente de la forma en que lo hacemos. Esto podría ser un complemento interesante para los enfoques de caja blanca discutidos. Además, en el sistema ASys como plataforma web tiene una orientación muy distinta al de otras escuelas virtuales, en el cual se traslada la corrección automática de los ejercicios al propio ordenador del usuario, liberando de esta carga al servidor, realizando correcciones más rápidas dependiendo del equipo del usuario, y añadiendo un soporte offline al no necesitar uso de la red.

2.1 Critica al estado del arte

ASys cuenta con una primera versión web con una construcción tradicional (implementado en HTML, CSS y Javascript sin usar ningún *framework* de desarrollo), la cual hace un uso intensivo de los recursos del servidor y no tanto de los recursos del navegador o equipo del usuario. Este enfoque era muy típico hace algunos años, cuando los equipos de los usuarios eran poco potentes y se necesitaba que el servidor realizara la mayor parte del cómputo. Dejando a los navegadores con una lógica muy básica, la cual constaba práctica y únicamente de la interacción. Este enfoque más tradicional, requiere una mayor inversión en infraestructura ya que es en esta donde se realiza la mayor parte del trabajo.

2.2 Propuesta

El motivo que ha provocado la realización de este TFG es el de migrar la plataforma web actual de ASys a un enfoque más actual, acorde a las tendencias seguidas por el mercado.

La tendencia actual, y lo que se va a realizar en este trabajo, es la de desarrollar una aplicación web y un nuevo servidor que traslade la mayor parte posible del cómputo de los servidores a las máquinas de los usuarios. Invirtiendo más esfuerzo en las aplicaciones web que se ejecutan en los equipos de los usuarios, y reduciéndolo en los servidores. Esto viene dado por el incremento en la capacidad de computo en los dispositivos de los usuarios de a pie, los cuales cada vez están más preparadas para ejecutar tareas de mayor complejidad. Esto tiene numerosas ventajas, entre las cuales se encuentran, una mejora significativa en la experiencia de usuario y un ahorro considerable en los costes de infraestructura.

3 Análisis del problema

3.1 Especificación de requisitos

3.1.1 Visión

El propósito de este apartado es definir formalmente los requerimientos del producto que se va a desarrollar. Para ello, se proporciona una visión general de todo lo relacionado con el sistema ASys, como por ejemplo los actores que interactuarán con el sistema o las características que va a poseer. Además, servirá de apoyo a la hora de determinar si el sistema ha pasado o no ha pasado las pruebas en base a su especificación, teniendo en cuenta que la finalidad de este trabajo es la de crear una estructura para la construcción del sistema y no su finalización. Esta especificación de requisito se llevará a cabo siguiendo un proceso semi-formal, el cual consta de utilizar una especificación de requisitos en lenguaje natural, acompañado de diagramas para sustituir o complementar el lenguaje natural [31, 32].

3.1.1.1 Actores

Un actor representa un conjunto coherente de roles que los usuarios de los casos de uso representan al interactuar con el sistema (tipos de usuarios).

- *Usuario Anónimo*: Persona que hace uso del sistema y aún no ha sido identificado.
- *Usuario Autenticado*: Persona que hace uso del sistema y ha sido previamente identificado.
- *Estudiante*: Persona que hace uso del sistema, ha sido identificado y cuya finalidad es la realización de ejercicios.
- *Profesor*: Persona que hace uso del sistema, ha sido identificado y cuya finalidad es la de crear ejercicios en la plataforma y corregir soluciones realizadas por los alumnos.
- *Administrador*: Persona que hace uso del sistema, ha sido identificado y cuya finalidad es la de modificar cualquier información contenida en el sistema.

3.1.1.2 Características

Las características representan requisitos de alto nivel del sistema software, los cuales a posteriori se deben descomponer en requisitos más específicos denominados casos de uso.

El sistema se compondrá de tres artefactos software independientes que interactuarán para llevar a cabo las funciones que se definen en este documento. Podemos resumir las funciones principales de cada artefacto así:

Artefacto software 1: ASys Web Client (Aplicación web)

- *Sistema de autenticación*: Todas las operaciones necesarias para que un usuario pueda entrar en la sección privada de la aplicación.
- *Gestión de ejercicios*: Todas las operaciones relacionadas con los ejercicios y su realización.
- *Ajustes*: Tareas relacionadas con la configuración de la aplicación.

Artefacto software 2: ASys Web Server (Servidor REST)

- *Sistema de autenticación*: Todas las operaciones necesarias para que un usuario pueda tener acceso a la sección privada del servidor.
- *Gestión de ejercicios*: Todas las operaciones relacionadas con los ejercicios y su realización.
- *Gestión de usuarios*: Todas las tareas relacionadas con la modificación de la información sobre los usuarios.

Artefacto software 3: ASys Client (Aplicación de escritorio)

- *Sistema de corrección*: Todas las tareas cuyo fin es entregar información sobre una solución a un ejercicio.

3.1.1.3 Requisitos No Funcionales

Los requisitos no funcionales son un tipo de requisitos en los cuales se especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales o casos de uso.

- *Fiabilidad en la aplicación*: El usuario debe recibir la información mínima sobre errores.
- *Seguridad y protección de datos*: La aplicación debe cumplir con la normativa vigente.
- *Facilidad de uso*: El usuario debe ser capaz de usar la aplicación de forma intuitiva.
- *Tiempos de respuesta bajos*: El usuario no debe recibir esperas prolongadas.
- *Alto nivel de carga*: Capacidad para soportar muchos usuarios en la aplicación al mismo tiempo.
- *Disponibilidad*: La aplicación no debe dejar de funcionar aunque se provoquen fallos.
- *Transparencia de acceso*: El usuario no debe saber con cuantas entidades se está comunicando el sistema.
- *Portabilidad*: La aplicación se debe poder usar en cualquier sistema de escritorio.

3.1.1.4 Modelo de dominio

El modelo de dominio es un diagrama de clases UML en el cual solo se representan aspectos del dominio, pero no aspectos del diseño. Algunos de sus objetivos son el identificar y nombrar conceptos importantes en el contexto del sistema a desarrollar (entidades y eventos), establecer un lenguaje común entre clientes y desarrolladores, e identificar y nombrar las relaciones en estos conceptos [33].

3.1.1.5 Modelo de contexto

El modelo de contexto es un diagrama de clases UML, en el cual se representan o se definen los límites entre los subsistemas o partes del sistema y su ambiente, mostrando las entidades que interactúan con él. Este tipo de modelos son una vista de alto nivel de un sistema.

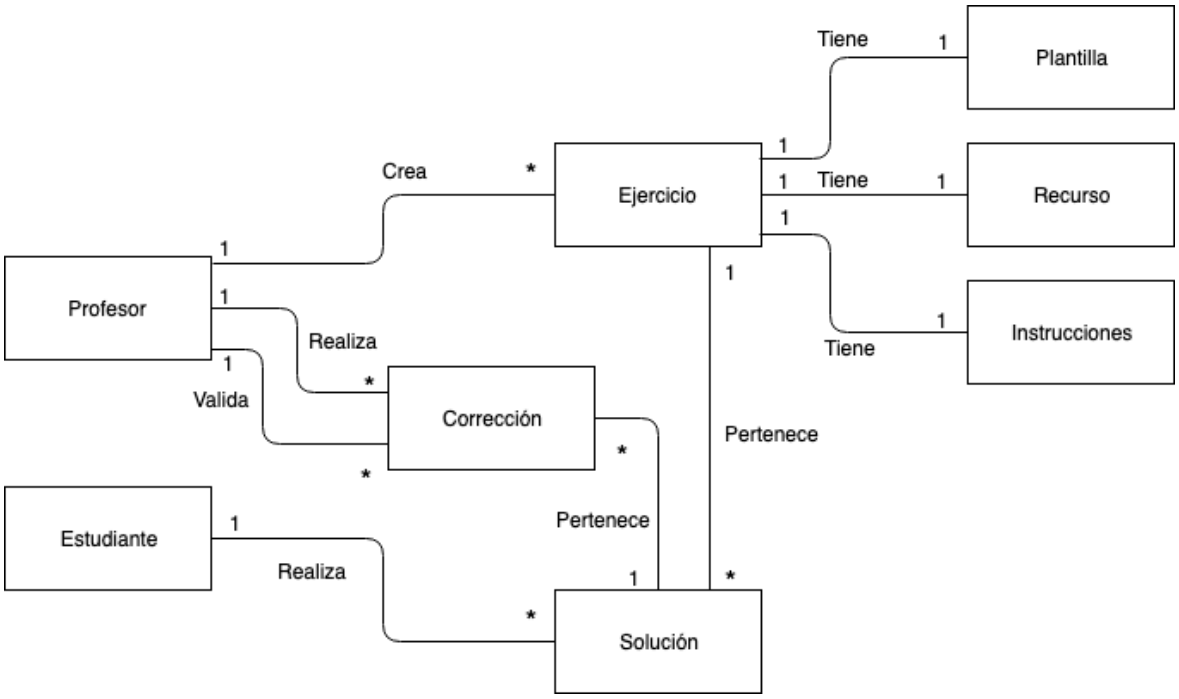


Ilustración 2: Modelo de dominio

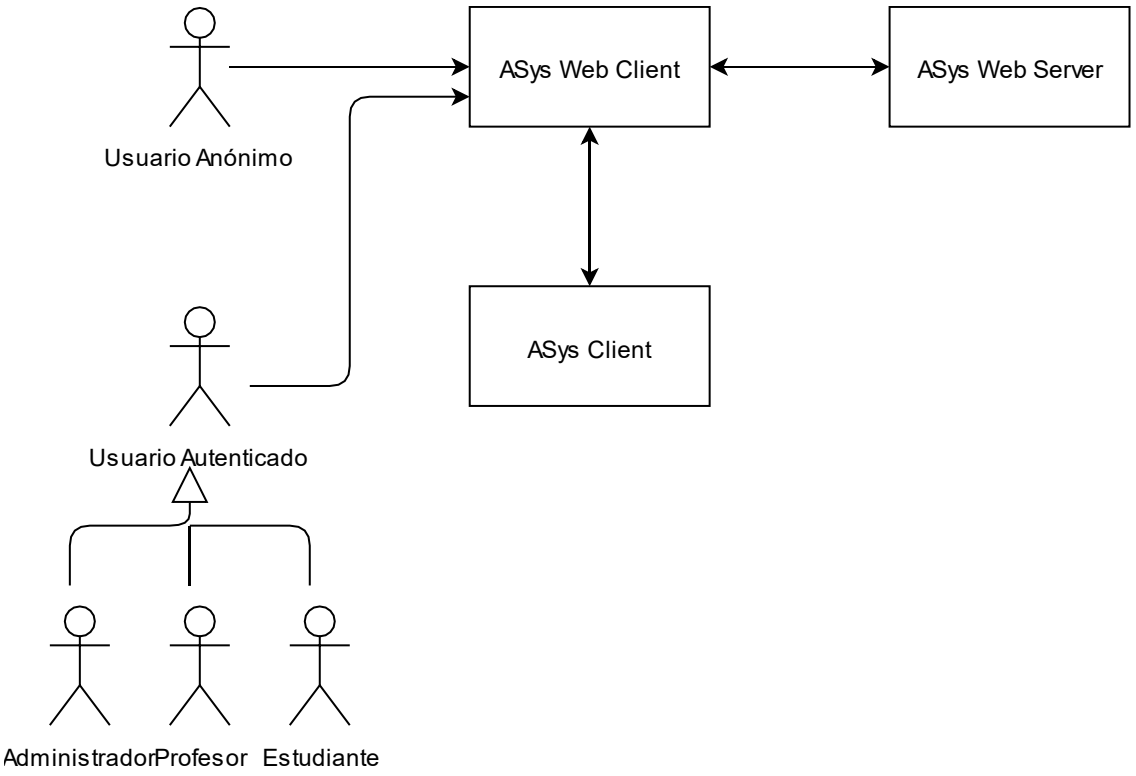


Ilustración 3: Modelo de contexto

3.1.2 Casos de Uso

El objetivo de esta sección es definir los casos de uso de acuerdo a las características del sistema, proporcionando los diagramas de casos de uso para cada característica.

Además, se proporciona información sobre cada caso de uso (nombre, definición, actores implicados y precondición).

3.1.2.1 Sistema de autenticación (ASys Web Client)

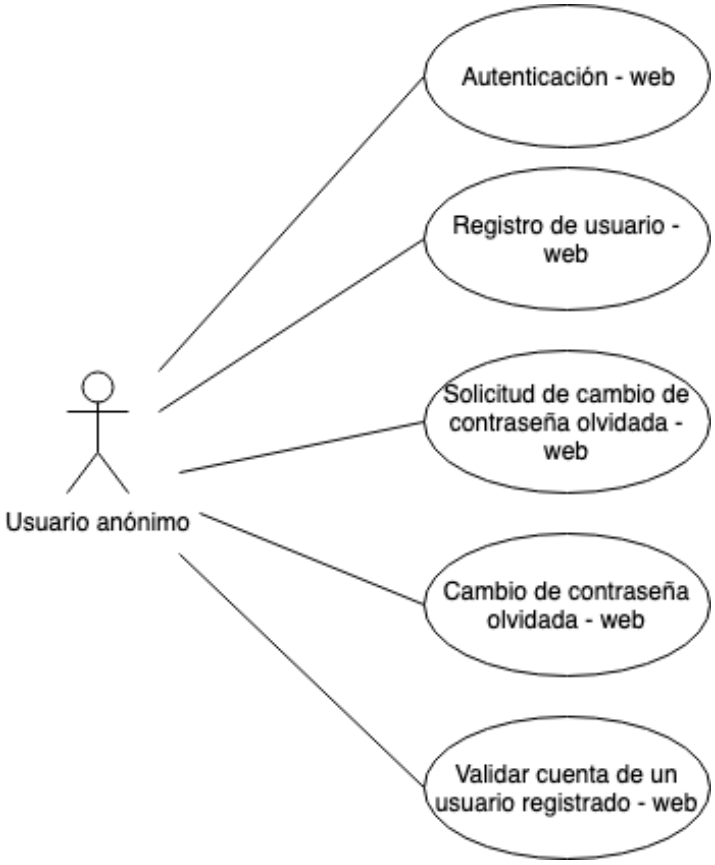


Ilustración 4: Diagrama de casos de uso autenticación web

Autenticación - web

Descripción	Un usuario no autenticado con una cuenta validada deberá poder autenticarse con sus credenciales mediante un formulario. Si se ha efectuado con éxito será enviado a la zona privada de la aplicación. En caso de no superar la autentificación, se mostrará un mensaje de error.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta validada.
Postcondición	Autenticación del usuario.

Registro de usuario - web

Descripción	Un usuario no autenticado deberá poder rellenar un formulario para crear una cuenta. Indicando como mínimo su nombre de usuario, contraseña y correo. El usuario debe ser único; por lo que en caso de estar en uso se notificará.

	La contraseña habrá que introducirla dos veces para evitar confusiones. El correo debe de estar en el formato correcto y debe ser único. En caso de estar en uso se notificará.
Actor	Usuario anónimo.
Precondición	Que la cuenta de correo y el nombre de usuario no estén en uso.
Postcondición	Creación de cuenta de usuario pendiente de validar.

Solicitud de cambio de contraseña olvidada - web

Descripción	Un usuario anónimo podrá rellenar un formulario indicando su dirección de correo para realizar la solicitud de cambio de contraseña por olvido. En caso de no existir una cuenta con el correo indicado se notificará un error.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta validada.
Postcondición	Envió de email con un enlace para cambiar la contraseña.

Cambio de contraseña olvidada - web

Descripción	Un usuario, tras pulsar en el enlace recibido por correo, será redirigido a una página donde visualizará un formulario. En este formulario deberá introducir la nueva contraseña dos veces por motivos de seguridad. Tras esto, se efectuará el cambio de contraseña y se redirigirá a la zona privada de la aplicación. En caso de no coincidir las contraseñas se notificará el error.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta validada. Haber solicitado el cambio de contraseña por olvido correctamente.
Postcondición	Cambio de la contraseña en la cuenta del usuario.

Validar cuenta de un usuario registrado - web

Descripción	Un usuario no autenticado que haya completado el formulario de registro recibirá un correo con un enlace que al seleccionarlo redirigirá al usuario a una página donde se efectuará la validación. Tras esto, se le redireccionará a la zona privada de la aplicación.
Actor	Usuario anónimo.

Precondición	Haber cumplimentado correctamente el formulario de registro.
Postcondición	Validación de la cuenta de usuario.

3.1.2.2 Gestión de ejercicios (ASys Web Client)

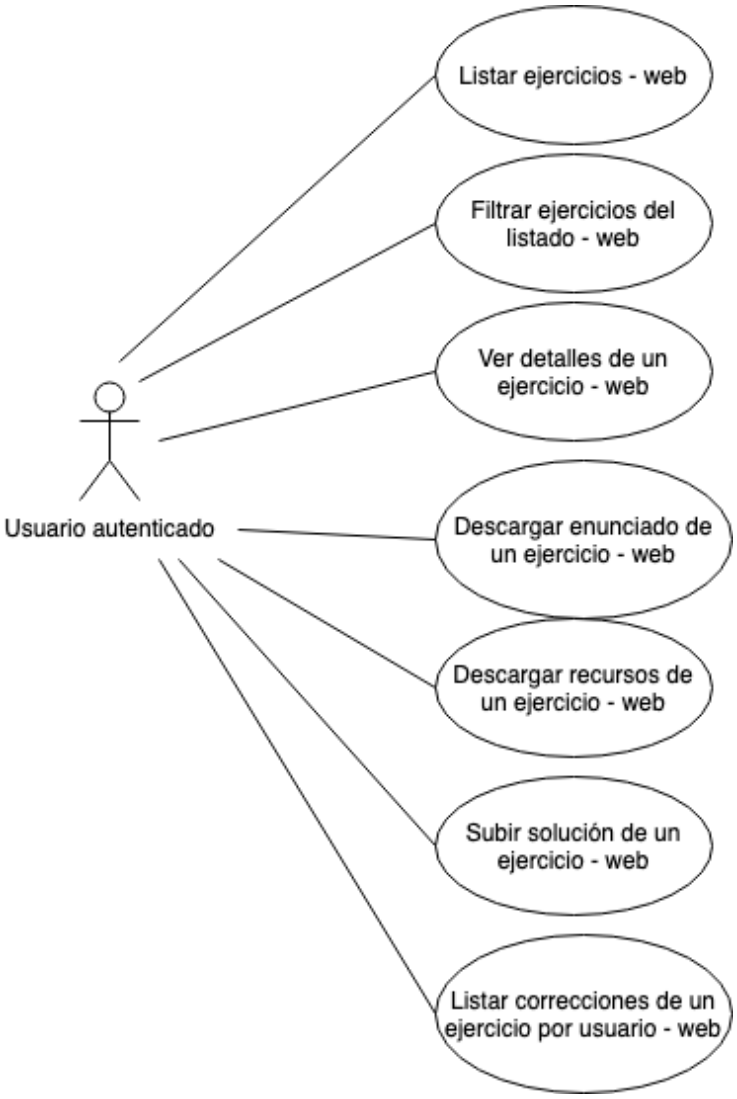


Ilustración 5: Diagrama de casos de uso gestión de ejercicios web

Listar ejercicios - web

Descripción	Se deberá poder observar los diferentes ejercicios en una página, la cual se ira completando con más ejercicios a medida que el usuario lo desee con algún sistema de paginación.
Actor	Usuario autenticado.
Precondición	Estar situado en la página correspondiente al listado de ejercicios.
Postcondición	Mostrar el listado de ejercicios.

Filtrar ejercicios del listado - web

Descripción	Un usuario podrá filtrar los ejercicios mostrados en el listado en base a diferentes criterios de búsqueda.
Actor	Usuario autenticado.
Precondición	Estar visualizando el listado de ejercicios.
Postcondición	Mostrar listado de ejercicios con el filtro indicado.

Ver detalles de un ejercicio - web

Descripción	Un usuario debe poder seleccionar un ejercicio que este visualizando para ver más información sobre el mismo.
Actor	Usuario autenticado.
Precondición	Estar visualizando un ejercicio.
Postcondición	Visualizar todos los datos relativos a un ejercicio en una ventana emergente.

Descargar enunciado de un ejercicio - web

Descripción	Un usuario autenticado debe poder descargar el enunciado de un ejercicio para poder realizarlo.
Actor	Usuario autenticado.
Precondición	Pulsar el botón correspondiente a la descarga
Postcondición	Descargar el enunciado.

Descargar recursos de un ejercicio - web

Descripción	Un usuario autenticado deberá poder descargarse los recursos de un ejercicio para poder realizarlo.
Actor	Usuario autenticado.
Precondición	Pulsar el botón correspondiente a la descarga.
Postcondición	Descargar el recurso.

Subir solución de un ejercicio - web

Descripción	Un usuario autenticado deberá poder subir la solución a un ejercicio para que se le efectué la corrección.
Actor	Usuario autenticado.
Precondición	Añadir la solución a la página.
Postcondición	Guardar la solución.

Listar correcciones de un ejercicio por usuario - web

Descripción	Un usuario autenticado que haya subido soluciones de un ejercicio deberá poder observar los datos de este en un listado.
Actor	Usuario autenticado.
Precondición	Haber subido una solución a un ejercicio.
Postcondición	Mostrar un listado con el histórico.

3.1.2.3 Ajustes (ASys Web Client)

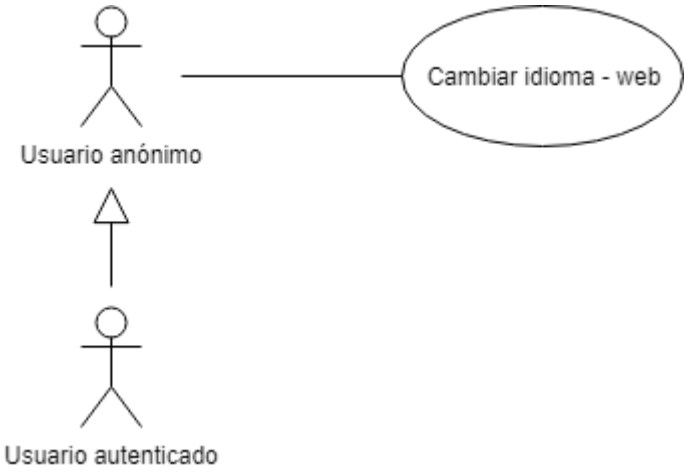


Ilustración 6: Diagrama de casos de uso ajustes web

Cambiar idioma - web

Descripción	El usuario deberá poder cambiar el idioma de la aplicación a su gusto dado un número de idiomas soportados. Inicialmente Inglés y Español.
Actor	Usuario anónimo y usuario autenticado.
Precondición	Seleccionar un idioma diferente al que está en uso.
Postcondición	Cambiar el idioma de la aplicación.

3.1.2.4 Sistema de autenticación (ASys Web Server)

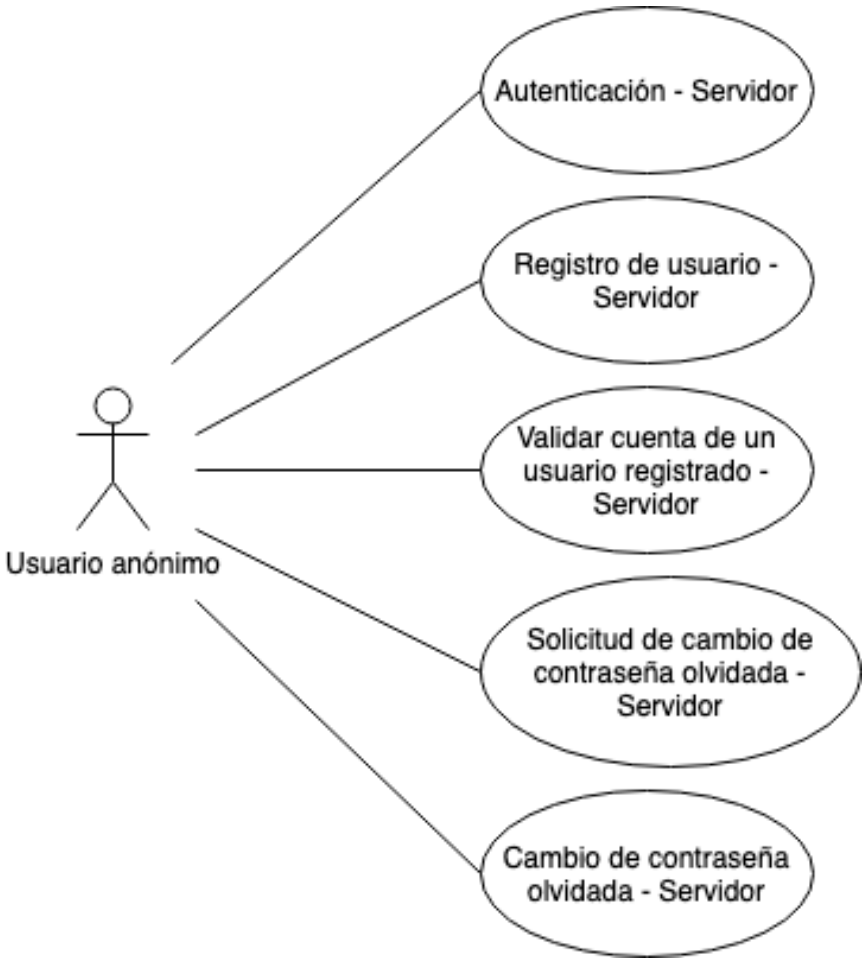


Ilustración 7: Diagrama de casos de uso autenticación servidor

Autenticación - Servidor

Descripción	Dados los datos de un usuario y una contraseña, en caso de ser correctos, se entregará un token o código de sesión que identifique que ese usuario está autenticado. En caso contrario se notificará del error.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta validada, enviar las credenciales del usuario.
Postcondición	Devolver un token identificativo.

Registro de usuario - Servidor

Descripción	Dados los datos de usuario, correo y contraseña, se crearán los datos de un usuario no validado y se enviará un correo a la cuenta indicada con un código autogenerado para poder realizar la validación. En caso de existir el usuario o el correo se notificará del error.

Actor	Usuario anónimo.
Precondición	Enviar los datos del nuevo usuario.
Postcondición	Crear un usuario pendiente de validar y enviar un correo con enlace para validar la cuenta.

Validar cuenta de un usuario registrado - Servidor

Descripción	Dado un código previamente generado y enviado, se identificará el usuario pertinente y se efectuará la validación del mismo.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta no validada y enviar el código del correo del usuario.
Postcondición	Validar la cuenta del usuario.

Solicitud de cambio de contraseña olvidada - Servidor

Descripción	Dado un correo, se comprobará que existe un usuario con dicho correo y en caso afirmativo se generará y enviará un código a la dirección para poder realizar el cambio. En caso de no existir un usuario con dicho correo se notificará.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta validada y enviar el correo de la cuenta.
Postcondición	Enviar un correo al usuario con un enlace para poder realizar un cambio de contraseña.

Cambio de contraseña olvidada - Servidor

Descripción	Dado un código autogenerado y una contraseña, se identificará el usuario pertinente y se efectuará el cambio de contraseña.
Actor	Usuario anónimo.
Precondición	Poseer una cuenta validada y haber solicitado el cambio de contraseña por olvido. Enviar la nueva contraseña.
Postcondición	Realizar un cambio de contraseña en la cuenta del usuario.

3.1.2.5 Gestión de ejercicios (ASys Web Server)

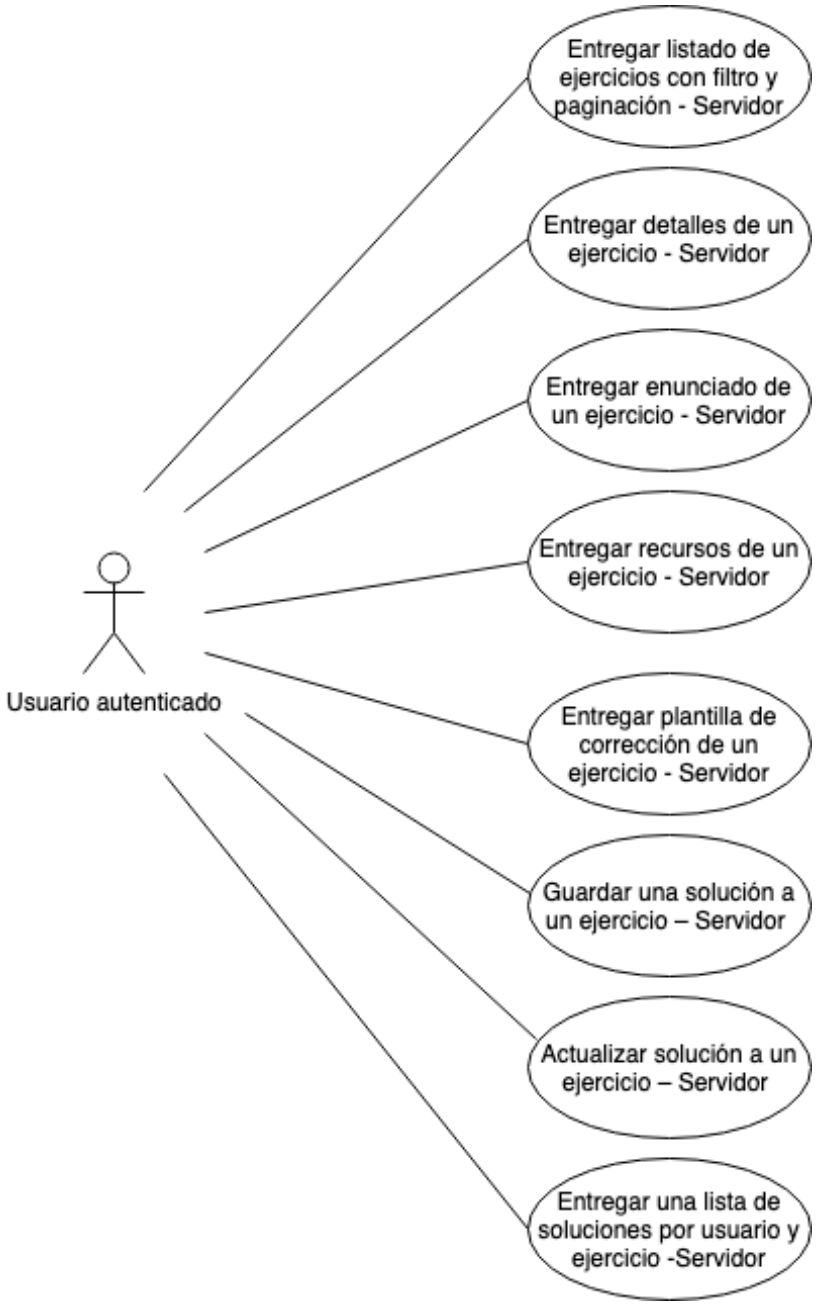


Ilustración 8: Diagrama de casos de uso gestión de ejercicios servidor

Entregar listado de ejercicios con filtro y paginación - Servidor

Descripción	Dados los datos: tamaño de página, número de página y un filtro. Se entregarán la información de los ejercicios que cumpla estas características.
Actor	Usuario autenticado.
Precondición	Entregar la información relativa al filtro la paginación.
Postcondición	Entregar listado de ejercicios correspondientes al filtro indicado.

Entregar detalles de un ejercicio - Servidor

Descripción	Dado el identificador de un ejercicio se entregará la información de este. En caso de no existir un ejercicio con ese identificador se notificará del error.
Actor	Usuario autenticado.
Precondición	Entregar el identificador del ejercicio del que se desee la información.
Postcondición	Devolver toda la información relativa a un ejercicio.

Entregar enunciado de un ejercicio - Servidor

Descripción	Dado un identificador de ejercicio se devolverá su enunciado en formato pdf. En caso de no existir un ejercicio con ese identificador se notificará del error.
Actor	Usuario autenticado.
Precondición	Debe existir el ejercicio implicado y debe tener un enunciado asociado. Enviar el identificador del ejercicio.
Postcondición	Devolver el enunciado en formato pdf de un ejercicio.

Entregar recursos de un ejercicio - Servidor

Descripción	Dado un identificador de ejercicio se devolverán sus recursos asociados comprimidos en un fichero zip. En caso de no existir un ejercicio con ese identificador se notificará del error.
Actor	Usuario autenticado.
Precondición	Debe existir el ejercicio implicado y debe tener un zip de recursos asociado. Enviar el identificador del ejercicio.
Postcondición	Devolver los recursos de un ejercicio comprimidos en un zip.

Entregar plantilla de corrección de un ejercicio - Servidor

Descripción	Dado un identificador de ejercicio se devolverá la plantilla de corrección en formato zip de este. En caso de no existir un ejercicio con ese identificador se notificará del error.

Actor	Usuario autenticado.
Precondición	Debe existir el ejercicio implicado y debe tener un zip con la plantilla asociado. Enviar el identificador del ejercicio.
Postcondición	Devolver la plantilla de corrección de ejercicio en formato zip.

Guardar una solución a un ejercicio – Servidor

Descripción	Dado un zip y un identificador del ejercicio, se guardará una solución con los datos pertinentes. Entre los cuales se deberá de encontrar la fecha de entrega.
Actor	Usuario autenticado.
Precondición	Adjuntar una solución en formato zip. Enviar el identificador del ejercicio.
Postcondición	Almacenar la solución en formato zip e insertar los datos en el histórico del alumno.

Actualizar solución a un ejercicio – Servidor

Descripción	Dado un identificador de solución y los datos que cambiar se actualizará aquella información que haya cambiado. En caso de no existir dicha solución se notificará de un error.
Actor	Usuario autenticado.
Precondición	Entregar los datos a cambiar en la solución. Enviar el identificador de la solución.
Postcondición	Modificar los datos de una solución con los nuevos datos recibidos.

Entregar una lista de soluciones por usuario y ejercicio - Servidor

Descripción	Dado un identificador de usuario y un identificador de ejercicio, se entregará un listado con la información de las soluciones de ese usuario en ese ejercicio.
Actor	Usuario autenticado.
Precondición	Debe existir el ejercicio implicado.
Postcondición	Devolver los datos relativos a las soluciones de un usuario en el ejercicio.

3.1.2.6 Gestión de usuarios (ASys Web Server)

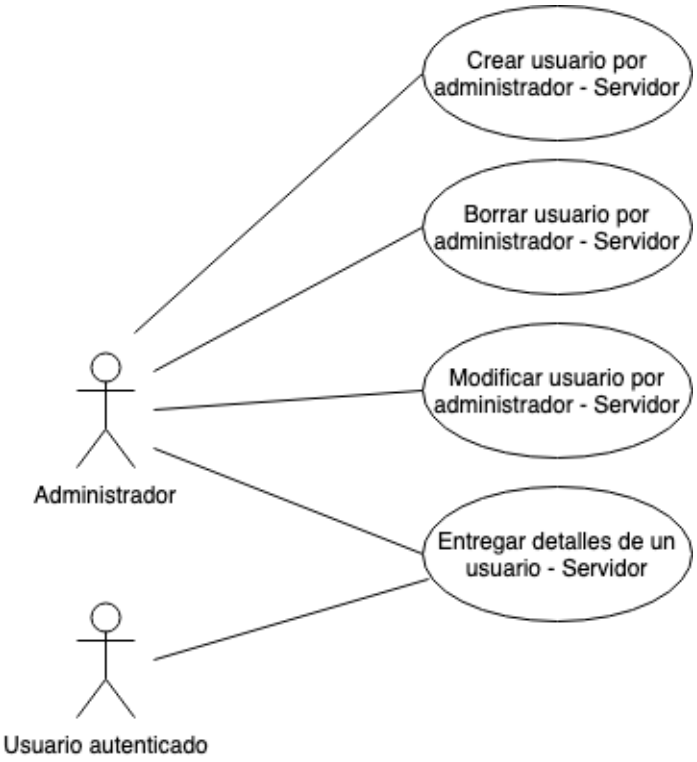


Ilustración 9: Diagrama de casos de uso gestión de usuarios servidor

Crear usuario por administrador - Servidor

Descripción	Dados los datos de un usuario se debe guardar la información de este. Comprobando que el nombre y el correo no están en uso por otros usuarios. En caso de que ya existan se notificara del error.
Actor	Enviar datos del nuevo usuario.
Precondición	Crear un nuevo usuario con los datos indicados.

Borrar usuario por administrador - Servidor

Descripción	Dado el identificador de un usuario, se eliminará la información de datos personales de su cuenta, pero no la información relacionada con ejercicios asociada a la misma. Como puedan ser las soluciones subidas, correcciones etc... Se notificará el éxito o el fracaso de la operación en caso de fallo.
Actor	Administrador.
Precondición	Entregar el identificador del usuario.
Postcondición	Eliminar la cuenta del usuario.

Modificar usuario por administrador - Servidor

Descripción	Dados los datos de una cuenta de un usuario se sobrescribirán la información del este que haya cambiado. Se notificará el éxito o el fracaso de la operación en caso de fallo.
Actor	Administrador.
Precondición	Enviar los datos a cambar relativos a un usuario.
Postcondición	Modificar la información de un usuario con los nuevos datos.

Entregar detalles de un usuario - Servidor

Descripción	Dado un identificador de usuario se proveerá de la información de la cuenta asociada a ese usuario. Se notificará de un error en caso de no existir el usuario correspondiente.
Actor	Administrador y usuario autenticado.
Precondición	En el caso del usuario autenticado debe ser el mismo que el solicitado. Indicar el identificador del usuario.
Postcondición	Enviar información de un usuario.

3.1.2.7 Sistema de corrección (ASys Client)

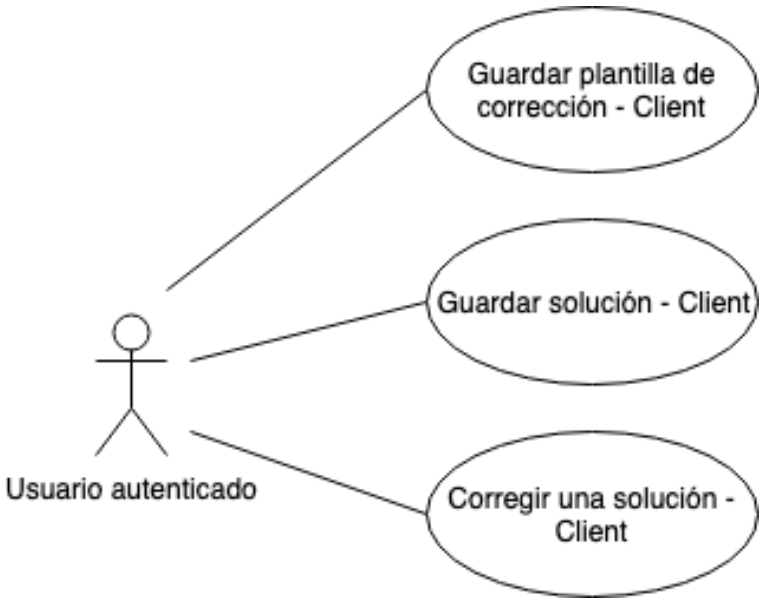


Ilustración 10: Diagrama de casos de uso corrección

Guardar plantilla de corrección - Client

Descripción	Dado un zip con una plantilla de corrección, se guardará y se descomprimirá en una carpeta localizada. En caso de poseer una ya guardada se eliminará.
Actor	Usuario autenticado.
Precondición	La plantilla deberá de cumplir con el formato pertinente.
Postcondición	Almacenar la plantilla de corrección descomprimida.

Guardar solución - Client

Descripción	Dado un zip con la solución, se guardará y se descomprimirá en una carpeta localizada. En caso de poseer una ya guardada se eliminará.
Actor	Usuario autenticado.
Precondición	La solución deberá de cumplir con el formato pertinente.
Postcondición	Almacenar la solución del ejercicio descomprimida.

Corregir una solución - Client

Descripción	Teniendo una plantilla de corrección y la solución a un ejercicio se procederá a realizar la corrección y se entregará la información pertinente.
Actor	Usuario autenticado
Precondición	Haber guardado previamente la plantilla y una solución. La solución deberá de ser del mismo ejercicio que la plantilla guardada.
Postcondición	Entregar las propiedades de la corrección de un ejercicio con las calificaciones correspondientes.

3.2 Análisis de riesgos

ASys es un sistema que va a ser usado por una gran cantidad de personas y por tanto está sujeto a diversos riesgos que debemos controlar. En particular, es importante crear mecanismos de seguridad para proteger el sistema y es importante prever el coste que las contramedidas tomadas tendrán, ya que los recursos necesarios (especialmente humanos y hardware) son costosos.

A continuación, se van a definir los principales riesgos detectados en el sistema. Se van a clasificar en tres tipos, de aceptación, de satisfacción y tecnológicos o de

integración. Para cada riesgo se van a definir dos apartados, el impacto que podrá producir el riesgo y las contramedidas a tomar para reducir ese impacto.

3.2.1 Riesgos de aceptación

Título	El diseño de la interfaz gráfica no cumple visualmente con lo esperado por el cliente
Impacto que producirá	Aumento en el tiempo de desarrollo y retrabajo
Medidas que se tomarán	Se validarán las interfaces frecuentemente para asegurar que se está cumpliendo con la expectativa esperada.

Título	El sistema no cumple con el rendimiento esperado
Impacto que producirá	No soportará los usuarios concurrentes necesarios y por tanto puede dar lugar a lentitud o caídas. Retomar el desarrollo para mejorar el sistema. Aumentar los recursos de la infraestructura donde se despliega.
Medidas que se tomarán	Se realizarán pruebas de rendimiento progresivas, aumentando poco a poco la cantidad de usuarios concurrentes para medir la escalabilidad y averiguar cuál es el límite. Se realizará un análisis y desarrollo del sistema teniendo como prioridad la optimización.

3.2.2 Riesgos de satisfacción

Título	A los usuarios les resulta complicado de usar el sistema.
Impacto que producirá	Aumento de tiempo requerido por el usuario para emplear la plataforma. Descontento en el usuario
Medidas que se tomarán	Se realizará un manual de usuario para facilitar el aprendizaje. Se validará la herramienta con diferentes usuarios para obtener retroalimentación.

Título	El sistema puede ser difícil de ampliar por los desarrolladores

Impacto que producirá	Abandono del sistema para desarrollar uno nuevo desde cero. Grandes cantidades de tiempo a invertir por los desarrolladores para ampliar el sistema.
Medidas que se tomarán	Se realizará un diseño y desarrollo teniendo como objetivo que el sistema sea fácilmente mantenible.

3.2.3 Riesgos tecnológicos y de integración

Título	Fin del soporte a las herramientas utilizadas
Impacto que producirá	No se recibirán actualizaciones de las herramientas. En caso de haber errores en estas, habrá que solucionarlos a mano. Se estará abierto a vulnerabilidades.
Medidas que se tomarán	Se elegirán cuidadosamente las herramientas utilizadas para el desarrollo. Se realizará un análisis tecnológico basándose en la fiabilidad de las herramientas a largo plazo.

Título	El sistema no es escalable en cuanto a recursos
Impacto que producirá	No se podrán ampliar adecuadamente los recursos del sistema. No se podrán llevar a cabo mejoras que requieran de un mayor uso de recursos.
Medidas que se tomarán	Se llevará a cabo un diseño y desarrollo teniendo presente que debe ser un sistema escalable. Por otro lado, se podrá renunciar a ampliar el sistema con funcionalidades que requieran muchos recursos.

3.3 Identificación y análisis de soluciones posibles

En este apartado se van a explicar las diferentes tecnologías tenidas en cuenta para el desarrollo del sistema, sus virtudes y los posibles defectos que puedan poseer. Este análisis se realizará teniendo en cuenta los riesgos del apartado anterior para poder minimizarlos en gran medida.

El análisis se va a dividir en tres partes, una para el planteamiento del sistema web en general y otras dos según el tipo de proyecto. Con respecto a los proyectos tendremos,

por un lado, las tecnologías estudiadas para el desarrollo del servidor, y por otro lado tendremos aquellas tecnologías estudiadas para el desarrollo de la aplicación web.

3.3.1 Análisis de arquitectura

En la primera versión de ASys se siguió una arquitectura basada en un monolito altamente acoplado y con un gran uso de las sesiones. Este planteamiento desencadenó numerosos problemas de escalabilidad, mantenimiento y rendimiento; y fue el motivo por el cual se produjo la necesidad de realizar una nueva versión con otra aproximación. Algunos de estos problemas pueden ser:

Problemas de disponibilidad en el despliegue: Al tener la aplicación web y el servidor en el mismo proyecto, cuando se realiza un despliegue por haber añadido cambios a uno de los dos proyectos, ambos dejan de funcionar, aunque uno de ellos no se haya visto afectado por estos cambios.

Limitación ante cambios tecnológicos: Una vez se han desarrollado ambos sistemas en una tecnología, ya no se podrán cambiar, pues al estar tan fuertemente acoplados es muy probable que si hay cambios en una parte deje de funcionar el resto.

El uso de sesiones y páginas generadas en el servidor: El uso de sesiones es el culpable del alto acoplamiento entre la aplicación web y el servidor, pues requiere que tanto la aplicación web como el servidor se ejecuten en el mismo contexto para tener acceso a las variables comunes. Es difícil de depurar pues resulta muy complicado seguir la traza de ejecución ya que en cualquier punto del sistema se pueden obtener, añadir y modificar las variables utilizadas para cierta funcionalidad. Es difícil de mantener pues un cambio en el contenido de una sesión puede dar lugar a comportamientos inesperados en cualquier punto del resto de la aplicación.

3.3.1.1 Autenticación

Teniendo en cuenta los problemas mencionados anteriormente y los posibles riesgos, se optó por una aproximación STATELESS, es decir, sin estado [33]. Para así dejar atrás el uso de sesiones que tantos problemas ha ocasionado.

Las aproximaciones STATELESS hacen uso de un token identificativo de cada usuario, el cual sirve para acceder a los datos mínimos y necesarios del mismo. El uso de tokens provoca que los servidores no guarden información de contexto y cada llamada o cada acción sea totalmente independiente de la anterior. Por tanto, el servidor únicamente sabrá la información del usuario a través de un token recibido en cada petición.

Esto permite tener múltiples aplicaciones que consuman información/recursos del servidor (API). Pues con que una aplicación posea el token del usuario ya puede acceder al resto de recursos. Por tanto, ya abrimos la puerta a tener tanto una aplicación web como una aplicación de escritorio que consuman los recursos en el mismo lugar centralizado, el servidor común. Además, también da pie a poder descomponer el servidor en diferentes módulos y máquinas puesto que únicamente requieren de un token que puede ser transferido y no necesitan ejecutarse en el mismo contexto para tener acceso a la información del usuario.

Para la autentificación del usuario y el uso de tokens existen varias aproximaciones: OAuth 2.0, OpenID Connect y Json Web Tokens (JWT) [34, 35, 36, 37].

Antes de entrar a comparar los diferentes sistemas de tokens hay que distinguir lo que es la autenticación de lo que es la autorización [38]:

La autentificación es el proceso de verificar una identidad, es decir confirmar que una persona es quien dice ser.

La autorización es el proceso de verificar lo que un usuario puede hacer.

Dicho esto, OAuth es un sistema de autorización, pero no de autenticación y OpenID es un sistema de autenticación que usa un estándar para el formato de los tokens (JWT). Mientras OAuth es necesario combinarlo con otra tecnología para la autentificación, en OpenID accedes a la identidad del usuario directamente.

3.3.1.2 Estándares de Servidores

Teniendo claro que se desea un servidor STATELESS, actualmente existen varias aproximaciones muy usadas en la industria. Por un lado, tenemos los servidores en forma de monolito y los servidores con una arquitectura de microservicios. Además, existe el estándar REST para diseñarlos como una interfaz desacoplada.

REST no es una arquitectura al uso sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software [33]. Por tanto. ¿qué restricciones debe cumplir el servidor para ser REST?

- Servidor asociado a información
- Permitir listar, crear, actualizar y borrar información
- Para las operaciones anteriores se necesitan una URL y un método HTTP para accederlas
- El servidor usualmente regresa la información en formato JSON
- Se retornan códigos de respuesta HTTP, por ejemplo 200, 400, 500 etc.

Una vez definido lo que es REST y sus principios, se pueden utilizar dos tipos de arquitecturas en base a estos. La arquitectura monolítica y la arquitectura con microservicios.

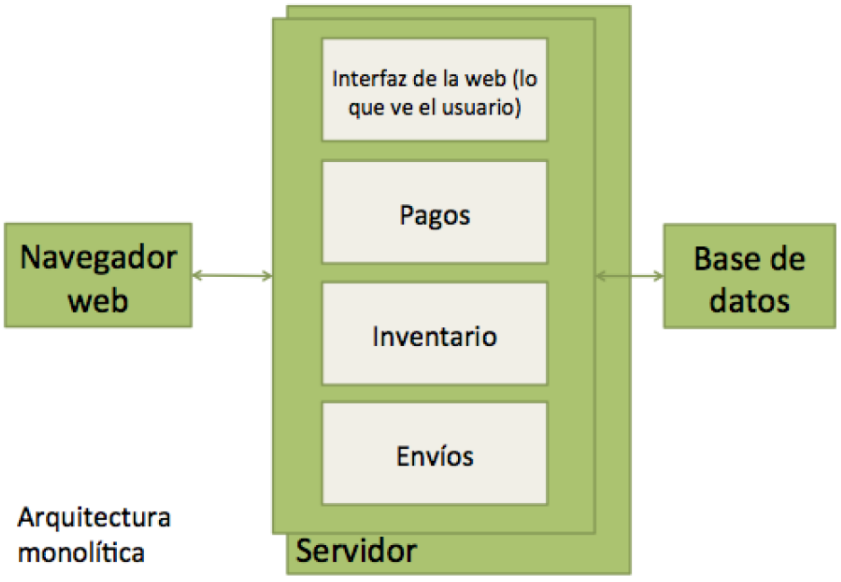


Ilustración 11: Diagrama de arquitectura monolítica

En la arquitectura monolítica el servidor es un único proyecto software en el cual se encuentra toda la lógica para todos los recursos, facilitando así la comunicación entre las diferentes partes del proyecto Entre una de las contras se encuentra la dificultad de

escalado horizontal o la falta de optimización en el mismo. En caso de necesitar un escalado horizontal se replicará el monolito en su totalidad, aunque realmente solo uno de los módulos sea el que necesite más recursos [39].

La otra aproximación es la de los microservicios. Una arquitectura de microservicios es un enfoque para desarrollar una aplicación software como una serie de pequeños servicios cada uno ejecutándose de forma autónoma y comunicándose entre sí; por ejemplo, a través de peticiones HTTP a sus API [40]. Una de las ventajas de los microservicios es su alta capacidad para escalar horizontalmente, pues al tenerlo todo dividido en pequeños módulos, únicamente necesitaremos replicar el que necesite más recursos y no todo el conjunto. Por otro lado, una de las contras es que se necesita una mayor cantidad de recursos de infraestructura para aprovechar este alto desacoplamiento. Si todos los microservicios funcionasen en una misma máquina sería notablemente más costoso que un monolito, esta arquitectura está pensada para distribuir entre distintas máquinas los pequeños servicios [41].

3.3.2 Análisis de tecnologías servidor

Partiendo de la base de que el servidor ASys debe construirse utilizando el lenguaje de programación Java debido a que todo el proyecto en su primera versión está construido en ese lenguaje, así como sus componentes y sistemas asociados, y por tanto es más fácil para el desarrollador usar el mismo lenguaje en todos los proyectos utilizados.

Después de haber realizado un estudio sobre los diferentes *frameworks* de este lenguaje en los que apoyar el desarrollo para agilizarlo, se han clasificado en los siguientes grupos:

Frameworks ligeros y *frameworks* completos.

Los *frameworks* ligeros son pequeñas librerías con las herramientas mínimas y necesarias para desarrollar una característica concreta del servidor. Y los *frameworks* completos ofertan un gran abanico de librerías para facilitar el desarrollo de todas las necesidades que pueda tener el proyecto. Estas necesidades o características pueden ser desde el desarrollo de controladores, hasta la seguridad, pasando por el acceso a datos, etc.

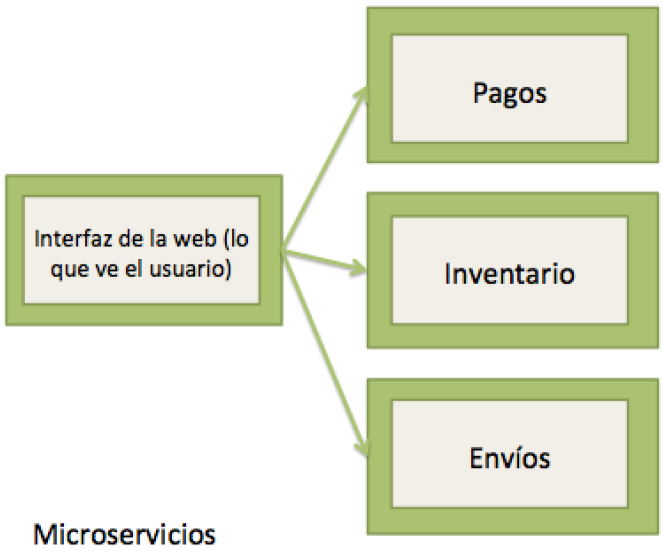


Ilustración 12: Diagrama de arquitectura de microservicios

Como *frameworks* ligeros se han tenido en cuenta Spark Framework y Restlet [42, 43].

Para *frameworks* completos se han tenido en cuenta Struts + JSF y Spring Framework [44, 45]. Pero entre estos se ha observado que Struts y JSF tienen un enfoque más tradicional y más enfocado a aplicaciones con renderizado en el servidor, que es de donde veníamos y lo que se quiere evitar.

En la primera versión de ASys se hace un uso intensivo de recursos en formato zip puesto que se usan para almacenar ejercicios, plantillas de corrección y las soluciones de los alumnos.

En la anterior versión de ASys estos recursos se guardaban haciendo uso del sistema de archivos del sistema operativo, guardándolos todos en una carpeta concreta. Para esta nueva versión se realizó una pequeña búsqueda para averiguar si se podían gestionar mejor este tipo de recursos y se descubrió una herramienta denominada MinIO [46, 47].

MinIO es un servidor que permite almacenar y gestionar nuestros recursos de forma descentraliza, es independiente del sistema operativo y está preparado para trabajar con él mediante comunicaciones HTTP, lo que nos permite situar este servidor en una o varias máquinas aparte e interactuar con él, haciendo mucho más eficiente y escalable la gestión de recursos.

3.3.3 Análisis de tecnología aplicación web

El principal enfoque para el desarrollo de la aplicación web, como se ha mencionado con anterioridad, es el de crear una app desacoplada del servidor y que funcione por si misma sin necesidad de que el servidor renderice las páginas. Este tipo de aplicaciones se denominan aplicaciones de una sola página o SPA (*Single Page Application*).

El mundo del desarrollo *frontend* tiene una cantidad inmensa de librerías y *frameworks* para realizar casi cualquier cosa, lo que ha dificultado mucho el estudio debido al gran abanico donde elegir. Por ello, se ha acotado la búsqueda a los *frameworks* más populares o más empleados por grandes compañías.

Angular [48]

Es un *framework* desarrollado por Google muy completo y que contiene librerías para hacer prácticamente cualquier cosa relacionada con SPA. La principal ventaja de esta librería es la de contar con una empresa tan grande detrás destinando muchos recursos para mejorarla constantemente, así como la gran comunidad de desarrolladores que hacen uso de esta tecnología. Como contras hay que mencionar que el avance de la herramienta es demasiado rápido y se genera una fragmentación constante entre las diferentes versiones. Esto sumado a la "obligación" de emplear Typescript como lenguaje para desarrollo, provoca que la curva de aprendizaje sea mucho más dura que en sus competidores y haya que estar adaptando el código a las últimas versiones más regularmente.

React.js [49]

Es una librería desarrollada por Facebook, la cual únicamente sirve para el renderizado de la aplicación mediante componentes. No es un *framework* en sí mismo, sino una librería de renderizado que requiere el uso de otras librerías a parte para poder desarrollar una aplicación de una sola página completa. Además, se puede añadir como posible contra la licencia añadida por Facebook a la misma la cual ha generado un gran debate entre la comunidad de desarrolladores web.

Polymer [50]

Polymer es otra librería desarrollada por Google, la cual consta principalmente de un abanico de componentes ya desarrollados, así como una serie de herramientas para desarrollar nuevos. Esto implica que el desarrollo se asemeje más a un juego de lego en el cual únicamente haya que ir juntando las piezas para construir una aplicación. Esta librería funciona muy bien para pequeños desarrollos, pero se echa en falta más personalización en aplicaciones de más envergadura puesto que su objetivo es ir uniendo bloques.

Vue.js [51]

Vue.js es la librería más novedosa de las mencionadas anteriormente. Está desarrollada por un exempleado de Google, el cual trabajó en el desarrollo de Angular y publicó su desarrollo como un proyecto de código libre, por lo que no hay ningún tipo de problema con las licencias y además al estar abierto a todo el mundo hay una gran cantidad de desarrolladores contribuyendo a su mejora. Como ventajas de Vue.js hay que destacar su fuerte enfoque en hacer la curva de aprendizaje muy ligera. Sus creadores se han centrado en realizar una documentación muy completa y muy bien explicada, además de según ellos ser un *framework* progresivo, es decir, se ha dividido en diferentes librerías que van aumentando en funcionalidad y complejidad, y que se pueden ir añadiendo al proyecto conforme se desee. Como contra de Vue.js hay que destacar su reducido número de recursos de apoyo debido a su corto periodo de vida debido a que es una tecnología muy novedosa, aunque ahora mismo es el *framework* de moda y está creciendo en popularidad muy rápidamente.

3.4 Solución propuesta

En este apartado se van a mencionar las tecnologías escogidas entre las estudiadas en el apartado anterior, así como realizar una pequeña descripción argumentando el motivo.

Autentificación

En ASys se desea identificar a un usuario y en base a qué usuario sea, asignarle un rol o tipo de usuario y por tanto autorizarlo. También es necesario identificar el usuario para realizar distintas operaciones asociadas como puede ser saber quién ha realizado un ejercicio o una corrección. Por este motivo se ha escogido la autentificación de OpenID frente a la autorización por oAuth. Además, el hecho de que OpenID haga uso del estándar Json Web Tokens (JWT) es un punto extra, pues nos permite encriptar la información del usuario en el token para así con la clave de desencriptación (que únicamente poseerá el servidor) acceder a la información directamente sin consultar la base de datos, esto dará lugar a una mayor eficiencia por parte del servidor.

Arquitectura del servidor

ASys es un sistema que en su inicio únicamente va a disponer de una máquina para ser desplegada, por lo que una arquitectura de microservicios no estaría bien aprovechada y seria mucho peor que una en monolito en cuanto a rendimiento se refiere. Además, las arquitecturas de microservicios requieren un fuerte esfuerzo en el diseño de la persistencia para poder desacoplarse en diferentes módulos, lo que provocaría romper en gran parte con el diseño inicial de ASys en su anterior versión. Por este motivo, se ha decidido desarrollar el servidor con una arquitectura monolítica siguiendo los principios REST para crear una API totalmente desacoplada.

Tecnologías servidor

Uno de los objetivos del proyecto es el de mejorar en gran medida la capacidad para ser mantenido con respecto a su versión anterior, y por tanto mejorar el tiempo de desarrollo. Por esto se ha decidido hacer uso de un *framework* más completo que obligue a desarrollar las diferentes capas del proyecto de forma muy similar, por lo que deberá de poseer herramientas para desarrollar cada una de las diferentes capas.

Por este motivo, junto con la gran cantidad de recursos debido a que es el *framework* más usado en Java, sumado a la capacidad para adaptarse al enfoque REST deseado, se ha escogido Spring *framework* como mejor tecnología para el desarrollo del servidor [45].

Para la gestión de recursos se añadirá MinIO; y como motor de base de datos se mantendrá MySql, puesto que es el mismo motor utilizado en la versión anterior de ASys.

Tecnologías aplicación web

Finalmente, la tecnología escogida para el desarrollo de la aplicación web de una sola página ha sido Vue.js. A pesar de que es una apuesta por su reciente creación, Vue.js se está convirtiendo en una tendencia para los nuevos desarrollos, debido a su orientación *open source* la cual garantiza un soporte infinito por parte de la comunidad. Además, su rendimiento es tremendamente bueno en comparación con sus competidores y el gran esfuerzo invertido en la documentación es un factor que se ha tenido muy en cuenta.

4 Diseño de la solución

En este apartado se va a explicar en base a los requisitos y el análisis detallado anteriormente, cual es la arquitectura del sistema, es decir, como se ha dividido el sistema en proyectos, módulos y bloques para poder realizar una buena implementación que cumpla con los objetivos de mantenibilidad y eficiencia deseados.

4.1 Arquitectura del sistema

Dadas las necesidades de ASys, en las cuales se va a hacer un alto uso de recursos tanto de datos como de binarios. Se quiere hacer un diseño muy modular y escalable en el cual cada subsistema pueda funcionar independientemente en una maquina dedicada para facilitar su disponibilidad y su rendimiento.

En base a las tecnologías propuestas obtenidas del análisis, la representación del sistema después de la fase de implementación debe ser la siguiente.

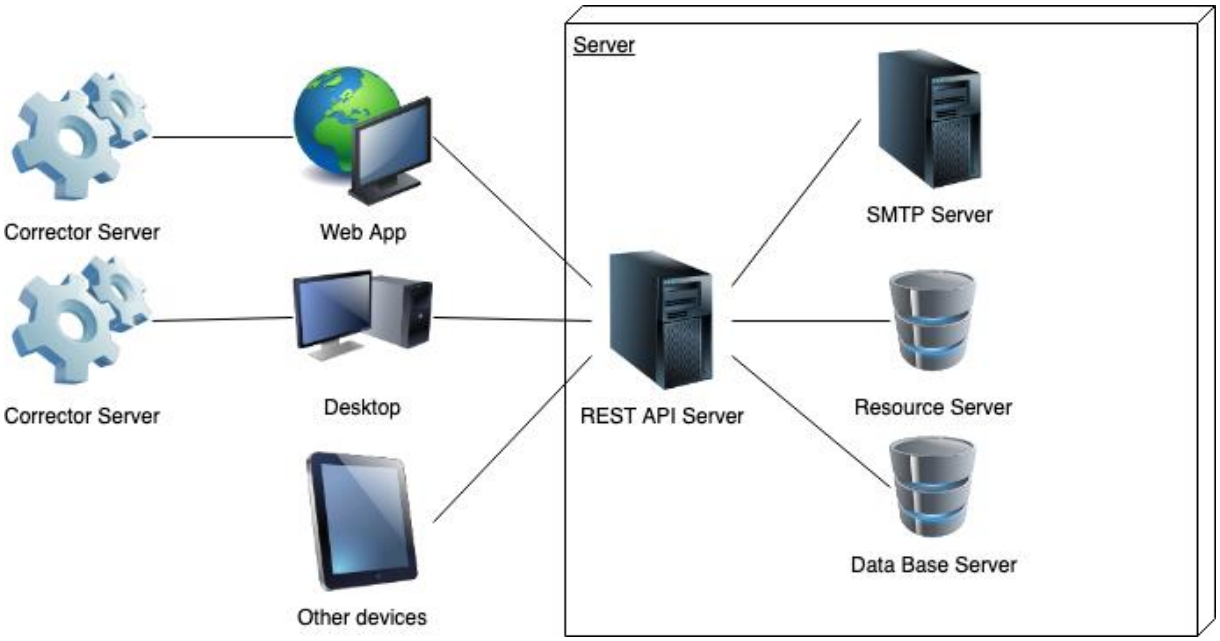


Ilustración 13: Diagrama de la arquitectura del sistema ASys v2

Como podemos observar el sistema se compondrá de diferentes aplicaciones relacionadas con la interacción de los usuarios (Web App, Desktop, Other Devices), las cuales para su funcionamiento por un lado requerirán comunicarse con un servidor REST que se encontrara en una máquina remota, y por otro lado se comunicara localmente con un servidor que se encargara de realizar las correcciones de los ejercicios. Como excepción se encuentran otros posibles dispositivos a los que se les pueda dar soporte en el futuro, pero cuya funcionalidad estará limitada de acuerdo con sus características. Como pueden ser los dispositivos móviles o tabletas en los cuales no se puede arrancar un servidor para realizar correcciones.

El sistema requiere de cuatro servidores, aunque en un principio todos van a funcionar en una misma máquina, el estar desacoplados unos de otros, nos permite poder dividirlos en un futuro en máquinas diferentes para poder mejorar la escalabilidad del sistema y poder dar servicio a una mayor cantidad de usuarios. Si bien es cierto que

hacer un diseño dividido en diferentes servidores mejora notablemente la escalabilidad por otro lado repercute en gran medida en el tiempo de desarrollo, ya que no es lo mismo desarrollar un único servidor que sea un único proyecto y todo este en el mismo lugar, que lidiar con cuatro servidores diferentes para los cuales hay que tener en cuenta las comunicaciones y adaptar el código a las necesidades de cada uno.

Para el alcance de este trabajo nos vamos a centrar en los proyectos Web App, REST API Server y los servidores asociados al mismo. Ya que debido a las grandes dimensiones del sistema es inabarcable desarrollarlo todo en único TFG individual.

4.2 Diseño Detallado

En este apartado se va a especificar más en detalle el diseño individual de los proyectos Web App y REST API Server con sus respectivos servidores colindantes. Cada uno de estos dos proyectos puede alcanzar unas dimensiones importantes y puede requerir un notable esfuerzo por parte de los desarrolladores, por lo que es necesario realizar un diseño de la arquitectura interna de cada uno de estos proyectos. Por otra parte, los servidores colindantes del servidor REST, son servidores de uso comercial ya desarrollados por corporaciones pero que requieren de una configuración para adaptarlos a las necesidades del sistema.

4.2.1 Arquitectura REST API Server (ASys Web Server)

Para este proyecto se ha decidido seguir una arquitectura vertical típica en capas muy propia de los servidores web, en la cual la descomposición es en base a su funcionalidad. La estructura vertical está dividida en Controladores, Servicios y Repositorios y luego existe una capa más transversal u horizontal que es la de utilería.

En una arquitectura por capas la comunicación entre las mismas se realiza entre las capas inmediatamente inferior o inmediatamente superior. Una capa solo puede realizar llamadas a función de la capa inmediatamente inferior, recibir el resultado de la misma y devolver la información a la capa inmediatamente superior. Como excepción a esta jerarquía de capas se encuentra la capa de clases de utilería. En esta capa se encuentran métodos que son completos por sí mismos y pueden ser llamados desde cualquier punto del proyecto, aunque en la mayoría de los casos suelen ser utilizados por la capa de servicio.

Cada una de las capas tendrá una división horizontal en base a elementos del dominio o a funcionalidades muy específicas. Como ejemplo se puede tener la división de la figura 14 en la cual las diferentes capas verticales están divididas en capas horizontales como son las de autenticación, usuarios, ejercicios, etc.

La capa de control es la parte más externa del servidor, es donde se definen los puntos de entrada de las peticiones que detonan una acción y donde se devuelve la información a las mismas. El objetivo de la capa de control es recibir información, realizar una llamada a la capa inferior y devolver la información de la misma al exterior. Por tanto, los controladores podrán contener lógica relacionada con definir los puntos de entrada, capturar los parámetros o posible información enviada, comprobar los permisos y realizar una llamada a la capa inmediatamente inferior. En ningún caso se debe de realizar lógica de comprobación de ningún otro tipo.

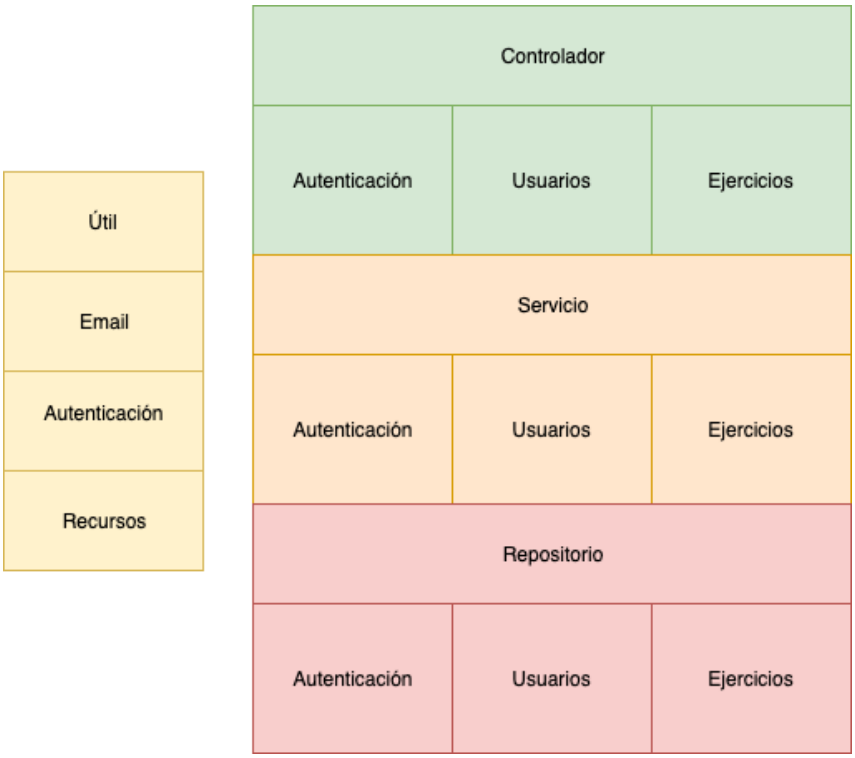


Ilustración 14: Diagrama de la arquitectura del servidor ASys v2

La capa de servicio es la capa donde se desarrolla toda la lógica de la aplicación, aquí es donde se escribe la mayoría del código y es donde se obtienen datos o recursos, se manipulan, se realizan validaciones, se llama a otros servicios, etc. En esta capa se recibirá como entrada la información obtenida por los controladores, y por otro lado se devolverá la información obtenida del proceso. El cómo completar el proceso depende mucho de la funcionalidad en sí misma, pero se pueden realizar llamadas a la capa inferior, a otros servicios o a métodos de utilería.

Como dato mencionar que la comunicación con sistemas externos suele estar contenida en los servicios que a su vez utilizan métodos de utilería o librerías para poder realizar esta comunicación.

En la capa repositorio es donde se desarrolla toda la lógica de acceso a datos. Es donde se realizan las comunicaciones con la base de datos y se definen las operaciones necesarias para cada uno de los elementos del dominio de los cuales se necesite guardar información persistente.

4.2.2 Configuración del servidor de recursos (Resource Server)

El servidor de recursos es un servidor muy sencillo en el cual únicamente se van a almacenar ciertos tipos de binarios. Este servidor no hay que desarrollarlo puesto que se va a utilizar uno ya desarrollado denominado MinIO, como se ha explicado en el apartado de análisis.

Se ha realizado un estudio de los posibles ficheros binarios necesarios para persistir y se ha decidido organizar el servidor de recursos en las siguientes secciones:

- Enunciados
- Recursos
- Soluciones
- Plantillas

- Utilería

En la sección de enunciados se almacenarán todos los pdfs utilizados para redactar los enunciados de los ejercicios creados por el profesorado.

En la sección de recursos se almacenarán archivos zip, los cuales constan de proyectos vacíos con la estructura base para poder realizar un ejercicio. Es lo que los alumnos rellenaran para poder realizar una solución.

En la sección de soluciones se almacenarán todas las soluciones en formato zip de los alumnos.

En la sección de plantillas es donde se almacenarán en formato zip los recursos utilizados para realizar las correcciones de los ejercicios.

En utilería se almacenarán recursos varios que no tengan nada que ver con las secciones anteriores, como puede ser los programas servidores de corrección.

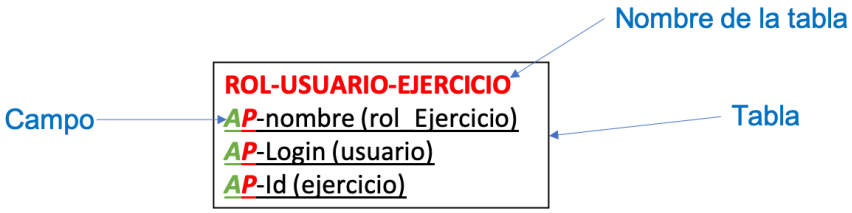
4.2.3 Diseño de la base de datos

Para ASys se ha fijado como objetivo el hacer un diseño lo más completo posible, pensando en todas las necesidades que pueda tener el sistema. El motivo de este objetivo es el de tener que realizar los mínimos cambios necesarios en un futuro, para ahorrar tiempo en tareas de mantenimiento y no dar pie a comportamientos inesperados. Este diseño ha sido realizado en su totalidad por mi tutor Josep Silva. Teniendo como primera aproximación el modelo relacional utilizado en la primera versión de ASys web.

El modelo relacional está fuertemente relacionado con el modelo de dominio mostrado en el apartado 3.1.1.4 del análisis.

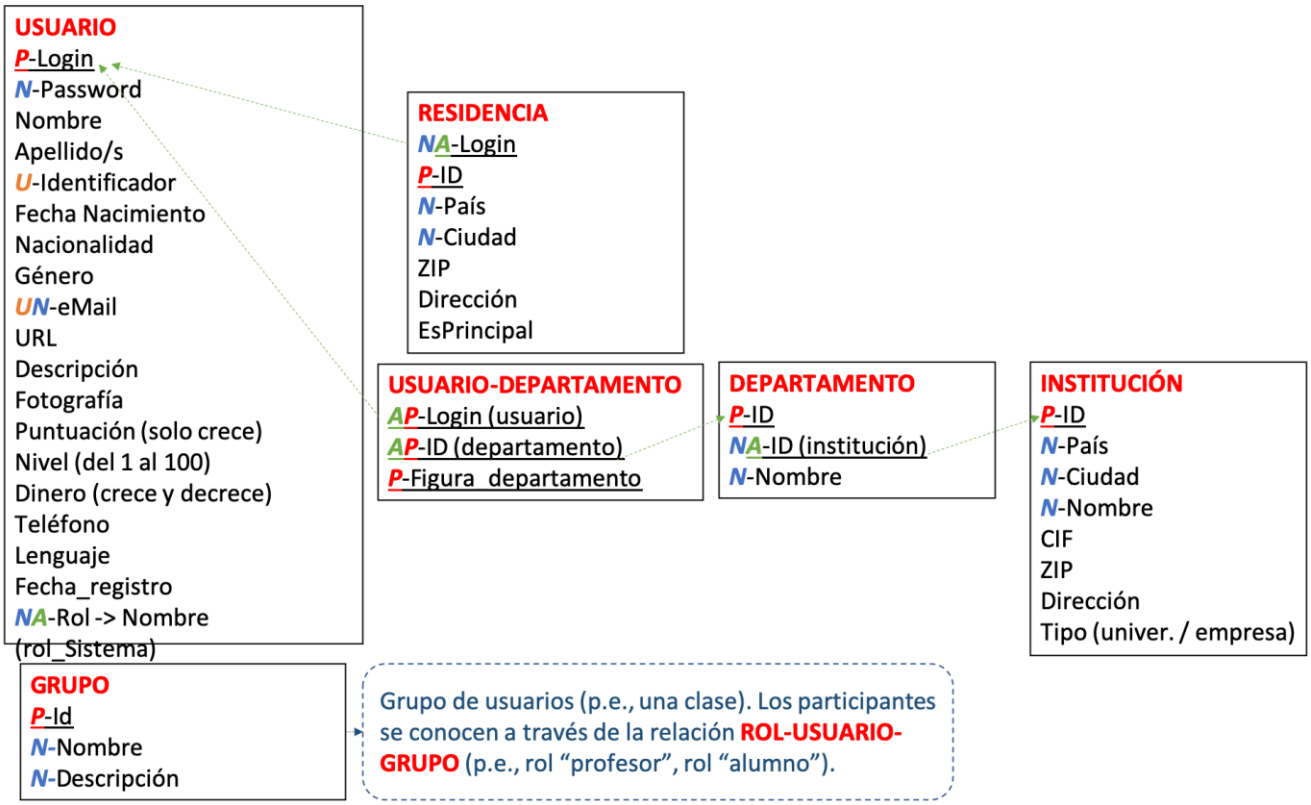
A continuación, se va a mostrar el modelo relacional diseñado para este proyecto.

Las tablas y relaciones se representan con tablas, donde cada fila de la tabla corresponde a un campo.

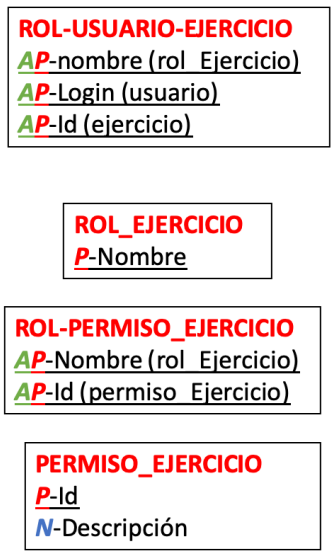


Como prefijo de cada campo se indican las restricciones del campo:

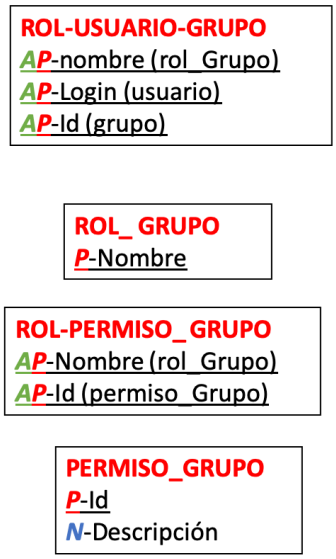
- Clave primaria: P (Atención: los campos de la clave primaria están subrayados)
- Clave ajena: A
- Unicidad: U
- Valor no-nulo: N



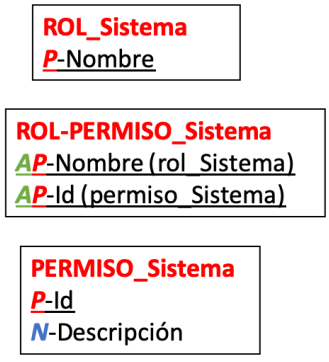
Rol de los usuarios en un ejercicio



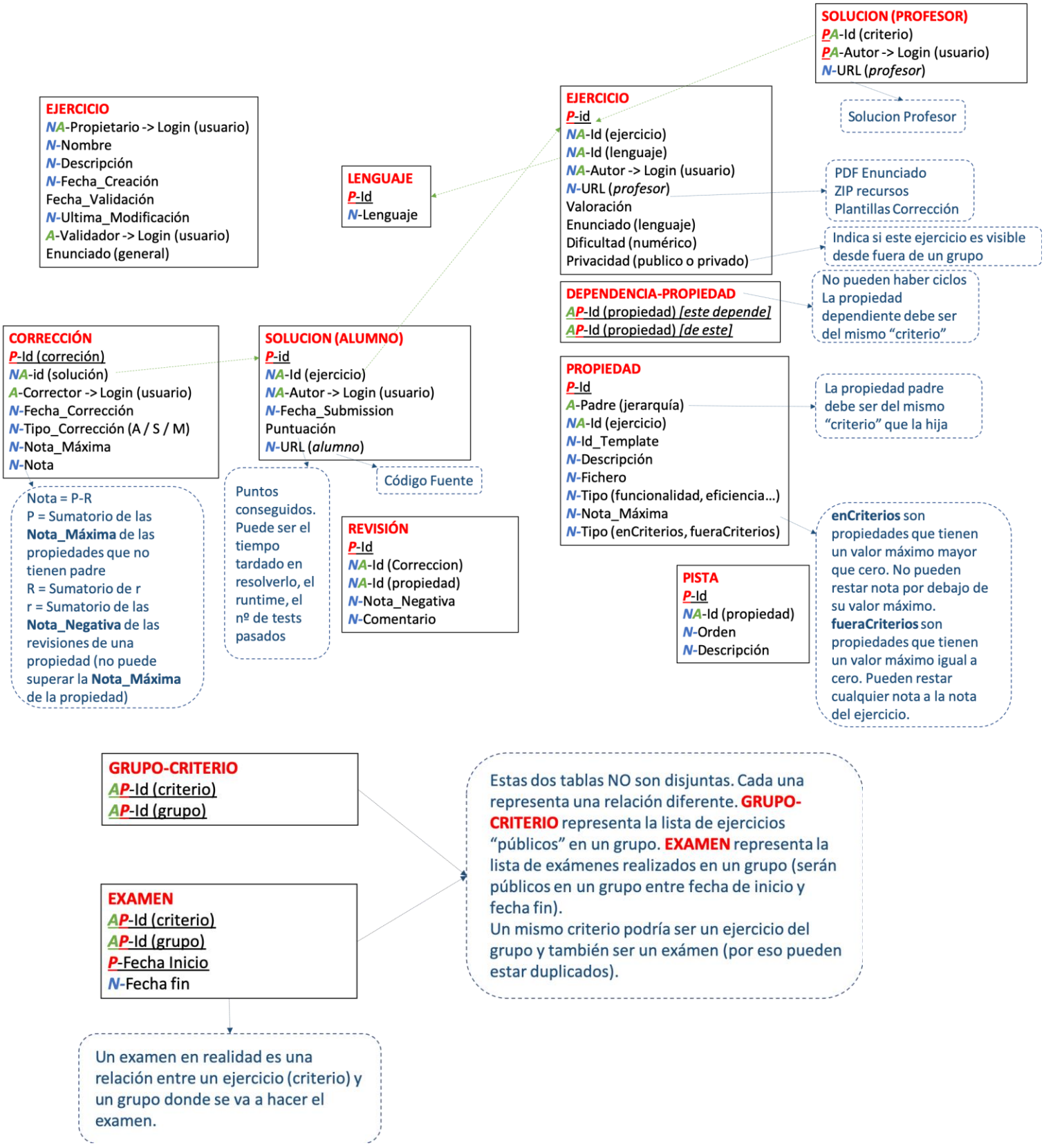
Rol de los usuarios en un grupo

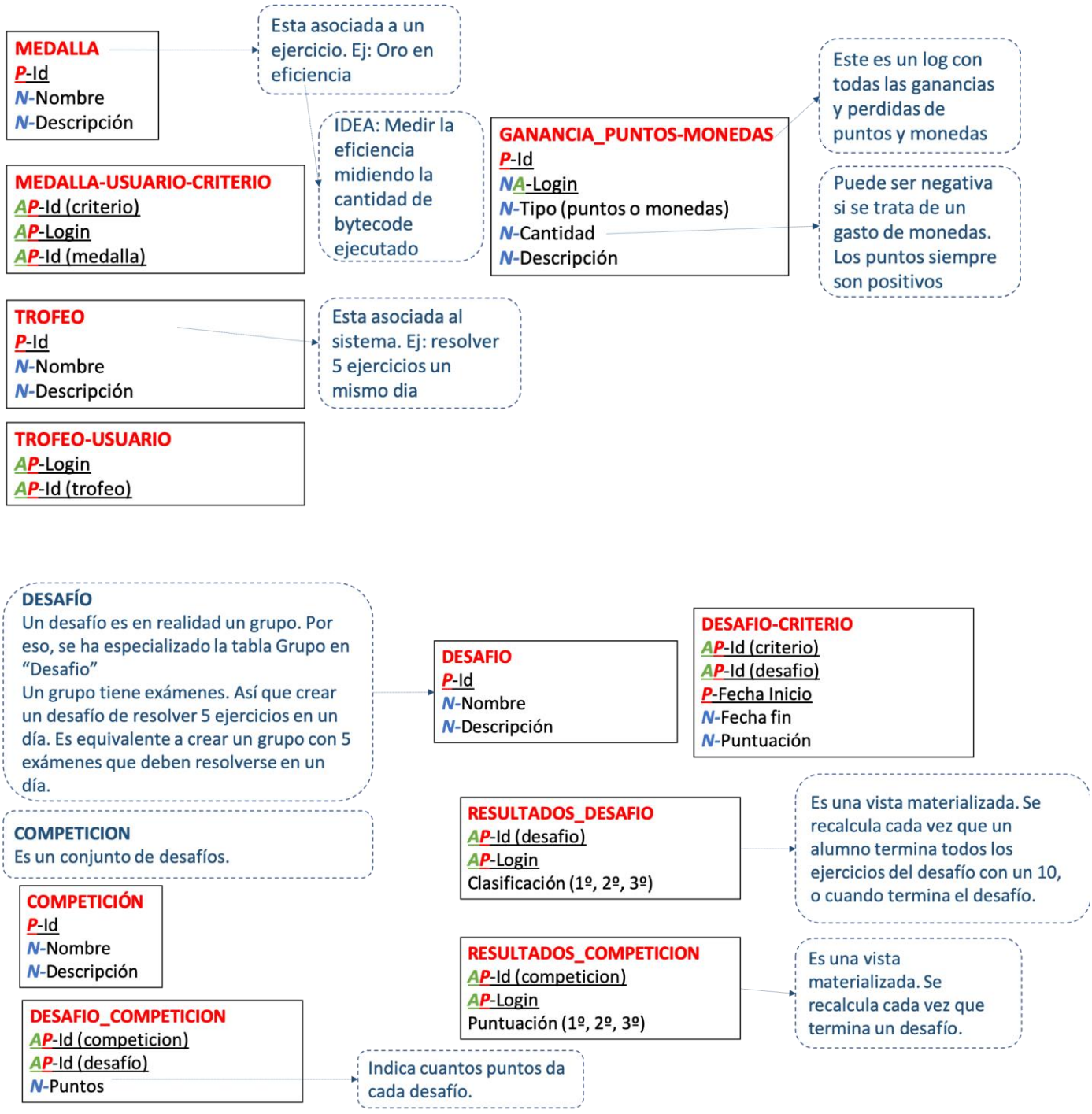


Rol de los usuarios en el sistema



Un rol es en realidad un conjunto de permisos. Los roles están especializados. (De momento) existen el rol de un usuario en un ejercicio, el rol de un usuario en un grupo, y el rol de un usuario en el sistema.


SOLUCION (PROFESOR)
PA-Id (criterio)
PA-Autor -> Login (usuario)
N-URL (profesor)
Solucion Profesor
EJERCICIO
NA-Propietario -> Login (usuario)
N-Nombre
N-Descripción
N-Fecha_Creación
Fecha_Validación
N-Ultima_Modificación
A-Validador -> Login (usuario)
Enunciado (general)
LENGUAJE
P-Id
N-Lenguaje
EJERCICIO
P-id
NA-Id (ejercicio)
NA-Id (lenguaje)
NA-Autor -> Login (usuario)
N-URL (profesor)
Valoración
Enunciado (lenguaje)
Dificultad (numérico)
Privacidad (publico o privado)
PDF Enunciado
ZIP recursos
Plantillas Corrección
Indica si este ejercicio es visible desde fuera de un grupo
DEPENDENCIA-PROPIEDAD
AP-Id (propiedad) [este depende]
AP-Id (propiedad) [de este]
No pueden haber ciclos
La propiedad dependiente debe ser del mismo "criterio"
CORRECCIÓN
P-Id (correción)
NA-id (solución)
A-Corrector -> Login (usuario)
N-Fecha_Corrección
N-Tipo_Corrección (A / S / M)
N-Nota_Máxima
N-Nota
SOLUCION (ALUMNO)
P-id
NA-Id (ejercicio)
NA-Autor -> Login (usuario)
N-Fecha_Submission
Puntuación
N-URL (alumno)
Código Fuente
PROPIEDAD
P-Id
A-Padre (jerarquía)
NA-Id (ejercicio)
N-Id_Template
N-Descripción
N-Fichero
N-Tipo (funcionalidad, eficiencia...)
N-Nota_Máxima
N-Tipo (enCriterios, fueraCriterios)
La propiedad padre debe ser del mismo "criterio" que la hija
Nota = P-R
P = Sumatorio de las Nota_Máxima de las propiedades que no tienen padre
R = Sumatorio de r
r = Sumatorio de las Nota_Negativa de las revisiones de una propiedad (no puede superar la Nota_Máxima de la propiedad)
Puntos conseguidos. Puede ser el tiempo tardado en resolverlo, el runtime, el nº de tests pasados
REVISIÓN
P-Id
NA-Id (Correccion)
NA-Id (propiedad)
N-Nota_Negativa
N-Comentario
PISTA
P-Id
NA-Id (propiedad)
N-Orden
N-Descripción
enCriterios son propiedades que tienen un valor máximo mayor que cero. No pueden restar nota por debajo de su valor máximo.
fueraCriterios son propiedades que tienen un valor máximo igual a cero. Pueden restar cualquier nota a la nota del ejercicio.
GRUPO-CRITERIO
AP-Id (criterio)
AP-Id (grupo)
EXAMEN
AP-Id (criterio)
AP-Id (grupo)
P-Fecha Inicio
N-Fecha fin
Estas dos tablas NO son disjuntas. Cada una representa una relación diferente. GRUPO-CRITERIO representa la lista de ejercicios "públicos" en un grupo. EXAMEN representa la lista de exámenes realizados en un grupo (serán públicos en un grupo entre fecha de inicio y fecha fin).
Un mismo criterio podría ser un ejercicio del grupo y también ser un exámen (por eso pueden estar duplicados).
Un examen en realidad es una relación entre un ejercicio (criterio) y un grupo donde se va a hacer el examen.



4.2.4 Configuración del servidor de correo SMTP

Con fines de agilizar el desarrollo se utilizará un servidor SMTP virtual denominado Mailtrap, en el cual se podrán observar los correos que vaya enviando la aplicación, pero en ningún caso se propagarán al exterior para evitar problemas con los proveedores de correo actuales [52].

Para la utilización del servidor virtual se creará una cuenta ficticia relacionada con el proyecto para que todos los desarrolladores puedan realizar pruebas.

Una vez se haya completado el desarrollo del sistema y se vaya a realizar una puesta en producción abierta para todos los usuarios, se procederá a montar un servidor de correo real.

4.2.5 Arquitectura Web App (ASys Web Client)

En este proyecto la arquitectura es más complicada y más informal debido a las características intrínsecas del proyecto y la tecnología en sí misma. Por lo que dará lugar a una arquitectura más irregular o de conceptos. Esta arquitectura se ira modificando a medida que avance el desarrollo en base a las necesidades detectadas.

La idea inicial del diseño de la aplicación es el de respetar las arquitecturas utilizadas por los *frameworks* que utilizan componentes web. Que básicamente consiste en dividir el proyecto en base a los recursos. Por tanto, la estructura constara de las siguientes secciones.

- Vistas
- Persistencia
- Router
- Almacén
- Componentes
- Recursos

Las vistas hacen referencia a las diferentes páginas web desarrolladas en la aplicación, en las cuales se incluirá todo el contenido visual y se capturará la interacción con el usuario.

La persistencia será la capa encargada de realizar comunicaciones externas a la aplicación para obtener información o recursos.

El *router* es el elemento encargado de asociar una visa a una URL y definir los permisos o las comprobaciones necesarias. Como el tamaño del *router* depende en gran medida de las vistas y suele haber grandes cantidades de estas. Se ha creado una sección para el *router* por si es necesario hacerlo modular en un futuro.

El almacén es una sección muy novedosa, utilizado en las aplicaciones reactivas para gestionar la información que se va almacenando en una caché global a toda la aplicación. En esta sección se gestiona toda la información de la aplicación y por tanto seguramente haya que descomponerla en diferentes archivos según las características del dominio (Ejercicios, usuarios, correcciones, autentificación etc..).

Los componentes son la base de todas las aplicaciones web modernas. Los componentes se encargan de encapsular funcionalidades completas de la aplicación, es decir, que contengan tanto la vista, la interacción y la lógica. Las web apps se descomponen todas en pequeños componentes que después se utilizan para montar vistas como si fueran piezas de Lego reutilizables. Se espera que durante el desarrollo se cree una gran cantidad de componentes por lo tanto se necesita organizarlos en una sección dedicada entera para ellos mismos.

Recursos en todas las aplicaciones visuales se hace uso de una gran cantidad de recursos como pueden ser imágenes, iconos, audios etc... Por lo tanto, también se ha decidido crear una sección para organizarlos.

5 Desarrollo de la solución

En esta sección se van a explicar los detalles de implementación de los proyectos ASys Web Client y ASys Web Server. Para facilitar el trabajo a un posible desarrollador en un futuro. No se pretende hacer una explicación extensa de todo el proyecto sino de factores a tener en cuenta en ciertas partes.

5.1 Desarrollo del servidor ASys Web Server

Para la aplicación servidor se va a proceder a explicar cómo hacer uso de las librerías de Spring, para poder ampliar cada una de las capas detalladas en el apartado de diseño de la arquitectura en el punto 4.2.1. Estas librerías se han ido descubriendo en el periodo previo de formación, y algunas durante el desarrollo ante el surgimiento de problemas y necesidades, pues inicialmente se quería desarrollar el proyecto en Spring pero no se tenían conocimientos de la existencia de estas librerías debido al extenso catálogo que posee el *framework* [45]. Como referencia se va a mostrar el desglose de paquetes contenidos en el proyecto.

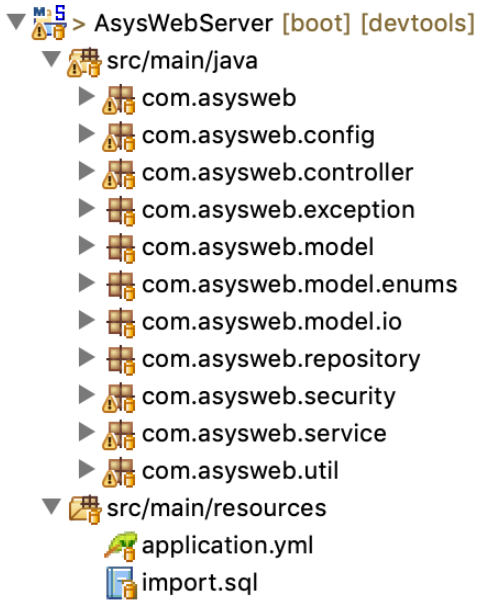


Ilustración 15: Estructura de paquetes proyecto servidor

5.1.1 Modelos

El desarrollo comienza creando las clases que representan las tablas del modelo relacional situados en el paquete com.asysweb.model. Como Spring por defecto utiliza ingeniería directa, es necesario crear los modelos de la base de datos para que él automáticamente al ejecutar la aplicación cree las tablas.

Para crear las tablas es necesario crear un objeto java nuevo con el nombre de la tabla (se puede poner cualquier nombre y después usar una anotación java para especificar el nombre de la tabla, pero por defecto Spring utiliza el nombre de la clase), y situar dentro del código, encima de la definición de la clase, la anotación @Entity. La cual indicara a spring que esta clase es una entidad que deberá mapearse en una base de datos.

A continuación, añadiremos las propiedades deseadas a la clase que se mapearan como columnas de la tabla de la base de datos, y necesitaremos añadir los *getters* y

setters correspondientes para poder manipularlos. Spring provee de una serie de anotaciones especiales para hacer la creación de la tabla más detallada. Algunas de estas anotaciones son @Id, @GeneratetedValue, @NotNull, @Unic y sobre todo las que definen las claves ajenas entre tables como son @OneToOne, @OneToMany y @ManyToOne. La única anotación de las anteriores que es obligatoria es @Id, pues indicará la columna que será la clave primaria de la tabla [53].

Una vez creados los modelos necesarios, al ejecutar la aplicación se crearán las tablas. A estas tablas podremos insertar datos de prueba, añadiendo consultas *insert into* en el archivo import.sql situado en la carpeta *resources*. Este archivo se ejecutará automáticamente cada vez que se cree la base de datos al ejecutar la aplicación.

Un ejemplo orientativo de un modelo usando la librería Spring core sería el siguiente:

```
1. @Entity
2. public class User {
3.
4.          @Id
5.          @GeneratedValue
6.          private Long id;
```

5.1.2 Repositorios

Tras haber creado los modelos, lo natural es crear el repositorio. El repositorio es la capa de acceso a datos, encargada de manipular y obtener la información de la base de datos. En Spring crear repositorios es muy sencillo y ahorra enormemente el tiempo necesario para desarrollar esta capa. Únicamente hay que crear una interfaz que extienda de JpaRepository<Param1, Param2>, donde param1 es la entidad para la cual queramos crear un repositorio y param2 es el tipo del identificador de la entidad. Con esto Spring ya nos proveerá de métodos genéricos de acceso básico, como puede ser obtener un elemento o una lista de elementos, añadir, actualizar y borrar un elemento.

Si queremos añadir métodos más específicos o realizar consultas más complejas, bastara con crear dentro de la interfaz el perfil del método utilizando una serie de patrones para el nombre. Por ejemplo si queremos obtener un usuario por su nombre, bastaría con añadir este perfil a la interfaz `User findByUsername(String username)` y Spring automáticamente generará internamente la implementación del método, por lo que podremos usarlo sin crearla nosotros mismos [54, 55]. La herramienta de patrón de nombre facilita muchísimo el trabajo pero si queremos realizar una consulta más compleja habrá que añadir la anotación @Query("mi consulta") justo encima del método. Esta consulta puede ser parametrizada añadiendo dentro de la mismas dos puntos y el nombre del parámetro que hayamos definido en el perfil del método.

Un ejemplo orientativo de un repositorio creado con la librería Spring JPA Data Repositories sería el siguiente:

```
1. public interface UserRepository extends JpaRepository<User,
   Long> {
2.          User findByUsername(String username);
```

5.1.3 Controladores

Los controladores son la parte fundamental del servidor, pues es el punto de entrada de las peticiones, es el lugar donde empieza una acción. Para crear un controlador en Spring es muy sencillo, basta con crear una clase con las anotaciones

@RestController y @RequestMapping("/sección") encima de su definición, donde sección será la porción de URL añadida a la URL del servidor base que será donde se añadirán las escuchas para las peticiones. Una vez definido un controlador, en su interior podremos definir los endpoints o puntos de entrada para las peticiones (los métodos que desencadenan las acciones). Para ello habrá que definir un método con las anotaciones @GetMapping("/subSección"), @PostMapping("/subSección"), @PutMapping("/subSección") y @DeleteMapping(/subsección). Estas anotaciones definen el método http y la sección que se añadirá a la URL del controlador necesarios para que una petición inicie el método al que están relacionados. Ademas con las anotaciones @RequestParam, @PathVariable y @RequestBody podremos pasar parámetros de las peticiones http a los métodos de entrada [45].

Un ejemplo orientativo de un controlador y un *endpoint* haciendo uso de la librería Spring Web MVC sería el siguiente:

```
1. @RestController
2. @RequestMapping("/user")
3. public class UserRestController {
4.         @GetMapping("/{id}")
5.         public User getProfileById(@PathVariable long id) {
6.                 return userService.getUserbyId(id);
7.         }
```

5.1.4 Servicios

Para terminar con la estructura de capas vertical del servidor, explicare brevemente como es la capa de servicio, puesto que es la que menos dependencia de Spring tiene. La capa de servicio es la que se encarga de realizar toda la lógica operacional de la aplicación. En esta capa únicamente deberemos definir clases java con la anotación @Service("nombre de instancia") encima de la definición de clase. Con esta anotación podremos invocar a las clases sin necesidad de crear instancias de estas mediante el uso de la anotación @Autowire. Los métodos contenidos en estas clases son métodos java normales y corrientes no tienen nada en especial

Un ejemplo orientativo de creación e invocación de una clase de servicio utilizando la librería Spring core sería el siguiente:

```
1. @Autowired
2. private ExerciseServiceImpl oExerciseService;
3.
4. @Service("oExerciseService")
5. public class ExerciseServiceImpl {
```

5.1.5 Seguridad

En Spring existe una librería llamada Spring Security que nos provee de muchas herramientas para proteger nuestra aplicación de diferentes vulnerabilidades, además de poder definir un sistema de permisos para los posibles puntos de entrada de nuestra aplicación. Algunos de las protecciones que nos ofrece Spring security pueden ser protección contra ataques como la fijación de sesiones, el *clickjacking*, la falsificación de solicitudes entre sitios, etc... [56]. Spring security es probablemente el apartado más complejo del servidor, por ello no se va a entrar muy en detalles con respecto a su implementación, únicamente se van a mencionar dos aspectos. En el paquete com.asysweb.config existe una clase llamada WebSecurityConfig, en esta clase se define mediante patrones los permisos de acceso a las URLs y se habilita CORS (*Cross Origin Resource Sharing*), para controlar las aplicaciones web que pueden solicitar peticiones al servidor [57].

Para el control de permisos mediante patrones de las URLs se sobrescribe el método configure y en su interior se definen los mismos. Un ejemplo orientativo de este método sería el siguiente:

```
1. @Override
2. protected void configure(HttpSecurity
   httpSecurity) throws Exception {
3.         httpSecurity
4.                 .antMatchers(
5.                         HttpMethod.GET,
6.                         "/",
7.                         "/*.html",
8.                         "/favicon.ico",
9.                         "/**/*.html",
10.                              "/**/*.css",
11.                              "/**/*.js"
12.                     ).permitAll()
13.                     .antMatchers("/auth/**").permitAll()
14.                     .anyRequest().authenticated();
15.
```

Para activar cors y añadir excepciones al registro habría que crear el método corsConfigurer. Un ejemplo orientativo de como habilitar cors para una aplicación web situada en la URL `localhost:8081` seria la siguiente.

```
1.     @Bean
2.     public WebMvcConfigurer corsConfigurer() {
3.         return new WebMvcConfigurerAdapter() {
4.             @Override
5.             public void addCorsMappings(CorsRegistry registry) {
6.                 registry.addMapping("/**").allowedOrigins("http://l
   ocalhost:8081");
7.             }
8.         };
9.     }
```

5.1.6 Acceso al servidor de recurso

Para el acceso a los recursos del servidor MinIO se descargó una librería llamada MinioClient [47]. Esta librería es más sencilla de usar que la oficial, haciendo el acceso mucho más intuitivo y menos engorroso. Para el acceso a recursos se ha creado un servicio con métodos para cada una de las operaciones necesarias de un recurso, como pueden ser obtener, borrar y añadir. MinioClient se inicializa con los datos de acceso del servidor de la siguiente manera:

```
1. public AmazonS3ServiceImpl() throws InvalidEndpointException,
   InvalidPortException {
2.      minioClient = new MinioClient("http://localhost:9000/",
3. "apikey", "secretKey");
4. }
```

Además, un ejemplo orientativo de un método de acceso a recursos ene l servicio sería el siguiente.

```
1. public InputStream downloadFile(String bucket, String file) throw
   s Exception {
2.         return minioClient.getObject(bucket, file);
3. }
```

En MinIO los recursos únicamente se pueden organizar en carpetas de un solo nivel de anidamiento llamadas *buckets* [46]. Por lo que para acceder a un recurso hay que indicar el *bucket* donde se encuentra y el nombre completo del recurso.

5.2 Desarrollo de la aplicación web ASys Web Client

Para el desarrollo de la aplicación web se ha utilizado un gran número de librerías y recursos externos para facilitar el desarrollo [58, 59, 60, 61, 62, 63, 64, 65]. Por ello no se van a explicar todos, sino los de más relevancia acorde a las secciones en las que se ha estructurado el proyecto, que son las que tienen que ver con el *framework* Vue.js.

La estructura utilizada en el proyecto es la siguiente:

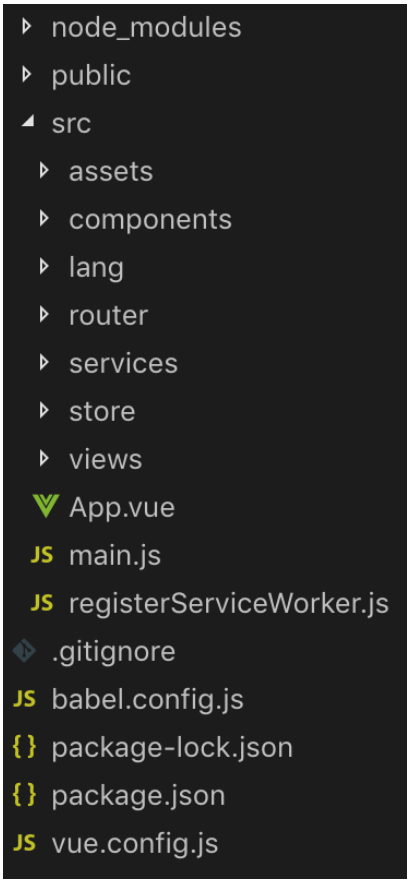


Ilustración 16: Estructura de carpetas de la aplicación web

5.2.1 Componentes

En las aplicaciones web modernas es muy utilizado el desarrollo orientado a componentes. Los componentes han supuesto una revolución en el mundo del desarrollo web en términos de reutilización y encapsulación de código cliente, puesto que permite encapsular nuestro código HTML, CSS y Javascript de forma que no se vea afectado por el código de la página que lo incluye (en caso de no querer forzarlo explícitamente), haciéndolos totalmente autosuficientes para un propósito concreto [66].

Cada *framework* tiene su propia manera de definir los componentes y realizar su propia interpretación de estos. En Vue.js se pueden definir de varias formas, una de ellas es la llamada 'componentes de un solo archivo' o en ingles *Single File Components*. Estos componentes nos permiten, como su nombre indica, encapsular el código CSS, HTML y Javascript en un único archivo [51]. Esto es un modo muy útil de definir los componentes

puesto que cuando la aplicación escale en tamaño nos será más fácil encontrar el código deseado y reduciremos en gran medida el tamaño en archivos del proyecto.

Un ejemplo sencillo de como sería un componente de un solo archivo en Vue.js sería el siguiente:

```
1. <style lang="scss" scoped>
2.     .contenerdor{ color: red;}
3. </style>
4.
5. <template>
6.     <div class="contenedor">
7.         Hola
8.     </div>
9. </template>
10.
11.      <script>
12.      export default {
13.          name: "Ejemplo",
14.          data() {
15.              return{
16.                  localVar: 'Hola'
17.              }
18.          },
19.          methods: {
20.              test() {
21.                  alert('Hola')
22.              }
23.          }
24.      }
25.      </script>
```

La librería que provee de las herramientas para la definición de componentes en Vue.js es la base.

En el proyecto ASys Web Client existen componentes creados a mano, y componentes externos creados por otras personas para su reutilización en distintos proyectos. Los componentes desarrollados especialmente para ASys se encuentran en la carpeta *components* de la imagen mostrada anteriormente. Y los componentes externos suelen añadirse como librerías y se encuentran en la carpeta *node_modules*.

5.2.2 Vistas

Las vistas en Vue.js no son más que componentes que representan una página y por tanto se utilizaran de forma diferente. Estos componentes especiales llamados vistas, se encargan de montar una estructura de componentes reutilizables como si fueran piezas de Lego, y además de encargarse de gestionar los datos necesarios que puedan requerir para suministrárselos a unos componentes u a otros, definiendo de esta manera las relaciones entre los mismos.

Las vistas siguen siendo componentes normales de Vue.js por lo que la forma de crearlos y la librería utilizada para su desarrollo sigue siendo la mismas [51].

5.2.3 Router

El *router* es una pieza fundamental de las aplicaciones de una sola página (*SPA Single Page Applications*) y por lo tanto es una pieza que no puede faltar en ningún *framework* completo. Un *router* es el elemento software encargado de relacionar una URL a una vista, de manera que cuando se escriba una URL en el navegador, automáticamente se visualizara la vista relacionada. Además, un *router* debe proveer de

diferentes herramientas para poder enviar datos dinámicos a las vistas y para establecer un sistema de permisos.

El *router* desarrollado para el proyecto ASys Web Client consta de las diferentes asociaciones URL / vista, y es donde se definen los permisos de entrada [67]. Por ejemplo, para las vistas de la intranet no se deberá poder acceder a no ser que el usuario se encuentre autentificado, y en caso de querer acceder a la mismas sin estarlo se debe bloquear o redirigir a otra diferente.

Un ejemplo simplificado de como sería una implementación de un *router* en ASys Web Client con Vue.js sería el siguiente:

```
1. Vue.use(Router)
2.
3. const routes = [
4.   {
5.     path: '/',
6.     name: 'index',
7.     component: Index
8.   },
9.   {
10.          path: '/intranet',
11.          name: 'intranet',
12.          meta: { requiresAuth: true },
13.          component: () => import('../views/private/Intranet.vue'
   ),
14.         }
15.      ]
16.      const router = new Router({
17.        routes
18.      })
```

Para el desarrollo de *routers* Vue.js provee de una librería denominada Vue Router, con todas las herramientas necesarias para su implementación. El *router* del proyecto ASys Web Client se encuentra en la carpeta *router* de la figura 16 con la estructura del proyecto.

5.2.4 Store

La *store* es el módulo más complejo de entender a nivel conceptual. La *store* en Vue.js es una implementación propia usando la librería Vuex, del patrón de diseño Redux, creado por Facebook [68]. Esta implementación es mucho más sencilla y fácil de usar que la original de Facebook, además de integrarse mucho mejor y de forma más natural con el resto de librerías en Vue.js.

La incorporación de una *store* viene dado por la necesidad de hacer una mejor gestión de los datos de forma centralizada en aplicaciones reactivas orientadas a componentes. En una aplicación orientada a componentes cada uno de ellos suele necesitar unos datos y los gestiona internamente. Cuando se añade interacción entre los mismo la complejidad crece enormemente, así como la capacidad para hacer un seguimiento de estos haciendo debug. Esto es debido a la gran cantidad de componentes que puede haber en una aplicación conectados formando un árbol de conexiones por las que se moverán los datos de unos componentes a otros.

Una *store* es como un almacén de datos global a la aplicación en el cual los componentes podrán añadir y obtener estos de forma organizada, haciendo uso de una serie de herramientas que facilitan esta serie de interacciones de forma ordenada para que se pueda realizar un seguimiento de los cambios en los datos.

Con Vuex esta *store* puede dividirse en secciones a gusto del desarrollador para poder organizar los datos de la forma que se desee. En ASys esta división viene dada por los elementos del dominio. Como pueden ser usuarios, ejercicios, ajustes etc...

Debido a la extensión que tienen los modulas de una *store* no se va a proveer un ejemplo, pero se puede observar a uno si se desea, en la carpeta store del proyecto.

5.2.5 Service

La capa de servicio se creó por la necesidad de centralizar en un único lugar todas las llamadas de acceso a datos del servidor. Puesto que en un principio se realizaban en los diferentes componentes, pero había que repetir el mismo código de llamadas cuando dos componentes requerían los mismos datos. Después de esto se añadieron a la *store* pero también daba lugar a código duplicado. Finalmente se optó por centralizar las llamadas en su propia sección del proyecto dando lugar a la carpeta *service* mostrada en la figura 16.

En esta sección se ha seguido la misma estructura que en la *store*, creando una interfaz común pero descompuesta en archivos que contienen las diferentes secciones de las llamadas en base a los elementos del dominio.

Por los mismos motivos que la *store* no se va a proveer de un ejemplo de módulo contenido en la capa de servicio, pero si se desea se puede acceder al proyecto para visualizar la estructura.

5.2.6 Internacionalización

La internacionalización es un requisito que apareció a posteriori durante el desarrollo, debido a la necesidad de contemplar los usuarios de diferentes países que pudieran utilizar la plataforma.

Tras realizar una pequeña investigación de las herramientas utilizadas para la internacionalización en este tipo de aplicaciones, se pudo observar que en muchos de los proyectos analizados se empleaba la herramienta i18n [69].

I18n es una herramienta muy sencilla que permite definir variables de texto globales. Las cuales estarán replicadas en diferentes archivos, en los cuales se cambiará el contenido de las variables acuerdo al idioma deseado. Por ello existe una carpeta *lang* en el proyecto que contiene los archivos de idioma con los textos que emplea la aplicación traducidos.

Por limitaciones de tiempo y de conocimiento por mi parte, solo se ha dado soporte al castellano y al inglés.

6 Implantación

En este apartado se explica cómo trasladar la solución propuesta a un entorno de explotación. En una primera instancia se va a describir el procedimiento necesario para la preparación de los proyectos y acto seguido se comentará cómo instalarlos ya preparados en el entorno de explotación.

6.1 Preparación

En primer lugar, voy a preparar, configurar y adaptar el proyecto ASys Web Client al nuevo entorno de explotación. Para ello voy a enumerar los pasos necesarios.

1. Descargar las dependencias del proyecto mediante el comando `npm install` en el directorio del proyecto. Este comando se encargará de descargar todas las dependencias contenidas en el archivo package.json.

2. Abrir el cliente visual de configuración web de Vue mediante el comando `vue ui` en el directorio del proyecto [70]. Una vez arrancado el servidor accederemos con el navegador a la URL que nos indique el terminal donde hayamos ejecutado el comando.

3. En el cliente visual de vue, importaremos el proyecto y accederemos a la sección `Project Configuration` -> `Vue Cli`. En este apartado en el campo Base URL añadiremos la siguiente ruta `./`. Tras modificarlo haremos clic en el botón `Save changes` situado en la parte inferior izquierda de la página. Esta ruta hará que el proyecto se sitúe relativamente en una ruta cualquiera del servidor. Si hemos realizado correctamente el procedimiento, se nos creara en el proyecto un archivo llamado `vue.config.js`.

4. Una vez configurado Vue Cli, eliminaremos la opción *mode historic* del *router* en el archivo index.js dentro de la carpeta *router*.

5. Tras realizar todos los pasos anteriores construiremos el proyecto para producción mediante el comando `npm run build`. Si el proyecto se ha construido correctamente se nos habrá creado una carpeta *dist* en el proyecto con la app lista para ser desplegada en el servidor de explotación.

A continuación, configuraremos y generaremos el proyecto ASys Web Server para ser desplegado [71].

1. Configuraremos las credenciales del servidor SMTP o en caso de no disponer de uno, insertaremos los datos de nuestra cuenta de Mailtrap con un servidor virtual [52]. Para configurarlo tendremos que modificar el contenido de las propiedades estáticas en la clase `src / main / java / com / asysweb / útil / JavaEmailUtil.java` con los datos necesarios [72].

2. Configuraremos las credenciales del servidor de recursos MinIO modificando las propiedades de la clase `src / main / java / com / asysweb / service / AmazonS3ServiceImpl.java`. En estas propiedades deberemos meter la URL donde este desplegado el servidor y las credenciales que se le hayan asignado.

3. Configuraremos las propiedades del archivo `src / main / resources / application.yml` de la siguiente manera

    a. La propiedad `jpa / hibernate / ddl-auto` la pondremos a `none`

    b. Las propiedades situadas bajo la propiedad `spring / datasource` las configuraremos acorde a los parámetros de la base de datos a utilizar. Indicando correctamente el usuario y la contraseña.

    c. Cambiaremos el valor de la propiedad 'cors / path1' por la URL donde vayamos a desplegar la aplicación web ASys Web Client.

4. Cambiaremos el método principal de arranque de la aplicación que hace uso de Spring Boot. Para ello comentaremos el método `main` y descomentaremos el método `configure` situado en la parte inferior. Ademas en la misma clase extenderemos de `SpringBootServletIniatilizer`.

5. En el pm.xml situado en la raíz del proyecto modificaremos la propiedad packaging de jar a war.

6. Finalmente construiremos el proyecto y se generará un archivo war en la carpeta target. Este archivo es el que utilizaremos para desplegar la aplicación en el servidor de explotación. A este archivo le deberemos de cambiar el nombre según como queramos acceder al mismo a través de la URL una vez desplegado. Para desplegarlo en la URL base del servidor habrá que ponerle el nombre ROOT.

Tras realizar estos pasos ya tendremos nuestros proyectos preparados para ser desplegados en el entorno de explotación.

NOTA: Conviene exportar el script con la base de datos generada automáticamente al ejecutar el proyecto ASys Web Server en local. En caso de no disponer de la base de datos creada deberemos tener la propiedad `jpa / hibernate / ddl-auto` del archivo application.yml a `créate`, y una vez generada la base de datos deberemos de ponerla a `none`, puesto que si se deja con el valor `create`, cada vez que se despliegue el servidor creara de nuevo la base de datos borrando todo el contenido del mismo.

6.2 Instalación

Antes de desplegar los proyectos ya preparados hay que instalar los servidores necesarios para su correcto funcionamiento.

En la maquina o maquinas donde se desee desplegar las aplicaciones descargaremos un servidor Tomcat, un servidor Mysql, un servidor SMTP en caso de no usar Mailtrap, y finalmente descargaremos el binario de MinIO correspondiente al sistema operativo en el que estemos desplegando.

En la instalación del servidor Mysql indicaremos el usuario y contraseña que hayamos puesto en el archivo de configuración de ASys Web Server. En caso de no poder especificar el usuario, por defecto es *root*. Tras esto crearemos una nueva BDs con el nombre que hayamos puesto en el archivo de configuración del proyecto, ya sea mediante la línea de comandos o con algún cliente que facilite su uso. En mi caso la base de datos se llamará asys-v2. A continuación importaremos el script de la base de datos generado con anterioridad.

Para instalar el servidor MinIO extraeremos del proyecto ASys Web Server la carpeta con su mismo nombre, y añadiremos el ejecutable descargado previamente en su interior. Tras esto ejecutaremos el binario indicando el puerto donde deseemos arrancar el servidor y el directorio que queremos que utilice para almacenar los recursos. Según el sistema operativo este comando puede variar. En el entorno utilizado para las pruebas de este trabajo hay instalado un sistema operativo Linux, por lo que el comando utilizado es el siguiente `exec ./minio server ./ --address ":9001"`. Una vez desplegado por primera vez el servidor se nos creara una carpeta de configuración oculta denominada `.minio.sys.` Dentro de esta carpeta en el directorio *config* existe un archivo config.json que deberemos modificar para cambiar las credenciales a las que hayamos definido en el proyecto ASys Web Server. Las propiedades a cambiar en el archivo son *accessKey* y *secretKey* principalmente. Tras haber realizado el cambio es necesario reiniciar el servidor para aplicar lo.

Para finalizar en el servidor Tomcat deberemos arrastrar los proyectos generados previamente a la carpeta webapps. El archivo *war* que contiene la aplicación ASys Web Server y la carpeta *dist* que contiene la aplicación ASys Web Client. Para indicar a Tomcat en que URL queremos desplegar los proyectos bastara con renombrar los archivos a arrastrar en la carpeta *webapp*. Si se quiere usar la URL base, se utilizará el nombre ROOT. Una vez tengamos los proyectos en la carpeta *webapps*, arrancaremos el servidor con el archivo *startup* que se encuentra en la carpeta *bin* dentro del servidor Tomcat.

NOTA: Es importante saber que para un entorno de producción es imprescindible instalar un certificado SSL en el servidor para hacer uso del protocolo https. Si no se hace uso de este protocolo la aplicación es muy insegura y se podría dar pie con facilidad a suplantaciones de identidad y robo de cuentas. Además, no se podrá hacer uso de la instalación de la aplicación web desde el navegador.

7 Pruebas

En este apartado se va a presentar una batería de mediciones y pruebas para comparar mediante una evaluación empírica la primera versión de ASys Web con la segunda versión desarrollada en este trabajo. Con estas pruebas se pretende evaluar la mejoría de la nueva versión con respecto a la primera y comprobar si se han conseguido los objetivos propuestos. Para ello, se van a realizar pruebas para evaluar la calidad del proyecto.

Por un lado, se va a realizar un estudio de la calidad interna del proyecto, ejecutando un analizador estático muy utilizado en el mundo empresarial denominado SonarQube, el cual se encarga de ejecutar una serie de algoritmos que miden la calidad interna del proyecto y ofrecen los resultados como métricas [73].

Por otro lado, se va a realizar un análisis de calidad externa del producto, para comparar el rendimiento en el mismo entorno de explotación de la primera versión de ASys Web, con la desarrollada para este trabajo. Para ello se va a medir el tiempo de respuesta de ambas versiones para observar cuan rápidos son ambos proyectos, y además se va a medir la cantidad de comunicaciones realizadas durante la navegabilidad de las aplicaciones, para de esta manera observar cuantos usuarios podría soportar al mismo tiempo un proyecto con respecto al otro.

7.1 Análisis de calidad interna

Primero se va a mostrar el resultado del proyecto relacionado con este trabajo, nada más terminar la etapa de desarrollo, y se va a comparar con el resultado de haberle dedicado unas cuantas sesiones de refactorización para mejorar su calidad interna y corregir los bugs detectados por SonarQube [73].

Cabe mencionar que el análisis de calidad interna **únicamente se ha realizado sobre los servidores escritos en Java**. Se ha descartado el análisis de las aplicaciones web escritas en Javascript debido a que no habría sido justo, ya que son aplicaciones con orientaciones totalmente diferentes, una es una aplicación web orientada a componentes, que hace uso de un *framework* progresivo y está totalmente desacoplada del servidor, y la otra es una aplicación más artesanal, que hace uso de diferentes librerías que serían difíciles de excluir del análisis y además el código está bastante acoplado con el del servidor, puesto que se encuentra todo en el mismo proyecto y hace uso de páginas en Jsp que mezcla código Java con código Javascript.

Antes de mostrar el análisis obtenido por SonarQube cabe mencionar que para cada métrica ofrece una valoración representada con letras de la A a la E, siguiendo el orden alfabético. Las cuales representan la severidad o la gravedad de los resultados obtenidos. Por tanto una calificación con la letra A representaría que el proyecto está sano y no es necesario tomar medidas, una calificación con la letra B se puede considerar que los resultados obtenidos son aceptables pero es conveniente tomar medidas para que no empeore, una calificación con la letra C entramos en un rango en el que el proyecto no está sano y hay que empezar a tomar medidas correctivas, una valoración con la letra D quiere decir que es un urgente tomar medidas para llevar al proyecto a zonas más saludables, y una valoración con la letra E representa que hemos entrado en la zona de rescate, hay que valorar si merece la pena invertir recursos en rescatar el proyecto y mejorarlo o es mejor dejar que muera y tomar otro tipo de medidas [74].

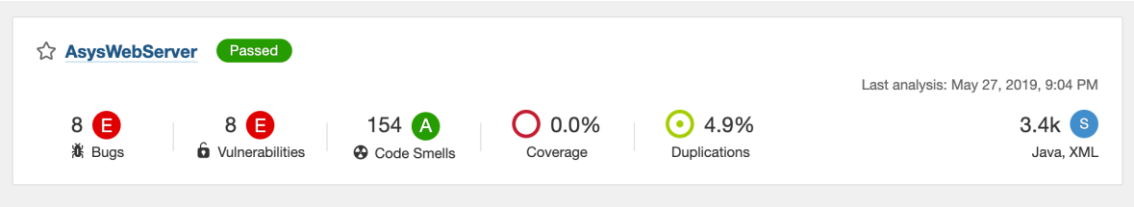


Ilustración 17: Análisis general SonarQube versión de desarrollo

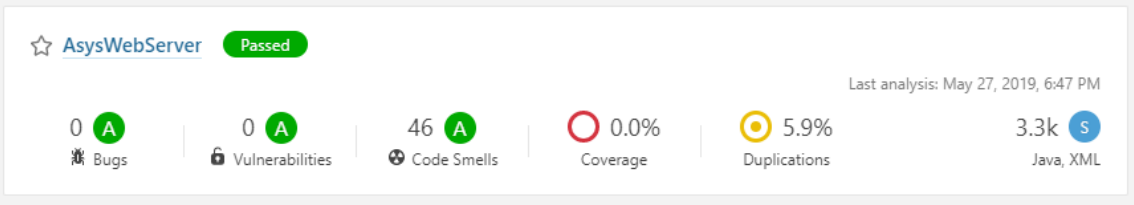


Ilustración 18: Análisis general SonarQube versión de mantenimiento

A la vista de los resultados obtenidos se puede comprobar que tras las sesiones de refactorización se han eliminado en su totalidad los bugs y las vulnerabilidades. Además, se ha reducido en gran medida el número de *code smells*, los cuales son consejos o recomendaciones que ofrece SonarQube para prevenir posibles problemas futuros o simplemente recomendaciones para mejorar la mantenibilidad del código. La cobertura es 0.0% en ambos casos puesto que no se han desarrollado test debido a la importancia de desarrollar nuevas funcionalidades frente a implementar casos de prueba. Y finalmente ha aumentado ligeramente el número de código duplicado debido a las correcciones realizadas para bugs y vulnerabilidades en diferentes puntos de la aplicación.

Tras los resultados de haber realizado sesiones de refactorización para mejorar la calidad interna del código, se va a proceder a comparar la versión corregida del proyecto con la de la primera versión de ASys Web, y así comprobar si realmente existe una mejora sustancial en la mantenibilidad de la nueva versión y por tanto a merecido la pena empezar un nuevo proyecto desde cero.

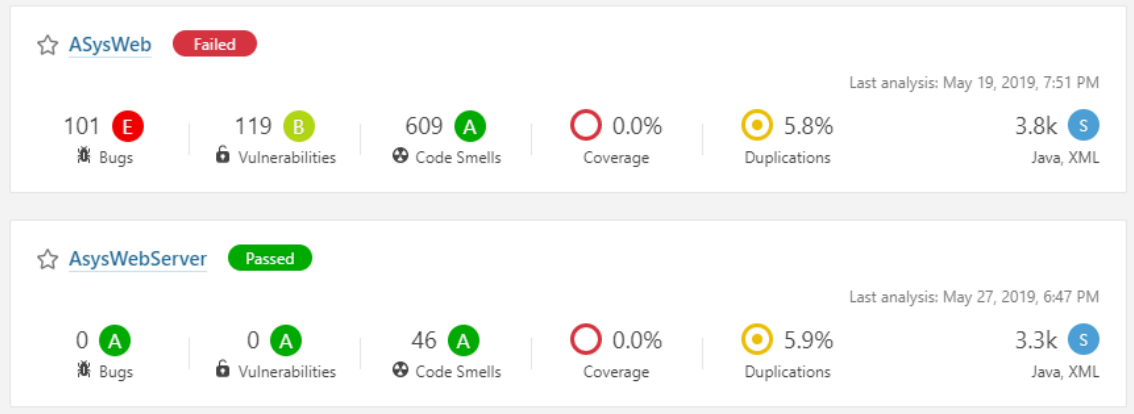


Ilustración 19: Comparación general SonarQube ASys v1 y ASys v2

En la parte superior de la figura 19 con el nombre ASysWeb podemos observar la primera versión del servidor ASys, y en la parte inferior se encuentra la segunda versión desarrollada fruto de este trabajo final de grado.

En las métricas generales se puede observar que en la primera versión de ASys existen 101 bugs, que ha recibido una calificación de gravedad E por parte de SonarQube,

la cual es la más alta dentro de las calificaciones, y habría que hacer un estudio más en profundidad del coste de solucionar estos bugs. Existen 119 vulnerabilidades para las cuales habría que tomar medidas preventivas, 609 *code smells* que no son de gran importancia, pero son detalles que habría que cuidar para mejorar la mantenibilidad. Y finalmente en la primera versión existe un menor número de código duplicado en proporción al número de líneas escritas con respecto a la segunda versión.

Tras los resultados obtenidos en el análisis general, podemos garantizar que la nueva solución de ASys es mucho más segura, mucho más robusta y a falta de un análisis más detallado, que veremos a continuación, parece ser que es mucho más mantenible.

A continuación, se van a mostrar métricas más detalladas, relacionadas con las obtenidas en el análisis general.

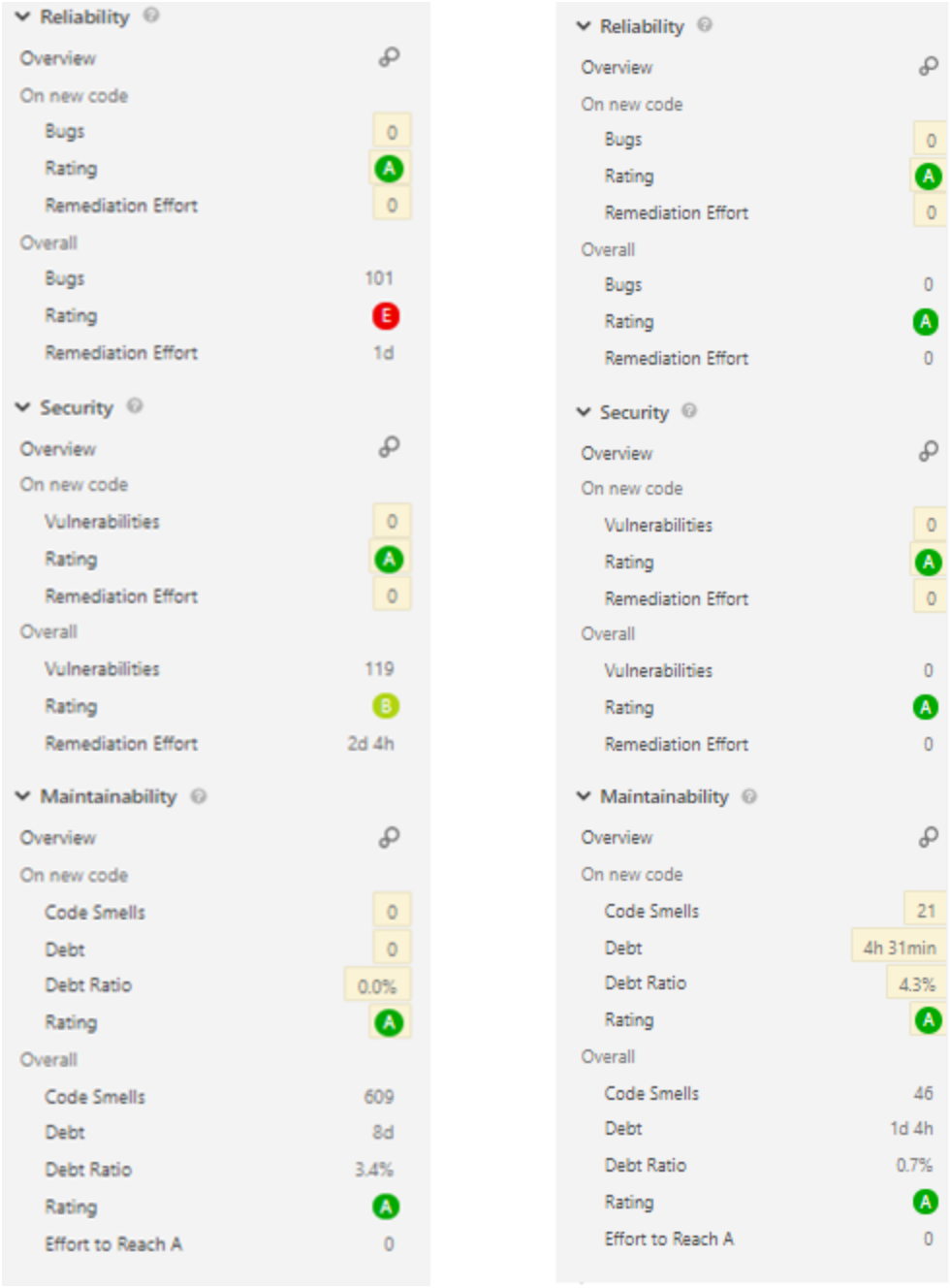


Ilustración 20: Comparación de métricas detalladas SonarQube

A la izquierda se encuentran los datos relacionados con la primera versión de ASys y a la derecha los de la segunda versión.

En las métricas más detalladas podemos observar una estimación en tiempo que nos provee SonarQube para reducir a cero cada una de las métricas generales provistas anteriormente. Comenzando por lo más grave que son los bugs y las vulnerabilidades, podemos observar que en los campos *Overall / Remediation Effort* de las secciones *Reliability y Security*, nos indica que necesitaremos invertir 1 día para solucionar todas los bus y 2 días con 4 horas para solucionar todas las vulnerabilidades, que teniendo en cuenta que SonarQube contempla 1 día como una jornada de 8 horas para un trabajador promedio, en total suman 28 horas de trabajo estimado para un desarrollador promedio, que a priori parece una cifra asumible, pero se encuentra muy por debajo de la necesaria en la nueva versión de ASys. Para reducir al mínimo los *code smells* se necesitan bastantes más horas, pero al no ser elementos críticos, no son muy relevantes para cumplir con los objetivos.

Uno de los mayores problemas que presentaba la primera versión con respecto a la calidad interna y que fue un factor importante para desarrollar la nueva versión, era la gran dificultad que existía para ampliar el código y mantenerlo. Esto era debido a la poca estructuración del código en métodos pequeños, y a la longitud que presentaban las clases. Existían números clases que incorporaban métodos con una extensión de entre 300 y 400 líneas. Este problema se puede visualizar en la métrica ofrecida por sonar relativa a la complejidad ciclomática y a la complejidad cognitiva.

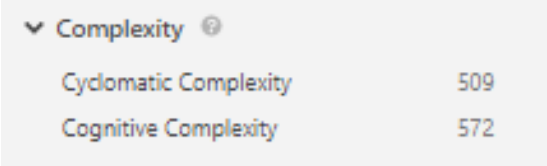


Ilustración 21: Complejidad ASys v1 por SonarQube

La complejidad ciclomática es una métrica relativa a la ingeniería del software, la cual mide el número de flujos distintos de ejecución que puede tener el código de un artefacto software. Nos va a dar una medida de como de dificultosa es la lógica de un programa [75, 76, 77, 78]. Por otro lado, existe la complejidad cognitiva, la cual nos da una medida que nos indica como es de complejo entender un bloque de código, y por tanto de lo mantenible que es [75, 76, 77, 78]. .

A la vista de los resultados anteriores podemos observar que la complejidad cognitiva supera bastante a la complejidad ciclomática, y por tanto esto quiere decir que es tan inmantenible que percibimos mucha más complejidad de la que realmente existe. Comparemos ahora los resultados con los datos obtenidos en la segunda versión de ASys.

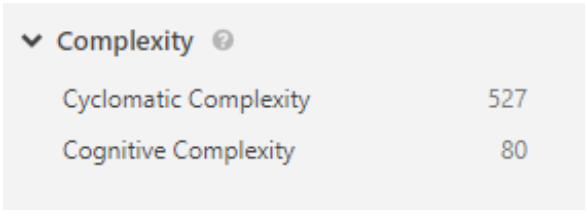


Ilustración 22: Complejidad ASys v2 por SonarQube

En la segunda versión de ASys la complejidad ciclomática es ligeramente superior a la de la primera versión, lo que quiere decir que el código del proyecto es más complejo, pero podemos observar una muy notable mejoría en la percepción de la misma, reduciendo la percepción de la complejidad en casi siete veces. Por tanto, estos resultados

nos indican que el proyecto de la nueva versión de ASys es mucho más mantenible que el de la segunda versión, cumpliendo de esta manera con el objetivo propuesto.

7.2 Análisis de calidad externa

En este análisis se va a proceder a comparar el rendimiento de la primera versión de ASys Web, con la segunda versión desarrollada para este trabajado. Para ello ambas aplicaciones se encuentran desplegadas en un mismo servidor cedido por la universidad, para que las pruebas se realicen en igualdad de condiciones. Como inconveniente cabe resaltar que al estar el servidor cerrado a la universidad y requerir de la VPN para tener acceso externo, no se han podido utilizar herramientas más completas para la obtención de datos como pueden ser Page Insights, Monitis o StressStimulus [79, 80, 81]. Por este motivo los datos se han obtenido de forma manual, haciendo uso de las herramientas para desarrolladores provistas en el navegador Chrome. Esto ha provocado que no se hayan podido obtener una gran cantidad de datos, pero si las suficientes para realizar una estimación aproximada.

En primer lugar, se va a proceder a comparar los tiempos de carga en las diferentes páginas de las aplicaciones. Intentando comparar las páginas homologas de una aplicación con respecto a la otra. Estas métricas serán representativas de la velocidad de ambas aplicaciones.

Las gráficas se realizarán sobre un total de 11 datos estadísticos recogidos para cada una de las versiones de ASys. Ninguno de estos datos se eliminará en base al tamaño de la ventana, para poder visualizar la influencia de la caché sobre el rendimiento del servidor. Todos los valores de las gráficas y los cálculos de las tablas se encuentran en milisegundos.

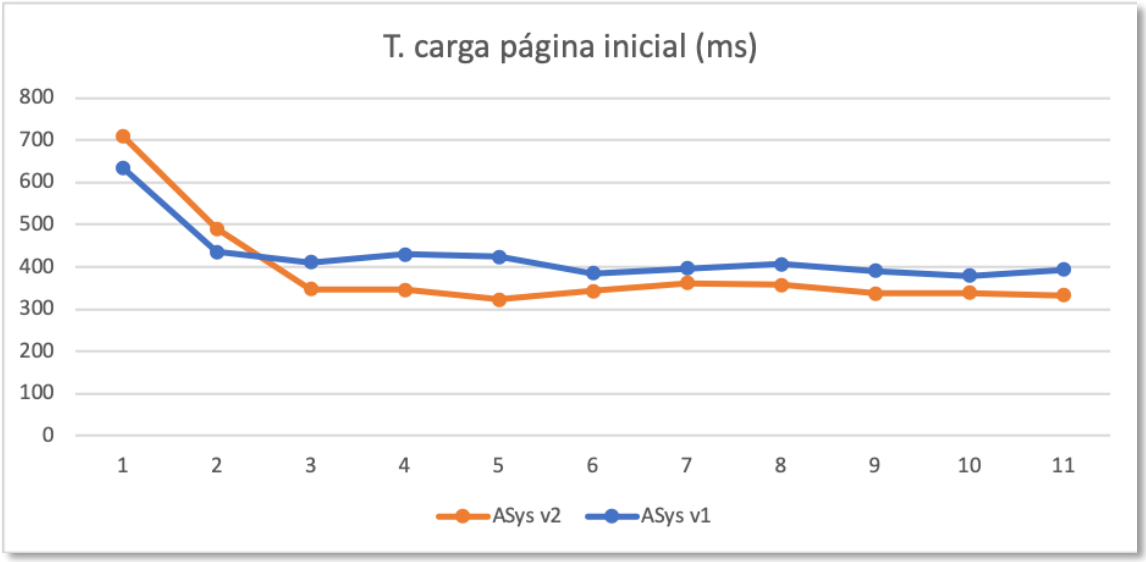


Ilustración 23: Gráfica tiempo de carga página inicial

ASys v1	Valores (ms)	ASys v2	Valores2 (ms)
Media	405,3	Media	389,64
Varianza	4669,06	Varianza	12137,87
Desv. Estándar	71,67	Desv. Estándar	115,55
Cuartil 1	392,5	Cuartil 1	337,5

Mediana	407	Mediana	346
Cuartil 2	407	Cuartil 2	346
Cuartil 3	426,5	Cuartil 3	359,5
Cuartil 4	635	Cuartil 4	710
Moda	#N/D	Moda	#N/D

Tabla 1: Detalles gráfica tiempo de carga página inicial

En la primera página es donde se descargan la mayoría de los archivos para el funcionamiento futuro de las aplicaciones, esto da lugar a que los resultados sean muy parecidos. También, para realizar la medición del tiempo de carga de las aplicaciones, se han tomado los datos sin hacer uso de la caché en ninguna de las dos versiones.

En esta primera carga podemos observar que las medias de las versiones ASys v1 y ASys v2, son bastante parecidas, habiendo una diferencia de alrededor de unos 15 ms, lo cual no es muy significativo. Donde sí hay una diferencia notable, es en la desviación estándar de ambas versiones, siendo la de la segunda versión de ASys mucho mayor. Esto puede ser debido a la mayor cantidad de archivos descargados para visualizar la primera página, los cuales pueden dar lugar a una mayor irregularidad de la carga. Una tabla comparativa de los archivos descargados por página se mostrará más adelante.

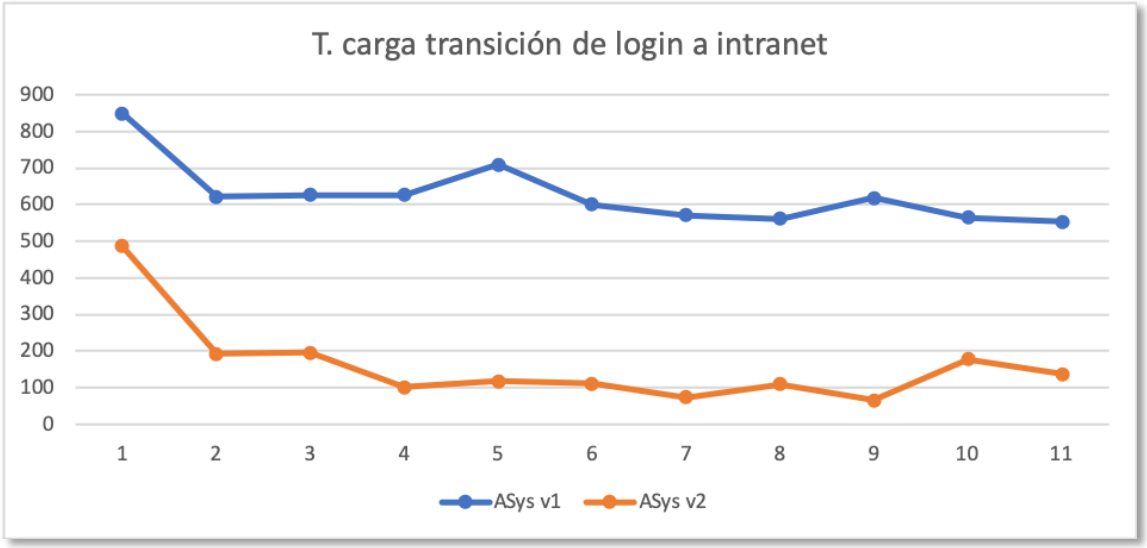


Ilustración 24: Gráfica tiempo de carga transición de login a intranet

ASys v1	Valores (ms)	ASys v2	Valores2 (ms)
Media	627,64	Media	160,82
Varianza	6727,69	Varianza	12435,97
Desv. Estándar	86,03	Desv. Estándar	116,96
Cuartil 1	568,5	Cuartil 1	105,5
Mediana	618	Mediana	117
Cuartil 2	618	Cuartil 2	117
Cuartil 3	626	Cuartil 3	185,5
Cuartil 4	850	Cuartil 4	487

Moda	626	Moda	#N/D

Tabla 2: Detalles tiempo de carga transición de login a intranet

Visualizando la gráfica anterior podemos observar que existen diferencias significativas a la hora de realizar el *login* y acceder a la zona privada de ambas aplicaciones. Si comparamos las medias, observamos que en la segunda versión de ASys el rendimiento es casi 4 veces superior al de la primera versión, lo cual es una mejora muy notable. Como contra, seguimos observando una mayor desviación en la segunda versión, pero en total siempre se encuentra muy por debajo de los tiempos ofrecidos por la primera.

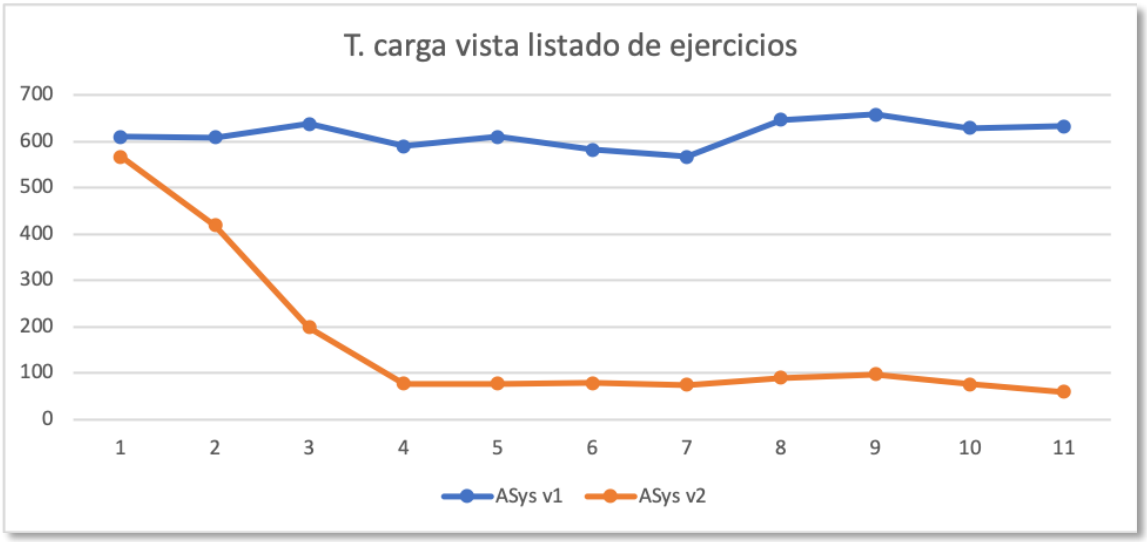


Ilustración 25: Gráfica tiempo de carga listado de ejercicios

ASys v1	Valores (ms)	ASys v2	Valores2 (ms)
Media	615,18	Media	165
Varianza	719,79	Varianza	26086,18
Desv. Estándar	28,14	Desv. Estándar	169,4
Cuartil 1	598,5	Cuartil 1	76,5
Mediana	610	Mediana	79
Cuartil 2	610	Cuartil 2	79
Cuartil 3	635	Cuartil 3	148
Cuartil 4	657	Cuartil 4	567
Moda	#N/D	Moda	77

Tabla 3: Detalles tiempo de carga listado de ejercicios

En esta vista, en la segunda versión de Asys, después de haber descargado los archivos necesarios para su visualización, entre los cuales se encuentran las imágenes de los ejercicios, los cachea y no se los vuelve a descargar, únicamente se realizan las peticiones necesarias para obtener los datos de los ejercicios. Estas peticiones son muy livianas y a penas cuestan tiempo por parte del servidor. Esto hace que la diferencia sea muy significativa, debido a que la primera versión se descarga todos los archivos necesarios para su ejecución en cada visita.

Observando la gráfica, podemos visualizar la mejora sustancial que provoca el uso de la caché. Vemos como el primer dato es prácticamente idéntico el de la segunda versión con respecto a la primera, pero mejora de forma notable en los siguientes accesos, hasta un punto estable en el cual, si nos fijamos en las medianas de ambas gráficas, la mejora en velocidad es de 7.7 veces superior la segunda versión con respecto a la primera.

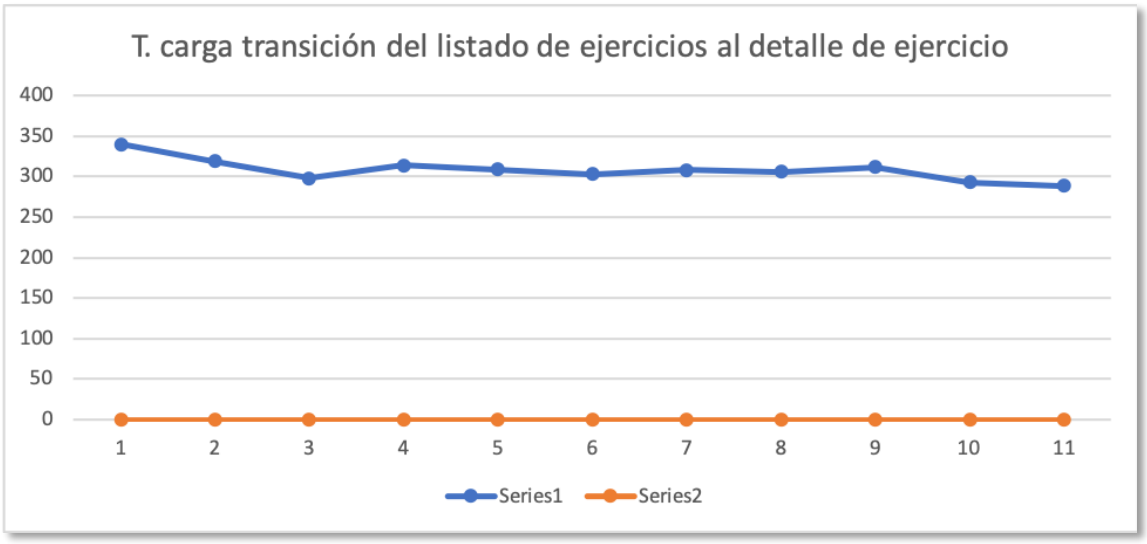


Ilustración 26: Gráfica tiempo de carga transición del listado de ejercicios al detalle

ASys v1	Valores (ms)	ASys v2	Valores2 (ms)
Media	308,27	Media	0
Varianza	173,83	Varianza	0
Desv. Estandar	13,83	Desv. Estandar	0
Cuartil 1	300,5	Cuartil 1	0
Mediana	308	Mediana	0
Cuartil 2	308	Cuartil 2	0
Cuartil 3	313	Cuartil 3	0
Cuartil 4	340	Cuartil 4	0
Moda	#N/D	Moda	0

Tabla 4: Detalle gráfica tiempo de carga transición del listado de ejercicios al detalle

Esta es la gráfica con el mayor uso de caché por parte de la segunda versión de ASys, y por tanto con la diferencia más palpable en cuanto a rendimiento se refiere. En la segunda versión solo se puede acceder al detalle de un ejercicio desde el listado de los mismos. En este listado se obtiene toda la información de cada uno de los ejercicios, y se almacena en la caché local, para cuando se quiera acceder a la información de un ejercicio. Por tanto, al visualizar el detalle no se realiza ninguna petición, sino que se recogen los datos del listado guardado previamente, esto sumado a que todos los archivos necesarios para el funcionamiento de la aplicación ya están cargados, da lugar a que el tiempo de carga en esta vista sea cero.

A continuación, vamos a visualizar las peticiones realizadas para la visualización de las diferentes páginas en cada versión. Y de esta manera poder comprobar el uso requerido por parte del servidor

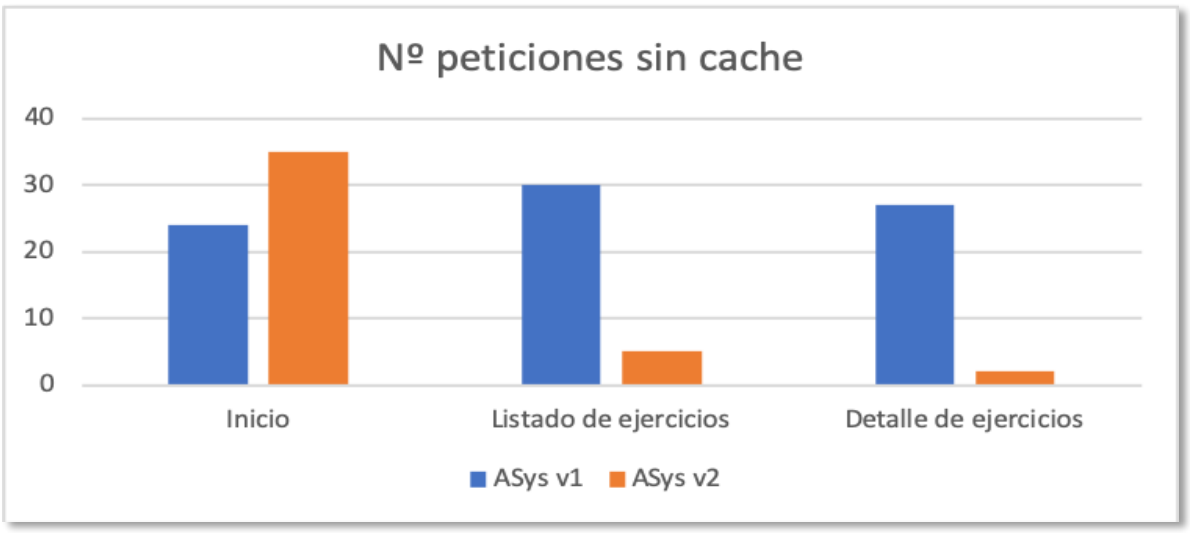


Ilustración 27: Gráfica nº peticiones por página sin caché

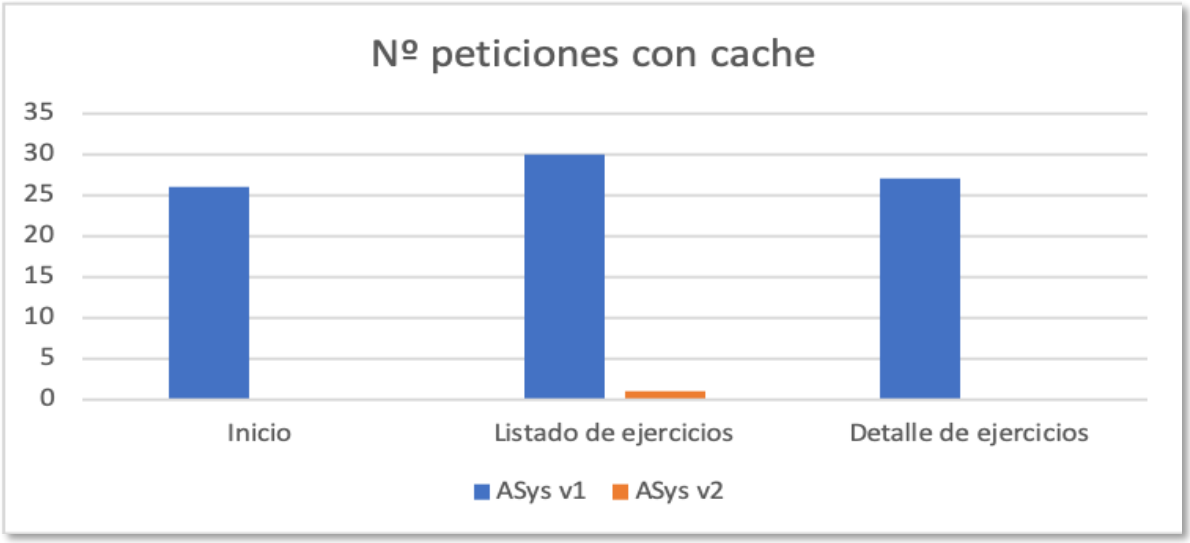


Ilustración 28: Gráfica nº peticiones por página con caché

En estas gráficas se puede observar como en la segunda versión de ASys se realiza un mayor número de peticiones en la carga inicial en comparación con la primera versión. Debido a que requiere de muchos más archivos para su funcionamiento. Pero esto no es un problema debido a que lo compensa de gran manera en la futura navegación por la aplicación. Podemos observar cómo mientras se navega apenas se realizan peticiones al acceder por primera vez a las vistas, y además cuando se vuelve a navegar por una vista ya accedida, el número de peticiones es prácticamente inexistente. Esto da lugar a la capacidad para soportar una mayor cantidad de usuarios concurrentes en la segunda versión, puesto que el uso del servidor por parte de cada usuario, se ha visto muy reducido en la navegación general de la aplicación. Aunque habrá que tener cuidado con los posibles picos de carga en la página inicial, este problema se puede disminuir en gran medida mediante la instalación de la aplicación como si fuera una aplicación web progresiva.

8 Conclusión

La realización de la segunda versión del sistema Asys, desde mi punto de vista como alumno, fue muy atractiva desde el principio, y ha sido muy gratificante en su consecución.

Desde el principio, el aspecto más importante y del que más orgulloso me siento, es de haber recibido la confianza y la propuesta por parte de mi tutor Josep Silva para colaborar en este proyecto de investigación de vanguardia; que no solo es un reto a nivel técnico, sino que además es un proyecto que va a trascender en el entorno académico en el que me he formado. Para mí ha sido un gran honor y un orgullo trabajar en este proyecto.

Por otro lado, realizar este proyecto ha supuesto un enorme aprendizaje a nivel técnico de un gran número de tecnologías, algunas de ellas muy novedosas y de vanguardia, otras de ellas muy robustas y asentadas en el mercado profesional, así como otras nuevas que no existían en otros ámbitos y que han sido fruto de una larga investigación. Esto junto con el aprendizaje más metódico y de proceso, impartido por mi tutor Josep Silva, el cual me ha guiado y orientado en todo momento, ha hecho que el alumno que empezó este trabajo fin de carrera y el alumno que se encuentra ahora mismo escribiendo estas palabras sea totalmente diferente. El resultado de este aprendizaje se ha visto reflejado como éxito profesional, siendo contratado por una empresa de vanguardia, para seguir desarrollando como experto en las tecnologías aprendidas a lo largo de este trabajo fin de grado.

En lo que respecta a la consecución de los objetivos propuestos para este trabajo, cabe mencionar que este trabajo tuvo como objetivo el realizar una migración de tecnología de un sistema. El desarrollo de la nueva versión consistía en crear un esqueleto y/o una estructura base, realizando un cambio total del planteamiento inicial, el cual debería mejorar tanto los aspectos de mantenibilidad como los aspectos de rendimiento con respecto a versiones anteriores. A la vista de los resultados obtenidos en las pruebas realizadas, el trabajo ha sido un completo éxito. La nueva versión de ASys es mucho más mantenible, es mucho más rápida, es mucho más eficiente y es mucho más escalable de cara a un futuro.

Durante la realización de este trabajo han surgido numerosas dificultades, que a pesar del esfuerzo que ha supuesto afrontarlas todas, se han visto reflejadas en un crecimiento personal y profesional. Como decía un buen profesor de mi infancia, “Todo crecimiento conlleva un sufrimiento”. Algunas de las dificultades más destacadas han sido las siguientes:

- La formación en una cantidad de librerías y *frameworks*, para los cuales ha habido que dedicar muchísimo esfuerzo y tiempo (casi el mismo que para el desarrollo).
- La realización de varios proyectos desde cero con tecnologías totalmente desconocidas para los cuales se ha invertido mucho tiempo en hacer que todo funcione como los engranajes de un reloj. Hacer que todas las piezas encajen y funcionen bien juntas siempre supone una gran dificultad.
- Y, finalmente, quiero destacar la tarea de realizar una memoria tan extensa y formal como esta, para la cual no poseía los conocimientos necesarios y no estoy acostumbrado a realizar, pero gracias a la orientación de mi tutor he podido mejorar mucho mi capacidad de redacción.

Por último, además del producto desarrollado, los conocimientos y las habilidades de trabajo en equipo que he adquirido son un resultado de este proyecto de un enorme

valor. Estos conocimientos combinan e integran por primera vez en la carrera las competencias adquiridas en las diferentes asignaturas de la carrera. En el siguiente apartado explico esta relación con las asignaturas y su influencia en el trabajo realizado.

8.1 Relación del trabajo desarrollado con los estudios cursados

Durante toda la carrera se han obtenido numerosos conocimientos y habilidades que se han visto reflejados directa o indirectamente en el desarrollo de este trabajo. Algunas de ellas han influido en un cambio en mi forma de pensar, que ahora es mucho más crítico y más analítico, y muchas otras se pueden ver de forma más directa, las cuales mencionaré a continuación.

Por lenguajes de programación y temática, se han utilizado en gran medida los conocimientos obtenidos en las asignaturas de Concurrencia y Sistemas Distribuidos (CSD), Redes de computadores y Tecnologías de sistemas de la información en red (TSR). En estas asignaturas enseñan los conocimientos teóricos y prácticos para realizar comunicaciones entre varios artefactos software, así como en desarrollar aplicaciones concurrentes y en red.

Por metodología, proceso y documentación, han sido muy útiles casi todos los conocimientos relativos a las asignaturas de la rama de especialización en ingeniería del software. En estas asignaturas se han adquirido conocimientos en análisis y especificación de requisitos (AER), mejora del mantenimiento y la calidad del proyecto (CSO y MES), así como en el proceso de desarrollo necesario para realizar un proyecto software exitoso (PSW y PIN). Y, por supuesto, la introducción a la ingeniería del software (ISW) impartida en el primer cuatrimestre del tercer curso.

9 Trabajos futuros

ASys es un proyecto muy grande y ambicioso que requiere mucho desarrollo para ser completado.

Algunas funcionalidades que se desarrollarán en un futuro son las siguientes:

- Desarrollo de la lógica necesaria para subir ejercicios por hacer por parte del profesorado.
- Desarrollo de un sistema de creación de ejercicios para el profesorado.
- Desarrollo de un sistema de grupos y permisos para relacionar en salas los profesores con sus alumnos, y que solo ellos puedan acceder a los ejercicios publicados.
- Crear un IDE embebido en la plataforma para la realización de ejercicios por parte del alumno.
- Desarrollo de una plataforma de corrección manual para el profesorado.
- Desarrollo de una plataforma de corrección semiautomática para el profesorado
- Desarrollar un sistema de subida de recursos (pdfs, videos, docx, etc.) por parte de los profesores para los alumnos en un grupo.
- Desarrollo de diferentes ASys Client para soportar otros tipos de correcciones para otras asignaturas que no sean de programación.
- Desarrollar un modo sin conexión para que los usuarios puedan trabajar con la plataforma sin necesidad de acceso a internet.

10 Bibliografía

[1] Insa D., Silva J., «Semi-Automatic Assessment of Unrestrained Java Code: A Library, a DSL, and a Workbench to Assess Exams and Exercises,» *ITiCSE,* pp. 39-44, 2015.

[2] Insa D., Silva J., «Automatic assessment of Java code,» *Computer Languages, Systems & Structures,* pp. 53-72, 2018.

[3] Insa D., Silva J., Tamarit S., «Where You Sit MattersHow Classroom Seating Might Affect Marks,» *ITiCSE,* pp. 212-217, 2016.

[4] España S., Insa D., Silva J., Tamarit S., «In what order should i correct the exercises? Determining the evaluation order for the automatic assessment of programming exercisses,» *ITHET,* pp. 1-3, 2017.

[5] Universidad Politècnica de València, «PoliformaT,» 2003. [En línea]. Available: https://poliformat.upv.es.

[6] MIT, «Cursos online gratis de Harvard, MIT y más - edX,» 2011. [En línea]. Available: https://www.edx.org/.

[7] Udemy Inc, «Udemy: Online Courses - Learn Anything, On Your Schedule,» 2010. [En línea]. Available: https://www.udemy.com/.

[8] Udacity Inc, «Udacity: Learn the Latest Tech Skills; Advance Your Career,» 2011. [En línea]. Available: https://eu.udacity.com/.

[9] Sims Z., Bubinski R., «Codecademy: Learn to Code - for Fre,» 2011. [En línea]. Available: https://www.codecademy.com/.

[10] Pluralsight LLC, «Pluralsight + Code School,» 2004. [En línea]. Available: https://www.pluralsight.com/codeschool.

[11] Demir et al., Demir, Ö., Soysal, A. S., Arslan, A., Yürekli, Ö. Y. B., and lmazel. «Automatic grading system for programming homework.,» *Computer Sciencie Education: Innovation and Technology,* 2010.

[12] Denny et al., Denny, P., Luxton-Reilly, A., Tempero, E., and Hendrickx, J., «CodeWrite: Supporting student-driven practice of Java,» ACM technical symposium on Computer sciencie education, pp. 471-476.

[13] Benac Earle et al., Benac Earle, C., Fredlung, L.-A., and Hughes, J., «Automatic grading of programming exercises using property-based testing.,» *ACM Conference on Innovation and Technology in Computer Sciencie Education,* 2016.

[14] Krusche S., Seitz, A., «Artemis: An automatic assessment management system for interactive learning.,» ACM Technical Symposium on Computer Sciencie Education, 2018, pp. 284-289.

[15] Bey et al. Bey, A., Jermann, P., and Dillenbourg, P., «A comparison between two automatic assessment approaches for programming: An empirical study on moocs,» Journal of Educational Technology & Society, pp. 259-272.

[16] Ala-Mutka K. M., «A survey of automated assessment approaches for programming assignments,» de *Computer Sciencie Education*, 2005, pp. 83-102.

[17] Abd Rahman K., Jan Nordin M., «A review on the static analysis approach in the automated programming assessment systems,» *National Conference on Programming 07,* 2007.

[18] Liang et al., Liang, Y., Liu, Q., Xu, J., and Wang, D., «The recent development of automated programming assessment,» de *International Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1-5.

[19] Ihantola et al., Ihantola, P., Ahoniemi, T., Karavirta, V., and Seppala, O., «Review of recent systems for automatic assessment of programming assigments,» de *In Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, 2010, pp. 86-93.

[20] Tung et al., Tung, S.-H., Lin, T.-T., and Lin, Y.-H., «An exercise management systems for teaching programming,» de *Journal of Software*, 2013, pp. 1718-1725.

[21] Kitaya, H., Inoue, U., «An online automated scoring system for Java programming assignments,» de *International Journal of Information and Education Technology*, 2014, pp. 275-279.

[22] Beierle et al., Beierle, C., Kulas, M., and Widera, M., «Automatic analysis of programming assignments,» In proceedings of the 1. E-Learning Fachtagung Informatik, pp. 144-153.

[23] Edwards S. H. and Pérez-Quiñones, M., «Web-CAT: Automatically grading programming assignments,» ACM SIGCSE Bulletin, 2008, pp. 328-328.

[24] Naude et al. Naude, K. A., Greyling, J. H., and Vogts, D., «Making student programs using graph similarity,» Computers & Education, 2010, pp. 545-561.

[25] Wang et al. Wang, T., Su, X., Wang, Y., and Ma, P., «Semantic similarity-based grading of student programs.,» Information and Software Technology, pp. 99-107.

[26] Vujosevic-Janicic et al. Vujosevic-Janicic, M., Nikolic, M., Tosic, D., and Kuncak, V., «Software verification and graph similarity for automated evaluation of students assignments,» de *Information & Software Technology*, pp. 1004-1016.

[27] Marin et al. Marin, V. J., Pereira, T., Sridharan, S., and Rivero, C. R., «Automated personalized feedback in introductory java programming moocs,» de *IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017, pp. 1259-1270.

[28] Jamil, H. M., «Automated personalized assessment of computational thinking mooc assignments,» de *IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, 2017, pp. 261-263.

[29] Muuli et al. Muuli, E., Papli, K., Tonisson, E., Lepp, M., Palts, T., Suviste, R., Sade, M., and Luik, P., «Automatic assessment of programming assignments using

image recognition,» de *2017 European Conference on Technology Enhanced Learning*, 2017, pp. 153-163.

[30] Xiong, Y. and Suen, H. K., «Assessment approaches in massive open online courses: Possibilities, challenges and future directions,» de *International Review of Education*, 2018, pp. 241-263.

[31] Lamsweerde, A. V., Requirements Engineering: From System Goals to UML Models to Software Specifications, John Wiley & Sons, 2009.

[32] Kotonya G., Sommerville I., Requirements Engineering: Processes and Techniques (Worldwide Series in Computer Science), John Wiley & Sons, 1998.

[33] Fielding, R. T., Architectural Styles and the Design of Network-based Software Architectures, University of California, Irvine, 2000.

[34] Parecki A., «OAuth 2.0,» 2012. [En línea]. Available: https://oauth.net/2/.

[35] Nascimento A. E., OAuth 2.0 Cookbook: Protect your web applications using Spring Security, Packt Publishing, 2017.

[36] OpenID Foundation, «OpenId Connect,» 2014. [En línea]. Available: https://openid.net/connect/.

[37] Jones M., «JSON Web Token (JWT, RFC 7519),» 2015. [En línea]. Available: https://tools.ietf.org/html/rfc7519.

[38] Khillar S., «Difference between Authentication and Authorization,» 2017. [En línea]. Available: http://www.differencebetween.net/technology/difference-between-authentication-and-authorization/.

[39] Richardson C., «Pattern: Monolithic Architecture,» 2018. [En línea]. Available: https://microservices.io/patterns/monolithic.html.

[40] Richardson C., «Pattern: Microservice Architecture,» 2018. [En línea]. Available: https://microservices.io/patterns/microservices.html.

[41] Garzas J., «¿Qué es eso de los microservicios?,» 2015. [En línea]. Available: https://www.javiergarzas.com/2015/06/microservicios.html.

[42] Spark Team, «Spark Framework: An expressive web framework for Kotlin and Java,» 2011. [En línea]. Available: http://sparkjava.com/.

[43] Restlet Inc, «Restlet documentation,» 28 03 2019. [En línea]. Available: https://restlet.com/documentation/.

[44] The Apache Software Foundation , «Welcome to the Apache Struts project,» 2000. [En línea]. Available: https://struts.apache.org/.

[45] Pivotal Software Inc, «Spring,» 2003. [En línea]. Available: https://spring.io/.

[46] MinIO Inc, «MINIO object storage for AI,» 14 03 2019. [En línea]. Available: https://min.io/.

[47] MinIO Inc, «MinIO Client SDK for Java,» 17 03 2019. [En línea]. Available: https://github.com/minio/minio-java.

[48] Google, «AngularJS — Superheroic JavaScript MVW Framework,» 2010. [En línea]. Available: https://angular.io/.

[49] Facebook Inc, «React Una biblioteca de JavaScript para construir interfaces de usuario,» 2013. [En línea]. Available: https://es.reactjs.org/.

[50] Google, «Polymer Project,» 2012. [En línea]. Available: https://www.polymer-project.org/.

[51] Vuejs Team, «Guide to Vue.js,» 01 07 2016. [En línea]. Available: https://vuejs.org/v2/guide/index.html.

[52] Railsware Products Inc, «Mailtrap,» 22 08 2017. [En línea]. Available: https://mailtrap.io/.

[53] Suárez, J. M. S., «Introducción a Spring Data: soporte para JPA.,» 14 08 2017. [En línea]. Available: https://www.adictosaltrabajo.com/2012/10/08/spring-data-jpa/.

[54] Kainulainen P., «Spring Data JPA Tutorial: Sorting,» 14 08 2017. [En línea]. Available: https://www.petrikainulainen.net/programming/spring-framework/spring-data-jpa-tutorial-part-six-sorting/.

[55] Pollack M., Risberg T., Gierke O., «Working with Spring Data Repositories,» 14 07 2017. [En línea]. Available: https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html.

[56] Gigsterous Team, «Spring Boot REST API (4) - Security with OAuth2,» 20 07 2017. [En línea]. Available: https://gigsterous.github.io/engineering/2017/03/01/spring-boot-4.html.

[57] Pivotal Software Inc, «Enabling Cross Origin Requests for a RESTful Web Service,» [En línea]. Available: https://spring.io/guides/gs/rest-service-cors/.

[58] Vue Examples, «Vue.js Examples,» 05 07 2018. [En línea]. Available: https://vuejsexamples.com/.

[59] Vue Toolbox, «Vue Toolbox,» 05 07 2018. [En línea]. Available: https://www.vuetoolbox.com/.

[60] Vuejs Team, «Awesome Vue,» 07 09 2018. [En línea]. Available: https://github.com/vuejs/awesome-vue.

[61] Winsemius R., «Vue-dropzone,» 17 02 2018. [En línea]. Available: https://github.com/rowanwins/vue-dropzone.

[62] Thibaud, «Authentication best practices for vue,» 03 09 2018. [En línea]. Available: https://blog.sqreen.com/authentication-best-practices-vue/.

[63] NightCatSama, «vue-slider-component,» 25 01 2019. [En línea]. Available: https://nightcatsama.github.io/vue-slider-component/#/basics/range.

[64] Oberlehner M., «Skeleton Loading Animation with Vue.js,» 15 02 2019. [En línea].

[65] Moura M., «Vue Material,» 10 08 2018. [En línea]. Available: https://vuematerial.io/getting-started.

[66] Saquete R., «Qué son y en qué consisten los Web Components Estás aquí:,» 27 07 2015. [En línea]. Available: https://www.humanlevel.com/articulos/desarrollo-web/que-son-y-en-que-consisten-los-web-components.html.

[67] Vuejs Team, «Guide to Vue router,» 15 07 2018. [En línea]. Available: https://router.vuejs.org/.

[68] Vuejs Team, «Guide to Vuex,» 02 08 2018. [En línea]. Available: https://vuex.vuejs.org/guide/.

[69] Kawaguchi K.,«Vue i18n,» 2018. [En línea]. Available: https://kazupon.github.io/vue-i18n/.

[70] Vuejs Team, «Guide to Vuecli,» 03 07 2018. [En línea]. Available: https://cli.vuejs.org/guide/.

[71] Kargopolov S., «Create a Deployable WAR File with Spring Boot,» 03 06 2019. [En línea]. Available: http://appsdeveloperblog.com/create-a-deployable-war-file-with-spring-boot/.

[72] Baeldung, «Sending Emails with Java,» 17 09 2017. [En línea]. Available: https://www.baeldung.com/java-email.

[73] SonarSource S.A, «SonarQube: Continuous Inspection,» 2008. [En línea]. Available: https://www.sonarqube.org/.

[74] SonarSource S.A, «SonarQube Documentation,» 2008. [En línea]. Available: https://docs.sonarqube.org/latest/.

[75] Beizier B., Testing and quality assurance, von Nostrand Reinhold, 1984.

[76] Pfleeger, S. L, Ingeniería del Software: Teoría y Práctica., Prentice Hall, 2002.

[77] Sommerville I., Software Engineering, Addison Wesley, 2005.

[78] Collard, J. F., Burnstein, I, Practical Software Testing: A Process-Oriented Approach, Springer, 2003.

[79] Google, «PageSpeed Insights - Google Developers,» 2013. [En línea]. Available: https://developers.google.com/speed/pagespeed/insights/?hl=es.

[80] TeamViewer US, Inc., «Monitis,» 2006. [En línea]. Available: http://www.monitis.com/pageload/.

[81] Stimulus Technology, «StresStimulus,» 2011. [En línea]. Available: https://www.stresstimulus.com/es.

[82] Böck M., «Building Skeleton Screens with CSS Custom Properties,» 15 02 2019. [En línea]. Available: https://css-tricks.com/building-skeleton-screens-css-custom-properties/.

[83] Chau G., «Detect when an element is becoming visible or hidden on the page,» 23 01 2018. [En línea]. Available: https://vuejsexamples.com/detect-when-an-element-is-becoming-visible-or-hidden-on-the-page/.

[84] Coolors, «The super fast color schemes generator!,» 08 09 2018. [En línea]. Available: https://coolors.co/5b0c20-931621-ce2d2d-9b7874-666370.

[85] Colorlib, «Free 404 error page templates,» 24 02 2019. [En línea]. Available: https://colorlib.com/wp/free-404-error-page-templates/.

[86] Unsplash, «Unsplash photos for everyone,» 03 07 2018. [En línea]. Available: https://unsplash.com/.

[87] Monty J., «Devicon,» 18 02 2019. [En línea]. Available: https://konpa.github.io/devicon/.

[88] Paraschiv E., «REST Query Language with Spring and JPA Criteria,» 23 09 2017. [En línea]. Available: https://www.baeldung.com/rest-search-language-spring-jpa-criteria.

[89] Paraschiv E., «REST Query Language with RSQL,» 23 09 2017. [En línea]. Available: https://www.baeldung.com/rest-api-search-language-rsql-fiql.

[90] Perplexhub, «rsql-jpa-specification,» 23 09 2017. [En línea]. Available: https://github.com/perplexhub/rsql-jpa-specification.

[91] Jirutka J., «Parser for RSQL / FIQL – query language for RESTful APIs,» 24 09 2017. [En línea]. Available: https://github.com/jirutka/rsql-parser.

[92] Cepr0, «How to easy implement 'REST API query language' with Querydsl and Spring Data,» 23 09 2017. [En línea]. Available: https://stackoverflow.com/questions/51127468/how-to-easy-implement-rest-api-query-language-with-querydsl-and-spring-data-to.

[93] «Spring Data Querydsl Value Operators,» 23 09 2017. [En línea]. Available: https://gt-tech.bitbucket.io/spring-data-querydsl-value-operators/README.html.

[94] Websystiqueadmin, «Secure Spring REST API using OAuth2,» 20 07 2017. [En línea]. Available: http://websystique.com/spring-security/secure-spring-rest-api-using-oauth2/.

[95] Baeldung, «Security with Spring,» 19 07 2017. [En línea]. Available: https://www.baeldung.com/security-spring.

[96] Spring, «spring-security-oauth,» 19 07 2017. [En línea]. Available: https://github.com/spring-projects/spring-security-oauth.

[97] Monteagudo J. L., «generator-spring-rest-jwt,» 21 07 2017. [En línea]. Available: https://github.com/jlmonteagudo/generator-spring-rest-jwt.

[98] Templier T., «How to use CORS in Restlet 2.3.1,» 28 03 2019. [En línea]. Available: https://stackoverflow.com/questions/28988671/how-to-use-cors-in-restlet-2-3-1.

11 Glosario

Backend: Aplicación que se ejecuta en un servidor y atiende peticiones otorgándoles una respuesta.

Frontend: Aplicación que se ejecuta en el navegador del usuario e interactúa con el.

Base de datos: Aplicación encargada de almacenar de forma persistente grandes cantidades de datos.

Modelo relacional: Diagrama que representa la estructura de la información a almacenar en una base de datos.

Actores: Tipos de usuarios que interactúan de forma diferente con la aplicación.

IDE (Integrated Drive Electronics): Es una aplicación informática con diversas herramientas para facilitar el desarrollo. Entre ellas se suele encontrar un editor de código, herramientas de construcción automática y un depurador.

Api Rest: Tipo de servidor que sigue un estándar concreto para la comunicación entre aplicaciones.

Web App (Aplicación Web): Aplicación informática que se ejecuta dentro de un navegador.

Progressive Web App (PWA): Tipo de aplicación web avanzada que utiliza diversas herramientas para equipararse en funcionalidad con aplicaciones móviles o de escritorio

Login (Autenticación): Proceso en el cual se identifica a un usuario y se le otorga acceso a una zona privada.

Web Component (Componente web): Fragmento de código web reutilizable en el cual está incluida la parte gráfica y su funcionalidad.

CORS (Cross Origin Resource Sharing): Es un estándar para el acceso de recursos web entre diferentes sitios web.

Caso de uso: Un caso de uso es la descripción de una acción o actividad.

Diagrama de casos de uso: es una descripción visual que deberá realizar alguien o algo para llevar a cabo algún proceso.

Refactorización: es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Bug: es un problema en un sistema software que desencadena un comportamiento indeseado.

Vulnerabilidad: es un punto débil en un sistema informático que permite que un atacante comprometa la integridad, disponibilidad o confidencialidad del mismo.

Code smell: es cualquier síntoma en el código que posiblemente indique un problema más profundo.

12 Índice de Ilustraciones

13 Índice de tablas