ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN WEB PARA LA VISUALIZACIÓN Y ANÁLISIS DE ÁRBOLES DE CONOCIMIENTOS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

STEVEN ANDRES CUASQUI PONCE

steven.cuasqui@epn.edu.ec

Director: PhD. EDISON LOZA AGUIRRE

edison.loza@epn.edu.ec

Quito, diciembre 2021

DECLARACIÓN DE AUTORÍA

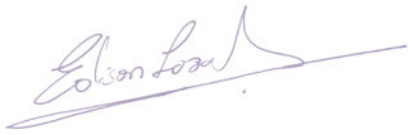
Yo, Steven Andres Cuasqui Ponce, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Steven Andres Cuasqui Ponce

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Steven Andres Cuasqui Ponce, bajo nuestra supervisión.



PhD. EDISON LOZA AGUIRRE
DIRECTOR DEL PROYECTO

PhD. ROSA NAVARRETE
CODIRECTORA DEL PROYECTO

DEDICATORIA

Este trabajo va dedicado a mi familia, Daniel, Mónica y Nelson, por su confianza y apoyo incondicional ante cualquier circunstancia que se nos presentó en el camino.

A mi mascota Toby, por su compañía hasta altas horas de la noche transmitiéndome tranquilidad.

A mis compañeros de la FIS, por ser confiables colegas dentro de las aulas y los mejores amigos fuera de ellas.

A mis profesores, siempre dispuestos a compartir su experiencia y conocimientos para mi desarrollo en el ámbito profesional.

A mi tutor Edison, por su compromiso con el presente trabajo de titulación.

AGRADECIMIENTOS

El presente trabajo fue posible gracias a la oportunidad que la Escuela Politécnica Nacional me brindó para formarme profesionalmente. Agradezco a todo el personal docente, administrativo y de apoyo de la Facultad de Ingeniería de Sistemas, en especial a Msc. Carlos Montenegro, PhD. José Lucio y PhD. Rosa Navarrete por su importante apoyo durante el período que tuve el cargo de presidente de los estudiantes de la FIS. Agradezco, mi tutor PhD. Edison Loza quién con su paciencia y experiencia supo brindarme guía a través de las dificultades encontradas en el desarrollo del presente proyecto.

Finalmente, agradezco a todas las personas con quienes alguna vez pude compartir una conversación. La suma de sus experiencias ayudó en la construcción del camino que me llevaron a este logro.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La gestión del conocimiento dentro de una organización es la manera en la cual se analiza las competencias del personal y como se relacionan estas entre sí. Los Árboles de Conocimiento (ADC) son herramientas muy útiles para este fin, ya que permiten mapear el conocimiento y representarlo de manera gráfica para analizar el estado del saber de la organización. En el presente trabajo, se desarrolla una herramienta web que permita la construcción automática de estas representaciones gráficas.

La aplicación web se construyó en 6 iteraciones utilizando el marco de trabajo SCRUM. Para la implementación se utilizó un stack MEAN para el desarrollo con MongoDB como base de datos no relacional, SailsJs en el servidor Back-End y Angular para el Front-End. Respondiendo a los requerimientos del presente trabajo, fue posible generar un algoritmo de renderizado de ADC que permite adaptar esta estrategia de mapeo de conocimiento para diferentes contextos y realizar un amplio análisis de las relaciones de los nodos del ADC. Además, la aplicación web se convierte en una de las pocas herramientas existentes en la actualidad para el mapeo y renderizado de ADC.

Palabras clave: Gestión del Conocimiento, Árboles de Conocimiento, renderizado web, aplicaciones web, SCRUM.

ABSTRACT

Knowledge management within an organization is how the competencies of the staff are analyzed and related to each other. Knowledge Trees (KT) are handy tools for this purpose because they allow mapping knowledge and graphically represent it as trees of nodes that help analyze the organization's state of knowledge. In this paper, a web application is developed to build this kind of representation automatically.

The web application was built in six sprints using the SCRUM framework. For its implementation, we used a MEAN stack for development with MongoDB as a non-relational database, SailsJs on the Back-End server and Angular for the Front-End. Responding to the requirements of the present work, an ADC rendering algorithm was developed, allowing adapting a knowledge mapping strategy for different contexts and carrying out a broad analysis of the relationships of the ADC nodes. Furthermore, the web application becomes one of the few tools available today for ADC mapping and rendering.

Keywords: Knowledge management, Knowledge trees, web rendering, web application, SCRUM.

CAPÍTULO 1: INTRODUCCIÓN

1.1. CONTEXTO

En la actualidad, hay quienes afirman que la competencia entre las organizaciones no se libra en el campo de los recursos sino más bien en el de los conocimientos [1]. Así, el saber adquirido mediante formación, capacitación, experiencia y/o autoaprendizaje ha transformado la manera en la que el personal es apreciado dentro de una organización. Con esto, se ha desmentido la creencia de que el individuo es fácilmente reemplazable dentro de su estructura organizacional [2]. De esta manera, la gestión de conocimiento se vuelve indispensable en los departamentos de Recursos Humanos y similares para estructurar y fortalecer las diferentes unidades, además de potenciar las competencias de la organización mediante la creación de grupos de trabajo con intereses similares o complementarios [3].

La gestión de conocimiento permite identificar fortalezas dentro de cualquier organización, a la vez que resalta las falencias o la poca satisfacción de necesidades de los grupos de análisis [4]. Los conocimientos pueden ser representados en estructuras que los mapeen, como los Árboles de Conocimiento (ADC). Estos árboles pueden presentar tanto el conocimiento existente como el requerido.

Una de las misiones del Departamento de Informática y Ciencias de la Computación (DICC) es contar con las competencias necesarias para satisfacer los requerimientos de las carreras de la Facultad de Ingeniería en Sistemas. Estos requerimientos, representados a manera de materias, pueden ser relacionados como nodos y mapeados como ADCs. Con esto, se podrán atacar las debilidades del departamento mediante planes de acción que busquen una corrección y prevención de amenazas para la gestión de las competencias del personal actual y potencial.

Dentro del sistema educativo, no existen casos de implementación de herramientas informáticas para la visualización y análisis de ADC en el reconocimiento y gestión de competencias del personal. Algunos productos tienen un acercamiento superficial a la gestión de conocimiento; como el software Badgr, que permite a los usuarios contar con un almacenamiento de sus emblemas, compartirlos e imprimirlos [5]. Para nombrar otro caso, es posible identificar la construcción de árboles de conocimiento de parte del equipo de Bresson, en representación de la Universidad de Reims, en Francia, que presenta un enfoque a futuro para una escuela primaria de la localidad [4]. Sin embargo,

este último trabajo utiliza para la construcción de ADC en un software de nombre GINGO, producto propietario cuyo último registro de utilización data del año 2003 y que ya no se encuentra disponible en el mercado actual [6].

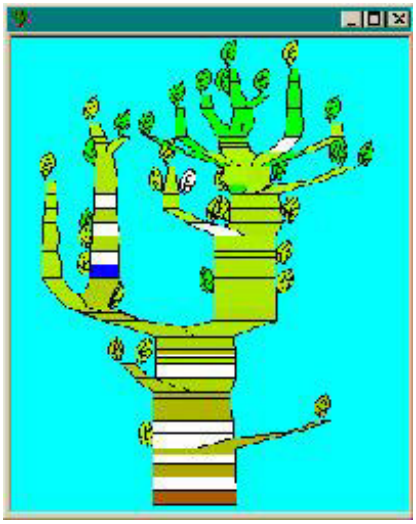


Figura 1. Árbol de conocimiento en GINGO [4].

La gestión del conocimiento, y en específico de los ADC, no cuenta actualmente con un producto de software que permita una visualización adecuada de los emblemas del personal en una organización. Actualmente, en la mayoría de las organizaciones, se opta simplemente por almacenar de manera documental registros y resúmenes sobre las competencias de los individuos, sin establecer relaciones ni interacciones entre el conocimiento del grupo [7]. Y eso, cuando se hace alguna actividad de Gestión de Conocimiento. Esta situación limita el reconocimiento del potencial de cada persona y de las falencias de la organización. Por tanto, el desarrollo de una aplicación que permita analizar y visualizar ADC se vuelve esencial para cualquier tarea de gestión de conocimientos en distintos contextos. En consecuencia, en el presente proyecto se propone el desarrollo de una aplicación web para la visualización y análisis de un ADC.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

- Desarrollar una aplicación web para la visualización y análisis de árboles de conocimientos.

1.2.2. OBJETIVOS ESPECÍFICOS

- Investigar literatura existente sobre gestión por competencias y la construcción de árboles de conocimiento.

- Implementar una herramienta gráfica que permita la visualización y análisis de árboles de conocimientos a través de una plataforma web.
- Crear un árbol de conocimientos en base a los emblemas identificados en el interior del DICC de la Facultad de Ingeniería de Sistemas de la EPN.

1.3. ALCANCE

El presente trabajo de titulación es desarrollado para el personal a cargo del DICC de la Facultad de Sistemas de la Escuela Politécnica Nacional. La propuesta consiste en el desarrollo de un algoritmo para generar un ADC basado que permita representar los requerimientos de materias dictadas y conocimientos de los profesores de la facultad. Además, junto con este algoritmo, se propone una aplicación web que se encargue del renderizado y gestión de los ADCs.

Para la realización de la aplicación web se consideran las secciones de Registro de Usuarios, Gestión de ADC, Renderizado de ADC, Gestión del Conocimiento del Árbol y Gestión de Profesores. El desarrollo de la aplicación se gestiona con la metodología ágil SCRUM para un prototipado en iteraciones incrementales. El trabajo se centra solamente en un ambiente local de desarrollo, con el motivo de focalizar la atención en el desarrollo del algoritmo y la construcción del Minimum Viable Product (MVP), y evitar actividades exhaustivas de administración de infraestructura.

Con el objetivo de probar la herramienta, y para contribuir a solucionar un requerimiento del DICC de la EPN, la aplicación será alimentada con la información de las materias que deben dictar los profesores del departamento. Este trabajo se implementará complementariamente con un proyecto de desarrollo que se está elaborando en la Maestría de Sistemas de Información, el cual se enfoca en el levantamiento de los conocimientos del DICC que complementará la herramienta aquí desarrollada.

1.4. MARCO TEÓRICO

1.4.1. DESARROLLO DE SOFTWARE ÁGIL

El desarrollo de software ágil es una respuesta al constante cambio de las necesidades del mercado actual. De acuerdo con el Manifiesto Ágil de Software, se deben priorizar a los individuos, software trabajando, colaboración con el cliente y la respuesta al cambio [8]. Estos cuatro criterios forman los valores centrales en los cuales se debe centrar el desarrollo de software para adaptarlo a la dinámica actual de la industria tecnológica.

1.4.2. SCRUM

SCRUM es un marco de trabajo que se alinea a los valores del Manifiesto Ágil definiendo eventos, artefactos y roles para entregar valor al desarrollo de software de manera incremental [9]. De esta manera, SCRUM es aplicable a proyectos de desarrollo de software de cualquier escala para aprovechar los beneficios que se obtienen al alinearse con el desarrollo de software ágil.

1.4.3. GESTIÓN DE CONOCIMIENTO

La gestión de conocimiento involucra actividades de generación, identificación y mapeo del capital intelectual dentro de una organización para fortalecer ventajas competitivas y transmitir información relevante a través de toda la institución. Esto permite refinar estrategias, políticas y prácticas para aprovechar conocimiento explícito y tácito [10].

El conocimiento explícito es aquel expresado de manera formal, fácil de describir y compartible a manera de especificaciones. Por otro lado, el conocimiento tácito es aquel informal con el que se encuentran dificultades para poder comunicarlo. Ambos empiezan en el individuo y pueden transmitirse de manera valiosa para el resto de la organización [11]. De esta manera, los dos tipos de saberes necesitan ser, de alguna manera, documentados para volverlos relevantes, generar innovación y aportar valor.

La identificación oportuna de nuevas potencialidades no solo le permite a una organización mejorar su productividad, sino ser más competente [12]. Es decir, que el conocimiento de su personal necesita ser gestionado de tal manera que se administren óptimamente las capacitaciones, entrenamientos y experiencia para fortalecer a la organización [2].

Los ADCs, desarrollados en los años noventa por Michel Authier y Pierre Levy, establecen principios que resuelven las problemáticas referentes a la gestión de conocimiento. Por ejemplo, los ADCs permiten identificar talentos “escondidos”, aparte de aquellos reconocidos de manera oficial por títulos o diplomas. Además, esta aproximación hace visible todo el espacio del saber de una organización, permitiendo transformar las condiciones de capacitación y la identidad cognitiva de los individuos [2].

Es posible relacionar conocimientos formales y no formales, además de representar de manera gráfica la abstracción de las relaciones entre los nodos de los ADCs [4]. Para esto, los “saberes” personales se representan como un “emblema” que, al ser

relacionado con los conocimientos de un grupo, puede ser organizado dentro de una de las diferentes maneras de mapear conocimientos como es un ADC [4].

Estos emblemas son una manera de identificar al conocimiento para caracterizarlo, validarlo y transformarlo en información [13]. Estos emblemas permiten encapsular la información referente a un conocimiento, de tal manera que contienen a los sujetos, su evidencia y el uso de su saber [4]. Se puede observar esto en la Figura 2.

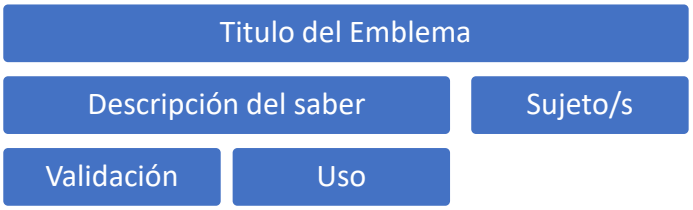


Figura 2. Contenido de un Emblema [4].

1.4.4. APLICACIÓN WEB

Las aplicaciones web son un tipo de sistema informático que utilizan los navegadores web a manera de cliente para comunicarse, utilizando el protocolo HTTP, con servidores web a través de una conexión de internet [14]. Actualmente, las aplicaciones web son sistemas con gran interoperabilidad debido a que los navegadores web pueden ser ejecutados en cualquier sistema operativo. De esta manera, el desarrollo de este tipo de sistemas permite poner a disposición la información para los usuarios utilizando cualquier tipo de dispositivo que cuente con acceso a internet.

1.4.5. BASE DE DATOS NO-SQL

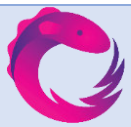
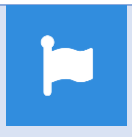
Las bases de datos no relaciones o No-SQL se caracterizan por no guardar los datos en tablas, sino que utilizan estructuras documentales del tipo llave - valor. Este tipo de estructuras son muy flexibles y facilitan la adaptabilidad a los diferentes modelos que pueden construirse en una aplicación web [15].

MongoDB es uno de los Database Management Systems (DBMS) más populares para la gestión de bases de datos no relacionales y utiliza documentos tipo JSON. De esta manera, el uso del formato JSON es muy útil para la construcción de servicios API REST, que son la convención más utilizada para poder comunicar las consultas del Front-End con las respuestas del Back-End [16].

1.5. HERRAMIENTAS UTILIZADAS

A continuación, se describen las herramientas utilizadas en el desarrollo a nivel de Front-End, Back-End y Uso General.

1.5.1. FRONT-END

Ítem	Descripción	Uso
 **Angular**	Angular es un framework de desarrollo basado en componentes para la construcción de aplicaciones web [17].	Utilizado para la implementación de componentes y su comunicación con el Back-End del programa.
 **JavaScript**	JavaScript es un lenguaje de scripting para páginas web que proporciona comportamiento a los sitios y es reconocido por los navegadores web [18].	Este lenguaje es implementado para proporcionar funcionalidad al sitio web tanto para el Front-End y Back-End.
 **TypeScript**	TypeScript es un lenguaje de programación construido sobre JavaScript que proporciona un completo sistema de tipado para agregar escalabilidad [19].	TypeScript se utiliza para aprovechar su robusto sistema de tipado y envolver las abstracciones del presente trabajo en estructuras de datos completas.
 **Babylon.js**	Babylon.js es un motor de renderizado web open-source escrito en TypeScript que permite crear escenarios y elementos en 3D [20].	El uso de este motor de renderizado 3D permite observar visualmente las abstracciones construidas de Árboles de Conocimiento.
 **RxJs**	RxJs es una librería de JavaScript para la composición y manejo de eventos asíncronos [21].	Esta librería permite gestionar los eventos asíncronos de datos con el Back-End.
 **Bootstrap**	Bootstrap es un framework de UI para la construcción de sitios web responsivos que contienen estilos y componentes personalizables potenciados con JavaScript [22].	Bootstrap es utilizado en la aplicación web para la construcción de componentes de UI de manera rápida y robusta.
	Font Awesome es una librería de íconos para su utilización en cualquier proyecto web o de	El conjunto de íconos de uso libre que Font Awesome proporciona son utilizados en la aplicación web

Ítem	Descripción	Uso
Font Awesome	escritorio con estilos y formatos determinados [23].	para complementar los diseños de los componentes del sitio.
 **CSS**	Cascading Style Sheets (CSS) es un lenguaje de estilo para determinar la manera en que los elementos HTML serán renderizados por el navegador web [24].	Las hojas de estilo permiten personalizar el renderizado de los componentes en el navegador web cuando el usuario interactúe con estos.
 **HTML**	HyperText Markup Language (HTML) es el lenguaje de marcado que determina la estructura y composición de las páginas web [25].	El HTML es utilizado para establecer la estructura general de la aplicación web y describir los vínculos entre los componentes.

Tabla 1. Herramientas Front-End.

1.5.2. BACK-END

Ítem	Descripción	Uso
 **SailsJs**	SailsJs es un marco de trabajo enfocado en el lado del Back-End, basado en el patrón MVC y que permite construir APIs de manera rápida para aplicaciones web de cualquier dimensión [26].	SailsJs es utilizado en el presente trabajo para la construcción del Back-End. Con este framework se establecerá la comunicación entre la Base de Datos y el Front-End de la aplicación web.
 **Express**	Express es un framework web que provee de varias características fundamentales para el funcionamiento de aplicaciones web y móviles, como métodos HTTP y enrutamiento [27].	Express se utiliza en la aplicación web para aprovechar las funciones HTTP que el framework tiene disponibles. Además, se usan sus características de enrutamiento y de control de errores de comunicación.
 **NodeJs**	NodeJs es un entorno para ejecutar JavaScript fuera del navegador web. Se orienta a eventos asíncronos y permite crear conexiones a través del protocolo HTTP [28].	En el presente trabajo se utiliza NodeJs para poder correr JavaScript como lenguaje del servidor web y soportar a todos los frameworks basados en JavaScript descritos anteriormente.

Ítem	Descripción	Uso
 **Docker**	Docker permite la creación de contenedores para controlar en un ambiente aislado partes específicas de una aplicación. Docker virtualiza el sistema operativo Linux para ejecutar aplicaciones en contenedores [29].	Docker es utilizado para virtualizar la Base de Datos en un contenedor. De esta manera es posible instalar la Base de Datos en un ambiente aislado y evitar problemas con actualizaciones o modificaciones en el sistema operativo nativo.
 **MongoDB**	MongoDB es una base de datos NoSQL open-source con gran flexibilidad que permite gestionar grandes volúmenes de datos a manera de documentos [30].	MongoDB es el sistema de Base de Datos utilizado para la aplicación web. Su formato documental permite tener flexibilidad al momento de determinar el modelo de los datos a gestionar.

Tabla 2. Herramientas Back-End.

1.5.3. USO GENERAL

Ítem	Descripción	Uso
 **VSCode**	VSCode es un poderoso editor de código de escritorio que cuenta con soporte para JavaScript y NodeJs [31].	El código del presente trabajo fue escrito completamente utilizando VSCode.
 **Git**	Git es un sistema de control de versiones open-source diseñado para poder ser utilizado en proyectos de cualquier dimensión [32].	Git es el sistema de control de versiones utilizado para todo el trabajo.
 **GitHub**	GitHub es un sistema que gestiona repositorios basados en Git [33].	El código fuente de toda la aplicación fue almacenado en un repositorio gestionado por GitHub.
 **Npm**	Npm es un gestor de paquetes para NodeJs [34].	Se utilizo npm para gestionar las librerías implementadas en el Front-End y Back-End de la aplicación web.

Tabla 3. Herramientas de Uso General.

CAPÍTULO 2: METODOLOGÍA

2.1. METODOLOGÍA

A continuación, se describen las consideraciones metodológicas utilizadas para el desarrollo del presente trabajo, junto con el proceso empleado para la realización del sistema.

2.1.1. METODOLOGÍA DE DESARROLLO

El desarrollo basado en SCRUM propone una división del trabajo en iteraciones incrementales. La utilización de esta metodología hace posible agregar funcionalidad crítica para el sistema muy rápido, y además se mejora la capacidad para reaccionar pronto a cualquier solicitud de cambios. De esta manera, es posible obtener un Minimum Viable Product (MVP) con características básicas para entregar valor al cliente de manera temprana [9].

Para el presente trabajo de titulación se consideran roles, artefactos y ceremonias reducidas debido a que el equipo de desarrollo es integrado solamente por una persona.

Los roles asignados para conformar el equipo SCRUM son de Product Owner, Scrum Master y Equipo de Desarrollo. El rol de Product Owner es asignado a PhD. Edison Loza, para representar al cliente y las necesidades que el producto debe cubrir. El rol de Scrum Master es asignado al autor del presente trabajo, siendo el encargado de gestionar el proyecto siguiendo la metodología de SCRUM. Además, el Equipo de Desarrollo se conforma también por el autor, quien es el encargado de la implementación de las Historias de Usuarios en el proyecto.

Los artefactos utilizados son el Product Backlog, que se describe como el conjunto de requerimientos obtenidos de las necesidades de los interesados; y el Sprint Backlog, que son el conjunto de requerimientos a ser desarrollados en el sprint en curso. Los requerimientos son especificados mediante Historias de Usuario con la finalidad de tener claro las funciones del sistema considerando el punto de vista del cliente y evitar documentación exhaustiva [8].

Los eventos considerados para el desarrollo del trabajo son el Sprint, con una duración aproximada de 3 semanas; la Planificación del Sprint, para estimar y establecer las Historias de Usuario que van a ser parte del Sprint; y la Revisión del Sprint, para aceptar

o rechazar las Historias de Usuario entregadas. El proceso definido se observa en la Figura 3.

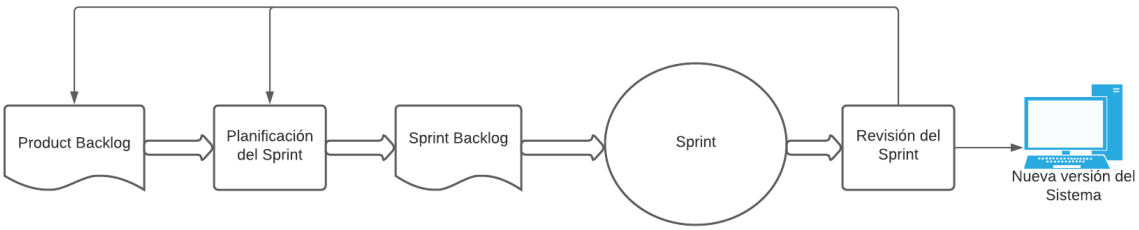


Figura 3. Ciclo de Scrum adaptado.

Cada Historia de Usuario se estima con puntos que representan el esfuerzo necesario para su desarrollo y que sigue los valores de la sucesión de Fibonacci. Además, se considera la utilización de Spikes, Historias de Usuario enfocadas en tareas de investigación que no entregan valor directo al producto, pero son necesarias para progresar con el desarrollo. De igual modo, se da lugar para incorporar Epics, Historias de Usuario de alto nivel, que no tienen criterios de aceptación bien definidos todavía y se parecen más a una idea general de una funcionalidad del sistema [9].

Tanto Historias de Usuario como Spikes se ubican como tarjetas en un tablero Scrum, utilizando la herramienta Trello (Figura 4), junto con la información solicitada por defecto (Figura 5) y que cuenta con las siguientes columnas:

- Product Backlog
- Sprint Backlog
- In Progress
- Finished

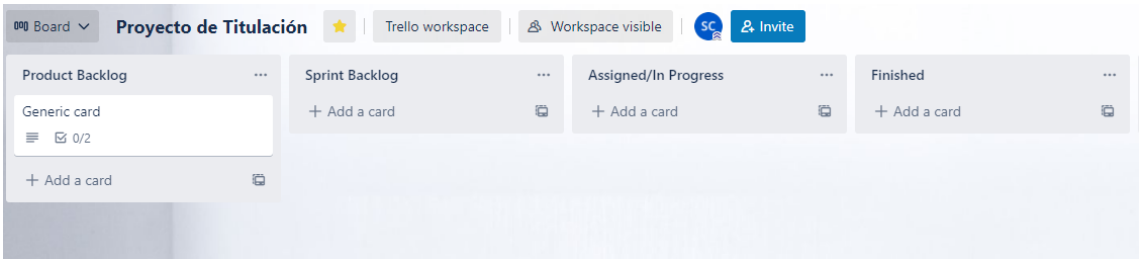


Figura 4. Tablero Scrum.

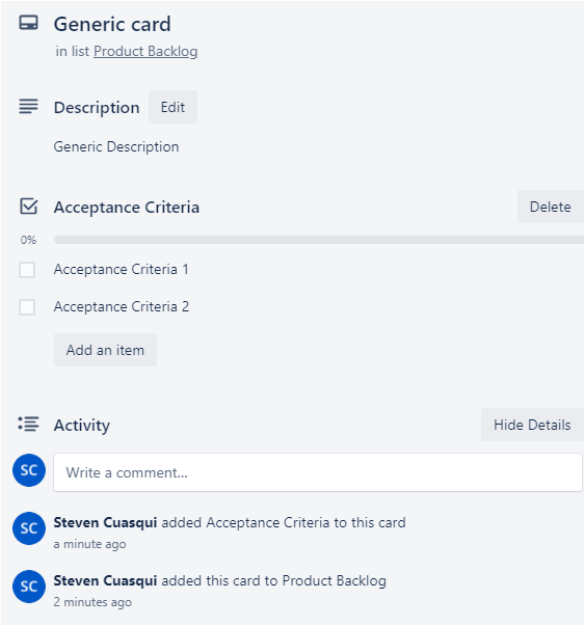


Figura 5. Modelo de Tarjeta.

2.1.2. FLUJO DE TRABAJO

El flujo de trabajo adoptado está basado en GitFlow con una rama maestra que sirve como versión funcional del producto y una rama de desarrollo que registra todos los commits a manera de historial [35] (Figura 6).

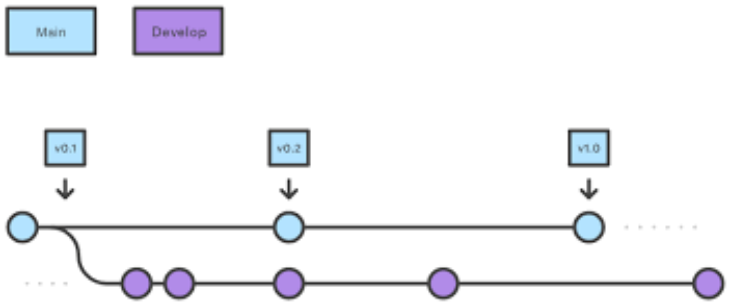


Figura 6. Flujo de Trabajo de GitFlow.

La rama máster es la encargada de tener el código de producción. Cualquier actualización o cambio que se requiera realizar en el código de producción se lo hace obedeciendo los siguientes pasos: primero, se crea una rama de desarrollo en la que se va a trabajar las Historias de Usuario; a continuación, se abren distintos pull requests contra la rama maestra con la identificación de las Historias de Usuario implementadas; y finalmente, cuando el pull request es aprobado, los cambios son combinados en la rama maestra y se cierra el pull request [35]. De esta manera, se mantiene una consistencia en el desarrollo de la aplicación con cada pull request representando una Historia de Usuario distinta y bien identificada.

2.2. PRODUCT BACKLOG

El Product Backlog se compone por una lista de requerimientos para el producto, generada por los interesados del proyecto y su función es ser la fuente que alimenta el trabajo que se va a realizar en todos los Sprints [9].

El Product Backlog se generó mediante sesiones virtuales de discusión con el Product Owner para determinar los requerimientos del sistema. Además, se estableció la posibilidad de revisitar los requerimientos si se requiere algún cambio o ajuste de estos.

En la Tabla 4 se detalla el Product Backlog inicial obtenido como resultado de las diferentes reuniones. Los requerimientos fueron agrupados de manera modular y con una calificación de prioridad baja, media o alta.

En cada Sprint se seleccionaron los requerimientos con la consideración de que pueden generarse nuevos, debido a la descripción de alto nivel y la falta de contexto al generar el Product Backlog inicial.

Código	Nombre	Descripción	Prioridad
Usuario			
US01	Registro Usuario	Como Usuario, requiero registrar mis datos para poder crear un usuario del sistema	Alta
US02	Inicio de sesión	Como Usuario, requiero iniciar sesión para ingresar al sistema	Alta
US03	Validación información de registro	Como Usuario, necesito saber si los datos ingresados en el registro son válidos	Media
US04	Validación información de inicio de sesión	Como Usuario, necesito saber si los datos ingresados en el inicio de sesión son válidos	Media
Gestión de Árboles de Conocimiento			
GA01	Creación de Árbol	Como Usuario, deseo crear un nuevo ADC para realizar la Gestión del Conocimiento	Alta
GA02	Modificación de Árbol	Como Usuario, necesito cambiar los datos del ADC para actualizar la información	Alta
GA03	Eliminar Árbol	Como Usuario, requiero eliminar el ADC para removar todos los nodos innecesarios	Alta
GA04	Listar Árboles	Como Usuario, deseo observar un listado de los Árboles de Conocimiento para proceder con su visualización como se desee	Alta
Gestión de Nodos			
GN01	Creación de Nodos	Como Usuario, deseo crear un nuevo nodo para representar un conocimiento	Alta
GN02	Eliminación de Nodos	Como Usuario, necesito eliminar los nodos para construir correctamente los Árboles de Conocimiento	Alta
GN03	Modificar Nodos	Como Usuario, requiero cambiar la información de los nodos para actualizar los conocimientos	Alta
GN04	Visualización de Nodos	Como Usuario, necesito ver la información de cada nodo para visualizar los detalles del conocimiento	Alta

Código	Nombre	Descripción	Prioridad
GN05	Relacionar Nodos	Como Usuario, deseo relacionar a los nodos para conectarlos a manera de árbol	Alta
GN06	Descarga de ADC	Como Usuario, deseo descargar el ADC como un archivo	Baja
Gestión de Profesores			
GP01	Asignar Profesores	Como Usuario, deseo relacionar profesores a un nodo para asignar profesores a los conocimientos	Alta
GP02	Eliminar Profesores	Como Usuario, deseo eliminar profesores para quitar las relaciones con los nodos que fueron asignados	Alta
GP03	Listar Profesores	Como Usuario, requiero observar los profesores relacionados con un nodo	Alta
GP04	Elección de Conocimiento	Como Usuario, necesito determinar el tipo de conocimiento de un profesor para diferenciar el conocimiento formal del informal	Alta
GP05	Evidencia de Conocimiento	Como Usuario, deseo registrar las evidencias del conocimiento para cada profesor	Alta
Visualización de Árbol de Conocimiento			
VA01	Render de Nodo	Como Usuario, necesito visualizar a un conocimiento a manera de nodo de un árbol	Alta
VA02	Relación de Nodos	Como Usuario, necesito visualizar la relación entre nodos para conectar los conocimientos	Alta
VA03	Render de Hoja	Como Usuario, necesito visualizar los nodos sin hijos a manera de hojas de un árbol	Alta
VA04	Render de Árbol	Como Usuario, necesito visualizar los nodos simulando un árbol en su render	Alta
VA05	Selección de Nodo	Como Usuario, necesito seleccionar un nodo para poder revisar los detalles de su contenido	Alta
VA06	Movimiento de Render	Como Usuario, necesito moverme por el árbol para ajustar mi foco de atención	Media

Tabla 4. Product Backlog inicial.

2.3. SPRINT 1

Para el primer Sprint, se decidió trabajar en los requerimientos de prioridad alta y media con bajo acoplamiento. Además, se consideró la opción para agregar historias con respecto a la preparación del ambiente de desarrollo.

2.3.1. PLANIFICACIÓN DEL SPRINT

Para el primer Sprint, se seleccionaron las Historias de Usuario que se consideran pueden ser completadas en un Sprint de aproximadamente 3 semanas. En la Figura 7 se puede observar el tablero al inicio del Sprint con las Historias de Usuario seleccionadas.

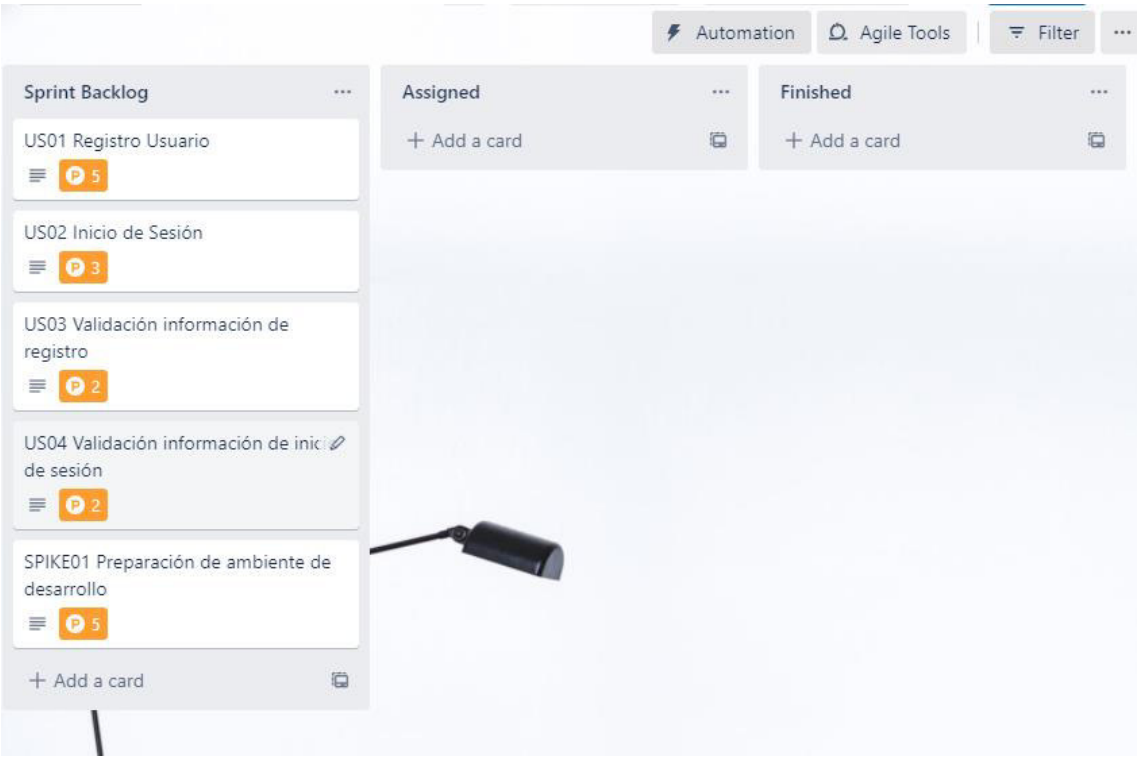


Figura 7. Tablero SCRUM del Sprint 1.

En este Sprint se añadió también el SPIKE01 para realizar el trabajo de configuración del ambiente de desarrollo para el Front-End, Back-End y Base de Datos. Es así como la lista de requerimientos definitiva para el Sprint 1 se indica en la Tabla 5.

Código	Nombre	Descripción	Prioridad
Usuario			
US01	Registro Usuario	Como Usuario, requiero registrar mis datos para poder crear un usuario del sistema	Alta
US02	Inicio de sesión	Como Usuario, requiero iniciar sesión para ingresar al sistema	Alta
US03	Validación información de registro	Como Usuario, necesito saber si los datos ingresados en el registro son válidos	Media
US04	Validación información de inicio de sesión	Como Usuario, necesito saber si los datos ingresados en el inicio de sesión son válidos	Media
SPIKE01	Preparación de ambiente de desarrollo	Como Desarrollador, necesito instalar las librerías necesarias para empezar el trabajo del proyecto	Alta

Tabla 5. Requerimientos para el Sprint 1.

2.3.2. HISTORIAS DE USUARIO

A continuación, se detallan los requerimientos del Sprint 1 a manera de Historias de Usuario incluyendo una estimación en puntos y criterios de aceptación, en las Tablas 6 a la Tabla 10.

US01 Registro Usuario	
Descripción:	Como Usuario, requiero registrar mis datos para poder crear un usuario del sistema
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema contará con un formulario de registro de usuarios en una pantalla.	
2. El sistema permitirá al usuario ingresar su nombre, apellido, correo, institución y contraseña.	

Tabla 6. Historia de Usuario US01.

US02 Inicio de Sesión	
Descripción:	Como Usuario, requiero iniciar sesión para ingresar al sistema
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El usuario puede iniciar sesión con su correo y contraseña registrada.	
2. El sistema mostrará un botón para proceder con el inicio de sesión	

Tabla 7. Historia de Usuario US02.

US03 Validación información de registro	
Descripción:	Como Usuario, necesito saber si los datos ingresados en el registro son válidos
Prioridad:	Media
Usuarios:	Usuario
Puntos:	2
Asignado:	Autor
Criterios de Aceptación	
1. El sistema debe alertarme cuando ingrese un nombre, apellido, correo, institución y/o contraseña inválidos.	
2. El sistema no me dejará enviar el formulario con datos inválidos.	
3. El sistema me indicará cuando el formulario de registro sea válido.	

Tabla 8. Historia de Usuario US03.

US04 Validación información de inicio de sesión	
Descripción:	Como Usuario, necesito saber si los datos ingresados en el inicio de sesión son válidos
Prioridad:	Media
Usuarios:	Usuario
Puntos:	2
Asignado:	Autor
Criterios de Aceptación	
1. El sistema debe alertarme cuando ingrese un correo y/o contraseña inválidos.	
2. El sistema alertará cuando las credenciales de inicio de sesión consultados sean inválidas.	
3. El sistema me indicará cuando el formulario de inicio de sesión sea válido.	

Tabla 9. Historia de Usuario US04.

SPIKE01 Preparación de ambiente de desarrollo	
Descripción:	Como Desarrollador, necesito instalar las librerías necesarias para empezar el trabajo del proyecto
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El Front-End está configurado con Angular y Bootstrap para el desarrollo del trabajo.	
2. El Back-End tiene instalado SailsJs y conectado con el Front-End para aceptar peticiones.	
3. El Back-End puede conectarse con la Base de Datos de MongoDB virtualizada en un contenedor de Docker.	
4. El sistema puede reflejar un cambio desde el Front-End y persistirlo en la Base de Datos.	

Tabla 10. Spike SPIKE01.

2.3.3. IMPLEMENTACIÓN

A continuación, se detalla la implementación de las Historias de Usuario del Sprint 1.

US01 Registro Usuario

Para el registro de usuarios, se implementó un formulario con los campos requeridos en la Historia de Usuario para que el usuario pueda ingresar los datos. Este formulario es una nueva ruta y es accesible a través de un botón “Registrarse” presente en la pantalla de inicio de la aplicación. En la Figura 8 y Figura 9 se puede observar la implementación del botón de registro y el formulario de registro respectivamente.

Figura 8. Botón de Registro e Inicio de Sesión.

Figura 9. Formulario de Registro de usuario.

US02 Inicio de Sesión

Para el inicio de sesión, se habilitó un botón “Ingresar” en la barra de navegación principal para ejecutar un modal que contiene un formulario en el cual el usuario ingresa las credenciales. En la Figura 10 se observa el modal de inicio de sesión.

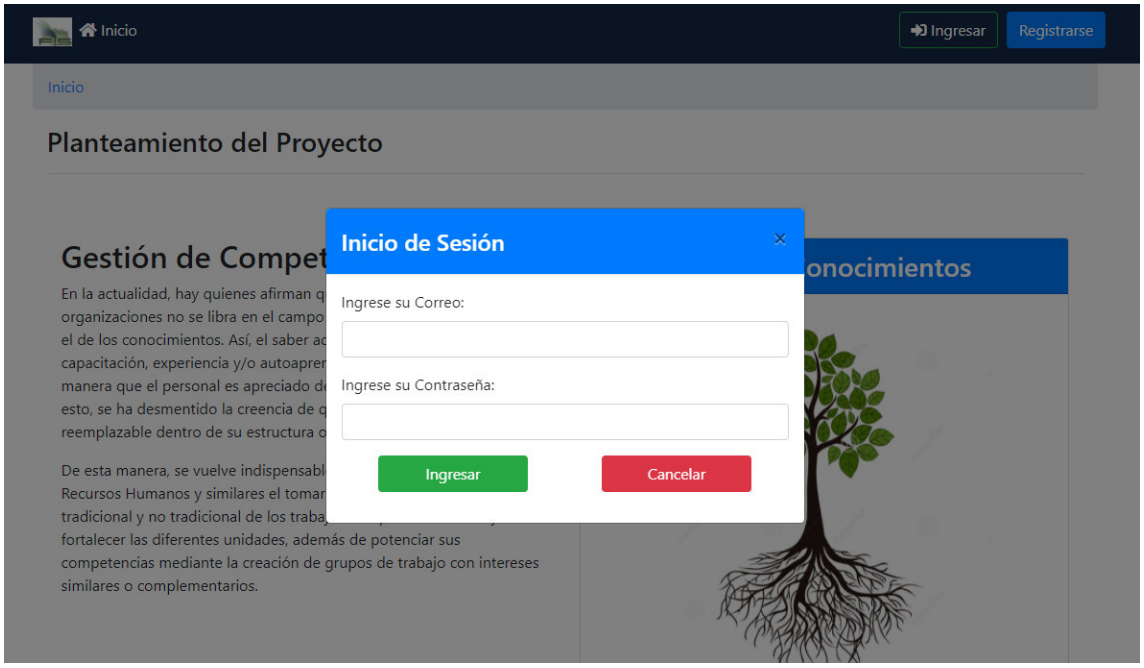


Figura 10. Modal de Inicio de Sesión.

US03 Validación información de registro

Para la validación del formulario de registro, se implementaron restricciones a nivel del documento HTML para establecer las condiciones cuando un campo es válido o inválido. Después, se detecta el estado del componente para mostrar los mensajes de error de acuerdo con el caso de validación. Un formulario válido es la única manera de habilitar el botón “Registrar” para crear el usuario.

En la Figura 11 se muestra un formulario inválido para los campos nombre, apellido, correo y contraseña, junto con los mensajes de error correspondientes. En la Figura 12 se observa un formulario válido con el botón “Registrar” habilitado para proceder con la creación del usuario.

Figura 11. Formulario de registro inválido.

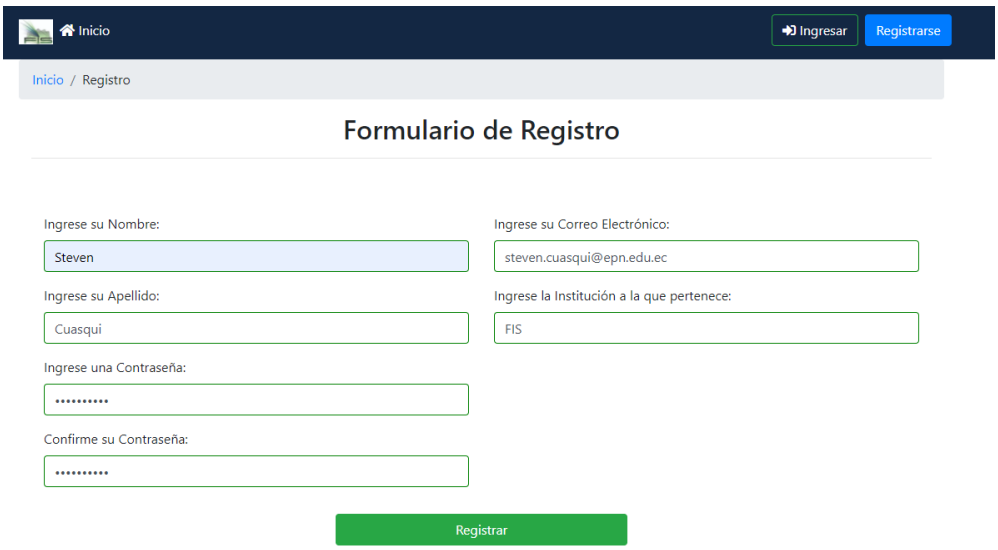


Figura 12. Formulario de registro válido.

US04 Validación información de inicio de sesión

Para realizar la validación de inicio de sesión, se procede de manera similar como con la historia US03, pero para el formulario correspondiente del modal. En la Figura 13 se observa un formulario inválido, mientras que en la Figura 14 se muestran los campos con entradas válidas.

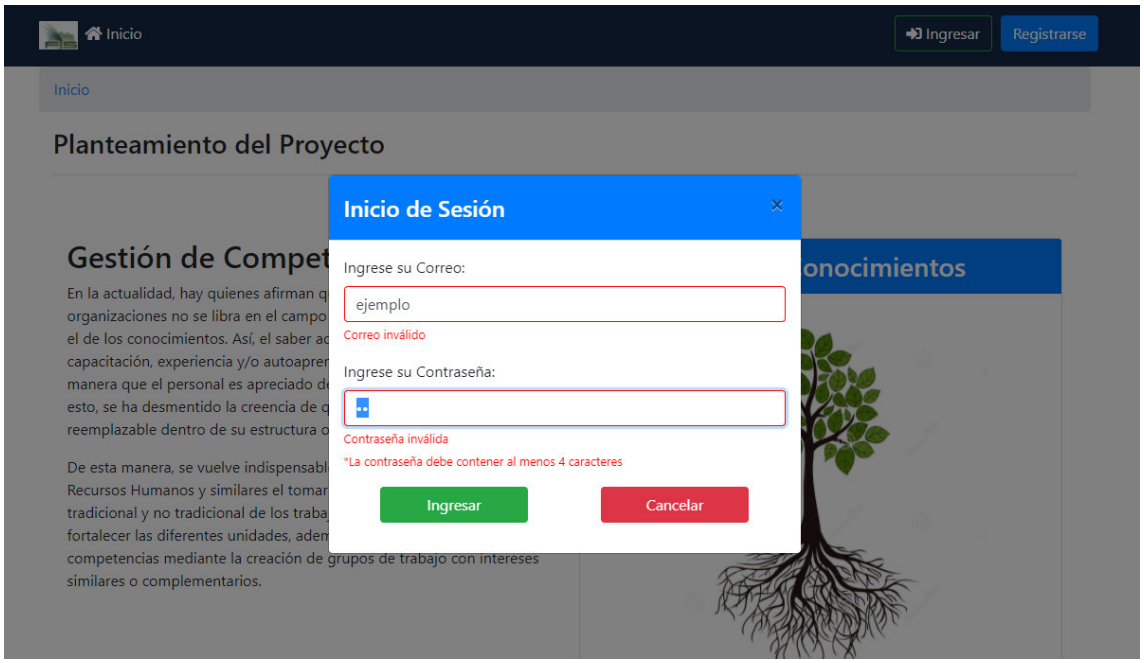


Figura 13. Formulario de inicio de sesión inválido.

Figura 14. Formulario de inicio de sesión válido.

SPIKE01 Preparación del ambiente de desarrollo

Para el Sprint 1, la implementación del Spike fue necesaria para la puesta a punto del ambiente de desarrollo del proyecto. El Front-End, Back-End y base de datos fueron generados junto con las librerías necesarias para su correcto funcionamiento.

En el Front-End, se generó un archivo *package.json* para ingresar las dependencias requeridas para esta parte del proyecto. Para el Back-End, se generó un archivo similar con sus respectivas dependencias. En el caso de la base de datos, se instaló el software Docker y se creó un contenedor con una imagen de MongoDB para ubicar la base documental. Además, se incluyó en la configuración del Back-End el puerto que Docker pone disponible para realizar conexiones con la base de datos.

En la Figura 15, se puede observar una lista de las librerías instaladas en el Front-End. En la Figura 16, se muestra la lista de dependencias en el Back-End. En la Figura 17, se observa el contenedor de Docker corriendo la imagen con la base de datos.

Figura 15. Lista de dependencias en el Front-End.

Figura 16. Lista de dependencias en el Back-End.

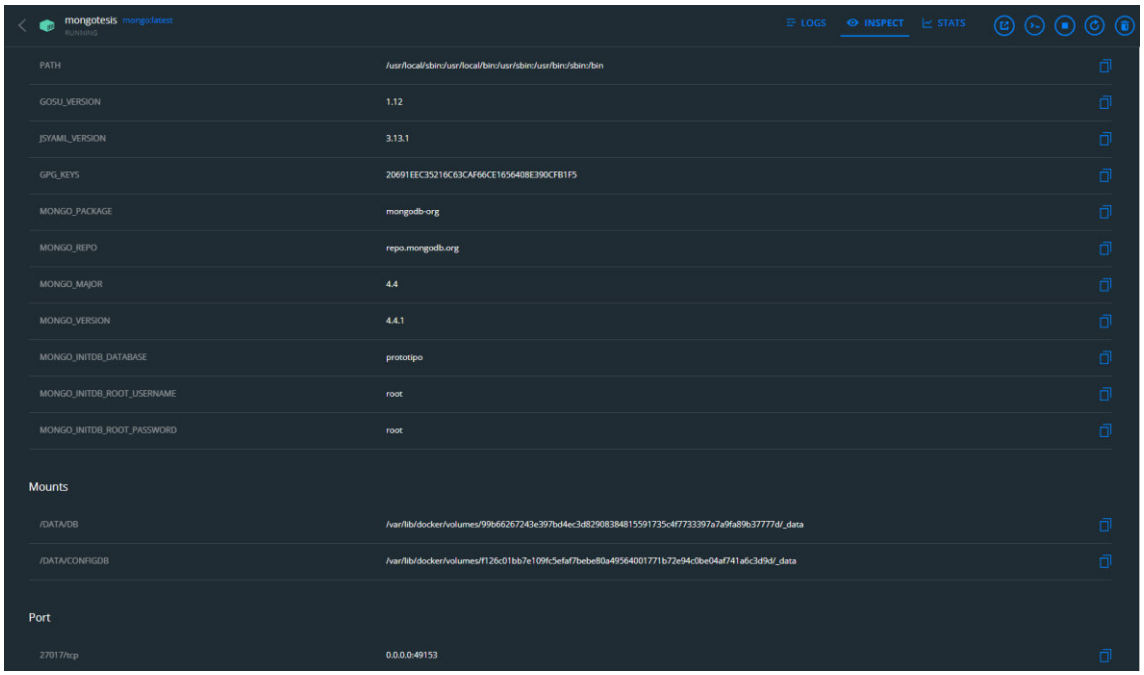


Figura 17. Contenedor de Docker con MongoDB.

2.3.4. REVISIÓN DEL SPRINT

Al finalizar el Sprint 1 se procede con la revisión del tablero Scrum para analizar la conformidad de las Historias de Usuario. En la Figura 18, se puede observar el tablero con todas las tarjetas terminadas.

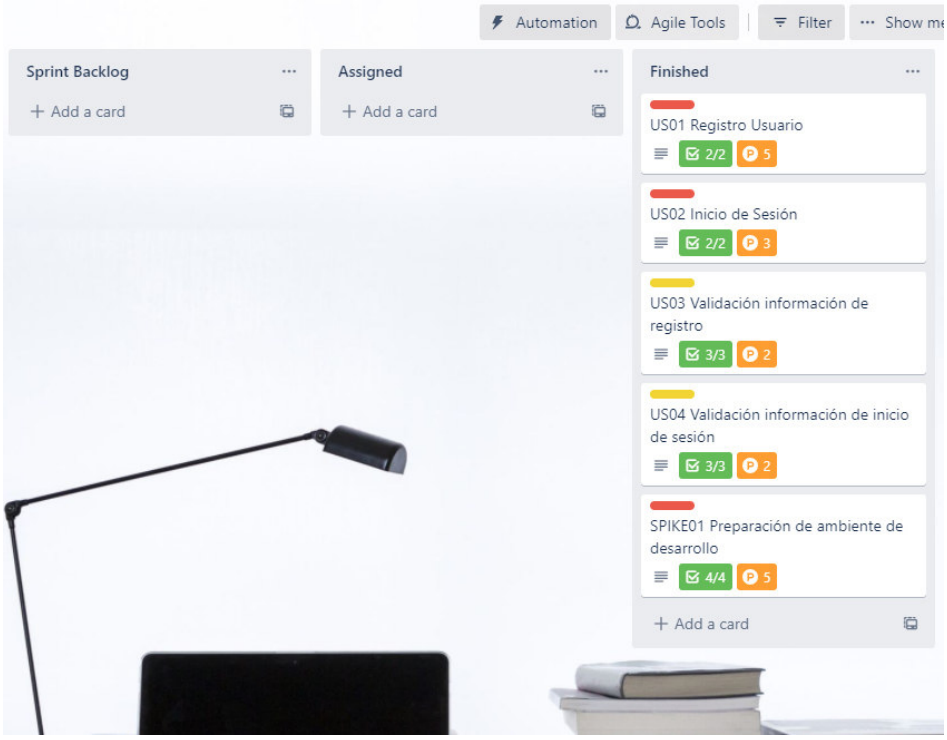


Figura 18. Tablero para revisión del Sprint 1.

Para el análisis del Sprint, se puede observar en la Tabla 11 un resumen de la implementación de las Historias de Usuarios basado en los puntos de esfuerzo completados en cada historia y su estado al finalizar el Sprint 1.

Código	Nombre	Prioridad	Puntos	Estado	Aceptación
US01	Registro Usuario	Alta	5	Terminada	Aceptada
US02	Inicio de sesión	Alta	3	Terminada	Aceptada
US03	Validación información de registro	Media	2	Terminada	Aceptada
US04	Validación información de inicio de sesión	Media	2	Terminada	Aceptada
SPIKE01	Preparación de ambiente de desarrollo	Alta	5	Terminada	Aceptada

Tabla 11. Revisión del Sprint 1.

Observaciones

Se observa que el Sprint pudo completarse sin mayor complicación con un total de 17 puntos de esfuerzo en las historias. Como resultado de la revisión con los interesados se genera un nuevo requerimiento para ser incorporado en el siguiente Sprint. En la Tabla 12, se observa el nuevo requerimiento US05 y su descripción.

Código	Nombre	Descripción	Prioridad
Usuario			
US05	Alerta de no existencia	Como Usuario, deseo saber si mis credenciales ingresadas en el inicio de sesión son correctas	Media

Tabla 12. Nuevos requerimientos del Sprint 1.

2.4. SPRINT 2

Para el Sprint 2, se decide mantener un ritmo de 17 puntos tomando en cuenta los resultados del Sprint 1 y la incorporación de un nuevo requisito. Es así como, se seleccionan requerimientos con prioridad alta y con bajo acoplamiento.

2.4.1. PLANIFICACIÓN DEL SPRINT

En la Figura 19 se observa el estado del tablero SCRUM al momento de dar inicio al Sprint 2.

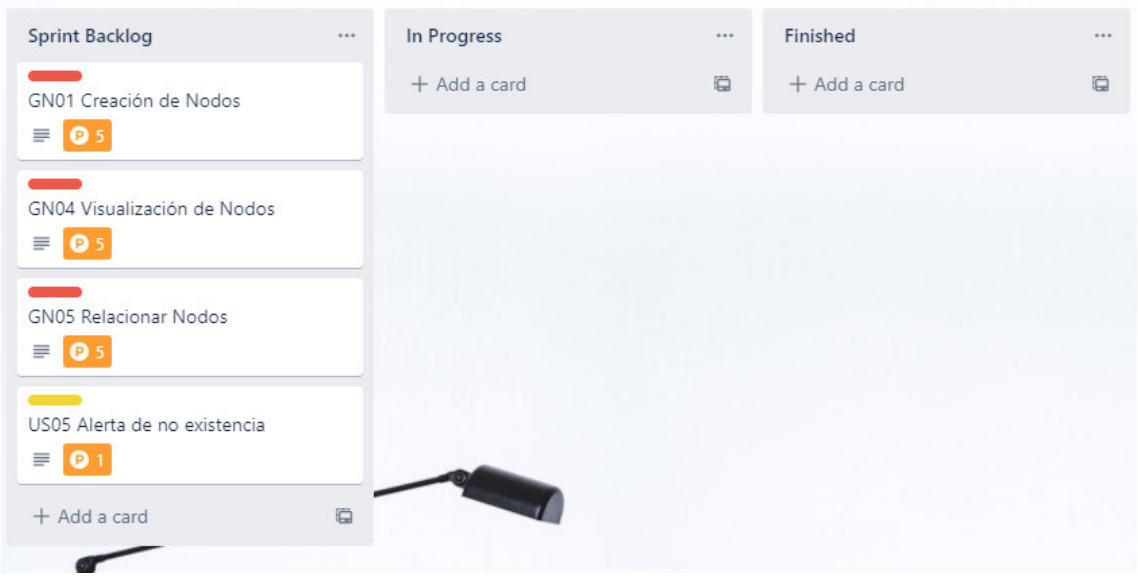


Figura 19. Tablero SCRUM del Sprint 2.

En el Sprint 2 se incorpora el requisito US05 que fue el resultado de la revisión del Sprint anterior. De esta manera, la lista de requisitos a satisfacer en el Sprint se observa en la Tabla 13.

Código	Nombre	Descripción	Prioridad
Gestión de Nodos			
GN01	Creación de Nodos	Como Usuario, deseo crear un nuevo nodo para representar un conocimiento	Alta
GN04	Visualización de Nodos	Como Usuario, necesito ver la información de cada nodo para visualizar los detalles del conocimiento	Alta
GN05	Relacionar Nodos	Como Usuario, deseo relacionar a los nodos para conectarlos a manera de árbol	Alta
Usuario			
US05	Alerta de no existencia	Como Usuario, deseo saber si mis credenciales ingresadas en el inicio de sesión son correctas	Media

Tabla 13. Requerimientos para el Sprint 2.

2.4.2. HISTORIAS DE USUARIO

A continuación, se representan los requerimientos para el Sprint 2 como Historias de Usuario. Se pueden observar desde la Tabla 14 hasta la Tabla 17.

GN01 Creación de Nodos	
Descripción:	Como Usuario, deseo crear un nuevo nodo para representar un conocimiento
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor

Criterios de Aceptación
1. El sistema debe permitir la creación de un nuevo nodo en el ADC deseado.
2. El nodo creado debe ser renderizado a manera de un cubo.
3. El nodo creado debe representar un conocimiento con atributos identificador, número de profesores e índice de paternidad.

Tabla 14. Historia de Usuario GN01

GN04 Visualización de Nodos	
Descripción:	Como Usuario, necesito ver la información de cada nodo para visualizar los detalles del conocimiento
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá seleccionar un nodo específico para visualizar sus datos.	
2. El sistema permitirá visualizar los atributos identificador, número de profesores e índice de paternidad del nodo seleccionado.	

Tabla 15. Historia de Usuario GN04.

GN05 Relacionar Nodos	
Descripción:	Como Usuario, deseo relacionar a los nodos para conectarlos a manera de árbol
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá diferenciar un nodo padre de un nodo hijo.	
2. El sistema permitirá crear un nodo hijo al seleccionar un nodo específico.	
3. El sistema debe mostrar la información del nodo padre cuando se selecciona un nodo.	

Tabla 16. Historia de Usuario GN05.

US05 Alerta de no existencia	
Descripción:	Como Usuario, deseo saber si mis credenciales ingresadas en el inicio de sesión son correctas
Prioridad:	Media
Usuarios:	Usuario
Puntos:	1
Asignado:	Autor
Criterios de Aceptación	
1. El sistema alertará al usuario cuando las credenciales de inicio de sesión sean correctas.	
2. El sistema notificará al usuario cuando las credenciales de inicio de sesión sean incorrectas.	

Tabla 17. Historia de Usuario US05.

2.4.3. IMPLEMENTACIÓN

A continuación, se muestra la implementación de cada Historia de Usuario del Sprint 2.

GN04 Visualización de Nodos

Para la visualización de los nodos del ADC, se genera un componente exclusivo para la renderización de una escena. Esta escena es construida con BabylonJs y su motor de renderizado integrado. El componente genera la escena desde el principio de su ciclo de vida, y procede a alterarla para agregar, quitar o modificar objetos a través de funciones que quedan esperando por una acción del usuario.

Para revisar los detalles de un nodo, se implementa la acción de hacer clic sobre el objeto, se cambia el color del nodo seleccionado y se procede a mostrar la información de sus atributos en el componente de “Información detallada”.

En la Figura 20 se muestra el componente de renderización con dos nodos de prueba y la información correspondiente en el componente de información. En la Figura 21 se muestra el componente “Información detallada” con los datos del nodo seleccionado.

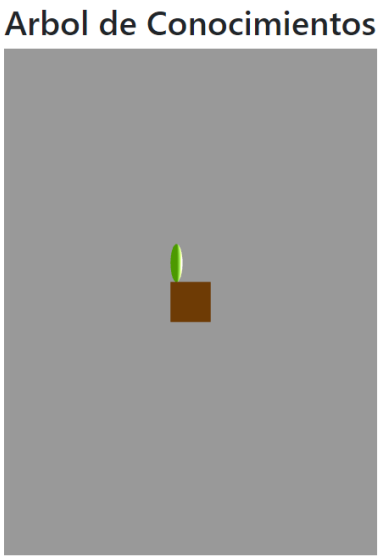


Figura 20. Componente de renderizado de nodos.

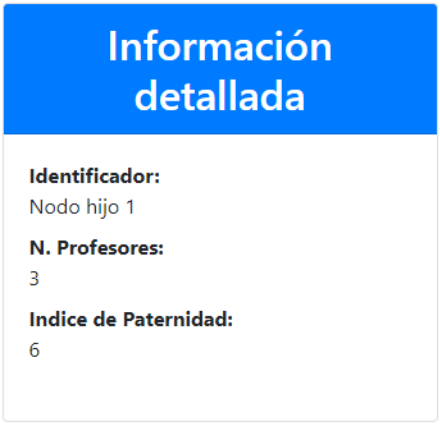


Figura 21. Información detallada de un nodo seleccionado.

GN05 Relacionar Nodos

Para crear la relación padre-hijo entre los nodos del árbol, se procede a agregar en cada objeto una función de escucha para el momento en que el usuario procede a hacer clic sobre un nodo. Entonces, se activa un menú fuera del componente de visualización del ADC que tiene la opción “Añadir”. De esta manera, el nodo seleccionado toma el rol de nodo padre y el nodo creado se añade a su correspondiente lista “nodosHijo”, que es una relación establecida en el modelo Nodo ubicado en el Back-End.

En la Figura 22 se observa el modelo Nodo con el atributo “nodosHijo” para soportar la relación padre-hijo con varios nodos.

Figura 22. Relación nodo padre-hijo.

GN01 Creación de Nodos

Para la creación de nodos, es necesario habilitar un botón “Añadir” ubicado en un nuevo menú de nodo. Este botón muestra un formulario para crear un nuevo nodo solicitando los datos de identificador, número de profesores y el índice de paternidad. Una vez que se han ingresado los datos del nodo a crear, se debe observar un objeto renderizado en el componente de visualización del ADC.

En la Figura 23 se puede observar el botón para la creación de un nodo. En la Figura 24 se muestra el formulario para la creación del nodo. En la Figura 25 se observa el nodo renderizado en el componente de visualización.

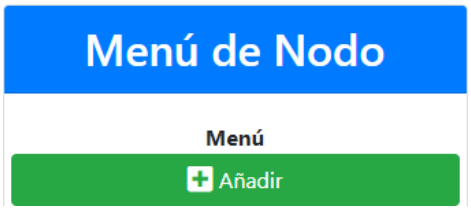


Figura 23. Botón de creación de nodo

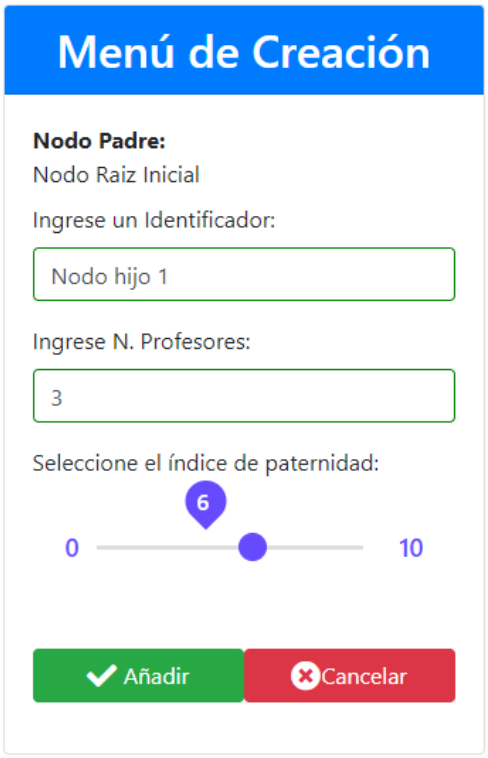


Figura 24. Formulario de creación de nodo.

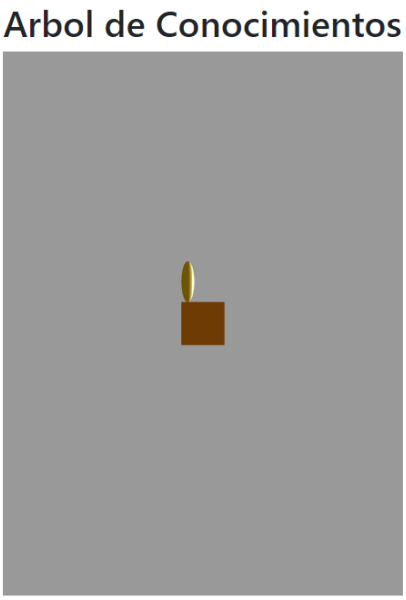


Figura 25. Nodo renderizado.

US05 Alerta de no existencia

Para alertar de la no existencia de un usuario con las credenciales proporcionadas, se procede a evaluar la consulta realizada por el Front-End al Back-End que busca recuperar un usuario existente con las credenciales ingresadas en el formulario de inicio de sesión. Si el usuario no existe, se procede a lanzar una alerta en el navegador con el mensaje “Credenciales incorrectas”.

En la Figura 26, se observa la alerta cuando las credenciales son incorrectas en el formulario de inicio de sesión. En la Figura 27, se muestra una alerta con un mensaje de éxito para un inicio de sesión correcto.

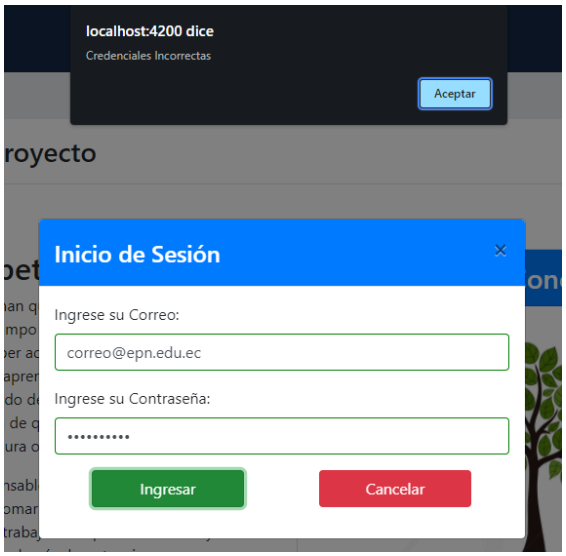


Figura 26. Alerta con inicio de sesión fallido.

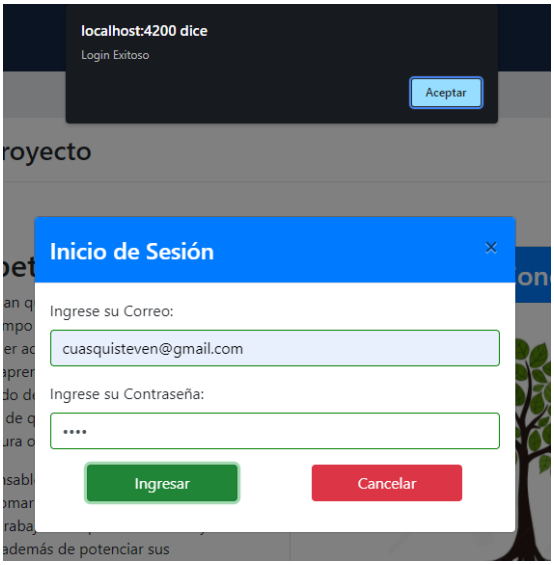


Figura 27. Alerta con inicio de sesión exitoso.

2.4.4. REVISIÓN DEL SPRINT

Al terminar el Sprint 2, se procede con la revisión del tablero Scrum para analizar la conformidad de las Historias de Usuario. En la Figura 28, se observa el tablero con todas las tarjetas terminadas.

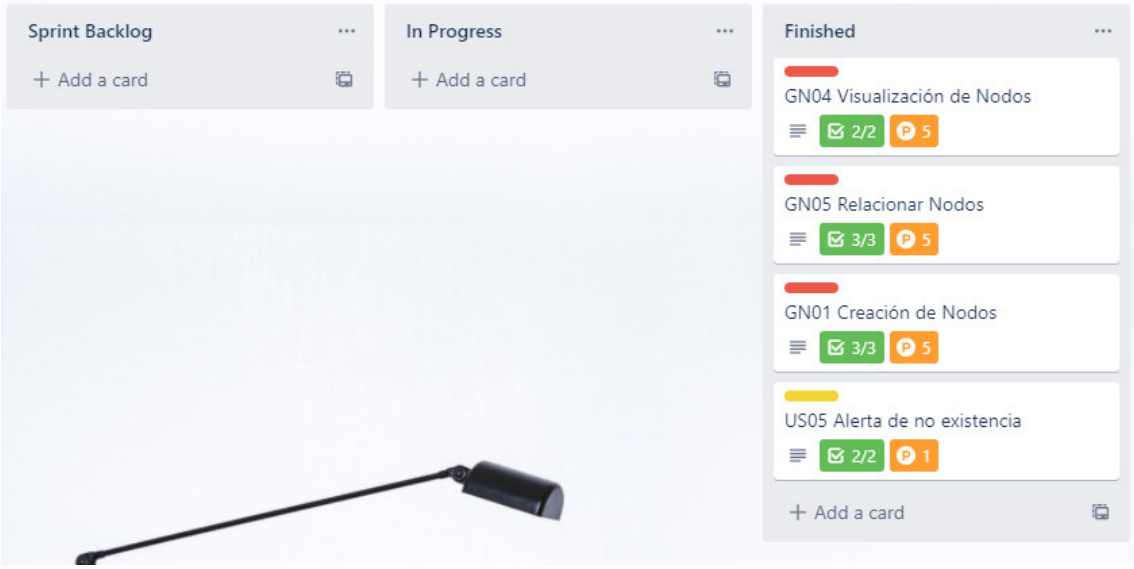


Figura 28. Tablero para revisión del Sprint 2.

Para analizar el Sprint 2, se puede observar en la Tabla 18 un resumen de la implementación de las Historias de Usuarios.

Código	Nombre	Prioridad	Puntos	Estado	Aceptación
GN04	Visualización de Nodos	Alta	5	Terminada	Aceptada

GN05	Relacionar Nodos	Alta	5	Terminada	Aceptada
GN01	Creación de Nodos	Alta	5	Terminada	Aceptada
US05	Alerta de no existencia	Media	1	Terminada	Aceptada

Tabla 18. Revisión del Sprint 2.

Observaciones

Para el Sprint 2, se tuvo que reordenar las Historias de Usuario para trabajar antes en aquellas que bloqueaban a la consecución de otras. El orden en que las historias fueron implementadas es el que se observa en la Tabla 18. No se generaron nuevas Historias de Usuario en el Sprint 2.

2.5. SPRINT 3

Para el Sprint 3, se decide mantener un ritmo de 17 puntos tomando en cuenta los resultados del Sprint 2. Es así como, se seleccionan requerimientos con prioridad alta y media-baja que entregan más valor al sistema.

2.5.1. PLANIFICACIÓN DEL SPRINT

En la Figura 29, se puede observar el tablero al inicio del Sprint 3 con las Historias de Usuario seleccionadas.

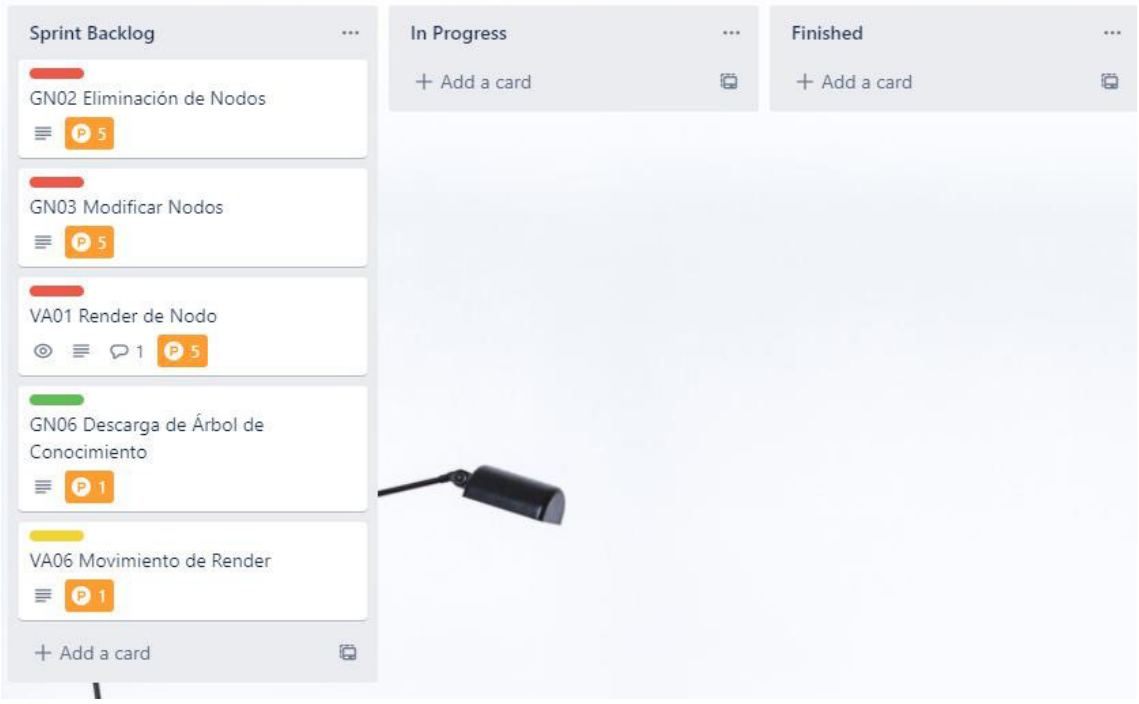


Figura 29. Tablero SCRUM del Sprint 3

En la Tabla 19, se listan los requerimientos a desarrollar en el Sprint 3.

Código	Nombre	Descripción	Prioridad
Gestión de Nodos			
GN02	Eliminación de Nodos	Como Usuario, necesito eliminar los nodos para construir correctamente los Árboles de Conocimiento	Alta
GN03	Modificar Nodos	Como Usuario, requiero cambiar la información de los nodos para actualizar los conocimientos	Alta
GN06	Descarga de ADC	Como Usuario, deseo descargar el ADC como un archivo	Baja
Visualización de ADC			
VA01	Render de Nodo	Como Usuario, necesito visualizar a un conocimiento a manera de nodo de un árbol	Alta
VA06	Movimiento de Render	Como Usuario, necesito moverme por el árbol para ajustar mi foco de atención	Media

Tabla 19. Requerimientos para el Sprint 3.

2.5.2. HISTORIAS DE USUARIO

A continuación, se presentan los requerimientos para el Sprint 3 como Historias de Usuario. Se pueden observar desde la Tabla 20 hasta la Tabla 24.

GN02 Eliminación de Nodos	
Descripción:	Como Usuario, necesito eliminar los nodos para construir correctamente los Árboles de Conocimiento
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá seleccionar el nodo específico para removerlo.	
2. El sistema no debe permitir que se elimine un nodo que tenga hijos.	
3. El sistema solamente debe mostrar la opción de eliminar nodo para nodos que no tienen hijos.	

Tabla 20. Historia de Usuario GN02.

GN03 Modificar Nodos	
Descripción:	Como Usuario, requiero cambiar la información de los nodos para actualizar los conocimientos
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá seleccionar el nodo específico para modificar.	
2. El sistema debe dar un indicio de los valores previos a la modificación.	

3. El sistema actualiza la información inmediatamente después de ocurrida la modificación de un nodo.

Tabla 21. Historia de Usuario GN03.

GN06 Descarga de ADC	
Descripción:	Como Usuario, deseo descargar el ADC como un archivo
Prioridad:	Baja
Usuarios:	Usuario
Puntos:	1
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá descargar un archivo en formato JSON con la información actual de los nodos del ADC.	

Tabla 22. Historia de Usuario GN06

VA01 Render de Nodo	
Descripción:	Como Usuario, necesito visualizar a un conocimiento a manera de nodo de un árbol
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema mostrará un cubo cuando el nodo sea central en el ADC.	
2. El sistema debe mostrar trapecios inclinados para representar ramas abriéndose en el ADC.	
3. Los nodos en forma de trapecio deben poder abrirse a la izquierda y a la derecha.	

Tabla 23. Historia de Usuario VA01.

VA06 Movimiento de Render	
Descripción:	Como Usuario, necesito moverme por el árbol para ajustar mi foco de atención
Prioridad:	Media
Usuarios:	Usuario
Puntos:	1
Asignado:	Autor
Criterios de Aceptación	
1. El componente de visualización del ADC debe permitirme acercarme y alejarme del árbol.	
2. El componente de visualización del ADC debe permitirme mover la perspectiva de la cámara.	

Tabla 24. Historia de Usuario VA06.

2.5.3. IMPLEMENTACIÓN

A continuación, se muestra la implementación de cada Historia de Usuario del Sprint 3.

GN02 Eliminación de Nodos

Para eliminar nodos del ADC, se debe considerar la condición de que no se puede eliminar un nodo si este tiene nodos hijo. Para esto, es necesario implementar un control para revisar la longitud del atributo “nodosHijo” del nodo a remover. Si al evaluar este atributo se obtiene como resultado una longitud de cero se puede proceder con renderizar un botón “Eliminar” en el menú del nodo. Caso contrario, se evita mostrar el botón en cuestión.

En la Figura 30, se observa la opción de eliminar un nodo cuando este no tiene hijos. Mientras que, en la Figura 31 se observa que si el nodo tiene al menos un hijo el botón no se mostrará.

Figura 30. Nodo con opción “Eliminar”.

Figura 31. Nodo sin opción “Eliminar”.

GN03 Modificar Nodos

La modificación de un nodo tiene una consideración importante, es necesario actualizar el ADC una vez que se ha editado uno de sus nodos. Para lograr este comportamiento es necesario implementar una función asíncrona que actualice el estado del árbol cuando se ejecute la petición de actualización de un nodo. Esto se logró utilizando el recurso de Observables, de la librería RxJs.

En la Figura 32, se puede observar la opción de “Editar” para el nodo seleccionado. Además, se muestra el formulario de edición requerido para actualizar los datos del nodo.

Figura 32. Formulario de edición de un nodo seleccionado.

VA01 Render de Nodo

Para que el renderizado de los nodos logre simular un árbol, es necesario considerar una representación especial para las ramas. Para lograr esto, se recurre a representar estos nodos como un polígono trapezoidal, como se puede observar en la Figura 33. Esta figura no altera el volumen total de un nodo ya que se considera la idea de que al descomponer un cubo en dos partes iguales por su diagonal y después al acomodar las partes rotando la cara de una de estas, se llega a este trapecio regular.

La implementación de este tipo de figura requiere de un análisis de las variables requeridas para lograr la construcción adecuada. Entonces, se establece que las variables necesarias para construir la figura son un ángulo de inclinación y una dimensión de altura o de ancho. De esta manera, es posible renderizar la figura haciendo uso de operaciones relacionadas a la geometría espacial. En la Figura 33, se observa el renderizado de nodos rama implementando esta nueva figura.

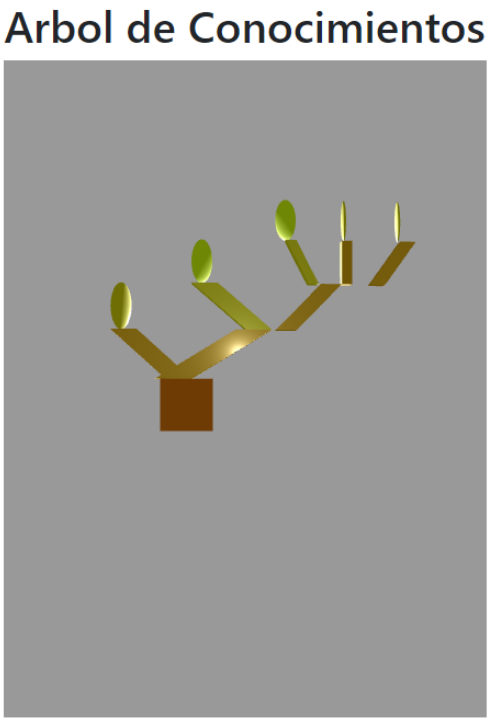


Figura 33. Renderizado de trapecios para ramas.

GN06 Descarga de ADC

Para realizar la descarga del ADC, se aprovechó que la estructura de los documentos en que estos son almacenados es del tipo JSON. Debido a esto, se cuenta con una armonía al momento de implementar la lógica de descarga; ya que, JavaScript utiliza objetos JSON para la mayoría de sus interacciones. De esta manera, al tener el ADC en este formato desde la base de datos, solamente se requiere una conversión simple desde un objeto a cadena de texto. Finalmente, esta cadena de texto es entregada cuando se hace clic en el botón “Descargar arbol.json”.

En la Figura 34, se observa el resultado de accionar el botón “Descargar arbol.json” presente en el componente principal del ADC presentado en la Figura 30.

Figura 34. Archivo JSON del ADC.

VA06 Movimiento de Render

Para este requerimiento solamente se necesita de agregar un atributo del tipo Camera en la clase que define una escena. En consecuencia, se habilita la navegación en primera persona para la escena renderizada. En la Figura 35, se observa un acercamiento al ADC de la Figura 30.

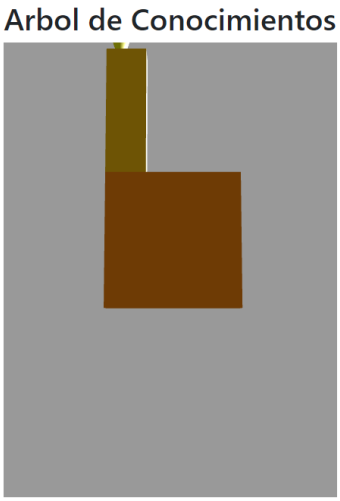


Figura 35. Acercamiento en la navegación de la escena.

2.5.4. REVISIÓN DEL SPRINT

Al terminar el Sprint 3, se procede con la revisión del tablero Scrum para analizar la conformidad de las Historias de Usuario. En la Figura 36, se observa el tablero con todas las tarjetas terminadas.

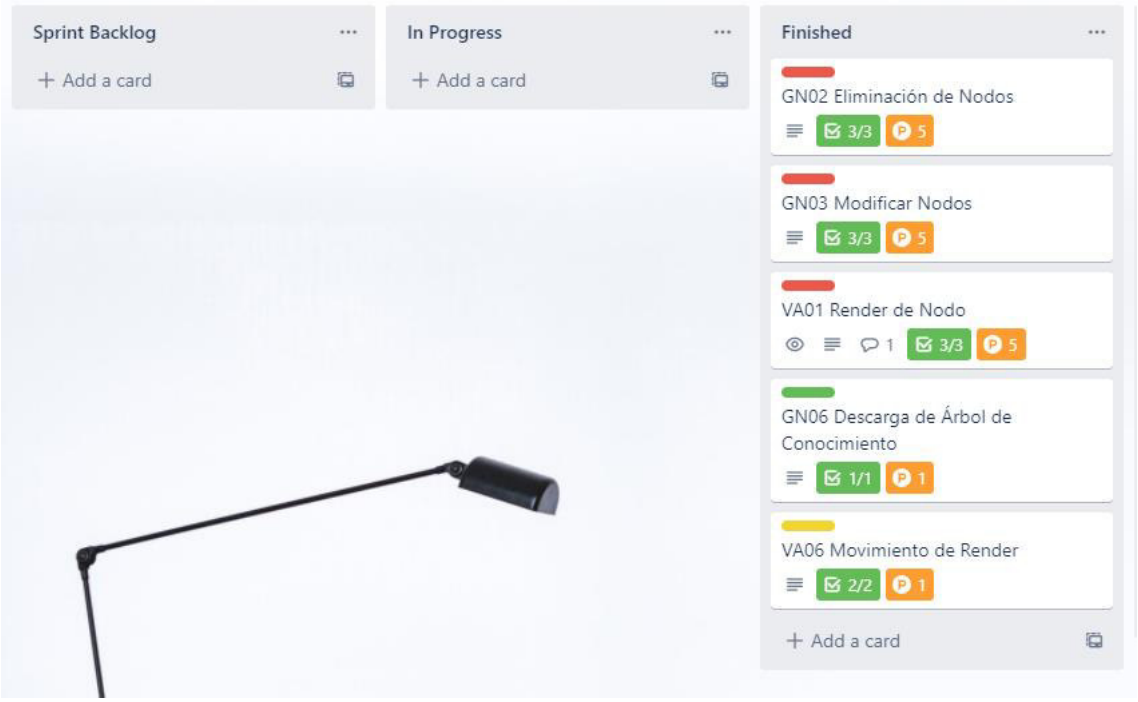


Figura 36. Tablero para revisión del Sprint 3

Para analizar el Sprint 3, se puede observar en la Tabla 25 un resumen de la implementación de las Historias de Usuarios.

Código	Nombre	Prioridad	Puntos	Estado	Aceptación
GN02	Eliminación de Nodos	Alta	5	Terminada	Aceptada
GN03	Modificar Nodos	Alta	5	Terminada	Aceptada
VA01	Render de Nodo	Alta	5	Terminada	Aceptada
GN06	Descarga de ADC	Baja	1	Terminada	Aceptada
VA06	Movimiento de Render	Media	1	Terminada	Aceptada

Tabla 25. Revisión del Sprint 3.

Observaciones

Para el Sprint 3 no se cuenta con nuevos requisitos y se pudieron completar las Historias de Usuario correspondientes sin la necesidad de trasladarlas al siguiente Sprint.

2.6. SPRINT 4

Para el Sprint 4, se decide mantener un ritmo de 17 puntos tomando en cuenta los resultados del Sprint 3. Es así como, se seleccionan requerimientos con prioridad alta.

2.6.1. PLANIFICACIÓN DEL SPRINT

En la Figura 37 se observa el estado del tablero SCRUM al momento de dar inicio al Sprint 4.

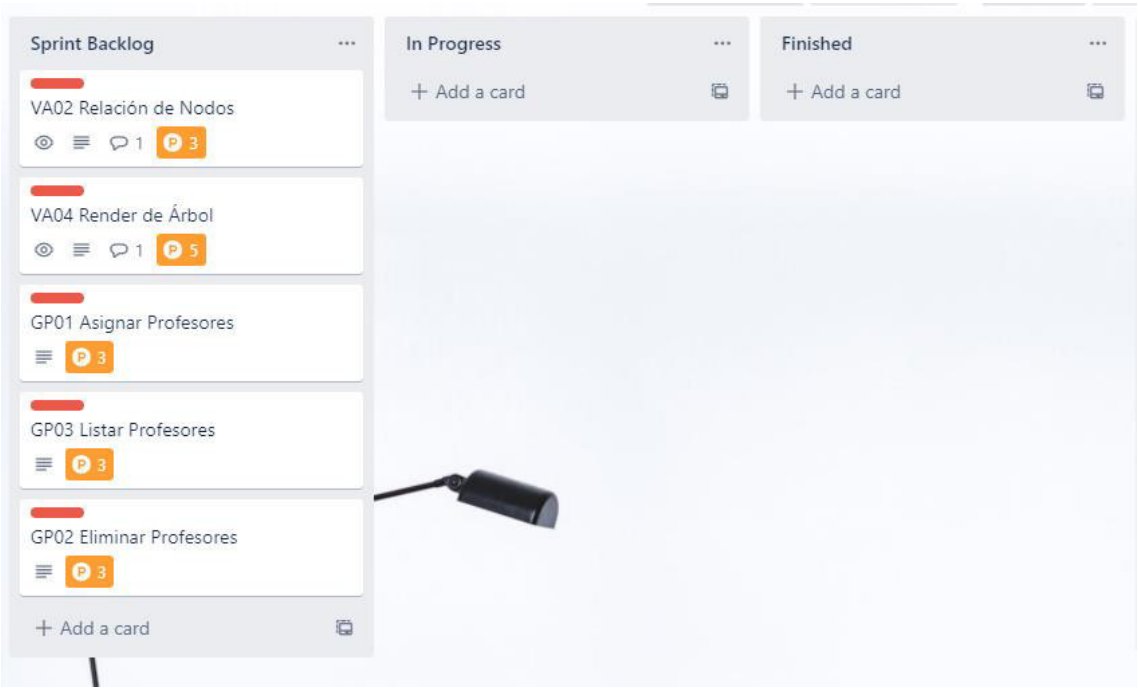


Figura 37. Tablero SCRUM del Sprint 4.

En la Tabla 26, se listan los requerimientos a desarrollar para el Sprint 4.

Código	Nombre	Descripción	Prioridad
Gestión de Profesores			
GP01	Asignar Profesores	Como Usuario, deseo relacionar profesores a un nodo para asignar profesores a los conocimientos	Alta
GP03	Listar Profesores	Como Usuario, requiero observar los profesores relacionados con un nodo	Alta
GP02	Eliminar Profesores	Como Usuario, deseo eliminar profesores para quitar las relaciones con los nodos que fueron asignados	Alta
Visualización de ADC			
VA03	Relación de Nodos	Como Usuario, necesito visualizar la relación entre nodos para conectar los conocimientos	Alta
VA04	Render de Árbol	Como Usuario, necesito visualizar los nodos simulando un árbol en su render	Alta

Tabla 26. Requerimientos para el Sprint 4.

2.6.2. HISTORIAS DE USUARIO

A continuación, se presentan los requerimientos para el Sprint 4 como Historias de Usuario. Se pueden observar desde la Tabla 27 hasta la Tabla 31.

VA02 Relación de Nodos	
Descripción:	Como Usuario, necesito visualizar la relación entre nodos para conectar los conocimientos
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El sistema debe mantener la relación padre-hijos siempre disponible para mejorar la renderización del árbol.	
2. El ADC puede tener solamente un nodo raíz.	
3. El sistema debe permitir que un nodo padre pueda tener varios nodos hijo.	

Tabla 27. Historia de Usuario VA02.

VA04 Render de Árbol	
Descripción:	Como Usuario, necesito visualizar los nodos simulando un árbol en su render
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	

1. El ADC debe tener ramas que no se sobrepongan entre sí.
2. Las ramas del ADC debe soportar más nodos hijo.
3. Las ramas que tienen nodos hijos deben estar alineados en su eje x

Tabla 28. Historia de Usuario VA04.

GP01 Asignar Profesores	
Descripción:	Como Usuario, deseo relacionar profesores a un nodo para asignar profesores a los conocimientos
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El nodo seleccionado debe permitir asociar profesores.	
2. El nodo debe asociar profesores con los atributos nombre, apellido, fecha de nacimiento y cargo.	

Tabla 29. Historia de Usuario GP01.

GP03 Listar Profesores	
Descripción:	Como Usuario, requiero observar los profesores relacionados con un nodo
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El nodo seleccionado debe listar todos los profesores asociados a ese conocimiento.	
2. El nodo debe mostrar los atributos nombre, apellido, fecha de nacimiento y cargo de los profesores.	

Tabla 30. Historia de Usuario GP03.

GP02 Eliminar Profesores	
Descripción:	Como Usuario, deseo eliminar profesores para quitar las relaciones con los nodos que fueron asignados
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El nodo seleccionado debe permitir eliminar los profesores asociados a ese conocimiento.	

Tabla 31. Historia de Usuario GP02.

2.6.3. IMPLEMENTACIÓN

A continuación, se muestra la implementación de cada Historia de Usuario del Sprint 4.

VA02 Relación de Nodos

Para mantener la consistencia entre las relaciones padre-hijo de los nodos, se procede a garantizar que los modelos cumplan con las siguientes condiciones:

- Un nodo tiene una altura determinada por el nivel del árbol que está ocupando y un ancho que representa la carga relacionada a ese conocimiento.
- Un nodo puede tener varios nodos hijos.
- Un ADC puede tener solo un nodo sin un padre asociado, este es un “nodo raiz”.

Para cumplir con estas condiciones es necesario incorporar un control en el renderizado del ADC. Este control establece una altura inicial para el nivel 0 del árbol y va disminuyendo linealmente cada vez que se sube de nivel en el ADC.

Para el resto de las condiciones, el modelo Nodo mantiene su consistencia. Esto se puede observar en la Figura 22.

VA04 Render de Árbol

Para darle un aspecto más apegado a como un árbol es en la vida real es necesario realizar unos ajustes con respecto a la separación y alineamiento de los nodos rama.

Se incorporan variables de separación entre nodos hermanos para controlar los espacios intermedios entre nodos de un mismo nivel. Además, se crea una variable de ajuste en el eje x para tomar en cuenta la diferencia que se genera al darle al trapecio un ángulo de inclinación. De esta manera, es posible renderizar los nodos con un aspecto más pulido. En la Figura 38, se observa el resultado de incorporar estas variables en el código que renderiza los nodos.

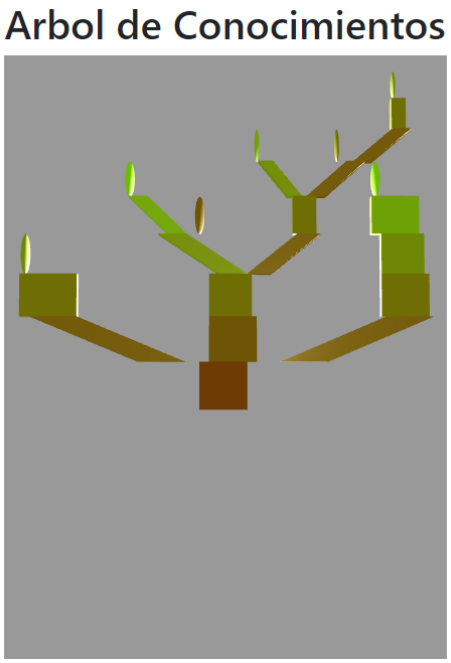


Figura 38. ADC ajustado.

GP01 Asignar Profesores

Para asignar profesores a un nodo se necesita construir el modelo Profesor con los atributos nombre, apellido, fecha de nacimiento y cargo. Además, se incluye un atributo en el modelo Nodo para permitir asociar varios profesores a un nodo del árbol. Para agregar profesores es necesario seleccionar el nodo deseado para habilitar un componente que muestra el botón “Añadir Profesor/a al Nodo”. Este botón despliega un formulario con los atributos necesarios para crear al profesor asociado al conocimiento.

En la Figura 39, se presenta el formulario creado para la asignación de un profesor a un nodo seleccionado.

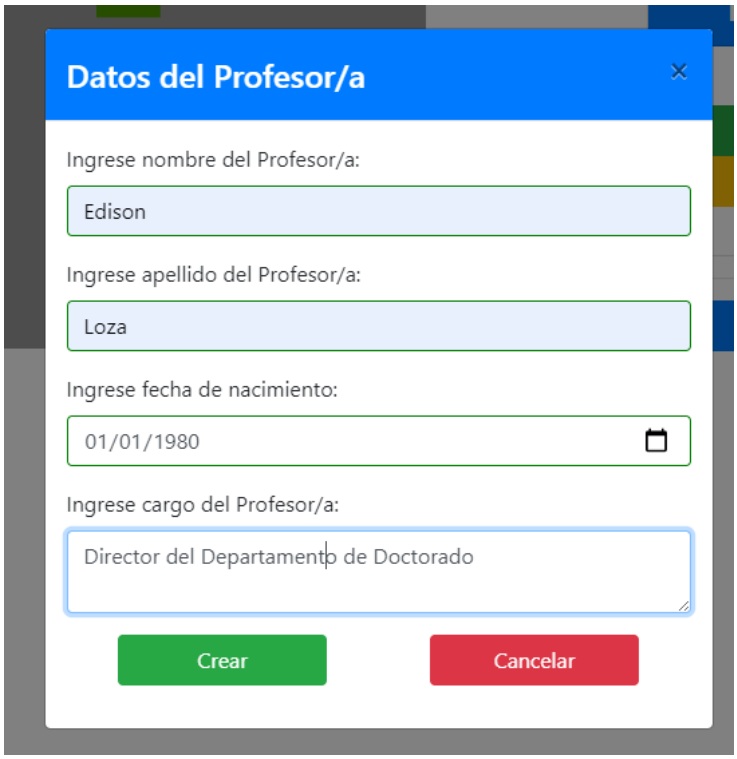


Figura 39. Formulario para creación de profesor.

GP03 Listar Profesores

Para visualizar todos los profesores relacionados con un nodo, se utiliza un componente del tipo acordeón de Bootstrap. Cada parte tiene como título la identificación del profesor, además cuenta con la funcionalidad de expandirse para mostrar botones para gestionar a cada profesor.

En la Figura 40, se muestra el listado de profesores de un conocimiento en el componente acordeón.

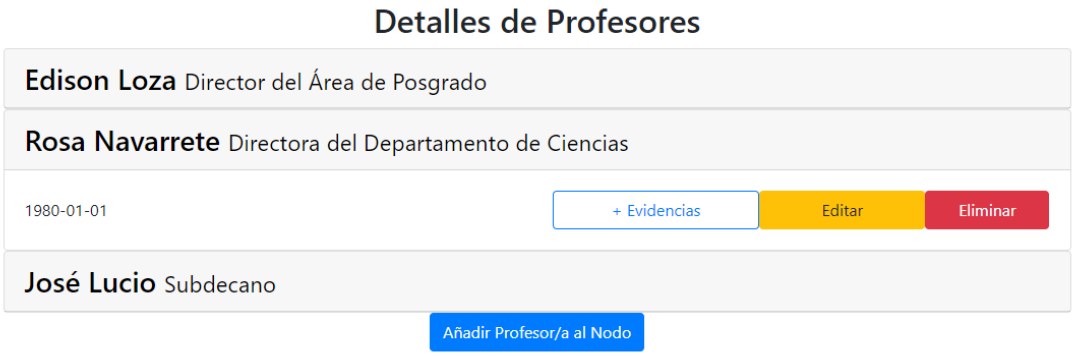


Figura 40. Listado de profesores asociados.

GP02 Eliminar Profesores

Para eliminar profesores, es necesario seleccionar un nodo que tenga a un profesor asociado. Luego, se ubica a un profesor de la lista y se hace clic para expandir el acordeón y mostrar el botón “Eliminar”. Haciendo clic en el botón el profesor queda eliminado del nodo. La Figura 40 brinda una vista al botón de eliminar, mientras que en la Figura 41 se puede observar el resultado de ejecutar la acción de eliminación.

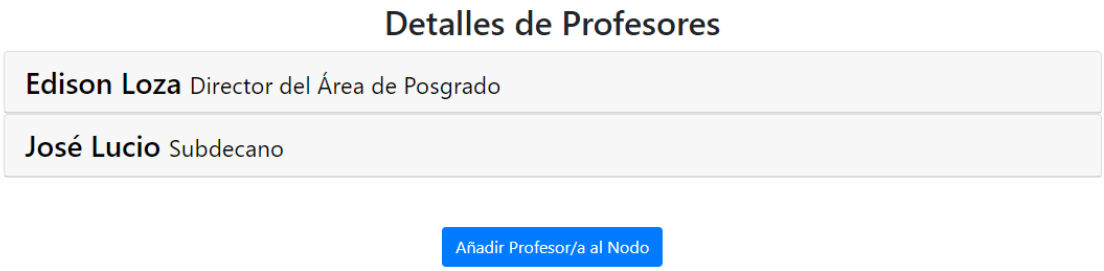


Figura 41. Eliminación de Profesor.

2.6.4. REVISIÓN DEL SPRINT

Al terminar el Sprint 4, se procede con la revisión del tablero Scrum para analizar la conformidad de las Historias de Usuario. En la Figura 42, se observa el tablero con todas las tarjetas terminadas.

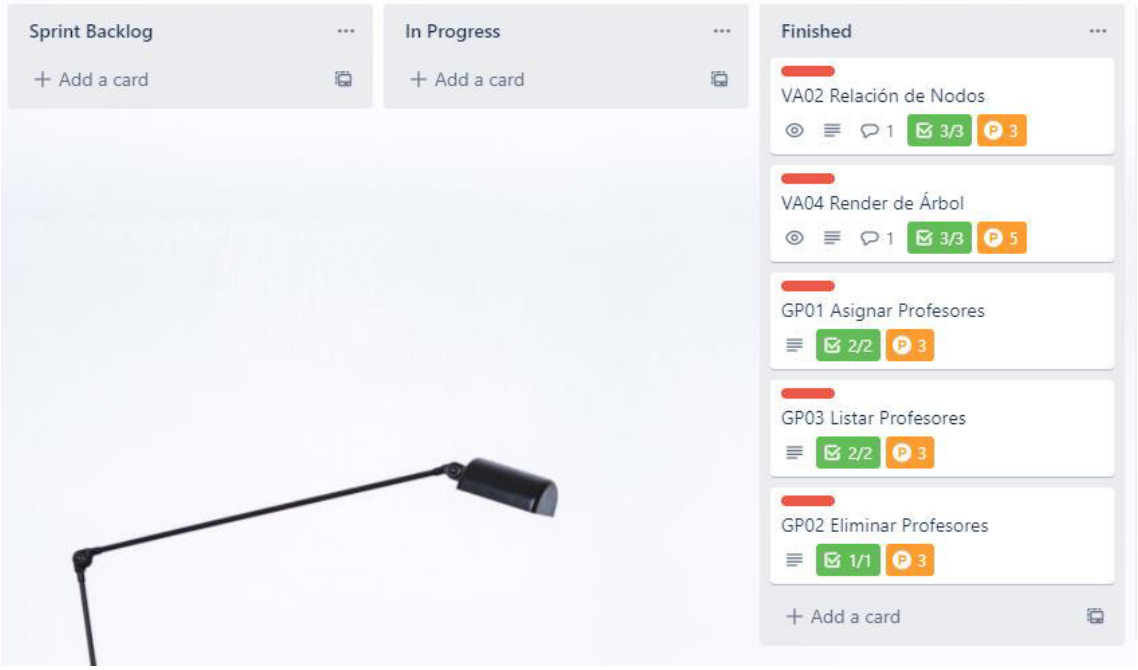


Figura 42. Tablero para revisión del Sprint 4.

Para analizar el Sprint 4, se muestra en la Tabla 32 un resumen de la implementación de las Historias de Usuarios.

Código	Nombre	Prioridad	Puntos	Estado	Aceptación
VA02	Relación de Nodos	Alta	3	Terminada	Aceptada
VA04	Render de Árbol	Alta	5	Terminada	Aceptada
GP01	Asignar Profesores	Alta	3	Terminada	Aceptada
GP03	Listar Profesores	Alta	3	Terminada	Aceptada
GP02	Eliminar Profesores	Alta	3	Terminada	Aceptada

Tabla 32. Revisión del Sprint 4.

Observaciones

Se observa en el Sprint 4 cierta holgura con respecto a los 17 puntos de trabajo, ya que muchas de las tareas de CRUD son familiares para el equipo de desarrollo. No se generan nuevos requerimientos para el siguiente Sprint.

2.7. SPRINT 5

Para el Sprint 4, se decide mantener un ritmo de 17 puntos tomando en cuenta los buenos resultados con los anteriores Sprints. Es así como, se seleccionan requerimientos con prioridad alta.

2.7.1. PLANIFICACIÓN DEL SPRINT

En la reunión planificación se decide dar por satisfecho el requerimiento VA05 Selección de Nodo, debido a que esta funcionalidad fue agregada durante el desarrollo de las historias del Sprint 2.

En la Figura 43, se puede observar el tablero al inicio del Sprint 5 con las Historias de Usuario seleccionadas.

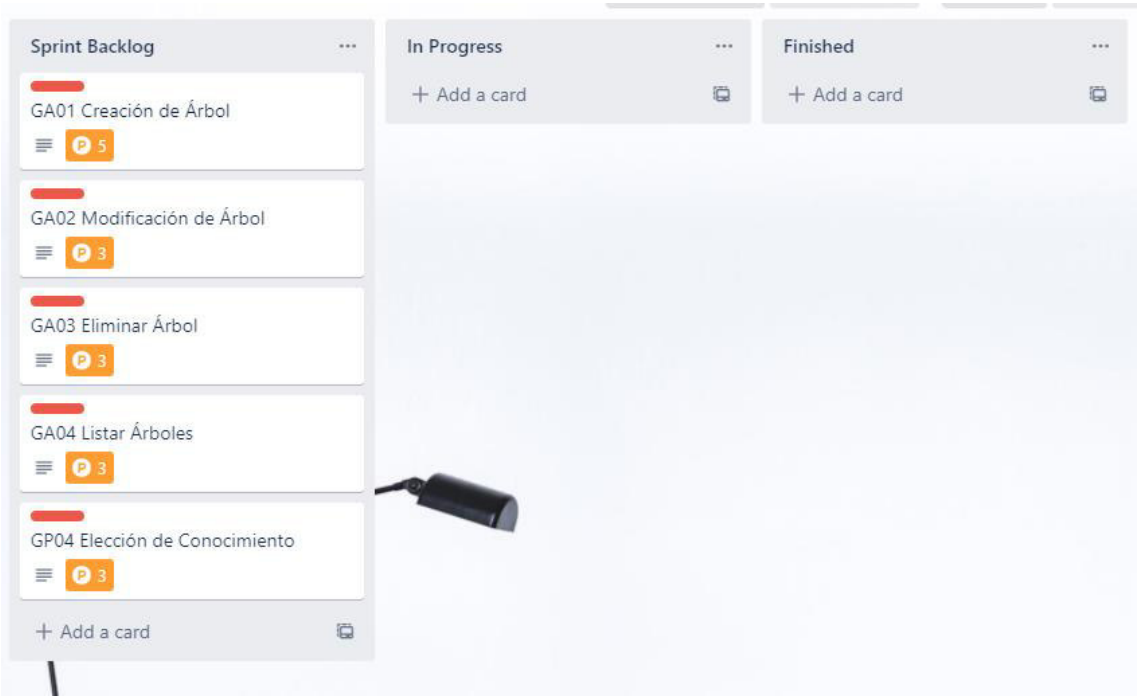


Figura 43. Tablero SCRUM del Sprint 5

En la Tabla 33, se listan los requerimientos seleccionados para el Sprint 5.

Código	Nombre	Descripción	Prioridad
Gestión de Árboles de Conocimiento			
GA01	Creación de Árbol	Como Usuario, deseo crear un nuevo ADC para realizar la Gestión del Conocimiento	Alta
GA02	Modificación de Árbol	Como Usuario, necesito cambiar los datos del ADC para actualizar la información	Alta
GA03	Eliminar Árbol	Como Usuario, requiero eliminar el ADC para remover todos los nodos innecesarios	Alta
GA04	Listar Árboles	Como Usuario, deseo observar un listado de los Árboles de Conocimiento para proceder con su visualización como se desee	Alta
Gestión de Profesores			
GP04	Elección de Conocimiento	Como Usuario, necesito determinar el tipo de conocimiento de un profesor para diferenciar el conocimiento formal del informal	Alta

Tabla 33. Requerimientos para el Sprint 5.

2.7.2. HISTORIAS DE USUARIO

A continuación, se presentan los requerimientos para el Sprint 5 como Historias de Usuario. Se pueden observar desde la Tabla 34 hasta la Tabla 38.

GA01 Creación de Árbol	
Descripción:	Como Usuario, deseo crear un nuevo ADC para realizar la Gestión del Conocimiento
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	5
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá crear un ADC con los atributos nombre y descripción.	

Tabla 34. Historia de Usuario GA01.

GA02 Modificación de Árbol	
Descripción:	Como Usuario, necesito cambiar los datos del ADC para actualizar la información
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá actualizar los datos de nombre y descripción de un ADC.	
2. El sistema deberá actualizar la fecha y hora de última modificación cuando se realice un cambio.	

Tabla 35. Historia de Usuario GA02.

GA03 Eliminar Árbol	
Descripción:	Como Usuario, requiero eliminar el ADC para remover todos los nodos innecesarios
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá eliminar el registro de un ADC.	
2. El sistema debe eliminar también los nodos asociados al ADC removido.	

Tabla 36. Historia de Usuario GA03.

GA04 Listar Árboles	
Descripción:	Como Usuario, deseo observar un listado de los Árboles de Conocimiento para proceder con su visualización como se desee
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor

Criterios de Aceptación
1. El sistema deberá presentar un listado de los Árboles de Conocimiento existentes.
2. El sistema debe presentar los atributos nombre, descripción, fecha de creación y de última modificación de cada ADC.

Tabla 37. Historia de Usuario GA04.

GP04 Elección de Conocimiento	
Descripción:	Como Usuario, necesito determinar el tipo de conocimiento de un profesor para diferenciar el conocimiento formal del informal
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El sistema deberá asignar a los profesores conocimientos formales o informales.	
2. El sistema permitirá agregar una descripción al conocimiento ingresado.	

Tabla 38. Historia de Usuario GP04.

2.7.3. IMPLEMENTACIÓN

A continuación, se muestra la implementación de cada Historia de Usuario del Sprint 5.

GA01 Creación de Árbol

Para la creación del Árbol se requiere crear un nuevo modelo en el Back-End. Este modelo viene a hacer de cabecera de la relación Arbol-Nodo. De esta manera, se tiene metadatos sobre un ADC. Para este caso se le agregan los atributos nombre y descripción. Para crear un registro se habilita una nueva ruta “inicio/arboles” previo a la renderización del ADC, con un botón “Crear Árbol” para desplegar un formulario de creación.

En la Figura 44, se puede observar el formulario incluido en el modal para crear un nuevo ADC.

Figura 44. Nuevo ADC.

GA02 Modificación de Árbol

La modificación de un ADC se lo realiza con la implementación de un botón “Editar” que lanza un modal solicitando por un nuevo nombre y descripción para el Árbol. Además, toda acción de actualización de registro se guarda en la base de datos y se puede solicitar para mostrar este atributo del ADC. En la Figura 45, se observa el formulario para actualizar la información de un ADC.

Figura 45. Edición de ADC.

GA03 Eliminar Árbol

Para eliminar el ADC deseado, se tiene un botón “Eliminar”. Este botón no solo borra el registro del ADC, además realiza una remoción en cascada de los nodos asociados al ADC eliminado. En la Figura 46, se observa la opción incorporada para remover un ADC.

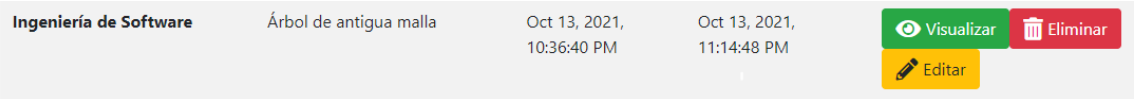


Figura 46. Eliminar ADC.

GA04 Listar Árboles

Se desea visualizar los ADCs registrados en la base de datos. Para esto, se utiliza una tabla que contiene los datos de los árboles además de una columna que contiene las acciones que pueden realizarse en cada registro. En la Figura 47, se listan los ADCs creados en el proyecto.

Figura 47. Listado de Árboles de Conocimiento.

Se añade en cada registro un botón “Visualizar” para proseguir con la renderización del ADC seleccionado en una ruta distinta.

GP04 Elección de Conocimiento

En la interfaz de gestión de profesores, se habilita un botón “+ Evidencias” que permite ingresar, en un modal, la descripción del conocimiento del profesor que está relacionado al nodo en que se encuentra. Para esto se hace una distinción entre conocimiento Formal y No Formal, aunque en ambos casos se solicita una simple descripción.

En la Figura 48, se observa el formulario del conocimiento con la pestaña de No Formal activa.

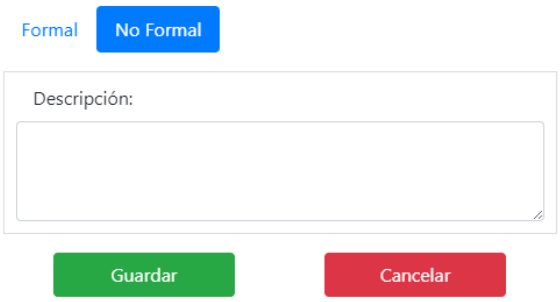


Figura 48. Formulario de Elección de Conocimiento.

2.7.4. REVISIÓN DEL SPRINT

Al terminar el Sprint 5, se procede con la revisión del tablero Scrum para analizar la conformidad de las Historias de Usuario. En la Figura 49, se observa el tablero con todas las tarjetas terminadas.

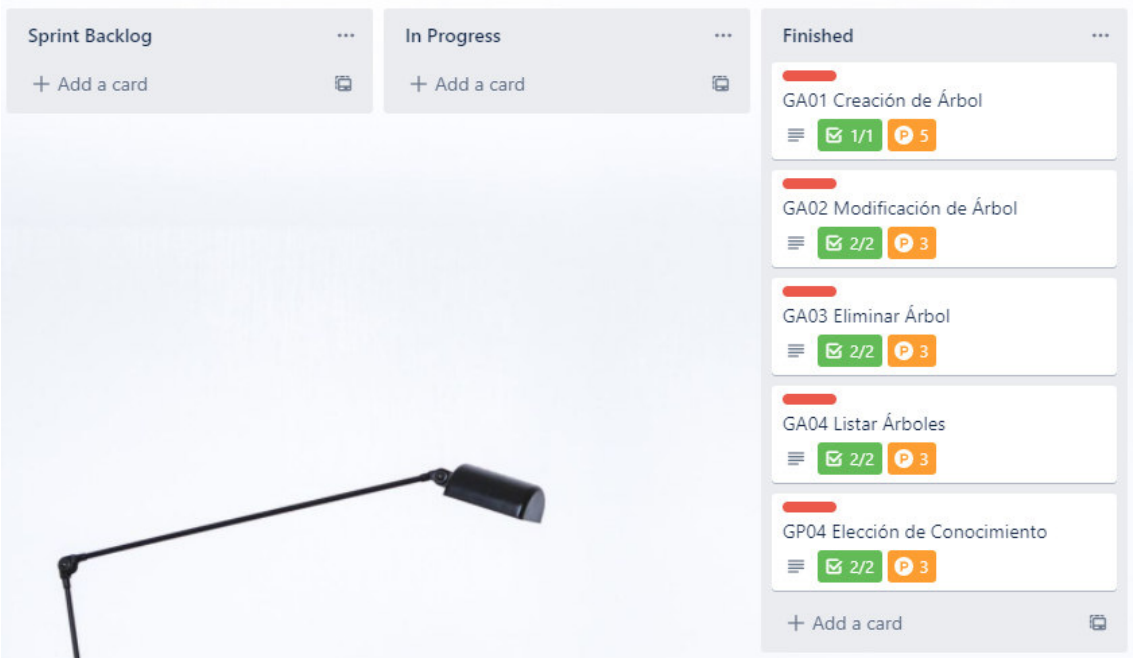


Figura 49. Tablero para revisión del Sprint 5.

Para hacer el análisis del Sprint 5, se puede observar en la Tabla 39 un resumen de la implementación de las Historias de Usuarios.

Código	Nombre	Prioridad	Puntos	Estado	Aceptación
GA01	Creación de Árbol	Alta	5	Terminada	Aceptada
GA02	Modificación de Árbol	Alta	3	Terminada	Aceptada
GA03	Eliminar Árbol	Alta	3	Terminada	Aceptada

GA04	Listar Árboles	Alta	3	Terminada	Aceptada
GP04	Elección de Conocimiento	Alta	3	Terminada	Aceptada

Tabla 39. Revisión del Sprint 5.

Observaciones

El Sprint 5 cumplió con los 17 puntos antes que el Sprint finalice, así que se contó con disposición de tiempo para buscar errores en el sistema. Este trabajo no se incluyó en una tarjeta. No se generaron nuevos requisitos para el siguiente Sprint.

2.8. SPRINT 6

Para el Sprint 6, se trabajó en los requerimientos restantes en el Product Backlog, sin llegar a ocupar los 17 puntos que se trabajó en los anteriores Sprints. Es así como, se seleccionan requerimientos con prioridad alta.

2.8.1. PLANIFICACIÓN DEL SPRINT

En la Figura 50, se puede observar el tablero al inicio del Sprint 6 con las Historias de Usuario seleccionadas.

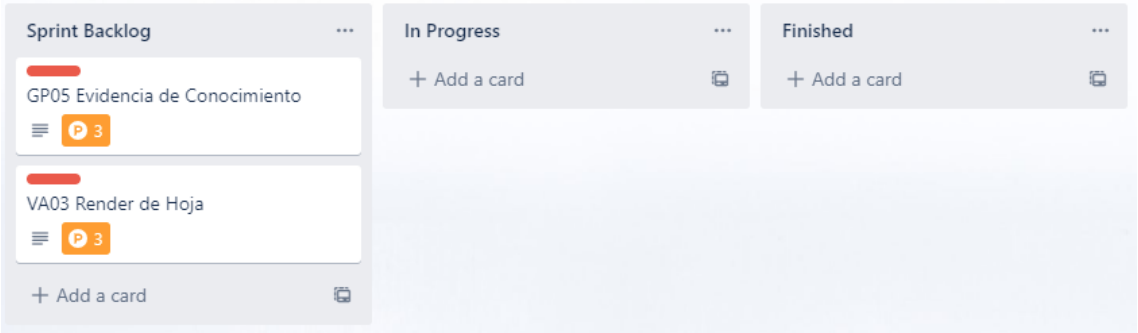


Figura 50. Tablero SCRUM del Sprint 6.

En la Tabla 40, se listan los requerimientos restantes para vaciar el Product Backlog que son los que se implementarán en el Sprint 6.

Código	Nombre	Descripción	Prioridad
Gestión de Profesores			
GP05	Evidencia de Conocimiento	Como Usuario, deseo registrar las evidencias del conocimiento para cada profesor	Alta
Visualización de Árboles de Conocimiento			
VA03	Render de Hoja	Como Usuario, necesito visualizar los nodos sin hijos a manera de hojas de un árbol	Alta

Tabla 40. Requerimientos para el Sprint 6.

2.8.2. HISTORIAS DE USUARIO

A continuación, se presentan los requerimientos para el Sprint 6 como Historias de Usuario. Se pueden observar en las Tablas 41 y 42.

GP05 Evidencia de Conocimiento	
Descripción:	Como Usuario, deseo registrar las evidencias del conocimiento para cada profesor.
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El sistema permitirá adjuntar evidencia del conocimiento descrito en el profesor como archivos .pdf o imágenes.	

Tabla 41. Historia de Usuario GP05.

VA03 Render de Hoja	
Descripción:	Como Usuario, necesito visualizar los nodos sin hijos a manera de hojas de un árbol
Prioridad:	Alta
Usuarios:	Usuario
Puntos:	3
Asignado:	Autor
Criterios de Aceptación	
1. El sistema renderizará un objeto elipsoide para representar a nodos tipo hoja.	

Tabla 42. Historia de Usuario VA03.

2.8.3. IMPLEMENTACIÓN

A continuación, se muestra la implementación de cada Historia de Usuario del Sprint 6.

GP05 Evidencia de Conocimiento

Para el conocimiento Formal, es necesario adjuntar una evidencia que respalde la adquisición del saber por parte del profesor. Esta evidencia puede ser un certificado o diploma en formato pdf o como imagen. Para esto, se habilita en el componente de gestión de profesores, dentro del modal de evidencia, un botón “Seleccionar archivo” para agregar un archivo como evidencia. En la Figura 51, se muestra un ejemplo del modal con un archivo subido.

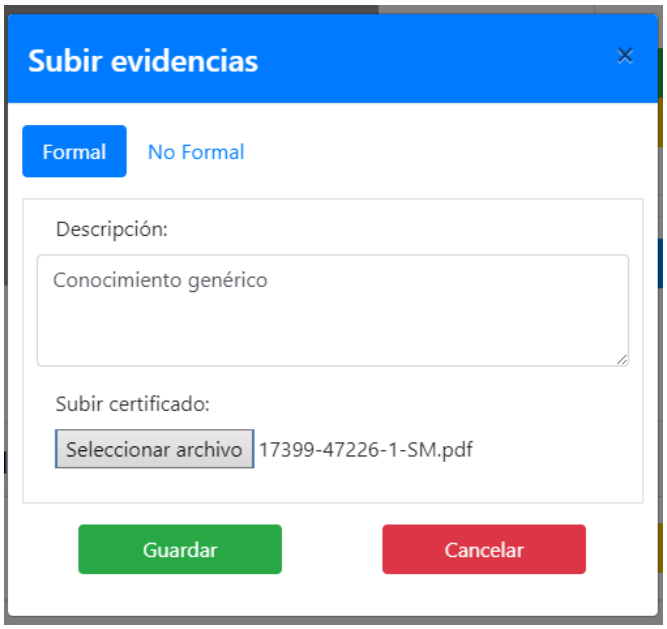


Figura 51. Evidencia para conocimiento Formal.

VA03 Render de Hoja

Para representar una hoja, se genera un objeto elipsoide provisto por la librería BabylonJs. La construcción del objeto se lo hace pasando dos parámetros, eje mayor y eje menor. Estos ejes son equivalentes al ancho y alto que se necesitaba para renderizar un cubo o un trapezoide. De esta manera, el objeto renderizado cumple con las dimensiones que el nodo posee. En la Figura 52, se puede apreciar el objeto siendo renderizado a manera de hoja del árbol.

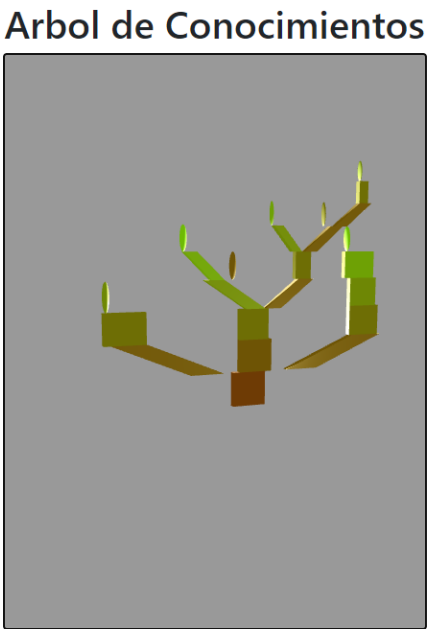


Figura 52. Render de objeto elipsoide.

2.8.4. REVISIÓN DEL SPRINT

Al terminar el Sprint 6, se procede con la revisión del tablero Scrum para analizar la conformidad de las Historias de Usuario. En la Figura 53, se observa el tablero con todas las tarjetas terminadas.

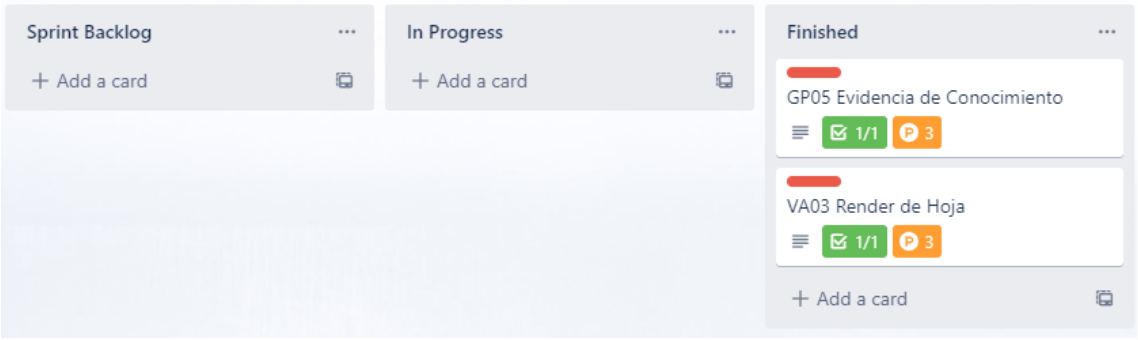


Figura 53. Tablero para revisión del Sprint 6.

Para hacer el análisis del Sprint 6, se puede observar en la Tabla 43 un resumen de la implementación de las Historias de Usuarios.

Código	Nombre	Prioridad	Puntos	Estado	Aceptación
GP05	Evidencia de Conocimiento	Alta	3	Terminada	Aceptada
VA03	Render de Hoja	Alta	3	Terminada	Aceptada

Tabla 43. Revisión del Sprint 6.

Observaciones

El Sprint 6 es el último ya que no se cuenta con más requerimientos en el Product Backlog y tampoco se generaron nuevos requerimientos.

CAPÍTULO 3: RESULTADOS

En este, se describen los resultados de implementar la aplicación web utilizando la metodología de desarrollo SCRUM. Se explica el modelo y funcionamiento del sistema, además de detallar dos casos en los que se representan ADCs dentro de la aplicación. Finalmente, se muestran los resultados las pruebas de integración realizadas al sistema.

3.1. DESCRIPCIÓN DEL SISTEMA

3.1.1. ARQUITECTURA DEL SISTEMA

Para el presente trabajo se utilizó una arquitectura Cliente-Servidor para la comunicación entre el Front-End y Back-End. Además, se implementó un contenedor de Docker para ubicar la base de datos y conectarla con el Back-End. La arquitectura del sistema se puede observar gráficamente en la Figura 54.

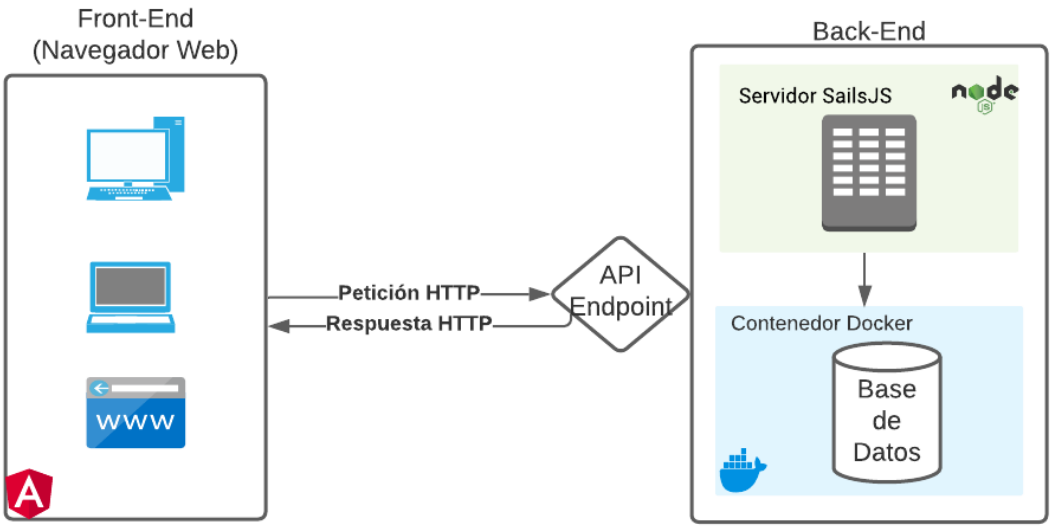


Figura 54. Arquitectura del Sistema

El Front-End del sistema tiene como clientes a cualquier dispositivo que cuente con un navegador web para cargar la aplicación web implementada en Angular y con Bootstrap como framework de UI.

El Back-End es un servidor web construido con el framework SailJs, que a su vez corre un proceso de NodeJs para ejecutar código JavaScript fuera de un navegador web. El servidor utiliza a SailsJs para construir y exponer API endpoints con la capacidad de

recibir y responder las peticiones a través del protocolo HTTP. Los endpoints expuestos se comunican en su totalidad mediante archivos tipo JSON. Además, el servidor se encarga de establecer conexión con la base de datos MongoDB utilizando un puerto expuesto por el contenedor de Docker para realizar la creación, lectura, actualización y eliminación de documentos JSON.

3.1.2. MODELO DE DATOS

Para el Back-End, se construyó un modelo de base de datos que se puede observar en la Figura 55. El modelo fue implementado utilizando el Object Relational Mapping (ORM) Waterline que viene incorporado en SailsJs. Para esto, se construyen las tablas implementando la estructura de clases en TypeScript, junto con las relaciones entre los modelos. El framework se encarga de generar la base de datos al momento de realizar la conexión con MongoDB si no existe una instancia existente. De esta manera, se puede manipular la estructura de la base de datos desde el Back-End sin tener que manipular en ningún momento MongoDB.

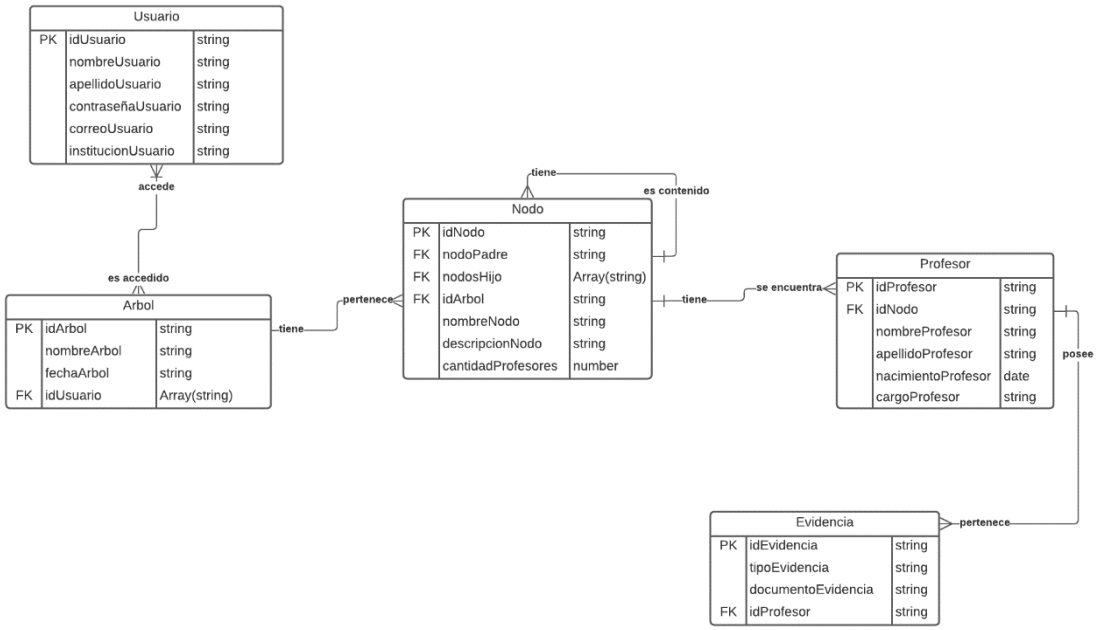


Figura 55. Modelo de Base de Datos.

Finalmente, el Back-End se configuró para establecer comunicación con MongoDB a través de un adaptador de SailsJs que viene incorporado en el framework. Este adaptador requiere el nombre de la base de datos, el puerto para la comunicación y credenciales del usuario de MongoDB. Al levantar el Back-End, SailsJs revisa si existe una instancia de la base de datos con la cual conectarse; caso contrario, se procede

con la creación de una nueva base considerando los modelos implementados con Waterline.

3.1.3. ALGORITMO DEL ADC

El análisis del algoritmo construido para generar un ADC requiere de una explicación previa de la manera en que se realiza el renderizado dentro del sistema; y después, se procede con el detalle de los pasos que sigue el algoritmo para construir la abstracción de ADC y su representación visual.

Proceso de Renderizado

Para lograr visualizar el ADC se utilizó la librería de renderizado 3D BabylonJS. El proceso de renderizado necesitó ser dividido en dos secciones principales. La primera fue encargada de la creación de una escena junto con la configuración adecuada para su posterior manipulación. Mientras que, la segunda sección se encargaba de instanciar esta escena, desde un componente de AngularJs, para manipular su estado.

Para la primera sección, se construyó una clase denominada “MotorRenderService” a manera de servicio. De esta manera, la clase creada era capaz de proveer una escena al componente cada vez que fuera instanciada, además de incluir los métodos necesarios para renderizar figuras específicas para lograr la abstracción de ADC (Figura 56).

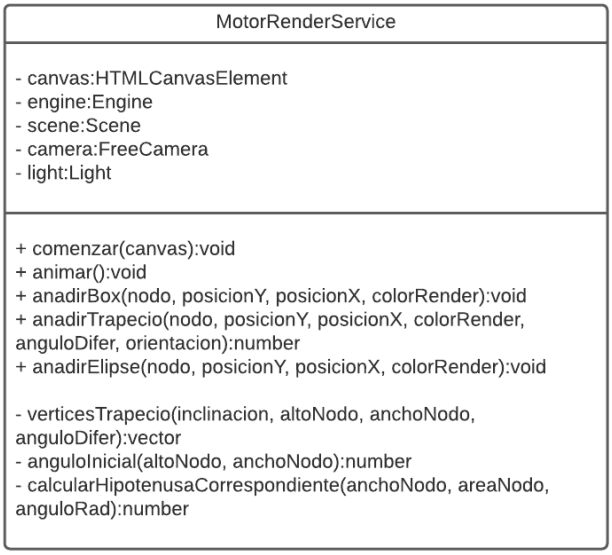


Figura 56. Clase “MotorRenderService”.

Para implementar la clase, fue necesario una trabajo en dos pasos. En el primer paso, se recibe como parámetro la referencia del elemento HTML, un “canvas”, que va a contener la escena. Además, se establecen la ubicación de la cámara al empezar la escena, la iluminación y el control para mover la cámara por la escena. En el segundo paso, se procede a ejecutar la función de renderizado de la escena con el método “runRenderLoop” de BabylonJS para renderizar cualquier cambio que exista dentro de la misma. Es importante considerar que dentro de la clase “MotorRenderService” se construyeron métodos para añadir objetos a la escena y que pueden ser llamados en tiempo de ejecución cuando el componente lo requiera.

Para la segunda sección, es necesario un componente de Angular que cuente con un elemento HTML del tipo “canvas”. Este componente es el encargado de llamar al servicio de renderizado del ADC, proporcionando la referencia del “canvas” en el que se va a visualizar la escena. Además, en este componente se procede con la llamada de los métodos disponibles en la clase “MotorRenderService” para añadir objetos a la escena según el algoritmo determine necesario. En la Figura 57, se muestra el diagrama para renderizar una escena y sus objetos.

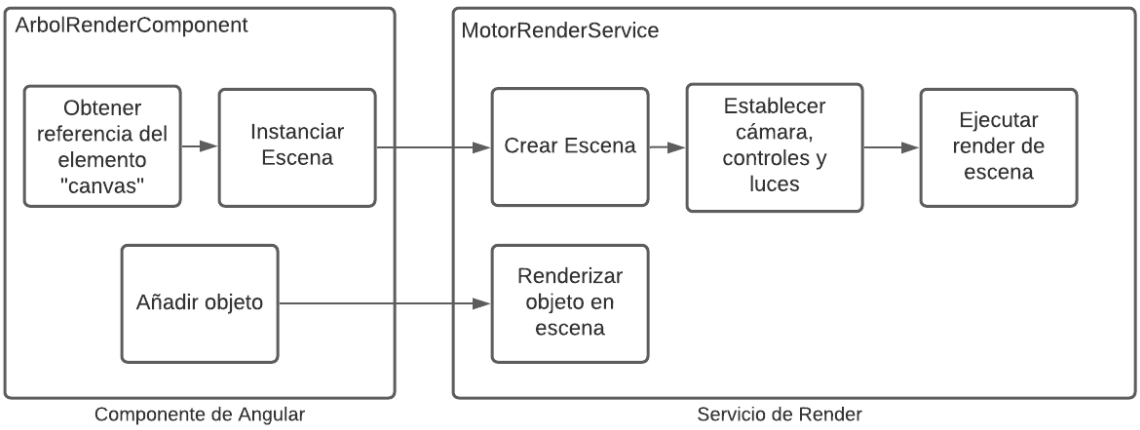


Figura 57. Diagrama de renderizado.

Construcción del ADC

El algoritmo encargado de la construcción del ADC necesita obtener los datos de los nodos antes de realizar la renderización del árbol. Se obtiene el objeto JSON completo, desde el Back-End, para identificar el nodo raíz. Desde este nodo se procede a recorrer el árbol en pre-orden, ejecutar las acciones de renderización y repetir con cada nodo hijo.

A continuación, se describen los pasos del algoritmo para la construcción del ADC y su visualización.

1. El algoritmo evalúa si el nodo tiene hijos para ordenarlos con un criterio de pirámide de acuerdo con el atributo “Índice de Paternidad”. De esta manera, se tiene un arreglo ordenado con nodos de menor índice acomodados en los extremos y los de mayor índice en el centro. Esto se lo realiza para mantener a los nodos con mayor potencial de tener hijos en la mitad y representar un tronco de un árbol.
2. El siguiente paso es obtener el arreglo de nodos hijos ordenado en una variable. Este arreglo va a ser útil en posteriores pasos del algoritmo.
3. Es necesario calcular la suma de los anchos que los nodos hijo tienen para determinar el espacio potencial que van a ocupar en la escena.
4. A continuación, se utiliza la suma de los anchos para calcular el ángulo de inclinación que las ramas tendrán para evitar la superposición entre nodos hermanos.
5. A continuación, se procede con la renderización del nodo actual considerando las propiedades de altura y ancho. Para esta renderización, se evalúa la condición del nodo; si es una rama, se renderiza el trapezoide regular; caso contrario, se renderiza un cubo.
6. Como resultado de la renderización, se procede con el cálculo los ejes de los nodos hijo para a posterior determinar la posición horizontal de estos.
7. Una vez obtenido el arreglo de ejes de los nodos hijo, se procede a cambiar el color que el siguiente nivel del árbol va a obtener, con el objetivo de hacerlo más verde a medida que va creciendo en altura. Además, se realiza un desplazamiento en los ejes de los nodos hijo a ser renderizados si estos son trapezoides.
8. Finalmente, el algoritmo procede a realizar el mismo recorrido para cada nodo hijo del actual a manera de pre-orden, siempre pasando la información de si el nodo padre es un trapezoide junto con su tipo de inclinación. Además, se revierte el color del objeto a renderizar cada que se pasa a una rama hermana en el árbol.

En la Figura 58, se puede observar el flujograma que describe los pasos a seguir por el algoritmo que renderiza los nodos a manera de ADC.

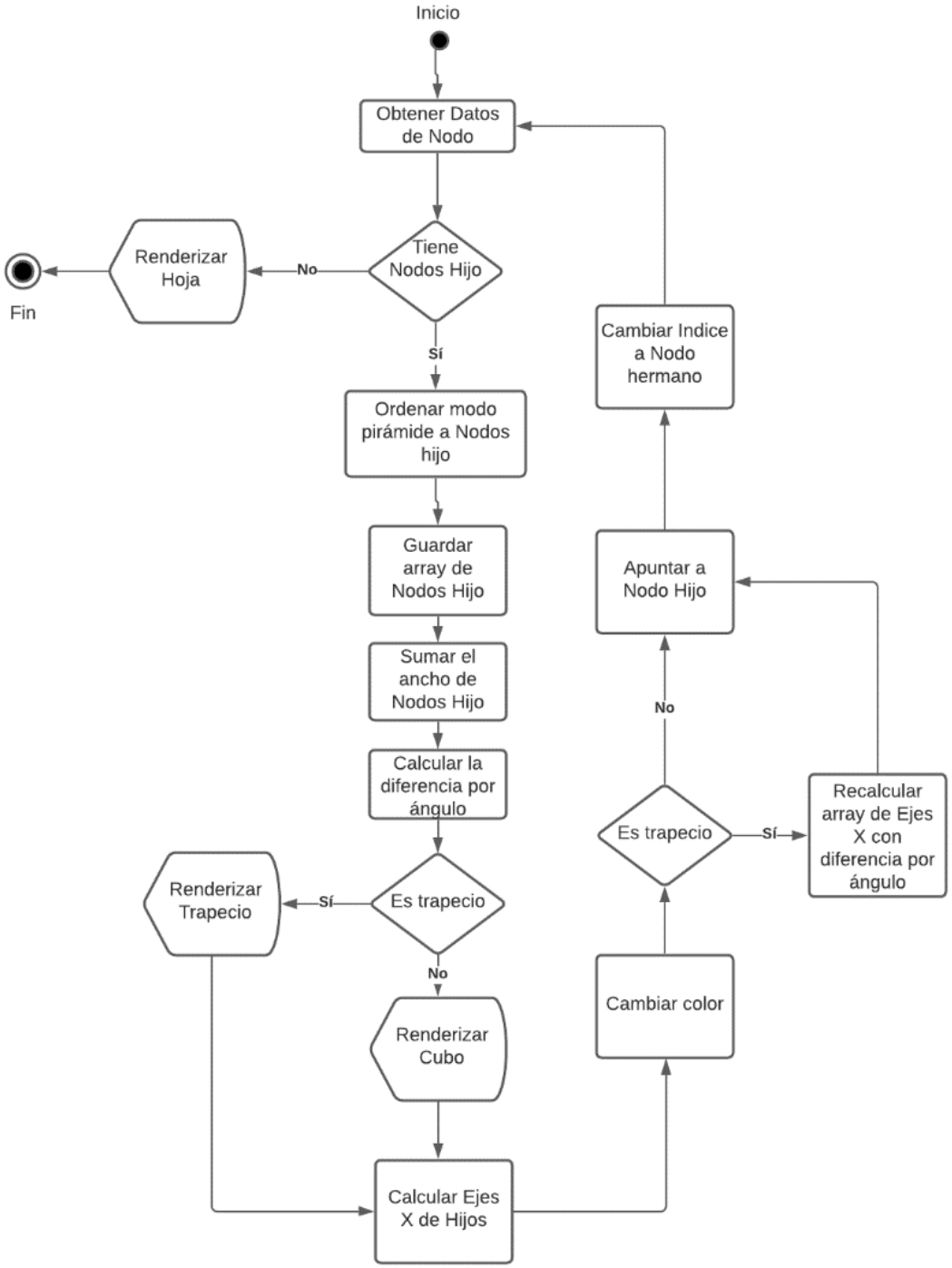


Figura 58. Flujograma de algoritmo de construcción de ADC.

3.2. REPRESENTACIÓN DE CASOS DE ESTUDIO

Para probar el algoritmo se decidió analizarlo con casos reales en los que se necesita mapear el conocimiento, como sucede en el DICC con el pensum de las carreras de Ingeniería en Software e Ingeniería en Computación.

Cada materia se representará como un nodo del ADC y las condiciones de prerrequisitos son las que determinan la relación padre-hijo entre nodos. Además, se procede a considerar la duración en horas de la materia como medida para determinar el ancho de su respectivo nodo. De este modo, nodos con mayor área representan materias con mayor carga horaria.

Para mejor visibilidad del Árbol, se aplica una transformación de las horas con una relación de 50 a 1. Por ejemplo, si una materia tiene una asignación de 200 horas, el correspondiente ancho del nodo va a ser de 4. La lista de las materias con su respectiva duración y equivalencia se encuentra en el Anexo 4.

3.2.1. CASO 1. INGENIERÍA EN SOFTWARE

Para construir el ADC de Ingeniería en Software, se utiliza la malla curricular de la carrera con un pensum aprobado en el año 2020, mismo que se encuentra como el Anexo 2 del presente trabajo. En el documento se detallan las materias que van a ser utilizadas junto con su duración en horas y las relaciones entre materias.

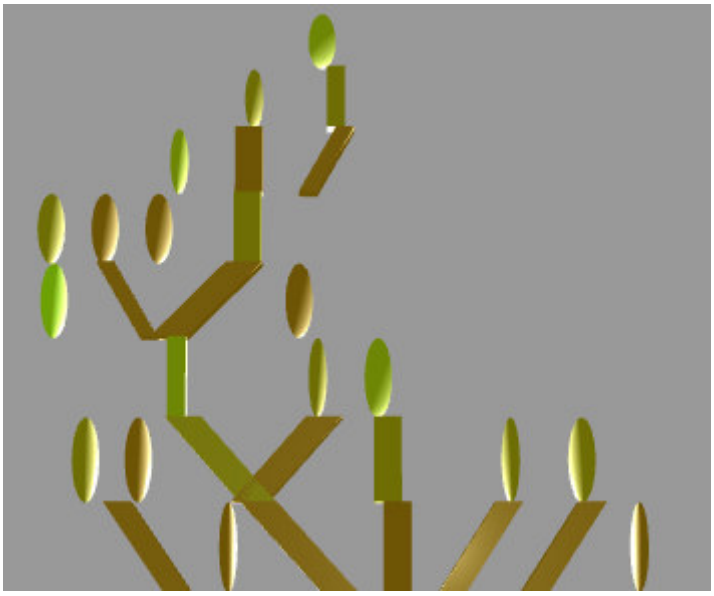


Figura 59. ADC de ingeniería en Software.

En la Figura 59, se observa un ADC con características asimétricas. La principal característica de esta representación es que no se tiene un tronco común inicial desde el que partan las ramas. Esto es debido a que en el primer semestre de la carrera de Ingeniería en Software se empieza con materias que no tienen una relación de prerrequisito; es decir, no se tiene un nodo inicial que sea padre del resto de nodos.

Además, se observan varias ramas sin mucho recorrido, inclusive un par de hojas pueden ser observadas desde el inicio del Árbol. Esto representa a las materias de prerrequisito que no se prolongan hasta semestres superiores. Es decir que, la rama se termina en los semestres iniciales de la carrera.

Finalmente, se observa una rama fuerte que se prolonga hasta las hojas de más alto nivel del Árbol. El nodo inicial de esta rama corresponde a la materia de "Programación I", siendo necesario destacar la importancia de esta materia para poder avanzar en los niveles de la carrera y el fuerte prerrequisito que tiene sobre materias de niveles superiores.

3.2.2. CASO 2. INGENIERÍA EN COMPUTACIÓN

Para construir el ADC de Ingeniería en Computación, se utiliza la malla curricular de la carrera con un pensum aprobado en el año 2020, mismo que se encuentra como el Anexo 1 del presente trabajo.

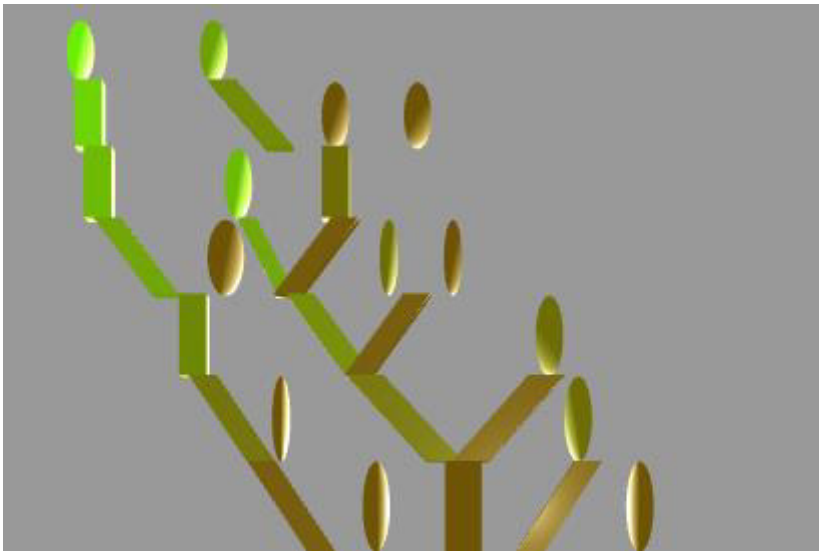


Figura 60. ADC de Ingeniería en Computación.

En la Figura 60, se observa un ADC con ramas más fuertes comparado con el de la Figura 59. Se cuentan con dos ramas que destacan sobre el resto y que cuyos nodos iniciales corresponden a las materias de "Fundamentos de electrónica para computación" y "Programación I".

El ADC mantiene las ramas iniciales sin conectarse a un tronco común; sin embargo, disminuye la cantidad de estas en un 50%. Esto implica que la carrera cuenta con pocas

materias de prerrequisito en las etapas iniciales, pero aumenta la dependencia de conocimientos previos a medida que se avanza en los semestres.

Finalmente, es necesario observar que, para ambos casos representados, cada nivel del árbol no corresponde a un periodo académico, o semestre, del pénsum de la carrera. Esto debido que el árbol representa la dependencia directa entre los nodos sin considerar el nivel en que los conocimientos se ubiquen dentro de la malla curricular. Por ejemplo, si una materia A es prerrequisito de una materia B estas se observan renderizadas en niveles adyacentes dentro del ADC para representar la relación padre-hijo, sin considerar que la materia A puede ser tomada en el primer semestre y la materia B pertenecer al tercer o cuarto semestre de la carrera.

3.3. PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación se utilizan para lograr la aprobación del cliente y proceder con el despliegue a producción. De manera que, en esta ección, se describen casos de uso para realizar su validación por parte del Product Owner. Los casos de uso son una evolución de las Historias de Usuario y del Product Backlog [36]. Se caracteriza a cada caso de uso con una identificación, nombre, descripción, precondiciones, pasos a seguir, los resultados esperados y observaciones importantes.

A continuación, se presenta el modelo de caso de uso diseñado para cubrir los principales requerimientos funcionales del sistema, en la Tabla 44. El proceso de construcción dio como resultado la generación de 16 casos de uso, mismos que pueden ser encontrados en el Anexo 3.

UC01 Registro de Usuario	
Descripción:	El usuario puede registrar sus datos en el sistema.
Precondiciones:	Ninguna
Pasos:	1. El usuario selecciona la opción de “Registrarse” de la barra de navegación. 2. El sistema muestra un formulario con los datos del usuario requeridos. 3. El usuario introduce la información en el formulario. 4. El sistema valida los datos ingresados en el formulario. 5. El usuario presiona el botón “Registrar”.
Resultados:	1. El sistema muestra un mensaje de éxito 2. El sistema guarda al nuevo usuario.
Observaciones:	- El botón de “Registrar” no estará habilitado al menos que los datos ingresados por el usuario sean válidos.

Tabla 44. Modelo de Caso de Uso.

El proceso para la validación de cada caso de uso se constituyó de los siguientes pasos:

1. Determinar el caso de uso a probar.
2. Obedecer las precondiciones descritas en el caso.
3. Seguir en el orden descrito cada paso del caso de uso.
4. Evaluar si el resultado del sistema coincide con lo establecido en el caso de uso.
5. Se aprueba el caso de uso.

Las pruebas de los casos de uso fueron realizadas junto con el Product Owner. Además, se verificó que las acciones que manipulaban los datos del sistema se vean reflejadas con éxito en la base de datos. El resultado de probar cada caso de uso se puede observar a manera de resumen en la Tabla 45.

Aceptación del Sistema	
Caso de Uso	**Estado**
UC01 Registro de Usuario	Aprobado
UC02 Inicio de Sesión	Aprobado
UC03 Visualización de Árboles de Conocimiento	Aprobado
UC04 Creación de ADC	Aprobado
UC05 Eliminación de ADC	Aprobado
UC06 Modificación de ADC	Aprobado
UC07 Visualización de Nodo	Aprobado
UC08 Creación de nuevo Nodo	Aprobado
UC09 Eliminación de Nodo	Aprobado
UC10 Modificación de Nodo	Aprobado
UC11 Renderización diferenciada	Aprobado
UC12 Descargar Árbol en archivo	Aprobado
UC13 Asociar Profesor con Nodo	Aprobado
UC14 Modificación de Profesor	Aprobado
UC15 Eliminación de Profesor	Aprobado
UC16 Categorizar conocimiento de Profesor	Aprobado

Tabla 45. Resumen de pruebas de los Casos de Uso.

El resultado de probar los casos de uso del sistema completo fue favorable, denotando que se cumple con todos los requisitos descritos en el Product Backlog. De esta manera, se procedió con la aceptación del sistema por parte del Product Owner y declarar al producto como terminado.

CAPÍTULO 4

4.1. CONCLUSIONES

- Los resultados del presente trabajo sugieren una importante flexibilidad por parte de los ADC para analizar diferentes dimensiones del saber dentro de una organización. Debido a que, los nodos pueden asociarse con distintas unidades de medida para cambiar el contexto del análisis; por ejemplo, si en los nodos se reemplaza la métrica de horas de duración de una materia con la métrica de número de publicaciones realizadas por la materia, se puede determinar ramas con mayor relevancia científica en lugar de ramas con más carga horaria.
- El sistema desarrollado en el presente trabajo facilita el proceso de análisis del pénsum de una carrera, por encima del uso tradicional de archivos de texto plano u hojas de cálculo; ya que, al comparar visualmente las ramas del ADC se puede determinar con mayor facilidad su relevancia, y la de los nodos que la conforman, dentro del pénsum de estudios.
- En el presente trabajo, la metodología de desarrollo SCRUM fue fundamental para la rápida comunicación entre el equipo de desarrollo y el Product Owner. Esta metodología permitió recibir retroalimentación de manera rápida para corregir el comportamiento del sistema en sus versiones tempranas. Además, el uso de Historias de Usuario para definir los requerimientos permitió centrar la atención en la satisfacción de las necesidades de los interesados por encima de las consideraciones técnicas del sistema.
- El presente trabajo requirió de una focalización del esfuerzo en tareas relacionadas al renderizado 3D y su incorporación dentro del algoritmo de Árboles de Conocimiento. Debido a esto, la implementación del sistema en sus componentes de autenticación, transferencia de datos y seguridad se limitó a lo básico. Sin embargo, la aprobación total del sistema por parte del Product Owner sugiere que las estrategias implementadas de parte del equipo de desarrollo fueron acertadas.
- La construcción del algoritmo para representar visualmente los ADC tiene un importante grado de dificultad al momento de evitar la superposición de objetos. Se requiere de un conocimiento previo en el área de geometría plana y espacial para implementar las soluciones a este problema.

4.2. RECOMENDACIONES

- El sistema fue desarrollado para construir visualmente un ADC; sin embargo, existen elementos de las interfaces de usuario que tienen potencial de mejora. Se podría invertir recursos para trabajar en tareas de diseño de interfaces y experiencia de usuario.
- Se recomienda incorporar una sección de comparación para observar dos ADCs a la vez con el objetivo de analizar diferencias o características en común entre nodos y ramas.
- Se recomienda el incorporar un módulo de estadística del ADC con el motivo de estudiar las consecuencias si se realizan distintos cambios en un nodo o rama.
- Se recomienda implementar mensajes de ayuda o tutoriales para que usuarios con poca experiencia en utilización de aplicaciones web tengan la oportunidad de aprovechar el sistema.

REFERENCIAS BIBLIOGRÁFICAS

[1] D. Di Tullio, «Knowledge-based theory of the firm,» IS Theory, 21 Noviembre 2014. [En línea]. Available: https://is.theorizeit.org/wiki/Knowledge-based_theory_of_the_firm. [Último acceso: 23 Julio 2020].

[2] R. Colle, «La Representación Individual y Colectiva del Conocimiento Adquirido,» Razón, Estado de Mexico, 2004.

[3] L. Vallejo, Gestión del Talento Humano, vol. I, E. S. P. d. Chimborazo, Ed., Riobamba: La Caracola Editores, 2016.

[4] J. Bresson, «Construction et questionnement informatisé d'un arbre de connaissances,» Institut français de l'éducation, Reims.

[5] Badgr, «Using the Badgr Backpack,» Badgr Support, Febrero 2020. [En línea]. Available: https://support.badgr.com/portal/en/kb/articles/using-the-badgr-backpack. [Último acceso: 10 Julio 2020].

[6] B. Villeret, «Trivium s'associe aux cabinets conseil autour d'une offre dédiée aux RH,» Consulting News Line, 11 Junio 2003. [En línea]. Available: https://www.consultingnewsline.com/Info/Attach%E9%20Case/Solutions/Trivium.html. [Último acceso: 18 Julio 2020].

[7] Broadcom, «CA SERVICE MANAGEMENT,» Broadcom, 4 Septiembre 2019. [En línea]. Available: https://techdocs.broadcom.com/content/broadcom/techdocs/es/es/ca-enterprise-software/business-management/ca-service-management/14-1/uso/gesti_n-del-conocimiento/creaci_n-de-documentos-de-_rbol-de-conocimiento.html. [Último acceso: 19 Julio 2020].

[8] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber y J. Sutherland, Manifesto for Agile Software Development, 2001.

[9] K. Schwaber y J. Sutherland, Scrum Guide, Scrum Org., 2020.

[10] R. Ruggles, Knowledge Management Tools, Newton: Butterworth-Heinemann, 1997.

[11] K. Torres y P. Lamenta, «La Gestión del Conocimiento y los Sistemas de Información en las organizaciones,» *Negotium,* vol. 11, nº 32, pp. 3-20, Noviembre 2015.

[12] Wikipedia, «Arbre de connaissances,» Wikipédia. L'encyclopédie libre., 28 Noviembre 2019. [En línea]. Available: https://fr.wikipedia.org/w/index.php?title=Arbre_de_connaissances&oldid=164940404. [Último acceso: 17 Julio 2020].

[13] M. Bellinza, «Gestión del Conocimiento: Aproximaciones teóricas,» *Clío América,* vol. 5, nº 10, pp. 257-271, Diciembre 2011.

[14] G. Rossi, O. Pastor, D. Schwabe y L. Olsina, Web Engineering: Modelling and Implementing Web Applications, Springer, 2008.

[15] A. Nayak, A. Poriya y D. Poojary, «Type of NOSQL Databases and its Comparison with,» *International Journal of Applied Information Systems (IJAIS),* vol. 5, nº 4, pp. 16-19, 2013.

[16] K. Chodorow, MongoDB: The Definitive Guide., Segunda ed., Sebastopol: O'Reilly, 2013.

[17] Google, «Introduction to the Angular Docs,» Google, 2021. [En línea]. Available: https://angular.io/docs. [Último acceso: 10 2021].

[18] Mozilla, «Javascript,» MDN Web Docs, 2021. [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Último acceso: 10 2021].

[19] Microsoft, «Typescript Documentation,» Microsoft, 2021. [En línea]. Available: https://www.typescriptlang.org/docs/. [Último acceso: 10 2021].

[20] Vercel, «Welcome to Getting Started with Babylon.js,» Vercel, 2021. [En línea]. Available: https://doc.babylonjs.com/start. [Último acceso: 10 2021].

[21] RxJs, «Get Started,» 2021. [En línea]. Available: https://rxjs.dev/guide/overview. [Último acceso: 10 2021].

[22] Bootstrap team, «Bootstrap Introduction,» Bootstrap, 2021. [En línea]. Available: https://getbootstrap.com/docs/5.1/getting-started/introduction/. [Último acceso: 10 2021].

[23] Fonticons, Inc., «Basic Use,» Font Awesome, 2021. [En línea]. Available: https://fontawesome.com/v5.15/how-to-use/on-the-web/referencing-icons/basic-use. [Último acceso: 10 2021].

[24] Mozilla, «CSS,» MDN Web Docs, 2021. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/CSS. [Último acceso: 10 2021].

[25] W3Schools, «HTML Tutorial,» W3Schools, 2021. [En línea]. Available: https://www.w3schools.com/html/. [Último acceso: 10 2021].

[26] The Sails Company, «Sails Documentation,» 2021. [En línea]. Available: https://sailsjs.com/documentation/reference. [Último acceso: 10 2021].

[27] OpenJS Foundation, «Express. Fast, unopinionated, minimalist web framework for Node.js,» Express, 2017. [En línea]. Available: https://expressjs.com. [Último acceso: 10 2021].

[28] OpenJS Foundation, «Acerca de Node.js,» Joyent, Inc., 2020. [En línea]. Available: https://nodejs.org/es/docs/. [Último acceso: Octubre 2021].

[29] Docker Inc., «Get started,» Docker, 2021. [En línea]. Available: https://docs.docker.com/get-started/. [Último acceso: Octubre 2021].

[30] MongoDB, Inc., «Introduction to MongoDB,» 2021. [En línea]. Available: https://docs.mongodb.com/manual/introduction/. [Último acceso: Octubre 2021].

[31] Microsoft, «Getting Started,» Visual Studio Code, 2021. [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: Octubre 2021].

[32] Git, «Git Documentation,» 2021. [En línea]. Available: https://git-scm.com/docs. [Último acceso: Octubre 2021].

[33] GitHub,Inc, «About Git,» 2021. [En línea]. Available: https://docs.github.com/en/get-started/using-git/about-git. [Último acceso: Octubre 2021].

[34] npm, Inc., «About npm,» 2021. [En línea]. Available: https://docs.npmjs.com/about-npm. [Último acceso: Octubre 2021].

[35] L. Rivero, I. Silva, P. Cutrim, D. Dias, G. Braz, A. Paiva, E. Alves y M. Oliveira, «Implementación de Gitflow para la gestión de la configuración en un proyecto de desarrollo de software ágil. Un informe de experiencia.,» *Computer on the Beach,* vol. XII, pp. 178-185, 2021.

[36] R. Miller y C. Collins, «Acceptance Testing,» *Proc. XPUniverse,* vol. 238, 2001.

ANEXOS

ANEXO 1. MALLA CURRICULAR DE INGENIERÍA EN COMPUTACIÓN

Nombre del documento anexo: MallaCurricularComputacionPensum2020.pdf

ANEXO 2. MALLA CURRICULAR DE INGENIERÍA EN SOFTWARE

Nombre del documento anexo: MallaCurricularSoftware2020.pdf

ANEXO 3. CASOS DE USO DEL SISTEMA

Nombre del documento anexo: CasosdeUso.xlsx

ANEXO 4. LISTA DE MATERIAS

Nombre de documento anexo: MateriasEquivalencias.xlsx