Facultad de Economía y Negocios

Escuela de Ingeniería Informática Empresarial

Diseño e implementación de software guía para el modelado de Dinámica de Sistemas

Autores:

Pablo Andrés Abarca Martínez

Benjamín Osvaldo Moya Calderón

Profesor Guía:

Luis Eduardo Canales Carrasco

Proyecto de memoria para optar al título de INGENIERO INFORMÁTICO EMPRESARIAL

Talca – Chile

2022

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su unidad de procesos técnicos certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Talca, 2023

Índice de contenido

Índice de tablas

Índice de figuras

Resumen ejecutivo

Los sistemas de información proveen de información vital a las organizaciones y suponen un gran apoyo a sus operaciones. En el ámbito de la dinámica de sistemas, el profesor Martin Schaffernicht ha trabajado hace años en mejorar la enseñanza de dicha disciplina a pesar de los contratiempos que supusieron la pandemia, la educación en línea y las limitaciones de una guía en formato Word. En un ánimo de continuar innovando en su propuesta docente, el profesor Schaffernicht plantea la idea de implementar un sistema de información, en forma de plataforma web, que sirva como guía para el aprendizaje del modelado conceptual de dinámica de sistema.

El objetivo de esta plataforma es solventar las brechas que ha detectado en las versiones anteriores de su guía de modelado conceptual. Esto es, a través de una interfaz amigable con los alumnos, que ofrezca una experiencia intuitiva para obtener los conocimientos del curso. Al mismo tiempo, se busca obtener retroalimentación de los avances de los alumnos, lo que le permite tanto a ellos como a los profesores del módulo conocer cómo progresan en los aprendizajes.

Este proyecto se centra en el desarrollo de la plataforma web antes mencionada, así como en el modo en el que se abordan los desafíos que conllevan la adaptación de los contenidos de la guía de modelado conceptual a un formato de web interactiva, apoyándolo desde el punto de vista de la experiencia de usuario (UX) y la gamificación.

Abstract

Information systems provide vital information to organizations and support their operations. In the field of system dynamics, Professor Martin Schaffernicht has been working for years to improve the teaching of this discipline despite the setbacks of the pandemic, online education, and the limitations of a Word guide. In an effort to continue innovating his teaching proposals, Professor Schaffernicht suggests the idea of implementing an information system, in the form of a web platform, to serve as a guide for learning the conceptual modeling of system dynamics.

The objective of this platform is to overcome the gaps that have been detected in previous versions of its conceptual modeling guide. That is, through a student-friendly interface that offers an intuitive experience to obtain the knowledge of the course. At the same time, it seeks to obtain feedback on the progress of the students, which allows them as well as the teachers of the module to know how they are progressing in their learning.

This project focuses on the development of the previously mentioned web platform, as well as the way in which the challenges involved in adapting the contents of the conceptual modeling guide to an interactive web format are addressed, supporting it from a user experience (UX) and gamification point of view.

Agradecimientos

“A mi familia, y no me refiero solamente con quienes comparto un lazo sanguíneo. Gracias por siempre estar ahí, entregando apoyo, palabras de aliento y ánimo para lograr alcanzar mis metas, por dar una mano frente a cualquier circunstancia y gracias por permitirme seguir mis sueños.”

Pablo Andrés Abarca Martínez

“A mis padres, mis abuelos y mis amigos por ser un constante apoyo y motivación durante todo este proceso. El viaje no fue fácil, pero su ayuda y compañía me ha motivado a completar este camino.”

Benjamín Osvaldo Moya Calderón

1. Introducción

Según Schaffernicht (2021) y la System Dynamics Society, la dinámica de sistemas es una disciplina que permite analizar y modelar el comportamiento de entornos complejos y dinámicos dentro de un determinado periodo de tiempo. Esta práctica requiere la comprensión de conceptos clave, la habilidad de aplicarlos libremente en el momento de modelar y la utilización de herramientas para analizar sistemas dinámicos. Además de lo anterior, la dinámica de sistemas está basada en supuestos fundamentales los cuales son:

- Los responsables de la toma de decisiones tienen un profundo conocimiento de los sistemas en los que trabajan.
- Tratan de desarrollar recursos que se acumulan y disminuyen dentro y fuera de las existencias.
- Los sistemas de poblaciones interactúan entre sí mediante circuitos de retroalimentación intencionados o no intencionados.

Los resultados obtenidos a través del modelado se pueden utilizar para generar conclusiones que aporten en la comprensión del comportamiento de un sistema complejo.

A pesar de la existencia de guías asociadas a las competencias del modelado sistémico-dinámico, el cual se encarga de simular sistemas para su análisis y estudio, existe una brecha en el material de aprendizaje para estudiantes nuevos de esta área en particular, pues se asume una formación previa de pregrados de ingeniería y en métodos cuantitativos (Schaffernicht, 2021). Esto provoca que nuevos estudiantes que muchas veces no poseen esta formación se les complique de sobremanera y generen modelos de mala calidad.

Durante el año 2021 el profesor Martin Schaffernicht de la Universidad de Talca, diseñó una “Guía práctica para la conceptualización y formalización de modelos de dinámica de sistemas”, documento de 53 páginas en Microsoft Word utilizada por los estudiantes del curso dinámica de sistemas de la carrera Ingeniería Informática Empresarial (IIE). Luego de su aplicación, el profesor Schaffernicht determinó que el conjunto de tareas, procedimientos y reglas de la guía es útil. Sin embargo, su implementación en formato Word posee carencias:

- El proceso de búsqueda es ineficiente y una potencial fuente de frustración, pues los alumnos deben registrar todo el documento a través del panel de navegación para encontrar la tarea, subtarea, procedimiento, subprocedimiento o regla que necesitan.
- No es posible hacer un seguimiento de la navegación de los alumnos en la guía, sin poder retroalimentarlos por su progreso en el avance del curso. Además, es una fuente de información desaprovechada para continuar mejorando el conjunto de tareas, procedimientos y reglas de la guía para facilitar su navegación.

En esta tesis se propuso implementar una plataforma web diseñada con los principios de la experiencia de usuario (UX), que permita a los estudiantes que se inician en la dinámica de sistemas acceder de forma rápida y cómoda a los conceptos, definiciones y procedimientos necesarios para aplicar los conocimientos que aún no manejan de manera autónoma. Esto sería a través de la entrega de cápsulas con una interfaz amigable que facilite la navegación para que los alumnos puedan visitarlas todas las veces que necesiten. A la vez, se realizará un seguimiento del tiempo empleado en una sesión de trabajo y el recorrido de los estudiantes en la guía. El monitoreo de la actividad de los estudiantes en la plataforma genera información relevante para que los profesores puedan realizar análisis de los avances de sus estudiantes en el curso y realizar retroalimentación respecto a éstas. Además de lo anterior, la plataforma cuenta con una estructura escalable, evolutiva y multi idioma, permitiendo en un futuro utilizarla en otras universidades donde se enseñe dinámica de sistema, dando la posibilidad de traducirla al inglés para ser empleada en otras instituciones alrededor del mundo.

El desarrollo de la plataforma fue a través de metodología ágil para iterar constantemente, consiguiendo versiones de prueba a la brevedad y con esto recibiendo la retroalimentación correspondiente para generar los cambios necesarios en la plataforma, mejorando el resultado final que se aspiraba conseguir.

Bajo la rendición normal del curso dinámica de sistemas II en base al calendario académico 2022, se realizó una primera etapa de implementación en

el módulo. Durante esta etapa solamente se contó con idioma español, un número de cápsulas preparadas y la capacidad de monitorear los avances de los estudiantes que utilizaron la plataforma. La información recopilada por el uso del sistema fue analizada para medir el progreso de los estudiantes y comparada con el avance de estos en el contenido del curso en años anteriores.

Los objetivos definidos para este proyecto son:

Objetivo general:

Implementar una plataforma web, usando los principios de experiencia de usuario, que sirva como guía interactiva para el aprendizaje de modelado en dinámica de sistemas.

Objetivos específicos:

1. Realizar un diagnóstico a partir del caso de estudio.
2. Establecer los requerimientos, el diseño y modelo de datos a emplear para el desarrollo.
3. Desarrollar la plataforma web con los principios de experiencia de usuario.
4. Implementar la plataforma en el curso dinámica de sistemas II.
5. Medir el rendimiento de la plataforma en el curso de dinámica de sistemas II.

   Observación: Los objetivos 4 y 5 quedan sujetos a la rendición del curso dinámica de sistemas II en condiciones normales (de acuerdo con el calendario académico 2022).

2. Marco Teórico

2.1 Sistema de información

Los sistemas de información, basados en la definición de Andreu, Ricart y Valor (1991), se entienden por una serie de procesos que trabajan sobre una base de datos que responde a las necesidades de una empresa, y permite recopilar, elaborar y distribuir información necesaria para el funcionamiento de dicha empresa y sus actividades de control correspondientes, ayudando de esta forma al proceso de toma de decisiones para realizar sus operaciones de negocio definidas por su estrategia.

Otra definición es la de Bravo (1994), la que propone que un sistema de información es:

> Un conjunto de elementos o componentes, los llamados recursos de información (personas, *hardware*, *software*, datos e información), interrelacionados (organización y estructura de roles intencionadas) para la consecución de fines comunes (información o producto final oportuno, de calidad y en cantidad adecuadas, en primera instancia, y los propios fines de la empresa en último término), dadas unas condiciones ambientales internas y externas a la compañía, cuya consecución se puede detectar mediante una adecuada información de retroalimentación y unos apropiados mecanismos de control. (p.64)

Con base en estas dos definiciones se puede concluir que un sistema de información es un conjunto de elementos y/o procesos, los cuales consideran personas, equipos informáticos, datos e información. Estos utilizados para la búsqueda de objetivos y toma de decisiones.

2.1.1 Características

Según lo propuesto por Laudon y Laudon (2016) definen un sistema de información como *"componentes interrelacionados que trabajan en conjunto para recolectar, procesar, almacenar y diseminar información para apoyar la toma de decisiones, la coordinación, el control, el análisis y la visualización en una organización" (p.16).* Con base en esta definición y lo propuesto por Hernández (2003), los componentes básicos de los sistemas de información comprenden personas/usuarios, infraestructura física/equipos e información/datos, y estos datos en cuestión se: *"Almacena, procesa y transforma para obtener como resultado final información" (p.1).*

En conjunto a lo previo, Hernández (2003) además propone que esta información final busca ciertos objetivos:

- Apoyar los objetivos y estrategias de la empresa
- Proporcionar información para el control de la totalidad de actividades de la empresa
- Adaptar las necesidades de información a la evolución de la empresa
- Interactuar con los diferentes agentes de la organización

Laudon y Laudon (2016) de igual manera proponen que los sistemas de información se agrupan según su utilidad en los cuatro niveles básicos de la organización:

- Nivel operativo: comprende las operaciones diarias de la organización.
- Nivel del conocimiento: afecta a los encargados del manejo de la información.
- Nivel administrativo: incumbente a los gerentes medios de la organización.
- Nivel estratégico: la alta dirección de la empresa.

En base a estos niveles, Laudon y Laudon (2016) definen la siguiente clasificación de sistemas de información:

Tabla 1: Clasificación de sistemas de información. Adaptado de Laudon y Laudon (2016)

Tipo de Sistema de Información	Definición	Nivel
Sistemas de Procesamiento de Operaciones (SPO)	Encargados de la administración de aquellas operaciones diarias de rutina necesarias en la gestión empresarial.	Operativo
Sistemas de Trabajo del Conocimiento (STC)	Encargados de apoyar a los agentes que manejan información en la creación e integración de nuevos conocimientos para la empresa.	Conocimiento
Sistemas de automatización en la oficina (SAO)	Empleados para incrementar la productividad de los empleados que manejan la información en los niveles inferiores de la organización.	Conocimiento
Sistemas de información para la administración (SIA)	Empleados en el proceso de planificación, control y toma de decisiones proporcionando informes sobre las actividades ordinarias.	Administrativo
Sistemas para el soporte de decisiones (SSD)	Ayudan a los distintos usuarios en el proceso de toma de decisiones, a la hora de utilizar diferentes datos y modelos para la resolución de problemas no estructurados.	Administrativo
Sistemas de Soporte Gerencial (SSG)	Diseñados para tomar decisiones estratégicas mediante el empleo de gráficos y comunicaciones avanzadas.	Estratégico

2.1.2 Ventajas y desventajas

La utilización de sistemas de información en una organización trae consigo una amplia variedad de ventajas para las operaciones de esta. En base a lo planteado por Hamidian y Ospino (2015), se destacan las siguientes ventajas y desventajas:

- **Ventajas:**

- Integración de las áreas que componen la organización, tecnología y herramientas.
- Proporcionar valor y aumento en efectividad de la operación de la organización.
- Optimización de recursos y riesgo.
- Mayor control en las actividades de la organización.
- Mayor seguridad y cumplimiento de reglas normativas.

- **Desventajas:**

- Tiempo empleado en su implementación.
- Resistencia al cambio por parte de sus usuarios.
- Problemas técnicos (fallas de *hardware* o *software*).

2.2 Aplicación Web

Las aplicaciones web son programas capaces de ejecutarse dentro de un navegador web y hoy en día han cobrado gran importancia con la diversificación de la internet. Mora (2001) define una aplicación web como:

> Un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador o visualizador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (HyperText Transfer Protocol (HTTP)) están estandarizados y no han de ser creados por el programador de aplicaciones. (p.8)

Estas funcionan en base a un navegador conectado a internet, pues deben comunicarse activamente con un servidor para ejecutar sus funcionalidades. Esta clase de aplicaciones usan tres tipos de tecnologías

básicas para su desarrollo, estas son: HTML, CSS y JavaScript. (Llamuca, Vera y Tapia, 2021, p.3)

2.2.1 Características

Basándose en las definiciones de Aplicaciones Web presentadas por diversos autores, se pueden identificar los siguientes componentes o características:

- **Cliente**

  Es la interfaz con la que el usuario interactúa para utilizar la aplicación web, generalmente son navegadores web como Google Chrome o Mozilla Firefox. Mora (2001) los define como un programa con el que el usuario interactúa para solicitar a un servidor la información que este desea obtener a través de un protocolo HTTP. El lado del cliente de una aplicación web está formada por código HTML (HyperText Markup Language) que construye la página web, junto a código ejecutable en lenguaje de script de navegador (JavaScript). (Mora, 2001, p.8)

- **Servidor**

  Un servidor web es un programa que se encuentra permanentemente a la espera de solicitudes de conexión a través del protocolo HTTP de parte de los clientes web. (Mora, 2001, p.9)

  Según la definición de los autores, un servidor corresponde a la contraparte del "cliente" en un esquema de aplicación web, y se encuentra a la espera de una conexión para responder una determinada solicitud. Esta solicitud puede devolver cierto contenido almacenado en el servidor, el cual se explica como:

  "Contiene páginas con contenido sin determinar, parcialmente o en su totalidad. El contenido final de una página se determina sólo cuando el usuario solicita una página del servidor Web." (Adobe Inc., 2021)

  La Figura 1: Esquema básico de una aplicación Web. Basado en Mora (2001) ilustra de manera simple la interacción entre los tres actores de una aplicación Web:

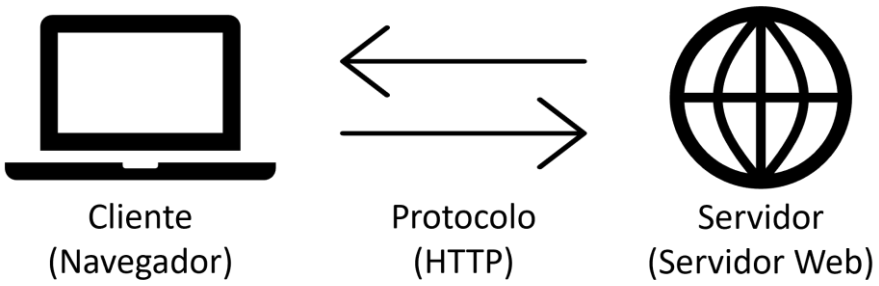

Figura 1: Esquema básico de una aplicación Web. Basado en Mora (2001)

- **Protocolos**

Los protocolos web son definidos por el Centro de investigación de la Universidad de Chile como:

> Desde un punto de vista más técnico, uno necesita un protocolo que permita enviar y traer información en HTML desde un lugar (sitio) a otro en esta gigantesca red que es la Web. El protocolo HTTP (sigla del inglés Hyper Text Transfer Protocol) […] es un protocolo de transmisión entre clientes y servidores. […] Entre ellos puede haber varios intermediarios, como *proxies*, *gateways* y túneles. A través de instrucciones simples, pero poderosas, el cliente indica al servidor qué acciones realizar para recibir o entregar datos. (Centro de Investigación de la Web, 2008, p.13)

- **Base de datos**

Camps (2005) describe una base de datos como:

> "*Un conjunto estructurado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas*" (p.8).

> "*Las bases de datos son el método preferido para el almacenamiento estructurado de datos. Una base de datos contiene unos datos que, en cada momento, deben reflejar la realidad o, más concretamente, la situación de una porción del mundo real*" (p.5).

Es decir, las bases de datos corresponden a un conjunto estructurado de datos interrelacionados, los cuales deberían ser posibles de usar y de reflejar información recopilada de la realidad.

- **HTML**

  Es un lenguaje estándar de etiquetas para representar información en un cliente Web. Gauchat (2017) lo explica como un lenguaje compuesto por etiquetas definidas por un nombre rodeadas de paréntesis angulares, estos paréntesis marcan el alcance de la etiqueta y su nombre define el tipo de contenido que representa.

  Es utilizado para generar la estructura de una página web y establecer un contenido estático.

- **PHP**

  PHP es un lenguaje de programación Web que permite la codificación de aplicaciones dinámicas. The PHP Group (2022) describe PHP como un lenguaje de script de código abierto diseñado para la web que puede ser incorporado dentro de código HTML. Su sintaxis está basada en C, Java y Perl, resultando sencillo de aprender por su similitud. Su objetivo principal es permitir que los desarrolladores puedan crear páginas web dinámicas rápidamente.

- **CSS**

  > CSS es un lenguaje que facilita instrucciones que podemos usar para asignar estilos a los elementos HTML, como colores, tipos de letra, tamaños, etc. Los estilos se deben definir con CSS y luego asignar a los elementos hasta que logramos el diseño visual que queremos para nuestra página. (Gauchat, 2017, p.83)

  Tal como está planteado en la definición, se utiliza para la personalización de los elementos de una página y con ello obtener algún diseño en específico.

- **JavaScript**

Es un lenguaje de programación que permite la ejecución de código dentro de un cliente Web. Gauchat (2017) lo explica como un lenguaje de programación utilizado para procesar información y manejar documentos HTML. Permite la ejecución de código secuencial para indicarle al sistema lo que deseamos que haga, como una operación matemática o asignar un valor a un elemento. Cuando un navegador encuentra este tipo de código en el documento, ejecuta las instrucciones al inmediatamente, mostrando cualquier cambio realizado al documento en pantalla.

Incorporar este lenguaje y sus funciones permiten generar dinamismo en la página, además de la capacidad de implementar algoritmos para ejecutar ciertas operaciones de variados tipos.

2.2.2 Tipos de aplicaciones web.

- **Estáticas**

"Las páginas estáticas son aquellas que existen todo el tiempo en un archivo en algún servidor Web." (Centro de Investigación de la Web, 2008, p.24)

"Página Web que el servidor de aplicaciones no modifica antes de que la página se envíe a un navegador." (Adobe Inc., 2021)

Considerando las anteriores definiciones, se puede dar cuenta de que este tipo de aplicaciones y páginas web son elementos estáticos los cuales fueron definidos durante su desarrollo y que no sufren modificaciones al momento de que el usuario tenga acceso a estos.

- **Progressive Web Apps (PWA)**

Las PWA son aplicaciones web desarrolladas con una serie de tecnologías específicas y patrones estándar que les permiten aprovechar las funciones de las aplicaciones nativas y web. Por ejemplo, las aplicaciones web son más fáciles de detectar que las aplicaciones nativas; es mucho más fácil y rápido visitar un sitio web

que instalar una aplicación, y también puedes compartir aplicaciones web simplemente enviando un enlace. (MDN contributors, 2020)

- **Públicas**

"Las páginas públicas son las que todas las personas pueden ver" (Centro de Investigación de la Web, 2008, p.25) Como se propone, las páginas web públicas son de libre acceso para cualquier usuario que quiera conocer su contenido y accesibles desde cualquier lugar.

- **Privadas**

Según la definición entregada por el Centro de Investigación de la Web (2008), las páginas web privadas son aquellas que se encuentran protegidas por una clave o se encuentran en una intranet. Por lo que un grupo específico de usuarios tiene acceso a estas páginas o tipo de aplicaciones. Además, con base en Mora (2001):

> Una intranet es una red de ordenadores basada en los protocolos que gobiernan Internet (TCP/IP) que pertenece a una organización y que es accesible únicamente por los miembros de la organización, empleados u otras personas con autorización. Una intranet puede estar o no conectada a Internet. Un sitio web en una intranet es y actúa como cualquier otro sitio web, pero los cortafuegos (*firewall*) lo protegen de accesos no autorizados. (p.12)

Con lo anterior, ejemplos de este tipo de páginas web pueden ser sitios Intranet que las universidades provean a sus alumnos u organizaciones que trabajen desde un sistema interno.

2.3 Ingeniería de software

En base a la ISO/IEC 2382:2015 (2015) se define a la ingeniería de *software* como la: *"aplicación sistemática de conocimientos, métodos y experiencias científicas y tecnológicas al diseño, la implementación, las pruebas y la documentación del software para optimizar su producción, soporte y calidad"*.

Además, Sommerville (2011) define que la ingeniería de *software* como una disciplina de la ingeniería enfocada en todos los aspectos de la producción

de *software*, abarcando desde las especificaciones del sistema hasta el mantenimiento luego de que este entre en operaciones.

Considerando ambas definiciones, la ingeniería de *software* contempla un proceso que implica el conocimiento y los aspectos de la producción de *software* desde las bases de un sistema hasta su posterior mantenimiento y soporte. Es decir, que se aplica conocimiento y metodologías de desarrollo a lo largo del ciclo de vida del producto.

2.3.1 Etapas de ingeniería de software tradicional

Las metodologías tradicionales de ingeniería de *software* definen una serie de etapas, las cuales han variado conforme nuevas metodologías más modernas se han creado. Sommerville (2011) define cinco etapas pertenecientes a la metodología tradicional en cascada, tal como se aprecia en la Figura 2: Etapas de la metodología en cascada. Adaptado de Sommerville (2011).

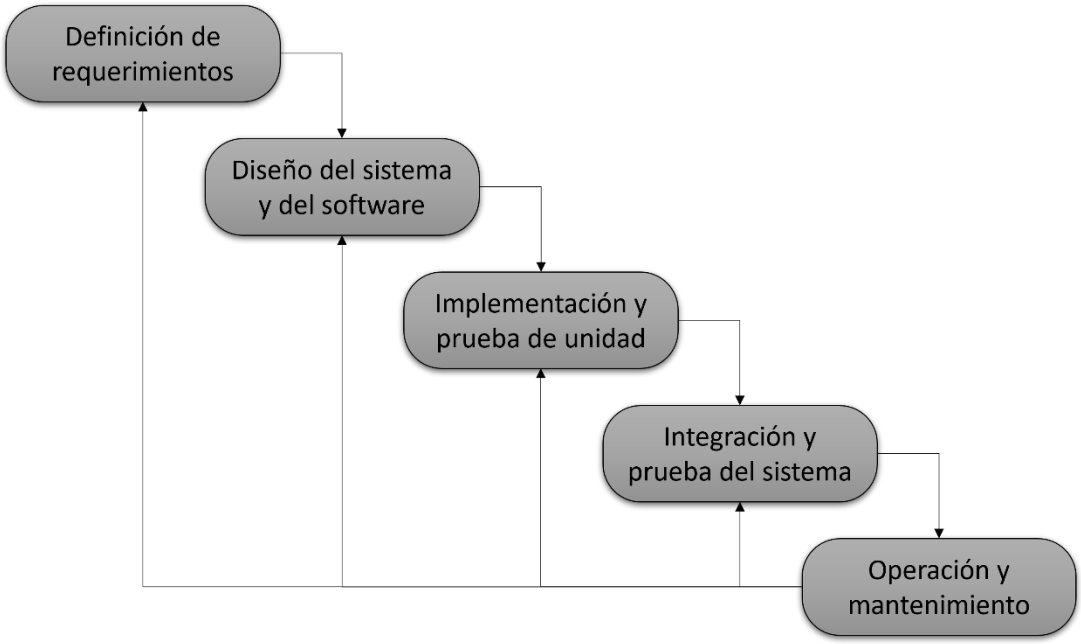

Figura 2: Etapas de la metodología en cascada. Adaptado de Sommerville (2011)

Sommerville (2011) define estas etapas como:

- **Definición de requerimientos**

  Los servicios, las restricciones y metas del sistema se establecen mediante consulta a los usuarios del sistema. Luego, se definen con detalle y sirven como una especificación del sistema.

- **Diseño del sistema y del *software***

  El proceso de diseño de sistemas asigna los requerimientos, para sistemas de *hardware* o de *software*, para establecer una arquitectura de sistema global. El diseño de *software* implica identificar y describir las abstracciones fundamentales del sistema de *software* y sus relaciones.

- **Implementación y prueba de unidad**

  Durante esta etapa, el diseño de *software* se realiza como un conjunto de programas o unidades de programa. La prueba de unidad consiste en verificar que cada unidad cumpla con su especificación.

- **Integración y prueba del sistema**

  Las unidades del programa o los programas individuales se integran y prueban como un sistema complejo para asegurarse de que se cumplan los requerimientos de *software*. Después de probarlo, se libera el sistema de *software* al cliente.

- **Operación y mantenimiento**

  Esta es la fase más larga del ciclo de vida, donde el sistema se instala y se pone en práctica. El mantenimiento incluye corregir los errores que no se detectaron en etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema e incrementar los servicios del sistema conforme se descubren nuevos requerimientos.

2.3.2 Metodologías tradicionales

En base a lo propuesto por Sommerville (2011) respecto al proceso de desarrollo de *software*, propone que:

- **Modelo en Cascada**

  Éste toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y, luego, los representa como fases separadas del proceso, tal como especificación de requerimientos, diseño de *software*, implementación, pruebas, etcétera.

- **Modelo de proceso incremental**

Este enfoque vincula las actividades de especificación, desarrollo y validación. El sistema se desarrolla como una serie de versiones (incrementos), y cada versión añade funcionalidad a la versión anterior. Durante este proceso, el desarrollo se genera en base a incrementos como está planteado, estos incrementos gradualmente incorporan las funcionalidades que finalmente componen el sistema en desarrollo.

- **Ingeniería de software orientada a la reutilización**

Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en la integración de estos componentes en un sistema, en vez de desarrollarlo desde cero. Es decir, el desarrollo se basa en incorporar código reutilizable, adaptándolo a las necesidades del sistema y reduciendo de este modo los costos de producción asociados al *software*.

2.4 Ingeniería de requerimientos

Sommerville (2011) explica que los requerimientos de un sistema son las descripciones que lo que dicho sistema debe hacer, refiriéndose a los servicios que ofrece y las restricciones que posee. Estos requerimientos reflejan las necesidades de un cliente que deben ser satisfecha a través del sistema. El proceso de descubrir, analizar, documentar y verificar estas funcionalidades y limitaciones se le llama ingeniería de requerimientos.

Basado en la definición del autor, la ingeniería de requerimientos es el proceso de comprensión e identificación de las necesidades de los usuarios y todas las partes involucradas, que un *software* debe satisfacer para dar solución a un problema en específico.

Complementando, Hull, Jackson & Dick. (2005) definen un requerimiento como: “L*a base para todo proyecto, definen lo que las partes interesadas – usuarios, clientes, proveedores, desarrolladores, negocios - necesitan de un potencial nuevo sistema y también lo que el sistema debe hacer para satisfacer esa necesidad.”* (p.2)

Este es considerado una de las etapas más importantes dentro del ciclo del desarrollo de *software* por diversos autores, pues la calidad del proceso de ingeniería de requerimientos influye considerablemente en las posibilidades de éxito que el proyecto pueda tener. A continuación, Hull et al. (2005) presenta una tabla comparativa de las principales causas del fracaso de un proyecto de desarrollo de *software*:

Tabla 2: Razones del fracaso de un proyecto. Basada en Hull et al. (2005)

Razones del fracaso de un proyecto	
Requerimientos incompletos	13.1%
Falta de participación del usuario	12.4%
Falta de recursos	10.6%
Expectativas no realistas	9.9%
Falta de apoyo ejecutivo	9.3%
Cambio en los requerimientos/especificaciones	8.7%
Falta de planeación	8.1%
Ya no se necesitaba	7.5%

Analizando la información contenida en la tabla, se observa que la principal razón del fracaso corresponde a un mal proceso de identificación de requerimientos y ocurre en el 13,1% de los casos. Junto al otro motivo asociado a requerimientos de *software* (Cambio en los requerimientos/especificaciones), se tiene que la razón del fracaso del 21,8% de los proyectos de desarrollo de *software* está vinculado a la ingeniería de requerimientos.

2.4.1 Tipos de requerimientos

Sommerville (2011) clasifica los requerimientos en dos tipos:

- **Requerimientos funcionales**

Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos

casos, los requerimientos funcionales también explican lo que no debe hacer el sistema. (Sommerville, 2011, p.84-85)

Se da a entender que este tipo de requerimientos apuntan a las funciones que debe desempeñar el sistema con base en los casos de uso en los que interactúa el usuario con este.

- **Requerimientos no funcionales**

Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema. (Sommerville, 2011, p.85)

En este caso, con lo propuesto, los requerimientos no funcionales se enfocan en elementos que no apuntan al uso del sistema, sino, a elementos y especificaciones técnicas que debe cumplir el sistema fuera de las funcionalidades que debe contener.

2.4.2 Beneficios

La ingeniería de requerimientos ofrece una variedad de beneficios al mejorar la calidad de productos en desarrollo, como proponen Sawyer, Sommerville & Viller (1999) dentro de estos se consideran:

- El documento de requisitos: estructurar y organizar el documento de requisitos para comunicar eficazmente los requisitos a los clientes, los gestores y los desarrolladores.
- Obtención de requisitos: Adquisición de requisitos y restricciones de los interesados en el sistema, el dominio de la aplicación y los entornos operativos y organizativos del sistema.
- Análisis y negociación de requisitos: Identificación y resolución de los problemas que surgen de los requisitos obtenidos.
- Describir los requisitos: Redactar los requisitos para facilitar la comprensión de los lectores.

- Modelización de sistemas: Desarrollo de modelos conceptuales para ayudar a la comprensión y el análisis de los requisitos y sus implicaciones para el sistema propuesto.
- Validación de requisitos: Establecer procedimientos para comprobar la corrección, integridad, coherencia y compatibilidad. Garantizar que los requisitos son verificables y que se cumplen las normas de calidad.
- Gestión de requisitos: Gestión de la información sobre los requisitos a lo largo del ciclo de vida del desarrollo.
- Ingeniería de requisitos para sistemas críticos: Prácticas para tratar los requisitos de los sistemas críticos, como los requisitos de seguridad o fiabilidad.

Estos beneficios aportan durante todo el ciclo de vida del desarrollo, además de que ofrecen la utilidad para comunicarse con los distintos usuarios, la capacidad de resolver problemas de requisitos y con ello lograr entregar al usuario lo que requiere como producto.

2.5 Frameworks

En base a la definición de Galindo y Camps (2010), un *framework es* un conjunto de clases colaborativas que construyen un diseño reusable para un tipo específico de *software*. Estos entregan una arquitectura en base a clases abstractas, definiendo sus responsabilidades y colaboraciones. El desarrollador construye la aplicación en base a subclases y componiendo instancias a partir de las clases definidas por el *framework*.

2.5.1 Ventajas y Desventajas

Utilizar *frameworks* en el proceso de desarrollo implica la obtención tanto de beneficios como desventajas, el seguir una arquitectura de desarrollo con estos marcos permite seguir ciertos estándares, lo cual según Galindo y Camps (2010) ofrece:

- **Ventajas**
  - Minimizar tiempos de desarrollo.
  - Reduce los riesgos del desarrollo.
  - Proporciona una arquitectura consistente entre aplicaciones.

- **Desventajas**
  - Limitación de la flexibilidad.
  - Dificultad de aprendizaje.
  - Reducción de la creatividad.

2.5.2 Frameworks de desarrollo web.

Sierra, Acosta, Ariza y Salas (2013) define los *frameworks* de desarrollo web como herramientas que entregan a los desarrolladores una base sólida, a través de la entrega de un esquema o patrón estandarizado, para el desarrollo e implementación de una aplicación web.

Así mismo, Sierra et al. (2013) señala que: *"Los frameworks poseen características que satisfacen en su gran mayoría a todos los programadores web según el estilo de desarrollo que deseen. Ahora bien, existen frameworks con todo tipo de características como la seguridad, robustez, facilidades de uso" (p.1).*

A continuación, se hará un breve repaso a una selección de *frameworks* de desarrollo web comúnmente utilizados:

- **Laravel**

Este es un *framework* web *open source* basado en el lenguaje PHP que permite la creación de código elegante, simple y claro. Su base es la creencia de que el desarrollo debe ser una experiencia creativa y disfrutable para que sea verdaderamente satisfactoria. (Laravel LLC, 2022)

Laravel se encuentra en constante desarrollo por su comunidad, y ofrece una amplia variedad de *plugins* que aumentan aún más sus funcionalidades.

- **CodeIgniter**

CodeIgniter es un *framework* web *open source* basado en el lenguaje PHP desarrollado por la empresa EllisLab. Permite la creación de aplicaciones con una complejidad y configuración mínima, con un alto rendimiento respecto a otros *frameworks* basados en PHP. (CodeIgniter Foundation, 2022)

Como plantean en su página web, está basado en el Modelo-Vista-Controlador (MVC) pero no fuerza al usuario a dar uso de este modelo para la creación de páginas web.

- **Django**

Como lo definen en su página web (Django Software Foundation, 2022) es un "*Framework web open source basado en el lenguaje Python que permite un desarrollo rápido, además de un diseño pragmático y limpio*" además de esto proponen la utilización de Django por características como:

- Ridículamente rápido
- Completamente cargado
- Tranquilizadoramente seguro
- Excesivamente escalable
- Increíblemente versátil

Este *framework* además está basado en MVC con cambios en la forma de llamar a ciertos componentes de este patrón de diseño.

2.5.3 Elección del Framework a utilizar.

Para el desarrollo de esta tesis se empleó el *framework* de desarrollo web Laravel. Como plantea Sierra et al. (2017), *"Este Framework usa el paradigma orientado a objetos, permite el uso del patrón MVC, ORM" (p.6).*

Por el uso del paradigma orientado a objetos, el producto de *software* desarrollado permite la implementación de nodos y un modelo de árbol para la distribución de las cápsulas de aprendizaje y la interrelación de conceptos.

Las desventajas de "*Limitación de Flexibilidad*" y "*Reducción de Creatividad*" se ven fuertemente reducidas al utilizar Laravel, puesto que este *framework* ofrece una amplia variedad de *plugins* que amplían en gran medida su flexibilidad y la capacidad de crear cosas nuevas con ellos.

2.6 Dinámica de Sistemas

Con base en lo planteado por Schaffernicht (2021) y la *System Dynamics Society*, la Dinámica de Sistemas es una disciplina que permite analizar y

modelar el comportamiento de entornos complejos y dinámicos dentro de un determinado periodo de tiempo. *"Es una aproximación de modelado basado en la variación de acumuladores y flujos a lo largo del tiempo."* (Strapasson et al., 2022)

2.6.1 ¿Para qué sirve?

Tal como propone la *System Dynamics Society* (2022):

> El objetivo principal es ayudar a las personas a tomar mejores decisiones cuando se enfrentan a sistemas complejos y dinámicos. El enfoque proporciona métodos y herramientas para modelar y analizar sistemas dinámicos. Los resultados del modelo pueden utilizarse para comunicar conclusiones esenciales que ayuden a todos a comprender el comportamiento del sistema.

Con esto presente, su fin es el entendimiento de sistemas complejos y mejorar la toma de decisiones al afrontar estos escenarios.

2.6.2 Proceso de modelado

El proceso de modelado comprende una serie de componentes que interactúan entre ellos para simular el comportamiento de un sistema. Schaffernicht (2021) los describe de la siguiente manera:

- Acumulador (o stock): Tipo de variable que representa la cantidad o estado de algo en un determinado momento. Un acumulador solo puede ser afectado por un flujo.
- Flujos: Tipo de variable que representa el cambio de algo durante un intervalo de tiempo. Un flujo origen y destino de un flujo siempre es un acumulador, aunque este no siempre sea parte del modelo en estudio.
- Variable intermedia: Tipo de variable que representa un indicador, una razón (relación entre 2 o más variables), u otro paso intermedio para un cálculo.
- Vínculos causales: Representa una relación causa-efecto entre dos variables. Poseen una polaridad que puede ser positiva o negativa, y esta variable puede presentar un retraso: significando que afecta el modelo desde el inicio, o desde un momento determinado. No pueden

existir vínculos causales entre dos acumuladores o entre una variable intermedia y un acumulador, pues solo los flujos pueden cambiar un acumulador.

- Bucle de retroalimentación: Es una secuencia cerrada de variables y vínculos causales. Este conjunto comprende al menos un acumulador y un flujo. Un bucle puede presentar una polaridad reforzadora (genera aceleración) o compensadora (genera desaceleración).

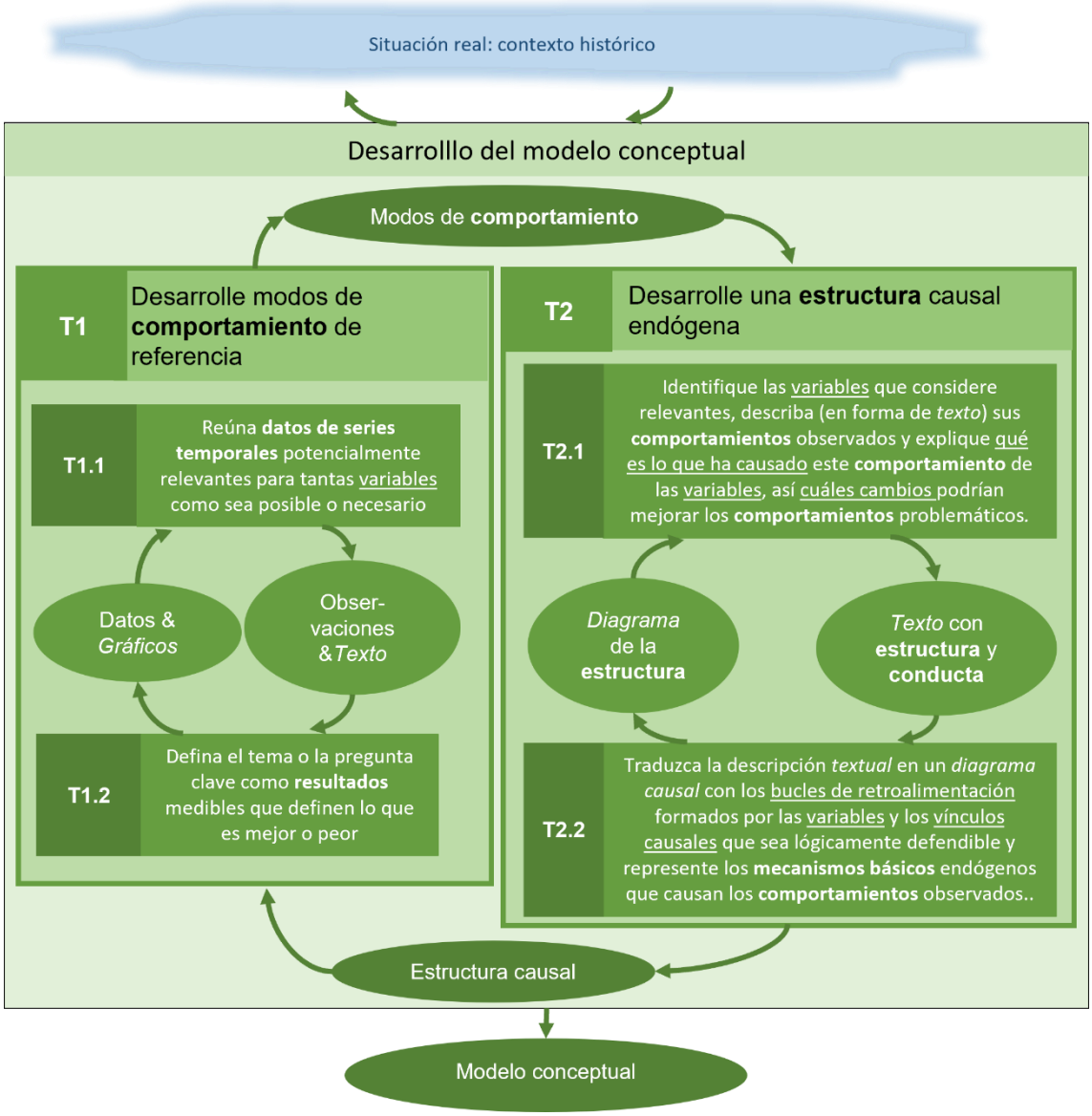

Figura 3: La conceptualización de modelos como conjunto de tareas interdependientes. Schaffernicht (2021)

El proceso de modelado contempla la realización de un modelo conceptual, este proceso tal como Schaffernicht (2021) propone, es un proceso iterativo que contempla varios ciclos basados en tareas:

En primer lugar, se debe definir un modo de comportamiento de referencia, esto se realiza a base de reunir datos de series de tiempo que contemplen una cantidad amplia de variables, luego de esto se procede a definir una pregunta clave con relación a resultados medibles. Durante este proceso, el comportamiento de referencia se basa en comprender cómo se comporta cierto sistema en un periodo de tiempo.

En segundo lugar, luego de desarrollar el modo de comportamiento de referencia, se desarrolla una estructura causal endógena, para esto se analizan las variables, identificando las relevantes en conjunto con su comportamiento. Este proceso de recolección de variables se implementa en un diagrama causal que, tal cual se presenta en la Figura 3: La conceptualización de modelos como conjunto de tareas interdependientes. Schaffernicht (2021), implementa *"los bucles de retroalimentación formados por las variables y los vínculos causales que sea lógicamente defendible y represente los mecanismos básicos endógenos que causan los comportamientos observados"*, con esto se crea el diagrama de la estructura.

Finalmente, tal como se planteó en un inicio, este proceso iterativo puede volver a realizarse, esta vez con una estructura causal construida (el diagrama) se analiza nuevamente el sistema, incorporando o eliminando variables que se vean afectadas y mejorando el diagrama para obtener un mejor análisis del sistema en cuestión. Por otra parte, se puede proceder con la estructura generada, utilizándola como modelo conceptual para efectuar los correspondientes análisis y tomar decisiones correspondientes al sistema bajo estudio.

2.7 Experiencia de usuario (UX)

La experiencia de usuario es definida por la ISO 9241-11:2018 (2018) como: "*Las percepciones y respuestas del usuario que resultan del uso o participación en el uso de un sistema, producto o servicio."*

Para López (2018), la finalidad del estudio de la experiencia de usuario es la de conocer las emociones y sensaciones que experimenta el usuario al interactuar con un determinado sistema o producto, y que conllevan a una percepción positiva o negativa de estos. *"Ésta depende no sólo de los factores relativos al diseño sino además de aspectos relativos a las emociones, sentimientos, construcción y transmisión de la marca, confiabilidad del producto, etc."* (López, 2018, p.12)

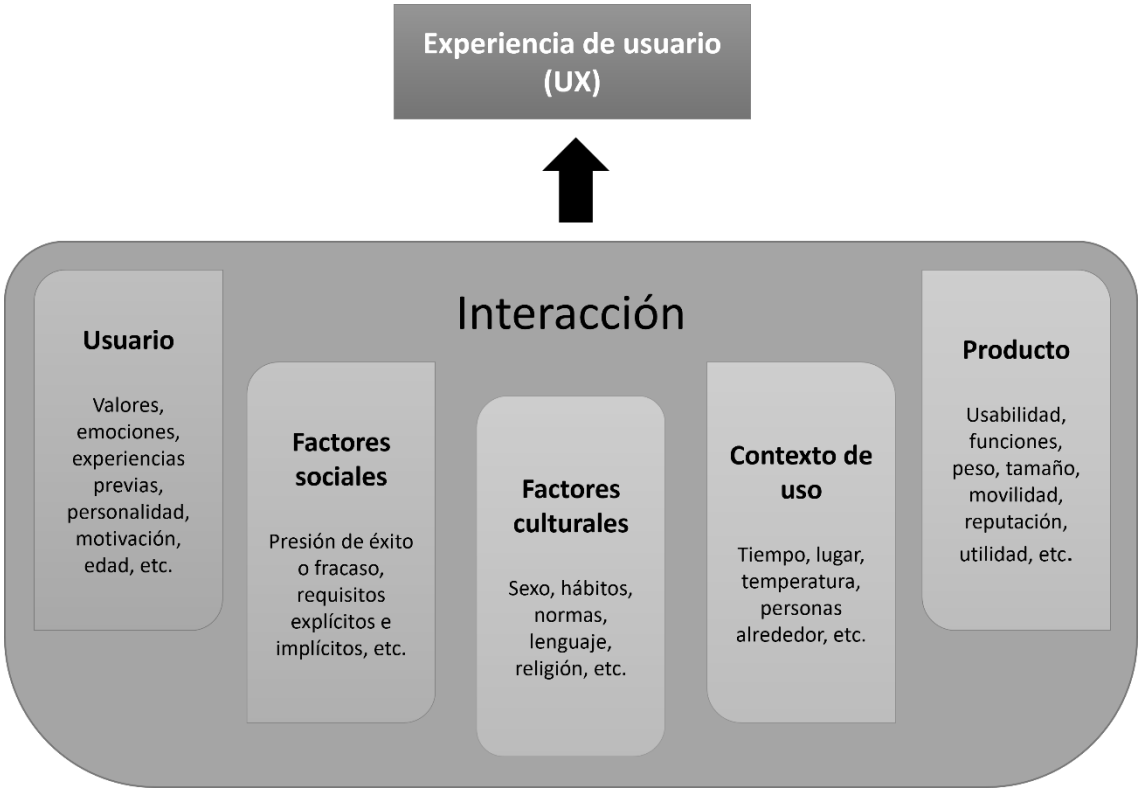

Figura 4: La experiencia del usuario se forma en la interacción con el usuario y el producto en el contexto particular, incluidos los factores sociales y culturales. Adaptado de Arhippainen & Tähti, (2003).

Basándose en la figura propuesta por (Arhippainen & Tähti, 2003) existen una variedad de factores los cuales su interacción influye en la UX, es decir, la experiencia misma varía en dependencia del contexto del usuario, además de su entorno. Con esto ninguna experiencia es igual para cada usuario al momento de interactuar con un producto.

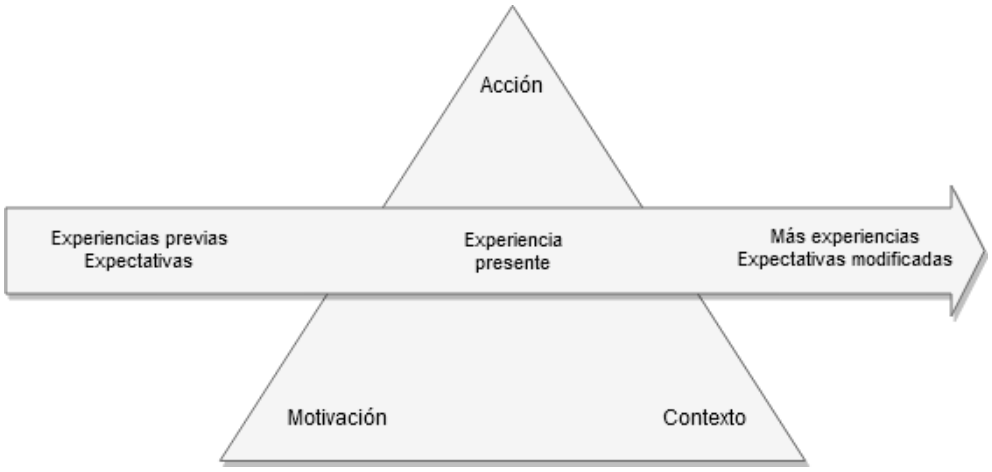

Figura 5: Un modelo conceptual de la experiencia del usuario. Adaptado de Kankainen (2002).

Kankainen (2002) propone que para la UX además de los factores de contexto, se incorporan la motivación y la acción de los usuarios, incluyendo además las expectativas a futuro que se generen en el usuario. En el caso de Arhippainen & Tähti (2003) también consideran las experiencias previas, sin embargo, en este caso consideran la temporalidad tanto de experiencias previas como futuras experiencias y las expectativas generadas.

2.7.1 Usabilidad

> La usabilidad es un atributo de calidad de un producto que se refiere sencillamente a su facilidad de uso. No se trata de un atributo universal, ya que un producto será usable si lo es para su audiencia específica y para el propósito específico con el que fue diseñado. (Montero, 2015)

Con esta definición en mente, Montero (2015) identifica dos dimensiones en la usabilidad:

- **Dimensión objetiva o inherente**

  Esta dimensión puede ser medida a través de la observación, se puede descomponer en los siguientes atributos:

  - Facilidad de aprendizaje: ¿Qué tan fácil resulta para el usuario realizar tareas básicas por primera vez con el diseño del producto?
  - Eficiencia: ¿Cuánto tiempo emplea el usuario en completar las tareas al aprender su funcionamiento básico?

- Cualidad de ser recordado: Luego de no usar el diseño por un tiempo, ¿Cuánto tiempo demoran en volver a aprender lo necesario para usar el producto eficientemente?
- Eficacia: Al realizar una tarea, ¿cuántos errores comete el usuario, y qué tan graves son sus consecuencias?, ¿cómo de rápido el usuario puede deshacer las consecuencias de sus errores?

- **Dimensión subjetiva o aparente**

  Esta dimensión se basa en la percepción del usuario:

  - Satisfacción: ¿Qué tan agradable y sencillo le parece al usuario realizar las tareas?

2.7.2 Beneficios

Según lo propuesto por Gualtieri (2009) una gran experiencia de usuario debería ser:

- Útil: Los clientes pueden cumplir sus metas.
- Usable: Los clientes pueden realizar sus tareas fácilmente.
- Deseable: Los clientes disfrutan de la experiencia.

Estos contemplan los beneficios de una buena UX, puesto que el enfoque es realizado en el usuario, considerando el ambiente en el cual se encuentra y las motivaciones de este.

2.8 Gamificación

La gamificación según Deterding, Dixon, Khaled y Nacke (2011) se propone como: "*El uso de elementos de diseño de juegos en contextos de no-juegos"* (p.10).

Otra definición es la propuesta por Teixes (2015) explica: *"La gamificación es la aplicación de recursos propios de los juegos (diseño, dinámicas, elementos, etc.) en contextos no lúdicos, con el fin de modificar los comportamientos de los individuos, actuando sobre su motivación, para la consecución de objetivos concretos"* (p.18).

En base a ambas definiciones se da a entender que la gamificación incorpora elementos de juego en contextos diferentes para motivar al usuario a conseguir objetivos.

2.8.1 Ventajas y desventajas

En base a lo propuesto por Prieto (2017), se pueden mencionar las siguientes ventajas de la gamificación en entornos de enseñanza:

- **Motivación**: La ludificación puede incrementar el atractivo de ciertas tareas académicas mejorando la calidad de enseñanza y aprendizaje.
- **Alfabetización tecnológica**: El uso de videojuegos y tareas gamificadas con las TIC favorece que el niño/a desarrolle habilidades en el manejo del ordenador, el *software* y las redes.
- **Mentalidad multitarea**: Es posible mejorar la capacidad de captar distintos detalles de una o varias pantallas lo cual supone una evolución en la lectura en pantallas y en el acceso general a la información digital.
- **Trabajo en equipo**: Los juegos actuales basados en las redes sociales facilitan la comunicación e intercambio con los demás.
- **Instrucción individualizada**: Cada alumno/a puede jugar y aprender por sí mismo siguiendo su propio ritmo.

Por otra parte, también clasifica las siguientes desventajas:

- **Elevado coste**: Conseguir videojuegos de calidad en un programa educativo resulta muy costoso.
- **Distracción y pérdida de tiempo**: Los juegos no desarrollan de forma suficiente habilidades valiosas desde el punto de vista educativo.
- **Inadecuada formación de valores**: Los alumnos son competitivos y desean ganar al sistema de cualquier forma dando lugar en muchas ocasiones a escasos o no deseados resultados de aprendizaje.

- **Equilibrio entre lo lúdico y lo formativo**: Es muy difícil encontrar el término medio que permita disponer de un juego atractivo donde se realice un aprendizaje efectivo desde el ámbito educativo.
- **Motivación efímera**: Las ganas de obtener premios y recompensas no perduran en el tiempo y terminan aburriendo una vez superada la novedad inicial.

2.8.2 Gamificación en Dinámica de sistemas

Cunico, Mollona y Aivazidou (2020) analizan la historia de la gamificación en dinámica de sistemas, y la definen como: *"El proceso de desarrollar y diseñar entornos de aprendizaje mediados o en base a medios para ser utilizados como herramientas de aprendizaje basados en el conocimiento de Dinámica de Sistemas y modelos formales*". (p.2)

Cunico et al. (2020) además mencionan que el acercamiento de la gamificación en el ámbito de dinámica de sistemas se enfoca principalmente en los simuladores. Estos son principalmente utilizados como herramientas de toma de decisiones con propósitos educativos. Con ellos el usuario puede generar una interacción y experimentar con los diferentes valores de un modelo para generar diversos comportamientos.

3. Metodología

Este capítulo contempla explicar la metodología utilizada para llevar a cabo el desarrollo del proyecto.

Un hecho a considerar es la naturaleza del proyecto la cual es aplicada, siendo este, el desarrollo de una aplicación web la cual sirve como guía para el aprendizaje de dinámica de sistemas. Además, el contenido de esta guía se compone por la información del documento realizado por Martin Schaffernicht “Guía práctica para la conceptualización y formalización de modelos de dinámica de sistemas”.

3.1 Metodologías ágiles

Las metodologías ágiles surgieron como nuevas técnicas para la gestión de proyectos, enfocadas en el cliente además del producto y evitando la documentación extensiva, tal como propone el manifiesto ágil (Beck K., Beedle M., Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R., Mellor S., Schwaber K., Sutherland J., y Thomas D., 2001).

Pressman (2010) identifica las metodologías ágiles por su enfoque en la satisfacción del cliente, logrando combinar diferentes aspectos como son la colaboración entre actores por ejemplo desarrolladores, clientes y usuarios finales. Además de considerar como el único producto importante el “incremento de *software*”, el cual es el producto entregado al cliente según coordinación con el mismo.

Respecto a metodologías ágiles en este proyecto, se mantuvieron como guía los principios del manifiesto ágil, destacando dos principios:

- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara. Puesto que la comunicación fue de los elementos más relevantes para tomar decisiones y compartir ideas.

- El *software* funcionando es la medida principal de progreso. Para lograr un seguimiento claro de las funciones y necesidades de la aplicación en desarrollo.

3.2 Metodologías empleadas

Existen variadas prácticas de metodologías ágiles que pueden ser implementadas dentro de proyectos de desarrollo de *software*. A continuación, se describen las metodologías utilizadas durante el desarrollo del proyecto y como fueron implementadas a lo largo de este.

3.2.1 Extreme programming (XP)

La programación extrema o *Extreme programming* es una práctica que como Beck (1999) propone: *"Más que planear, analizar y diseñar para un futuro lejano, XP explota la reducción en el costo de software cambiante para hacer todas estas actividades un poco a la vez, a través del desarrollo del software"*. Es decir, que las prácticas de XP a diferencia de otras metodologías, XP no realiza un trabajo de forma tan sistemática como lo son metodologías en cascada, se realiza trabajo de forma incremental lo que permite flexibilidad en el caso de ocurrir cambios en los requisitos, logrando con ello reducir costos de *software* al permitir correcciones en el trabajo realizado de ser necesario.

Extreme programming también considera reglas para el uso de esta metodología, Wells (2013) propone las siguientes reglas del XP:

- Planear
- Administrar
- Diseñar
- Codificar
- Probar

Durante el transcurso de este proyecto las reglas anteriormente nombradas se integraron durante la fase de desarrollo de la siguiente forma:

Planeación: Siguiendo esta regla se realizaron procedimientos y actividades de distinto tipo, considerándose diagnosticar el problema a resolver, generar historias de usuarios con base en las necesidades planteadas por el

cliente, realizar periódicamente lanzamientos con nuevas funcionalidades implementadas en el sistema desarrollado, planificar iteraciones dedicadas a funciones específicas del *software*, realizar planes de contingencia en base a problemáticas que pudieron surgir durante el desarrollo.

Administrar: con esta regla se realizaron ciertas recomendaciones al inicio del desarrollo como, por ejemplo, el trabajar en conjunto participando de videoconferencias de ser posible, el establecer ritmos de desarrollo que nos permitieran alcanzar ciertos objetivos cada semana correspondiente a funciones del sistema y corregir la metodología de ser necesario al estancarse o no funcionar bajo los ambientes en los cuales no encontramos. Otros procedimientos se formularon de forma más natural, por ejemplo, el realizar reuniones breves que nos permitieran entregar síntesis del trabajo realizado, manifestar problemas que nacieron durante el desarrollo o reuniones para tomar decisiones referentes tanto a funciones del sistema como al modelo de datos.

Diseñar: bajo esta regla se mantuvieron focos tales como, mantener el código con la mayor simpleza y claridad, lo que además implica refactorizarlo siempre y cuando fuera posible para mantener consistencia durante el desarrollo. Sumado a lo anterior, mantener diseños simples y funcionales para el usuario final, además del cliente, buscando con ello generar una buena experiencia. Un ejemplo de consistencia aplicado es la generación de rutas en el sistema, puesto que todas las vistas accesibles para los usuarios poseen rutas en español con la única excepción de la pantalla de trabajo de los alumnos que posee un nombre en inglés.

Codificar: siguiendo esta regla se establecieron ciertos procedimientos para trabajar, como, codificar en parejas cuando fuera posible, reduciendo de este modo los errores cometidos y evaluando en conjunto si los objetivos buscados están siendo obtenidos. El código desarrollado debe seguir ciertos estándares de escritura, manteniendo la consistencia a lo largo del proyecto. La integración del código era realizada por uno de nosotros, evitando incidentes relacionados a esto.

Probar: en este punto se siguió de forma importante el realizar pruebas constantes luego de integrar funciones, esto se debe a que las incorporaciones eran evaluadas según su funcionamiento y como se incorporaban al resto del sistema, evitando así generar errores entre los módulos del sistema.

Sumado a lo anterior, debemos considerar que como Wells (2013) explica: *"La programación extrema enfatiza el trabajo en equipo (…). El equipo se auto organiza alrededor de un problema para resolverlo de la forma más eficientemente posible"*. En este caso el trabajo en equipo es uno de los factores clave en conjunto a la comunicación, logrando con esto un trabajo eficiente y consiguiendo un producto de calidad que responda a las necesidades del cliente.

3.2.2 Kanban

Kanban es una metodología que implementa un cuadro con columnas a las cuales se incorporan tarjetas visuales, cada columna se asocia a un estado y las tarjetas establecen tareas específicas las cuales deben ser completadas.

Implementamos Kanban en el desarrollo utilizando la herramienta web Trello. En esta herramienta utilizamos 4 columnas diferentes para información relevante, conformadas por:

- Lista de tareas: como su nombre indica, incorporamos las tareas relevantes que se requerían para el desarrollo del producto de *software*, los documentos asociados y actividades por realizar.
- En proceso: en esta columna integramos tareas conforme el desarrollo se desempeñaba, añadiendo actividades según desarrollaba cada uno de los integrantes del equipo y detallando en cada tarjeta diferentes elementos que se debían resolver.
- Hecho: la columna correspondiente se compuso de las diferentes tareas que fueron completadas en etapa previa "En proceso", además incorporamos tarjetas a causa de hitos relevantes ocurridos en el desarrollo.
- Anotaciones: Este espacio fue utilizado para almacenar tarjetas extra que contienen datos relacionados a: el proyecto, el *software*,

enlaces de interés y recomendaciones realizadas por parte de interesados en el proyecto.

Distintos tipos de tarea complementaron la distribución del tablero, siendo el caso de tareas tipo “Reunión” uno relevante por su composición, integrando los tópicos correspondientes a la reunión que se asociaba. Otro tipo es el de “Tarea”, que corresponde a actividades a realizar. Sumado a los anteriores el tipo “Entrega” corresponde a tarjetas compuestas por evidencias de ciertas actividades. Los otros tipos correspondían a “Futuro” que son actividades del tipo “Tarea” las que no fueron necesarias para completar el desarrollo y “Pendiente” para aquellas que no podíamos controlar directamente para completar.

A continuación, la Figura 6: Tablero Kanban durante el desarrollo del proyecto. Creación propia, 2022., presenta un vistazo del tablero utilizado durante el desarrollo:

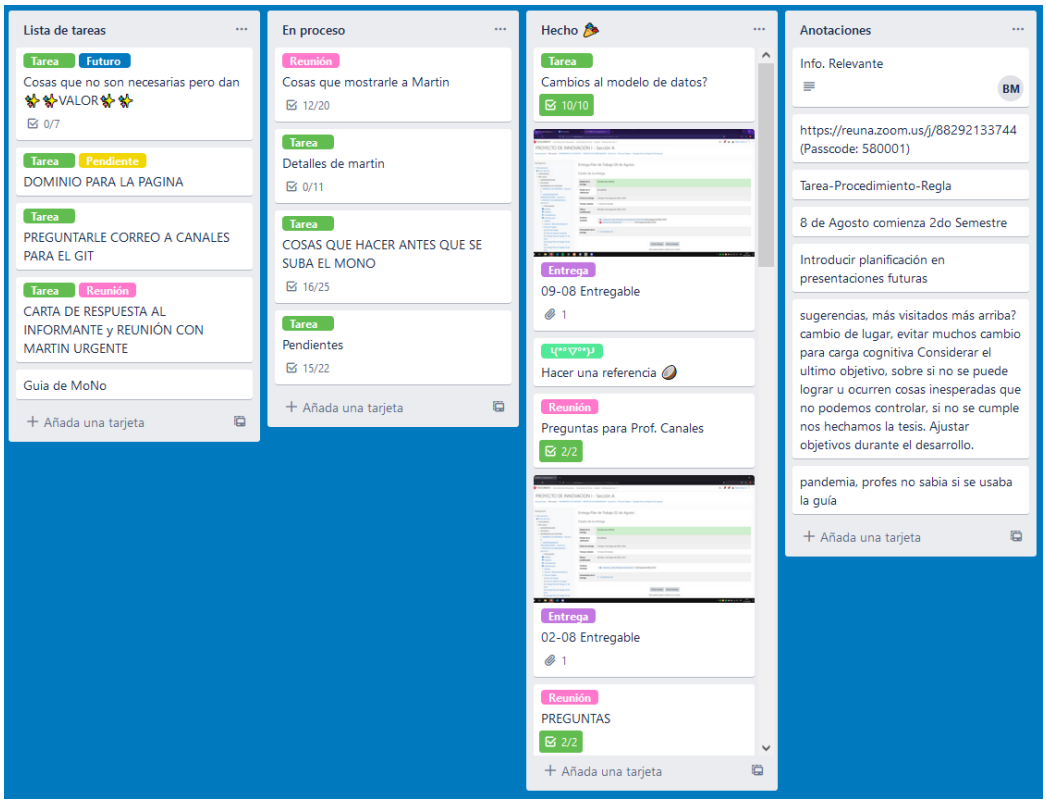

Figura 6: Tablero Kanban durante el desarrollo del proyecto. Creación propia, 2022.

3.3 Elección de metodología

Se implementaron metodologías ágiles justificados en la posibilidad de realizar retrabajo de ser necesario, ya que, considerando los nuevos requerimientos nacidos a lo largo del proyecto era necesario cambiar parte del desarrollo ya preparado con el objetivo de satisfacer las necesidades del cliente y la usabilidad de la plataforma. Otro motivo es la flexibilidad de implementar cambios comparados a una metodología en cascada, puesto que no existía un limitante para alterar código ya escrito que pudiera causar conflicto con las nuevas incorporaciones realizadas en la aplicación.

Durante el desarrollo de este proyecto, fue de vital importancia realizar ajustes rápidos en las estructuras y modelos planteados en un primer momento, por lo que la elección de esta metodología demostró ser la adecuada para enfrentar el desarrollo de esta tesis.

4. Presentación y análisis de los resultados

4.1 Fase de diagnóstico

Durante esta fase del desarrollo del proyecto, se llevaron a cabo varias reuniones tanto con la contraparte (profesor Martin Schaffernicht) como con el profesor guía. Luego de analizar los documentos generados por el profesor Schaffernicht y junto a la información recopilada, llegamos a las siguientes conclusiones:

- Existe un trabajo de investigación por parte del profesor Schaffernicht desde 2019, con la finalidad de mejorar el material sobre el modelado de trabajo conceptual. Luego de varias iteraciones en generaciones de estudiantes de ingeniería informática empresarial, el profesor llegó a la versión actual de la guía, contando de 53 páginas que contemplan la definición de tareas, procedimientos, axiomas y definiciones.
- Paralelamente, el profesor Schaffernicht en el curso de interfaces de usuario el año 2021, entregó el trabajo a sus estudiantes de modelar

un sistema guía para dinámica de sistemas, pero lamentablemente ninguno de los prototipos realizados cumplía con todos los requisitos descritos por el profesor.

- Por otra parte, la guía de Word debido a su formato no brindaba la ayuda requerida, pues presentaba diversas carencias, por mencionar:
  - La navegación a través del documento.
  - Falta de seguimiento en el progreso de los estudiantes.
  - Extensión del documento.
  - Facilidad de perderse en el documento.
  - Falta de orientación.
  - Sistema de códigos para navegar.
- El profesor Schaffernicht desarrolló una serie de modelos que representan la relación de los conceptos de la guía, lo que puede servir para la posterior creación de los modelos de datos y las cápsulas de aprendizaje.
- Producto de las medidas tomadas durante la pandemia, la guía nunca llegó a emplearse en clases presenciales. Esto perjudicó el desarrollo de la guía por parte del profesor Schaffernicht, así como el uso de ésta por parte de los estudiantes.
- Debido a las limitaciones planteadas por las clases en línea, resultó complicado para el profesor recibir retroalimentación respecto al uso que entrega la guía a los estudiantes, señalando que no podía saber si la guía era utilizada o no.

4.2 Fase de definición de requerimientos

Junto a los antecedentes recopilados y las reuniones realizadas con el profesor Schaffernicht y el profesor guía de este proyecto, se elaboró la siguiente lista de requerimientos para este proyecto:

Requerimientos funcionales:

- Revisión del avance de los alumnos en el curso.
- Creación y edición de cápsulas.

- Gestión de usuarios.
- No perder pantalla de trabajo.
- Roles de alumno, profesor y diseñador.
- Panel de información del curso.
- Panel de información de cápsulas.
- Soporte para múltiples idiomas.
- Estructura escalable.
- Seguimiento del avance de los alumnos.
- Búsqueda rápida de conceptos.
- Interfaz intuitiva.

Requerimientos no funcionales:

- Plataforma web.
- Interfaz vía navegador Web.
- Utiliza principios de experiencia de usuario.
- Disponibilidad en múltiples plataformas (A través de interfaz web)
- Disponibilidad en múltiples navegadores (Edge, Chrome, Firefox)
- Implementado en un servidor que el profesor Schaffernicht pueda manejar.
- *Framework* de desarrollo web PHP: Laravel.
- Elementos de gamificación.
- Tiempos de respuesta rápidos.

4.2.1 Tipos de usuario

Tomando en consideración las funcionalidades recabadas durante la etapa de definición de requerimientos, se detectaron los siguientes cuatro tipos de usuario que emplearán la plataforma:

- Administrador(a): Son los(as) responsables de la mantención y funcionamiento de la plataforma, así como de gestionar el registro de nuevos usuarios, manejar las instituciones inscritas y sus cursos respectivos, así como ayudar frente a cualquier eventualidad en la plataforma.

- Super usuario(a): Son los(as) responsables de cumplir funciones del rol de administrador, además de tener el poder de gestionar usuarios a un nivel más alto, permitiendo un control mayor sobre la administración en la plataforma.
- Diseñador(a): Son los(as) responsables de mantener en orden y actualizar la información de la guía contenida en los nodos y cápsulas. Pueden hacer cambios en la información y editar el mapa de navegación del sitio.
- Profesor(a): Son los(as) responsables de sus cursos particulares, pueden hacer un seguimiento y retroalimentar a sus alumnos según sus avances en el curso. Deben gestionar la inscripción, des inscripción y la creación de nuevos cursos para sus planes de estudio.
- Alumno(a): Son los(as) encargados de hacer uso de la plataforma para consultar la información que esta ofrece, pueden ver un resumen de su actividad en el curso, así como su avance en los contenidos de la guía.

4.2.2 Historias de usuario

En base a los requerimientos anteriores, se generaron las siguientes historias de usuario:

1.	Ver información de los alumnos.
Como profesor **quiero** revisar los alumnos de mi curso **para** ver su progreso en el curso.	

2.	Crear cápsulas de contenido.
Como diseñador **quiero** crear y editar cápsulas **para** modificar y actualizar la guía.	

3.	Gestionar usuarios.
Como administrador **quiero** gestionar usuarios **para** crearlos, modificarlos y eliminarlos según corresponda.	

4.	Mantener el espacio de trabajo.
Como usuario **quiero** mantener mi pantalla de trabajo **para** concentrarme en mi tarea y evitar pérdidas de trabajo.	

5.	Clasificar los usuarios en diferentes roles.
Como administrador **quiero** roles de alumno, profesor y diseñador **para** gestionar la plataforma y las acciones que puede realizar cada usuario.	

6.	Ver panel de información del curso.
Como profesor **quiero** un panel de información del curso **para** hacer seguimiento centralizado de los avances del curso.	

7.	Ver panel de información de cápsulas.
Como profesor **quiero** ver un panel de información de cada cápsula **para** obtener información relacionada al avance de los estudiantes.	

8.	Seleccionar idioma.
Como alumno o profesor **quiero** poder cambiar el idioma **para** acceder a la información en el idioma de mi preferencia.	

9.	Mantener una plataforma escalable.
Como administrador **quiero** una estructura escalable **para** ampliar el alcance de la plataforma y expandirla posteriormente.	

10.	Seguir el avance de los alumnos.
Como profesor **quiero** hacer seguimiento del avance de los alumnos **para** entregarles retroalimentación.	

11.	Buscar conceptos.
Como usuario **quiero** buscar fácilmente conceptos **para** entender la guía sin interrumpir mi trabajo y aprendizaje.	

12.	Poseer una interfaz intuitiva.
Como usuario **quiero** una interfaz intuitiva	

para no perder tiempo en mi trabajo.

13.	Múltiples conceptos en el mismo lugar.
Como estudiante **quiero** trabajar múltiples conceptos al mismo tiempo **para** entender el tema que estoy estudiando.	

4.3 Fase de diseño

La fase de diseño está enfocada en la implementación de los requisitos levantados para generar un diseño preliminar de la estructura de la plataforma. En este ámbito, se generó un modelo de datos y una serie de prototipos que se detallan a continuación:

4.3.1 Diseño de datos

En base a los requerimientos presentados previamente en conjunto con el profesor Martin Schaffernicht, se diseñó el siguiente modelo de datos que soportaría las funcionalidades de la plataforma. El diseño de la base de datos relacional se ilustra en la Figura 7: Modelo de datos preliminar. Creación propia, 2022.

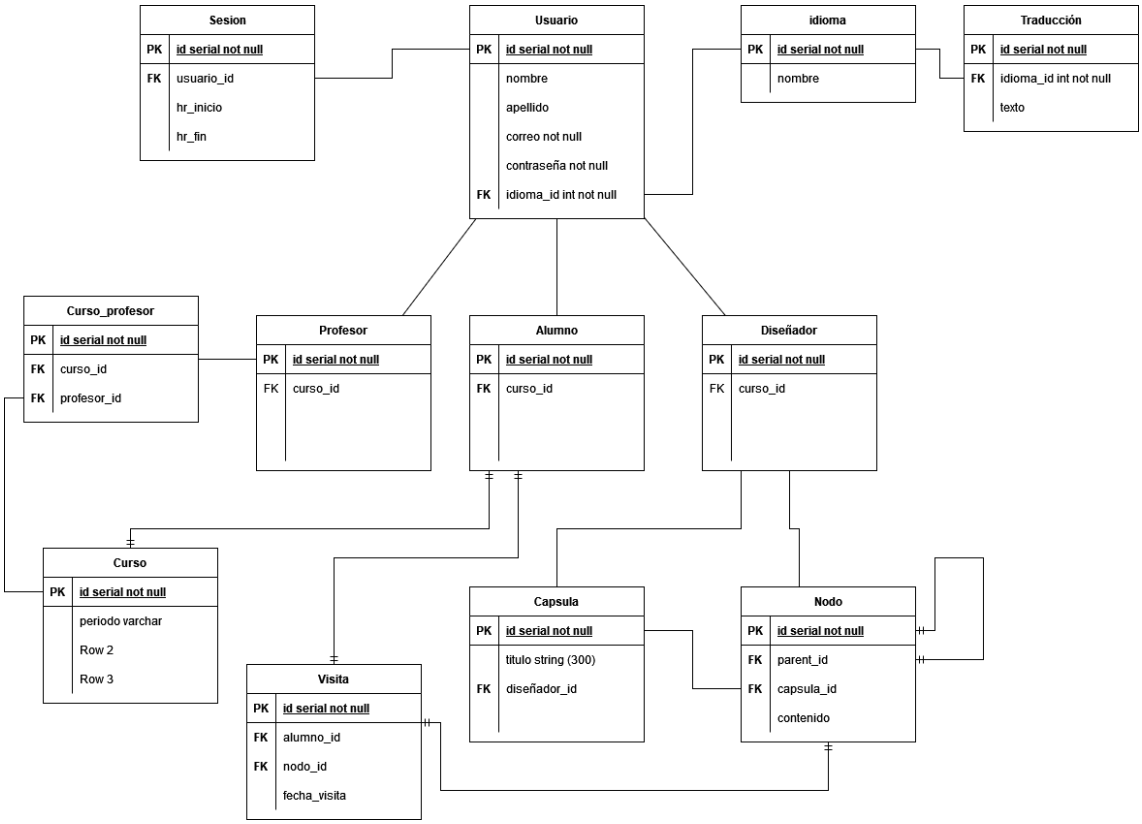

Figura 7: Modelo de datos preliminar. Creación propia, 2022.

El uso de metodologías ágiles permitió modificar este modelo de datos para adaptarlo a las nuevas necesidades detectadas durante el proceso de desarrollo de *software*, por lo que este fue evolucionando hasta llegar al esquema que fue utilizado finalmente.

4.3.2 Diseño de prototipado

Fueron diseñados una serie prototipos de la aplicación a modo de producto mínimo viable, con la finalidad de comunicar visualmente las funcionalidades presentadas en los requerimientos y recibir retroalimentación tanto de la contraparte, como del profesor guía de este proyecto. A continuación, se presentan algunos de los prototipos más relevantes generados durante esta etapa del desarrollo:

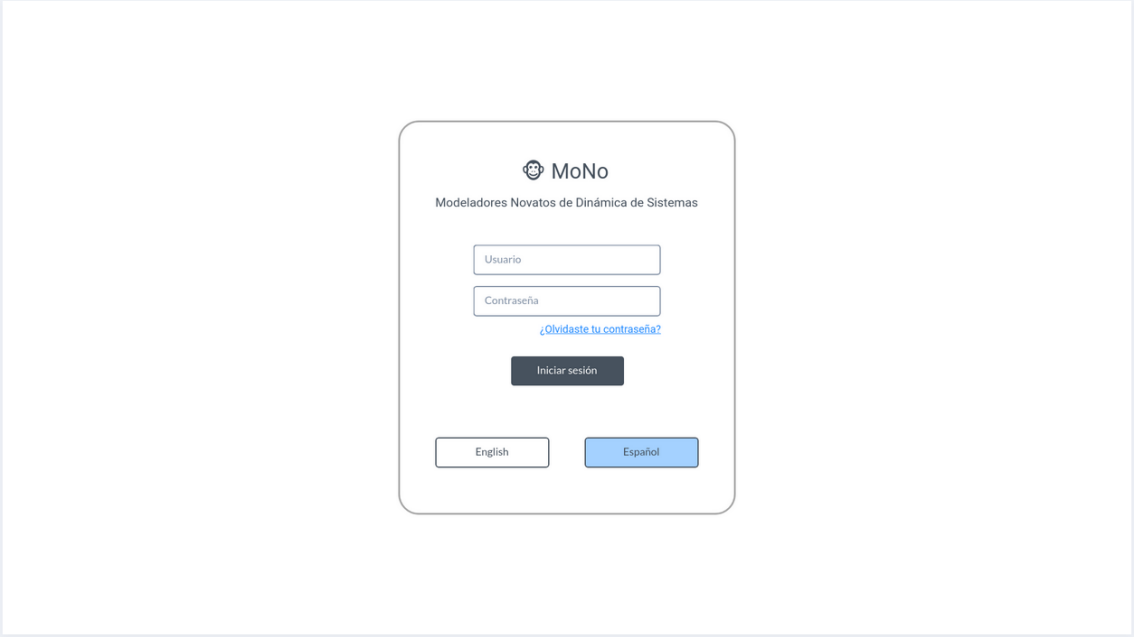

Figura 8: Prototipo pantalla de inicio. Creación propia, 2022.

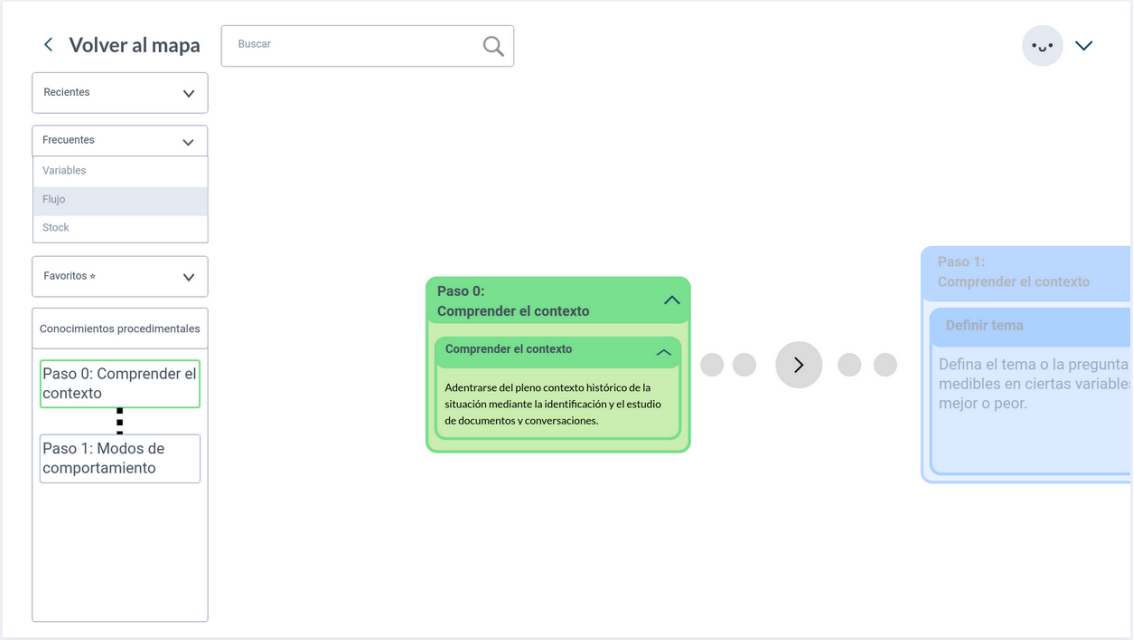

Figura 9: Prototipo pantalla de trabajo. Creación propia, 2022.

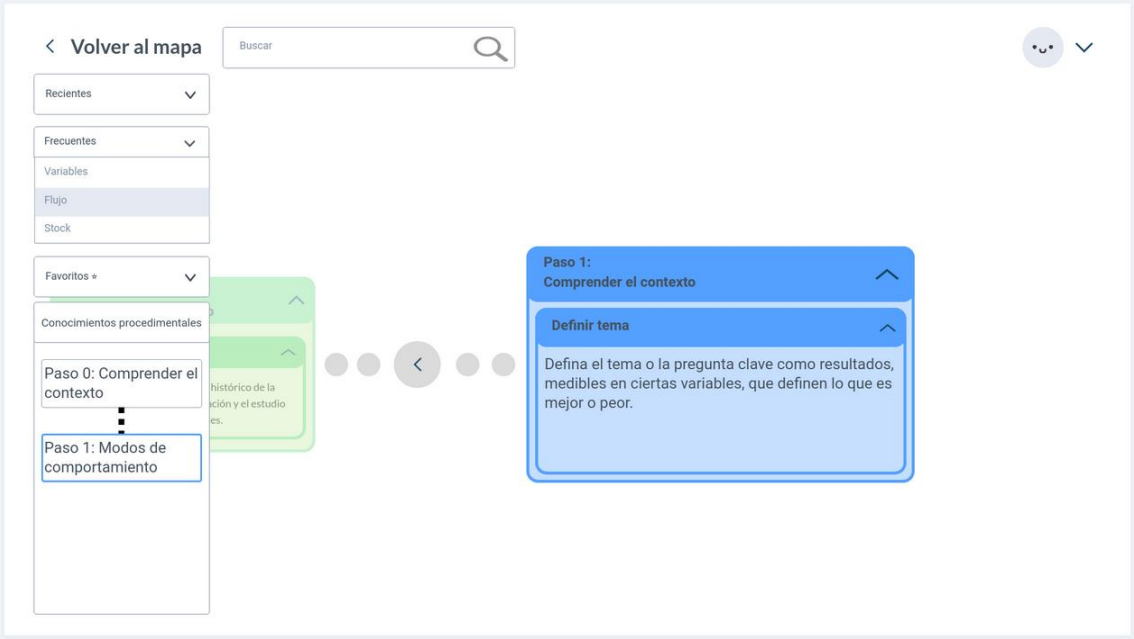

Figura 10: Prototipo pantalla de trabajo. Creación propia, 2022.

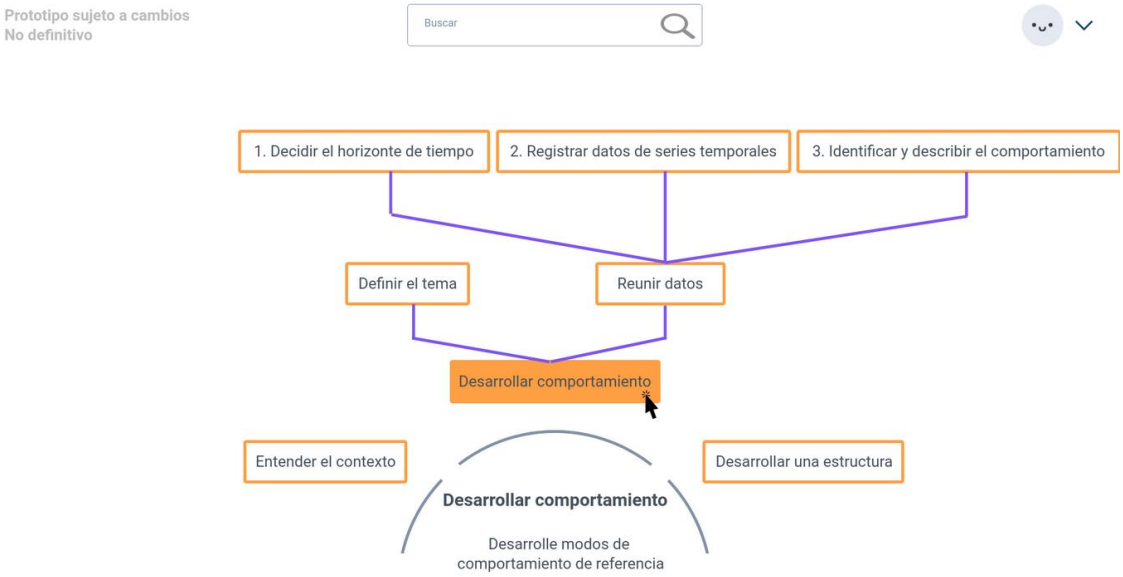

Figura 11: Prototipo mapa de navegación. Creación propia, 2022.

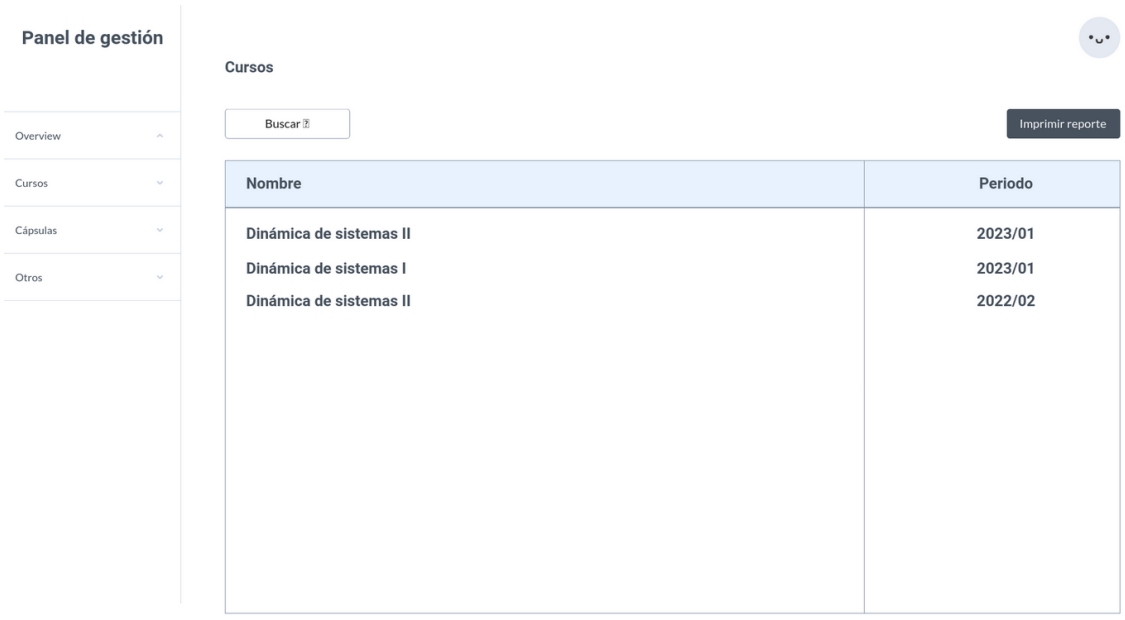

Figura 12: Prototipo pantalla de cursos. Creación propia, 2022.

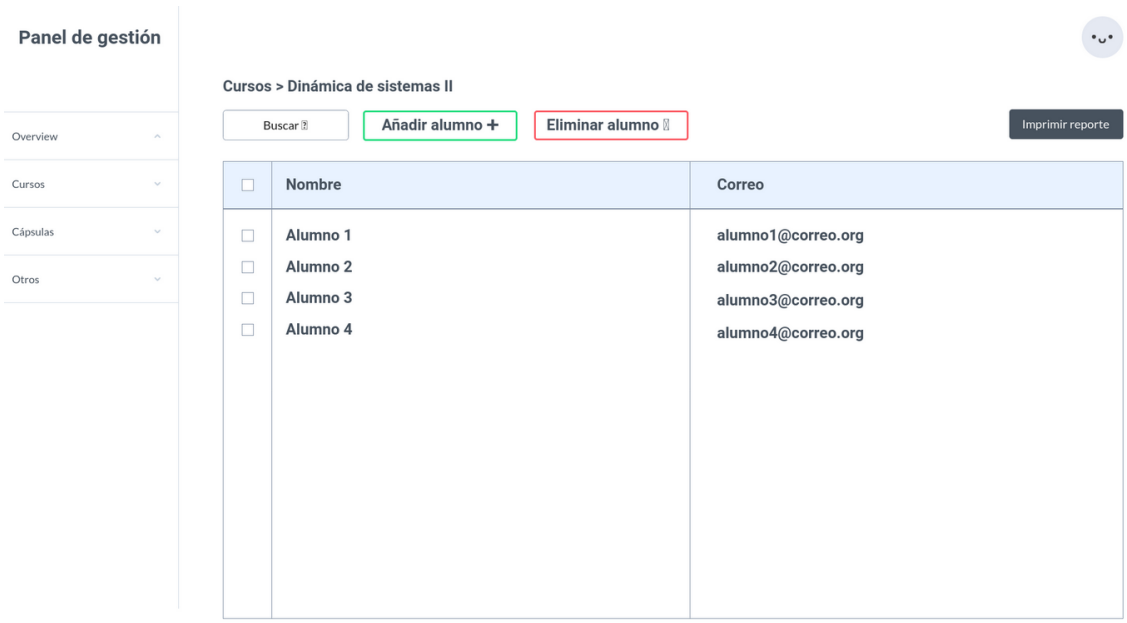

Figura 13: Prototipo vista interna de curso. Creación propia, 2022.

4.4 Fase de desarrollo

Durante esta etapa del desarrollo se llevaron a cabo todos los diseños generados previamente para ser implementados a través de una plataforma web.

4.4.1 Herramientas utilizadas

- Tal como se mencionó durante los capítulos previos, el desarrollo se llevó a cabo bajo el *framework* de desarrollo web en PHP

Laravel por sugerencia del profesor guía de este proyecto. Revísese el apartado dedicado a Laravel en la sección *Frameworks* del capítulo Marco teórico para más información.

- Visual Studio Code como editor de código fuente. Visual Studio Code es un editor de código fuente de código abierto desarrollado por Microsoft, disponible en plataformas Windows, macOS, Linux y a través de navegadores Web. Brinda soporte a múltiples lenguajes de programación como Python, Java, JavaScript, entre muchos otros (Microsoft, 2022).
- MariaDB como base de datos. MariaDB es un sistema de gestión de bases de datos relacionales desarrollado por la fundación MariaDB bajo licencia de código abierto (MariaDB Fundation, 2022).
- DBeaver como herramienta de base de datos. DBeaver es una herramienta universal de base de datos de código abierto desarrollado por su comunidad. Entrega soporte a los gestores más conocidos, como PostgreSQL, Oracle, MySQL, MariaDB, entre muchos otros (DBeaver Community, 2022).
- GitLab como plataforma de control de versiones. GitLab es un servicio web de control de versiones que además ofrece servicios de *DevOps* y *Forja* basado en Git. Es desarrollado por GitLab Inc. bajo licencia de código abierto (GitLab Inc., 2022).
- NGROK como herramienta de creación de servidores temporales. NGROK es un servicio de red que permite conectar de manera segura ambientes locales con la internet y la nube. Es desarrollado por NGROK Inc. (NGROK Inc., 2022)

4.4.2 Implementación del diseño

Durante esta etapa del desarrollo del proyecto surgieron diversos cambios tanto al modelo de datos como a los prototipos planteados a medida que se construían nuevas funcionalidades de la aplicación. El modelo de datos definitivo se puede observar en la Figura 14: Modelo de datos final. Creación propia, 2022.

Se destaca la creación de nuevas tablas en respuesta a cambios de los requerimientos y funcionalidades de la plataforma, tal como es contemplado durante los procesos de desarrollo con metodologías ágiles. De este modo, se pudo dar respuesta a dichos cambios sin afectar otros aspectos del proyecto, entregando mayor valor a nuestra contraparte.

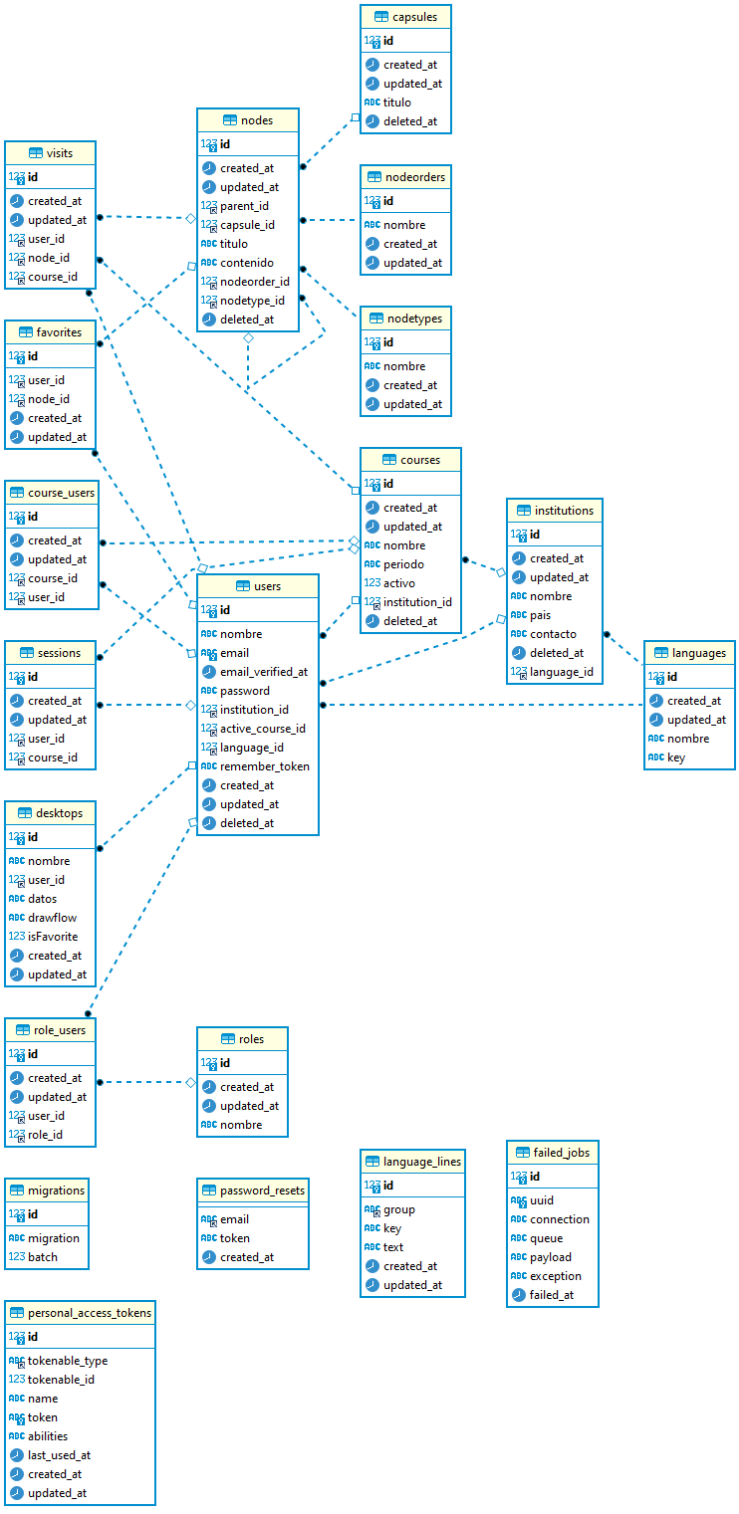

Figura 14: Modelo de datos final. Creación propia, 2022.

Por otra parte, la interfaz de usuario empleó un diseño redondeado con sombras, para lograr un estilo moderno, sencillo y agradable a la vista que evite las distracciones visuales, mejorando su usabilidad. A continuación, se

presentan las pantallas más relevantes del sistema, con una breve explicación de su funcionamiento:

La Figura 15: Pantalla de trabajo. Creación propia, 2022. muestra la funcionalidad principal de la guía. Se puede apreciar un nodo desplegado con información del curso. A la izquierda se encuentran tres pestañas que despliegan, respectivamente, una lista con las búsquedas recientes del usuario, las consultas más frecuentes a la plataforma, y una lista de nodos elegidos como favoritos por el alumno. En la esquina superior derecha se ubica la gestión de espacios de trabajo, que permite guardar y cargar una configuración de nodos para ser desplegados posteriormente. Por último, se encuentra el botón de usuario que despliega una ventana para acceder a su perfil y cerrar sesión.

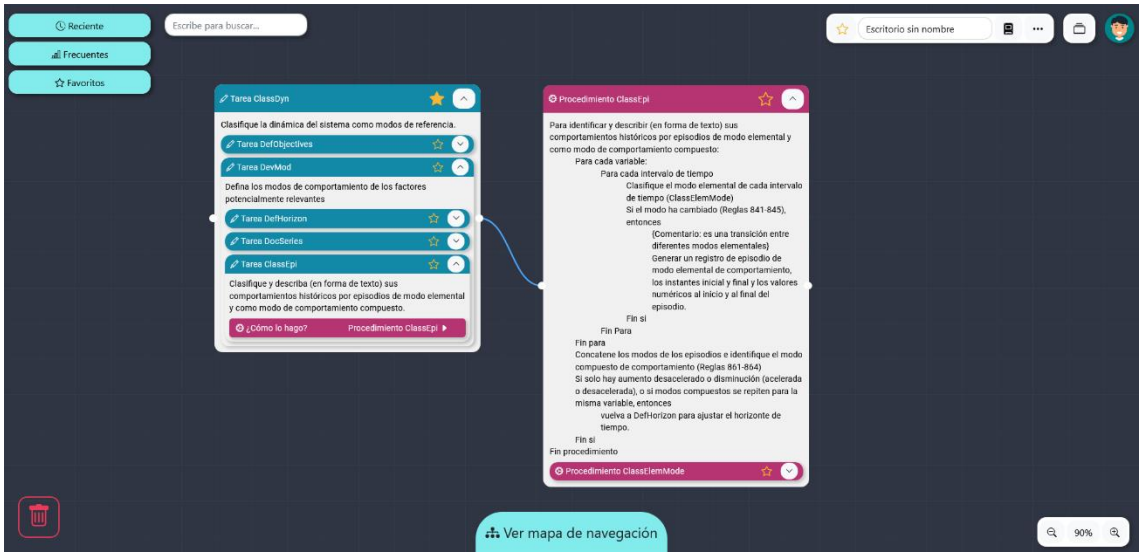

Figura 15: Pantalla de trabajo. Creación propia, 2022.

En la Figura 16: Pantalla de inicio del panel de administración. Creación propia, 2022. nos encontramos con la pestaña de bienvenida al panel de administración. Despliega información general relevante del curso a modo de *dashboard*. A la izquierda se encuentra un panel de navegación con accesos a otros lugares del panel.

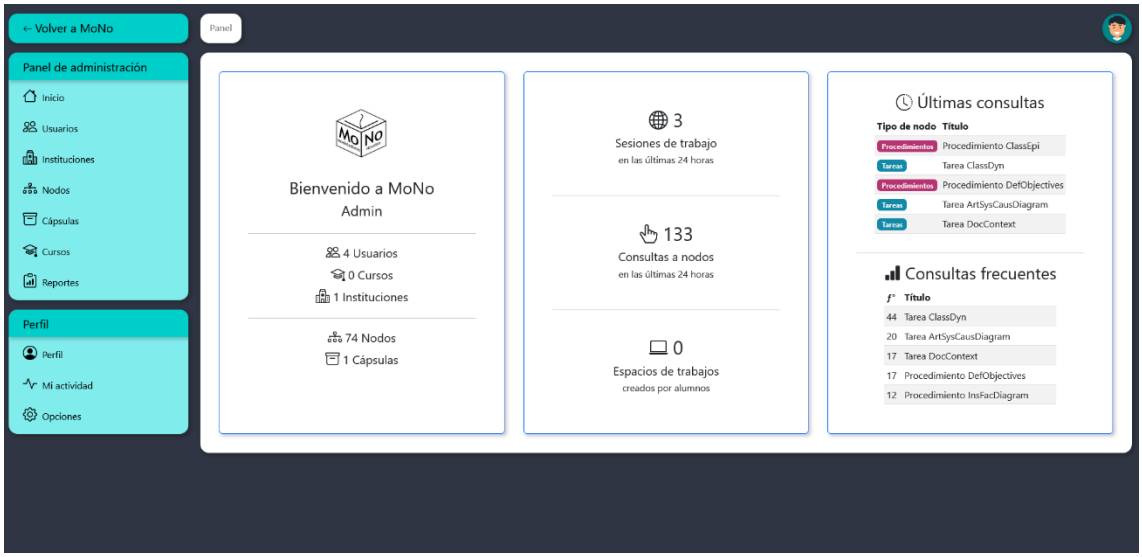

Figura 16: Pantalla de inicio del panel de administración. Creación propia, 2022.

La Figura 17: Panel de administración en sección nodos. Creación propia, 2022. destaca la vista de nodos, que involucra la creación, edición y modificación de los nodos que contienen la información de la guía de modelado.

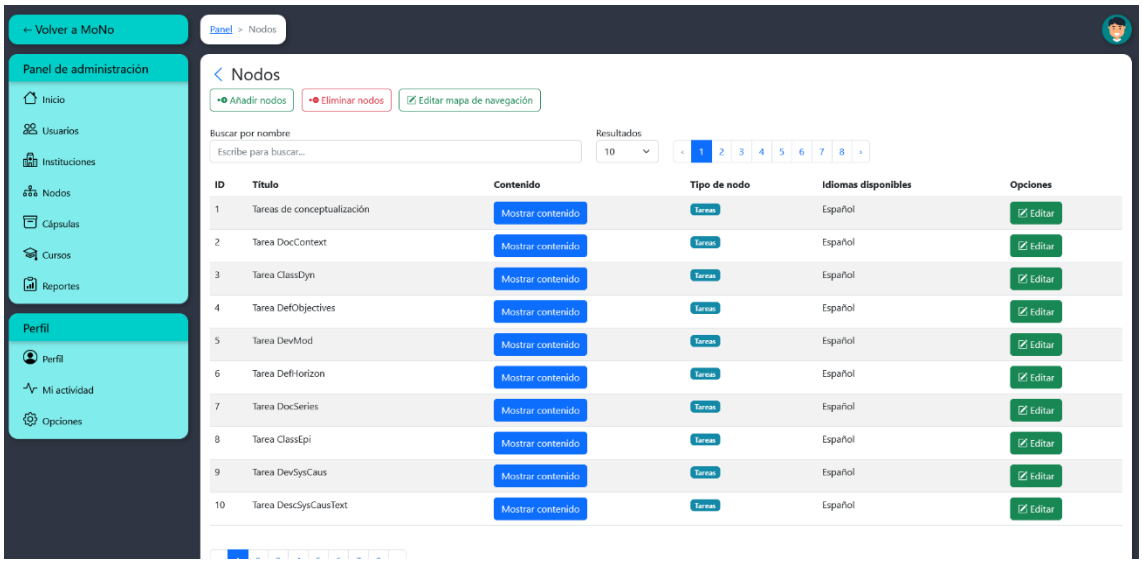

Figura 17: Panel de administración en sección nodos. Creación propia, 2022.

La Figura 18: Perfil sección actividad. Creación propia, 2022. contiene la pantalla de “Mi actividad”, que entrega información resumida del itinerario de los alumnos en la guía, junto a gráficos que resumen el tiempo de actividad en la plataforma, las consultas a los nodos según el tipo y el avance en las cápsulas del sistema.

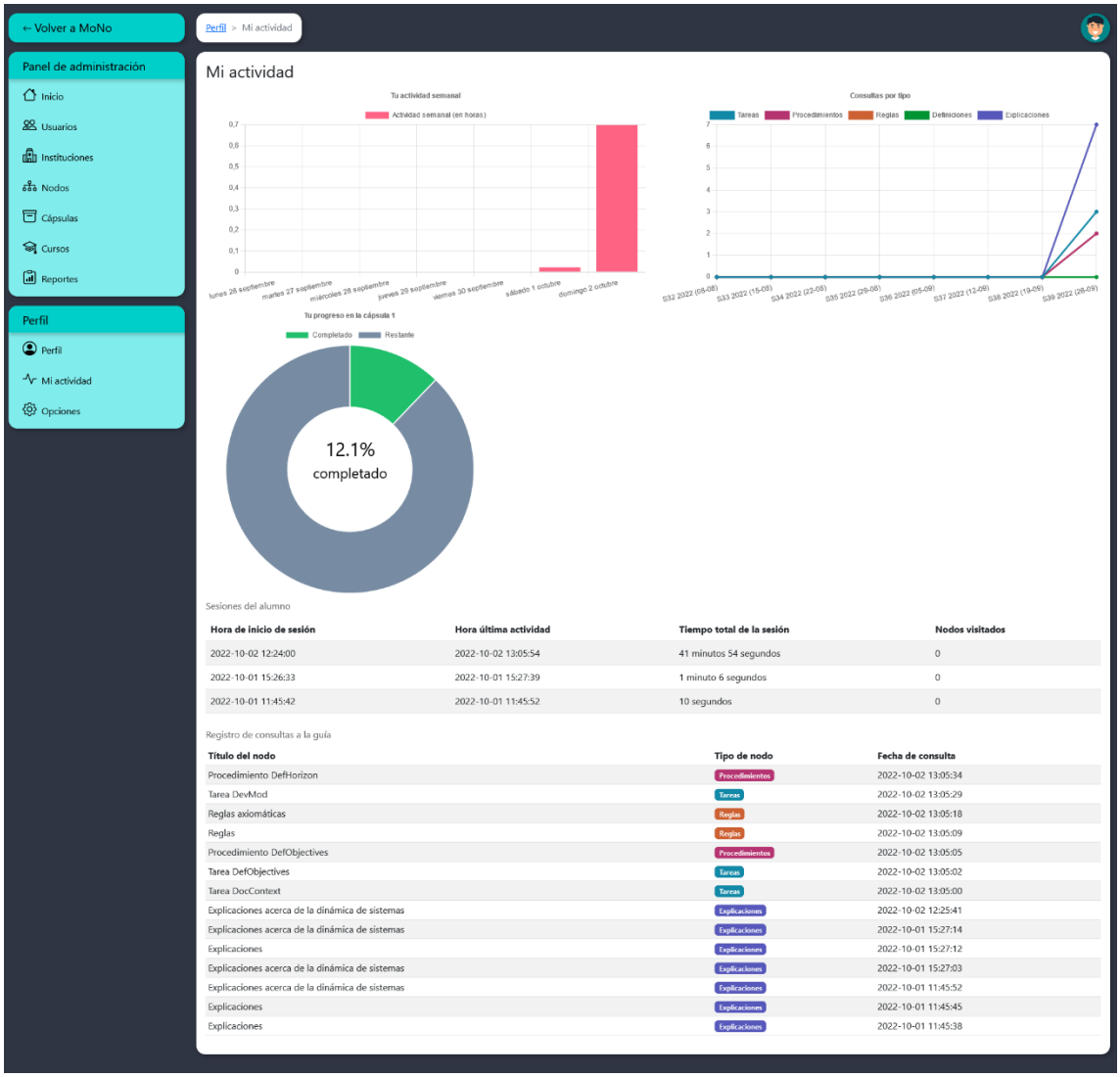

Figura 18: Perfil sección actividad. Creación propia, 2022.

La Figura 19: Panel de administración en sección instituciones. Creación propia, 2022. permite ver la pantalla de instituciones de la plataforma, que entrega un listado de todos los alumnos pertenecientes a dicha institución, junto con todos los cursos impartidos por ésta en la plataforma.

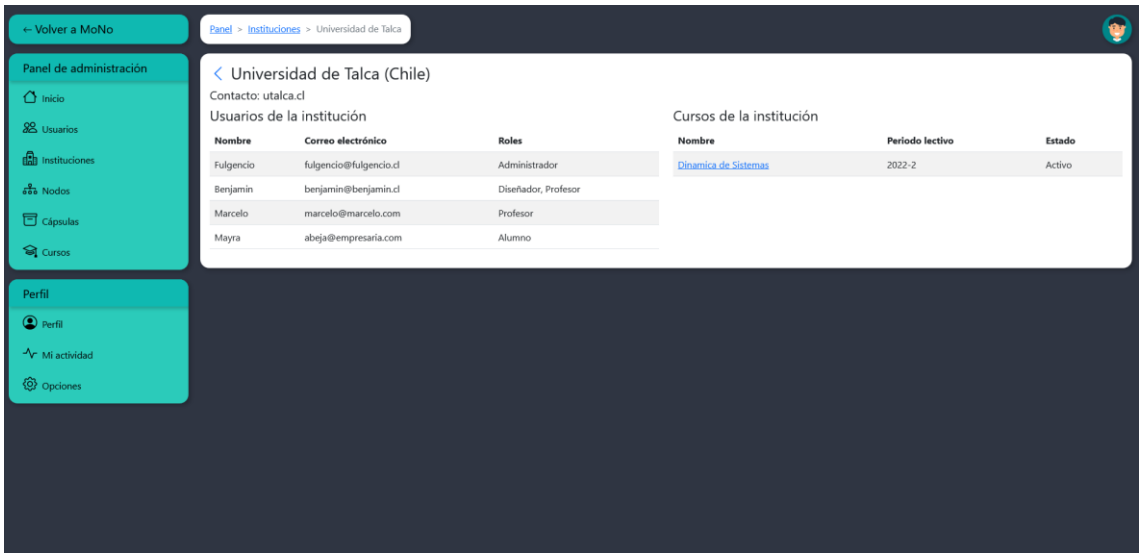

Figura 19: Panel de administración en sección instituciones. Creación propia, 2022.

Finalmente, en la Figura 20: Panel de administración en sección cursos. Creación propia, 2022. encontramos la vista de cursos, que presenta un gráfico que contabiliza la cantidad de consultas en un periodo de tiempo, una tabla mostrando las consultas frecuentes a los nodos y un listado de todos los alumnos y profesores registrados en estos. Permite la inscripción y desvinculación de usuarios, la desactivación y activación de estos en el curso, así como la posibilidad de revisar el avance de estos en los contenidos de la guía.

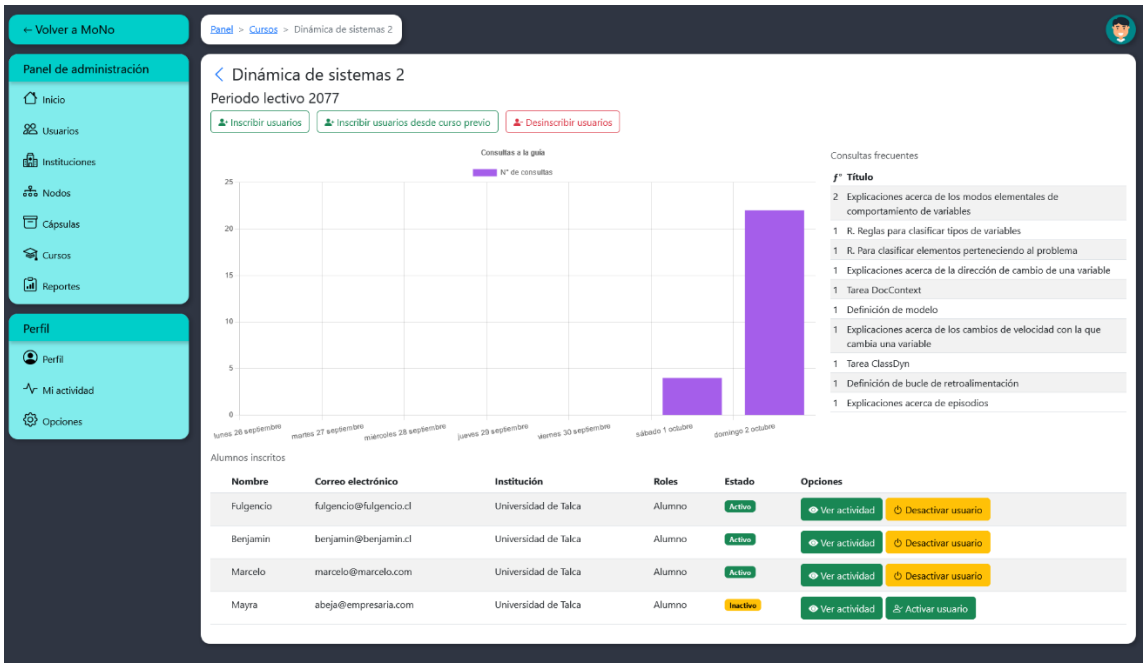

Figura 20: Panel de administración en sección cursos. Creación propia, 2022.

Otras pantallas relevantes pueden encontrarse en el capítulo de anexos: Vista de la plataforma.

4.5 Fase de implementación

Esta etapa del desarrollo fue destinada a la implementación de este proyecto en el curso dinámica de sistemas II impartido por el profesor Martin Schaffernicht con la finalidad de probar las funcionalidades del sistema, así como realizar pruebas de rendimiento con el servidor.

En etapas tempranas de la implementación, se ejecutaron una serie de pruebas internas a través de un servidor local montado para realizar pruebas preliminares antes de contar con los servidores definitivos de la plataforma, permitiendo realizar pruebas de rendimiento en los computadores de los laboratorios Arrayán de la carrera Ingeniería Informática Empresarial.

Se hizo uso del *software* “NGROK” para la generación de un servidor local temporal en uno de los equipos personales de desarrollo, logrando poner a prueba la plataforma con sus diferentes funcionalidades, como la creación de usuarios y cursos, la generación de reportes, los resúmenes de actividad de los estudiantes y la interfaz interactiva de la guía de modelado conceptual MoNo.

Posterior a ejecutar las pruebas necesarias, se procedió a la selección del servicio de hosting, en este caso el servidor se instaló empleando los servicios de Google Cloud, utilizando el periodo de prueba gratuita de Pablo Abarca. Este otorga $300 USD en créditos para ser usados en 90 días. Se implementó dentro de una máquina virtual modelo “e2-medium” con el sistema operativo Debian 11, teniendo 4GB de memoria RAM y un disco SSD de 10GB para almacenamiento. El servidor fue ubicado en la región “southamerica-west1-a”, perteneciente a Santiago (Chile) para reducir en la mayor medida posible la latencia y tiempos de carga de la plataforma.

Por otra parte, el profesor Martin Schaffernicht gestionó el dominio web *mono-guide.cl* para la implementación del sitio web. Se decidió que el dominio fuese gestionado por el profesor Schaffernicht para tener el control del sitio luego de terminado este proyecto de tesis, con la finalidad de continuar usando la plataforma cuando se decida expandir a otras instituciones en los años próximos.

El martes 25 de octubre fue realizada una reunión para generar las credenciales de acceso a los alumnos del curso usando la planilla de alumnos de Educandus, para ser utilizadas al día siguiente. La plataforma fue implementada finalmente el miércoles 26 de octubre durante una de las clases del curso Dinámica de sistemas II en conjunto al profesor Martin Schaffernicht. En esta instancia se presentó el proyecto a los alumnos del curso, enseñando cómo acceder a la plataforma y las funciones que estaban disponibles dentro del espacio de trabajo. Durante la clase, los alumnos ingresaron a la plataforma sin dificultad y observaron por primera vez la pantalla de trabajo desde la perspectiva del usuario. Luego de un breve periodo de tiempo se les enseñó a los alumnos las funciones para cambiar sus contraseñas de acceso personal, esto con el fin de mejorar la privacidad de cada cuenta.

A lo largo de la hora de clase, los alumnos comenzaron a generar ciertos comentarios sobre la plataforma y su navegación, entregando como recomendación durante ese momento mejorar el movimiento del mapa de navegación, permitiéndoles usar las flechas del teclado y también incorporar la capacidad de marcar el contenido de los nodos, puesto que en los nodos de procedimiento los cuales incorporan contenido algorítmico, se perdían al realizar el seguimiento del texto.

La presentación del proyecto en el curso mostró resultados favorables en términos del uso de la aplicación, permitiendo evaluar el uso del servidor con múltiples usuarios de forma instantánea y su impacto en el desempeño de este, con lo cual, la instancia de implementación fue clasificada exitosa y generó información útil para mejorar la aplicación.

Cabe mencionar que, además de la retroalimentación realizada en vivo durante la clase, a los alumnos se les compartió una encuesta en la misma plataforma, la cual también presentó un espacio para recibir comentarios y recomendaciones con el fin de seguir obteniendo información para continuar mejorando el sistema.

5. Conclusiones

Durante el desarrollo de este proyecto se han sorteado una serie de problemáticas tanto técnicas como conceptuales desde el punto de vista de la disciplina de Dinámica de Sistemas. La plataforma MoNo, nombre definido por el profesor Martin Schaffernicht, es la iteración más reciente en su trayectoria de mejoras a su propuesta docente para los ramos de dinámica de sistema I y II impartidos por la carrera de Ingeniería Informática Empresarial.

MoNo es una nueva manera de entregar los conocimientos para la conceptualización y formulación de modelos de dinámica de sistemas. Alimentándose de todas las versiones previas de la guía de modelado en formato físico, y tomando en cuenta toda la retroalimentación recopilada a través de los años en que estuvo en desarrollo, el profesor Schaffernicht realiza una propuesta que busca entregar estos conocimientos de una forma interactiva a través de una plataforma web que ayude a solventar los problemas que presentaban las guías previas.

Con este objetivo en mente, el resultado final fue una plataforma Web interactiva y didáctica que permita a los estudiantes navegar a través de esta y encontrar los conocimientos que necesitan para el desarrollo de su aprendizaje en los módulos dinámica de sistemas I y II. El diseño de este sistema permite una rápida navegación y alta interactividad, usando un diseño simple y agradable que incorpora principios de experiencia de usuario (UX) y gamificación para entregar una experiencia nueva y gratificante en su uso a los estudiantes. A sí mismo, logra solventar muchos de los problemas planteados por las guías anteriores en formato de texto, pues permite a los profesores del módulo hacer un seguimiento en el trabajo de sus estudiantes para entregar retroalimentación y entender los patrones de aprendizaje de estos.

Desde la vista de los estudiantes, MoNo entrega un escritorio de trabajo donde los usuarios pueden organizarse y moverse libremente, tienen además acceso a un mapa de navegación que muestra de manera ordenada los contenidos de la guía de modelado, las cuales se despliegan en pequeñas ventanas o nodos que los estudiantes pueden distribuir de la forma que deseen. Cada nodo despliega su propio contenido, así como el de sus niveles jerárquicos

inferiores, permitiendo una rápida navegación para profundizar en los contenidos de la guía. A su vez, también es posible buscar un nodo directamente por su nombre con la barra de búsqueda ubicada en el sector superior de la pantalla. Además, MoNo lleva registro de las consultas más recientes y las despliega a un costado de la pantalla para que el usuario pueda revisar sus últimas consultas. También lleva un recuento de las consultas más frecuentes para sugerirlas a los alumnos que lo deseen. Por último, cada nodo puede ser guardado como favorito para ser accedido de manera más rápida por el usuario.

MoNo es una plataforma interactiva, por lo tanto, esto se ve plasmado en el uso de los nodos. Estos pueden ser minimizados y arrastrados para organizarlos de la manera que mejor le parezca al usuario, además estos pueden ser eliminados de la pantalla al arrastrarlos al basurero ubicado en la esquina inferior izquierda de la pantalla. Junto a lo anterior, el control del espacio de trabajo permite a los estudiantes moverse por los escritorios arrastrándose con el cursor y hacer zoom para alejar o acercar los contenidos en pantalla. Por último, es posible guardar la distribución de nodos en el escritorio para ser utilizado más tarde, además de poder cargar aquellos que hayan sido subidos por los alumnos previamente. El control de escritorio a su vez permite despejar el espacio de trabajo con un solo botón eliminando todos los nodos, además de permitir descargar los escritorios en un archivo de extensión ".mono" para ser cargado posteriormente.

Todas estas funcionalidades han sido diseñadas para que los estudiantes puedan interactuar activamente con los conocimientos y alentarlos a explorar la plataforma de una manera que no existía previamente. El uso de una interfaz simple e intuitiva a través del uso de colores, simbología e iconografía permite una rápida comprensión del funcionamiento de MoNo.

Cabe añadir que durante el desarrollo de MoNo fueron propuestas ciertas funcionalidades que se pueden incorporar dentro de la plataforma, permitiendo continuar con el desarrollo del proyecto en estudios futuros, lo que permitiría analizar más a fondo el impacto de la aplicación y convertirla en un espacio de trabajo más completo para el aprendizaje. Dentro de las posibles funcionalidades a incorporar se consideran: notas propias del usuario, las cuales sean guardadas

en conjunto a sus escritorios de trabajo, la capacidad de compartir estos escritorios con otros usuarios dentro de la misma aplicación e incorporar dentro de la misma plataforma un simulador, elemento que puede ser de gran valor para el aprendizaje del comportamiento de variables en el área de la dinámica de sistemas. Otras propuestas se detallan en más profundidad en el capítulo de Anexo, Tabla 3: Lista de sugerencias para futuras versiones de MoNo. Elaboración propia (2022)

MoNo contempla ser usado en el largo plazo no tan solo por alumnos del módulo de dinámica de sistemas en la carrera de ingeniería informática empresarial, sino también por estudiantes de otras universidades a lo largo del mundo que compartan la dinámica de sistemas como área de estudio, por lo que, es necesario considerar seguir trabajando en este proyecto, entregando mantenimiento y mejoras para lograr alcanzar mejores versiones del sistema, las que logren integrar de manera correcta la optimización y usabilidad necesarias para un buen rendimiento de la plataforma.

Para dar cierre, MoNo como plataforma sumada a los esfuerzos del profesor Martin Schaffernicht presenta una salida a los métodos tradicionales del aprendizaje para novatos de dinámica de sistemas, logrando innovar con el uso de tecnologías de información en la presentación y seguimiento de la información. Corresponde añadir que esta salida de los métodos tradicionales se puede explorar en otras áreas de estudio, las cuales no presenten una ruta de estudio secuencial, lo que permitiría generar plataformas de aprendizaje similares a MoNo con sus respectivas ventajas o, por otra parte, continuar con la búsqueda de métodos de aprendizaje con el fin de adaptarse a los nuevos estudiantes que buscan incorporarse a estas áreas del conocimiento.

6. Bibliografía

Schaffernicht, M. (2021). MoNo – una guía interactiva para Modeladores Novatos. Universidad de Talca.

SYSTEM DYNAMICS SOCIETY. (2022). *System Dynamics Society*. https://systemdynamics.org/

Schaffernicht, M. (2021). Guía práctica para la conceptualización y formalización de modelos de dinámica de sistemas. Universidad de Talca.

Andreu, Rafael, Joan E. Ricart, and Josep Valor. Estrategia Y Sistemas De Información. Madrid: McGraw-Hill, 1991.

Bravo, D. G. (1994). *Sistemas y tecnologías de la información en las organizaciones. Repercusiones para la administración* (Doctoral dissertation, Universitat d'Alacant-Universidad de Alicante).

Adobe Inc. (2021). *Aspectos básicos de las aplicaciones web*. https://helpx.adobe.com/es/dreamweaver/using/web-applications.html

Arhippainen, L., & Tähti, M. (2003). *Empirical Evaluation of User Experience in Two Adaptive Mobile Application Prototypes*.

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70–77. https://doi.org/10.1109/2.796139

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham W., Fowler, M., Grenning, J., Highsmith J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifiesto por el Desarrollo Ágil de Software*. https://agilemanifesto.org/iso/es/manifesto.html

Centro de Investigación de la Web. (2008). *Cómo funciona la Web*.

CodeIgniter Foundation. (2022). *CodeIgniter*. https://www.codeigniter.com/

Cunico, G., Mollona, E., & Aivazidou, E. (2020). *System dynamics gamification: Lessons learned from the PERCEIVE Simulation Lab development*. https://www.perceiveproject.eu/simulation/

DBeaver Community. (2022). *DBeaver Webpage*.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining “gamification.” *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011*, 9–15. https://doi.org/10.1145/2181037.2181040

Django Software Foundation. (2022). *Django project*. https://www.djangoproject.com/

Galindo, J., & Camps, J. (2010). *Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE*. http://hdl.handle.net/10609/876

Gauchat, J. (2017). *El gran libro de HTML5, CSS3 y JavaScript*.

GitLab Inc. (2022). *GitLab Webpage*.

Gualtieri, M. (2009). *Best Practices In User Experience (UX)*. www.forrester.com.

Hernández, A. (2003). *LOS SISTEMAS DE INFORMACIÓN: EVOLUCIÓN Y DESARROLLO*.

Hull, E., Jackson, K., & Dick, J. (2005). *Requirements engineering*. Springer.

ISO 9241-11:2018. (2018). *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*.

ISO/IEC 2382:2015. (2015). *Information technology — Vocabulary*.

Kankainen, A. (2002). *Thinking model and tools for understanding user experience related to information appliance product concepts*.

Laravel LLC. (2022). *Laravel*. https://laravel.com/

Laudon, J. P., & Laudon, K. (2016). *Sistemas de información gerencial* (14th ed.). Pearson Educación.

Llamuca, J., Vera, Y., & Tapia, V. (2021). Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una

aplicación web progresiva. *TecnoLógicas*, *24*(51), e1892. https://doi.org/10.22430/22565337.1892

López, A. (2018). *INFORME DEL ESTADO DEL ARTE EN METODOLOGÍAS UX*.

MariaDB Fundation. (2022). *MariaDB*.

MDN contributors. (2020, December 8). *Introducción a aplicaciones web progresivas*. https://developer.mozilla.org/es/docs/Web/Progressive_web_apps/Introduction

Microsoft. (2022). *Documentation Visual Studio Code* .

Montero, Y. H. (2015). *Experiencia de Usuario: Principios y Métodos*. www.yusef.esTodoslosderechosreservados,2015EstelibrohasidoescritousandoCalmlyWriter

Mora, S. (2001). *Programación en Internet: clientes web*.

NGROK Inc. (2022). *NGROK*. https://ngrok.com/

Pressman, R. S. (2010). *Ingeniería del software : un enfoque práctico*. McGraw-Hill.

Prieto, F. P. (2017). *Gamifica tu aula: experiencia de gamificación TIC para el aula*.

Sawyer, P., Sommerville, I., & Viller, S. (1999). Capturing the benefits of requirements engineering. *IEEE Software*, *16*(2), 78–85. https://doi.org/10.1109/52.754057

Sierra, F., Acosta, J., Ariza, J., & Salas, M. (2017). Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web. *Investigación y Desarrollo En TIC*, *4*(2).

Sommerville, I. (2011). *Ingeniería de Software* (Novena edición). Pearson Educación.

Strapasson, A., Ferreira, M., Cruz-Cano, D., Woods, J., do Nascimento Maia Soares, M. P., & da Silva Filho, O. L. (2022). The use of system dynamics for energy and environmental education. *International Journal of Educational Technology in Higher Education*, *19*(1). https://doi.org/10.1186/s41239-021-00309-3

System Dynamics Society. (2022). *System Dynamics Society*. https://systemdynamics.org/

Teixes, F. (2015). *Gamificación*. Universitat Oberta de Catalunya. https://www.digitaliapublishing.com/a/43875

The PHP Group. (n.d.). *PHP Documentation*. Retrieved May 7, 2022, from https://www.php.net/manual/es/preface.php

Wells, D. (2013, October 8). *Extreme Programming: A gentle introduction*. Octubre 8. http://www.extremeprogramming.org/

7. Anexos

7.1 Vistas de la plataforma

Vista 1: Pantalla de nodos – mostrar contenido

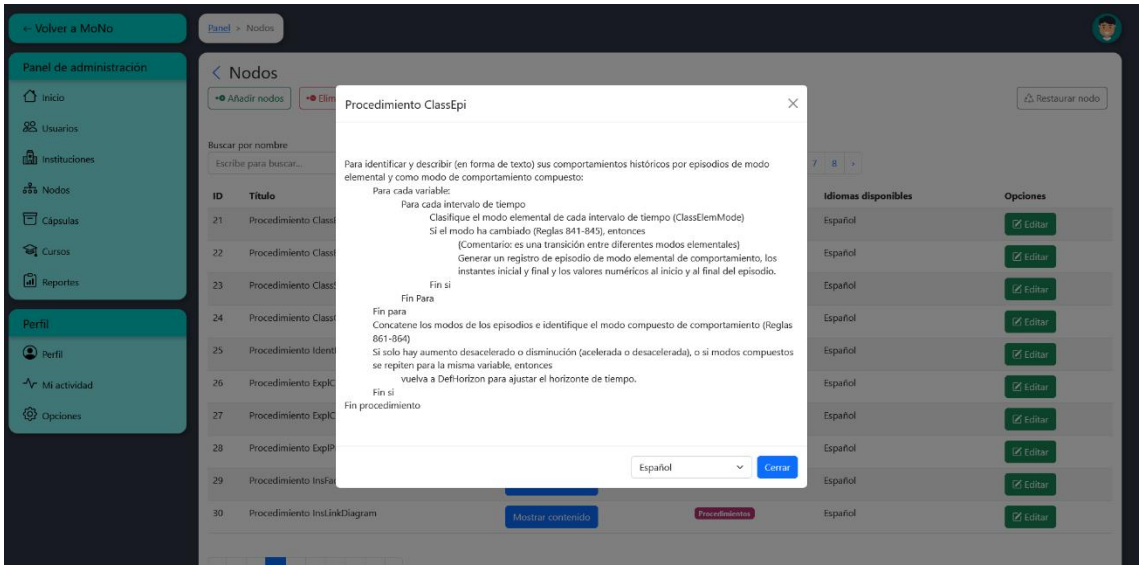

Figura 21: Pantalla de trabajo – Mostrar contenido. Creación propia, 2022.

Vista 2: Pantalla de reportes

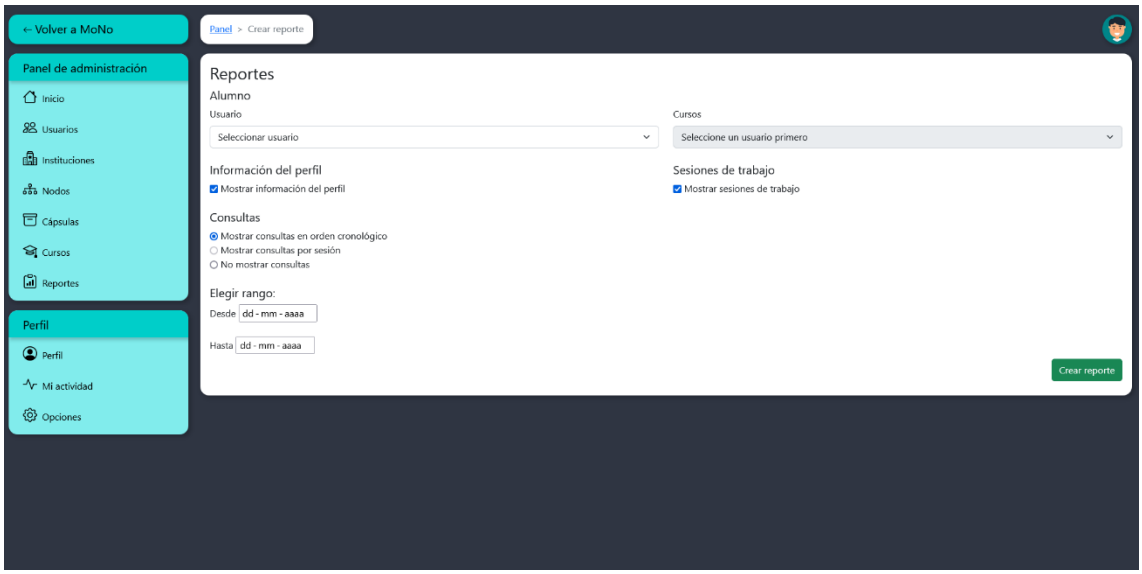

Figura 22: Pantalla de reportes. Creación propia, 2022.

Vista 3: Pantalla de usuarios

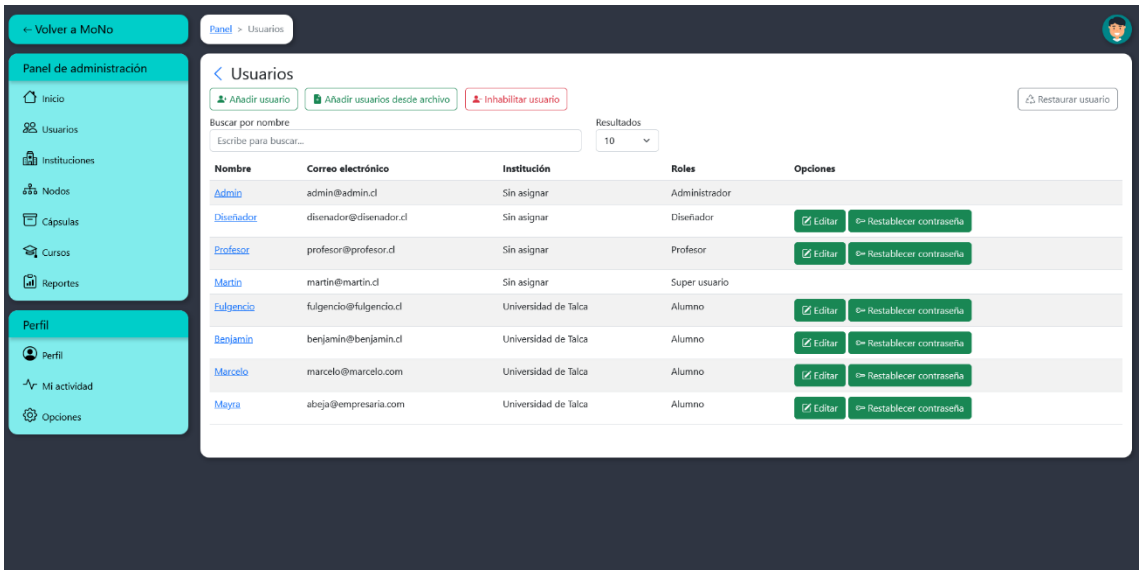

Figura 23: Pantalla de usuarios. Creación propia, 2022.

Vista 4: Pantalla de cápsulas

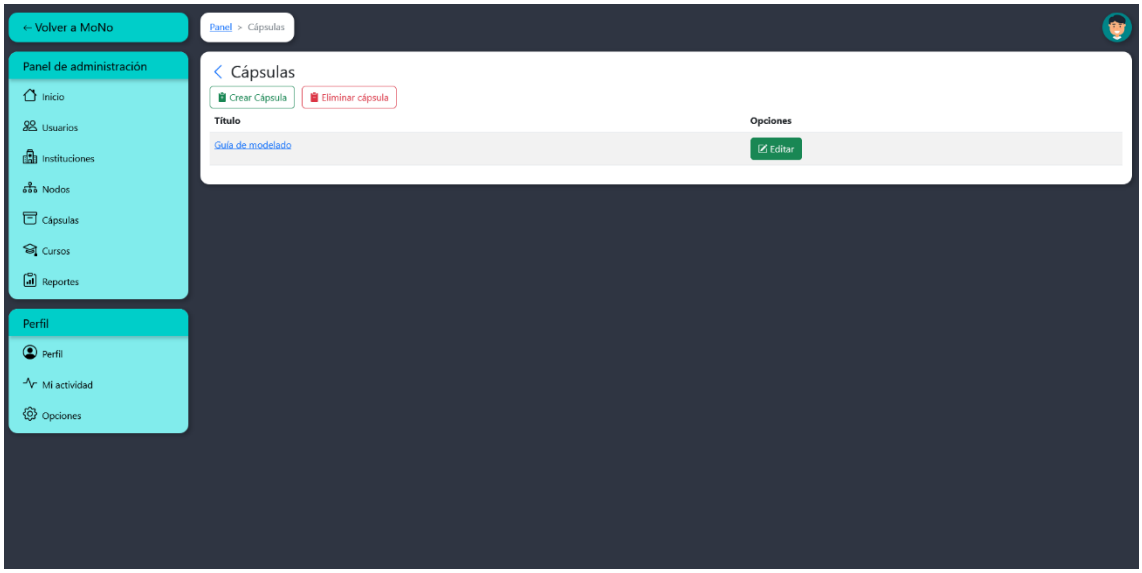

Figura 24: Pantalla de cápsulas. Creación propia, 2022.

Vista 5: Pantalla perfil

Figura 25: Pantalla perfil. Creación propia, 2022.

Vista 6: Pantalla de añadir usuarios desde archivo paso 1

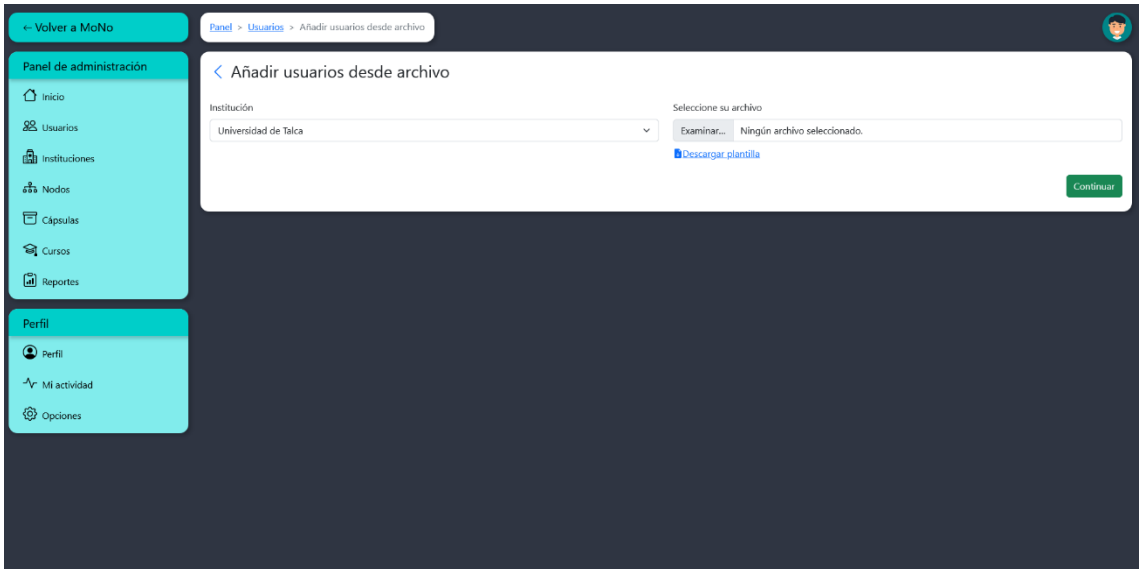

Figura 26: Pantalla de añadir usuarios desde archivo paso 1. Creación propia, 2022.

Vista 7: Pantalla de añadir usuarios desde archivo paso 2

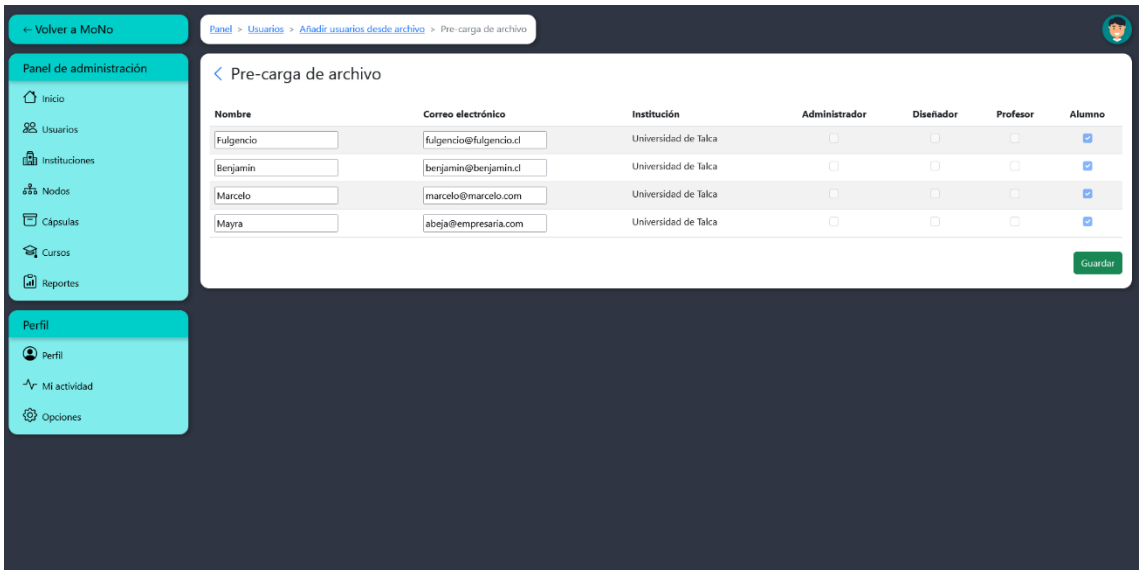

Figura 27: Pantalla de añadir usuarios desde archivo paso 2. Creación propia, 2022.

Vista 8: Pantalla de la aplicación interactiva con función favoritos desplegada

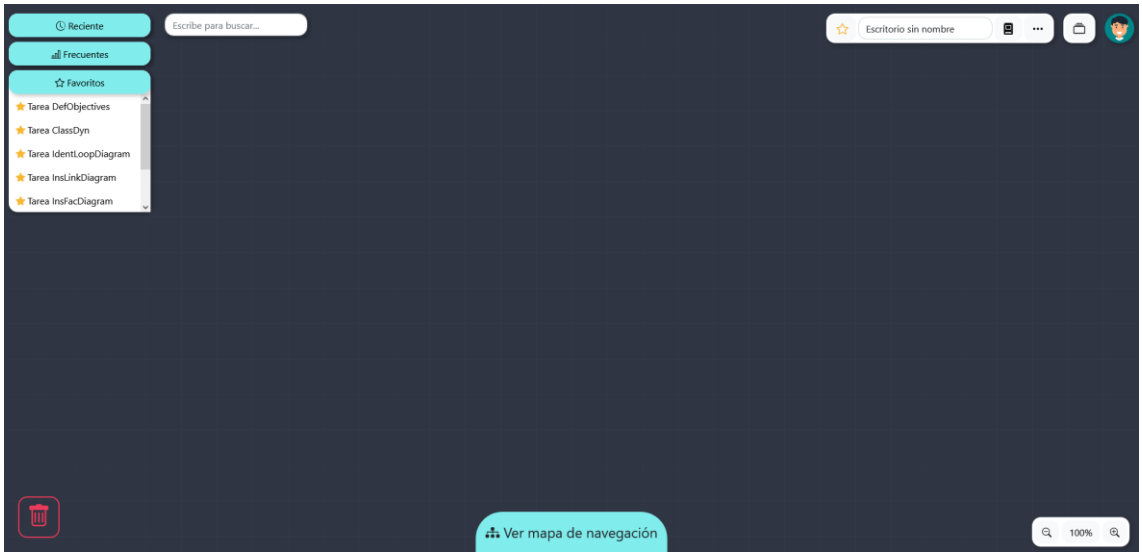

Figura 28: Pantalla de la aplicación interactiva con función favoritos desplegada. Creación propia, 2022.

Vista 9: Pantalla de la aplicación interactiva con función escritorios desplegada

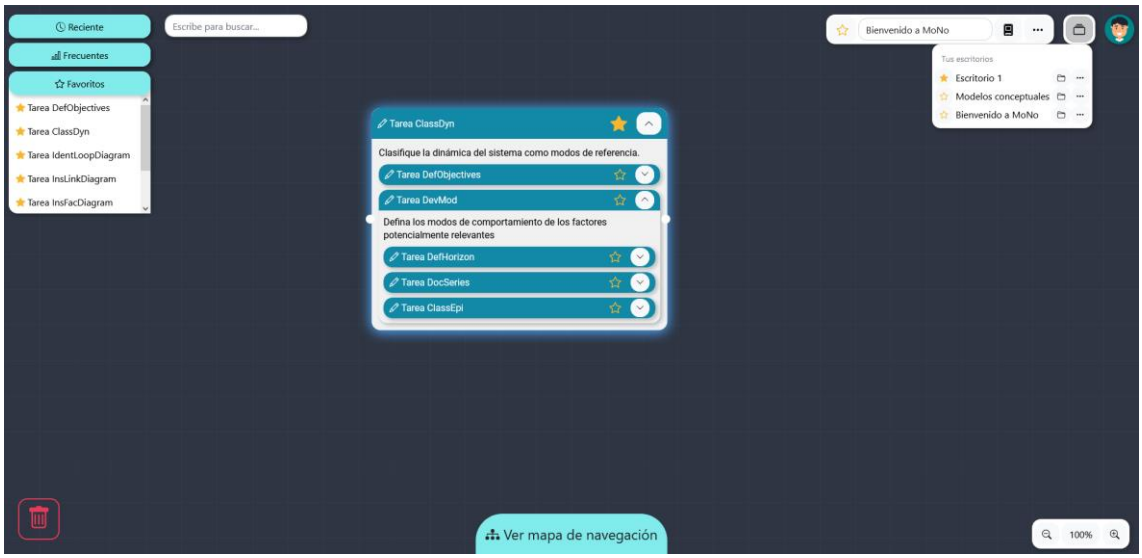

Figura 29: Pantalla de la aplicación interactiva con función escritorios desplegada. Creación propia, 2022.

Vista 10: Pantalla de la aplicación interactiva en mapa de navegación

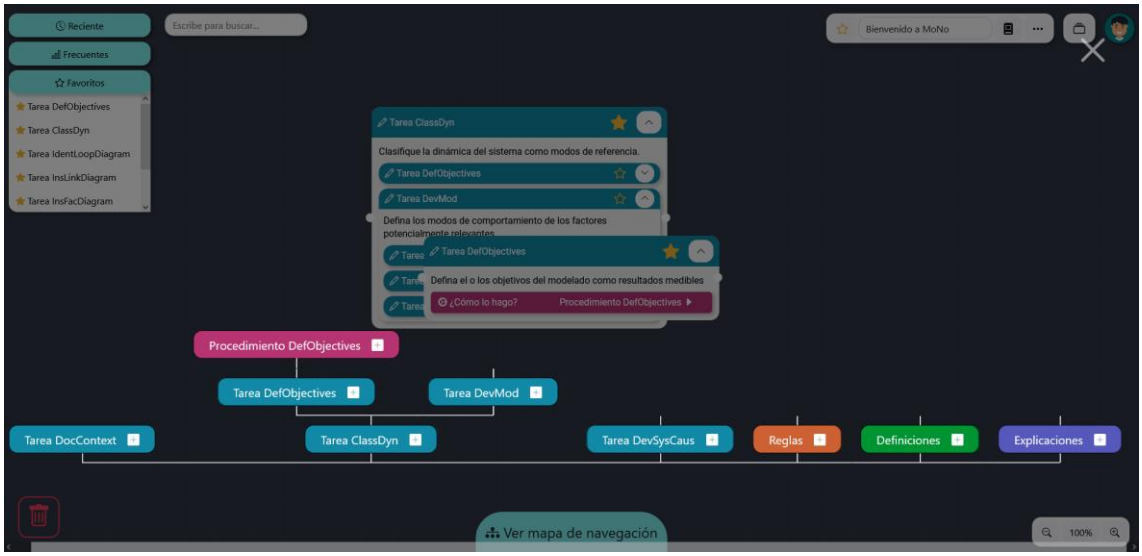

Figura 30: Pantalla de la aplicación interactiva en mapa de navegación. Creación propia, 2022.

7.2 Anexo: Lista de sugerencias para futuras versiones de MoNo

Durante el transcurso del desarrollo de la plataforma, surgieron diversas ideas que podrían agregar valor a MoNo. Sin embargo, debido a limitaciones del alcance del proyecto y tiempo, se decidieron dejar como ideas para las nuevas versiones de MoNo a los próximos posibles tesistas que decidan continuar con su desarrollo.

Tabla 3: Lista de sugerencias para futuras versiones de MoNo. Elaboración propia (2022)

N°	Descripción de la sugerencia	Potencial valor de la sugerencia
1	Agregar una función para importar y exportar la estructura del mapa de navegación en archivos JSON.	Permitiría llevar un respaldo de los mapas generados, así como también permitiría replicar las estructuras en otros lugares donde se implemente MoNo (si se decide implementar en servidores en otros lugares del mundo).
2	Agregar la función de crear "*sticky notes*" dentro de los escritorios de trabajo.	Permitiría a los alumnos escribir sus propios apuntes en la plataforma. La librería *Drawflow* utilizada actualmente permite guardar su contenido sin mayores modificaciones al código.
3	Reposicionamiento de nodos creados para evitar solapamiento.	Actualmente al generar un nodo desde otro este ese ubica a la misma altura, al lado derecho. Al abrir múltiples estos se generan unos sobre otros y deben ser movidos manualmente. Este cambio permitiría mejorar la usabilidad.

4	Implementación de Livewire en el escritorio de trabajo.	Livewire permite modificar el código de la página web directamente desde back-end, lo que permitiría, por ejemplo, cambiar el idioma de la interfaz sin necesidad de recargar la página.
5	Advertencia al no guardar los escritorios al salir.	Para evitar perder información al cerrar la página por accidente. Se sugiere que con ello también se implemente la opción de “nunca preguntar antes de salir” para estos casos.
6	Búsqueda de usuarios por roles y/o instituciones.	Entregaría una mejor experiencia de búsqueda en la medida que se ampliase la plataforma y más usuarios estén registrados en esta.
7	Gestión de versiones de MoNo.	Guardar escritorios de una versión antigua podría causar problemas al actualizar la estructura de la guía en profundidad, por eso se sugiere limitar estos a versiones en concreto para no perjudicar la experiencia de usuario.
8	Autenticación de acciones.	Entregaría mayor seguridad para ataques a la plataforma, mientras más crezca MoNo, más necesario será reducir las posibles vulnerabilidades de la plataforma.

9	Personalización de temas de MoNo.	Durante las etapas tempranas de MoNo se implementaron una serie de plantillas intercambiables de colores para el tema de MoNo. Esa iniciativa se podría retomar para entregar a los usuarios una experiencia más personalizable e interactiva.
10	Servicio de correo de MoNo.	Permitiría, por ejemplo, el envío de correos electrónicos para restaurar contraseñas y otros asuntos de la plataforma, como comunicaciones a los usuarios.
11	Gráficos y modelos en MoNo.	Se pueden generar ciertas plantillas HTML para ser implementadas en ciertos nodos con *Drawflow*, con contenidos como gráficos, modelos, líneas de tiempo, etc. En formato SVG.
12	Rol de ayudante o invitado.	Este rol permitiría agregar a los ayudantes de los módulos de Dinámica de sistemas sin tener que ocupar el rol de profesor. Este rol podría limitar los permisos de inscripción de alumnos, pero conservar la capacidad de revisar el progreso de los estudiantes.

7.3 Anexo: Pruebas de latencia con servidor NGROK y servidor de Google Cloud

Tabla 4: Pruebas de latencia primera iteración. Elaboración propia (2022)

Versión App	0.9
SO	Windows 10
Tipo Conexión	Internet LAN UTalca Lab. Arrayán 2
Desc. PC	PC escritorio laboratorio Arrayán 2
Tipo Servidor	NOTEBOOK PERSONAL NGROK

	Instancia 1	**Instancia 2**	**Instancia 3**
Navegador	Firefox 105.0.1	Firefox 105.0.1	Chrome 105.0
Idioma	Inglés	Español	Inglés
CARGA DE LOGIN	2,55 seg.	2,49 seg.	2,25 seg.
CARGA MONO APP TOTAL	5,38 seg.	-	2,71 seg.
CARGA MONO DOM	1,44 seg.	-	1,54 seg.
CARGA MONO CSS	789 ms	-	458 ms
CARGA JS	948 ms	-	671 ms
REFRESCARSESION	1454 ms	-	1150 ms
GETALL DESKTOP	1285 ms	-	1010 ms
GETALL FAVORITOS	833 ms	-	737 ms
FRECUENTES	675 ms	-	573 ms
RECIENTES	1093 ms	-	877 ms
CARGA PANEL INICIO	2,47 seg.	2,27 seg.	6,27 seg.
CARGA USUARIOS	2,10 seg.	2,38 seg.	12,63 seg.
CARGA CURSOS	2,27 seg.	2,83 seg.	6,74 seg.
CARGA MI ACTIVIDAD	2,52 seg.	3,40 seg.	8,49 seg.

Tabla 5: Pruebas de latencia, segunda iteración. Elaboración propia (2022)

Versión App	0.9
SO	Windows 10
Tipo Conexión	Internet Wi-Fi UTalca Lab. Arrayán 2
Desc. PC	PC Notebook Marcelo H.
Tipo Servidor	NOTEBOOK PERSONAL NGROK

	Instancia 1	**Instancia 2**
Velocidad conexión	160 Mbps	190 Mbps
Navegador	Edge 105.0	Opera GX 90.0
Idioma	Español	Español
CARGA DE LOGIN	2,88 seg.	2,74 seg.
CARGA MONO APP TOTAL	3,37 seg.	3,36 seg.
CARGA MONO DOM	1,89 seg.	2,02 seg.
CARGA MONO CSS	487 ms	542 ms
CARGA JS	683 ms	815 ms
REFRESCARSESION	732 ms	1140 ms
GETALL DESKTOP	1470 ms	1310 ms
GETALL FAVORITOS	1130 ms	794 ms
FRECUENTES	946 ms	966 ms
RECIENTES	1300 ms	653 ms
CARGA PANEL INICIO	2,12 seg.	2,16 seg.
CARGA USUARIOS	1,82 seg.	1,96 seg.
CARGA CURSOS	2,10 seg.	1,96 seg.
CARGA MI ACTIVIDAD	2,39 seg.	2,28 seg.

Tabla 6: Pruebas de latencia, tercera iteración. Elaboración propia (2022)

Versión App	0.9
SO	Windows 10
Tipo Conexión	Internet LAN UTalca Lab. Arrayán 2
Desc. PC	PC escritorio laboratorio Arrayán 2
Tipo Servidor	NOTEBOOK PERSONAL NGROK

	Instancia 1	**Instancia 2**
Velocidad conexión	370 Mbps	220 Mbps
Navegador	Chrome 105.0	Firefox 105.0.1
Idioma	Inglés	Español
CARGA DE LOGIN	2,34 seg.	2,26 seg.
CARGA MONO APP TOTAL	3,13 seg.	3,88 seg.
CARGA MONO DOM	1,79 seg.	2,42 seg.
CARGA MONO CSS	495 ms	468 ms
CARGA JS	688 ms	650 ms
REFRESCARSESION	1320 ms	1300 ms
GETALL DESKTOP	964 ms	790 ms
GETALL FAVORITOS	1140 ms	1144 ms
FRECUENTES	792 ms	975 ms
RECIENTES	616 ms	638 ms
CARGA PANEL INICIO	1,76 seg.	4,40 seg.
CARGA USUARIOS	2,0 seg.	8,62 seg.
CARGA CURSOS	1,75 seg.	7,90 seg.
CARGA MI ACTIVIDAD	2,89 seg.	7,05 seg.

Tabla 7: Pruebas de latencia, cuarta iteración. Elaboración propia (2022)

Versión App	1.0
SO	Windows 10
Tipo Conexión	Internet Wi-Fi Hogar
Desc. PC	PC Notebook Pablo
Tipo Servidor	Servidor Google (Santiago-Debian11-10GB SSD-4GB RAM)

	Instancia 1	**Instancia 2**	**Instancia 3**	**Instancia 4**
Velocidad conexión	250 Mbps	250 Mbps	250 Mbps	250 Mbps
Navegador	Firefox 106.0	Firefox 106.0	Chrome 106.0	Chrome 106.0
Idioma	Español	Inglés	Español	Inglés
CARGA DE LOGIN	514 ms	555 ms	255 ms	172 ms
CARGA MONO APP TOTAL	1,34 seg.	1,92 seg.	541 ms	498 ms
CARGA MONO DOM	1,07 seg.	1,47 seg.	467 ms	424 ms
CARGA MONO CSS	497 ms	1,42 seg.	116 ms	109 ms
CARGA JS	446 ms	532 ms	104 ms	78 ms
REFRESCARSESION	714 ms	1,77 seg.	60 ms	64 ms
GETALL DESKTOP	697 ms	1,77 seg.	44 ms	46 ms
GETALL FAVORITOS	670 ms	1,76 seg.	57 ms	54 ms
FRECUENTES	684 ms	1,75 s	38 ms	37 ms
RECIENTES	692 ms	1,74 seg.	34 ms	36 ms
CARGA PANEL INICIO	541 ms	841 ms	274 ms	249 ms
CARGA USUARIOS	908 ms	1,40 s	320 ms	283 ms
CARGA CURSOS	480 ms	1,31 seg.	202 ms	173 ms
CARGA MI ACTIVIDAD	606 ms	1,67 seg.	284 ms	290 ms

Tabla 8: Pruebas de latencia, quinta iteración. Elaboración propia (2022)

Versión App	1.1
SO	Windows 11
Tipo Conexión	Internet Wi-Fi UTalca Lab. Arrayán 3
Desc. PC	PC Notebook Benjamín
Tipo Servidor	Servidor Google (Santiago-Debian11-10GB SSD-4GB RAM)

	Instancia 1
Velocidad conexión	140 Mbps
Navegador	Firefox 106.1
Idioma	Español
CARGA DE LOGIN	396ms
CARGA MONO APP TOTAL	956 ms
CARGA MONO DOM	792 ms
CARGA MONO CSS	295 ms
CARGA JS	64 ms
REFRESCARSESION	32ms
GETALL DESKTOP	25ms
GETALL FAVORITOS	24ms
FRECUENTES	27ms
RECIENTES	20ms
CARGA PANEL INICIO	343 ms
CARGA USUARIOS	282 ms
CARGA CURSOS	118 ms
CARGA MI ACTIVIDAD	353 ms