**UNIVERSIDAD CATÓLICA DEL MAULE**

**Facultad de Ciencias de la Ingeniería** **Profesor Guía**

**Escuela de Ingeniería Civil Informática Dr. Marco Toranzo Céspedes**

**“COMPARTE CONOCIMIENTO UCM” DIFUSIÓN DE CONTENIDO E INTERACCIÓN UNIVERSITARIA**

**MANUEL ANTONIO CASTILLO GÓMEZ**

**NICOLAS ALEXIS SALINAS FLORES**

**Memoria para optar al**

**Título Profesional de**

**Ingeniero de Ejecución en Computación e Informática**

**TALCA, NOVIEMBRE 2022.**

**UNIVERSIDAD CATÓLICA DEL MAULE**

**FACULTAD DE CIENCIAS DE LA INGENIERÍA**

**ESCUELA DE INGENIERÍA CIVIL INFORMÁTICA**

**MEMORIA PARA OPTAR AL**

**TÍTULO PROFESIONAL DE**

**INGENIERO DE EJECUCIÓN EN COMPUTACIÓN E INFORMÁTICA**

**“COMPARTE CONOCIMIENTO UCM” DIFUSIÓN DE CONTENIDO E INTERACCIÓN UNIVERSITARIA**

**MANUEL ANTONIO CASTILLO GÓMEZ  
NICOLAS ALEXIS SALINAS FLORES**

**COMISIÓN EXAMINADORA FIRMA**

PROFESOR GUÍA

MARCO TORANZO CÉSPEDES \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROFESOR COMISIÓN

HUGO ARAYA CARRASCO \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROFESOR COMISIÓN

ROBERTO AHUMADA GARCÍA \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOTA FINAL EXAMEN DE TÍTULO \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TALCA, NOVIEMBRE 2022.**

**Agradecimientos.**

*Agradezco a Dios, a la Universidad Católica del Maule y a sus docentes involucrados en el programa de la prosecución de estudios que brindaron sus conocimientos para llegar a ser un profesional en el mundo de la informática. A mi profesor guía el Dr. Marco Toranzo Céspedes por brindar su apoyo, comprensión, sabiduría y conocimientos en este proyecto de memoria y en los ámbitos laborales que se presentaron en el camino.*

*Son muchas las personas a quienes me gustaría agradecer por su constante apoyo, por la eterna compañía, por la luz que, en momentos de oscuridad y duda durante el proceso, entregaron la revitalización necesaria para seguir adelante. Doy gracias al cielo porque cada uno de ellos siguen a mi lado afrontando los nuevos desafíos de la vida. Mi eterno agradecimiento a las personas más importantes de mi vida: mis padres, mis hermanos y mi prometida.*

**Sumario**

Las instituciones académicas están en permanente flujo de información y recursos, contribuyendo con las necesidades de las personas que, en estos tiempos, muestran un gran potencial académico en conjunto con las tecnologías.

Actualmente en la Universidad Católica del Maule (UCM), alumnos y académicos utilizan el sistema de gestión de aprendizaje online LMS, el cual se ha empleado para distribuir material académico por parte de docentes. Dicho sistema entrega una experiencia con los usuarios netamente académica, la cual es sumamente útil, pero que muchas veces no incentiva a los usuarios a compartir y crecer como comunidad universitaria.

Este proyecto de memoria expone el desarrollo de una aplicación multiplataforma que entrega las herramientas para interactuar entre los distintos tipos de usuarios, tales como, alumnos, académicos, y todo interesado perteneciente a todas las carreras y departamentos mediante la implementación de una red social académica, la cual dispondrá de mensajería instantánea privada, la posibilidad de crear avisos y preguntas entre usuarios, subir archivos directamente a la nube para su posterior descarga, permitiendo así la optimización de la comunicación e interacción dentro de la comunidad universitaria al tratarse de una aplicación (APP) intuitiva.

Se llevó a cabo una investigación de tecnologías, metodologías, y marcos de trabajo actuales, referentes al desarrollo de aplicaciones multiplataformas para proponer una solución óptima a implementar como una opción amigable al sistema actual. Basado en la evidencia, se determinó que la mejor alternativa a emplear en este proyecto es un modelo de datos relacional y arquitectura cliente/servidor, donde se utilizaron las tecnologías más recientes y eficientes como: PostgreSQL, Node.JS, Angular, Ionic, Trello, entre otras. Además de integraciones en la nube como con el sistema S3 de Amazon Web Services, cuyo aporte tecnológico contribuye a favor de las soluciones optimas al momento de cubrir las necesidades del proyecto a ejecutar.

**Índice General**

[Capítulo 1 : Introducción 1](#_Toc92435081)

[**1.1. Introducción.** 2](#_Toc92435082)

[**1.2.** **Descripción del problema.** 3](#_Toc92435089)

[**1.3.** **Objetivos.** 5](#_Toc92435089)

[**1.4.** **Motivación.** 5](#_Toc92435091)

[**1.5.** **Contribuciones del sistema.** 6](#_Toc92435094)

[**1.6.** **Alcance de la memoria.** 6](#_Toc92435097)

[**1.7.** **Organización de la memoria.** 7](#_Toc92435099)

[**1.8.** **Acrónimos y Abreviaturas.** 7](#_Toc92435099)

[Capítulo 2 : Marco Teórico. 10](#_Toc92435100)

[**2.1.** **Conceptos técnicos relevantes.** 11](#_Toc92435103)

[**2.2.** **Especificación de requerimientos del sistema.** 11](#_Toc92435104)

[**2.2.1.** **Requisitos Funcionales.** 11](#_Toc92435105)

[**2.2.2.** **Requisitos No Funcionales.** 14](#_Toc92435107)

[**2.3.** **UX/UI.** 15](#_Toc92435108)

[**2.4.** **Figma.** 15](#_Toc92435111)

[**2.5.** **Framework.** 16](#_Toc92435114)

[**2.6.** **Angular.** 16](#_Toc92435120)

[**2.7.** **Ionic.** 18](#_Toc92435133)

[**2.8.** **API** 19](#_Toc92435141)

[**2.8.1.** **API REST.** 20](#_Toc92435148)

[**2.8.2.** **Protocolo HTTP.** 21](#_Toc92435151)

[**2.8.3.** **Respuestas HTTP.** 22](#_Toc92435153)

[**2.9.** **NodeJs** 23](#_Toc92435141)

[**2.10.** **ORM** 23](#_Toc92435141)

[**2.11.** **Secuelize** 23](#_Toc92435141)

[**2.12.** **Socket.io** 25](#_Toc92435141)

[**2.13.** **PostgreSQL** 25](#_Toc92435141)

[**2.14.** **Postman** 26](#_Toc92435141)

[**2.15.** **MVC.** 27](#_Toc92435154)

[**2.16.** **VPS.** 28](#_Toc92435169)

[**2.17.** **AWS.** 28](#_Toc92435172)

[**2.18.** **Git.** 29](#_Toc92435175)

[**2.19.** **Trello.** 29](#_Toc92435180)

[**2.20.** **Netlify.** 30](#_Toc92435185)

[**2.21.** **Marco de trabajo SCRUM** 30](#_Toc92435187)

**[2.22.](#_Toc92435187)****[Justificación de Tecnologías.](#_Toc92435187)** [33](#_Toc92435187)

[Capítulo 3 : Marco de Trabajo y Propuesta de Solución 34](#_Toc92435194)

[**3.1.** **Análisis.** 35](#_Toc92435195)

[**3.2. Funcionalidad Organizacional de la Aplicación** 35](#_Toc92435211)

[**3.3. Modelo Relacional de la Base de Datos.** 37](#_Toc92435211)

[Capítulo 4 : Sistema Comparte Conocimientos UCM. 39](#_Toc92435212)

[**4.1. Backlog del Producto.** 40](#_Toc92435214)

* [**Definición del Modelo y Arquitectura de Datos.** 41](#_Toc92435220)
* [**Creación de Vistas del Sistema.** 41](#_Toc92435221)
* [**Interconexiones Entre el Cliente y el Servidor.**. 41](#_Toc92435222)
* [**Implementación con AWS.**. 41](#_Toc92435223)

[**4.2. Sprint Número Uno.** 41](#_Toc92435224)

[**4.2.1. Backlog del Sprint.** 41](#_Toc92435226)

* [**Diseñar el mapa de la aplicación.** 41](#_Toc92435228)
* [**Toma de ideas generales.**. 41](#_Toc92435229)
* [**Creación de un boceto o *Wireframe*.**. 42](#_Toc92435230)
* [**Traspaso del boceto a Figma.**. 42](#_Toc92435231)

[**4.2.2. Incremento.** 42](#_Toc92435232)

[**4.2.2.1. Mapa de la aplicación.** 42](#_Toc92435234)

[**4.2.2.2. Diseño UX/UI.** 43](#_Toc92435239)

[**4.2.3. Retrospectiva del Sprint..** 43](#_Toc92435239)

[**4.3. Sprint Número Dos.** 44](#_Toc92435224)

[**4.3.1. Backlog del Sprint.** 44](#_Toc92435226)

* [**Diseño del modelo relacional.** 44](#_Toc92435228)
* [**Normalización.**. 44](#_Toc92435229)
* [**Implementación mediante SQL.**. 44](#_Toc92435230)
* [**Testeo…**…………………………………………………………………………….44](#_Toc92435231)

[**4.3.2. Incremento.** 44](#_Toc92435232)

[**4.3.2.1. Modelo relacional** 45](#_Toc92435234)

[**4.3.2.2. Código SQL.** 46](#_Toc92435239)

[**4.3.2.3. Base de Datos Implementada.** 47](#_Toc92435239)

[**4.3.3. Retrospectiva del Sprint..** 47](#_Toc92435239)

[**4.4. Sprint Número Tres.** 48](#_Toc92435224)

[**4.4.1. Backlog del Sprint.** 48](#_Toc92435226)

* [**Creación Vistas de Inicio de Sesión.** 48](#_Toc92435228)
* [**Creación de la Navegación.**. 48](#_Toc92435229)
* [**Creación de la Vista Home**. 48](#_Toc92435231)
* [**Creación Vistas de Perfil.**. 48](#_Toc92435230)
* [**Creación Vistas de Avisos/Preguntas**. 48](#_Toc92435231)
* [**Creación Vistas Compartir**. 48](#_Toc92435231)
* [**Creación Vista Chat.**. 48](#_Toc92435229)
* [**Creación de Vistas Complementarias.**. 49](#_Toc92435230)

[**4.4.2. Incremento.** 49](#_Toc92435232)

[**4.4.2.1. Vistas de Control de Acceso** 49](#_Toc92435234)

[**4.4.2.2. Vista Principal y Sus Interacciones.** 51](#_Toc92435239)

[**4.4.2.3. Perfil de Usuario y Chat.** 53](#_Toc92435239)

[**4.4.2.4. Vistas Complementarias.** 54](#_Toc92435239)

[**4.4.3. Retrospectiva del Sprint..** 55](#_Toc92435239)

[**4.5. Sprint Número Cuatro** 55](#_Toc92435224)

[**4.5.1. Backlog del Sprint.** 55](#_Toc92435226)

* [**Implementación Base de la API.** 55](#_Toc92435228)
* [**Integración de ExpressJS y Sequelize.**. 55](#_Toc92435229)
* [**Implementación de Modelos**. 55](#_Toc92435231)
* [**Implementación de Controladores.**. 55](#_Toc92435230)
* [**Implementación de Rutas**. 55](#_Toc92435231)
* [**Configuración de CORS y HTTPS.**. 56](#_Toc92435229)
* [**Integración con AWS.**. 56](#_Toc92435230)
* [**Integración con Socket.IO**. 56](#_Toc92435231)
* [**Implementación de Servicios Angular.**. 56](#_Toc92435229)
* [**Implementación de Componentes Angular**. 56](#_Toc92435230)

[**4.5.2. Incremento.** 56](#_Toc92435232)

[**4.5.2.1. Creación de la API** 56](#_Toc92435234)

[**4.5.2.2. Conexión a la Base de Datos.** 58](#_Toc92435239)

[**4.5.2.3. Seguridad.** 61](#_Toc92435239)

[**4.5.2.4. Integraciones.** 62](#_Toc92435239)

[**4.5.2.5. Implementación en el Cliente.** 65](#_Toc92435239)

[**4.5.3. Retrospectiva del Sprint..** 66](#_Toc92435239)

[**4.6. Sprint Número Cinco** 67](#_Toc92435224)

[**4.6.1. Backlog del Sprint.** 67](#_Toc92435226)

* [**Configuración del VPS.** 67](#_Toc92435228)
* [**Pruebas de conexión.**. 67](#_Toc92435229)
* [**Despliegue en Netlify**. 67](#_Toc92435231)
* [**Pruebas de la Aplicación.**. 67](#_Toc92435230)

[**4.6.2. Incremento.** 67](#_Toc92435232)

[**4.6.2.1. Contabo VPS** 68](#_Toc92435234)

* [**Configuración del sistema Ubuntu.** 69](#_Toc92435228)
* [**Instalación del Software Necesario.**. 69](#_Toc92435229)
* [**Implementación de la Base de Datos**. 69](#_Toc92435231)
* [**Clonar Repositorio Github de la API**. 69](#_Toc92435230)

[**4.6.2.2. Pruebas con Postman** 69](#_Toc92435239)

[**4.6.2.3. Netlify y Pruebas funcionales.** 70](#_Toc92435239)

[**4.6.3. Retrospectiva del Sprint..** 73](#_Toc92435239)

[**4.7. Información general final** 73](#_Toc92435224)

[Capítulo 5 : Conclusiones. 75](#_Toc92435243)

[**5.1. Conclusión de las herramientas.** 76](#_Toc92435248)

[**5.2.** **Conclusión del marco de trabajo.** 76](#_Toc92435250)

[**5.3.** **Conclusiones generales.** 76](#_Toc92435252)

[**5.3.** **Trabajos Futuros.** 76](#_Toc92435252)

[Bibliografía. 79](#_Toc92435253)

* [**Citas de Enlaces bibliográficos:** 80](#_Toc92435254)
* [**Citas de Textos bibliográficos:** 82](#_Toc92435275)

**Índice de Figuras**

[**Figura 2.1** Componente de la página “home” del proyecto. 17](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634145) [**Figura 2.2** Código fuente en la creación de una carta con Ionic. 19](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 2.3** Representación de funcionamiento de una API.. 20](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 2.4** Esquema de comunicación entre los sistemas pertenecientes al Backend... 24](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 2.5** Gráfica con los tiempos de ejecución de PostgreSQL frente a MongoDB.... 26](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 2.6** Proceso del marco de trabajo Scrum..... 31](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 3.1** Organización y distribución modular de Comparte Conocimientos UCM..... 35](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 3.2** Base de datos relacional de Comparte Conocimientos UCM...... 38](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.1** Backlog del producto para Backend y Frontend con la herramienta Trello....... 40](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.2** Mapa de la aplicación Comparte Conocimientos UCM........ 42](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.3** Segmento de diseño preliminar aplicación Comparte Conocimientos UCM........ 43](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.4** Modelo Relacional hasta su tercera forma normal......... 45](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.5** Creación de base de datos mediante pgAdmin 4......... 46](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.6** Fragmento de código SQL correspondiente a la base de datos del proyecto. 47](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.7** Vistas correspondientes al control de acceso del portal Comparte Conocimientos UCM.. 50](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.8** Inicio de sesión en versión de escritorio al portal Comparte Conocimientos UCM.. 50](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146)

[**Figura 4.9** Barras de navegación junto a diferentes vistas pertenecientes al sistema Comparte Conocimientos UCM.. 52](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.10** Versión de escritorio correspondiente a la interfaz visual de “Compartir Archivos”.. 52](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146)

[**Figura 4.11** Vistas correspondientes al perfil de usuario y chat, pertenecientes a la aplicación Comparte Conocimientos UCM.. 53](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.12** Vistas correspondientes a favoritos, búsquedas y contacto pertenecientes al sistema Comparte Conocimientos UCM. 54](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) **[Figura 4.13](file:///C:\\Users\\JulioFdo\\Desktop\\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx" \l "_Toc57634146)** [Estructura básica del Servidor. 57](file:///C:\\Users\\JulioFdo\\Desktop\\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx" \l "_Toc57634146) [**Figura 4.14** Directorios creados por Sequelize.. 58](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.15** Modelo lógico de la API REST... 59](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) **[Figura 4.16](file:///C:\\Users\\JulioFdo\\Desktop\\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx" \l "_Toc57634146)** [Función del Controlador perteneciente al Modelo Usuarios... 60](file:///C:\\Users\\JulioFdo\\Desktop\\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx" \l "_Toc57634146) [**Figura 4.17** Rutas pertenecientes al Controlador de Usuarios.... 60](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.18** Integración de CORS en la API REST de Comparte Conocimientos UCM..... 61](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.19** Implementación del protocolo HTPPS en la API...... 62](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.20** Conexión emisor – receptor NodeJs....... 63](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.21** Conexión emisor – receptor Ionic Angular........ 64](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.22** Integración de la API con AWS......... 64](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.23** Servicio perteneciente a la vista Home de la aplicación Comparte Conocimientos UCM.......... 65](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.24** Componente perteneciente a la vista home de la aplicación CCUCM......... 66](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.25** Interfaz perteneciente al VPS de Contabo.......... 68](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.26** Test correspondiente a petición GET a la API REST del proyecto........... 70](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.27** Interfaz perteneciente a la herramienta Netlify............ 70](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.28** Pruebas de control de acceso a plataforma Comparte Conocimientos UCM........ 71](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146) [**Figura 4.29** Prueba de subida de archivos a la plataforma de Comparte Conocimientos UCM……………………………………………………………………………………….….72](file:///C:\Users\JulioFdo\Desktop\Final_version_MATC_REVISION____4_Noviembre19___NewMemory.docx#_Toc57634146)

**Índice de Tablas**

[**Tabla 1.1** Acrónimos y Abreviaturas usadas a lo largo de este documento 7](#_Toc57634183) [**Tabla 2.1** Requisitos funcionales de la APP. 12](#_Toc57634184) [**Tabla 2.2** Requisitos no funcionales de la APP. 14](#_Toc57634185)

# **: Introducción**

Este capítulo presenta la perspectiva global, problemática, solución propuesta, motivaciones, contribuciones, los objetivos relacionados, además de la contribución a la Universidad Católica del Maule.

* 1. **Introducción**

La UCM es una institución educacional superior tradicional privada, acreditada por la Comisión Nacional de Acreditación (CNA), organismo de carácter público y autónomo que tiene como fin verificar y promover la calidad de las instituciones de educación superior en Chile y los programas que estos imparten. La universidad cuenta con la Facultad de Ciencias de Computación ubicada en la ciudad de Talca, dentro de la cual imparte la carrera de Ingeniería Civil Informática, junto a sus salidas intermedias, como lo es Ingeniería Ejecución en Computación e Informática, la cual surgió como consecuencia lógica de la carrera de Técnico Universitario en Programación Computacional que históricamente se venía desarrollando desde 1989.

En la actualidad, tanto alumnos como académicos de la Universidad Católica del Maule cuentan con acceso a la plataforma UCM Virtual o Sistema de Gestión de Aprendizaje (LMS) el cual permite a los docentes compartir material referente a los distintos cursos que estos imparten. Por otro lado, la comunidad universitaria cuenta con el sistema SIBIB, el cual corresponde al sistema de bibliotecas de la universidad, que ofrece una amplia colección bibliográfica, en formato físico como electrónico, para satisfacer las distintas necesidades académicas de dichos usuarios.

Sin embargo, las herramientas a disposición en la UCM y la situación sanitaria actual han puesto en evidencia nuevas demandas tecnológicas que apremia implementar. Debido a las nuevas necesidades y herramientas expuestas anteriormente, los alumnos se encuentran cada vez más desconectados con respecto a sus grupos de estudio, al compartir material académico, e incluso la prestación de servicios tales como tutorías. Considerando el contexto anterior, es oportuno la creación de una plataforma denominada “Comparte Conocimientos UCM”, la cual permita a los estudiantes, docentes y todo interesado, formar una comunidad educacional a través de avisos de todo tipo, creando preguntas de diversas temáticas referentes al ámbito universitario, compartir archivos institucionales tales como libros, informes, pruebas, guías, entre otros. Además de la posibilidad de comunicarse internamente mediante la creación de un CHAT privado.

* 1. **Descripción del Problema**

Las plataformas disponibles para todo el alumnado de la Universidad Católica del Maule, tales como el LMS y el sistema de bibliotecas UCM (SIBIB) ofrecen una variedad de herramientas que aportan continuamente en el proceso de formación de cada uno de los futuros profesionales pertenecientes a la Universidad Católica del Maule. Dentro de dicha variedad de instrumentos, se encuentra la posibilidad de la entrega de información, evaluación y administración de los procesos académicos, así como una amplia colección de libros y revistas tanto en formato físico como digital. Con motivo de esclarecer la problemática se detectó que dichas herramientas son limitadas en su forma de compartir la información, ya que, en la actualidad el material solo es proporcionado por el docente a cargo de una asignatura o por el organismo que compone el SIBIB.

Disponer del material del UCM Virtual implica que los estudiantes deben estar matriculados en las asignaturas de la malla curricular de la carrera de interés. Este requerimiento supone una limitante para el usuario, lo que trae por consecuencia que, no cualquier persona puede compartir información en la plataforma. Esto también queda sujeto a los tiempos en los cuales está accesible el material, ya sea por la disponibilidad del académico, o por la naturaleza temporal de los archivos en el sistema.

En el contexto de la contingencia sanitaria a nivel mundial, los estudiantes han debido enfrentar nuevos desafíos referentes a la pérdida temporal de la posibilidad de organización en grupos de estudios junto con la provisión de la prestación de servicios académicos (tutorías, apoyo estudiantil, entre otros), así como de disfrutar de diversas actividades extracurriculares. Este fenómeno afecta directamente a la salud mental de un gran número de jóvenes universitarios (en especial a quienes ingresan a su primer año), ya que la privación social, como medida de adaptación a la situación de pandemia, es de gran impacto cuando dichos individuos se encuentran en un período caracterizado por una mayor necesidad de contacto con otras personas, debiendo amoldarse a la modalidad de enseñanza online. Tanto en este grupo etario como en otros, la sociabilización entre pares es muy importante, donde además se crean y se afiatan los vínculos en un ambiente donde los estudiantes desde sus inicios comparten experiencias comunes, que les permiten conocer y avanzar en la vida universitaria. Dicha adaptación se complejizó aún más debido a que el sistema educacional chileno no estaba adaptado para la enseñanza a distancia. Según el estudio de Imhay, un 47% de estudiantes reconoció dificultades para acceder y seguir las clases en modalidad online, y un "82,5% reportó haber sufrido problemas de concentración en los estudios y actividades cotidianas" [HER2021].

Si bien, es cierto que cada uno de los alumnos tienen acceso a las diferentes plataformas de Redes Sociales (RRSS) disponibles en la actualidad, esto significa organización y tiempo extra para poder reunirse con su comunidad universitaria, además de considerar que no todos los usuarios están abiertos a exponer sus datos y sucesos personales con su ambiente académico. Cabe mencionar que, debido al gran flujo de información en Redes Sociales, junto con su rapidez de propagación, es complejo asegurar la veracidad de la información, donde los únicos responsables de asegurar y acreditar dicha información son los usuarios [CAS2016].

En resumen, los puntos de mayor interés a tratar en la problemática son:

* Restricción para compartir material o información académica, ya sea formal o informal, la cual fuese duradera en el tiempo.
* La existencia de un medio institucional virtual que permita interactuar con la comunidad universitaria de manera pública o privada, y de forma segura.
* La falta de espacios digitales académicos para exponer consultas, tanto donde los estudiantes, como también los docentes sean quienes puedan responder, creando una dinámica de comunicación transparente.
* La posibilidad de poseer un canal digital, donde los diferentes usuarios puedan publicar avisos de diferente índole, como, por ejemplo, tutorías, eventos sociales, compra/venta de artículos, entre otros.
* Poseer de forma práctica un solo “lugar” donde poder concentrar información y crear una comunidad de forma segura y fidedigna.

Como conclusión a la problemática, Comparte Conocimientos UCM entrega un espacio digital donde los usuarios pueden crear comunidades junto a sus compañeros, no solo de carrera, sino que, de todo el conglomerado de carreras universitarias que conforman la Universidad Católica del Maule. Además, ser una aplicación multiplataforma que brinde diversas funcionalidades como compartir documentos, publicar avisos, crear preguntas, comunicación directa mediante un chat dentro de un mismo lugar, incentiva a los usuarios a mejorar el flujo de información y comunicación con los diferentes tipos de usuarios, tales como: estudiantes y docentes.

* 1. **Objetivos**

La conformación de este proyecto tiene por finalidad los siguientes objetivos:

* **Objetivo general:** Desarrollar una aplicación multiplataforma que permita crear comunidad, comunicación transparente y un flujo de información duradero para los estudiantes y docentes de la Universidad Católica del Maule
* **Objetivos específicos:** Con el fin de satisfacer el objetivo general anteriormente descrito, se crean los siguientes objetivos específicos:
* Crear un diseño UX/UI responsivo.
* Identificar requerimientos funcionales y no funcionales.
* Aplicar el marco de trabajo Scrum para fomentar el trabajo en equipo, asegurando el correcto desarrollo del software.
* Integrar tecnologías de Front-End y Back-End para el desarrollo modular de la APP.
* Aplicar métodos para la producción de una salida móvil del objetivo general.
  1. **Motivación**

En la actualidad, la contingencia sanitaria sigue siendo un factor importante en el día a día de muchos estudiantes y docentes en la Universidad Católica del Maule, ya que esta ha afectado profundamente en cómo se llevan a cabo diferentes actividades, algunas tan simples como lo es la comunicación. En contexto de lo anteriormente expuesto es que se debe tener a la disposición de cada uno de los usuarios una plataforma que les permita no solo compartir información en diferentes formatos, sino que también comunicarse de manera pública y privada. Esta mejora supone ser un gran aporte para reforzar el sentimiento de compañerismo, que por consecuencia ayuda en el crecimiento educacional de cada uno de los miembros de la comunidad.

Si bien es cierto que hoy en día existen diferentes plataformas para poder comunicarse, compartir archivos, y generar grupos, desde la experiencia y comentarios de diferentes personas, no siempre son los más idóneos, siendo esto por la poca privacidad, por la dudosa veracidad de la información o simplemente por ser un mundo externo a su casa de estudios. Es de esta problemática, surgida por la situación actual de pandemia y los años experimentados como alumnos de la UCM, que surge la idea de crear un portal multiplataforma que contribuya a las diversas necesidades de estudiantes y docentes.

* 1. **Contribuciones del Sistema**

La principal contribución de la plataforma Comparte Conocimientos UCM es facilitar la comunicación transparente de los diversos usuarios pertenecientes a la Universidad Católica del Maule, entregando herramientas como el albergar y descargar diferentes tipos de documentos en y desde la nube. Además de relacionarse con estudiantes y docentes mediante avisos y preguntas, junto con la posibilidad de comunicarse de manera privada entre sí.

La aplicación no pretende reemplazar a los actuales sistemas internos de la UCM, como los son el LMS o el sistema de biblioteca, ni mucho menos a portales externos que permiten la comunicación entre usuarios. El sistema está creado con la intención de entregarle a los estudiantes y académicos un lugar “propio”, donde poder tener al alcance diferentes herramientas que apoyen la experiencia universitaria en tiempos de pandemia, y en un futuro, situaciones de educación remota.

* 1. **Alcance de la Memoria**

Dentro del proyecto de memoria se llevó a cabo el diseño y desarrollo de una aplicación multiplataformas de índole social y académico para los estudiantes y docentes de la Universidad Católica del Maule, la cual es capaz de almacenar contenido educativo suministrado por los usuarios en diversas categorías, además de la posibilidad de publicar avisos y preguntas, junto con una herramienta de chat. Para ello, fueron integradas tecnologías actuales de almacenamiento en la nube, diseño de interfaces atractivo e intuitivo, entre otras características que brindan un sistema amigable, seguro y sumamente útil.

* 1. **Organización de la Memoria**

Este documento se compone de los siguientes capítulos:

* **Capítulo 1:** Introducción y descripción de la problemática, objetivo general y específicos del proyecto, motivación, contribuciones del sistema y alcance del proyecto de memoria.
* **Capítulo 2:** Exposición del marco teórico, donde se detalla conceptos importantes de las tecnologías informáticas (TI) utilizadas en la creación de la aplicación, que entrega solución a la problemática planteada Capítulo 1.
* **Capítulo 3:** Sección queexplica la disposición de los elementos utilizados para la solución de la problemática expuesta en el Capítulo 1,
* **Capítulo 4:** Apartado que abarca la puesta en marcha de las metodologías y tecnologías presentadas a lo largo del documento.
* **Capítulo 5:** Manifiesta de forma resumida las principales conclusiones, trabajos futuros y algunas consideraciones con respecto al proyecto de memoria.
  1. **Acrónimos y Abreviaturas**

A continuación, se muestra una Tabla explicativa con los acrónimos y abreviaturas utilizadas a lo largo de esta memoria para facilitar la comprensión del documento:

**Tabla 1.1:** *Abreviaturas y acrónimos utilizadas en esta memoria.*

|  |  |  |
| --- | --- | --- |
| **Ítem** | **Acrónimo** | **Significado** |
| 1 | UCM | Universidad Católica del Maule. |
| 2 | LMS | UCM Virtual. |
| 3 | SIBIB | Sistema de Bibliotecas de la Universidad Católica del Maule. |
| 4 | API | Interfaz de Programación de Aplicaciones. |
| 5 | CRUD | C=Registrar, R=Consultar, U=Actualizar y D=Borrar |
| 6 | HTTP | Protocolo de transferencia de hipertexto. |
| 7 | HTML | Lenguaje de marcas de hipertexto. |
| 8 | TI | Tecnologías informáticas. |
| 9 | TIC | Tecnología de la información y comunicación. |
| 10 | BD | Base de datos. |
| 11 | CHAT | Mensajería instantánea en tiempo real. |
| 12 | RRSS | Redes sociales. |
| 13 | CNA | Comisión Nacional de Acreditación. |
| 14 | CC | Comparte conocimiento UCM. |
| 15 | MVC | Modelo - Vista – Controlador. |
| 16 | RF | Requisitos funcionales. |
| 17 | RNF | Requisitos no funcionales. |
| 18 | URL | Localizador de recursos uniformes. |
| 19 | OS | Sistema operativo. |
| 20 | Web | Word Wide Web. |
| 21 | CSS | Hojas de estilo en cascada. |
| 22 | JS | JavaScript. |
| 23 | TS | TypeScript. |
| 24 | BDOO | Base de Datos relacional orientada a objetos. |
| 25 | JSON | Notación de objetos JS. |
| 26 | XML | Lenguaje de Marcado Extensible. |
| 27 | AWS | Amazon Web Services. |
| 28 | ORM | Objeto de Mapeo Relacional. |
| 29 | VPS | Servidor Virtual Privado. |
| 30 | UX/UI | Experiencia de usuario. / Interfaz de usuario. |

**Fuente*:*** *Elaboración propia.*

# **: Marco Teórico**

Se presentan los principales conceptos referentes a las diversas tecnologías empleadas para el desarrollo del portal multiplataformas.

1. 1. **Conceptos Técnicos Relevantes**

El desarrollo de una aplicación que brinda soluciones informáticas conlleva una variedad de procesos, técnicas de desarrollo, metodologías de trabajo y gestiones.

En el proceso de desarrollo de la aplicación Comparte Conocimientos UCM, se utilizaron diferentes tecnologías y terminologías, entre las cuales es de gran relevancia destacar conceptos como UX/UI, Figma, Framework, Angular, Ionic, API, REST, NodeJs, Sequelize, ORM, Socket.io, PostgreSQL, Postman, MVC, VPS, AWS, Git, Trello y Netlify.

* 1. **Especificación de Requerimientos del Sistema**

Para llevar a cabo un desarrollo eficiente y ordenado de un producto de software, es necesario definir de forma clara y concisa los requerimientos necesarios para su posterior implementación.

En la siguiente sección se mencionan los requerimientos funcionales y no funcionales más destacados del sistema, los cuales aportan en la comprensión de los puntos clave en el desarrollo de la plataforma Comparte Conocimientos UCM. Estos se reflejan tanto en el diseño del cliente, como del servidor, transformándose así en un pilar fundamental en el proceso de desarrollo.

* + 1. **Requisitos Funcionales**

Los requisitos funcionales son aquellos que detallan el comportamiento explicito que debe realizar la aplicación o producto de software, además de la información que debe manejar. A su vez, son considerados el usuario y cualquier otro sistema externo que posea alguna interacción con el sistema. Estos, por otro lado, indican situaciones que el sistema no debe realizar.

Los requisitos funcionales se componen de los siguientes elementos que se detallan a continuación:

* **Número de requisito:** Señala la indexación asociada a los requisitos. No necesariamente los desarrolladores deben seguir el orden de la numeración.
* **Título:** Texto descriptivo y acotado que identifica textualmente al requisito.
* **Descripción:** Breve explicación de la funcionalidad asociada al requisito en cuestión.

A continuación, se muestra la tabla 2.1, la cual muestra los requisitos funcionales utilizados para el desarrollo de la plataforma Comparte Conocimientos UCM:

**Tabla 2.1:** *Requisitos funcionales de la APP.*

|  |  |  |
| --- | --- | --- |
| **Número requisito** | **Título** | **Descripción** |
| RF1 | Creación de usuarios | La aplicación dispondrá de un apartado donde los usuarios podrán crear cuentas nuevas mediante una serie de datos.  Los usuarios podrán emplear la red social si aceptan los términos y condiciones de uso. Solo si aceptan el acuerdo podrán utilizar la plataforma. En el documento se enumeran los tipos de denuncias entre usuarios, como, por ejemplo, afirmaciones racistas.  Todos los usuarios podrán denunciar diferentes elementos.  El administrador dispondrá de un listado de denuncias, para así determinar la sanción respectiva al caso. |
| RF2 | Recuperar contraseñas | La aplicación dispondrá de un apartado donde cada usuario, mediante su correo electrónico, podrá solicitar recuperar su contraseña en caso de perderla. |
| RF3 | Autenticación de usuarios | La aplicación dispondrá de un sistema para que los usuarios puedan autenticar sus datos, para de esta forma lograr ingresar el contenido de la aplicación. |
| RF4 | Creación de cursos | La aplicación tendrá la capacidad de que un usuario del tipo administrador pueda hacer CRUD de los diferentes cursos. |
| RF5 | Subida de archivos | La aplicación dispondrá de un sistema con la capacidad de permitir a los usuarios subir archivos desde sus ordenadores o aparato móvil, para posteriormente, ser almacenados y expuestos en las secciones pertenecientes a la aplicación.  Todos los archivos dispondrán de una fecha de caducidad. El sistema dispondrá de parámetros donde se estipulará la cantidad de días para el proceso de eliminado. |
| RF6 | Cursos favoritos | La aplicación dispondrá de un sistema que permita a los usuarios agregar/eliminar diferentes cursos de interés a una sección de favoritos. |
| RF7 | Funciones de perfil de usuario | La aplicación dispondrá de un sistema donde el usuario podrá almacenar/modificar/eliminar en su perfil diferente información personal, tal como: nombre completo, carrera, año de ingreso, foto de perfil, entre otros. |
| RF8 | Sistema de avisos | La aplicación dispondrá de un sistema de avisos, donde cada usuario podrá escribir mensajes donde da aviso de diferentes temas de interés. Estos temas podrán ser comentados con un sistema de foro. |
| RF9 | Sistema de chat | La aplicación dispondrá de un sistema de chat, donde será posible entablar comunicación entre diferentes usuarios mediante mensajes privados. |
| RF10 | Sistema de preguntas | La aplicación dispondrá de un sistema donde cada usuario podrá generar preguntas respecto a una determinada temática. Los demás usuarios podrán responder mediante un sistema de foro. |
| RF11 | Contacto | La aplicación dispondrá de un sistema de contacto mediante la disposición de un formulario, el cual permitirá comunicarse con el administrador(es) de aplicación. |
| RF12 | Inspeccionar cursos | La aplicación permitirá a los usuarios inspeccionar los cursos, donde será posible visualizar los diferentes archivos almacenados. |
| RF13 | Descarga de archivos | La aplicación permitirá a los usuarios descargar los archivos subidos por otros usuarios. |

**Fuente:** *Elaboración propia.*

* + 1. **Requisitos No Funcionales**

Los requisitos no funcionales son aquellos que no hacen referencia directa a las funcionalidades del sistema, sino que, más bien explican las propiedades del sistema, tales como son la seguridad, rendimiento, tiempos de espera, entre otros. En ocasiones, estos también explican la accesibilidad, como los tipos de dispositivos donde se dispondrá del proyecto en cuestión.

Los requisitos no funcionales se componen de los mismos elementos utilizados para la confección y descripción de los requisitos funcionales, tales como: Número de requisito, título y por último su respectiva descripción. A continuación, se expone la tabla 2.2, la cual describe los requisitos no funcionales utilizados en la implementación de la plataforma Comparte Conocimientos UCM:

**Tabla 2.2:** *Requisitos no funcionales de la APP.*

|  |  |  |
| --- | --- | --- |
| **Número requisito** | **Título** | **Descripción** |
| RNF1 | Control de acceso | La aplicación no permitirá que cualquier usuario ingrese a su contenido. Los usuarios deben estar registrados para poder ingresar a sus vistas y funcionalidades. |
| RNF2 | Tiempos de espera bajos | La aplicación entregará tiempos de respuesta bajos, tanto en el ingreso de datos, como su navegación y sus funcionalidades. |
| RNF3 | Versión móvil | La aplicación contará con una versión móvil disponible tanto en sistemas Android, sistemas iOS, entre otros. |
| RNF4 | Almacenamiento | La aplicación será capaz de almacenar todos los archivos, avisos, preguntas y respuestas, además de información de los usuarios. |
| RNF5 | Comunicación | La aplicación está diseñada para facilitar la comunicación entre los diferentes usuarios, por lo que el sistema de chat y foros debe ser accesible e intuitivo. |
| RNF6 | Accesibilidad | La aplicación será capaz de mostrar todos los archivos de forma ordenada e intuitiva, para de esta forma entregar al usuario la mejor experiencia posible. |

**Fuente:** *Elaboración propia.*

* 1. **UX/UI**

Según el artículo escrito sobre el tema en la revista *Tecnología en Marcha,* se define la experiencia de usuario (UX – *user experience*) como la forma en que el usuario específicamente experimenta la interfaz de usuario (UI – *user interface*), que en palabras simples es la interfaz visual del producto de software en cuestión [RAM2017].

Estos conceptos van de la mano, ya que, para construir un diseño atractivo y liviano para el usuario, es necesario enfocarse en las necesidades de este último. No es idóneo tener un diseño llamativo si su experiencia es frustrante. Por lo tanto, el foco principal al momento de construir un software es que su diseño esté enfocado en que el usuario logre sus objetivos mediante el uso de la herramienta de una forma sencilla y amigable. Es por esto por lo que, en el desarrollo de este proyecto de memoria, se enfocó en una experiencia multiplataforma, dando gran realce al diseño, para atraer a los distintos tipos de usuarios de esta aplicación mediante la entrega de soluciones más simples.

* 1. **Figma**

Figma es un editor de gráficos vectoriales y una herramienta de prototipado [CIR2020]. En otras palabras, Figma es un poderoso editor de gráficos de código abierto, además de una muy útil herramienta de prototipos con un enfoque en el diseño Web. Entre sus características se encuentra la posibilidad de trabajar en grupos de trabajos de manera simultánea desde diferentes dispositivos, como un navegador o su aplicación de escritorio.

Esta herramienta entrega una gran variedad de complementos para el diseño UX/UI, y al ser de acceso libre a todo público, se convierte en una plataforma práctica y a la medida para el diseño de una aplicación multiplataformas.

* 1. **Framework**

Para la mejor comprensión de futuras secciones, es necesario que el lector obtenga el conocimiento de algunos términos referentes a las herramientas utilizadas para el desarrollo de la aplicación. Entre estos se encuentra el concepto de *framework*, que corresponde a una variada colección de clases abstractas y objetos, además de ser un conjunto de patrones de diseño que facilitan el trabajo reutilizando código, siendo de esta forma la mejor opción para la producción eficiente y eficaz. El uso de estas herramientas agiliza en gran medida la creación de software, ya que se optimiza el tiempo de desarrollo, además de entregar una buena consistencia en las aplicaciones.

Para la materialización de Comparte Conocimientos UCM, se utilizaron los siguientes *frameworks*:

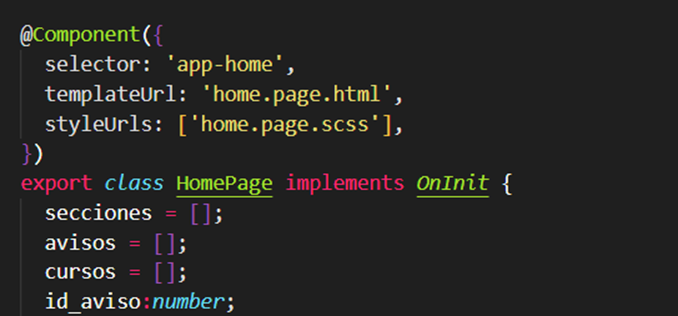
* Ionic, como *framework* Front-End.
* Angular, como *framework* JavaScript/TypeScript.
* ExpressJS, como *framework* de NodeJs.
  1. **Angular**

El *framework* Angular es considerado como una herramienta diseñada para el desarrollo eficiente y sofisticado de aplicaciones de una página (*single-page apps*). Crea aplicaciones móviles nativas con estrategias de desarrollo como Cordova, Ionic o NativeScript [ANG2020]. Angular es generalmente utilizado en proyectos robustos, los cuales poseen un crecimiento a largo plazo exponencial. Esto lo hace ideal para proyectos que nacen con un solo desarrollador y luego son adoptados por grandes empresas.

La plataforma de desarrollo de Angular está construida en base de TypeScript, aunque también hace referencia al uso de JavaScript, HTML, CSS, entre otros. Entre sus principales características se encuentran:

* Ser un *framework* basado en componentes, generando aplicaciones web escalables.
* Poseer una colección bien integrada de librerías, las cuales complementan diferentes funcionalidades como lo son el enrutamiento, formularios, comunicación cliente-servidor, entre otras.
* Herramientas complementarias que ayudan en el desarrollo, construcción y testeo de las aplicaciones desarrolladas.

Dentro de la amplitud de sus características más importantes, para la comprensión integral del desarrollo del proyecto de memoria, se encuentran los componentes anteriormente mencionados. Los componentes son esencialmente los bloques de código que componen la aplicación. Estos incluyen clases escritas en TypeScript, una plantilla HTML y algunos estilos CSS. A continuación, se expone la figura 2.1, la cual muestra un componente propio del proyecto Comparte Conocimientos UCM:

****Figura 2.1:** *Componente de la página “home” del proyecto.* **Fuente:** *Elaboración propia.*

La ventaja de utilizar componentes en el desarrollo de aplicaciones de este tipo es que ofrecen un fuerte encapsulamiento, lo cual permite un desarrollo intuitivo del proyecto en cuestión.

* 1. **Ionic**

Ionic es un *framework* Frontend de código abierto, el cual utiliza herramientas UI para crear aplicaciones multiplataforma haciendo uso de tecnologías Web. Este posee integraciones en los *framework* más populares tales como Angular, React y Vue. Ionic se centra en la interfaz de usuario y su experiencia de uso al utilizar la aplicación, además de controladores, interacciones, gestos y animaciones [ION2013].

Una de las principales ventajas al momento de elegir a Ionic como *framework* de Frontend es la capacidad de crear versiones del software tanto Web, de escritorio, Android y iOS con un solo código fuente. Esto es gracias a la integración dentro del mismo entorno de trabajo de Capacitor, el cual es una herramienta de código abierto para la creación nativa de aplicaciones multiplataformas.

El desarrollo con Ionic está basado en los diseños UX/UI anteriormente creados, ya que brinda todas las herramientas para crear proyectos con estilos adaptativos, siendo así para los usuarios, las aplicaciones visualmente atractivas en todas sus versiones. La elección de este *framework* como parte fundamental de Comparte Conocimientos UCM, es por su posibilidad de entregar soluciones en diversas plataformas, lo cual entrega a los usuarios la capacidad de permanecer conectados donde ellos prefieran sin importar el dispositivo que estén utilizando.

A continuación, con fines explicativos se ilustra la figura 2.2, la cual corresponde a un fragmento de código implementando en Ionic:

Texto

Descripción generada automáticamente

**Figura 2.2:** *Código fuente en la creación de una carta con Ionic.* **Fuente:** *Elaboración propia.*

* 1. **API**

El término API (Interfaz de Programación de Aplicaciones) cada día cobra más importancia cuando se habla de soluciones y desarrollos tecnológicos. Instituciones y empresas en múltiples áreas se enfrentan a la necesidad de construir o utilizar una API, ya sea para ofrecer un servicio o para mejorar el desempeño de alguna operación [ISA2018]. En otras palabras, se establece como un conjunto de definiciones y protocolos, los cuales son utilizados para desarrollar e integrar software a diferentes aplicaciones. Estas permiten la comunicación con otros sistemas mediante sus productos y servicios, sin la necesidad de que se conozca la implementación de esta misma.

Su ventaja está basada en permitir facultar la comunicación con diferentes sistemas, entregando funcionalidades hospedadas en infraestructuras en la nube. Esto a su vez ofrece un cierto nivel de seguridad, ya que al solo poder consumir servicios que entregan funcionalidades, el usuario no correrá el riesgo de modificaciones en el código.

Entre las arquitecturas más utilizadas para el desarrollo de una API es posible identificar SOAP (Protocolo de Acceso a Objetos Simples) y SOA (Arquitectura Orientada a Servicios), además de la arquitectura orientada a microservicios.

En el diseño del proyecto de memoria, se utilizó la arquitectura cliente – servidor, y es en el lado del servidor (Backend) que se desarrolla a modo de API, bajo la arquitectura SOA y la especificación de REST.

Para facilitar la comprensión del funcionamiento de las API, se muestra la figura 2.3, la cual es un diagrama explicativo:

Diagrama

Descripción generada automáticamente

**Figura 2.3:** *Representación de funcionamiento de una API.* **Fuente:** *Elaboración propia.*

* + 1. **API REST**

Una API de REST, o API de RESTful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. El informático Roy Fielding es el creador de la transferencia de estado representacional (REST) [RED2020].

Estas cuentan con las características de poseer una arquitectura cliente – servidor, la cual es gestionada mediante solicitudes del protocolo HTTP. Esta comunicación no posee estado, por lo cual la información no se almacena entre las solicitudes, siendo cada una de ellas independientes y desconectadas del resto. Por otro lado, los datos contenidos en las solicitudes pueden ser almacenados en caché, ayudando a las interacciones entre el cliente y el servidor.

* + 1. **Protocolo HTTP**

El protocolo HTTP (Protocolo de Transferencia de Hipertexto) corresponde a un [procedimiento de comunicación](https://definicion.de/protocolo-de-comunicacion/) que se utiliza para el tráfico de información por la Web. Esta es la base de cualquier intercambio de datos en Internet, siguiendo el modelo cliente-servidor, lo cual infiere en que una petición de datos es iniciada por el elemento que recibirá los datos, normalmente mediante un navegador Web [MDN2021].

El protocolo HTTP incluye ocho diferentes tipos de métodos. No obstante, son destacables los más utilizados, los cuales son empleados en la plataforma Comparte Conocimiento UCM.

A continuación, se entrega una breve definición de los métodos más utilizados pertenecientes al protocolo:

* **GET:** Mediante este método, se realiza una solicitud de un recurso (información) específica, la cual está determinada en la URL. Como un ejemplo práctico, dentro de la API REST perteneciente al proyecto de memoria, para obtener una lista de usuarios es utilizada la ruta “https://url/api/usuarios”.
* **POST:** El presente método se emplea con el fin de insertar recursos en el servidor. La información es enviada en el cuerpo de la solicitud hacia el servidor junto con la ruta correspondiente. A modo de ejemplo práctico, para llevar a cabo una solicitud POST dentro de la API del proyecto se debe utilizar la ruta “https://url/api/usuario”.
* **PUT:** Este método permite actualizar un sinnúmero de campos alojados en el servidor. Homologo al método POST, la información es enviada en el cuerpo de la solicitud. Como ejemplo, se muestra la ruta utilizada para actualizar usuarios dentro de la API del proyecto: “https://url/api/updateusuario/id”.
* **DELETE:** El método es utilizado con el fin de eliminar recursos almacenados en el servidor mediante la ruta en cuestión. Para comprender mejor la estructura de cómo se utiliza la ruta del método, se ejemplifica con la ruta perteneciente al servidor del proyecto que elimina usuarios: “https://url/api/eliminarusuario/id”.
  + 1. **Respuestas HTTP**

Los códigos de estado de respuesta HTTP indican cuando se ha completado satisfactoriamente una solicitud específica [MDN2021a]. Dicho de otro modo, en el momento de efectuar una solicitud es de gran valor recibir cierta información de parte del servidor, con el propósito de poseer conocimiento respecto a la resolución de la operación, la cual puede indicar el éxito o fallo de dicha solicitud. Por lo cual, reunidos en cinco categorías, surgen variados códigos de estado de respuesta HTTP, los cuales son descritos en el siguiente listado:

* Respuestas informativas (100-199).
* Respuestas satisfactorias (200-299).
* Redirecciones (300-399).
* Errores de los clientes (400-499).
* Errores de los servidores (500-599).

Es de suma importancia que se tenga en conocimiento la información que representa cada código de estado y su significado en dicho ámbito. Para contextualizar, se describen algunos códigos entregados por diversos sistemas:

* **200**: Indica que la solicitud ha sido exitosa. Sin embargo, su significado puede variar dependiendo del método usado en la petición.
* **201**: Informa que el recurso ha sido creado satisfactoriamente.
* **403**: Indica que no se cuenta con los permisos para acceder al recurso.
* **409**: Indica que ha ocurrido un conflicto en la petición, esto está relacionado con diversos factores.
* **500**: Señala que ha ocurrido un error en el servidor.
  1. **NodeJs**

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, NodeJs está diseñado para crear aplicaciones network escalables [NOD2016]. Aplicaciones tales como sitios Web, API, entre otros. Este entorno se basa en el modelo de eventos, además de poseer la adaptabilidad necesaria para poder integrar diferentes bibliotecas y *frameworks*.

Para el manejo de paquetes, NodeJs utiliza el gestor de paquetes NPM, desde donde diferentes desarrolladores de proyectos de código abierto suben sus contribuciones [NPM2019].

En contexto de la plataforma Comparte Conocimientos UCM, se utilizó esta herramienta junto con el *framework* ExpressJS, para así generar una API REST que mantiene comunicación directa con la base de datos mediante el ORM Sequelize.

* 1. **ORM**

Según el artículo escrito por Deloitte, un ORM (*Object Relational Mapping*) es un modelo de programación que permite mapear las estructuras contenidas dentro de una base de datos relacional [MUR2018], todo esto en el contexto de una sólida base lógica de entidades de objetos que ayudan a acelerar y sobre todo a simplificar el desarrollo de diferentes tipos de aplicaciones.

Las estructuras almacenadas en la base de datos son vinculadas con las entidades lógicas generadas en base a la utilización del ORM, facilitando a que el desarrollador evite escribir directamente código SQL para crear los diferentes CRUD (*Create, Read, Update, Delete*) a utilizar, generando un gran impacto en tiempo empleado en el desarrollo de una aplicación.

Para la plataforma Comparte Conocimientos UCM, se implementó como gestor de base de datos relacional PostgreSQL. Ahora bien, para generar una comunicación consistente y eficiente, se implementó el ORM Sequelize.

* 1. **Sequelize**

El ORM basado en promesas Sequelize es una herramienta utilizada para la comunicación con diferentes bases de datos relacionales del mercado como: PostgreSQL, MySQL, MariaDB, SQL Server, entre otros. Entre sus principales características está poseer un sólido soporte de transacciones, relaciones, carga diferida, entre otros. [DEP2014].

Para ejemplificar de mejor manera la integración de Sequelize en la arquitectura del Backend, se presenta la figura 2.4, la cual representa gráficamente la comunicación entre los sistemas:

Logotipo, nombre de la empresa

Descripción generada automáticamente

**Figura 2.4:** *Esquema de comunicación entre los sistemas pertenecientes al Backend*. **Fuente:** *Elaboración propia.*

* 1. **Socket.IO**

Socket.IO es una librería que, en palabras simples, permite la comunicación bidireccional en tiempo real entre el cliente y el servidor. Su funcionamiento se sustenta en establecer una conexión mediante *WebSocket*, el cual consiste en un protocolo de comunicación que entrega un canal de baja latencia entre el servidor y cliente [SOC2022].

Esta poderosa herramienta provee todo lo necesario para la creación de una aplicación de chat, y debido a su compatibilidad con JavaScript y TypeScript, lo hace perfecto para el desarrollo del apartado comunicacional del proyecto Comparte Conocimientos UCM.

* 1. **PostgreSQL**

Para la elección acertada de una base de datos, se llevó a cabo una investigación en busca de las ventajas del uso de bases de datos relacionales y no relacionales. Para esto se buscó información referente a PostgreSQL y MongoDB.

PostgreSQL es un sistema de base de datos relacional de alta disponibilidad. Es capaz de funcionar de manera estable en el servidor y, por lo tanto, resulta robusto, siendo esta una de las principales características que buscan las empresas. Además, es consistente, tolerante a fallos y es compatible con el modelo relacional, asegurando siempre su integridad referencial [BLA2018].

MongoDB es una base de datos orientada a documentos lo cual quiere decir que, en lugar de guardar los datos en registros, guarda los datos en documentos y estos son almacenados en BSON, que es una representación binaria de JSON [ROB2019].

En consecuencia, las bases de datos no relacionales poseen una flexibilidad al estar enfocadas en el manejo de documentos, lo cual permite adaptarse a los diversos cambios del sistema. La solidez que entregan las bases de datos relacionales, considerando la integridad y baja redundancia de los datos, además de un mejor rendimiento, llevó a la elección de PostgreSQL como la base de datos a utilizar.

Se muestra en la figura 2.5 un gráfico comparativo del rendimiento entre MongoDB y PostgreSQL, perteneciente al documento: *Comparación de rendimiento entre bases de datos Relacionales, NoSQL y Blockchain*:

Gráfico

Descripción generada automáticamente **Figura 2.5:** *Gráfica con los tiempos de ejecución de PostgreSQL frente a MongoDB*. **Fuente:** *[PER2020].*

PostgreSQL, al ser una base de datos relacional orientada a objetos de código abierto, entrega toda la seguridad, consistencia y robustez que el proyecto demanda. Entre sus principales características se encuentran: tipos de datos más utilizados, integridad de datos, concurrencia y buen rendimiento, seguridad, extensibilidad, entre otros [POS2022].

* 1. **Postman**

Postman es una aplicación que funciona de un modo simple: como un cliente que envía peticiones HTTP hacia una API. Esta permite la creación de espacios de trabajo, colecciones de peticiones, entre otras herramientas [AST2021]. Por otro lado, permite conocer en mayor profundidad la API con la que se está interactuando, lo cual es idóneo en el desarrollo de esta, ya que evita la implementación incompleta.

Esta herramienta toma gran relevancia al momento de desarrollar la API REST perteneciente al proyecto de memoria, entregando incluso un monitoreo externo de algunos datos almacenados en el servidor.

* 1. **MVC**

El Modelo Vista Controlador corresponde a una arquitectura de software que separara los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos: Modelos, Vistas y Controladores [ALI2012]. A continuación, se definen dichos componentes:

* **Modelo:** Este componente es el que contiene una representación lógica de toda la estructura de datos que maneja el sistema, como lo es, por ejemplo, la base de datos. Además, maneja su lógica de negocios y su mecanismo de persistencia. El modelo es responsable de:
  + Acceder a la capa de almacenamiento de datos. Es pertinente que el almacenamiento sea independiente de los otros módulos del sistema.
  + Definir las reglas de negocio (funcionalidad del sistema).
  + Llevar el registro de las vistas y controladores del sistema.
  + El desarrollador, al estar frente a un modelo activo, debe notificar con las vistas los cambios que se puedan producir en los datos a causa de agentes externos.
* **Vista:** Este componente se encarga de mostrar al usuario final las pantallas, ventanas, páginas, formularios, y los resultados de una solicitud. Las vistas son responsables de:
  + Recibir datos del modelo y mostrarlo al usuario.
  + Tener un registro de su controlador asociado.
* **Controlador:** Este componente actúa como intermediario entre los modelos y las vistas, gestionando activamente el flujo de información entre estos dos. También se dedica a transformar los datos dependiendo de las necesidades de cada componente. El controlador es responsable de:
  + Recibir los eventos de entrada y retornar las respuestas.
  + Contener las reglas de gestión de eventos, del tipo “Si evento Z, entonces acción W”. Estas acciones pueden suponer peticiones al modelo o a las vistas.
  1. **VPS**

Un Servidor Virtual Privado es básicamente una partición virtual dentro de un servidor físico al cual se le asignan recursos exclusivos dependiendo de las necesidades de este. Su funcionamiento consiste en la creación de una capa virtual sobre el sistema operativo del servidor físico base, la cual se divide en particiones entregando así la posibilidad de instalar el sistema operativo que el cliente estima conveniente, además de todo el software necesario para su utilidad como servidor. Cabe mencionar que, al ser privado, el usuario posee control absoluto del servidor [ALB2021]. Para almacenar todas las tecnologías utilizadas en el desarrollo del Backend, se contrató un VPS del sitio web Contabo.

* 1. **AWS**

Amazon Web Services es una colección de servicios integrales ofrecidos por la empresa Amazon, que brinda diferentes soluciones de computación, almacenamiento, aplicaciones móviles, gestión de datos, entre otros. En la actualidad, es de los servicios más demandados por las empresas que ofertan posibilidades laborales.

Dentro del proyecto, se utilizó el servicio de *Amazon Simple Storage Service* o también conocido como Amazon S3. Este consiste, básicamente, en un servicio de almacenamiento de objetos que ofrece escalabilidad a nivel industrial, accesibilidad de datos, seguridad y sobre todo un excelente rendimiento [AWS2022].

* 1. **Git**

Git es, hoy en día, el sistema de control de versiones más utilizado en el mundo. Siendo un proyecto de código abierto desarrollado por el creador del Kernel de Linux, Linus Torvalds. Git Corresponde a una herramienta madura que apoya continuamente el proceso de desarrollo de software gracias a la posibilidad de utilizar ramas con versiones anteriores del sistema [CHA2008]. Esto tiene como ventaja la posibilidad de que, si se lleva a cabo un cambio no deseado o desafortunado, es posible encontrar en las versiones anteriores un respaldo completo del proyecto.

Git presenta una arquitectura distribuida, siendo un ejemplo de DVCS (Sistema de Control de Versiones Distribuido, por sus siglas en inglés). Dentro de sus características, en lugar de tener un único espacio para todo el historial de versiones del software, como sucede de manera habitual en los sistemas de control de versiones antaño populares, como CVS o subversión (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios [BIT2019]. Para el desarrollo de la aplicación Comparte Conocimientos UCM, se utilizaron tres repositorios en GitHub, donde se almacena el proyecto y todas sus versiones.

* 1. **Trello**

Trello es una herramienta flexible utilizada para la gestión de trabajo, con la que diversos equipos pueden diseñar, colaborar y organizar flujos de trabajo y hacer seguimiento activo del estado de este [ATL2017]. Los principales atributos de Trello son su facilidad de uso y versatilidad, atributos que le permiten al desarrollador una plena organización, desde una lista de compras, hasta grandes proyectos [PAR2016].

Esta APP basa su metodología en el sistema de trabajo japonés llamado Kanban, el cual incorpora tableros y tarjetas en un espacio de trabajo físico, para coordinar diferentes actividades [PAR2016a]. Gracias a las posibilidades de trabajo que permite la herramienta, su facilidad y naturaleza para el trabajo en equipo, es que se transforma en un punto ideal para el trabajo con metodologías ágiles, como lo son Scrum.

* 1. **Netlify**

El servicio proporcionado por Netlify consiste en una plataforma todo-en-uno para automatizar proyectos Webs modernos. Como característica principal, y más destacable, se encuentra la posibilidad de desplegar un proyecto de lado del cliente de manera gratuita, permitiendo un tráfico de hasta 100 GB [SRI2020].

* 1. **Marco de Trabajo Scrum**

Scrum es un marco ágil de trabajo en el que se aplican de manera regular un conjunto de buenas prácticas que permiten de manera colaborativa, autoorganizada, y en continuo aprendizaje, la obtención de resultados óptimos dentro de un proyecto [LGS2017]. Este marco de trabajo es muy utilizado en el desarrollo de software, aunque cabe mencionar que estos procesos pueden utilizarse en cualquier tipo de trabajo en equipo.

Entre las características más beneficiosas que presenta este marco de trabajo se encuentra la entrega incremental de producto en cortos periodos de tiempo. Esto entrega un valor agregado, ya que en cada incremento se mejora el aprendizaje tanto del equipo de trabajo como del cliente con respecto al sistema. Por otro lado, Scrum se basa en un proceso de control empírico, lo cual señala que las cosas pueden cambiar en cualquier momento, y la mejor forma de responder al cambio es mediante la transparencia, la inspección y adaptación. Este marco de trabajo no busca hacer más cosas en un menor tiempo, sino que, busca entregar un valor al proyecto en el cual es aplicado [HUA2020].

La figura 2.6 muestra de forma gráfica el proceso del marco de trabajo Scrum, lo que entrega una visión general de su funcionamiento:

Diagrama

Descripción generada automáticamente con confianza baja

**Figura 2.6**: *Proceso del marco de trabajo Scrum.* **Fuente:** *Elaboración propia.*

En base al artículo escrito por Claire Drumond para el portal Atlassian se expone lo siguiente: Aunque Scrum está estructurado, no es del todo rígido. Su ejecución se puede adaptar a las necesidades de cualquier organización, de lo cual existen muchas teorías acerca de cómo deben trabajar los equipos de Scrum exactamente para tener éxito. De Atlassian, fue aprendido que la comunicación clara, la transparencia y la dedicación a la mejora continua siempre deben ser el núcleo del marco de trabajo [DRU2022]. Tras el estudio del documento de Claire, se destacan y explican los siguientes artefactos y protocolos Scrum, los cuales poseen relevancia en la confección de la aplicación Comparte Conocimientos UCM:

* **Backlog del Producto:** Básicamente, consiste en un conjunto de tareas que el propietario del producto asigna, y funcionan como la entrada para la confección del *sprint backlog.* Esta agrupación de tareas no es inmutable, por lo cual reacciona según el estado y prioridades del proyecto.
* **Backlog del Sprint:** Consiste en un listado de elementos, historias de usuario, corrección de errores o subtareas seleccionadas por el equipo de trabajo, con la finalidad de ser implementadas en el ciclo actual del sprint. Este backlog es dinámico, ya que evoluciona en base al desarrollo del sprint. Un punto importante para tener en consideración a la hora de confeccionar el backlog del sprint es que las tareas no pueden poner en peligro el objetivo principal del sprint mismo.
* **Sprint:** Es el periodo de tiempo en el cual se desarrollan las diversas tareas que representan un incremento. Dentro de este espacio, es posible renegociar el alcance, entre otros factores con respecto al incremento. Un sprint por lo general dura dos semanas, aunque el tiempo depende naturalmente del equipo de trabajo.
* **Incremento:** Este artefacto consiste en el producto terminado en contexto de la finalización de un sprint. En esta etapa el equipo expone el producto finalizado del sprint, generando a su vez una retroalimentación, además de un valor al proyecto.
* **Retrospectiva del Sprint:** Llegado este evento, el equipo realiza una retroalimentación de análisis de todos los acontecimientos que han funcionado y los que no al momento de desarrollar el sprint. También se examina al equipo, las herramientas, el proyecto, etc. Este es el punto donde se documenta lo realizado en el periodo de desarrollo.
  1. **Justificación de Tecnologías**

El objetivo de este proyecto de memoria es crear una aplicación multiplataforma, intuitivo, atractivo y de uso simple. Es debido a esto que se adoptaron las diversas tecnologías expuestas dentro del marco teórico. A modo de resumen, se dispone a continuación de la distribución de las tecnologías empleadas en el desarrollo de la plataforma Comparte Conocimientos UCM:

* **Diseño UX/UI (Interfaz y experiencia de usuario)**: Para la confección de este apartado se utilizó el software de código abierto Figma.
* **Backend**: El sistema de lado del servidor consiste en un conjunto de tecnologías bien organizadas compuesto por: NodeJs junto al *framework* ExpressJS y la librería Socket.IO. Además de integraciones con Amazon Web Services y su servicio de alojamiento de datos S3. Como base de datos se utilizó PostgreSQL, el cual se comunica directamente con NodeJs mediante el ORM Sequelize.
* **Frontend**: Para el desarrollo visual, o, en otras palabras, la visión del cliente, se utilizaron dos tecnologías: Angular, como *framework* de JavaScript/TypeScript, y por otro lado Ionic, como *framework* de diseño.

Para el alojamiento del servidor se utilizó el VPS Contabo, y para alojar el cliente se utilizó la herramienta Netlify.

1. **:** **Propuesta de Solución**

El presente capítulo explica la propuesta de solución al problema expuesto en el primer Capítulo.

* 1. **Análisis**

La creación de software es una tarea indispensable que requiere una estructura de trabajo definida y un flujo comunicacional activo, así como una gran pasión por entregar soluciones creativas a los diversos problemas que existen hoy en día en las diferentes áreas que comprenden la vida. La creación de Comparte Conocimientos UCM está basada en el marco de trabajo ágil Scrum, lo cual implica que tanto su planificación, diseño de interfaces, experiencia de usuario, así como el desarrollo del software en cuestión, fueron regulados bajo este marco de trabajo, aplicando todas sus buenas prácticas y procedimientos.

2. 4. **Funcionalidad Organizacional de la Aplicación**

El desarrollo de la aplicación contempló diferentes subsistemas requeridos para su desempeño óptimo. La organización y distribución por módulos de la aplicación Comparte Conocimientos UCM se expone en la Figura 3.2, para así, de este modo, mejorar la comprensión de la plataforma:

Diagrama

Descripción generada automáticamente

**Figura 3.1.** *Organización y distribución modular de Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

La aplicación, al ser diseñada de tal manera que sea responsiva a los diferentes dispositivos, se ejecuta tanto en sistemas de escritorio, como móviles. Su funcionamiento, explicado de manera simple, consiste en una vista principal, en la cual se presenta el control de acceso o *login*, además de la posibilidad de registrar una cuenta. Al iniciar sesión, se hace una petición al servidor donde este responde con el acceso a la aplicación. Luego, desde la vista de inicio, se dispone de una barra de navegación para el acceso a las diferentes herramientas que brinda el sistema, tales como: avisos, preguntas, subida de archivos, cursos, búsqueda, favoritos. Por otra parte, existe una barra superior, donde se encuentra el perfil de usuario, desde donde es posible acceder a los contactos y a la funcionalidad de chat.

En base a la Figura 3.2. se extraen diferentes conceptos que son de gran relevancia a la hora de comprender las funcionalidades y propuesta de solución que entrega la plataforma Comparte Conocimientos UCM. Estos se explicados a continuación:

* **Subsistema de Control de Plataforma de Usuarios:** Este permite identificar la plataforma en que el usuario está ingresando al sistema y, por consiguiente, evidencia el formato óptimo para lograr un mejor desempeño de la aplicación y una visualización responsiva según el dispositivo de acceso.
* **Subsistema de Control de Acceso:** Permite tener el control y/o registro de los usuarios que intentan acceder a la plataforma, entregando un medio de seguridad a los mismos garantizando un uso seguro de la aplicación.
* **Subsistema de Control de Contactos:** Módulo encargado de la gestión de los contactos junto con los respectivos datos. Permite tener conocimiento del universo de usuarios disponibles con los cuales es posible entablar comunicación.
* **Subsistema de Perfil de Usuario:** Herramienta encargada de la gestión de datos por parte del usuario, el cual entrega la posibilidad de gestionar datos personales, como lo son nombre, apellido, carrera, contraseña, además de agregar una foto de perfil.
* **Subsistema de Gestión de Roles:** Consiste en un subsistema de dos roles: Administrador y usuario final. El administrador posee acceso al manejo de recursos dentro de la aplicación, como lo son gestión de cursos y secciones, además de moderador con respecto a los archivos.
* **Subsistema de Interconexión:** Corresponde a la conexión directa con el servidor, la cual es llevada a cabo mediante una API REST propia. Esta permite ejecutar los diferentes servicios utilizados en la plataforma.
* **Subsistema de Chat:** Módulo encargado del funcionamiento de chat, permitiendo la interacción entre usuarios asegurando una comunicación privada entre estos.
* **Subsistema de Secciones:** Módulo que da lugar a la gestión de las secciones dentro de la aplicación, lo cual facilita la experiencia de usuario al momento de utilizar la plataforma.
* **Subsistema de Publicaciones:** Se encarga de la creación de publicaciones del tipo: aviso, pregunta o archivo. Cada uno de estos tipos de publicaciones se almacenan en sus respectivas vistas dentro del sistema, las cuales son de acceso intuitivo para el usuario.
* **Subsistema de Comentarios en Publicaciones:** Subsistema que se encarga de generar comentarios dentro de las publicaciones, sean estas del tipo de avisos o preguntas, lo cual aporta en crear el sentido de comunidad.
* **Subsistema de Búsqueda:** Motor de búsqueda que permite la rápida localización de archivos en la aplicación, proporcionando un ahorro considerable de tiempo y procesamiento, ayudando en lograr una navegación más agradable y óptima.
* **Subsistema de Descarga de Archivos:** Herramienta que permite a los usuarios poder descargar los diferentes archivos alojados en el sistema.
  1. **Modelo Relacional de la Base de Datos**

Para el desarrollo de la plataforma Comparte Conocimientos UCM, como parte de su modelo de solución, se implementó una base de datos relacional bajo la tecnología de PostgreSQL.

La figura 3.2 muestra una visión global de las entidades y relaciones participantes en la base de datos:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**Figura 3.2.** *Base de datos relacional de Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

Es oportuno mencionar que el desarrollo de la base de datos, sus puntos clave e interacción con el sistema serán explicados en profundidad en el siguiente capítulo.

1. **: Sistema Comparte Conocimientos UCM**

A lo largo de este capítulo se presenta el desarrollo de la aplicación Comparte Conocimientos UCM, desde el diseño UX/UI hasta su puesta en marcha.

1. 1. **Backlog del Producto**

Antes del inicio del desarrollo del proyecto, se llevaron a cabo diversas reuniones para poder generar toda la planificación, enfoque y alcance respecto al producto de software. Como resultado a dichos encuentros, se hizo toma de requerimientos, dando origen a los requisitos funcionales y no funcionales (expuestos en el capítulo dos), los cuales, por consecuencia, ayudaron en la confección del backlog del producto. -

Para mantener un control de todo lo realizado, se hizo uso de la herramienta Trello, donde fueron definidos dos backlogs de producto, considerando la naturaleza cliente – servidor del sistema, un backlog para el Backend y otro para el Frontend. La figura 4.1 presenta una captura de la herramienta Trello y la disposición de algunas tareas desarrolladas en el proceso de creación del software:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

**Figura 4.1** *Backlog del producto para Backend y Frontend con la herramienta Trello.* **Fuente:** *Elaboración propia.*

Entre los puntos más relevantes del desarrollo de Comparte Conocimientos UCM a destacar se encuentran los siguientes:

* **Definición del Modelo y Arquitectura de Datos:** Confección organizada y estructurada de los modelos de datos que interactúan dentro del sistema. Este artefacto compone el corazón principal de la API REST.
* **Creación de Vistas del Sistema:** Diseño e implementación de las vistas responsables de interactuar con el usuario, a modo que estas sean simples, responsivas y atractivas.
* **Interconexiones Entre el Cliente y el Servidor:** Diseño de rutas desde el servidor para la conexión directa con el cliente mediante el protocolo HTTPS, el cual brinda una conexión segura.
* **Implementación con AWS:** Conexión desde el servidor con S3 de Amazon Web Services, ayudando al manejo de archivos directamente desde la nube.
  1. **Sprint Número Uno**

En el desarrollo de este primer sprint se fijó como objetivo el diseño de usabilidad e interfaces de usuarios (UX/UI), mediante la herramienta Figma, para de esta forma tener un esquema base de cómo serán implementadas las vistas. Este sprint posee una duración de dos semanas, en las cuales fueron distribuidas tareas específicas en un Backlog del sprint.

* + 1. **Backlog del Sprint**

En la siguiente sección se describen las tareas asignadas para el desarrollo del sprint número uno.

* **Diseñar el Mapa de la Aplicación:** Diseño general que expone a grandes rasgos las diferentes vistas que interactúan entre sí.
* **Toma de Ideas Generales:** Colección de ideas provenientes desde diversos portales relacionados a la temática del proyecto, además de conceptos ideados propiamente por el equipo.
* **Creación de un Boceto o *Wireframe*:** Confección de los primeros bocetos en papel de cómo sería visualmente el proyecto.
* **Traspaso del Boceto a Figma:** Digitalización del boceto de diseño y mapa de la aplicación en la herramienta Figma, generando de esta forma el Mockup que entrega forma al proyecto.
  + 1. **Incremento**

Como consecuencia al sprint número uno, el cual como objetivo principal consiste en el diseño UX/UI, se obtuvieron los siguientes resultados expuestos en subsecciones.

* + - 1. **Mapa de la Aplicación**

Se toma como punto base de la navegación entre vistas la página “*home*”, ya que, al ser un punto céntrico, da entrada a la creación de una barra de navegación que permite acceder a todos los demás puntos de la plataforma. En la figura 4.2 se expone el resultado obtenido del proceso de creación del mapa de la aplicación:

Diagrama

Descripción generada automáticamente

**Figura 4.2** *Mapa de la aplicación Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

* + - 1. **Diseño UX/UI**

Una de las ideas principales en el desarrollo de Comparte Conocimientos UCM es la conectividad sin importar donde se encuentren los usuarios, y sin importar que dispositivo estos usen. En base a esta premisa es que el diseño se genera lo más simple posible, fomentando que la experiencia de usuario no sea tediosa, entregando una experiencia adaptativa para enfrentarse a cualquier ambiente. La figura 4.3 muestra el resultado de algunos diseños preliminares, los cuales fueron base para el resultado final del producto de software:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Figura 4.3** *Segmento de diseño preliminar aplicación Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

* + 1. **Retrospectiva del Sprint**

Entre los puntos importantes obtenidos como experiencia dentro del sprint número uno, el cual se debe aplicar en futuros sprint, es la aplicación de una paleta atractiva de colores, además de una redistribución para dispositivos más pequeños que sea semejante al diseño original.

* 1. **Sprint Número Dos**

En el desarrollo de este segundo sprint, fue asignado como objetivo principal el diseño e implementación de la base de datos en PostgreSQL. Para esto, el punto principal fue desglosado en pequeñas subtareas descritas en el backlog del sprint. La duración de este ciclo corresponde a dos semanas.

* + 1. **Backlog del Sprint**

La siguiente sección describe de forma explicativa las tareas designadas para el sprint número dos:

* **Diseño del modelo relacional:** En base a la idea principal del proyecto, se diseña un modelo relacional con todos los elementos participantes y sus respectivas relaciones.
* **Normalización:** Procedimiento en el cual es tomado el diseño posteriormente creado y se aplica el proceso de normalización para evitar redundancias, dependencias innecesarias, crear flexibilidad, entre otros.
* **Implementación mediante SQL:** Se toma el resultado de la normalización y se implementa en el gestor de bases de datos PostgreSQL mediante la programación de un script SQL.
* **Testeo:** Se prueba la base de datos mediante la creación, modificación, actualización y eliminación de datos y tablas.
  + 1. **Incremento**

Como consecuencia del desarrollo del sprint número dos, el cual finalmente se centra en el desarrollo e implementación del modelo de datos en el gestor de base de datos PostgreSQL, se obtienen los siguientes resultados, ordenados en subsecciones.

* + - 1. **Modelo Relacional**

Para comenzar a implementar de manera efectiva la base de datos responsable de almacenar toda la información importante para la aplicación, se requiere planificación, un modelo gráfico en donde apoyarse, el cual responde a los requerimientos funcionales y a la problemática de la cual fue gestado el proyecto.

Para llevar a cabo este proceso, se identifican todos los elementos de relevancia que interactúan entre sí, como lo son: el usuario, secciones, avisos, documentos, entre otros. Estos generan relaciones sólidas mediante sus atributos, más específicamente, con sus claves primarias y claves foráneas. Llegado a este punto, se posee un modelo relacional, aunque precisa optimización mediante la normalización. La figura 4.4 muestra el modelo relacional final, luego de haber sido normalizado hasta su tercera forma normal:

Diagrama

Descripción generada automáticamente

**Figura 4.4** *Modelo Relacional hasta su tercera forma normal.* **Fuente:** *Elaboración propia.*

* + - 1. **Código SQL**

Adoptando como base el modelo relacional anteriormente construido, se prosigue con la adaptación de este último a código SQL. Para llevar a cabo esta tarea se utiliza el gestor de base de datos relacionales PostgreSQL, además de un software complementario: DBeaver, el cual consiste en un cliente de SQL y administrador de bases de datos.

Antes de comenzar a escribir código, se precisa crear la base de datos que alojará dicho contenido. Para facilitar este proceso, se utilizó la interfaz gráfica de PostgreSQL, llamada pgAdmin 4. La figura 4.5 muestra la interfaz que crea la base de datos, a la cual fue asignada el nombre de “app”:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Figura 4.5** *Creación de base de datos mediante pgAdmin 4.* **Fuente:** *Elaboración propia.*

Ya con la base de datos creada, se utiliza el software complementario DBeaver para la escritura del código y su ejecución. La figura 4.6 expone un fragmento del código SQL utilizado para la estructura de la base de datos.

Se utilizaron atributos como “createdAt” y “updatedAt”, por requerimientos del ORM Sequelize, explicado en secciones posteriores:

Texto

Descripción generada automáticamente

**Figura 4.6** *Fragmento de código SQL correspondiente a la base de datos del proyecto.* **Fuente:** *Elaboración propia.*

* + - 1. **Base de Datos Implementada**

Luego del proceso de ejecución del código expuesto en la subsección anterior, se pasó al proceso de pruebas con la base de datos, haciendo ingreso de diferentes volúmenes de datos, actualización de estos y eliminación en cascada. Además, se hicieron pruebas con respecto a las tablas, como son la actualización de atributos, remoción de tablas, diferentes tipos de consultas, entre otras. Todo esto con el fin de garantizar la robustez del modelo de datos.

La base de datos final implementada es expuesta en el capítulo tres, más explícitamente en la figura 3.3, donde es posible apreciar el modelo completo de datos.

* + 1. **Retrospectiva del Sprint**

Como resultado del sprint número dos, se toma como experiencia a aplicar en futuros sprint la incorporación de nuevos atributos que contribuyan con la funcionalidad óptima de la aplicación final.

* 1. **Sprint Número Tres**

El fin de este tercer sprint consiste en el desarrollo de las vistas responsables de la interacción con el usuario, utilizando como base el diseño UX/UI anteriormente creado, siendo estas adaptables a dispositivos móviles. La duración del sprint es de cuatro semanas. Sus tareas específicas son listadas en el backlog del sprint.

* + 1. **Backlog del Sprint**

La siguiente sección describe de forma explicativa las tareas designadas para el sprint número tres.

* **Creación de Vistas de Inicio de Sesión:** Confección de todas las vistas respectivas al control de acceso de usuarios, como, por ejemplo: registro.
* **Creación de la Navegación:** Confección de la barra de navegación y la barra superior (*header*) de la aplicación.
* **Creación de la Vista Home:** Confección de la vista principal del sistema, la cual funciona como puerta de enlace a otras herramientas de la aplicación, complementándose con la barra de navegación.
* **Creación de Vistas de Perfil:** Implementación según el diseño de las vistas referentes al perfil de usuario, donde es posible obtener información de este mismo, como editar sus datos y observar los contactos disponibles.
* **Creación de Vistas de Avisos y Preguntas:** Desarrollo de las vistas referente a la creación de avisos y preguntas, además de las vistas para comentar estas últimas.
* **Creación de Vista Compartir:** Desarrollo de la vista encargada de compartir archivos asignándolos a los diferentes cursos predeterminados.
* **Creación de Vista Chat:** Confección de la vista que compone el servicio de chat, disponible desde la vista de perfil de usuario.
* **Creación de Vistas Complementarias:** Desarrollo de las diversas vistas complementarias del sistema, las cuales generan un apoyo al funcionamiento completo del mismo, como lo son, por ejemplo: contacto, búsquedas, favoritos, entre otros.
  + 1. **Incremento**

A modo de resultado, luego del desarrollo exitoso del sprint número tres, el cual como objetivo principal se encarga del desarrollo de las interfaces visuales responsables de la interacción con el usuario final, se obtuvieron los siguientes productos ordenados en subsecciones.

* + - 1. **Vistas de Control de Acceso**

Ya que el sistema Comparte Conocimientos UCM funciona como un portal para diversos tipos de usuario, la mejor forma de comenzar a confeccionar las interfaces visuales que interactúan con el usuario final es mediante el control de accesos. Esto es debido a que es el punto de partida, tanto para crear una cuenta, como para ingresar al sistema y disfrutar de las herramientas que ofrece.

Basado en los diseños de usabilidad e interfaz de usuarios creados anteriormente, se confecciona tres vistas: ingreso o *login*, registro y recuperación de contraseña. Para llevar a cabo este proceso, se toma como principio que estas interfaces visuales sean adaptativas a los diferentes tipos de dispositivos.

El resultado de este proceso se puede apreciar en la figura 4.7 y 4.8, donde la primera muestra la interfaz correspondiente a un dispositivo móvil y la segunda expone la vista de un dispositivo de escritorio:

Interfaz de usuario gráfica

Descripción generada automáticamente

**Figura 4.7** *Vistas correspondientes al control de acceso del portal Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

**Figura 4.8** *Inicio de sesión en versión de escritorio al portal Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

Es necesario precisar que se tomó la decisión de utilizar una paleta de colores fría, ya que va acorde al logo de la Universidad Católica del Maule, además de generar a nivel visual un ambiente de tranquilidad y concentración, lo cual lo hace idóneo para su uso diario.

* + - 1. **Vista Principal y Sus Interacciones**

Anteriormente, en la sección 4.2, se comenta como es que la vista de inicio o home sería el punto de conexión entre las diferentes herramientas que brinda el sistema. Es por esto por lo que se diseña una atractiva e intuitiva barra de navegación inferior, la cual permite la interacción entre las diferentes interfaces visuales, con la ventaja de no perder las acciones que se realizan en las demás vistas. Su diseño está basado en las principales y más populares aplicaciones móviles del mercado, creando una sensación de familiaridad y una rápida curva de aprendizaje para los nuevos usuarios de la plataforma Comparte Conocimientos UCM. Entre los accesos que entrega la barra de navegación encontramos el mismo inicio/avisos, favoritos, búsquedas, compartir, y preguntas.

Junto con la adición de la barra inferior de navegación, se implementa una barra superior, la cual da acceso tanto a las funcionalidades de perfil de usuario, y la sección de contacto/soporte. Esta barra lleva la paleta de colores característica de la aplicación, lo cual mantiene el ambiente de tranquilidad que se desea generar en los usuarios, además de crear un contraste, haciendo llamativo y de fácil acceso con sus enlaces. En la figura 4.9 se observa la versión móvil de la implementación de las vistas de inicio, compartir archivos y preguntas.

En la figura 4.10 se contempla la versión de escritorio para la sección correspondiente a compartir archivos:

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

**Figura 4.9** *Barras de navegación junto a diferentes vistas pertenecientes al sistema Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Figura 4.10** *Versión de escritorio correspondiente a la interfaz visual de “Compartir Archivos”.* **Fuente:** *Elaboración propia.*

* + - 1. **Perfil de Usuario y Chat**

Cuando se crea un proyecto que tiene un fuerte flujo de usuarios, y todos se reúnen con el fin de interactuar entre ellos, sea por diferentes fines, como es, por ejemplo, el que da vida al presente proyecto: compartir conocimientos y generar comunidades, es necesario la implementación de un sistema de perfil de usuarios. Esta necesidad nace desde algo tan primitivo en nuestra existencia como es la necesidad de poseer una personalidad, una autenticidad y diferencia con los demás.

Otro punto crucial al momento de crear un proyecto social es la posibilidad de comunicarse entre los diferentes usuarios que navegan a través del portal. Para esto es que se implementaron sistemas de comunicación transparentes y globales, como lo son los avisos y preguntas, pero sobre todo un canal directo de mensajería instantánea: un chat. En la figura 4.11 es posible apreciar la interfaz correspondiente al perfil de usuario, así como la ventana de chat:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**Figura 4.11** *Vistas correspondientes al perfil de usuario y chat, pertenecientes a la aplicación Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

Es la vista de perfil de usuario la que da acceso al Chat mediante un listado de usuarios que se encuentra registrados en la aplicación. Por otro lado, los colores utilizados como burbujas de chat siguen la paleta utilizada para el sistema, con la diferencia de ser más claros. Esto con la finalidad de no generar cansancio visual en el usuario.

* + - 1. **Vistas Complementarias**

Luego de cubrir la gran parte de diseños de interfaces de usuario, donde las grandes funcionalidades son representadas, es importante asignar espacios a otras vistas, las cuales entregan un valor agregado a los puntos fuertes del sistema.

Entre las funcionalidades complementarias existentes para las cuales es necesario confeccionar vistas, es posible encontrar algunas como las búsquedas, contacto/soporte, favoritos, mis documentos, mis preguntas, mis avisos y cursos. La figura 4.12 muestra algunas de las interfaces visuales descritas anteriormente:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**Figura 4.12** *Vistas correspondientes a favoritos, búsquedas y contacto pertenecientes al sistema Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

* + 1. **Retrospectiva del Sprint**

Como conclusión al sprint número tres, se generan nuevas experiencias con el fin de poder aplicarse en futuras iteraciones, entre las que se encuentra la reutilización inteligente de estilos para agilizar la producción de diversas vistas. Por otra parte, la optimización de algunos enrutamientos puede ser mejorada para crear una fluidez en la navegación de la aplicación.

* 1. **Sprint Número Cuatro**

En el desarrollo de este cuarto sprint, se tomó como objetivo principal el desarrollo del servidor utilizando NodeJs y el *framework* ExpressJS. Además, la creación de las configuraciones para las conexiones con AWS y Socket.io. Por último, la creación de los componentes y servicios que generan un enlace a con el cliente. Para mejorar el desempeño del desarrollo, los objetivos son separados en subobjetivos enlistados en el backlog del sprint. La duración asignada a este ciclo de producción es de cinco semanas.

* + 1. **Backlog del Sprint**

La siguiente sección describe de forma explicativa las tareas designadas para el sprint número cuatro.

* **Implementación Base de la API:** Programación base de la API mediante NodeJs, donde se levanta un servidor local con el protocolo HTTP.
* **Integración de ExpressJS y Sequelize:** Implementación del *framework* ExpressJS y el ORM Sequelize para el manejo y comunicación de la base de datos PostgreSQL.
* **Implementación de Modelos:** Creación de los modelos lógicos, responsables de la comunicación con la base de datos.
* **Implementación de Controladores:** Creación de los diversos controladores, quienes aportan al sistema con funcionalidades como los son control de accesos, filtros, CRUD.
* **Implementación de Rutas:** Confección de las diversas rutas de salida, las cuales generan la puerta de enlace con la API REST.
* **Configuración de CORS y HTTPS:** Ajustes de seguridad mediante control de acceso HTTP (CORS) y el protocolo HTTPS mediante un certificado SSL.
* **Integración con AWS:** Configuración con los servicios S3 de Amazon Web Services, generando de esta manera una conexión directa con la API.
* **Integración con Socket.IO:** Configuración e implementación del paquete Socket.IO para establecer la comunicación bidireccional entre el cliente y el servidor.
* **Implementación de Servicios Angular:** Diseño y desarrollo de las diferentes peticiones HTTP desde el cliente para la comunicación con la API REST.
* **Implementación de Componentes Angular:** Diseño y desarrollo de los principales métodos responsables de consumir las peticiones realizada por los servicios de Angular.
  + 1. **Incremento**

Como resultado al desarrollo del sprint número cuatro, que como objetivo principal consiste en el desarrollo completo del servidor, además de incluir sus correspondientes conexiones con el cliente para el funcionamiento efectivo del proyecto, se obtuvieron los siguientes resultados, los cuales son ordenados en subsecciones.

* + - 1. **Creación de la API**

El proyecto de memoria Comparte Conocimientos UCM, está diseñado y ejecutado mediante el estilo de arquitectura MVC, por lo que, al poseer las vistas implementadas, es necesario integrar tanto los modelos como los controladores. Es aquí donde entra en juego la confección de la API REST.

Para llevar a cabo esta tarea, se comienza un proyecto mediante el comando “*npm init”*, el cual nos entrega la estructura básica para comenzar a trabajar con NodeJs y los módulos necesarios para el desarrollo de la API, obtenidos mediante el gestor de paquetes de NodeJs: npm.

Entre los módulos iniciales instalados para crear el servidor básico del proyecto se encuentran: *express, dotenv, body-parser, sequelize, pg* y *pg-htstore*. La funcionalidad de estos se explica brevemente en el listado a continuación:

* **Express:**Consiste en el *framework* de NodeJs utilizado para generar un servidor básico.
* **dotenv:** Módulo utilizado para configurar y utilizar variables de entorno.
* **Body-parser**: Permite tener acceso al cuerpo de las peticiones HTTP. En el caso del servidor de Comparte Conocimientos UCM, este funciona bajo el formato JSON.
* **Sequelize:** ORM encargado de la comunicación con la base de datos.
* **pg y pg-htstore:** Módulos complementarios a Sequelize para el trabajo con bases de datos PostgreSQL.

A continuación, se muestra en la figura 4.13 la estructura básica del servidor de NodeJs utilizada en el sistema del proyecto de memoria. Al ser una versión temprana, es utilizada la plataforma de control de versiones GitHub para entregar dicha información:

Texto

Descripción generada automáticamente

**Figura 4.13** *Estructura básica del Servidor*. **Fuente:** *Elaboración propia.*

* + - 1. **Conexión a la Base de Datos**

Al tener integrados los módulos necesarios para comenzar el trabajo con la base de datos, es necesario crear una lógica de programación que permita la comunicación entre el servidor de NodeJs y el modelo de datos PostgreSQL. Para esto se crean tres artefactos: modelos, controladores y rutas.

Sequelize facilita enormemente el trabajo de estructurar los directorios y archivos. Para esto se ejecuta el siguiente comando en el directorio del servidor de NodeJs: *sequelize init*. La figura 4.14 muestra de manera explícita la estructura de directorios que crea dicho comando:

Texto, Chat o mensaje de texto

Descripción generada automáticamente

**Figura 4.14** *Directorios creados por Sequelize.* **Fuente:** *Elaboración propia.*

Para efectos prácticos, se explicará en grandes rasgos como son implementado los tres diferentes artefactos encargados de la comunicación con la base de datos, así como la salida mediante rutas que permite a otras aplicaciones consumir los servicios de la API.

Los modelos consisten en una representación lógica de una tabla o entidad existente en la base de datos, además de las diferentes relaciones que estas poseen. A modo de ejemplo, la figura 4.15 muestra el modelo creado para la tabla de usuario:

Texto

Descripción generada automáticamente

**Figura 4.15** *Modelo lógico de la API REST.* **Fuente:** *Elaboración propia.*

Por otro lado, los controladores entregan la funcionalidad necesaria para llevar a cabo diferentes acciones, como lo son CRUD, control de accesos, proceso de seguridad, control de sesiones, entre otros.

La figura 4.16 muestra una función perteneciente al controlador que hace referencia al modelo de usuario:

Texto

Descripción generada automáticamente

**Figura 4.16** *Función del Controlador perteneciente al Modelo Usuarios.* **Fuente:** *Elaboración propia.*

El ultimo artefacto responsable de la comunicación son las rutas. Estas básicamente se encargan de entregar una dirección para acceder a las funciones pertenecientes a un controlador determinado. La figura 4.17 muestra las rutas pertenecientes al controlador del modelo de usuarios:

Texto

Descripción generada automáticamente

**Figura 4.17** *Rutas pertenecientes al Controlador de Usuarios.* **Fuente:** *Elaboración propia.*

* + - 1. **Seguridad**

Con el servidor NodeJs funcionando ya a modo de API REST gracias a las conexiones creadas con la base de datos, los controladores que aplican diversas funcionalidades, y las rutas que generan la puerta de conexión hacia el exterior, es importante tener en cuenta un nivel de seguridad para la comunicación con el servicio.

El primer nivel de seguridad a implementar son los CORS, o Intercambio de Recursos de Origen Cruzado, el cual permite decidir, en palabras resumidas, quien posee acceso a los recursos almacenados en la API REST. Para mejorar la comprensión de este nivel de seguridad y observar su implementación en código, la figura 4.18 muestra como está integrado en NodeJs:

Texto

Descripción generada automáticamente

**Figura 4.18** *Integración de CORS en la API REST de Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

La segunda capa de seguridad presente en el sistema es el protocolo HTTPS, o Protocolo Seguro de Transferencia de Hipertexto. Este, mediante un certificado SSL (Protocolo criptográfico de transporte) permiten establecer una conexión privada y segura con el cliente o cualquier servicio que intente consumir la API REST. La figura 4.19 muestra la implementación del protocolo HTTPS en el sistema:

Texto

Descripción generada automáticamente

**Figura 4.19** *Implementación del protocolo HTPPS en la API.* **Fuente:** *Elaboración propia.*

* + - 1. **Integraciones**

Entre las utilidades más importantes para la aplicación Comparte Conocimientos UCM, se encuentra la posibilidad de alojar archivos de diferente índole, además de la herramienta de mensajería instantánea. Para llevar a cabo estas funcionalidades, se integraron al servidor dos herramientas de gran utilidad: Socket.io y S3 de Amazon Web Services. La implementación de Socket.io es simple, ya que requiere de la instalación mediante el gestor de paquetes de NodeJs *npm*, además de las funciones de emisor – receptor tanto en el cliente como en el servidor. Para mejorar la comprensión de cómo se debe implementar esta comunicación bidireccional entre cliente y servidor, la figura 4.20 y 4.21 muestran su implementación en código respectivamente:

Texto

Descripción generada automáticamente

**Figura 4.20** *Conexión emisor – receptor NodeJs.* **Fuente:** *Elaboración propia.*

Texto

Descripción generada automáticamente

**Figura 4.21** *Conexión emisor – receptor Ionic Angular.* **Fuente:** *Elaboración propia.*

Por otro lado, una de las integraciones más importantes es la llevada a cabo con AWS, ya que, al ser una tecnología robusta, entrega un gran beneficio al proyecto de memoria, en especial su servicio de almacenamiento S3. Para crear la conexión entre el servicio de Amazon y la API REST se precisa de la instalación de los módulos *aws-sdk* y *express-fileupload*.

La integración de S3 con la API REST, en palabras resumidas, consiste en crear una cuenta de capa gratis en Amazon Web Services, para de este modo obtener los parámetros de acceso a los servicios, además de configurar los contenedores donde se almacenarán los archivos que serán enviados desde el cliente. La figura 4.22 muestra la integración en código de las tecnologías de AWS:

Texto

Descripción generada automáticamente

**Figura 4.22** *Integración de la API con AWS.* **Fuente:** *Elaboración propia.*

* + - 1. **Implementación en el Cliente**

Cuando los servicios pertenecientes a la API REST están en total funcionamiento, es el momento ideal para crear los puentes de lado del cliente que consuman las funcionalidades alojadas en esta. Para esto, Angular posee dos artefactos que permiten consumir de manera eficiente las funcionalidades de las API REST, estos son los componentes y los servicios.

Los servicios son clases encargadas de crear métodos a través de protocolos HTTP, los cuales consumen las diferentes funciones provenientes desde el servidor, para luego comunicarse con los componentes, quienes procesan los datos mediante métodos versátiles que permiten redirigir datos a diferentes secciones de la aplicación.

Una ventaja de los componentes es que integran un valor agregado al tener la posibilidad de adicionar nuevas funciones a través de métodos. Las figuras 4.23 y 4.24 muestran la estructura de un servicio y un componente respectivamente:

Texto

Descripción generada automáticamente

**Figura 4.23** *Servicio perteneciente a la vista Home de la aplicación Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

Texto

Descripción generada automáticamente

**Figura 4.24** *Componente perteneciente a la vista home de la aplicación CCUCM.* **Fuente:** *Elaboración propia.*

* + 1. **Retrospectiva del Sprint.**

Al finalizar el sprint número cuatro, se toma como experiencia diferentes factores, los cuales son buenos candidatos para la integración de estos en futuras iteraciones. Uno de los puntos fuertes es la integración y configuración de CORS, los cuales son de gran importancia para asegurar una conexión segura entre el cliente y el servidor. A su vez, el trabajo con AWS abre las puertas a futuros trabajos de integración con sus tecnologías, las cuales pueden optimizar de gran manera nuevas herramientas del proyecto.

* 1. **Sprint Número Cinco**

El objetivo por cumplir en este quinto sprint fue el despliegue de la aplicación a un entorno de producción, haciendo uso de un servidor virtual privado (VPS), además de la herramienta Web Netlify. El inicio de la iteración se comenzó con el proceso de test del proyecto. Para ello se asignó un tiempo de tres semanas para el término del sprint.

* + 1. **Backlog del Sprint**

La siguiente sección describe de forma explicativa las tareas designadas para el sprint número cinco:

* **Configuración del VPS:** Instalación del sistema operativo Ubuntu versión 20.04, y posterior configuración del entorno para alojar la API REST confeccionada con anterioridad.
* **Pruebas de conexión:** Proceso de pruebas hacia el servidor privado mediante la herramienta Postman.
* **Despliegue en Netlify:** Carga del proyecto construido al servicio de gestión Web Netlify, configurando así un dominio público.
* **Pruebas de la Aplicación:** Como equipo se llevan a cabo diversas pruebas al sistema ya desplegado en busca de errores, para de esta forma aplicar parches y futuros trabajos.
  + 1. **Incremento**

Como consecuencia del desarrollo exitoso del sprint número cinco, el cual lleva como objetivo principal el despliegue tanto del cliente como del servidor en los diferentes servicios de alojamiento, se obtuvieron los siguientes productos finalizados, los cuales están ordenados en diferentes subsecciones.

* + - 1. **Contabo VPS**

La primera labor antes de desplegar la sección del proyecto correspondiente al cliente es subir a producción toda la sección correspondiente a la API (servidor), ya que es gracias a la comunicación que se entabla con esta y la lógica de programación que hay detrás que es posible su funcionalidad y conexión con la base de datos.

Para lograr este cometido, es necesario la contratación de un servicio de VPS, la cual entrega, en palabras simples, acceso a una maquina física con una partición virtual privada donde es posible instalar un sistema operativo a conveniencia para desplegar todo tipo de servicios. En el caso correspondiente al proyecto Comparte Conocimientos UCM, se utiliza el servicio de Contabo, una galardonada empresa de hosting ubicada en Alemania especializada en alojamiento Web y servidores dedicados, la cual ofrece planes por bajos precios. En la figura 4.25 se expone la consola del VPS, con la distribución de Linux Ubuntu, en su versión 20.04.

Texto

Descripción generada automáticamente

**Figura 4.25** *Interfaz perteneciente al VPS de Contabo.* **Fuente:** *Elaboración propia.*

El proceso de despliegue de la API REST perteneciente al sistema Comparte Conocimientos UCM en el servicio previamente contratado consiste en las siguientes partes listadas a continuación:

* **Configuración del sistema Ubuntu:** Para la configuración del servidor Ubuntu, se aplicaron diversos comandos mediante la terminal, además de la instalación de una interfaz de escritorio para el manejo de algunos procesos de manera más amigable con el fin de agilizar los tiempos de desarrollo.
* **Instalación del Software Necesario:** Entre el software necesario para la implementación y despliegue del Backend o API REST del sistema, se encuentra PostgreSQL, NodeJs, NPM, y DBeaver.
* **Implementación de la Base de Datos:** Para la implementación de la base de datos se envía el Script correspondiente (script SQL) mediante SSH. Luego es ejecutado de igual forma a su implementación en desarrollo.
* **Clonar Repositorio GitHub de la API:** Para finalizar el proceso de despliegue del Backend perteneciente al proyecto de memoria, se clona el repositorio donde está alojado el código de la API. Luego se accede a su archivo de configuración, para asignar variables correspondientes a la conexión con la base de datos, para posteriormente ser ejecutada.
  + - 1. **Pruebas con Postman**

Luego de la implementación de la API REST en el servicio de VPS, al corroborar que todo está funcionando en orden, se procede a la realización de pruebas de comunicación mediante la herramienta Postman. En este proceso se llevaron a cabo diferentes tipos de test referente a los diferentes métodos HTTP que fueron implementados en el servicio de API.

A continuación, se presenta la figura 4.26 que muestra una de las pruebas realizadas, correspondiente a una petición del tipo GET, que trae como resultado un listado de usuarios:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Figura 4.26** *Test correspondiente a petición GET a la API REST del proyecto.* **Fuente:** *Elaboración propia.*

* + - 1. **Netlify y Pruebas Funcionales**

Al tener en funcionamiento la API REST alojada ya en un servidor, lista para ser consumida, comienza el proceso de despliegue del cliente. Para esto se utiliza el servicio de Netlify, el cual de manera rápida e intuitiva permite completar este proceso. La figura 4.27 muestra la interfaz de administración de Netlify:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Figura 4.27** *Interfaz perteneciente a la herramienta Netlify.* **Fuente:** *Elaboración propia.*

Luego de completar el proceso de despliegue en el servicio de Netlify, este permite al administrador de la cuenta crear un dominio público personalizado. En el caso del software perteneciente al proyecto se asignó “https://comparteconocimientosucm.netlify.app/”.

Al terminar con la configuración del cliente, y ya en correcto funcionamiento, se procede a generar pruebas funcionales, entre las que se encuentran pruebas respecto al control de acceso, tales como usuarios no existentes, claves erróneas, suplantación de datos, entre otros. La figura 4.28 muestra el resultado en sistema de dichas pruebas:

Interfaz de usuario gráfica

Descripción generada automáticamente

**Figura 4.28** *Pruebas de control de acceso a plataforma Comparte Conocimientos UCM.* **Fuente:** *Elaboración propia.*

El resultado de estas pruebas fue satisfactorio, debido a las funcionalidades implementadas para el control de usuarios en la API, solo se permite el ingreso a usuarios registrados, al igual que para el registro, los datos deben ser válidos y no pertenecientes al sistema previamente.

Por motivos prácticos, se describirá a grandes rasgos los diferentes tipos de pruebas realizados al sistema. Entre estos se crearon pruebas para la creación de avisos y preguntas, la posibilidad de creación de comentarios por parte de diferentes usuarios, entre otras. También se llevaron a cabo pruebas de comunicación respecto al chat, donde diferentes usuarios mantuvieron conversaciones activas y continuas en tiempo real. Por otro lado, se probó el estado de sesión activa, el cual se lleva a cabo gracias al funcionamiento de tokens llamados *guards*, los cuales son propios del sistema de Angular. Esto implicó que dos usuarios no pueden iniciar sesión con la misma cuenta al mismo tiempo. A continuación, se expone la figura 4.29, la cual muestra una prueba exitosa de subida de archivos a un curso predeterminado:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Figura 4.29** Prueba de subida de archivos a la plataforma de Comparte Conocimientos UCM. **Fuente:** *Elaboración propia.*

* + 1. **Retrospectiva del Sprint**

Como último sprint, se toma a modo de experiencia la configuración de un servidor Linux, lo cual es útil para trabajos futuros del proyecto, donde puede ser necesario la configuración de nuevos servicios en dicho sistema operativo. Además de la labor de aplicar pruebas funcionales a la aplicación, mostrando diversos puntos a fortalecer en otros desafíos.

* 1. **Información General Final**

Los lenguajes utilizados dentro del desarrollo de la aplicación Comparte Conocimientos UCM son:

**• Backend:** SQL y JavaScript.

**• Frontend:** JavaScript, TypeScript, HTML y CSS.

1. **: Conclusiones**

En el presente capitulo se lleva a cabo la exposición de las conclusiones del desarrollo de la memoria.



**5.1. Conclusión**

A continuación, se presentan las conclusiones respecto a las herramientas utilizadas, el marco de trabajo y generales:

* **Conclusión de las Herramientas:** La evolución en el desarrollo de la tecnología contribuye a los usuarios en el momento de simplificar y hacer autónomos los procesos que requieran de trabajo y mucho tiempo. Cuando se comienza a desarrollar software, los variados marcos de trabajo, o más conocidos como *frameworks*, son un recurso imprescindible al momento de proporcionar soluciones a problemáticas que se presentan en el día a día, mejorando la productividad a través del uso de estas tecnologías, ya que suponen ser una base estructural agregando utilidades y/o librerías que facilitan al programador centrarse de lleno a la solución de la problemática. Por otro lado, la utilización de recursos externos a la programación del software facilita la verificación de las funciones creadas, asegurándose que estas cumplan con el funcionamiento deseado.
* **Conclusión del Marco de Trabajo:**Usar el marco de trabajo ágil SCRUM dispone de enormes ventajas tales como: promover la autoevaluación en base a los incrementos, fomentar el trabajo en equipo, crecimiento mediante la autogestión de tareas, entre otros. No obstante, como en todo proyecto, son las personas las que logran que un trabajo sea exitoso.  
   Como el marco de trabajo sugiere, a medida que el proyecto avanzó, fue necesaria la autogestión del trabajo para adquirir nuevos conocimientos, para de este modo, seguir el desarrollo óptimo de los incrementales en cada sprint, por lo cual, este proceso entrega un valor como profesional al fomentar la obtención de conocimientos de manera autodidacta.
* **Conclusiones Generales:** Tras el desarrollo y análisis de la presente memoria, es posible deducir que se alcanzaron los objetivos planteados al inicio de la elaboración de este proyecto, concretando el desarrollo de la aplicación por medio de la utilización del marco de trabajo ágil Scrum. Además, se evidencio que el equipo, gracias a las sugerencias de Scrum, más la implementación de diversos procesos para adaptar el marco de trabajo a las necesidades personales, funcionó entregando soluciones eficientes y eficaces.   
   Dado las necesidades de los usuarios que pertenecen a la Universidad Católica del Maule, como lo son alumnos y docentes, se comprende que la solución proporcionada por la aplicación Comparte Conocimientos UCM es acertada, ya que, en los tiempos actuales y venideros, donde la situación sanitaria mundial es incierta, poseer un portal donde es posible organizarse como comunidad universitaria, compartir eventos, crear comunicación transparente entre los diferentes tipos de usuarios, además de compartir documentos de interés para un amplio tipo de público, tiene un valor incalculable. Esto debido a el aporte social y educativo que fomenta el presente proyecto de memoria, ayudando incluso a combatir diferentes enfermedades psicológicas, producto de los efectos del aislamientos social.  
   Finalmente, mediante la elaboración de una aplicación multiplataforma, con una interfaz sencilla y amigable para todo usuario, ejecutable en cualquier dispositivo, se cubrieron las necesidades manifestadas en este último tiempo de contingencia donde la labor de los profesionales de las comunicaciones e información es fundamental.
  1. **Trabajos Futuros**

El objetivo principal de Comparte Conocimientos UCM es ser una herramienta útil para todos quienes deseen crear comunidad y compartir información con su colectivo universitario. También busca satisfacer toda clase de incógnita manteniendo una comunicación transparente con los diversos tipos de usuarios en el sistema, así como la organización en grupos para llevar a cabo diferentes actividades. Pero su alcance no llega hasta aquí, existen diversos trabajos futuros para la expansión de la plataforma en miras a convertirse en una red social robusta y predilecta para todos los estudiantes y docentes de la UCM.

De entre los trabajos futuros mencionados anteriormente, se expone a continuación una lista de forma explicativa:

* **Nuevas Salas de Chat:** La implementación de salas de chat compuesta de un número predeterminado de usuarios, definidos por un anfitrión.
* **Notificaciones:** La implementación de un sistema completo de notificaciones que den aviso de diversos tipos de sucesos de interés para los usuarios.
* **Métricas de Usuario:** Posibilidad de crear un contador de archivos, preguntas, y avisos que lleva a cabo un usuario, entregando información útil de dichos elementos.
* **Conexión con Servicios de Email:** Conexiones con un servidor de correos electrónicos (SMTP), con el fin de agilizar la comunicación de soporte con los diferentes usuarios.
* **Conexión con Portales Externos:** Conexiones con Google y Apple, con el fin de generar más versatilidad a la hora de registrarse en la aplicación.

Para la realización de todos estos trabajos futuros y la contribución al crecimiento del proyecto, es ideal que se instaure una nueva etapa de la aplicación.

**Bibliografía**

[ALB2021]. DEYIMAR ALBORNOZ. “¿Qué es un VPS? Todo lo que necesitas saber sobre servidores virtuales”, <https://www.hostinger.es/tutoriales/que-es-un-vps>, Equipo de SEO & Localization de Hostinger.

[ALI2012]. Servicio de Informática ASP.NET MVC 3 Framework, *“Modelo vista controlador (MVC)”,* [Modelo vista controlador (MVC). Servicio de Informática ASP.NET MVC 3 Framework (ua.es),](https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html) , Universidad de Alicante.

[ANG2020]. GOOGLE LLC. *“Funcionalidades y Ventajas”,* <https://docs.angular.lat/features>, Google llc.

[AST2021]. ABHINAV ASTHANA. *“Postman Learning Center”,* <https://learning.postman.com/>,Postman's CEO and co-founder.

[ATL2017]. ATLASSIAN. *“Acerca de Trello: Lo que hay detrás de los tableros”,* <https://trello.com/about>, Trello.com.

[AWS2022]. AMAZON WEB SERVICES. *“Amazon Simple Storage Service Documentation”*, <https://docs.aws.amazon.com/s3/?id=docs_gateway>, Amazon Web Services, Inc., or its affiliates.

[BIT2019]. BITBUCKET. *“Qué es Git: Rendimiento Seguridad Flexibilidad Control de versiones con Git”,* <https://www.atlassian.com/es/git/tutorials/what-is-git>, Bitbucket.

[BLA2018] ALBERTO BLANCH, MANUEL LEÓN, MARCO LOZANO, MARÍA GARCÍA RUESGAS, SERGIO ARIAS, FERNÁN GARCÍA DE ZÚÑIGA. *“Por qué elegir PostgreSQL y llevarlo a Cloud”,* <https://www.arsys.es/blog/soluciones/postgresql-servidores/>, Arsys.

[CAS2016]. Ma. Cecilia Castillo Valenzuela, Leticia Cruces Hernández, Ma. de los Ángeles Guerra Goglios, *“La Veracidad de la Información Expuesta en las Redes Sociales”* Revista Educación y Tecnología, Año 05. Número 08. Vol 2. 2016, Pág. 26-40.

[CHA2008]. SCOTT CHACON, JASON LONG. “*Git --everything-is-local”,* <https://git-scm.com/about>, Git-scm.com.

[CIR2020]. ANA CIRUJANO. “Qué es Figma”, <https://3ymedia.school/que-es-figma/>, 3ymedia school.

[DEP2014]. SASCHA DEPOLD UHLANDSTR. *“Sequelize”,* <https://sequelize.org/master/>, sequelize.org.

[DRU2022]. CLAIRE DRUMOND. *“Scrum: Aprende a utilizar scrum con lo mejor de él”,* <https://www.atlassian.com/es/agile/scrum>, estratega de marketing, ponente y redactora de Atlassian.

[GUS2019] GUSTAVO B. *“¿Qué es un hosting y cómo funciona?”*, <https://www.hostinger.es/tutoriales/que-es-un-hosting>, Hostinger tutorials.

[HER2021]. RENATO HENRÍQUEZ SALAZAR, “*El golpe de la pandemia en la salud mental de las y los estudiantes universitarios”*, <https://www.uchile.cl/noticias/176250/el-golpe-de-la-pandemia-en-la-salud-mental-de-las-y-los-estudiantes>, Universidad de Chile.

[HUA2020]. JOEL FRANCIA HUAMBACHANO. “Scrum no es una metodología, es un marco de trabajo”, <https://www.scrum.org/resources/blog/scrum-no-es-una-metodologia-es-un-marco-de-trabajo>,

[ION2013] IONICFRAMEWORK. *“Introduction to Ionic”,* <https://ionicframework.com/docs>, ionicframework.com.

[ISA2019] ISARQ. *¿Qué es una API y para qué sirve?,* [¿Qué es una API y para qué sirve? – Proyectos y Desarrollos ISARQ C.A.,](https://isarq.com/index.php/2018/06/29/ques-es-api-para-que-sirve/) isarq.com.

[LGS2017] Scrum, <https://www.scrumguides.org/docs/scrumguide/v2017/2017>, Schwaber, K

[MDN2021]. BY MDN CONTRIBUTORS. *“Generalidades del protocolo HTTP”,* <https://developer.mozilla.org/es/docs/Web/HTTP/Overview>, Mozilla.org.

[MUR2018]. JOSÉ ANTONIO MURO. *“¿Qué es un ORM?”*, <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html>, jefe de equipo en la competencia Microsoft dentro del área de DxD de Deloitte.

[NPM2019]. NPMJS. *“About npm”*, <https://docs.npmjs.com/about-npm>, docs.npmjs.com

[NOD2016]. OPENJS FOUNDATION. *“Acerca de Node.js®”,* <https://nodejs.org/es/about/>, NodeJS.org.

[PAR2016]. MARINA PARRA. *“¿Qué es y cómo funciona Trello?”*, <https://www.postedin.com/blog/que-es-y-como-funciona-trello/>, Periodista digital freelance Nombre de la empresa The Branding Brothers / Postedin Profesional independiente.

[PAR2016a] MARINA PARRA. *“¿Qué es y cómo funciona Trello?”*, <https://www.postedin.com/blog/que-es-y-como-funciona-trello/>, Periodista digital freelance Nombre de la empresa The Branding Brothers / Postedin Profesional independiente.

[PER2020]. ALBERTO PÉREZ ROMÁN. *“Comparación de rendimientos entre bases de datos Relacionales, NoSQL y Blockchain”*, Universidad de Málaga, Pág. 56.

[POS2022]. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *“Why use PostgreSQL?”,* <https://www.postgresql.org/about/>,postgresql.org.

[RED2020] REDHAT. *“¿Que es una API REST?”*, <https://www.redhat.com/es/topics/api/what-is-a-rest-api>, Redhat.com.

[RED2021] REDHAT. *“¿Qué es el middleware?”,* [¿Qué es el middleware? (redhat.com),](https://www.redhat.com/es/topics/middleware/what-is-middleware) Redhat.com.

[ROB2019]. ÁNGEL ROBLEDANO. *“Qué es MongoDB”,* <https://openwebinars.net/blog/que-es-mongodb/>, Product Manager – OpenWebinars.

[SCH2014b]. KEN SCHWABER Y JEFF SUTHERLAND. *“Scrum Team”*, La Guía Definitiva de Scrum: Las Reglas del Juego. (Vol. 1.0, p. 21).

[SOC2022]. SOCKET.IO. *“What Socket.IO is”,* <https://socket.io/docs/v4/>, Copyright © 2022 Socket.IO.

[SRI2020]. BHAVANA SRINIVAS. *“Welcome to Netlify”,* <https://docs.netlify.com/>, Senior solution manager.

[RAM2017]. KEREN RAMÍREZ-ACOSTA. *“Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo”*, Ramírez-Acosta, K. Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo. Tecnología en Marcha. Número Especial Movilidad Estudiantil 4. Pág. 49-54.