**Documento de arquitectura del software**

**Versión 2.0**

**Autores:**

Miguel Ramírez Osorio

Valentín Martínez Motta

**Proyecto:**

Knowl App

**Fecha:**

08 de julio de 2021

**1. Documento de arquitectura del software**

**1.1. Objetivo del documento:**

Este documento es actualizable de acuerdo a los cambios técnicos de arquitectura que vayan apareciendo. Se detalla el grupo que mantiene la infraestructura técnica y los requerimientos de cambios que el grupo pudiera tener.

El objetivo del documento es sustentar de manera organizada la Arquitectura Técnica Organizacional y proporcionar una fuente de referencia para los analistas y diseñadores de la aplicación.

Los analistas y desarrolladores de la aplicación usaran este documento con fines informativos de la comprensión de la actual arquitectura, y para los propósitos del análisis, con el fin de discernir si la arquitectura, como es, apoyará sus requisitos de la aplicación.

Si esto exigiera unos cambios a la arquitectura, estos se harán previa revisión con el equipo de Infraestructura para su resolución. El diseñador de la aplicación hará las modificaciones apropiadas entonces a este documento.

**1.2. Definición de la arquitectura:**

Respecto a la arquitectura, se ha decidido separar el backend (la parte del servidor) del frontend (el sitio web). Manteniéndolos de forma independiente ganamos comodidad a la hora de desarrollar cada componente y flexibilidad si se desea cambiar alguna tecnología.

**1.3. Tecnologías a emplear:**

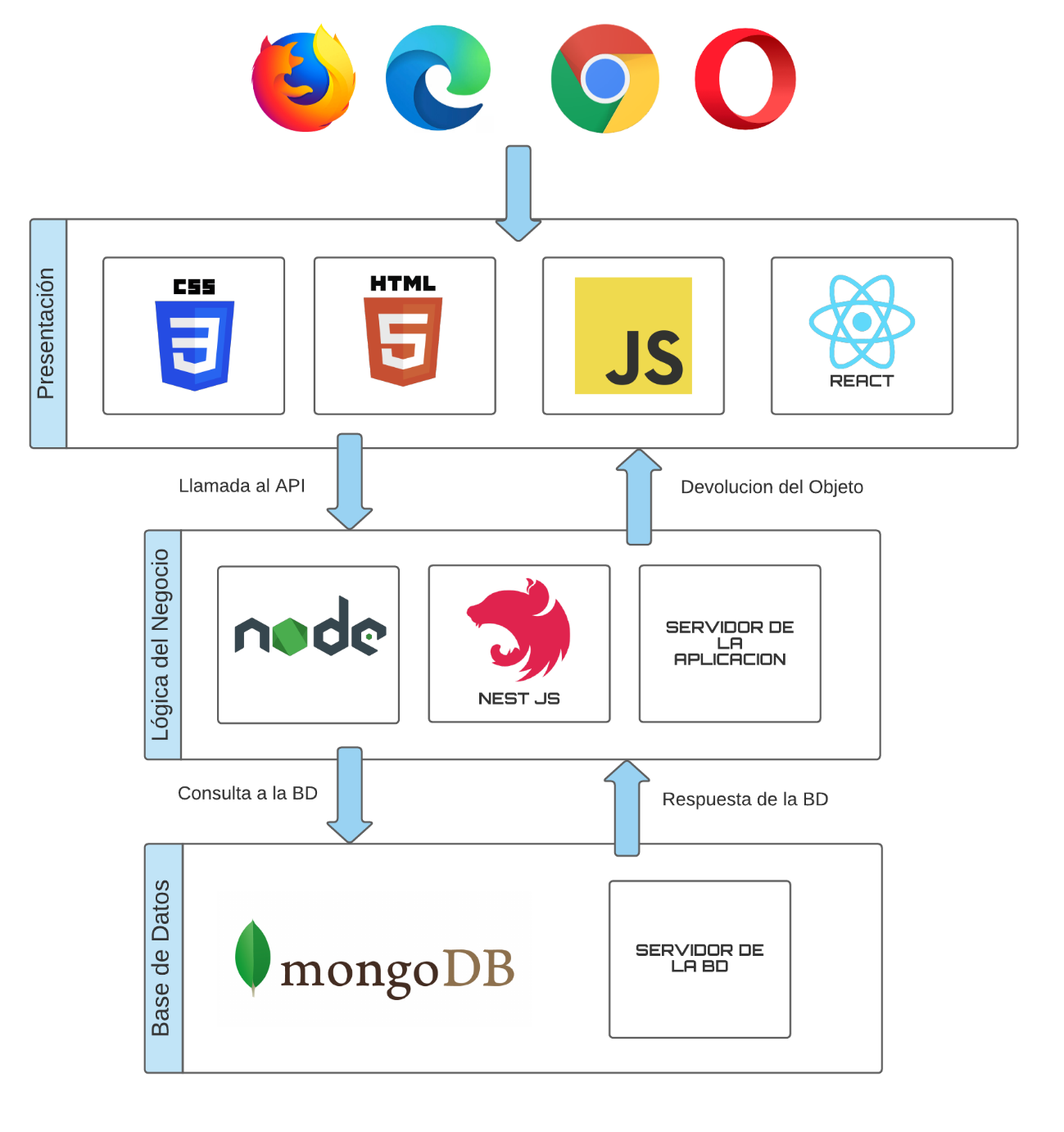
* **NodeJS:** Es un entorno de desarrollo del lado del servidor basado en JavaScript que usa el motor V8 de Google, asíncrono y gran uso de I/O de datos basado en eventos; creado para ser útil en programas altamente escalables. Se ha escogido NodeJS como entorno en el lado del servidor principalmente porque nos permite usar JavaScript para el desarrollo del backend. Por otro lado, cuenta con un gestor de paquetes (NPM) con una enorme comunidad de desarrolladores que le da soporte, esto nos permite encontrar fácilmente paquetes que resuelven problemas sencillos.
* **NestJS:** Es un framework desarrollado completamente en TypeScript para NodeJS con un enfoque de lado servidor. Este framework copia muchas de las ideas que tiene detrás Spring como Framework clásico. Se ha escogido porque tiene la ventaja de que se tenga un único runtime (NodeJS) y se simplifique la gestión de la configuración de los proyectos, así como los procesos de integración continua; pero también hace que un solo desarrollador tenga que cambiar mucho de chip para mantener las dos capas.
* **React:** Es una librería JavaScript focalizada en el desarrollo de interfaces de usuario. Así se define la propia librería y evidentemente, esa es su principal área de trabajo. Se trabajará con React debido a que es un excelente aliado para hacer todo tipo de aplicaciones web, SPA (Single Page Application) o incluso aplicaciones para móviles.
* **MongoDB:** Es una base de datos NoSQL de carácter general, basada en documentos para el desarrollo de aplicaciones modernas. Utiliza estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, lo que facilita su acceso desde las aplicaciones. Se ha decidido usar MongoDB porque suponía un reto mayor el cambiar de paradigma de base de datos. A lo largo de la carrera estudiamos SQL y bases de datos relacionales, viendo solo de forma teórica otros tipos de bases de datos.
* **Git:** Es un software de control de versiones, y junto con GitHub, es lo que se empleará a lo largo del desarrollo del proyecto para mantener una organización adecuada

**1.4. Capas arquitectónicas:**

El sistema Knowl-app estará compuesto por las siguientes capas arquitectónicas:

* **Presentación:** Toda la parte de la presentación se realizará con React. Hay que diferenciar entre vistas y componentes; las vistas son las diferentes ventanas de la página web (página principal, la sección de notas, la lista de usuarios…), en cambio, los componentes son pequeños trozos de código que se repiten en las diferentes vistas (barra de navegación, footer…). El script es el controlador, que manejará las distintas acciones (pulsación de botones, peticiones a la API).
* **Lógica de negocios**: El API se ha desarrollado en el entorno de ejecución de NodeJS; usando NestJS como framework para manejar las peticiones (procesamiento de las peticiones, aplicar los middlewares necesarios, etc.) a los diferentes endpoints.
* **Capa de datos:** Mediante Mongoose se creará la conexión con la base de datos y así, disponer de una interfaz para tratar con esta.

El diagrama inicial de la arquitectura del sistema es el siguiente:



**1.4. Estructura de los componentes arquitectónicos:**

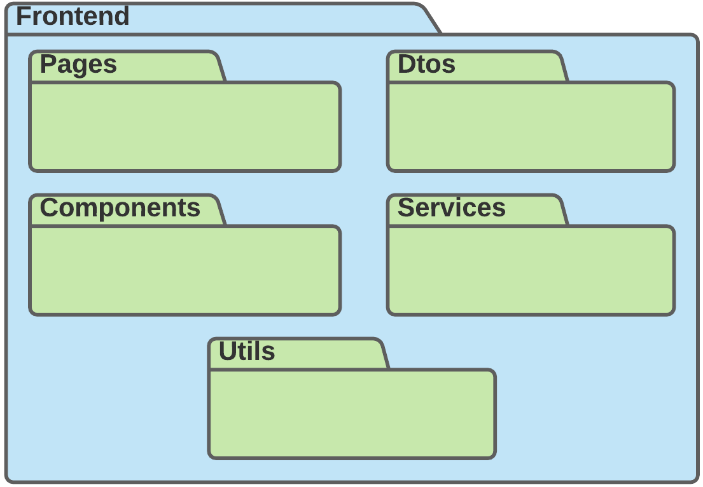
**1.4.1. Estructura de la capa de presentación**

La capa de presentación, o front-end, se caracteriza por emplear componentes de React reutilizables, por ese motivo estos se colocarán en una carpeta llamada “Components”, en ella tendremos tanto componentes de grano fino como de grano grueso. Técnicamente, las páginas que React Router DOM utiliza también son componentes, pero al tener una complejidad mayor y ser usualmente únicos en cada página, se guardarán en la carpeta “Pages”.

La carpeta “Services” guardará los archivos que contienen las funciones que permiten comunicarse con el back-end, a las cuales accederán los componentes para darle funcionalidad completa al sistema.

La carpeta “Dtos” contendrá las interfaces que definen los tipos a emplear al realizar peticiones hacia el back-end y al recibir respuestas del mismo. Por último, la carpeta “Utils” almacenará archivos que contengan cualquier tipo de código utilitario.

En resumen, la estructura de la capa de presentación, o front-end, será la siguiente:



**1.4.2. Estructura de la capa de lógica de negocio**

La capa de lógica de negocio, o back-end, se caracteriza por seguir una arquitectura modular gracias al empleo de NestJS. Los módulos principales, el módulo de usuarios (en “UserModule”), el módulo de inicio de sesión (en “LoginModule”) y el módulo de tecnologías (en “TechnologyModule”), son componentes principales y contienen internamente funcionalidades como: el definir los endpoints y rutas, el contener los dtos necesarios para validar los datos recibidos y enviados, el realizar transacciones con la base de datos, etc.

Por último, la carpeta “Middlewares” contiene diversas funciones a ejecutar antes de ejecutar el código contenido en cada ruta, y la carpeta “Config” contiene diversa información necesaria para configurar el sistema.

En resumen, la estructura de la capa de lógica de negocio, o back-end, será la siguiente:

