**Reporte del desarrollo del software**

**Versión 2.0**

**Autores:**

Martín Pérez Puchuri

Valentín Martínez Motta

Roberto Meregildo Guillén

**Proyecto:**

Knowl App

**Fecha:**

2 de agosto de 2021

**1. Reporte del desarrollo del software**

**1.1. Objetivo del documento:**

El objetivo de este documento es reportar los avances que se vienen realizando hasta la fecha en el desarrollo del software correspondiente al proyecto.

**2. Desarrollo del software durante el primer sprint**

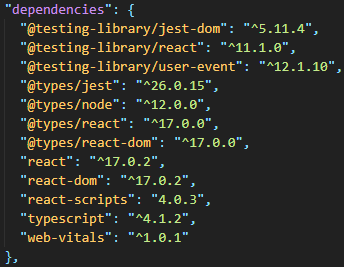
**2.1. Desarrollo de la aplicación del lado del cliente:**

***2.1.1. Inicialización de la aplicación***

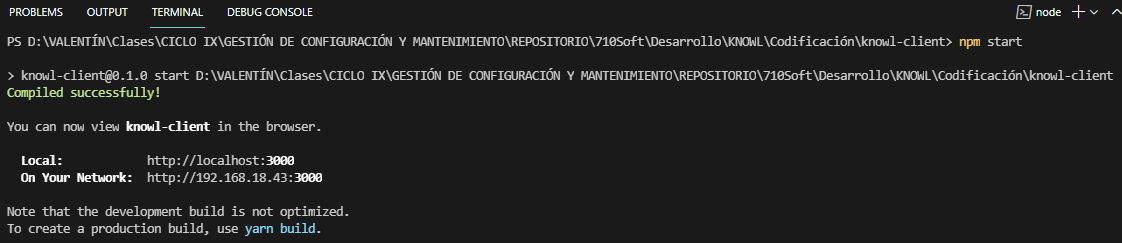
En el presente sprint se instaló la librería de UI principal y se inicializó el sistema. El código de la aplicación del lado del cliente se encuentra en la carpeta **knowl-client** del repositorio del proyecto.

En KW\_AS (Documento de arquitectura del software) se detallaron las tecnologías de la aplicación del lado de cliente a emplear. La librería sobre la que toda la aplicación funcionará es React, la cual fue instalada en su versión de TypeScript.

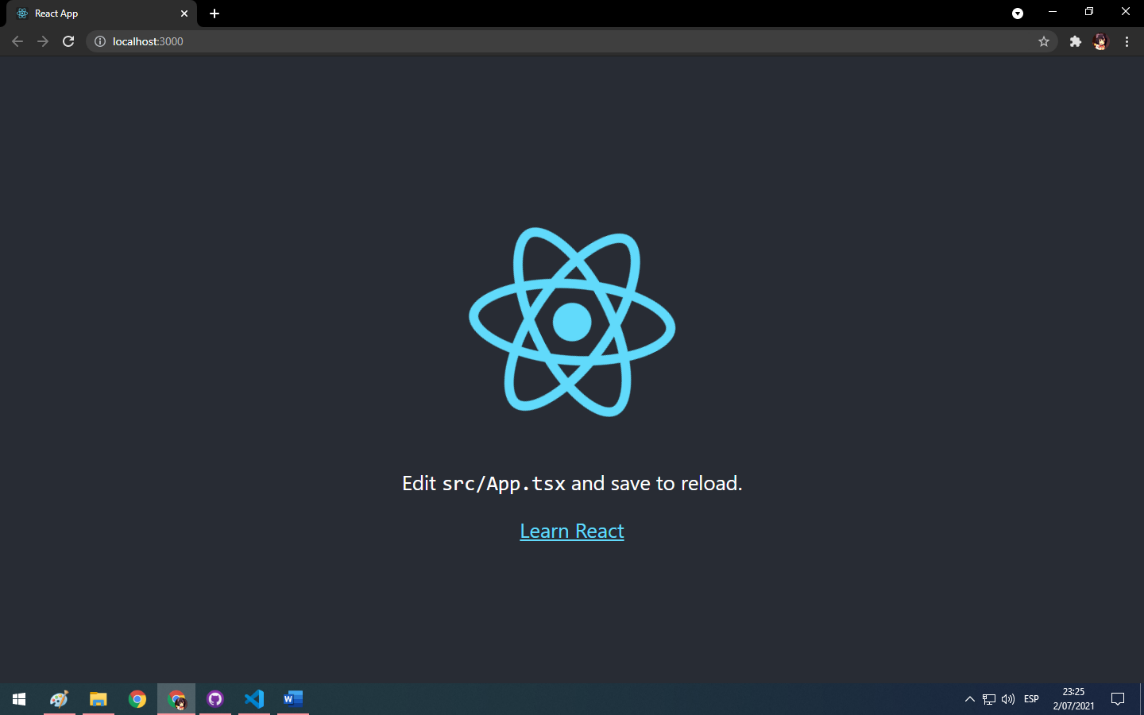
Las dependencias que se añadieron al proyecto con la instalación de React, indicadas en el archivo “package.json”, fueron las siguientes:



Para comprobar que la aplicación del lado del cliente funciona correctamente hasta el momento, fue ejecutada de manera local. Los mensajes provistos por el terminal indicaron que la ejecución fue exitosa. Dichos mensajes del terminal fueron los siguientes:



Podemos acceder a “localhost:3000” para comprobar que la aplicación está funcionando. Se muestra la siguiente pantalla plantilla generada por React:



**2.2. Desarrollo de la aplicación del lado del servidor:**

***2.2.1. Inicialización de la aplicación***

En el presente sprint se instalaron las dependencias principales y se inicializó el sistema. El código de la aplicación del lado del servidor se encuentra en la carpeta **knowl-server** del repositorio del proyecto.

En KW\_AS (Documento de arquitectura del software) se detallaron las tecnologías de la aplicación del lado de servidor a emplear, siendo NestJS el framework principal para el desarrollo y la primera tecnología instalada.

Las dependencias que se añadieron al proyecto con la instalación de NestJS, indicadas en el archivo “package.json”, fueron las siguientes:



Al instalarse, NestJS genera varios archivos que contienen código plantilla que debe modificarse conforme se va desarrollando el sistema.

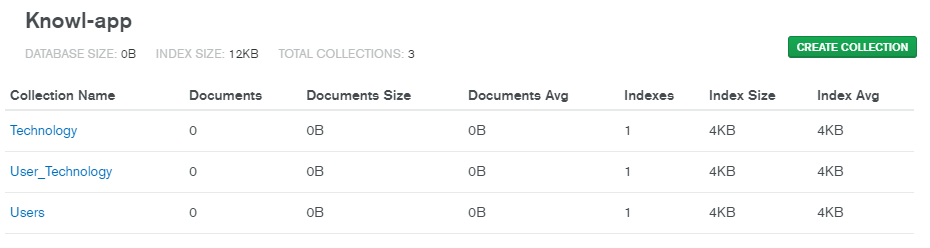
Para no incurrir en hardcoding, se emplearán algunas variables de entorno que facilitarán la gestión de ciertos datos de manera global. NestJS posee un módulo oficial que permite utilizar dichas variables de entorno, el cual viene incluido en el paquete “@nestjs/config”.

Respecto a la base de datos, se mencionó en documentación previa que se trabajaría con MongoDB. NestJS también posee un módulo oficial que permite el empleo de Mongoose como ODM para interactuar con las bases de datos en MongoDB. Este módulo viene incluido en el paquete “@nestjs/mongoose”, el cual para funcionar también requiere de la instalación del paquete Mongoose.

La lista de dependencias hasta el momento será la siguiente:

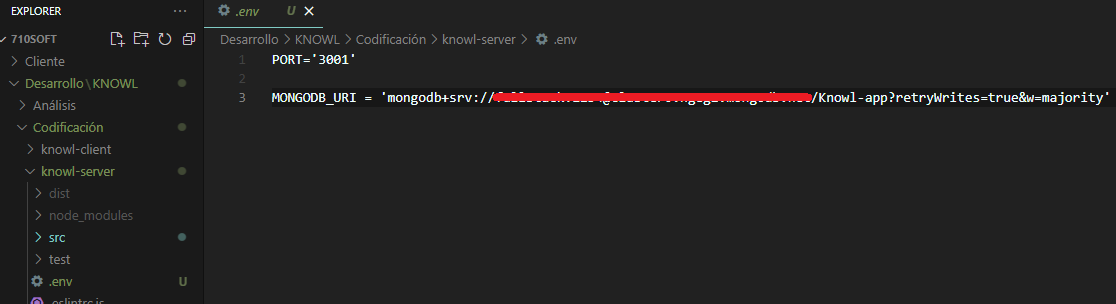


Se ha inicializado la base de datos en el servicio MongoDB Atlas y las colecciones a emplear siguiendo las indicaciones del KW\_DB (Doc. de especificación de la base de datos). En la siguiente imagen se puede observar la base de datos y sus colecciones tal como lo muestra MongoDB Atlas:



Se creó el archivo “.env” en la carpeta del software del lado del servidor “knowl-server”, el cual incluirá las variables de entorno que serán inicializadas con el módulo de configuración de NestJS. Este archivo por el momento incluirá dos variables de entorno: la variable de entorno PORT que almacena el valor del puerto a utilizar (en este caso 3001), y la variable de entorno MONGODB\_URI que almacena el URI para conectarse a la base de datos, el cual será generado por el servicio de MongoDB Atlas.

El archivo “.env” será entonces:

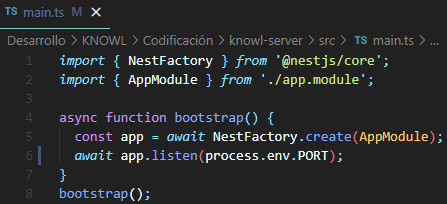


Por último, debemos importar e inicializar tanto el módulo de configuración como el módulo de Mongoose en el módulo principal de la aplicación. El módulo de Mongoose, para inicializarse, emplea la URI que debe estar almacenada en la variable de entorno correspondiente.

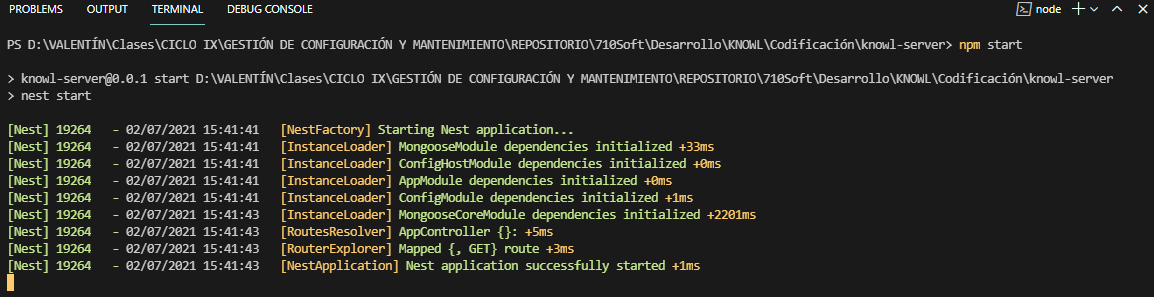
El módulo principal de la aplicación (“app.module.ts”) entonces se verá de la siguiente manera:



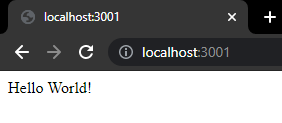
El archivo que se ejecuta para iniciar el sistema, “main.ts”, también debe emplear la variable de entorno correspondiente al puerto. Por tanto quedará de la siguiente manera:



Para comprobar que la aplicación del lado del servidor funciona correctamente hasta el momento, fue ejecutada de manera local. Los mensajes provistos por el terminal indicaron que la ejecución fue exitosa (incluyendo la inicialización de los módulos instalados y la conexión a la base de datos). Dichos mensajes del terminal fueron los siguientes:



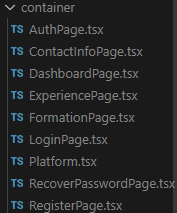
Podemos acceder a “localhost:3001” para comprobar que la aplicación está funcionando. Se muestra el siguiente mensaje (que se encuentra en la ruta generada a modo de plantilla por el mismo NestJS):



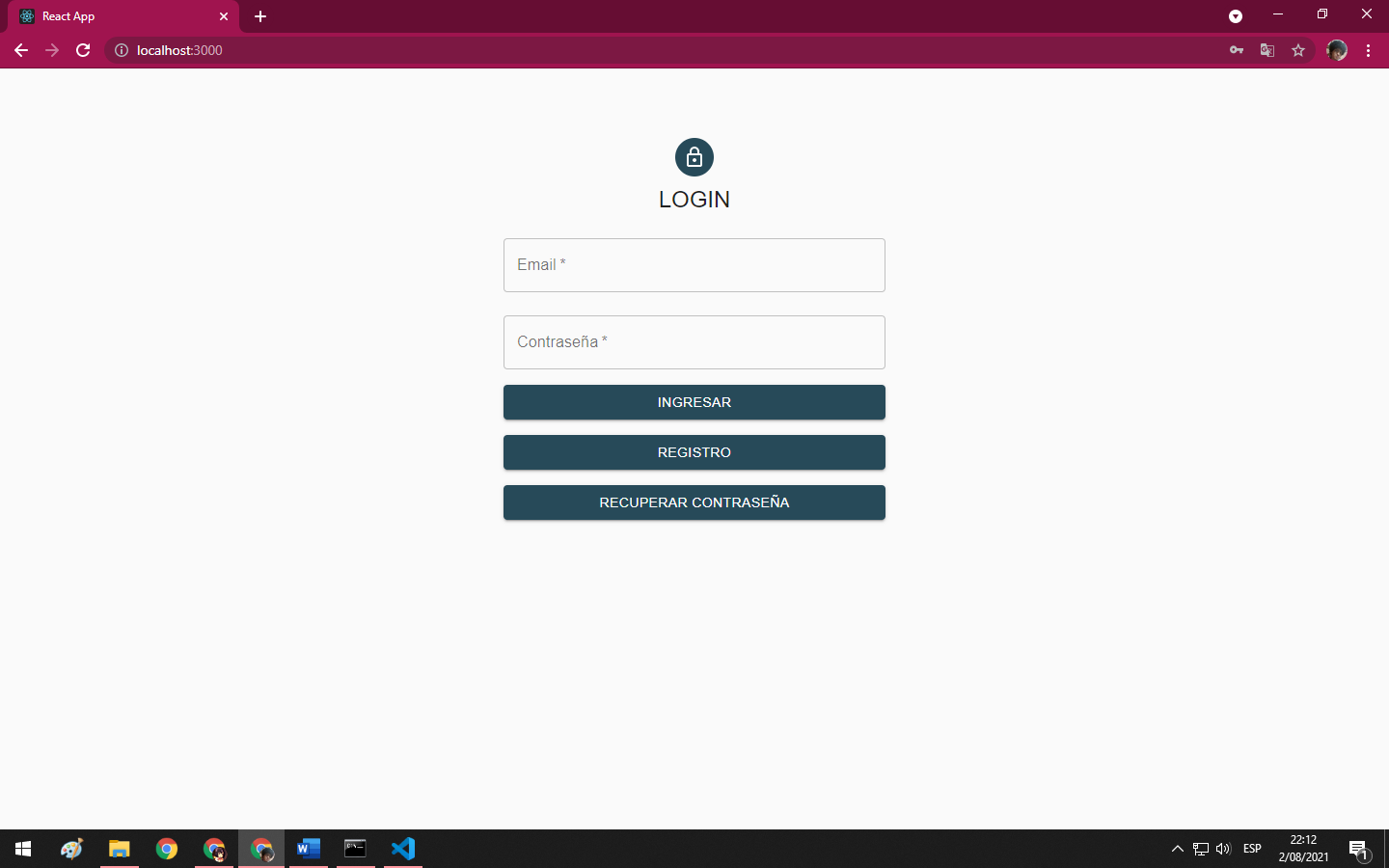
**3. Desarrollo del software durante el segundo sprint**

**3.1. Desarrollo de la aplicación del lado del cliente:**

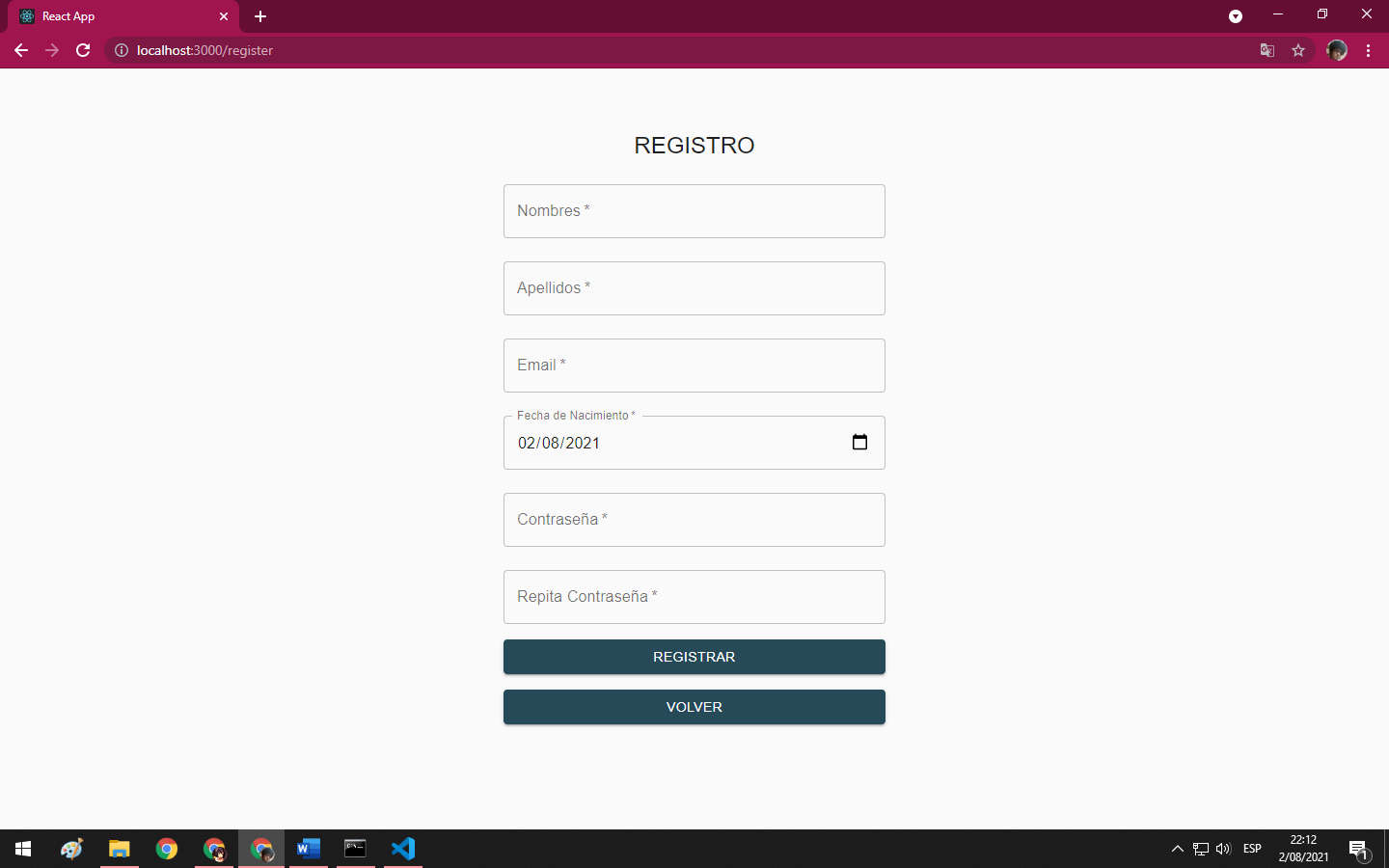
Respecto al front-end, se elaboraron varias páginas y sus componentes. Las páginas, que en realidad son componentes de grano grueso, se encuentran en “src/presentation/container”.



Entre algunas de estas páginas tenemos a la página de inicio de sesión o login, la cual renderiza un formulario que incluye botones que permiten redirigirse a otras páginas, o iniciar sesión si se ingresan las credenciales correctas.



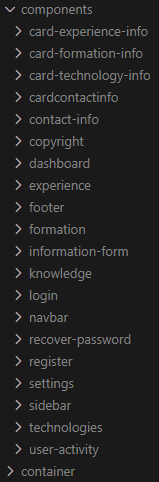
De la misma manera tenemos a la página de registro, que permite recibir los datos de un nuevo usuario y darle de alta en el sistema.



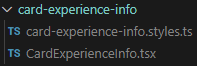
También tenemos a la página de recuperar contraseña, que ingresando el correo de un usuario previamente registrado, le envía un código que le permita recuperar la contraseña.



Estas páginas hacen uso de los múltiples componentes que podemos encontrar en “src/presentation/components”:



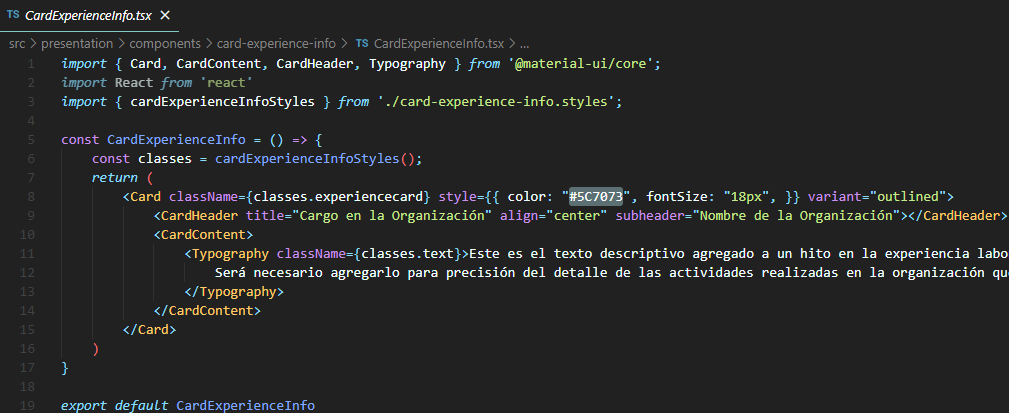
La mayoría de componentes cuenta con un archivo de estilos:



Los estilos de cada componente se aplican en el tema global de la aplicación, gestionado por medio de la librería MaterialUI. A modo de ejemplo, parte de los estilos del componente “CardExperienceInfo” se pueden observar en la siguiente imagen:

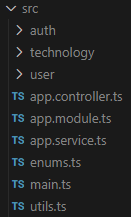


Mientras que los componentes de React son funciones anónimas que retornan su representación en TSX que permite renderizarlos. A modo de ejemplo, podemos ver el componente “CardExperienceInfo” en la siguiente imagen:

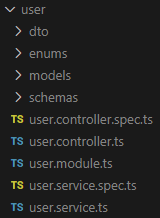


**3.2. Desarrollo de la aplicación del lado del servidor:**

En lo que respecta al back-end, se agregaron los tres módulos principales del sistema: el módulo de autenticación, el módulo de usuarios y el módulo de tecnologías. Cada módulo se encuentra en su carpeta respectiva.



El módulo de usuarios y el de tecnologías tienen una estructura semejante, debido a que gestionan entidades de negocio. Por ejemplo, la estructura del módulo de usuarios es:



En donde “dto” contiene todos los data transfer objects asociados con los usuarios, “enums” contiene los mensajes de error necesarios de incluir en las respuestas a enviar al cliente, “models” contiene la estructura del objeto que manipula la base de datos, y “schemas” contiene las validaciones necesarias a la hora de almacenar o recibir objetos de la base de datos. “user.controller.ts” contiene los endpoints, “user.module” se encarga de instanciar los diversos componentes del módulo y “user.service” hace de mediador entre el controlador y la base de datos.