

TODO app

Introducción.....	2
Objetivos.....	3
Tecnologías.....	4
Node.....	4
React.....	5
Vite.....	5
heroicons.....	5
moment.js.....	5
Express.....	6
jsonwebtoken.....	6
bcrypt.....	6
MongoDB.....	7
Mongoose.....	7
Diseño.....	8
Casos de uso.....	8
Pantallas.....	8
Vista escritorio.....	8
Vista móvil.....	9
Modelo de dominio.....	13
Arquitectura.....	14
Estructura del frontend.....	14
Estructura del backend.....	15
Manual de despliegue y usuario.....	16
Requisitos previos.....	16
Despliegue del backend en Render.....	16
Despliegue del frontend en Vercel.....	16
Configuración de la conexión entre el Frontend y Backend.....	16
Uso de la Aplicación.....	17
Gestión del proyecto.....	18
Problemas encontrados.....	18
Modificaciones.....	18
Mejoras.....	18
Conclusión.....	19
Bibliografía.....	20

Introducción

La *TODO app* es un proyecto motivado por la necesidad de contar con una herramienta sencilla y eficiente para administrar tareas simples. Su objetivo principal es proporcionar a los usuarios una interfaz minimalista, intuitiva y fácil de usar, que les permita gestionar sus tareas de manera eficaz.

La aplicación se desarrolla utilizando el enfoque de *Single Page Application* (SPA). Esto significa que la aplicación carga un único documento web como punto de entrada y, posteriormente, actualiza su contenido de forma dinámica utilizando JavaScript y peticiones de datos asíncronas. Este enfoque mejora la experiencia de usuario al evitar recargas completas de la página y permitir una navegación fluida y rápida (1).

En cuanto al alcance del proyecto, la *TODO app* se enfoca en implementar las operaciones CRUD (2) básicas (Crear, Leer, Actualizar y Eliminar) sobre una base de datos o cualquier otro tipo de almacenamiento de datos persistente. Los usuarios podrán crear nuevas tareas, leer y visualizar las existentes, actualizar su estado o detalles, y eliminar tareas que ya no sean necesarias. Esto brinda una funcionalidad completa para administrar la lista de tareas de manera eficiente.

Para garantizar la seguridad de la aplicación, se ha implementado un esquema de autenticación basado en tokens. Esto significa que los usuarios deben autenticarse proporcionando credenciales válidas para acceder a las funcionalidades de la *TODO app* (3). Este esquema de autenticación simple, pero efectivo, garantiza que solo los usuarios autorizados puedan acceder y modificar la lista de tareas.

En resumen, la *TODO app* es un proyecto de aplicación SPA sencilla para administrar tareas simples. Su objetivo es proporcionar una interfaz minimalista y fácil de usar, permitiendo a los usuarios realizar operaciones CRUD sobre una base de datos. Además, se implementa un esquema de autenticación basado en tokens para garantizar la seguridad de la aplicación.

Objetivos

Los objetivos del proyecto se resumen en:

- Desarrollar un *frontend* (4) SPA minimalista, sencillo y *mobile first* (5) para facilitar de esta manera su uso al usuario final.
- Desarrollar un *backend* (4) que implemente operaciones CRUD básicas junto a una autenticación por *token*.
- Aprovisionarse y emplear una base de datos como almacenamiento persistente para las tareas que almacena nuestra aplicación.
- Desplegar la aplicación de modo que esté disponible en internet para su uso generalista.

TODO app tiene como objetivo principal permitir al usuario final crear tareas pendientes de manera personalizada. Estas tareas estarán disponibles exclusivamente para el usuario a través del uso de credenciales individuales, asegurando así la privacidad y confidencialidad de la información. La aplicación se encarga de almacenar estas tareas de manera segura en una base de datos, garantizando el cumplimiento de las políticas de privacidad establecidas por la propia aplicación.

A través de nuestro sitio web, el usuario podrá crear, modificar y eliminar las tareas según sus necesidades. Además, podrá marcar las tareas como completadas para llevar un seguimiento de su progreso. Es importante destacar que las tareas almacenadas en la cuenta del usuario sólo podrán ser consultadas utilizando sus credenciales de acceso, lo que brinda una capa adicional de seguridad y confidencialidad.

La flexibilidad de la *TODO app* permite al usuario acceder a sus tareas desde cualquier dispositivo con un navegador web y conexión a internet. Esto significa que podrá gestionar su lista de tareas pendientes de forma conveniente y sin limitaciones de ubicación o dispositivo.

Tecnologías

Para desarrollar *TODO app* se ha optado por el *stack* MERN (6), esto consiste en desarrollar una aplicación con las siguientes tecnologías:

- ***MongoDB***
- ***Express***
- ***React***
- ***Node***

La elección de la combinación MERN para desarrollar la *TODO app* se basa en la flexibilidad, escalabilidad y sinergias que estas tecnologías ofrecen en el desarrollo de aplicaciones web modernas. Cada componente de MERN cumple un rol específico en el *stack* tecnológico, permitiendo construir una aplicación robusta, eficiente y fácil de mantener.

Además, se han usado otras librerías como: *Mongoose*, *Vite*, *heroicons*, *moment.js* *jsonwebtoken* y *bcrypt*.



Node

Node (7) es un entorno de ejecución multiplataforma (*Linux*, *Windows* y *macOS*) para *Javascript* que emplea el motor V8 de *Chrome*. Este entorno permite la ejecución de *Javascript* en servidores (fuera del navegador).

La ejecución en servidores habilita que desde *Javascript* se pueda acceder al sistema de archivos del sistema operativo y crear herramientas de consola o servidores web, algo que desde el navegador es imposible. Además, su modelo de asincronía está pensado para escalar y responder adecuadamente a altas cargas de trabajo.

Node también permite crear proyectos y manejar sus dependencias gracias a *npm* (*node package manager*), así como crear *scripts* para las tareas habituales en el proyecto. A día de hoy, existe un rico ecosistema de herramientas para desarrollar aplicaciones con *Node*, ejecutarlas, empaquetarlas, ejecutar pruebas o analizar el código fuente en busca de errores o para aplicar convenciones de estilo.



React

React (8) es una librería de *Javascript* diseñada para la creación de interfaces de usuario. Esta librería es declarativa y hace ameno la creación de interfaces de usuario interactivas. Tiene la capacidad de crear vistas simples para cada estado de la aplicación, además, *React* actualizará los componentes cuyos datos hayan sido modificados utilizando el DOM virtual y un algoritmo diferencial.

Los componentes de *React* son fácilmente reutilizables y aceptan argumentos (*props*) que los hacen verdaderamente flexibles y funcionales.



Vite

Vite (9) es un *framework* moderno y ligero para el desarrollo de aplicaciones web. Este es compatible con *JavaScript* y *TypeScript* y ha sido creado por Evan You, también creador de *Vue*. *Vite* se enfoca en proporcionar un entorno de desarrollo rápido y eficiente para proyectos de *frontend*. Utiliza una arquitectura de servidor de desarrollo basada en ESM (*ECMAScript modules*) y técnicas de análisis en tiempo real para permitir una experiencia de desarrollo ultra rápida. Además, es compatible con varios *frameworks*, lo que lo hace altamente adaptable y fácil de integrar en proyectos existentes.



heroicons

heroicons (10) es una librería de *Javascript* que proporciona un conjunto de iconos escalables y personalizables para la interfaz de usuario. Esta librería provee una gran cantidad de iconos que pueden ser utilizados en diferentes situaciones, como botones, menús, barras de navegación, entre otros. Los iconos son vectoriales y pueden ser redimensionados sin perder calidad, además, se pueden personalizar con diferentes colores y efectos. *Heroicons* es una librería de código abierto y gratuita, lo que la hace muy accesible para los desarrolladores web y diseñadores.



moment.js

Moment.js (11) es una librería de *Javascript* que permite trabajar fácilmente con fechas y horas. Proporciona una gran cantidad de funciones para manipular, analizar y

mostrar fechas y horas, así como también para realizar cálculos y comparaciones entre ellas.

Express

Express

Express.js, o simplemente *Express* (12), es un *framework* para desarrollar aplicaciones web *backend* en *Node*. Ha sido llamado el *framework* de servidor estándar para *Node*. El autor original, TJ Holowaychuk, lo describió como un servidor inspirado en Sinatra, lo que significa que es relativamente minimalista, con muchas características disponibles como complementos. *Express* es el componente de *backend* de stacks tecnológicos populares como MEAN, MERN o MEVN.



jsonwebtoken

JSON Web Token (JWT) (3) es un estándar abierto basado en JSON, propuesto por IETF (RFC 7519), para la creación de *tokens* de acceso que permiten la propagación de identidad y privilegios o *claims*. Los *JSON Web Tokens* están diseñados para ser compactos y poder ser enviados en las URLs o a través de cabeceras HTTP(s).

Por ejemplo, un servidor puede generar un *token* indicando que un usuario tiene privilegios de administrador. El cliente, que recibe ese *token*, entonces puede utilizarlo para probar que está actuando como un administrador. El *token* está firmado por la clave del servidor, así que el cliente y el servidor son ambos capaces de verificar que el token es legítimo.

bcrypt

La librería **bcrypt** (13) está disponible para múltiples lenguajes de programación. Esta librería permite encriptar cómodamente contraseñas antes de guardarlas en la base de datos, además aporta métodos para comparar contraseñas en texto plano con las contraseñas encriptadas.



MongoDB

MongoDB (14) es un sistema de base de datos NoSQL (no relacional), de código abierto y orientado a documentos. Se caracteriza por su flexibilidad y escalabilidad, permitiendo el almacenamiento y la gestión eficiente de grandes volúmenes de datos no estructurados. *MongoDB* utiliza un modelo de datos basado en documentos JSON, lo que significa que los datos se almacenan en documentos flexibles y anidados en lugar de en filas y columnas.

La flexibilidad de sus documentos facilita la adaptación de la estructura de los datos a medida que evoluciona la aplicación. *MongoDB* también ofrece características como replicación, fragmentación y consultas flexibles, lo que lo convierte en una opción popular para aplicaciones web y proyectos que requieren almacenamiento de datos ágil y escalable.



Mongoose

Mongoose (15) es una librería que permite escribir consultas para una base de datos de *MongoDB*, con características como validaciones, construcción de consultas, *middlewares* o conversión de tipos.

La principal ventaja de *Mongoose* es que permite definir esquemas para los documentos de *MongoDB*. *MongoDB* no permite definir esquemas para sus documentos, esta funcionalidad tiene que ser aportada por librerías como la propia *Mongoose* u otros ORM (16).

En el esquema mencionado se especifican los campos que pertenecen a un documento y su tipo, validaciones y configuraciones especiales para su consulta. El esquema también aporta la posibilidad de crear métodos que facilitan la realización de tareas repetitivas contra un documento de la base de datos. Un ejemplo habitual sería la creación de métodos *signUp* y *login* para un documento que almacena usuarios.

Mongoose además, abre las puertas a un conjunto de *plugins* que se pueden emplear para automatizar tareas comunes, tales como el encriptado de información, paginación o consultas adicionales.

Diseño

Casos de uso

TODO app solo tiene un perfil de usuario, este usuario puede:

- Registrarse en la aplicación
- Iniciar sesión en la aplicación
- Crear sus propias tareas
- Consultar sus propias tareas
- Modificar sus tareas
- Añadir una fecha de finalización a sus tareas
- Marcar sus tareas como realizadas
- Borrar sus tareas

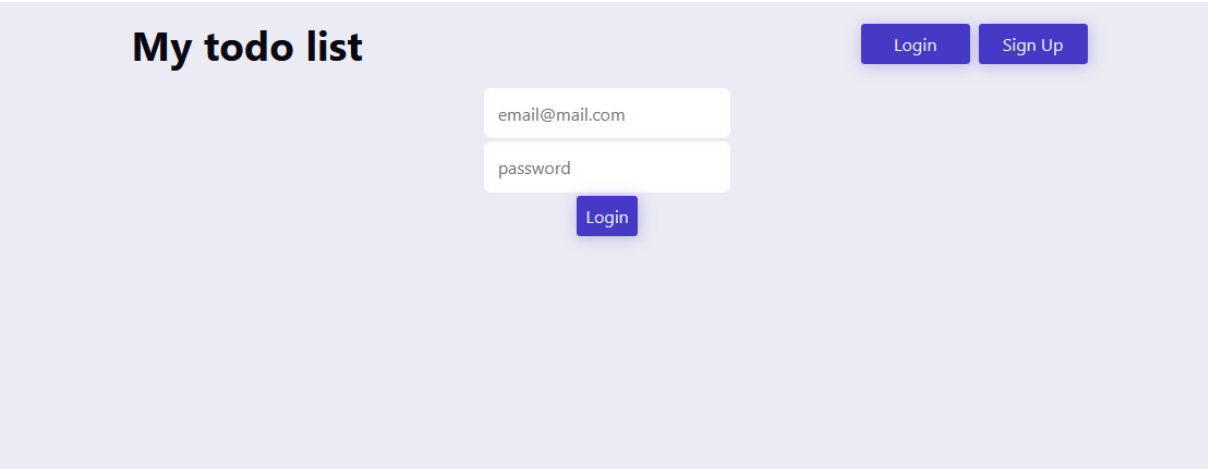
Pantallas

La aplicación consta de tres vistas y una adicional en dispositivos móviles. La primera vista de la aplicación permite al usuario iniciar sesión, se dispone de otra para que pueda registrarse cuando aún no tiene cuenta y, por último, la vista principal donde se muestran las tareas y el usuario puede interaccionar con estas.

En dispositivos móviles es necesaria una vista adicional, esto se debe a que no hay espacio disponible para incluir todas las acciones disponibles en una barra de navegación. Por esto, en lugar de la barra de navegación se incluye un menú hamburguesa que despliega la vista adicional con las acciones correspondientes.

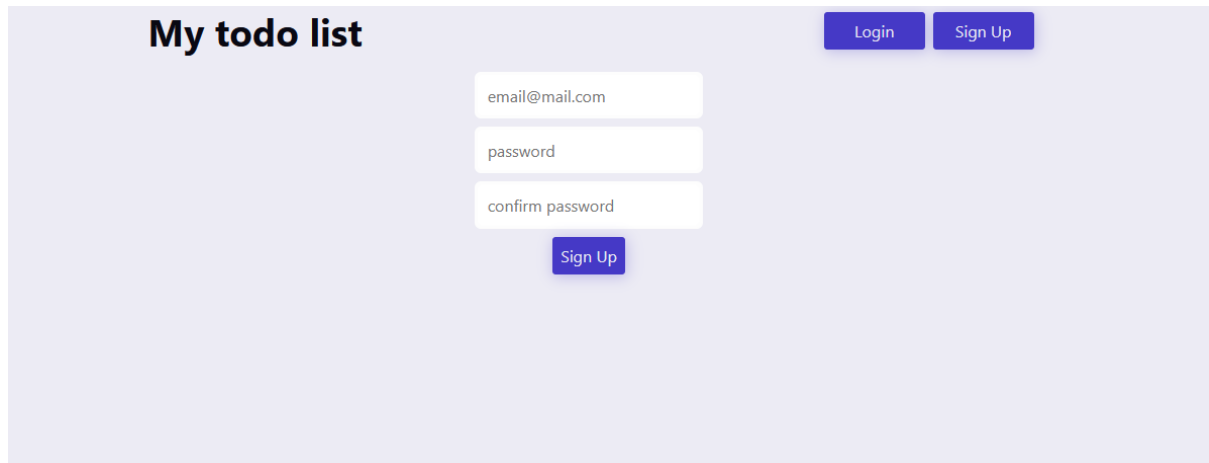
Vista escritorio

En las vistas de escritorio el contenido se muestra centrado con una barra superior. Esta contiene la imagen corporativa a la izquierda y a la derecha los enlaces a las diferentes partes de la aplicación.



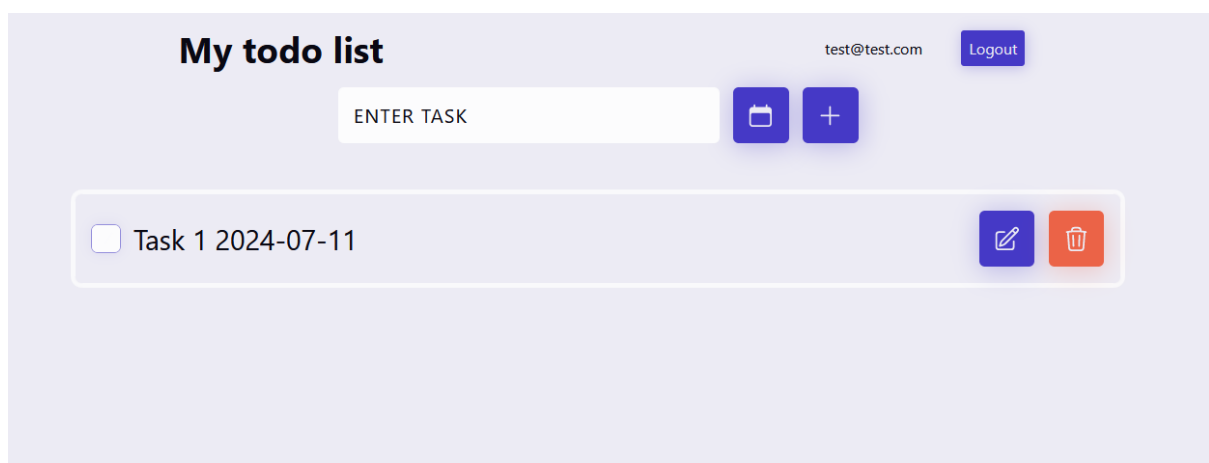
The mockup shows a desktop login interface. At the top left, the text 'My todo list' is displayed in a bold, dark font. At the top right, there are two blue buttons labeled 'Login' and 'Sign Up'. In the center, there is a white login form with two input fields: the first is labeled 'email@mail.com' and the second is labeled 'password'. Below these fields is a blue 'Login' button.

Figura 1. Vista de inicio de sesión (Escritorio).



The registration view for the 'My todo list' app on a desktop screen. The title 'My todo list' is in the top left. In the top right, there are 'Login' and 'Sign Up' buttons. The registration form consists of three input fields: 'email@mail.com', 'password', and 'confirm password'. Below these fields is a 'Sign Up' button.

Figura 2. Vista de registro (Escritorio).

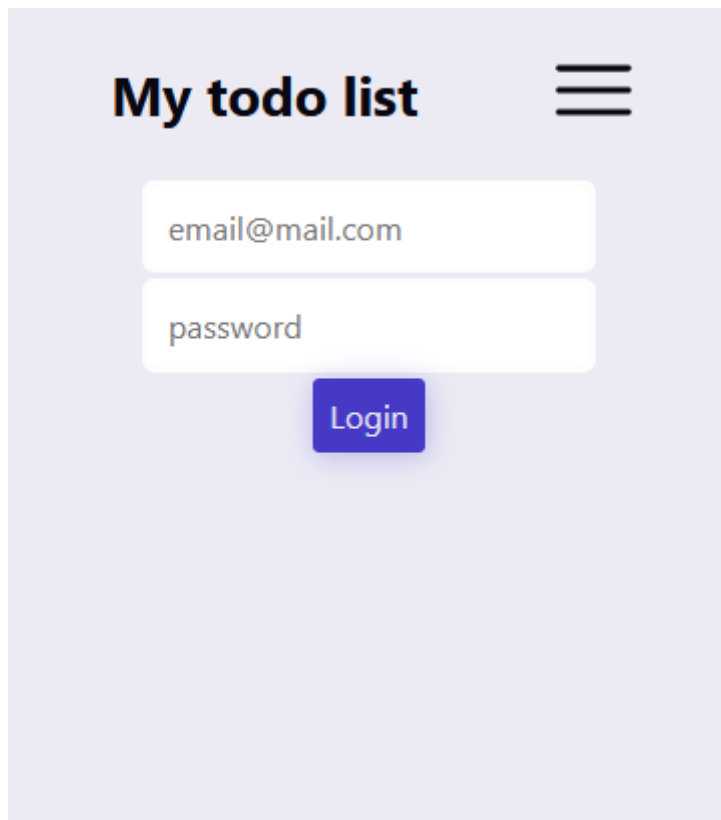


The main view for the 'My todo list' app on a desktop screen. The title 'My todo list' is in the top left. In the top right, the user is logged in as 'test@test.com' with a 'Logout' button. Below the title is an 'ENTER TASK' input field, followed by a calendar icon and a '+' button. A task list is shown below, with the first task being 'Task 1 2024-07-11'. To the right of the task is a blue button with a pencil icon and a red button with a trash icon.

Figura 3. Vista principal (Escritorio).

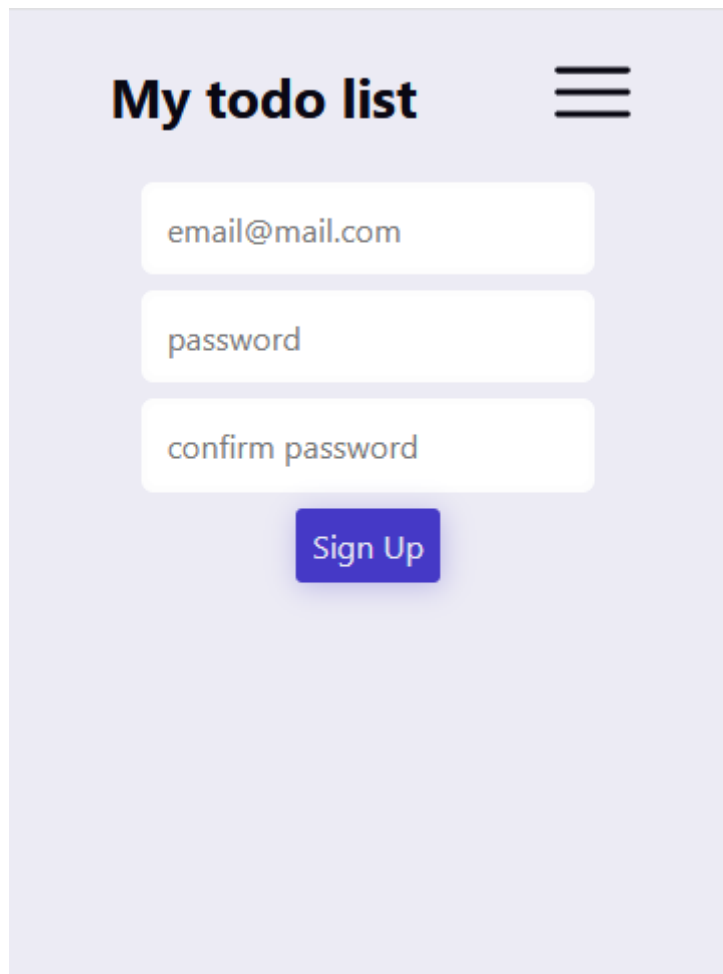
Vista móvil

En las vistas de dispositivos móviles se puede observar el contenido de la aplicación centrado con un logo corporativo y un menú hamburguesa en la parte superior. Este sustituye a los enlaces antes mencionados. Al hacer click en el menú se despliega la cuarta vista con la que se puede navegar por la aplicación.



The image shows a mobile application interface for a login screen. At the top left, the text "My todo list" is displayed in a bold, black font. To its right is a hamburger menu icon consisting of three horizontal lines. Below the title, there are two white input fields with rounded corners. The first field contains the placeholder text "email@mail.com" and the second field contains "password". Both fields have a subtle shadow. Below these fields is a blue rectangular button with the word "Login" in white text. The entire interface is set against a light purple background.

Figura 4. Vista de inicio de sesión (Móvil).



The image shows a mobile registration screen for an app titled "My todo list". The title is in bold black text at the top left, and a hamburger menu icon is at the top right. Below the title, there are three white input fields with rounded corners, each containing placeholder text: "email@mail.com", "password", and "confirm password". Below these fields is a blue button with white text that says "Sign Up". The entire form is set against a light purple background.

Figura 5. Vista de registro (Móvil).

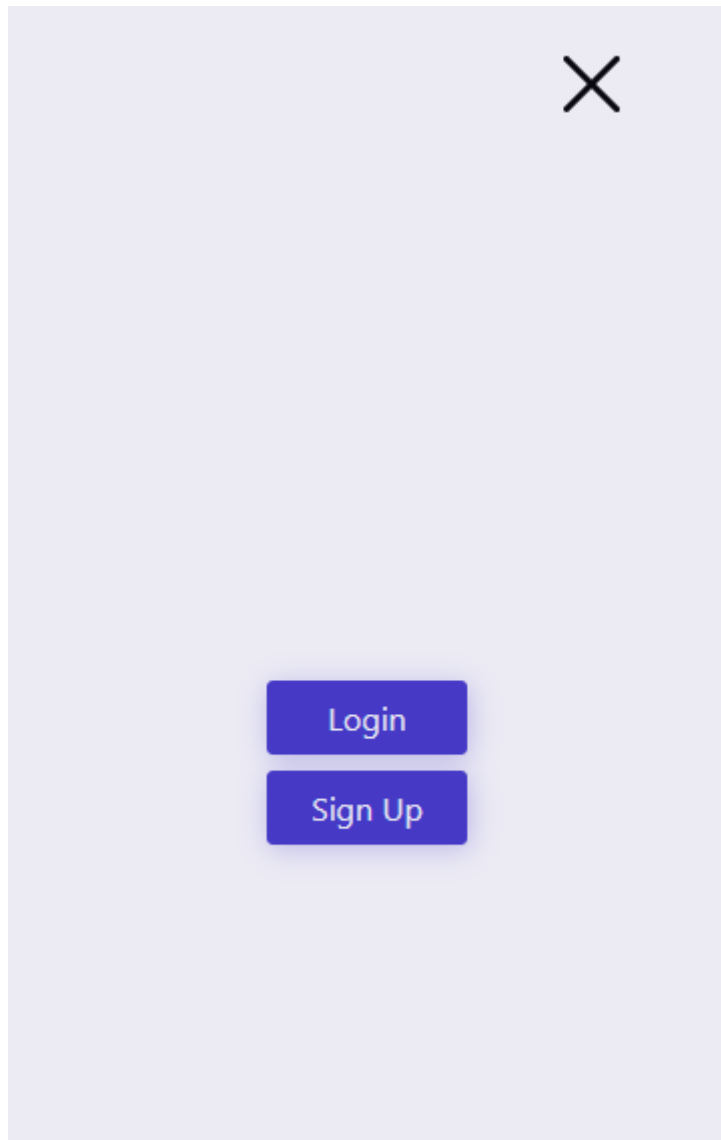


Figura 6. Vista de navegación (Móvil).

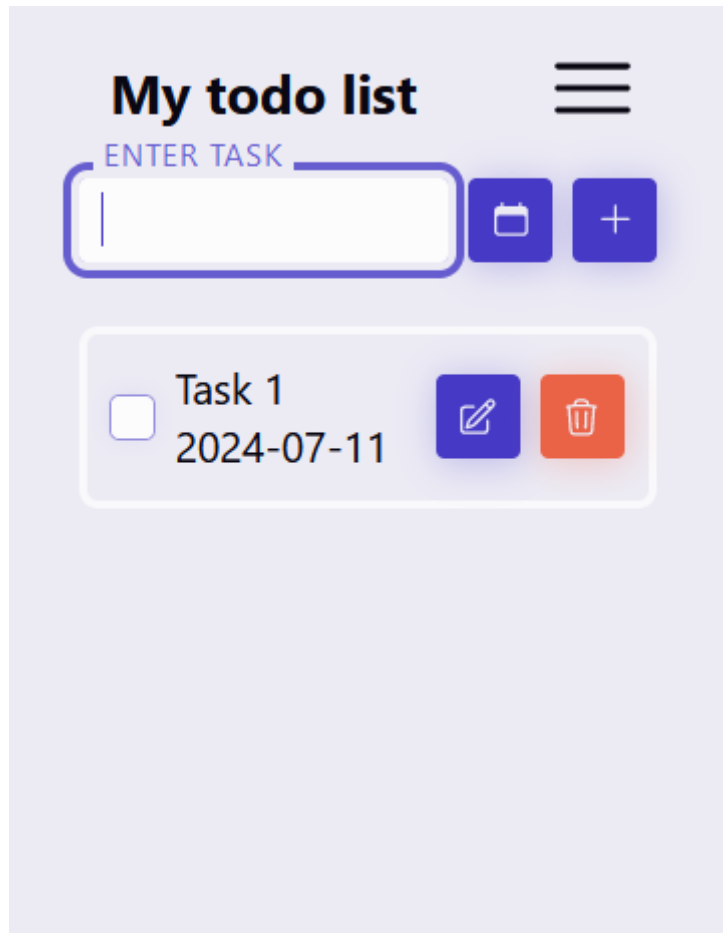


Figura 7. Vista principal (Móvil).

Modelo de dominio

TODO app presenta el siguiente modelo de base de datos :

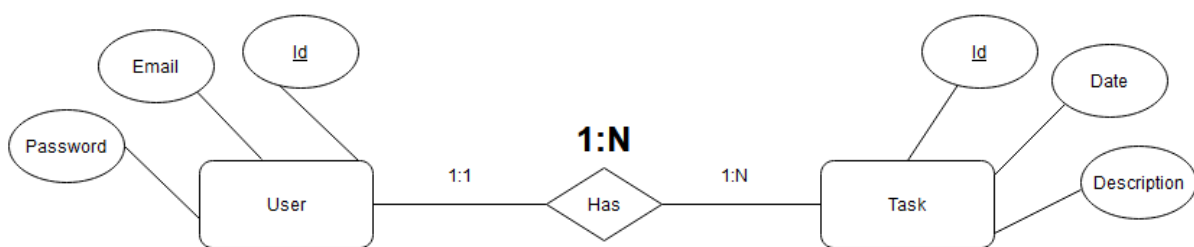


Figura 8. Modelo de base de datos.

Teniendo dos tablas en la base de datos siendo una *User* y la otra *Task* teniendo una relación 1:N

Arquitectura

Estructura del *frontend*

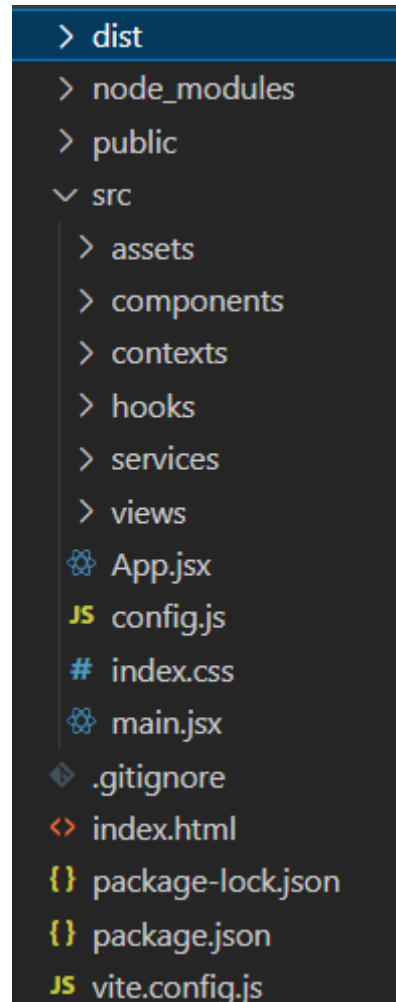


Figura 9. Arquitectura del *frontend*.

La estructura del proyecto es la típica de un proyecto de *react*, las principales carpetas son:

- **PROYECTO_REACT:** contiene la aplicación y los ficheros de configuración.
 - **src:** Contiene el código fuente de la app.
 - **assets:** Imágenes, logotipos y otros estáticos similares que se referencian en los componentes.
 - **components:** contiene componentes de *react* que son utilizados para el desarrollo de la interfaz de usuario.
 - **contexts:** contiene los contextos de la aplicación (17). En nuestro caso, el contexto de autenticación.
 - **hooks:** Contiene *hooks* personalizados (18) para su uso en distintos componentes de la aplicación.

- **services:** contiene los ficheros donde se definen las acciones de acceso a datos que luego son empleadas en los hooks personalizados.
- **views:** Contiene componentes de *react* que definen las diferentes vistas de la aplicación.
- **node_modules:** Contiene las distintas librerías que emplea el proyecto.
- **public:** Contiene los ficheros estáticos que se entregan al cliente sin pasar por ninguna transformación.

Estructura del *backend*

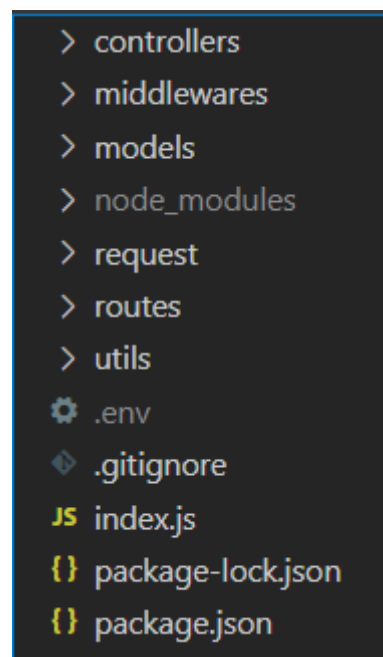


Figura 10. Arquitectura del *backend*.

- **express** contiene las carpetas:
 - **controllers:** Contiene los diferentes controladores (19) de la app.
 - **middlewares:** Incluye los *middlewares* (19) de la app.
 - **models:** Contiene los modelos de *task* y *user* que usamos en la bbdd.
 - **node_modules:** Contiene las distintas librerías que emplea el proyecto.
 - **request:** Almacena una serie de ficheros con peticiones HTTP definidas para su cómodo uso en conjunto con la extensión de *Visual Studio Code REST Client*.
 - **routes:** Contiene los ficheros que definen las rutas del servidor.
 - **utils:** Funciones no relacionadas.

Manual de despliegue y usuario

Este manual proporciona instrucciones detalladas para el despliegue y uso de una aplicación MERN con el *frontend* alojado en *Vercel* (20), el *backend* en *Render* (21) y todo el código fuente almacenado en GitHub.

Requisitos previos

Antes de comenzar, asegúrese de tener lo siguiente:

- Cuentas creadas en MongoDB Atlas (22), *Vercel*, *Render* y GitHub.
- Git instalado en su máquina local.
- Node.js y npm (Node Package Manager) instalados en su máquina local.
- Acceso al código fuente de la aplicación almacenado en un repositorio de GitHub.

Despliegue del *backend* en *Render*

El *backend* de la aplicación, que utiliza Node.js y Express, se desplegará en *Render*. Siga los pasos a continuación:

- Inicie sesión en su cuenta de *Render*.
- Cree un nuevo *web service*.
- Asigne un nombre al servicio y siga las instrucciones proporcionadas por *Render* para clonar el repositorio en el que se encuentra el código fuente de su *backend*.
- Configure las variables de entorno necesarias para su backend, como la cadena de conexión de la base de datos MongoDB.
- Una vez configurado, *Render* se encargará automáticamente de desplegar su *backend* y lo hará nuevamente con cada nuevo cambio en la rama principal.

Despliegue del *frontend* en *Vercel*

El *frontend* de la aplicación, desarrollado con React, se desplegará en *Vercel*. Siga los pasos a continuación:

- Inicie sesión en su cuenta de *Vercel*.
- En el *dashboard* de *Vercel*, haga clic en "Importar proyecto" y seleccione el repositorio de GitHub que contiene el código fuente de su *frontend*.
- Se dejan las opciones de despliegue por defecto.
- Una vez configurado, *Vercel* se encargará automáticamente de desplegar su frontend y lo hará nuevamente con cada nuevo cambio en la rama principal.

Configuración de la conexión entre el Frontend y Backend

Para que el frontend pueda comunicarse correctamente con el backend, es necesario configurar la URL de la API en el código del frontend. Siga los pasos a continuación:

- Abra el código fuente de su frontend en su máquina local.
- Busque el archivo `config.js` y proporcione la URL base de la API en la variable `API_URL`.

Uso de la Aplicación

Una vez que el backend y el frontend estén desplegados correctamente, se puede acceder a la aplicación a través de la URL proporcionada por Vercel.

Gestión del proyecto

El proyecto se ha ido desarrollando a lo largo de las semanas poniendo en común los avances con el tutor de manera quinquenal. Se tenía una idea más o menos clara de lo que se quería hacer en un principio, y se iban descubriendo necesidades durante el proceso de desarrollo. Además, se tenían en cuenta las sugerencias de características del propio tutor.

Problemas encontrados

El menú hamburguesa dio muchos problemas debido a que su funcionalidad atravesaba diversos componentes.

El manejo de fechas es problemático en *JavaScript*, especialmente cuando hay que serializar la fecha y compartirla entre cliente y servidor. Se optó por utilizar la librería *moment.js* para lidiar de una manera más cómoda con estas.

El almacenamiento de *tokens* de autenticación ya expirados en el navegador provocaba un mal funcionamiento de la app. Se optó por eliminar el *token* en el cliente después de una petición no autorizada, redireccionando al usuario a la vista de *Login*.

Modificaciones

El proyecto se modifica debido a la falta de tiempo. Iba ser una aplicación de gestión de notas y se optó por continuar el proyecto de *frontend* que se presentó para la asignatura de Desarrollo en entorno cliente. Aquel proyecto utilizaba una API falsa con JSON server (23), no tenía ninguna base de datos, no tenía un diseño adaptativo, constaba de una única vista y no tenía un sistema de autenticación.

Mejoras

Una posible mejora del proyecto sería el desarrollo de una módulo de usuarios con algunas funcionalidades importantes como por ejemplo: Cambio de contraseña, olvido de contraseña, preferencias de usuario (tema oscuro o luminoso) y otras.

Conclusión

El proyecto de *TODO app* es una aplicación web sencilla pero muy útil para administrar nuestras tareas diarias de manera fácil y eficiente.

La interfaz minimalista e intuitiva facilita su uso y la implementación de operaciones CRUD básicas junto a una autenticación por *token* la hace segura. Además, su soporte para dispositivos móviles permite que podamos acceder a nuestras tareas desde cualquier lugar con conexión a internet.

En resumen, es un proyecto que me ha permitido poner en práctica y aprender cómo enfrentarme a un desarrollo web moderno basado en *JavaScript* y SPA. En este, he abordado tanto el *frontend* como el *backend* completamente desde cero. Además he consumido servicios de infraestructura para desplegar el proyecto (*Vercel*, *Render*) y para la base de datos (*MongoDB Atlas*).

Bibliografía

1. [SPA \(Single-page application\) - MDN Web Docs Glossary: Definitions of Web-related terms](#) (10/05/2023)
2. [CRUD - MDN Web Docs Glossary: Definitions of Web-related terms](#) (10/05/2023)
3. [¿Qué es la autenticación basada en Token?](#) (06/06/2023)
4. [Qué es Frontend y Backend, diferencias y características - Platzi](#) (10/05/2023)
5. [Mobile First - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web](#) (10/05/2023)
6. [Qué es el stack MERN de JavaScript](#) (10/05/2023)
7. [About | Node.js](#) (06/06/2023)
8. [React](#) (06/06/2023)
9. [Guía tutorial inicial de Vite - Javascript en español - Lenguaje JS](#) (06/06/2023)
10. [Heroicons](#) (06/06/2023)
11. [Moment.js](#) (06/06/2023)
12. [Express.js](#) (06/06/2023)
13. [Hashing in Action: Understanding bcrypt](#) (06/06/2023)
14. [What is MongoDB? Features and how it works – TechTarget Definition](#) (06/06/2023)
15. [Mongoose](#) (06/06/2023)
16. [What is an ORM \(Object Relational Mapper\)?](#) (15/05/2023)
17. [Utiliza Context Para el Manejo de Estado en tu Aplicación React](#) (06/06/2023)
18. [React Hooks](#) (06/06/2023)
19. [What is the difference between Middleware and Controllers in Node REST APIs? – Corey Cleary](#) (06/06/2023)
20. [Vercel](#) (06/06/2023)
21. [Render](#) (06/06/2023)
22. [MongoDB Atlas Database | Multi-Cloud Database Service](#) (06/06/2023)
23. [GitHub - typicode/json-server: Get a full fake REST API with zero coding in less than 30 seconds \(seriously\)](#) (06/06/2023)