

MEDUP

Marta Villanueva Muñoz

Proyecto Final

Desarrollo Fullstack

Punto de partida

Intencionalidad

El reto en esta website consistía en crear una aplicación de gestión de items. Siendo un punto de partida genérico, decidí crear un **sistema de gestión de citas médicas, cuya particularidad es la adaptación dependiendo del usuario al que se muestra.**

Inicialmente, la intención era crear una aplicación de gestión hospitalaria en la que se encontrasen todos los datos y documentos relacionados con un hospital. Desde la gestión de salas y habitaciones hasta las pruebas, así como sus resultados.

A continuación, dividiré la presentación en todos los pasos realizados hasta la fecha, creados en un tablero FigJam para completar el proyecto, explicando las dificultades y datos relevantes de cada etapa.

Como dato adicional a tener en cuenta para la corrección, he realizado el proyecto, así como el tablón de Figjam en inglés, ya que la programación en general está dominada por la lengua anglosajona (incluidos los tutoriales o guías) y es un beneficio para mi propio conocimiento y mi portfolio.

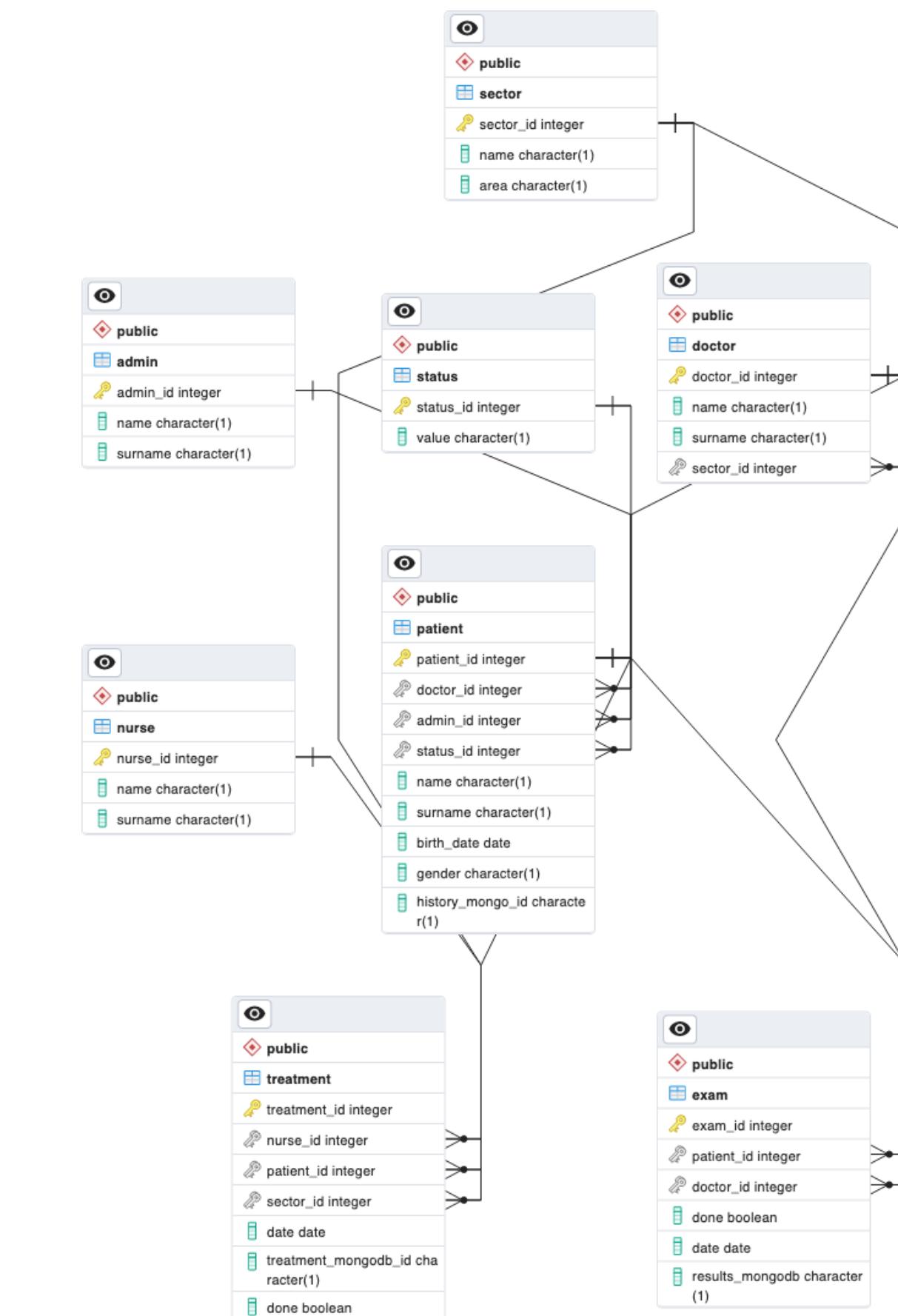
Paso

1

11.06.2023

Siguiendo este esquema, se basaba en **diferentes acciones y permisos** y debía contar con dos bases de datos: una en Postgres y otra en MongoDB.

Razón: en MongoDB podía almacenar un mayor volumen de información, aquella correspondiente a imágenes (radiografías, ecografías..) y, al ser no relacional, me otorgaba mayor libertad. Los usuarios, junto con sus datos y las del propio hospital tendrían sus datos en Postgres.



	ADMIN	NURSE	DOCTOR
PSQL	Read / Write	Read	Read
MONGO	Read	Read / Write	Read / Write

Paso 2

11.06.2023

Posteriormente, planteé los **Casos de uso** en torno a los que giraría la funcionalidad de la aplicación.

Se han mantenido de forma general, dado que la única diferencia es que habría más datos para gestionar (resultados y hospitalización), pero la base seguiría siendo la misma: asignar citas/pruebas y que el usuario pudiera verlas, editarlas y eliminarlas, asignando permisos a través de estructuras condicionales.

Descarté esta idea, con el planteamiento del DB, porque el proyecto adquiriría una gran complejidad que no reflejaba mi intención en este trabajo: crear una web sencilla y práctica que pudieran usar dos tipos de usuarios diferentes, aprendiendo por el camino. Así, podría enfocarme en perfeccionar sus funcionalidades.

REFERENCIAS:

Ref

Serial in Postgresql

<https://www.postgresql.org/docs/current/datatype-numeric.html#DATATYPE-SERIAL>

Marta Villanueva Muñoz

Ref

How to make a diagram

<https://miro.com/es/diagrama/que-es-diagrama-entidad-relacion/>

Marta Villanueva Muñoz

Paso 2

24.06.2023

Una vez fijado el nuevo rumbo, volví a plantear el stack, las funciones de usuario, la estructura del DB y los casos de uso.

Una vez inicializado el proyecto tanto en Node como en React y con el DB a punto, empecé a plantear el **Router** con React.

REFERENCIAS:



createBrowserRouter v6.14.0

↗ reactrouter.com

PSQL

PATIENT

Read/
Write

DOCTOR

Read/
Write

Paso

2

24.06.2023

RELACIONES FINALES ENTRE LAS TABLAS DEL DB

TO ORDER (paciente->cita)

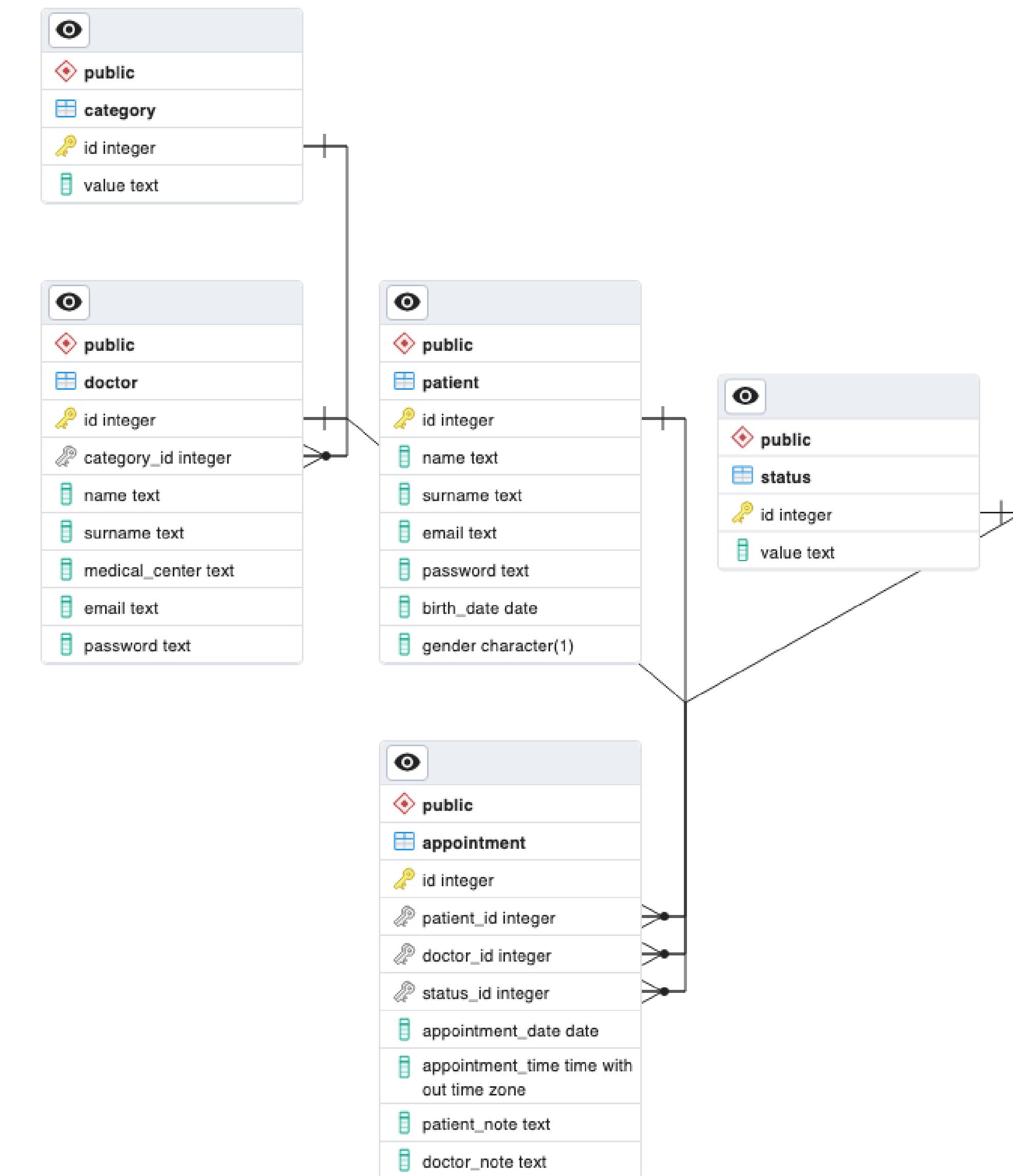
TO CONFIRM (doctor->cita)

TO BE (status<-cita)

TO BELONG (doctor-> categoría)

Son las bases de las relaciones en la estructura de datos de Medup, con la **cita** como item fundamental.

Para poblar las tablas de usuarios y pacientes hice uso de la inteligencia artificial con ChatGPT, lo que me permitió ahorrar tiempo y dedicar más esfuerzo a las funciones



Paso 3

25-26.06.2023

FORMULARIOS

Caso de uso: solicitud de cita, algo que solo podrá hacer el paciente.

Los formularios han sido una de las tareas más complicadas y satisfactorias, dada la necesidad de conectar la llamada al backend con el frontend. Para ello he utilizado el método **fetch()** con promesas, gestionándolas con **await**.

Esta fase contenía la funcionalidad principal del proyecto, pero surgieron retos que solucioné instalando la librería **cors**, que aparecía continuamente como error, bloqueando en los principales browsers peticiones http tanto request como response del mismo dominio.

Fijé como content-type **application/json** porque iba a trabajar con objetos json.

Por último, para solventar problemas de navegación, utilicé el hook **useNavigate** de React, que posibilitaría la navegación dinámica en la web, pasando datos.

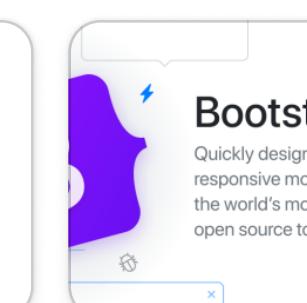
REFERENCIAS:



Alerts

Provide contextual feedback messages for typical...

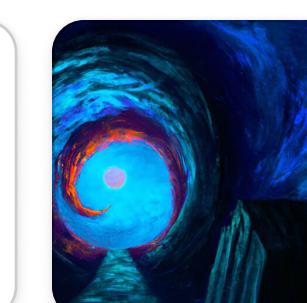
 getbootstrap.com



Layout

Give your forms some structure—from inline to hor...

 getbootstrap.com



How to Redirect to URL in ReactJS

Learn to redirect and navigate to external URLs in...

 codefrontend.com

Task:

- Form interface
- Send form

Problems solved:

- cors: with npm, check on package.json
- content-type:application/json
- redirect using useNavigate

Paso 4

27.06.2023

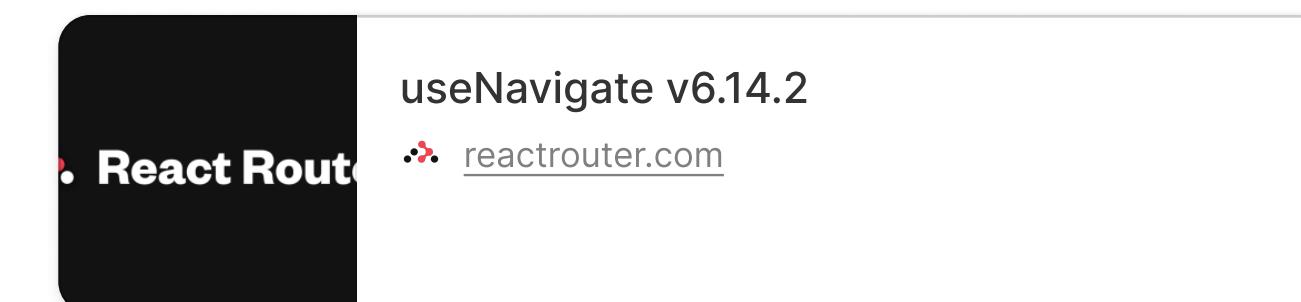
NAVEGACIÓN

Para continuar con el user journey, el siguiente paso era posibilitar la **búsqueda de doctores**. Y, además, poder acceder a la **página de reserva** clicando sobre un doctor concreto. Así, la categoría y el nombre seleccionados aparecerán como datos ya registrados en el formulario.

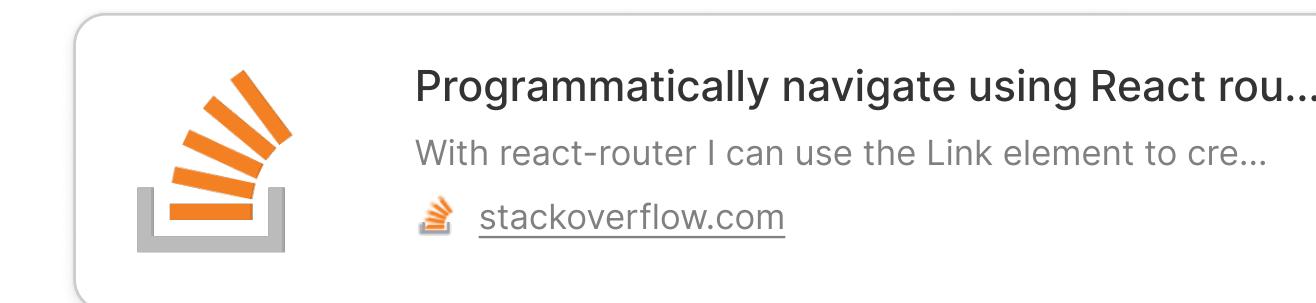
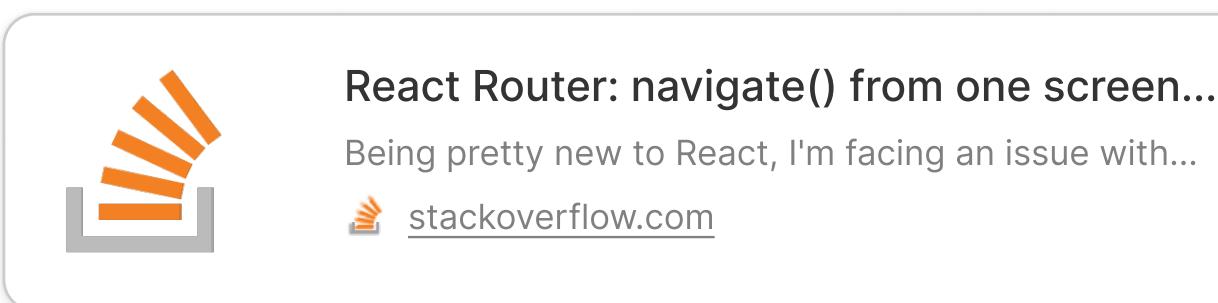
Aquí, de nuevo utilicé **useNavigate** y **useLocation** para navegar de una ruta a otra enviando objetos de estado. Ha sido especialmente útil en la página de reserva de cita.

He utilizado el componente nativo **NavLink** para sustituir anchor, para hacer que useNavigate y useLocation comprendieran las rutas en un contexto de navegación 100% React, en lugar de HTML.

En simultáneo, completé tareas de diseño.



REFERENCIAS:



Paso 4

27.06.2023

CAMBIO DE ESTADO

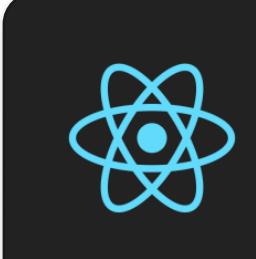
Un hook fundamental en el desarrollo del proyecto han sido los hooks **useState**, con el cual he registrado variables con un valor inicial que cambiaban dependiendo de la interacción con la webapp, así como **useEffect**, para aplicar efectos cuando esto sucedía.

Desde el propio inicio de sesión, hasta el saludo en la Home, la obtención de citas o la propia navBar. En este último caso lo empleé para poder cerrar el menú de navegación en mobile, como última solución después de probar varios métodos.

En esta fase creé la mayor parte de los componentes fundamentales, utilizando en la mayoría de los casos el standard de definición de **funciones como constantes**.

Ej.: `const exampleFunction = () => {}` en lugar de `function exampleFunction(){}`

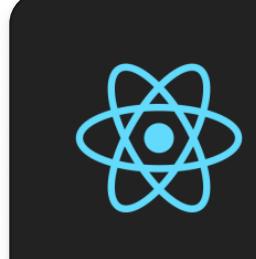
REFERENCIAS:



Using the State Hook – React

A JavaScript library for building user interfaces

legacy.reactjs.org



Using the Effect Hook – React

A JavaScript library for building user interfaces

legacy.reactjs.org



The useState set method is not reflecting a...

I am trying to learn hooks and the useState...

stackoverflow.com

Paso

5

28.06.2023

CONTEXTO

Aunque en un inicio la gestión de estos dos tipos de usuario (doctor y paciente) se realizaba pasando el booleano `isDoctor` en cada componente que fuera necesario, decidí cambiarlo utilizando el hook `useContext` de React.

Para conseguirlo, tuve que crear un contexto utilizando `createContext`. Posteriormente, encapsular toda mi app dentro de él y pasándole como valor la variable `loggedUser`. Esta, a su vez, llega desde el backend con el endpoint `/getUserSession` y devuelve el booleano `isDoctor` y el id del usuario.

Sobre el id del usuario en backend: para comprobar que la conexión con el `login` es satisfactoria tanto para paciente como para doctor, incluí un `console.log` con el id del usuario que accede (dinámico). De forma adicional, he incluido alertas que aparecen si los datos introducidos en esta fase no son correctos.

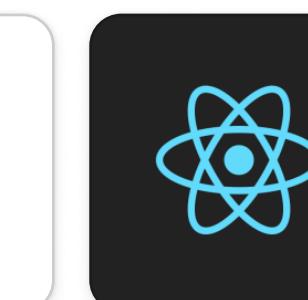
REFERENCIAS:



Alerts

Provide contextual feedback messages for typical...

 getbootstrap.com



Context – React

A JavaScript library for building user interfaces

 legacy.reactjs.org



How to Redirect to URL in ReactJS

Learn to redirect and navigate to external URLs in...

 codefrontend.com

Pasos 5-6

28-30.06.2023

SESIONES

Este paso se puede considerar una prolongación del anterior. La diferencia es que encontré más dificultades en el registro de la sesión. Finalmente, lo resolví con **cookie-parser** y un middleware en index.js añadiendo la propiedad **cookie** en el objeto session, conteniendo al mismo tiempo la propiedad secure con valor false.

Task:

Implement sessions

Log in interface

Unable comments on Appointment page

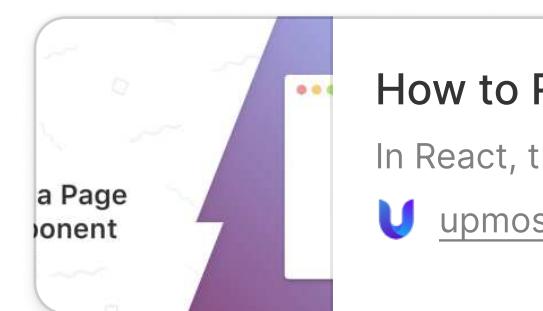
Doctors interface, allow and not allow

Problems solved:

Sessions.

Understanding the difference between components and index permissions

REFERENCIAS:



How to Refresh a Page or Component in R...

In React, there are two ways to refresh a page: upd...

upmostly.com



401 Unauthorized - HTTP | MDN

The HyperText Transfer Protocol (HTTP) 401 Unau...

developer.mozilla.org

Redirects in React Router

If you're a developer working on any modern web application, you're prob...

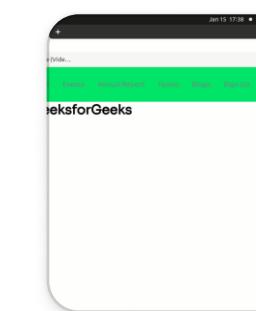
stackabuse.com



Session Management in Node.js using Exp...

This tutorial will help the reader develop a session...

section.io



Create a Responsive Navbar using...

A Computer Science portal for geeks. It contains w...

geeksforgeeks.org

Create 404 page in react

To create a 404 page in React using React Router, Create a...

datainfinities.com

Pasos

6-

30.06.2023 -
22.07.2023

INTERFAZ

Con la base construida y funcionando, he modificado el **diseño** inicial para hacerlo más usable y atractivo. Es por eso que durante las últimas semanas el desarrollo se ha centrado en detalles y solución de errores, como por ejemplo el envío del formulario en Login. En esta ocasión, al haber cambiado los tabs por links en el diseño, ariaSelected dejaba de funcionar y, por tanto, no dejaba marcada la opción. En consecuencia, no llegaba a registrar si era doctor o paciente, la base del proyecto.

El resto de modificaciones han ido dirigidas a la detección de errores, validaciones, alertas y mejoras de la interfaz, teniendo en cuenta el diseño responsive. Hacia el final, pude incluir una página 404 para añadir algo más de personalización a la página.

En cuanto a las imágenes, interpreté por las indicaciones del proyecto que debían ser imágenes y estar siempre almacenadas en el fichero, por lo que incluso los svg están incluidos como img.

ANOTACIONES

Como anotación, he utilizado para todo este tipo de detalles tanto **condicionales** if/else como switch/case y ternarios, siendo estos últimos de mi preferencia por su inclusión en el propio componente sin necesidad de crear un función al margen o incluir un gran bloque de código que afecte a la legibilidad.

En este periodo de tiempo me dediqué a estudiar para el examen, practicando con ejercicios hechos en clase y mi propio proyecto, dejando el diseño para momentos más puntuales.

Por último, aunque esta webapp podría incorporar muchas más funcionalidades, agradezco haber tenido que completar esta tarea, porque me ha ayudado a entender una gran cantidad de conceptos que antes no comprendía.

Indicaciones

Links del proyecto

PASOS PARA LA CORRECCIÓN

Leer README (fichero adjunto en la carpeta principal del proyecto).

O a través de GitHub

Iniciar sesión en cualquiera de los usuarios, ya que todos los datos de acceso están en el DB. Dadas las indicaciones por parte de la escuela, adjunto las **credenciales de tres usuarios:**

USER 1

- liam.smith@patient.com
- password1

USER 2 (aún sin citas, para comprobar que se pueden crear sin problemas)

- emma.johnson@patient.com
- password2

USER 3

- hans.schmidt@doctor.com
- password1

Repositorio en GitHub

[Visitar](#)

Figjam

[Visitar](#)

¡¡Gracias!!

