Indice

Que es duacenter?

De manera resumida duacenter en una aplicación hecha para los empleados de duacode para que puedan obtener y manejar información interna, del estilo de los empleados que hay tanto sus nombres como donde se encuentran entre otra gran variedad de datos, tambien se podran ver un mapa con las salas, su ocupación, un organigrama de la empresa un apartado para ver el resumen de la empresa otro para eventos, noticias comunicados, clientes que actuales y pasados un calendario con las reuniones y eventos un apartado de protocolos con manuales en pdf.

Motivos para su implementación

Centralización de la Información: Facilita a los empleados acceder a toda la información interna en un solo lugar, reduciendo tiempos de búsqueda y aumentando la eficiencia.

Mejor Comunicación Interna: Permite compartir noticias, eventos, comunicados, y mantener a todos actualizados de manera inmediata.

Organización y Planificación: El calendario integrado ayuda a gestionar reuniones y eventos, optimizando la coordinación y evitando solapamientos.

Gestión de Recursos: Con el mapa de salas y su ocupación, los empleados pueden planificar mejor reuniones y aprovechar espacios disponibles.

Transparencia y Conexión: Mostrar el organigrama y datos de empleados fomenta la colaboración, facilita identificar roles y fomenta un ambiente de equipo.

Acceso a Protocolos: Los manuales en PDF aseguran que todos los empleados puedan consultar procedimientos de manera rápida y fácil, promoviendo el cumplimiento de normas.

Historial de Clientes y Proyectos: La sección de clientes y proyectos pasados y actuales muestra el alcance de la empresa y fortalece la identidad corporativa.

Fomento de la Cultura Empresarial: El resumen de la empresa y los apartados dedicados refuerzan el sentido de pertenencia y orgullo en la organización.

Documentacion sobre el codigo

Inicio (Inicio.js)

Descripción General
El componente Inicio representa la página principal de la aplicación de comunicación interna. Ofrece una vista inicial con enlaces a las diferentes secciones de la aplicación, cada una representada como una tarjeta interactiva.

Dependencias Importadas
Cabecera:
Componente que muestra la barra de navegación superior. Se pasa la prop activePage para destacar la página actual.
../css/inicio.css:
Archivo CSS que define los estilos del componente.
Link (de react-router-dom):
Componente para la navegación interna de la aplicación.
FontAwesomeIcon:
Iconos utilizados para representar visualmente las secciones. Se importan de la biblioteca @fortawesome/react-fontawesome.
Iconos específicos:
faUser: Icono para la sección de empleados.
Info: Icono para la información de la empresa.
faMap: Icono para el mapa de salas.
MapPin: Icono para la disponibilidad de salas.
fa Calendar Days: Icono para el calendario informativo.
Sitemap: Icono para el organigrama.
Fa File Pdf: Icono para los protocolos.
faJar: Icono para la configuración de sede.
Estructura del Componente
Contenedor Principal ( ):

Incluye el componente Cabecera.
Organiza el contenido principal con un contenedor adicional (inició-container).
Encabezado (<header className="inicio-header">):

Muestra un título de bienvenida y una breve descripción de la aplicación.
Secciones ( ):

Cada sección está representada por un componente Link con la clase section-card.
Contenido de cada tarjeta:
Título: Incluye un icono y el nombre de la sección.
Descripción: Breve explicación de la funcionalidad de la sección.

Secciones Disponibles:

Empleados: Información detallada sobre los empleados.
Información de la Empresa: Noticias y proyectos.
Mapa Salas: Visualización de salas según la sede seleccionada.
Salas y disponibilidad: Estado de ocupación de las salas.
Calendario Informativo: Eventos y días relevantes.
Organigrama: Estructura organizativa de la empresa.
Protocolos: Acceso a manuales en PDF.
Configuración de Sede: Cambio de sede para visualizar información específica.
Props Utilizadas
activePage (en Cabecera):
Tipo: string.
Descripción: Define qué página está activa para resaltar en la barra de navegación.
Valor en este componente: "inicio".

Cabecera (Cabecera.js)

Descripción General
El componente Cabecera representa la barra de navegación superior de la aplicación. Incluye el logotipo de la empresa, un menú principal con enlaces a las diferentes secciones, y un submenú desplegable para las opciones relacionadas con la empresa.

Dependencias Importadas
React y hooks (useState, useEffect, useRef):

Sus State: Manejo del estado del submenú.
use Effect: Detectar clics fuera del menú para cerrarlo automáticamente.
useR: Referencia al submenú para manejar interacciones fuera de este.
react-router-dom:

Link: Para la navegación interna entre secciones de la aplicación.
FontAwesomeIcon:

Utilizado para el icono de hamburguesa en el menú de configuración (faBars).
../css/cabecera.css:

Archivo de estilos CSS para personalizar el diseño de la cabecera.
Logotipo (../imagenes/logo-duacode-negro.svg):

Imagen del logotipo de la empresa que se muestra en la parte superior izquierda.
Estructura del Componente
Encabezado Principal (<header className="header">):

Contiene dos secciones principales: la parte superior con el logotipo e icono de menú, y el menú de navegación.
Parte Superior ( ):

Logotipo: Enlaza a la página de inicio (/).
Icono de Configuración: Enlace al apartado de ajustes (/ajustes).
Menú de Navegación (<nav className="nav-menu">):

Contiene elementos del menú principal, cada uno vinculado a una sección específica de la aplicación.
Submenú:
Asociado a la sección "Empresa".
Incluye enlaces adicionales para:
Información de la empresa.
Eventos.
Noticias y comunicados.
Proyectos y clientes.
El submenú se muestra u oculta según el estado Submenú Open.
Props
activePage:
Tipo: string.
Descripción: Identifica la página activa para aplicar estilos específicos, como negrita en el menú correspondiente.
Ejemplo de valores:
"empleados"
"empresa"
"calendario"
Estado Interno
isSubmenuOpen:
Tipo: boolean.
Descripción: Controla si el submenú de la sección "Empresa" está abierto (true) o cerrado (false).
Eventos y Lógica
Cierre Automático del Submenú:

Usa Effect para registrar un evento mousedown que detecta clics fuera del submenú.
Si se hace clic fuera del menú, se actualiza el estado Submenú Open a false.
Apertura/Cierre del Submenú:

Al hacer clic en la sección "Empresa", se alterna el estado is Submenú Open.

Empleados (Empleado.js, EmpleadoEspecifico.js, menuEmpleados, organigrama.js)

1. EmpleadoEspecifico.js
Este componente muestra los detalles de un empleado específico, incluyendo su información personal, supervisor, y otros datos relevantes.

Funciones principales:
petición empleado: Hace una solicitud GET al backend para obtener los datos del empleado actual utilizando su ID (X) desde los parámetros de la URL.
Obtener Nombre Supervisor: Si el empleado tiene un supervisor, realiza otra solicitud para obtener el nombre del supervisor por su ID.
use Effect: Llama a petición del empleado al cargar el componente o cuando cambia el parámetro X.
Estado:
employee: Almacena los datos del empleado.
supervisor Name: Contiene el nombre del supervisor.
Error: Indica si hubo un error al recuperar datos.
isLoading: Maneja el estado de carga.
Renderizado:
Si hay un error, muestra un mensaje informando del problema.
Utiliza el componente With Loader para mostrar un indicador de carga mientras se obtienen los datos.
Muestra los detalles del empleado y un botón para regresar a la lista de empleados.
2. Empleados.js
Este componente muestra una lista de todos los empleados, con opciones de búsqueda y filtrado.

Funciones principales:
petición empleados: Solicita al backend todos los empleados y los almacena en dos estados (data Employee Filtrada y data Employed).
handleChange: Maneja los cambios en el campo de búsqueda.
filtrar: Filtrar los empleados basándose en el nombre ingresado en el campo de búsqueda.
Estado:
Data Employee Filtrada: Contiene todos los empleados obtenidos del servidor.
Data Employee: Lista filtrada de empleados.
búsqueda: Contiene el texto ingresado en el campo de búsqueda.
Error y isLoading: Manejan errores y estados de carga.
Renderizado:
Si ocurre un error, muestra un mensaje informando del problema.
Utiliza el componente With Loader para gestionar el estado de carga.
Presenta una lista de tarjetas con la información básica de cada empleado. Cada tarjeta enlaza al componente Empleado Específico.
3. menuEmpleados.js

Este componente genera un menú de navegación para las secciones relacionadas con empleados.

Características principales:
Contiene enlaces a las secciones Información Empleados y Organigrama.
Proporciona estilos personalizados definidos en empleados.
Uso:
Este componente se importa en las páginas principales relacionadas con empleados (Empleados.js, Organigrama.js).

4. Organigrama.js
Este componente construye y muestra un organigrama interactivo de la empresa.

Funciones principales:
render Employees: Genera nodos para empleados individuales.
Prender Subordinates: Construye de manera recursiva los nodos de subdepartamentos y empleados.
use Effect: Carga los datos del organigrama desde un archivo JSON cuando se monta el componente.
Estado:
data: Contiene los datos del organigrama obtenidos del archivo JSON.
isLoading: Maneja el estado de carga.
Renderizado:
Utiliza react-organizational-chart para mostrar una estructura jerárquica.
Diseña tarjetas de miembros con estilos personalizados.
Incluye el componente MenuEmpleados para la navegación.

5. Componente Reutilizable: WithLoader.js y Loaders.js
Un componente HOC (High Order Componente) que gestiona los estados de carga para otros componentes.

Props:
isLoading: Booleano que indica si se deben mostrar los datos o el indicador de carga.
Uso:
Envuelve las secciones principales de Empleados.js, EmpleadoEspecifico.js y Organigrama.js para mostrar un spinner mientras se cargan los datos.

Empresa

Menú e Información

Documentación: Componente Información Empresa

Descripción General

El componente Información Empresa forma parte de la sección de empresa de la aplicación de gestión. Está diseñado para proporcionar una visión detallada de la compañía, incluyendo su misión, presencia en distintas localizaciones y patrocinio deportivo.

Funcionalidades Principales

Cabecera Activa:

Incluye el componente Cabecera, que destaca la página activa "empresa" para facilitar la navegación.

Menú Lateral Dinámico:

Utiliza el componente Menú Empresa con la opción activa "infoEmpresa".

Proporciona navegación a otras secciones relacionadas como eventos, noticias, comunicados y proyectos.

Información sobre Duacode:

Contenido detallado sobre las actividades principales de la empresa:

Desarrollo de software y soluciones tecnológicas personalizadas.

Enfoque en eficiencia interna y adaptabilidad a las necesidades del cliente.

Presencia Geográfica:

Destaca la ubicación estratégica de las oficinas principales en Madrid y en otras ciudades para ofrecer colaboración nacional e internacional.

Compromiso Deportivo:

Muestra la participación de la empresa en patrocinios deportivos, reforzando su identidad empresarial y conexión con la comunidad.

Imagen Representativa:

Incluye una imagen destacada (decoders-desk-1.webp) que complementa la sección de información.

Estructura Visual

División de contenido:

menú-info Importante: Divide la sección en un menú lateral y un área principal para la información.

información-lateralMente: Presenta un texto introductorio sobre Duacode.

imagen infoEmpresa: Muestra una imagen relacionada con la empresa.

información Extra: Proporciona detalles adicionales sobre la presencia geográfica y compromiso deportivo.

Archivos CSS Relacionados

información Empresa.css: Define el diseño visual de los elementos.

Recursos Adicionales

Imagen:

Archivo decoders-desk-1.webp utilizado para complementar la presentación visual.

Documentación: Componente Menú Empresa

Descripción General

El componente Menú Empresa representa un menú lateral interactivo utilizado en la sección de "empresa" para facilitar la navegación entre subsecciones.

Funcionalidades Principales

Navegación Dinámica:

Lista de enlaces a las diferentes subsecciones:

Información sobre la empresa.

Eventos.

Noticias y comunicados.

Proyectos y clientes.

Estado Activo:

Destaca la sección activa mediante una clase CSS dinámica (bold). Esto mejora la experiencia del usuario al indicar visualmente la página seleccionada.

Parámetros

EmpresaMenuActivo:

Propiedad que indica cuál es la sección activa.

Archivos CSS Relacionados

menuEmpresa.css:

Define el estilo del menú lateral, incluyendo el estado destacado para el elemento activo.

Flujo de Trabajo

Al hacer clic en un enlace, se redirige al usuario a la página correspondiente utilizando react-router-dom.

Eventos

El componente Eventos es parte de la sección de Empresa y se encarga de mostrar información sobre los próximos eventos relevantes para Duacode. Su objetivo principal es recuperar datos de eventos desde una API externa y organizarlos de manera clara para los usuarios.

Funcionalidad
Carga de datos de eventos desde la API:

Se conecta al endpoint https://idkmen.pythonanywhere.com/event/ para recuperar información sobre los eventos.
Implementa un sistema de manejo de errores y un indicador de carga para ofrecer una experiencia de usuario más fluida.
Filtrado y organización de eventos:

Identifica los eventos futuros en función de su fecha de inicio (date_start).
Ordena los eventos por fecha para mostrar el más cercano primero.
Divide los eventos en:
Próximo evento: El evento más cercano a la fecha actual.
Eventos restantes: Una lista con los eventos que ocurrirán después del próximo evento.
Interacción con el usuario:

Muestra un mensaje en caso de que no existan eventos futuros disponibles.
Proporciona detalles de cada evento, como título, fecha y contenido.
Estructura del componente
Encabezado y menú lateral:

Incluye un encabezado con el componente Cabecera, destacando la sección activa.
Usa el componente Menú Empresa para navegar entre las distintas secciones de Empresa.
Próximo evento:

Destaca el evento más cercano en un contenedor exclusivo.
Presenta su título, fecha y descripción.
Eventos futuros restantes:

Muestra el resto de los eventos en un formato ordenado.
Cada evento incluye su título, fecha y contenido.
Indicador de estado:

Un mensaje indica si:
No hay eventos futuros.
Ocurrió un error al conectar con la API.
Consideraciones técnicas
Estado del componente:

data Eventos: Almacena los datos recuperados de la API.
Error: Indica si ocurrió un error durante la recuperación de datos.
isLoading: Determina si los datos aún están cargándose.
Manejo de errores y carga:

Muestra un mensaje de error si la API no está disponible o hay problemas de conexión.
Utiliza el componente With Loader para mostrar un indicador de carga mientras se obtienen los datos.
Filtrado y ordenamiento:

Utiliza Array.filter() para filtrar eventos futuros basándose en su fecha de inicio.
Aplica Array.sort() para ordenar los eventos por fecha, de menor a mayor.
Diseño y estilo
Clase principal: información Empleados, que define el diseño base de la página.
Próximo evento: Diseñado con la clase prox Evento para destacar visualmente el evento más cercano.
Lista de eventos futuros: Usa ordenar Eventos para un diseño uniforme, y evento Futuro para estilizar cada evento.
Mensajes y logs de depuración
Mensajes en consola:

console.log se utiliza para:
Verificar la estructura de los datos recibidos de la API.
Comprobar el proceso de filtrado y la interpretación de las fechas.
Mensajes de usuario:

Cuando no hay eventos futuros: "No hay eventos futuros disponibles."
Cuando no hay más eventos futuros: "No hay más eventos futuros."
En caso de error: "Puede que el servidor esté apagado o exista algún problema con el."

Noticias y comunicados

Componente Comunicados
Este componente muestra una lista de comunicados filtrados por aquellos que son importantes. Realiza una solicitud a una API externa para obtener los datos y los muestra de forma interactiva.

Características principales:

Carga de datos:

Utiliza Axios para obtener los datos de los comunicados desde una API externa.
El estado de la carga (isLoading) se maneja con un componente de orden superior (With Loader).
Se maneja un estado de error (has Error) para mostrar un mensaje si ocurre un problema con la solicitud.
Filtrado de datos:

Los comunicados se filtran para mostrar solo aquellos con importante comunicación === true.
Renderizado:

Los comunicados importantes se muestran como tarjetas, donde se incluye el título, contenido y la imagen (si está disponible).
Cada comunicado es un enlace a una página más detallada.
Estructura y navegación:

Incluye un enlace para volver a la página anterior (Volver).
Se utilizan componentes reutilizables como Cabecera para el encabezado y Menú Empresa para el menú lateral.
Componente Noticias
Este componente muestra una lista de noticias filtradas por aquellas que no son importantes. Similar a Comunicados, este componente realiza una solicitud a la misma API para obtener los datos y filtra aquellos con importante comunicación === false.

Características principales:

Carga de datos:

Utiliza Axios para obtener los datos de las noticias desde la API externa.
El estado de la carga se maneja con el componente With Loader.
Se maneja un estado de error para mostrar un mensaje si hay problemas con la API.
Filtrado de datos:

Las noticias se filtran para mostrar solo aquellas con importante comunicación === false.
Renderizado:

Las noticias se muestran en tarjetas, con el título, contenido y una imagen si está disponible.
Cada noticia es un enlace que lleva a una página más detallada de la noticia.
Estructura y navegación:

Incluye un botón de navegación para regresar a la página de noticias y comunicados (Volver).
Componente Noticias Comunicados
Este componente muestra un resumen de las noticias y comunicados importantes. Realiza una solicitud a la misma API externa para obtener los datos y muestra los artículos destacados en una vista simplificada.

Características principales:

Carga de datos:

Realiza una solicitud a la API para obtener todas las noticias y comunicados.
Al igual que en los otros componentes, utiliza Axios para la solicitud de datos y el estado isLoading para manejar la carga.
Filtrado de datos:

Muestra los comunicados importantes y las noticias no importantes de forma separada.
Usa los métodos find y filter para obtener la información específica de cada tipo.
Renderizado:

El componente muestra enlaces a las páginas de noticias y comunicados.
Cada enlace lleva a una página con más detalles sobre la noticia o el comunicado.
Estructura y navegación:

Incluye un menú lateral para acceder a las noticias y comunicados y enlaces para ver más detalles.
Usando With Loader, se gestiona la carga de los datos.
Componente Noticias Específicas
Este componente muestra la información detallada de una noticia específica. Obtiene el ID de la noticia desde la URL (mediante seParamos) y realiza una solicitud a la API para recuperar los datos de esa noticia.

Características principales:

Obtención del parámetro de URL:

Utiliza separamos de React Router para obtener el newsId y cargar los datos específicos de esa noticia.
Carga de datos:

Realiza una solicitud a la API para obtener los detalles de una noticia usando Axios.
Maneja el estado de carga y el manejo de errores utilizando los estados isLoading y haz Error.
Renderizado:

Si la noticia se encuentra, muestra su título, fecha, contenido y una imagen si está disponible.
Si no se encuentra la noticia, se maneja el caso con un mensaje de error o cargando.
Estructura y navegación:

Incluye un botón para volver a la lista de noticias.
Componente MenuEmpresa
Este es un componente de navegación lateral que se utiliza para mostrar el menú de la sección de empresa, permitiendo a los usuarios navegar entre las noticias, comunicados y otras páginas relevantes.

Características principales:

Enlaces de navegación:
Contiene enlaces a las páginas de información de empleados, organigrama, noticias, etc.
Cada enlace redirige a la ruta correspondiente dentro de la aplicación.

Proyectos y clientes

Componente ProyectosClientes
El componente Proyectos Clientes es responsable de mostrar una lista de proyectos actuales y pasados asociados a los clientes de la empresa. Esta sección permite a los usuarios navegar a proyectos específicos a través de un enlace, y visualizar detalles más completos de cada proyecto.

Estructura y Funcionalidad
Importaciones:

Cabecera: Componente que se muestra en la parte superior de la página para la navegación.
Menú Empresa: Muestra un menú lateral con opciones relacionadas con la empresa.
With Loader: Componente de carga para indicar cuando los datos están siendo cargados.
axios: Librería utilizada para realizar solicitudes HTTP.
Link: Componente de React Router utilizado para navegar entre páginas sin recargar el navegador.
Estados:

Data Clientes: Almacena los datos de los clientes obtenidos de la API.
Proyecto Inicial: Almacena los proyectos obtenidos de la API.
Error: Indicador de si hubo un error al obtener los datos.
isLoading: Indicador de si los datos están siendo cargados.
Métodos:

Petición proyectos Inicial: Realiza una solicitud GET a la API para obtener los proyectos. Los proyectos se almacenan en el estado data Proyecto Inicial.
petición clientes: Realiza una solicitud GET a la API para obtener la lista de clientes, que se almacena en el estado data Clientes.
Uso de useEffect:

Se ejecuta al montar el componente para realizar las solicitudes de datos (petición proyectos Inicial y petición clientes).
Filtrado de Proyectos:

Se filtran los proyectos para separar los proyectos actuales y proyectos pasados basándose en el campo active:
Proyectos Actuales: Proyectos donde active es true.
proyectosPasados: Proyectos donde active es false.
Renderizado:

Si hay un error (haz Error), se muestra un mensaje indicando que puede haber problemas con el servidor.
Si los datos están siendo cargados (isLoading), se muestra un cargador.

Se muestran los proyectos actuales y pasados con un enlace para acceder a los detalles completos de cada uno.
Componente Proyectos Actuales
Este componente muestra solo los proyectos actuales, proporcionando enlaces a sus detalles específicos. Cada tarjeta de proyecto incluye el título y los objetivos del proyecto.

Filtrado:

Se filtran los proyectos para mostrar solo aquellos cuyo campo activo es true.
Renderizado:

Cada proyecto es representado como una tarjeta con el título, objetivos y un enlace para ver más detalles.
Componente Proyectos Antiguos
Este componente es similar a Proyectos Actuales, pero se enfoca en mostrar los proyectos pasados. Utiliza el mismo proceso de filtrado, pero para proyectos donde active es false.

Filtrado:

Se filtran los proyectos para mostrar solo aquellos cuyo campo activo es falso.
Renderizado:

Al igual que en Proyectos Actuales, cada proyecto pasado es representado como una tarjeta con información básica y un enlace a más detalles.
Componente Proyecto Actual Específico y Proyecto Antiguo Específico
Estos componentes se encargan de mostrar los detalles específicos de un proyecto cuando el usuario hace clic en él. La solicitud GET para obtener los detalles se realiza utilizando axios y se muestra la información del proyecto, como el título y los objetivos. Además, se proporciona un botón para volver a la lista de proyectos.

Petición de datos:
Al cargar el componente, se realiza una solicitud GET a la API para obtener los detalles del proyecto seleccionado usando el ID del proyecto extraído de la URL.
Renderizado:
Si los datos no están disponibles, se muestra un mensaje de carga o error. Si los datos están disponibles, se muestran los detalles completos del proyecto.

Distribución e Información Salas

DistribucionInfo.js
Este componente gestiona la visualización del mapa de distribución de las salas de la oficina seleccionada. Además, permite seleccionar una sala y ver sus detalles.

Estado y Props:
rooms: Almacena la lista de salas obtenidas del backend.
selected Room Id: ID de la sala seleccionada.
Seleccionado Office: Oficina seleccionada (obtenida del contexto use Office).
isLoading: Indica si los datos están siendo cargados.
Funciones:
petición habitaciones: Realiza una petición HTTP para obtener los datos de las salas desde un backend.
handleRoomSelect: Cambia la sala seleccionada al hacer clic en una sala o un botón.
Filtered Rooms: Filtra las habitaciones según la oficina seleccionada (Galicia o Valencia).
use Effect: Llama a petición habitaciones al montar el componente para obtener los datos de las habitaciones.
Comportamiento:
Muestra un mapa de distribución con las habitaciones y permite seleccionar una.
Si una sala está ocupada, se indica visualmente.
Permite visualizar la información de la sala seleccionada, como nombre, capacidad y ocupación.
Componentes utilizados:
Cabecera: Muestra la cabecera de la página.
MenuDistribucionInfo: Menú de navegación con enlaces a diferentes vistas de distribución de salas.
With Loader: Componente que muestra un indicador de carga mientras los datos se están recuperando.
InfoSalas.js
Este componente muestra una lista de las salas disponibles en la oficina seleccionada, junto con su capacidad y estado de ocupación.

Estado y Props:
rooms: Lista de habitaciones obtenidas del backend.
selected Room Id: ID de la sala seleccionada.
Seleccionado Office: Oficina seleccionada (obtenida del contexto use Office).
isLoading: Indica si los datos están siendo cargados.
Funciones:
petición de habitaciones: Realiza una solicitud HTTP para obtener las habitaciones desde el backend.
handle Room Select: Permite seleccionar una sala al hacer clic sobre un elemento de la lista.
Comportamiento:
Filtra las salas según la oficina seleccionada (Galicia o Valencia).

Muestra una lista con los nombres de las salas, su capacidad y si están ocupadas o no.
Permite seleccionar una sala al hacer clic sobre un ítem de la lista.
Componentes utilizados:
Cabecera: Muestra la cabecera de la página.
Menu Distribución Info: Menú de navegación para seleccionar entre el mapa o la lista de información de salas.
With Loader: Muestra un indicador de carga mientras se obtienen los datos.
MenuDistribucionInfo.js
Este componente contiene el menú de navegación para la sección de distribución de salas.
Proporciona enlaces a dos vistas: el mapa de distribución y la lista de información de las salas.

Comportamiento:
Ofrece enlaces para navegar entre las dos vistas de distribución de salas: "Mapa" y "Información de las Salas".
Componentes utilizados:
Link: Utiliza el componente Link de react-router-dom para la navegación entre rutas.
Ajustes.js
Este componente permite a los usuarios cambiar la oficina seleccionada, mostrando un formulario con opciones para seleccionar entre Galicia y Valencia.

Estado y Props:
Selected Office: Estado obtenido del contexto use Office, que indica la oficina actualmente seleccionada.
setSelected Office: Función para actualizar el estado de la oficina seleccionada.
Funciones:
handleChange: Actualiza la oficina seleccionada cuando el usuario elige una opción en el formulario de radio buttons.
Comportamiento:
Permite al usuario seleccionar entre dos oficinas (Galicia o Valencia).
Actualiza el contexto global usando Office con la oficina seleccionada.
Componentes utilizados:
Link: Utiliza el componente Link de react-router-dom para la navegación a otras páginas.
useOffice: Hook personalizado que maneja el contexto de la oficina seleccionada.
Notas adicionales:
Los socios se utilizan en todos los componentes para realizar solicitudes HTTP al backend para obtener las habitaciones.
Estilos CSS: Se utilizan archivos CSS como distribución Info.css y ajustes.css para dar estilo a los componentes.
Contexto use Office: Todos los componentes relacionados con las oficinas usan el contexto use Office para obtener y modificar la oficina seleccionada globalmente, asegurando una experiencia coherente entre los diferentes componentes.

Calendario

Descripción General
El componente Calendario es responsable de mostrar un calendario interactivo con eventos de una API externa. El calendario utiliza el paquete @fullcalendar/react para renderizar los eventos de forma visual en un mes, semana o día, y permite realizar búsquedas por nombre de evento y activar/desactivar la visibilidad de los fines de semana.

Dependencias
React: Para la creación del componente y manejo de estado.
FullCalendar: Para mostrar y manejar el calendario.
Taxis: Para realizar peticiones HTTP.
FontAwesome: Para los iconos, en este caso el icono de búsqueda.
With Loader: Un componente auxiliar para mostrar un estado de carga mientras los datos son recuperados.
Estado del Componente
Calendar Filtrada: Almacena los eventos del calendario filtrados por la búsqueda.
Weekend Visible: Estado que controla si los fines de semana están visibles en el calendario.
Calendar: Almacena todos los eventos del calendario.
Error: Indica si ocurrió un error al cargar los datos.
búsqueda: Contiene el término de búsqueda introducido por el usuario.
isLoading: Indica si los datos están siendo cargados.
Funciones
formatDate(dateString):

Formatea las fechas en formato día/mes/año.
botonMostrarSemanas():

Cambia el estado de visibilidad de los fines de semana.
petición calendario():

Realiza una petición GET a la API (https://idkmen.pythonanywhere.com/event/) para obtener los eventos.
Los eventos se procesan y se almacenan en los estados dataCalendar y dataCalendarFiltrada.
useEffect():

Llama a petición calendario() cuando el componente se monta para obtener los eventos.
handleChange(e):

Maneja los cambios en el campo de búsqueda y filtra los eventos según el término de búsqueda.
filtrar(término Búsqueda):

Filtra los eventos en función del término de búsqueda y actualiza el estado dataCalendarFiltrada.
Renderizado Condicional
Error al cargar los datos: Si ocurre un error, se muestra un mensaje indicando que el servidor podría estar apagado o haya un problema con él.
Eventos del Calendario: Se dividen en dos partes:
Primer Evento: El primer evento se muestra por separado con información detallada.
Resto de Eventos: Los eventos restantes se muestran en una lista, donde cada uno tiene información detallada como el título, ID, contenido, fecha de inicio y fin, y la sala.
Interfaz de Usuario
Calendario: Utiliza el componente FullCalendar con los siguientes plugins:
Day Grid Plugin: Para la vista del mes.
Time Grid Plugin: Para la vista de semana y día.
list Plugin: Para la vista de lista de eventos.
Barra de Búsqueda: Permite buscar eventos por nombre.
Checkbox de fines de semana: Permite activar o desactivar la visualización de los fines de semana en el calendario.
Loader: Se muestra un componente With Loader mientras los datos están siendo cargados.
Estilos
Los estilos para este componente están definidos en el archivo calendario.css. Se usan clases como:

.PaginaCalendario: Contenedor principal.
Contenido Calendario: Contenedor que organiza el calendario y la barra lateral.
Calendario: Estilo del calendario en sí.
.sidebar Calendario: Contenedor de la barra lateral con instrucciones y controles.
.restoS Eventos: Contenedor que muestra el resto de los eventos.

Acciones
Buscar eventos: Al escribir en la barra de búsqueda, los eventos que coincidan con el término se muestran en el calendario.
Activar/Desactivar fines de semana: El checkbox permite alternar la visibilidad de los fines de semana en el calendario.

Protocolos

1. Pdf Comp (Componente para mostrar y navegar por archivos PDF)
El componente PdfCamp se encarga de la visualización y navegación de archivos PDF. Permite cargar un archivo PDF de manera dinámica desde un servidor y visualizarlo, ya sea en modo página por página o todo el documento de una vez.

Propiedades:
id (propiedad obligatoria): Recibe el id del archivo PDF a cargar.
Estado del Componente:
numPages: Número total de páginas del documento PDF.
pageNumber: Página actual del PDF, iniciando en la primera.
mostrar: URL del archivo PDF que se está mostrando.
viewMode: Modo de visualización del documento. Puede ser 'page-by-page' o 'all-pages'.
isLoading: Estado de carga del archivo PDF.
Efectos (use Effect):
Se ejecuta cuando el componente se monta, recuperando el PDF correspondiente usando axios para hacer una solicitud HTTP.
Cuando el componente se desmonta, se revoca la URL del archivo PDF para liberar recursos.
Funciones:
onDocumentLoad Success: Maneja la carga exitosa del documento PDF, actualizando el número de páginas.
toggleViewMode: Alterna entre los modos de visualización 'page-by-page' y 'all-pages'.
goToNextPage: Cambia a la siguiente página del PDF.
goToPreviousPage: Cambia a la página anterior.
Renderizado:
Si el archivo PDF aún no ha sido cargado, se muestra un componente de carga (With Loader).
Si el PDF se carga correctamente, se renderiza utilizando el componente Document de react-pdf, mostrando las páginas según el modo seleccionado.
2. Protocolos (Componente para mostrar lista de protocolos y permitir ver archivos PDF)
El componente Protocolos es el encargado de listar los protocolos disponibles (en formato PDF) y permitir su visualización cuando el usuario selecciona uno.

Estado del Componente:
pdf Files: Lista de archivos PDF de los protocolos recuperados desde el servidor.
Seleccione Pdf: id del archivo PDF seleccionado para su visualización.
Error: Indica si hubo un error al cargar los datos de los protocolos.
isLoading: Estado de carga de los datos.
Efectos (use Effect):
Se ejecuta cuando el componente se monta, recuperando la lista de protocolos disponibles a través de una solicitud HTTP utilizando axios.
Funciones:
petición protocolos: Realiza la solicitud HTTP para obtener los archivos PDF desde el servidor y los almacena en el estado pdf Files.

Renderizado:
Si ocurre un error al cargar los protocolos, se muestra un mensaje de error.
Si los protocolos se cargan correctamente, se muestra una lista de los títulos de los protocolos disponibles con un botón que permite seleccionar uno. Cuando se selecciona un protocolo, se muestra su contenido utilizando el componente PdfCamp.
Si el archivo PDF seleccionado está en proceso de carga, se muestra un componente de carga (With Loader).
Elementos de la interfaz:
Cabecera: Muestra el encabezado de la página con el enlace a la sección activa.
Botones de visualización: Cada protocolo tiene un botón para mostrar el PDF correspondiente.
Vista del PDF: Al seleccionar un protocolo, se carga y se visualiza mediante el componente PdfCamp.
Dependencias utilizadas:
react-pdf: Biblioteca para renderizar archivos PDF dentro de los componentes React.
Baixos: Biblioteca para realizar solicitações HTTP.
With Loader: Componente utilizado para mostrar un indicador de carga mientras se están recuperando los datos o el archivo PDF.
FontAwesome Icon: Para mostrar el ícono de PDF en la interfaz.
Estilos:
Los estilos de los componentes están definidos en el archivo protocolos.css y otros archivos CSS importados, los cuales permiten configurar la disposición y el diseño de los elementos (como los botones, listas y visualización de PDFs).