Seminario de Lenguajes
Opción: PHP, React y API Rest
2024

React

React es una librería de JavaScript que se utiliza para construir interfaces de usuario. Toda aplicación web React se construye a través de componentes reutilizables que conforman partes de la interfaz de usuario — podemos tener un componente distinto para nuestra barra de navegación, otro para el pie de página, otro para el contenido principal, etc.

Un componente es una pieza de UI (siglas en inglés de interfaz de usuario) que tiene su propia lógica y apariencia. Un componente puede ser tan pequeño como un botón, o tan grande como toda una página.

Arquitectura base

Este es un ejemplo de organización de carpetas una vez creado el proyecto de React, pueden tomarlo como base, agregar o modificar cosas para el desarrollo de esta entrega

- **public/**: Contiene archivos estáticos que se sirven directamente, como index.html y el favicon.
- **src/**: Contiene todo el código fuente del proyecto.
    - **assets/**: Archivos estáticos como imágenes y hojas de estilo.
        - **images/**: Imágenes del proyecto.
        - **styles/**: Hojas de estilo CSS.
    - **components/**: Componentes reutilizables de la interfaz de usuario.
    - **pages/**: Componentes de página que representan vistas completas.
    - **utils/**: Funciones utilitarias y helpers.
    - **App.jsx**: Componente raíz de la aplicación.
    - **index.jsx**: Punto de entrada de la aplicación que renderiza el componente raíz en el DOM.
- **/**
    - **package.json**: Contiene las dependencias del proyecto y scripts de comandos.
    - **README.md**: Documentación del proyecto.

Segunda entrega

Esta entrega consistirá en el desarrollo de una aplicación React que consumirá los endpoints de la API creada en la entrega anterior. De ser necesario, se pueden crear nuevos endpoints si lo requieren.

1) Dentro de la carpeta *src* crear una carpeta llamada *components*

Esta carpeta contendrá todos los componentes reutilizables. Desde la cátedra, se les indica que deben definir los siguientes tres componentes, pero también se espera que piensen y definan otros componentes reutilizables que puedan mejorar la eficiencia y la calidad del proyecto.

a. **HeaderComponent**: el cual debe contener un logo que identifique su página (puede ser una imagen de internet o creada) y un título.
b. **FooterComponent** el cual debe contener los nombres de los integrantes del grupo y el año en curso.
c. **NavBarComponent:** el cual debe permitir navegar las diferentes páginas.

2) Todas la secciones (o páginas) deben contener los siguientes componentes:

a. Header
b. Footer
c. Navbar para navegar la página

3) Módulos a desarrollar de forma obligatoria:

a. Tipos de propiedad

Crear un archivo dentro de la carpeta ***pages/tipoPropiedad*** llamado ***TipoPropiedad*****Page**

Dicha página debe contener:

1. Listado de todos los tipos de propiedad
    a. nombre
    b. Cada ítem del listado debe tener las acciones para editar y eliminar

2. Un Botón para crear una nuevo tipo de propiedad. Crear un archivo dentro de ***pages/tipoPropiedad*** llamado **New*TipoPropiedad***. Dicha página debe contener un formulario con lo siguiente:
    a. Input para ingresar el **nombre**(se debe validar que el input no sea vacío antes enviarlo)
    b. Un botón para realizar el submit del formulario (Una vez enviados los datos del formulario al backend, se debe mostrar el mensaje devuelto por servicio)

3. Crear un archivo dentro de ***pages/tipoPropiedad*** llamado **Edit*TipoPropiedad*** que se carga al presionar editar:
    a. El formulario debe venir precargado con los valores del tipo de propiedad
    b. Se debe validar lo mismo que en el **New*TipoPropiedad***
    c. Se debe mostrar el mensaje devuelto por el endpoint (éxito o error)

4. Al presionar en la acción "**Eliminar**" del listado se debe borrar el item del listado. En caso que no se pueda eliminar, se debe mostrar un mensaje indicando lo ocurrido. **Se debe pedir confirmación antes de realizar la acción.**

b. Propiedades

Crear un archivo dentro de la carpeta ***pages/propiedad*** llamado **PropiedadPage**. Esta página deberá mostrarse como página principal (como **index.php** en la entrega uno).

Dicha página debe contener:

1. Listado de todos las propiedades

    a. Domicilio
    b. Localidad (nombre de localidad, no **localidad_id**)
    c. Tipo de propiedad (nombre del tipo, no **tipo_propiedad_id**)
    d. Fecha de inicio disponibilidad
    e. Cantidad de Huéspedes
    f. Valor noche
    g. Cada ítem del listado debe tener las acciones para ver detalle, editar y eliminar

2. En la sección del listado, habrá un formulario que permitirá filtrar las propiedades y modificar la información mostrada en el listado principal. Los campos del formulario serán los siguientes:
    a. Disponible (input de tipo checkbox)
    b. Localidad: (select)
    c. Fecha de inicio: (input de tipo date)
    d. Cantidad de huéspedes: (input de tipo number)

3. Crear un archivo dentro de ***pages/propiedad*** llamado **DetailPropiedad**. Dicha página deberá mostrar únicamente los datos de la propiedad seleccionada.

4. Un Botón para crear una nueva propiedad. Crear un archivo dentro de ***pages/propiedad*** llamado **NewPropiedad**. Dicha página debe contener un formulario con lo siguiente:

    a. Input para ingresar el **domicilio** (se debe validar que el input no sea vacío antes enviarlo)

b. Select que permite elegir una **localidad** (se debe validar que se seleccione algo antes de enviar el formulario)
c. Input para el **cantidad_habitaciones**(se debe validar que el input que sea un número si se ingresó algo)
d. Input para el **cantidad_banios**(se debe validar que el input que sea un número si se ingresó algo)
e. Input checkbox para el **cochera**
f. Input para el **cantidad_huespedes**(se debe validar que el input no sea vacío antes enviarlo y que sea un número)
g. Input Date para el **fecha_inicio_disponibilidad**(se debe validar que el input no sea vacío antes enviarlo)
h. Input para el **cantidad_días**(se debe validar que el input no sea vacío antes enviarlo y que sea un número)
i. Input checkbox para el **disponible**
j. Input para el **valor_noche**(se debe validar que el input no sea vacío antes enviarlo y que sea un número)
k. select que permite elegir el **tipo de propiedad** (se debe validar que se seleccione algo antes de enviar el formulario)
l. Input file para la **imagen** (el tipo de la imagen lo pueden obtener de este mismo file antes de enviarlo al backend)
m. Y un botón para realizar el submit del formulario (Una vez enviados los datos del formulario al backend, se debe mostrar el mensaje devuelto por servicio)

5. Crear un archivo dentro de ***pages/propiedad*** llamado **EditPropiedad** que se carga al presionar editar:
    a. El formulario debe venir precargado con los valores de la propiedad
    b. Se debe validar lo mismo que en el **NewPropiedad**
    c. Se debe mostrar el mensaje devuelto por el endpoint (éxito o error)

6. Al presionar en la acción "**Eliminar**" del listado se debe borrar el item del listado. En caso que no se pueda eliminar, se debe mostrar un mensaje indicando lo ocurrido. **Se debe pedir confirmación antes de realizar la acción.**

c. Reservas

Crear un archivo dentro de la carpeta ***pages/reserva*** llamado **ReservaPage**.

Dicha página debe contener:

1. Listado de todos las reservas

    a. Propiedad (domicilio, no **propiedad_id**)
    b. Inquilino(nombre y apellido del inquilino, no **inquilino_id**)
    c. Fecha desde
    d. Cantidad noches
    e. valor total

f. Cada ítem del listado debe tener las acciones para editar y eliminar

2. Un botón para crear una nueva reserva. Crear un archivo dentro de ***pages/reserva*** llamado **NewReserva**. Dicha página debe contener un formulario con lo siguiente:

    a. Select que permite elegir una **propiedad** (se debe validar que se seleccione algo antes de enviar el formulario)
    b. Select que permite elegir un **inquilino** (se debe validar que se seleccione algo antes de enviar el formulario)
    c. Input Date para el **fecha_desde** (se debe validar que el input no sea vacío antes enviarlo)
    d. Input para el **cantidad_noches** (se debe validar que el input no sea vacío antes enviarlo y que sea un número)
    e. Input para el **valor_total** (se debe validar que el input no sea vacío antes enviarlo y que sea un número)
    f. Y un botón para realizar el submit del formulario (Una vez enviados los datos del formulario al backend, se debe mostrar el mensaje devuelto por servicio)

3. Crear un archivo dentro de ***pages/reserva*** llamado **EditReserva** que se carga al presionar editar:
    a. El formulario debe venir precargado con los valores de la reserva
    b. Se debe validar lo mismo que en el **NewReserva**
    c. Se debe mostrar el mensaje devuelto por el endpoint (éxito o error)

4. Al presionar en la acción "**Eliminar**" del listado se debe borrar el item del listado. En caso que no se pueda eliminar, se debe mostrar un mensaje indicando lo ocurrido. **Se debe pedir confirmación antes de realizar la acción.**

4) Módulos a desarrollar de forma <u>opcional</u>:

a. Localidades

Crear un archivo dentro de la carpeta ***pages/localidad*** llamado **LocalidadPage**

Dicha página debe contener:

2. Listado de todos las localidades:
    a. nombre
    b. cada ítem del listado debe tener las acciones para editar y eliminar

3. Un botón para crear una nueva localidad. Crear un archivo dentro de ***pages/localidad*** llamado **NewLocalidad**. Dicha página debe contener un formulario con lo siguiente:
    a. Input para ingresar el **nombre** (se debe validar que el input no sea vacío antes enviarlo)

b. Un botón para realizar el envío del formulario (una vez enviados los datos del formulario al backend, se debe mostrar el mensaje devuelto por servicio)

4. Crear un archivo dentro de ***pages/localidad*** llamado **EditLocalidad** que se carga al presionar editar:
    a. El formulario debe venir precargado con los valores de la localidad
    b. Se debe validar lo mismo que en el **NewLocalidad**
    c. Se debe mostrar el mensaje devuelto por el endpoint (éxito o error)

5. Al presionar en la acción "**Eliminar**" del listado se debe borrar el item del listado. En caso que no se pueda eliminar, se debe mostrar un mensaje indicando lo ocurrido. **Se debe pedir confirmación antes de realizar la acción.**

b. Inquilinos

Crear un archivo dentro de la carpeta ***pages/inquilino*** llamado **InquilinoPage**

Dicha página debe contener:

1. Listado de todos los inquilinos
    a. Nombre
    b. Apellido
    c. Documento
    d. email
    e. activo
    f. Cada ítem del listado debe tener las acciones para ver detalle, editar y eliminar

2. Crear un archivo dentro de ***pages/*inquilino** llamado **DetailInquilino**. Dicha página deberá mostrar los datos del inquilino seleccionado, y un listado de todas sus reservas.

3. Un Botón para crear un nuevo inquilino. Crear un archivo dentro de ***pages/*inquilino** llamado **NewInquilino**. Dicha página debe contener un formulario con lo siguiente:
    a. Input para ingresar el **nombre** (se debe validar que el input no sea vacío antes enviarlo)
    b. Input para ingresar el **apellido** (se debe validar que el input no sea vacío antes enviarlo)
    c. Input para el **documento** (se debe validar que el input no sea vacío antes enviarlo y que sea un número)
    d. Input Email para el **email** (se debe validar que el input no sea vacío antes enviarlo y que sea un email válido)
    e. Input checkbox para el **activo**
    f. Y un botón para realizar el submit del formulario (Una vez enviados los datos del formulario al backend, se debe mostrar el mensaje devuelto por servicio)

4. Crear un archivo dentro de ***pages/*inquilino** llamado **EditInquilino** que se carga al presionar editar:
    a. El formulario debe venir precargado con los valores del inquilino
    b. Se debe validar lo mismo que en el **NewInquilino**
    c. Se debe mostrar el mensaje devuelto por el endpoint (éxito o error)

5. Al presionar en la acción "**Eliminar**" del listado se debe borrar el item del listado. En caso que no se pueda eliminar, se debe mostrar un mensaje indicando lo ocurrido. **Se debe pedir confirmación antes de realizar la acción.**

Requisitos obligatorios

- Usar las últimas versiones estables de **nodejs** y **ReactJS.**
- El nombre del proyecto debe ser **inmobiliaria**
- Los estilos NO pueden estar embebidos en el código
- Validar todos los formularios antes de realizar los llamados al backend
- Si instalan librerias externas dejarlas documentadas en el archivo **README.md**
- Si modifican algunos endpoint/agregan, documentarlos en el **README.md** especificando ruta del servicio y definir porque se realizó la modificación.
- Para la creación de componentes se debe utilizar componentes de funciones (Function Components).

Metodología para la entrega en Ideas

- Hacer un archivo zip o rar con todos los archivos de **ambos proyectos** (excepto la carpetas ***vendor*** —API— y ***node_modules*** —React—)
- Subir a la sección del grupo asignada

Fecha de entrega: a definir