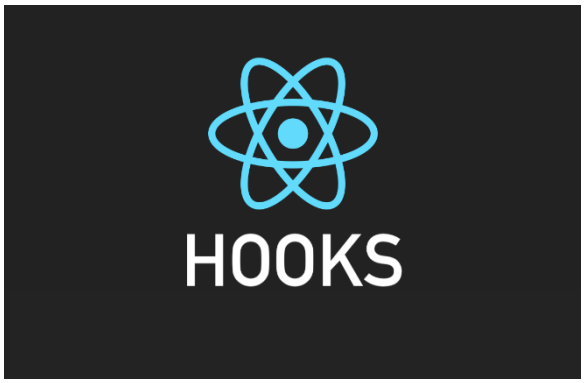
CUADERNO PERSONAL

DESARROLLO DE APLICACIONES WEB

SEMANA 7 – Uso de Hooks



Introducción a los Hooks

Los **Hooks** en React son funciones que permiten utilizar el estado y otras características de React sin necesidad de escribir una clase. Fueron introducidos en React versión 16.8 y solucionan algunas de las limitaciones que se tenían al trabajar únicamente con componentes de clase. Entre sus ventajas, los Hooks simplifican el uso y la reutilización de lógica de estado, mejoran la organización del código, y permiten un mejor manejo del ciclo de vida de los componentes.

Ventajas de Usar Hooks

1. **Reutilización de lógica de estado:** Los Hooks permiten extraer lógica de estado de un componente para reutilizarla en otros. Esto es especialmente útil para crear componentes más pequeños y modulares.
2. **Composición de funciones:** Los Hooks promueven la composición de funciones, permitiendo que los desarrolladores combinen hooks en un mismo componente para realizar tareas complejas.
3. **Mejora la legibilidad y organización del código:** Al no tener que gestionar el ciclo de vida del componente directamente, el código se vuelve más limpio y fácil de entender.
4. **Sin clases:** Los Hooks permiten usar el estado y otras características de React en componentes funcionales, eliminando la necesidad de clases y las confusiones que estas conllevan, como el manejo del this.

Principales Hooks de React

1. useState

useState es un Hook que permite añadir estado a los componentes funcionales de React. Este hook devuelve un par: el valor del estado actual y una función que permite actualizarlo.

- **Definición:** useState acepta un único argumento que es el valor inicial del estado y retorna un array con dos elementos: el valor actual del estado y una función que se utiliza para actualizarlo.

2. useEffect

useEffect es un Hook que permite realizar efectos secundarios en los componentes funcionales. Algunos ejemplos de efectos secundarios son las llamadas a APIs, las suscripciones a eventos y las manipulaciones del DOM.

- **Definición:** useEffect acepta una función que representa el efecto a ejecutar y un array de dependencias que determina cuándo debe ejecutarse el efecto.

3. useContext

useContext es un Hook que permite acceder al contexto en componentes funcionales. Los contextos en React permiten compartir valores como temas, configuración de localización y datos de usuario entre componentes sin necesidad de pasar props manualmente por cada nivel del árbol de componentes.

- **Definición:** useContext acepta un contexto y retorna el valor actual del contexto, que se determina a partir del componente proveedor más cercano en el árbol.

4. useRef

useRef es un Hook que permite crear una referencia mutable que persiste durante todo el ciclo de vida de un componente. Es útil para acceder a elementos del DOM o para almacenar valores que no requieren una nueva renderización al cambiar.

- **Definición:** useRef devuelve un objeto de referencia mutable cuya propiedad .current se inicializa con el valor proporcionado.

5. useReducer

useReducer es un Hook que es una alternativa a useState para gestionar el estado en componentes que tienen lógica compleja o múltiple. Es similar al patrón de reducer en Redux, donde se utiliza una función reductora para manejar las actualizaciones de estado.

- **Definición:** useReducer acepta una función reductora y un estado inicial, retornando el estado actual y una función de despacho.

6. useCallback

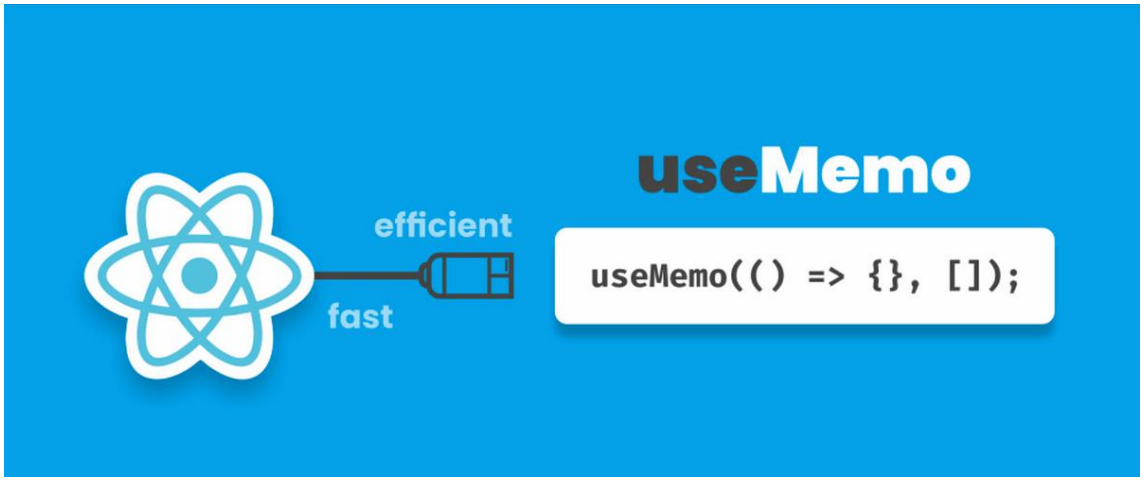
useCallback es un Hook que memoriza una función para evitar su recreación en cada renderización del componente. Esto es útil cuando se pasan funciones como props a componentes hijos que dependen de que la referencia de la función no cambie, mejorando el rendimiento.

- **Definición:** useCallback acepta una función y un array de dependencias, retornando una versión memorizada de la función que sólo se actualiza cuando cambian las dependencias.

7. useMemo

useMemo es un Hook que memoriza el resultado de una función computacional para evitar cálculos costosos en cada renderización del componente. Es similar a useCallback, pero se utiliza para memorizar valores computados.

- **Definición:** useMemo acepta una función computacional y un array de dependencias, retornando el valor memorizado que sólo se actualiza cuando cambian las dependencias.



Hooks Personalizados

Los **Hooks Personalizados** permiten crear funciones de Hook que encapsulan lógica de estado reutilizable en componentes de React. Estos hooks son funciones de JavaScript que pueden utilizar otros hooks de React internamente para implementar lógica personalizada.

Definición

Un Hook personalizado es una función de JavaScript que sigue la convención de nombrar comenzando con use y puede llamar a otros hooks de React en su implementación. Permiten compartir lógica de estado y efectos entre componentes de forma modular y reutilizable.

Practica:

Código:

```
// ColorContext.js
import React, { createContext, useState } from "react";

export const ColorContext = createContext();

export const ColorProvider = ({ children }) => {
  const [color1, setColor1] = useState("blue");
  const [color2, setColor2] = useState("green");
  const [color3, setColor3] = useState("lightblue");

  const getRandomColor = () => {
    const letters = "0123456789ABCDEF";
    let color = "#";
    for (let i = 0; i < 6; i++) {
      color += letters[Math.floor(Math.random() * 16)];
    }
    return color;
  };

  const changeColors = () => {
    setColor1(getRandomColor());
    setColor2(getRandomColor());
    setColor3(getRandomColor());
  };

  return (
    <ColorContext.Provider value={{ color1, color2, color3, changeColors
}}>
```

```
      {children}
    </ColorContext.Provider>
  );
};


// Componente1.js
import React, { useContext } from "react";
import { ColorContext } from "./ColorContext";

function Componente1() {
  const { color1, changeColors } = useContext(ColorContext);
  return (
    <div className="componente1" style={{ backgroundColor: color1 }}>
      <button onClick={changeColors}>Cambiar color</button>
    </div>
  );
}

export default Componente1;

// Componente2.js
import React, { useContext } from "react";
import { ColorContext } from "./ColorContext";

function Componente2() {
  const { color2, changeColors } = useContext(ColorContext);
  return (
    <div className="componente1" style={{ backgroundColor: color2 }}>
      <button onClick={changeColors}>Cambiar color</button>
    </div>
  );
}

export default Componente2;

// Componente3.js
import React, { useContext } from "react";
import { ColorContext } from "./ColorContext";

function Componente3() {
  const { color3, changeColors } = useContext(ColorContext);
  return (
    <div className="componente3" style={{ backgroundColor: color3 }}>
      <img src="/myImage.png" alt="My photo" />
      <button onClick={changeColors}>Cambiar color</button>
    </div>
  );
}
```

```
export default Componente3;

import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);

import React, { useState } from "react";
import Componente1 from "./Componente1";
import Componente2 from "./Componente2";
import Componente3 from "./Componente3";

const getRandomColor = () => {
  const letters = "0123456789ABCDEF";
  let color = "#";
  for (let i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
};

const ParentComponent = () => {
  const [backgroundColor, setBackgroundColor] = useState("#ffffff");

  const handleChangeColor = () => {
    setBackgroundColor(getRandomColor());
  };

  return (
    <div>
      <button onClick={handleChangeColor}>Cambiar Color de Todos</button>
      <Componente1 initialColor={backgroundColor} />
      <Componente2 initialColor={backgroundColor} />
      <Componente3 initialColor={backgroundColor} />
    </div>
  );
};

export default ParentComponent;

import { useState, useEffect } from "react";
```

```
const images = [
  "/image1.jpg",
  "/image2.jpg",
  "/image3.jpg",
  "/image4.jpg",
  "/image5.jpg",
];

const getRandomImage = () => {
  const randomIndex = Math.floor(Math.random() * images.length);
  return images[randomIndex];
};

const useRandomImage = () => {
  const [image, setImage] = useState(getRandomImage());

  useEffect(() => {
    const interval = setInterval(() => {
      setImage(getRandomImage());
    }, 1000); // 30 segundos

    return () => clearInterval(interval); // Limpiar intervalo al
desmontar el componente
  }, []);

  return image;
};

export default useRandomImage;

import React from "react";
import useRandomImage from "./useRandomImage";
import "./ImageWidget.css";

const ImageWidget = () => {
  const image = useRandomImage();

  return (
    <div className="image-widget">
      <img src={image} alt="Random" />
    </div>
  );
};

export default ImageWidget;

import React from "react";
```

```
const EmployeeCard = ({ employee, onClose }) => {
  return (
    <div className="employee-card">
      <h2>{employee.name}</h2>
      <p>
        <strong>Dirección:</strong> {employee.address}
      </p>
      <p>
        <strong>Ciudad:</strong> {employee.city}
      </p>
      <button onClick={onClose}>Cerrar</button>
    </div>
  );
};

export default EmployeeCard;

import React, { useState, useContext, useEffect } from "react";
import { EmployeeContext } from "../context/EmployeeContext";

const EmployeeForm = ({ editableEmployee, setEditableEmployee }) => {
  const { dispatch } = useContext(EmployeeContext);
  const [employee, setEmployee] = useState({
    id: "",
    name: "",
    address: "",
    city: "",
  });

  useEffect(() => {
    if (editableEmployee) {
      setEmployee(editableEmployee);
    }
  }, [editableEmployee]);

  const handleChange = (e) => {
    setEmployee({ ...employee, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (employee.id) {
      dispatch({ type: "EDIT_EMPLOYEE", payload: employee });
    } else {
      dispatch({
        type: "ADD_EMPLOYEE",
        payload: { ...employee, id: Date.now() },
      });
    }
```

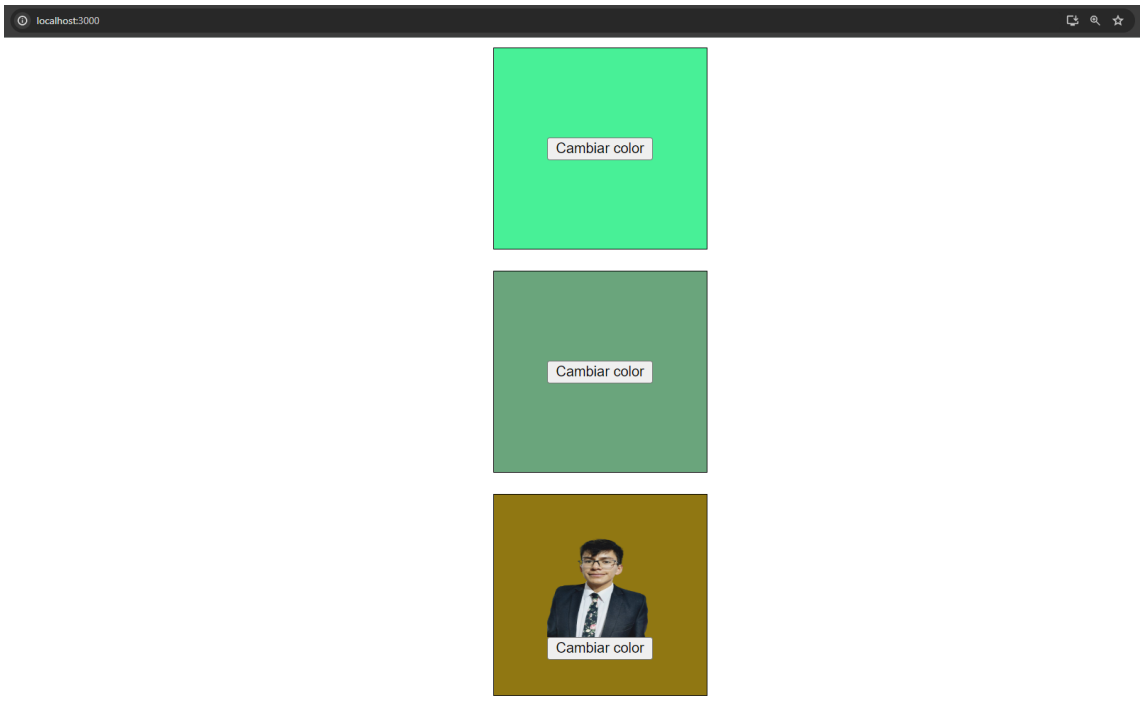
```
    setEmployee({ id: "", name: "", address: "", city: "" });
    setEditableEmployee(null);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        name="name"
        placeholder="Nombre"
        value={employee.name}
        onChange={handleChange}
        required
      />
      <input
        type="text"
        name="address"
        placeholder="Dirección"
        value={employee.address}
        onChange={handleChange}
        required
      />
      <input
        type="text"
        name="city"
        placeholder="Ciudad"
        value={employee.city}
        onChange={handleChange}
        required
      />
      <button type="submit">
        {employee.id ? "Editar" : "Agregar"} Empleado
      </button>
    </form>
  );
};

export default EmployeeForm;

import React, { useContext } from "react";
import { EmployeeContext } from "../context/EmployeeContext";

const EmployeeItem = ({
  employee,
  setEditableEmployee,
  setDetailedEmployee,
}) => {
  const { dispatch } = useContext(EmployeeContext);
```

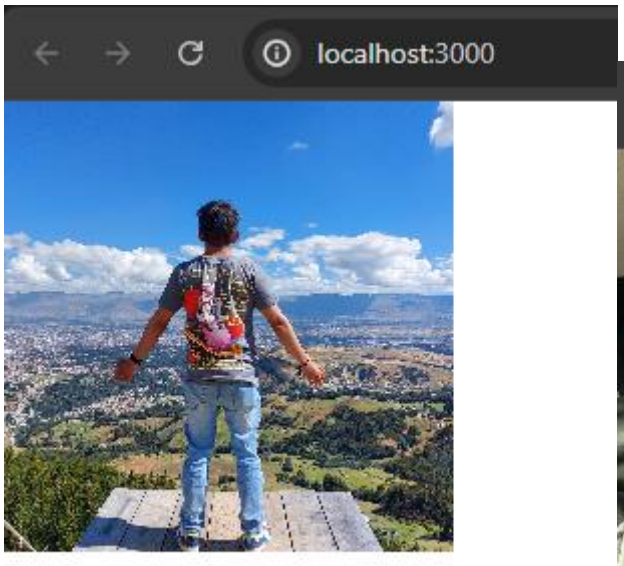
```
  const handleDelete = () => {
    dispatch({ type: "DELETE_EMPLOYEE", payload: employee.id });
  };

  const handleEdit = () => {
    setEditableEmployee(employee);
  };

  const handleShowDetails = () => {
    setDetailedEmployee(employee);
  };

  return (
    <li>
      {employee.name} - {employee.address} - {employee.city}
      <button onClick={handleEdit}>Editar</button>
      <button onClick={handleDelete}>Eliminar</button>
      <button onClick={handleShowDetails}>Mostrar Detalles</button>
    </li>
  );
};

export default EmployeeItem;

import React, { useContext, useState } from "react";
import { EmployeeContext } from "../context/EmployeeContext";
import EmployeeItem from "./EmployeeItem";
import EmployeeForm from "./EmployeeForm";
import EmployeeCard from "./EmployeeCard";

const EmployeeList = () => {
  const { state } = useContext(EmployeeContext);
  const [editableEmployee, setEditableEmployee] = useState(null);
  const [detailedEmployee, setDetailedEmployee] = useState(null);
  const [filter, setFilter] = useState("");
  const [currentPage, setCurrentPage] = useState(1);
  const employeesPerPage = 5;

  const handleFilterChange = (e) => {
    setFilter(e.target.value);
    setCurrentPage(1); // Reset to first page when filter changes
  };

  const handleFilter = () => {
    return state.employees.filter(
      (employee) =>
        employee.name.toLowerCase().includes(filter.toLowerCase()) ||
        employee.address.toLowerCase().includes(filter.toLowerCase()) ||
        employee.city.toLowerCase().includes(filter.toLowerCase())
```

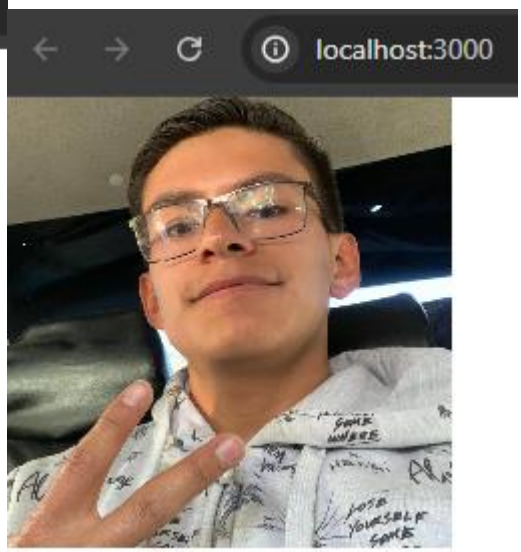
```
  );
};

const filteredEmployees = handleFilter();

// Logic for displaying employees
const indexOfLastEmployee = currentPage * employeesPerPage;
const indexOfFirstEmployee = indexOfLastEmployee - employeesPerPage;
const currentEmployees = filteredEmployees.slice(
  indexOfFirstEmployee,
  indexOfLastEmployee
);

const paginate = (pageNumber) => setCurrentPage(pageNumber);

const pageNumbers = [];
for (
  let i = 1;
  i <= Math.ceil(filteredEmployees.length / employeesPerPage);
  i++
) {
  pageNumbers.push(i);
}

return (
  <>
    <EmployeeForm
      editableEmployee={editableEmployee}
      setEditableEmployee={setEditableEmployee}
    />
    <div className="filter-container">
      <input
        type="text"
        placeholder="Filtrar empleados"
        value={filter}
        onChange={handleFilterChange}
      />
    </div>
    <ul>
      {currentEmployees.map((employee) => (
        <EmployeeItem
          key={employee.id}
          employee={employee}
          setEditableEmployee={setEditableEmployee}
          setDetailedEmployee={setDetailedEmployee}
        />
      ))}
    </ul>
    {detailedEmployee && (
```

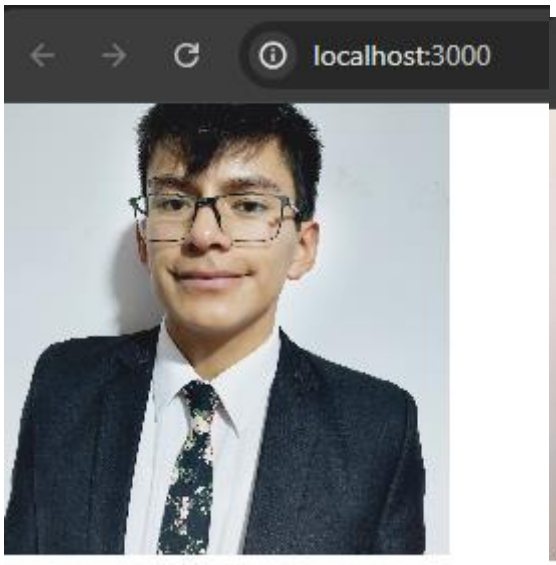
```
        <EmployeeCard
          employee={detailedEmployee}
          onClose={() => setDetailedEmployee(null)}
        />
      )}
      <div className="pagination">
        {pageNumbers.map((number) => (
          <button key={number} onClick={() => paginate(number)}>
            {number}
          </button>
        ))}
      </div>
    </>
  );
};

export default EmployeeList;

import React from "react";
import EmployeeProvider from "./context/EmployeeContext";
import EmployeeList from "./components/EmployeeList";

const App = () => {
  return (
    <EmployeeProvider>
      <div className="container">
        <h1>Administrar Empleados</h1>
        <EmployeeList />
      </div>
    </EmployeeProvider>
  );
};

export default App;
```

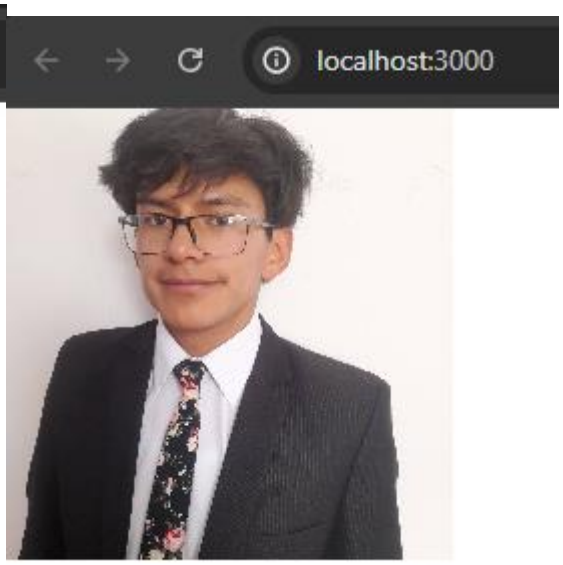
Vista desde el LocalHost:

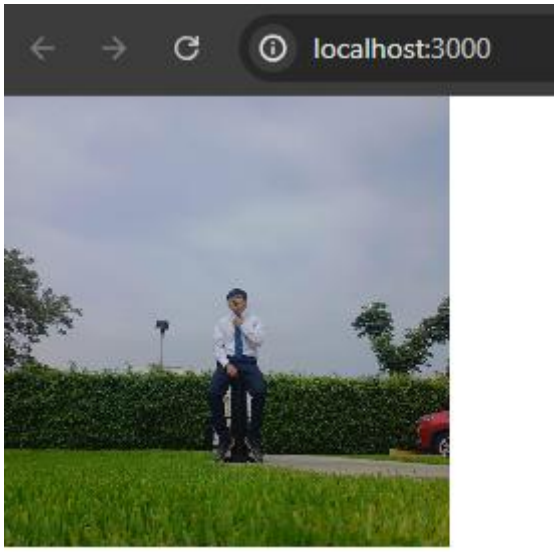

localhost:3000
Cambiar color
Cambiar color
Cambiar color


localhost:3000


localhost:3000


localhost:3000


localhost:3000


localhost:3000


localhost:5173
Administrar Empleados
Nombre
Dirección
Ciudad
Agregar Empleado
Filtrar empleados
Cristian Torres Cordova - Pasaje. Rio Chilca #304 - Chilca
Editar
Eliminar
Mostrar Detalles
Cristian Torres Cordova - Pasaje. Rio Chilca #304 - Chilca
Editar
Eliminar
Mostrar Detalles
Cristian Torres Cordova - Pasaje. Rio Chilca #304 - Chilca
Editar
Eliminar
Mostrar Detalles
Cristian Torres Cordova - Pasaje. Rio Chilca #304 - Chilca
Editar
Eliminar
Mostrar Detalles
Cristian Torres Cordova - Pasaje. Rio Chilca #304 - Chilca
Editar
Eliminar
Mostrar Detalles
1
2