Fixing a Small Bug

The `<dialog>` element can also be closed by pressing the `ESC` key on the keyboard. In that case, the dialog will disappear but the state passed to the `open` prop (i.e., the `modalIsOpen` state) will not be set to `false`.

Therefore, the modal can't be opened again (because `modalIsOpen` still is `true` - the UI basically now is not in sync with the state anymore).

To fix this issue, we must listen to the modal being closed by adding the built-in `onClose` prop to the `<dialog>`. The event is then "forwarded" to the `App` component by accepting a custom `onClose` prop on the `Modal` component.

The `Modal` component therefore should look like this:

```
1. import { useRef, useEffect } from 'react';
2. import { createPortal } from 'react-dom';
3.
4. function Modal({ open, children, onClose }) {
5.   const dialog = useRef();
6.
7.   useEffect(() => {
8.     if (open) {
9.       dialog.current.showModal();
10.    } else {
11.      dialog.current.close();
12.    }
13.  }, [open]);
14.
15.  return createPortal(
16.    <dialog className="modal" ref={dialog} onClose={onClose}>
17.      {children}
18.    </dialog>,
19.    document.getElementById('modal')
20.  );
21. }
22.
23. export default Modal;
```

In the `App` component, we can now set the `handleStopRemovePlace` function as a value for the `onClose` prop on the `<Modal>` component:

```
1. <Modal open={modalIsOpen} onClose={handleStopRemovePlace}>
2.   <DeleteConfirmation
3.     onCancel={handleStopRemovePlace}
4.     onConfirm={handleRemovePlace}
5.   />
6. </Modal>
```