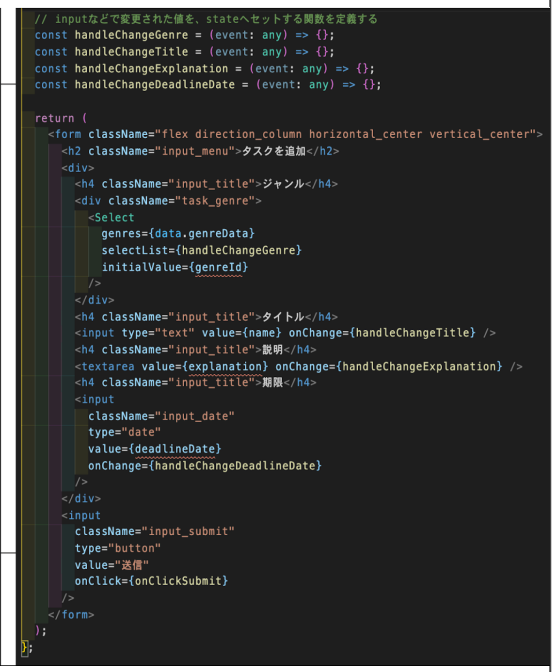
タスクの作成/編集を実装

達成要件	実装の例
■「実装の例」を参考に、タスクを追加するモーダルを完成させてください。また、タスクの propsがある場合には編集になるよう実装してください。 ■ 編集の場合は、Taskコンポーネントのクラス名がtask_text_contentsの要素をクリックすると、編集用のモーダルが開くようにしてください。その時に、選択したタスクの情報をpropsでモーダルへ渡してください。	 

```tsx
import React, { useContext, useState } from "react";
import { Select } from "../../components/select";
import { DataContext } from "../../pages/home";
import { TaskType } from "../../interfaces/Task";
import { taskRequest } from "../../requests/taskRequest";
import "./style.css";

interface Props {
  // 編集の場合は、タスクを受け取る
  task?: TaskType;
  handleClose: () => void;
}

export const TaskBody = (props: Props) => {
  const date: Date = new Date();

  const { data, dispatch } = useContext(DataContext);

  // name,genreId,explanation,deadlineDateをstateで管理する

  const onClickSubmit = async () => {
    const requestData = {
      name: name,
      genre_id: genreId,
      explanation: explanation,
      deadline_date: deadlineDate,
      status: (props.task && props.task.status) || 0,
    };

    if (props.task !== undefined) {
      try {
        // props.taskがundefinedではない場合、すなわち編集の場合は、requestDataをパラメーターとして、編集するリクエストを行う
      } catch (err) {
        dispatch({ type: "failure", payload: { error: err.message } });
      }
    } else {
      try {
        // requestDataをパラメータとして、新規作成するリクエストを行う
      } catch (err) {
        dispatch({ type: "failure", payload: { error: err.message } });
      }
    }
    props.handleClose();
  };
```

```tsx
  // inputなどで変更された値を、stateへセットする関数を定義する
  const handleChangeGenre = (event: any) => {};
  const handleChangeTitle = (event: any) => {};
  const handleChangeExplanation = (event: any) => {};
  const handleChangeDeadlineDate = (event: any) => {};

  return (
    <form className="flex direction_column horizontal_center vertical_center">
      <h2 className="input_menu">タスクを追加</h2>
      <div>
        <h4 className="input_title">ジャンル</h4>
        <div className="task_genre">
          <Select
            genres={data.genreData}
            selectList={handleChangeGenre}
            initialValue={genreId}
          />
        </div>
        <h4 className="input_title">タイトル</h4>
        <input type="text" value={name} onChange={handleChangeTitle} />
        <h4 className="input_title">説明</h4>
        <textarea value={explanation} onChange={handleChangeExplanation} />
        <h4 className="input_title">期限</h4>
        <input
          className="input_date"
          type="date"
          value={deadlineDate}
          onChange={handleChangeDeadlineDate}
        />
      </div>
      <input
        className="input_submit"
        type="button"
        value="送信"
        onClick={onClickSubmit}
      />
    </form>
  );
};
```