src\component\SearchForm\SearchForm.hook.ts

```ts
import { useEffect } from "react";
import { pokemonListServices, pokemonDetailServices } from "@/services";
import { useForm } from "react-hook-form";
import { usePokemonListStore } from "@/store/pokemonList";
import { generationList, typesList, sortList } from "@/utils/optionList";
import { IPokemonDetailResponse } from "@/interface/pokemonDetail";

const useSearchForm = () => {
  const { register, handleSubmit, watch, formState } = useForm();

  const { setFetchPokemonList, fetchPokemon, setPokemonList } =
    usePokemonListStore();

  const keyword = watch("keyword");
  const generation = watch("generation");
  const type = watch("type");
  const sort = watch("sort");

  const callData = async (filter: {
    name: string;
    limit: number;
    offset: number;
  }) => {
    const responseList = await pokemonListServices.getPokemonList(
      filter.limit,
      filter.offset
    );
    const pokelist = [];
    setFetchPokemonList({ data: [], loading: true, error: null });

    if (responseList.status === 200) {
      const responseResults = responseList.data?.results || [];
      for (const pokemon of responseResults) {
        const response = await pokemonDetailServices.getPokemonDetail(
          pokemon.name
        );
        const pokeData = response.data;
        if (pokeData)
          pokelist.push({
            ...pokeData,
            image:
              pokeData.sprites.other.dream_world.front_default ||
              pokeData.sprites.other["official-artwork"].front_default,
          });
      }
      setFetchPokemonList({ data: pokelist, loading: false, error: null });
      setPokemonList({
        data: pokelist,
        loading: false,
        error: null,
      });
```

```
    } else {
      setFetchPokemonList({
        data: [],
        loading: false,
        error: responseList.error,
      });
    }
  };

  const filterPokemon = (
    keyword: string,
    type: string,
    sort: string,
  ) => {
    console.log(`keyword`,keyword)
    console.log(`type`,type)
    console.log(`sort`,sort)

    const keywordFilter = fetchPokemon.data.filter((item) =>
      item.name.toLowerCase().includes(keyword?.toLowerCase())
    );

    const typeFilter =
      type !== "all types"
        ? keywordFilter.filter((item) =>
            item.types.find((f) =>
              f.type.name.toLowerCase().includes(type.toLowerCase())
            )
          )
        : keywordFilter;

    return sortBy(typeFilter, sort);
  };

  const sortBy = (data: IPokemonDetailResponse[], type: string) => {
    switch (type) {
      case "id":
        return data.sort((a, b) => a.id - b.id);
      case "name":
        return data.sort((a, b) =>
          a.name > b.name ? 1 : b.name > a.name ? -1 : 0
        );
      default:
        return data.sort((a, b) => a.id - b.id);
    }
  };

  useEffect(() => {
    if (generation !== undefined) callData(generationList[generation]);
  }, [generation]);

  useEffect(() => {
    const data = filterPokemon(keyword, typesList[type], sortList[sort]);
    setPokemonList({
```

```
      data: data,
      loading: false,
      error: null,
    });
  }, [keyword, type, sort]);

  return {
    fieldKeyword: register("keyword"),
    fieldGeneration: register("generation"),
    fieldType: register("type"),
    fieldSort: register("sort"),
  };
};

export { useSearchForm };

```