

# Критерії прийому

Використовуй цей [шаблон React-проекту](#) як стартову точку своєї програми.

- Створено репозиторій `goit-react-hw-07-phonebook`
- Використана бібліотека `Redux Toolkit`

## Книга контактів

Виконайте рефакторинг коду програми «Книга контактів». Видали код, що відповідає за зберігання та читання контактів з локального сховища, та додай взаємодію з бекендом для зберігання контактів.

### Бекенд

Створи свій персональний бекенд для розробки за допомогою UI-сервісу [mockapi.io](#). Зареєструйся використовуючи свій обліковий запис GitHub. Створи ресурс `contacts` щоб отримати ендпоінт `/contacts`. Використай конструктор ресурсу та опиши об'єкт контакту як на ілюстрації.



### Resource name

Enter meaningful resource name, it will be used to generate API endpoints.

contacts

### Schema (optional)

Define Resource schema, it will be used to generate mock data.

id

Object ID

createdAt

Faker.js

Recent

name

Faker.js

Find name

phone

Faker.js

Phone number form

+

### Object template (optional)

To define more complex structure for your data use JSON template. You can

CREATE

CANCEL

## Форма стану

Додай у стан Redux обробку індикатора завантаження та помилки. Для цього змieni форму стану.

```
{
  contacts: {
    items: [],
    isLoading: false,
    error: null
  },
  filter: ""
}
```

## Операції

Використовуй функцію `createAsyncThunk` для оголошення асинхронних генераторів екшенів та виконання HTTP-запитів. Обробку екшенів та зміну даних у стані Redux зроби за допомогою `createSlice`.

Оголоши наступні операції:

- `fetchContacts` - одержання масиву контактів (метод GET) запитом. Базовий тип екшену `"contacts/fetchAll"`.
- `addContact` - додавання контакту (метод POST). Базовий тип екшену `"contacts/addContact"`.
- `deleteContact` - видалення контакту (метод DELETE). Базовий тип екшену `"contacts/deleteContact"`.