Building a Basic DApp on ICP Local Network

Overview:

This guide walks through building a simple DApp on the Internet Computer (ICP) local network with a Rust backend and React.js frontend. The DApp performs CRUD (Create, Read, Update, Delete) operations and is deployed locally.

1. Prerequisites:

- Node.js and npm

- Rust and cargo

- dfx (Internet Computer SDK)

- React.js (via Create React App)

2. Setup ICP Project:

```
$ dfx new icp_dapp
$ cd icp_dapp
```

3. Backend (Rust):

Edit `src/icp_dapp_backend/src/lib.rs`:

```rust
#[derive(Debug, Clone, CandidType, Deserialize)]
pub struct Item {
    id: u64,
    name: String,
}
```

```rust
static mut ITEMS: Vec<Item> = Vec::new();

#[update]
fn create_item(id: u64, name: String) {

    unsafe {

        ITEMS.push(Item { id, name });

    }

}


#[query]
fn read_items() -> Vec<Item> {

    unsafe { ITEMS.clone() }

}


#[update]
fn update_item(id: u64, new_name: String) {

    unsafe {

        if let Some(item) = ITEMS.iter_mut().find(|i| i.id == id) {

            item.name = new_name;

        }

    }

}


#[update]
fn delete_item(id: u64) {

    unsafe {

        ITEMS.retain(|item| item.id != id);
```

```
    }
}
```

4. Frontend (React.js):

$ cd src/icp_dapp_frontend

$ npm install

Use agent-js to interact with canister.

Example React Component:

```
import { createActor } from '../declarations/icp_dapp_backend';

const actor = createActor();

const createItem = async () => {
  await actor.create_item(1, "Sample Item");
};
```

5. Deploy on Local:

Start the local replica:

$ dfx start --background

Deploy the canisters:

$ dfx deploy

Open in browser:

$ dfx webserver

Access the frontend: http://localhost:8000

Conclusion:

You now have a simple CRUD DApp running locally on the Internet Computer using Rust and React.js.