1. Create project by dotnet CLI:

--clear

--dotnet --info

--dotnet -h

--dotnet new -l

--mkdir Reactivities

--cd Reactivities

--dotnet new sln

--ls

--dotnet new webapi -n API

--dotnet new classlib -n Application

--dotnet new classlib -n Domain

--dotnet new classlib -n Persistence

--dotnet sln add API/API.csproj

--dotnet sln add Application

--dotnet sln add Persistence

--dotnet sln add Domain

--dotnet sln list

--cd/API

--dotnet add reference ../Application

--cd ..

--cd Application

--dotnet add reference ../Persistence

--dotnet add reference ../Domain

--cd ..

--cd Persistence

--dotnet add reference ../Domain

2. Generate assesst for run and debug: choose Yes to add/ Or ctr+shift+p type generate

--exclude obj and bin

--edit appsettings.Development.json: "Microsoft": "Information",

"Microsoft.Hosting.Lifetime": "Information",

to get more log details

3. Setting up Postman

4. Create Domain files: Activity. These properties inside the class will form columns by using EF Core

5. EF Core create database

--Object Relational Mapper: Allow using C# code to querry the database.

--Check the tool to use entity framework: dotnet tool list --global

--https://www.nuget.org/packages/dotnet-ef/7.0.0

--Update to the newest version: dotnet tool update --global dotnet-ef --version 7.0.0

--Create code first EF Migration:

dotnet ef migrations add InitialCreate -p Persistence -s API

--Create seed data: Using task to implement Asynchronous Programming i.e. executing operations asynchronously

--use the other watch run version:

dotnet watch --no-hot-reload //work 99% of the cases

--Create a basic Controller

6. Create a basic React Application

--npx create-react-app client-app --use-npm --template typescript

React use the concept of virtual DOM

--React Hook to build a function application: Hook in react is functions that allows we hook into React state and LifeCycle fearture from function component. Function

component is literaly a function that return JSX

--useState(): assign and track state of a component

--useEffect(): this allow we hook into a life cycle event inside our component. When our component mount or initialize on our page, we can use this hook to add a

side effect when something happen when our component mount, or take an action when our component unmount

--useWhateverWeWant()

--Typescript adv: strong typing = IDE will get error before combining code; Object oriented; better intellisense; Access modifiers (use public or private in class or function)

and it will be removed when the code complied to JS; Future JS Features; Catch silly mistakes in dev = save tons of time; 3rd party lib are vast

Easy to learn if you know JS; much improved in React

--Typescript pros: More upfront code; 3rd party libraries don't support typescript; strict mode is strict

--Don't use conventional naming in Interface: Don't use I...

--Child component of the App component: Using Props to pass property into a child component:

{ducks.map(duck => (

<DuckItem duck={duck} key={duck.name}/>

))}

--install react developer tools

--fetch data by using axios: Main reason to use is it allows to intercept request and response

--npm install axios

--create a variable : const

--Call Back function (() =>)

--when using useEffect, it is important to give it dependency, otherwise it will be fired up:

axios.get('http://localhost:5116/api/activities')

.then(response => {

setActivities(response.data);

})

This function will loop forever

--Different domain fixed by enable CORS

--Install CSS Framework: Semantic UI

https://react.semantic-ui.com/

import the css framework into index.tsx

--install extension: Auto Rename Tag

7. Create CRUD application using CQRS + Mediator pattern

--Clean architecture

--CQRS: Command Query Responsibility Segalation: Separate Command vs Query

--Command: Does st, Modifies State, Should not return a value

--Query: Answers a question, Does not modify State, and Should return a value

--CQRS Read/Write Database can reduce the join table. Read database can be noSql such as mongoDB

7.1. Query Handler

7.1 Architecture

