Buổi 18. Tương tác hợp đồng Helloworld với giao diện frontend.

Xin chào các bạn , chào các bạn đã quay trở lại với khóa học lập trình aiken cơ bản trong bài học trước các bạn cũng đã biết cách viết một hợp đồng thông minh Helloworld và tương tác với chúng bằng commandline interface . trong bài học này mình giới thiệu cho các bạn một bài học thú vị hơn đó là tương tác với hợp đồng này bằng giao diện frontend.Oke chúng ta cùng bắt đầu bài học.

Trong bài học này gồm những nội dung chính sau. Thứ nhất,chúng ta sẽ Tạo một dự ám nextjs cơ bản trong bài học này mình sẽ dung nextjs làm framework frontend chính để thực hiện tướng tác với hợp đồng thông minh. Thư Hai, Chúng ta sẽ thực hiện đọc validator từ hợp đồng thông minh và hiển thì ra màn hình sau đó. Mình sẽ thực hiện viết ra các chức năng để kết nối Ví. Dự án này chúng ta sẽ sử dụng Nami làm ví chính để thực hiện tương tác với hệ sinh thái Cardano. Sau đó mình sẽ nhập 1000 ada để lock vào hợp đồng thông minh Oke bây giờ mình sẽ nhấn lock . Mình sẽ kí một giao dịch mình sẽ kí . Như vậy là mình đã kí thành công giao dịch khi mà transaction Hash được in ra màn hình chứng tỏ quá trình lock tài sản của mình thành công . trong bài học này sẽ dung lucid và api blockfrost làm công cụ chính để tương chính với hợp đồng thông minh Như vậy transactionHash đã được in ra và mình đã bị lock thành công 1000 ada vào hợp đồng thông minh Tiếp đến mình sẽ thực hiện unlock 1000 ada vào hợp đồng thông minh để lấy lại 1000 ada vừa lock vào mình sẽ thực hiện kí giao dịch để unlock mình sẽ nhập mật khẩu để kí Khi transaction hash được in ra chứng to quá trinh unlock tài sản của chúng ta thành công. Như vậy để biết đên bài học này các bạn cần chắc kiến thực frontend.

Bây giờ mình sẽ kiểm tra xem giao dịch của mình như thế nào bằng cách nhấn vào từng transaction Hash Như khóa tài sản mình sẽ xem và tada quá trình giao dịch tài sản đã được ghi lại như vậy quá trình khóa tài sản vào hợp đồng thực hiện thành công bây giờ chúng ta sẽ thực hiện kiểm tra hợp đồng unlock tài sản bằng việc nhấn vào transaction Hash

Như vậy bài học chúng ta hôm nay sẽ thực hiệc các chức năng đó Bây giờ chúng ta sẽ bắt dầu bài học này. Trước tiên mình cần setup cài đặt môi trường cho dự án Trong bài này mình sẽ làm việc với Nextjs mà để làm việc với Nextjs thì mình phải có nodejs. Để kiểm tra nodejs có hay chưa chúng ta sẽ sử dụng Node -v để kiểm tra nếu các bạn chưa có NodeJs thì lên Google và search Nodejs để tải nodejs . Mình khuyên các bạn nên tải Nodejs cho người sử dụng vì nó đỡ lỗi hơn so với các bản thường. Như vậy mình đã có Nodejs rồi. À các bạn nên sử dung Nodejs phiên bản 16 trở lên vì phiên bản đó sẽ ổn định hơn. Đi cùng với Nodejs có 2 công cụ khá quan trong là Npx và Npm các package này sẽ được tải về cùng với NodeJs Để kiểm tra cái này mình sẽ sử dụng npm -v và npx -v để kiểm tra

Như vậy việc cài đặt và kiểm tra Nodejs đã thành công. Bây giờ hãy bắt đầu với bước thú vị hơn mình sẽ cài đặt nextjs về để cho dự án để cài nextjs mình sẽ sử dụng npx create-next-app + tên dự án trong dự án này mình sẽ đặt tên là Helloworld-frontend . Oke mình sẽ chọn Typescript làm ngôn ngữ để dev chính và mình sẽ sử dụng tailwind css để thực hiện styles cho dự án sau đó mình sẽ đợi để dự án được cài đặt.

Oke như vậy dự án đã được cài đặt thành công. Bây giờ mình sẽ thực hiện việc tải các thư viện cần thiết cho dự án . Trong dự án này minh sẽ sử dụng 2 thư viện quan trong là lucid cardano và cbor-x để install và biết cách sử dụng cbor-x mình sẽ lên trang lucid cardano nom và xem cách tải cũng như cách hoạt động và cbor-x cũng vậy như vậy mình sẽ thực hiện việc cài đặt lucid cardano bằng cách npm install –save lucid-cardano cbor-x. Như vậy quá trình cà đặt 2 thư viện này thành công. Bây giờ chúng ta sẽ chạy dự án của mình và thực hiện dev chúng như những gì mà chúng ta đã xây dựng ở đầu video. Bằng câu lệnh npm run dev Dự án của chúng ta được chạy ở PORT 3000 chúng ta sẽ chạy nó lên bằng câu localhost:3000. Như vậy dự án của mình được chạy thành công

Tiêp đến hay thực hiện đọc các validator từ hợp đồng thông minh helloworld . Để đọc validator chúng ta sẽ copy file plutus.json để thực hiện đọc file này File này là các tài nguyên để tương tác với onchain minh sẽ tạo một folder tên là libs để lưu file plustus.json tương tác và lưu file plutus.json vào trong đó. Đồng thời mình sẽ tạo folder utils để thực hiện đọc validator trong file Read validator

import { encode } from "cbor-x"

import { SpendingValidator, fromHex, toHex } from "lucid-cardano"

import helloWorld from "~/libs/helloworld.json"

const readValidator = function(): SpendingValidator {

    const helloWorldValidator = helloWorld.validators.find(function(validator) {

        return validator.title === "contract.hello\_world"

    })

    if(!helloWorldValidator) {

        throw new Error("Hello world validator not found.")

    }

    const helloWorldScript: string = toHex(encode(fromHex(helloWorldValidator.compiledCode)))

    return {

        type: "PlutusV2",

        script: helloWorldScript,

    }

}

export default readValidator

Như vậy trong đây mình đã đọc thành công validator, Bây giờ mình sẽ thực hiện Việc hiển thị các validator ra màn hình . Oke được rồi chúng ta cần làm nha Bây giờ chúng ta cần vào file file app.tsx để thực hiện đọc validator đã tạo từ hợp đồng thông minh. Oke chúng ta cũng thực hiện nhé. Bây giờ chúng ta sẽ thực hiện import validator from Validor sau đó chúng ta sẽ in validator ra màn hình. Mình cũng xem kết quả nhé.

Được rồi như vậy quá trinh invalidator ra màn hình được thực hiện thành công . Bây giừ chúng ta thấy nó khá xấu vì vậy hãy cùng mình custom lại giao diện một chút nhé. Mình sẽ thực hiện xóa một và cái không cần thiết và để theo tailwind css bây giờ chúng ta cùng viết code giao diện để viết được những cái này các bạn cần phải có kiến thức cơ bản như html css cũng như react

import React from "react";

import readValidator from "~/utils/read-validators";

import HelloWorld from "~/components/HelloWorld";

export default function Home() {

    const validators = readValidator();

    return (

        <main>

            <div className="max-w-2xl mx-auto mt-20 mb-10">

                <div className="mb-10">

                    <h2 className="text-lg font-semibold text-gray-900">

                        Make a lock and un lock tada contract

                    </h2>

                    <h3 className="mt-4 mb-2">Hello world</h3>

                    <div className="bg-gray-200 p-2 rounded overflow-x-scroll">

                        {validators.script}

                    </div>

                </div>

                <HelloWorld />

            </div>

        </main>

    );

}

Chúng ta sẽ thực hiện kiểm tra các validator xem nó đã đẹp chưa Oke nhìn khá ổn ấy. Tiếp đến chúng ta sẽ thực hiện connect wallet . Mình sẽ sử dụng lucid và api của blockfrost để tương tác và kết nối với ví trong hướng dẫn này mình sẽ dung nami là ví chính để thực hiện tương tác với mạng lưới. Oke minh sẽ tạo một thư mục context trong có provider và các components để thực hiện tương tác trong các component này mình sẽ tạo một folder types để thực hiện định nghĩa các dự liệu cho đúng mình sẽ tạo file có tên là Lucid context type.d.ts để thực hiện định nghĩa type

import { Lucid } from "lucid-cardano";

export type LucidContextType = {

    lucid: Lucid;

    setLucid: React.Dispatch<React.SetStateAction<Lucid>>

}

Bây giờ mình sẽ tạo lucid context và create chúng

"use client"

import React, { createContext } from "react";

import { LucidContextType } from "~/types/contexts/LucidContextType";

const LucidContext = createContext<LucidContextType>(null!);

export default LucidContext;

Đồng thời với context mình sẽ sử dụng Provider

"use client";

import React, { ReactNode, useState } from "react";

import { Lucid } from "lucid-cardano";

import LucidContext from "~/contexts/components/LucidContext";

type Props = {

    children: ReactNode;

};

const LucidProvider = function ({ children }: Props) {

    const [lucid, setLucid] = useState<Lucid>(null!);

    return (

        <LucidContext.Provider

            value={{

                lucid,

                setLucid,

            }}

        >

            {children}

        </LucidContext.Provider>

    );

};

export default LucidProvider;

"use client";

import React, { ReactNode, lazy } from "react";

const LucidProvider = lazy(() => import("~/contexts/providers/LucidProvider"));

const WalletProvider = lazy(

    () => import("~/contexts/providers/WalletProvider")

);

const SmartContractProvider = lazy(

    () => import("~/contexts/providers/SmartContractProvider")

);

type Props = {

    children: ReactNode;

};

const ContextProvider = function ({ children }: Props) {

    return (

        <LucidProvider>

            <WalletProvider>

                <SmartContractProvider>{children}</SmartContractProvider>

            </WalletProvider>

        </LucidProvider>

    );

};

export default ContextProvider;

export type WalletContextType = {

    connect: () => Promise<void>,

    disconnect: () => Promise<void>,

}

"use client"

import React, { createContext } from "react";

import { WalletContextType } from "~/types/contexts/WalletContextType";

const WalletContext = createContext<WalletContextType>(null!);

export default WalletContext;

"use client";

import React, { ReactNode, useContext, useState } from "react";

import { Blockfrost, Lucid } from "lucid-cardano";

import WalletContext from "~/contexts/components/WalletContext";

import LucidContext from "~/contexts/components/LucidContext";

import { LucidContextType } from "~/types/contexts/LucidContextType";

type Props = {

    children: ReactNode;

};

const WalletProvider = function ({ children }: Props) {

    const { setLucid } = useContext<LucidContextType>(LucidContext);

    const connect = async function () {

        try {

            const lucid: Lucid = await Lucid.new(

                new Blockfrost(

                    process.env.BLOCKFROST\_PROPROD\_RPC\_URL as string,

                    process.env.BLOCKFROST\_PROPROD\_API as string

                ),

                "Preprod"

            );

            lucid.selectWallet(await window.cardano.nami.enable());

            setLucid(lucid);

        } catch (error) {

            console.log(error);

        }

    };

    const disconnect = async function () {

        try {

            setLucid(null!);

        } catch (error) {

            console.log(error);

        }

    };

    return (

        <WalletContext.Provider value={{connect,disconnect}}>

            {children}

        </WalletContext.Provider>

    );

};

export default WalletProvider;