To structure a smart contract on the Cardano platform, following the defined parameters, we can conceptualize the process as follows:

1. Definition of the Contract and Main Variables

- Applicant address: The person requesting the creation of the NFT.

- Receiver Address: The person who authorizes the payment and receives the funds.

- Payment amount: The amount of ADA to be transferred for the creation of the NFT.

- Title and notes: Information that will be stored in the NFT.

- Transaction hash: The hash that represents the transaction in the blockchain.

2. Main Components of the Contract

to. NFT Creation Request

- The applicant initiates an application by providing the title, notes, payment amount and recipient's address.

- The contract verifies that the payment amount is correct and stores the NFT data in a temporary state.

b. Authorization and Making Payment

- The payment recipient reviews the request and digitally signs the transaction to authorize the payment.

- The contract validates the recipient's signature to guarantee authorization.

c. Registration on the Blockchain

- Once payment is authorized, the contract transfers the funds to the recipient.

- The NFT data (title, notes, recipient address) is recorded on the blockchain.

- A hash of the transaction is generated, which is stored along with the NFT data.

3. Conceptual Diagram

```
+--------------------------------------+
|         Contrato Inteligente         |
+--------------------------------------+
|                                      |
|  1. Solicitud de Creación del NFT    |
|  - Dirección del solicitante         |
|  - Dirección del receptor            |
|  - Monto del pago                    |
|  - Título y notas                    |
|                                      |
|  2. Autorización y Realización del Pago |
|  - Validación de la firma del receptor|
|  - Transferencia de fondos            |
|                                      |
|  3. Registro en la Cadena de Bloques |
|  - Almacenamiento de datos del NFT   |
|  - Generación del hash de transacción|
|                                      |
+--------------------------------------+
```

4. Smart Contract Steps

- Creation Request

• The requester calls the smart contract and provides the necessary details (title, notes, payment amount, recipient address).

• The contract verifies that the payment amount is correct.

- Payment Authorization

• The recipient reviews the request and digitally signs to authorize payment.

• The contract validates the recipient's signature to ensure that the authorization is valid.

- Payment and Registration

• Once authorized, the contract transfers the funds to the recipient.

• The NFT data (title, notes, recipient address) is recorded on the blockchain.

• A hash of the transaction is generated and stored.

5. Hash Code and Registration on the Blockchain

• The transaction hash is generated using a cryptographic hash function (such as SHA-256) applied to the transaction data.

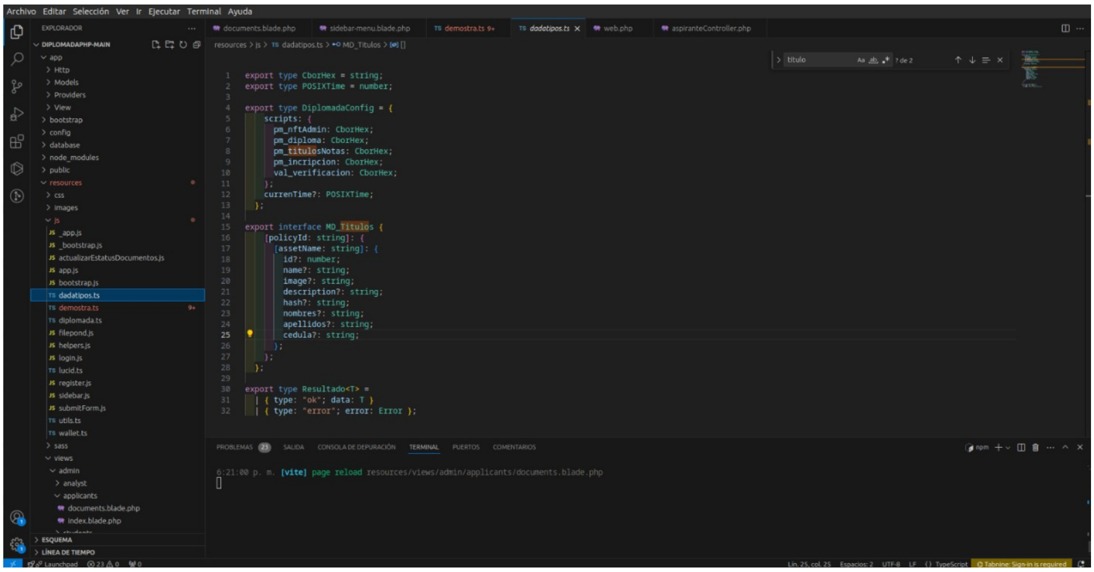
• The hash is stored alongside the NFT data on the blockchain, providing a unique and secure reference for the transaction.

6. Implementation in Plutus (Conceptual)

Although not complete Plutus code, the following structure provides an idea of what the implementation might look like:

```
{-# LANGUAGE DataKinds           #-}
{-# LANGUAGE DeriveAnyClass      #-}
{-# LANGUAGE DeriveGeneric       #-}
{-# LANGUAGE FlexibleContexts    #-}
{-# LANGUAGE NoImplicitPrelude   #-}
{-# LANGUAGE OverloadedStrings   #-}
{-# LANGUAGE ScopedTypeVariables #-}
{-# LANGUAGE TemplateHaskell     #-}
{-# LANGUAGE TypeApplications    #-}
{-# LANGUAGE TypeFamilies        #-}
{-# LANGUAGE TypeOperators       #-}

module CertifiedTitleNFT where

import           Plutus.V1.Ledger.Api
import           Plutus.V1.Ledger.Contexts
import           Plutus.V1.Ledger.Scripts
import           Plutus.V1.Ledger.Value
import           PlutusTx
import           PlutusTx.Prelude          hiding (Semigroup(..), unless)
import           Ledger                    hiding (singleton)
import           Ledger.Constraints        as Constraints
import           Ledger.Typed.Scripts      as Scripts
import           Playground.Contract
import           Prelude                    (Semigroup (..), Show (..), String)

data NFTParams = NFTParams
{ npTitle         :: !String
, npNotes         :: !String
, npRecipient     :: !PubKeyHash
, npPaymentAmount :: !Integer
} deriving (Generic, ToJSON, FromJSON, ToSchema)

data NFTDatum = NFTDatum
{ ndTitle     :: !String
, ndNotes     :: !String
, ndRecipient :: !PubKeyHash
, ndHash      :: !BuiltinByteString
} deriving (Show, Generic, ToJSON, FromJSON)

PlutusTx.makeLift ''NFTParams
PlutusTx.unstableMakeIsData ''NFTDatum

{-# INLINABLE mkNFTValidator #-}
mkNFTValidator :: NFTDatum -> () -> ScriptContext -> Bool
mkNFTValidator datum _ ctx =
traceIfFalse "Recipient's signature missing" signedByRecipient &&
traceIfFalse "Incorrect payment amount" correctPayment
where
info :: TxInfo
info = scriptContextTxInfo ctx

signedByRecipient :: Bool
signedByRecipient = txSignedBy info $ ndRecipient datum

correctPayment :: Bool
correctPayment  =  valuePaidTo  info  (ndRecipient  datum)  ==  Ada.lovelaceValueOf
(npPaymentAmount params)

data NFT
instance Scripts.ValidatorTypes NFT where
type instance DatumType NFT = NFTDatum
type instance RedeemerType NFT = ()

typedNFTValidator :: Scripts.TypedValidator NFT
typedNFTValidator = Scripts.mkTypedValidator @NFT
$$(PlutusTx.compile [|| mkNFTValidator ||])
$$(PlutusTx.compile [|| wrap ||])
where
wrap = Scripts.wrapValidator @NFTDatum @()

validator :: Validator
validator = Scripts.validatorScript typedNFTValidator

valHash :: Ledger.ValidatorHash
valHash = Scripts.validatorHash typedNFTValidator

scrAddress :: Ledger.Address
scrAddress = scriptAddress validator

createNFT :: NFTParams -> Contract w s Text ()
createNFT params = do
let datum = NFTDatum
{ ndTitle     = npTitle params
, ndNotes     = npNotes params
, ndRecipient = npRecipient params
, ndHash      = sha2_256 $ BuiltinByteString $ npTitle params ++ npNotes params
}
```

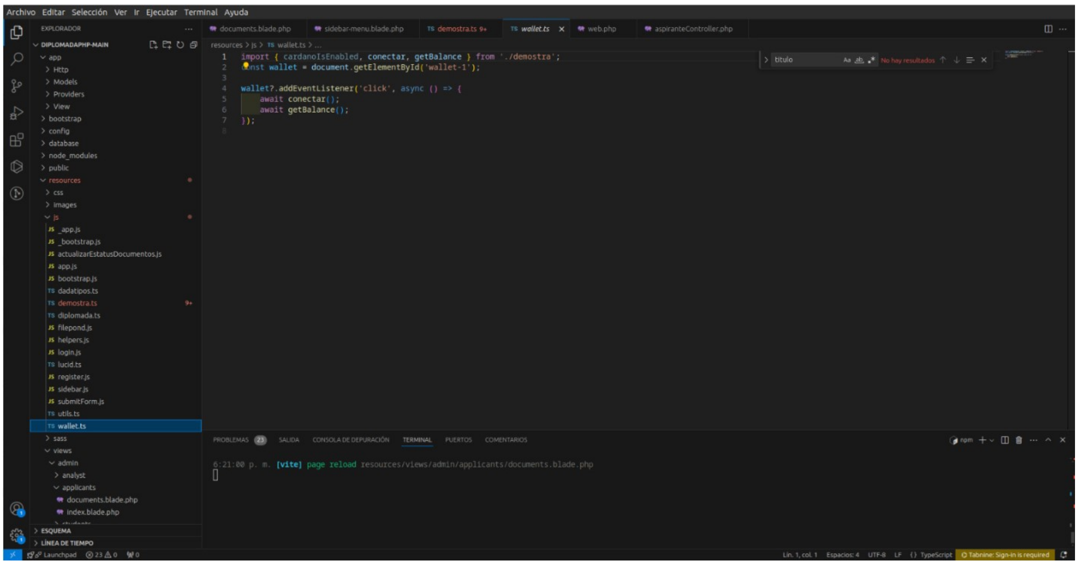
```
tx = Constraints.mustPayToTheScript datum $ Ada.lovelaceValueOf (npPaymentAmount
params)
ledgerTx <- submitTxConstraints typedNFTValidator tx
awaitTxConfirmed $ getCardanoTxId ledgerTx
logInfo @String $ "NFT created with title: " ++ npTitle params

endpoints :: Contract () NFTSchema Text ()
endpoints = createNFT'>> endpoints
where
createNFT' = endpoint @"createNFT">>= createNFT

type NFTSchema = Endpoint "createNFT" NFTParams

mkSchemaDefinitions ''NFTSchema

mkKnownCurrencies []
```



resources > js > dadatipos.ts > MD_Titulos > []

```
export type CborHex = string;
export type POSIXTime = number;

export type DiplomadaConfig = {
    scripts: {
      pm_nftAdmin: CborHex;
      pm_diploma: CborHex;
      pm_titulosNotas: CborHex;
      pm_incripcion: CborHex;
      val_verificacion: CborHex;
    };
    currenTime?: POSIXTime;
  };

export interface MD_Titulos {
    [policyId: string]: {
      [assetName: string]: {
        id?: number;
        name?: string;
        image?: string;
        description?: string;
        hash?: string;
        nombres?: string;
        apellidos?: string;
        cedula?: string;
      };
    };
  };

export type Resultado<T> =
  | { type: "ok"; data: T }
  | { type: "error"; error: Error };
```

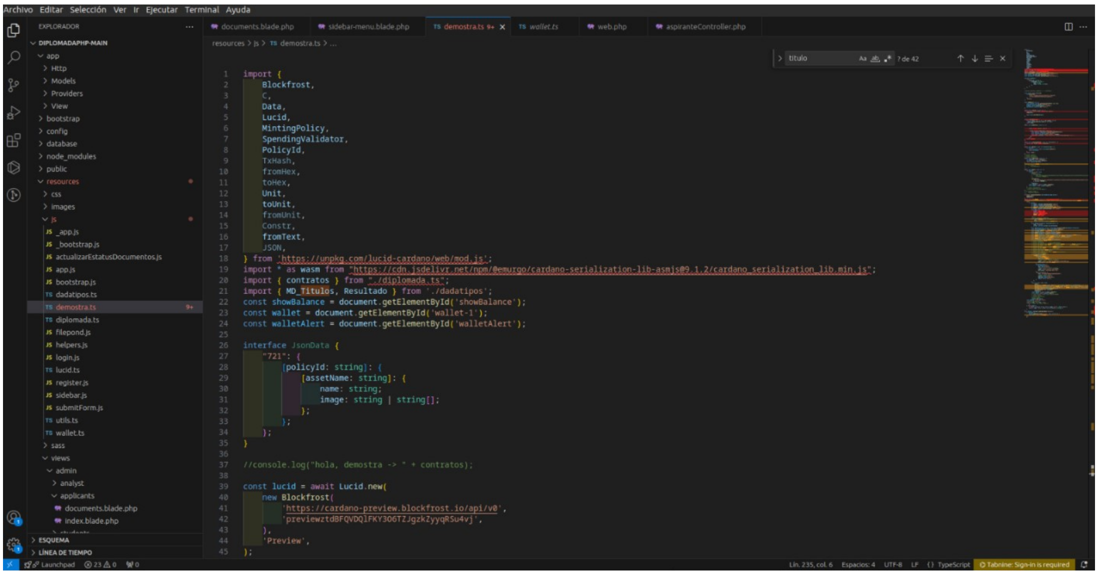
```
6:21:00 p. m. [vite] page reload resources/views/admin/applicants/documents.blade.php
```



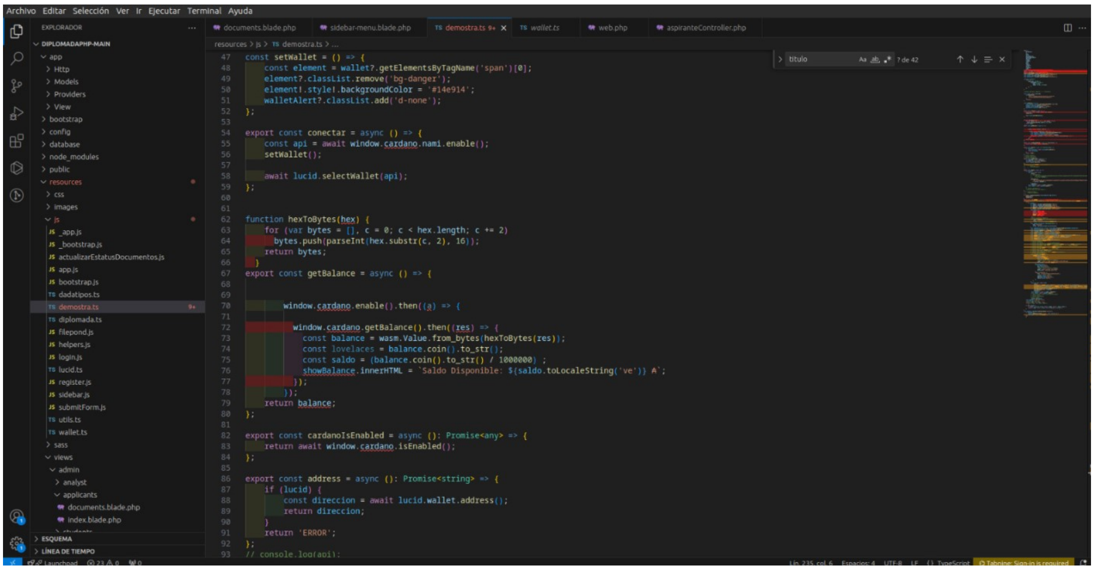
resources > js > wallet.ts > ...

```
import { cardanoIsEnabled, conectar, getBalance } from './demostra';
const wallet = document.getElementById('wallet-1');

wallet?.addEventListener('click', async () => {
    await conectar();
    await getBalance();
});
```

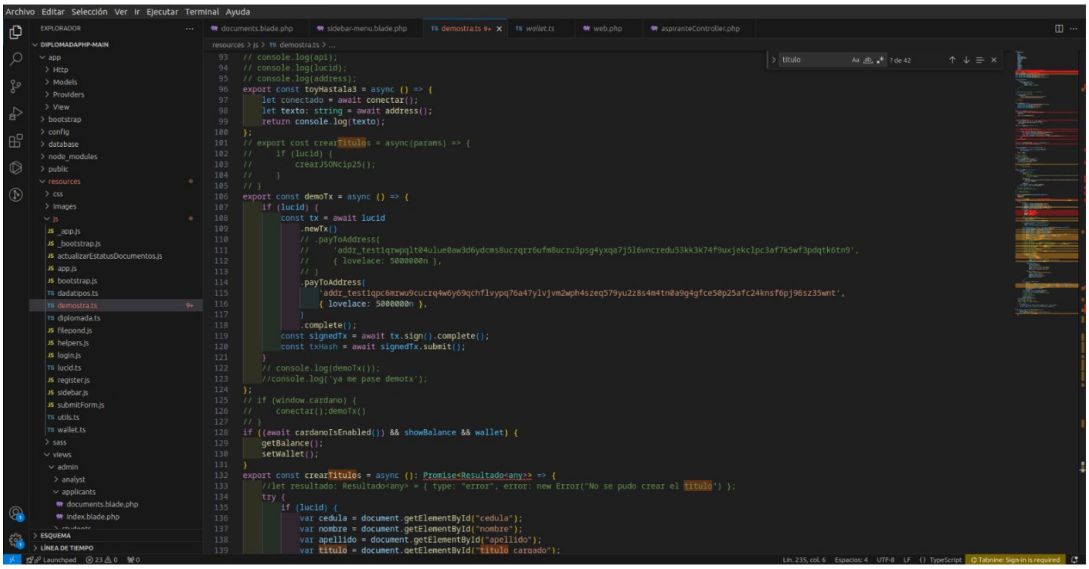
```
6:21:00 p. m. [vite] page reload resources/views/admin/applicants/documents.blade.php
```



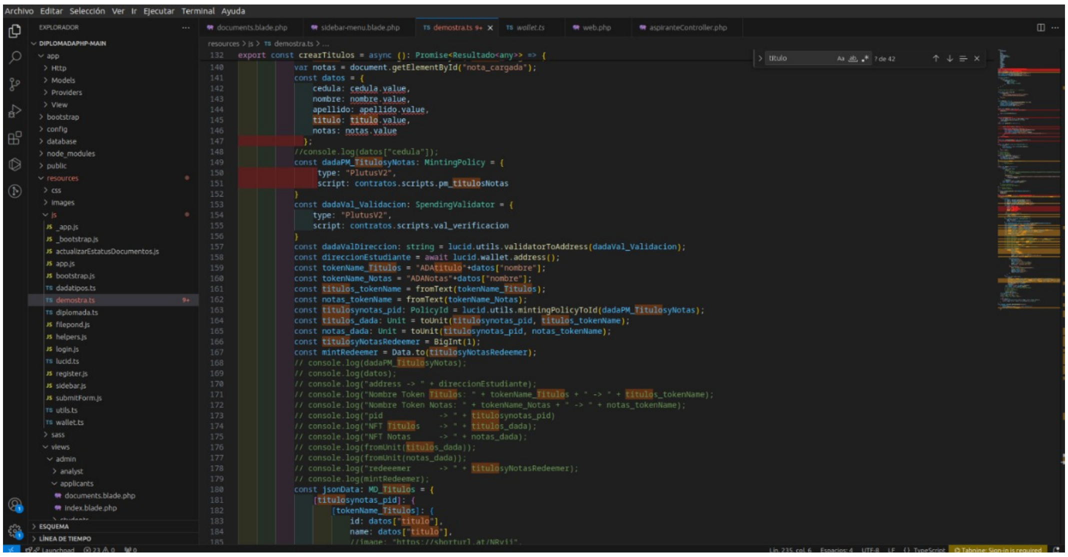
```
import {
    Blockfrost,
    C,
    Data,
    Lucid,
    MintingPolicy,
    SpendingValidator,
    PolicyId,
    TxHash,
    fromHex,
    toHex,
    Unit,
    toUnit,
    fromUnit,
    Constr,
    fromText,
    JSON,
} from 'https://unpkg.com/lucid-cardano/web/mod.js';
import * as wasm from "https://cdn.jsdelivr.net/npm/@emurgo/cardano-serialization-lib-asmjs@9.1.2/cardano_serialization_lib.min.js";
import { contratos } from "./diplomada.ts";
import { MD_Titulos, Resultado } from './dadatipos';
const showBalance = document.getElementById('showBalance');
const wallet = document.getElementById('wallet-1');
const walletAlert = document.getElementById('walletAlert');

interface JsonData {
    "721": {
        [policyId: string]: {
            [assetName: string]: {
                name: string;
                image: string | string[];
            };
        };
    };
}

//console.log("hola, demostra -> " + contratos);

const lucid = await Lucid.new(
    new Blockfrost(
        'https://cardano-preview.blockfrost.io/api/v0',
        'previewztdBFQVDQlFKY3O6TZJgzkZyyqRSu4vj',
    ),
    'Preview',
);
```



```
const setWallet = () => {
    const element = wallet?.getElementsByTagName('span')[0];
    element?.classList.remove('bg-danger');
    element!.style!.backgroundColor = '#14e914';
    walletAlert?.classList.add('d-none');
};

export const conectar = async () => {
    const api = await window.cardano.nami.enable();
    setWallet();

    await lucid.selectWallet(api);
};


function hexToBytes(hex) {
    for (var bytes = [], c = 0; c < hex.length; c += 2)
     bytes.push(parseInt(hex.substr(c, 2), 16));
    return bytes;
 }
export const getBalance = async () => {


        window.cardano.enable().then((a) => {

          window.cardano.getBalance().then((res) => {
            const balance = wasm.Value.from_bytes(hexToBytes(res));
            const lovelaces = balance.coin().to_str();
            const saldo = (balance.coin().to_str() / 1000000) ;
            showBalance.innerHTML = `Saldo Disponible: ${saldo.toLocaleString('ve')} ₳`;
          });
        });
    return balance;
};

export const cardanoIsEnabled = async (): Promise<any> => {
    return await window.cardano.isEnabled();
};

export const address = async (): Promise<string> => {
    if (lucid) {
        const direccion = await lucid.wallet.address();
        return direccion;
    }
    return 'ERROR';
};
// console.log(api);
```

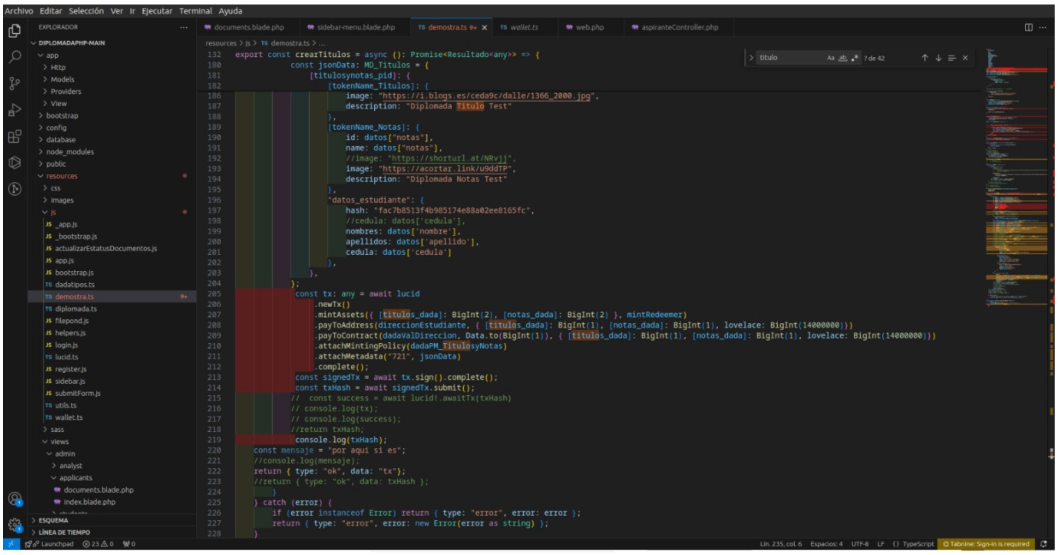


```
// console.log(api);
// console.log(lucid);
// console.log(address);
export const toyHastala3 = async () => {
    let conectado = await conectar();
    let texto: string = await address();
    return console.log(texto);
};
// export cost crearTitulos = async(params) => {
//     if (lucid) {
//         crearJSONcip25();
//     }
// }
export const demoTx = async () => {
    if (lucid) {
        const tx = await lucid
            .newTx()
            // .payToAddress(
            //     'addr_test1qrwpqlt04ulue0aw3d6ydcms8uczqrr6ufm8ucru3psg4yxqa7j5l6vncredu53kk3k74f9uxjekclpc3af7k5wf3pdqtk6tn9',
            //     { lovelace: 5000000n },
            // )
            .payToAddress(
                'addr_test1qpc6mrwu9cucrq4w6y69qchflvypq76a47ylvjvm2wph4szeq579yu2z8s4m4tn0a9g4gfce50p25afc24knsf6pj96sz35wnt',
                { lovelace: 5000000n },
            )
            .complete();
        const signedTx = await tx.sign().complete();
        const txHash = await signedTx.submit();
    }
    // console.log(demoTx());
    //console.log('ya me pase demotx');
};
// if (window.cardano) {
//     conectar();demoTx()
// }
if ((await cardanoIsEnabled()) && showBalance && wallet) {
    getBalance();
    setWallet();
}
export const crearTitulos = async (): Promise<Resultado<any>> => {
    //let resultado: Resultado<any> = { type: "error", error: new Error("No se pudo crear el titulo") };
    try {
        if (lucid) {
            var cedula = document.getElementById("cedula");
            var nombre = document.getElementById("nombre");
            var apellido = document.getElementById("apellido");
            var titulo = document.getElementById("titulo cargado");
```



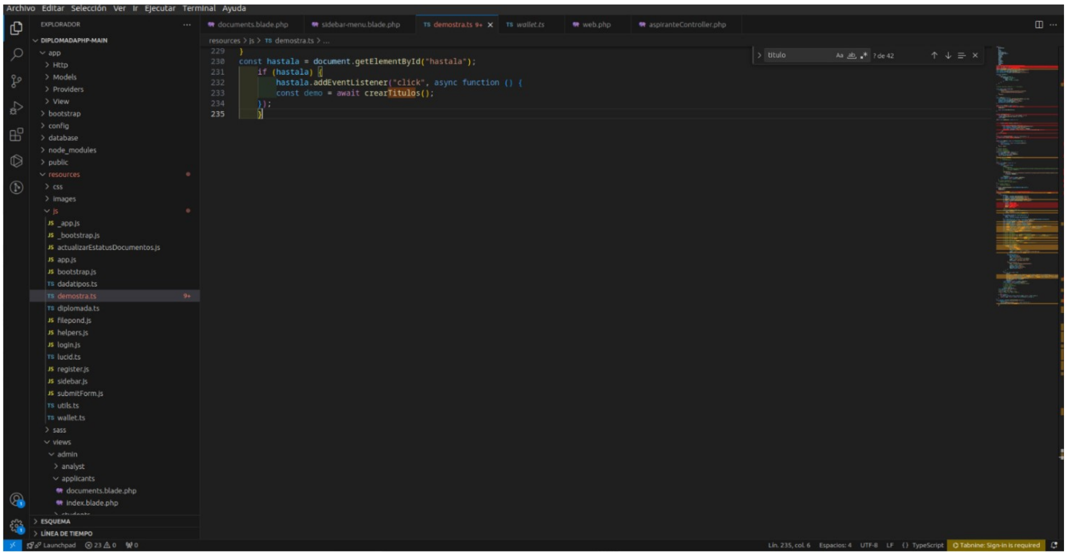
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
documents.blade.php
sidebar-menu.blade.php
demostra.ts 9+
wallet.ts
web.php
aspiranteController.php
resources > js > demostra.ts > ...
titulo
? de 42
Lín. 235, col. 6 Espacios: 4 UTF-8 LF {} TypeScript
Tabnine: Sign-in is required

```
export const crearTitulos = async (): Promise<Resultado<any>> => {
            var notas = document.getElementById("nota_cargada");
            const datos = {
                cedula: cedula.value,
                nombre: nombre.value,
                apellido: apellido.value,
                titulo: titulo.value,
                notas: notas.value
            };
            //console.log(datos["cedula"]);
            const dadaPM_TitulosyNotas: MintingPolicy = {
                type: "PlutusV2",
                script: contratos.scripts.pm_titulosNotas
            }
            const dadaVal_Validacion: SpendingValidator = {
                type: "PlutusV2",
                script: contratos.scripts.val_verificacion
            }
            const dadaValDireccion: string = lucid.utils.validatorToAddress(dadaVal_Validacion);
            const direccionEstudiante = await lucid.wallet.address();
            const tokenName_Titulos = "ADAtitulo"+datos["nombre"];
            const tokenName_Notas = "ADANotas"+datos["nombre"];
            const titulos_tokenName = fromText(tokenName_Titulos);
            const notas_tokenName = fromText(tokenName_Notas);
            const titulosynotas_pid: PolicyId = lucid.utils.mintingPolicyToId(dadaPM_TitulosyNotas);
            const titulos_dada: Unit = toUnit(titulosynotas_pid, titulos_tokenName);
            const notas_dada: Unit = toUnit(titulosynotas_pid, notas_tokenName);
            const titulosyNotasRedeemer = BigInt(1);
            const mintRedeemer = Data.to(titulosyNotasRedeemer);
            // console.log(dadaPM_TitulosyNotas);
            // console.log(datos);
            // console.log("address -> " + direccionEstudiante);
            // console.log("Nombre Token Titulos: " + tokenName_Titulos + " -> " + titulos_tokenName);
            // console.log("Nombre Token Notas: " + tokenName_Notas + " -> " + notas_tokenName);
            // console.log("pid            -> " + titulosynotas_pid)
            // console.log("NFT Titulos    -> " + titulos_dada);
            // console.log("NFT Notas      -> " + notas_dada);
            // console.log(fromUnit(titulos_dada));
            // console.log(fromUnit(notas_dada));
            // console.log("redeeemer      -> " + titulosyNotasRedeemer);
            // console.log(mintRedeemer);
            const jsonData: MD_Titulos = {
                [titulosynotas_pid]: {
                    [tokenName_Titulos]: {
                        id: datos["titulo"],
                        name: datos["titulo"],
                        //image: "https://shorturl.at/NRvjj",
```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
documents.blade.php
sidebar-menu.blade.php
demostra.ts 9+
wallet.ts
web.php
aspiranteController.php
resources > js > demostra.ts > ...
titulo
? de 42
Lín. 235, col. 6 Espacios: 4 UTF-8 LF {} TypeScript
Tabnine: Sign-in is required

```
export const crearTitulos = async (): Promise<Resultado<any>> => {
            const jsonData: MD_Titulos = {
                [titulosynotas_pid]: {
                    [tokenName_Titulos]: {
                        image: "https://i.blogs.es/ceda9c/dalle/1366_2000.jpg",
                        description: "Diplomada Titulo Test"
                    },
                    [tokenName_Notas]: {
                        id: datos["notas"],
                        name: datos["notas"],
                        //image: "https://shorturl.at/NRvjj",
                        image: "https://acortar.link/u9ddTP",
                        description: "Diplomada Notas Test"
                    },
                    "datos_estudiante": {
                        hash: "fac7b8513f4b985174e88a02ee8165fc",
                        //cedula: datos['cedula'],
                        nombres: datos['nombre'],
                        apellidos: datos['apellido'],
                        cedula: datos['cedula']
                    },
                },
            };
            const tx: any = await lucid
                .newTx()
                .mintAssets({ [titulos_dada]: BigInt(2), [notas_dada]: BigInt(2) }, mintRedeemer)
                .payToAddress(direccionEstudiante, { [titulos_dada]: BigInt(1), [notas_dada]: BigInt(1), lovelace: BigInt(14000000)})
                .payToContract(dadaValDireccion, Data.to(BigInt(1)), { [titulos_dada]: BigInt(1), [notas_dada]: BigInt(1), lovelace: BigInt(14000000)})
                .attachMintingPolicy(dadaPM_TitulosyNotas)
                .attachMetadata("721", jsonData)
                .complete();
            const signedTx = await tx.sign().complete();
            const txHash = await signedTx.submit();
            //  const success = await lucid!.awaitTx(txHash)
            // console.log(tx);
            // console.log(success);
            //return txHash;
            console.log(txHash);
    const mensaje = "por aqui si es";
    //console.log(mensaje);
    return { type: "ok", data: "tx"};
    //return { type: "ok", data: txHash };
        }
    } catch (error) {
        if (error instanceof Error) return { type: "error", error: error };
        return { type: "error", error: new Error(error as string) };
    }
```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
documents.blade.php
sidebar-menu.blade.php
demostra.ts 9+
wallet.ts
web.php
aspiranteController.php
resources > js > demostra.ts > ...
titulo
? de 42
Lín. 235, col. 6 Espacios: 4 UTF-8 LF {} TypeScript
Tabnine: Sign-in is required

```
}
const hastala = document.getElementById("hastala");
    if (hastala) {
        hastala.addEventListener("click", async function () {
        const demo = await crearTitulos();
    });
    }
```