Curso de

DApps: Introducción al Desarrollo de Aplicaciones Descentralizadas

Ernesto García Nevares

@ernestognw

Tecnológico de Monterrey

BLOCKCHAIN ACADEMY

yotepresto.

Google

zenfi

JavaScript in Plain English

¿Qué aprenderás en este curso?

- Arquitectura de aplicaciones descentralizadas.
- Uso de herramientas para desarrollar contratos inteligentes.
- Estructura de un proyecto para *EVM compatible chains*.
- Estándares de desarrollo más importantes.
- Despliegue de tu proyecto.

Objetivos

- Incrementar tus **habilidades** como desarrollador de software para que colabores con proyectos y protocolos en blockchains compatibles con la **EVM.**
- **Crear un proyecto para tu portafolio** que te abrirá las puertas para trabajar en los protocolos más importantes en el ecosistema Blockchain.
- **Desarrollar** tu capacidad crítica para evaluar los riesgos de centralización en aplicaciones descentralizadas.

No contempla...

- Dar recomendaciones de inversión.
- Estudiar los economics de tu proyecto.

Conocimientos previos

- Fundamentos de blockchain (protocolos de consenso, contratos inteligentes).
- Fundamentos de Solidity.
- Conocimientos de redes de prueba (Rinkeby, Ropsten).
- Javascript (Node.js).
- Teoría de desarrollo de aplicaciones (modelo cliente-servidor).

#BUIDL > #HODL

PlatziPunks:
Marketplace de NFT

El proyecto

PlatziPunks es un **marketplace** de **NFTs** creados a partir de **datos guardados completamente en la blockchain** y mostrados a través de una aplicación desplegada en un sistema de archivos descentralizado.

Créditos a Pablo Stanley 🎨

https://twitter.com/pablostanley

avataaars

Mix & Match Avatars with a Sketch library

Create avatar illustrations in Sketch App with this free library. Combine clothes, hair, emotions, accesories, and colors. Video 📺

Get the Libraaary

Designed by **Pablo Stanley** 🎨
Free for personal and commercial use. 😇
Get an email when there's an update. 📪
Try the **web editor** by Fang-Pen Lin

Tweet 👈🙏

Curso de DApps: Introducción al Desarrollo de Aplicaciones Descentralizadas

Curso de Desarrollo Frontend de Aplicaciones Descentralizadas con Web3.js



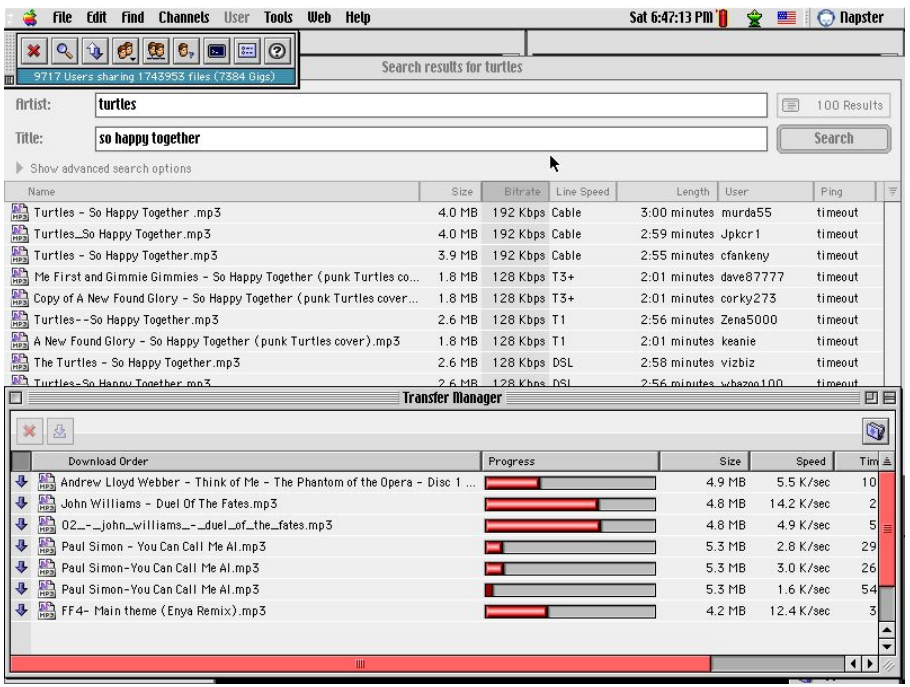




¿Qué es una **aplicación descentralizada?**

¿Alguien recuerda Napster?

- Los archivos eran compartidos P2P.
- No existía una autoridad central que validara los archivos.
- Cualquier persona con una copia del software podía compartir y descargar archivos.



DApp

Por sus siglas en inglés (Decentralized Application), se considera como una aplicación que posee la **mayor cantidad de componentes descentralizados.**

Características

Resistentes a la censura: no existe gobierno, empresa o autoridad que pueda censurar el acceso a una aplicación descentralizada.

Resilientes: debido a que la lógica de negocio está contenida en tecnologías P2P, es imposible detener su funcionamiento.

Transparentes: el uso de tecnología blockchain y de almacenamiento distribuido permite que todo su código, información y estado esté siempre disponible para consulta.

En los primeros días de Ethereum, la visión de los fundadores para las DApps iba más allá de **Smart Contracts**

Eventualmente estos se han convertido en un medio para la reinvención de la internet: **la web3.0**

Componentes descentralizables de una aplicación

Persistencia

Lógica

Cliente

Base de Datos

Almacenamiento

Lógica de Negocios

API

DNS

Web App

RTC

Web App

RTC

Web App

Aplicaciones tradicionales

Persistencia

aws

mongoDB.

Lógica

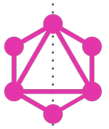
node JS

GO

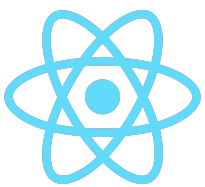
PYTHON

Cliente

{REST:API}



DNS
Domain Name System



Aplicaciones descentralizadas

Persistencia

Lógica

Cliente

JSON-RPC

Grados de descentralización

El frontend de mi aplicación no puede ser censurado.

La gobernanza de mi organización está descentralizada.

0%

100%

La lógica de negocios corre en un Smart Contract.

Los datos no están bajo control de los desarrolladores.

Grados de descentralización

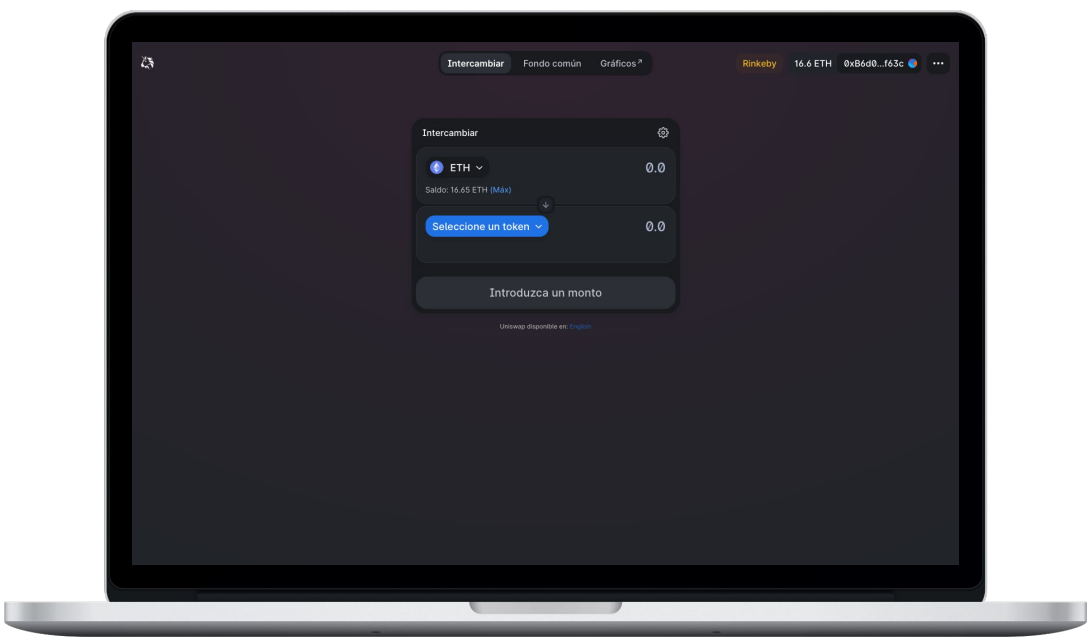
- ¿Qué tan descentralizado es el blockchain donde está mi Smart Contract?
- ¿Quiénes y bajo qué circunstancias pueden alterar el funcionamiento de un protocolo?
- ¿Los usuarios pueden usar mi frontend desde canales alternativos?
- ¿Pueden construir un frontend de código abierto?

Entre mayor sea la cantidad de componentes descentralizados de tu aplicación, estará más cerca del ideal de la web3

Ejemplos de **aplicaciones descentralizadas**

Uniswap

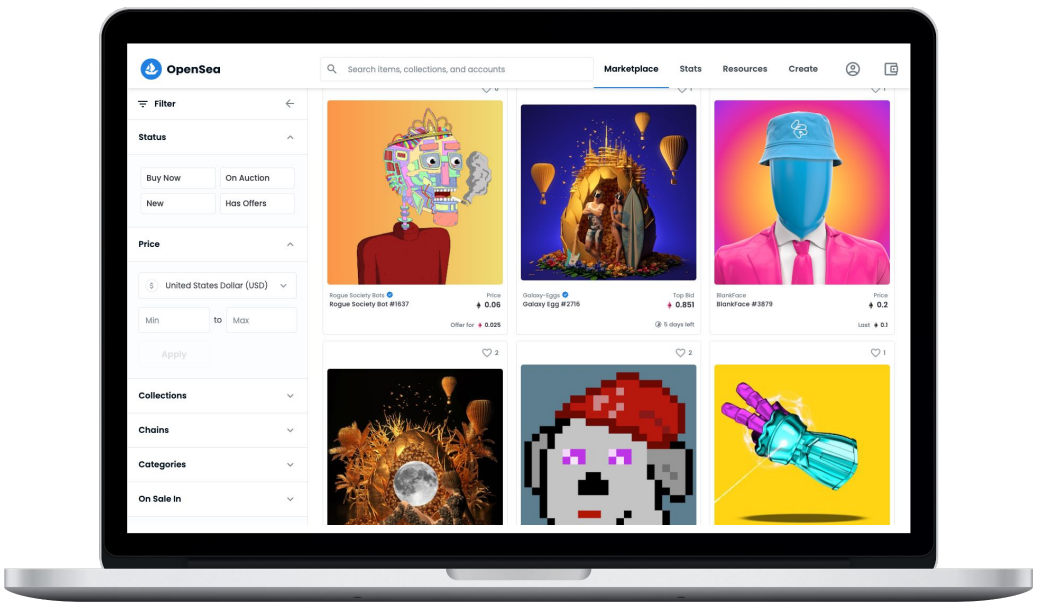
- ✅ Basado en Smart Contracts.
- ✅ Open source.
- ✅ Frontend Descentralizado por IPFS.
- ✅ Controlado a través de un token de gobernanza.
- ⚠️ Ya se han censurado tokens en su interfaz.



Opensea

OpenSea

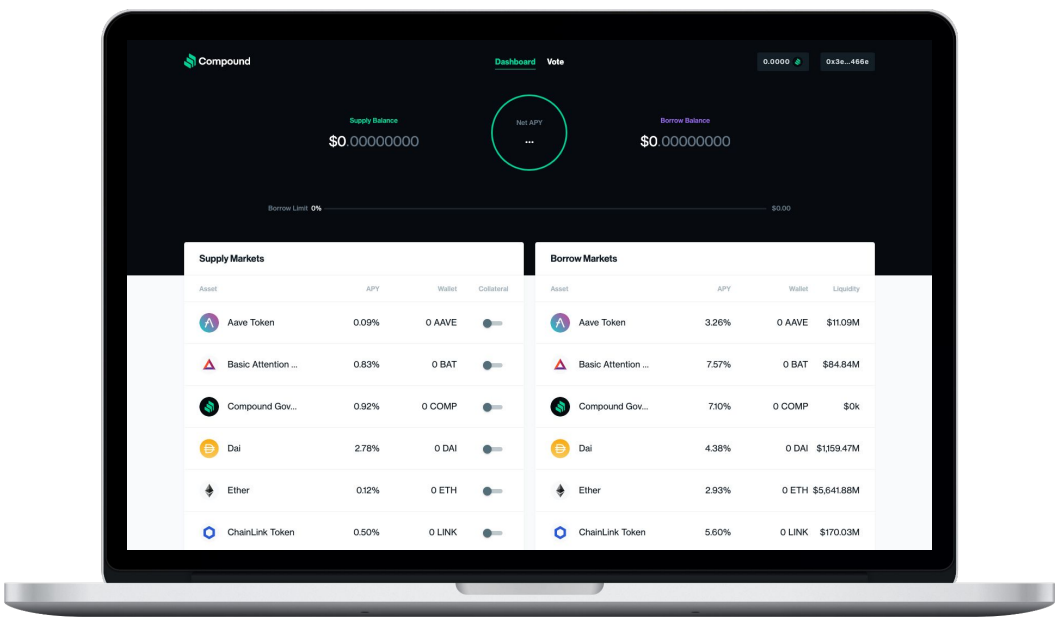
- ✅ Basado en Smart Contracts.
- ❌ Open source.
- Puedes subir un NFT a IPFS.
- ✅ Compatibilidad universal con estándares de NFT.
- ❌ Backend centralizado, podrían censurar NFT.



Compound

Compound

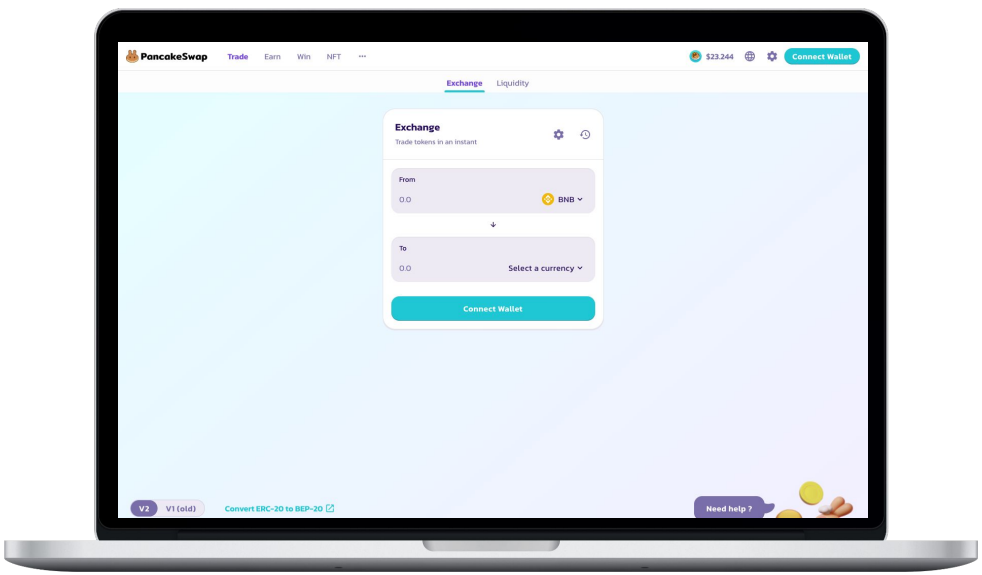
- ✅ Basado en Smart Contracts.
- ✅ Open source.
- ✅ Frontend Descentralizado por IPFS.
- ✅ Controlado a través de un token de gobernanza.



PancakeSwap

- 🤔 Basado en Smart Contracts en Binance Smart Chain.
- ✅ Open source.
- ❌ Frontend Descentralizado por IPFS.
- 🤔 Controlado a través de un token de gobernanza.

PancakeSwap

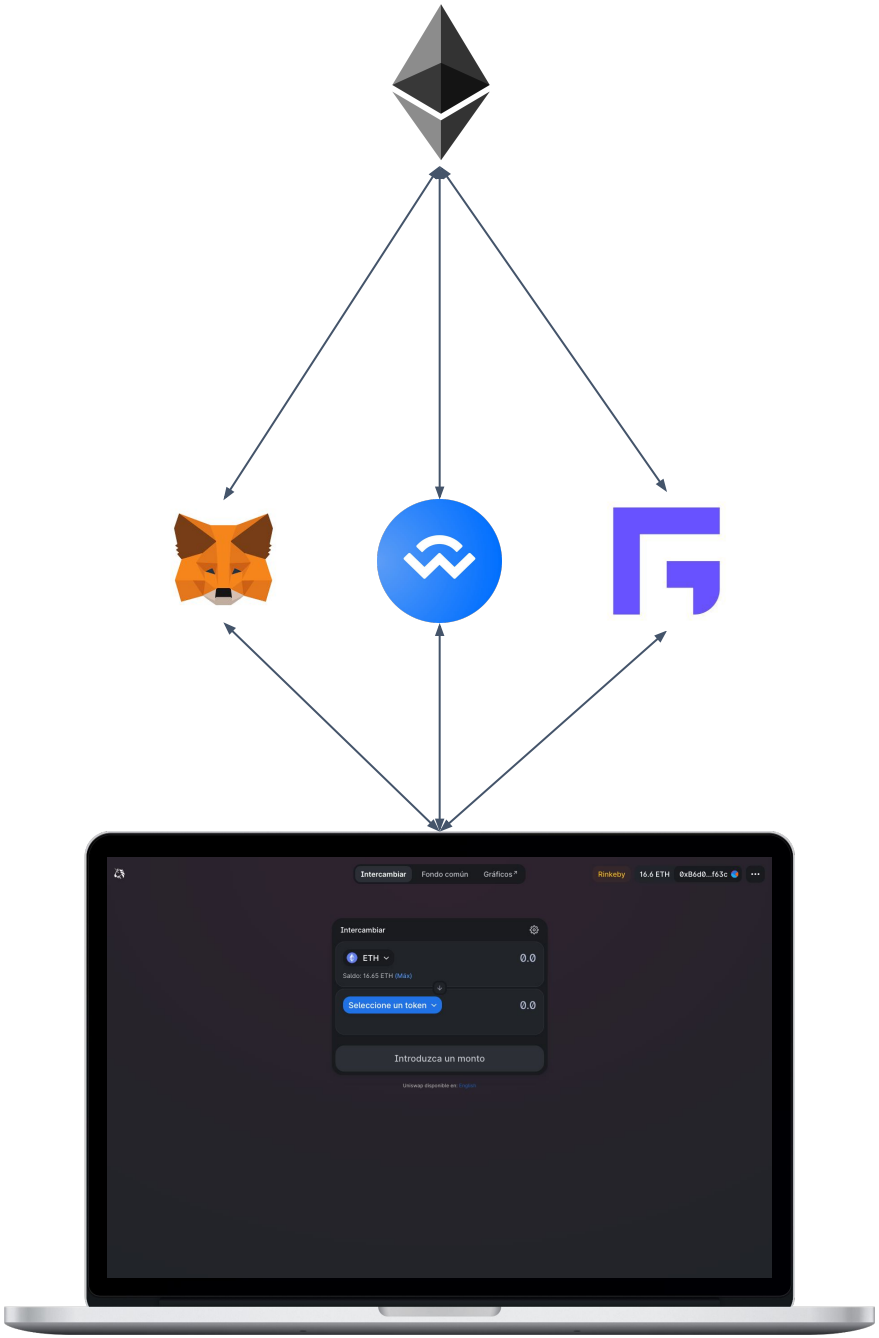


¿Qué otras aplicaciones descentralizadas conoces?

Déjalas en los comentarios

¿Cómo interactuar con **aplicaciones descentralizadas?**

Dado que las aplicaciones descentralizadas cuentan con Smart Contracts como **backend**, es necesario que las aplicaciones se **conecten a un nodo de Ethereum** (o cualquier otra blockchain) a través de un **proveedor.**



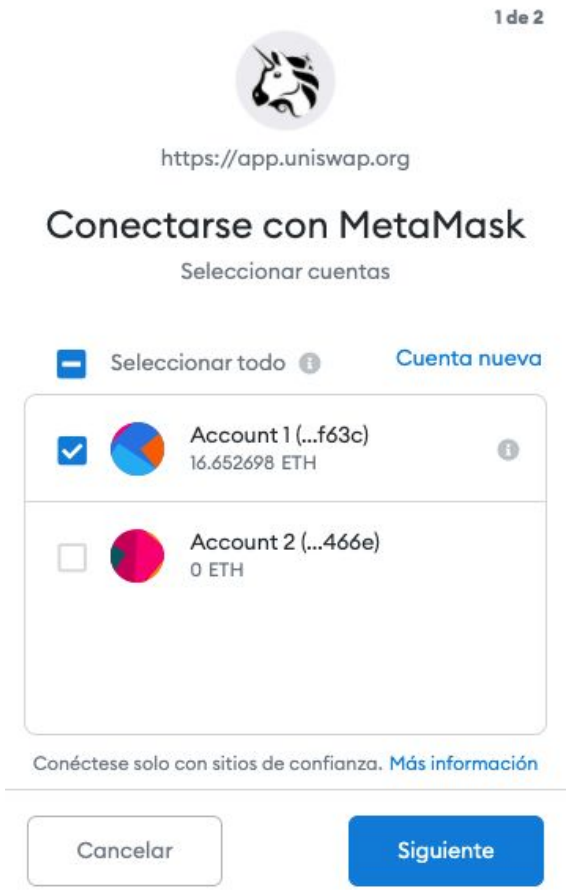
Aplicación

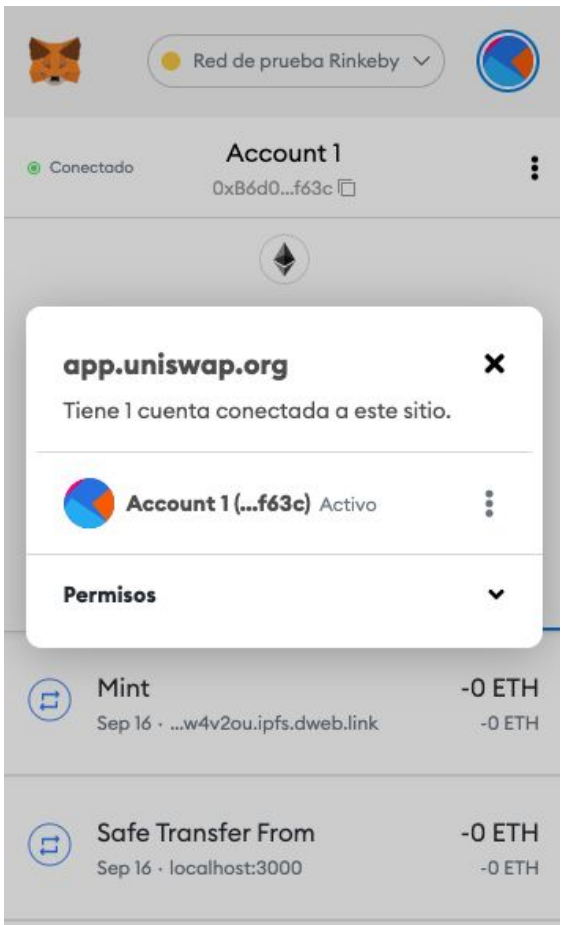
Proveedor

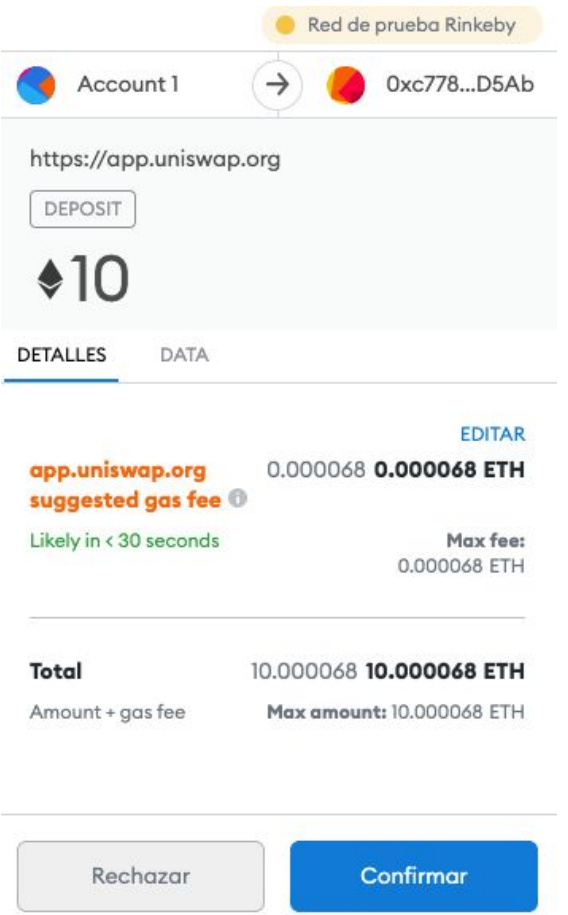
Blockchain

Los proveedores sirven como un puente entre tu aplicación y la blockchain. **Cada mensaje/transacción/función** va criptográficamente firmada por tu wallet.

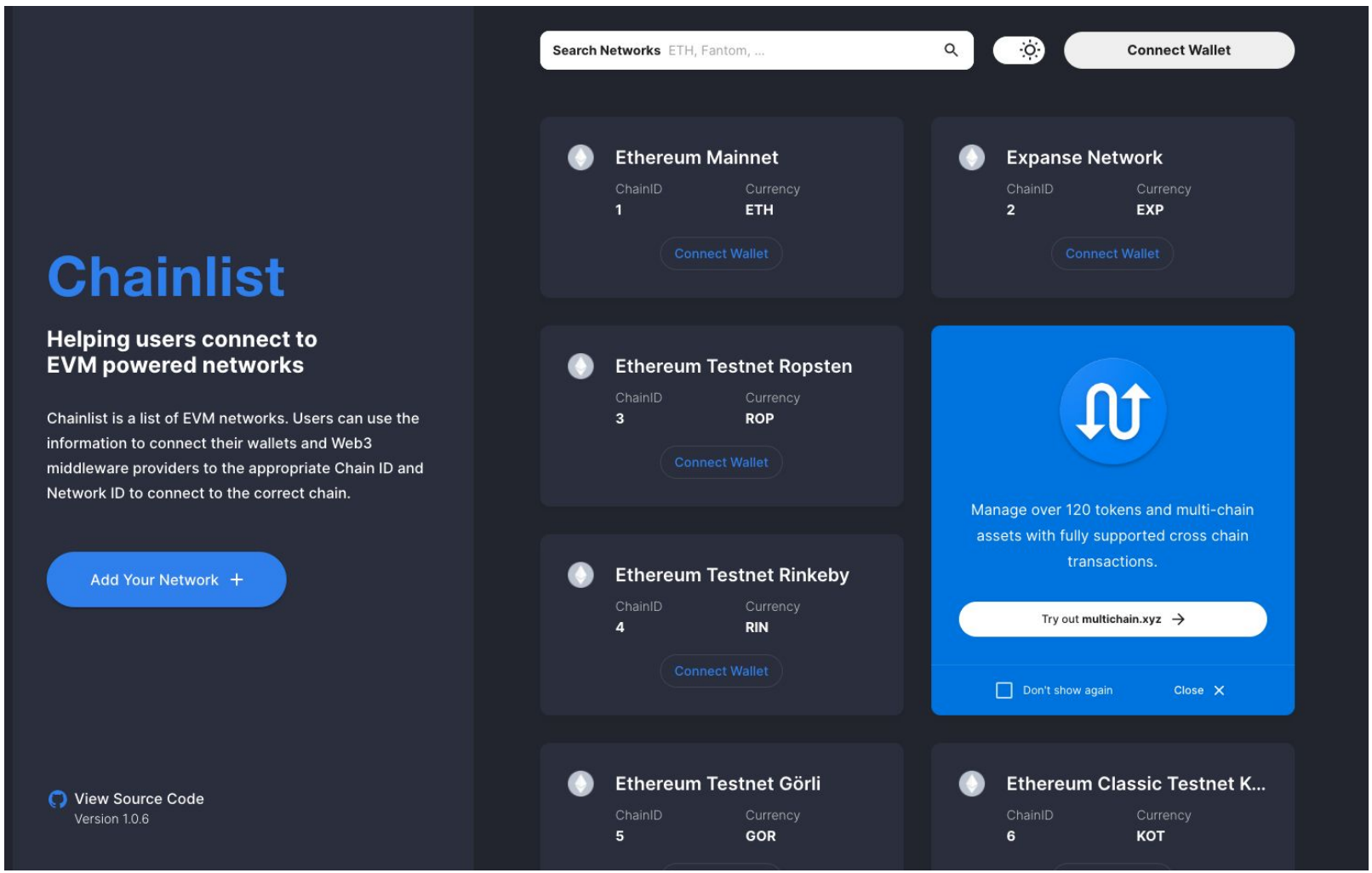
En la web3 **tu identidad es criptográfica** y administrada a través de un proveedor. No son necesarios los proveedores de autenticación.





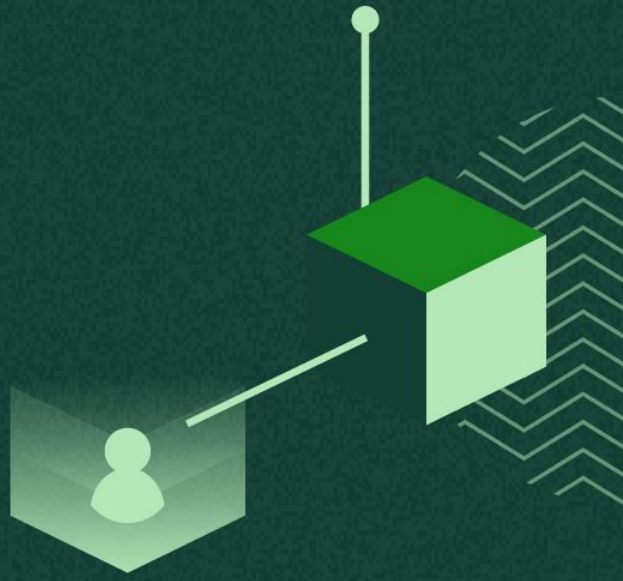


Si la aplicación lo soporta, puedes incluso elegir tu propio backend cambiando el endpoint de tu proveedor. **https://chainlist.org**

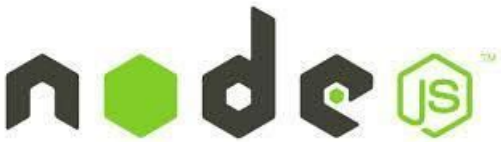


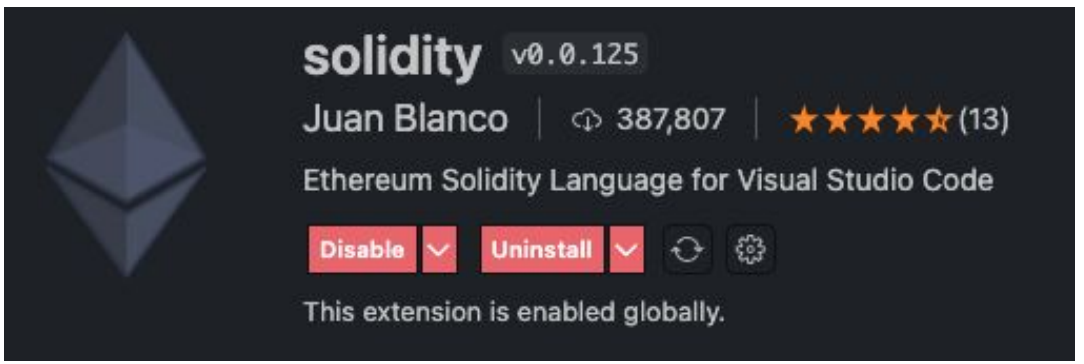
Setup de **PlatziPunks**

Módulo 3



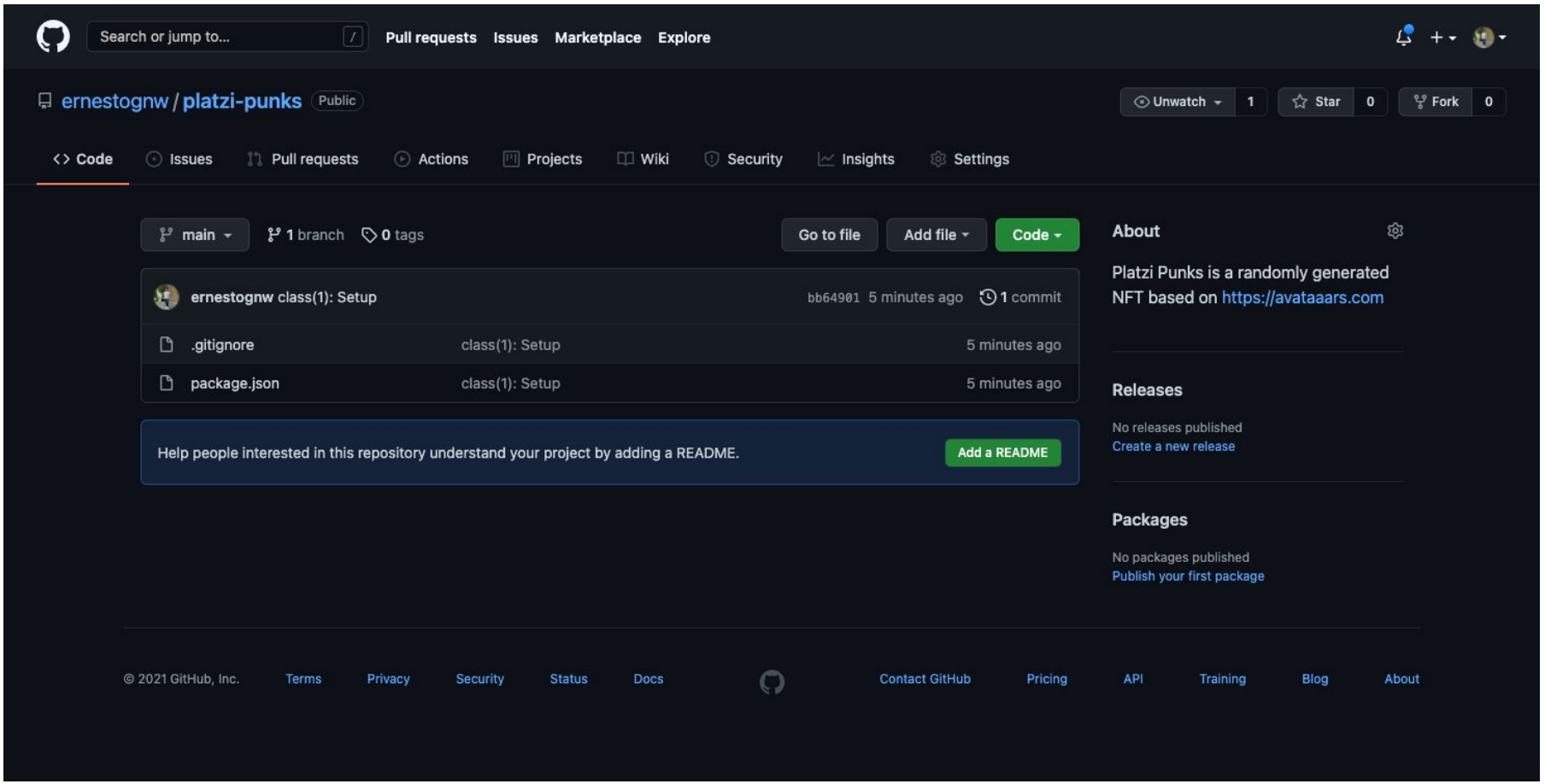
JavaScript
y Node






```
mkdir platzi-punks
cd platzi-punks
yarn init -y
touch .gitignore
```

```
git add .
git commit -m "class(1): Setup"
```


Search or jump to...
Pull requests
Issues
Marketplace
Explore
ernestognw / platzi-punks
Public
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
ernestognw class(1): Setup
bb64901 5 minutes ago
1 commit
.gitignore
class(1): Setup
5 minutes ago
package.json
class(1): Setup
5 minutes ago
Help people interested in this repository understand your project by adding a README.
Add a README
About
Platzi Punks is a randomly generated NFT based on https://avataaars.com
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

Instalación y **overview de Hardhat**

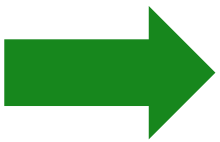
Ethereum development environment for professionals

Hardhat es un entorno de desarrollo que nos permite compilar, probar y desplegar contratos inteligentes.

Compilación, **tests** y despliegue

Compilación

```
//SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.0;

import "hardhat/console.sol";

contract Greeter {
    string private greeting;

    constructor(string memory _greeting) {
        console.log("Deploying a Greeter with greeting:", _greeting);
        greeting = _greeting;
    }

    function greet() public view returns (string memory) {
        return greeting;
    }

    function setGreeting(string memory _greeting) public {
        greeting = _greeting;
    }
}
```



Compilación

```
npx hardhat compile
```

Tests

```
npx hardhat test
```

Despliegue

polygon

OP

Despliegue

```
npx hardhat run scripts/sample-script.js
```

Despliegue en **Rinkeby con Infura**

¿Cómo enviamos a la blockchain nuestro contrato compilado?

El código objeto generado de un contrato en Solidity debe ser enviado **a través de una transacción** hacia un nodo **compatible con la máquina virtual de Ethereum.**

Dado que un despliegue es **una transacción**, es necesario **fondear nuestra** cuenta con Ether.

Transacción

Tradicionalmente, deberías de correr tu propio nodo de Ethereum para enviar tu código a producción.

No obstante, los **costos de manutención** y el **tiempo de sincronización** no suelen ser alternativas cuando estás desarrollando un Smart Contract.

https://geth.ethereum.org/

Tradicionalmente

- Para aplicaciones normales, el problema del costo de **mantenimiento de infraestructura** privada también es frecuente.

- Este problema lo solucionan los mayores proveedores de **infraestructura en la nube.**

Con DApps

Existen alternativas de **infraestructura como servicio** para montar y sincronizar un nodo de Ethereum.

Estos nodos suelen ser utilizados para desplegar nuestros contratos inteligentes.

alchemy

INFURA

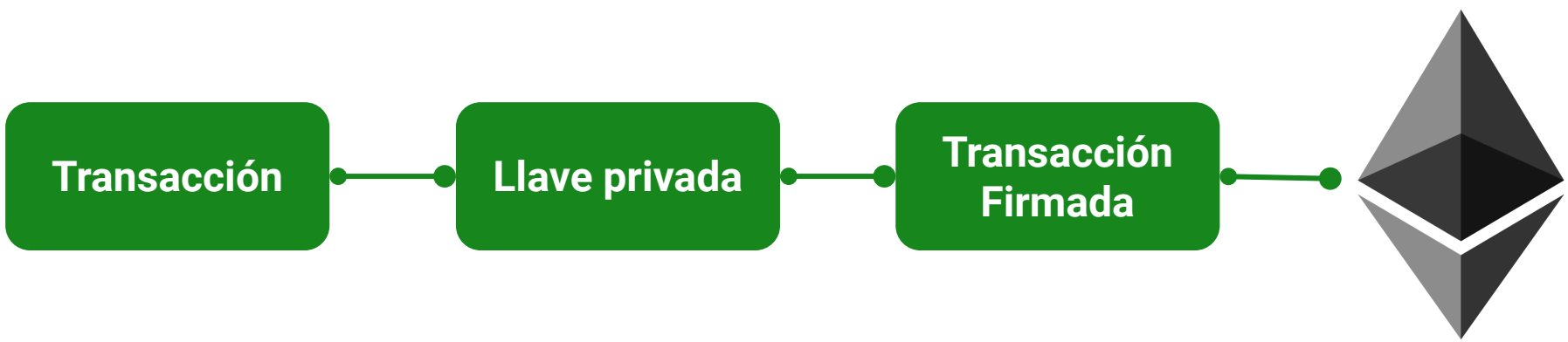
Configuración

```
module.exports = {
  solidity: "0.8.7",
  networks: {
    rinkeby: {
      url: "https://rinkeby.infura.io/v3/<<PASTE YOUR INFURA PROJECT ID (DANGEROUS)>>",
      accounts: ["<<PASTE YOUR PRIVATE KEY HERE (DANGEROUS)>>"],
    },
  },
};
```

```
npx hardhat run scripts/example-script.js --network rinkeby
```

Manejo de **llaves privadas**

Llaves privadas



Cualquiera con tus llaves privadas puede firmar transacciones y robar tus fondos

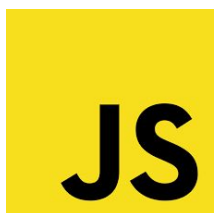
No las pongas en Github

Alternativas **a Hardhat**

¿Solo existe Hardhat?

https://eth-brownie.readthedocs.io

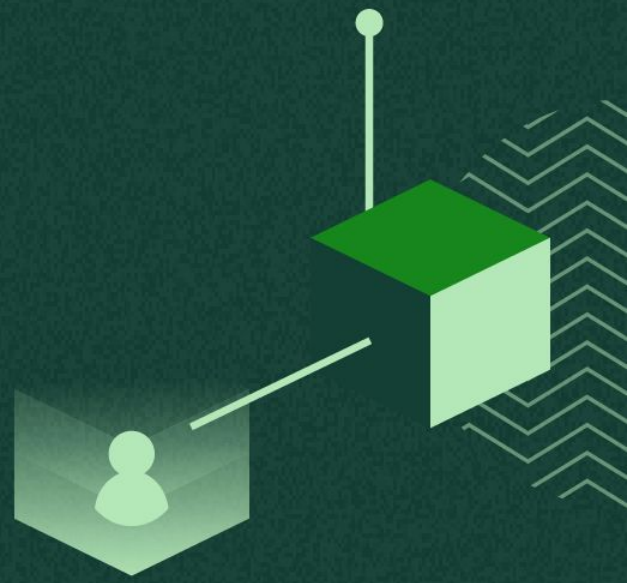
https://trufflesuite.com/



dapp.tools

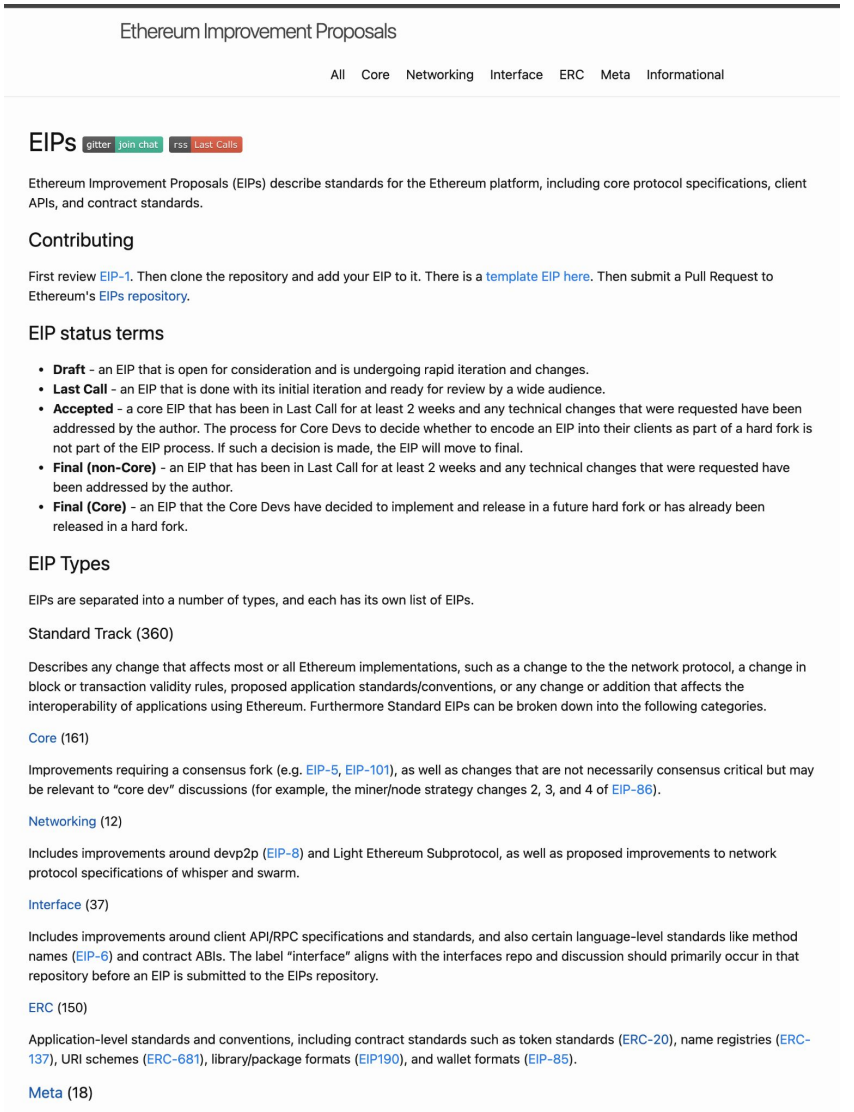
https://dapp.tools/

Ethereum **Improvement Proposals**



EIP

La comunidad de Ethereum tiene una lista de **propuestas de mejora** que se han convertido en estándares por **consenso de la comunidad.**



Ethereum Improvement Proposals

All Core Networking Interface ERC Meta Informational

EIPs

Ethereum Improvement Proposals (EIPs) describe standards for the Ethereum platform, including core protocol specifications, client APIs, and contract standards.

Contributing

First review EIP-1. Then clone the repository and add your EIP to it. There is a template EIP here. Then submit a Pull Request to Ethereum's EIPs repository.

EIP status terms

- **Draft** - an EIP that is open for consideration and is undergoing rapid iteration and changes.
- **Last Call** - an EIP that is done with its initial iteration and ready for review by a wide audience.
- **Accepted** - a core EIP that has been in Last Call for at least 2 weeks and any technical changes that were requested have been addressed by the author. The process for Core Devs to decide whether to encode an EIP into their clients as part of a hard fork is not part of the EIP process. If such a decision is made, the EIP will move to final.
- **Final (non-Core)** - an EIP that has been in Last Call for at least 2 weeks and any technical changes that were requested have been addressed by the author.
- **Final (Core)** - an EIP that the Core Devs have decided to implement and release in a future hard fork or has already been released in a hard fork.

EIP Types

EIPs are separated into a number of types, and each has its own list of EIPs.

Standard Track (360)

Describes any change that affects most or all Ethereum implementations, such as a change to the the network protocol, a change in block or transaction validity rules, proposed application standards/conventions, or any change or addition that affects the interoperability of applications using Ethereum. Furthermore Standard EIPs can be broken down into the following categories.

Core (161)

Improvements requiring a consensus fork (e.g. EIP-5, EIP-101), as well as changes that are not necessarily consensus critical but may be relevant to "core dev" discussions (for example, the miner/node strategy changes 2, 3, and 4 of EIP-86).

Networking (12)

Includes improvements around devp2p (EIP-8) and Light Ethereum Subprotocol, as well as proposed improvements to network protocol specifications of whisper and swarm.

Interface (37)

Includes improvements around client API/RPC specifications and standards, and also certain language-level standards like method names (EIP-6) and contract ABIs. The label "interface" aligns with the interfaces repo and discussion should primarily occur in that repository before an EIP is submitted to the EIPs repository.

ERC (150)

Application-level standards and conventions, including contract standards such as token standards (ERC-20), name registries (ERC-137), URI schemes (ERC-681), library/package formats (EIP190), and wallet formats (EIP-85).

Meta (18)

EIP

- Protocolo de Comunicación
- Mercado de fees
- Mecanismos de Consenso
- **Tokens (ERC20 y ERC721)**

https://eips.ethereum.org/

PlatziPunks será un proyecto compatible con el estándar EIP 721 de Ethereum para desarrollar **Tokens No Fungibles (NFTs).**

EIP-721: Non-Fungible Token Standard ‹›

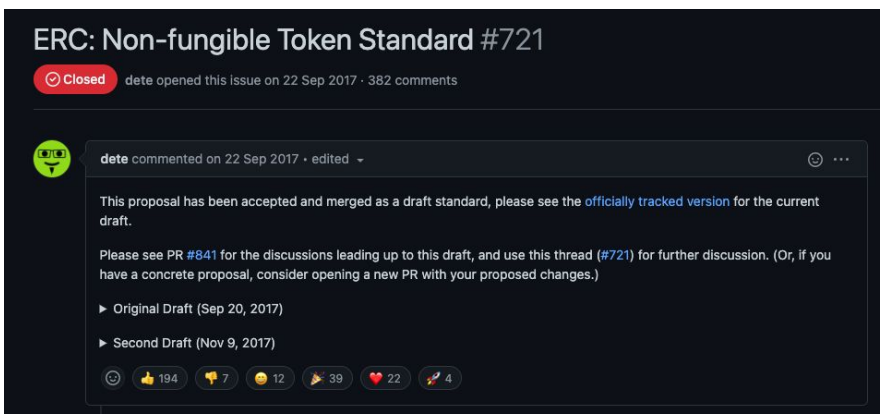
Author	William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs
Discussions-To	https://github.com/ethereum/eips/issues/721
Status	Final
Type	Standards Track
Category	ERC
Created	2018-01-24
Requires	165

https://eips.ethereum.org/EIPS/eip-721

Discusiones

Todos los estándares de la comunidad han sido (o están siendo) activamente discutidos en los foros de Github con el mismo número con el que se aprueban.

Inicialmente se les conoce como **Ethereum Request for Comments (ERCs).**



https://github.com/ethereum/EIPs/issues/721

Open Zeppelin Contracts

Seguridad en Smart Contracts

Dado que los contratos inteligentes están **estandarizados** y hacen **manejo de fondos**, la mayor preocupación es la seguridad.

Herramientas como Open Zeppelin proveen **contratos inteligentes estándar, open source y previamente auditados.**

OpenZeppelin

Implementaciones base

```
// contracts/GLDToken.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract GLDToken is ERC20 {
    constructor(uint256 initialSupply) ERC20("Gold", "GLD") {
        _mint(msg.sender, initialSupply);
    }
}
```

Control de acceso

```sol
// contracts/MyContract.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/Ownable.sol";

contract MyContract is Ownable {
    function normalThing() public {
        // anyone can call this normalThing()
    }

    function specialThing() public onlyOwner {
        // only the owner can call specialThing()!
    }
}
```

Componentes

```
using ECDSA for bytes32;

function _verify(bytes32 data, address account) pure returns (bool) {
    return keccak256(data)
        .toEthSignedMessageHash()
        .recover(signature) == account;
}
```

¿Customizaciones?

```solidity
pragma solidity ^0.8.0;

import "../../utils/introspection/IERC165.sol";

/**
 * @dev Required interface of an ERC721 compliant contract.
 */
interface IERC721 is IERC165 {
    /**
     * @dev Emitted when `tokenId` token is transferred from `from` to `to`.
     */
    event Transfer(address indexed from, address indexed to, uint256 indexed tokenId);

    /**
     * @dev Emitted when `owner` enables `approved` to manage the `tokenId` token.
     */
    event Approval(address indexed owner, address indexed approved, uint256 indexed tokenId);

    /**
     * @dev Emitted when `owner` enables or disables (`approved`) `operator` to manage all of its assets.
     */
    event ApprovalForAll(address indexed owner, address indexed operator, bool approved);

    /**
     * @dev Returns the number of tokens in ``owner``'s account.
     */
    function balanceOf(address owner) external view returns (uint256 balance);
```

Implementando el ERC721 en PlatziPunks

Extendiendo la funcionalidad de PlatziPunks

¿Qué es la **metadata del ERC721?**

Metadata del ERC721

El ERC721 Metadata es una extensión al estándar ERC721 para NFT que añade 3 funciones:

- **name:** el nombre del token.
- **symbol:** el símbolo con el que se identifica en el mercado.
- **tokenURI:** una URL que debe regresar un archivo JSON con las propiedades del NFT.

Ya viene implementado en OpenZeppelin, pero debemos extenderlo.

Metadata del ERC721

```
interface ERC721Metadata {
    /// @notice A descriptive name for a collection of NFTs in this contract
    function name() external view returns (string _name);

    /// @notice An abbreviated name for NFTs in this contract
    function symbol() external view returns (string _symbol);

    /// @notice A distinct Uniform Resource Identifier (URI) for a given asset.
    /// @dev Throws if `_tokenId` is not a valid NFT. URIs are defined in RFC
    ///  3986. The URI may point to a JSON file that conforms to the "ERC721
    ///  Metadata JSON Schema".
    function tokenURI(uint256 _tokenId) external view returns (string);
}
```

PlatziPunks

balanceOf()

safeTransferFrom()

...

tokenURI()

JSON File

Asset Metadata

```
{
    "title": "Asset Metadata",
    "type": "object",
    "properties": {
        "name": {
            "type": "string",
            "description": "Identifies the asset to which this NFT represents"
        },
        "description": {
            "type": "string",
            "description": "Describes the asset to which this NFT represents"
        },
        "image": {
            "type": "string",
            "description": "A URI pointing to a resource with mime type image/* representing the asset
to which this NFT represents. Consider making any images at a width between 320 and 1080 pixels and
aspect ratio between 1.91:1 and 4:5 inclusive."
        }
    }
}
```

PlatziPunks

balanceOf()

safeTransferFrom()

...

tokenURI()

La Metadata hace que aplicaciones como OpenSea puedan **registrar la imagen y los datos del NFT.**

```
{
  ...
  "image": "https://someurl.com/token/:id"
  ...
}
```

¿Has escuchado que la imagen de un NFT no vive en Blockchain?

Esta es la razón

Implementando la metadata de PlatziPunks

Como **no tenemos un servidor** donde guardar los JSON de la Metadata, usaremos un truco muy común de **NFTs on-chain** para guardarlo en el contrato inteligente.

```
{
  ...
  "image": "https://someurl.com/token/:id"
  ...
}
```

Acorde con los estándares de la web, una URL válida es una **DATA URL,** que consiste en una URI con los datos de respuesta codificados en **Base64.**

Data URLs

Data URLs, URLs prefixed with the `data:` scheme, allow content creators to embed small files inline in documents. They were formerly known as "data URIs" until that name was retired by the WHATWG.

> **Note:** Data URLs are treated as unique opaque origins by modern browsers, rather than inheriting the origin of the settings object responsible for the navigation.

Syntax

Data URLs are composed of four parts: a prefix (`data:`), a MIME type indicating the type of data, an optional `base64` token if non-textual, and the data itself:

```
data:[<mediatype>][;base64],<data>
```

tokenURI()

```
data:application/json;base64,ZWplbXBsbw...
```

```
{
  ...
  "image": "https://someurl.com/token/:id"
  ...
}
```

Diseñando el ADN de PlatziPunks

Imagen del Punk

Para implementar la imagen de nuestro PlatziPunk es necesario apuntar a una **URL que regrese la imagen.**

Afortunadamente, **existe un API** para traer las diferentes combinaciones de Punks.

https://avataaars.io/

El API retorna un SVG

```
https://avataaars.io/
    ?accessoriesType=Sunglasses
    &clotheColor=Gray02
    &clotheType=GraphicShirt
    &eyeType=Happy
    &eyebrowType=UpDownNatural
    &facialHairColor=Platinum
    &facialHairType=BeardMagestic
    &hairColor=Red
    &hatColor=PastelRed
    &graphicType=Deer
    &mouthType=Smile
    &skinColor=Light
    &topType=WinterHat1
```

Propiedades

Existen 13 propiedades combinables de los PlatziPunks.

Necesitamos una forma de calcular lo más **aleatoriamente** posible las propiedades para cada NFT.

- accesoriesType
- clotheColor
- clotheType
- eyeType
- eyebrowType
- facialHairColor
- facialHairType
- hairColor
- hatColor
- graphicType
- mouthType
- skinColor
- topType

ADN

El ADN de cada PlatziPunk será una cadena numérica de **al menos 26 caracteres** (2 por atributo).

... 01 23 45 67 89 01 23 45 67 89 01 23 45

ADN

uint256


accesoriesType
clotheType
eyebrowType
facialHairType
hatColor
mouthType
topType
... 01 23 45 67 89 01 23 45 67 89 01 23 45
clotheColor
eyeType
facialHairColor
hairColor
graphicType
skinColor

Cálculo de atributos

Vamos a implementar una función en Solidity que **recorte la parte que nos interesa** del ADN para convertirlo en un atributo.

Operación	
(... 678901234567890123**45** % 1**00**) / 1	45
(... 678901234567890**23**45 % 1**0000**) / 1**00**	23
(... 678901234567890**1**2345 % 1**000000**) / 1**0000**	01
...	

Transformando en string

(... 012345678901234567**89**012345 % 1**00000000**) / 1**000000**

= 89

```
_skinColor = [
    "Tanned",
    "Yellow",
    "Pale",
    "Light",
    "Brown",
    "DarkBrown",
    "Black"
]
```

89 % _skinColor.length → **5** → **“DarkBrown”**

Implementando el ADN de PlatziPunks

Calculando el ADN de PlatziPunks

Aleatoriedad

- Idealmente, los PlatziPunks deberían tener un ADN aleatorio. No obstante, **la aleatoriedad no existe en la blockchain.**

- La razón es que todas las operaciones son **deterministas.**

Determinismo

El determinismo es una propiedad computacional en la que, dado un **estado inicial** y una **acción,** siempre **genera el mismo resultado.**

Esto garantiza que los nodos puedan ponerse de acuerdo. Si no, todos tendrían información distinta.

Proyectos como **Chainlink** resuelven este problema.

Mueven el procesamiento no determinista off-chain, y cobran un fee por reinsertarlo en blockchain.

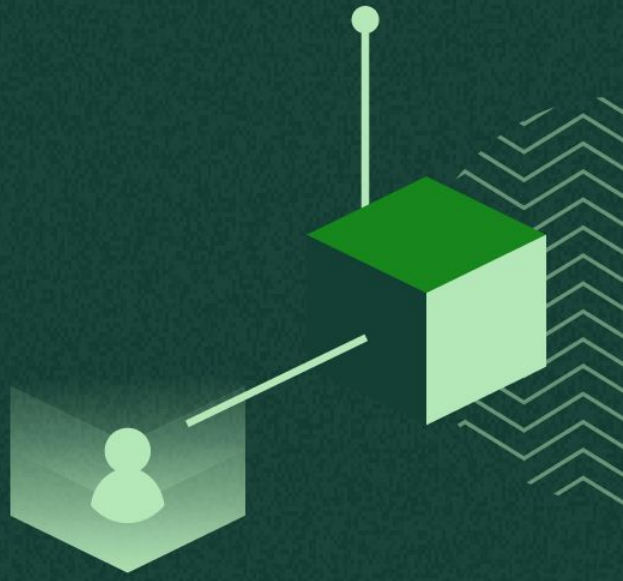
Estos servicios se conocen como **oráculos.**

Chainlink

Nosotros implementaremos una función pseudoaleatoria para calcular el ADN basado en el tokenId y el address del usuario

Usando el ADN **para calcular la imagen del NFT**

Probando nuestro **Smart Contract**

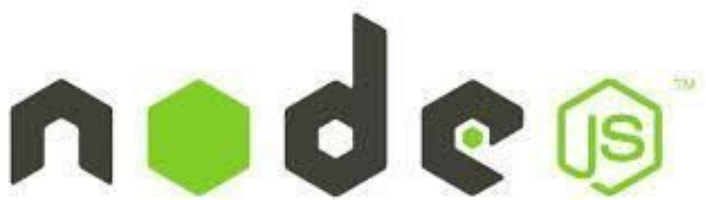


Despliegue **en redes de prueba**

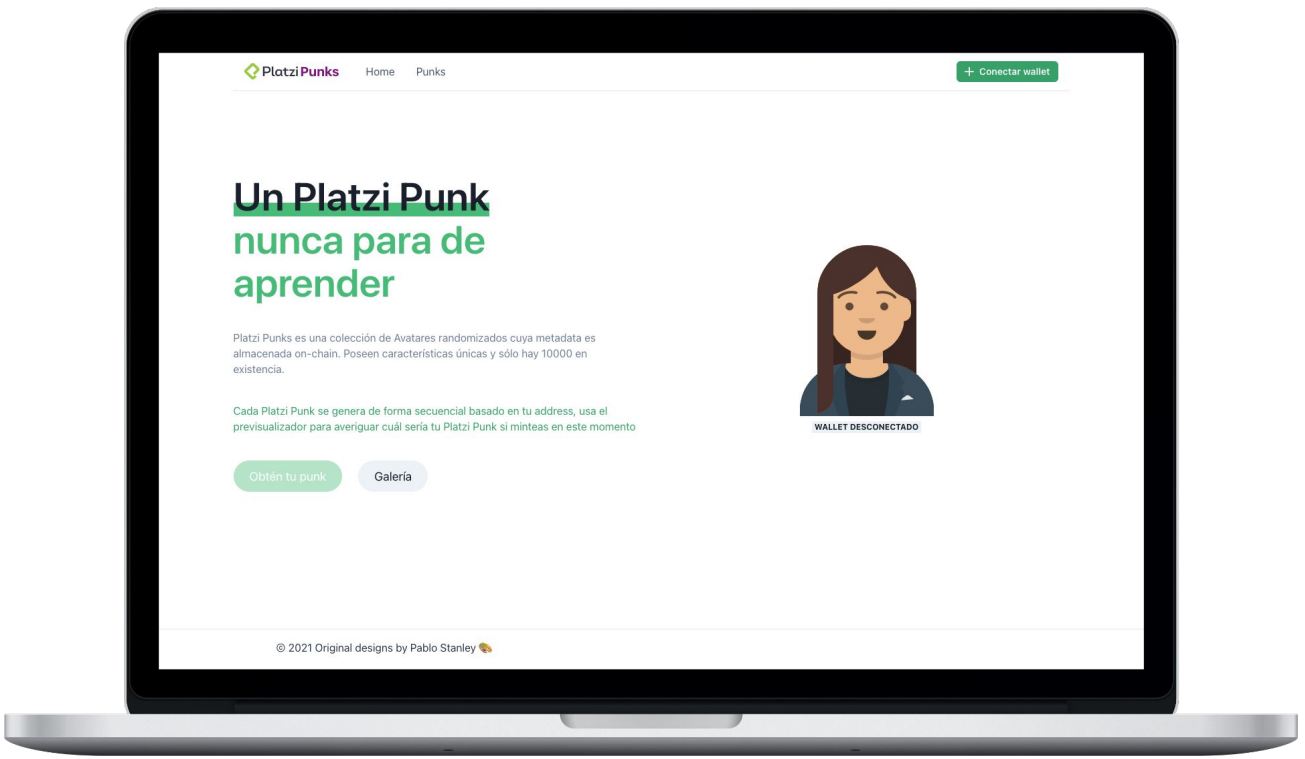
Verificando a PlatziPunks **en Etherscan**

¡Crea tu PlatziPunk y **visualízalo en OpenSea!**

Continúa con el proyecto PlatziPunks




Platzi Punks
0x3ebf...466e
NEXT ID: 11
ADDRESS: 0X3EBF...466E
Actualizar
Un Platzi Punk
nunca para de aprender
Platzi Punks es una colección de Avatares randomizados cuya metadata es almacenada on-chain. Poseen características únicas y sólo hay 10000 en existencia.


Platzi Punks
Home
Punks
Conectar wallet
Un Platzi Punk
nunca para de aprender
Platzi Punks es una colección de Avatares randomizados cuya metadata es almacenada on-chain. Poseen características únicas y sólo hay 10000 en existencia.
Cada Platzi Punk se genera de forma secuencial basado en tu address, usa el previsualizador para averiguar cuál sería tu Platzi Punk si minteas en este momento
Obtén tu punk
Galería
WALLET DESCONECTADO
© 2021 Original designs by Pablo Stanley