Blockchain y smart contracts

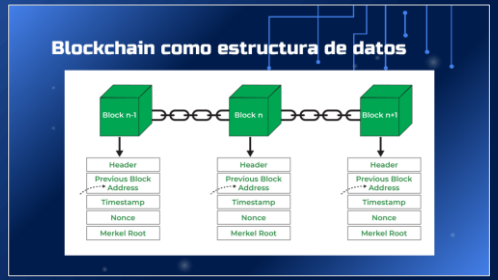
Arquitectura de Software

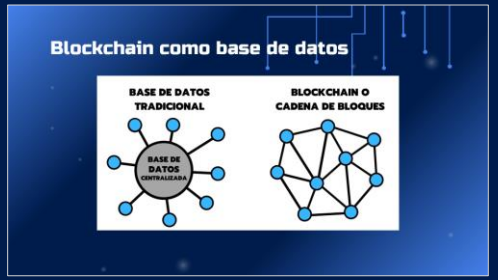
Luisa Parra
Fabio Buitrago
Erick Garavito

”Para comprender el poder de los sistemas blockchain y las cosas que pueden hacer, es importante distinguir entre tres cosas que comúnmente se confunden, a saber, la moneda bitcoin, la blockchain específica que la sustenta y la idea de blockchain en general”

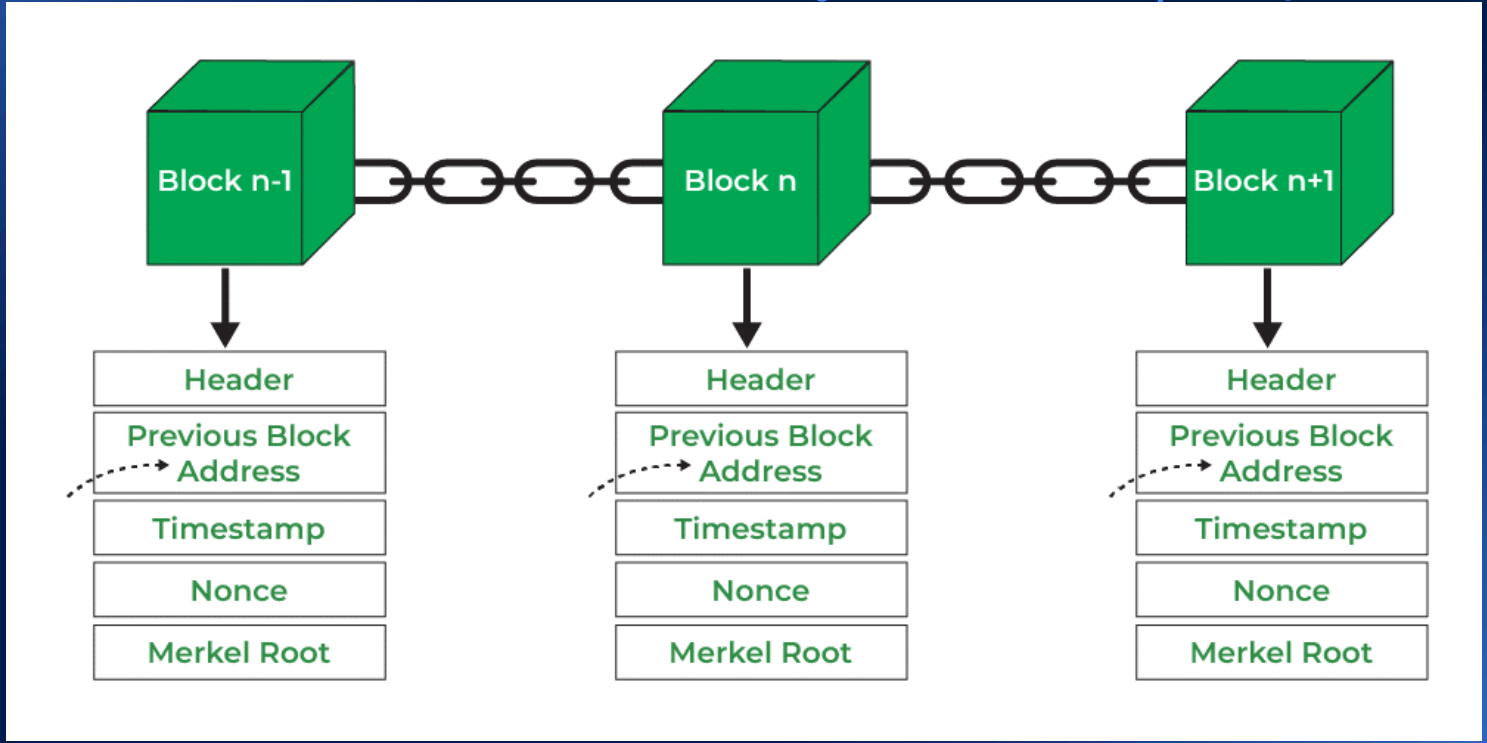
— The Trust Machine, THE ECONOMIST, Oct. 31, 2015

Blockchain como: estructura de datos, base de datos y plataforma

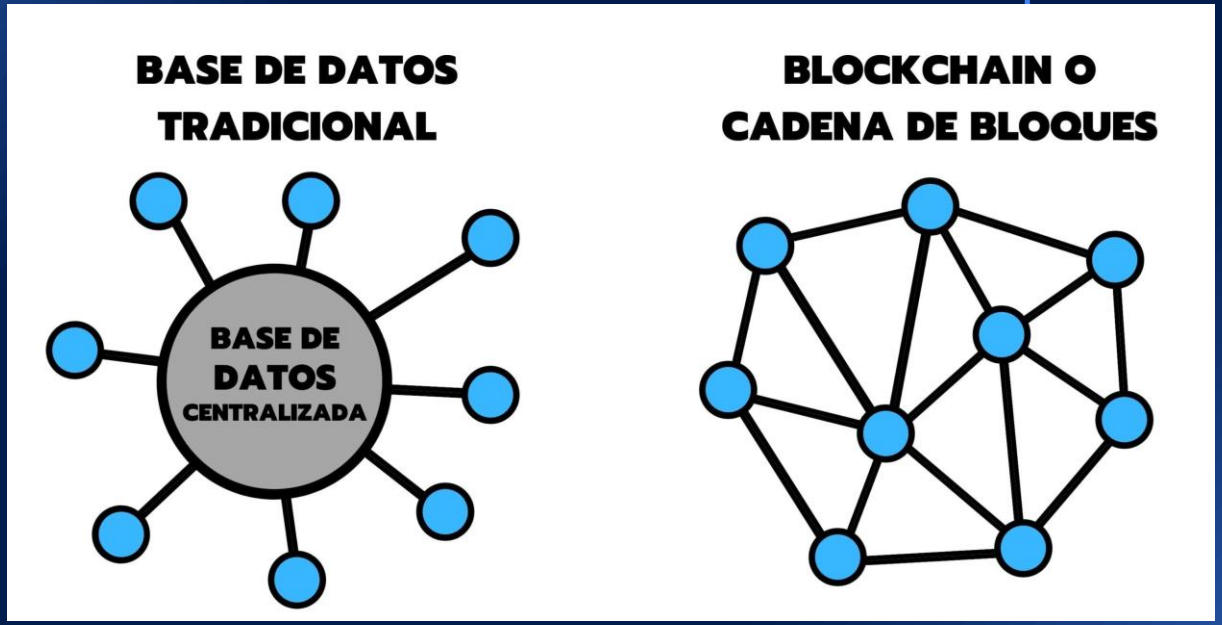




Blockchain como estructura de datos



Blockchain como base de datos



Blockchain como plataforma

Historia y evolución

Precursores (1980s - 1990s)

1982
David Chaum introduce el concepto de ”firmas ciegas”

1991
Stuart Haber y W. Scott Stornetta proponen ”Bestamp”

1993
Eli Shamir y Adi Shamir introducen el concepto de ”firmas de grupo”

Nacimiento de Bitcoin y la era de las criptomonedas (2008 - 2015)

2008

Satoshi Nakamoto publica el whitepaper de Bitcoin

2009

Se lanza la red Bitcoin, demostrando la viabilidad de la tecnología blockchain.

2013

Se lanza Ripple, una red de pagos diseñada para facilitar transacciones internacionales rápidas y económicas

2014

Se lanza Ethereum, plataforma blockchain que permite la creación de contratos inteligentes y DApps.

Expansión y madurez (2016 - actualidad)

2016

Hyperledger Fabric es lanzada por la Fundación Linux, una plataforma blockchain modular y escalable diseñada para empresas

2018

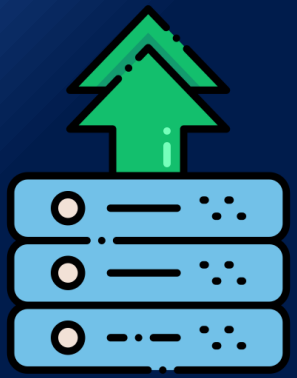
Se forma la Alianza para la Economía Digital (WEF), una iniciativa global para promover el desarrollo y la adopción responsable de la tecnología blockchain

Actualidad

Auge de las DeFi y los NFT y el uso de la tecnología blockchain en nuevas áreas

Características

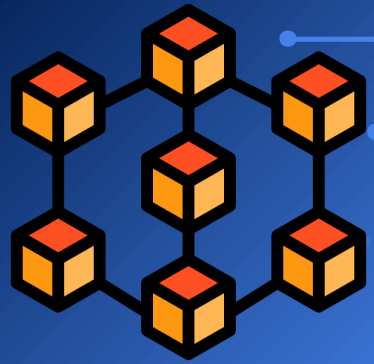
Escalabilidad



Trazabilidad

Privacidad

Descentralización



Trilema de la blockchain

Algoritmos de consenso

Proof of Work
PoW

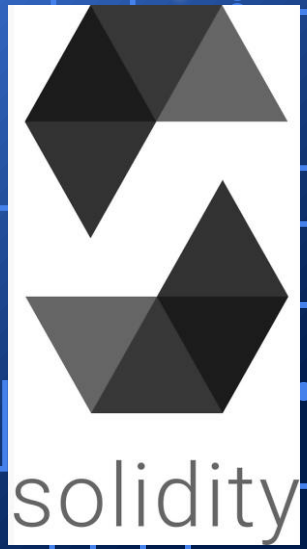
Proof of Stake
PoS

Delegated Proof of Stake
DPoS

Proof of Authority
PoA

Ethereum

- Es una plataforma blockchain descentralizada que permite la creación y ejecución de contratos inteligentes (smart contracts) y aplicaciones descentralizadas (DApps). Fundada en 2015, Ethereum permite a los desarrolladores construir aplicaciones que operan sin intermediarios, utilizando su criptomoneda nativa, Ether (ETH), para facilitar transacciones y procesos automatizados dentro de la red.



Solidity

- Es un lenguaje de programación de alto nivel, específicamente diseñado para crear contratos inteligentes en la plataforma blockchain de Ethereum. Con una sintaxis similar a la de JavaScript, Solidity permite a los desarrolladores escribir y desplegar contratos inteligentes que se ejecutan en la máquina virtual de Ethereum (EVM). Estos contratos pueden automatizar y garantizar la ejecución de acuerdos, facilitando la creación de aplicaciones descentralizadas (DApps).

Contratos inteligentes (Smart contracts)

- son programas autoejecutables con los términos del acuerdo directamente escritos en líneas de código. Se ejecutan en una red blockchain, como Ethereum, lo que garantiza que sean inmutables y transparentes.

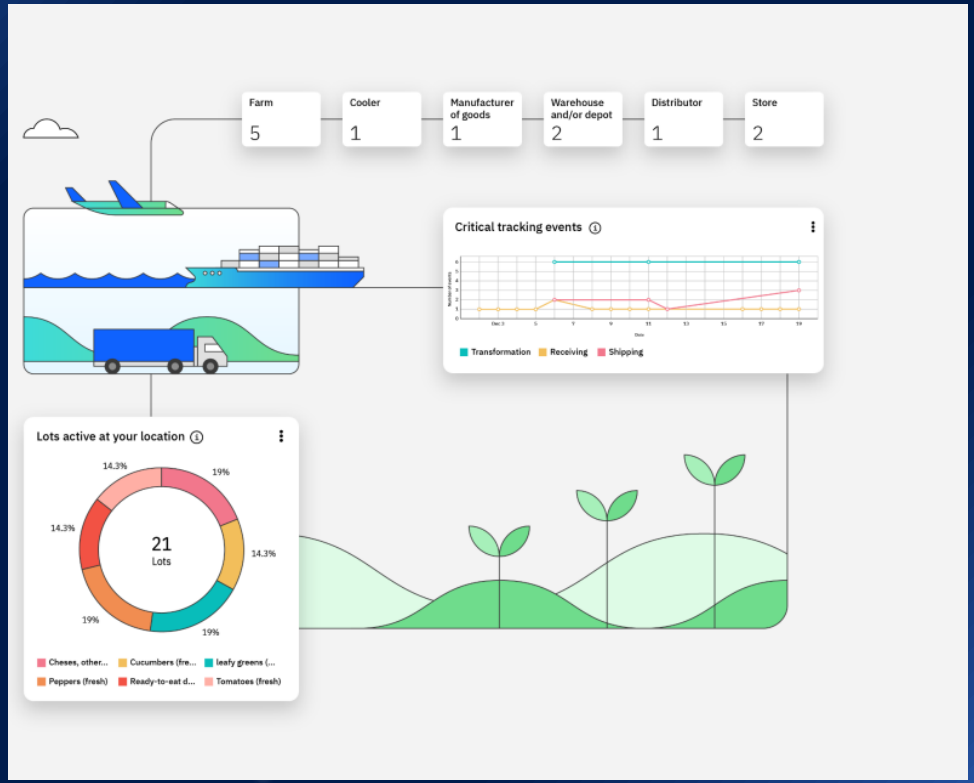
Ganache

- Ganache es una herramienta de desarrollo para la plataforma Ethereum, creada por Truffle Suite. Esencialmente, Ganache permite a los desarrolladores de blockchain crear una cadena de bloques local privada para probar y desarrollar contratos inteligentes y aplicaciones descentralizadas (DApps) sin necesidad de interactuar con la red principal de Ethereum.

Casos de uso

- Finanzas Descentralizadas (DeFi)
- Propiedad Intelectual y Derechos de Autor
- Votación electrónica
- Registro de Propiedad Inmobiliaria
- Gestión de la Cadena de Suministro
- Salud y Expedientes Médicos
- Gestión de Identidad Digital
- Seguros

Caso de aplicación
IBM Food Trust



Principios SOLID vs Temas

Principios SOLID \ Temas	Blockchain	Smart Contracts	Solidity
S **Single Responsibility Principle (SRP)**	Definir componentes específicos para cada función dentro de la cadena de bloques.	Cada contrato inteligente debe tener una única responsabilidad.	Las clases y contratos en Solidity deben tener una única responsabilidad.
O **Open/Closed Principle (OCP)**	La blockchain debe ser abierta a extensiones, pero cerrada a modificaciones directas.	Los contratos inteligentes deben permitir extensiones sin modificar el código original.	Utilizar herencia y bibliotecas para extender funcionalidades sin modificar el código existente.
L **Liskov Substitution Principle (LSP)**	Los componentes de la blockchain deben ser reemplazables por otros que respeten la misma interfaz.	Contratos inteligentes derivados deben poder reemplazar contratos base sin alterar el comportamiento.	Contratos y clases derivados en Solidity deben mantener el comportamiento esperado del contrato o clase base.
I **Interface Segregation Principle (ISP)**	Interfaces específicas para distintas funciones dentro de la blockchain.	Descomponer contratos en múltiples interfaces específicas.	Dividir interfaces en Solidity para evitar dependencias innecesarias.
D **Dependency Inversion Principle (DIP)**	Los módulos de alto nivel no deben depender de módulos de bajo nivel, ambos deben depender de abstracciones.	Los contratos deben depender de interfaces y no de implementaciones concretas.	Utilizar interfaces y contratos abstractos en Solidity para depender de abstracciones.

Atributos de calidad vs Temas

Atributos de Calidad \ Temas	Blockchain	Smart Contracts	Solidity
Seguridad	Mecanismos de consenso seguros y resistentes a ataques.	Verificación formal y auditoría de contratos inteligentes.	Prácticas seguras de codificación y uso de herramientas de análisis estático.
Escalabilidad	Soluciones de Layer 2 y sharding.	Optimización de gas y gestión eficiente de recursos.	Optimizaciones específicas de Solidity para reducir el consumo de gas.
Mantenibilidad	Diseño modular y actualización de nodos sin interrupciones.	Contratos inteligentes bien documentados y modulares.	Código claro, comentado y modular en Solidity.
Rendimiento	Tiempos de bloque eficientes y procesamiento rápido de transacciones.	Ejecución eficiente de contratos inteligentes.	Código Solidity optimizado para un rendimiento rápido.
Usabilidad	Interfaces de usuario amigables para interacción con la blockchain.	Interfaces claras y fáciles de entender para los usuarios de contratos.	Simplificación del lenguaje Solidity para facilitar su aprendizaje y uso.

Tácticas vs Temas

Tácticas \ Temas	Blockchain	Smart Contracts	Solidity
Tácticas de Seguridad	Implementación de criptografía robusta.	Uso de patrones de diseño seguros como multisig.	Uso de bibliotecas de seguridad y patrones como checks-effects-interactions.
Tácticas de Escalabilidad	Sharding y soluciones de capa 2.	Fragmentación de contratos y optimización de gas.	Optimizaciones de código y uso de opcodes eficientes.
Tácticas de Mantenibilidad	Modularidad y actualizaciones sin interrupciones.	Contratos modulares y actualizables.	Uso de patrones de diseño que faciliten la actualización y mantenimiento del código.
Tácticas de Rendimiento	Mejora en los algoritmos de consenso.	Optimización en la ejecución de contratos.	Uso de estructuras de datos y algoritmos eficientes en Solidity.
Tácticas de Usabilidad	Mejora de las interfaces de usuario para nodos y wallets.	Contratos con interfaces de usuario intuitivas.	Mejoras en el lenguaje y herramientas de desarrollo para Solidity.

Patrones vs Temas

Patrones de Arquitectura \ Temas	Blockchain	Smart Contracts	Solidity
Patrón de Microservicios	Descomposición de la blockchain en microservicios para escalabilidad.	Contratos inteligentes modulares y desplegables individualmente.	Implementación de contratos en Solidity como microservicios independientes.
Patrón de Event Sourcing	Registro de todas las transacciones en la blockchain como eventos inmutables.	Uso de eventos en contratos inteligentes para registrar cambios de estado.	Emisión de eventos en Solidity para monitoreo y registro de actividades.
Patrón de CQRS	Separación de operaciones de lectura y escritura en la blockchain.	Uso de comandos y consultas separados en contratos inteligentes.	Implementación de CQRS en el diseño de contratos Solidity.
Patrón de Singleton	Nodo único que gestiona la configuración global de la blockchain.	Contrato único que maneja ciertas operaciones globales.	Uso del patrón Singleton en contratos Solidity para gestión de datos globales.
Patrón de Proxy	Uso de proxies para actualización de contratos inteligentes sin cambiar la dirección.	Implementación de contratos proxy para gestionar actualizaciones.	Uso de contratos proxy en Solidity para permitir actualizaciones sin modificar la dirección del contrato principal.

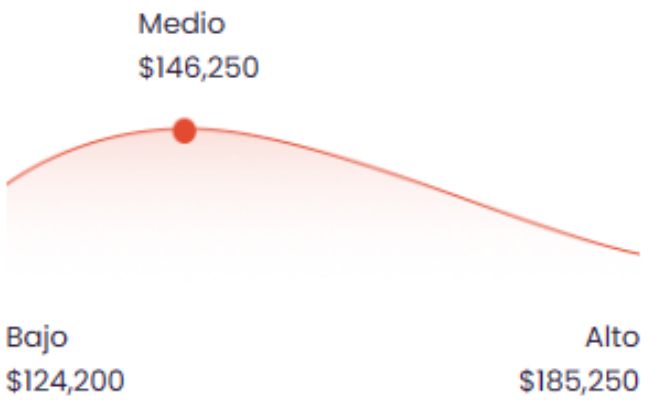
Mercado laboral

¿Cuánto gana un Blockchain Developer en Estados Unidos?

$146,250 / Año

Basado en 934 salarios

El salario **blockchain developer** promedio en **Estados Unidos** es de **$146,250** al año o **$70.31** por hora. Los cargos de nivel inicial comienzan con un ingreso de **$124,200** al año, mientras que profesionales más experimentados perciben hasta **$185,250** al año.



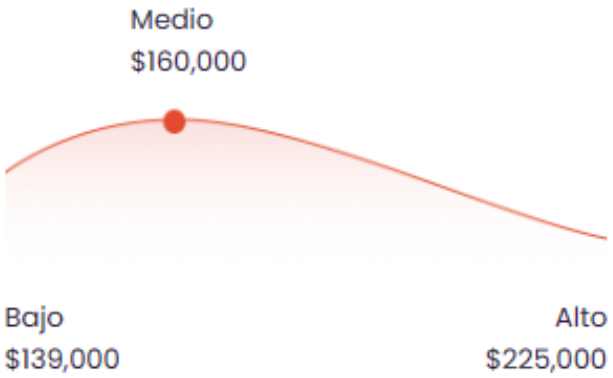
Mercado laboral

¿Cuánto gana un Smart Contract Developer en Estados Unidos?

$160,000 / Año

Basado en 102 salarios

El salario **smart contract developer** promedio en **Estados Unidos** es de **$160,000** al año o **$76.92** por hora. Los cargos de nivel inicial comienzan con un ingreso de **$139,000** al año, mientras que profesionales más experimentados perciben hasta **$225,000** al año.



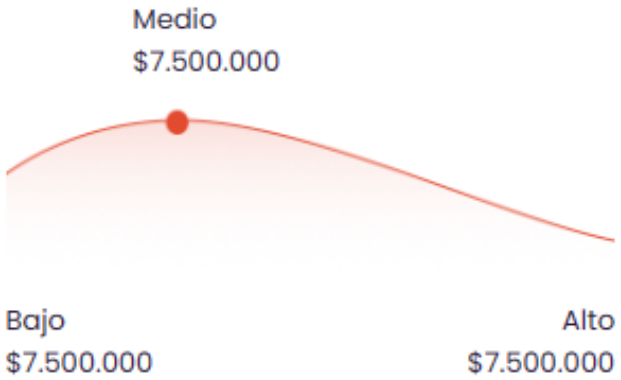
Mercado laboral

¿Cuánto gana un Desarrollador blockchain en Colombia?

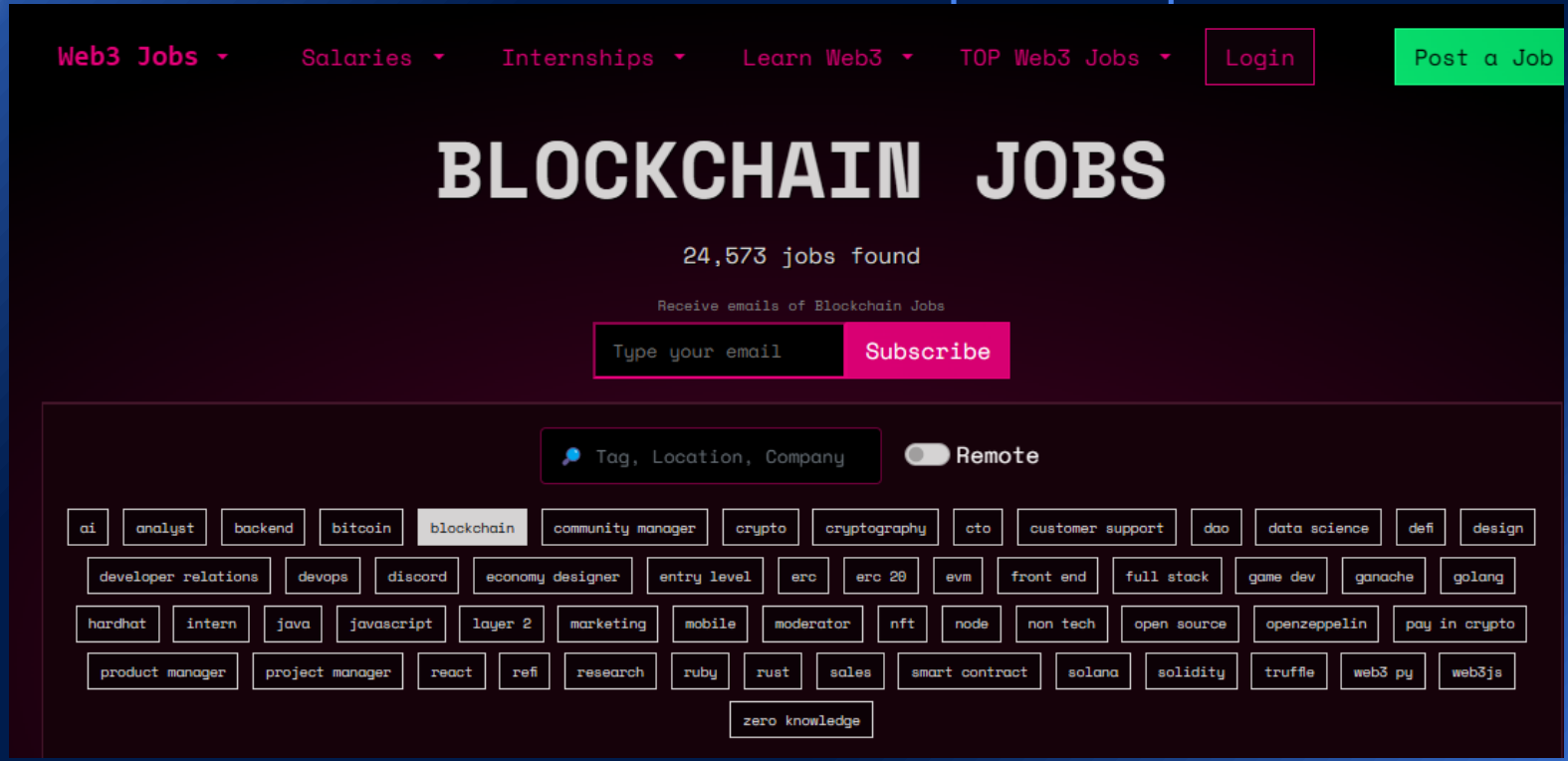
$7.500.000 / Mes

Basado en 1 salarios

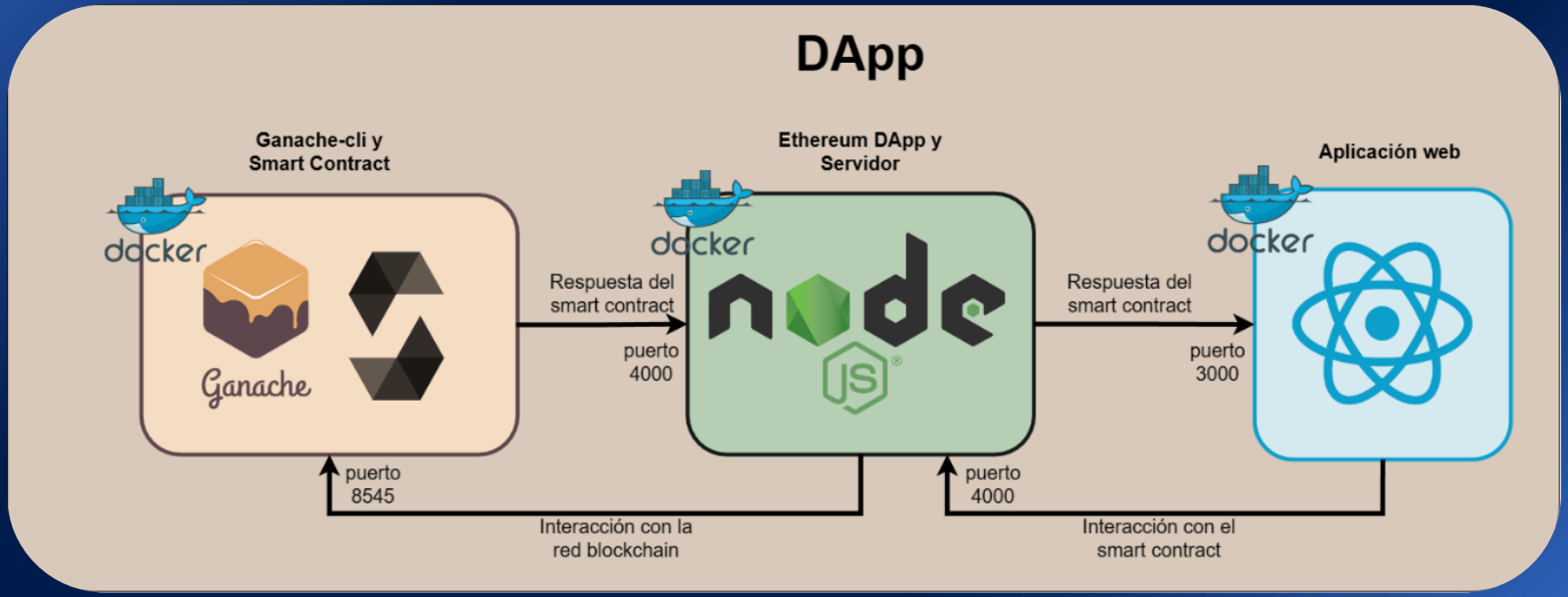
El salario **desarrollador blockchain** promedio en **Colombia** es de **$90.000.000** al año o **$41.209** por hora. Los cargos de nivel inicial comienzan con un ingreso de **$90.000.000** al año, mientras que profesionales más experimentados perciben hasta **$90.000.000** al año.



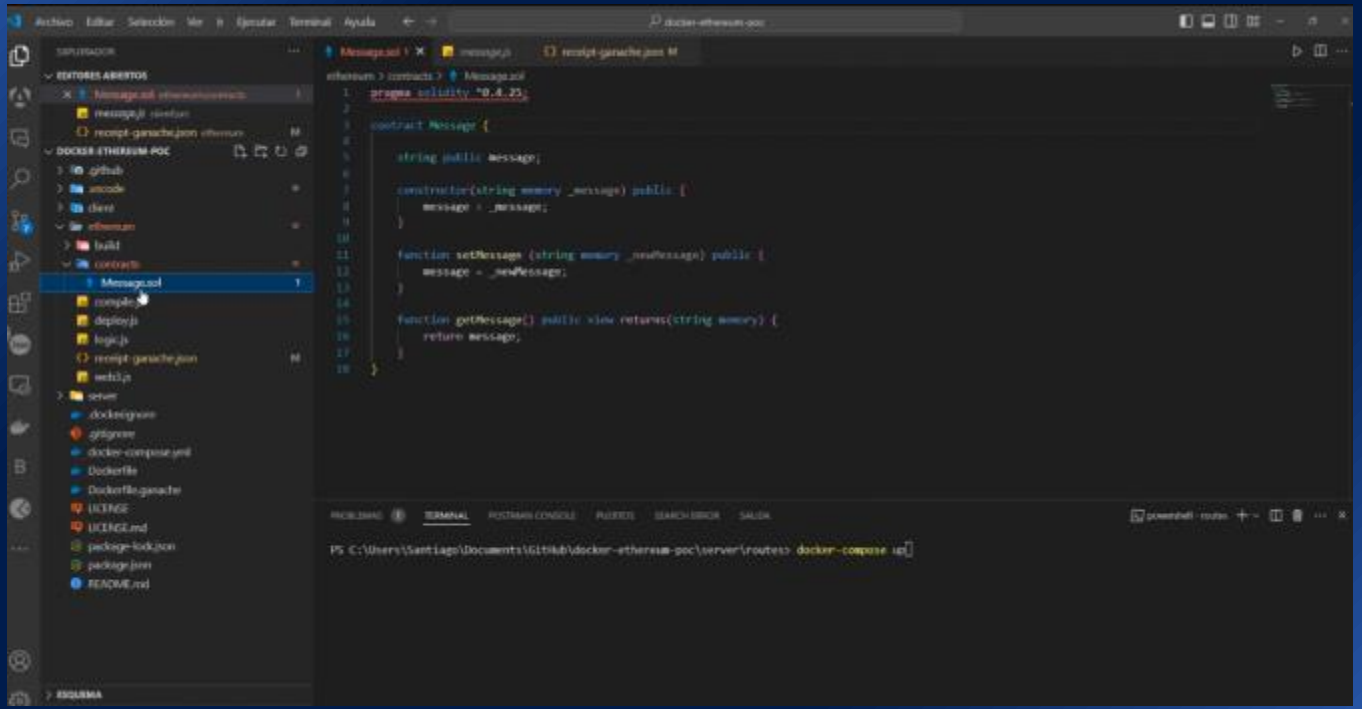
Mercado laboral



Ejemplo práctico



Demostración



https://youtu.be/7Bm39RnRQa4

Gracias!