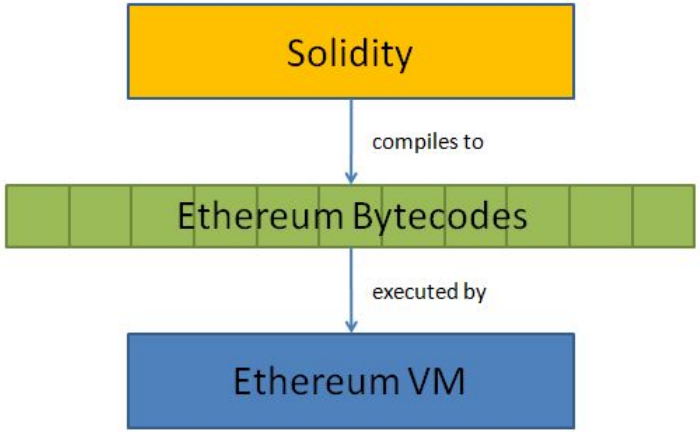
Desarrollo Blockchain Ethereum con Solidity

Módulo 1 - EVM (Ethereum Virtual Machine)

Ethereum Virtual Machine

Ethereum Virtual Machine

- La EVM es la que soporta realmente la ejecución de los smart contracts.
- Al estar estandarizada, pueden existir muchísimas implementaciones diferentes siempre que soporten las operaciones definidas. (por ej. GETH y Parity).
- La EVM trabaja en un lenguaje intermedio, es decir, trabaja en Bytecode.
- Cuando programamos para Ethereum, podemos hacerlo en diferentes lenguajes, da igual ya que finalmente se compilarán a Bytecode que es lo que realmente entiende la EVM.



Solidity vs. Python vs. Viper vs. LLL

- Solidity es el lenguaje que, podríamos decir, es el estándar para el desarrollo de Smart Contracts.
- No es el único, existen otros como LLL, Serpent, Viper, etc.
- Solidity es verdaderamente sencillo, su sintaxis es similar a C++/Javascript (se ha tomado parte de la sintaxis ECMASCRIPT) pero a la vez es diferente (Solidity, a diferencia de JavaScript, es tipado).
- Solidity no solo sirve para Ethereum, sino para otras Blockchain como RSK o Tendermint.
- Serpent y Viper tienen una sintaxis similar.

SOLIDITY

Bytecode

- Cuando compilamos un contrato en Solidity obtendremos dos datos importantes:
  - **ABI:** Interface utilizada para las dApps
  - **Bytecode:** Código de máquina que representa las operaciones codificadas previamente en Solidity. Es similar a un “assembler” y es interpretado por la EVM en tiempo de ejecución.
- El *bytecode* no es legible por el humano, sin embargo, puede realizarse cierta ingeniería inversa sobre el mismo.
- Al almacenarse en la blockchain, el bytecode de un contrato es público y puede utilizarse para validar si el código de un contrato es el mismo que ya fue subido en la blockchain
- No importa qué lenguaje utilicemos, para el nodo lo que importa es el bytecode.

¡Muchas gracias!

¡Sigamos trabajando!