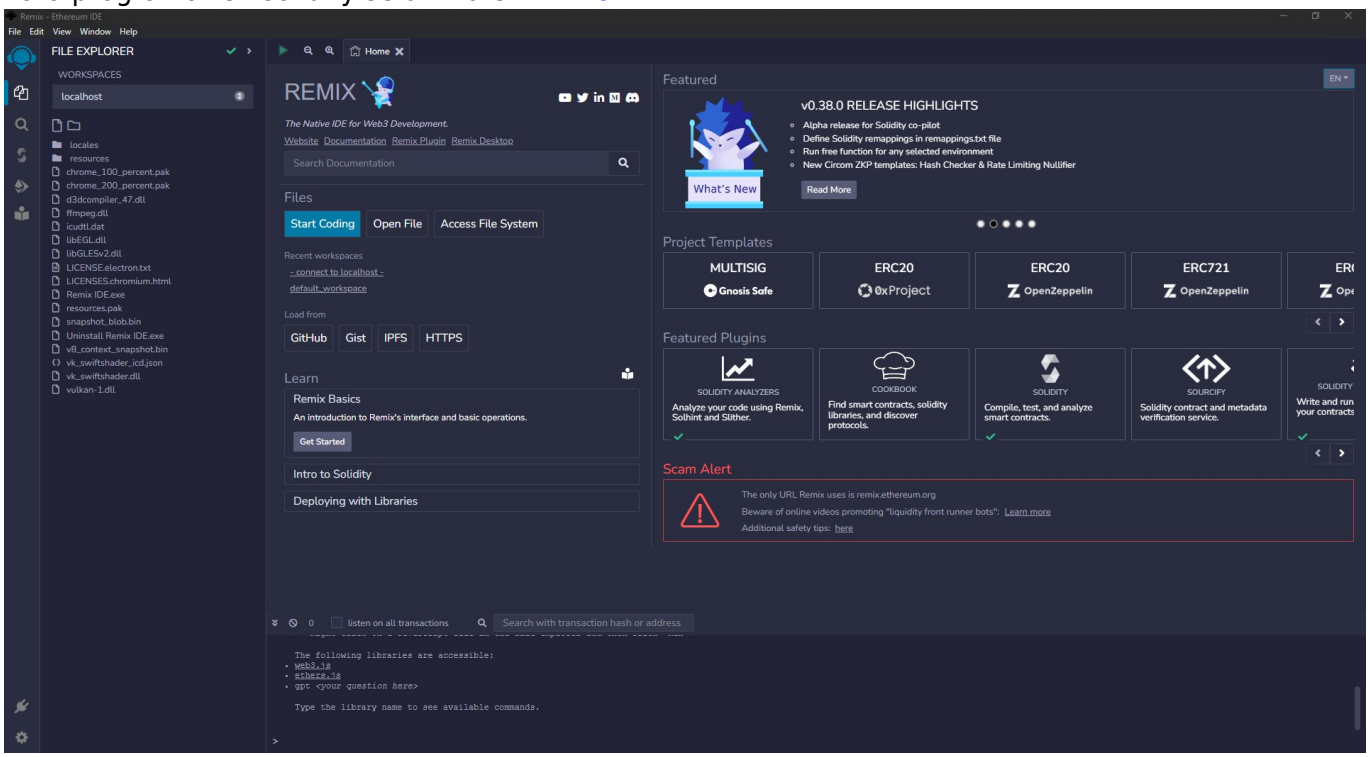
Introducción a la Programación en Solidity

Para programar en solidity se utiliza el IDE Remix:



componentes de un Smart Contract

Pragma

Lo primero que se escribe en un contrato inteligente es la versión pragma, que indica la versión del compilador que debe usar el código. Generalmente se debe poner un rango de versiones que sean compatibles con el código, por ejemplo, si quisiéramos que el código fuera compilable por las versiones entre la 0.6.12 y la 0.9.0 escribiríamos:

```
pragma solidity >=0.6.12 <0.9.0;
```

Para crear un contrato vacío, a continuación del pragma se puede introducir lo siguiente:

```
contract NombreDelContrato{

}
```

Variables y constantes

Las variables de estado se almacenan permanentemente en el almacenamiento de un contrato. Se podría decir que es como si se escribiera en una base de datos. La asignación de memoria es estática y no se puede cambiar. Las variables locales deben ser declaradas dentro de una función y no se

almacenará en la blockchain, mientras que las variables globales proporcionan información sobre la blockchain.

```
contract variable {
  uint IntSinSigno = 10; //Variable de tipo Unisgned integer
}
```

Las constantes son variables que no se pueden modificar, se usan para ahorrar costes de gas (Gas es una cantidad que se cobra por cada transacción.) Las constantes se suelen poner en mayusculas por convención para diferenciarlas de las variables.

```
contract Constante {
  address public constant MY_ADDRESS = 0x29384093845093ifSDOAsdjas;
  uint public constant MY_UINT = 325;
}
```

Operaciones matemáticas

Se pueden realizar más o menos las mismas operaciones que en otros lenguajes de programación:

- Suma: $x+y$
- Resta: $x-y$
- Multiplicación: $x*y$
- División: x/y
- Módulo: $x%y$
- Exponenciación: x^y

En algunas ocasiones es necesario una conversión entre tipos de datos:

```
uint8 x =1;
uint y = 2;

uint8 z = x* uint8(y) //Se convierte y al mismo tipo que X para la operación
```

Estructuras de datos

En Solidity, al igual que en C, tenemos el tipo struct que se puede utilizar para agrupar elementos relacionados, estas estructuras pueden ser declaradas de la siguiente forma:

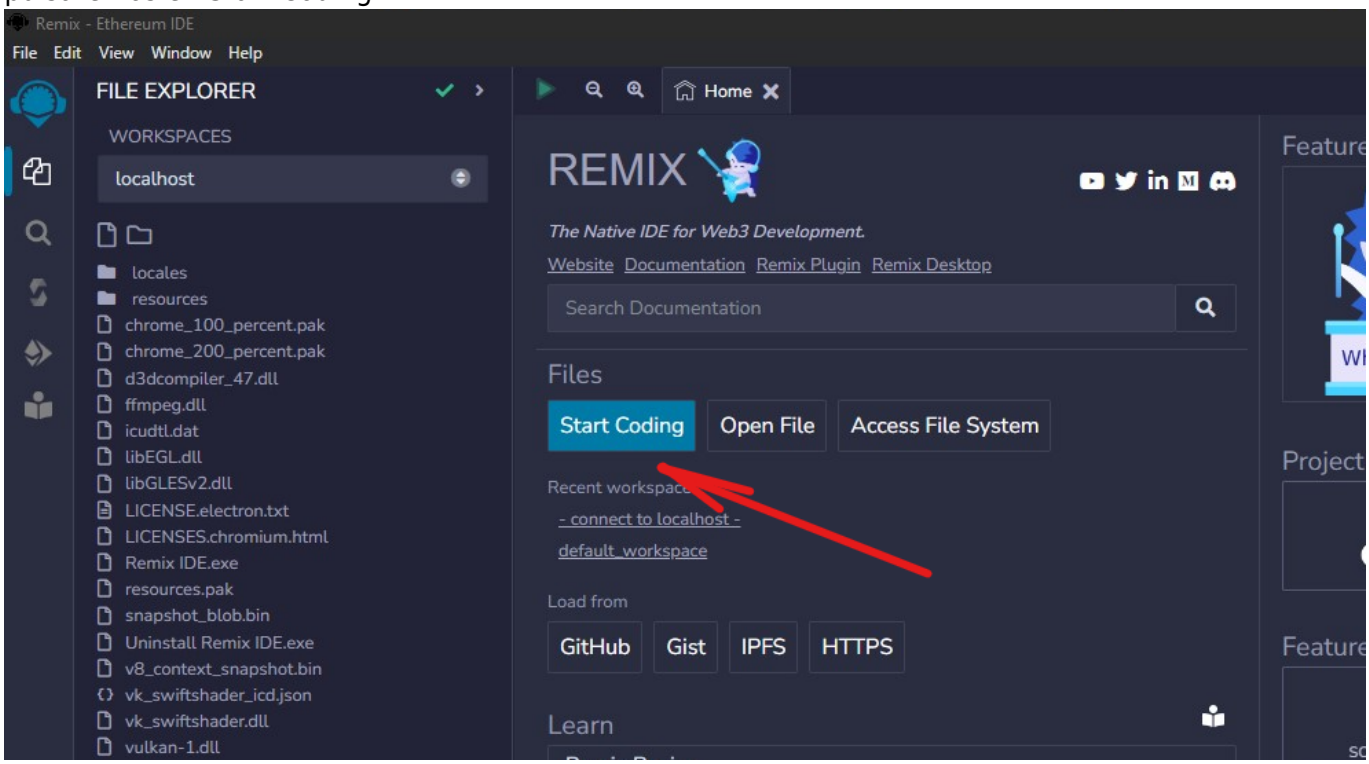
```
struct Persona {
  uint edad;
  uint nombre;
}
```

Para crear una instancia de esta estructura se hace lo siguiente:

```
Persona manuel = Persona(25, "Manuel");
```

Hola Mundo en un Smart Contract

Comenzaremos creando un Smart Contract de prueba con el típico “Hello World”, para ello pulsaremos en Start Coding:



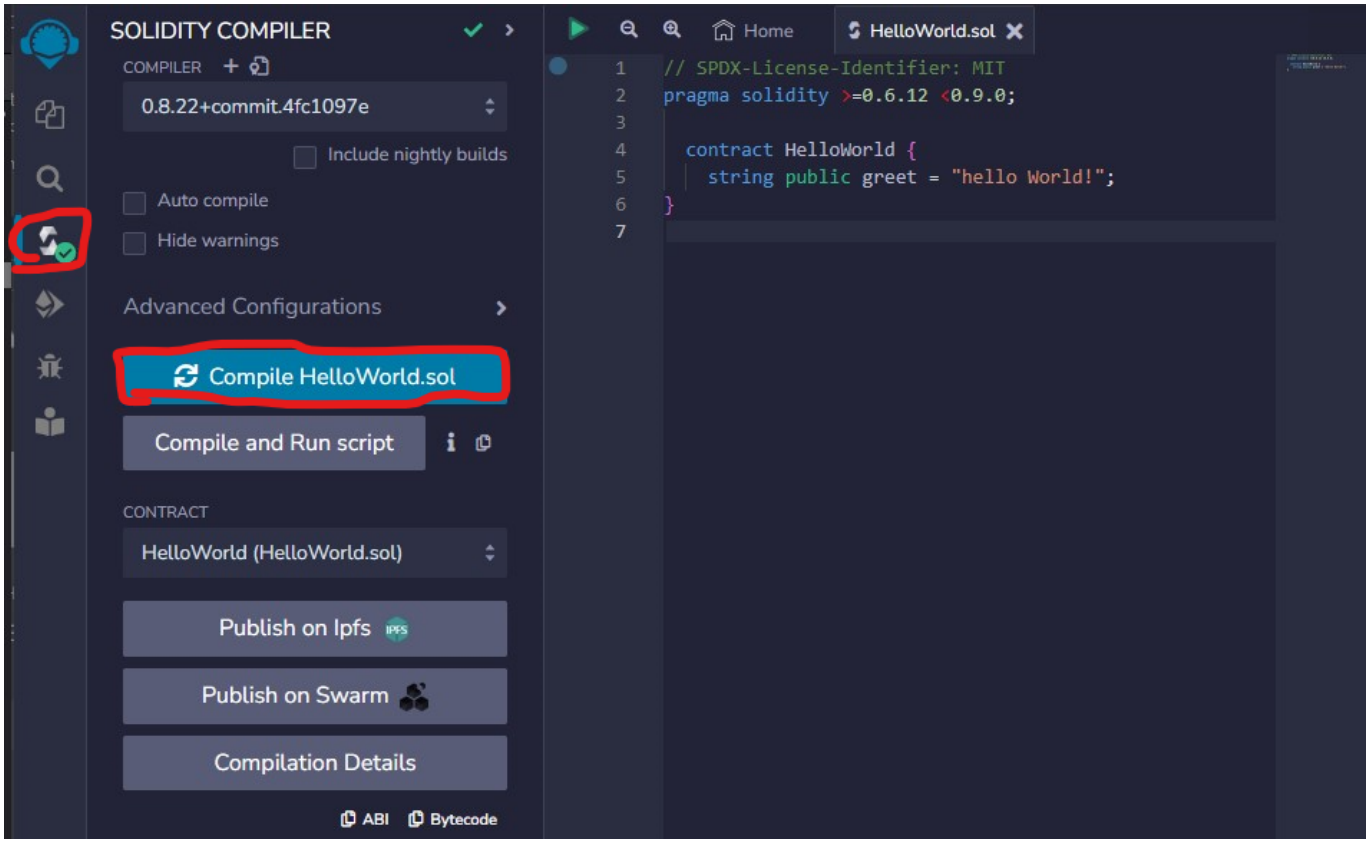
Para hacer un “Hola Mundo” escribimos el siguiente código:

HolaMundo.sol

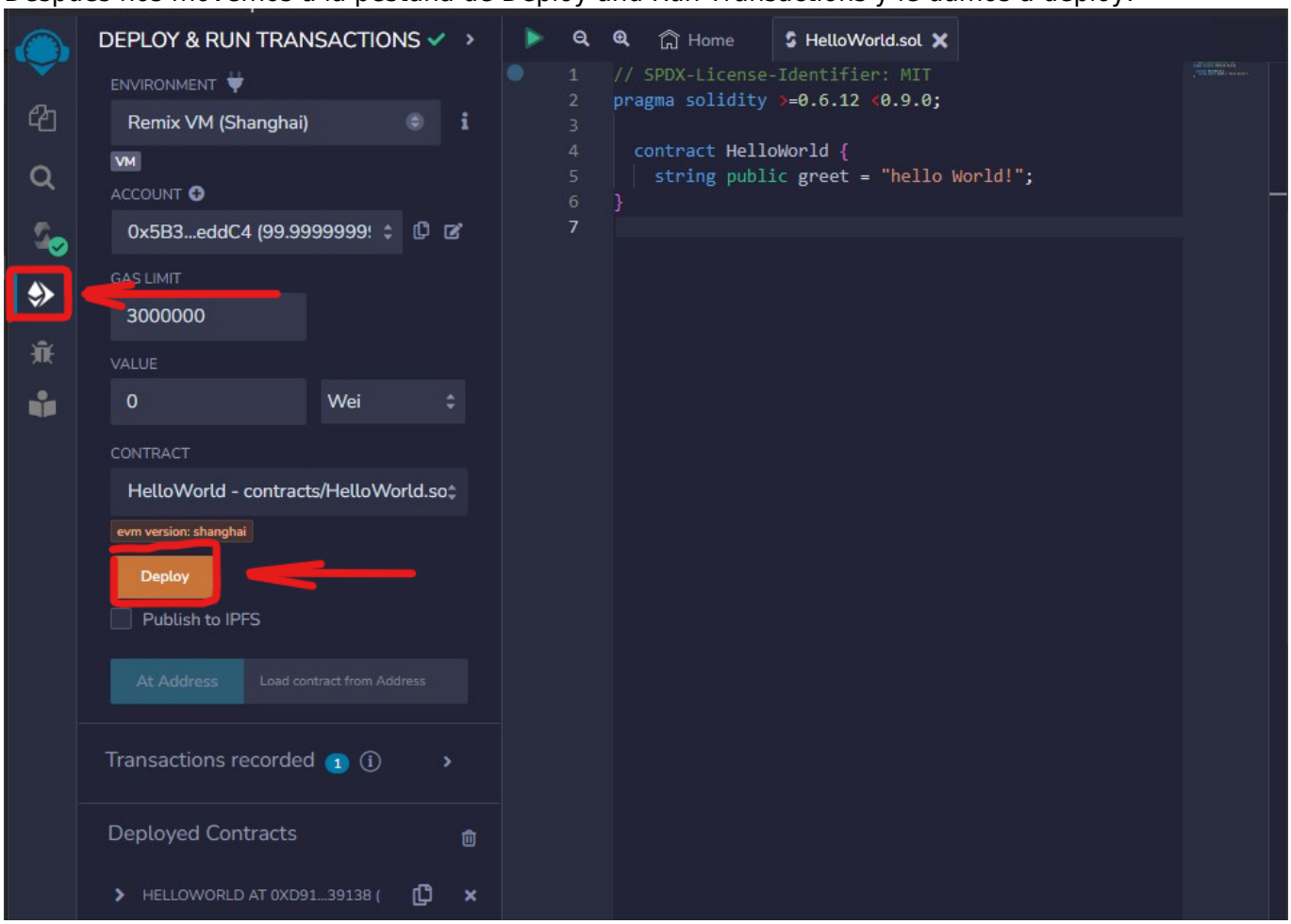
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

contract HelloWorld {
    String public greet = "Hello World!";
}
```

Tras eso iremos a la pestaña de solidity compiler y le daremos a compile:



Después nos movemos a la pestaña de Deploy and Run Transactions y le damos a deploy:



Finalmente podemos ir a la pestaña de Deployed Contracts, seleccionar el contrato que acabamos de enviar y pulsar en el botón greet para ver e mensaje;