Essay – Lab 3 – Florin-George Bărcan

Immediately as we start the 3rd lesson, we are introduced to the fact that in Ethereum, if we deploy our contract, it becomes immutable and we can no longer patch it after release if we are to discover some errors or bugs in code. Due to this fact, we are assured that calling a function will render the same results that the code says it will yield. *The code is law*.

In this lesson we are also pressured to give a huge importance to the security of our contract. We are given the library OpenZeppelin that has the contract Ownable which offers a degree of security for our blockchain by making us, the deployers, the *owner*.

Besides that, we are taught that we can create modifiers for our smart contracts that we can use, such as the *Ownable* modifier *onlyOwner*, which restricts access to certain functions if you are not the owner of the contract. This way we can protect certain core functions inside our contract. The modifier *onlyOwner* is added at the end of our function together with the *external* keyword and can be called only if our contract inherits the *Ownable* contract.

In Ethereum, there is the concept of *gas*. Each time a function is executed, it consumes gas. Gas is bought with Ether, the currency of Ethereum. The gas cost of executing a function depends on the complexity of the function and is based on how many resources are used to perform certain operations. If our code is sloppy, the users will pay millions of money unnecessarily. If we pay attention to structuring the contract well, the cost of gas will be very small compared to our slipshod approach.