

Design and Implementation of a Token-NFT-Liquidity Smart Contract Suite

Roberto Di Rosa, Luca Sforza

Sapienza

3-12-2025

1. Introduction

1. Introduction

—

We aim to present a suite of smart contracts designed to manage NFT auctions using a custom token called SapiCoin.

We used Solidity language for developing this suite of smart contracts.

A Token can be viewed as a class in other programming languages (e.g. Java), however to avoid binding a smart contract to a specific Token we used an abstraction standardized by the Ethereum Community.

In addition to token transfers and auctions, our system includes a liquidity pool implemented using Uniswap v3, which allows users to trade between Ether and SapiCoin.

2. Tokens

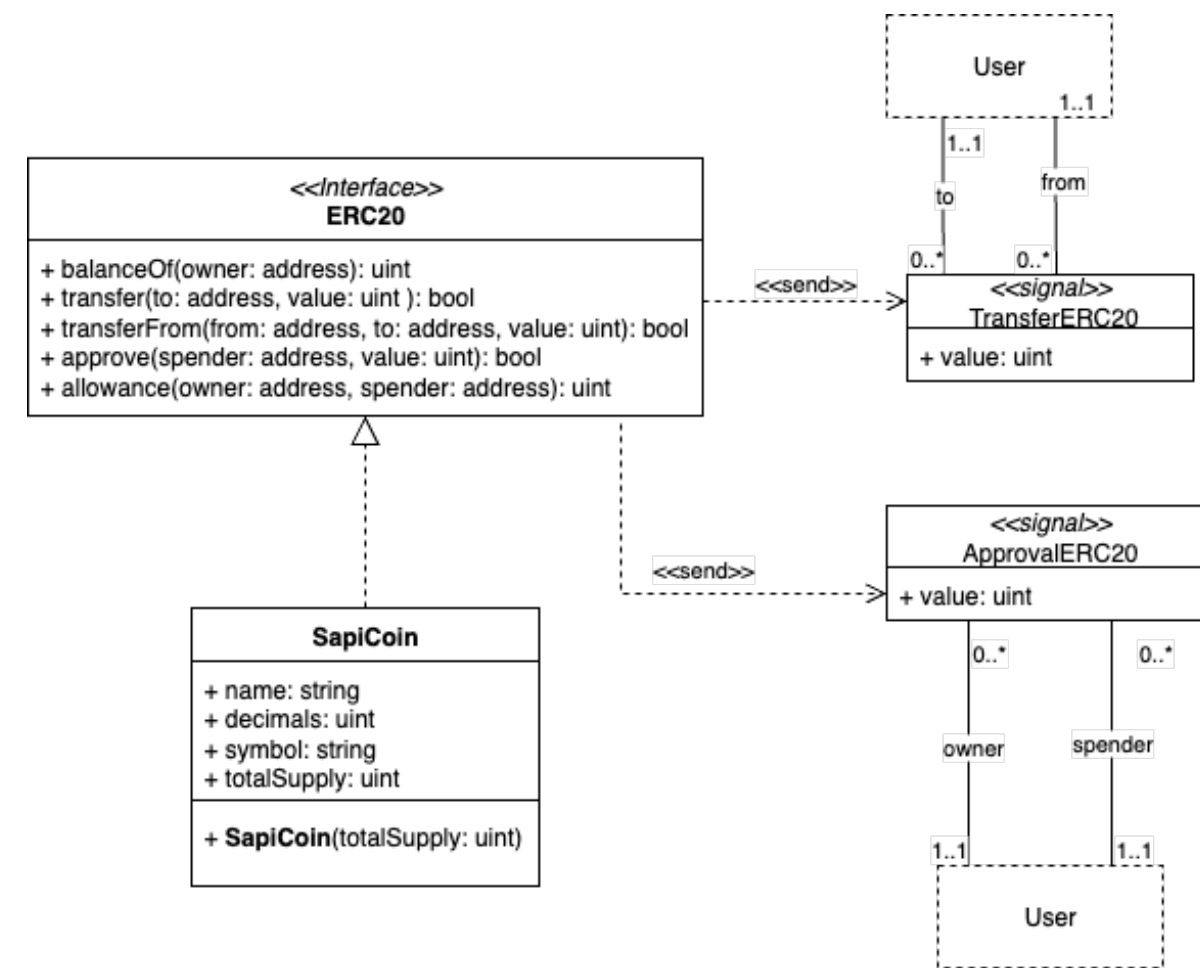
2. Tokens

—

<div data-bbox="31 18 403 81" data-label="Section-Header"><h2>2.1 ERC-20</h2></div> <div data-bbox="120 113 2114 170" data-label="Text"><p>ERC-20 is the standard that represents a token on EVM compatible blockchains.</p></div> <div data-bbox="120 226 2114 522" data-label="Text"><p>There is no central authority responsible for issuing these standards; instead, communities such as Ethereum Magicians provide a space to discuss improvements to the Ethereum standard through EIPs (Ethereum Improvement Proposals).</p></div> <div data-bbox="120 592 680 648" data-label="Section-Header"><h3>2.1.1 Polymorphism</h3></div> <div data-bbox="120 680 1097 737" data-label="Text"><p>Achieving this requires polymorphism.</p></div> <div data-bbox="120 793 1971 932" data-label="Text"><p>The Ethereum Virtual Machine does not provide instructions for handling abstract classes, but the VM itself is polymorphic.</p></div>	<div data-bbox="2325 18 2643 81" data-label="Section-Header"><h2>2. Tokens</h2></div> <div data-bbox="2325 113 2679 170" data-label="Text"><p>— 2.1 ERC-20</p></div>
--	--

<div>2.1 ERC-20</div> <div>When a method of a smart contract is invoked through a transaction, the memory address of the method is accessed by computing the hash of the function signature. If the invoked method does not exist, a default fallback function is executed (which can be overridden).</div> <div><pre>contract SapiCoin is ERC20 { /* Implementation */ }</pre></div> <div>So Solidity exploit this behavior to defines interfaces. So ERC-20 can be viewed as an Interface.</div>	<div>2. Tokens</div> <div>— 2.1 ERC-20</div>
--	--

Design of the standard ERC-20



1. A Model Driven Architecture approach provides a structured method for designing blockchain smart contracts. Using UML diagrams—such as Class and State Machine diagrams—developers can model both the structure and behavioral logic of smart contracts across multiple abstraction layers, improving clarity, analysis, and maintainability.
2. In this UML we have the ERC20 seen as an Interface and SapiCoin is a Class that implementes the methods. The methods are:
 1. `balanceOf`: returns the balance locked in a address
 2. `transfer`: transfer token to another address
 3. `approve`: Approve a Spender to spend our money
 4. `allowance`: returns the amount of token that a spender can spend
 5. `transferFrom`: trasfer token from an address to another if we are

2.2 Token ERC-721

2. Tokens

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

2.2.1 ERC-165

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

2. Tokens

— 2.2 Token ERC-721

3. Auction

3. Auction

—

**Another variant with primary color in
background...**

3. Auction

—

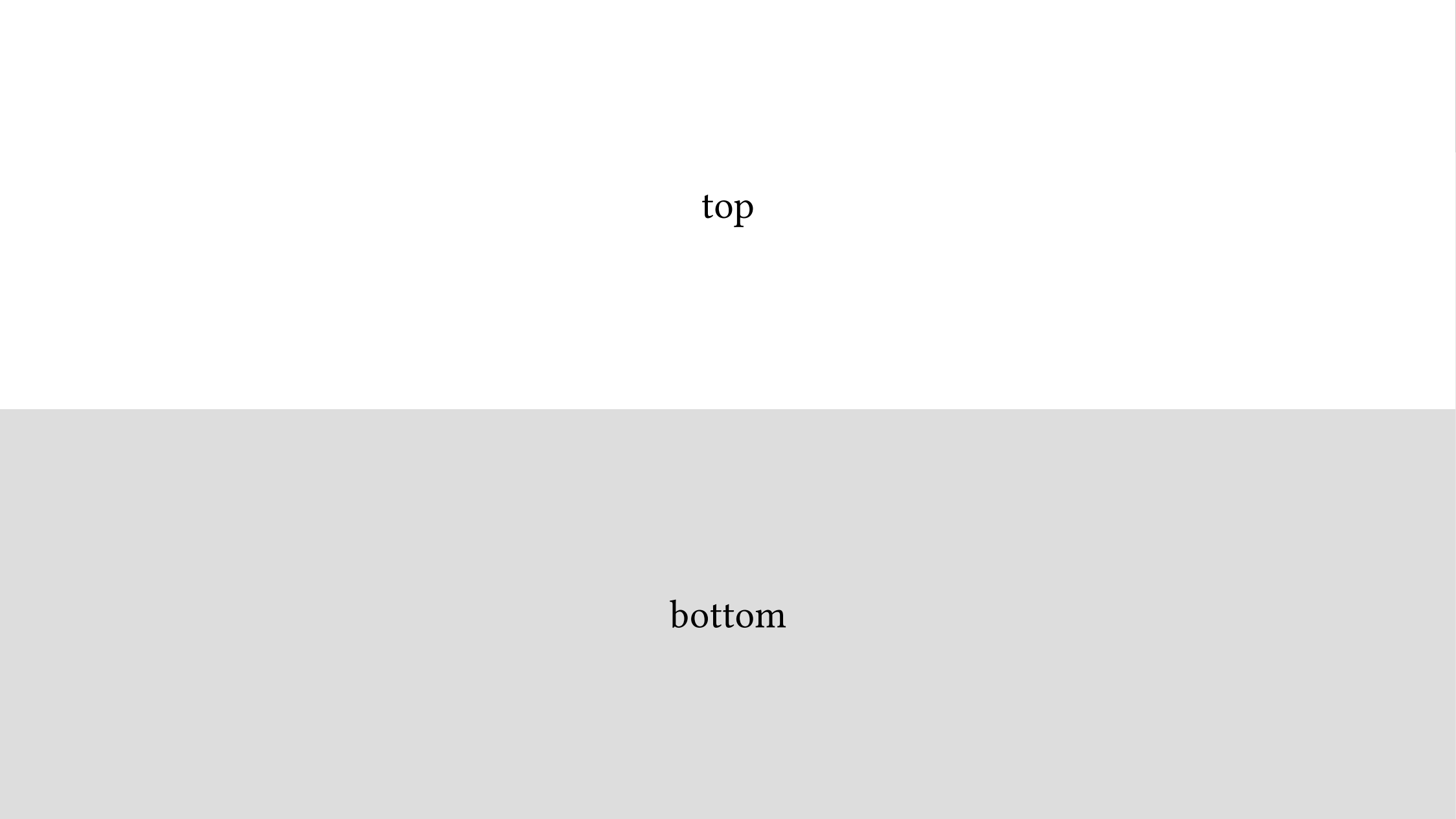
left

middle

right

3. Auction

—



3. Auction

—

Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	3. Auction —
Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	
Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

—

4. Implementation details

**Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do.**

4. Implementation details

—

5. Liquidity Pool

**Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do.**

5. Liquidity Pool

—

6. Conclusions

**Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do.**

6. Conclusions

—