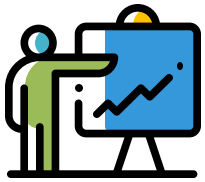
BAZINGA!

Marketplace for Digital Assets


I'm going to change the world.
For the better, right?
For the better, right?

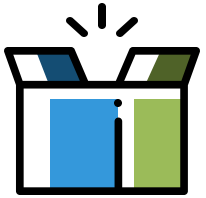
1. PROBLEM

PROBLEM



UNDEFINED OWNERSHIP

DISADVANTAGE FOR CREATORS



COPYRIGHT LAWS IN GENERAL

2. SOLUTION AND PRODUCT

THE SOLUTION

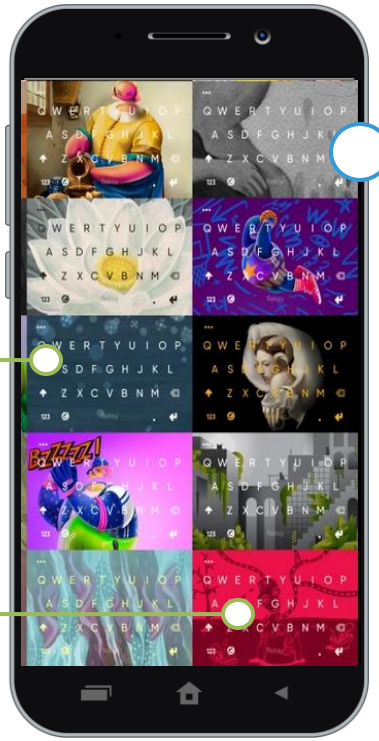
PLACE FOR THE OWNERSHIP ADMINISTRATION

MARKETPLACE FOR IMMATERIAL ASSETS

HOW IT WORKS:

„OWNERSHIP REGISTRATION“

Creators are able to register their Ownership on the platform, so that the copyrights are not violated any more

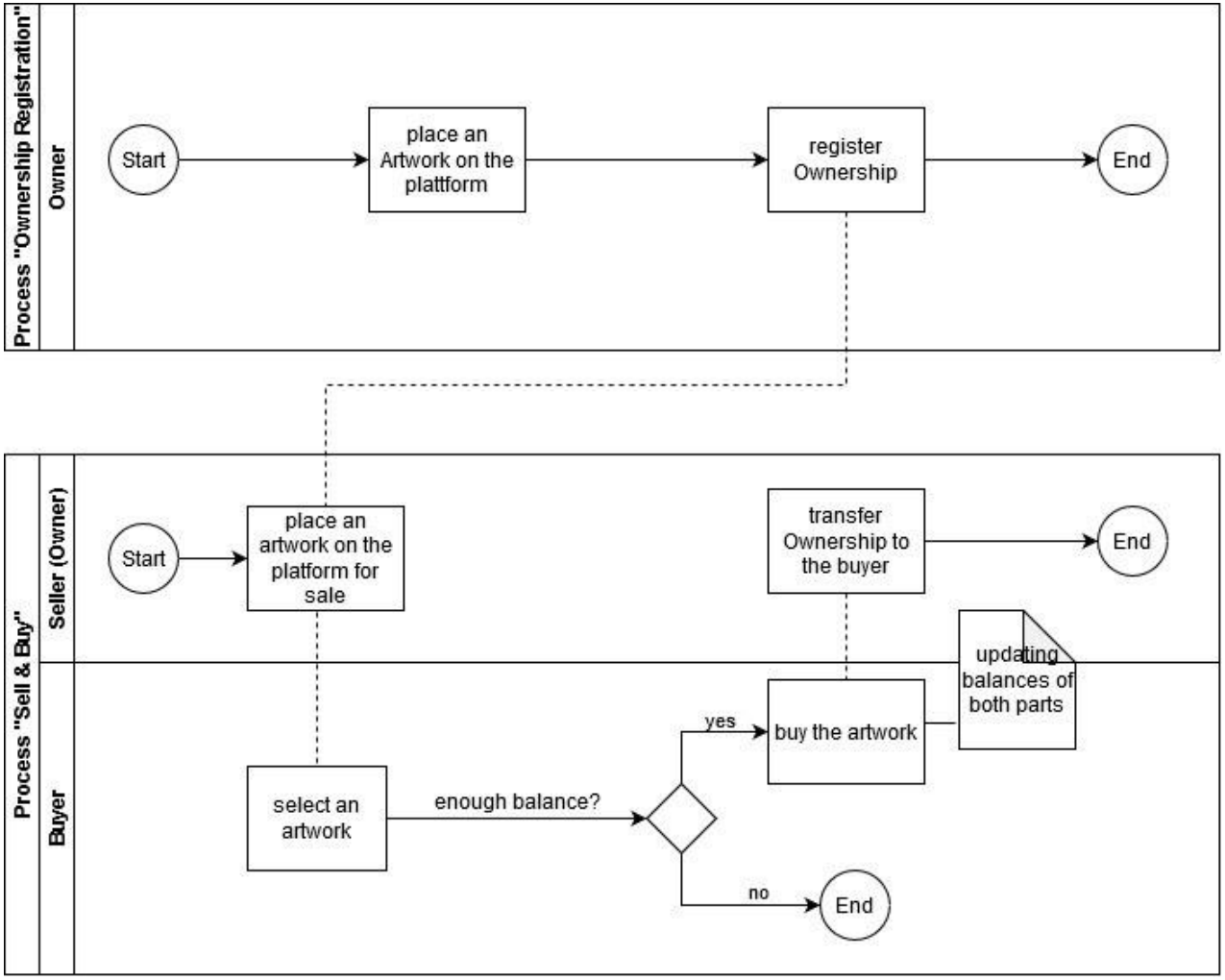


„SELL & BUY“

Users who are interested in buying an immaterial asset has a possibility to choose one and become its owner in exchange for a transaction fee

PROCESS MODEL

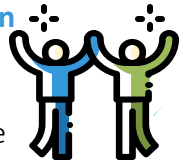
BPMN

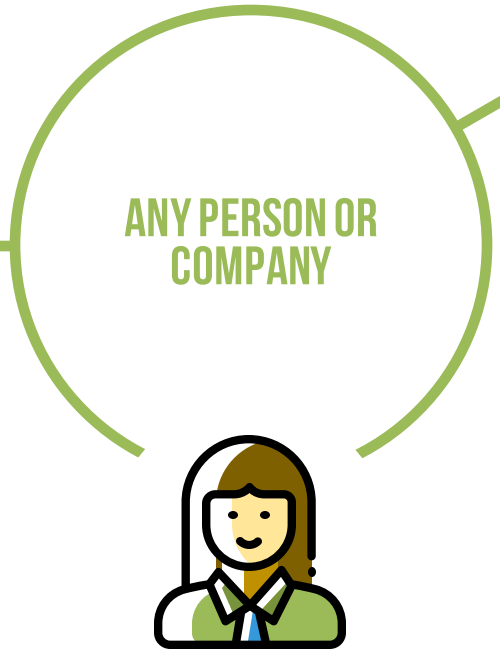



3. MARKET

WHO ARE YOUR CUSTOMERS?

Ownership Administration
Artists
15+ years
50% of our customer base





Large selection of immaterial assets, Legal process in terms of copyrights

Buyers
15+ years
50% of our customer base


4. DEMO

SMART CONTRACTS

Marketplace

```
7   /// @title Platform for registrating, buying and selling artworks
8   /// @notice Is created by the marketplace creator
9   contract Marketplace is Ownable{
10      address[] public artworks;
11      uint constant public registrationFee = 0.01 ether;
12
13      /// @notice Is used for marketplace and artwork
14      constructor(address _owner) {
15          owner = _owner;
16      }
17
18      /// @notice Register new artwork
19      function registerArtwork(string memory _artworkName, string memory _artworkUrl) public payable {
20          require(msg.value == registrationFee, 'Need to send a fee of 0.01 ether.');
21
22          payable(owner).transfer(msg.value);
23
24          Artwork artwork = new Artwork(_artworkName, _artworkUrl);
25          artwork.transferOwnership(msg.sender);
26          artworks.push(address(artwork));
27      }
28
29      /// @notice Get all registered artworks
30      function getArtworks() public view returns(address[] memory) {
31          return artworks;
32      }
33  }
```

SMART CONTRACTS

Artwork

```
/// @title Representation of an immaterial asset
/// @notice Will be created when registering a artwork in the marketplace
contract Artwork is Ownable {
    string public artworkName;
    string public artworkUrl;
    bytes32 public artworkHash;
    uint public artworkPrice;
    bool public isArtworkForSale;

    /// @notice Set attributes
    constructor(string memory _artworkName, string memory _artworkUrl) Ownable() {
        artworkName = _artworkName;
        artworkUrl = _artworkUrl;
        artworkHash = keccak256(abi.encode(_artworkUrl));
        isArtworkForSale = false;
    }

    /// @notice Offer artwork for sale
    function sellArtwork (uint _artworkPrice) public onlyOwner {
        artworkPrice = _artworkPrice;
        isArtworkForSale = true;
    }

    /// @notice Withdraw offer
    function cancelSellArtwork () public onlyOwner {
        isArtworkForSale = false;
    }

    /// @notice Transfer price and change ownership
    function buyArtwork() public payable {
        require(owner != msg.sender, 'Owner can not buy their own artwork.');
        require(isArtworkForSale, 'Artwork is currently not for sale.');
        require(msg.value == artworkPrice, 'Need to send the exact price.');

        payable(owner).transfer(msg.value);

        transferOwnershipInternal(msg.sender);
        isArtworkForSale = false;
    }
}
```

SMART CONTRACTS

Ownable

```
4   /// @title Contract module which provides a basic access control mechanism
5   /// @notice Is used for marketplace and artwork
6   contract Ownable {
7     address public owner;
8
9     /// @notice Set owner
10    constructor() {
11      owner = msg.sender;
12    }
13
14    /// @notice Check if caller is owner
15    modifier onlyOwner() {
16      require(msg.sender == owner, "You are not the Owner.");
17      _;
18    }
19
20    /// @notice Transfer ownership (public, with modifier)
21    function transferOwnership(address newOwner) onlyOwner public {
22      if (newOwner != address(0)) owner = newOwner;
23    }
24
25    /// @notice Transfer ownership (internal, no modifier)
26    function transferOwnershipInternal(address newOwner) internal {
27      if (newOwner != address(0)) owner = newOwner;
28    }
29  }
```

THANK YOU FOR YOUR ATTENTION!

Group 3:

Fabian Rittmeier

Simon Hirner

Vitaliia Savchyn