09 - Ideas

Overview

- Intro
- Think
  - 100 Words
  - 10 Sentences
  - Rate & Select
- Sketch
  - Idea
  - Requirments
  - Stories
  - Diagrams
- Paint
  - Write
  - Build
  - Test

🧠 Think

100 Words

- Rich pictures

Earings, Shirt, Chair, Microphon, Technican
Phala, Computiation, Finance, Offchain, Polygon
Mandala, Nightshift, Google Meet, Starbucks, Fanta,
Swiss, Berry, Communication, Thailand, TukTuk,

10 Sentences

Theme	Word	Phrase	Rating
Marketing	Berry	Your animals of exelcene	5
Right Items	Fanta	Right items on the go	6
Computiation	Blender	Mint on the fly	5
Call out	Microphon	You cards on the spot	2

Rate & Select

The winner is `animo.xyz`

📝 Sketch

Idea

- Problem: The marketing value of your animal is hard to share
- Idea: Tokenize your public persona animal
- Solution: `animalo.xyz`

Requirements

Functional Requirments

- The App must connect to Instagrams
- The App must show a selection of prominent animals
- The Owner must be ablet to tokenize the rewards

No Functional Requirmetns

- The App must take care of the animal automaticall

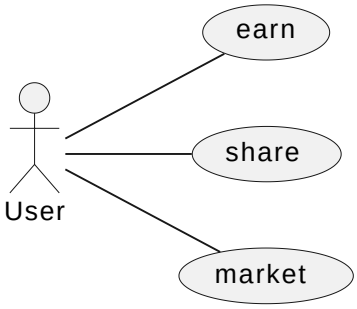
User Stories

- As a User I want to earn money
- As a User I want to share my animal sucess
- As a User I want to I want to market my product

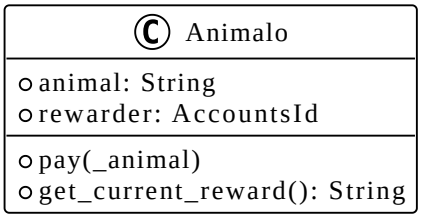
> Currently creates a animal rewards system.
> I bet my ass off that exist in the web2 way and uses a centralized entity.

Diagram

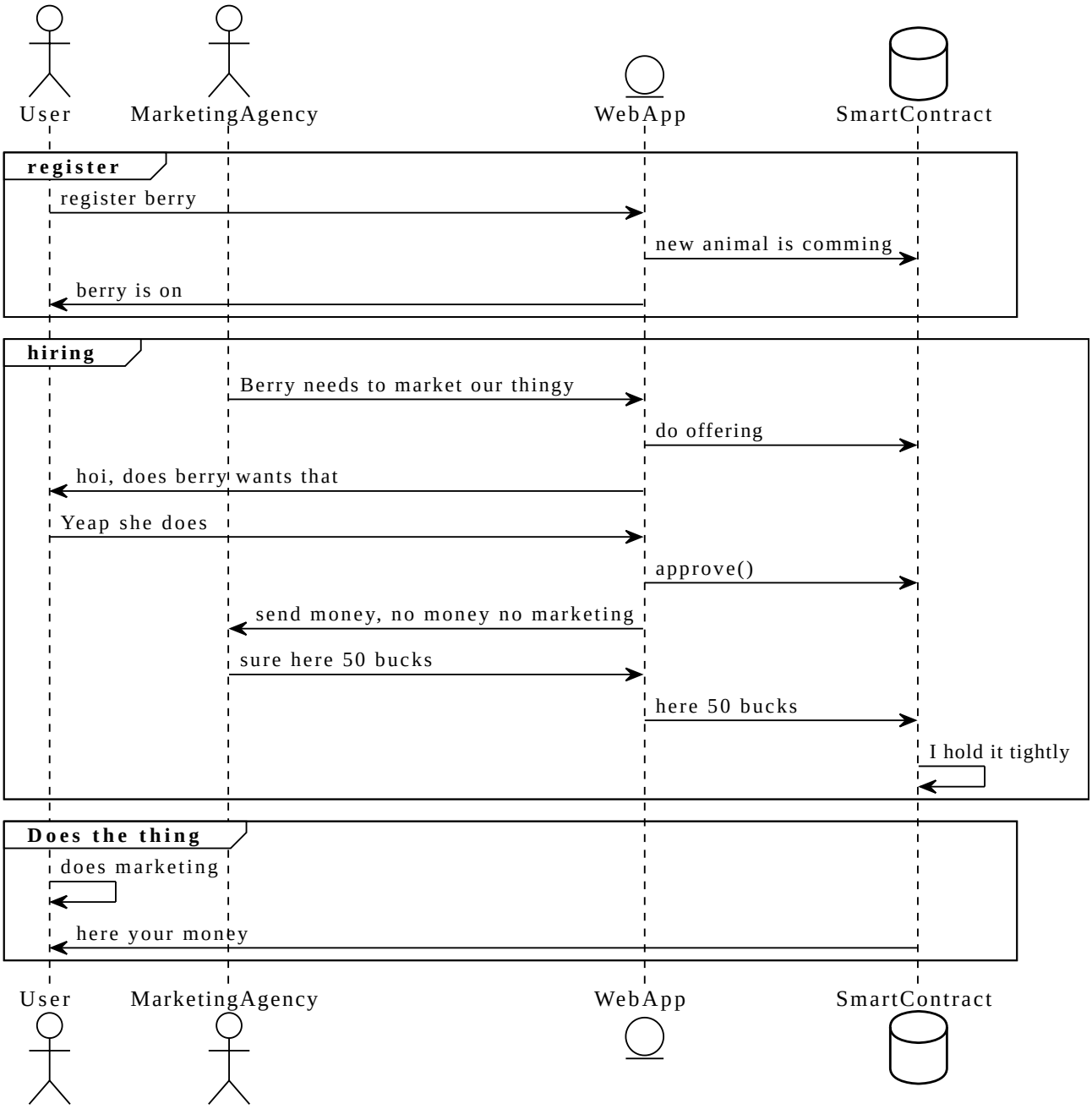
Use Case Diagram



Class Diagramm



Sequence Diagram



🎨 Paint

```
#![cfg_attr(not(feature = "std"), no_std, no_main)]

#[ink::contract]
mod animalo {
    use ink::prelude::string::String;

    #[ink(storage)]
    pub struct Animalo {
        value: bool,
        animal: String,
        rewarder: AccountId,
    }

    impl Animalo {
        #[ink(constructor)]
        pub fn new() -> Self {
            Self {
                value: true,
                animal: "Berry".into(),
                rewarder: AccountId::from([0xff; 32]),
            }
        }

        #[ink(message)]
        pub fn get_current_rewarder(&self) -> AccountId {
            self.rewarder
        }

        #[ink(message)]
        pub fn pay(&self) {}

        #[ink(message)]
        pub fn flip(&mut self) {
            self.value = !self.value;
        }

        #[ink(message)]
        pub fn get(&self) -> bool {
            self.value
        }
    }
}
```