Blockchain Smart Contract Project

Nguyen Van Tien

Kim Thi Lien

Le Thi Huynh Thao

30/05/2021

# Project Overview

This project implements a simple clone of Twitter, where any users can send tweets on the web page as well as update, delete their tweets. All tweet operations are sent to a blockchain network and executed by a smart contract. Each operation has a small cost to perform. A user, therefore, must own a blockchain wallet with positive balance to create, update or delete a tweet.

The project is split into two components: the smart contract, and the front-end.

# 

# Smart Contract: TweetFactory

There is only a single smart contract that handles all operations. This contract is named TweetFactory, coded in Solidity programming language, and is compiled by Truffle suite.

The contract implements 4 functions, which correspond to Create-Read-Update-Delete operations of the project:

* function createTweet(string memory \_content) public
* function readTweets() public view returns (Tweet[] memory)
* function updateTweet(uint256 \_id, string calldata \_content) public ownerOf(\_id)
* function deleteTweet(uint256 \_id) public ownerOf(\_id)

The contract source code can be found in: **./contracts/TweetFactory.sol.**

Test: TweetFactory.test.js

There is only one file that test all the above functions in the smart contract:

* Creating a new Tweet: this will check if the function createTweet() returns the right content and the right owner of a created tweet.
* Reading all tweets: check if the function readTweets() returns all tweets.
* Updating a tweet:
  + Check if the updated tweet has true content.
  + Check if only owner has permission to update a tweet
* Deleting a tweet:
  + Check if the deleted is no longer exist!
  + Check if only owner has permission to delete a tweet.

Docker set up:

* Using docker-compose to set up to run test, migrate, deploy and coverage. Each has one Dockerfile (ie: test.Dockerfile,…) and docker-compose file (ie: docker-compose.test.yaml)
* Set up scripts in the package.json to run docker-compose:

"docker:migrate": "docker-compose -f docker-compose.migrate.yaml up",

"docker:test": "docker-compose -f docker-compose.test.yaml up",

"docker:deploy": "docker-compose -f docker-compose.deploy.yaml up",

"docker:coverage": "docker-compose -f docker-compose.coverage.yaml up",

"docker:start": "docker-compose -f docker-compose.start.yaml up",

The following command lines can be used to compile and build :

* Compile and Migrate: **npm run docker:migrate**
* This command will compile and migrate the contract to the Ropsten network. The details of the Ropsten network can be modified in the **.env** file.
* Test: **npm run docker:test**
* Coverage: check how code lines are coveraged: **npm run docker:coverage**
* Start the website application: **npm run docker:start**

**! Note:** in the first time of starting the web application, you need to agree with continue with a new image by entering y or Y in command line.

# 

# Front-End: Twitter Clone

The front-end implements a simple clone of Twitter and is completed in React and JavaScript technologies. In order to save time, we *copied and reused* the original front-end implementation from the following repository: **https://github.com/tnguyen42/twitter-front**.

The front-end source code can be found in:

* Source code: **./src**
* Resource: **./public**

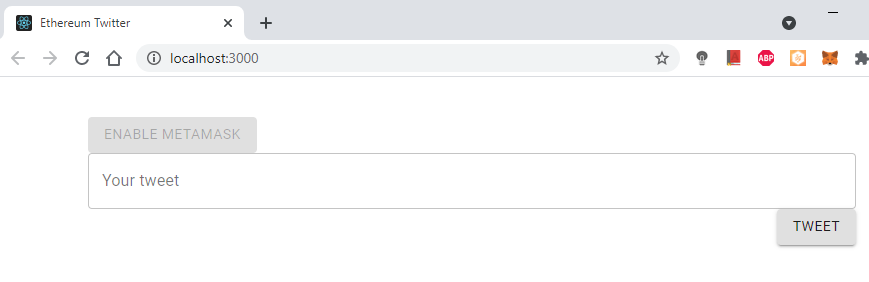
The following command line can be used to run and test the front-end on a local machine:

* **npm run docker:start**
* The web application can be accessed at URL [**http://localhost:3000/**](http://localhost:3000/)after running this command.

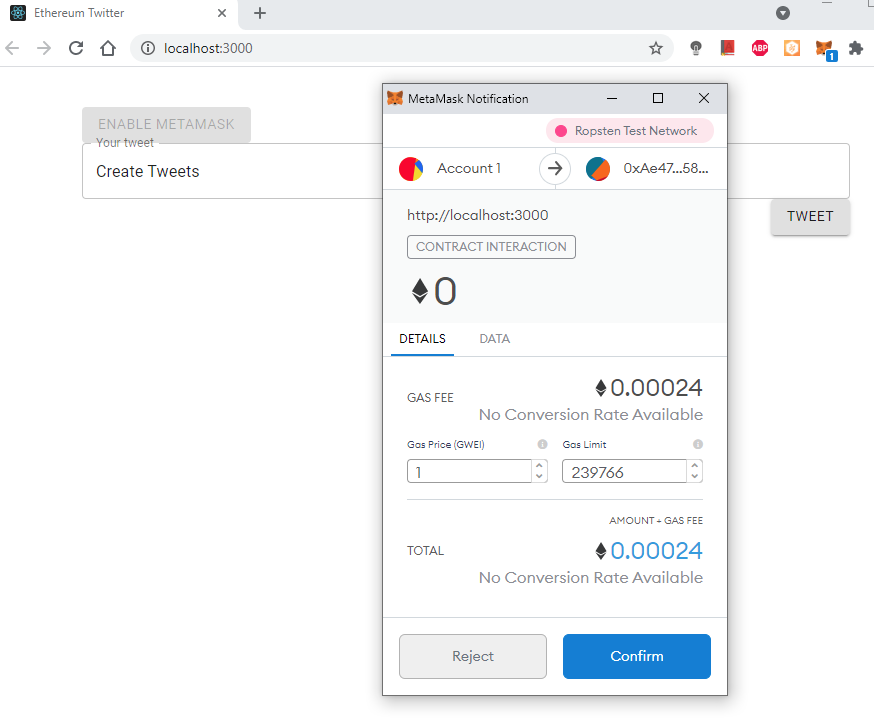
In addition, to use or test this web application, a user must set up the MetaMask plugin in his browser application (Firefox, Chrome, …). The MetaMask plugin allows the user to create his or her blockchain accounts and use their Ether currency to perform the Tweet operations. After account setup, a user can request 1 Eth at the following page: <https://faucet.ropsten.be/> or 5 Eth from <https://faucet.dimensions.network/>

## Screenshots

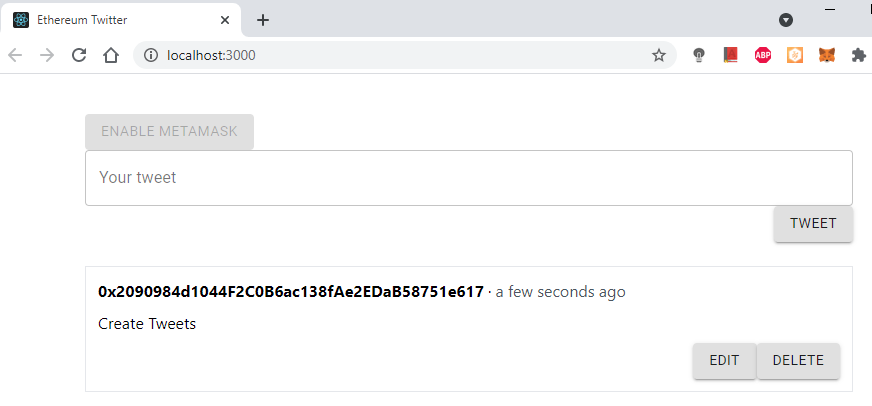
Initial state of the web application:



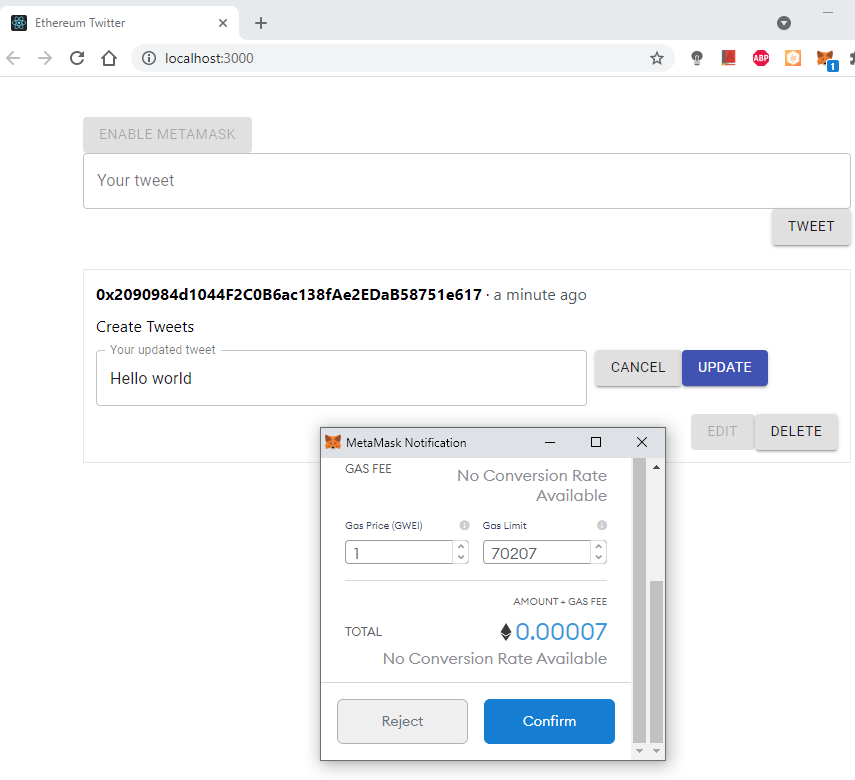
Create a tweet with a cost of 0.00024 Eth:



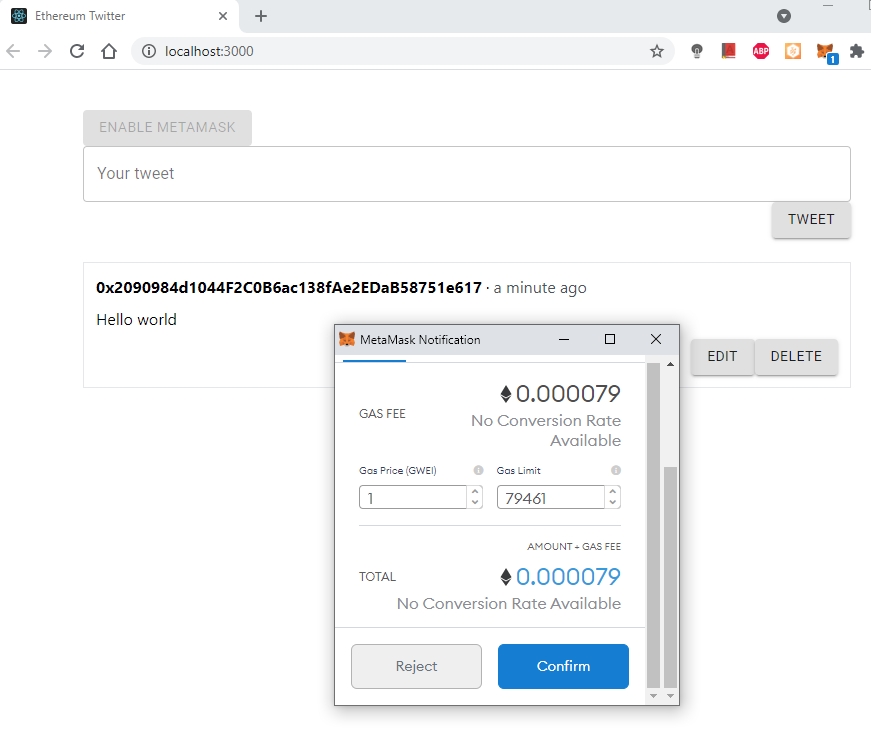
The new tweet is created:



Update a tweet



Delete a tweet



Multiple tweets in descending created timestamp

