RESUMO
ABSTRACT

1. introdução

Parte inicial do texto, na qual devem constar o tema e a delimitação do assunto tratado, objetivos da pesquisa e outros elementos necessários para situar o tema do trabalho, tais como: objetivos do trabalho (geral e específico), caracterização do problema, justificativa, embasamento teórico (principais bases sintetizadas) e estrutura do trabalho, tratados de forma sucinta. Salienta-se que os procedimentos metodológicos e o embasamento teórico são tratados, posteriormente, em capítulos próprios e com a profundidade necessária ao trabalho de pesquisa.

Subseções sugeridas:

1.1 Problema
1.2 justificativa
1.3 objetivos
    1.3.1 Objetivos Gerais
    1.3.2 Objetivos Específicos
1.4 organização do texto

2. FUNDAMENTAÇÃO TEÓRICA

O capítulo de Fundamentação Teórica deve definir os principais conceitos do trabalho e apontar os **trabalhos relacionados**. Contém a exposição ordenada e pormenorizada do assunto. É composta de revisão de literatura, dividida em seções e subseções descritas detalhadamente. Cada seção ou subseção deverá ter um título apropriado ao conteúdo.

3. PROPOSTA

No capítulo de proposta é elaborada a proposta propriamente dita. São apresentados com clareza os objetivos a serem alcançados. Além disso, deve ser apresentados:

A proposta deste trabalho é desenvolver um Decentralized App (DAPP) baseado na Web para a denúncia de crimes ambientais. Para tal, será utilizado uma abordagem de desenvolvimento de DAPP, onde as informações fornecidas pelo denunciante serão armazenadas dentro de uma blockchain, evitando assim que haja adulteração ou exclusão das informações por pessoas mal-intencionadas.

3.1 TECNOLOGIAS E FERRAMENTAS

- Apresentar uma descrição das tecnologias que serão utilizadas
- Descrever as ferramentas a serem utilizadas

3.1.1 BLOCKCHAIN

A tecnologia Blockchain (BC) tem sido bastante difundida durante a última década. Uma das principais motivações para isso se dá por conta do seu papel na difusão das criptomoeadas, como o bitcoin. *A*

BC é uma estrutura de dados distribuída, gerenciada por um conjunto de nós conectados, caracterizada pelos seguintes elementos:

- *é redundante, sendo que cada nó possui uma cópia da BC;*
- *é append-only (somente adição), uma vez que escrita, a informação não pode ser alterada ou deletada;*
- *o estado da BC mudada pelo envio de transações pela rede, numa BC pública, todo mundo pode enviar uma transação;*
- *todas as transações são verificadas pelos nós alcançados, inválidos são ignorados;*
- *as transações válidas são tipicamente guardadas em uma sequência ordenada de blocos, cujo a criação é gerenciada por um algoritmo de consenso entre os nós da rede;*
- *todas as transações são enviadas de um endereço único, que por sua vez é calculado a partir de uma chave pública. Somente o proprietário da chave privada associada a ela pode assinar as transações provenientes desse endereço usando criptografia, validando-os;*
- *Se o blockchain for capaz de executar SCs, uma transação pode criar um SC, ou executar uma de suas funções públicas; dentro neste caso, a função é executada por todos os nós, quando a transação é avaliada – a execução do programa é a execução da transação.*

O Algoritmo de Consenso (AC), mencionado anteriormente, tem participação fundamental quando levamos em consideração o funcionamento da rede que gerencia a BC. Dentre os principais e mais conhecidos AC's utilizados atualmente temos o Algoritmo de Prova Por Trabalho, ou mais conhecido pelo seu termo em inglês de Proof of Work (PoF), e o Algoritmo de Prova Por Participação, ou em inglês Proof of Stake (PoS). Cada um deles possuem diferenças que afetam da velocidade de criação de blocos à participação dos nós da rede.

Neste trabalho será utilizado a Ethereum, que tem em sua estruturação a utilização do PoW. Essa BC *tem a intenção de prover uma linguagem de programação Turing completa que permite a criação de Smart Contracts (SC), ou Contratos Inteligentes (CI) em português, que podem ser usados para a codificação de funções de transição de estado arbitrária*. Ela tem a motivação de *fundir e melhorar os conceitos de scripting, altcoins e on-chain meta-protocolos e permitir que os desenvolvedores criem aplicativos baseadoos em consenso arbitrário com alta escalabilidade, padronização, completude de recursos, facilidade de desenvolvimento e interoperabilidade entre diferentes paradigmas tudo ao mesmo tempo.*

Pode ser definido como SC um código compilável que tem seu binário resultante da compilação executado dentro da BC e que também está distribuído pela rede, assim como todos as outras informações gravadas nela. *O código em SC da Ethereum é escrito em um baixo nível, em uma linguagem de bytecode baseada em pilha, referido como "Código de Máquina Virtual da Ethereum" ou "código EVM (Ethereum Virtual Machine)"*. A EVM é Turing completa, isto significa que o código EVM pode realizar codificar qualquer computação que possa ser convenientemente realizada, incluindo laços de repetição infinitos.

3.1.2 DECENTRALIZED APPS

Como mencionado anteriormente, o foco do Ethereum é fornecer uma plataforma que permita o desenvolvimento de SC. O mecanismo de código EVM *habilita qualquer um a construir o que é essencialmente uma aplicação de linha de comando executada em uma máquina virtual que é executada por consenso através da rede inteira.* Porém a utilização de uma aplicação de linha de comando não é muito amigável para a maior parte dos usuários. Para fazer um bom uso dessa finalidade da BC do Ethereum é

necessário a utilização de mais tecnologias que não dependem diretamente da BC ou da linguagem de script dela.

Nesse sentido podemos definir uma Aplicação Decentralizada completa, ou no termo em inglês Decentralized App (DAPP), como sendo uma aplicação que *pode consistir em dois componentes de lógica de négocio de baixo nível, se implementada inteiramente na Ethereum, usando a combinção da Ethereum e outros sistemas (por exemplo uma camada P2P de mensagem, um dos quais está previsto para ser posto em um cliente Ethereum) ou outros sistemas inteiramente, e componentes gráficos de interface de usuário de alto nível. Um DAPP é usualmente composto de SC's implementados em uma BC, e um software que é habilitado a criar e enviar transações para eles. Este software usualmente provem uma interface, rodando em um Computador Pessoal ou dispositivo móvel. Informações adicionais podem ser armazenadas em um servidor, e lógicas de negócio adicionais podem ser executadas sobre isso.*

Uma arquitetura típica de um DAPP é composta de um software rodando em dispositivos móveis e/ou em servidores, possivelmente na nuvem, trocando informação com usuários e dispositivos externos, que podemos chamar "Sistema da aplicação". Esta Interface de Usuário (IU), ou em inglês User Interface (UI) tipicamente é executada em um navegador Web. Pode haver um componente servidor, para armazenar dados que não podem ser armazenados na BC, e para realizar cálculos de negócio. Na Ethereum, o Sistema de Aplicação tipicamente comunica-se com a BC usando a "web3.js", uma biblioteca JavaScript, quer gerencia a criação e envio de transações.

Na prática pode-se classificar a arquitetura de um DAPP com sendo uma das seguintes: Direta, onde o cliente interage diretamente com o SC implantado na BC; Indireta, que tem um serviço back end funcionando em um servidor centralizado, e um cliente que interage com o SC através deste servidor; Mista, que combina as duas arquiteturas anteriores onde o cliente interage com o SC diretamente e indiretamente através de um back end.

3.1.3 SOLIDITY E GAS

Para realizar o desenvolvimento é necessário uma linguagem Turing completa. Como falado anteriormente, essa linguagem deve permitir, teoricamente, que contratos arbitrários possam ser criados a partir de qualquer tipo de transação da aplicação. Para este trabalho usaremos a Solidity, uma Linguagem de Programação Orientada a Objetos (LPOO), é uma linguagem semelhante ao JavaScript. Os contratos são definidos como classes, eles possuem variáveis internas, funções públicas e privadas. *Solidity adiciona conceitos específicos como eventos e modificadores, e exibe uma limitação nos tipos disponíveis de estruturas de dados para o SC, e no gerenciamento de coleções de dados[…]. O código fonte do SC é compilado em bytecode para ser implantado na BC da Ethereum. Após o desenvolvimento o SC receberá um endereço.*

Como a mudança de estado da BC e a execução de SC necessitam que hajam transações, então a solução achada pelas pessoas que desenvolveram a plataforma de desenvolvimento de SC da Ethereum para evitar que houvesse uma alta latência na execução destas foi a cobrança de uma taxa chamada gas. O custo do SC em um DAPP inclui duas partes: custo de implantação e custo de execução. Inplatação e execução são feitos por transações, que custam gas. O Gas é pago com Ethers, a moeda da Ethereum, e o montante de gas usado é a medida da complexidade da execução de um contrato. Uma conta paga um pouco de gas na execução de um contrato, e então obtém o gas restante quando a execução do SC é confirmada. Se na transação for usado todo o gas enviado pelo requerinte, a conta receberá um erro informando "out of gas", "sem gás" em português, e perderá todo o gas enviado.

3.2 DA ARQUITETURA DA APLICAÇÃO

A solução proposta conta com uma abordagem de desenvolvimento de um DAPP indireto, tendo serviços de back end rodando em um servidor centralizado, enquanto o cliente interage com o SC através destes. Como falado anteriormente, e na seção de Fundamentação Teórica, a blockchain possui um desempenho que deixa a desejar para alguns contextos, porém não para o nosso.

Parte da arquitetura da aplicação será desenvolvida utilizando tecnologias mais difundidas e que não terão o papel principal na descentralização da informação. Essa última necessidade estará a cargo da blockchain, que para o nosso desenvolvimento será utilizada via rede de teste.

Para o front end será feito uso da principal linguagem de desenvolvimento para a web da atualidade, o JavaScript. Pode-se ser feito o uso dos principais frameworks ou bibliotecas da linguagem, como o Angular, React ou Vue, dentre outras menos difundidos. Também será feito o uso de um navegador de internet.

No desenvolvimento do back end pode-se fazer uso de bibliotecas, API's e frameworks que agilizam a integração com a BC da Ethereum e que permitem que o programador não precise implementar do início as principais funcionalidades de uma BC. É possível acessar uma quantidade grande dessas ferramentas através da própria documentação da Ethereum.org, a fonte primária e online para a comunidade da Ethereum, de acordo com o repositório do próprio projeto aberto do site no github.

A imagem a seguir fornece uma visão geral da arquitetura da aplicação. Nela podemos ver a divisão entre das camadas mencionadas; front end, back end e API de conexão com a BC da Ethereum.

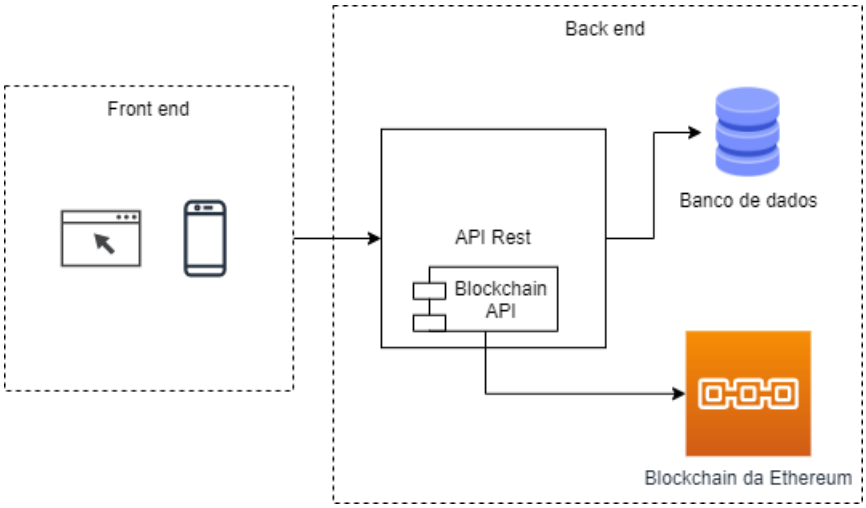

Figura 1: Arquitetura básica da aplicação

Um banco de dados também será utilizado para armazenar as informações de usuário, como dados de autenticação, identificação ou demais informações que não precisam ser armazenadas utilizando a BC. Será feito o uso de um banco de dados não relacional, pois as tabelas do banco de dados podem apresentar características dinâmicas, o que é interessante para aplicação de uma tecnologia que permita esquemas de tabela mais flexíveis, como bancos de dados orientados a documentos.

3.3 MÉTODO

- Apresentar o processo de desenvolvimento a ser utilizado

Com a finalidade de atingir uma gama maior de usuários, o aplicativo deverá contar com uma interface amigável e de fácil interação. Ou seja, para este trabalho será necessário o desenvolvimento de um

front end. Já para tratar e trabalhar com as informações fornecidas pelo denunciante, será necessário o desenvolvimento de uma outra parte da aplicação, um back end.

A blockchain da Ethereum (BC) será utilizada neste trabalho por possuir uma gama muito alta de ferramentas, além de uma linguagem Turing completa para o desenvolvimento de Smart Contracts (SC). Ela deve manipular e armazenar o DAPP por pura descentralização, porém, devido ao gargalo de desempenho da blockchain, será escolhido uma implementação onde somente algumas partes do sistema estarão na estrutura decentralizada.

3.3 ANÁLISE E DESENVOLVIMENTO

- Apresentar os principais diagramas: Caso de Uso, Diagrama de Classes, etc.
- Apresentar a modelagem preliminar do banco de dados
- Apresentar a validação da modelagem no banco de dados
- Apresentar os protótipos iniciais das telas do sistema
- Discutir como será realizada a validação do software

5. cronograma

Devem ser definidas/listadas as principais atividades a serem realizadas e a previsão para serem desempenhadas até a conclusão do trabalho.

6. CONSIDERAÇÕES FINAIS

Devem ser apresentadas as principais limitações e restrições do trabalho.

REFERÊNCIAS

Para as regras gerais de apresentação das referências, ver página 77 das Normas para Elaboração de Trabalhos Acadêmicos da UTFPR, disponível em:
http://www3.utfpr.edu.br/dibib/normas-para-elaboracao-de-trabalhos-academicos/normas_trabalhos_utfpr.pdf

Observações:

As regras de formação são as disponibilizadas no site da biblioteca em:
http://www.utfpr.edu.br/cornelioprocopio/biblioteca-e-producao-academica/normas-para-elaboracao-de-trabalhos-academicos

Para a proposta o limite de páginas será de máximo 20.