ARCADA

Blockkedjeteknik och utvecklingen av en decentraliserad applikation (dApp)

Atte Marjamäki

Lärdomsprov

Informationsteknik

2024

Lärdomsprov

Atte Marjamäki

Blockkedjeteknik och utvecklingen av en decentraliserad applikation (dApp)

Yrkeshögskolan Arcada: Informationsteknik, 2024.

Sammandrag:

Detta lärdomsprov utforskar blockkedjeteknik i två huvuddelar. Först ges en introduktion till teknologin, inklusive dess historik och dess potential inom olika sektorer. Det diskuteras även grundläggande komponenter och egenskaper hos framstående blockkedjor som Bitcoin och Ethereum. Sedan fokuserar texten på utvecklingen av ett proof-of-concept för en enkel decentraliserad social media-applikation som körs på Polygon. Användare kan göra inlägg, gilla andra användares inlägg samt ge dem dricks. Dessutom kan användare ställa in ett användarnamn som är kopplat till deras plånboksadress och se statistik över hur många likes, inlägg och dricks de har fått totalt. Användning av applikationen kräver att man har en plånbok som Metamask och en viss mängd av MATIC. Syftet är att utforska blockkedjans potential och dess förmåga att decentralisera olika sektorer, med särskild betoning på dess tillämpning inom sociala medieplattformar. Projektet inkluderar en djupare förståelse för Solidity-programmeringsspråket och syftar till att illustrera dess roll i decentraliserade system. Slutligen diskuteras för- och nackdelar med att använda blockkedjor för sociala medieplattformar samt bredare frågor och utmaningar inom blockkedjetekniken.

Nyckelord:

Blockkedja, Bitcoin, Ethereum, smarta kontrakt, dApp, Web3

Degree Thesis

Atte Marjamäki

Blockchain technology and the development of a decentralized application (dApp)

Arcada University of Applied Sciences: Information Technology, 2024

Abstract:

This thesis explores blockchain technology in two main parts. First, an introduction to the technology is provided, including its history and its potential across various sectors. It also discusses fundamental components and features of prominent blockchains such as Bitcoin and Ethereum. Then, the text focuses on the development of a proof-of-concept for a simple decentralized social media application running on Polygon. Users can make posts, like other users' posts, and tip them. Additionally, users can set a username linked to their wallet address and see statistics on the total number of likes, posts, and tips they have received. Using the application requires having a wallet like MetaMask and a certain amount of MATIC. The purpose is to explore the potential of blockchain and its ability to decentralize various sectors, with a particular emphasis on its application in social media platforms. The project includes a deeper understanding of the Solidity programming language and aims to illustrate its role in decentralized systems. Finally, the advantages and disadvantages of using blockchains for social media platforms are discussed, as well as broader issues and challenges in blockchain technology.

Keywords:

Blockchain, Bitcoin, Ethereum, Smart Contracts, dApp, Web3

Opinnäyte

Atte Marjamäki

Lohkoketjuteknologia ja hajautetun sovelluksen (dApp) kehittäminen

Yrkeshögskolan Arcada: Informaatiotekniikka, 2024

Tiivistelmä:

Tämä opinnäytetyö tutkii lohkoketjuteknologiaa kahdessa pääosassa. Aluksi annetaan johdanto teknologiaan, mukaan lukien sen historia ja sen potentiaali eri aloilla. Lisäksi käsitellään johtavien lohkoketjujen, kuten Bitcoinin ja Ethereumin, peruskomponentteja ja ominaisuuksia. Tämän jälkeen teksti keskittyy yksinkertaisen hajautetun sosiaalisen median sovelluksen proof-of-conceptin kehittämiseen, joka toimii Polygonilla. Käyttäjät voivat tehdä julkaisuja, tykätä muiden käyttäjien julkaisuista ja antaa heille tippejä. Lisäksi käyttäjät voivat asettaa käyttäjänimen, joka on yhteydessä heidän lompakkonsa osoitteeseen, ja nähdä tilastot siitä, kuinka monta tykkäystä, julkaisua ja tippiä he ovat saaneet yhteensä. Sovelluksen käyttö edellyttää lompakkoa, kuten Metamaskia, ja tiettyä määrää MATICia. Tarkoituksena on tutkia lohkoketjun potentiaalia ja sen kykyä hajauttaa eri aloja, erityisesti sen soveltamista sosiaalisen median alustoilla. Projekti sisältää syvällisemmän ymmärryksen Solidity ohjelmointikielen roolista ja pyrkii havainnollistamaan sen roolia hajautetuissa järjestelmissä. Lopuksi käsitellään lohkoketjujen käytön etuja ja haittoja sosiaalisen median alustoilla sekä laajempia kysymyksiä ja haasteita lohkoketjuteknologiassa.

Avainsanat:

Lohkoketju, Bitcoin, Ethereum, älysopimukset, dApp, Web3

Figurer

Innehåll

1 Inledning

Detta arbete är strukturerat i två huvuddelar, vilka tillsammans syftar till att ge en omfattande översikt och praktisk insikt i användningen av blockkedjeteknik.

I den första delen ges en introduktion till blockkedjeteknik, inklusive en översikt över dess historiska utveckling och det potentiella inflytandet på diverse sektorer. Denna del omfattar en genomgång av de grundläggande komponenterna i en blockkedja: transaktioner, block, miners och konsensusmekanismer. Vidare utforskas de distinkta egenskaperna hos framstående blockkedjor såsom Bitcoin och Ethereum, med särskilt fokus på deras unika mekanismer och implementeringar av smarta kontrakt, vilka har introducerat en ny dimension till blockkedjeteknologin. Dessutom behandlas konceptet med Lager 2-blockkedjor och dess bidrag till enklare implementering inom Web3. Avslutningsvis diskuteras de grundläggande egenskaperna hos blockkedjeteknologi, såsom transparens, säkerhet och decentralisering, samt de potentiella utmaningar och nackdelar som dessa medför.

Den andra delen av arbetet ägnas åt implementeringen av en decentraliserad applikation (dApp), med särskilt fokus på de specifika verktyg och metoder som används i utvecklingsprocessen. Här utforskas det praktiska tillvägagångssättet för att skapa en interaktion mellan en webbapplikation och ett smart kontrakt på blockkedjan. Detaljerade steg för utvecklingen av både applikationen och det smarta kontraktet presenteras, inklusive hur man programmerar ett smart kontrakt med hjälp av Solidity och distribuerar detta till en blockkedja. Vidare demonstreras processen för att etablera kommunikation med det smarta kontraktet, med stöd av verktyg som Thirdweb. Slutligen behandlas utvecklingen av en modern front-end applikation med hjälp av Next.js och TailwindCSS, vilket illustrerar hur samtida webbteknologier kan integreras med blockkedjeteknologi för att skapa användarvänliga och funktionella decentraliserade applikationer.

1.1 Bakgrund

Inspirationen för detta arbete härrör från mitt personliga intresse för blockkedjeteknik. Jag har länge varit fascinerad av denna teknik och önskade att ta mitt intresse ett steg längre genom att fördjupa mig i ämnet. Detta inkluderar att förstå hur teknologin fungerar, att identifiera skillnader mellan olika protokoll, samt att utforska den omgivande hypen.

Valet att utveckla en social medieapplikation baserad på blockkedjeteknik motiverades av den rådande centraliseringen inom medieindustrin. Plattformar som Instagram och Facebook har visat en tendens att censurera innehåll som inte överensstämmer med deras åsikter, vilket inkluderar att förbjuda användare med avvikande åsikter, radera opassande inlägg, och manipulera informationsflöden. Ett noterbart exempel är Twitter, före dess övertagande av Elon Musk och namnändringen till X, där det avslöjades i så kallade ”Twitter Files” att den amerikanska regeringen hade påverkat modereringen av innehåll på plattformen. (Mattox, 2023)

Denna utveckling har bidragit till en minskning av yttrandefriheten globalt. Exemplifierat genom en artikel från Helsingin Sanomat, (Bäckgren, 2024) där det framkom att studenter vid Helsingfors universitet med politiskt högerinriktade åsikter upplevde svårigheter att uttrycka sig på grund av rädsla för att bli dömda. Denna situation har lett till vad som beskrivs som en "cancel-kultur", där individer som uttrycker åsikter som avviker från normen kring frågor som invandring, matvanor och kön riskerar att bli ostraciserade.

I ljuset av dessa överväganden framstår blockkedjeteknik och decentralisering som en lovande möjlighet att utforska potentialen för att skapa en decentraliserad social medieplattform. En sådan plattform skulle möjliggöra för användare att fritt uttrycka sina åsikter utan rädsla för censur eller att bli uteslutna från samtalet. Denna studie ämnar därför att utforska hur blockkedjeteknik kan användas för att främja yttrandefriheten och motverka de centraliserade mediernas begränsningar.

1.2 Syfte och mål

Syftet med detta arbete är att undersöka utvecklingsprocessen för en decentraliserad applikation (dApp) som interagerar med blockkedjan. Detta inkluderar att utforska blockkedjans funktioner i praktiken, att förstå dess potentiellt revolutionära natur, och att bedöma dess förmåga att decentralisera olika sektorer. Studien syftar till att bedöma om blockkedjetekniken verkligen representerar ett revolutionärt framsteg för framtiden eller om den endast utgör en övergående trend, ofta kritiserad som ett "Ponzi-schema" eller som en form av "magisk internetvaluta", såsom den ibland har beskrivits av dess kritiker.

Vidare avser arbetet att förmedla en djupare förståelse för programmeringsspråket Solidity, vilket är centralt för utvecklingen av dApps och interaktion med smarta kontrakt på blockkedjan. Genom praktisk tillämpning siktar studien på att utveckla en applikation som inte bara demonstrerar teknisk kompetens i att använda Solidity utan också illustrerar hur smarta kontrakt kan fungera som en kärnkomponent i decentraliserade system. Genom detta tvärvetenskapliga tillvägagångssätt hoppas arbetet bidra med insikter i blockkedjeteknologins möjligheter och utmaningar, samt dess långsiktiga hållbarhet och relevans i ett bredare sammanhang.

Målet med detta arbete är att lära mer om blockkedjeteknik och hur man programmerar en enkel decentraliserad applikation med hjälp av denna teknologi. I detta fall en decentraliserad socialmediaapplikation där användare kan skapa och interagera med andras inlägg på ett decentraliserat sätt utan att behöva oroa sig över censur. Samtidigt undersöka vilka olika komponenter och teknologier som kan användas för att skapa applikationen.

2 Teoretisk bakgrund

Tanken på ett elektroniskt betalningssystem som utnyttjade kryptografi framträdde redan under 1980-talet, då David Chaum grundade företaget DigiCash. Det introducerade ett system som möjliggjorde säkra och anonyma överföring av pengar på internetet, vilket i sin tur lade grunden för modern blockkedjeteknik. (Rasure, 2023)

2.1 Bitcoin

När diskussionen rör sig kring ursprunget för blockkedjor och Bitcoin är det vanligt att namnet Nick Szabo lyfts fram. Szabo, en kryptograf som tidigare också hade arbetat för DigiCash, spelade en central roll i utvecklingen av tidiga koncept som lade grunden för Bitcoin. År 1998 föreslog Szabo idén om en decentraliserad digital valuta som han kallade "bit gold", vilket många betraktar som en tidig föregångare till Bitcoin. (Peck, 2012)

Bitcoin var ett av de första protokollen som använde denna revolutionerande teknologi. I oktober 2008 publicerades Bitcoins vitbok av den pseudonyma Satoshi Nakamoto. (Nakamoto, 2008) Vitboken beskrev hur Bitcoin kunde möjliggöra peer-to-peer transaktioner i ett decentraliserat nätverk. (Collins, 2023)

Detta nätverk, baserat på kryptografi och decentralisering, erbjöd en revolutionerande möjlighet för individer att delta i finansiella transaktioner utan behov av mellanhänder. På grund av dessa egenskaper uppfattas Bitcoin av många som en digital värdeförvaring, ofta liknad vid ”digitalt guld”. Liksom guld har Bitcoin en begränsad mängd tillgänglig för hela världen, vilket bidrar till dess attraktivitet som värdeförråd. (Collins, 2023)

2.2 Ethereum

Vissa individer såg potentialen i Bitcoins teknologi och strävade efter att vidareutveckla den. Ett antal år efter Bitcoins uppkomst, publicerade Vitalik Buterin sin vitbok för ett nytt protokoll som kom att kallas Ethereum. Detta protokoll utnyttjade samma blockkedjeinfrastruktur som Bitcoin, men med en betydande utökning av dess funktionalitet. År 2015, tillsammans med sina medgrundare, introducerade Buterin projektet till allmänheten. Den avgörande innovationen som Ethereum förde med sig var förmågan att inte enbart utföra decentraliserade transaktioner, utan även att möjliggöra decentraliserade avtal och andra former av integration utan behov av en centraliserad mellanhand eller auktoritet. (Collins, 2023)

För att ge en enkel jämförelse, om Bitcoin kan liknas vid en huvudbok där all transaktionsdata registreras i en linjär tabell, kan Ethereum betraktas som en dynamisk Excel-

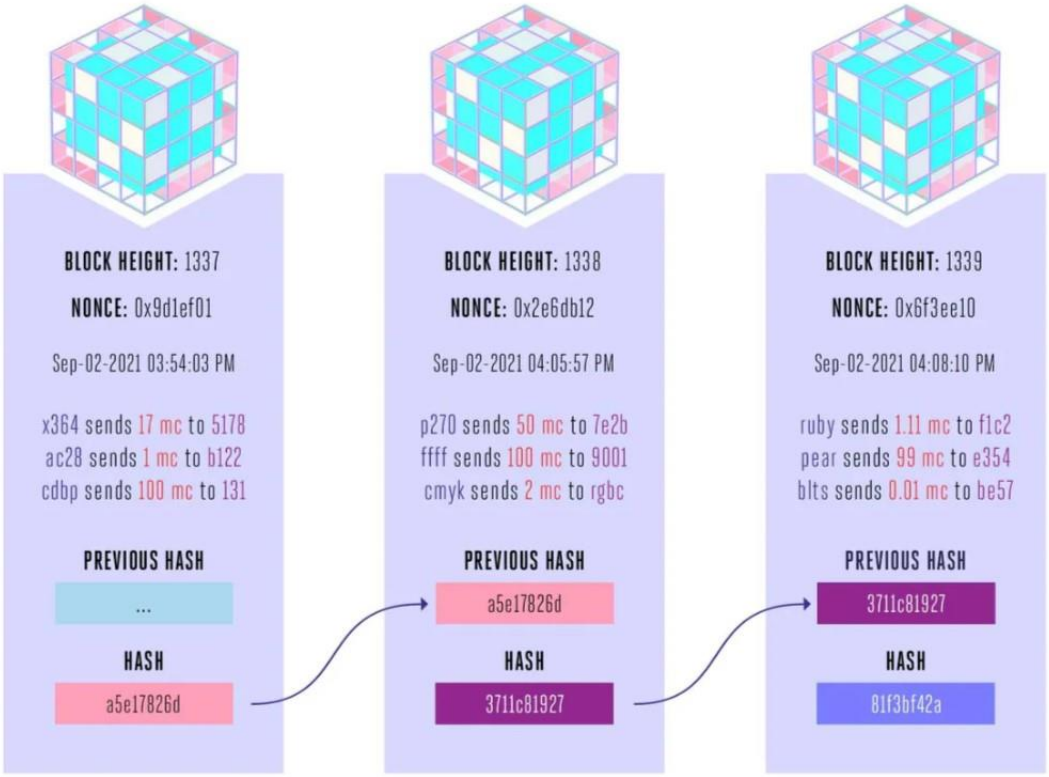
tabell där olika typer av funktioner kan implementeras. (Jani, 2023) I grunden var deras vision att bygga vidare på det som gjorde Bitcoin så banbrytande och att inkludera möjligheten till decentraliserade avtal, även kända som smarta kontrakt.

2.3 Blockkedjeteknik

Teknologin bakom både Bitcoin och Ethereum kallas blockkedja. Det utgör en form av en databas som representerar en distribuerad huvudbok inom ett peer-to-peer-nätverk. Varje block inom kedjan innehåller information om blocket självt, en huvudbok över transaktioner samt det föregående blockets hashvärde. (Hayes, 2023) Ursprunget till namnet "blockkedja" refererar till hur varje huvudbok (block) är länkad till nästa via ett hashvärde (kedja).

När en transaktion initieras, exempelvis överföring av kryptovaluta, distribueras informationen till nätverket. Deltagande noder verifierar transaktionens legitimitet genom att kontrollera dess digitala signaturer och andra relevanta data. Efter verifiering samlas transaktionerna i ett nytt block som sedan läggs till kedjan av tidigare block. (Binance Academy, 2023)

Denna process av validering och blockintegration genomförs med hjälp av olika konsensusmekanismer. Varje nod i nätverket behåller en kopia av hela blockkedjan, vilket i praktiken gör datamanipulation omöjligt. (Ravikiran, 2023) I enklare termer baseras säkerheten på kryptografi och datorkod och förhindrar försök till manipulation av data i kedjan.



Figur 1. Hur en blockkedja ser ut och vad den innehåller (The Verge, 2021)

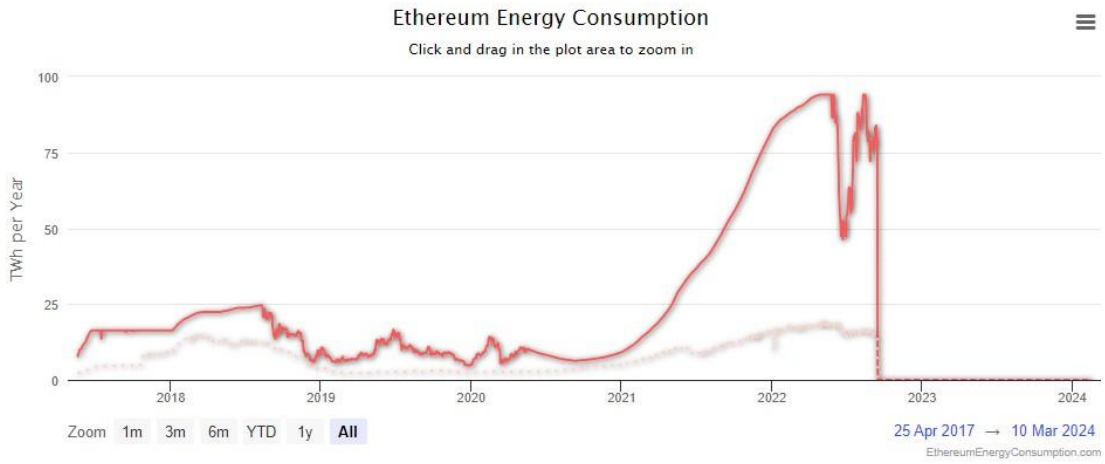
2.3.1 Konsensusmekanismer

Det finns olika konsensusmekanismer för att validera nya block och transaktioner men de vanligaste är bevis på arbete (Proof of Work, PoW) och bevis på insats (Proof of Stake, PoS),

I bevis på arbete (PoW) som används till exempel av Bitcoin. Där tävlar deltagare om att lösa komplicerade matematiska pussel för att lägga till ett nytt block i blockkedjan, dessa deltagare kallas för miners. Arbetet för att lösa pusslen görs av en dators hårdvara. Denna metod ger stark säkerhet och decentralisering, men medför långsamma transaktionshastigheter och väldigt hög energiförbrukning. (Becher, 2023) Idag konsumerar Bitcoin ungefär 160TWh elektricitet i året, vilket är ungefär lika mycket som hela Egypten. (Digiconomist, 2024)

I bevis på insats (PoS) som används till exempel av Ethereum väljs validerare för att skapa nya block av de som har av kryptovalutan som insats (staked) i protokollet. Rätten

att validera ett block kan variera på hur mycket kryptovaluta man har satsat, hur länge man har vari en validator eller helt slumpmässigt. De olika protokollen brukar mixa mellan dessa men oftast är det så att ju mer man har som insats, desto större chans har man att bli vald. (Binance Academy, 2018) Detta leder till mera centralisering eftersom det som har mera valuta har mera makt. Men bevis på insats ger betydande energibesparingar och bättre skalbarhet jämfört med bevis på arbete. (Becher, 2023) När Ethereum övergick i september av 2022 från PoW till PoS minskades protokollets energikonsumtion med mera än 99%. (Digiconomist, 2024)



Figur 2. Diagram av Ethereums energikonsumtion. (Digiconomist, 2024)

2.3.2 Transaktioner och transparens

Inom blockkedjor används ofta asymmetrisk kryptografi, även känd som kryptering med offentlig nyckel, som en kritisk metod för att säkra och verifiera transaktioner mellan dess användare. Varje deltagare i nätverket innehar ett unikt nyckelpar, bestående av en privat och en offentlig nyckel. När en användare initierar en transaktion, använder hen sin privata nyckel för att skapa en digital signatur. Genom att tillämpa avsändarens offentliga nyckel på denna signatur kan andra användare i nätverket verifiera transaktionens äkthet. Denna process säkerställer att endast ägaren till den privata nyckeln kan godkänna en transaktion, medan alla kan verifiera dess autenticitet med den offentliga nyckeln. (Binance Academy, 2023)

En av de mest framträdande egenskaperna hos blockkedjan är dess transparens. All information på blockkedjan, inklusive transaktioner, saldon och blockdata, är synliga för alla som bara har intresse att besöka en av de blockkedjewebbplatserna som har informationen. Denna transparens ger flera fördelar såsom enkel spårning och verifiering av transaktioner, vilket skapar förtroende mellan användarna i nätverket. Dessutom har varje nod i nätverket en kopia på kedjan, vilket underlättar bekämpning av brottslighet. Detta innebär att endast personen tilldelad en adress kan avslöja sin identitet. Som ett resultat kan användare av blockkedjan förbli anonyma samtidigt som transparensen bibehålls. (Hayes, 2023)

2.4 Smarta kontrakt

Begreppet ”smarta kontrakt” kan också spåras tillbaka till Nick Szabo och år 1994. Han föreslog självutförande kontrakt lagrade i en distribuerad huvudbok, vilket möjliggjorde avtal utan behov för en tredje part. (Collins, 2023) Men det var inte förrän lanseringen av Ethereum år 2015, vilket gjorde dem allmänt kända.

Så vad är egentligen smarta kontrakt? I grund och botten är de självutförande digitala avtal eller uppsättningar av instruktioner som implementeras på decentraliserade blockkedjor. När dessa instruktioner har implementerats kan de inte ändras. De utförs automatiskt under förbestämda förutsättningar, och utfallen är synliga för alla involverade parter. Den grundläggande principen bakom smarta kontrakt är att de utförs av en decentraliserad kollektiv, såsom ett blockkedjenätverk, där flera parter verifierar och validerar utförandet av koden. Denna decentraliserade natur säkerställer att ingen enskild person eller enhet har makten att ändra avtalen eller förändra villkoren för överenskommelsen. (Collins, 2023)

De potentiella tillämpningarna för smarta kontrakt är omfattande och sträcker sig över olika branscher. Här är några framstående exempel:

Tokens: Tokens är digitala tillgångar som representerar ägande av en tillgång eller nytta inom ett decentraliserat nätverk, liknande kryptovalutor eller lojalitetspoäng. (Investopedia, 2024)

Non-Fungible Tokens (NFT): NFT:er är unika digitala tillgångar som representerar ägande av unika föremål eller samlarobjekt, utnyttjar blockkedjeteknik för att verifiera äkthet och ägande som ofta används för digital konst eller in-gameföremål. (Sharma, 2024)

Decentralized Finance (DeFi): DeFi syftar till ett finansiellt system som syftar till att tillhandahålla traditionella finansiella tjänster utan mellanhänder, vilket möjliggör utlåning, lån, handel och mer genom decentraliserade applikationer och smarta kontrakt. (Sharma, 2024)

2.5 Web3

Internetet har utvecklats från statiska webbsidor (Web1) till interaktiva plattformar (Web2), där användarna skapar och delar innehåll. Web3 tar nästa steg genom att använda blockkedjeteknik för att decentralisera kontrollen. Denna förändring ger användarna ökad säkerhet och integritet när de integrerar med online applikationer och ger dem större kontroll över sin data på internetet.

Web3 representerar en transformerande metod för internetinfrastruktur genom att utnyttja blockkedjeteknik för decentralisering och förbättrad säkerhet. Den främjar ett internetekosystem som ger användarna möjlighet att hantera sin data, vilket eliminerar behovet av en central myndighet. Detta paradigmskifte är betydelsefullt av olika skäl. För det första ger det individer större äganderätt över digitala tillgångar, exemplifierat genom ägande av NFT:er, vilket säkerställer kontroll över spelobjekt oavsett spelaktivitet eller kontoborttagning. För det andra främjar Web3 motstånd mot censur genom att låta användare övervaka sina data inom ett decentraliserat ramverk, vilket eliminerar beroendet av centraliserade enheter. Dessutom introducerar det decentraliserade autonoma organisationer (DAOs), vilket underlättar gemensamt ägarskap och beslutsfattande om plattformsstyre och potentiellt främjar jämlikhet och transparens. I grund och botten erbjuder Web3 en mängd fördelar som kan revolutionera internetanvändning och styrningsdynamik. (CoinMarketCap, n.d.)

2.6 EVM

Ethereum Virtual Machine (EVM) är ett speciellt program som används för att köra smarta kontrakt på Ethereum nätverket. Tusentals datorer som kör Ethereum-blockkedjan driver denna ”virtuella” maskin. EVM håller koll på det aktuella tillståndet för Ethereum-nätverket, bearbetar transaktioner och lagrar data. Den är central för alla funktioner i Ethereums dAppar och plånböcker.

Ethereums dominerande ställning inom Web3 driver på en trend där nya blockkedjor byggs kompatibla med dess EVM. Denna kompatibilitet gör det enkelt för utvecklare att flytta sina Ethereum-baserade applikationer till dessa nya kedjor, vilket potentiellt når en bredare publik och drar nytta av funktioner som lägre avgifter och snabbare transaktioner. Populära dAppar som Uniswap finns redan på flera EVM-kompatibla kedjor. Vissa av dessa kedjor är helt separata blockkedjor som Avalanche och BNB Smart Chain, medan andra är ”Layer 2” lösningar byggda ovanpå Ethereum som erbjuder snabbare och billigare transaktioner. En av dessa är till exempel Polygon. (Worldcoin, 2023)

2.6.1 Layer 2 och Polygon

På grund av Ethereums popularitet och designval lider nätverket av långsamma transaktionshastigheter och höga gasavgifter. Detta leder till att användarbasen inte kan öka och expandera och stödet för mer komplicerade applikationer begränsas. För att hantera dessa begränsningar har olika skalningslösningar dykt upp med så kallade ”Layer 2”-lösningar. Layer 2 fungerar i huvudsak som separata blockkedjor byggda ovanpå Ethereums blockkedja (Layer 1). De bearbetar transaktioner utanför kedjan, vilket minskar både transaktionshastigheten och kostnaden.

En av dessa Layer 2 lösningar är Polygon. Som en parallell blockkedja gör Polygon det möjligt att ”bridga” det vill säga flytta över sitt krypto från Ethereums huvudblockkedja, vilket gör det möjligt för dem att integrera med olika populära kryptoapplikationer som tidigare var exklusiva för Ethereum. MATIC, Polygons kryptovaluta används för nätverksavgifter, insats och styrning, vilket gör det möjligt för MATIC innehavare att delta i omröstningar och förändringar på Polygon. (Coinbase, n.d.)

Genom att behandla transaktioner på en separat, Ethereum-kompatibel blockkedja och sedan återintegrera dem i Ethereum-huvudnätverket, minskar Polygon effektivt nätverksbelastningen, ökar transaktionshastigheterna och sänker kostnaderna till mindre än en cent. Detta etablerar Polygon som en skalbar lösning för både nya och befintliga applikationer på Ethereum, där man adresserar skalbarhetsutmaningarna direkt. (Kaur, 2023)

3 Metoder

3.1 Verktyg och teknologier

Detta kapitel omfattar de olika verktygen och teknologierna som används för att utveckla både smarta kontraktet och applikationen.

3.1.1 VS Code och Remix IDE

Visual Studio Code (VS Code) är en utvecklingsmiljö med öppen källkod utvecklad av Microsoft. Den erbjuder en kraftfull miljö för programmering, kodredigering och felsökning. VS Code stöder ett brett utbud av programmeringsspråk och olika tillägg, vilket gör det till ett av de populäraste val bland programmerare.

Remix IDE är en webbläsarbaserad utvecklingsmiljö med öppen källkod som är designad för utveckling av smarta kontrakt på blockkedjeplattformar. Den erbjuder ett användarvänligt interface och verktyg för att skriva, testa och distribuera smarta kontrakt. Den har funktioner som kodredigering, felsökning och en inbyggd Solidity kompilator, vilket gör det enklare att skapa och integrera med smarta kontrakt på olika EVM baserade blockkedjor.

3.1.2 Solidity

Solidity är ett programmeringsspråk gjort för att utveckla smarta kontrakt på Ethereum- och andra EVM-baserade blockkedjor. Det utgör grunden för att skapa decentraliserade applikationer och automatisera utförande av kontrakt på EVM. Den fungerar som ett

verktyg för att omvandla människoläsbarkod och kompilera det till bytekod på EVM. (Simplilearn, n.d.)

Solidity är ett statiskt skrivet, klammerbaserat programmeringsspråk som är influerat av JavaScript, Python och C++. Solidity stöder funktioner som arv, bibliotek och komplexa användardefinierade typer. I struktur och koncept liknar smarta kontrakt skrivna i Solidity Java-klasser. Dessa kontrakt omfattar flera komponenter som är väsentliga för deras funktion. (Solidity, n.d.).

3.1.3 Thirdweb

Thirdweb är en utvecklingsplattform som gör det enklare att bygga Web3 applikationer. Thirdweb tillhandahåller en kraftfull API (Application Programming Interface) och SDK (Software Develompment Kit) för olika programmeringsspråk som JavaScript, React och Python. Dessa verktyg möjliggör det att lätt kunna interagera med smarta kontrakt på blockkedjan. (Shevchuk, 2023)

Dessutom erbjuder Thridweb en samling förbyggda, anpassningsbara smarta kontrakt som man kan dra nytta av istället för att bygga ett smart kontrakt från början. Dessa förbyggda kontrakt granskas på förhand av säkerhetsexperter, vilket ger ett lager av förtroende och minskar risken för sårbarheter som ofta är förknippade med smarta kontrakt. (Lemmens, 2023)

3.1.4 MetaMask

MetaMask är en gratis kryptovaluta plånboksprogramvara. Den möjliggör för användare att integrera med Ethereum-blockkedjan och andra nätverk för transaktioner, byte av tillgångar och anslutning till decentraliserade applikationer. Den är tillgänglig som ett webbläsartillägg och mobil app. Den har över 30 miljoner användare, vilket gör den till en av det mest populära krypto-plånböckerna som används idag. (Cointelegraph, n.d.)

3.1.5 React och Next.js

React är ett JavaScript baserat bibliotek för att skapa användargränssnitt, utvecklat av Facebook år 2011. Det möjliggör att skapa återanvändbara UI-komponenter som effektivt uppdateras och renderas baserat på förändringar i applikationens tillstånd, vilket ger ett effektivt sätt att bygga interaktiva webbapplikationer. (Deshpande, 2023)

Next.js, utvecklat av Vercel är ett ramverk med öppen källkod byggt ovanpå React. Den erbjuder flera funktioner som gör det enklare och snabbare att bygga moderna webapplikationer. Några av de viktigaste funktionerna är server-side-rendering (SSR), statisk sidgenerering (SSG) och client-side routing. Detta gör Next.js enkelt att skapa högpresterande, SEO-vänliga webbapplikationer med minimal konfiguration. (Abu Bakr, 2023)

3.1.6 TailwindCSS

TailwindCSS är ett CSS-ramverk med fokus på verktyg (utility-first) som gör det möjligt att snabbt bygga och designa moderna webbsidor direkt i HTML-koden utan behov för att skriva vanlig CSS. Den erbjuder en samling av lågmälda verktygsklasser för att designa element, vilket ger mer kontroll och flexibilitet jämfört andra CSS-ramverk. (Fitzgerald, 2022)

3.2 Utvecklingen av applikationen

Detta kapitel kommer att gå igenom steg för steg hur man bygger en simpel decentraliserad applikation med hjälp av alla teknologier som nämnts tidigare. Det kommer att finnas kodsnuttar och exempel bland texten. Den fullständiga koden för projektet finns på Github. (https://github.com/attemarjamaki/pigeonpost)

3.2.1 Utvecklingen av smarta kontrakt

För att inleda utvecklingen av smarta kontraktet navigerar vi till Remix IDE-webbplatsen och initierar processen där. Användningen av Remix motiveras av kontraktets relativa

enkelhet. Remix tillhandahåller en inbyggd EVM-miljö som möjliggör smidig testning och distribution av kontraktet, vilket effektiviserar processen.

När man konstruerar ett smart kontrakt från grunden utförs två initiala steg:

1. Val av SPDX-licensidentifierare: Licensidentifieraren specificerar kontraktets licensvillkor.
2. Val av kompilatorversion för Solidity: Solidity-kompilatorn används för att översätta koden till bytecode som kan utföras på EVM.

Eftersom detta inte utgör en introduktion till Solidity kommer vi inte att fördjupa oss i språkets funktionalitet och samtliga typer. I stället fokuserar vi på relevanta exempel och de centrala komponenter som utgör grunden för det aktuella kontraktet.

Mappings: Dessa fungerar som datastrukturer som länkar en informationspunkt till en annan. I vårt kontrakt använder vi mappningar för att lagra data som inlägg per användare *tweets*, antalet likes för ett inlägg *totalLikesReceived* och användarnamn *usernames*.

Events: Dessa är permanenta register över specifika åtgärder som inträffar inom kontraktet. De ger en transparent logg över aktiviteter som att posta inlägg *Tweetposted*, gilla inlägg *TweetLiked* och skicka tips *TipSent*.

Functions: Dessa är exekverbara delar av kontraktet som tillåter användare att integrera med den. Kontraktet erbjuder funktioner för att posta inlägg *postTweet*, gilla inlägg *likeTweet*, ge tips till skapare av inlägg *tipTweet* och skapa ett användarnamn *setUsername*. Sen kan de också hämta olika användardata via getter-funktioner som *getLikesCount*, som hämtar likes för ett specifikt inlägg och *getTweet* som hämtar en användares inlägg.

3.2.2 Testande av smarta kontrakt

Testning av smarta kontrakt är en avgörande del av deras utvecklingsprocess och kan anses vara lika viktig som själva konstruktionen av kontraktet. Eftersom smarta kontrakt utförs på blockkedjan med fullständig kodtransparens, blir de attraktiva mål för exploatering om säkerhetsbrister existerar. Till skillnad från traditionella programvara kan inte

smarta kontrakt modifieras efter implementering på blockkedjan. Att upptäcka och åtgärda buggar kräver därför distribution av ett helt nytt kontrakt. Därför är rigorös testning av avgörande vikt för att säkerställa funktionalitet och säkerhet.

Populära testramverk som Hardhat och Foundry tillhandahåller verktyg för att skapa skript som simulerar interaktioner med kontraktet och identifierar potentiella problem. I detta fall, på grund av kontraktets relativa enkelhet och begränsande ekonomiska incitament används Remix IDE för testning. Remix erbjuder en intuitiv miljö med en inbyggd lekblockkedja, vilket gör det enkelt att testa funktionalitet och identifiera kodfel.

Testprocessen i Remix:

1. **Kompilering:** Koden kompileras och distribueras till en av Remix inbyggda lekblockkedjor.
2. **Interaktion:** Användarinteraktionen med kontraktets funktioner simuleras via Remix gränssnitt.
3. **Funktionalitet och fel:** Noggrann granskning av kontraktets funktioner utförs för att identifiera eventuella felaktigheter eller oväntade resultat.

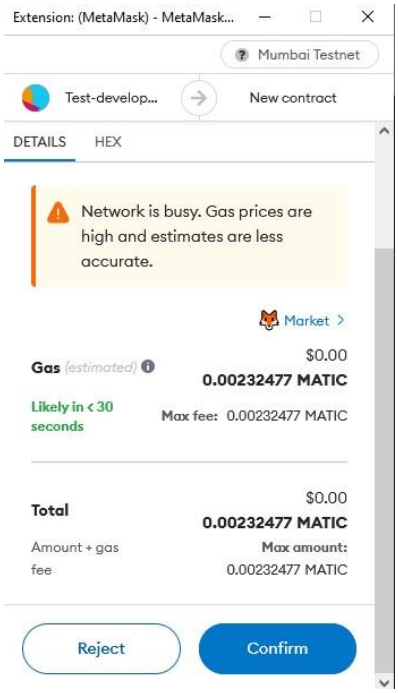
3.2.3 Distributionen av kontrakt till ett testnätverk

Efter framgångsrik testning i Remix distribueras kontraktet till en testblockkedja, i detta fall Mumbai, Polygons testnätverk. Populära blockkedjor med smarta kontraktfunktionalitet har community-drivna testnätverk som är gratis att använda. För att kunna distribuera kontraktet till testblockkedjan måste vi först installera MetaMask.

När MetaMask är installerat och konfigurerat för att ansluta till Mumbais nätverk måste vi skaffa testvaluta. Detta kan vi få från en plattform som Alchemy där man kan få testvaluta gratis för att testa sitt kontrakt.

När vi har en plånbok med testvaluta kan vi gå över till Remix och välja ”Injected Provider” som vår miljö. Det innebär att Remix använder MetaMask för att hantera transaktioner och integrera med blockkedjan. Efter det klickar vi på ”deploy” knappen i Remix. Därefter visas MetaMask-fönstret med information om transaktionen och sedan behöver

vi bara bekräfta transaktionen och betala med vår testvaluta. När detta är gjort finns vårt kontrakt på en testblockkedja för alla att se. Därefter kan vi gå till Mumbai PolygonScan för att se transaktionen.



Figur 3. Skärmdump av MetaMask på transaktionen när kontraktet distribueras.

3.2.4 Hämtande av kontraktet till Thirdweb

För att dra nytta av alla funktioner som Thirdweb erbjuder behöver vi först en API-nyckel och dessutom importera vårt kontrakt till deras plattform. För att få tag på API-nyckeln behöver vi bara skapa en användare på Thirdweb och sedan generera en API-nyckel för oss. Nästa steg är att importera kontraktet till plattformen. Steget för detta är något mer komplicerande. Först måste vi ange adressen för vårt kontrakt och på vilket nätverk kontraktet är distribuerat till. Sedan måste vi verifiera kontraktkoden så att Thirdweb kan extrahera relevant information korrekt från kontraktet.

3.2.5 Setup för front-end

Vi inleder processen genom att starta VS Code och navigera till den mapp där vi önskar skapa vårt nya projekt. Därefter öppnar vi terminalfönstret och utför kommandot ”npx thidweb create app”. Vanligtvis, om vi skulle genomföra en standardinstallation av

Next.js, skulle vi använda kommandot ”npx create-next-app”. Men Thirdweb tillhandahåller ett eget kommando som gör det möjligt för oss att skapa en Next.js applikation med JavaScript och alla nödvändiga paket redan installerade. Detta ger oss en solid grund att strata från när vi utvecklar vår front-end applikation.

Nästa steg är att installera TailwindCSS, vilket är enkelt om vi följer dokumentationen på deras webbplats. Processen involverar i huvudsak två kommandon: ”npm install -D tailwindcss postcss autoprefixer” följt av ”npx tailwindcss init -p”. Dessa kommandon installerar TailwindCSS och dess beroende samt skapar en grundläggande konfigurationsfil, ”tailwind.config.js”. Efter detta steg anpassar vi ”tailwind.config.js” enligt våra behov.

Slutligen integrerar vi TailwindCSS i vårt projekt genom att inkludera TailwindCSS: direktiv i vår ”global.css”-fil. Med det avklarat är installationen färdig och vi är redo att börja utforma vår applikation med ett stilrent och responsivt gränssnitt.

3.2.6 Plånboksanslutning

Nästa steg är att ansluta till webbplatsen med sin digitala plånbok. Vi använder MetaMask, men vi gör det möjligt att också kunna ansluta med Coinbase och Trust Wallet. Processen för att ansluta sin plånbok till webbplatsen är relativt enkel. Först öppnar vi vår _app.js-fil och konfigurerar *activeChain* variabeln till ”mumbai”. Därefter, inom funktionen *MyApp*, inuti *ThirdwebProvider* komponenten som omsluter hela applikationen, måste vi lägga till några egenskaper. Vår CLIENT_ID (API-nyckeln vi erhållit från Thirdweb) och *supportedWallets* så att vi accepterar MetaMask, Coinbase och TrustWallet.

```
const activeChain = "mumbai";

function MyApp({ Component, pageProps }) {
  return (
    <ThirdwebProvider
      activeChain={activeChain}
      clientId={process.env.NEXT_PUBLIC_TEMPLATE_CLIENT_ID}
      supportedWallets={[metamaskWallet(), coinbaseWallet(), trustWallet()]}
```

Figur 4. Kodsnutt av _app.js från Github. (Marjamäki, 2024)

När _app.js är korrekt konfigurerad, fortsätter vi till vår fil där vi vill ha vår anslutning. I vår app, har vi en *Splash.jsx*-fil som kommer att fungera som en landning page som användaren landar till ifall deras plånbok inte är ansluten. På denna sida infogar vi *ConnectWallet*-komponenten med olika egenskaper som påverkar utseende och därmed skapar vi en vacker och interaktiv anslutningsknapp för plånboken.

3.2.7 Datahämtning från blockkedjan till applikationen

Det första steget av att hämta data innebär att fånga upp och visa alla användarinlägg på en så kallad feed. Detta uppnås genom att lyssna på *TweetPosted* -händelsen (Events) som vårt smarta kontrakt sänder ut varje gång ett nytt inlägg skapas. Genom att använda *useContractEvents* -hooken från Thirdweb, prenumererar vi på dessa händelser och hämtar en array av inlägg. Denna metod säkerställer att vi fångar varje inlägg som görs på plattformen, vilket övervinner begränsningen av att endast hämta det senaste inlägget för en användare, vilket skulle vara fallet om vi enbart förlitade oss på direkta anrop till smarta kontaktsfunktioner för inläggshämtning.

```
const { data: userTweets, isLoading: isUserTweetsLoading } =
  useContractEvents(contract, "TweetPosted", { subscribe: true });
```

Figur 5. Kodsnutt av PostFeed.jsx från Github. (Marjamäki, 2024)

Denna kodsnutt prenumererar på *TweetPosted* händelsen, hämtar alla inlägg och lagrar dem i variabeln *userTweets*.

Det andra steget kompletterar den initiala inläggsdatan med ytterligare detaljer nödvändiga för en omfattande visning, såsom användarnamn, likes, tidsstämplar och plånboksadresser. Denna detaljhämtning görs genom att använda specifika get-funktioner definierade i det smarta kontraktet. Genom *useContractRead* -hooken gör vi anrop till funktioner som *getTips*, *getLikesCount* och *getUsername*, där relevanta parametrar såsom *walletAddress* och *tweetId* används för att hämta den information vi vill ha.

```
const { data: likes, isLoading: likesLoading } = useContractRead(
  contract,
  "getLikesCount",
  [tweetId]
);

const { data: _username, isLoading: usernameLoading } = useContractRead(
  contract,
  "getUsername",
  [walletAddress]
);
```

Figur 6. Kodsnutt av PostCardTest.jsx från Github. (Marjamäki, 2024)

Denna kodsnutt anropar de respektive getterfunktionerna, hämtar likes för ett inlägg och användarnamnet associerat med en plånboksadress.

Sedan bygger vi vidare på metoden som används i det andra steget. Vi utökar vår strategi för datahämtning ytterligare för att inkludera kontospecifik statistik, såsom det totala antalet inlägg, likes och tips. Denna strategi använder samma *useContractRead* -hook från Thirdweb. Vi kommer inte att visa exempel på detta eftersom processen är liknande, men med andra parametrar som den tar emot.

3.2.8 Postande från applikationen till blockkedjan

För att möjliggöra att posta från vår applikation skapar vi en *Post* komponent, som fungerar som gränssnittet för att posta innehåll till blockkedjan. Denna komponent är utformad för att fånga användarinput via en textarea, med återkoppling i realtid av antalet tecken så att inlägget inte överskrider 140 tecken eftersom det även är gränsen vi kodade i vårt kontrakt. Sedan utnyttjar vi *Web3Button* komponenten från Thirdweb som låter oss interagera med det smarta kontraktet. Genom att anropa *postTweet* funktionen på det smarta kontraktet vid knapptryckningen möjliggör vi för användaren att publicera sina inlägg direkt på blockkedjan.

```
<Web3Button
  contractAddress={CONTRACT_ADDRESS}
  action={(contract) => contract.call("postTweet", [newPost])}
  isDisabled={newPost.length === 0 || newPost.length > 140}
  onSuccess={() => {
    setNewPost("");
  }}
```

Figur 7. Kodsnutt av Post.jsx från Github. (Marjamäki, 2024)

För att gilla ett inlägg är processen simplare. Vi använder oss igen av *Web3Button* för att anropa *likeTweet* funktionen på det smarta kontraktet och skickar in nödvändiga parametrar såsom användarens plånboksadress och inläggets ID. Denna åtgärd utlöser en transaktion på blockkedjan och registrerar att användaren gillat inlägget.

```
<Web3Button
  contractAddress={CONTRACT_ADDRESS}
  action={(contract) =>
    contract.call("likeTweet", [walletAddress, tweetId])
  }
```

Figur 8. Kodsnutt av PostCardTest.jsx från Github. (Marjamäki, 2024)

Att däremot ge tips till ett inlägg introducerar en liten komplexitet på grund av Ethereums oförmåga att hantera decimalvärden. För att kringgå denna begränsning formerar vi tipsbeloppet på lämpligt sätt genom att konvertera det från ett decimalvärde till dess ekvivalent i Wei.

```
const weiEquivalent = maticAmount * 10 ** 18;
```

Figur 9. Kodsnutt av PostCardTest.jsx från Github. (Marjamäki, 2024)

När användaren har valt det önskade tips mängden utlöser de tipsåtgärden genom den angivna knappen. Bakom kulisserna anropar *Web3Button 'tipTweet'* funktionen på det smarta kontraktet. Detta initierar en transaktion på blockkedjan och överför det specificerade mängden av MATIC som tips till inläggets skapare.

```
<Web3Button
  contractAddress={CONTRACT_ADDRESS}
  action={(contract) =>
    contract.call("tipTweet", [walletAddress, tweetId], {
      value: weiEquivalent.toString(),
    })
  }
```

Figur 10. Kodsnutt av PostCardTest.jsx från Github. (Marjamäki, 2024)

3.2.9 Designen av applikationen

Som inspiration för designen av vår applikation har vi tagit intryck från både Facebook och X (tidigare känd som Twitter). Eftersom vår app också är en social plattform, strävade vi efter att skapa en bekant användarupplevelse så att appen blir enkel att navigera. Designen är avskalad, med en navigeringsrad längst upp som innehåller logo till vänster, anslutningen till plånboken till höger, samt flikarna *Home* och *Profile* i mitten Dessa flikar leder användaren antingen till startsidan eller till deras egen profil.

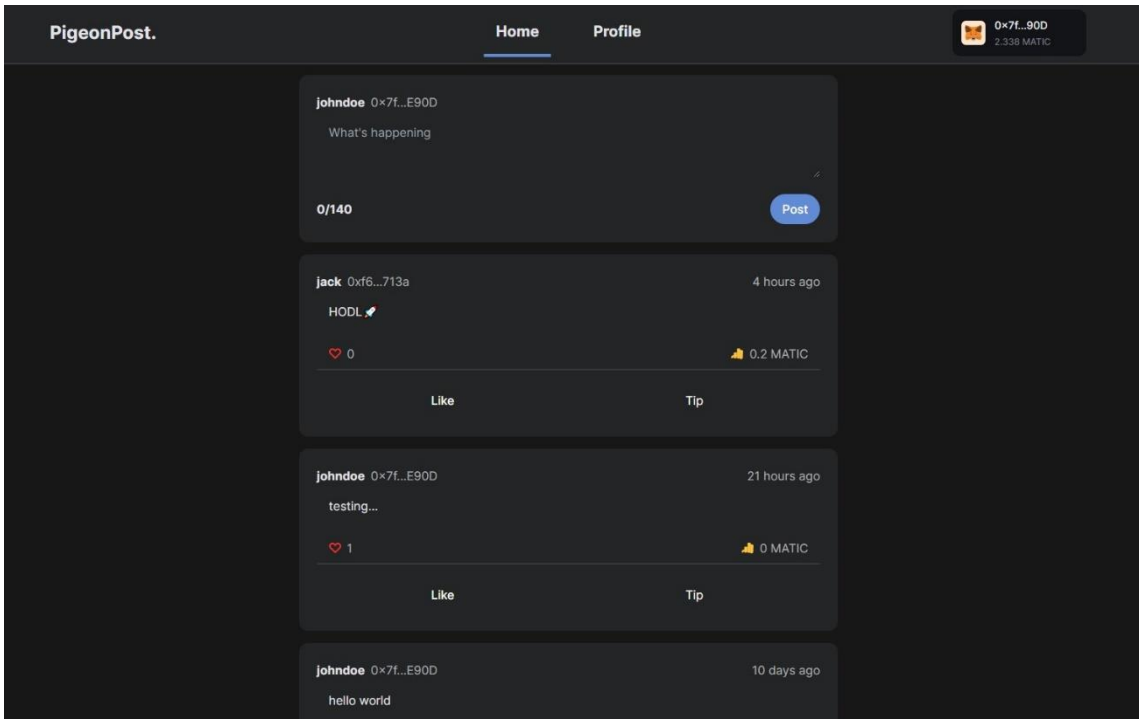
På startsidan möts användaren först av en sektion där de kan skapa inlägg, följt av ett flöde med inlägg från andra användare. Denna del är starkt inspirerad av Twitter. På profilsidan visas användarens information och här finns även möjligheten att ändra sitt användarnamn. Nedanför detta finns statistiksektion för profilen samt användarens egna inlägg.

Färgpaletten för applikationen är liknande den som används Facebooks mörkläge, alternativt en färgpalett som hittades på nätet. Implementeringen i appen är enkel tack vare att vi använder TailwindCSS; vi kan enkelt inkludera dessa färger i vår tailwind.config.css fil och sedan använda dem i designen. Vi har även en färgpalett från Polygon, eftersom det är den blockkedjan vi använder.

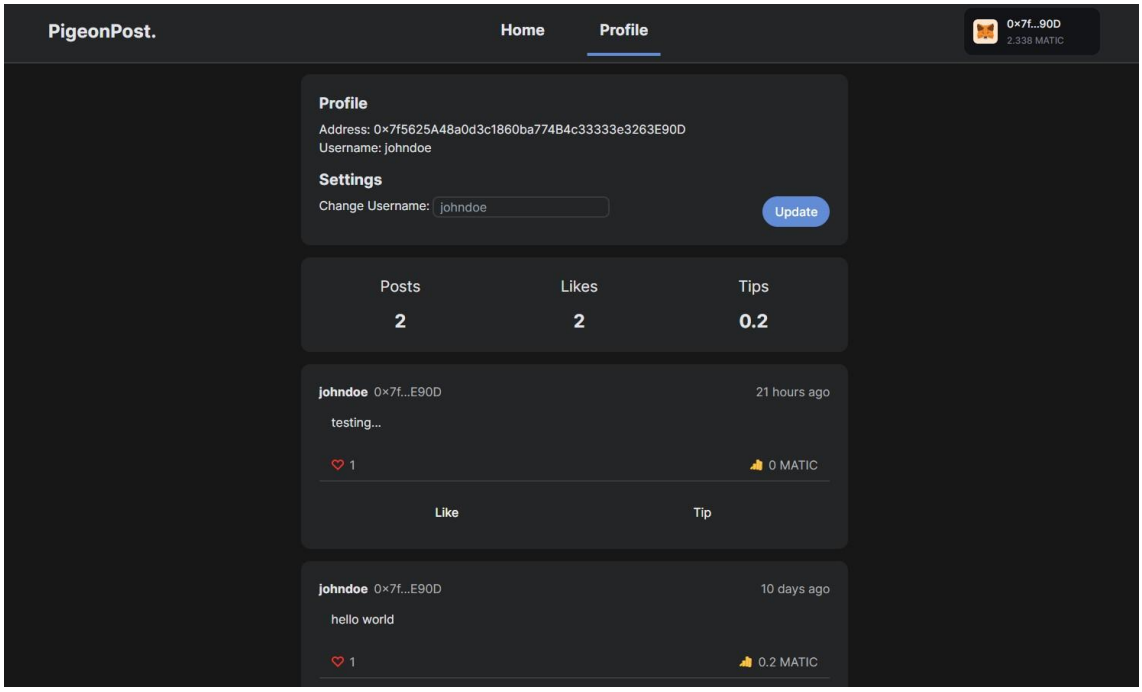
```
theme: {
  colors: {
    transparent: "transparent",
    current: "currentColor",
    white: "#ffffff",
    blue: "#628CD6",
    "dark-blue": "#494592",
    purple: "#9C0C63",
    "purple-dark": "#66095A",
    violet: "#4D0A62",
    "fb-black": "#18191a",
    "fb-dark-gray": "#242526",
    "fb-gray": "#3a3b3c",
    "fb-white": "#e4e6eb",
    "fb-silver": "#b0b3b8",
    "red-error": "#dc2626",
```

Figur 11. Kodsnutt av tailwind.config.js från Github. (Marjamäki, 2024)

Denna kodsnutt visar de olika färgerna som används i applikationen, där exempelvis de lila färgerna representerar Polygons färgschema och de med prefixet ”fb” är inspirerade av Facebook.



Figur 12. Skärmdump av hemsidan (PigeonPost, 2024)



Figur 13. Skärmdump av profilsidan (PigeonPost, 2024)

3.3 Distribution av kontraktet till Polygons huvudnät

Efter att applikationen har blivit färdigställd och testad lokalt är det dags att distribuera smarta kontraktet till Polygons huvudnät. Dessa steg liknar dem som togs när kontraktet distribuerades till Mumbais testnätverk, med den skillnaden att vi nu använder Polygons huvudnät. Efter att kontraktet har distribuerats måste det sedan också verifieras för att kunna importeras till Thirdweb och därmed använda deras tjänster.

Eftersom det nya kontraktet på Polygons huvudnät är identiskt med kontraktet på testnätverket krävs det inte mycket ändringar i front-end-koden. Det enda som behöver ändras är variabeln *activeChain* från ”mumbai” till ”polygon” i *_app.js* filen och att ändra variabeln *CONTRACT_ADDRESS* i *addresses.js* filen till den nya kontraktadressen.

3.4 Distributionen till Vercel

För att publicera applikationen så att alla kan använda den och interagera med vårt smarta kontrakt kommer vi att använda Vercel. Valet föll naturligt på Vercel eftersom vi använder Next.js som vårt ramverk för att bygga applikationen. Processen är så enkel som att ladda upp sin kod till GitHub, därefter besöker man Vercels webbplats, kopplar sitt GitHub konto till deras tjänst, importerar projektet dit och lägger till de miljövariabler som behövs, vilket i vårt fall är API-nyckeln vi använt. Sedan är det bara att klicka på ”deploy” och vänta några minuter, så är webbsidan online. Det kostar ingenting att använda tjänsten, men den innebär att man får en domän som slutar på vercel.app, vilket kanske inte är det mest önskvärda. Dock finns det möjlighet att ändra detta senare om man så önskar.

4 Resultat

Den decentraliserade applikationen uppnådde framgångsrikt sina primära funktioner, användare kan skapa och se inlägg på en blockkedjebaserad plattform. Detta säkerställer att inläggen är oföränderliga och inte kan censureras. Ett smart kontrakt, utvecklat med Solidity, implementerades på Polygons testnät för att hantera lagring och validering av användardata. Integrering med Thirdweb effektiviserade kommunikationen med detta

kontrakt. Frontend, byggt med Next.js och TailwindCSS, visualiserar effektivt användarinteraktioner.

4.1 Funktionalitet och decentralisering

Det finns dock begränsningar att ta hänsyn till. Varje interaktion med blockkedjan, såsom att posta eller gilla, kräver gasavgifter. Även om Thirdweb förenklar utvecklingsprocessen är applikationen beroende av dem som en tredjepartsleverantör, vilket skapar en potentiell svag punkt. Dessutom undergräver Vercels centraliserade hostning delvis idealet om fullständig decentralisering. Prestandamässigt kanske Thirdweb API inte alltid ger den smidigaste användarupplevelsen.

4.2 Framtida förbättringar

Framöver finns det utrymme för förbättringar. Att distribuera applikationen på IPFS (InterPlanetary File System) skulle uppnå sann decentralisering, även om det kan innebära sämre prestanda. Att hitta en balans mellan decentralisering och användarupplevelse är avgörande. Säkerheten skulle kunna förbättras ytterligare genom att granska det smarta kontraktet. Ramverk som Hardhat kan användas för att automatisera tester och simulera realistiska användningsscenarier. För extra säkerhetsgarantier kan experts anlitas för att utföra revisioner av kontraktet.

För större projekt och bästa praxis kan TypeScript förbättra kodeffektiviteten i framtiden. Slutligen skulle mappstrukturen kunna organiseras bättre, separera komponenter som navigationsfältet till en annan mapp. Även om dessa aspekter inte prioriterades initialt, kan de hanteras i framtida förbättringar.

5 Slutledning

Detta projekt utforskade potentialen av att använda blockkedjateknik för att skapa en decentraliserad applikation, mera specifikt en social medieplattform. Vi vägde för- och nackdelarna och det medför blandade känslor.

Från den ena synvinkeln är tanken på en helt decentraliserad social medieplattform mycket tilltalande. Ur ett annat perspektiv är det inte möjligtvis genomförbart att lagra all data på blockkedjan för storskalig användning för tillfället. Transaktionskostnaderna utgör en betydande utmaning. Gasfria transaktioner, där den tredje parten täcker kostnaderna, kan vara en lösning, men det återinför en centraliserad komponent.

En hybridinsats kan vara svaret. Den avgörande informationen som användarposter och bildlänkar skulle kunna finnas på blockkedjan, medan mindre kritiska funktioner som användarnamn, gillande och kommentarer kan lagras på en centraliserad server för effektivitet och kostnadseffektivitet.

Web3 som är den nästa iterationen av webben, innebär inte nödvändigtvis en fullständig övergång till decentralisering. Även om blockkedjan är revolutionerande kanske den inte är den perfekta lösningen för allt. Vi försöker fortfarande ta reda på dess bästa användningsområden.

Ta till exempel Bitcoin. Satoshi Nakamoto såg det som en peer-to-peer betalningsmetod, utan att behöva vara beroende av en tredje part. Över ett decennium senare betraktas Bitcoin främst som en värdebevarare. Det används sällan för vardagliga transaktioner, förutom för olagliga varor. Även om det inte uppfyller sitt ursprungliga syfte har Bitcoin hittat en värdefull nisch.

Vad sägs om andra protokoll med smart kontraktfunktionalitet? Liknande Bitcoin är deras idealiska användningsfall ännu inte fastställda.

Den nuvarande högkonjunkturen på kryptomarknaden är en bra tidpunkt att reflektera över några av de största frågorna inom blockkedjerummet. Girighet verkar vara en dominerande kraft i samband med människor som satsar pengar på protokoll för potentiell vinst, i stället för att fokusera på verkliga tillämpningar. Boom- och bustcyklerna förvärrar detta: under högkonjunkturer är hypen stor, följt av kraschar som rensar ut dem som

bara är intresserade av en snabb vinst. Dessa perioder efter kraschar är ofta de mest produktiva, eftersom sanna troende på tekniken förblir hängivna åt dess utveckling.

Kärnan i lockelsen med blockkedjan ligger i decentraliseringen, den tar makten från centraliserade enheter som banker och regeringar. Tanken på ett community-drivet tillvägagångssätt är spännande. Verkligheten är däremot att många krypto investerare förvarar sina tillgångar på centraliserade börser, vilket motsäger decentraliseringens anda. De nyligen inträffade konkursfallen för FTX och Celsius fungerar som starka påminnelser om riskerna förknippade med centraliserade börser. I dessa fall förlorade många människor sina kryptotillgångar, inte på grund av en minskning av värde, utan eftersom de inte kunde ta ut sina medel från plattformarna. Detta beror på att, till skillnad från att lagra krypto i en personlig plånbok där du kontrollerar de privata nycklarna, håller centraliserade börser de privata nycklarna till dina tillgångar.

Decentralisering väcker också frågor. Den nyligen godkända Bitcoin ETF:en är motstridig. Medan Bitcoin nu värderas mer som en värdebevarare går köp av Bitcoin ETF – en centraliserad produkt – emot idén av att äga och kontrollera sin egen Bitcoin.

En annan utmaning kommer från regeringar och centralbankers digitala valutor (CBDC). Förståeligt nog är regeringen och banker försiktiga med decentraliserade valutor som hotar deras kontroll. CBDC:er, stabila mynt knutna till befintliga valutor, presenteras som ett sätt att dra nytta av blockkedjetekniken samtidigt som man behåller tillsynen. Även om detta kan verka som ett steg framåt kan det leda till ökad statlig kontroll och underminera användarens integritet. Blockkedjans skönhet ligger i dess förmåga att erbjuda anonymitet och öppenhet. Om transaktioner kan kopplas till enskilda personer försvinner anonymiteten. CBDC:er kan vara ett verktyg för regeringar att utöva större kontroll över sina medborgare.

Detta projekt har visat att blockkedjan är en kraftfull teknik med enorm potential, men mycket återstår att spekulera om. Framtiden håller nyckeln till att låsa upp dess verkliga möjligheter. För närvarande befinner vi oss i upptäckfasen, där vi utforskar hur vi bäst kan utnyttja denna revolutionerande teknik.

Källor

Abu Bakr, A. (2023) *A Comprehensive Guide to Next.js* https://medium.com/@ahmed.nums345/a-comprehensive-guide-to-next-js-5f3b03b49def

Becher, B. (2023) *What is Consensus Mechanism* https://builtin.com/blockchain/consensus-mechanism

Binance Academy (2018) *Vad är bevis på insats (PoS)?* https://academy.binance.com/sv/articles/proof-of-stake-explained

Binance Academy (2023) *Vad är blockkedjan och hur fungerar det?* https://academy.binance.com/sv/articles/what-is-blockchain-and-how-does-it-work

Bäckgren, N. (2024) *Tutkija huolissaan ilmiöstä Helsingin yliopistolla: Oikeistolaiset kokevat puhumisen usein vaikeaksi* https://www.hs.fi/pkseutu/art-2000010144049.html

Coinbase (n.d.) *What is Polygon (MATIC)* https://www.coinbase.com/learn/crypto-basics/what-is-polygon

CoinMarketCap (n.d.) *Web 3.0* https://coinmarketcap.com/academy/glossary/web-3-0

Cointelegraph. (n.d.) *MetaMask tutorial for beginners: How to set up a MetaMask wallet?* https://cointelegraph.com/learn/metamask-tutorial-for-beginners-how-to-set-up-a-metamask-wallet

Collins, P. (2022) *Learn Solidity, Blockchain Development, & Smart Contracts | Powered By AI – Full Course (0 - 6)* [Video] YouTube. https://www.youtube.com/watch?v=umepbfKp5rI&t=926s

Deshpande, C. (2023). *The Best Guide to Know What Is React* https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

Digiconomist (n.d. a) *Bitcoin Energy Consumption Index* https://digiconomist.net/bitcoin-energy-consumption Hämtad: 3.4.2024

Digiconomist (n.d. b) *Ethereum Energy Consumption Index* https://digiconomist.net/ethereum-energy-consumption Hämtad: 16.2.2024

Fitzgerald, A. (2022) *Tailwind CSS: What It Is, Why Use It & Examples* https://blog.hubspot.com/website/what-is-tailwind-css

Hayes, A. (2012) *Blockchain Facts: What Is It, How It Works, and How It Can Be Used* https://www.investopedia.com/terms/b/blockchain.asp

Investopedia (2024) *What Are Crypto Tokens, and How Do They Work?* https://www.investopedia.com/terms/c/crypto-token.asp#toc-what-are-crypto-tokens

Jani, J. (2023) *Crypto: The World's Greatest Scam.* [Video] YouTube. https://www.youtube.com/watch?v=ORdWE_ffirg&t=1373s

Kaur, G. (2023) *Polygon blockchain explained: A beginner's guide to MATIC* https://cointelegraph.com/learn/polygon-blockchain-explained-a-beginners-guide-to-matic

Lemmens, C. (2023). *Thirdweb, the Super App for Airdrops* https://www.altcoin-buzz.io/spotlight/thirdweb-the-super-app-for-airdrops/

Marjamäki, A. (2024) *PigeonPost* [Source code] GitHub https://github.com/attemar-jamaki/pigeonpost

Mattox, C. (2023) *Here's what you need to know about the Twitter Files* https://ameri-cansforprosperity.org/blog/heres-what-you-need-to-know-about-the-twitter-files/

Nakamoto, S. (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System* https://bitcoin.org/bitcoin.pdf

Peck, M. (2012) *Bitcoin: The Cryptoanarchists' Answer to Cash* https://spec-trum.ieee.org/bitcoin-the-cryptoanarchists-answer-to-cash

PigeonPost (2024) *PigeonPost* https://pigeonpost.vercel.app/

Rasure, E. (2023) *DigiCash: Meaning, History, Implications* https://www.in-vestopedia.com/terms/d/digicash.asp

Ravikiran, A. S. (2023) *What is Blockchain Technology? How Does Blockchain Work?* https://www.simplilearn.com/tutorials/blockchain-tutorial/blockchain-technol-ogy

Sharma, R. (2023) *Bitcoin: What Is Decentralized Finance (DeFi) and how Does It Work?* https://www.investopedia.com/decentralized-finance-defi-5113835

Sharma, R. (2024) *Non-Fungible Token (NFT): What It Means and How It Works* https://www.investopedia.com/non-fungible-tokens-nft-5115211

Shevchuk, V. (2023). *The Future of Web 3.0 Development: Thirweb SDK* https://easy-web.medium.com/the-future-of-web-3-0-development-thirdweb-sdk-a2d35c2cf3dd

Simplilearn (2023) *What is Solidity Programming: Data Types, Smart Contracts, and EVM?* https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-soli-dity-programming

Solidity (n.d.) *Solidity Documentation* https://docs.soliditylang.org/en/v0.8.25/

The Verge (2021) *Blockchain, explained* https://www.theverge.com/22654785/blockchain-explained-cryptocurrency-what-is-stake-nft

Worldcoin (2023) *What Is the Ethereum Virtual Machine? A 101 Guide* https://worldcoin.org/articles/what-is-the-ethereum-virtual-machine