Sistem Voting Elektronik Berbasis Ethereum Smart Contract

***Ethereum Smart Contract Based E-Voting System***

**I Dewa Gede Dirga Yasa1, I Putu Arya Dharmadi2, A. A. Kt. Agung Cahyawan Wiranatha3**

123Teknologi Informasi, Fakultas Teknik, Universitas Udayana, Jimbaran, Badung, Bali

*1email penulis pertama, 2email penulis kedua, 3email penulis ketiga*

Naskah diterima: ­­­­, direvisi: ,disetujui:

***Abstract***

The voting method has often made joint decisions, but digital-based voting system is still rare because of digital data is easy to manipulate, causing public distrust. The purpose of this study is to develop a system for voting with maintained data integrity. The development of this voting system uses Scrum method, which is every feature has a planning, working and testing cycle. The results obtained from this study are that the integrity of the election results is very well maintained because if the manipulate data function on the smart contract cannot be used anymore then there is no way to manipulate data because the database used by the Ethereum node is locked. Based on data integrity testing, blockchain that supports smart contracts such as Ethereum can be used as a data storage medium for election results because data integrity is very well maintained.

**Keywords:** voting, blockchain, ethereum, smart contract.

Abstrak

Metode voting banyak dilakukan dalam pengambilan keputusan, namun sistem voting digital masih jarang ditemui karena sifat data digital yang mudah dimanipulasi sehingga menimbulkan ketidakpercayaan masyarakat. Penelitian ini bertujuan untuk mengembangkan sistem voting dengan integritas data yang terjaga. Pengembangan ini menggunakan metode Scrum yang memampukan setiap pengerjaan fitur memiliki siklus perencanaan, pengerjaan dan pengujiannya masing-masing yang terstruktur dan terencana. Pengujian integritas data dilakukan dengan percobaan mengubah variabel yang tersimpan pada database blockchain dan diperoleh hasil bahwa tidak memungkinkan dilakukan perubahan data secara langsung ke dalam database yang digunakan oleh node ethereum. Hal ini dikarenakan ketika fungsi pada smart contract untuk melakukan manipulasi data tidak lagi bisa digunakan, maka tidak ada cara lain yang dapat digunakan untuk melakukan manipulasi data, dengan kata lain database yang digunakan oleh node Ethereum sudah terkunci. Hal ini tentunya dapat membuktikan bahwa blockchain yang mendukung smart contract seperti Ethereum mampu menyimpan data dengan integritas yang sangat terjaga.

**Kata kunci:** voting, blockchain, ethereum, smart contract.

PENDAHULUAN

Pengambilan keputusan bersama memiliki beberapa cara dalam pelaksanaannya, diantaranya musyawarah mufakat, voting dan aklamasi. Metode pengambilan keputusan yang paling sering digunakan adalah voting karena keputusan dapat dihasilkan dalam jangka waktu yang relatif cepat. Seiring dengan berkembangnya teknologi, kecepatan dalam mengakses informasi adalah prioritas sehingga cara voting konvensional yang menggunakan kertas dan paku bukan menjadi pilihan yang optimal. Berdasarkan permasalahan di atas, negara-negara di dunia mulai mengembangkan voting elektronik atau yang biasa disebut dengan *e-voting. E-voting* memiliki beberapa model yang digunakan yaitu *internet poll site voting, kiosk voting* dan *internet voting. Internet poll site voting* merupakan model *e-voting* yang menggunakan komputer di lokasi pemilihan dan menggunakan internet untuk mengirim data dari lokasi pemilihan ke otoritas penyelenggara pemilihan. Sehingga pemilih masih harus datang ke lokasi pemilihan untuk melakukan voting. *Kiosk voting* adalah model *e-voting* yang pemilihnya bisa menggunakan komputer di lokasi yang sudah menjalankan kerja sama dengan otoritas penyelenggara pemilihan misalnya komputer di perpustakaan daerah, sekolah dan mall. Sedangkan *internet voting* adalah model *e-voting* yang sepenuhnya menggunakan internet sehingga pemilih dapat melakukan voting dari perangkat masing-masing (Habibi and Nurmandi 2018).

*E-voting* juga memiliki beberapa masalah yang membuat *e-voting* mendapatkan penolakan oleh masyarakat yaitu masalah kerahasiaan dan kepercayaan terhadap data. Salah satu teknologi yang dapat mengatasi masalah tersebut adalah Ethereum Smart Contract. Ethereum *Smart Contract* adalah program komputer yang berjalan di dalam jaringan *blockchain* Ethereum. Program komputer tersebut melakukan transaksi dalam jaringan *blockchain* dan program komputer tersebut tidak dapat dikontrol oleh seorang pengguna. Jadi *smart contract* hanya akan berjalan sesuai dengan kode program yang disimpan. Karena sifatnya yang berjalan secara autonomous maka dapat menjadi solusi terhadap masalah kepercayaan penyimpanan data sensitif.

Berdasarkan permasalahan diatas, maka penelitian ini dilakukan dengan tujuan dapat memahami proses kerja sistem voting elektronik yang berbasis Ethereum *smart contract*, mengetahui perancangan dan proses pembangunannya serta mengetahui hasil pengujian dan kinerja sistem. Selain itu, penelitian ini juga dilakukan dengan harapan mampu menghilangkan *trust issue* dalam pelaksanaan voting, meningkatkan keamanan data hasil voting. Sistem voting yang dilakukan secara elektronik tentunya juga dapat menghemat waktu dalam proses pengambilan data voting, meminimalisir biaya yang perlu dikeluarkan dalam proses pengambilan data voting, dan juga mmpermudah proses pengambilan data voting karena dapat dilakukan dari mana saja melalui perangkat pengguna yang terkoneksi internet. Karena luasnya permasalahan dan pemecahan masalah yang mungkin dilakukan, penelitian juga dibatasi dengan berfokus pada perancangan Sistem Voting Elektronik dan *platform blockchain* yang digunakan adalah Ethereum.

Penelitian ini tentunya memiliki persamaan dan atau keterikatan dari beberapa penelitian yang telah dilakukan sebelumnya baik dari segi teori maupun metode yang digunakan. Beberapa diantaranya adalah penelitian mengenai pengembangan sistem *e-voting* berbasis web menggunakan keamanan algoritma Rivest Shamir Adleman (RSA). Aplikasi ini dibangun menggunakan bahasa pemrograman PHP dan *database* MySQLi sebagai *database server*. Fokus dari sistem voting elektronik ini adalah melakukan enkripsi hak suara pemilih pada sistem menggunakan *public key* dan *private key* untuk proses verifikasi sehingga tidak ada manipulasi data hak suara pemilih. Hasil yang diperoleh adalah sistem dapat melakukan validasi pemilih dan melakukan verifikasi hak suara apakah telah terjadi manipulasi data selama pengiriman atau tidak (Ridwan, Arifin and Yulianto 2016). Penelitian selanjutnya adalah penelitian mengenai sistem *e-voting* berbasis SMS *gateway* yang dirancang menggunakan *software* Gammu dan menggunakan bahasa pemrograman PHP dan *database* MySQL. Uji coba pemungutan suara dilakukan dengan mengirim pemberitahuan melalui SMS kepada pemilih sesuai dengan data yang telah disimpan dan pemilihan dilakukan dengan cara mengirim SMS balasan sesuai dengan format yang telah ditentukan (Adhi and Harjono 2014). Perancangan sistem voting elektronik juga diterapkan dalam penelitian menggunakan *smart contract blockchain*. Sistem dirancang menggunakan bahasa pemrogaman *solidity* dan *smart contract* menghasilkan kode unik pada setiap pemilihan baru. Pengujian menghasilkan bahwa manipulasi hasil pemungutan suara tidak dapat dilakukan karena setiap pemilih hanya dapat memiliki satu akun dan satu *address blockchain* (Setia and Susanto 2019). Perancangan sistem *e-voting* juga diterapkan menggunakan metode enkripsi *blockchain*. Enkripsi *blockchain* merupakan salah satu cara pengamanan data yang tidak memungkinkan untuk dilakukan penyusupan data ke dalamnya serta menjaga kerahasiaan pada data pemilih karena dilakukan proses enkripsi pada data pemilih dan apa yang dipilih, akan tetap dapat diperoleh informasi pemilik suara terbanyak (Ardilla 2018). Penelitian lain mengenai sistem *e-voting* menggunakan teknologi *blockchain* adalah penelitian oleh Kurnia Hu, dkk menggunakan *multichain tools*. Penelitian ini menghasilkan sistem voting elektronik yang dapat menyimpan data yang transparan dan dapat diakses oleh publik, menjaga identitas pemilih, menyimpan data suara yang tidak dapat diubah, digandakan, atau dihapus. Berdasarkan hasil kuesioner diperoleh persentase sebesar 55,6 % pengguna menilai baik dan 44,4 % pengguna menilai baik sekali mengenai keseluruhan aplikasi (Kurnia Hu, Palit and Handojo 2019). Aplikasi *e-voting* selanjutnya adalah SEVA: *Secure E-Voting Application*. Sistem dirancang bertujuan untuk memungkinan bagi para pemilih dapat melakukan pemilihan melalui web tanpa terbatas oleh lokasi. Aplikasi dikembangkan dengan *Azure Service Fabric* menggunakan arsitektur *microservice*. Aplikasi ini diterapkan pada *cluster five-node* menggunakan *Server Fabric Local Cluster* (Johari, et al. 2020). Penelitian selanjutnya adalah penerapan teknologi *blockchain* dalam layanan *E-commerce* pada *OJS (Open Journal System)*. Penggunaan teknologi *blockchain* dalam penelitian ini bertujuan untuk mengefisiensi manajemen identitas, membangun sistem pelacakan dan mengidentifikasi keaslian produk, dapat menyinkronkan data yang tersimpan dalam *blockchain* ke semua jaringan pengguna, dapat menjadikan sistem pembayaran yang lebih mudah, efisien dan terdokumentasikan dengan baik. Serta kemudahan bagi mahasiswa, dosen dan pihak lain dalam melakukan proses transaksi (Rahardja, et al. 2020). Berdasarkan penelitian-penelitian tersebut, maka dilakukanlah penelitian ini sebagai pengembangan lebih lanjut di bidang teknologi *blockchain* pada sistem voting elektronik.

*Blockchain* adalah teknologi pencatatan transaksi yang menghilangkan pihak ketiga atau *central authority* dalam sebuah transaksi. *Blockchain* juga dapat dikatakan sebagai buku besar digital dimana setiap transaksi yang terjadi di jaringan *blockchain* akan dicatat dan diamankan di dalam *peer* atau *node* yang terlibat di jaringan *blockchain*. Secara umum cara kerja *blockchain* adalah transaksi yang terjadi di jaringan *blockchai*n akan dicatat dalam blok. Blok akan menghasilkan sebuah *hash*. *Hash* blok pada transaksi sebelumnya akan dicatat di blok saat ini. Begitu pula *hash* blok saat ini akan dicatat di blok selanjutnya sehingga akan menghasilkan struktur seperti rantai (Pierro 2017). Jika ada pihak yang ingin melakukan kecurangan dengan cara mengubah data transaksi dalam suatu blok, maka pihak tersebut harus melakukan *hashing* ulang dari blok yang diubah hingga blok yang terjadi saat ini. Tidak hanya itu, karena blok disimpan secara terdistribusi, maka pihak tersebut juga harus mengubah data blok di setiap *node* yang terlibat dalam jaringan. Hal inilah yang menjadi kekuatan *blockchain* sehingga dapat menghilangkan *central authority* dalam melakukan transaksi.

Ethereum adalah sebuah *blockchain* yang didirikan pada tahun 2015 yang digagas oleh Vitalik Buterin. Salah satu tujuan pengembangan Ethereum adalah untuk memberikan kemampuan eksekusi program komputer pada *blockchain*. Aplikasi terdesentralisasi yang memanfaatkan kemampuan eksekusi program komputer ini sering disebut dengan *dapps. Smart Contract* adalah kumpulan dari kode dan data yang teridentifikasi menggunakan sebuah *address* pada *blockchain* Ethereum. *Smart contract* merupakan salah satu jenis Ethereum *account* sehingga sebuah *smart contract* bisa menyimpan ETH dan mengirim transaksi di jaringan *blockchain*. Akan tetapi *smart contract* tidak dikendalikan oleh seorang pengguna atau EOA. Sebuah *smart contract* akan di-*deploy* ke jaringan *blockchain* dan bekerja sesuai kode yang dimilikinya. Seorang pengguna bisa beriteraksi dengan sebuah *smart contract* dengan cara membuat sebuah transaksi dengan menggunakan *address* *smart contract* sebagai *recipient*. Kode *smart contract* dapat dibuat dengan bahasa pemrograman yang lebih mudah dipahami yaitu Solidity. Solidity adalah bahasa pemrograman yang diusulkan oleh Gavin Wood pada tahun 2014. Hingga saat ini Solidity menjadi bahasa pemrograman terpopuler untuk membuat *smart contract* karena Solidity didesain menyerupai ECMAScript sehingga lebih familiar di kalangan *developer.*

Konsensus dalam *blockchain* sangat penting karena data yang terdesentralisasi sehingga sangat memungkinkan terjadinya perbedaan daya yang tercatat. Secara umum pengertian konsensus adalah sebuah kesepakatan bersama yang telah dicapai oleh suatu kelompok terhadap suatu masalah. Dengan adanya algoritma konsensus ini maka jaringan tidak akan kebingungan ketika terjadi perbedaan pencatatan data dan menentukan data mana yang akan digunakan. Sebagian besar ekosistem *blockchain* saat ini menggunakan algoritma *Proof of Work* sebagai konsensus di jaringan *blockchain*. Cara kerja algoritma konsensus *proof of work* adalah setiap *node* yang tergabung dalam sebuah jaringan *blockchain* akan berlomba untuk memproses transaksi yang terjadi dalam kurun waktu tertentu dan *node* tercepat yang bisa menyelesaikan pembuatan blok baru akan menyebarkan blok tersebut ke jaringan. *Account* yang terhubung dengan *node* tersebut akan mendapatkan insentif sesuai dengan mata uang jaringan *blockchain*. Sedangkan algoritma konsensus *Proof of Authority*, dalam jaringan sudah di atur *account* mana saja yang boleh melakukan pembuatan blok. Karena *account* yang melakukan pembuatan blok sudah ditentukan sehingga tidak akan terjadi persaingan dalam hal pembuatan blok sehingga tingkat *difficulty* dari pembuatan blok bisa diturunkan. Dengan tingkat *difficulty* yang rendah maka jaringan dapat berjalan dengan komputer dengan daya komputasi rendah sehingga bisa menghemat biaya infrastruktur.

Besar biaya yang dibutuhkan untuk melakukan tindakan pada jaringan *blockchain* dikalkulasi menggunakan *Gas*. Semua aktivitas yang dilakukan pada jaringan Ethereum seperti pengiriman ETH hingga pemanggilan metode *smart contract* membutuhkan *gas*. *Gas* yang dibutuhkan harus dibayar menggunakan ETH. Besarnya biaya *gas* yang harus dibayarkan ini biasanya disebut dengan *gas fee* sedangkan batas maksimal *gas* yang boleh digunakan untuk melakukan sebuah transaksi disebut dengan *Gas limit*. *Gas limit* sangat penting karena jika tidak ada *gas limit* maka jika terjadi suatu kesalahan misalnya kesalahan *smart contract* yang menyebabkan pemanggilan berulang akan terus menggunakan ETH yang terdapat pada *account eksekutor* untuk membayar *gas* hingga ETH pada *account* tersebut habis.

METODE

Penelitian ini bertempat di Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana Bali yang berlokasi di Jimbaran*,* Kabupaten Badung, Bali, dilaksanakan pada bulan Agustus 2020 sampai dengan Januari 2022. Tempat dan waktu tersebut diatas ditentukan karena mencakup semua aspek pendukung sebagai upaya agar penelitian berjalan dengan baik.

Penelitian ini menggunakan metode pengembangan *Agile Software Development Methods.* Metode *agile* merupakan sekumpulan metode pengembangan perangkat lunak yang berbasis pada pengembangan iteratif yang dilakukan dengan kolaborasi yang terorganisir. Metode *agile* berfokus pada perkembangan yang cepat, perangkat lunak yang dirilis bertahap, mengurangi *overhead* proses, dan menghasilkan kode berkualitas tinggi dan pada proses perkembangannya melibatkan pelanggannya secara langsung (Sommerville 2011). *Agile Software Development Methods* masih terbagi lagi menjadi beberapa jenis bagian dan penelitian ini menggunakan jenis metode *Scrum*. Metode *scrum* membagi proses *development* menjadi beberapa *sprint*. *Sprint* ini akan bertindak sebagai *milestone* dalam pembuatan *software* (Pressman 2010)*.*

Tahap pertama adalah tahap *backlog.* Tahap ini melakukan proses menyusun rincian prioritas pada fitur-fitur yang akan dibangun pada sistem yang akan dikembangkan. Umumnya, tahapan ini bertujuan untuk merancang pekerajaan-pekerjaan yang akan dilakukan berdasarkan permintaan atau kebutuhan *user.* Berdasarkan implementasi tahap *backlog* ini diperoleh hasil bahwa proses-proses yang akan dilakukan pada perancangan sistem *e-voting* adalah perancangan *database*, perancangan *API* *backend,* integrasi dengan *ethereum node,* dan perancangan *frontend.* Isi pada masing-masing proses secara rinci ditambahkan setiap memulai siklus pengembangan karena satu proses dan lainnya tidak diselesaikan secara langsung dalam suatu waktu sehingga dalam setiap proses dapat terjadi perubahan dan penambahan lebih lanjut. Tahap selanjutnya adalah memasukan pekerjaan-pekerjaan yang harus dilakukan dan menyusun prioritasnya ke dalam *sprints.* Proses ini menyusun kegiatan yang akan dilakukan untuk memenuhi kebutuhan dan ditetapkan dalam *backlog* dengan durasi realisasi selama hari kerja yang ditentukan. Setiap *sprint* mengerjakan hampir keseluruhan proses yang disebutkan dalam *backlog* atau paling sedikit 3 dari 4 proses. Durasi realisasi selama hari kerja yang ditentukan dalam penelitian ini adalah selama 2 minggu. Prioritas pekerjaan yang harus dikerjakan sesuai waktu yang ditentukan adalah perancangan *database*, perancangan *API* *backend,* integrasi dengan *ethereum node,* sedangkan untuk perancangan *frontend,* pada tahap awal pengembangan masih belum mendapat prioritas utama, namun akan menjadi fokus setelah proses-proses lainnya mencapai hasil dengan persentase yang lebih besar. Berikutnya dilakukan tahap *scrum meeting*, dengan menganalisis dari sisi pengembang untuk mengetahui seberapa banyak kemajuan dari kegiatan pengembangan sistem yang telah dilakukan atau dicapai. Tahap ini dilakukan dengan pengujian jenis *automated testing* yaitu *unit testing* dan *e2e testing* terhadap 10 *file* dengan masing-masing *file* terdiri atas kurang lebih 5 skenario pengujian. Langkah selanjutnya adalah tahap *demos*, yaitu memperlihatkan fitur-fitur *software* yang telah dihasilkan dalam proses pengembangan untuk dievaluasi oleh pengguna yang kemudian memberi masukan dan saran atas pengembangan yang telah dilakukan untuk kemudian dilakukan proses evaluasi kembali atas hasil kerja berdasarkan proses penerapan tahap-tahap diatas sesuai dengan waktu yang telah ditentukan.

Implementasi aplikasi sistem voting elektronik ini memiliki beberapa kebutuhan perangkat lunak untuk mengoptimalkan jalannya aplikasi, diantaranya sistem operasi berbasis Linux dan *javascript runtime*. *Virtual Private Server* menggunakan sistem operasi Ubuntu versi 18.04 dan Node.js versi 14. Komunikasi dengan *Virtual Private Server* memerlukan koneksi internet yang stabil serta kecepatan *ping* yang digunakan untuk implementasi adalah dibawah 500 ms. Selain kebutuhan perangkat lunak, terdapat juga beberapa kebutuhan perangkat keras yaitu empat *virtual private server* yang disediakan oleh Google Cloud Platform dan spesifikasi dari *virtual private server* adalah Processor 2 vCPU, RAM 1GB, dan Penyimpanan 10GB. Setelah kebutuhan sistem terpenuhi, implementasi perancangan dan pengujian sistem dapat dilakukan sesuai dengan skenario sistem. Setiap komponen dalam penelitian ini memiliki peran dan keterkaitan antara satu dengan yang lainnya. Gambaran umum sistem ini dapat dilihat pada Gambar 1.

|  |
| --- |
| C:\Users\sarah\Downloads\Telegram Desktop\Gabungan Rev.5 Potrait.png  Gambar 1. Gambaran Umum Sistem |

Semua kebutuhan komputasi yaitu *virtual machine* menggunakan penyedia Google Cloud. Terdapat empat *virtual machine* yang digunakan dalam sistem. Masing-masing *VM* menggunakan OS Ubuntu 18.04. Sebuah *VM* digunakan untuk menjalankan *RDBMS* MySQL, dua buah *VM* untuk menjalankan Openethereum dan yang terakhir digunakan untuk menjalankan dua proses *node* untuk aplikasi *backend* dan *frontend*. Terdapat dua buah *node* Ethereum untuk membuat jaringan *private* yang semirip mungkin dengan jaringan *public* Ethereum misalnya Ethereum Mainnet dan Ethereum Testnet dimana terdapat lebih dari satu *node* yang saling berkomunikasi. Nestjs adalah *framework* yang digunakan untuk menjalankan *backend* sedangkan Angular adalah *framework* yang digunakan untuk menjalankan *frontend*. Baik Nestjs dan Angular dijalankan menggunakan *runtime* Nodejs versi 14. Alur kerja sistem dibagi menjadi beberapa tahap, yaitu pertama *super admin* akan membuat akun untuk *election authority*. *Election authority* akan membuat membuat pemilihan dan menambahkan kandidat pada pemilihan. Pemilihan akan di *review* oleh *super admin* apakah diijinkan untuk di-*deploy* ke *blockchain*. Jika pemilihan sudah di-*deploy* maka *voter* dapat melakukan pemilihan dan data akan terekam pada *blockchain*. Kemudian pemilihan akan dihentikan oleh *election authority* dan *voter* dapat melihat hasil dari pemilihan tersebut.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\usecase.png  Gambar 2. Use Case Diagram |

Interaksi antara *user* dengan sistem yang dikembangkan ini dapat dilihat melalui u*se case* *diagram* pada Gambar 2, dimana terdapat 3 jenis aktor/*user*, diantaranya *super admin, election authority* dan *voter*. Masing-masing aktor memiliki peran dan akses yang berbeda-beda. *Super admin* dapat melakukan penambahan atau mendaftarkan *election authority* ke dalam sistem, melakukan proses *send ether* danproses *deploy* atau aktivasi *election* sehingga dapat digunakan, serta melihat hasil *election*. *Election authority* dapat melakukan pembuatan *election* baru, menambahkan kandidat dalam *election* yang dibuat, menentukan awal dan berakhirnya *election,* serta melihat hasil *election. Voter* dapat melakukan pendaftaran dalam setiap *election* yang diikuti dan melakukan proses *vote* dalam *election* yang dipilih, serta melihat hasil *election.*

HASIL DAN PEMBAHASAN

Pengujian Sistem Voting Elektronik menggunakan dua metode yang berbeda yaitu pengujian otomatis dan pengujian manual. Pengujian otomatis adalah pengujian yang menggunakan *automation tools* untuk membandingkan hasil yang diharapkan dengan hasil yang diperoleh dalam suatu proses. Karena penelitian ini berfokus pada penggunaan *smart contract* maka pengujian dibatasi hanya pada *backend* dan *node* Ethereum saja.

1. Pengujian API Aplikasi

*Test runner* adalah *automation tools* yang digunakan untuk menjalankan skenario pengujian yang telah disiapkan. Pada penelitian ini *test runner* yang digunakan adalah *Jest*. Pengujian dibagi menjadi 3 buah *file* yang masing-masing menguji *endpoint* untuk *Super Admin*, *Election Authority* dan *Voter.*

|  |
| --- |
| Gambar 3. Struktur File Pengujian |

Gambar 3 menunjukkan struktur file yang dibutuhkan Jest untuk melakukan pengujian *endpoint*. *File* shared.ts digunakan untuk menyimpan nilai *constant* yang dibutuhkan untuk pengujian. *File* jest-e2e.json adalah konfigurasi Jest menjalankan pengujian. *File* super-admin.e2e-spec.tsadalah pengujian untuk *endpoint Super Admin*. *File* election-authority.e2e-spec.ts adalah pengujian untuk *endpoint Election Authority*, dan *file* voter.e2e-spec.ts adalah pengujian untuk *endpoint Voter.*

Pengujian yang pertama adalah pengujian untuk *endpoint Super Admin,* pengujian manipulasi data yang dilakukan adalah melakukan *http request* ke *endpoint* yang hanya boleh diakses oleh *Super Admin* kemudian mencocokan *http response* yang diperoleh.

Tabel 1. Skenario Pengujian Endpoint Super Admin

| **No** | **Endpoint yang Diuji** | **Skenario Pengujian** | **Hasil yang Diharapkan** |
| --- | --- | --- | --- |
| 1. | /super-admin/election-authority | Menggunakan *POST request* untuk membuat *Election Authority* | * Data berhasil disimpan di *database* * Mendapatkan *http code* 201 |
| 2. | /super-admin/election-authority | Menggunakan *GET request* | * Mendapatkan semua *Election Authority* * Mendapatkan *http code* 200 |
| 3. | /super-admin/election-authority/:id | Menggunakan *GET request* dengan *id* = 2 | * Mendapatkan *Election Authority* dengan *id* = 2 * Mendapatkan *http code* 200 |
| 4. | /super-admin/election-authority/set-wallet-address/:id | Menggunakan *POST request* untuk menambahkan *wallet address* | * *Address* berhasil tersimpan di *database* * Mendapatkan *http code* 201 |
| 5. | /super-admin/election/ready-to-deploy | Menggunakan *GET request* | * Mendapatkan Pemilihan yang siap di-*deploy* * Mendapatkan *http code* 200 |
| 6. | /super-admin/deploy-election/:electionId | Menggunakan *POST request* untuk *deploy* pemilihan | * Berhasil *deploy* pemilihan ke jaringan *Ethereum* * Mendapatkan *http code* 201 |

Pengujian terhadap *endpoint Super Admin* dijalankan dengan cara menjalankan perintah npm run test:e2e super-admin.e2e-spec.ts pada *terminal*. Berdasarkan skenario pengujian diatas, diperoleh hasil bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Super Admin* berhasil dijalanakan, dapat dilihat pada Gambar 4.

|  |
| --- |
| Gambar 4. Hasil Pengujian Endpoint Super Admin |

Pengujian selanjutnya adalah pengujian untuk *endpoint Election Authority*, pengujian manipulasi data yang dilakukan adalah melakukan *http request* ke *endpoint* yang hanya boleh diakses oleh *Election Authority* kemudian mencocokan *http response* yang diperoleh.

Tabel 2. Skenario Pengujian Endpoint Election Authority

| **No** | **Endpoint yang Diuji** | **Skenario Pengujian** | **Hasil yang Diharapkan** |
| --- | --- | --- | --- |
| 1. | /election-authority/election | Menggunakan *POST request* untuk membuat pemilihan | * Data berhasil disimpan di *database* * Mendapatkan *http code* 201 |
| 2. | /election-authority/elections | Menggunakan *GET request* | * Mendapatkan semua pemilihan * Mendapatkan *http code* 200 |
| 3. | /election-authority/election/:id | Menggunakan *GET request* dengan *id* = 1 | * Mendapatkan pemilihan dengan *id* 1 * Mendapatkan *http code* 200 |
| 4. | /election-authority/add-candidate/:id | Menggunakan *POST request* untuk menambahkan kandidat | * Data berhasil disimpan di *database* * Mendapatkan *http code* 201 |
| 5. | /election-authority/election/:id/ candidates | Menggunakan *GET request* | * Mendapatkan kandidat dari sebuah pemilihan * Mendapatkan *http code* 200 |
| 6. | /election-authority/election/:id/  ready | Menggunakan *POST request* untuk mengubah status pemilihan | * Status pemilihan di *database* berubah * Mendapatkan *http code* 200 |
| 7. | /election-authority/election-participant/accept/:id | Menggunakan *POST request* untuk menerima pemilih | * Status pemilih di *database* berubah * Mendapatkan *http code* 201 |
| 8. | /election-authority/election-participant/reject/:id | Menggunakan *POST request* untuk menolak pemilih | * Status pemilih di *database* berubah * Mendapatkan *http code* 201 |
| 9. | /election-authority/start-election/:id | Menggunakan *POST request* untuk memulai pemilihan | * Status pemilihan berubah di *database* * Mendapatkan *http code* 201 |
| 10. | /election-authority/end-election/:id | Menggunakan *POST request* untuk menghentikan pemilihan | * Status pemilihan berubah di *database* * Mendapatkan *http code* 201 |

Pengujian terhadap *endpoint Election Authority* dijalankan dengan cara menjalankan perintah npm run test:e2e election-authority.e2e-spec.ts pada *terminal.* Berdasarkan skenario pengujian diatas, diperoleh hasil bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Election Authority* berhasil dijalanakan, dapat dilihat pada Gambar 5.

|  |
| --- |
| Gambar 5. Hasil Pengujian Endpoint Election Authority |

Pengujian selanjutnya adalah pengujian untuk *endpoint Voter*, Pengujian manipulasi data yang dilakukan adalah melakukan *http request* ke *endpoint* yang hanya boleh diakses oleh *Voter* kemudian mencocokan *http response* yang diperoleh.

Tabel 3. Skenario Pengujian Endpoint Voter

| **No** | **Endpoint yang Diuji** | **Skenario Pengujian** | **Hasil yang Diharapkan** |
| --- | --- | --- | --- |
| 1. | /voter/join/:id | Menggunakan *POST request* untuk mengikuti pemilihan | * Data partisipasi berhasil disimpan di *database* * Mendapatkan *http code* 201 |
| 2. | /voter/election-participation | Menggunakan *GET request* | * Mendapatkan semua partisipasi *user* terkait * Mendapatkan *http code* 200 |
| 3. | /voter/available-election | Menggunakan *GET request* | * Mendapatkan semua pemilihan yang tersedia * Mendapatkan *http code* 200 |
| 4. | /voter/followed-election | Menggunakan *GET request* | * Mendapatkan semua pemilihan yang diikuti * Mendapatkan *http code* 200 |
| 5. | /voter/ended-election | Menggunakan *GET request* | * Mendapatkan semua pemilihan yang telah selesai * Mendapatkan *http code* 200 |
| 6. | /voter/election-detail/:id | Menggunakan *GET request* | * Mendapatkan detail sebuah pemilihan * Mendapatkan *http code* 200 |
| 7. | /voter/vote | Menggunakan *POST request* untuk memilih kandidat | * Data berhasil tersimpan di *database* * Mendapatkan *http code* 201 |
| 8. | /voter/ended-election-detail/:id | Menggunakan *GET request* | * Mendapatkan detail pemilihan yang telah selesai * Mendapatkan *http code* 200 |

Pengujian terhadap *endpoint Voter* dijalankan dengan cara menjalankan perintah npm run test:e2e voter.e2e-spec.ts pada *terminal.* Berdasarkan skenario pengujian diatas, diperoleh hasil bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Voter* berhasil dijalanakan, dapat dilihat pada Gambar 6.

|  |
| --- |
| Gambar 6. Hasil Pengujian Endpoint Voter |

1. Pengujian API Aplikasi

Penelitian ini menggunakan Openethereum sebagai Ethereum client sehingga pengujian ini akan melakukan percobaan untuk mengubah data yang tersimpan pada database yang digunakan oleh Openethereum. Berdasarkan repositori Openethereum, database yang digunakan adalah RocksDB. RocksDB adalah sebuah database dengan paradigma key-value yang mengutamakan performance sehingga hampir tidak ada client GUI yang dapat digunakan untuk melakukan interaksi dengan database tersebut. Pada pengujian ini ada beberapa hal yang perlu dipersiapkan, yang pertama adalah *Node* Openethereum, yang dibutuhkan adalah *node* yang memiliki konfigurasi semirip mungkin dengan *node* Openethereum yang digunakan untuk Sistem Voting Elektronik namun dengan langkah persiapan yang lebih mudah. Jumlah validator yang dibutuhkan adalah 1 dan dijalankan pada *localhost*. Selanjutnya adalah *Block Explorer*, digunakan untuk memberikan gambaran bagaimana kondisi *block* yang telah terbentuk dalam *blockchain*. Repositori untuk projek ini terdapat pada https://github.com/cekingx/explorer.git. Langkah-langkah yang perlu dijalankan adalah *clone repository* ke *local*. Kemudian mengubah variabel *remoteUrl* pada *file app.js* menjadi http://localhost:8540. Kemudian jalankan perintah npm install. Setelah selesai melakukan instalasi, jalankan perintah npm start dan buka *url* http://localhost:8000 pada *browser.* Ketika melakukan pemanggilan fungsi sebuah *smart contract*, hal yang tersimpan didalam *blockchain* adalah *function signature* dari fungsi yang dipanggil dan disimpan dalam bentuk *hex* sehingga diperlukan *decoder* untuk bisa membaca *hex* tersebut. *Ethereum decoder* yang akan digunakan dapat diakses pada *url* https://eth-decoder.cekingx.com. Selanjutnya, *smart contract* yang digunakan adalah *smart contract* dengan logika yang sederhana dimana hanya terdapat dua fungsi yang bisa digunakan yaitu getPerson dan setPerson. *Smart contract* yang digunakan pada pengujian ini memiliki sebuah fungsi untuk melakukan perubahan data yaitu setPerson. Namun fungsi tersebut hanya bisa digunakan sekali saja, sehingga pengujian ini akan berusaha mengubah data yang telah tersimpan di dalam *database* Openethereum.

|  |
| --- |
| Gambar 7. Block Explorer |

Gambar 7 menunjukkan tampilan dari *block explorer* terhadap jaringan Ethereum yang telah disiapkan. Bisa terlihat terdapat 2 *block* dimana *block* 1 terdapat *data deploy smart contract* dan *block* 2 tersimpan data pemanggilan fungsi setPerson dari proses sebelumnya.

|  |
| --- |
| Gambar 8. Detail Block 2 |

Informasi penting yang nantinya digunakan adalah *block hash* dengan nilai 0xb323b235f56bd0276fad76d4f3b21bf83b205c71ace2a97203c80ae0aa1463f0 serta nilai *hash* yang terdapat *field input* pada bagian Transaction #1.

|  |
| --- |
| Gambar 9. Ethereum Decoder 1 |

Gambar 9 menunjukkan tampilan dari hasil decoding dari input parameter yang tersimpan di dalam *blockchain* menggunakan *ethereum decoder*. Seperti yang ditampilkan, *string parameter* yang tersimpan adalah “Dirga” sesuai dengan proses penggunaan *smart contract.*

|  |
| --- |
| Gambar 10. Param Generator |

Gambar 10 menunjukkan tampilan untuk membuat *input parameter* untuk *blockchain*. Pada pengujian ini akan berusaha mengubah *string parameter* yang tersimpan dari “Dirga” menjadi “Yasa”.

|  |
| --- |
| Gambar 11. Ethereum Decoder 2 |

Gambar 11 menunjukkan hasil *decoding input parameter* yang dihasilkan oleh *param generator*. Seperti yang ditampilkan, *string parameter* dari *input parameter* tersebut adalah “Yasa”.

|  |
| --- |
| Gambar 12. RocksDB Administrative Tools |

Gambar 12 menunjukkan *value* yang dipasangkan terhadap sebuah *key* yaitu *hash* dari *block 2. Hash* yang ditampilkan memiliki struktur data *rlp* sehingga diperlukan *tools rlpdump* untuk bisa melihat isi di dalamnya.

|  |
| --- |
| Gambar 13. Struktur Data RLP |

Gambar 13 menunjukkan *rlp* yang telah di-*decoding* menggunakan *rlpdump*. Hal yang perlu diubah adalah elemen ke-6 pada *array* terkecil pertama menjadi *hash* yang telah diperloleh menggunakan *param generator.*

|  |
| --- |
| Gambar 14. Encoding RLP |

Gambar 14 menunjukkan proses untuk mendapatkan struktur *rlp* yang baru. Proses yang dilakukan adalah memasukkan *rlp* yang telah di-*decode* dan elemen ke-6 pada *array* terkecil sudah dimodifikasi disimpan pada *file* file.txt.

|  |
| --- |
| Gambar 15. Update Database RocksDB |

Gambar 15 menunjukkan proses *update value* untuk *key* 0xb323b235f56bd0276fad76d4f3b21bf83b205c71ace2a97203c80ae0aa1463f. Seperti yang ditampilkan, hasilnya kita tidak bisa melakukan perubahan data secara langsung ke *database* yang digunakan oleh *node ethereum*. Sehingga ini membuktikan data yang tersimpan pada *blockchain* sangat terjaga integritasnya.

KESIMPULAN

Kesimpulan yang dapat diperoleh dari penelitian ini adalah Jaringan Ethereum *private* yang digunakan oleh sistem voting elektronik ini menggunakan Openethereum sebagai Ethereum *client*. Dokumentasi yang dimiliki oleh Openethereum sangat lengkap sehingga pengguna bisa membuat jaringan Ethereum *private* sendiri sesuai dengan kebutuhan. Sebagian pengujian *backend* sistem voting elektronik dilakukan secara otomatis menggunakan *library* Jest. Implementasi *automated testing* dalam sistem voting elektronik ini bisa mempersingkat proses *iterasi* pengerjaan sebuah fitur karena pengujian bisa dijalankan dalam hitungan detik. Uji coba integritas terhadap data yang tersimpan dalam Ethereum *client* menunjukkan bahwa *database* yang digunakan oleh Ethereum *client* terkunci ketika Ethereum *client* dijalankan. Cara lain untuk melakukan manipulasi data adalah dengan cara memanggil fungsi yang terdapat pada *smart contract*. Jika fungsi pada *smart contract* tersebut sudah terkunci maka tidak ada cara lain lagi untuk melakukan manipulasi data.

Penelitian ini masih jauh dari kata sempurna dan masih banyak memiliki kekurangan. Namun, Penulis berharap bahwa penelitian ini dapat menjadi inspirasi dalam pengembangan sistem yang lebih baik lagi kedepannya. Hal ini dikarenakan penelitian terhadap Ethereum s*mart contract* saat ini masih jarang ditemui dan minimnya studi literatur yang membahas penelitian yang serupa. Namun, hal ini justru dapat dijadikan peluang bagi para *developer* untuk melakukan eksplorasi, pengembangan dan penyempurnaan lebih lanjut. Masih sangat banyak potensi sistem berbasis *smart contract* yang bisa digali, dengan kata lain tidak menutup kemungkinan bahwa *smart contract* dapat diimplementasikan terhadap sistem-sistem lainnya, mengingat integritas data yang dihasilkan dari penelitian ini sangat baik

UCAPAN TERIMA KASIH

Penelitian ini tentunya dapat terlaksana karena dukungan dari banyak pihak, untuk itu Penulis secara khusus mengucapkan terima kasih kepada Bapak I Putu Arya Dharmadi, S.T.,M.T. dan Bapak A. A. Kt. Agung Cahyawan Wiranatha, S.T.,M.T. selaku dosen pembimbing yang telah banyak meluangkan waktu memberikan dorongan, bimbingan, tuntunan dan kesabaran selama penelitian ini. Penulis juga ingin mengucapkan terima kasih kepada berbagai pihak yang belum dapat disebutkan satu-persatu yang juga berperan penting dalam membatu, memberikan sumbangan ide, pemikiran dan dukungan dalam penelitian ini.

DAFTAR PUSTAKA

Adhi, Rizqi Andhestria, and Harjono. 2014. "Rancang Bangun Sistem Informasi E-Voting Berbasis SMS." *JUITA* III (2): 85-93. doi:10.30595/juita.v3i2.863.

Ardilla, Rizal. 2018. "Rancang Bangun Sistem E – Voting Dengan Metode Enkripsi Blockchain Di Kota Mojokerto." *Repository Universitas Islam Majapahit.* http://repository.unim.ac.id/id/eprint/276.

Habibi, Muhammad, and Achmad Nurmandi. 2018. "Dinamika Implementasi E-Voting di Berbagai Negara." *INA-Rxiv Papers* 1-23. doi : 10.31227/osf.io/bu2ax.

Johari, Rahul, Arvinder Kaur, Mohammad Hashim, Prateek Kumar Rai, and Kanika Gupta. 2020. "SEVA: Secure E-Voting Application in Cyber Physical System." *Cyber-Physical Systems* 1-31. doi : 10.1080/23335777.2020.1837250.

Kurnia Hu, Satria Damai, Henry Novianus Palit, and Andreas Handojo. 2019. "Implementasi Blockchain: Studi Kasus e-Voting." *Jurnal Infra* VII (1).

Pierro, M. Di. 2017. "What Is the Blockchain?" *Computing in Science & Engineering* 19: 92-95.

Pressman, R.S. 2010. *Software Engineering : a practitioner’s approach.* New York: McGraw - Hill.

Rahardja, Untung, Qurotul Aini, M. Yusup, and Aulia Edliyanti. 2020. "Penerapan Teknologi Blockchain Sebagai Media Pengamanan Proses Transaksi E-Commerce." *Journal of Computer Engineering System and Science* V (1): 28--32. https://doi.org/10.24114/cess.v5i1.14893.

Ridwan, Muhammad, Zainal Arifin, and Yulianto. 2016. "Rancang Bangun E-Voting Menggunakan Keamanan RSA." *Jurnal Informatika Mulawarman.* http://dx.doi.org/10.30872/jim.v11i2.210.

Setia, Teresa Enades Hari, and Ajib Susanto. 2019. "Smart Contract Blockchain pada E-Voting." *Jurnal Informatika Upgris* V (2): 188-191. https://doi.org/10.26877/jiu.v5i2.4160.

Sommerville, Ian. 2011. *Software Engineering.* 9. United States of America: Pearson Education Inc., publishing as Addison-Wesley.