**SISTEM VOTING ELEKTRONIK BERBASIS ETHEREUM *SMART CONTRACT***

**TUGAS AKHIR**

Diajukan guna memenuhi sebagian persyaratan dalam rangka menyelesaikan Pendidikan Sarjana Strata Satu (S1) Program Studi Teknologi Informasi



**I DEWA GEDE DIRGA YASA**

**NIM: 1705551062**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS UDAYANA**

**2022**

# HALAMAN SAMPUL

**SISTEM VOTING ELEKTRONIK BERBASIS ETHEREUM *SMART CONTRACT***

**TUGAS AKHIR**

Diajukan guna memenuhi sebagian persyaratan dalam rangka menyelesaikan Pendidikan Sarjana Strata Satu (S1) Program Studi Teknologi Informasi



**I DEWA GEDE DIRGA YASA**

**NIM: 1705551062**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS UDAYANA**

**2022**

# HALAMAN JUDUL

# KATA PENGANTAR

Puji dan syukur saya panjatkan kehadapan Tuhan Yang Maha Esa sehingga penulis dapat menyelesaikan tugas akhir dengan judul **“Sistem Voting Elektronik Berbasis Ethereum Smart Contract”**. Dalam penyusunan Tugas Akhir ini, penulis mendapatkan masukan dan bimbingan dari berbagai pihak. Sehubungan dengan hal tersebut pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada:

1. Prof. Ir. Ngakan Putu Gede Suardana, MT, Ph.D selaku Dekan Fakultas Teknik Universitas Udayana.
2. Dr. Ir. I Made Oka Widyantara, ST., MT., IPM, ASEAN Eng. selaku Koordinator Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana.
3. Bapak I Putu Arya Dharmadi, S.T.,M.T. selaku dosen pembimbing I yang telah banyak meluangkan waktu memberikan dorongan, bimbingan, tuntunan dan kesabaran selama penyusunan Tugas Akhir ini.
4. Bapak A. A. Kt. Agung Cahyawan Wiranatha, S.T.,M.T. selaku dosen pembimbing II yang juga telah banyak meluangkan waktu memberikan dorongan, bimbingan, tuntunan dan kesabaran selama penyusunan Tugas Akhir ini.
5. Segenap dosen pengajar yang turut memberikan ilmu dan pengetahuan selama penulis menempuh perkuliahan di Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana.
6. Bapak dan Ibu pegawai di Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana yang telah membantu selama proses perkuliahan dan administrasi.
7. Orang tua serta anggota keluarga yang telah banyak memberikan motivasi dan dukungan baik jasmani maupun rohani selama penulis menyusun Tugas Akhir ini.
8. Teman-teman seperjuangan dan segenap civitas di Program Studi Teknologi Informasi Universitas Udayana yang telah memberikan sumbangan ide dan dukungan dalam penyusunan Tugas Akhir ini.
9. Berbagai pihak yang belum dapat disebutkan satu-persatu yang juga berperan penting dalam membatu, memberikan sumbangan ide, pemikiran dan dukungan dalam penyusunan Tugas Akhir ini

Penulis menyadari bahwa laporan ini jauh dari sempurna baik dalam materi maupun penulisannya. Berkaitan dengan hal tersebut, maka kritik dan saran yang bersifat membangun dari semua pihak sangat diharapkan. Akhir kata, semoga laporan ini dapat memberikan manfaat bagi semua pihak sesuai dengan yang diharapkan.

Denpasar, Januari 2022

I Dewa Gede Dirga Yasa

# ABSTRAK

Pengambilan keputusan bersama menggunakan metode voting sudah banyak dilakukan, namun penggunaan sistem voting berbasis digital masih jarang ditemui. Hal ini dikarenakan sifat data digital yang mudah dimanipulasi tanpa meninggalkan jejak sehingga menimbulkan ketidakpercayaan masyarakat untuk menggunakan sistem digital sebagai media untuk mengambil keputusan yang bersifat sensitif. Penelitian ini bertujuan untuk mengembangkan sistem voting dengan integritas data yang terjaga sehingga dapat mengurangi ketidakpercayaan masyarakat tersebut. Pengembangan sistem voting ini menggunakan salah satu metode yang tergabung di dalam metode pengembangan Agile yaitu Scrum. Metode ini memampukan setiap pengerjaan fitur yang dibutuhkan oleh sistem memiliki siklus perencanaan, pengerjaan dan pengujiannya masing-masing yang terstruktur dan terencana. Pengujian terhadap integritas data dilakukan dengan percobaan mengubah variabel yang tersimpan pada *database blockchain.* Berdasarkan pengujian tersebut, diperoleh hasil bahwa tidak memungkinkan untuk 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*

# ABSTRACT

Joint decision-making using the voting method has been widely used, but the use of digital-based voting systems is still rare. This is due to the nature of digital data that is easy to manipulate without leaving a trace, causing public distrust to use digital systems as a medium for making sensitive decisions. This study aims to develop a voting system with maintained data integrity so as to reduce public distrust. The development of this voting system uses one of the methods incorporated in the Agile development method, namely Scrum. This method enables each feature work required by the system to have its own structured and planned planning, execution and testing cycle. Testing the integrity of the data is done by trying to change the variables stored in the blockchain database. Based on these tests, the results obtained that it is not possible to make changes to the data directly into the database used by the ethereum node. This is because when the smart contract function to manipulate data can no longer be used, then there is no other way that can be used to manipulate data, in other words the database used by the Ethereum node is locked. This can certainly prove that blockchains that support smart contracts like Ethereum are able to store data with very high integrity.

**Keywords :** Voting, Blockchain, Ethereum, Smart Contract

# DAFTAR ISI

[HALAMAN SAMPUL i](#_Toc92832548)

[HALAMAN JUDUL ii](#_Toc92832549)

[KATA PENGANTAR iii](#_Toc92832550)

[ABSTRAK v](#_Toc92832551)

[DAFTAR ISI vii](#_Toc92832553)

[DAFTAR GAMBAR x](#_Toc92832554)

[DAFTAR KODE PROGRAM xiv](#_Toc92832555)

[DAFTAR TABEL xv](#_Toc92832556)

[BAB I PENDAHULUAN 1](#_Toc92832557)

[1.1 Latar Belakang 1](#_Toc92832558)

[1.2 Rumusan Masalah 2](#_Toc92832559)

[1.3 Tujuan 3](#_Toc92832560)

[1.4 Manfaat Penelitian 3](#_Toc92832561)

[1.5 Batasan Masalah 3](#_Toc92832562)

[1.6 Sistematika Penulisan 4](#_Toc92832563)

[BAB II TINJAUAN PUSTAKA 5](#_Toc92832564)

[2.1 *State of the Art* 5](#_Toc92832565)

[2.2 Blockchain 7](#_Toc92832566)

[2.3 Ethereum 8](#_Toc92832567)

[2.3.1 Ethereum Accounts dan Transaction 8](#_Toc92832568)

[2.3.2 Ethereum Code Execution 8](#_Toc92832569)

[2.3.3 Openethereum 9](#_Toc92832570)

[2.3.4 Smart Contract 9](#_Toc92832571)

[2.3.5 Konsensus 10](#_Toc92832572)

[2.3.6 Gas, Gas Fee dan Gas Limit 11](#_Toc92832573)

[2.4 VSCode 12](#_Toc92832574)

[2.5 NodeJS 13](#_Toc92832575)

[2.5.1 NestJS 14](#_Toc92832576)

[2.5.2 Angular 15](#_Toc92832577)

[2.5.3 Web3JS 17](#_Toc92832578)

[2.6 MySQL 18](#_Toc92832579)

[2.7 JSON 20](#_Toc92832580)

[2.8 Google Cloud Platform 20](#_Toc92832581)

[2.8.1 Compute Engine 20](#_Toc92832582)

[2.9 *Agile Software Development Method* 21](#_Toc92832583)

[2.9.1 Backlog 22](#_Toc92832584)

[2.9.2 Sprints 22](#_Toc92832585)

[2.9.3 Scrum Meeting 23](#_Toc92832586)

[2.9.4 Demos 23](#_Toc92832587)

[BAB III METODOLOGI DAN PERANCANGAN SISTEM 24](#_Toc92832588)

[3.1 Tempat dan Waktu Penelitian 24](#_Toc92832589)

[3.2 Metode Penelitian 24](#_Toc92832590)

[3.2.1 Penerapan Tahap Backlog 24](#_Toc92832591)

[3.2.2 Penerapan Tahap Sprints 25](#_Toc92832592)

[3.2.3 Penerapan Tahap Scrum Meeting 25](#_Toc92832593)

[3.2.4 Penerapan Tahap Demos 25](#_Toc92832594)

[3.3 Alat Penelitian Aplikasi 25](#_Toc92832595)

[3.3.1 Kebutuhan Perangkat Lunak Implementasi Aplikasi 26](#_Toc92832596)

[3.3.2 Kebutuhan Perangkat Keras Implementasi Aplikasi 26](#_Toc92832597)

[3.4 Gambaran Umum Sistem 26](#_Toc92832598)

[3.4.1 Use Case Diagram 28](#_Toc92832599)

[3.4.2 Rancangan Alur Sistem 30](#_Toc92832600)

[3.5 Rancangan Basis Data 43](#_Toc92832601)

[3.5.1 Implementasi Basis Data 43](#_Toc92832602)

[3.5.2 Struktur Data Tabel 44](#_Toc92832603)

[3.6 Bahasa Pemrograman 47](#_Toc92832604)

[3.7 Rancangan Antarmuka Aplikasi 47](#_Toc92832605)

[3.8 Rancangan Spesifikasi API 53](#_Toc92832606)

[BAB IV PENGUJIAN DAN ANALISA HASIL 61](#_Toc92832607)

[4.1 Infrastruktur Google Cloud 61](#_Toc92832608)

[4.1.1 Persiapan Infrastruktur 61](#_Toc92832609)

[4.1.2 Persiapan Virtual Machine 68](#_Toc92832610)

[4.2 Konfigurasi Openethereum 71](#_Toc92832611)

[4.2.1 Genesis Block 72](#_Toc92832612)

[4.2.2 Konfigurasi Node Openethereum 74](#_Toc92832613)

[4.2.3 Menjalankan Jaringan 76](#_Toc92832614)

[4.3 Implementasi Desain Antarmuka 78](#_Toc92832615)

[4.3.1 Membuat Pemilihan 78](#_Toc92832616)

[4.3.2 Menambahkan Kandidat 79](#_Toc92832617)

[4.3.3 Deploy Pemilihan 81](#_Toc92832618)

[4.3.4 Mengikuti Pemilihan 84](#_Toc92832619)

[4.3.5 Daftar Peserta Pemilihan 85](#_Toc92832620)

[4.3.6 Memulai Pemilihan 86](#_Toc92832621)

[4.3.7 Vote 87](#_Toc92832622)

[4.3.8 Menghentikan Pemilihan 88](#_Toc92832623)

[4.3.9 Melihat Hasil Pemilihan 90](#_Toc92832624)

[4.4 Implementasi Rancangan API 90](#_Toc92832625)

[4.4.1 Membuat Pemilihan 90](#_Toc92832626)

[4.4.2 Menambahkan Kandidat 92](#_Toc92832627)

[4.4.3 Deploy Pemilihan 93](#_Toc92832628)

[4.4.4 Mengikuti Pemilihan 94](#_Toc92832629)

[4.4.5 Daftar Peserta Pemilihan 95](#_Toc92832630)

[4.4.6 Detail Pemilihan 96](#_Toc92832631)

[4.4.7 Memulai Pemilihan 98](#_Toc92832632)

[4.4.8 Vote 98](#_Toc92832633)

[4.4.9 Menghentikan Pemilihan 99](#_Toc92832634)

[4.4.10 Melihat Hasil Pemilihan 100](#_Toc92832635)

[4.5 Pengujian Sistem Voting Elektronik 102](#_Toc92832636)

[4.5.1 Pengujian API Aplikasi 102](#_Toc92832637)

[4.5.2 Pengujian Integritas Data Blockchain 109](#_Toc92832638)

[BAB V PENUTUP 122](#_Toc92832639)

[5.1 Kesimpulan 122](#_Toc92832640)

[5.2 Saran 122](#_Toc92832641)

[DAFTAR PUSTAKA 124](#_Toc92832642)

# DAFTAR GAMBAR

[Gambar 2. 1 Arsitektur *Blockchain* 7](#_Toc93349613)

[Gambar 2. 2 Visual Studio Code 13](#_Toc93349614)

[Gambar 2. 3 Node.js 14](#_Toc93349615)

[Gambar 2. 4 NestJS 15](#_Toc93349616)

[Gambar 2. 5 Angular 16](#_Toc93349617)

[Gambar 2. 6 Interaksi Web3.JS dengan Node Ethereum 17](#_Toc93349618)

[Gambar 2. 7 *Agile Software Development Method* 21](#_Toc93349619)

[Gambar 3. 1 Gambaran Umum Sistem 27](#_Toc93349622)

[Gambar 3. 2 *Use Case Diagram* 29](#_Toc93349623)

[Gambar 3. 3 Rancangan Alur Sistem *(Super Admin)* 31](#_Toc93349624)

[Gambar 3. 4 Rancangan Alur Sistem *(Election Authority)* 32](#_Toc93349625)

[Gambar 3. 5 Rancangan Alur Sistem *(Voter )* 33](#_Toc93349626)

[Gambar 3. 6 Rancangan Alur Proses *Login* 35](#_Toc93349627)

[Gambar 3. 7 Rancangan Alur Penambahan *Election Authority* 36](#_Toc93349628)

[Gambar 3. 8 Rancangan AlurPembuatan *Election* Baru 37](#_Toc93349629)

[Gambar 3. 9 Rancangan AlurPenambahan Kandidat 38](#_Toc93349630)

[Gambar 3. 10 Rancangan Alur *Deploy Election* 39](#_Toc93349631)

[Gambar 3. 11 Rancangan Alur Registrasi *Voter* 40](#_Toc93349632)

[Gambar 3. 12 Rancangan Alur Pendaftaran *Voter* dalam *Election* 41](#_Toc93349633)

[Gambar 3. 13 Rancangan Alur Proses *Vote* 42](#_Toc93349634)

[Gambar 3. 14 Implementasi Basis Data 43](#_Toc93349635)

[Gambar 3. 15 Rancangan Antarmuka Buat Pemilihan 48](#_Toc93349636)

[Gambar 3. 16 Rancangan Antarmuka Menambahkan Kandidat 49](#_Toc93349637)

[Gambar 3. 17 Rancangan Antarmuka Daftar Pemilihan Siap *Deploy* 50](#_Toc93349638)

[Gambar 3. 18 Rancangan Antarmuka Daftar Peserta Pemilihan 51](#_Toc93349639)

[Gambar 3. 19 Rancangan Antarmuka Detail Pemilihan 52](#_Toc93349640)

[Gambar 3. 20 Rancangan Antarmuka Hasil Pemilihan 53](#_Toc93349641)

[Gambar 4. 1 Halaman Google Cloud untuk Mengelola *Firewall Rule* 62](#_Toc93349642)

[Gambar 4. 2 *Firewall Rule* yang Diperlukan 63](#_Toc93349643)

[Gambar 4. 3 Halaman Google Cloud untuk Mengelola *VM* 64](#_Toc93349644)

[Gambar 4. 4 Spesifikasi *Virtual Machine* 65](#_Toc93349645)

[Gambar 4. 5 Konfigurasi Jaringan *Virtual Machine* 66](#_Toc93349646)

[Gambar 4. 6 Konfigurasi Security *Virtual Machine* 67](#_Toc93349647)

[Gambar 4. 7 Halaman Compute Engine ketika berhasil membuat *VM* 67](#_Toc93349648)

[Gambar 4. 8 *SSH* ke *Virtual Machine* 68](#_Toc93349649)

[Gambar 4. 9 Proses Instalasi *Package* 69](#_Toc93349650)

[Gambar 4. 10 Proses *Download Binary* Openethereum 69](#_Toc93349651)

[Gambar 4. 11 Proses *Extract File* 70](#_Toc93349652)

[Gambar 4. 12 Proses Ubah *Permission File* 70](#_Toc93349653)

[Gambar 4. 13 Proses *Download* Konfigurasi *Node* Ethereum 71](#_Toc93349654)

[Gambar 4. 14 Struktur *File* Konfigurasi 72](#_Toc93349655)

[Gambar 4. 15 *Node* Terhubung 77](#_Toc93349656)

[Gambar 4. 16 Tampilan Halaman Pembuatan Pemilihan 78](#_Toc93349657)

[Gambar 4. 17 Tampilan Pesan Sukses 79](#_Toc93349658)

[Gambar 4. 18 Tampilan Halaman Penambahan Kandidat 80](#_Toc93349659)

[Gambar 4. 19 Tampilan Pesan Sukses 80](#_Toc93349660)

[Gambar 4. 20 Tampilan *Detail* Pemilihan 81](#_Toc93349661)

[Gambar 4. 21 Tampilan Konfirmasi 82](#_Toc93349662)

[Gambar 4. 22 Tampilan Pesan Sukses 82](#_Toc93349663)

[Gambar 4. 23 Tampilan Daftar Pemilihan yang Siap Deploy 83](#_Toc93349664)

[Gambar 4. 24 Tampilan Pesan Sukses 83](#_Toc93349665)

[Gambar 4. 25 Tampilan Daftar Pemilihan 84](#_Toc93349666)

[Gambar 4. 26 Tampilan Pesan Sukses 85](#_Toc93349667)

[Gambar 4. 27 Tampilan Daftar Peserta Pemilihan 85](#_Toc93349668)

[Gambar 4. 28 Tampilan Konfirmasi Mulai Pemilihan 86](#_Toc93349669)

[Gambar 4. 29 Tampilan Pesan Sukses 87](#_Toc93349670)

[Gambar 4. 30 Tampilan *Vote* Sebuah Pemilihan 87](#_Toc93349671)

[Gambar 4. 31 Tampilan Konfirmasi Memilih Kandidat 88](#_Toc93349672)

[Gambar 4. 32 Tampilan Konfirmasi Menghentikan Pemilihan 89](#_Toc93349673)

[Gambar 4. 33 Tampilan Pesan Sukses 89](#_Toc93349674)

[Gambar 4. 34 Tampilan Hasil Pemilihan 90](#_Toc93349675)

[Gambar 4. 35 *HTTP Request* untuk Membuat Pemilihan 91](#_Toc93349676)

[Gambar 4. 36 *HTTP Response* Pembuatan Pemilihan 91](#_Toc93349677)

[Gambar 4. 37 *HTTP Request* untuk Menambahkan Kandidat 92](#_Toc93349678)

[Gambar 4. 38 *HTTP Response* Penambahan Kandidat 93](#_Toc93349679)

[Gambar 4. 39 *HTTP Request Deploy* Pemilihan 94](#_Toc93349680)

[Gambar 4. 40 *HTTP Response Deploy* Pemilihan 94](#_Toc93349681)

[Gambar 4. 41 *HTTP Request* untuk Mengikuti Pemilihan 94](#_Toc93349682)

[Gambar 4. 42 *HTTP Response* Mengikuti Pemilihan 95](#_Toc93349683)

[Gambar 4. 43 *HTTP Request* Daftar Peserta Pemilihan 95](#_Toc93349684)

[Gambar 4. 44 *HTTP Response* Daftar Peserta Pemilihan 96](#_Toc93349685)

[Gambar 4. 45 *HTTP Request* Detail Pemilihan 96](#_Toc93349686)

[Gambar 4. 46 *HTTP Response* Detail Pemilihan 97](#_Toc93349687)

[Gambar 4. 47 *HTTP Request* Memulai Pemilihan 98](#_Toc93349688)

[Gambar 4. 48 *HTTP Response* Memulai Pemilihan 98](#_Toc93349689)

[Gambar 4. 49 *HTTP Request* Pemilihan Kandidat 99](#_Toc93349690)

[Gambar 4. 50 *HTTP Response* Pemilihan Kandidat 99](#_Toc93349691)

[Gambar 4. 51 *HTTP Request* Menghentikan Pemilihan 100](#_Toc93349692)

[Gambar 4. 52 *HTTP Response* Menghentikan Pemilihan 100](#_Toc93349693)

[Gambar 4. 53 *HTTP Request* Hasil Pemilihan 100](#_Toc93349694)

[Gambar 4. 54 *HTTP Response* Hasil Pemilihan 101](#_Toc93349695)

[Gambar 4. 55 Struktur *File* Pengujian 102](#_Toc93349696)

[Gambar 4. 56 Hasil Pengujian *Endpoint Super Admin* 104](#_Toc93349697)

[Gambar 4. 57 Hasil Pengujian *Endpoint Election Authority* 106](#_Toc93349698)

[Gambar 4. 58 Hasil Pengujian *Endpoint Voter* 108](#_Toc93349699)

[Gambar 4. 59 Tampilan 111](#_Toc93349700)

[Gambar 4. 60 *Rlpdump Help* 111](#_Toc93349701)

[Gambar 4. 61 *Ldb Help* 112](#_Toc93349702)

[Gambar 4. 62 Tampilan Awal 113](#_Toc93349703)

[Gambar 4. 63 *Deployed Smart Contract* 113](#_Toc93349704)

[Gambar 4. 64 *Connected Smart Contract* 114](#_Toc93349705)

[Gambar 4. 65 Tombol Get Person 114](#_Toc93349706)

[Gambar 4. 66 Tombol Set Person 115](#_Toc93349707)

[Gambar 4. 67 Tombol Get Person 115](#_Toc93349708)

[Gambar 4. 68 Tombol Set Person 116](#_Toc93349709)

[Gambar 4. 69 *Block Explorer* 117](#_Toc93349710)

[Gambar 4. 70 Detail *Block* 2 117](#_Toc93349711)

[Gambar 4. 71 Ethereum *Decoder* 118](#_Toc93349712)

[Gambar 4. 72 *Param Generator* 118](#_Toc93349713)

[Gambar 4. 73 Ethereum *Decoder* 119](#_Toc93349714)

[Gambar 4. 74 *RocksDB Administrative Tools* 119](#_Toc93349715)

[Gambar 4. 75 Struktur Data *RLP* 120](#_Toc93349716)

[Gambar 4. 76 *Encoding RLP* 120](#_Toc93349717)

[Gambar 4. 77 *Update Database RocksDB* 121](#_Toc93349718)

# DAFTAR KODE PROGRAM

[Kode Program 3. 1 Spesifikasi *Request* Pembuatan Pemilihan 53](#_Toc93349719)

[Kode Program 3. 2 Spesifikasi Response Pembuatan Pemilihan 54](#_Toc93349720)

[Kode Program 3. 3 Spesifikasi Request Penambahan Kandidat 54](#_Toc93349721)

[Kode Program 3. 4 Spesifikasi *Response* Penambahan Kandidat 55](#_Toc93349722)

[Kode Program 3. 5 Spesifikasi *Request Deployment* Pemilihan 55](#_Toc93349723)

[Kode Program 3. 6 Spesifikasi *Response Deployment* Pemilihan 55](#_Toc93349724)

[Kode Program 3. 7 Spesifikasi *Request* Peserta Pemilihan 55](#_Toc93349725)

[Kode Program 3. 8 Spesifikasi *Resposne* Peserta Pemilihan 56](#_Toc93349726)

[Kode Program 3. 9 Spesifikasi *Request* *Detail* Pemilihan 56](#_Toc93349727)

[Kode Program 3. 10 Spesifikasi *Response* *Detail* Pemilihan 57](#_Toc93349728)

[Kode Program 3. 11 Spesifikasi *Request* Hasil Pemilihan 57](#_Toc93349729)

[Kode Program 3. 12 Spesifikasi *Response* Hasil Pemilihan 58](#_Toc93349730)

[Kode Program 3. 13 Spesifikasi *Request* Memulai Pemilihan 58](#_Toc93349731)

[Kode Program 3. 14 Spesifikasi *Response* Memulai Pemilihan 58](#_Toc93349732)

[Kode Program 3. 15 Spesifikasi *Request* Mengikuti Pemilihan 58](#_Toc93349733)

[Kode Program 3. 16 Spesifikasi *Response* Mengikuti Pemilihan 59](#_Toc93349734)

[Kode Program 3. 17 Spesifikasi *Request* untuk Memilih Kandidat 59](#_Toc93349735)

[Kode Program 3. 18 Spesifikasi *Response* untuk Memilih Kandidat 59](#_Toc93349736)

[Kode Program 3. 19 Spesifikasi *Request* untuk Menghentikan Pemilihan 59](#_Toc93349737)

[Kode Program 3. 20 Spesifikasi *Response* untuk Menghentikan Pemilihan 60](#_Toc93349738)

[Kode Program 4. 1 *File* simvoni-spec.json 74](#_Toc93349739)

[Kode Program 4. 2 *File* simvoni-config.toml 75](#_Toc93349740)

[Kode Program 4. 3 Perintah Menjalankan Openethereum 76](#_Toc93349741)

[Kode Program 4. 4 CURL untuk Membuat *Validator Account* 76](#_Toc93349742)

[Kode Program 4. 5 CURL untuk Membuat *Validator Account* 76](#_Toc93349743)

[Kode Program 4. 6 Menyimpan *Password Validator* 76](#_Toc93349744)

[Kode Program 4. 7 Mendapatkan Enode 77](#_Toc93349745)

[Kode Program 4. 8 Menambahkan Enode 77](#_Toc93349746)

[Kode Program 4. 9 *Smart Contract* 110](#_Toc93349747)

# DAFTAR TABEL

[Tabel 2. 1 MySQL *Function* 18](#_Toc93349751)

[Tabel 3. 1 Kebutuhan Perangkat Keras Implementasi Aplikasi 26](#_Toc93349752)

[Tabel 3. 2 Struktur Data Tabel ref\_user 44](#_Toc93349753)

[Tabel 3. 3 Struktur Data Tabel ref\_user\_role 44](#_Toc93349754)

[Tabel 3. 4 Struktur Data Tabel ta\_candidate 45](#_Toc93349755)

[Tabel 3. 5 Struktur Data Tabel ta\_misi 45](#_Toc93349756)

[Tabel 3. 6 Struktur Data Tabel ta\_pengalaman 46](#_Toc93349757)

[Tabel 3. 7 Struktur Data Tabel ta\_election 46](#_Toc93349758)

[Tabel 3. 8 Struktur Data Tabel ref\_election\_status 47](#_Toc93349759)

[Tabel 4. 1 Skenario Pengujian *Endpoint* *Super Admin* 103](#_Toc93349760)

[Tabel 4. 2 Skenario Pengujian *Endpoint Election Authority* 105](#_Toc93349761)

[Tabel 4. 3 Skenario Pengujian *Endpoint Voter* 107](#_Toc93349762)

# BAB I PENDAHULUAN

Bab I berisi bahasan mengenai pendahuluan dari penelitian ini. Pendahuluan dijabarkan menjadi beberapa bagian, diantaranya latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika penulisan.

## Latar Belakang

Pengambilan keputusan bersama memiliki beberapa cara dalam pelaksanaannya, diantaranya musyawarah mufakat, voting dan aklamasi. Masing-masing metode pengambilan keputusan memiliki kelebihan dan kekurangan. Metode pengambilan keputusan yang paling sering digunakan adalah voting karena keputusan dapat dihasilkan dalam jangka waktu yang relatif cepat. Kegiatan yang paling sering menggunakan metode voting adalah pemilihan kepala daerah karena voting sesuai dengan prinsip demokrasi yang memberikan kesempatan setiap kalangan untuk memberikan hak suara. 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 (Muhammad Habibi, 2018). Model *e-voting internet poll site voting* dan *kiosk voting* dirasa kurang optimal karena walaupun sudah menggunakan teknologi internet, pemilih masih belum bisa menggunakan perangkat yang dimiliki sehingga tetap harus datang ke tempat pemilihan untuk melakukan voting. Saat ini, era dimana kecepatan dan kemudahan akses informasi menjadi prioritas, internet voting menjadi model *e-voting* yang paling optimal untuk dikembangkan.

Beberapa negara yang sudah mengembangkan *e-voting* model internet voting diantaranya Australia, Kanada, Jepang, Jerman dan Inggris (Muhammad Habibi, 2018). *E-voting* berjalan dengan lancar di Australia, Kanada dan Jepang namun gagal di Jerman dan Inggris. *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.

## Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya, maka diperoleh beberapa rumusan masalah sebagai berikut.

1. Bagaimana proses kerja Ethereum *smart contract* pada Sistem Voting Elektronik*?*
2. Bagaimana perancangan dan proses pembangunan Sistem Voting Elektronik berbasisEthereum *smart contract*?
3. Bagaimana hasil penelitian dan kinerja Sistem Voting Elektronik berbasisEthereum *smart contract*?

## Tujuan

Berdasarkan rumusan masalah yang telah dipaparkan sebelumnya, maka diperoleh tujuan sebagai berikut.

1. Memahami proses kerja Sistem Voting Elektronik berbasisEthereum *smart contract*.
2. Mengetahui perancangan dan proses pembangunan Sistem Voting Elektronik berbasisEthereum *smart contract*.
3. Mengetahui hasil penelitian dan kinerja Sistem Voting Elektronik berbasisEthereum *smart contract*.

## Manfaat Penelitian

Penelitian ini dalam pelaksanaannya memiliki beberapa manfaat yang dijabarkan sebagai berikut.

1. Menghilangkan *trust issue* dalam pelaksanaan voting.
2. Meningkatkan keamanan data hasil voting.
3. Menghemat waktu dalam proses pengambilan data voting.
4. Meminimalisir biaya yang perlu dikeluarkan dalam proses pengambilan data voting.
5. Mempermudah proses pengambilan data voting karena dapat dilakukan dari mana saja melalui perangkat pengguna yang terkoneksi internet.

## Batasan Masalah

Penelitian ini dalam pelaksanaannya memiliki beberapa batasan yang bertujuan agar penggunaan aplikasi tidak terlalu luas. Batasan masalah yang dijabarkan adalah sebagai berikut.

1. Penelitian yang dilakukan hanya berfokus pada Sistem Voting Elektronik.
2. *Platform blockchain* yang digunakan adalah Ethereum*.*

## Sistematika Penulisan

Sistematika penulisan merupakan bagian yang menjelaskan gambaran laporan dari semua dasar teori dan metode yang digunakan serta hasil yang diperoleh selama pengerjaan tugas akhir. Laporan tugas akhir ini dibagi menjadi lima bab sebagai berikut.

BAB I : **Pendahuluan**

Bagian ini membahas mengenai latar belakang, perumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika penulisan laporan tugas akhir sistem *e-voting* berbasis Ethereum *smart contract*.

BAB II : **Tinjauan Pustaka**

Bagian ini menjelaskan tentang dasar teori, pengetahuan, dan referensi yang digunakan untuk memperkuat projek tugas akhir sistem *e-voting* berbasis Ethereum *smart contract*.

BAB III : **Metodologi dan Perancangan Sistem**

Bagian ini memuat metode yang digunakan dalam penelitian tugas akhir sistem *e-voting* berbasis Ethereum *smart contract* beserta dengan proses pembuatan laporan yang meliputi tempat dan waktu penelitian, sumber data dan metode pengumpulan, instrumen perancangan, algoritma pemrograman, hingga perancangan sistem dimulai dari rancangan prosedural hingga antarmuka (*interface*) sistem.

BAB IV : **Pengujian dan Analisa Hasil**

Bagian ini memuat rancangan yang telah dibuat, implentasi dari perangkat yang digunakan serta pengujian sistem dari sistem *e-voting* berbasis Ethereum *smart contract.*

BAB V : **Penutup**

Bagian ini memuat kesimpulan yang mengacu pada rumusan masalah dan tujuan penelitian beserta saran yang diberikan peneliti untuk kepentingan pengembangan aplikasi selanjutnya.

# BAB II TINJAUAN PUSTAKA

Bab II berisi bahasan mengenai teori atau materi pendukung yang digunakan sebagai acuan atau dasar teori dalam sistem *e-voting* berbasis Ethereum *smart contract*.

## *State of the Art*

Perancangan untuk pengembangan Sistem Voting Elektronik memiliki persamaan dan atau keterikatan dari beberapa penelitian yang dilakukan beberapa diantaranya adalah sebagai berikut.

Penelitian pertama yang digunakan sebagai acuan dalam penelitian ini 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, et al., 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 & Harjono, 2014).

Perancangan sistem votingelektronik 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 & 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 terhadap 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, et al., 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)*.*

## Blockchain

*Blockchain* adalah teknologi pencatatan transaksi yang menghilangkan pihak ketiga atau *central authority* dalam melakukan pertukaran data atau 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).

|  |
| --- |
|  |

Gambar 2. 1 Arsitektur *Blockchain*

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

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.*

### Ethereum Accounts dan Transaction

*Accounts* adalah entitas yang terdapat dalam Ethereum yang dapat membuat *Transaction* dimana setiap *account* memiliki *address* yang panjangnya 20-*bytes*. Sebuah Etherum *account* memiliki empat *field* yaitu *nonce, ether balance, contract code* dan *storage*. *Nonce* adalah *counter* yang digunakan untuk memastikan bahwa setiap transaksi yang dibuat oleh sebuah *account* hanya diproses sekali. Ether atau (ETH) adalah mata uang digital yang digunakan untuk membayar biaya transaksi. Secara umum ada dua jenis account yang terdapat di Ethereum. Yang pertama adalah *Externally Owned Account (EOA)* yang dikontrol menggunakan *private key* dan yang kedua adalah *contract account* yang dikontrol oleh *contract code*-nya. *Contract account* bisa dikatakan sebagai *autonomous agents* yang hidup didalam Ethereum *execution environment* yang selalu menjalankan perintah spesifik dari kode ketika menerima transaksi.

Istilah transaksi yang dimaksud dalam Ethereum adalah data yang telah ditandatangani secara digital oleh sebuah *EOA*. Sebuah transaksi berisi *address* penerima, *signature* untuk identifikasi pengirim, jumlah *ether* yang akan dikirim, *data field, gas* dan *gas price*. *Gas* adalah nilai yang merepresentasikan seberapa besar transaksi tersebut boleh menggunakan *resource* komputasi.

### Ethereum Code Execution

Kode dalam kontrak Ethereum ditulis dalam sebuah bahasa *bytecode low-level* berbasis *stack* yang disebut sebagai Ethereum Virtual Machine Code atau EVM Code. Kode terdiri dari beberapa kumpulan *byte* dimana setiap kumpulan *byte* merepresentasikan sebuah operasi. Secara umum, *code execution* ada sebuah perulangan tak hingga yang melakukan *increment* penambahan satu yang dimulai dari nol hingga program mencapai *error* atau terdeteksi intruksi *return*. Setiap operasi memiliki akses ke tiga jenis penyimpanan data yaitu *stack, memory* dan *storage*. *Stack* adalah sebuah *contrainer last-in-first-out* dimana nilainya bisa di *pushed* atau *pop*. *Memory* adalah *expandable byte array* yang akan hilang ketika komputasi selesai dilakukan. *Storage* adalah penyimpanan persistent yang menyimpan data menggunakan model *key/value*.

### Openethereum

Jaringan suatu *blockchain* tercipta dari jaringan komunikasi *peer-to-peer* dari beberapa *nodes. Nodes* adalah suatu program komputer yang berjalan berdasarkan spesifikasi suatu *blockchain*. Hal yang sama juga berlaku pada jaringan Ethereum. *Nodes* pada jaringan Ethereum adalah program yang berjalan berdasarkan spesifikasi Ethereum. Ada beberapa Ethereum *client* yang ada saat ini yaitu Geth, Openethereum, Nethermind dan Hyperledger Besu. Ethereum *client* tersebut menggunakan bahasa pemrograman yang berbeda namun tetap bisa saling berkomunikasi karena mengimplementasikan spesifikasi yang sama.

Openethereum adalah sebuah Ethereum *client* yang dibuat menggunakan bahasa pemrograman Rust. Keunggulan yang dimiliki oleh Openethereum adalah program yang ringan, kecepatan sinkronisasi dan dokumentasi yang jelas tentang bagaimana cara mengoptimalisasikan Openethereum untuk skenario penggunaan yang berbeda.

### Smart Contract

*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

Secara umum pengertian konsensus adalah sebuah kesepakatan bersama yang telah dicapai oleh suatu kelompok terhadap suatu masalah. Konsensus dalam *blockchain* sangat penting karena data yang terdesentralisasi sehingga sangat memungkinkan terjadinya perbedaan daya yang tercatat. 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.

### Gas, Gas Fee dan Gas Limit

*Gas* adalah sebuah kalkulasi untuk menentukan seberapa besar biaya yang dibutuhkan untuk melakukan tindakan pada jaringan *blockchain*. 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. Gas limit* adalah batas maksimal *gas* yang boleh digunakan untuk melakukan sebuah transaksi. *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.

### Protokol Komunikasi

Jaringan Ethereum adalah jaringan yang terdiri dari banyak *node* yang dikelola oleh pihak-pihak independen sehingga sangat mustahil untuk memelihara jaringan secara manual oleh sebuah pihak sentral karena *uptime* dari *node-node* yang terhubung berbeda-beda. Untuk memelihara jaringan tetap berjalan tanpa diganggu oleh *uptime node* yang berbeda-beda sehingga diperlukan sebuah *dicovery protocol* untuk melakukan *lookup* terhadap *node* yang masih aktif pada jaringan. *Discovery protocol* yang digunakan oleh Ethereum dibuat seperti desain jaringan *peer-to-peer* yang didesain oleh Petar Maymounkov and David Mazières pada tahun 2002 yang diberi nama *Kademlia*. Secara sederhana *Kademlia* adalah *Distributed Hash Table (DHT)* yang menyimpan informasi alamat *node* yang terhubung pada suatu *node* sehingga masing-masing *node* pada jaringan bisa melakukan *lookup* secara berkala. *Discovery protocol* yang digunakan oleh Ethereum berjalan pada protokol *UDP*.

RLPx adalah protokol komunikasi berbasis *TCP* yang digunakan oleh Ethereum *node* untuk bertukar data. Protokol ini mewajibkan setiap *node* yang akan melakukan pertukaran data untuk melakukan serangkaian proses enkripsi. Dalam jaringan Ethereum, *request* dan *response* memiliki waktu timeout sebesar 500 milidetik sedangakan operasi untuk protokol RLPx memiliki timeout sebesar 5 detik. Tabel yang menyimpan data peer diupdate secara berkala dalam waktu 30 detik.

|  |
| --- |
|  |

Gambar 2. Diagram Komunikasi Node Ethereum

Gambar 2.2 menjelaskan bahwa untuk memeriksa apakah sebuah *node* masih aktif atau tidak dengan mengirimkan pesan "ping". Jika dibalas dengan "pong" maka proses pengiriman data akan dilakukan yang diawali dengan pertukaran kunci ECIES. Sedangkan jika tidak ada balasan maka alamat *peer* akan dihapus dari DHT. Proses sinkronisasi terjadi dengan cara mengirim dan menerima data *block headers*, *block bodies* dan *receipts* secara berkala menggunakan protokol RLPx.

## VSCode

Visual Studio Code adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung bahasa pemrograman JavaScript, Typescript, dan Node.js, serta bahasa pemrograman lainnya dengan bantuan *plugin* yang dapat dipasang melalui *marketplace* Visual Studio Code (seperti C++, C#, Python, Go, Java, dst).

Fitur-fitur yang disediakan oleh Visual Studio Code, diantaranya *Intellisense, Git Integration, Debugging,* dan fitur ekstensi yang menambah kemampuan teks editor. *Editor* ini dirancang untuk pengembang yang bekerja dengan teknologi *open source* *cloud* serta menggunakan NET untuk memberikan dukungan untuk ASP. Antarmukanya tergolong mudah untuk digunakan karena didasarkan pada gaya *explorer* umum, dengan *panel* di sebelah kiri yang menunjukkan semua *file* dan *folder* serta *panel editor* di sebelah kanan, yang menunjukkan isi dari *file* yang telah dibuka.

|  |
| --- |
| https://code.visualstudio.com/opengraphimg/opengraph-home.png |

Gambar 2. 3 Visual Studio Code

Microsoft telah menyediakan dokumentasi yang lengkap untuk membantu dan memudahkan penggunaan bagi para *developer*. Visual Studio Code ditargetkan pada pengembang scripting *server-side* dan dapat digunakan oleh siapa saja untuk membangun aplikasi berbasis *web* (Code, 2021)*.*

## NodeJS

Node.js merupakan salah satu *platform* pengembang yang dapat digunakan untuk membuat aplikasi berbasis *Cloud*. Node.js dikembangkan dari *engine* JavaScript yang dibuat oleh Google untuk *browser* Chrome ditambah dengan *libuv* serta beberapa pustaka lainnya. Node.js menggunakan JavaScript sebagai bahasa pemrograman dan *event-driven, non-blocking I/O (asynchronous)* model yang membuatnya ringan dan efisien. Node.js memiliki fitur built-in HTTP *server library* yang menjadikannya mampu menjadi sebuah *web server* tanpa bantuan *software* lainnya seperti Apache dan Nginx.

|  |
| --- |
| https://miro.medium.com/max/724/1*ODU5V_oAmYmzvZ1wIw3rDw.png |

Gambar 2. 4 Node.js

Sumber : https://ichi.pro/id

Node.js adalah sebuah *runtime environment* dan *script library*. Sebuah *runtime environment* adalah sebuah *software* yang berfungsi untuk mengeksekusi, menjalankan dan mengimplementasikan fungsi-fungsi serta cara kerja inti dari suatu bahasa pemrograman. Sedangkan *script library* adalah kumpulan, kompilasi atau bank data berisi skrip/kode-kode pemrograman. Node.js dibangun menggunakan JavaScript dan C++, terdapat arsitektur serta fungsi dari Google V8 di dalamnya yang berfungsi sebagai *compiler* ditulis dalam C++ dan library Libuv bekerja untuk menangani operasi *asynchronous I/O* dan *main event loop* (Wilson, 2013).

### NestJS

Nest (NestJS) adalah *framework* untuk membangun *server-side applications* Node.js yang efisien. NestJS menggunakan JavaScript progresif yang dibangun dengan mendukung TypeScript secara penuh dan menggabungkan elemen *OOP (Object Oriented Programming), FP* (*Functional Programming*), dan *FRP* (*Functional Reactive Programming*). Nest menggunakan kerangka kerja *Server HTTP default* Express dan secara opsional dapat dikonfigurasi untuk menggunakan Fastify juga. Nest menyediakan level abstraksi di atas *framework* Node.js (Express / Fastify), tetapi juga mengekspos *API*-nya secara langsung ke *developer* sehingga memberi kebebasan bagi *developer* untuk menggunakan banyak sekali modul pihak ketiga yang tersedia untuk *platform* yang mendasarinya.

|  |
| --- |
| https://www.digitalkode.com/images/uploads/1_czxaov35etfe545eiugfqq.png |

Gambar 2. 5 NestJS

Sumber: https://www.digitalkode.com/

NestJS memiliki beberapa kelebihan atau fitur unggulan, diantaranya memberikan fleksibilitas dengan memungkinkan untuk penggunaan pustaka lain dengan adanya arsitektur modular *(extensible),* juga merupakan ekosistem yang dapat beradaptasi untuk semua jenis *server-side applications* serta bersifat progresif karena memanfaatkan fitur JavaScript yang terbaru (Mysliwiec, 2017).

### Angular

Angular adalah *framework open-source single-page apps* berbasis *TypeScript*. Angular dirancang dari penulisan ulang Angular JS oleh Google. Angular dapat dikembangkan di seluruh *platform* baik *web, mobile web, native mobile,* maupun *native desktop*. Angular melakukan proses *load* dengan cepat melalui *component router* baru, yang memberikan *code-splitting* otomatis sehingga pengguna hanya perlu melakukan *load* kode yang diperlukan untuk me-*render* tampilan yang diminta. Angular dirancang untuk membuat sebuah pembaruan semudah mungkin, sehingga *developer* dapat memanfaatkan perkembangan terbaru dengan *effort* yang minimal.

|  |
| --- |
| https://miro.medium.com/max/700/1*1LQEcTLv8rBn3SixcRcbxw.png |

Gambar 2. 6 Angular

Angular menyediakan beberapa *library* pihak pertama*. Library-library* tersebut berfungsi untuk memperluas fungsionalitas aplikasi sekaligus berfokus pada fitur yang membuat aplikasi yang dikembangkan menjadi lebih unik. *Library* tersebut dirancang untuk diintegrasikan dengan lancar dan diperbarui secara bersamaan dengan kerangka kerja Angular. *Library-library* yang disediakan, antara lain (Angular, 2010) :

1. *Angular Router*, navigasi dan *routing client-side* lanjutan berdasarkan komponen Angular. Mendukung *lazy-loading, nested routes, custom path matching* dan lainnya,
2. *Angular Forms*, sistem yang seragam untuk partisipasi dan validasi *form*.
3. *Angular HttpClient*, HTTP *client* yang kuat yang dapat mendukung komunikasi client -server yang lebih baik.
4. *Angular Animations*, sistem yang kaya untuk menggerakkan animasi berdasarkan status aplikasi.
5. *Angular PWA*, tools untuk membangun *Progressive Web Applications* (*PWA*) termasuk *service worker* dan *web app manifest*.
6. *Angular Schematics*, *automated scaffolding*, *refactoring*, dan *update tools* yang menyederhanakan pengembangan dalam skala besar.

### Web3JS

Web3js memungkinkan untuk berinteraksi dengan *node* Ethereum lokal atau jarak jauh, menggunakan koneksi *HTTP* atau *IPC*. Web3js merupakan pustaka yang memungkinkan untuk melakukan tindakan pengiriman seperti mengirim *ether* dari akun satu ke akun lain, membaca dan menulis *smart contract* dan lain sebagainya.

|  |
| --- |
|  |

Gambar 2. 7 Interaksi Web3.JS dengan Node Ethereum

Web3.js dapat digunakan untuk terhubung ke jaringan Ethereum melalui *node* Ethereum yang memungkinkan akses melalui HTTP. Akses tersebut dapat berupa *node* lokal, *node* yang di-*hosting* oleh penyedia DApp, atau *gateway* publik seperti Infura, yang mengoperasikan titik akses Ethereum gratis. Salah satu cara umum untuk mengintegrasikan aplikasi *browser web* dengan Ethereum adalah dengan menggunakan ekstensi *browser* Metamask dengan kombinasi Web3js (Web3JS, 2016).

## MySQL

MySQL merupakan *software* yang tergolong sebagai *DBMS* (*Database Management System*) yang bersifat *open source*. *Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* selain bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi. MySQL memiliki fungsi-fungsi MySQL yang digunakan untuk mengakses *database server* MySQL. Fungsi ini berguna untuk mengantarkan perintah SQL pada PHP menuju ke *server* sehingga perintah tersebut dapat dieksekusi oleh semua *server* MySQL.

Tabel 2. 1 MySQL *Function*

| **No.** | **Function** | **Kegunaan** |
| --- | --- | --- |
|  | *Mysql\_connect()* | Membuat hubungan ke *database* MySQL yang terdapat pada suatu *host* |
|  | *Mysql\_close()* | Menutup hubungan ke *database MySQL* |
|  | *Mysql\_select\_db()* | Memilih *database* |
|  | *Mysql\_query()* | Mengeksekusi permintaan terhadap sebuah tabel atau sejumlah tabel |
|  | *Mysql\_db\_query()* | Menjalankan suatu permintaan terhadap suatu *database* |
|  | *Mysql\_num\_rows()* | Memperoleh jumlah baris dari suatu hasil permintaan yang menggunakan *SELECT* |
|  | *Mysql\_affected\_rows()* | Memperoleh jumlah baris yang dikenai operasi *INSERT, DELETE, UPDATE* |
|  | *Mysql\_num\_fields()* | Memperoleh jumlah kolom pada suatu hasil permintaan |
|  | *Mysql\_fecth\_row()* | Menghasilkan *array*/baris yang berisi seluruh kolom dari sebuah baris pada suatu himpunan hasil |
|  | *Mysql\_fecth\_array()* | Menghasilkan *array*/baris yang berisi seluruh kolom dari sebuah baris pada suatu himpunan hasil yang akan disimpan dua kali pada *array* hasil |
|  | *Mysql\_fecth\_field()* | Menghasilkan informasi suatu kolom |
|  | *Mysql\_data\_seek()* | Memindahkan pointer pada suatu himpunan hasil supaya menunjuk ke baris tertentu |
|  | *Mysql\_field\_seek()* | Memindahkan pointer pada suatu himpunan hasil supaya menunjuk ke kolom tertentu |
|  | *Mysql\_create\_db()* | Membuat *database MySQL* |
|  | *Mysql\_drop\_db()* | Menghapus *database MySQL* |
|  | *Mysql\_list\_dbs()* | Menghasilkan daftar *database MySQL* |
|  | *Mysql\_list\_tables()* | Memperoleh daftar nama tabel dalam suatu *database* |
|  | *Mysql\_list\_fields()* | Memperoleh daftar nama kolom dalam suatu *database* |
|  | *Mysql\_fetch\_assoc()* | Mendapatkan array baris dari suatu *recordset* |
|  | *Mysql\_fetch\_lengths()* | Mendapatkan panjang baris pada setiap isi *field* |
|  | *Mysql\_fetch\_object()* | Menghasilkan baris dari *recordset* sebagai sebuah objek |
|  | *Mysql\_field\_len()* | Mendapatkan informasi panjang maksimum *field* dalam sebuah *recorset* |
|  | *Mysql\_field\_name()* | Mendapatkan informasi nama *field* dalam *recordset* |
|  | *Mysql\_ping()* | Memeriksa koneksi *server* dan akan mencoba untuk melakukan koneksi ulang jika koneksi terputus |

Tabel 2.1 merupakan fungsi-fungsi dalam MySQL yang dapat digunakan untuk mengantarkan perintah SQL pada PHP menuju ke *server* sehingga perintah tersebut dapat dieksekusi oleh semua *server* MySQL.

Selain fungsi-fungsinya, MySQL juga memiliki beberapa keunggulan, seperti dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lainnya. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis serta dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik. MySQL memiliki kecepatan yang tinggi dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain. MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi *API* (*Application Programming Interface*) (Kadir, 2008).

## JSON

*JSON (JavaScript Object Notation)* adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 Desember 1999. *JSON* merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dan lainnya (Derizal, 2011).

## Google Cloud Platform

Google Cloud Platform merupakan layanan *public cloud computing* dari Google yang terdiri dari beragam layanan. *Platform* dari Google ini menyediakan beragam layanan *hosting* mulai dari komputasi, *storage* dan *aplication development* yang berjalan pada *hardware* Google. Google Cloud Platform Service dapat diakses oleh pengembang *software, administrator cloud* dan profesional IT lainnya menggunakan internet publik atau melalui koneksi jaringan *dedicated* (Google, 2021).

### Compute Engine

Google Compute Engine memungkinkan untuk menjalankan beban kerja komputasi skala besar dengan infrastruktur yang sama yang menjalankan Google Search, Gmail dan Ads. Pengguna dapat meluncurkan *virtual machines* sesuai permintaan, mengelola konektivitas jaringan menggunakan solusi jaringan yang sederhana namun fleksibel, dan mengakses berbagai alternatif penyimpanan data dari *virtual machines*. Fitur-fitur Google Compute Engine antara lain (Google, 2021) :

1. *Full Virtual Machines:* Mesin virtual yang di-*hosting* kernel yang menjalankan Ubuntu atau CentOs. Meluncurkan 1, 2, 4, atau 8 *virtual core instances* dengan memori 3.75 GB per *virtual core*.
2. *Flexible Storage:* menawarkan opsi penyimpanan yang berbeda untuk memenuhi berbagai kebutuhan.
3. *Flexible networking:* menawarkan solusi jaringan yang memudahkan untuk menghubungkan *virtual machines* satu sama lain dan ke Internet.
4. *Open tooling:* menawarkan *UI* sederhana dan *command-line tooling* untuk mengonfigurasi dan meluncurkan *virtual machines*.

## Agile Software Development Method

Metode *Agile* pertama kali diperkenalkan oleh Kent Beck bersama dengan 16 rekannya. 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).

|  |
| --- |
| https://pelajarindo.com/wp-content/uploads/2018/08/Pengertian-Metode-Agile.png |

Gambar 2. 8 *Agile Software Development Method*

Gambar 2.7 *Agile* sendiri merupakan gabungan dari metode *incremental* dan *iterative*. Dibandingkan dengan metode *waterfall*, *agile* tidak memiliki fase-fase yang paten. Metode *agile* berproses iteratif dengan perputaran yang pendek. Kebutuhan direncananakan, diimplementasikan, diuji, dan dievaluasi secara berulang.

Metode *agile* masih terbagi menjadi beberapa jenis bagian, diantaranya *Adaptive Software Development (ASD), Agile Modelling (AM), Crystal, Dynamic System Development Method (DSDM), Extreme Programming (XP), Feature Driven Development (FDD),* *Rational Unified Process*, dan *Scrum Methodology*. Jenis *agile* yang paling sering digunakan adalah *Adaptive Software Development* *(ASD), Dynamic System Development Method (DSDM), Extreme Programming (XP),* dan *Scrum* (Pressman, 2010).

Penelitian ini akan menggunakan jenis metode *Scrum*. Walaupun sebenarnya metode ini biasanya digunakan secara tim, namun secara khusus tidak ada larangan untuk menggunakannya secara perorangan. Metode *scrum* membagi proses *development* menjadi beberapa *sprint*. *Sprint* ini akan bertindak sebagai *milestone* dalam pembuatan *software.* Berikut ini merupakan beberapa tahapan dalam *Scrum.*

### 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*.

### Sprints

Tahapan selanjutnya yaitu 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.

### Scrum Meeting

Tahap ini dilakukan dengan menganalisis dari sisi pengembang untuk mengetahui seberapa banyak kemajuan dari kegiatan pengembangan sistem yang telah dilakukan atau dicapai.

### Demos

Langkah selanjutnya adalah memperlihatkan fitur-fitur *software* yang telah dihasilkan dalam proses pengembangan untuk dievaluasi oleh pengguna sesuai dengan waktu yang telah ditentukan.

Sementara itu, di dalam setiap iterasi kegiatan pengembangan sistem yang telah dijelaskan sebelumnya, terdapat beberapa rangkaian kegiatan di dalamnya, yaitu sebagai berikut.

1. Analisa Kebutuhan Sistem, yaitu Asesmen terhadap pengguna untuk menggali secara detail *software requirement* yang dibutuhkan oleh pengguna.
2. Desain, yaitu tahap perancangan sistem, mulai dari desain arsitektur sistem, desain proses bisnis, desain *database*, hingga desain *user interface*.
3. *Code Generation*, yaitu tahappenulisan kode program dan manajemen *database*
4. *Testing*, yaitu proses meminimalisir kesalahan (*error*) dan memastikan *output* yang dihasilkan telah sesuai dengan *user requirement.*
5. *Support*, yaitu proses menindaklanjuti perubahan yang diperlukan setelah proses *testing* dan evaluasi sesuai dengan kebutuhan pengguna.

# BAB III METODOLOGI DAN PERANCANGAN SISTEM

BAB III ini memuat metode yang digunakan dalam penelitian beserta dengan proses pembuatan laporan yang meliputi tempat dan waktu penelitian, sumber data dan metode pengumpulan, instrumen perancangan, dan gambaran umum sistem.

## Tempat dan Waktu Penelitian

Penelitian ini bertempat di Kampus Bukit Universitas Udayana, Fakultas Teknik, Program Studi Teknologi Informasi yang berlokasi di Jalan Kampus Bukit UNUD Jimbaran*,* Kabupaten Badung, Bali. Kampus Bukit Universitas Udayana dipilih karena memiliki semua aspek pendukung agar penelitian dapat berjalan dengan baik. Penelitian dilaksanakan pada semester genap tahun ajaran 2019-2020 sampai dengan semester genap tahun ajaran 2021-2022 bulan Januari.

## Metode Penelitian

Metode penelitian yang digunakan pada pembuatan sistem *e-voting* ini adalah *Agile Software Development Methods.* Seperti yang telah dijelaskan pada pembahasan sebelumnya, *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.* Berikut ini merupakan implementasi penelitian ini berdasarkan beberapa tahapan dalam *Scrum.*

### Penerapan Tahap Backlog

Tahap ini dilakukan dengan proses menyusun rincian prioritas pada fitur-fitur yang akan dikembangkan pada sistem *e-voting*. 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.

### Penerapan Tahap Sprints

Proses ini menyusun kegiatan dalam *backlog* dengan durasi realisasi selama hari kerja yang ditentukan yaitu 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,* di awal pengembangan masih belum mendapat prioritas utama, namun akan menjadi fokus setelah proses-proses lainnya mencapai hasil dengan persentase yang sudah lebih besar.

### Penerapan Tahap Scrum Meeting

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*.*

### Penerapan Tahap Demos

Tahapan ini dalam pengembangan sistem *e-voting* dibantu oleh Dosen Pembimbing 1 dan Dosen Pembimbing 2 tugas akhir secara rutin sebagai pihak yang memberi masukan dan saran atas pengembangan yang telah dilakukan untuk kemudian dilakukan proses evaluasi kembali atas hasil kerja berdasarkan proses penerapan tahap-tahap diatas.

## Alat Penelitian Aplikasi

Alat penelitian adalah media pendukung untuk melakukan implementasi aplikasi dalam penelitian ini. Alat penelitian dibagi menjadi dua bagian yaitu perangkat lunak dan perangkat keras. Kebutuhan perangkat perangkat lunak dan perangkat keras untuk implementasi aplikasi dalam penelitian ini dijelaskan pada sub bab dibawah ini.

### Kebutuhan Perangkat Lunak Implementasi Aplikasi

Implementasi aplikasi Sistem Voting Elektronik memiliki kebutuhan perangkat lunak. Perangkat lunak yang dimaksud adalah 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. Kecepatan *ping* yang digunakan untuk implementasi Sistem Voting Elektronik adalah dibawah 500 ms.

### Kebutuhan Perangkat Keras Implementasi Aplikasi

Implementasi aplikasi menggunakan empat *virtual private server* yang disediakan oleh Google Cloud Platform. Spesifikasi dari *virtual private server* disebutkan dalam Tabel

Tabel 3. 1 Kebutuhan Perangkat Keras Implementasi Aplikasi

|  |  |
| --- | --- |
| **Hardware** | **Virtual Private Server** |
| **Processor** | 2 vCPU |
| **RAM** | 1 GB |
| **Penyimpanan** | 10 GB |

Tabel 3.1 menunjukkan spesifikasi dari perangkat keras yaitu *VPS* yang disediakan oleh Google Cloud. Perangkat keras tersebut digunakan untuk melakukan implementasi terhadap Sistem Voting Elektronik.

## Gambaran Umum Sistem

Sistem Voting Elektronik berbasis Ethereum *smart contract* merupakan sistem yang digunakan sebagai sarana pemilihan elektronik dengan menerapkan teknologi *blockchain*. Gambar 3.1 merupakan gambaran umum arsitektur sistem *e-voting* berbasis Ethereum *smart contract*. Gambaran umum arsitektur teknologi merupakan bagan yang menjelaskan alur kerja teknologi yang digunakan secara menyeluruh secara umum.

|  |
| --- |
|  |

Gambar 3. 1 Gambaran Umum Sistem

Gambar 3.1 merupakan gambaran umum dari Sistem Voting Elektronik berbasis Ethereum *smart contract*. 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 karena 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.

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.

### Use Case Diagram

*Use case diagram* merupakan representasi interaksi *user* dengan sistem dan menggambarkan spesifikasi dari *use case*. *Use case* untuk sistem *e-voting* berbasis Ethereum *smart contract* ditunjukkan oleh Gambar 3.2.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\usecase.png |

Gambar 3. 2 *Use Case Diagram*

Gambar 3.2 menunjukkan *use case* dari sistem *e-voting* berbasis Ethereum *smart contract*. *Use case* tersebut menggambarkan interaksi pengguna terhadap sistem *e-voting* dan interaksi antara satu elemen dengan elemen lainnya. *Use case* aplikasi sistem *e-voting* memiliki 3 aktor, 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.*

### Rancangan Alur Sistem

Rancangan alur penggunaan aplikasi sistem *e-voting* berbasis Ethereum *smart contract* digunakan sebagai acuan untuk membuat antarmuka dari aplikasi.

1. Rancangan Alur Sistem *(Super Admin)*

Aplikasi sistem *e-voting* memiliki 3 jenis *user* yang memiliki peran dan akses yang berbeda-beda. Jenis *user* yang pertama adalah *super admin*. Rancangan alur sistem dengan *super admin* sebagai *user* dapat dilihat pada Gambar 3.3.

|  |
| --- |
|  |

Gambar 3. 3 Rancangan Alur Sistem *(Super Admin)*

Gambar 3.3 menggambarkan rancangan alur keseluruhan sistem dengan *super admin* sebagai *user*. Pertama-tama, *super admin* harus *login* terlebih dahulu ke dalam sistem. Setelah berhasil *login, super admin* dapat melakukan pembuatan akun bagi *election authority.* Apabila terdapat *election* baru yang siap di­-*deploy,* maka *super admin* bertugas melakukan proses *deploy* ke *blockchain.*

1. Rancangan Alur Sistem *(Election Authority)*

Jenis *user* selanjutnya adalah *election authority,* yaitu *user* yang bertanggung jawab untuk membuat dan menangani satu atau lebih *election.* Rancangan alur sistem dengan *election authority* sebagai *user* dapat dilihat pada Gambar 3.4.

|  |
| --- |
| C:\Users\sarah\Downloads\Telegram Desktop\flowchart-election-authority.png |

Gambar 3. 4 Rancangan Alur Sistem *(Election Authority)*

Gambar 3.4 menggambarkan rancangan alur keseluruhan sistem dengan *election authority* sebagai *user.* Alur sistem diawali dengan proses *login* oleh *election authority* menggunakan akun yang telah didaftarkan atau dibuat oleh *super admin* sebelumnya.Setelah berhasil *login,* maka *election authority* dapat melakukan proses pembuatan *election* baru, diikuti dengan peambahan daftar kandidatnya. Setelah itu, apabila election yang dibuat telah di-*deploy* oleh *super admin,* maka *election authority* dapat memulai *election.* Namun, apabila *election* belum di-*deploy,* maka *election authority* harus menunggu sampai *election* berhasil di­-*deploy* oleh *super admin* untuk dapat memulai *election.*

1. Rancangan Alur Sistem *Voter*

Jenis *user* yang terakhir adalah *voter,* yaitu *user* yang memiliki hak suara dan berhak memilih dalam satu atau lebih *election.* Rancangan alur sistem dengan *voter* sebagai *user* dapat dilihat pada Gambar 3.5.

|  |
| --- |
|  |

Gambar 3. 5 Rancangan Alur Sistem *(Voter )*

Gambar 3.5 menggambarkan rancangan alur keseluruhan sistem dengan *voter* sebagai *user.* Langkah pertama yang harus dilakukan oleh *voter* adalah *login* ke dalam sistem. Setelah berhasil *login,* maka *voter* dapat mendaftar ke dalam *election* yang akan diikuti. Pendaftaran *voter* dalam *election* harus terlebih dahulu disetujui untuk selanjutnya dapat melihat daftar kandidat dalam *election,* namun apabila pendaftaran ditolak, maka *voter* harus melakukan pendaftaran ulang sampai pendaftaran diterima untuk dapat mengikuti *election* tersebut. Proses pemungutan suara atau *vote* hanya dapat dilakukan oleh *voter* apabila *election* telah dimulai.

1. Rancangan Alur Proses *Login*

Proses *login* dapat dilakukan oleh 3 aktor yang terlibat dalam mengakses sistem. Rancangan alur untuk proses *login* pada sistem *e-voting* dapat dilihat pada Gambar 3.6.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\flowchart-login .png |

Gambar 3. 6 Rancangan Alur Proses *Login*

Gambar 3.6 memperlihatkan aktivitas untuk melakukan proses *login* untuk masuk atau mengakses sistem. Proses *login* juga menentukan hak akses dari masing-masing aktor yang terlibat. Setelah melakukan pengecekan data *login* yang dimasukkan maka dilakukan pengecekan terhadap *role* dari *user* yang akan *login.* Apabila dalam proses pengecekan tidak ada kesalahan, maka *user* akan dibawa menuju halaman sesuai dengan *user role.* Apabila terdapat kesalahan, maka *user* akan diminta memasukkan data *login* kembali.

1. Rancangan Alur Penambahan *Election Authority*

Rancangan alur untuk proses penambahan *election authority* pada sistem *e-voting* dapat dilihat pada Gambar 3.7.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\super-admin_create-ea.png |

Gambar 3. 7 Rancangan Alur Penambahan *Election Authority*

Gambar 3.7 memperlihatkan proses penambahan atau pembuatan *election authority* baru oleh *super admin.* Data yang dimasukkan untuk pembuatan *election authority* adalah nama, *usernme* dan *password. Username* harus bersifat *unique* atau belum pernah digunakan sebelumnya.

1. Rancangan Alur Pembuatan *Election*

Rancangan alur untuk proses pembuatan *election* baru pada sistem *e-voting* dapat dilihat pada Gambar 3.8.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\election-authority_create-election.png |

Gambar 3. 8 Rancangan AlurPembuatan *Election* Baru

Gambar 3.8 memperlihatkan aktivitas untuk melakukan penambahan atau pembuatan *election* baru. *Election authority* yang bertugas melakukan pembuatan *election* baru meng-*input*-kan nama*,* deskripsi, serta tanggal berlangsungnya pemilihan. Setelah proses pembuatan *election,* maka *election* baru berhasil dibuat.

1. Rancangan Alur Tambah Kandidat

Rancangan alur untuk proses penambahan kandidat yang dapat dipilih dalam sebuah *election* pada sistem *e-voting* dapat dilihat pada Gambar 3.9.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\election-authority_add-candidate.png |

Gambar 3. 9 Rancangan AlurPenambahan Kandidat

Gambar 3.9 memperlihatkan aktivitas untuk melakukan proses penambahan kandidat yang dapat dipilih dalam sebuah *election*. Sebelum melakukan penambahan kandidat, *election authority* harus terlebih dahulu membuat *election. Election* yang telah berhasil dibuat kemudian dapat dipilih untuk melakukan penambahan kandidat. *Election authority* memasukkan data nama, visi, misi dan pengalaman dari kandidat. Proses penambahan kandidat dapat dilakukan secara berulang sebanyak jumlah kandidat dalam *election* yang dipilih.

1. Rancangan Alur *Deploy Election*

Rancangan alur untuk proses aktivasi *election* yang telah dibuat sehingga dapat digunakan atau *deploy election* dapat dilihat pada Gambar 3.10.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\super-admin_deploy-election.png |

Gambar 3. 10 Rancangan Alur *Deploy Election*

Gambar 3.10 memperlihatkan aktivitas untuk melakukan proses mengaktifkan *election* yang telah dibuat oleh *election authority* sebelumnya atau proses membawa *smart contract* ke *blockchain* sehingga dapat digunakan. Setelah *election* yang akan di-*deploy* dipilih, maka akan dilakukan proses pengiriman *ether* dari *super admin* ke *wallet address election authority.* Selanjutnya dilakukan proses *deploy smart contract* ke *private blockchain* dan penentuan kandidat di *smart contract* bedasarkan data yang tersimpan di basis data. Setelah semua proses selesai dilakukan maka *smart contract* telah berhasil di-*deploy* dan sudah dapat digunakan.

1. Rancangan Alur Registrasi *Voter*

Sebelum dapat melakukan proses *voting, s*etiap *voter* harus terdaftar pada sistem terlebih dahulu. Rancangan alur untuk proses pendaftaran *voter* dapat dilihat pada Gambar 3.11.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\umum-register-voter.png |

Gambar 3. 11 Rancangan Alur Registrasi *Voter*

Gambar 3.11 memperlihatkan aktivitas untuk melakukan proses pendaftaran masing-masing *voter* dalam sistem *e-voting*. *Voter* memasukkan *username* yang bersifat unik dan *password* untuk melakukan proses pembuatan akun yang terdaftar dalam sistem.

1. Rancangan Alur Pendaftaran *Voter* dalam *Election*

Setiap *voter* yang telah terdaftar di sistem akan melakukan votingdan juga harus terdaftar pada masing-masing *election* yang akan diikuti. Rancangan alur untuk proses pendaftaran *voter* dapat dilihat pada Gambar 3.12.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\voter_voter-register-to-election.png |

Gambar 3. 12 Rancangan Alur Pendaftaran *Voter* dalam *Election*

Gambar 3.12 memperlihatkan aktivitas untuk melakukan proses pendaftaran masing-masing *voter* dalam setiap *election* yang diikuti. Setelah melakukan pemilihan *election* yang diikuti, maka *election authority* akan memverifikasi apakah *voter* tersebut memiliki hak suara atau tidak dalam *election* yang dipilih.

1. Rancangan Alur Proses *Vote*

Rancangan alur untuk pemberian suara atau *vote* oleh setiap *voter* dalam sebuah *election* pada sistem *e-voting* dapat dilihat pada Gambar 3.13.

|  |
| --- |
| C:\Users\sarah\AppData\Local\Microsoft\Windows\INetCache\Content.Word\voter_voter-vote.png |

Gambar 3. 13 Rancangan Alur Proses *Vote*

Gambar 3.13 memperlihatkan aktivitas untuk melakukan proses *vote* oleh masing-masing *voter. Voter* terlebih dahulu melakukan pemilihan *election* dan selanjutnya memilih kandidat yang akan di *vote.* Setelah itu dilakukan proses pengiriman *ether* dari *super admin* ke *wallet address voter* dilanjutkan dengan proses pemanggilan fungsi *vote* pada *smart contract* oleh *voter.* Apabila semua proses telah selesai dilakukan maka suara dari *voter* telah berhasil terekam di *blockchain.*

## Rancangan Basis Data

Perancangan basis data merupakan suatu langkah dalam merancang basis data yang akan digunakan pada sistem. Perancangan basis data digunakan sebagai pendekatan struktur dengan menggunakan prosedur, teknik, alat serta bantuan dokumen untuk membantu dan memudahkan dalam proses perancangan.

### Implementasi Basis Data

Implementasi basis data diambil berdasarkan perancangan basis data yang telah dibuat, secara fisik, implementasi basis data diimplementasikan menggunakan perangkat lunak MySQL. Struktur tabel basis data dapat dilihat pada Gambar 3.14.

|  |
| --- |
| db_simvoni |

Gambar 3. 14 Implementasi Basis Data

Gambar 3.14 merupakan implementasi basis data dari *e-voting* berbasis Ethereum *smart contract*. Basis data tersebut digunakan sebagai media penyimpanan informasi data *user*, *election* dan *candidate*.

### Struktur Data Tabel

Basis data merupakan kumpulan data yang disatukan di dalam kumpulan tabel. Struktur data dari masing-masing tabel dalam perancangan basis data dijelaskan sebagai berikut.

Tabel user diberi nama tabel ref\_user. Tabel ini menyimpan seluruh data dari masing-masing user yang terdaftar dalam sistem ­*e-voting.* Struktur data tabel ref\_user dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Struktur Data Tabel ref\_user

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| user\_id | int |  | Primary key |
| user\_nama | varchar | 255 |  |
| user\_username | varchar | 255 |  |
| user\_password | varchar | 255 |  |
| user\_wallet\_address | varchar | 255 |  |
| user\_role\_id | Int |  | Foreign key |
| user\_login\_at | datetime |  |  |

Tabel 3.2 merupakan struktur data dari tabel ref\_user yang memiliki user\_id sebagai *primary key* beserta dengan *field* user\_nama, user\_username, user\_password, user\_wallet\_address, user\_role\_id, dan user\_login\_at.

Tabel selanjutnya adalah tabel ref\_user\_role. Tabel ini menyimpan data dari *user role* yang ada dalam sistem ­*e-voting.* Struktur data tabel ref\_user\_role dapat dilihat pada Tabel 3.3.

Tabel 3. 3 Struktur Data Tabel ref\_user\_role

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| user\_role\_id | int |  | Primary key |
| user\_role\_nama | varchar | 255 |  |

Tabel 3.3 merupakan struktur data dari tabel ref\_user\_role yang memiliki user\_role\_id sebagai *primary key* beserta dengan *field* user\_role\_nama.

Tabel selanjutnya adalah tabel ta\_candidate. Tabel ini menyimpan seluruh data dari masing-masing kandidatyang terdaftar dalam sistem ­*e-voting.* Struktur data tabel ta\_candidate dapat dilihat pada Tabel 3.4.

Tabel 3. 4 Struktur Data Tabel ta\_candidate

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| candidate\_id | int |  | Primary key |
| candidate\_name | varchar | 255 |  |
| candidate\_election\_id | int | 255 | Foreign key |
| candidate\_visi | varchar | 255 |  |

Tabel 3.4 merupakan struktur data dari tabel ta\_candidate yang memiliki candidate\_id sebagai *primary key* beserta dengan *field* candidate\_name, candidate\_election\_id, dan candidate\_visi.

Tabel selanjutnya adalah tabel ta\_misi. Tabel ini menyimpan seluruh data misi dari masing-masing kandidatyang terdaftar dalam sistem ­*e-voting.* Struktur data tabel ta\_misi dapat dilihat pada Tabel 3.5.

Tabel 3. 5 Struktur Data Tabel ta\_misi

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| misi\_id | int |  | Primary key |
| misi\_deskripsi | varchar | 255 |  |
| misi\_candidate\_id | int | 255 | Foreign key |

Tabel 3.5 merupakan struktur data dari tabel ta\_misi yang memiliki misi\_id sebagai *primary key* beserta dengan *field* misi\_deskripsi dan misi\_candidate\_id.

Tabel selanjutnya adalah tabel ta\_pengalaman. Tabel ini menyimpan seluruh data pengalaman dari masing-masing kandidatyang terdaftar dalam sistem ­*e-voting.* Struktur data tabel ta\_pengalaman dapat dilihat pada Tabel 3.6.

Tabel 3. 6 Struktur Data Tabel ta\_pengalaman

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| pengalaman\_id | int |  | Primary key |
| pengalaman\_deskripsi | varchar | 255 |  |
| pengalaman\_candidate\_id | int | 255 | Foreign key |

Tabel 3.6 merupakan struktur data dari tabel ta\_pengalaman yang memiliki pengalaman\_id sebagai *primary key* beserta dengan *field* pengalaman \_deskripsi dan pengalaman \_candidate\_id.

Tabel selanjutnya adalah tabel ta\_election. Tabel ini menyimpan seluruh data *election* yang terdaftar dalam sistem ­*e-voting.* Struktur data tabel ta\_election dapat dilihat pada Tabel 3.7.

Tabel 3. 7 Struktur Data Tabel ta\_election

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| election\_id | int |  | Primary key |
| election\_name | varchar | 255 |  |
| election\_description | varchar | 255 |  |
| election\_authority | int |  | Foreign key |
| election\_status | int |  | Foreign key |
| election\_start | date |  |  |
| election\_end | date |  |  |
| election\_contract\_address | varchar | 255 |  |

Tabel 3.7 merupakan struktur data dari tabel ta\_election yang memiliki election\_id sebagai *primary key* beserta dengan *field* election\_name, election\_description, election\_authority, election\_status, election\_start, election\_end, dan election\_contract\_address

Tabel selanjutnya adalah tabel ref\_election\_status. Tabel ini menyimpan seluruh data *election status* yang terdaftar dalam sistem ­*e-voting.* Struktur data tabel ref\_election\_status dapat dilihat pada Tabel 3.8.

Tabel 3. 8 Struktur Data Tabel ref\_election\_status

|  |  |  |  |
| --- | --- | --- | --- |
| Nama Field | Tipe Data | Panjang | Keterangan |
| election\_status\_id | int |  | Primary key |
| election\_status | varchar | 255 |  |

Tabel 3.8 merupakan struktur data dari tabel ref\_election\_status yang memiliki election\_status\_id sebagai *primary key* beserta dengan *field* election\_status.

## Bahasa Pemrograman

Implementasi Sistem Voting Elektronik berbasis Ethereum *smart contract* merupakan aplikasi berbasis *web*. Aplikasi ini dibangun dengan bahasa pemrograman Typescript untuk *frontend* dan *backend* dan Solidity untuk membuat *smart contract*. Adapun *database* yang digunakan adalah *database* MySQL.

## Rancangan Antarmuka Aplikasi

Rancangan antarmuka aplikasi adalah rancangan terhadap desain tampilan dari sistem voting elektronik. Keseluruhan tampilan dari sistem voting elektronik adalah berbasis *web*.

|  |
| --- |
|  |

Gambar 3. 15 Rancangan Antarmuka Buat Pemilihan

Gambar 3.15 merupakan rancangan tampilan untuk membuat pemilihanpada sistem voting elektronik. Rancangan tampilan tersebut menampilkan sebuah *form* yang terdiri dari 4 *input* yaitu nama, deskripsi, mulai dan selesai serta sebuah tombol simpan.

|  |
| --- |
|  |

Gambar 3. 16 Rancangan Antarmuka Menambahkan Kandidat

Gambar 3.16 merupakan rancangan tampilan untuk menambahkan kandidat pada sistem voting elektronik. Rancangan tampilan tersebut menampilkan sebuah *form* dengan 4 *input* yaitu nama, visi, misi dan pengalaman. *Input* misi dan pengalaman digunakan untuk menyimpan data dari *multiple* *input* sehingga terdapat tombol tambah di sebelah kanan input *field.*

|  |
| --- |
|  |

Gambar 3. 17 Rancangan Antarmuka Daftar Pemilihan Siap *Deploy*

Gambar 3.17 merupakan rancangan tampilan untuk melihat daftar dari pemilihan yang siap di-*deploy* ke jaringan Ethereum. *Field* yang ditampilkan adalah nama pemilihan, tanggal mulai, tanggal selesai dan *election authority* yang membuat pemilihan tersebut serta sebuah tombol *deploy*.

|  |
| --- |
|  |

Gambar 3. 18 Rancangan Antarmuka Daftar Peserta Pemilihan

Gambar 3.18 merupakan rancangan tampilan untuk melihat daftar peserta suatu pemilihan pada sistem voting elektronik. *Field* yang ditampilkan adalah *username*, status keikutsertaan dan tombol untuk menerima atau menolak.

|  |
| --- |
|  |

Gambar 3. 19 Rancangan Antarmuka Detail Pemilihan

Gambar 3.19 merupakan rancangan tampilan untuk melihat detail dari sebuah pemilihan pada sistem voting elektronik. Informasi yang ditampilkan adalah nama pemilihan, tanggal mulai dan selesai dan deskripsi pemilihan. Selain itu terdapat daftar kandidat yang masing-masing menampilkan nama, visi, misi, pengalaman dan tombol untuk memilih kandidat tersebut.

|  |
| --- |
|  |

Gambar 3. 20 Rancangan Antarmuka Hasil Pemilihan

Gambar 3.20 merupakan rancangan tampilan untuk melihat hasil dari sebuah pemilihan pada sistem voting elektronik. Informasi yang ditampilkan adalah nama pemilihan, tanggal mulai dan selesai, deskripsi pemilihan dan pemenang. Selain itu terdapat daftar kandidat yang masing-masing menampilkan nama, visi dan jumlah suara yang diperoleh.

## Rancangan Spesifikasi API

Rancangan spesifikasi *endpoint* adalah rancangan terhadap *request* dan *response* yang diberikan oleh *backend* sistem *e-voting* melalui protokol *http*.

|  |
| --- |
| POST /election-authority/election  {  "name": string,  "description": string,  "start": date,  "end": date,  } |

Kode Program 3. 1 Spesifikasi *Request* Pembuatan Pemilihan

Kode Program 3.1 merupakan spesifikasi *request* untuk membuat pemilihan. Metode yang digunakan adalah POST serta *field* yang dibutuhkan adalah name dan description dengan tipe data string serta start dan end dengan tipe data date.

|  |
| --- |
| {  "message": string,  "data": {  "id": number,  "name": string,  "description": string,  "start": date,  "end": date,  "status": string,  "ea": string,  }  } |

Kode Program 3. 2 Spesifikasi Response Pembuatan Pemilihan

Kode Program 3.2 merupakan spesifikasi *response* dari hasil pembuatan pemilihan. *Field* yang ditampilkan adalah *field* yang sebelumnya digunakan untuk membuat pemilihan ditambah dengan id dengan tipe data number. status dan ea dengan tipe data string.

|  |
| --- |
| POST /election-authority/add-candidate/:electionId  {  "name": string,  "visi": string,  "misi": [string],  "pengalaman": [string]  } |

Kode Program 3. 3 Spesifikasi Request Penambahan Kandidat

Kode Program 3.3 merupakan spesifikasi *request* untuk menambahkan kandidat pada sebuah pemilihan. Metode yang digunakan adalah POST serta *field* yang dibutuhkan adalah name dan visi dengan tipe data string serta misi dan pengalaman dengan tipe data array of string.

|  |
| --- |
| {  "message": string,  "data": {  "id": number,  "name": string,  "visi": string,  "candidateSlug": string,  "election": string  }  } |

Kode Program 3. 4 Spesifikasi *Response* Penambahan Kandidat

Kode Program 3.4 merupakan spesifikasi *response* dari hasil penambahan kandidat. *Field* yang ditampilkan adalah id dengan tipe data number serta name, visi, candidateSlug dan election dengan tipe data string. *Field* candidateSlug adalah nama kandidat yang diolah sehingga bisa digunakan di *smart contract*.

|  |
| --- |
| POST /super-admin/deploy-election/:electionId |

Kode Program 3. 5 Spesifikasi *Request Deployment* Pemilihan

Kode Program 3.5 merupakan spesifikasi *request* untuk melakukan *deployment* sebuah pemilihan ke jaringan Ethereum. Metode yang digunakan adalah POST dan tidak menggunakan *request body*.

|  |
| --- |
| {  "message": string,  "data": {  "address": string  }  } |

Kode Program 3. 6 Spesifikasi *Response Deployment* Pemilihan

Kode Program 3.6 merupakan spesifikasi *response* dari hasil *deployment* sebuah pemilihan ke jaringan Ethereum. *Field* yang ditampilkan adalah address dengan tipe data string.

|  |
| --- |
| GET /election-authority/election-participant/:electionId |

Kode Program 3. 7 Spesifikasi *Request* Peserta Pemilihan

Kode Program 3.7 merupakan spesifikasi *request* untuk mendapatkan semua peserta yang mengikuti sebuah pemilihan. Metode yang digunakan adalah GET tanpa menggunakan *request body*.

|  |
| --- |
| {  "message": string,  "data": {  "electionId": number,  "electionName": string,  "participant": [  {  "participationId": number,  "userId": number,  "username": string,  "status": string  }  ]  }  } |

Kode Program 3. 8 Spesifikasi *Resposne* Peserta Pemilihan

Kode Program 3.8 merupakan spesifikasi *response* untuk menampilkan peserta sebuah pemilihan. *Field* yang ditampilkan adalah electionId dengan tipe data number. electionName dengan tipe data string. Dan untuk *field* participant adalah array dari suatu object. *Field* object tersebut adalah participationId dan userId dengan tipe data number serta username dan status dengan tipe data string.

|  |
| --- |
| GET /voter/election-detail/:electionId |

Kode Program 3. 9 Spesifikasi *Request* *Detail* Pemilihan

Kode Program 3.9 merupakan spesifikasi *request* untuk mendapatkan informasi detail sebuah pemilihan. Metode yang digunakan adalah GET tanpa menggunakan *request body*.

|  |
| --- |
| {  "message": string,  "data": {  "id": number,  "name": string,  "description": string,  "start": date,  "end": date,  "status": string,  "ea": string,  "participation\_status": string,  "candidates": [  {  "id": number,  "visi": string,  "misi": [string],  "pengalaman": [string]  }  ]  }  } |

Kode Program 3. 10 Spesifikasi *Response* *Detail* Pemilihan

Kode Program 3.10 merupakan spesifikasi *response* untuk menampilkan informasi detail sebuah pemilihan. *Field* yang ditampilkan adalah id dengan tipe data number. name, description, status, ea dan participation\_status dengan tipe data string, start dan end dengan tipe data date. Dan untuk *field* candidates adalah array dari suatu object. Spesifikasi object tersebut adalah id dengan tipe data number, visi dengan tipe data string serta misi dan pengalaman dengan tipe data array of string.

|  |
| --- |
| GET /voter/ended-election-detail/:electionId |

Kode Program 3. 11 Spesifikasi *Request* Hasil Pemilihan

Kode Program 3.11 merupakan spesifikasi *request* untuk mendapatkan informasi hasil dari sebuah pemilihan yang telah selesai. Metode yang digunakan adalah GET tanpa menggunakan *request body*.

|  |
| --- |
| {  "message": string,  "data": {  "id": number,  "name": string,  "description": string,  "start": date,  "end": date,  "status": string,  "ea": string,  "winner": string,  "candidates": [  {  "id": number,  "visi": string,  "vote\_count": number,  "misi": [string],  "pengalaman": [string]  }  ]  }  } |

Kode Program 3. 12 Spesifikasi *Response* Hasil Pemilihan

Kode Program 3.12 merupakan spesifikasi *response* untuk menampilkan hasil dari pemilihan yang telah selesai. *Field* yang ditampilkan adalah id dengan tipe data number. name, description, status, ea dan winner dengan tipe data string, start dan end dengan tipe data date. Dan untuk *field* candidates adalah array dari suatu object. Spesifikasi object tersebut adalah id dengan tipe data number. visi dengan tipe data string. vote\_count dengan tipe data number serta misi dan pengalaman dengan tipe data array of string.

|  |
| --- |
| POST /election-authority/start-election/:electionId |

Kode Program 3. 13 Spesifikasi *Request* Memulai Pemilihan

Kode Program 3.13 merupakan spesifikasi *request* untuk memulai sebuah pemilihan. Metode yang digunakan adalah POST tanpa *request body*. Parameter yang digunakan adalah id dari pemilihan yang akan dimulai.

|  |
| --- |
| {  "message": string,  "data": {  "election": string  }  } |

Kode Program 3. 14 Spesifikasi *Response* Memulai Pemilihan

Kode Program 3.14 merupakan spesifikasi *response* yang ditampilkan untuk memulai sebuah pemilihan. *Field* yang ditampilkan adalah election dengan tipe data string.

|  |
| --- |
| POST /voter/join/:electionId |

Kode Program 3. 15 Spesifikasi *Request* Mengikuti Pemilihan

Kode Program 3.15 merupakan spesifikasi *request* untuk mengikuti sebuah pemilihan. Metode yang digunakan adalah POST tanpa *request body*. Parameter yang digunakan adalah id dari pemilihan yang akan diikuti.

|  |
| --- |
| {  "message": string,  "data": {  "election": string  }  } |

Kode Program 3. 16 Spesifikasi *Response* Mengikuti Pemilihan

Kode Program 3.16 merupakan spesifikasi *response* yang ditampilkan untuk mengikuti sebuah pemilihan. *Field* yang ditampilkan adalah election dengan tipe data string.

|  |
| --- |
| POST /voter/vote  {  "election\_id": number,  "candidate\_id": number  } |

Kode Program 3. 17 Spesifikasi *Request* untuk Memilih Kandidat

Kode Program 3.17 merupakan spesifikasi *request* untuk memilih kandidat dalam sebuah pemilihan. Metode yang digunakan adalah POST dan *field* yang dibutuhkan adalah election\_id dan candidate\_id dengan tipe data number.

|  |
| --- |
| {  "message": string,  "data": {  "election": string  }  } |

Kode Program 3. 18 Spesifikasi *Response* untuk Memilih Kandidat

Kode Program 3.18 merupakan spesifikasi *response* yang ditampilkan untuk memilih kandidat dalam sebuah pemilihan. *Field* yang ditampilkan adalah election dengan tipe data string.

|  |
| --- |
| POST /election-authority/end-election/:electionId |

Kode Program 3. 19 Spesifikasi *Request* untuk Menghentikan Pemilihan

Kode Program 3.19 merupakan spesifikasi *request* untuk menghentikan sebuah pemilihan. Metode yang digunakan adalah POST tanpa *request body*. Parameter yang digunakan adalah id dari pemilihan yang akan dihentikan.

|  |
| --- |
| {  "message": string,  "data": {  "election": string  }  } |

Kode Program 3. 20 Spesifikasi *Response* untuk Menghentikan Pemilihan

Kode Program 3.20 merupakan spesifikasi *response* yang ditampilkan untuk menghentikan sebuah pemilihan. *Field* yang ditampilkan adalah election dengan tipe data string.

# BAB IV PENGUJIAN DAN ANALISA HASIL

BAB IV menjelaskan hasil dari penelitian ini yaitu berupa hasil pengembangan aplikasi, pengujian aplikasi, analisa aplikasi serta implementasinya ke inftrastruktur *production* yaitu *Virtual Machine* Google Cloud.

## Infrastruktur Google Cloud

Google Cloud adalah *cloud provider* berskala internasional yang sudah terbukti menyediakan layanan yang handal untuk keperluan industri di bidang teknologi informasi. Layanan yang ditawarkan oleh Google Cloud sangat beragam diantaranya yang digunakan dalam pengembangan aplikasi ini adalah *Virtual Machine* dengan spesifikasi yang fleksibel sesuai kebutuhan dan biaya sehingga semua sumber daya bisa digunakan secara efisien. Google Cloud juga telah menyelesaikan pembangunan *data center* untuk *region* Jakarta sehingga bisa meminimalisir latensi.

### Persiapan Infrastruktur

*Virtual Machine* yang dibutuhakan untuk membuat jaringan *private blockchain* adalah dua buah *virtual machine* sesuai dengan Gambaran Umum aplikasi.

|  |  |
| --- | --- |
| |  | | --- | |  | |

Gambar 4. 1 Halaman Google Cloud untuk Mengelola *Firewall Rule*

Gambar 4.1 menunjukkan halaman Google Cloud untuk mengelola *firewall rule*. *Firewall rule* adalah adalah informasi untuk melakukan *filter* terhadap *request* yang diterima *virtual machine* pada suatu *port*. Umumnya *firewall rule* diatur dalam *virtual machine* dengan perintah *ufw*. Namun untuk *virtual machine* pada Google Cloud terdapat halaman terpisah untuk mengelola *firewall rule* yang digunakan oleh *virtual machine*.

|  |
| --- |
|  |

Gambar 4. 2 *Firewall Rule* yang Diperlukan

Gambar 4.2 menunjukkan halaman pembuatan *firewall rule* untuk *virtual machine* yang akan dibuat. Hal yang perlu diperhatikan adalah *Direction of traffic* adalah *Ingress* yang artinya *trafick* berasal dari luar menuju ke arah *virtual machine*. *Action on match* adalah *Allow* yang artinya memperbolehkan *trafick* untuk diteruskan apabila sesuai dengan *firewall rule*. *Target tags* bisa disesuaikan dengan ketentuan penamaan masing-masing, untuk studi kasus pada penelitian ini menggunakan nama *all-in*. *Source IPv4 ranges* adalah 0.0.0.0/0 yang artinya *IP* apa saja boleh mengakses *virtual machine*. Sedangkan *Protocols and ports* adalah *Allow all* yang artinya memberbolehkan semua protokol komunikasi diantaranya *http* dan *udp*.

|  |  |
| --- | --- |
| |  | | --- | |  | |

Gambar 4. 3 Halaman Google Cloud untuk Mengelola *VM*

Gambar 4.3 menunjukkan halaman Google Cloud untuk mengelola Compute Engine. Compute Engine adalah nama layanan Google Cloud yang memberikan *virtual machine* yang dapat bekerja seperti layaknya *server*.

|  |  |
| --- | --- |
| |  | | --- | |  | |

Gambar 4. 4 Spesifikasi *Virtual Machine*

Gambar 4.4 menunjukkan spesifikasi *Virtual Machine* yang akan digunakan sebagai *Node* Ethereum. Spesifikasi dari *virtual machine* adalah menggunakan 2vCPU dengan 1GB RAM. Sistem Operasi yang digunakan adalah Ubuntu 18.04 LTS.

|  |
| --- |
|  |

Gambar 4. 5 Konfigurasi Jaringan *Virtual Machine*

Gambar 4.5 menunjukkan *form* untuk melakukan konfigurasi *networking virtual machine*. Gunakan *tags* yang sudah dibuat sebelumnya yaitu *all-in* dan *all-out.* Ini bertujuan untuk memperbolehkan akses ke *virtual machine* dari berbagai *port* dan berbagai protokol komunikasi.

|  |
| --- |
|  |

Gambar 4. 6 Konfigurasi Security *Virtual Machine*

Gambar 4.6 menunjukkan *form* untuk melakukan konfigurasi *security* *virtual machine*. Hal yang perlu diperhatikan adalah pada *input SSH key*. Pastikan menggunakan *public key* dari *ssh key* yang berada pada mesin lokal sehingga nantinya bisa diakses melalui *ssh*.

Lakukan hal yang sama dimulai dari pembuatan *virtual machine* sekali lagi sehingga terdapat dua *virtual machine* untuk dijadikan *node* Ethereum.

|  |
| --- |
|  |

Gambar 4. 7 Halaman Compute Engine ketika berhasil membuat *VM*

Gambar 4.7 menunjukkan pada halaman pengelolaan Compute Engine terdapat dua *virtual machine* yang telah dibuat. Masing-masing akan mendapatkan sebuah *IP Public* yang nantinya akan digunakan untuk mengakses *virtual machine*. *IP Public* yang diperoleh bersifat *ephemeral* atau tidak statis, jadi jika *virtual machine* dimatikan kemudian dihidupkan kembali pasti akan mendapatkan *IP* yang berbeda. Setiap pengguna Google Cloud hanya diberi kuota empat *IP Public* untuk sebuah *region* untuk *virtual machine* pada Compute Engine sehingga harus digunakan dengan bijak.

### Persiapan Virtual Machine

Setelah *virtual machine* berhasil dibuat, ada beberapa *package* yang perlu di-*install* pada *virtual machine* sebelum siap dijadikan *node ethereum*. Hal yang dibutuhkan antara lain adalah *package* *libsnappy-dev* dan *unzip*, *binary openethereum* serta *file* konfigurasi *node* Ethereum.

|  |
| --- |
|  |

Gambar 4. 8 *SSH* ke *Virtual Machine*

Gambar 4.8 menunjukkan sukses melakukan *ssh* ke *virtual machine* yang sebelumnya telah dibuat. Pada studi kasus ini *IP* dari *virtual machine* yang pertama adalah 34.101.182.96.

|  |
| --- |
|  |

Gambar 4. 9 Proses Instalasi *Package*

Gambar 4.9 menunjukkan proses instalasi *package* *libsnappy-dev* dan *unzip* menggunakan *package manager* yang terdapat pada sistem operasi Ubuntu. Perintah yang dijalankan adalah sudo apt install libsnappy-dev unzip.

|  |
| --- |
|  |

Gambar 4. 10 Proses *Download Binary* Openethereum

Gambar 4.10 menunjukkan proses *download* dari *binary* Openethereum. Proses yang dilakukan adalah mengubah *directory* menjadi /usr/local/bin. Kemudian menggunakan perintah wget <https://github.com/openethereum/openethereum/releases/download/v3.2.6/openethereum-linux-v3.2.6.zip>. Setelah *download* selesai maka akan ada *file* dengan nama openethereum-linux-v3.2.6.zip

|  |
| --- |
|  |

Gambar 4. 11 Proses *Extract File*

Gambar 4.11 menunjukkan proses *extract file* yang telah di-*download*. *Extract file* menggunakan perintah *unzip* yang telah di-*install* sebelumnya. Perintah yang dijalankan adalah sudo unzip openethereum-linux-v3.2.6.zip Ada empat *file* yang muncul jika proses *extract* berhasil yaitu ethkey, ethstore, openethereum dan openethereum-evm.

|  |
| --- |
|  |

Gambar 4. 12 Proses Ubah *Permission File*

Gambar 4.12 menunjukkan proses mengubah *permission* dari *file* yang sebelumnya di-*extract*. Perintah yang dijalankan adalah sudo chmod 755 openethereum openethereum-evm ethkey ethstore. Hal ini dilakukan agar *binary* bisa dijalankan.

|  |
| --- |
|  |

Gambar 4. 13 Proses *Download* Konfigurasi *Node* Ethereum

Gambar 4.13 menunjukkan proses *download* konfigurasi *node ethereum* untuk jaringan *private blockchain*. Hal yang dilakukan adalah mengubah *directory* menjadi ~. Dalam sistem operasi berbasis *linux directory* ~ artinya *directory home* dari sebuah *user*. Proses *download* dimulai dengan menjalankan perintah git clone <https://github.com/cekingx/simvoni-PoA.git> . Setelah *download* selesai maka akan ada *folder* dengan nama simvoni-PoA.

Lakukan proses hal yang sama pada *virtual machine* yang lainnya karena persiapan perlu dilakukan pada kedua *virtual machine*.

## Konfigurasi Openethereum

Openethereum memerlukan beberapa *file* konfigurasi untuk bisa membuat jaringan *private blockchain* karena secara *default* Openethereum akan berjalan pada jaringan utama Ethereum. Terdapat beberapa konfigurasi khusus untuk membuat sebuah *Private Network*. *Private network* adalah sebuah jaringan *blockchain* yang terisolasi dari jaringan *blockchain* utama sehingga kita bisa menyesuaikan jaringan sesuai dengan kebutuhan. Jaringan utama *blockchain* Ethereum menggunakan konsensus Proof of Work dimana konsensus ini menggunakan sangat banyak daya komputasi. Untuk keperluan voting lembaga independen seperti misalnya organisasi mahasiswa yang memiliki dana terbatas, tentunya penghematan biaya adalah prioritas. Menggunakan konsensus PoA dalam jaringan *blockchain private* adalah sebuah solusi karena jaringan tidak membutuhkan daya komputasi yang besar sehingga *VM* dengan spesifikasi rendah sudah cukup untuk menjalankan jaringan. Konsensus PoA tidak memerlukan daya komputasi yang besar karena *difficulty* untuk membuat sebuah blok relatif kecil

|  |
| --- |
|  |

Gambar 4. 14 Struktur *File* Konfigurasi

Gambar 4.14 menunjukkan struktur *file* dari *git repository* yang sudah di-*clone* sebelumnya. *File* yang dibutuhkan berupa simvoni-spec.json, simvoni-config.json dan node.pwds.

### Genesis Block

*Genesis block* adalah blok pertama yang terdapat dalam sebuah *blockchain* yang nantinya akan disambung oleh blok lainnya sehingga bisa menjadi sebuah rantai blok. Konfigurasi bagaimana sebuah jaringan *blockchain* bekerja tersimpan dalam *genesis block*. Jaringan *blockchain* *private* yang digunakan untuk implementasi Sistem Voting Elektronik bernama SimvoniPoA. Dalam jaringan SimvoniPoA terdapat dua *node* yang berada pada *Virtual Machine* yang berbeda. Kedua *node* tersebut berperan sebagai *sealer*. Dalam jaringan *blockchain* yang menggunakan konsensus PoW, *account* yang berhak membuat blok disebut dengan *miner*. Sedangkan dalam jaringan *blockchain* yang menggunakan konsensus PoA, *account* yang berhak membuat blok disebut dengan *sealer*.

|  |
| --- |
| {  "name": "SimvoniPoA",  "engine": {  "authorityRound": {  "params": {  "stepDuration": "5",  "validators" : {  "list": [  "0x00aa39d30f0d20ff03a22ccfc30b7efbfca597c2",  "0x002e28950558fbede1a9675cb113f0bd20912019"  ]  }  }  }  },  "params": {  "gasLimitBoundDivisor": "0x400",  "maximumExtraDataSize": "0x20",  "minGasLimit": "0x1388",  "networkID" : "0x2323",  "eip150Transition": "0x0",  "eip160Transition": "0x0",  "eip161abcTransition": "0x0",  "eip161dTransition": "0x0",  "eip155Transition": "0x0",  "eip98Transition": "0x7fffffffffffff",  "validateChainIdTransition": 0,  "eip140Transition": "0x0",  "eip211Transition": "0x0",  "eip214Transition": "0x0",  "eip658Transition": "0x0",  "eip145Transition": "0x0",  "eip1014Transition": "0x0",  "eip1052Transition": "0x0"  },  "genesis": {  "seal": {  "authorityRound": {  "step": "0x0",  "signature": "0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000"  }  },  "difficulty": "0x20000",  "gasLimit": "0x5B8D80"  },  "accounts": {  "0x0000000000000000000000000000000000000001": { "balance": "1", "builtin": { "name": "ecrecover", "pricing": { "linear": { "base": 3000, "word": 0 } } } },  "0x0000000000000000000000000000000000000002": { "balance": "1", "builtin": { "name": "sha256", "pricing": { "linear": { "base": 60, "word": 12 } } } },  "0x0000000000000000000000000000000000000003": { "balance": "1", "builtin": { "name": "ripemd160", "pricing": { "linear": { "base": 600, "word": 120 } } } },  "0x0000000000000000000000000000000000000004": { "balance": "1", "builtin": { "name": "identity", "pricing": { "linear": { "base": 15, "word": 3 } } } },  "0x00b108e445c6fb0e38ef3a7d4ba5b4f934471236": { "balance": "10000000000000000000000" }  }  } |

Kode Program 4. 1 *File* simvoni-spec.json

Kode Program 4.1 *genesis block* yang terdapat pada *file* simvoni-spec.json. *Genesis block* tersebut menunjukkan bahwa jaringan SimvoniPoA berjalan menggunakan *authorityRound* sebagain *engine* konsensus. *authorityRound* adalah implementasi dari konsensus PoA. *Difficulty* dari jaringan ini adalah 0x20000 atau jika dikonversi menjadi desimal adalah 131.072. Angka tersebut relatif kecil jika dibandingkan dengan *difficulty* jaringan utama yang bisa mencapai 17.179.869.184. *Gas limit* dari jaringan ini adalah 0x5B8D80 atau jika dikonversi menjadi desimal adalah 6.000.000 unit.

### Konfigurasi Node Openethereum

*Node* dalam sebuah jaringan Ethereum menggunakan protokol *devp2p* untuk berkomunikasi satu sama lain. *Devp2p* adalah protokol komunikasi *peer-to-peer* khusus yang dibuat untuk Ethereum. *Devp2p* menggunakan protokol komunikasi *UDP* yang secara *default* menggunakan *port* 30303. Selain berkomunikasi dengan *node* lainnya dalam jaringan Ethereum, *node* juga perlu dikonfigurasi agar bisa berkomunikasi dengan aplikasi yang membutuhkan integrasi dengan *blockchain*. Protokol komunikasi yang digunakan adalah protokol *http*. Bentuk pesan yang dikirimkan dari aplikasi ke *node* Ethereum adalah *jsonrpc* yang artinya aplikasi menggunakan pesan dengan format *json* untuk melakukan *remote procedure call* pada *node* Ethereum. Terdapat beberapa *jsonrpc api* yang disediakan oleh *node* Ethereum yaitu *all*, *safe*, *debug*, *web3*, *net*, *eth*, *pubsub*, *personal*, *signer*, *parity*, *parity\_pubsub*, *parity\_accounts*, *parity\_set*, *traces* dan *secretstore*. Konfigurasi *CORS* juga perlu diperhatikan apabila aplikasi yang ingin melakukan *http request* ke *node* menggunakan bahasa pemrograman JavaScript karena tanpa adanya *CORS*, *http request* yang dilakukan oleh aplikasi tersebut akan gagal.

|  |
| --- |
| [parity]  chain = "simvoni-spec.json"  base\_path = "$HOME/openethereum-node"  [network]  port = 30300  allow\_ips = "all"  nat = "0.0.0.0"  [rpc]  interface = "all"  port = 8540  apis = ["all"]  cors = ["all"]  hosts = ["all"]  [websockets]  port = 8450  interface = "all"  origins = ["all"]  [account]  password = ["node.pwds"]  [mining]  # engine\_signer = ""  reseal\_on\_txs = "all"  reseal\_max\_period = 4000 |

Kode Program 4. 2 *File* simvoni-config.toml

Kode Program 4.2 adalah konfigurasi dari *node* yang terdapat dalam *file* simvoni-config.toml. Dalam bagian *parity*, konfigurasi *chain* adalah *file* *genesis block* yang digunakan dalam jaringan. Konfigurasi *base\_path* adalah direktori penyimpanan data *blockchain*. Dalam bagian *network*, konfigurasi *port* adalah *port* yang digunakan oleh *node* untuk melakukan komunikasi *peer-to-peer* di dalam jaringan. Dalam bagian *rpc*, konfigurasi *interface* adalah *jsonrpc api* yang boleh digunakan pada *node*. Nilai *all* artinya semua *jsonrpc api* yang tersedia boleh untuk digunakan. Konfigurasi *cors* adalah daftar *IP* mana saja yang boleh melakukan *http request* menggunakan *javascript*. Nilai *all* artinya semua *IP* boleh melakukan *http request* dari *javascript*.

### Menjalankan Jaringan

Setelah memenuhi kebutuhan untuk menjalankan *binary* Openethereum diperlukan beberapa langkah tambahan agar jaringan *private blockchain* dapat berjalan.

|  |
| --- |
| openethereum --config simvoni-config.toml --nat extip:34.101.182.96 --no-color |

Kode Program 4. 3 Perintah Menjalankan Openethereum

Kode Program 4.3 adalah perintah yang digunakan untuk menjalankan Openethereum dalam sebuah *virtual machine*. Sesuaikan *IP* *virtual machine* pada bagian extip.

|  |
| --- |
| curl -X POST \  -H "Content-Type: application/json" \  -d '{  "jsonrpc":"2.0",  "method":"parity\_newAccountFromPhrase",  "params":["node1", "node1-password"],  "id":0  }' \  http://34.101.182.96:8540 |

Kode Program 4. 4 CURL untuk Membuat *Validator Account*

Kode Program 4.4 adalah *curl* yang digunakan untuk membuat *validator account* pada *virtual machine* dengan IP 34.101.182.96.

|  |
| --- |
| curl -X POST \  -H "Content-Type: application/json" \  -d '{  "jsonrpc":"2.0",  "method":"parity\_newAccountFromPhrase",  "params":["node2", "node2-password"],  "id":0  }' \  http://34.101.154.53:8540 |

Kode Program 4. 5 CURL untuk Membuat *Validator Account*

Kode Program 4.5 adalah *curl* yang digunakan untuk membuat *validator account* pada *virtual machine* dengan IP 34.101.154.53.

|  |
| --- |
| echo “node1-password” > node.pwds |

Kode Program 4. 6 Menyimpan *Password Validator*

Kode Program 4.6 adalah perintah yang digunakan untuk menyimpan *password* dari *validator account* pada *file* node.pwds.

Ulangi kembali langkah untuk menjalankan Openethereum sehingga Openethereum sudah mendapatkan *update* konfigurasi.

Setiap *node* Ethereum yang berjalan diidentifikasikan dengan sebuah *url* *enode*. Cara menghubungkan *node* Ethereum adalah dengan mendapatkan *url* *enode* dari sebuah *node* kemudian menambahkan *url* tersebut ke *node* lainnya

|  |
| --- |
| curl --data '{  "jsonrpc":"2.0",  "method":"parity\_enode",  "params":[],  "id":0  }' -H "Content-Type: application/json"  -X POST  34.101.182.96:8540 |

Kode Program 4. 7 Mendapatkan Enode

Kode Program 4.7 adalah perintah untuk melakukan *http request* dengan bentuk *jsonrpc* ke sebuah *node* Ethereum untuk mendapatkan *url* *enode* dari *node* tersebut.

|  |
| --- |
| curl --data '{  "jsonrpc":"2.0",  "method":"parity\_addReservedPeer",  "params":["enode://RESULT"],  "id":0  }' -H "Content-Type: application/json"  -X POST  34.101.154.53:8540 |

Kode Program 4. 8 Menambahkan Enode

Kode Program 4.8 adalah perintah untuk melakuan *http request* dengan bentuk *jsonrpc* ke *node* Ethereum lainnya dengan argumen *url* *enode* yang sudah didapatkan sebelumnya.

|  |
| --- |
|  |

Gambar 4. 15 *Node* Terhubung

Gambar 4.15 menunjukkan perbedaan ketika *node* Ethereum yang belum terhubung dan sudah terhubung. Perbedaannya adalah ketika *node* belum terhubung, *log* yang muncul adalah 0/25 *peer*. Jika sudah terhubung *log* akan berubah menjadi 1/25 *peer*.

## Implementasi Desain Antarmuka

Implementasi Desain Antarmuka adalah proses pembuatan tampilan yang bisa digunakan dalam sistem voting elektronik. Aplikasi dapat digunakan oleh 3 jenis user role yaitu *Super Admin*, *Election Authority* dan *Voter*.

### Membuat Pemilihan

Halaman pembuatan pemilihan adalah tampilan awal dari sebuah proses voting elektronik. Halaman ini terdiri dari sebuah *form* dengan empat buah *input* serta sebuah *button*.

|  |
| --- |
|  |

Gambar 4. 16 Tampilan Halaman Pembuatan Pemilihan

Gambar 4.16 menunjukkan *form* untuk membuat sebuah pemilihan terdiri dari empat buah *input* yaitu Nama, Deskripsi, Mulai dan Selesai. Serta sebuah *button* Simpan untuk melakukukan *http request* ke *backend*.

|  |
| --- |
|  |

Gambar 4. 17 Tampilan Pesan Sukses

Gambar 4.17 menampilkan pesan “Pemilihan Berhasil Dibuat” yang diberikan ketika berhasil melakukan *http request* untuk membuat sebuah pemilihan.

### Menambahkan Kandidat

Halaman penambahan kandidat adalah tampilan untuk menambahkan kandidat pada sebuah pemilihan yang telah berhasil dibuat. Halaman ini terdiri dari sebuah *form* yang terdiri dari empat *input* dan sebuah *button*.

|  |
| --- |
|  |

Gambar 4. 18 Tampilan Halaman Penambahan Kandidat

Gambar 4.18 menunjukkan *form* untuk menambahkan seorang kandidat pada sebuah pemilihan terdiri dari empat *input* yaitu Nama, Visi, Misi dan Pengalaman. Khusus untuk *input* Misi dan Pengalaman bersifat *multiple input*. Serta sebuah *button* Simpan untuk melakukan *http request* ke *backend.*

|  |
| --- |
|  |

Gambar 4. 19 Tampilan Pesan Sukses

Gambar 4.19 menampilkan pesan “Kandidat Berhasil Ditambahkan” yang diberikan ketika berhasil melakukan *http request* untuk menambahkan seorang kandidat.

### Deploy Pemilihan

Melakukan *deployment* sebuah pemilihan memerlukan langkah-langkah yang lebih panjang yaitu status pemilihan akan diubah mejadi “Siap Deploy” oleh *Election Authority* kemudian akan di-*deploy* oleh *Super Admin*.

|  |
| --- |
|  |

Gambar 4. 20 Tampilan *Detail* Pemilihan

Gambar 4.20 menunjukkan tampilan *detail* sebuah pemilihan dari sisi *Election Authority*. Tampilan *detail* pemilihan terdapat informasi dasar pemilihan seperti Nama, Deskripsi, Mulai dan Selesai serta daftar kandidat. Pada tampilan tersebut terdapat *button* “Siap Deploy” untuk mengubah status pemilihan.

|  |
| --- |
|  |

Gambar 4. 21 Tampilan Konfirmasi

Gambar 4.21 menampilkan *modal* untuk konfirmasi apakah yakin untuk mengubah status pemilihan tersebut. Jika memilih “Cancel” tidak akan terjadi apa-apa, sedangkan jika memilih “OK” maka akan membuat *http request* ke *backend*.

|  |
| --- |
|  |

Gambar 4. 22 Tampilan Pesan Sukses

Gambar 4.22 menampilkan pesan “Tunggu Super Admin Untuk Deployment” yang diberikan ketika berhasil melakukan *http request* untuk mengubah status pemilihan.

|  |
| --- |
|  |

Gambar 4. 23 Tampilan Daftar Pemilihan yang Siap Deploy

Gambar 4.23 menunjukkan daftar pemilihan yang statusnya siap untuk di-*deploy*. Halaman ini hanya bisa dilihat oleh *Super Admin*. Terdapat *button* “Deploy” untuk melakukan *deployment* terhadap sebuah pemilihan.

|  |
| --- |
|  |

Gambar 4. 24 Tampilan Pesan Sukses

Gambar 4.24 menampilkan pesan “Pemilihan Berhasil di Deploy” yang diberikan ketika sukses melakukan *http* *request* untuk *deploy* pemilihan.

### Mengikuti Pemilihan

Halaman ikuti pemilihan adalah tampilan *Voter* untuk mengikuti sebuah pemilihan. Halaman ini terdiri dari daftar pemilihan serta *button* pada masing-masing pemilihan.

|  |
| --- |
|  |

Gambar 4. 25 Tampilan Daftar Pemilihan

Gambar 4.25 menunjukkan tampilan halaman untuk mengikuti pemilihan. Terdapat daftar pemilihan yang tersedia di sistem voting elektronik. Informasi yang ditampilkan masing-masing pemilihan adalah Nama, Periode, Status serta *button* untuk mengikuti pemilihan.

|  |
| --- |
|  |

Gambar 4. 26 Tampilan Pesan Sukses

Gambar 4.26 menampilkan pesan “Pemilihan berhasil diikuti. *EA* adalah akronim dari *Election Authority*. Tunggu *EA* untuk melakukan persetujuan” ketika berhasil melakukan *http request* untuk mengikuti pemilihan.

### Daftar Peserta Pemilihan

Halaman daftar peserta pemilihan adalah tampilan pada *Election Authority* untuk melihat dan melakukan aksi terhadap peserta pemilihan.

|  |
| --- |
|  |

Gambar 4. 27 Tampilan Daftar Peserta Pemilihan

Gambar 4.27 menunjukkan daftar peserta dalam sebuah pemilihan. Informasi yang ditampilkan dari masing-masing peserta adalah *Username*, Status serta dua buah *button* aksi.

### Memulai Pemilihan

Halaman mulai pemilihan adalah tampilan *detail* pemilihan namun terdapat *button* yang berbeda dari detail pemilihan yang sebelumnya yaitu *button* Mulai Pemilihan.

|  |
| --- |
|  |

Gambar 4. 28 Tampilan Konfirmasi Mulai Pemilihan

Gambar 4.28 menunjukkan *modal* yang muncul ketika menekan *button* “Mulai Pemilihan” pada halaman detail pemilihan. Jika menekan “Cancel” maka tidak akan terjadi apa-apa sedangkan menekan tombol “OK” akan melakukan *http request* ke *backend* untuk memulai pemilihan.

|  |
| --- |
|  |

Gambar 4. 29 Tampilan Pesan Sukses

Gambar 4.29 menampilkan pesan “Pemilihan berhasil dimulai” ketika sukses melakukan *http request* ke *backend* untuk memulai pemilihan.

### Vote

Halaman *vote* adalah tampilan pada sisi *Voter* yang sudah terdaftar dalam sebuah pemilihan untuk memilih seorang kandidat. Terdapat daftar kandidat serta *button* untuk memilih.

|  |
| --- |
|  |

Gambar 4. 30 Tampilan *Vote* Sebuah Pemilihan

Gambar 4.30 menunjukkan tampilan halaman untuk memilih seorang kandidat. Terdapat *card* untuk masing-masing kandidat. Setiap *card* memiliki Nomor Urut, Nama, Visi, *button* untuk melihat Misi dan Pengalaman serta *button* untuk Memilih.

|  |
| --- |
|  |

Gambar 4. 31 Tampilan Konfirmasi Memilih Kandidat

Gambar 4.31 menunjukkan *modal* yang muncul untuk melakukan konfirmasi terhadap pemilihan kandidat. Jika memilih “Cancel” maka tidak akan terjadi apa-apa sedangkan menekan “OK” akan melakukan *http request* ke *backend* untuk memilih kandidat.

### Menghentikan Pemilihan

Halaman menghentikan pemilihan adalah tampilan pada sisi *Election Authority* untuk menghentikan sebuah pemilihan yang telah berjalan.

|  |
| --- |
|  |

Gambar 4. 32 Tampilan Konfirmasi Menghentikan Pemilihan

Gambar 4.32 menunjukkan modal yang muncul ketika menekan *button* “Hentikan Pemilihan” pada tampilan detail pemilihan. Jika menekan “Cancel” tidak akan terjadi apa-apa sedangkan menekan “OK” akan melakukan *http request* ke *backend* untuk menghentikan pemilihan.

|  |
| --- |
|  |

Gambar 4. 33 Tampilan Pesan Sukses

Gambar 4.33 menampilkan pesan “Pemilihan berhasil dihentikan” ketika sukses melakukan *http request* ke *backend* untuk menghentikan pemilihan.

### Melihat Hasil Pemilihan

Halaman hasil pemilihan adalah tampilan dari sisi *Voter* untuk melihat hasil dari pemilihan yang telah diikuti sebelumnya.

|  |
| --- |
|  |

Gambar 4. 34 Tampilan Hasil Pemilihan

Gambar 4.34 menunjukkan tampilan untuk melihat hasil dari sebuah pemilihan. Tampilan ini menyerupai tampilan *Vote* namun terdapat perbedaan yaitu terdapat *card* tambahan untuk melihat pemenang pemilihan serta *button* pada *card* kandidat diganti menjadi jumlah suara yang diperoleh.

## Implementasi Rancangan API

Implementasi rancangan *API* merupakan implementasi spesifikasi *request response* ketika melakukan komunikasi dengan aplikasi *backend*. Semua proses bisnis sistem voting elektronik termasuk penyimpanan data ke *database* dan komunikasi dengan *node* *blockchain* dilakukan oleh aplikasi *backend.*

### Membuat Pemilihan

Sebuah pemilihan dibuat dengan cara melakukan *http request* dengan metode POST ke *endpoint* /election-authority/election. *Request body* yang dibutuhkan adalah name, description, start dan end.

|  |
| --- |
|  |

Gambar 4. 35 *HTTP Request* untuk Membuat Pemilihan

Gambar 4.35 menunjukkan *http request* yang digunakan untuk membuat pemilihan Pemira HMTI 2020. *Format* tanggal yang digunakan pada *field* start dan end adalah *date string* yang disimpan dalam *database* yaitu YYYY-MM-DD. *Token* yang digunakan pada *header* Authorization berasal dari *credentials* yang dimiliki oleh *Election Authority* yaitu Pemira HMTI.

|  |
| --- |
|  |

Gambar 4. 36 *HTTP Response* Pembuatan Pemilihan

Gambar 4.36 menunjukkan *http response* yang diberikan ketika sukses membuat sebuah pemilihan. Pesan sukses yang diberikan adalah Sukses Membuat Pemilihan. Data yang ditampilkan memiliki tambahan beberapa *field* dari *request* yaitu id, status dan ea. Yang dimaksud *field* ea adalah *Election Authority* yang membuat pemilihan tersebut.

### Menambahkan Kandidat

Kandidat suatu pemilihan ditambahkan dengan cara melakukan *http request* dengan metode POST ke *endpoint* /election-authority/add-candidate/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan id dari pemilihan yang akan ditambahkan kandidat.

|  |
| --- |
|  |

Gambar 4. 37 *HTTP Request* untuk Menambahkan Kandidat

Gambar 4.37 menunjukkan *http request* yang digunakan untuk menambahkan kandidat pada pemilihan dengan id 16. Karena hubungan antara kandidat dengan misi dan pengalaman bersifat *one-to-many* sehingga nilai dari *field* misi dan pengalaman adalah array of string.

|  |
| --- |
|  |

Gambar 4. 38 *HTTP Response* Penambahan Kandidat

Gambar 4.38 menunjukkan *http response* yang diberikan ketika sukses menambahkan kandidat pada sebuah pemilihan. Pesan sukses yang diberikan adalah Sukses Menambahkan Kandidat. *Field* nameSlug adalah nama kandidat yang diubah menjadi kebab-case. Hal ini diperlukan untuk identifikasi kandidat pada smart contract nantinya.

### Deploy Pemilihan

Sebuah *smart contract* bisa digunakan apabila sudah di-*deploy* ke jaringan Ethereum untuk mendapatkan *address*. *Address* digunakan untuk mengidentifikasi setiap *smart contract* yang terdapat pada jaringan Ethereum. Jadi walaupun *source code* *smart contract* sama akan menghasilkan *address* yang berbeda setiap *deployment*. Sebuah pemilihan di-*deploy* ke dengan cara melakukan *http request* dengan metode POST ke *endpoint* /super-admin/deploy-election/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan di-*deploy.*

|  |
| --- |
|  |

Gambar 4. 39 *HTTP Request Deploy* Pemilihan

Gambar 4.39 menunjukkan *http request* yang digunakan untuk melakukan *deployment smart contract* pemilihan ke jaringan Ethereum. *Token* yang digunakan pada *header* Authorization berasal dari *credentials* *Super Admin* karena hanya *Super Admin* yang berhak melakukan *deployment*.

|  |
| --- |
|  |

Gambar 4. 40 *HTTP Response Deploy* Pemilihan

Gambar 4.40 menunjukkan *http response* yang diberikan ketika sukses melakukan *deployment smart contract* ke jaringan Ethereum. Pesan Sukses yang diberikan adalah Sukses Deploy Pemilihan. *Field* address adalah *address* dari *smart contract* untuk pemilihan yang di-*deploy*.

### Mengikuti Pemilihan

*User* dengan *role Voter* dapat ditambahkan sebagai pemilih dalam sebuah pemilihan dengan cara melakukan *http request* dengan metode POST ke *endpoint* /voter/join/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan diikuti.

|  |
| --- |
|  |

Gambar 4. 41 *HTTP Request* untuk Mengikuti Pemilihan

Gambar 4.41 menunjukkan *http request* yang digunakan untuk mengikuti sebuah pemilihan dengan id 16. *Token* yang digunakan pada *header* Authorization berasal dari *credentials Voter*.

|  |
| --- |
|  |

Gambar 4. 42 *HTTP Response* Mengikuti Pemilihan

Gambar 4.42 menunjukkan *http response* yang diberikan ketika sukses mengikuti sebuah pemilihan. Pesan sukses yang diberikan adalah Sukses Mengikuti Pemilihan.

### Daftar Peserta Pemilihan

*User* dengan *role Election Authority* dapat melihat daftar perserta yang mengikuti sebuah pemilihan.dengan cara melakukan *http request* dengan metode GET ke *endpoint* /election-authority/election-participant/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan dibutuhkan.

|  |
| --- |
|  |

Gambar 4. 43 *HTTP Request* Daftar Peserta Pemilihan

Gambar 4.43 menunjukkan *http request* yang digunakan untuk menampilkan daftar peserta yang mengikuti sebuah pemilihan dengan id 16. *Token* yang digunakan pada *header* Authorization adalah *credentials Election Authority*.

|  |
| --- |
|  |

Gambar 4. 44 *HTTP Response* Daftar Peserta Pemilihan

Gambar 4.44 menunjukkan *http response* yang diberikan ketika sukses menampilkan daftar peserta yang mengikuti pemilihan. Karena sebuah pemilihan dapat diikuti oleh banyak *user* maka tipe data dari *field* participant adalah array of object.

### Detail Pemilihan

*User* dengan *role Voter* dapat melihat detail dari sebuah pemilihan dengan cara melakukan *http request* dengan metode GET ke *endpoint* /voter/election-detail/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang dibutuhkan.

|  |
| --- |
|  |

Gambar 4. 45 *HTTP Request* Detail Pemilihan

Gambar 4.45 menunjukkan *http request* yang digunakan untuk mendapatkan detail sebuah pemilihan dengan id 16. *Token* yang digunakan pada *header* Authorization adalah *credentials Voter*.

|  |
| --- |
|  |

Gambar 4. 46 *HTTP Response* Detail Pemilihan

Gambar 4.46 menunjukkan *http response* yang diberikan ketika sukses mendapatkan detail dari sebuah pemilihan. Karena sebuah pemilihan terdapat lebih dari satu kandidat maka *field* candidates memiliki tipe data array of object.

### Memulai Pemilihan

*User* dengan *role Election Authority* dapat memulai sebuah pemilihan yang telah di-*deploy* ke jaringan Ethereum dengan cara melakukan *http request* dengan metode POST ke *endpoint* /election-authority/start-election/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan dimulai.

|  |
| --- |
|  |

Gambar 4. 47 *HTTP Request* Memulai Pemilihan

Gambar 4.47 menunjukkan *http request* yang digunakan untuk memulai sebuah pemilihan dengan id 16. *Token* yang digunakan pada *header* Authorization adalah *credentials Election Authority*.

|  |
| --- |
|  |

Gambar 4. 48 *HTTP Response* Memulai Pemilihan

Gambar 4.48 menunjukkan *http response* yang diberikan ketika sukses memulai sebuah pemilihan. Data yang diberikan adalah pesan sukses yaitu Sukses Memulai Pemilihan dan nama pemilihan yang berhasil dimulai yaitu Pemira HMTI 2020.

### Vote

*User* dengan *role Voter* dapat memilih seorang kandidat pada sebuah pemilihan dengan cara melakukan *http request* dengan metode POST ke *endpoint* /voter/vote.

|  |
| --- |
|  |

Gambar 4. 49 *HTTP Request* Pemilihan Kandidat

Gambar 4.49 menunjukkan *http request* yang digunakan untuk memilih seorang kandidat. *Request body* yang digunakan ada 2 yaitu election\_id dan candidate\_id. *Field* election\_id berisi nilai id dari sebuah pemilihan dan *field* candidate\_id berisi nilai id dari kandidat yang dipilih. *Token* yang digunakan pada *header* Authorization adalah *credentials Voter*.

|  |
| --- |
|  |

Gambar 4. 50 *HTTP Response* Pemilihan Kandidat

Gambar 4.50 menunjukkan *http response* pemilihan kandidat pada sebuah pemilihan. Data yang diberikan adalah pesan sukses yaitu Sukses Memberikan Suara serta nama pemilihan yang diikuti yaitu Pemira HMTI 2020.

### Menghentikan Pemilihan

*User* dengan *role Election Authority* dapat menghentikan pemilihan yang telah dimulai dengan cara melakukan *http request* dengan metode POST ke *endpoint* /election-authority/end-election/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan dihentikan.

|  |
| --- |
|  |

Gambar 4. 51 *HTTP Request* Menghentikan Pemilihan

Gambar 4.51 menunjukkan *http request* yang digunakan untuk menghentikan sebuah pemilihan dengan id 16. *Token* yang digunakan pada *header* Authorization adalah *credentials Election Authority*.

|  |
| --- |
|  |

Gambar 4. 52 *HTTP Response* Menghentikan Pemilihan

Gambar 4.52 menunjukkan *http response* yang diberikan ketika sukses menghentikan sebuah pemilihan. Data yang diberikan adalah pesan sukses yaitu Sukses Menghentikan Pemilihan serta nama pemilihan yang dihentikan yaitu Pemira HMTI 2020.

### Melihat Hasil Pemilihan

Hasil sebuah pemilihan dapat dilihat dengan cara melakukan *http request* dengan metode GET ke *endpoint* /voter/ended-election-detail/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang dibutuhkan.

|  |
| --- |
|  |

Gambar 4. 53 *HTTP Request* Hasil Pemilihan

Gambar 4.53 menunjukkan *http request* yang digunakan melihat hasil pemilihan dengan id 16. *Token* yang digunakan pada *header* Authorization adalah *credentials Voter*.

|  |
| --- |
|  |

Gambar 4. 54 *HTTP Response* Hasil Pemilihan

Gambar 4.54 menunjukkan *http response* yang diberikan ketika sukses mendapatkan hasil sebuah pemilihan. *Response* yang diberikan mirip seperti *response* detail pemilihan. Yang berbeda adalah pada *object candidate* terdapat *field* vote\_count yang berisi nilai total suara yang didapatkan oleh seorang kandidat. *Field* winner yang berisi nilai nama dari kandidat yang memenangkan sebuah pemilihan.

## Pengujian Sistem Voting Elektronik

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 pada *backend* dan *node* Ethereum.

### 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 4. 55 Struktur *File* Pengujian

Gambar 4.55 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.ts adalah 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*.

1. Pengujian *API 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 4. 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*.

|  |
| --- |
|  |

Gambar 4. 56 Hasil Pengujian *Endpoint Super Admin*

Gambar 4.56 menunjukkan bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Super Admin* berhasil dijalanakan.

1. Pengujian *API 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 4. 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*.

|  |
| --- |
|  |

Gambar 4. 57 Hasil Pengujian *Endpoint Election Authority*

Gambar 4.57 menunjukkan bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Election Authority* berhasil dijalanakan.

1. Pengujian *API 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 4. 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*.

|  |
| --- |
|  |

Gambar 4. 58 Hasil Pengujian *Endpoint Voter*

Gambar 4.58 menunjukkan bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Voter* berhasil dijalanakan.

### Pengujian Integritas Data Blockchain

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.

1. *Node* Openethereum

*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*.

1. *Block Explorer*

*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*.

1. Ethereum *Decoder*

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>.

1. *Smart Contract*

*Smart Contract* yang digunakan adalah *smart contract* dengan logika yang sederhana dimana hanya terdapat dua fungsi yang bisa digunakan yaitu getPerson dan setPerson.

|  |
| --- |
| contract OneTime {  bool private isLocked;  string private person;  constructor() {  isLocked = false;  person = "";  }  modifier unlocked() {  require(isLocked == false, "Contract Locked");  \_;  }  function getPerson() public view returns (string memory) {  return person;  }  function setPerson(string memory \_person) public unlocked {  person = \_person;  isLocked = true;  }  } |

Kode Program 4. 9 *Smart Contract*

Kode Program 4.9 merupakan *source code* dari *smart contract* yang digunakan pada pengujian ini. *Source code* tersebut hanya memiliki dua buah *function* yaitu getPerson dan setPerson.

1. Tampilan

Tampilan tidak terlalu dibutuhkan karena untuk berinteraksi dengan *ethereum client* hanya membutuhkan *library javascript ethers.js*. Menggunakan *runtime javascript* seperti *node* sudah bisa berinteraksi dengan *ethereum client* namun untuk memudahkan penjelasan pengujian ini akan menggunakan *tech stack* yang umum yaitu *html*, *css* dan *javascript*.

|  |
| --- |
|  |

Gambar 4. 59 Tampilan

Gambar 4.59 menunjukkan tampilan yang akan digunakan untuk berinteraksi dengan *smart contract* yang terdapat pada *blockchain*.

1. *Rlpdump*

*Rlpdump* adalah *tools* yang dapat digunakan untuk melakukan *encoding* maupun *decoding* struktur data *rlp*. *RLP(Recursive Length Prefix)* adalah standar digunakan oleh *ethereum* untuk menyimpan struktur data *array* dalam sebuah *hex*.

|  |
| --- |
|  |

Gambar 4. 60 *Rlpdump Help*

Gambar 4.60 menunjukkan perintah yang disediakan serta deskripsi singkat tentang apa saja yang bisa dilakukan oleh *rlpdump*.

1. *Ldb*

*Ldb* dideskripsikan sebagai *RocksDB Administrative Tools* yaitu *tools CLI* yang dapat melakukan operasi *read* dan *write* ke *database RocksDB*. *Ldb* berperan penting dalam pengujian ini karena *ldb* adalah satu-satunya cara untuk berinteraksi dengan data yang tersimpan pada *database RocksDB*.

|  |
| --- |
|  |

Gambar 4. 61 *Ldb Help*

Gambar 4.61 menunjukkan perintah yang disediakan serta deskripsi singkat tentang apa saja yang bisa dilakukan oleh *ldb*. Perintah yang dibutuhkan dalam pengujian ini adalah perintah *get* dan perintah *put*.

1. Penggunaan *Smart Contract*

*Smart Contract* yang digunakan untuk pengujian memiliki skenario penggunaan yang sederhana yaitu mendapatkan nama ketika selesai *deploy smart contract*, menentukan nama, mendapatkan nama setelah nama ditentukan dan mencoba kembali untuk menentukan nama.

|  |
| --- |
|  |

Gambar 4. 62 Tampilan Awal

Gambar 4.62 menunjukkan tampilan awal dari projek yang digunakan untuk melakukan interaksi dengan *smart contract*. Nama projek ini adalah *Mini Smart Contract*. Seperti yang terlihat karena belum dilakukan *deployment*, bagian *status* dan *address* masih kosong.

|  |
| --- |
|  |

Gambar 4. 63 *Deployed Smart Contract*

Gambar 4.63 menunjukkan tampilan ketika *smart contract* telah di-*deploy* pada jaringan Ethereum yang telah disiapkan. Pada bagian *status* akan menampilkan *Deployed* dan pada bagian *address* akan menampilkan *address* dari *smart contract* tersebut. Untuk bisa menggunakan fungsi yang terdapat pada *smart contract*, *ethers.js* harus dihubungkan dengan *smart contract* tujuan menggunakan *address* yang telah didapatkan.

|  |
| --- |
|  |

Gambar 4. 64 *Connected Smart Contract*

Gambar 4.64 menunjukkan tampilan ketika berhasil terhubung dengan *smart contract* yang diinginkan. Terdapat dua tombol yaitu *Get Person* dan *Set Person*. Tombol *Get Person* akan memanggil fungsi *getPerson* pada *smart contract*. Sedangkan tombol *Set Person* akan memanggil fungsi *setPerson* pada *smart contract* dengan menggunakan nilai dari *input* *Person Name* sebagai parameter.

|  |
| --- |
|  |

Gambar 4. 65 Tombol Get Person

Gambar 4.65 menunjukkan tampilan ketika tombol *Get Person* digunakan tanpa menentukan nama menggunakan tombol *Set Person*. Hasil yang terlihat adalah pada bagian pesan terdapat *timestamp* tanpa nilai.

|  |
| --- |
|  |

Gambar 4. 66 Tombol Set Person

Gambar 4.66 menunjukkan tampilan ketika menggunakan tombol *Set Person*. Tidak ada *feedback* dari tampilan namun proses yang berjalan adalah *ethers.js* akan mengambil nilai dari *input field* dan menggunakannya sebagai parameter ketika memanggil fungsi *setPerson* pada *smart contract*.

|  |
| --- |
|  |

Gambar 4. 67 Tombol Get Person

Gambar 4.67 menunjukkan tampilan ketika menggunakan tombol *Get Person* setelah menentukan nama. Hasil yang terlihat adalah terdapat *timestamp* dengan nilai yang sama seperti apa yang telah ditentukan menggunakan tombol *Set Person*.

|  |
| --- |
|  |

Gambar 4. 68 Tombol Set Person

Gambar 4.68 menunjukkan tampilan ketika menggunakan tombol *Set Person* lagi setelah sebelumnya sudah menentukan nama. Hasil yang didapatkan adalah akan ada *alert* dengan pesan *error* karena fungsi *setPerson* pada *smart contract* memang dibiat agar hanya bisa digunakan sekali saja.

1. Pengujian

*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 4. 69 *Block Explorer*

Gambar 4.69 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 4. 70 Detail *Block* 2

Gambar 4.70 menunjukkan tampilan dari 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 4. 71 Ethereum *Decoder*

Gambar 4.71 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 4. 72 *Param Generator*

Gambar 4.72 menunjukkan tampilan untuk membuat *input parameter* untuk *blockchain*. Pada pengujian ini akan berusaha mengubah *string parameter* yang tersimpan dari “Dirga” menjadi “Yasa”.

|  |
| --- |
|  |

Gambar 4. 73 Ethereum *Decoder*

Gambar 4.73 menunjukkan hasil *decoding input parameter* yang dihasilkan oleh *param generator*. Seperti yang ditampilkan, *string parameter* dari *input parameter* tersebut adalah “Yasa”.

|  |
| --- |
|  |

Gambar 4. 74 *RocksDB Administrative Tools*

Gambar 4.74 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 4. 75 Struktur Data *RLP*

Gambar 4.75 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 4. 76 *Encoding RLP*

Gambar 4.76 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 4. 77 *Update Database RocksDB*

Gambar 4.77 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.

# BAB V PENUTUP

BAB V membahas mengenai kesimpulan yang mengacu pada rumusan masalah dan tujuan penelitian beserta saran yang diberikan peneliti untuk kepentingan pengembangan aplikasi selanjutnya.

## Kesimpulan

Simpulan yang dapat diperoleh dari pengembangan Sistem Voting Elektronik berbasis Ethereum *smart contract* adalah sebagai berikut:

* 1. 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.
  2. 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.
  3. 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.

## Saran

Penelitian sistem ­e-voting berbasis Ethereum *smart contract* ini masih jauh dari kata sempurna dan masih banyak yang 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 *smart 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.

# DAFTAR PUSTAKA

Adhi, R. A. & Harjono, 2014. Rancang Bangun Sistem Informasi E-Voting Berbasis SMS. *JUITA,* III(2), pp. 85-93.

Angular, 2010. *Angular.* [Online] Available at: https://angular.io/  
[Accessed 31 March 2021].

Ardilla, R., 2018. Rancang Bangun Sistem E – Voting Dengan Metode Enkripsi Blockchain Di Kota Mojokerto. *Repository Universitas Islam Majapahit.*

Code, V. S., 2021. *Visual Studio Code.* [Online] Available at: https://code.visualstudio.com [Accessed 31 March 2021].

Copes, F., 2019. *The Next.js Handbook.* s.l.:s.n.

Derizal, 2011. Panduan Lengkap PHP, Ajax, jQuery. *PHP Ajax Javascript jQuery Tutorial Indonesia,* pp. 27-39.

Google, C., 2021. *Google Cloud.* [Online] Available at: https://cloud.google.com/  
[Accessed 31 March 2021].

Johari, R. et al., 2020. SEVA: Secure E-Voting Application in Cyber Physical System. *Cyber-Physical Systems,* pp. 1-31.

Kadir, A., 2008. *Tuntunan Praktis Belajar Database Menggunakan MySQL.* Yogyakarta: Andi Offset.

Kurnia Hu, S. D., Palit, H. N. & Handojo, A., 2019. Implementasi Blockchain: Studi Kasus e-Voting. *Jurnal Infra,* VII(1).

Mysliwiec, K., 2017. *NestJS.* [Online] Available at: https://nestjs.com/  
[Accessed 31 March 2021].

Petersen, K., Wohlin, C. & Baca, D., 2009. *The Waterfall Model in Large-Scale Development.* Findland, 10th International Conference.

Pierro, M. D., 2017. What Is the Blockchain?. *Computing in Science & Engineering,* Volume 19, pp. 92-95.

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

Rahardja, U., Aini, Q., Yusup, M. & Edliyanti, A., 2020. Penerapan Teknologi Blockchain Sebagai Media Pengamanan Proses Transaksi E-Commerce. *Journal of Computer Engineering System and Science,* V(1), pp. 28--32.

Ridwan, M., Arifin, Z. & Yulianto, 2016. Rancang Bangun E-Voting Menggunakan Keamanan RSA. *Jurnal Informatika Mulawarman.*

Setia, T. E. H. & Susanto, A., 2019. Smart Contract Blockchain pada E-Voting. *Jurnal Informatika Upgris ,* V(2), pp. 188-191.

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

Web3JS, 2016. *web3.js.* [Online] Available at: https://web3js.readthedocs.io/  
[Accessed 31 March 2021].

Wilson, J. R., 2013. *Node.js the Right Way.* 1st ed. United States of America: The Pragmatic Programmers, LLC..