

PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG



# **Pengembangan Aplikasi Berbasis *Blockchain* (DApps)**

**Kasus: BlockTrade Chain Untuk Manajemen Ekspor-Import**

Penulis:

13521051 - Manuella Ivana Uli Sianipar

13521135 - Nicholas Liem

13521162 - Antonio Natthan Krishna

**IF4035 - Blockchain**

**2024**

# BAB I

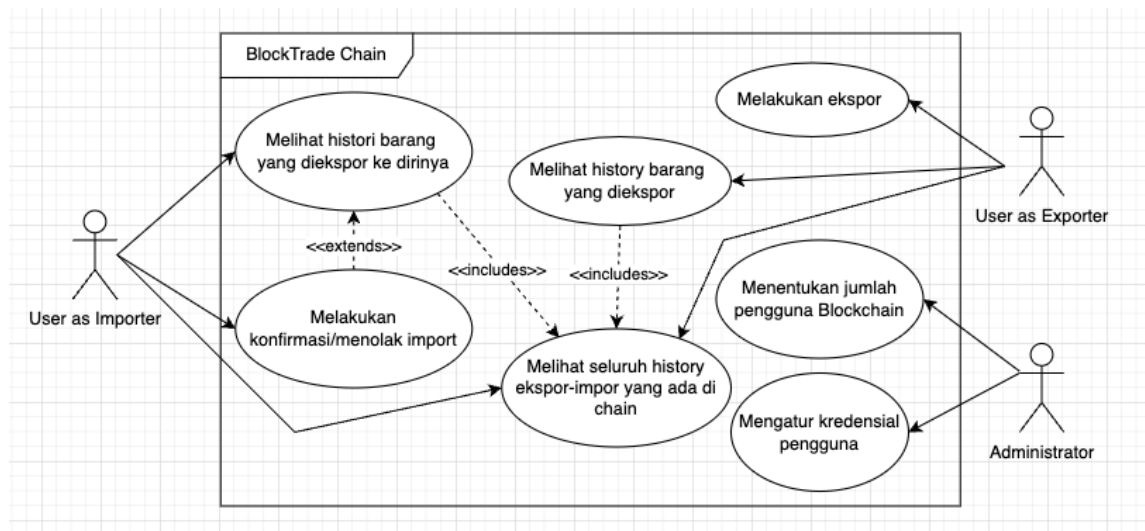
## DESKRIPSI APLIKASI

### I. *Problem Statement*

Sistem manajemen tradisional untuk supply chain dalam pengimporan barang masih memiliki beberapa inefisiensi, kurangnya transparansi, serta trust antar para pemegang kepentingan. Sangat mungkin untuk terjadinya keterlambatan, penipuan, dan error dikarenakan pemrosesan data yang manual sehingga bukan hanya itu tetapi malah tercipta suatu sistem yang terisolasi yang juga memiliki traceability yang terbatas.

Terlebih lagi, manajemen pembayaran dan obligasi kontrak biasanya membutuhkan intermediaries atau penengah sehingga menambah waktu dan biaya untuk melakukan proses tersebut. Kekurangan-kekurangan ini disebabkan karena sistem tersebut tidak terintegrasi dengan baik oleh sebab itu dibutuhkan suatu sistem baru yang menjamin kebutuhan pemegang kepentingan terutama kepercayaan antara mereka.

### II. *Use Case*



Gambar 1.1. Diagram *Use Case*

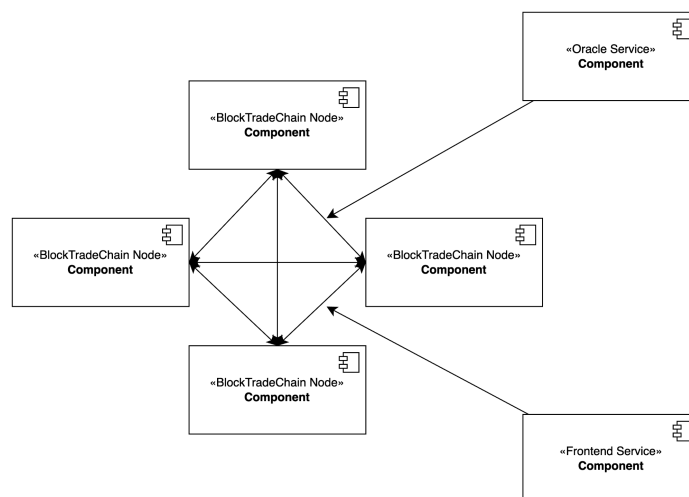
Sistem ini dirancang untuk mengelola proses ekspor-impor secara digital menggunakan teknologi blockchain. Dengan demikian, semua transaksi dan data terkait dapat dilacak, diverifikasi, dan disimpan secara aman dan transparan. Sistem ini melibatkan beberapa aktor,

1. User: Pengguna, setiap pengguna dapat melakukan ekspor-impor di dalam aplikasi.
  - a. User as Exporter: Pengguna yang memiliki peran sebagai eksportir.
  - b. User as Importer: Pengguna yang memiliki peran sebagai importir.
2. Administrator: Pengguna dengan hak akses penuh terhadap sistem.

Aplikasi dapat membantu aktor tersebut untuk melakukan beberapa tugas utama,

1. Melakukan Ekspor: Use case ini memungkinkan pengguna yang berperan sebagai eksportir untuk memulai proses ekspor barang.
2. Melakukan Konfirmasi/Menolak Impor: Use case ini memungkinkan pengguna yang berperan sebagai importir untuk memberikan persetujuan atau penolakan terhadap suatu impor.
3. Melihat Histori: Use case ini memungkinkan pengguna untuk kembali melihat seluruh transaksi yang tercatat pada *blockchain*.
4. Menetapkan Jumlah Pengguna Blockchain: Use case ini merupakan hak eksklusif administrator untuk menentukan berapa banyak pengguna yang dapat terhubung ke sistem.
5. Mengatur Kredensial Pengguna: Use case ini juga merupakan hak eksklusif administrator untuk mengelola data pengguna, seperti menambahkan, mengubah, atau menghapus pengguna.

### III. High Level Design



Gambar 1.2. High Level Design BlockTrade Chain

Interaksi antar komponen terlibat dan peran tiap komponen.

Desain sistem BlockTradeChain terdiri dari beberapa komponen utama yang saling berinteraksi dalam arsitektur terdistribusi. Komponen ini mencakup frontend, jaringan

blockchain, dan layanan oracle, yang masing-masing memiliki peran dan tanggung jawabnya. Jaringan blockchain tentunya digunakan sebagai *stable storage* untuk penyimpanan data dan fungsi *frontend* adalah menghubungkan pengguna dengan antarmuka web3 untuk berkomunikasi dengan jaringan blockchain. Fungsi oracle di sini adalah untuk memperbarui sebuah data publik yakni nilai tukar uang yang digunakan pada saat proses ekspor impor.

## BAB II

### KONTRAK PINTAR

#### I. Gambaran Besar Desain Kontrak Pintar

Kontrak Pintar (*Smart Contract*) adalah program yang berjalan di atas blockchain yang secara otomatis mengeksekusi transaksi atau instruksi sesuai dengan logika yang telah diimplementasikan. Dalam sistem ini, kontrak pintar digunakan untuk mengelola *supply chain* barang antar pihak, meliputi ekspor dan konfirmasi impor. Fungsi kontrak pintar ini adalah memastikan transparansi dan keamanan dalam pengelolaan transaksi logistik, termasuk mengelola nilai tukar antar mata uang dan status transaksi.

#### II. Deskripsi Properti

Kontrak pintar ini memiliki beberapa properti:

1. items (mapping)  
Menyimpan informasi barang berdasarkan hash transaksi.
2. userInbox (mapping)  
Daftar hash transaksi yang menunggu aksi dari pengguna (inbox).
3. userAssets (mapping)  
Daftar hash transaksi yang sudah dikonfirmasi oleh pengguna.
4. exchangeRates (mapping)  
Menyimpan nilai tukar antar mata uang.
5. oracle (address)  
Alamat oracle yang membantu memperbarui nilai tukar.
6. transactionHashes (bytes32[])  
Daftar semua hash transaksi yang telah terjadi.

#### III. Deskripsi *Method*

| Method             | Fungsi   |
|--------------------|--|
| setOracle          | Mengatur alamat oracle baru  |
| getInbox           | Mengembalikan daftar hash transaksi yang ada di inbox pengguna       |
| getAssets          | Mengembalikan daftar hash transaksi yang sudah menjadi aset pengguna |
| updateExchangeRate | Memperbarui nilai tukar antara dua mata uang tertentu                |

|                       |  |
|-----------------------|--|
| getExchangeRate       | Mengambil nilai tukar antara dua mata uang tertentu  |
| exportItem            | Membuat transaksi ekspor baru, mencatat detail barang, penerima, dan status awal                 |
| confirmItem           | Mengonfirmasi penerimaan barang oleh penerima, mengubah status transaksi menjadi "IMPORTED"      |
| denyItem              | Mengonfirmasi barang tidak diterima oleh penerima, mengubah status transaksi menjadi "CANCELLED" |
| getItemDetails        | Mengambil detail barang berdasarkan hash transaksi   |
| addToInbox            | Menambahkan hash transaksi ke inbox pengguna   |
| removeFromInbox       | Menghapus hash transaksi dari inbox pengguna   |
| addToAsset            | Menambahkan hash transaksi ke aset pengguna  |
| calculateExchangeRate | Menghitung nilai tukar antar mata uang berdasarkan kurs USD sebagai mata uang referensi          |

#### IV. Optimasi Algoritma Kontrak Pintar

Metode `removeFromInbox` dibuat untuk menghapus transaksi dari daftar inbox pengguna. Algoritma ini berfungsi untuk menghapus elemen di array dinamis Solidity. Karena Solidity tidak mendukung penghapusan elemen secara langsung dari array, algoritma ini menggunakan pendekatan *swap-and-pop*, yang merupakan metode hemat gas. Cara kerja algoritma ini adalah dengan menukar elemen yang ingin dihapus ke akhir array, lalu penghapusan elemen dilakukan dengan fungsi `pop`. Pendekatan ini memastikan bahwa array tetap terkompresi tanpa adanya elemen kosong.

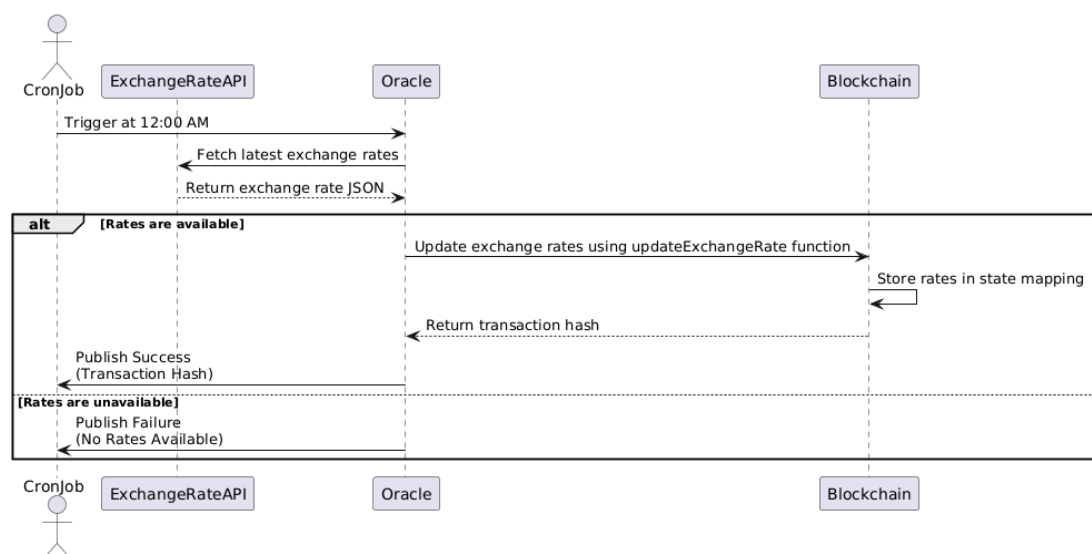
## BAB III

### ORACLE

#### I. Deskripsi Umum

Oracle pada sistem ini berfungsi sebagai *source of truth* yang dipakai dalam penggunaan kegiatan ekspor-impor. Kami memilih peran oracle untuk memegang data kebenaran dari *exchange rate* atau nilai tukar uang antara eksportir dan importir. Pengambilan data ini dilakukan terhadap API eksternal *ExchangeRateAPI*. Tetapi, karena penggunaannya berbayar, maka kami menggunakan data *mock* yang berisi contoh respons API dari *ExchangeRateAPI* jika benar-benar dipanggil. Selain itu, hanya ada satu endpoint jika pengguna hendak ingin melihat atau memperbarui data terbaru yakni `PUT /exchangeRate`.

#### II. Mekanisme Pengambilan Data dari API Eksternal



Rencana awalnya pengambilan data ini akan dilakukan setiap hari pada pukul 12 malam menggunakan *cron job*. Mekanisme ini akan memanggil server *oracle* secara otomatis untuk melakukan pemanggilan API ke *ExchangeRateAPI* kemudian datanya akan diparse dan dilakukan pembaruan ke dalam data yang disimpan pada *Blockchain* pada waktu itu juga.

#### III. Integrasi dengan Kontrak Pintar

Integrasi dilakukan ketika pemanggilan data dari API eksternal diterima oleh server oracle, kemudian melalui *interface* yang dihasilkan oleh abigen terhadap ABI

(Application Binary Interface) dari hasil *deployment* kontrak, data disimpan menggunakan fungsi yang hanya bisa diakses oleh Oracle dalam kontrak pintar. Fungsi tersebut adalah `UpdateExchangeRate(fromCurrency, toCurrency, rate)`. Fungsi ini akan memperbarui data di Blockchain khususnya pada sebuah *public state variable* bernama *exchangeRates* yang merupakan *nested mapping* berbentuk seperti ini: `mapping(string => mapping(string => ExchangeRate))`.

Data ini nantinya akan dipakai ketika terjadi pembuatan permintaan ekspor, di mana data *exchangeRate* akan diisi berdasarkan nilai ini dan bukan nilai dari mana pun karena *exchangeRate* pada *state variable* ini dinyatakan sebagai *truth* karena hanya oracle yang dapat mengubahnya.

#### IV. Kemungkinan Skenario Buruk

Ada beberapa kemungkinan skenario buruk dari penggunaan oracle ini, beberapa di antaranya adalah:

1. Nilai awal yang tidak terdefinisi

Jika nilai awal dari *exchangeRates* untuk suatu pasangan mata uang tidak terdefinisi, maka kontrak pintar tidak dapat memberikan nilai tukar yang valid. Hal ini dapat menyebabkan transaksi gagal atau tidak akurat. Mitigasi yang dilakukan adalah dengan cara: Menetapkan nilai *default* untuk *exchangeRates* (contoh: `rate = 1` untuk mata uang yang tidak ditemukan) agar sistem tetap dapat bekerja meskipun data dari oracle belum tersedia. Gunakan mekanisme *fallback* pada kontrak pintar untuk memberikan peringatan kepada pengguna jika nilai tukar belum diperbarui. Pastikan *cron job* oracle berfungsi dengan baik untuk memperbarui data setiap hari tanpa gagal.

2. Serangan dari sumber data eksternal (dimanipulasi nilainya)

Jika API eksternal *ExchangeRateAPI* dimanipulasi atau menyediakan data yang salah, oracle dapat memperbarui nilai tukar yang tidak akurat ke blockchain, menyebabkan kerugian finansial bagi pengguna.

Mitigasi yang dilakukan adalah dengan cara: verifikasi respons API dengan mekanisme validasi seperti *cross-check* dengan sumber data lain sebelum menulis data ke blockchain. Gunakan penyedia API tepercaya yang memiliki rekam jejak keamanan dan akurasi yang baik.



## **BAB IV**

### **DESKRIPSI IMPLEMENTASI**

#### **I. Platform *Blockchain* yang Digunakan**

Sistem ini menggunakan Ethereum sebagai platform pengembangannya. Beberapa pertimbangan yang mendasarinya adalah:

1. Teknologi kontrak pintar yang sudah matang (*mature*)

Etherum dikenal sebagai pelopor teknologi kontrak pintar dan sampai saat ini dinilai sebagai yang paling matang dan banyak diadopsi oleh banyak orang. Hal ini menjadi alasan yang cukup kuat untuk mengotomatisasi proses seperti pembayaran, eksekusi kontrak, dan mungkin juga penalti.

2. Platform yang sudah diadopsi secara luas

Platform Ethereum sudah diadopsi secara luas dan memiliki komunitas yang sangat besar. Platform ini telah digunakan secara global, tidak hanya untuk kebutuhan publik tetapi juga perusahaan dan pemerintah. Oleh sebab itu, kami memilih Ethereum sebagai platform pengembangan.

3. Desentralisasi dan keamanan

Platform Ethereum memiliki struktur yang terdesentralisasi untuk memastikan keutuhan data dan keamanan. Setiap transaksi diverifikasi oleh banyak node lain untuk memastikan sistem bebas dari *tampering* dan *fraud*. Dalam manajemen *supply chain*, karakteristik sistem yang dibutuhkan adalah kepercayaan dan transparansi oleh sebab itu platform ini sangat sesuai untuk digunakan.

#### **II. *Technology Stack* yang Terlibat**

Tech stack yang kami gunakan untuk pengembangan ini terdiri dari beberapa bagian:

1. Frontend

Pada bagian ini, kami menggunakan vite sebagai framework pengembangan supaya cepat dan efisien. Selain itu kami memanfaatkan web3.js untuk menghubungkan aplikasi dengan blockchain sehingga memungkinkan interaksi dengan kontrak pintar secara langsung dari antarmuka pengguna.

2. Kontrak Pintar

Kontrak pintar dikembangkan menggunakan Solidity (Sol), yakni sebuah bahasa pemrograman utama untuk blockchain Ethereum. Untuk pengelolaan siklus hidup dari kontrak ke dalam blockchain kami menggunakan truffle.

3. Backend / Oracle

Pada bagian ini, kami menggunakan Golang yang terkenal dengan performa tinggi, efisiensi, dan skalabilitasnya. Golang memungkinkan kami untuk membangun backend yang dapat menangani proses integrasi data eksternal ke blockchain melalui mekanisme oracle.

### III. Design Pattern

#### a. Oracle

##### i. Adapter Pattern

Respons dari eksternal diubah dan disesuaikan di server Oracle sebelum akhirnya datanya dimasukkan ke dalam Blockchain.

##### ii. Dependency Injection

Salah satu parameter pertama yang digunakan pada saat mengubah *exchange rate* adalah *auth*. Nilai *auth* diinisialisasi pada program utama kemudian diinjeksi ke dalam fungsi *UpdateExchangeRate*. Guna dari *auth* sendiri adalah sebagai antarmuka oracle untuk berkomunikasi dengan blockchain.

#### b. Kontrak Pintar

##### i. Factory Pattern

Fungsi *exportItem* merupakan fungsi yang mengenkapsulasi proses pembentukan dan inisialisasi objek *item*.

##### ii. Access Control

Ada dua *modifier* yang dipakai dalam kontrak pintar, yakni *onlyOwner* dan *onlyOracle* keduanya menggambarkan tentang desain RBAC (Role Base Access Control) yang diimplementasikan terhadap beberapa fungsi. Contohnya *getInbox* yang diharuskan menggunakan *onlyOwner* untuk mengaksesnya.

##### iii. Observer Pattern

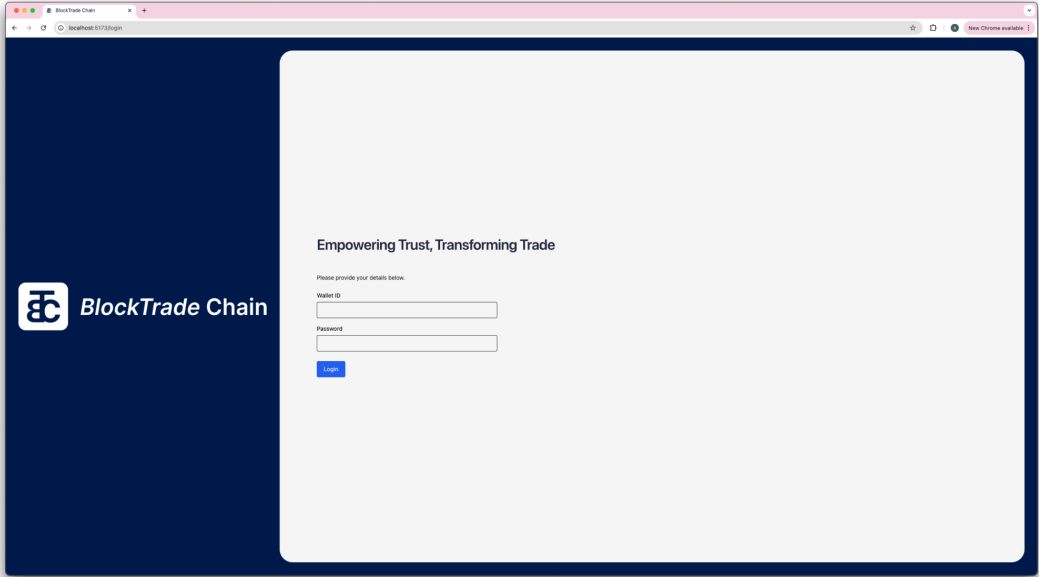
Desain ini digunakan pada kontrak khususnya ketika program mengimplementasikan *event-based paradigm*. Hal ini ditunjukkan dengan adanya penggunaan dari event seperti *ItemExported*, *ExchangeRateUpdated*, *StatusUpdated*.

##### iv. State Machine

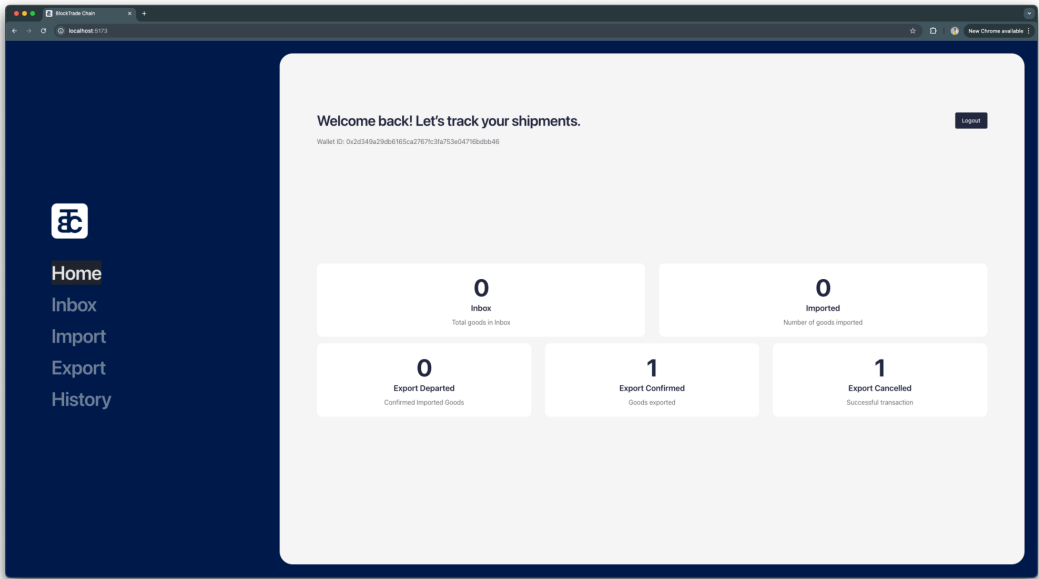
Pada item terdapat *field* bernama status yang digunakan untuk menunjukkan status sebuah *order* atau *item* tersebut dalam proses *supply chain*. Ada tiga status yang digunakan, yakni *EXPORTED*, *IMPORTED*, dan *CANCELLED*. Masing-masing flow dari perubahan status diatur oleh kontrak.

## IV. Tangkapan Layar Aplikasi

### 1. Halaman Login

|   |  |
|---|--|
| Deskripsi   | Halaman untuk memasukkan dan melakukan pengecekan kredensial |
|  |  |

### 2. Halaman Home

|  |  |
|--|--|
| Deskripsi  | Halaman utama dari aplikasi, berisikan data penting dan tombol <i>logout</i> |
|  |  |

### 3. Halaman *Inbox*

|           |   |
|-----------|---|
| Deskripsi | Berisikan data barang yang datang ke pengguna. Pengguna dapat menerima / menolak barang yang datang pada halaman ini. |
|-----------|---|

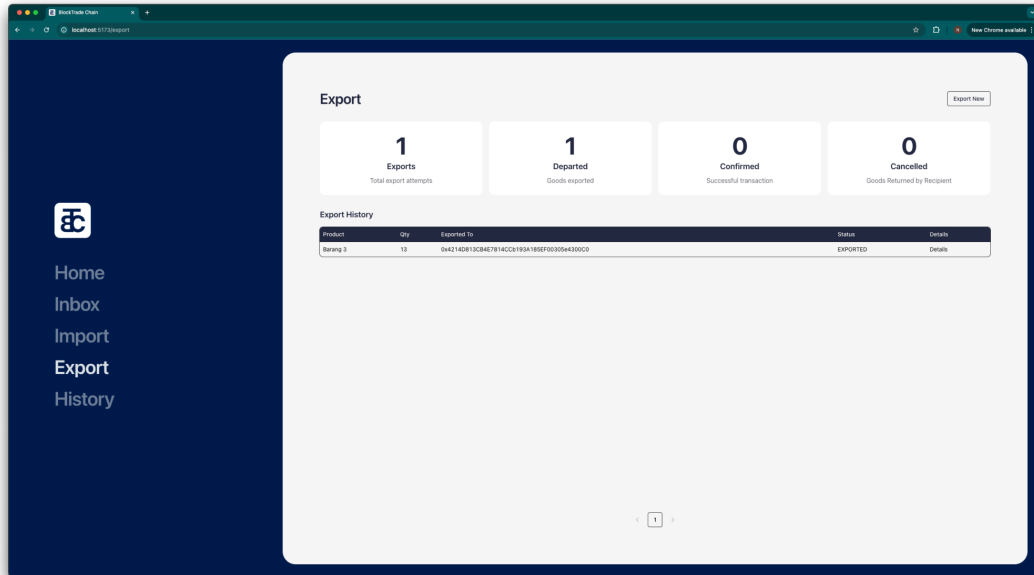
### 4. Halaman Imports

|           |   |
|-----------|---|
| Deskripsi | Berisikan data barang yang telah dikonfirmasi untuk diimpor oleh pengguna |
|-----------|---|

## 5. Halaman Export

### Deskripsi

Berisikan histori seluruh barang yang diekspor oleh pengguna, baik barang yang berhasil diimpor target maupun yang dikembalikan / ditolak oleh target



## 6. Sub Halaman Menambahkan Ekspor

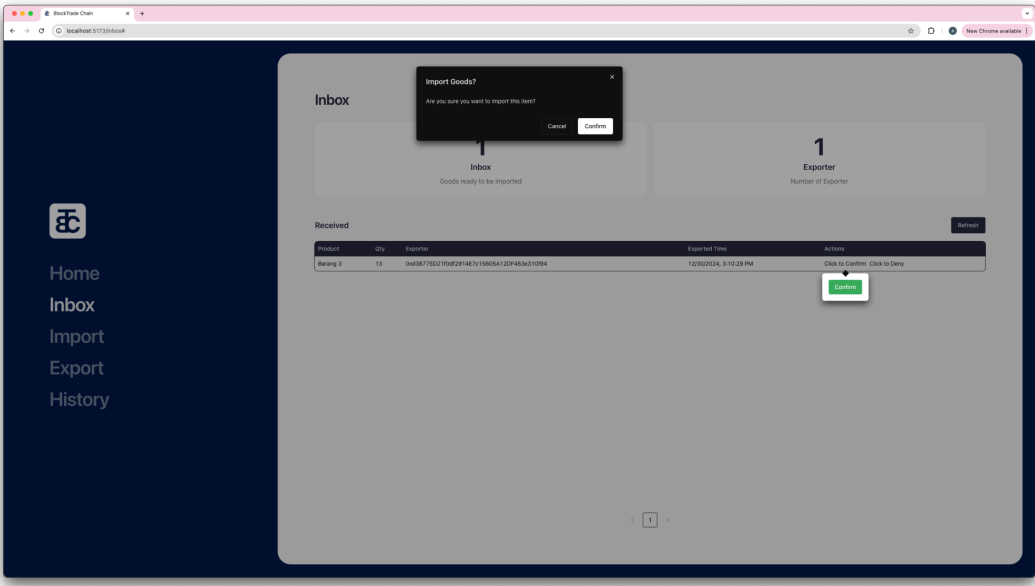
### Deskripsi

Berisikan formulir data barang yang akan diekspor

The screenshot shows a modal form titled 'Product' with a close button (X) in the top right corner. The form contains several input fields and dropdown menus: 'Product name' (text input), 'Quantity' (text input), 'Currency' (dropdown menu showing 'XXX'), 'Value' (text input), 'Target' (dropdown menu showing 'XXX'), 'Conversion Rate' (text input showing '0'), and 'Recipient Address' (text input). At the bottom of the form is a dark blue button labeled 'Export New'. The background is a blurred view of the 'Export' page.

## 7. Sub Halaman Konfirmasi

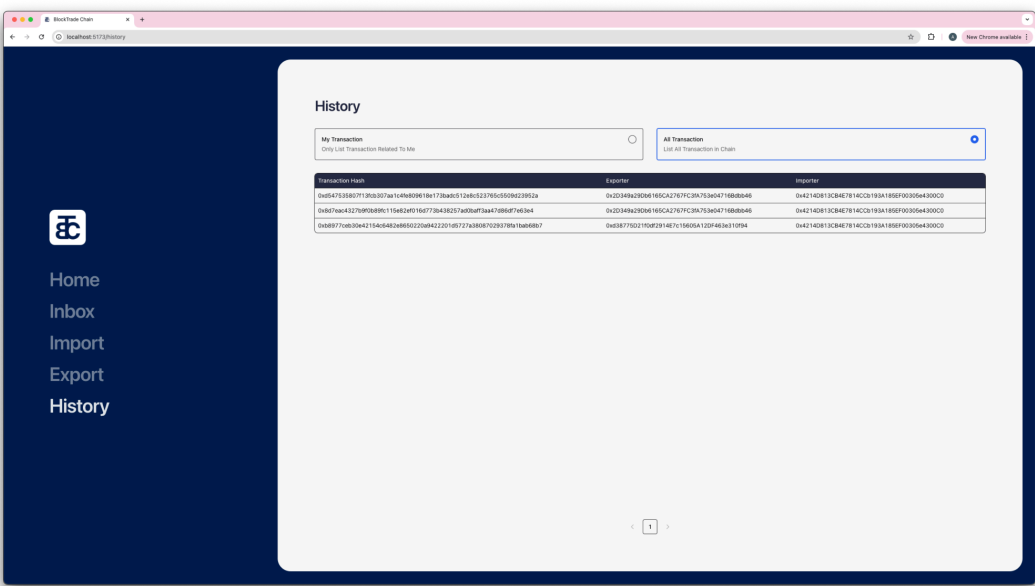
|           |   |
|-----------|---|
| Deskripsi | Berisikan konfirmasi apakah pengguna yakin untuk mengimpor/menolak impor suatu barang |
|-----------|---|



The screenshot shows a web application interface for 'BlockTrade Chain'. On the left is a dark blue sidebar with a logo and navigation links: Home, Inbox, Import, Export, and History. The main content area is titled 'Inbox' and displays a confirmation modal titled 'Import Goods?'. The modal contains the text 'Are you sure you want to import this item?' and two buttons: 'Cancel' and 'Confirm'. Below the modal, the 'Inbox' section shows 'Goods ready to be imported' and a table of received goods. The table has columns for Product, Qty, Exporter, Exported Time, and Actions. A single row is visible for 'Barang 1' with a quantity of 10. The 'Actions' column for this row contains 'Click to Confirm' and 'Click to Deny'. A green 'Confirm' button is highlighted over the 'Click to Confirm' link. To the right of the table, there is a summary card for the 'Exporter' showing a count of 1. At the bottom of the table, there is a pagination control showing '1'.

## 8. Halaman History

|           |   |
|-----------|---|
| Deskripsi | Berisikan data seluruh transaksi yang tercatat di <i>chain</i> . Dapat difilter dengan seluruh data yang terkait dengan pengguna atau untuk seluruh transaksi |
|-----------|---|



The screenshot shows the 'History' page of the 'BlockTrade Chain' application. The left sidebar is identical to the previous page. The main content area is titled 'History' and features two filter buttons: 'My Transaction' (selected) and 'All Transaction'. Below the filters is a table of transactions. The table has three columns: 'Transaction Hash', 'Exporter', and 'Importer'. There are three rows of transaction data. At the bottom of the table, there is a pagination control showing '1'.

## PEMBAGIAN TUGAS KELOMPOK

| NIM      | Nama                        | Pembagian Tugas   |
|----------|-----------------------------|---|
| 13521051 | Manuella Ivana Uli Sianipar | <ul style="list-style-type: none"> <li>• Integrasi Kontrak Pintar Dengan Wallet</li> <li>• Pengembangan Smart Contract Untuk Wallet Inbox dan Asset, History</li> <li>• Integrasi FE untuk History Page</li> <li>• Pengembangan FE untuk Home dan History Page</li> </ul>   |
| 13521135 | Nicholas Liem               | <ul style="list-style-type: none"> <li>• Inisiasi Private Chain, Smart Contract, dan Oracle</li> <li>• Integrasi Private Chain, Smart Contract, dan Oracle</li> <li>• Pengembangan Oracle</li> <li>• Pengembangan Smart Contract Untuk Ekspor/Impor, Confirm/Deny, Inbox, Asset, Exchange Rates</li> <li>• Integrasi FE untuk Exchange Rates, Ekspor/Impor, Confirm/Deny, Login (Unlock Account)</li> </ul>   |
| 13521162 | Antonio Natthan Krishna     | <ul style="list-style-type: none"> <li>• Inisiasi dan Pengembangan Antarmuka FE Aplikasi (seluruh <i>page</i>)</li> <li>• Integrasi Frontend dengan Metamask</li> <li>• Pengembangan FE untuk Ekspor/Import dan UI/UX secara keseluruhan</li> <li>• Integrasi FE untuk Exchange Rates</li> <li>• Pengembangan Smart Contract untuk Export/Import</li> <li>• Integrasi Fetching Data dengan Visible Data FE</li> <li>• Manajemen Tampilan Data pada FE (Pagination)</li> </ul> |

## REFERENSI

Vite. (n.d.). *Vite: Next-generation, front-end tool*. Diambil dari <https://vitejs.dev>.

Web3.js. (n.d.). *Web3.js documentation*. Diambil dari <https://web3js.readthedocs.io>.

Solidity. (n.d.). *Solidity documentation*. Diambil dari <https://soliditylang.org/docs>.

Truffle Suite. (n.d.). *Truffle framework*. Diambil dari <https://www.trufflesuite.com>.

Go Programming Language. (n.d.). *Go: The Go programming language*. Diambil dari <https://golang.org>.

## PRANALA VIDEO PENJELASAN

<https://youtu.be/uNvwT2ytmos>