

# Portofolio Interchain V2 /chain.co.id

[c.faishal.amrullah@gmail.com](mailto:c.faishal.amrullah@gmail.com)

@2020

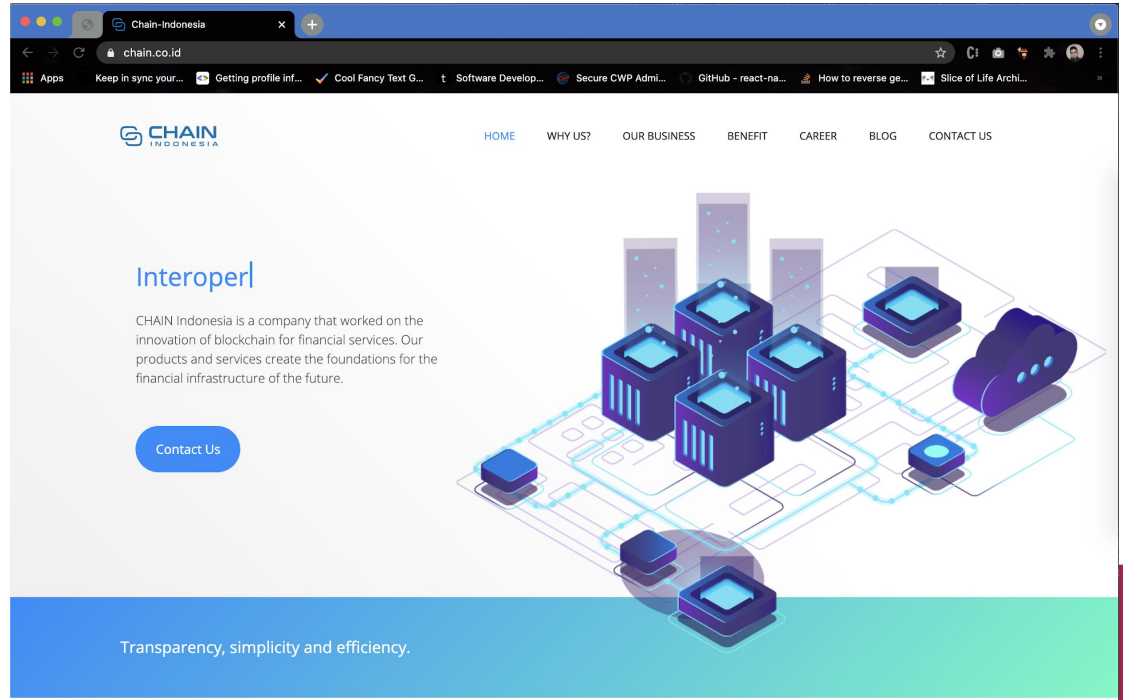
# Landing Page

Programming Language :

- Golang, HTML, CSS

Description :

Website company profile for Interchain



# Admin Panel & Backend API

Programming Language :

- PHP
- VueJs

Framework :

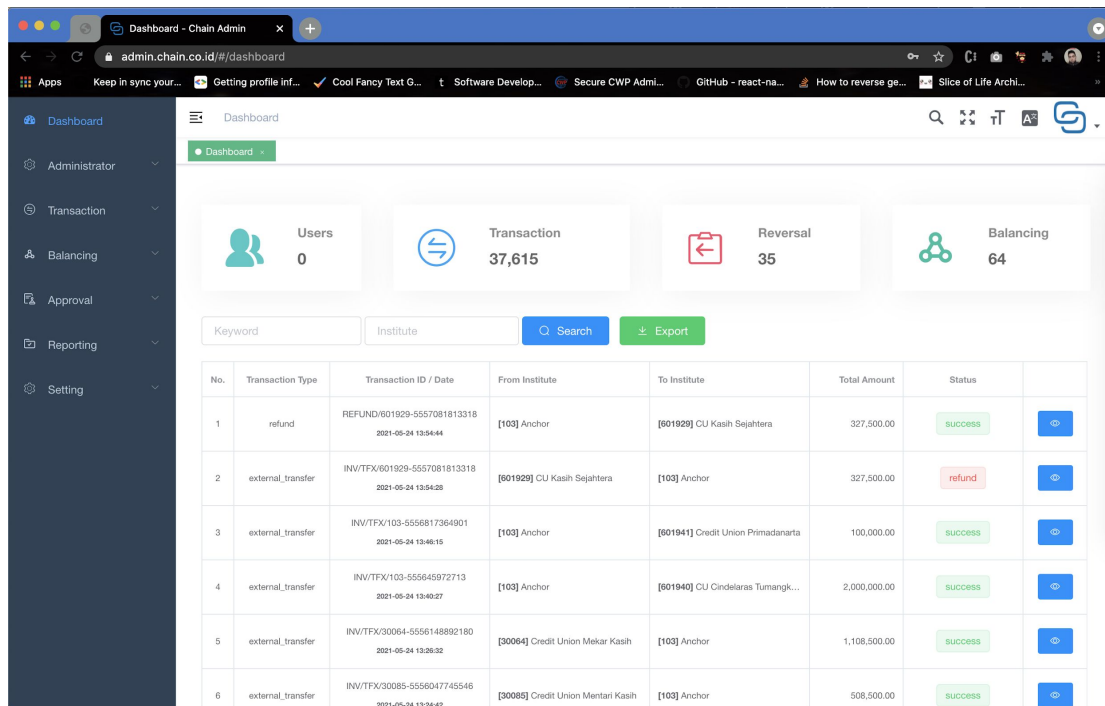
- Laravel (laravue.dev)
- VueJs

Database :

- PostgreSQL

Description :

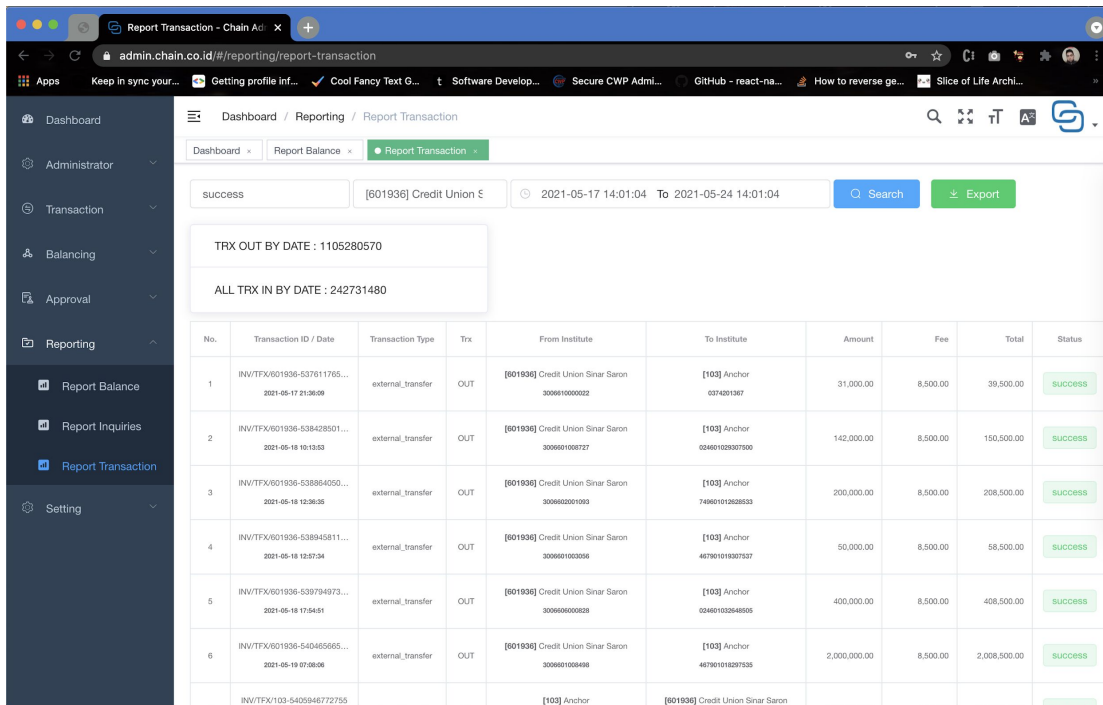
- Admin Panel for CURD, Monitoring and Reporting Transaction
- Backend API for receive request from Vendor and forward to Blockchain API



# Admin Panel & Backend API

## Description :

- Receive transaction status change from Blockchain API
- Use Websocket (socket.io) for auto refresh tables
- Module Deposit, TopUp and Redeem for blockchain token
- Upgrade system from V1



The screenshot shows a web application interface for reporting transactions. The browser address bar shows 'admin.chain.co.id/#reporting/report-transaction'. The left sidebar contains a menu with 'Dashboard', 'Administrator', 'Transaction', 'Balancing', 'Approval', 'Reporting', 'Report Balance', 'Report Inquiries', 'Report Transaction', and 'Setting'. The main content area has a breadcrumb 'Dashboard / Reporting / Report Transaction' and tabs for 'Dashboard', 'Report Balance', and 'Report Transaction'. Below the tabs, there are filters for 'success', '[601936] Credit Union', and a date range from '2021-05-17 14:01:04' to '2021-05-24 14:01:04'. There are 'Search' and 'Export' buttons. Two summary boxes show 'TRX OUT BY DATE : 1105280570' and 'ALL TRX IN BY DATE : 242731480'. A table lists transactions with columns: No., Transaction ID / Date, Transaction Type, Trx, From Institute, To Institute, Amount, Fee, Total, and Status. The table contains 6 rows of transaction data, all with a 'SUCCESS' status.

No.	Transaction ID / Date	Transaction Type	Trx	From Institute	To Institute	Amount	Fee	Total	Status
1	INV/TFX/601936-537611765... 2021-05-17 21:36:09	external_transfer	OUT	[601936] Credit Union Sinar Saron 3006010000022	[103] Anchor 0374201367	31,000.00	8,500.00	39,500.00	SUCCESS
2	INV/TFX/601936-538428501... 2021-05-18 19:13:53	external_transfer	OUT	[601936] Credit Union Sinar Saron 300601000727	[103] Anchor 024601029307509	142,000.00	8,500.00	150,500.00	SUCCESS
3	INV/TFX/601936-538864050... 2021-05-18 13:36:35	external_transfer	OUT	[601936] Credit Union Sinar Saron 3006002001093	[103] Anchor 748601012628553	200,000.00	8,500.00	208,500.00	SUCCESS
4	INV/TFX/601936-538945811... 2021-05-18 12:57:34	external_transfer	OUT	[601936] Credit Union Sinar Saron 3006010003056	[103] Anchor 467901010307537	50,000.00	8,500.00	58,500.00	SUCCESS
5	INV/TFX/601936-539794973... 2021-05-18 17:54:51	external_transfer	OUT	[601936] Credit Union Sinar Saron 3006000000628	[103] Anchor 024601032848505	400,000.00	8,500.00	408,500.00	SUCCESS
6	INV/TFX/601936-540465565... 2021-05-19 07:08:06	external_transfer	OUT	[601936] Credit Union Sinar Saron 300601000408	[103] Anchor 467901010297535	2,000,000.00	8,500.00	2,008,500.00	SUCCESS
...	INV/TFX/103-5405946772755	...	...	[103] Anchor	[601936] Credit Union Sinar Saron	...	...	...	...

# Blockchain API

Programming Language :

- Python

Framework :

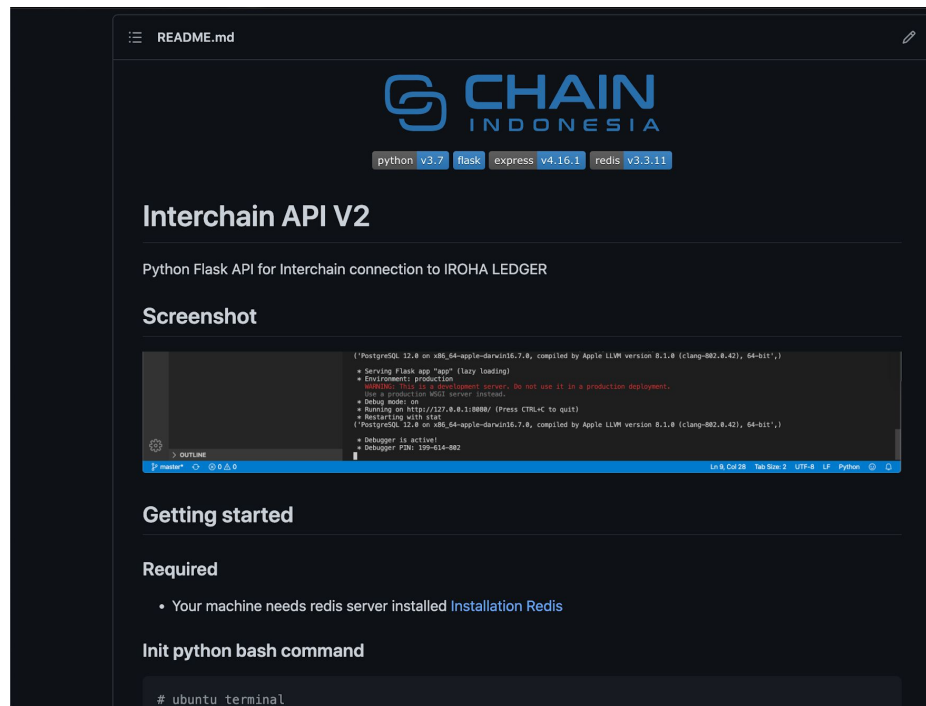
- Flask (Python web framework)

With :

- Redis
- Socket.IO
- Psycpg2

Description :

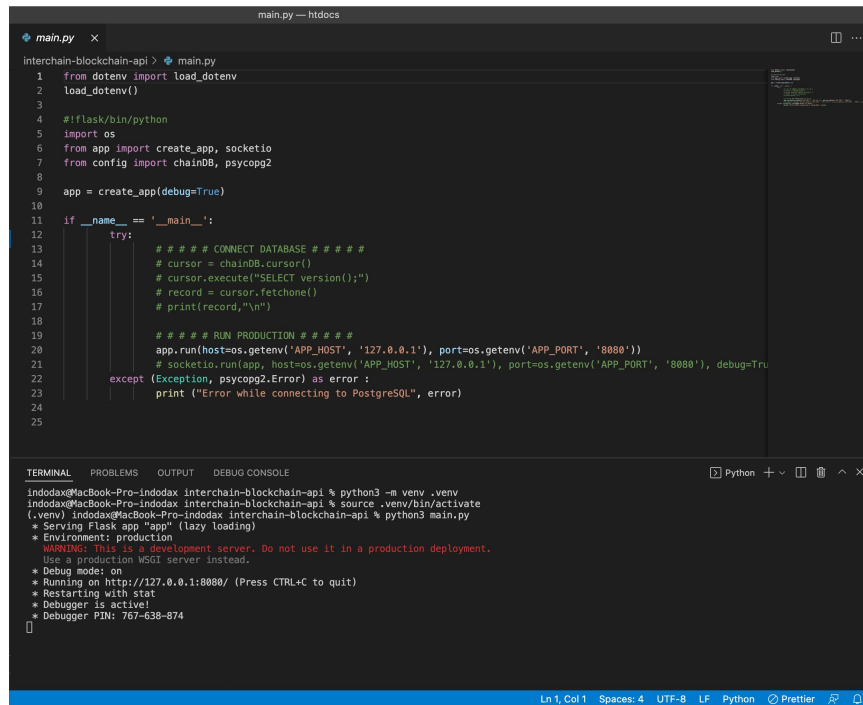
- Receive request transaction from Backend and send to Iroha Ledger
- Getting realtime transaction status from Iroha Ledger



# Blockchain API

## Description :

- Send Transaction status to Backend with API and broadcast notification to redis -> socket.io
- Available create account for transaction in Iroha Ledger



```
main.py — htdocs
main.py x
interchain-blockchain-api > main.py
1 from dotenv import load_dotenv
2 load_dotenv()
3
4 #flask/bin/python
5 import os
6 from app import create_app, socketio
7 from config import chainDB, pycpg2
8
9 app = create_app(debug=True)
10
11 if __name__ == '__main__':
12     try:
13         ##### CONNECT DATABASE #####
14         # cursor = chainDB.cursor()
15         # cursor.execute("SELECT version();")
16         # record = cursor.fetchone()
17         # print(record, "\n")
18
19         ##### RUN PRODUCTION #####
20         app.run(host=os.getenv('APP_HOST', '127.0.0.1'), port=os.getenv('APP_PORT', '8080'))
21         # socketio.run(app, host=os.getenv('APP_HOST', '127.0.0.1'), port=os.getenv('APP_PORT', '8080'), debug=True)
22     except (Exception, pycpg2.Error) as error :
23         print ("Error while connecting to PostgreSQL", error)
24
25
```

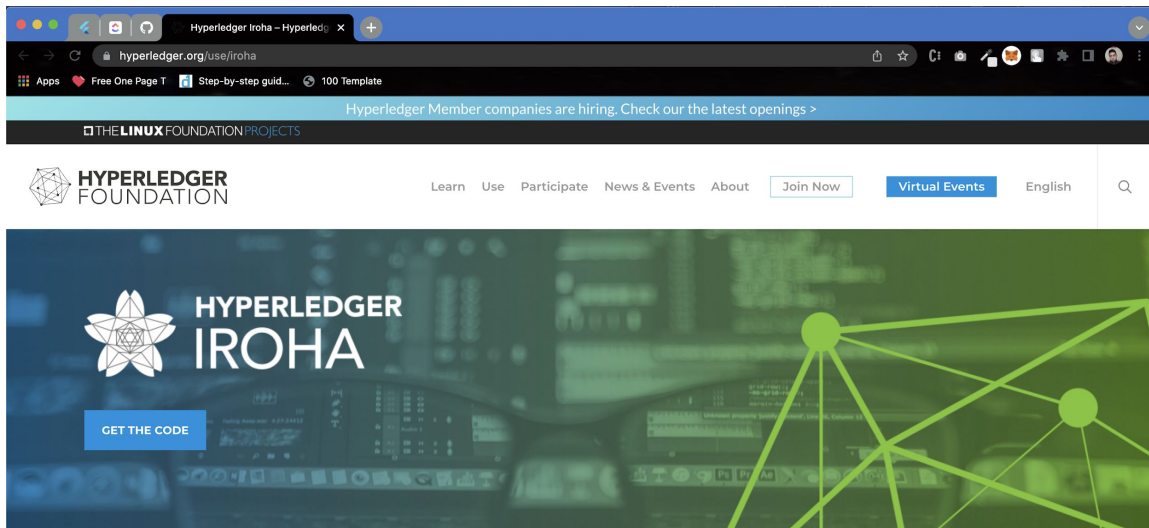
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
indodax@MacBook-Pro-indodax interchain-blockchain-api % python3 -m venv .venv
indodax@MacBook-Pro-indodax interchain-blockchain-api % source .venv/bin/activate
(.venv) indodax@MacBook-Pro-indodax interchain-blockchain-api % python3 main.py
* Serving flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active
* Debugger PIN: 767-638-874
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python Prettier

# Hyperledger Iroha

Hyperledger Iroha is a general purpose permissioned blockchain system that can be used to manage digital assets, identity, and serialized data. This can be useful for applications such as interbank settlement, central bank digital currencies, payment systems, national IDs, and logistics, among others.



## Type: Distributed ledger software

Hyperledger Iroha is designed to be simple and easy to incorporate into infrastructural or IoT projects requiring distributed ledger technology. Hyperledger Iroha features a simple construction, modular, domain-driven C++ design, emphasis on client application development and a new, crash fault tolerant consensus algorithm, called YAC.

