DAFTAR PUSTAKA

Al-Breiki, H., Rehman, M. H., Salah, K., & Sventinovic, D. (2020). Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges. *IEEE Access*, 85675-85685.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A. D., . . . Nguyen, B. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Distributed, Parallel, and Cluster Computing*, 30.

Azzi, R., Chamoun, R. K., & Sokhn, M. (2019). The power of a blockchain-based supply chain. *Computers & Industrial Engineering*, 582-592.

BPS. (2021). *Luas Panen Dan Produksi Padi Di Indonesia.* Jakarta: BPS.

BPS. (2022). *Bagaimana Stok Beras di Indonesia.* Jakarta: BPS.

Chopra, S., & Meindl, P. (2007). Supply Chain Management. Strategy, Planning & Operation. *Das Summa Summarum des Management*, 265-275.

Darmawan, O., & Wijaya, D. A. (2017). *Blockchain Dari Bitcoin Untuk Dunia.* Jakarta: Jasakom.

Docker. (2023, December 16). *Docker docs*. Retrieved from Docker docs: https://docs.docker.com/

Fabric, H. (2023, December 16). *Hyperledger Fabric Documentation*. Retrieved from Hyperledger Fabric Documentation: https://hyperledger-fabric.readthedocs.io/

Farouk, A., Alahmadi, A., Ghose, S., & Mashatan, A. (2020). Blockchain platform for industrial healthcare: Vision and future opportunities. *Computer Communications*, 1-24.

Gardy, J. S., Her, M., Moreno, G., Perez, C., & Yelinek, J. (2019). Emotions in storybooks: A comparison of storybooks that represent ethnic and racial groups in the United States. *Psychology of Popular Media Culture, 8*(3), 207-217. doi:https://doi.org/10.1037/ppm0000185

Gatteschi, V., Lamberti, F., Demartini, C., Pranteda, C., & Santamaria, V. (2018). Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough? *Future Internet*, 1-16.

[redacted] & Hao, Y. (2019). Blockchains in operations and supply chains: A model ıd reference implementation. *Computers & Industrial Engineering*, 242-51.

Kane, S. P., & Matthias, K. (2018). *Docker: Up & Running: Shipping Reliable Containers in Production.* O'Reilly Media.

Kushilevitz, E., & Malkin, T. (2016). Lecture notes in computer science: Vol. 9562. *Theory of cryptography.* Springer. doi:https://doi.org/10.1007/978-3-662-49096-9

Longo, F., Padovano, A., Nicoletti, L., & D'atri, G. (2019). Blockchain-enabled supply chain: An experimental study. *Computers & Industrial Engineering*, 57-69.

Perera, S., Nanayakkara, S., Rodrigo, M., Senaratne, S., & Weinand, R. (2020). Blockchain technology: Is it hype or real in the construction industry? *Journal of Industrial Information Integration*, 1-20.

Ravi, D., Ramachandran, S., Vignesh, R., Falmari, V. R., & Brindha, M. (2022). Privacy preserving transparent supply chain management through Hyperledger Fabric. *Blockchain: Research and Applications*.

Skudnov, R. (2013). Purely P2P Crypto-Currency With Finite Mini-Blockchain.

Stanton, D. (2017). *Supply Chain Management For Dummies.* For Dummies.

Tananto, F. F., Sari, W. S., Alzami, F., Nabila, M., Saputra, F. O., Umam, C., . . . Handoko, L. B. (2023). BUSINESS PROCESS REENGINEERING SUPPLY CHAIN MANAGEMENT SYSTEM BASED BLOCKCHAIN USING BPR LC. *Jurnal Teknik Informatika (JUTIF)*, 291-300.

Usman, M., Hermadi, I., & Arkeman, Y. (2021). Rancang Bangun Sistem Ketertelusuran Rantai Pasok Ayam Pedaging Melalui Aplikasi Android Berbasis Blockchain. *Jurnal Ilmu Komputer dan Agri-Informatika*, 105-114.

Vincent, N. E., Skjellum, A., & Medury, S. (2020). Blockchain architecture: A design that helps CPA firms leverage the technology. *International Journal of Accounting Information Systems*, 1-13.

Voulgaris, S., Fotiou, N., Siris, V. A., Polyzos, G. C., Jaatinen, M., & Oikonomidis, Y. (2019). Blockchain Technology for Intelligent Environments. *Future Internet*, 1-24.

Lampiran 1 Contoh lembar pengesahan untuk seminar proposal dan hasil

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI BLOCKCHAIN PADA SISTEM SUPPLY CHAIN BERAS UNTUK MENCEGAH MANIPULASI DATA (STUDI KASUS : PABRIK BERAS MAKASSAR)

Disusun dan diajukan oleh

Muh Nurfais Ramadhan
D121191022

Telah memenuhi syarat untuk melaksanakan Seminar Proposal/Hasil
Pada tanggal ……

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
_____	_____
NIP xxxxxxxxxxxxx	NIP xxxxxxxxxxxxx

Ketua Program Studi,

_____
NIP xxxxxxxxxxxxx

Lampiran 1 Source Code Smart contract

```
        asset = Asset{

            Tanggal_diolah:    tanggal_diolah,
            Alamat_perusahaan: alamat_perusahaan,
        }
    }
    if err != nil {
        return err
    }

    assetJSON, err := json.Marshal(asset)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutState(id, assetJSON)
}

// DeleteAsset deletes an given asset from the world state.
func (s *SmartContract) DeleteAsset(ctx contractapi.TransactionContextInterface, id string) error {
    exists, err := s.AssetExists(ctx, id)
    if err != nil {
        return err
    }
    if !exists {
        return fmt.Errorf("the asset %s does not exist", id)
    }

    return ctx.GetStub().DelState(id)
}

// AssetExists returns true when asset with given ID exists in world state
func (s *SmartContract) AssetExists(ctx contractapi.TransactionContextInterface, id string) (bool, error) {
    assetJSON, err := ctx.GetStub().GetState(id)
    if err != nil {
        return false, fmt.Errorf("failed to read from world state: %v", err)
    }

    return assetJSON != nil, nil
}

// TransferAsset updates the owner field of asset with given id in world state.
func (s *SmartContract) TransferAsset(ctx contractapi.TransactionContextInterface, id string, tanggal_diolah string, alamat_perusahaan string, jumlah string) error {
    asset, err := s.ReadAsset(ctx, id)
    if err != nil {
        return err
    }
    clientorg, err := ctx.GetClientIdentity().GetMSPID()
    if clientorg == "manufacturerMSP" {
        asset.Owner = "distributorMSP"
        asset.Tanggal_diolah = tanggal_diolah
        asset.Alamat_perusahaan = alamat_perusahaan

    }

    // timestamp, err := ctx.GetStub().GetTxTimestamp()
    dt := time.Now()
```

```
    dt := time.Now()
    timestampp := dt.Format("01-02-2006 15:04:05")

    if clientorg == "distributorMSP" {
        asset.Owner = "wholesalerMSP"
        asset.JumlahToWholesaler = jumlah
        asset.TimeStamp = timestampp

    }
    if clientorg == "wholesalerMSP" {
        asset.Owner = "retailerMSP"
        asset.JumlahToRetailer = jumlah
        asset.TimeStamp = timestampp

    }
    if clientorg == "retailerMSP" {
        asset.Owner = "consumerMSP"
    }

    assetJSON, err := json.Marshal(asset)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutState(id, assetJSON)
}

func (s *SmartContract) LockAsset(ctx contractapi.TransactionContextInterface, id string) error {
    clientorg, err := ctx.GetClientIdentity().GetMSPID()
    if err != nil {
        return err
    }

    assets, err := s.ReadAsset(ctx, id)
    if clientorg == "distributorMSP" {
        assets.Konfirmasi_distributor = true
    }
    if clientorg == "manufacturerMSP" {
        assets.Konfirmasi_manufacturer = true
    }
    if clientorg == "retailerMSP" {
        assets.Konfirmasi_retailer = true
    }
    if clientorg == "wholesalerMSP" {
        assets.Konfirmasi_wholesaler = true
    }
    if err != nil {
        return err
    }
    AssetJSON, err := json.Marshal(assets)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutState(id, AssetJSON)
}
```

```
    return ctx.GetStub().PutState(id, AssetJSON)
}

// GetAllAssets returns all assets found in world state
func (s *SmartContract) GetAllAssets(ctx contractapi.TransactionContextInterface) ([]*Asset, error) {
    // range query with empty string for startKey and endKey does an
    // open-ended query of all assets in the chaincode namespace.
    resultsIterator, err := ctx.GetStub().GetStateByRange("", "")
    if err != nil {
        return nil, err
    }
    defer resultsIterator.Close()

    var assets []*Asset
    for resultsIterator.HasNext() {
        queryResponse, err := resultsIterator.Next()
        if err != nil {
            return nil, err
        }

        var asset Asset
        err = json.Unmarshal(queryResponse.Value, &asset)
        if err != nil {
            return nil, err
        }
        assets = append(assets, &asset)
    }

    return assets, nil
}

func main() {
    assetChaincode, err := contractapi.NewChaincode(&SmartContract{})
    if err != nil {
        log.Panicf("Error creating asset-transfer-basic chaincode: %v", err)
    }

    if err := assetChaincode.Start(); err != nil {
        log.Panicf("Error starting asset-transfer-basic chaincode: %v", err)
    }
}
```

Lampiran 2 Source Code Golang untuk berinteraksi dengan jaringan *Hyperledger fabric*.

gatewayy > gateway.go

```go
package gatewayy

import (
    "database/sql"
    "encoding/json"
    "fmt"
    "log"
    "net/http"
    "os"
    "path/filepath"

    "github.com/gin-gonic/gin"
    "github.com/hyperledger/fabric-sdk-go/pkg/core/config"
    "github.com/hyperledger/fabric-sdk-go/pkg/gateway"
)

type Gatewayy struct {
    Db       *sql.DB
    Contract *gateway.Contract
    Network  *gateway.Network
    Wallet   *gateway.Wallet
}

type RequestGateway struct {
    Id                string
    Tanggal_Diolah    string
    Alamat_Perusahaan string
    Jumlah            string
    Jenis_Beras       string
    Nama_Petani       string
    Alamat            string
    Tanggal_Panen     string
    No_hp             string
    Npwp              string
}

func (gate *Gatewayy) ConnectToGateway(ctx *gin.Context) {
    var err error
```

```
38      var err error
39      gate.Wallet, err = gateway.NewFileSystemWallet("wallet")
40      if err != nil {
41          log.Fatalf("Failed to create wallet: %v", err)
42      }
43      // err = gate.populateWallet()
44      // if err != nil {
45      //  log.Fatalf("Failed to create Profile: %v", err)
46      // }
47      var ccpPath string
48      var WalletAccount string
49      baru := ctx.GetString("Organization")
50      if baru == "farmer" {
51          ccpPath = filepath.Join(
52              "config",
53              "connection-farmer.json",
54          )
55          WalletAccount = "Akunfarmer1"
56      }
57      if baru == "distributor" {
58          ccpPath = filepath.Join(
59              "config",
60              "connection-distributor.json",
61          )
62          WalletAccount = "Akundistributor1"
63      }
64      if baru == "retailer" {
65          ccpPath = filepath.Join(
66              "config",
67              "connection-retailer.json",
68          )
69          WalletAccount = "Akunretailer1"
70      }
71      if baru == "wholesaler" {
72          ccpPath = filepath.Join(
73              "config",
74              "connection-wholesaler.json",
75          )
```

```
75          )
76          WalletAccount = "Akunwholesaler1"
77      }
78      if baru == "manufacturer" {
79          ccpPath = filepath.Join(
80              "config",
81              "connection-manufacturer.json",
82          )
83          WalletAccount = "Akunmanufacturer1"
84      }
85      if baru == "consumer" {
86          ccpPath = filepath.Join(
87              "config",
88              "connection-consumer.json",
89          )
90          WalletAccount = "Akunconsumer1"
91      }
92
93      gw, err := gateway.Connect(
94          gateway.WithConfig(config.FromFile(filepath.Clean(ccpPath))),
95          gateway.WithIdentity(gate.Wallet, WalletAccount),
96      )
97      if err != nil {
98          log.Fatalf("Gagal Connect Ke Gateway: %v", err)
99      }
100     gate.Network, err = gw.GetNetwork("mychannel")
101     if err != nil {
102         // log.Fatalf("Failed to get network: %v", err)
103         log.Fatalf("Gagal Connect Ke Network: %v", err)
104     }
105     gate.Contract = gate.Network.GetContract("scberas")
        efer gw.Close()


        (gate *Gatewayy) AssetTransfer(ctx *gin.Context) {
        ar req = RequestGateway{}
```

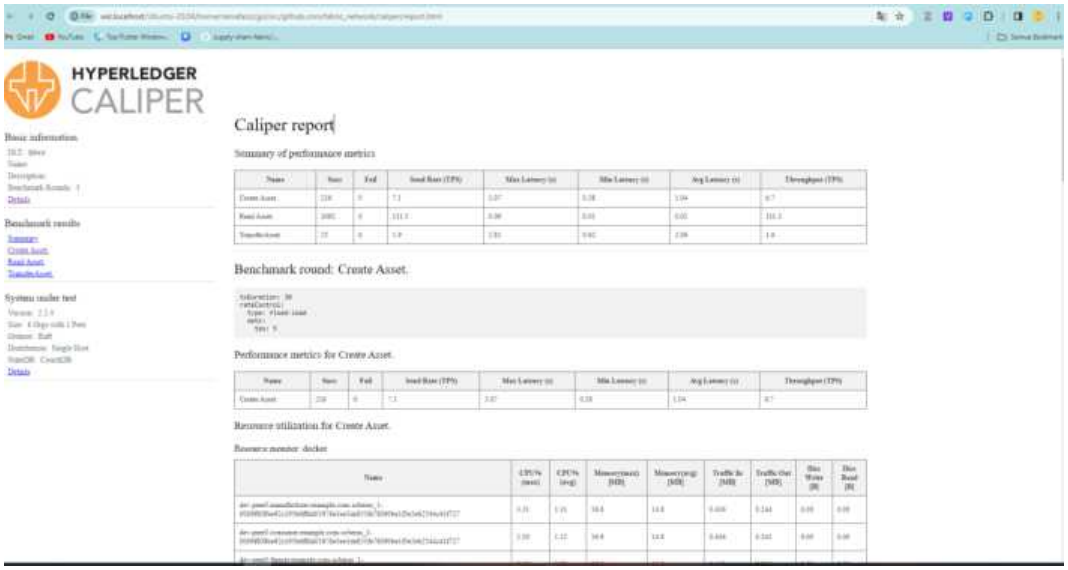
```
    var req = RequestGateway{}

    if err := ctx.ShouldBind(&req); err != nil {
        ctx.AbortWithStatusJSON(http.StatusBadRequest, gin.H{
            "Error": err.Error(),
        })
        return
    }
    result, err := gate.Contract.SubmitTransaction("TransferAsset", req.Id, req.Tanggal_Diolah, req.Alamat_Perusahaan, req.Jumlah)

    if err != nil {
        log.Fatalf("Gagal dalam submit transaksi: %v", err)
        // println("Gagal Dalam Submit Transaksi: %v", err)
    }

    ctx.JSON(http.StatusOK, gin.H{
        "result": string(result),
    })

}
func (gate *Gatewayy) CreateAsset(ctx *gin.Context) {
    var req = RequestGateway{}

    if err := ctx.ShouldBind(&req); err != nil {
        ctx.AbortWithStatusJSON(http.StatusBadRequest, gin.H{
            "Error": err.Error(),
        })
        return
    }
    result, err := gate.Contract.SubmitTransaction("CreateAssets", req.Id, req.Jenis_Beras, req.Nama_Petani, req.Alamat, req.Tanggal_Panen, req.No_hp, req.Npwp)
    if err != nil {
        ctx.AbortWithStatusJSON(http.StatusBadGateway, gin.H{
            "Errorki": err.Error(),
        })
        return
    }
    ctx.JSON(http.StatusOK, gin.H{
        "result": string(result),
```

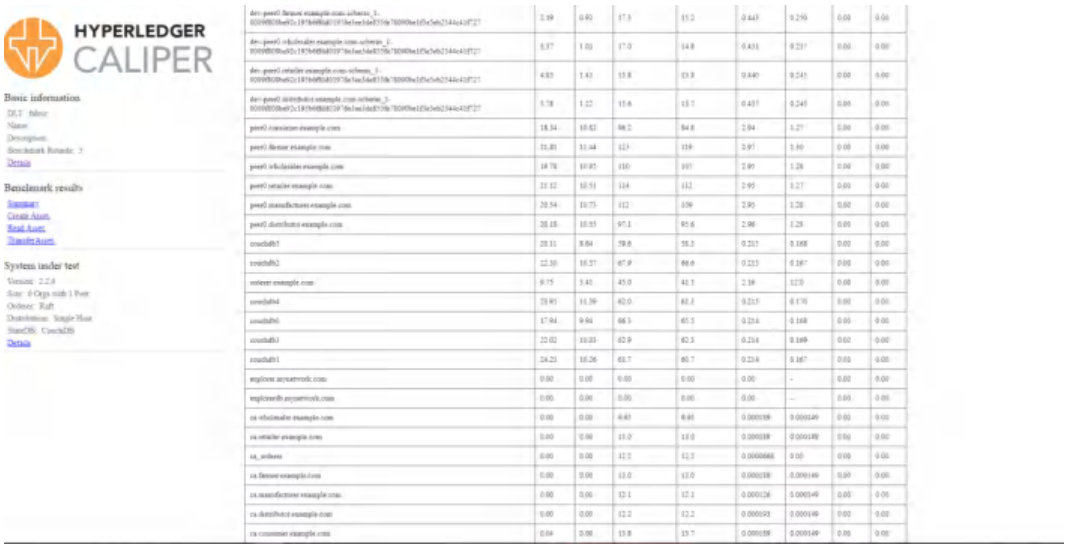
```
        "result": string(result),
    })
}

func (gate *Gatewayy) ReadAsset(ctx *gin.Context) {
    var req = RequestGateway{}
    var resultbaru map[string]interface{}
    if err := ctx.ShouldBind(&req); err != nil {
        ctx.AbortWithStatusJSON(http.StatusBadRequest, gin.H{
            "Error": err.Error(),
        })
        return
    }
    result, err := gate.Contract.EvaluateTransaction("ReadAsset", req.Id)
    if err != nil {
        ctx.AbortWithStatusJSON(http.StatusBadGateway, gin.H{
            "Errorki": err.Error(),
        })
        return
    }
    err = json.Unmarshal(result, &resultbaru)
    if err != nil {
        println("error")
    }
    ctx.JSON(http.StatusOK, gin.H{
        "result": resultbaru,
    })

}

func (gate *Gatewayy) LockAsset(ctx *gin.Context) {
    var req = RequestGateway{}
    // var resultbaru map[string]interface{}
    if err := ctx.ShouldBind(&req); err != nil {
        ctx.AbortWithStatusJSON(http.StatusBadRequest, gin.H{
            "Error": err.Error(),
        })
        return
```

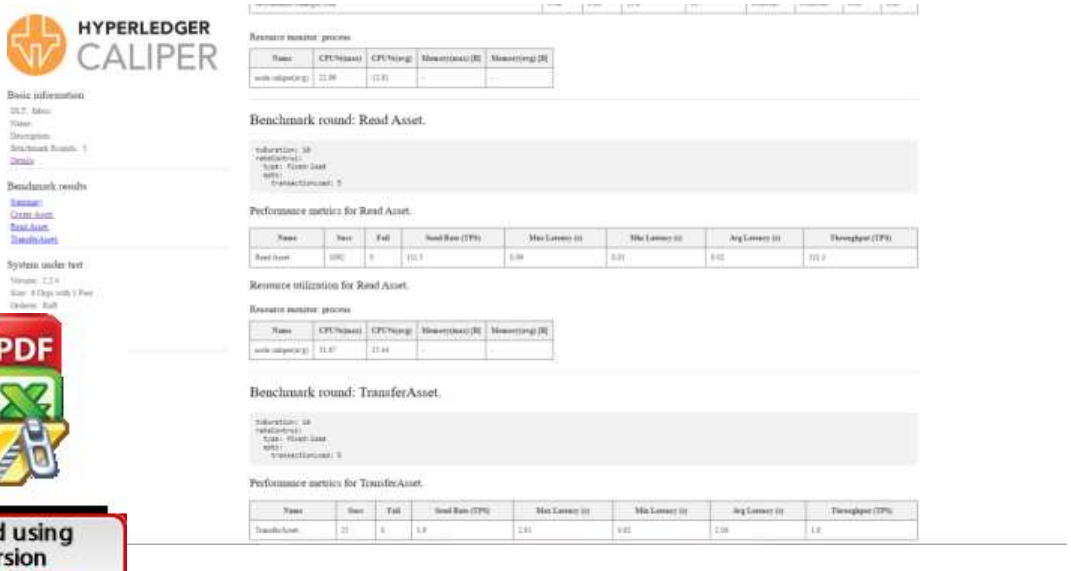
```
184         return
185     }
186     result, err := gate.Contract.SubmitTransaction("lockAsset", req.Id)
187     if err != nil {
188         ctx.AbortWithStatusJSON(http.StatusBadGateway, gin.H{
189             "Errorki": err.Error(),
190         })
191         return
192     }
193     // err = json.Unmarshal(result, &resultbaru)
194     // if err != nil {
195     //  println("error")
196     // }
197 
198     ctx.JSON(http.StatusOK, gin.H{
199         "result": string(result),
200     })
201 }
202 func (gate *Gatewayy) populateWallet() error {
203     credPath := filepath.Join(
204         "..",
205         "artifacts",
206         "channel",
207         "crypto-config",
208         "peerOrganizations",
209         "retailer.example.com",
210         "users",
211         "User1@retailer.example.com",
212         "msp",
213     )
214 
215     certPath := filepath.Join(credPath, "signcerts", "cert.pem")
216     // read the certificate pem
217     cert, err := os.ReadFile(filepath.Clean(certPath))
218     if err != nil {
219         return err
220     }
221 
```

```
221 
222     keyDir := filepath.Join(credPath, "keystore")
223     // there's a single file in this dir containing the private key
224     files, err := os.ReadDir(keyDir)
225     if err != nil {
226         return err
227     }
228     if len(files) != 1 {
229         return fmt.Errorf("keystore folder should have contain one file")
230     }
231     keyPath := filepath.Join(keyDir, files[0].Name())
232     key, err := os.ReadFile(filepath.Clean(keyPath))
233     if err != nil {
234         return err
235     }
236 
237     identity := gateway.NewX509Identity("retailerMSP", string(cert), string(key))
238 
239     return gate.Wallet.Put("Akunretailer1", identity)
240 }
241 
```

Lampiran 3 Laporan Evaluasi *Metriks* pada *Hyperledger Caliper*.







Lampiran 4 Source Code Frontend *Supply Chain*

Source Code
https://github.com/ramafaizz12/supplychain_frontend