Mini Project X-Mart dengan menerapkan stack React JS, Express JS, Java Spring Boot, dan teknologi GraphQL, Caching Redis, dan State Management Redux

Rizky Jubair · Follow
29 min read · Oct 23, 2023

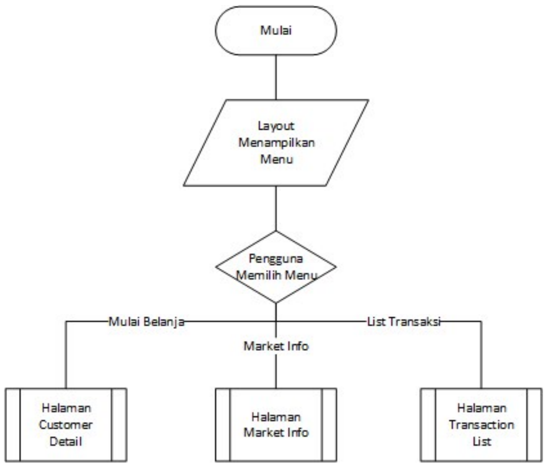
Assalamualaikum Wr. Wb.

Perkenalkan saya Andi Muhammad Rizky Akhmadi Jubair sebagai Software Developer di PT. Harmony Mitra Jayandra. Pada artikel ini saya akan sharing mengenai tahapan singkat membuat mini project X-Mart dengan menerapkan stack React JS, Express JS, dan Java Spring Boot. Untuk Front-End nya saya menggunakan React JS dan teknologi React Redux sebagai state management nya dan Axios sebagai tools untuk request API. Untuk Back-End nya terbagi dua yaitu menggunakan Java Spring Boot, dan Node JS dengan framework Express JS. Pada Node JS, saya menggunakan teknologi GraphQL sebagai REST API nya, Redis sebagai tools untuk caching, dan database MongoDB sebagai store datanya. Pada Java Spring Boot, saya menggunakan database PostgreSQL sebagai store datanya. Sebelum memulai membuat mini projectnya, pembaca diharapkan memahami konsep-konsep berikut sebagai dasar ilmunya.

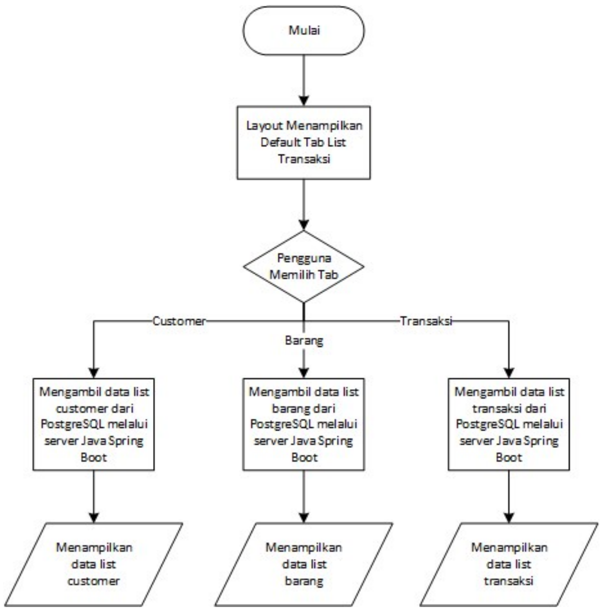
- useState dan useEffect pada React JS
- React Router DOM
- State Management menggunakan Redux
- Request API menggunakan Axios
- Membuat REST API menggunakan GraphQL
- Mengambil atau menyimpan data dalam Redis
- Operasi CRUD dalam MongoDB
- Operasi CRUD dalam PostgreSQL

Konsep Aplikasi

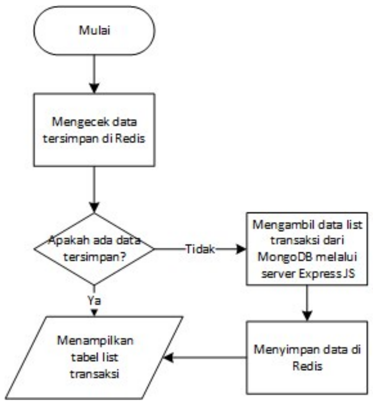
Aplikasi ini menyerupai aplikasi e-commerce lain untuk customer membeli suatu barang secara online. Customer akan menscan QRCode berisikan identitas dari customer tersebut agar terintegrasi dengan walletnya dan memiliki akses untuk pembelian barang secara online. Customer dapat memilih barang yang ingin dibelinya kemudian dimasukan kedalam keranjang untuk membuat transaksi pembelian. Setelah transaksi pembelian berhasil dibuat. Customer dapat mengecek hasil transaksi yang telah dibuat sebelumnya pada fitur riwayat transaksi. Berikut adalah alur aplikasi lengkapnya yang dibuat dalam bentuk flowchart.



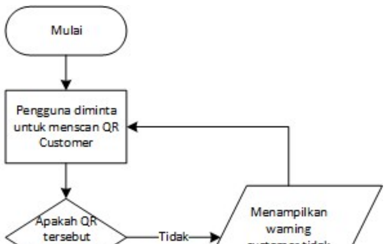
Halaman Home

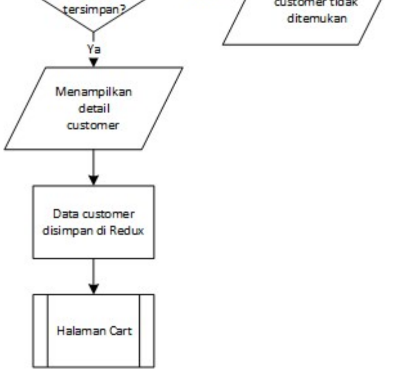

Mulai
Layout Menampilkan Default Tab List Transaksi
Pengguna Memilih Tab
Customer
Barang
Transaksi
Mengambil data list customer dari PostgreSQL melalui server Java Spring Boot
Mengambil data list barang dari PostgreSQL melalui server Java Spring Boot
Mengambil data list transaksi dari PostgreSQL melalui server Java Spring Boot
Menampilkan data list customer
Menampilkan data list barang
Menampilkan data list transaksi

Halaman Market Info — Spring Boot PostgreSQL

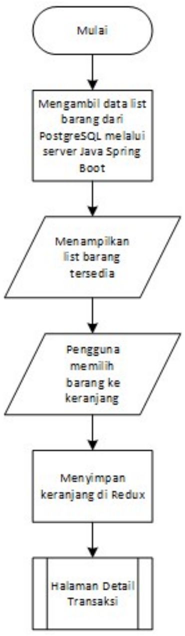

Mulai
Mengecek data tersimpan di Redis
Apakah ada data tersimpan?
Tidak
Ya
Mengambil data list transaksi dari MongoDB melalui server Express JS
Menyimpan data di Redis
Menampilkan tabel list transaksi

Halaman History Transaksi — Express JS MongoDB

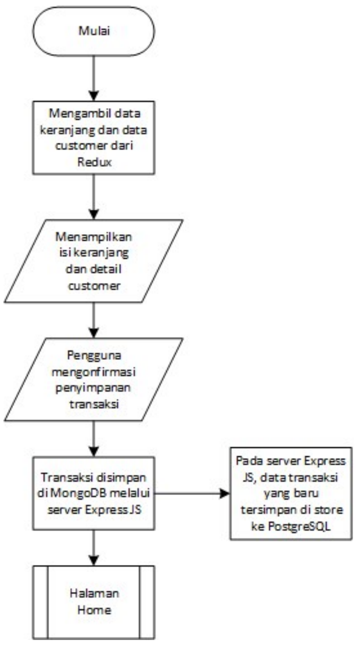




Halaman Customer Detail


Mulai
Mengambil data list barang dari PostgreSQL melalui server Java Spring Boot
Menampilkan list barang tersedia
Pengguna memilih barang ke keranjang
Menyimpan keranjang di Redux
Halaman Detail Transaksi

Halaman Cart


Mulai
Mengambil data keranjang dan data customer dari Redux
Menampilkan isi keranjang dan detail customer
Pengguna mengonfirmasi penyimpanan transaksi
Transaksi disimpan di MongoDB melalui server Express JS
Pada server Express JS, data transaksi yang baru tersimpan di store ke PostgreSQL
Halaman Home

Halaman Detail Transaksi

Java Spring Boot

Pembuatan backend java spring boot digunakan untuk service API customer, barang, transaksi dan transaksi split dengan database yang digunakan adalah PostgreSQL. Untuk entity customer memiliki property QR Code, nama customer, dan jenis wallet yang digunakan. QR Code itu sendiri bersifat unique karena merupakan property pembeda dari customer lain.

Untuk entity barang memiliki property RFID, nama barang, dan harga satuan untuk setiap barangnya. Property pembeda dari entity barang ini adalah pada RFID dimana setiap jenis barang memiliki RFID yang berbeda beda sehingga RFID bersifat unique. Kemudian untuk entity transaksi memiliki property id transaksi, QR Code, RFID, harga satuan, jumlah, dan waktu pesan. QR Code dari entity transaksi merupakan hasil join property dari entity customer sedangkan RFID merupakan hasil join property dari entity barang. Property unique dari transaksi ini adalah id transaksi. Pada entity transaksi, property yang ditampilkan merupakan property unique dari entity customer dan barang, untuk mencegah tertampilnya property unique tersebut, maka dibuat satu entity lagi yaitu transaksi split. Entity transaksi split memiliki property nama, nama barang, harga satuan, jumlah, dan waktu pesan, dimana property nama merupakan hasil join property dari entity customer sedangkan property harga satuan dan nama barang merupakan hasil join property dari barang. Dari konsep service API tersebut, berikut adalah detail project java spring bootnya dan untuk dapat melihat deskripsi dan dependencies dari inisialisasi project spring boot nya, silahkan klik link disini.

Struktur Folder :

```
xmart-java
├── src
│   └── main
│       ├── java
│       │   └── com
│       │       └── example
│       │           └── xmartjava
│       │               ├── controller
│       │               │   ├── BarangController.java
│       │               │   ├── CustomerController.java
│       │               │   └── TransaksiController.java
│       │               ├── entity
│       │               │   ├── Barang.java
│       │               │   ├── Customer.java
│       │               │   ├── Transaksi.java
│       │               │   └── TransaksiSplit.java
│       │               ├── exception
│       │               │   └── ResourceNotFoundException.java
│       │               ├── repository
│       │               │   ├── BarangRepository.java
│       │               │   ├── CustomerRepository.java
│       │               │   ├── TransaksiRepository.java
│       │               │   └── TransaksiSplitRepository.java
│       │               ├── service
│       │               │   ├── BarangService.java
│       │               │   ├── CustomerService.java
│       │               │   └── TransaksiService.java
│       │               └── XmartJavaApplication.java
│       └── resources
│           └── application.properties
└── pom.xml
```

Configurasi file application.properties :

```
spring.datasource.hikari.connection-timeout=20000
spring.datasource.hikari.maximum-pool-size=5

spring.datasource.url=jdbc:postgresql://localhost:5433/xmart
spring.datasource.username=postgres
spring.datasource.password=psql1234
spring.jpa.hibernate.ddl-auto=update
server.error.include-stacktrace=never
```

Entity Class :

```java
package com.example.xmartjava.entity;

import java.io.Serializable;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.Data;

@Entity
@Data
public class Customer implements Serializable {

    @Id
    private String qrCode;
    private String nama;
    private String wallet;
}
```

```java
package com.example.xmartjava.entity;

import java.io.Serializable;
import java.math.BigDecimal;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.Data;

@Entity
@Data
public class Barang implements Serializable {
    @Id
    private String rfid;
    private String namaBarang;
    private BigDecimal hargaSatuan;
}
```

```java
package com.example.xmartjava.entity;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Temporal;
import jakarta.persistence.TemporalType;
import lombok.Data;

@Entity
@Data
public class Transaksi implements Serializable {
    @Id
    private String id;
    private String qrCode;
    private String rfid;
    private BigDecimal hargaSatuan;
    private Integer jumlah;
    @Temporal(TemporalType.TIMESTAMP)
    private Date waktuPesan;
}
```

```java
package com.example.xmartjava.entity;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Temporal;
import jakarta.persistence.TemporalType;
import lombok.Data;
```

```
@Entity
@Data
public class TransaksiSplit implements Serializable {
    @Id
    private String id;
    private String nama;
    private String namaBarang;
    private BigDecimal hargaSatuan;
    private Integer jumlah;
    @Temporal(TemporalType.TIMESTAMP)
    private Date waktuPesan;
}
```

Repository Class :

```
package com.example.xmartjava.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.xmartjava.entity.Customer;

public interface CustomerRepository extends JpaRepository<Customer, String> {

}
```

```
package com.example.xmartjava.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.xmartjava.entity.Barang;

public interface BarangRepository extends JpaRepository<Barang, String> {

}
```

```
package com.example.xmartjava.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.xmartjava.entity.Transaksi;

public interface TransaksiRepository extends JpaRepository<Transaksi, String> {

}
```

```
package com.example.xmartjava.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.example.xmartjava.entity.TransaksiSplit;

public interface TransaksiSplitRepository extends JpaRepository<TransaksiSplit,
 String> {
    @Query(value = "SELECT id, nama, nama_barang, transaksi_customer.harga_satu
an, jumlah, waktu_pesan  FROM \r\n" + //
            "\t(SELECT * FROM transaksi JOIN customer ON transaksi.qr_code = cu
stomer.qr_code) \r\n" + //
            "AS transaksi_customer JOIN barang ON transaksi_customer.rfid = bar
ang.rfid ORDER BY waktu_pesan", nativeQuery = true)
    List<TransaksiSplit> findAllTransaksiSplit();
}
```

Exception Class :

```
package com.example.xmartjava.exception;
```

```
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(code = HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {
    public ResourceNotFoundException(String message) {
        super(message);
    }
}
```

Service Class :

```
package com.example.xmartjava.service;

import java.util.List;
import java.util.UUID;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.xmartjava.entity.Customer;
import com.example.xmartjava.exception.ResourceNotFoundException;
import com.example.xmartjava.repository.CustomerRepository;

@Service
public class CustomerService {
    @Autowired
    private CustomerRepository customerRepository;

    public Customer findById(String id) {
        return customerRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Customer deng
an QRCode tersebut tidak ditemukan"));
    }

    public List<Customer> findAll() {
        return customerRepository.findAll();
    }

    public Customer create(Customer customer) {
        customer.setQrCode(UUID.randomUUID().toString());
        return customerRepository.save(customer);
    }

    public Customer edit(Customer customer) {
        return customerRepository.save(customer);
    }

    public void deleteById(String id) {
        customerRepository.deleteById(id);
    }
}
```

```
package com.example.xmartjava.service;

import java.util.List;
import java.util.UUID;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.xmartjava.entity.Barang;
import com.example.xmartjava.exception.ResourceNotFoundException;
import com.example.xmartjava.repository.BarangRepository;

@Service
public class BarangService {
    @Autowired
    private BarangRepository barangRepository;

    public Barang findById(String id) {
        return barangRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Barang dengan
 RFID " + id + " tidak ditemukan"));
    }

    public List<Barang> findAll() {
        return barangRepository.findAll();
    }
    public Barang create(Barang barang) {
        barang.setRfid(UUID.randomUUID().toString());
        return barangRepository.save(barang);
    }

    public Barang edit(Barang barang) {
        return barangRepository.save(barang);
    }

    public void deleteById(String id) {
        barangRepository.deleteById(id);
    }
}
```

```
package com.example.xmartjava.service;

import java.util.List;
import java.util.UUID;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.xmartjava.entity.Transaksi;
import com.example.xmartjava.entity.TransaksiSplit;
import com.example.xmartjava.exception.ResourceNotFoundException;
import com.example.xmartjava.repository.TransaksiRepository;
import com.example.xmartjava.repository.TransaksiSplitRepository;

@Service
public class TransaksiService {
    @Autowired
    private TransaksiRepository transaksiRepository;

    @Autowired
    private TransaksiSplitRepository transaksiSplitRepository;

    public Transaksi findById(String id) {
        return transaksiRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Transaksi den
gan id " + id + " tidak ditemukan"));
    }

    public List<Transaksi> findAll() {
        return transaksiRepository.findAll();
    }

    public List<TransaksiSplit> findAllTransaksiSplit() {
        return transaksiSplitRepository.findAllTransaksiSplit();
    }

    public Transaksi create(Transaksi transaksi) {
        transaksi.setId(UUID.randomUUID().toString());
        return transaksiRepository.save(transaksi);
    }

    public Transaksi edit(Transaksi transaksi) {
        return transaksiRepository.save(transaksi);
    }

    public void deleteById(String id) {
        transaksiRepository.deleteById(id);
    }
}
```

Controller Class :

```
package com.example.xmartjava.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.xmartjava.entity.Customer;
import com.example.xmartjava.service.CustomerService;
```

```
@RestController
@RequestMapping("/api")
public class CustomerController {
    @Autowired
    private CustomerService customerService;

    @GetMapping("/customer")
    public List<Customer> findAll() {
        return customerService.findAll();
    }

    @GetMapping("/customer/{id}")
    public Customer findById(@PathVariable("id") String id) {
        return customerService.findById(id);
    }

    @PostMapping("/customer")
    public Customer create(@RequestBody Customer customer) {
        return customerService.create(customer);
    }

    @PutMapping("/customer")
    public Customer edit(@RequestBody Customer customer) {
        return customerService.edit(customer);
    }

    @DeleteMapping("/customer/{id}")
    public void deleteById(@PathVariable("id") String id) {
        customerService.deleteById(id);
    }
}
```

```
package com.example.xmartjava.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.xmartjava.entity.Barang;
import com.example.xmartjava.service.BarangService;

@RestController
@RequestMapping("/api")
public class BarangController {
    @Autowired
    private BarangService barangService;

    @GetMapping("/barang")
    public List<Barang> findAll() {
        return barangService.findAll();
    }

    @GetMapping("/barang/{id}")
    public Barang findById(@PathVariable("id") String id) {
        return barangService.findById(id);
    }

    @PostMapping("/barang")
    public Barang create(@RequestBody Barang barang) {
        return barangService.create(barang);
    }

    @PutMapping("/barang")
    public Barang edit(@RequestBody Barang barang) {
        return barangService.edit(barang);
    }

    @DeleteMapping("/barang/{id}")
    public void deleteById(@PathVariable("id") String id) {
        barangService.deleteById(id);
    }
}
```

```
package com.example.xmartjava.controller;

import java.util.List;
```

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.xmartjava.entity.Transaksi;
import com.example.xmartjava.entity.TransaksiSplit;
import com.example.xmartjava.service.TransaksiService;

@RestController
@RequestMapping("/api")
public class TransaksiController {
    @Autowired
    private TransaksiService transaksiService;

    @GetMapping("/transaksi")
    public List<Transaksi> findAll() {
        return transaksiService.findAll();
    }

    @GetMapping("/transaksi-split")
    public List<TransaksiSplit> findAllTransaksiSplits() {
        return transaksiService.findAllTransaksiSplit();
    }

    @GetMapping("/transaksi/{id}")
    public Transaksi findById(@PathVariable("id") String id) {
        return transaksiService.findById(id);
    }

    @PostMapping("/transaksi")
    public Transaksi create(@RequestBody Transaksi transaksi) {
        return transaksiService.create(transaksi);
    }

    @PutMapping("/transaksi")
    public Transaksi edit(@RequestBody Transaksi transaksi) {
        return transaksiService.edit(transaksi);
    }

    @DeleteMapping("/transaksi/{id}")
    public void deleteById(@PathVariable("id") String id) {
        transaksiService.deleteById(id);
    }
}
```

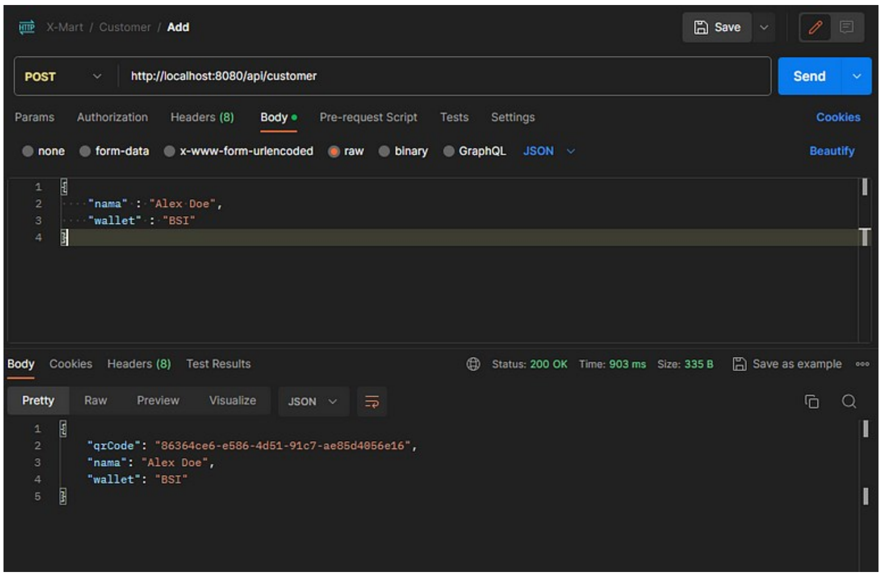
Main Class :

```java
package com.example.xmartjava;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@SpringBootApplication
public class XmartJavaApplication {

 public static void main(String[] args) {
  SpringApplication.run(XmartJavaApplication.class, args);
 }

 @Bean
 public WebMvcConfigurer corsConfigurer() {
  return new WebMvcConfigurer() {
   @Override
   public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**").allowedOrigins("*");
   };
  };
 }

}
```

Testing API Java Spring Boot

Untuk menguji fungsionalitas project spring boot yang telah dibuat, maka perlu menguji API untuk setiap controller menggunakan Postman. Untuk mempersingkat penulisan, saya hanya menguji untuk method GET dan POST saja untuk setiap controller. Berikut adalah hasil pengujiannya.

Add Customer :



X-Mart / Customer / **Add**

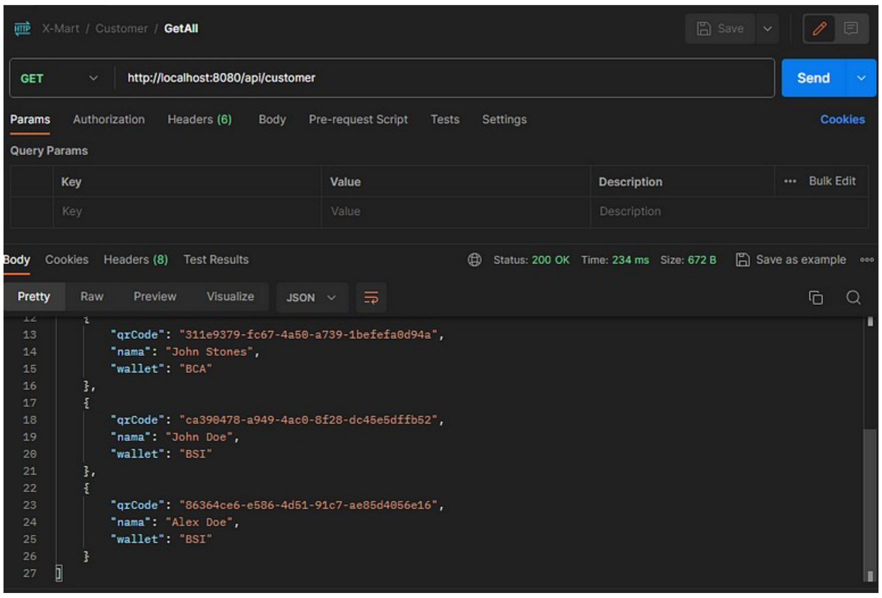
POST http://localhost:8080/api/customer

```
{
    "nama" : "Alex Doe",
    "wallet" : "BSI"
}
```

Status: 200 OK Time: 903 ms Size: 335 B

```
{
    "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
    "nama": "Alex Doe",
    "wallet": "BSI"
}
```

Get All Customer :



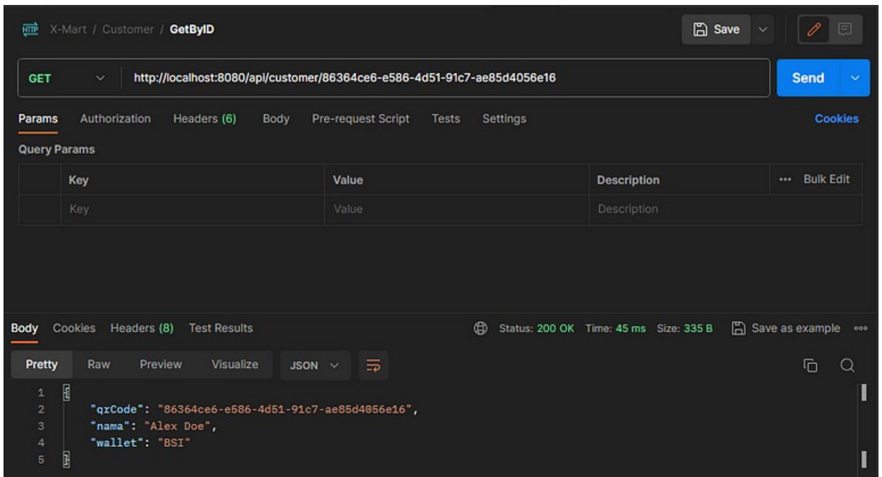
X-Mart / Customer / **GetAll**

GET http://localhost:8080/api/customer

Status: 200 OK Time: 234 ms Size: 672 B

```
    {
        "qrCode": "311e9379-fc67-4a50-a739-1befefa0d94a",
        "nama": "John Stones",
        "wallet": "BCA"
    },
    {
        "qrCode": "ca390478-a949-4ac0-8f28-dc45e5dffb52",
        "nama": "John Doe",
        "wallet": "BSI"
    },
    {
        "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
        "nama": "Alex Doe",
        "wallet": "BSI"
    }
]
```

Get Customer By Id :



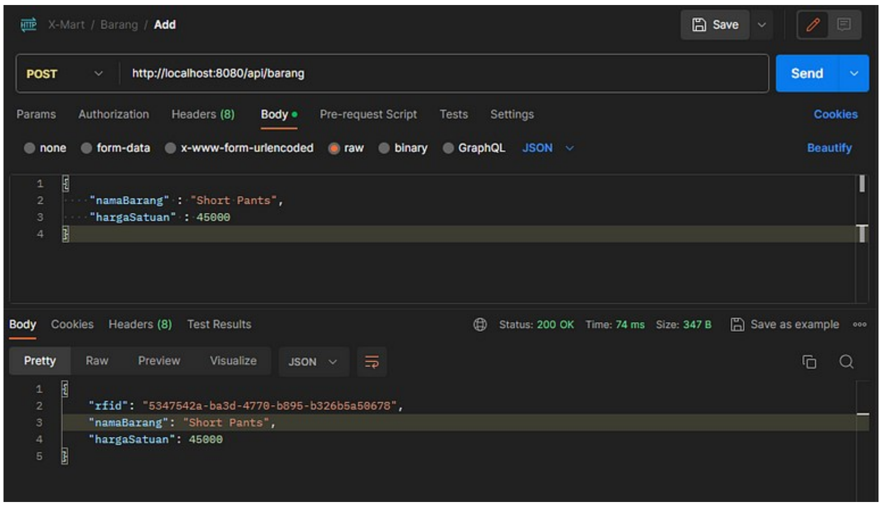
X-Mart / Customer / **GetByID**

GET http://localhost:8080/api/customer/86364ce6-e586-4d51-91c7-ae85d4056e16

Status: 200 OK Time: 45 ms Size: 335 B

```
{
    "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
    "nama": "Alex Doe",
    "wallet": "BSI"
}
```

Add Barang :


X-Mart / Barang / Add

POST http://localhost:8080/api/barang

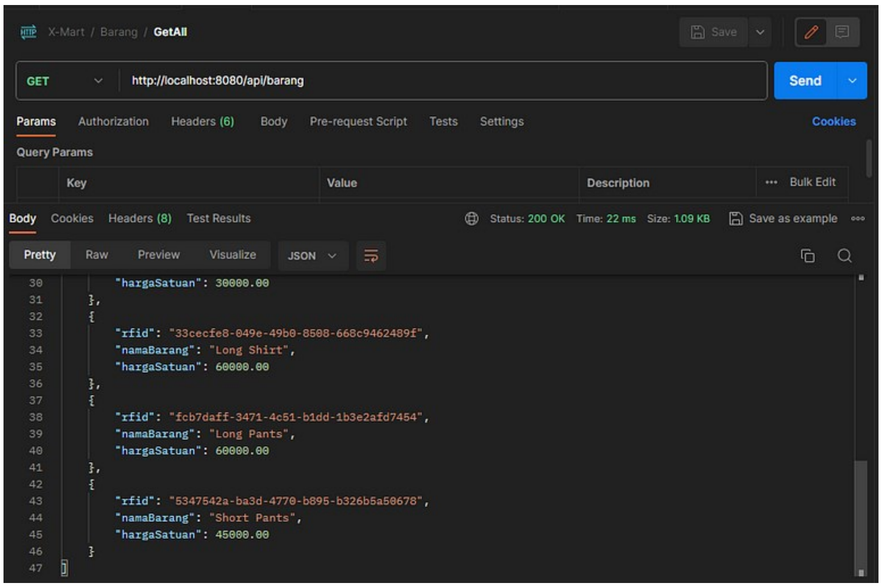
Body: raw JSON

```
{
    "namaBarang" : "Short Pants",
    "hargaSatuan" : 45000
}
```

Status: 200 OK Time: 74 ms Size: 347 B

```
{
    "rfid": "5347542a-ba3d-4770-b895-b326b5a50678",
    "namaBarang": "Short Pants",
    "hargaSatuan": 45000
}
```

Get All Barang :

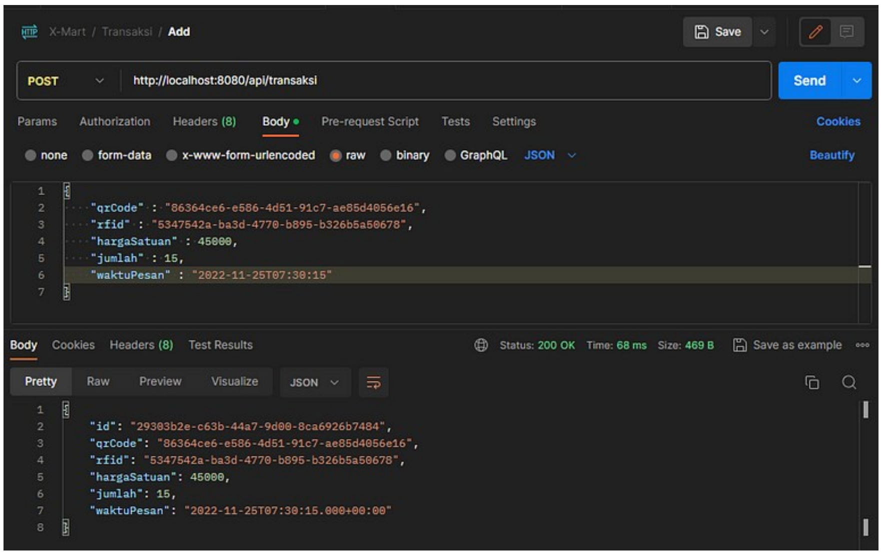

X-Mart / Barang / GetAll

GET http://localhost:8080/api/barang

Status: 200 OK Time: 22 ms Size: 1.09 KB

```
            "hargaSatuan": 30000.00
        },
        {
            "rfid": "33cecfe8-049e-49b0-8508-668c9462489f",
            "namaBarang": "Long Shirt",
            "hargaSatuan": 60000.00
        },
        {
            "rfid": "fcb7daff-3471-4c51-b1dd-1b3e2afd7454",
            "namaBarang": "Long Pants",
            "hargaSatuan": 60000.00
        },
        {
            "rfid": "5347542a-ba3d-4770-b895-b326b5a50678",
            "namaBarang": "Short Pants",
            "hargaSatuan": 45000.00
        }
]
```

Add Transaksi :


X-Mart / Transaksi / Add

POST http://localhost:8080/api/transaksi

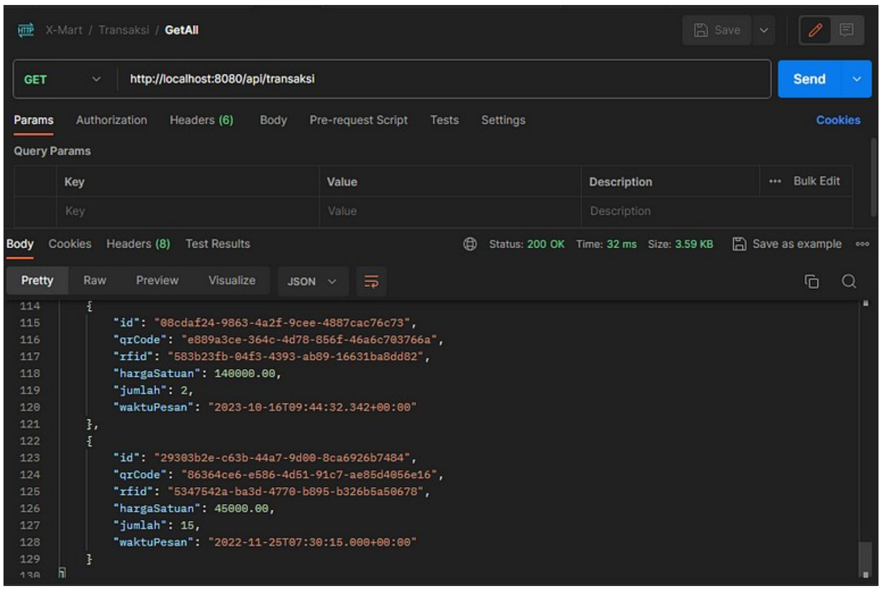
Body: raw JSON

```
{
    "qrCode" : "86364ce6-e586-4d51-91c7-ae85d4056e16",
    "rfid" : "5347542a-ba3d-4770-b895-b326b5a50678",
    "hargaSatuan" : 45000,
    "jumlah" : 15,
    "waktuPesan" : "2022-11-25T07:30:15"
}
```

Status: 200 OK Time: 68 ms Size: 469 B

```
{
    "id": "29303b2e-c63b-44a7-9d00-8ca6926b7484",
    "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
    "rfid": "5347542a-ba3d-4770-b895-b326b5a50678",
    "hargaSatuan": 45000,
    "jumlah": 15,
    "waktuPesan": "2022-11-25T07:30:15.000+00:00"
}
```

Get All Transaksi :

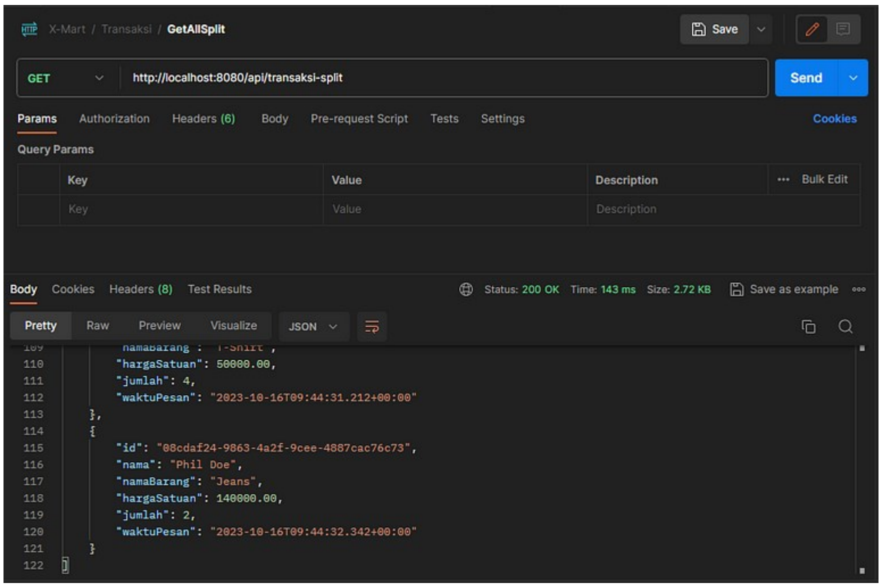


http://localhost:8080/api/transaksi

Status: 200 OK Time: 32 ms Size: 3.59 KB

```
    {
        "id": "08cdaf24-9863-4a2f-9cee-4887cac76c73",
        "qrCode": "e889a3ce-364c-4d78-856f-46a6c703766a",
        "rfid": "583b23fb-04f3-4393-ab89-16631ba8dd82",
        "hargaSatuan": 140000.00,
        "jumlah": 2,
        "waktuPesan": "2023-10-16T09:44:32.342+00:00"
    },
    {
        "id": "29303b2e-c63b-44a7-9d00-8ca6926b7484",
        "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
        "rfid": "5347542a-ba3d-4770-b895-b326b5a50678",
        "hargaSatuan": 45000.00,
        "jumlah": 15,
        "waktuPesan": "2022-11-25T07:30:15.000+00:00"
    }
]
```

Get All Transaksi Split :



http://localhost:8080/api/transaksi-split

Status: 200 OK Time: 143 ms Size: 2.72 KB

```
        "namaBarang": "T-Shirt",
        "hargaSatuan": 50000.00,
        "jumlah": 4,
        "waktuPesan": "2023-10-16T09:44:31.212+00:00"
    },
    {
        "id": "08cdaf24-9863-4a2f-9cee-4887cac76c73",
        "nama": "Phil Doe",
        "namaBarang": "Jeans",
        "hargaSatuan": 140000.00,
        "jumlah": 2,
        "waktuPesan": "2023-10-16T09:44:32.342+00:00"
    }
]
```

Express JS (GraphQL + Redis)

Berbeda dengan backend dengan java spring boot, pada express JS service API yang diterapkan menggunakan GraphQL sekaligus menerapkan Redis sebagai caching pada method GET transaksi. Entity yang digunakan hanya transaksi saja dan property nya sama seperti property pada entity transaksi di java spring boot. Database yang digunkan pada service ini adalah MongoDB. Service API ini digunakan untuk menyimpan hasil transaksi dari frontend yang kemudian distore data transaksinya ke PostgreSQL melalui service API java spring boot. Berikut adalah detail dari project express JS nya.

Struktur Folder :

```
xmart-nodejs
├── lib
│   ├── graphql
│   │   ├── mutations
│   │   │   ├── transaksi
│   │   │   │   ├── add.js
│   │   │   │   ├── index.js
│   │   │   │   └── remove.js
│   │   │   └── index.js
│   │   ├── queries
│   │   │   ├── transaksi
│   │   │   │   ├── index.js
│   │   │   │   ├── multiple.js
│   │   │   │   └── single.js
│   │   │   └── index.js
│   │   ├── types
│   │   │   └── transaksi.js
│   │   └── index.js
│   └── models
│       ├── redis.js
│       └── transaksi.js
├── node_modules
├── package-lock.json
├── package.json
└── server.js
```

Configurasi file package.json :

```json
{
  "name": "xmart-nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^1.5.1",
    "bluebird": "^3.7.2",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-graphql": "^0.12.0",
    "graphql": "^15.8.0",
    "graphql-date": "^1.0.3",
    "mongoose": "^7.5.3",
    "redis": "^4.6.10"
  }
}
```

Models :

- redis.js (path : ./xmart-nodejs/lib/models/redis.js)

```javascript
import { createClient } from 'redis';
const redisClient = createClient();
const DEFAULT_EXPIRATION = 3600;

//Redis Connection
(
    async () => {
        await redisClient
        .on('error', err => console.log('Redis client error', err))
        .connect();
    }
)();

export default redisClient;
```

- transaksi.js (path : ./xmart-nodejs/lib/models/transaksi.js)

```
import { Schema, model } from 'mongoose';

const transaksiSchema = new Schema({
    qrCode : {type : String, require : true},
    rfid : {type : String, require : true},
    hargaSatuan : {type : Number, require :true},
    jumlah : {type : Number, require : true},
    waktuPesan : Date
});

transaksiSchema.pre('save', function(next){
    let currentDate = new Date();
    this.waktuPesan = currentDate;
    next();
});

export default model('transaksi', transaksiSchema);
```

GraphQL Types :

- transaksi.js (path : ./xmart-nodejs/lib/graphql/types/transaksi.js)

```
import { GraphQLObjectType, GraphQLInputObjectType, GraphQLString, GraphQLID, G
raphQLInt } from 'graphql';

import GraphQLDate from 'graphql-date';

const transaksiType = new GraphQLObjectType({
    name : 'Transaksi',
    description : 'Transaksi Type',
    fields : () => ({
        _id : {type : GraphQLID},
        qrCode : {type : GraphQLString},
        rfid : {type : GraphQLString},
        hargaSatuan : {type : GraphQLInt},
        jumlah : {type : GraphQLInt},
        waktuPesan : {type : GraphQLDate}
    })
});

const transaksiInputType = new GraphQLInputObjectType({
    name : 'TransaksiInput',
    description : 'Transaksi Input Type',
    fields : () => ({
        qrCode : {type : GraphQLString},
        rfid : {type : GraphQLString},
        hargaSatuan : {type : GraphQLInt},
        jumlah : {type : GraphQLInt}
    })
});

export {transaksiType, transaksiInputType};
```

GraphQL Mutations :

- add.js (path : ./xmart-nodejs/lib/graphql/mutations/transaksi/add.js)

```
import { GraphQLNonNull } from 'graphql';
import axios from 'axios'

import { transaksiType, transaksiInputType } from '../../types/transaksi.js';
import TransaksiModel from '../../../models/transaksi.js';
import redisClient from '../../../models/redis.js';
```

```
const storeTransactions = async (transaction) => {
    try {
        const response = await axios.post('http://localhost:8080/api/transaksi'
, transaction)
        if(response.status === 200){
            console.log('Data berhasil di store ke postgreSQL')
        }
    } catch (error) {
        console.log(error)
    }
}

export default {
    type : transaksiType,
    args : {
        data: {
            name: 'data',
            type: new GraphQLNonNull(transaksiInputType)
        }
    },
    async resolve(root, params) {
        const transaksiModel = new TransaksiModel(params.data);
        const newTransaksi = await transaksiModel.save();
        if (!newTransaksi) {
            throw new Error('Terjadi kesalahan dalam menyimpan transaksi');
        }
        console.log(newTransaksi)
        storeTransactions(newTransaksi);
        redisClient.del('list-transaksi');
        return newTransaksi;
    }
}
```

- remove.js (path : ./xmart-nodejs/lib/graphql/mutations/transaksi/remove.js)

```
import { GraphQLNonNull, GraphQLID } from 'graphql';

import { transaksiType } from '../../types/transaksi.js';
import TransaksiModel from '../../../models/transaksi.js';
import redisClient from '../../../models/redis.js';

export default {
    type : transaksiType,
    args : {
        id: {
            name: 'id',
            type: new GraphQLNonNull(GraphQLID)
        }
    },
    resolve(root, params) {
        const removeTransaksi = TransaksiModel.findByIdAndRemove(params.id).exec
();
        if (!removeTransaksi) {
            throw new Error('Terjadi kesalahan dalam menghapus transaksi');
        }
        redisClient.del(`transaksi-${params.id}`);
        return removeTransaksi;
    }
}
```

- index.js (path : ./xmart-nodejs/lib/graphql/mutations/transaksi/index.js)

```
import AddTransaksi from './add.js';
import RemoveTransaksi from './remove.js';

export default {AddTransaksi, RemoveTransaksi};
```

- index.js (path : ./xmart-nodejs/lib/graphql/mutations/index.js)

```
import TransaksiMutation from './transaksi/index.js';
```

```
export default {...TransaksiMutation};
```

GraphQL Queries :

- single.js (path : ./xmart-nodejs/lib/graphql/queries/transaksi/single.js)

```
import { GraphQLID, GraphQLNonNull } from 'graphql';

import { transaksiType } from '../../types/transaksi.js';
import TransaksiModel from '../../../models/transaksi.js';
import redisClient from '../../../models/redis.js';

const DEFAULT_EXPIRATION = 3600;

export default {
    type : transaksiType,
    args : {
        id: {
            name: 'id',
            type: new GraphQLNonNull(GraphQLID)
        }
    },
    async resolve (root, params) {
        const cacheValue = await redisClient.get(`transaksi-${params.id}`);
        if (cacheValue) {
            const dataJSON = JSON.parse(cacheValue);
            dataJSON.waktuPesan = new Date(dataJSON.waktuPesan);
            return dataJSON;
        } else {
            const transaksi = await TransaksiModel.findById(params.id).lean();
            if (!transaksi) {
                throw new Error(`Gagal mendapatkan transaksi dengan id ${params
.id}`);
            }
            redisClient.setEx(`transaksi-${params.id}`, DEFAULT_EXPIRATION, JSON
.stringify(transaksi));
            return transaksi;
        }
    }
}
```

- multiple.js (path : ./xmart-nodejs/lib/graphql/queries/transaksi/multiple.js)

```
import { GraphQLList } from 'graphql';
// const { createClient } = require('redis');
// const redisClient = createClient();
const DEFAULT_EXPIRATION = 3600;

import { transaksiType } from '../../types/transaksi.js';
import TransaksiModel from '../../../models/transaksi.js';
import redisClient from '../../../models/redis.js';

export default {
    type : new GraphQLList(transaksiType),
    async resolve(root, params) {
        const cacheValue = await redisClient.get('list-transaksi');
        if (cacheValue) {
            const dataJSON = JSON.parse(cacheValue);
            dataJSON.map((data) => {
                data.waktuPesan = new Date(data.waktuPesan);
            });
            return dataJSON;
        } else {
            const transaksi = await TransaksiModel.find().lean();
            if (!transaksi) {
                throw new Error('Gagal mendapatkan transaksi');
            }
            redisClient.setEx("list-transaksi", DEFAULT_EXPIRATION, JSON.string
ify(transaksi));
            return transaksi;
        }
    }
}
```

- index.js (path : ./xmart-nodejs/lib/graphql/queries/transaksi/index.js)

```
import Transaksi from './single.js';
import ListTransaksi  from './multiple.js';

export default {Transaksi, ListTransaksi};
```

- index.js (path : ./xmart-nodejs/lib/graphql/queries/index.js)

```
import TransaksiQuery from './transaksi/index.js';

export default {...TransaksiQuery};
```

GraphQL Index :

- index.js (path : ./xmart-nodejs/lib/graphql/index.js)

```
import { GraphQLObjectType, GraphQLSchema } from 'graphql';

import mutations from './mutations/index.js';
import queries from './queries/index.js';

export default new GraphQLSchema({
    query : new GraphQLObjectType({
        name : 'Query',
        fields : queries
    }),
    mutation : new GraphQLObjectType({
        name : 'Mutation',
        fields : mutations
    })
});
```

Main server :

- server.js (path : ./xmart-nodejs/server.js)

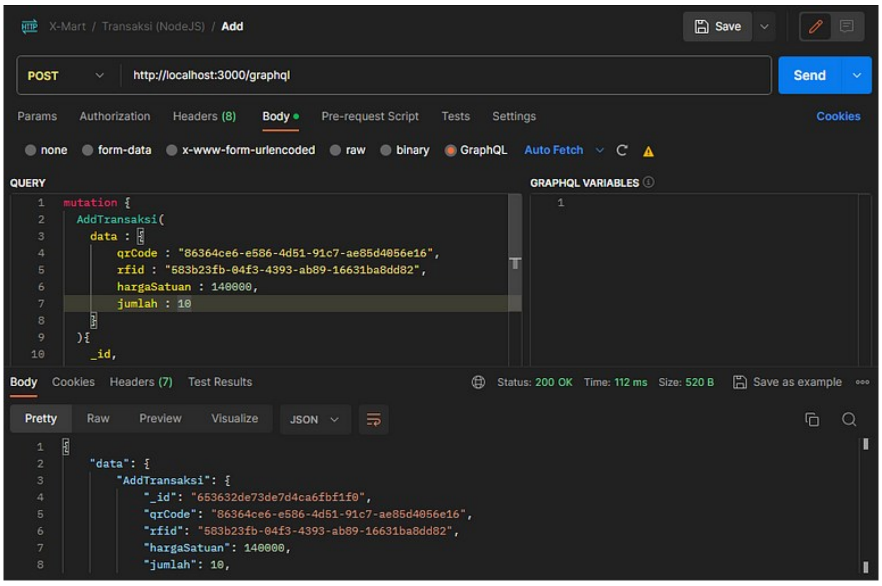
```
import express, { json } from 'express';
import cors from 'cors';
import { connect } from 'mongoose';
import { graphqlHTTP as graphQLHTTP } from 'express-graphql';
// const { createClient } = require('redis');
import schema from './lib/graphql/index.js';

const app = express();
// const redisClient = createClient();
// const DEFAULT_EXPIRATION = 3600;
const port = 3000;
const mongodbURL = 'mongodb://127.0.0.1:27017';

//MongoDB Connection
connect(mongodbURL, {'dbName' : 'XMart'})
.then( result => console.log('***** Connected to MongoDB *****'))
.catch( error => console.log(error));

//Middleware Setup
app.use(json());
app.use(cors());

app.get('/', (req, res, next) => {
    res.send('Hello World');
});
```

```
app.use('/graphql', graphQLHTTP({
    schema,
    graphiql : true,
    pretty : true
}));

app.listen(port, ()=> {
    console.log(`GraphQL server running on http://localhost:${port}/graphql`);
});
```

Testing API Express JS

Sama seperti backend pada java spring boot, service API express js pun perlu diuji. Method yang digunakan pada service ini hanya POST dan GET transaksi. Pada saat pengujian, pastikan server pada java spring boot telah aktif karena server express js membutuhkan API untuk store data transaksi ke postgreSQL melalui java spring boot. Berikut adalah detail pengujiannya.

Add Transaksi :



POST http://localhost:3000/graphql

Body: GraphQL

QUERY

```
mutation {
  AddTransaksi(
    data : {
        qrCode : "86364ce6-e586-4d51-91c7-ae85d4056e16",
        rfid : "583b23fb-04f3-4393-ab89-16631ba8dd82",
        hargaSatuan : 140000,
        jumlah : 10
    }
  ){
    _id,
```

GRAPHQL VARIABLES

Body

Status: 200 OK Time: 112 ms Size: 520 B

```
{
    "data": {
        "AddTransaksi": {
            "_id": "653632de73de7d4ca6fbf1f0",
            "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
            "rfid": "583b23fb-04f3-4393-ab89-16631ba8dd82",
            "hargaSatuan": 140000,
            "jumlah": 10,
```

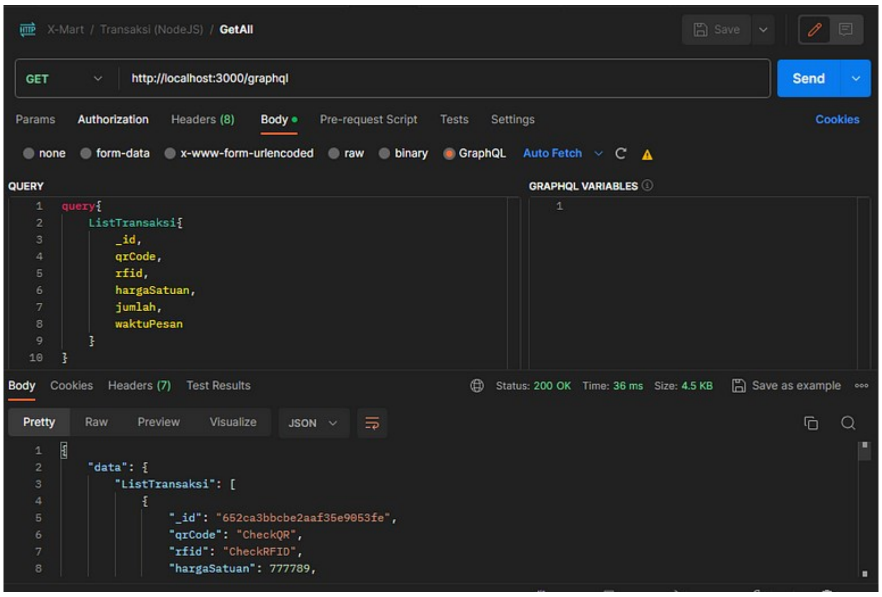
- query

```
mutation {
  AddTransaksi(
    data : {
        qrCode : "86364ce6-e586-4d51-91c7-ae85d4056e16",
        rfid : "583b23fb-04f3-4393-ab89-16631ba8dd82",
        hargaSatuan : 140000,
        jumlah : 10
    }
  ){
    _id,
    qrCode,
    rfid,
    hargaSatuan,
    jumlah,
    waktuPesan
  }
}
```

- response body

```
{
    "data": {
        "AddTransaksi": {
            "_id": "653632de73de7d4ca6fbf1f0",
            "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
            "rfid": "583b23fb-04f3-4393-ab89-16631ba8dd82",
            "hargaSatuan": 140000,
            "jumlah": 10,
            "waktuPesan": "2023-10-23T08:46:22.847Z"
        }
    }
}
```

Get Transaksi :



```
query{
    ListTransaksi{
        _id,
        qrCode,
        rfid,
        hargaSatuan,
        jumlah,
        waktuPesan
    }
}
```

```
{
    "data": {
        "ListTransaksi": [
            {
                "_id": "652ca3bbcbe2aaf36e9053fe",
                "qrCode": "CheckQR",
                "rfid": "CheckRFID",
                "hargaSatuan": 777789,
```

- query

```
query{
    ListTransaksi{
        _id,
        qrCode,
        rfid,
        hargaSatuan,
        jumlah,
        waktuPesan
    }
}
```

- response body

```
{
    "data": {
        "ListTransaksi": [
            {
                "_id": "652ca3bbcbe2aaf35e9053fe",
                "qrCode": "CheckQR",
                "rfid": "CheckRFID",
                "hargaSatuan": 777789,
                "jumlah": 10,
                "waktuPesan": "2023-10-16T02:45:15.175Z"
            },
            {
                "_id": "652cb897cbe2aaf35e905404",
                "qrCode": "e889a3ce-364c-4d78-856f-46a6c703766a",
                "rfid": "f7cffe98-8954-4682-ba9a-c3744d1efd5c",
                "hargaSatuan": 50000,
                "jumlah": 3,
                "waktuPesan": "2023-10-16T04:14:15.300Z"
            },
            {
                "_id": "652cb897cbe2aaf35e905406",
                "qrCode": "e889a3ce-364c-4d78-856f-46a6c703766a",
                "rfid": "583b23fb-04f3-4393-ab89-16631ba8dd82",
                "hargaSatuan": 140000,
                "jumlah": 2,
                "waktuPesan": "2023-10-16T04:14:15.315Z"
            },
            {
                "_id": "652cb897cbe2aaf35e905408",
                "qrCode": "e889a3ce-364c-4d78-856f-46a6c703766a",
                "rfid": "a6d05791-59f5-40a4-965f-6fcff82c4354",
                "hargaSatuan": 180000,
                "jumlah": 5,
                "waktuPesan": "2023-10-16T04:14:15.321Z"
            },
            {
                "_id": "652cb8cccbe2aaf35e90540c",
                "qrCode": "27a5ee6e-fde2-4740-a620-597d6c839d13",
                "rfid": "a6d05791-59f5-40a4-965f-6fcff82c4354",
                "hargaSatuan": 180000,
                "jumlah": 2,
                "waktuPesan": "2023-10-16T04:15:08.973Z"
            },
            {
                "_id": "652cb8cccbe2aaf35e90540e",
                "qrCode": "27a5ee6e-fde2-4740-a620-597d6c839d13",
                "rfid": "820544d3-fa18-4965-ba87-10ad2e2195d0",
                "hargaSatuan": 50000,
                "jumlah": 5,
                "waktuPesan": "2023-10-16T04:15:08.985Z"
            },
            {
                "_id": "652cb8cccbe2aaf35e905410",
                "qrCode": "27a5ee6e-fde2-4740-a620-597d6c839d13",
                "rfid": "d8467bb2-aa47-40e3-a94c-d4eb87b38d56",
                "hargaSatuan": 300000,
                "jumlah": 1,
                "waktuPesan": "2023-10-16T04:15:08.997Z"
            },
            {
                "_id": "652cb8eacbe2aaf35e905412",
                "qrCode": "ca390478-a949-4ac0-8f28-dc45e5dffb52",
                "rfid": "d8467bb2-aa47-40e3-a94c-d4eb87b38d56",
                "hargaSatuan": 300000,
                "jumlah": 5,
                "waktuPesan": "2023-10-16T04:15:38.856Z"
            },
            {
                "_id": "652cb8eacbe2aaf35e905414",
                "qrCode": "ca390478-a949-4ac0-8f28-dc45e5dffb52",
                "rfid": "ecf7d9c3-2d1c-4625-897e-b571f47c7e2e",
                "hargaSatuan": 30000,
                "jumlah": 1,
                "waktuPesan": "2023-10-16T04:15:38.867Z"
            },
            {
                "_id": "652cb8eacbe2aaf35e905416",
                "qrCode": "ca390478-a949-4ac0-8f28-dc45e5dffb52",
                "rfid": "33cecfe8-049e-49b0-8508-668c9462489f",
                "hargaSatuan": 60000,
                "jumlah": 2,
                "waktuPesan": "2023-10-16T04:15:38.879Z"
            },
            {
                "_id": "652cb90bcbe2aaf35e905418",
                "qrCode": "311e9379-fc67-4a50-a739-1befefa0d94a",
                "rfid": "ecf7d9c3-2d1c-4625-897e-b571f47c7e2e",
                "hargaSatuan": 30000,
                "jumlah": 1,
                "waktuPesan": "2023-10-16T04:16:11.030Z"
            },
            {
                "_id": "652cb90bcbe2aaf35e90541a",
                "qrCode": "311e9379-fc67-4a50-a739-1befefa0d94a",
                "rfid": "33cecfe8-049e-49b0-8508-668c9462489f",
                "hargaSatuan": 60000,
                "jumlah": 3,
                "waktuPesan": "2023-10-16T04:16:11.052Z"
            },
            {
                "_id": "652cb90bcbe2aaf35e90541c",
                "qrCode": "311e9379-fc67-4a50-a739-1befefa0d94a",
                "rfid": "fcb7daff-3471-4c51-b1dd-1b3e2afd7454",
                "hargaSatuan": 60000,
                "jumlah": 5,
                "waktuPesan": "2023-10-16T04:16:11.060Z"
            },
            {
                "_id": "652d05ff7ca051fbdbc1b8e9",
                "qrCode": "e889a3ce-364c-4d78-856f-46a6c703766a",
                "rfid": "f7cffe98-8954-4682-ba9a-c3744d1efd5c",
                "hargaSatuan": 50000,
                "jumlah": 4,
                "waktuPesan": "2023-10-16T09:44:31.212Z"
            },
            {
                "_id": "652d06007ca051fbdbc1b8eb",
                "qrCode": "e889a3ce-364c-4d78-856f-46a6c703766a",
                "rfid": "583b23fb-04f3-4393-ab89-16631ba8dd82",
                "hargaSatuan": 140000,
                "jumlah": 2,
                "waktuPesan": "2023-10-16T09:44:32.342Z"
            },
            {
                "_id": "653632de73de7d4ca6fbf1f0",
                "qrCode": "86364ce6-e586-4d51-91c7-ae85d4056e16",
                "rfid": "583b23fb-04f3-4393-ab89-16631ba8dd82",
                "hargaSatuan": 140000,
                "jumlah": 10,
                "waktuPesan": "2023-10-23T08:46:22.847Z"
            }
        ]
    }
}
```

React JS + Redux

Pada sisi frontend, framework yang digunakan adalah react JS yang digenerate menggunakan vite JS dan menggunakan Tailwind CSS sebagai framework CSS nya. Beberapa page yang digunakan diantaranya adalah home page sebagai menu utama dari suatu aplikasi, customer detail page sebagai akses login customer dan melihat detail info customer, cart page sebagai halaman untuk customer memilih barang yang tersedia dan dimasukan ke keranjang, transaction detail page sebagai halaman untuk pengguna mengonfirmasi transaksi, market info page untuk melihat data customer, barang, dan transaksi dari postgreSQL, dan terakhir transaction history page sebagai riwayat transaksi dari MongoDB. Untuk menyimpan data customer dan keranjang pada sisi frontend, digunakan state management Redux untuk store data tersebut. Berikut adalah detail project react JS nya.

Struktur Folder :

```
frontend
├── dist
├── node_modules
├── public
├── src
│   ├── assets
│   ├── components
│   │   ├── CartPage.jsx
│   │   ├── CustomerDetailPage.jsx
│   │   ├── HomePage.jsx
│   │   ├── ListTransactionPage.jsx
│   │   ├── MarketInfoPage.jsx
│   │   └── TransactionDetailPage.jsx
│   ├── redux
│   │   ├── slice
│   │   │   ├── cartSlice.js
│   │   │   └── customerSlice.js
│   │   └── store.js
│   ├── service
│   │   └── service.js
│   ├── App.css
│   ├── App.jsx
│   ├── index.css
│   └── main.jsx
├── .eslintrc.cjs
├── .gitignore
├── index.html
├── package-lock.json
├── package.json
├── postcss.config.js
├── README.md
├── tailwind.config.js
└── vite.config.js
```

Configurasi File package.json :

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-war
nings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@reduxjs/toolkit": "^1.9.7",
    "axios": "^1.5.1",
    "html5-qrcode": "^2.3.8",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.11.0",
    "react-qr-code": "^2.0.12",
    "react-redux": "^8.1.3",
    "react-router-dom": "^6.16.0",
    "redux": "^4.2.1"
  },
  "devDependencies": {
    "@types/react": "^18.2.15",
    "@types/react-dom": "^18.2.7",
    "@vitejs/plugin-react": "^4.0.3",
    "autoprefixer": "^10.4.16",
    "eslint": "^8.45.0",
    "eslint-plugin-react": "^7.32.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.3",
    "postcss": "^8.4.31",
    "tailwindcss": "^3.3.3",
    "vite": "^4.4.5"
  }
}
```

Redux :

- cartSlice.js (path : ./frontend/src/redux/slice/cartSlice.js)

```
import { createSlice } from '@reduxjs/toolkit'

const cartSlice = createSlice({
    name : 'cart',
    initialState : [],
    reducers : {
        addToCart(state, action){
            const itemExist = state.some((item) => item.rfid === action.payload.
rfid)
            if(itemExist){
                state = state.map((item) => {
                    if(action.payload.rfid === item.rfid){
                        item.jumlah = item.jumlah + 1
                    }
                    return item
                })
            } else {
                state.push({...action.payload, jumlah : 1})
            }
        },
        increaseItem(state, action){
            state = state.map((item) => {
                if(action.payload === item.rfid){
                    item.jumlah = item.jumlah + 1
                }
                return item
            })
        },
        decreaseItem(state, action){
            state = state.map((item) => {
                if(action.payload === item.rfid && item.jumlah > 1){
                    item.jumlah = item.jumlah - 1
                }
                return item
            })
        },
        removeItem(state, action){
            return state.filter((item) => item.rfid !== action.payload)
        },
        clearCart(state, action){
            return []
        }
    }
})


export const { addToCart, increaseItem, decreaseItem, removeItem, clearCart } =
 cartSlice.actions
export default cartSlice.reducer
```

- customerSlice.js (path :./frontend/src/redux/slice/customerSlice.js)

```
import { createSlice } from '@reduxjs/toolkit'

const customerSlice = createSlice({
    name : 'customer',
    initialState : {},
    reducers : {
        setCustomer(state, action){
            console.log(action.payload)
            return action.payload
        },
        clearCustomer(state, action){
            state = {}
        }
    }
})

export const { setCustomer, clearCustomer } = customerSlice.actions
export default customerSlice.reducer
```

- store.js (path :./frontend/src/redux/store.js)

```
import { configureStore } from '@reduxjs/toolkit'
import cartReducer from './slices/cartSlice'
import customerReducer from './slices/customerSlice'
```

```
export const store = configureStore({
    reducer : {
        cart : cartReducer,
        customer : customerReducer
    }
})
```

Service :

- service.js (path :./frontend/src/service/service.js)

```
import axios from 'axios'

const getCustomer = async () => {
    try {
        const response = await axios.get('http://localhost:8080/api/customer')
        return response
    } catch (error) {
        console.log(error)
    }
}

const getCustomerById = async (id) => {
    try {
        const response = await axios.get(`http://localhost:8080/api/customer/${
id}`)
        return response
    } catch (error) {
        console.log(error)
    }
}

const getItems = async () => {
    try {
        const response = await axios.get('http://localhost:8080/api/barang')
        return response
    } catch (error) {
        console.log(error)
    }
}

const getTransactions = async () => {
    try {
        const response = await axios.get('http://localhost:8080/api/transaksi')
        return response
    } catch (error) {
        console.log(error)
    }
}

const getTransactionsSplit = async () => {
    try {
        const response = await axios.get('http://localhost:8080/api/transaksi-s
plit')
        return response
    } catch (error) {
        console.log(error)
    }
}

const getTransactionsFromMDB = async () => {
    try {
        const endpoint = 'http://localhost:3000/graphql'
        const graphqlQuery = {
            query : `
                query {
                    ListTransaksi{
                        _id,
                        qrCode,
                        rfid,
                        hargaSatuan,
                        jumlah,
                        waktuPesan
                    }
                }
            `
        }
        const response = await axios.post(endpoint, graphqlQuery)
        return response
    } catch (error) {
        console.log(error)
    }
}
```

```
const addTransactionToMDB = async (transaction) => {
    try {
        const endpoint = 'http://localhost:3000/graphql'
        const graphqlMutation = {
            query : `
            mutation {
                AddTransaksi(
                    data : {
                        qrCode : "${transaction.qrCode}",
                        rfid : "${transaction.rfid}",
                        hargaSatuan : ${transaction.hargaSatuan},
                        jumlah : ${transaction.jumlah}
                    }
                ){
                    _id,
                    qrCode,
                    rfid,
                    hargaSatuan,
                    jumlah,
                    waktuPesan
                }
            }
        `
        }
        const response = await axios.post(endpoint, graphqlMutation)
        console.log(response)
        return response
    } catch (error) {
        console.log(error)
    }
}

export { getCustomer, getCustomerById, getItems, getTransactions, getTransactio
nsFromMDB, getTransactionsSplit, addTransactionToMDB }
```

Components :

- CartPage.jsx (path :./frontend/src/components/CartPage.jsx)

```
import React, { useEffect, useState} from 'react'
import {BsCartPlusFill} from 'react-icons/bs'
import {FiShoppingBag} from 'react-icons/fi'
import {IoIosAddCircle, IoIosRemoveCircle, IoIosTrash} from 'react-icons/io'
import { addToCart, increaseItem, decreaseItem, removeItem, clearCart } from '.
./redux/slices/cartSlice'
import { setCustomer, clearCustomer } from '../redux/slices/customerSlice'
import { useDispatch, useSelector } from 'react-redux'
import { useNavigate } from 'react-router-dom'
import { getItems } from '../service/service'

const CartPage = () => {
 const [listItem, setListItem] = useState([])
 const [customer, setCustomer] = useState({})
 const carts = useSelector((state) => state.cart)
 const customerSaved = useSelector((state) => state.customer)
 const navigate = useNavigate()

 useEffect(() => {
  if(Object.keys(customerSaved).length !== 0){
   setCustomer(customerSaved)
   getListItem()
  } else {
   alert('Please Input QR Customer first !')
   navigate('/customer-detail')
  }
 }, [])

 const getListItem = async () => {
  try {
   const listItemSaved = await getItems()
   setListItem(listItemSaved.data)
  } catch (error) {
   console.log(error)
  }
 }
```

```
  const getTotalPrice = () => {
   if(carts.length === 0){
    return 0
   } else {
    let totalPrice = 0
    carts.forEach((item) => {
     totalPrice = totalPrice + item.hargaSatuan * item.jumlah
    })
    return totalPrice
   }
  }

  return (
   <div className='flex flex-row w-full h-full gap-5 p-5'>
    <div className='bg-slate-400 rounded-2xl w-2/3 p-5 overflow-auto'>
     <button className='text-xl p-2 bg-blue-600 rounded-full self-center' onClic
k={() => navigate('/customer-detail')}>Kembali</button>
      <h1 className='text-3xl text-center mb-5'>Item List</h1>
     <div className='flex flex-row flex-wrap gap-5'>
      {
       listItem.map((item) => {
        return (
         <Card
          namaBarang={item.namaBarang}
          hargaSatuan={item.hargaSatuan}
          rfid={item.rfid}
         />
        )
       })
      }
     </div>
    </div>
    <div className='flex flex-col bg-slate-400 rounded-2xl w-1/3 p-5'>
     <h1 className='text-3xl text-center mb-5'>Cart</h1>
     <div className='flex flex-col flex-1 gap-5 overflow-auto'>
      {
       carts.map((item) => {
        return (
         <CartList
          hargaSatuan={item.hargaSatuan}
          namaBarang={item.namaBarang}
          rfid={item.rfid}
          jumlah={item.jumlah}
         />
        )
       })
      }
     </div>
     <div className='flex flex-col h-1/4 mt-2 '>
      <h2 className='text-xl'>Total Harga : Rp{getTotalPrice()}</h2>
      <h2 className='text-xl'>Customer : {customer.nama}</h2>
      <h2 className='text-xl'>Wallet : {customer.wallet}</h2>
      {
       (carts.length === 0)
        ?  <div className='text-xl p-2 bg-gray-400 rounded-full self-center'>
          Keranjang Kosong
         </div>
        :  <button
          className='text-xl p-2 bg-blue-600 rounded-full self-center'
          onClick={()=>navigate('/transaction-detail')}>
           Buat Transaksi
         </button>
      }
     </div>
    </div>
   </div>
  )
}
```

```
const Card = ({namaBarang, hargaSatuan, rfid}) => {
 const dispatch = useDispatch()
 return (
  <div className='w-1/5 aspect-square bg-slate-300 rounded-xl flex flex-col ite
ms-center p-1'>
   <div className='w-2/3 h-2/3 grid place-items-center'>
    <FiShoppingBag size={75}/>
   </div>
   <div className='flex flex-row justify-between items-center w-full'>
    <div className='flex flex-col'>
     <div className='text-base'>{namaBarang}</div>
     <div className='text-base font-bold'>Rp{hargaSatuan}</div>
    </div>
    <button
     className='w-10 h-10 bg-slate-400 grid place-items-center rounded-full'
     onClick={()=>dispatch(addToCart({
      namaBarang : namaBarang,
      hargaSatuan : hargaSatuan,
      rfid : rfid
     }))}>
      <BsCartPlusFill size={25} />
    </button>
   </div>
  </div>
 )
}

const CartList = ({namaBarang, hargaSatuan, rfid, jumlah}) => {
 const dispatch = useDispatch()
 return (
  <div className='flex flex-row justify-between items-center bg-slate-300 round
ed-xl px-2'>
   <div className='flex flex-col'>
    <h2 className='text-base'>{namaBarang}</h2>
    <h2 className='text-base font-bold'>Rp{hargaSatuan*jumlah}</h2>
   </div>
   <div className='flex flex-row'>
    <button><IoIosRemoveCircle size={30} onClick={()=>dispatch(decreaseItem(rfi
d))}/></button>
    <h3 className='text-xl mx-3'>{jumlah}</h3>
    <button><IoIosAddCircle size={30} onClick={()=>dispatch(increaseItem(rfid))
} /></button>
    <button><IoIosTrash size={30} color='#ff0000' onClick={()=>dispatch(removeI
tem(rfid))}/></button>
   </div>
  </div>
 )
}

export default CartPage
```

- CustomerDetailPage.jsx (path :./frontend/src/components/CustomerDetailPage.jsx)

```
import React, { useState } from 'react';
import { Html5QrcodeScanner, Html5QrcodeSupportedFormats } from 'html5-qrcode';
import QRCode from 'react-qr-code'
import { useDispatch, useSelector } from 'react-redux';
import { setCustomer, clearCustomer } from '../redux/slices/customerSlice'
import { useNavigate } from 'react-router-dom';
import { getCustomerById } from '../service/service'

const CustomerDetailPage = () => {
 const [scanResult, setScanResult] = useState()
 const [customer, setCustomer] = useState()
 const navigate = useNavigate()
 const startScanner = () => {
  const scanner = new Html5QrcodeScanner('reader', {
   qrbox: {
    width: 500,
    height: 500
   },
   fps: 1,
   disableFlip : false,
   formatsToSupport : [
    Html5QrcodeSupportedFormats.QR_CODE,
    Html5QrcodeSupportedFormats.CODE_128
   ]
  }, false);
  scanner.render(async (result) => {
   const customerExist = await checkCustomer(result)
   if(customerExist){
    setCustomer({
     nama : customerExist.nama,
     wallet : customerExist.wallet,
     qrCode : customerExist.qrCode
    })
   } else {
    setCustomer()
   }
   setScanResult(result)
   scanner.clear()
  }, (error) => {
   console.warn(error)
  });
 }

 const checkCustomer = async (qrResult) => {
  const customerExist = await getCustomerById(qrResult)
  if(customerExist){
   return customerExist.data
  } else {
   return null
  }
 }

 return (
  <div className='flex flex-row w-full h-full gap-5 p-5'>
   <div className='bg-slate-400 rounded-2xl w-1/2 p-5 flex flex-col items-cente
r'>
    <h1 className='text-3xl text-center mb-5'>QR Scanner for Customer</h1>
    <div id='reader' className='w-5/6 flex-1'></div>
    <div className='flex flex-row w-full justify-evenly items-center'>
     <button className='text-xl p-2 bg-blue-600 rounded-full self-center' onCli
ck={() => navigate('/')}>Kembali</button>
     <button className='text-xl p-2 bg-blue-600 rounded-full self-center' onCli
ck={()=>startScanner()}>Start Scanner</button>
    </div>
   </div>
   <div className='flex flex-col bg-slate-400 rounded-2xl w-1/2 p-5'>
    <h1 className='text-3xl text-center mb-5'>Detail Customer</h1>
    {
     (scanResult)
      ? (customer)
       ?
        <CustomerInfoCard
         customer={customer}
        />
       :
        <div className='flex-1 grid place-items-center'>
         <h2 className='text-xl p-2 bg-red-400 rounded-full self-center'>
          Customer with this QR Code is not found, please scan again
         </h2>
        </div>
      :
      <div className='flex-1 grid place-items-center'>
       <h2 className='text-xl p-2 bg-green-400 rounded-full self-center'>
        Scan QRCode for showing detail customer
       </h2>
      </div>
    }
   </div>
  </div>
 );
};

const CustomerInfoCard = ({customer}) => {
 const dispatch = useDispatch()
 const navigate = useNavigate()
 const startShopping = (customer) => {
  const data = dispatch(setCustomer(customer))
  if(data.payload){
   navigate('/cart')
  }
 }
 return (
  <div className='flex flex-col flex-1 justify-between items-center'>
   <div className='flex flex-col items-center'>
    <h2 className='text-xl'>QR Code Customer :</h2>
    <h2 className='text-xl font-bold'>{customer.qrCode}</h2>
    <QRCode
     value={customer.qrCode}
    />
   </div>
   <div className='flex flex-col items-center'>
    <h2 className='text-xl'>Nama Customer :</h2>
    <h2 className='text-xl font-bold'>{customer.nama}</h2>
   </div>
   <div className='flex flex-col items-center'>
    <h2 className='text-xl'>Wallet :</h2>
    <h2 className='text-xl font-bold'>{customer.wallet}</h2>
   </div>
    <button className='text-xl p-2 bg-blue-600 rounded-full self-center' onClic
k={() => startShopping(customer)}>Mulai Belanja</button>
  </div>
 )
}

export default CustomerDetailPage
```

- HomePage.jsx (path :./frontend/src/components/HomePage.jsx)

```
import React, { useEffect } from 'react'
import { useNavigate } from 'react-router-dom'

const HomePage = () => {
 const navigate = useNavigate()
 const checkTimeZone = () => {
  let currentDate = new Date(new Date().toLocaleString('en', {timeZone: 'Asia/J
akarta'}));
  let currentDateGlobal = new Date()
  console.log(currentDate)
  console.log(currentDateGlobal)
 }
 return (
  <div className='w-full h-full grid place-items-center' onClick={() => checkTi
meZone()}>
   <div className='bg-slate-400 flex flex-col justify-evenly items-center w-72
h-72 rounded-2xl'>
    <h1 className='text-3xl'>X-Mart</h1>
    <button className='bg-slate-300 p-2 rounded-2xl' onClick={() => navigate('/
customer-detail')}>Mulai Belanja</button>
    <button className='bg-slate-300 p-2 rounded-2xl' onClick={() => navigate('/
market-info')}>Market Info</button>
    <button className='bg-slate-300 p-2 rounded-2xl' onClick={() => navigate('/
transaction-list')} >Riwayat Transaksi</button>
   </div>
  </div>
 )
}

export default HomePage
```

- ListTransactionPage.jsx (path :./frontend/src/components/ListTransactionPage.jsx)

```
import React, { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { getTransactionsFromMDB } from '../service/service'

const ListTransactionPage = () => {
 const [historyTrx, setHistoryTrx] = useState([])
 const navigate = useNavigate()

 useEffect(() => {
  fetchHistoryTrx()
 }, [])
```

```
  const fetchHistoryTrx = async () => {
   try {
    const response = await getTransactionsFromMDB()
    setHistoryTrx(response.data.data.ListTransaksi)
   } catch (error) {
    console.log(error)
   }
  }

  const checkCurrentDate = (date) => {
   const currentDate = new Date(date)
   console.log(currentDate)
  }

  return (
   <div className='w-full h-full grid place-items-center'>
    <div className='w-5/6 h-5/6 bg-slate-400 rounded-3xl flex flex-col'>
     <h2 className='text-3xl text-center mb-5' onClick={() => getTransactionsFro
mMDB()}>Riwayat Transaksi</h2>
     <div className='flex-1 overflow-auto p-2'>
      <table className='table-auto border border-slate-800 border-collapse mt-5
mx-auto'>
       <thead className='border border-slate-800 bg-slate-500 font-bold'>
        <tr className='border border-slate-800'>
         <td className='p-2 border border-slate-800'>QR Code</td>
         <td className='p-2 border border-slate-800'>RFID</td>
         <td className='p-2 border border-slate-800'>Harga Satuan</td>
         <td className='p-2 border border-slate-800'>Jumlah</td>
         <td className='p-2 border border-slate-800'>Waktu Pesan</td>
        </tr>
       </thead>
       <tbody className='border border-slate-800'>
        {
         historyTrx.map((item, index) => {
          return (
           <tr className={`border border-slate-800 ${(index % 2 === 0)? 'bg-slat
e-300' : 'bg-slate-200'}`}>
            <td className='p-2 border border-slate-800'>{item.qrCode}</td>
            <td className='p-2 border border-slate-800'>{item.rfid}</td>
            <td className='p-2 border border-slate-800'>Rp{item.hargaSatuan}</td
>
            <td className='p-2 border border-slate-800'>{item.jumlah}</td>
            <td className='p-2 border border-slate-800' onClick={() => checkCurr
entDate(item.waktuPesan)} >{Date(item.waktuPesan)}</td>
           </tr>
          )
         })
        }
       </tbody>
      </table>
     </div>
     <button className='m-2 p-2 self-center bottom-2 bg-blue-500 rounded-3xl' on
Click={()=> navigate('/')}>Kembali ke home</button>
    </div>
   </div>
  )
}

export default ListTransactionPage
```

- MarketInfoPage.jsx (path :./frontend/src/components/MarketInfoPage.jsx)

```
import React, { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { getCustomer, getItems, getTransactions, getTransactionsSplit } from '.
./service/service'
```

```
const MarketInfoPage = () => {
 const [activeContent, setActiveContent] = useState('transaction')
 const navigate = useNavigate()

 return (
  <div className='w-full h-full grid place-items-center'>
   <div className='w-5/6 h-5/6 bg-slate-400 rounded-3xl flex flex-col mb-5 over
flow-auto'>
    <div className='flex flex-row justify-evenly items-center'>
     <button
      className={
       `m-2 p-2 self-center bottom-2 rounded-3xl text-3xl text-center
       ${(activeContent === 'customer')? 'bg-blue-500' : 'bg-blue-300'}`
       }
      onClick={()=> setActiveContent('customer')}>
        Customer
     </button>
     <button
      className={
       `m-2 p-2 self-center bottom-2 rounded-3xl text-3xl text-center
       ${(activeContent === 'items')? 'bg-blue-500' : 'bg-blue-300'}`
       }
      onClick={()=> setActiveContent('items')}>
        Barang
     </button>
     <button
      className={
       `m-2 p-2 self-center bottom-2 rounded-3xl text-3xl text-center
       ${(activeContent === 'transaction')? 'bg-blue-500' : 'bg-blue-300'}`
       }
      onClick={()=> setActiveContent('transaction')}>
        Transaksi
     </button>
    </div>
    <div className='h-fit p-2 overflow-auto'>
    {
     {
      'transaction' : <TransactionTable/>,
      'customer' : <CustomerTable/>,
      'items' : <ItemsTable/>
     }[activeContent]
    }
    </div>
    <button className='m-2 p-2 self-center bottom-2 bg-blue-500 rounded-3xl' on
Click={()=> navigate('/')}>Kembali ke home</button>
   </div>
  </div>
 )
}

const CustomerTable = () => {
 const [listCustomer, setListCustomer] = useState([])

 useEffect(() => {
  fetchListCustomer()
 }, [])
 const fetchListCustomer = async () => {
  try {
   const response = await getCustomer()
   setListCustomer(response.data)
  } catch (error) {
   console.log(error)
  }
 }
 return (
  <table className='table-auto border border-slate-800 border-collapse mt-5 mx-
auto'>
   <thead className='border border-slate-800 bg-slate-500 font-bold'>
    <tr className='border border-slate-800' onClick={() => getCustomer()}>
     <td className='p-2 border border-slate-800'>Nama Customer</td>
     <td className='p-2 border border-slate-800'>Jenis Wallet</td>
     <td className='p-2 border border-slate-800'>QR Code</td>
    </tr>
   </thead>
   <tbody className='border border-slate-800'>
    {
     listCustomer.map((item, index) => {
      return (
       <tr className={`border border-slate-800 ${(index % 2 === 0)? 'bg-slate-3
00' : 'bg-slate-200'}`}>
        <td className='p-2 border border-slate-800'>{item.nama}</td>
        <td className='p-2 border border-slate-800'>{item.wallet}</td>
        <td className='p-2 border border-slate-800'>{item.qrCode}</td>
       </tr>
      )
     })
    }
   </tbody>
  </table>
 )
}

const TransactionTable = () => {
 const [historyTrxs, setHistoryTrxs] = useState([])

 useEffect(() => {
  fetchHistoryTrxs()
 }, [])

 const fetchHistoryTrxs = async () => {
  try {
   const response = await getTransactionsSplit()
   setHistoryTrxs(response.data)
  } catch (error) {
   console.log(error)
  }
 }

 return (
  <table className='table-auto border border-slate-800 border-collapse mt-5 mx-
auto h-full'>
   <thead className='border border-slate-800 bg-slate-500 font-bold'>
    <tr className='border border-slate-800'>
     <td className='p-2 border border-slate-800'>Nama</td>
     <td className='p-2 border border-slate-800'>Nama Barang</td>
     <td className='p-2 border border-slate-800'>Harga Satuan</td>
     <td className='p-2 border border-slate-800'>Jumlah</td>
     <td className='p-2 border border-slate-800'>Waktu Pesan</td>
    </tr>
   </thead>
   <tbody className='border border-slate-800'>
    {
     historyTrxs.map((item, index) => {
      return (
       <tr className={`border border-slate-800 ${(index % 2 === 0)? 'bg-slate-3
00' : 'bg-slate-200'}`}>
        <td className='p-2 border border-slate-800'>{item.nama}</td>
        <td className='p-2 border border-slate-800'>{item.namaBarang}</td>
        <td className='p-2 border border-slate-800'>Rp{item.hargaSatuan}</td>
        <td className='p-2 border border-slate-800'>{item.jumlah}</td>
        <td className='p-2 border border-slate-800'>{Date(item.waktuPesan)}</td
>
       </tr>
      )
     })
    }
   </tbody>
  </table>
 )
}

const ItemsTable = () => {
 const [listItem, setListItem] = useState([])

 useEffect(() => {
  fetchListItem()
 },[])
 const fetchListItem = async () => {
  try {
   const response = await getItems()
   setListItem(response.data)
  } catch (error) {
   console.log(error)
  }
 }
 return (
  <table className='table-auto border border-slate-800 border-collapse mt-5 mx-
auto'>
   <thead className='border border-slate-800 bg-slate-500 font-bold'>
    <tr className='border border-slate-800' onClick={() => getItems()}>
     <td className='p-2 border border-slate-800'>Nama Barang</td>
     <td className='p-2 border border-slate-800'>Harga Satuan</td>
     <td className='p-2 border border-slate-800'>RFID</td>
    </tr>
   </thead>
   <tbody className='border border-slate-800'>
    {
     listItem.map((item, index) => {
      return (
       <tr className={`border border-slate-800 ${(index % 2 === 0)? 'bg-slate-3
00' : 'bg-slate-200'}`}>
        <td className='p-2 border border-slate-800'>{item.namaBarang}</td>
        <td className='p-2 border border-slate-800'>Rp{item.hargaSatuan}</td>
        <td className='p-2 border border-slate-800'>{item.rfid}</td>
       </tr>
      )
     })
    }
   </tbody>
  </table>
 )
}

export default MarketInfoPage
```

- TransactionDetailPage.jsx (path :./frontend/src/components/TransactionDetailPage.jsx)

```
import React, {useEffect, useState} from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { useNavigate } from 'react-router-dom'
import { clearCart } from '../redux/slices/cartSlice'
import { clearCustomer } from '../redux/slices/customerSlice'
import { addTransactionToMDB } from '../service/service'

const TransactionDetailPage = () => {
 const [cart, setCart] = useState([])
 const [customer, setCustomer] = useState({})
 const cartSaved = useSelector((state) => state.cart)
 const customerSaved = useSelector((state) => state.customer)
 const navigate = useNavigate()
 const dispatch = useDispatch()

 useEffect(() => {
  fetchDataTransaction()
 }, [])

 const getTotalPrice = () => {
  if(cart.length === 0){
   return 0
  } else {
   let totalPrice = 0
   cart.forEach((item) => {
    totalPrice = totalPrice + item.hargaSatuan * item.jumlah
   })
   return totalPrice
  }
 }

 const fetchDataTransaction = () => {
  if(Object.keys(customerSaved).length === 0){
   alert('Please Input QR Customer first !')
   navigate('/customer-detail')
  } else if (cartSaved.length === 0){
   alert('Keranjang kosong !')
   navigate('/cart')
  } else {
   setCart(cartSaved)
   setCustomer(customerSaved)
  }
 }
```

```
  const saveTransaction = async () => {
   console.log(customerSaved)
   console.log(cartSaved)
   try {
    cartSaved.forEach(async (cart) => {
     const transaction = {
      qrCode : customerSaved.qrCode,
      rfid : cart.rfid,
      hargaSatuan : cart.hargaSatuan,
      jumlah : cart.jumlah
     }
     const response = await addTransactionToMDB(transaction)
     console.log(response)
    })
    alert('Transaksi berhasil tersimpan !')
    dispatch(clearCart())
    dispatch(clearCustomer())
    navigate('/')
   } catch (error) {
    alert('Transaksi gagal !')
    console.log(error)
   }
  }
  return (
   <div className='w-full h-full grid place-items-center'>
    <div className='w-1/2 h-5/6 bg-slate-400 rounded-3xl flex flex-col'>
     <h2 className='text-3xl text-center mb-5'>Detail Transaksi</h2>
     <div className='flex flex-row justify-evenly items-center'>
      <div className='flex flex-col justify-between items-start'>
       <h2 className='text-lg font-bold'>Pembeli : </h2>
       <h2 className='text-lg'>{customer.nama}</h2>
      </div>
      <div className='flex flex-col justify-between items-start'>
       <h2 className='text-lg font-bold'>Wallet : </h2>
       <h2 className='text-lg'>{customer.wallet}</h2>
      </div>
      <div className='flex flex-col justify-between items-start'>
       <h2 className='text-lg font-bold'>Total Bayar : </h2>
       <h2 className='text-lg'>Rp{getTotalPrice()}</h2>
      </div>
     </div>
     <div className='flex-1 overflow-auto'>
      <table className='table-auto border border-slate-800 border-collapse mt-5
mx-auto'>
       <thead className='border border-slate-800 bg-slate-500 font-bold'>
        <tr className='border border-slate-800'>
         <td className='p-2 border border-slate-800'>Nama Barang</td>
         <td className='p-2 border border-slate-800'>Jumlah</td>
         <td className='p-2 border border-slate-800'>Harga Satuan</td>
         <td className='p-2 border border-slate-800'>Total Harga</td>
        </tr>
       </thead>
       <tbody className='border border-slate-800'>
        {
         cart.map((item, index) => {
          return (
           <tr className={`border border-slate-800 ${(index % 2 === 0)? 'bg-slat
e-300' : 'bg-slate-200'}`}>
            <td className='p-2 border border-slate-800'>{item.namaBarang}</td>
            <td className='p-2 border border-slate-800'>{item.jumlah}</td>
            <td className='p-2 border border-slate-800'>Rp{item.hargaSatuan}</td
>
            <td className='p-2 border border-slate-800'>Rp{item.hargaSatuan * it
em.jumlah}</td>
           </tr>
          )
         })
        }
       </tbody>
      </table>
     </div>
     <div className='flex flex-row w-full justify-evenly items-center'>
      <button className='text-xl p-2 bg-blue-600 rounded-full self-center' onCli
ck={() => navigate('/cart')}>Kembali</button>
      <button className='m-2 p-2 self-center bottom-2 bg-blue-500 rounded-3xl' o
nClick={() => saveTransaction()}>
       Simpan Transaksi
      </button>
     </div>
    </div>
   </div>
  )
}

export default TransactionDetailPage
```

React Main App :

- App.jsx (path :./frontend/src/App.jsx)

```
import { useEffect, useState } from 'react'
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'
import HomePage from './components/HomePage'
import CustomerDetailPage from './components/CustomerDetailPage'
import CartPage from './components/CartPage'
import TransactionDetailPage from './components/TransactionDetailPage'
import ListTransactionPage from './components/ListTransactionPage'
import MarketInfoPage from './components/MarketInfoPage'

function App() {
  const [count, setCount] = useState(0)

  return (
    <div className='w-full h-screen bg-slate-300'>
      <Router>
        <Routes>
          <Route path='/' element={<HomePage/>} />
          <Route path='/customer-detail' element={<CustomerDetailPage/>} />
          <Route path='/cart' element={<CartPage/>} />
          <Route path='/transaction-detail' element={<TransactionDetailPage/>}
/>
          <Route path='/transaction-list' element={<ListTransactionPage/>} />
          <Route path='/market-info' element={<MarketInfoPage/>} />
        </Routes>
      </Router>
    </div>
  )
}

export default App
```

- main.jsx (path :./frontend/src/main.jsx)

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import {Provider} from 'react-redux'
import {store} from './redux/store.js'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <Provider store={store} >
      <App />
    </Provider>
  </React.StrictMode>,
)
```

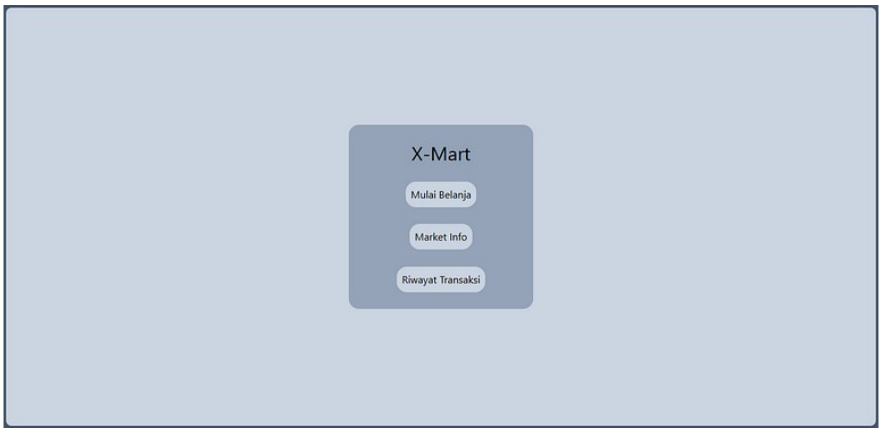
- index.css (path :./frontend/src/index.css)

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Demo Aplikasi

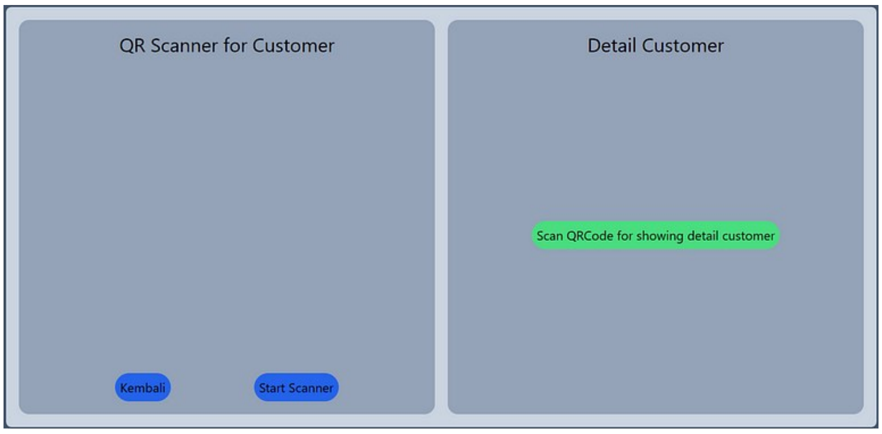
Setelah kedua project backend dan satu project frontend telah dibuat, saatnya mengecek integrasi aplikasi dengan mendemokan keseluruhan fitur yang dibuat. Fitur yang akan didemokan yang pertama adalah scan QR Code customer. Fitur ini akan menscan QR Code customer dan mengeceknya di database postgreSQL apakah QR Code tersebut terdaftar atau tidak. Kemudian yang kedua adalah fitur untuk menambahkan barang ke keranjang dan menambahkan kuantitasnya. Lalu setelah scan QR Code customer dan memilih barang ke keranjang, langkah selajutnya menyimpan data pesanan tersebut menjadi data transaksi. Langkah terakhir adalah melihat data transaksi yang telah dibuat sebelumnya pada tabel riwayat transaksi dari MongoDB dan dari PostgreSQL. Berikut adalah detail pengujiannya.

Home Page :

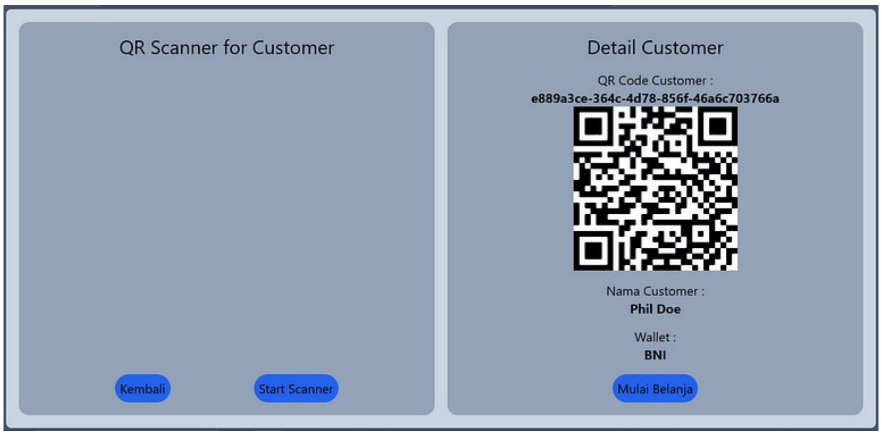


Customer Detail Page:

- Tampilan awal



- Tampilan QR Code terdaftar

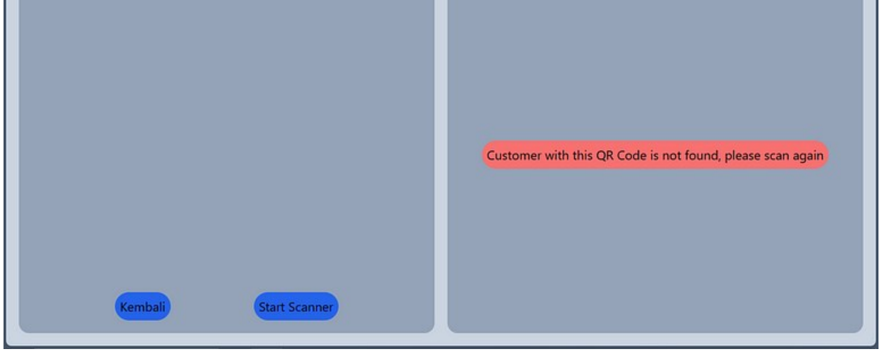


QR Code Customer :
e889a3ce-364c-4d78-856f-46a6c703766a

Nama Customer :
Phil Doe

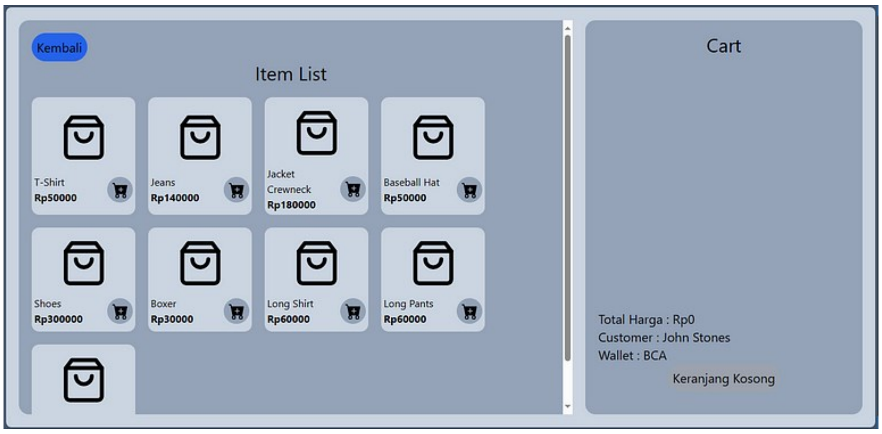
Wallet :
BNI

- Tampilan QR Code tidak terdaftar

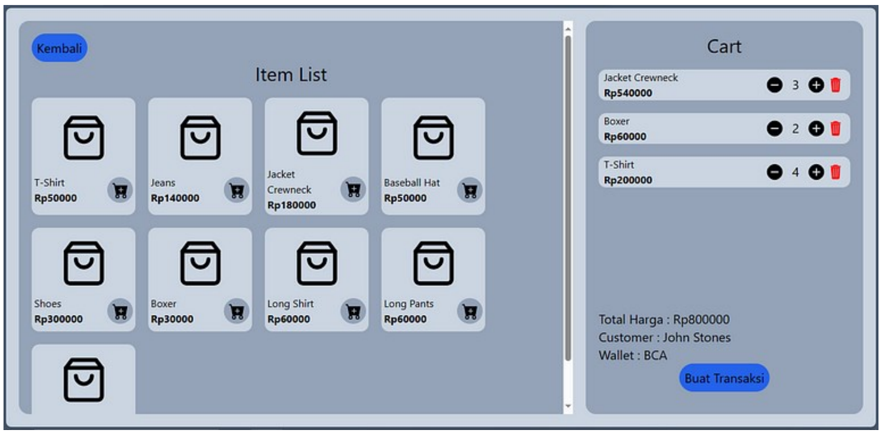


Cart Page :

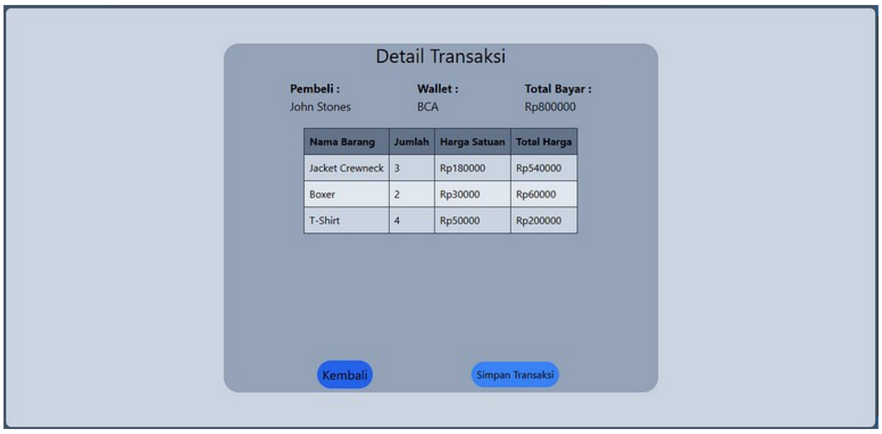
- Tampilan Awal



- Tampilan ketika beberapa barang ditambahkan



Detail Transaction Page :

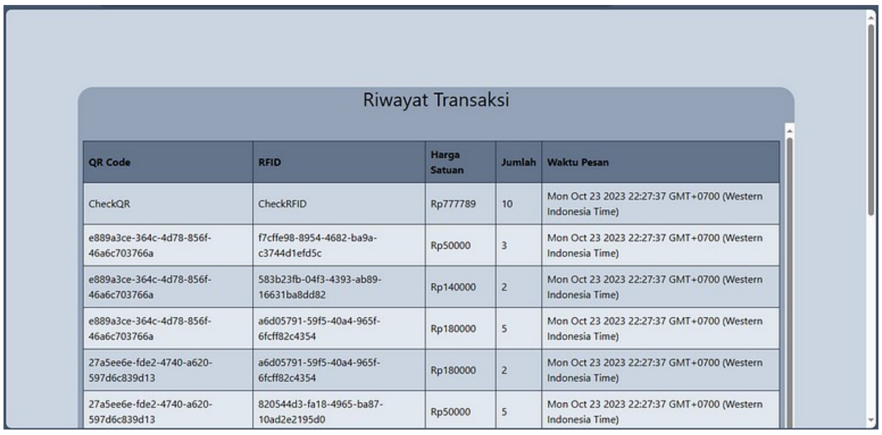


Detail Transaksi

Pembeli : John Stones **Wallet :** BCA **Total Bayar :** Rp800000

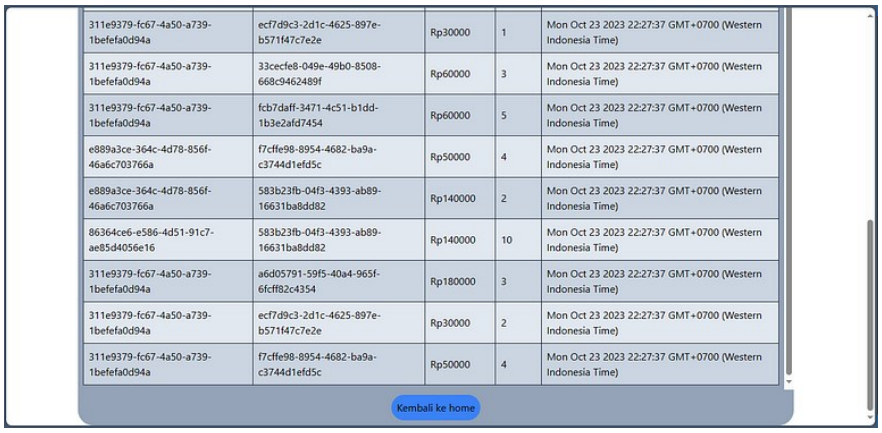
Nama Barang	Jumlah	Harga Satuan	Total Harga
Jacket Crewneck	3	Rp180000	Rp540000
Boxer	2	Rp30000	Rp60000
T-Shirt	4	Rp50000	Rp200000

History Transaction Page :



Riwayat Transaksi

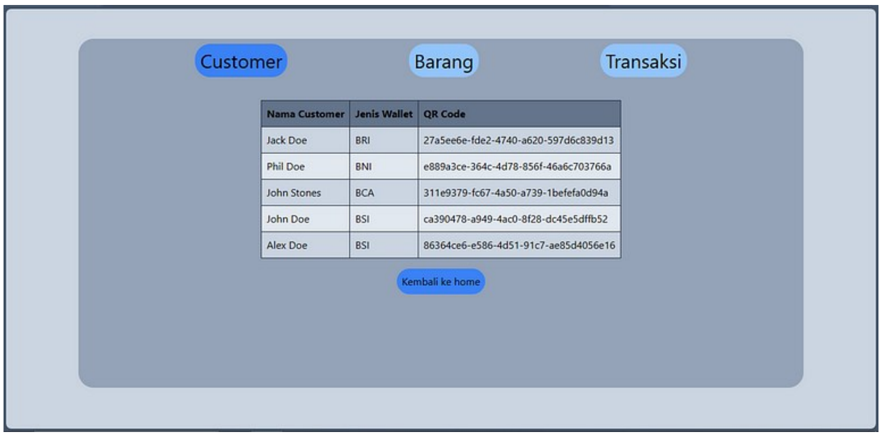
QR Code	RFID	Harga Satuan	Jumlah	Waktu Pesan
CheckQR	CheckRFID	Rp777789	10	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
e889a3ce-364c-4d78-856f-46a6c703766a	f7cffe98-8954-4682-ba9a-c3744d1efd5c	Rp50000	3	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
e889a3ce-364c-4d78-856f-46a6c703766a	583b23fb-04f3-4393-ab89-16631ba8dd82	Rp140000	2	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
e889a3ce-364c-4d78-856f-46a6c703766a	a6d05791-59f5-40a4-965f-6fcff82c4354	Rp180000	5	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
27a5ee6e-fde2-4740-a620-597d6c839d13	a6d05791-59f5-40a4-965f-6fcff82c4354	Rp180000	2	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
27a5ee6e-fde2-4740-a620-597d6c839d13	820544d3-fa18-4965-ba87-10ad2e2195d0	Rp50000	5	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)



311e9379-fc67-4a50-a739-1befefa0d94a	ecf7d9c3-2d1c-4625-897e-b571f47c7e2e	Rp30000	1	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
311e9379-fc67-4a50-a739-1befefa0d94a	33cecfe8-049e-49b0-8508-668c9462489f	Rp60000	3	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
311e9379-fc67-4a50-a739-1befefa0d94a	fcb7daff-3471-4c51-b1dd-1b3e2afd7454	Rp60000	5	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
e889a3ce-364c-4d78-856f-46a6c703766a	f7cffe98-8954-4682-ba9a-c3744d1efd5c	Rp50000	4	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
e889a3ce-364c-4d78-856f-46a6c703766a	583b23fb-04f3-4393-ab89-16631ba8dd82	Rp140000	2	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
86364ce6-e586-4d51-91c7-ae85d4056e16	583b23fb-04f3-4393-ab89-16631ba8dd82	Rp140000	10	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
311e9379-fc67-4a50-a739-1befefa0d94a	a6d05791-59f5-40a4-965f-6fcff82c4354	Rp180000	3	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
311e9379-fc67-4a50-a739-1befefa0d94a	ecf7d9c3-2d1c-4625-897e-b571f47c7e2e	Rp30000	2	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)
311e9379-fc67-4a50-a739-1befefa0d94a	f7cffe98-8954-4682-ba9a-c3744d1efd5c	Rp50000	4	Mon Oct 23 2023 22:27:37 GMT+0700 (Western Indonesia Time)

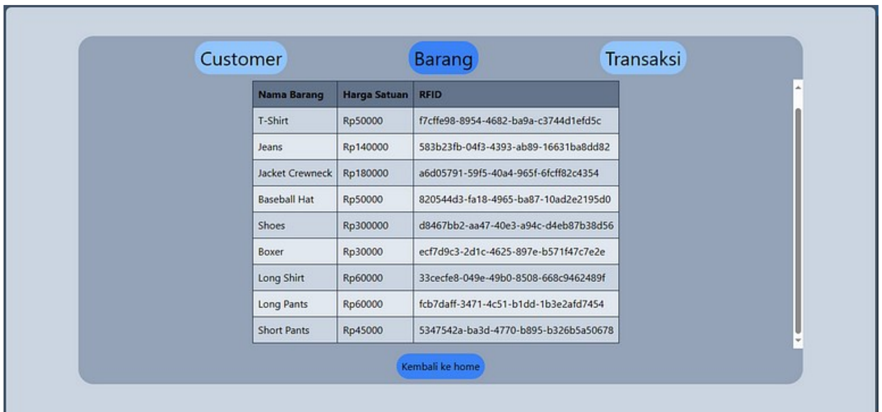
Market Info Page :

- Customers Table



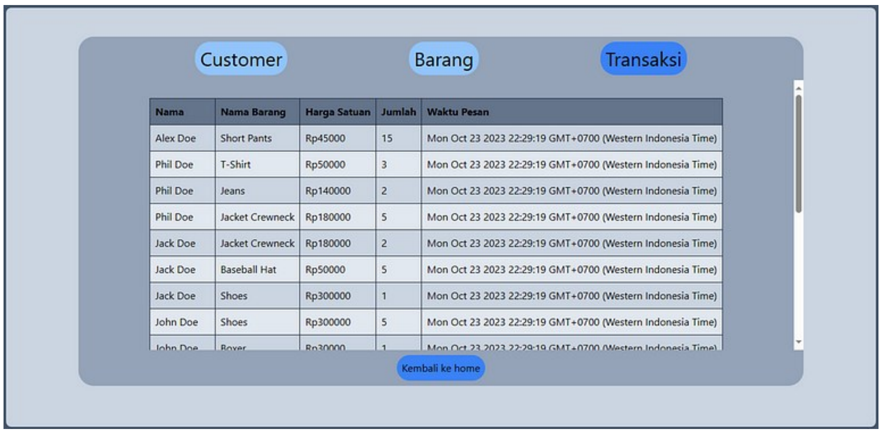
Nama Customer	Jenis Wallet	QR Code
Jack Doe	BRI	27a5ee6e-fde2-4740-a620-597d6c839d13
Phil Doe	BNI	e889a3ce-364c-4d78-856f-46a6c703766a
John Stones	BCA	311e9379-fc67-4a50-a739-1befefa0d94a
John Doe	BSI	ca390478-a949-4ac0-8f28-dc45e5dffb52
Alex Doe	BSI	86364ce6-e586-4d51-91c7-ae85d4056e16

- Items Table



Nama Barang	Harga Satuan	RFID
T-Shirt	Rp50000	f7cffe98-8954-4682-ba9a-c3744d1efd5c
Jeans	Rp140000	583b23fb-04f3-4393-ab89-16631ba8dd82
Jacket Crewneck	Rp180000	a6d05791-59f5-40a4-965f-6fcff82c4354
Baseball Hat	Rp50000	820544d3-fa18-4965-ba87-10ad2e2195d0
Shoes	Rp300000	d8467bb2-aa47-40e3-a94c-d4eb87b38d56
Boxer	Rp30000	ecf7d9c3-2d1c-4625-897e-b571f47c7e2e
Long Shirt	Rp60000	33cecfe8-049e-49b0-8508-668c9462489f
Long Pants	Rp60000	fcb7daff-3471-4c51-b1dd-1b3e2afd7454
Short Pants	Rp45000	5347542a-ba3d-4770-b895-b326b5a50678

- Transactions Table



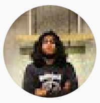
Nama	Nama Barang	Harga Satuan	Jumlah	Waktu Pesan
Alex Doe	Short Pants	Rp45000	15	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
Phil Doe	T-Shirt	Rp50000	3	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
Phil Doe	Jeans	Rp140000	2	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
Phil Doe	Jacket Crewneck	Rp180000	5	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
Jack Doe	Jacket Crewneck	Rp180000	2	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
Jack Doe	Baseball Hat	Rp50000	5	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
Jack Doe	Shoes	Rp300000	1	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
John Doe	Shoes	Rp300000	5	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)
John Doe	Boxer	Rp30000	1	Mon Oct 23 2023 22:29:19 GMT+0700 (Western Indonesia Time)

Penutup

Demikian penjelasan mengenai project fullstack mini project X-Mart menggunakan React JS, Express JS, dan Java Spring Boot serta teknologi Caching Redis, GraphQL dan Redux. Mohon maaf apabila ada kesalahan dalam postingan artikel ini. Saya harap pembaca mendapatkan ilmu baru setelah membaca artikel ini dan dapat menerapkan teknologi teknologi yang diterapkan pada project ini kedalam project yang dirancang pembaca sendiri seperti GraphQL, Caching Redis, Axios, Redux, sampai store data ke PostgreSQL atau MongoDB melalui service backend Express JS ataupun Java Spring Boot.

Full Stack GraphQL Redis Reactjs Nodejs

53

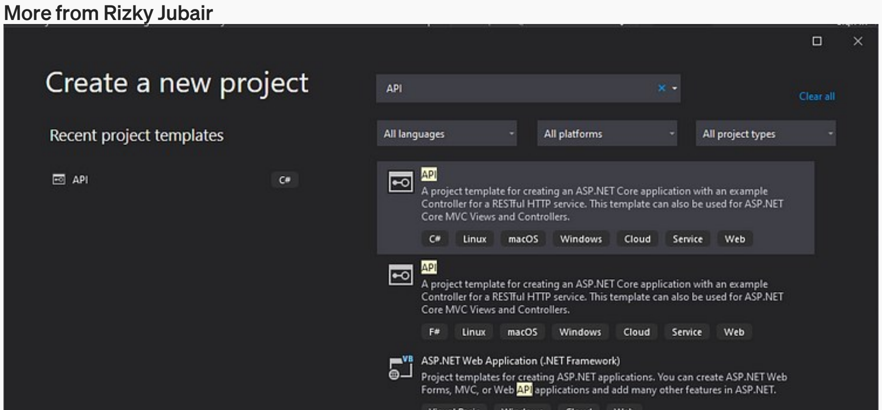


Written by Rizky Jubair

0 Followers

Follow

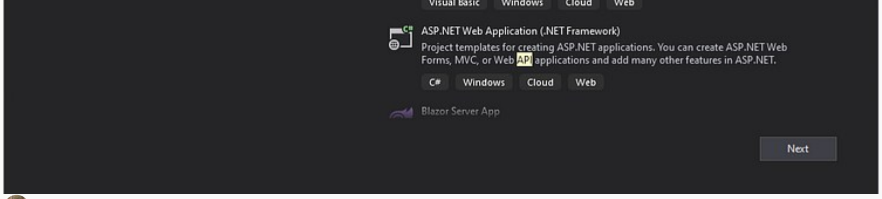
More from Rizky Jubair



API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
C# Linux macOS Windows Cloud Service Web

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
F# Linux macOS Windows Cloud Service Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

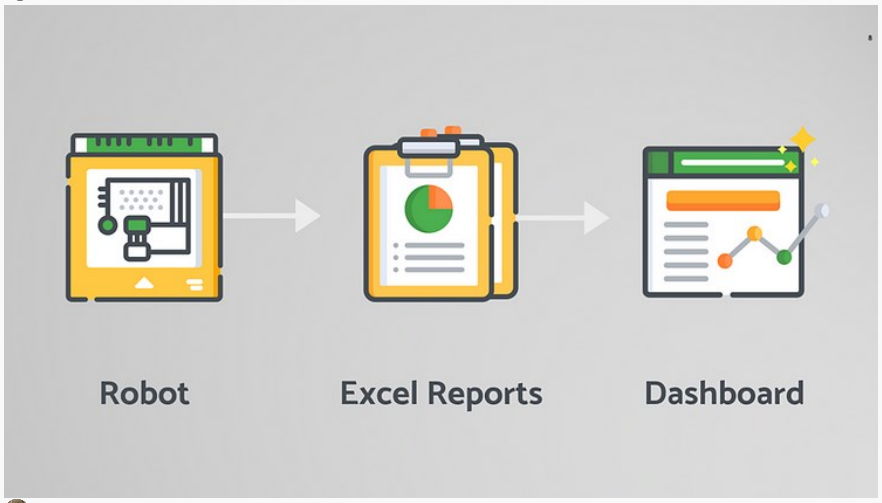

Visual Basic
Windows
Cloud
Web
ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C#
Windows
Cloud
Web
Blazor Server App
Next

Rizky Jubair

Penerapan CRUD REST API sederhana menggunakan .NET Core & SQL Server

Assalamualaikum Wr. Wb.

12 min read · Nov 2, 2023



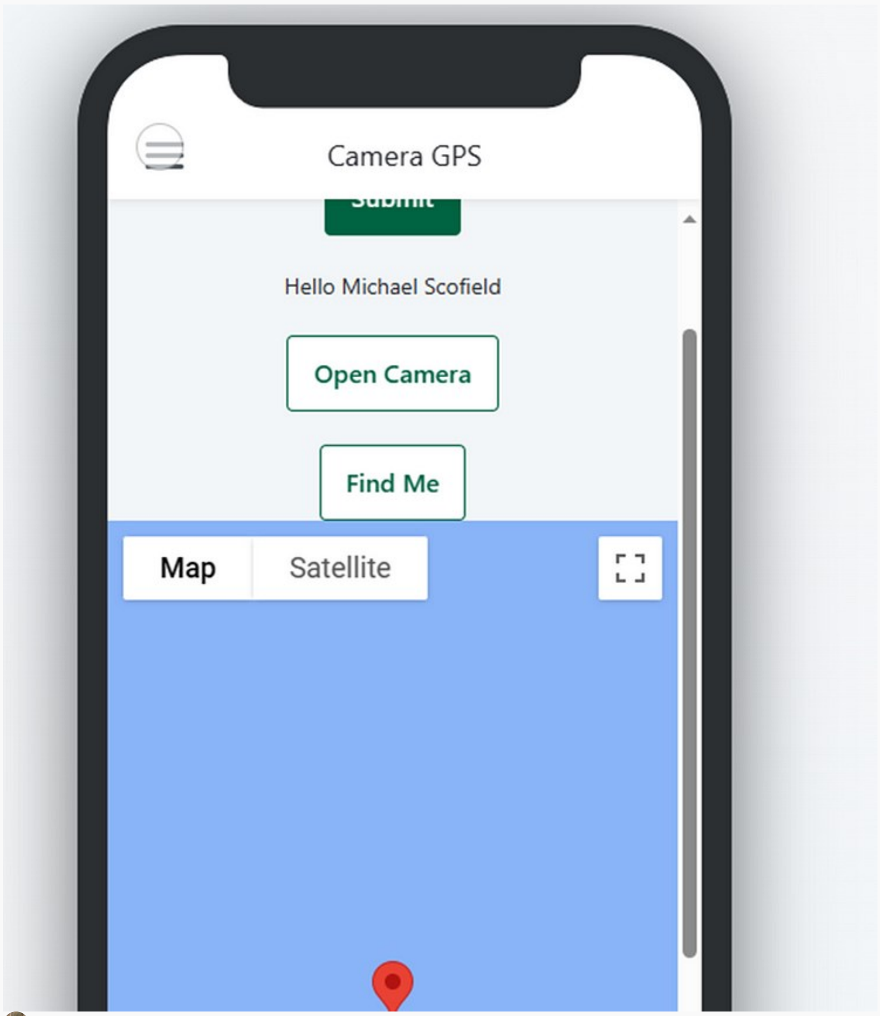
Rizky Jubair

Robot Proccess Automation Development Tool menggunakan UIPath

Assalamualaikum Wr. Wb.

4 min read · Sep 14, 2023

51

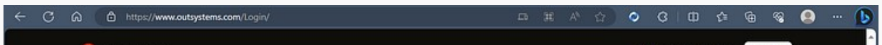


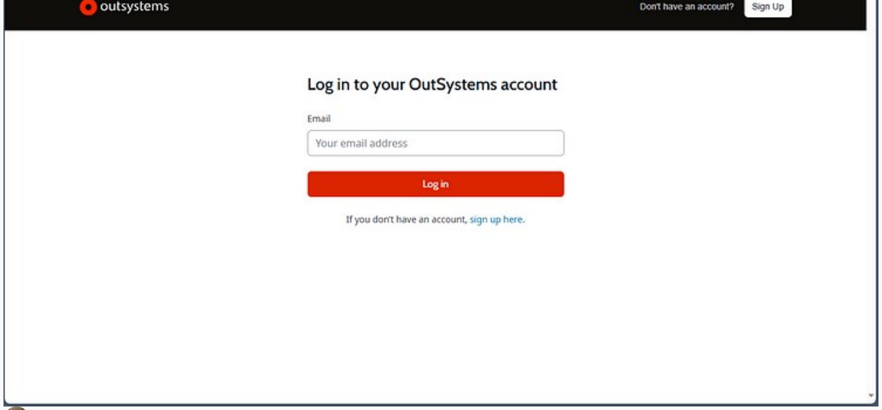
Rizky Jubair

Penerapan Kamera dan GPS pada Aplikasi Mobile menggunakan Outsystems

Assalamualaikum Wr. Wb.

7 min read · Sep 11, 2023




outsystems

Don't have an account? Sign Up

Log in to your OutSystems account

Email

Your email address

Log in

If you don't have an account, sign up here.

Rizky Jubair

Aplikasi CRUD Sederhana menggunakan Outsystems dan REST API GoREST

Assalamualaikum Wr. Wb.

9 min read · Sep 5, 2023

200

See all from Rizky Jubair

Recommended from Medium

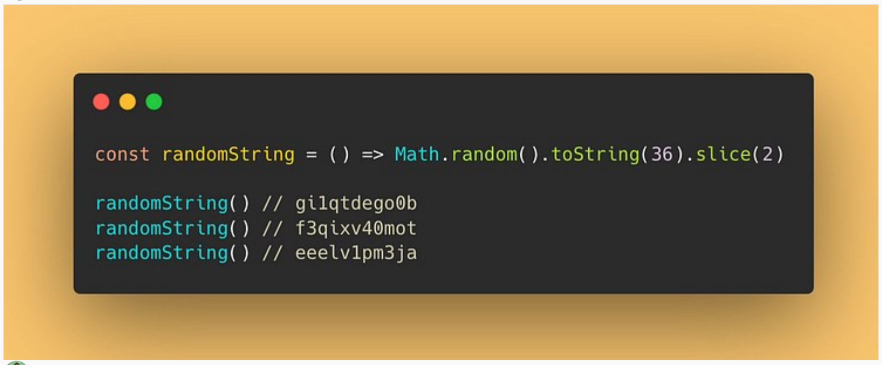
Whapi.Cloud API

How to Create a Whatsapp Bot With Node.js

This guide for programmers is a step-by-step instruction on setting up and launching your own chatbot using the Whapi.Cloud API on Node.JS...

13 min read · Nov 3, 2023

143 1



```
const randomString = () => Math.random().toString(36).slice(2)

randomString() // gi1qtdego0b
randomString() // f3qixv40mot
randomString() // eeelv1pm3ja
```

fatfishinJavaScript in Plain English

18 JavaScript One-Liners That'll Make You Look Like a Pro

A list of one-liners you should know to up your knowledge of JavaScript.

· 5 min read · 6 days ago

212 4

Lists

Stories to Help You Grow as a Software Developer

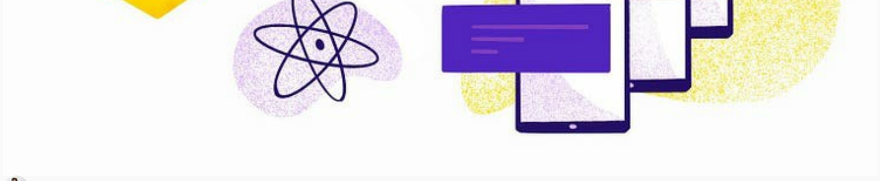
19 stories · 981 saves

General Coding Knowledge

20 stories · 1119 saves

data science and AI

40 stories · 130 saves



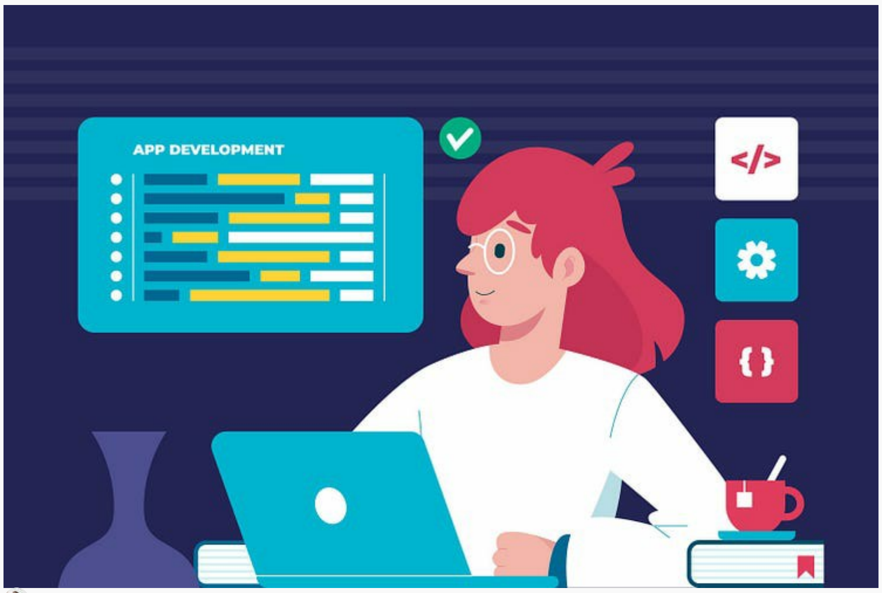
Abbas Ali Radiowala

React Native Push Notifications: Your Ultimate Guide

Hi Geeks !! In this fast-paced world of mobile app development, Push Notifications play a pivotal role in engaging users and delivering...

9 min read · Feb 13, 2024

228 1



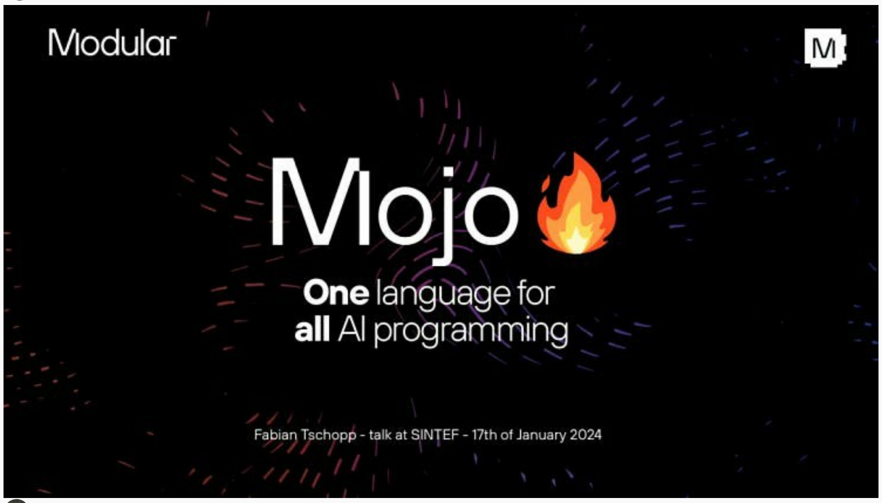
Héla Ben KhalfallahinITNEXT

Frontend Development Beyond React: Svelte (1/3)

Delving into Svelte, Solid, and Qwik

· 20 min read · Apr 11, 2024

2.2K 6


Modular
M
Mojo
One language for all AI programming
Fabian Tschopp - talk at SINTEF - 17th of January 2024

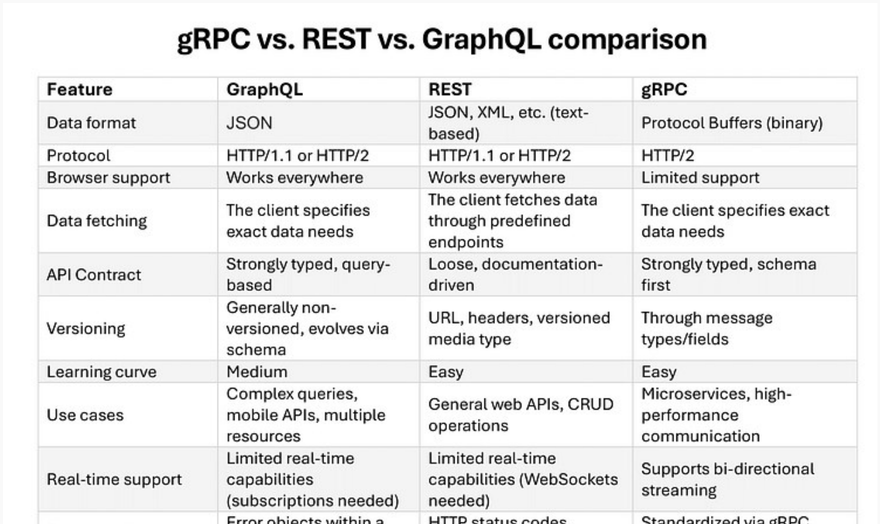
Dylan CooperinStackademic

Mojo, 90,000 Times Faster Than Python, Finally Open Sourced!

On March 29, 2024, Modular Inc. announced the open sourcing of the core components of Mojo.

· 10 min read · Apr 9, 2024

2.7K 21



gRPC vs. REST vs. GraphQL comparison

Feature	GraphQL	REST	gRPC
Data format	JSON	JSON, XML, etc. (text-based)	Protocol Buffers (binary)
Protocol	HTTP/1.1 or HTTP/2	HTTP/1.1 or HTTP/2	HTTP/2
Browser support	Works everywhere	Works everywhere	Limited support
Data fetching	The client specifies exact data needs	The client fetches data through predefined endpoints	The client specifies exact data needs
API Contract	Strongly typed, query-based	Loose, documentation-driven	Strongly typed, schema first
Versioning	Generally non-versioned, evolves via schema	URL, headers, versioned media type	Through message types/fields
Learning curve	Medium	Easy	Easy
Use cases	Complex queries, mobile APIs, multiple resources	General web APIs, CRUD operations	Microservices, high-performance communication
Real-time support	Limited real-time capabilities (subscriptions needed)	Limited real-time capabilities (WebSockets needed)	Supports bi-directional streaming
	Error objects within a	HTTP status codes	Standardized via gRPC

Error Handling	Error objects within a successful response	HTTP status codes, custom payloads	Standardized via gRPC status codes
Community	Growing	Large	Large

Dr Milan Milanović

When to use GraphQL, gRPC, and REST?

Building APIs is one of the most important tasks for developers in modern engineering. These APIs allow different systems to communicate...

11 min read · Apr 8, 2024

286 2

See more recommendations