PROJECT Documentation

Tsunami Marketplace - Web Application

Universitas Tarumanagara

Web Programming (C) – Kelompok 9

**DAFTAR ISI**

[**SECTION 1 About This Project** 1](#_Toc58871077)

[**1.1.** **Project Plan – 5W 1H** 1](#_Toc58871078)

[**1.2** **Software Quality - ISO 9126** 1](#_Toc58871079)

[**1.3** **Project Structure** 3](#_Toc58871080)

[**1.4** **Cara Menjalankan Project** 5](#_Toc58871081)

[**SECTION 2 Frontend Tsunami** 6](#_Toc58871082)

[**2.1 Pendahuluan Frontend** 6](#_Toc58871083)

[**2.2 Landing Page** 6](#_Toc58871084)

[**2.3 Homepage** 7](#_Toc58871085)

[**2.4 Cart – Checkout** 7](#_Toc58871086)

[**2.5 Catalog** 10](#_Toc58871087)

[**2.6 Product Search** 11](#_Toc58871088)

[**2.7 Account** 12](#_Toc58871089)

[**SECTION 3 Backend Tsunami** 14](#_Toc58871090)

[**3.1 Pendahuluan Backend** 14](#_Toc58871091)

[**3.2 Account** 14](#_Toc58871092)

[**3.3 Address** 15](#_Toc58871093)

[**3.4 Post** 16](#_Toc58871094)

[**3.5 Product** 17](#_Toc58871095)

[**3.6 Transaction** 19](#_Toc58871096)

# **SECTION 1 About This Project**

## **Project Plan – 5W 1H**

Pada bagian project planning ini, kami menggunakan pendekatan 5W 1H untuk merancang langkah-langkah dan tujuan yang ingin kami capai dalam pengerjaan project ini. Berikut project plan yang telah kami rancang:

Who

Orang-orang yang ingin mencari tempat melaksanakan kegiatan jual beli dengan mudah dan User Experience (UX) yang berbeda

What

Sebuah inovasi dari marketplace yang akan menutup kekurangan-kekurangan marketplace yang ada di Indonesia.

When

Project ini akan diselesaikan paling lambat bulan Desember 2020.

Where

Kami menggunakan Visual Studio Code dengan integrasi GitHub untuk membuat website ini.

Why

Project ini kami buat untuk menunjukan konsep marketplace yang unik dan agar para pengguna dapat menikmati User Experience (UX) yang berbeda dibandingkan marketplace lainnya

How

Pengerjaan project akan dilakukan sepenuhnya di local machine setiap anggota kelompok, lalu akan digabungkan menjadi 1 di repository GitHub utama.

## **Software Quality - ISO 9126**

Selain melakukan project planning, kami juga membuat ISO 9126 tentang kualitas web application yang kami buat

1. Functionality

Menyediakan platform dimana seorang penjual dan pembeli dapat bertemu dengan mudahnya karena website dapat diakses di perangkat desktop ataupun mobile.

2. Reliability

Memiliki fitur untuk bug report bila terjadi kesalahan/eror dalam website/aplikasi, dan sering melakukan patch minimal 1 minggu sekali untuk memperbaiki bug tersebut. Kami juga akan menerima kritik dan saran dari para pengguna website sebagai bahan referensi untuk pengembangan website.

3. Usablity

Mudah untuk digunakan dan dipahami oleh orang awam sekalipun karena memiliki tampilan yang sederhana namun unik serta menyerupai standard penempatan button marketplace pada umumnya. Selain itu terdapat juga halaman bantuan/FAQ untuk menjawab pertanyaan-pertanyaan pada pengguna.

4. Efficiency

Walaupun website ini memiliki tampilan yang unik serta UX yang berbeda, website kami tetaplah ringan karena kode/algoritma yang kami gunakan merupakan algoritma yang efisien secara resource sehingga web kami tidak memakan banyak penyimpanan/memori dan cepat untuk di-load.

5. Maintainability

Tersedia fitur untuk mendiagnosis/melihat kekurangan/eror yang terjadi dari para pengguna dan melaporkannya untuk diperbaiki, juga ada fitur untuk modifikasi kepada toko/profil pengguna

6. Portability

Bisa digunakan di device mana saja (PC, Smartphone, dll.) dan beradaptasi sesuai dengan platform device yang digunakan

## **Project Structure**

Tech Stack yang kami pakai dalam project ini adalah stack MEVN yang terdiri atas (MongoDB, Express.js, Vue.js, dan Node.js) dimana MongoDB merupakan database NoSQL yang akan kami pakai, Express.js dan Node.js sebagai backend serta Vue.js menjadi framework frontendnya.

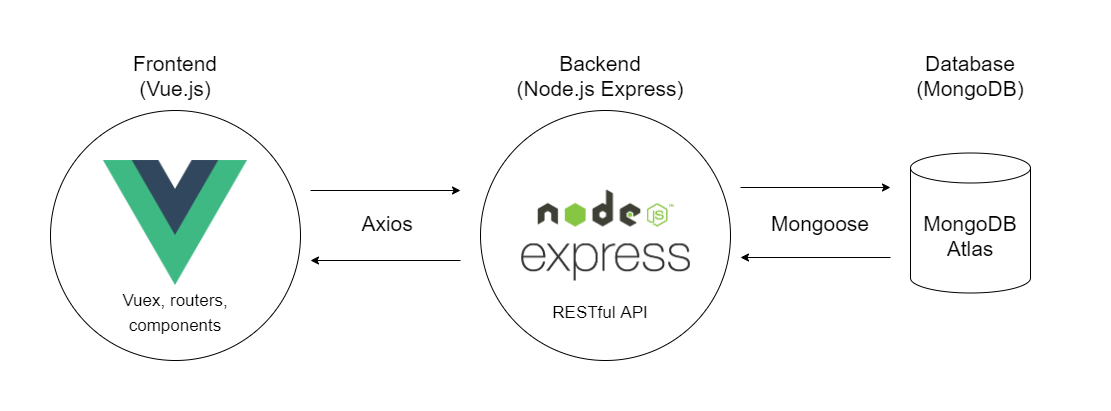


Figure 1. Tech Stack Tsunami Marketplace

Struktur file project kami mengikuti konsep server-client dimana terdapat folder khusus untuk file-file backend serta folder khusus untuk file-file frontend.

Figure 2. Struktur File Tsunami

Sehubungan dengan Figure 2 di atas, berikut fungsi setiap folder yang ada dalam project ini:

1. Root/Server  
   Ini merupakan folder pertama yang berada dalam project ini, fungsinya adalah sebagai tempat file-file backend seperti starting point server (server.js), config-config server, routes REST API dan model-model database.
2. Routes - api

Folder api dalam routes merupakan tempat diletakannya routes-routes server web, contohnya seperti routes product (/api/product) untuk mengambil data-data yang berkaitan dengan product.

1. Models

Folder models berfungsi untuk menyimpan model-model database yang dipakai di project, dalam kasus ini merupakan skema-skema MongoDB yang digunakan untuk mewakili suatu data.

1. Client

Folder client merupakan folder tempat dimana file-file frontend berada. Sebgaian besar folder disini merupakan hasil generate dari Vue CLI.

1. Src (Source)

Folder src merupakan tempat utama source code milik client, dalam kasus ini merupakan file-file .vue yang dirancang.

1. Assets

Folder assets digunakan untuk menyimpan gambar-gambar static yang digunakan dalam web

1. Components.

Folder ini digunakan untuk meletakan komponen-komponen (SFC) dari web berbasis Vue yang telah dibuat dan akan digunakan oleh view-view yang ada.

1. Mixins

Digunakan untuk meletakan mixins vue yaitu fungsi yang dikhususkan untuk reusability

1. Router

Digunakan untuk meletakan index.js yang berisi route-route dari Single Page Application yang terbentuk, contohnya untuk masuk dashboard routenya adalah /account

1. Store

Digunakan untuk meletakan file-file vuex store untuk local data

1. Views

Digunakan untuk meletakan view-view Vue.

## **Cara Menjalankan Project**

Clone project dari repo terbaru yang berada di GitHub, pastikan Git CLI telah terinstall, jika tidak, command “git” tidak akan berjalan:

Setelah proses clone selesai, buka cmd di directory hasil cloning lalu install package yang diperlukan dengan Node Package Manager (NPM):



Script di atas akan menginstall modul-modul server dan client yang dibutuhkan. Setelah proses penginstallan selesai silahkan jalankan server dan client dengan:



Jika command di atas berhasil dijalankan maka akan dilakukan kompilasi web dan server akan dijalankan. Terdapat feedback pertanda server/client berhasil dijalankan yaitu akan ada tulisan URL server/client berjalan pada console.

Selain dengan cara menjalankan pada server lokal di PC, web ini juga sudah dideploy ke Heroku yang dapat diakses seperti web pada umumnya, URL Heroku Tsunami Marketplace adalah <http://tsunami-marketplace.herokuapp.com/>

# **SECTION 2 Frontend Tsunami**

## **2.1 Pendahuluan Frontend**

Frontend pada Tsunami menggunakan Framework Vue.js sehingga susunan-susunan setiap halaman yang ada merupakan kombinasi dari berbagai komponen yang dapat saling berinteraksi melalui view utama (root view). Sehingga secara frontend, web kami telah memanfaatkan Single File Component (SFC) dan telah merupakan sebuah web Single Page Application (SPA) yang dimana komponen-komponen akan melakukan render ulang seperlunya dengan memanfaatkan 1 file root (index.html) yang ada dan setiap komponen akan mengurus state mereka sendiri.

Pada subsection-subsection berikutnya, akan dijabarkan penyusunan komponen-komponen setiap halaman yang ada di web

## **2.2 Landing Page**

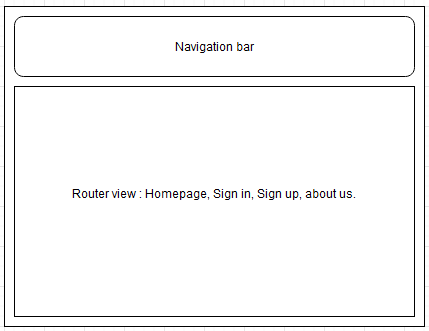


Figure 3. Skema Landing Page

Pada halaman Landing terdapat komponen Navigation bar. Selanjutnya terdapat router view yang akan menampilkan halaman lainnya sesuai dengan apa yang diinginkan oleh user.

## **2.3 Homepage**

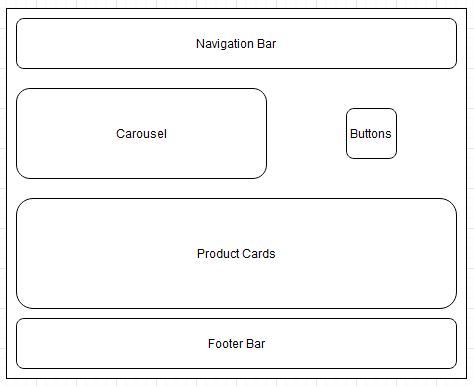


Figure 4. Skema Homepage

Pada halaman homepage, terdapat beberapa komponen seperti Navigation Bar, buttons yang digunakan untuk quick access halaman-halaman di web, kemudian terdapat bagian untuk menampilkan produk terbaru, produk yang sedang ada promosi seperti flash sale ataupun daily deals. Selain itu di paling bawah terdapat juga footer.

## **2.4 Cart – Checkout**

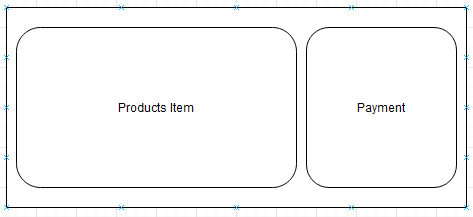


Figure 5. Skema Cart

Bagian halaman cart terdiri atas 3 komponen, yaitu bagian product items yang di sebelah kiri, bagian payment yang menyangkut subtotal, input voucher dan total harga yang dibeli, serta cart item yang mewakili setiap produk yang sedang berada di cart.

Komponen Cart Item memiliki bentuk seperti sebagai berikut:

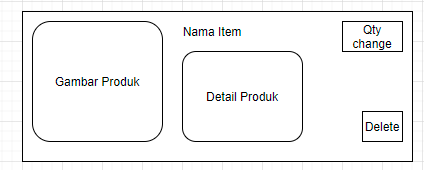


Figure 6. Skema Cart Item

Jadi di dalam komponen list cart (kiri) akan menampung Cart Item untuk setiap item yang berada di dalam keranjang.

Lalu setelah halaman cart, terdapat halaman untuk melakukan checkout dengan skema seperti di bawah ini

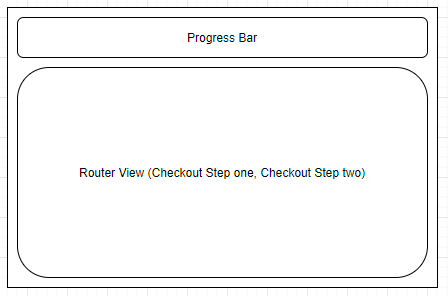


Figure 7. Skema Checkout

Pada bagian checkout, terdiri atas 2 bagian, yaitu bagian progress bar yang akan menandakan progress checkout saat ini (Step satu atau step dua) kemudian terdapat Router View yang akan digunakan untuk menampilkan component yang bertugas untuk melayani checkout tahap satu atau dua.

Detail dari komponen checkout tahap satu dan dua adalah sebagai berikut

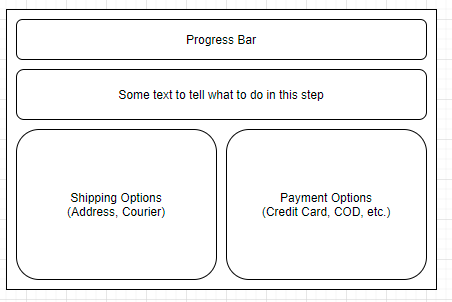


Figure 8. Skema Checkout Step 1

Pada checkout tahap 1, terdapat 2 komponen utama yaitu Shipping Options dan Payment Options yang berfungsi untuk memilih alamat pengiriman, kurir pilihan, serta untuk pemilih opsi pembayaran

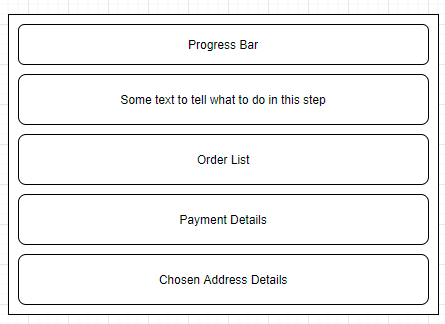


Figure 9. Skema Checkout Step 2

Sedangkan pada checkout tahap 2, desainnya akan berubah menjadi sistem list, yaitu seperti bertumpukan (stack). Pada bagian ini terdapat beberapa bagian penting yaitu order list, payment details, dan detail address yang sudah dipilih.

## **2.5 Catalog**

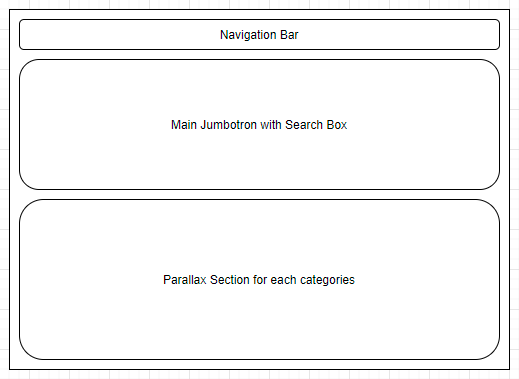


Figure 10. Skema Catalog

Halaman Catalog tersusun atas 3 komponen, yaitu navigation bar yang bersifat floating, lalu search box yang terdapat di main jumbotron, serta parallax section yang digunakan untuk menampilkan kategori tertentu dengan gambar backgroundnya.

Penyebab search box dijadikan komponen sendiri dikarenakan search box pada halaman catalog ini memiliki logikanya sendiri, seperti auto suggestion untuk kategori-kategori yang ada sehingga lebih baik jika dijadikan 1 komponen yang berdiri sendiri.

## **2.6 Product Search**

Figure 11. Skema Product Search

Pada bagian Product Search, desain yang dibuat cukuplah sederhana, hanya terdapat navigation bar yang berisi search box dan list product-product yang berhasil dicari dan diambil dari database.

Untuk menampilkan list produk, CSS Grid System digunakan agar susunannya dapat berupa grid-grid yang mewakili sebuah produk, dan agar design yang tercipta sudah responsive.

## **2.7 Account**

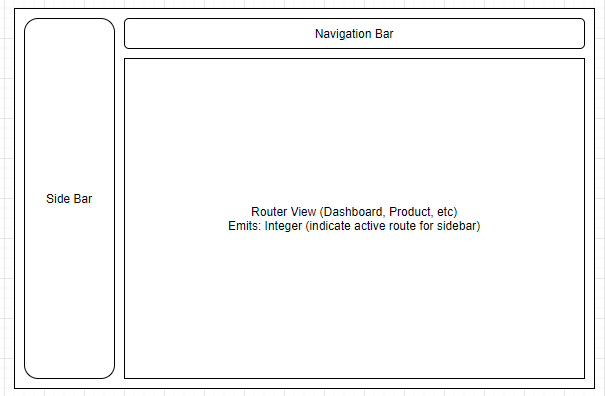


Figure 12. Skema View Account

Pada halaman account (Dashboard, Product, Transaction History, dan Account Setting), komponen-komponen yang dibentuk adalah komponen Side Bar dan komponen Navigation Bar seperti pada Figure 3. Kemudian di bagian router viewnya akan menampilkan halaman-halaman yang berbeda untuk setiap page accountnya seperti dashboard ataupun transaction.

Setiap subroute pada account (Dashboard, Product, Transaction History, dan Account) akan meng-emit sebuah flag yang menandakan bahwa halaman yang bersangkutan sedang aktif, variabel yang diemit adalah berupa sebuah Integer, sebagai contohnya pada saat dashboard dibuka maka component dashboard akan meng-emit angka 1 ke parentnya, yaitu Account View. Kemudian Integer tersebut akan diserahkan ke komponen Side Bar untuk menentukan tombol Side Bar yang mana yang harus menyala sesuai dengan halaman yang sedang terbuka.

Special Casenya terdapat pada bagian Account Setting, karena di dalam Account Setting, terdapat nested component pada bagian tab-tab setting seperti security ataupun address. Ilustrasinya dapat dilihat pada Figure 4.

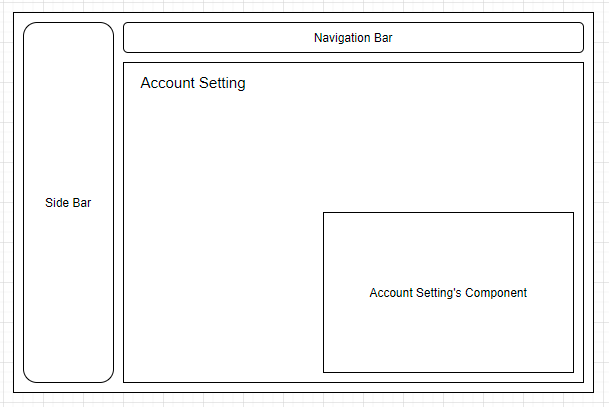


Figure 13. View Account Setting

Sesuai dengan penjelasan singkat di atas, di dalam halaman Account Setting memiliki komponen-komponen yang akan mewakili setiap tab di Account Setting. Sehingga sifat pada halaman Account Setting pada view Account ini memiliki konsep nested components.

# **SECTION 3 Backend Tsunami**

## **3.1 Pendahuluan Backend**

Backend pada Tsunami menggunakan Node.js dan Express.js sebagai servernya dan MongoDB sebagai databasenya. Untuk melakukan koneksi dari frontend (Vue.js) ke backend server, digunakan sebuah modul yaitu axios ([axios - npm (npmjs.com)](https://www.npmjs.com/package/axios)) yang akan mengurus hal-hal yang berhubungan dengan REST API pada web, seperti request GET / POST / ataupun PUT dari frontend ke backend database ataupun sebaliknya.

Database yang digunakan adalah MongoDB berbasis cloud yaitu MongoDB Atlas ([Managed MongoDB Hosting | Database-as-a-Service | MongoDB](https://www.mongodb.com/cloud/atlas)) yang akan menyimpan data-data yang dimiliki web seperti data account, produk, maupun transaksi-transaksi yang terjadi.

Pada subsection-subsection berikutnya akan dijelaskan routes-routes dan model database yang akan menghandle request-request API dari frontend.

## **3.2 Account**

Model Account yang digunakan:

|  |  |  |
| --- | --- | --- |
| Nama | Tipe Variabel | Constraint / Details |
| uid | String | Unique, Required |
| display\_name | String | Required |
| email\_address | String | Required |
| gender | String | Required |
| birthday | String | Required |
| phone\_number | String | Required |
| member | String | - |
| PIN | Number | Default: NULL |
| seller | Object | name: String  location: String  products\_sold: Number |
| cart | Array of Objects | product: ObjectID (ref: product)  price: Number  qty: Number |

Routes-routes pada /api/account

|  |  |  |
| --- | --- | --- |
| Nama | Tipe | Fungsi |
| /register | POST | Untuk membuat account baru dan menyimpannya di database |
| /:uid | GET | Mengambil data dari account yang memiliki UID sesuai pada parameter |
| /:uid | PUT | Melakukan update data dari account yang memiliki UID sesuai pada parameter |
| /:uid/cart | GET | Mengambil cart dari account yang memiliki UID sesuai pada parameter |
| /:uid/cart | POST | Menambahkan cart item baru pada account yang memiliki UID sesuai pada parameter |
| /:uid/cart/:id | DELETE | Melakukan delete pada cart item yang dimiliki account tertentu |

## **3.3 Address**

Model Address yang digunakan:

|  |  |  |
| --- | --- | --- |
| Nama | Tipe Variabel | Constraint / Details |
| owner\_uid | String | Required |
| is\_default | Boolean | - |
| name | String | Required |
| phone\_number | String | Required |
| shipping\_address | String | Required |
| details | Object | province: String  cityregency: String  district: String  subdistrict: String  coordinates: String |

Routes-routes pada /api/address

|  |  |  |
| --- | --- | --- |
| Nama | Tipe | Fungsi |
| /:uid | GET | Untuk mengambil semua address milik account dengan uid tertentu |
| /add | POST | Untuk menambahkan address baru |
| /setdefault/:id | PUT | Untuk menyetel sebuah address dengan id tertentu sebagai default |

## **3.4 Post**

Model Post yang digunakan:

|  |  |  |
| --- | --- | --- |
| Nama | Tipe Variabel | Constraint / Details |
| type | String | Required, berupa discussion, review, atau reply |
| product | ObjectID (ref: product) | Required |
| poster | ObjectID (ref: account) | Required |
| description | String | Required |
| posted\_date | Date | Required |
| review | String | - |
| replies | Array of ObjectID (ref: post) | - |

Routes-routes pada /api/post

|  |  |  |
| --- | --- | --- |
| Nama | Tipe | Fungsi |
| / | POST | Untuk menambahkan post baru |
| /reviews/:product\_id | GET | Untuk mengambil review dari product dengan id tertentu |
| /reviews/seller/:seller\_id | GET | Untuk mengambil review yang dimana penjual produknya adalah account dengan id tertentu |
| /discussions/:id | GET | Mengambil discussion dari product dengan id tertentu |
| /discussions/:id/reply | POST | Untuk menambahkan reply baru dari suatu diskusi dengan id tertentu |

## **3.5 Product**

Model Product yang digunakan:

|  |  |  |
| --- | --- | --- |
| Nama | Tipe Variabel | Constraint / Details |
| seller | ObjectID (ref: account) | Required |
| name | String | Required |
| category | String | Required |
| subcategory | String | Required |
| images | Array | Required |
| description | String | Required |
| detailed\_description | String | Required |
| price | Number | Required |
| stock | Number | Required |
| sold | Number | Required |
| specifications | Array | Required |
| views | Number | Default: 0 |
| posted\_date | Date | Required |
| flashsale | Object | sold: Number,  stock: Number,  price: Number,  start\_time: Date,  end\_time: Date |
| dailydeals | Object | price: Number,  date: Date |

Routes-routes pada /api/product

|  |  |  |
| --- | --- | --- |
| Nama | Tipe | Fungsi |
| / | GET | Untuk ngambil semua product yang tidak memiliki event (flashsale/dailydeals) berjalan |
| /details/:id | GET | Untuk mengambil details-details dari product dengan id tertentu |
| /search/:query | GET | Untuk mengambil product dengan nama yang mengandung kata yang diberikan dalam bentuk parameter “query” |
| /seller/:id | GET | Untuk mengambil product yang dijual oleh account dengan id tertentu |
| /add | POST | Untuk menambahkan produk baru |
| /edit/:id | PUT | Untuk mengupdate produk dengan id tertentu |
| /delete/:id | DELETE | Untuk menghapus produk dengan id tertentu |
| /flashsale/add/:id | POST | Untuk menambahkan event flashsale pada produk dengan id tertentu |
| /flashsale | GET | Untuk mengambil produk yang saat ini sedang memiliki flashsale berjalan |
| /dailydeals/add/:id | POST | Untuk menambahkan event dailydeals pada produk dengan id tertentu |
| /dailydeals | GET | Untuk mengambil produk yang memiliki event dailydeals berjalan |
| /:seller\_id/views | GET | Untuk mengambil jumlah views dari produk yang dimiliki oleh account dengan id tertentu |
| /:id/view | POST | Untuk melakukan increment pada view produk dengan id tertentu |

## **3.6 Transaction**

Model Transaction yang digunakan:

|  |  |  |
| --- | --- | --- |
| Nama | Tipe Variabel | Constraint / Details |
| \_id | Number | Required, Unique |
| buyer\_id | ObjectID (ref: account) | Required |
| date | Date | Required |
| status | String | Required  Default: ‘Processing’ |
| shipping\_address | ObjectID (ref: address) | Required |
| discount | Number | Required |
| total | Number | Required |
| cart | Array of Objects | product: ObjectID (ref: product)  price: Number  qty: Number  review: Boolean, default False |

Routes-routes pada /api/transaction

|  |  |  |
| --- | --- | --- |
| Nama | Tipe | Fungsi |
| / | POST | Untuk menambahkan transaksi baru ke database sekaligus mengurangi stok produk |
| /:id | GET | Untuk ngambil semua transaksi yang dimiliki suatu account dengan id tertentu |
| /:seller\_id/sales | GET | Untuk mengambil transaksi income yang dimiliki account dengan id tertentu |
| /:id/review | PUT | Untuk mengupdate flag review dari suatu produk dalam suatu transaksi (menandakan suatu produk telah direview) |