DOKUMEN PEMBANGUNGAN LAYANAN *MANAGING ORDER* API

Oleh

Mira Risty Masyita

18216037

SISTEM TEKNOLOGI INFORMASI

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2018

DAFTAR ISI

DAFTAR ISI 2
DAFTAR TABEL 3
DAFTAR GAMBAR 4
1. DEFINISI OBJEKTIF BISNIS 5
1.1. MASALAH 5
1.2. DAMPAK/MANFAAT UNTUK PENYEDIA API, PELANGGAN, DAN DEVELOPER PIHAK KETIGA 5
2. MELAKUKAN PENELITIAN 6
2.1. API YANG TERSEDIA 6
2.2. EVALUASI PERBEDAAN 7
3.* DENIFISI *USE CASE 7
4. MEMILIH TKNOLOGI 7
4.1. TEKNOLOGI REST 7
4.2. PRO DAN KONTRA 8
5. SPESIFIKASI TEKNIS API 9
5.1. Judul 9
5.2. Masalah 9
5.3. Solusi 9
5.4. Implementasi 9
5.5. Autektikasi 10
5.6. Rincian Spesifikasi API 11
DAFTAR PUSTAKA 16

DAFTAR TABEL

Tabel 1 Evaluasi Perbedaan 7
Tabel 2 Pro dan Kontra 8
Tabel 3 Spesifikasi Rinci API 12

DAFTAR GAMBAR

Gambar 1 Autentikasi 10
Gambar 2 JSON Web Token..... 11

1. DEFINISI OBJEKTIF BISNIS

1.1. MASALAH

Di Indonesia telah banyak *e-commerce* yang berkembang. Banyak para penjual atau yang memiliki usaha yang menjual barang-barangnya di *platform-platform e-commerce* yang ada. Penjual ingin mengetahui *order* apa saja yang dilakukan oleh *customer* terkait dengan produk-produk yang dijualnya melalui aplikasi yang dibangun oleh *developer*. *Developer* harus mampu memberikan layanan tersebut. *Order* atau pesanan harus dapat terintegrasi dengan data produk-produk yang dijual oleh penjual. Seorang pelanggan tidak dapat membeli barang yang tidak dijual oleh penjual. Pengelolaan data *order* dan *product* ini menjadi hal yang cukup rumit bagi *developer*. Oleh karena itu, diperlukan sebuah API yang dapat mengelola *order* dan *product* yang dijual oleh penjual.

1.2. DAMPAK/MANFAAT UNTUK PENYEDIA API, PELANGGAN, DAN DEVELOPER PIHAK KETIGA

Berdasarkan masalah yang ada, saya sebagai penyedia API, bertujuan membentuk sebuah API yang menyediakan data pengelolaan *order* dan *product* untuk *customer*. *Customer* yang dimaksud adalah *developer* pihak ketiga yang akan menggunakan data tersebut. Dengan adanya API ini, memberikan manfaat atau dampak yang dirasakan oleh penyedia. Penyedia memiliki jumlah *customer* yang cukup banyak menimbang semakin maraknya *e-commerce* yang memungkinkan semakin meningkat pula *developer*-nya.

Selain bermanfaat bagi penyedia API, dengan adanya API ini pula memberikan manfaat kepada *developer* pihak ketiga. *Developer* sangat dipermudah dengan adanya data ini sehingga dalam pengembangan *software* yang akan dibangunnya nanti.

Dan bagi *customer,* dalam kasus ini adalah penjual, dengan adanya *software* yang dibangun oleh *developer* terkait dengan pengelolaan *order* dan *product* yang dijual, sangat memberikan manfaat. *Customer* dapat dengan mudah memonitor produk-produk yang dijualnya melalui *software* yang dibangun oleh *developer* yang menggunakan API yang disediakan oleh penyedia terkait data *order* dan *product* tersebut.

2. MELAKUKAN PENELITIAN

2.1. API YANG TERSEDIA

Moltin adalah salah satu API yang menyediakan pengelolaan data-data mengenai *e-commerce*. Melalui visinya "Provides a simple solution to a complex problem", Moltin percaya bahwa *brand experience* tidak boleh dikaburkan oleh teknologi dan pengembangan perdagangan harus sederhana, ringan, dan menyenangkan. Sebagai pengembang, pendiri Moltin mengakui bahwa platform *e-commerce*, meskipun dioptimalkan untuk saluran tradisional, menghalangi kebebasan kreatif merek karena template yang kaku, bahasa pemrograman yang rumit, dan kurva belajar yang curam.

Moltin adalah solusi *microservices* yang memberi organisasi fleksibilitas tertinggi untuk mengambil pendekatan pengalaman pertama dalam perdagangan, dengan API paling sederhana dan paling kuat yang tersedia. Dengan Moltin, pengecer dapat membayangkan *brand experience* yang kaya dan dinamis yang melibatkan dan mengubah konsumen di saluran atau titik kontak mana pun. Sementara pengembang dapat membawa mereka untuk hidup dengan cepat dan mudah tanpa kendala teknologi warisan. Dengan Moltin, ide menjadi kenyataan, esok hari menjadi hari ini.

Moltin memiliki beberapa kategori *service*, yaitu *ventory, cart & checkout, orders, payment,* dan *content & schemas*. Pada kategori *orders*, Moltin memberikan layanan sinkronisasi *order retailer* dengan *inventory*-nya secara *real-time*. Tetap up-to-date dengan penjualan *retailer* saat terjadi. Dapatkan data penjualan real-time dan pelaporan lanjutan. Konversikan *cart*s pelanggan dengan mudah ke pesanan dengan proses pemeriksaan yang efisien kami atau buat pesanan secara otomatis dengan satu panggilan API. Moltin dapat membawa pesanan dari semua saluran penjualan *retailer* ke dalam satu sistem pusat. Membuat pesanan penjualan dari permintaan grosir, atau tambahkan secara otomatis melalui saluran penjualan online *retailer*. Dan Moltin mudah terintegrasi dengan dengan *software* pihak ketiga.

Moltin API mengikuti arsitektur *microservice* yang memungkinkan untuk menyambungkan perdagangan ke bagian mana pun dari aplikasi. API dibuat berdasarkan spesifikasi JSON API, mengikuti REST URL yang dapat diprediksi dan mendukung pembagian sumber daya lintas-asal.

2.2. EVALUASI PERBEDAAN

Setelah melakukan pencarian terkait API yang sejenis dengan API yang akan saya bangun, ditemukan Moltin. Pada kategori *orders*, API yang disediakan oleh Moltin memiliki konsep yang sama dengan API yang akan saya bangun. Berikut adalah penjelasan mengenai perbedaan antara Moltin dengan API yang akan saya bangun.

Tabel 1 Evaluasi Perbedaan

Moltin	API yang Akan Dibangun
Memiliki banyak *service*	Hanya terdapat satu fungsi *service* (*manage orders*)
Terintegrasi dengan saluran *e-commerce* sehingga bisa menyimpan data pesanan dari berbagai saluran	Tidak terintegrasi dengan saluran mana pun
Sudah sinkron dengan service lainnya (*inventory*, *payment,* dll)	Tidak tersinkronisasi dengan fungsi lain

3. DENIFISI *USE CASE*

"Sebagai *developer*, saya ingin membuat aplikasi yang dapat *create, read, update,* dan *delete* data produk"

"Sebagai *developer*, saya ingin membuat aplikasi yang dapat *create, read,* dan *delete* data pesanan"

"Sebagai *developer*, saya membutuhkan data yang telah terintegrasi antara produk dan pesanan"

Developer adalah *customer* dalam kasus ini, artinya *developer* yang akan menggunakan API yang akan dibangun. Ketiga *use case* di atas adalah bentuk *customer needs statement* yang nantinya dapat diterjemahkan menjadi kebutuhan fungsional. Ketiga *use case* di atas adalah gambaran bagaimana kebutuhan fungsionalitas yang harus dipenuhi dalam pembangunan API ini.

4. MEMILIH TKNOLOGI

4.1. TEKNOLOGI REST

REST (*Representational State Transfer*) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai protocol untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000.

Pada arsitektur REST, REST server menyediakan resources (sumber daya/data) dan REST client mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya. Setiap resource diidentifikasi oleh URIs (Universal Resource Identifiers) atau global ID. Resource tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

Berikut metode HTTP yang umum digunakan dalam arsitektur berbasis REST.

1. **GET**, menyediakan hanya akses baca pada resource.
2. **PUT**, digunakan untuk menciptakan resource baru.
3. **DELETE**, digunakan untuk menghapus resource.
4. **POST**, digunakan untuk memperbarui resource yang ada atau membuat resource baru.
5. **OPTIONS**, digunakan untuk mendapatkan operasi yang disupport pada resource.

4.2. PRO DAN KONTRA

Dalam menentukan teknologi, terdapat beberapa pertimbangan mengenai pro dan kontra terkait REST. Berikut adalah pro dan kontra dalam penggunaan REST.

Tabel 2 Pro dan Kontra

PRO	KONTRA
• Bahasa dan platform agnostic • Lebih sederhana/simpel untuk dikembangkan ketimbang SOAP • Mudah dipelajari, tidak bergantung pada tools • Ringkas, tidak membutuhkan layer pertukaran pesan (messaging) tambahan • Secara desain dan filosofi lebih dekat dengan web	• Mengasumsi model point-to-point komunikasi - tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara • Kurangnya dukungan standar untuk keamanan, kebijakan, keandalan pesan, dll, sehingga layanan yang mempunyai persyaratan lebih canggih lebih sulit untuk dikembangkan ("dipecahkan sendiri") • Berkaitan dengan model transport HTTP

5. SPESIFIKASI TEKNIS API

5.1. Judul

Managing Order API

5.2. Masalah

Di Indonesia telah banyak *e-commerce* yang berkembang. Banyak para penjual atau yang memiliki usaha yang menjual barang-barangnya di *platform-platform e-commerce* yang ada. Penjual ingin mengetahui *order* apa saja yang dilakukan oleh *customer* terkait dengan produk-produk yang dijualnya melalui aplikasi yang dibangun oleh *developer*. *Developer* harus mampu memberikan layanan tersebut. *Order* atau pesanan harus dapat terintegrasi dengan data produk-produk yang dijual oleh penjual. Seorang pelanggan tidak dapat membeli barang yang tidak dijual oleh penjual. Pengelolaan data *order* dan *product* ini menjadi hal yang cukup rumit bagi *developer*.

5.3. Solusi

Berdasarkan masalah yang telah dipaparkan di atas, *developer* memerlukan sebuah pengelolaan data terkait pesanan dan produk yang dijual oleh penjual. Oleh karena itu, sebagai solusi dari masalah tersebut, dibangunlah API yang menyediakan pengelolaan data terkait pesanan dan produk-produk yang dijual. Hal ini sangat membantu *developer* dalam membangun aplikasi *e-commerce* sehingga *developer* hanya perlu menggunakan API ini untuk pengelolaan pesanan.

5.4. Implementasi

Untuk mengimplementasikan API ini, telah ditentukan beberapa hal yang harus dipertimbangkan. Berikut adalah hal-hal yang telah ditentukan terkait implementasi.

1. Menggunakan bahasa pemrograman javascript

   Alasan menggunakan bahasa pemrograman javascript karena setelah saya melakukan riset dan pencarian bahasa yang apling sering digunakan di internet adalah bahasa javascript. Hal ini bertujuan untuk mempermudah saya dalam membangun API ini karena banyak contoh yang bisa didapatkan dari internet.

2. Menggunakan REST untuk akses

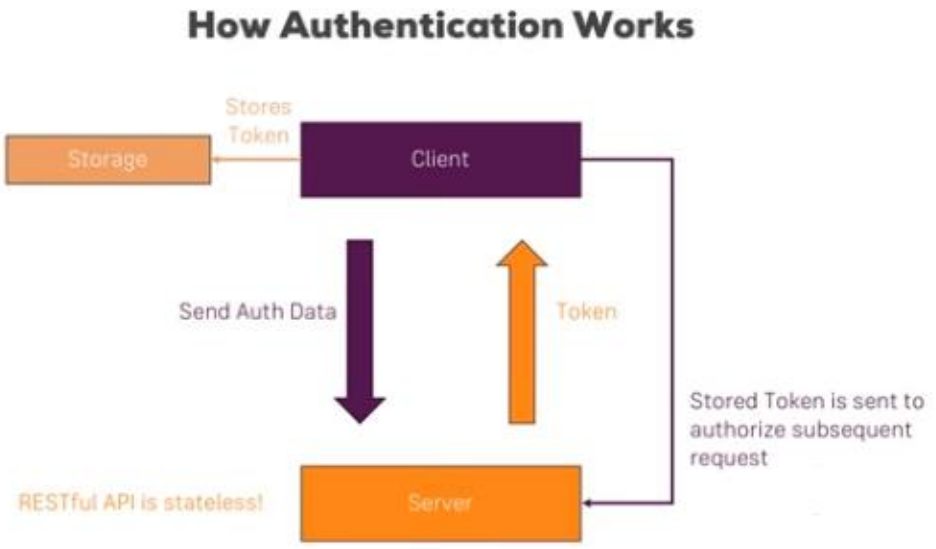
Alasan menggunakan REST adalah tidak adanya kebutuhan untuk *event transport* dan operasi data cocok dengan operasi CRUD.

3. Menggunakan JSON Web Token (JWT) untuk autentikasi

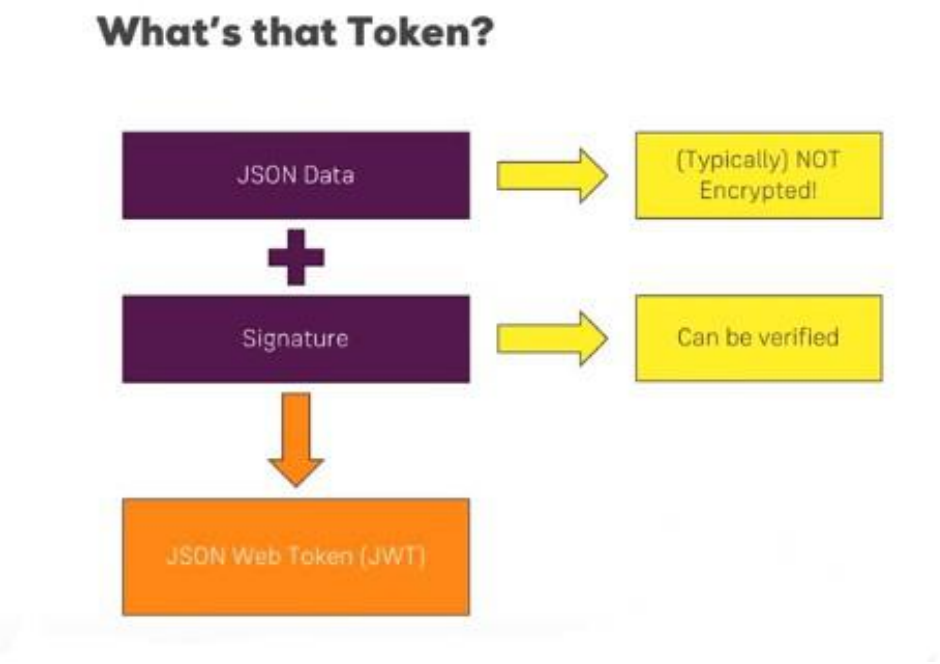
Alasan menggunakan JWT adalah data yang disimpan dalam bentuk JSON. JWT, sesuai dengan, namanya mendukung pengubahan JSON data + signature menjadi token.

5.5. Autektikasi

Di dalam layanan API ini akan menggunakan autentikasi yang mengimplementasikan token. Untuk menjaga kemanan dan kerahasiaan data yang disediakan, tentunya tidak ingin menyimpan informasi *password* dari pengguna. Hal ini sangat tidak aman karena apabila ada seseorang yang dapat meretas basis data, maka seluruh data *password* tersebut akan terbaca. Oleh karena itu, digunakanlah token yang berlaku hanya dalam kurun waktu tertentu. Dan setiap *login* akan menghasilkan token yang berbeda-beda. Berikut adalah gambaran autentikasi ini bekerja.



Gambar 1 Autentikasi



Gambar 2 JSON Web Token

5.6. Rincian Spesifikasi API

Tabel berikut adalah spesifikasi rinci mengenai *input*, *output*, dan lingkup dari masing-masing *request* yang diberikan. Perhatikan bahwa di bawha ini adalah spesifikasi output apabila input benar atau valid, apabila ada kesalahan input akan menghasilkan pesan error yang tidak ditampilkan dibawah ini.

Tabel 3 Spesifikasi Rinci API

ID	URI	Input	Output	Lingkup

S-01	POST
	/user/signup
		Required:
		email (string),
		password (string)
			201 Created
			{
			"message": "User created!"
			}
				create

S-02	POST
	/user/login
		Required:
		email (string),
		password (string)
			200 OK
			{
			"message": "Auth successful!",
			"token": $token
			}
				read

S-03	GET
	/products
		-	200 OK
			{
			"count": array.length,
			"products": [
			{
			"name": string,
			"price": number,
			"_id": $_id,
			"request": {
			"type": "GET",
			"url": "http://localhost:3000/products/$_id"
			}
			}
			]
				read

S-04	GET
	/products/productId
		-	200 OK
			{
			"product": {
			"_id": $_id,
			"name": string,
			"price": number
			},
				read

			"request": {
			"type": "GET",
			"url": "http://localhost:3000/products"
			}
			}	

S-05	POST
	/products
		Required:
		name (string),
		price (number)
			201 Created
			{
			"message": "Created product succesfully",
			"createProduct": {
			"name": string,
			"price": number,
			"_id": $_id,
			"request": {
			"type": "GET",
			"url": "http://localhost:3000/products/$_id"
			}
			}
			}
				create

S-06	PATCH
	/products/productId
		Option:
		name (string),
		price (number)
			200 OK
			{
			"message": “Product updated”
			}
				update

S-07	DELETE
	/products/productId
		-	200 OK
			{
			"message": "Product succesfully deleted!",
			"request": {
			"type": "POST",
			"url": "http://localhost:3000/products",
			"body": {
			"name": "String",
			"price": "Number"
			}
			}
			}
				delete

S-08	GET
	/order
		-	200 OK
			"count": array.length,
			"order": [
			{
			"_id": $order_id,
			"product": {
			"_id": $product_id,
			"name": string,
			},
			"quantity": number,
			"request": {
			"type": "GET",
			"url": "http://localhost:3000/order/$order_id"
			}
			}
			]
				read

S-09	GET
	/order/orderId
		-	200 OK
			{
			"order": {
			"quantity": number,
			"_id": $order_id,
			"product": {
			"_id": $product_id,
			"name": string,
			"price": number
			}
			},
			"request": {
			"type": "GET",
			"url": "http://localhost:3000/order"
			}
			}
				read

S-10	POST /order	Required: productId ($product_id)	201 Created { "message": "Order stored!",	create

		Option:
		quantity (number), default 1
			"createdOrder": {
			"_id": $order_id,
			"product": $product_id,
			"quantity": number
			},
			"request": {
			"type": "GET",
			"url": "http://localhost:3000/order/$order_id"
			}
			}	

S-11	DELETE
	/order/orderId
		-	200 OK
			{
			"message": "Order deleted!",
			"request": {
			"type": "GET",
			"url": "http://localhost:3000/order",
			"body": {
			"productId": "ID",
			"quantity": "Number"
			}
			}
			}
				delete

DAFTAR PUSTAKA

Moltin. “E-commerce API”. https://moltin.com/headless-commerce-platform/order-management/. Diakses pada bulan November, 2018.

Newcyber, Apradis. “Kelebihan dan Kekurangan Web Service SAP vs REST”. http://apradisnewcyber.blogspot.com/2014/12/kelebihan-dan-kekurangan-web-service.html. Diakses pada bulan November, 2018.

Feridi. “Mengenal RESTful Web Service”. https://www.codepolitan.com/mengenal-restful-web-services. Diakses pada bulan November, 2018.