Implementasi REST API pada Crino.id

Febby Kurniawan
Program Studi Sarjana Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
18523273@students.uii.ac.id

Irving Vitra Paputungan
Program Studi Sarjana Informatika
Universitas Islam Indonesia
Yogyarkata, Indonesia
irving@uii.ac.id

***Abstrak*—PT Bhinneka Mentari Dimensi yang memiliki toko online bhinneka.com memiliki sebuah platform yang dapat mempermudah mengelola penjualan dalam beberapa marketplace bernama crino.id. Namun sistem tersebut saat ini masih merupakan sistem monolith atau berdiri sendiri, dibutuhkan pengembangan sistem ke dalam web service API. Makalah ini menyajikan pengembangan web service API pada crino.id agar sistem tersebut dapat disatukan dengan dashboard seller center bhinneka.com. Hasil pengembangannya yaitu implementasi REST API pada sistem crino.id sehingga dapat digunakan untuk melakukan transaksi pertukaran data dengan sistem bhinneka.com yang terpisah dan telah mengadopsi teknologi web service.**

Kata Kunci— Django Rest Framework, Web Service Rest API

I. PENDAHULUAN

Crino.id adalah salah satu *platform multi-channel* yang memudahkan untuk mengelola produk, stok, harga, dan promosi di semua toko *online* dari satu *dashboard*. Dengan crino.id, penjual bisa memanajemen pengadaan barang, listing produk baru, mengatur stok, memproses pesanan dari pembeli, mengelola pengiriman, hingga melihat laporan penjualan dari beberapa *marketplace* popular di Indonesia yang telah terintegrasi hanya dalam satu *platform*. Crino.id merupakan *platform* yang dikembangkan oleh PT Bhinneka Mentari Dimensi untuk mempermudah para *merchant* atau *seller* bhinneka.com memanajemen penjualan produknya[1]. Namun, sistem crino.id yang ada saat itu masih merupakan sistem yang berdiri sendiri atau bisa dibilang sistem dengan arsitektur monolith dan terpisah dengan toko online bhinneka.com. Berawal dari masalah tersebut, terdapat gagasan untuk menyatukan crino.id ke dalam dashboard seller center bhinneka.com yang sudah mengadopsi teknologi web service. Sehingga dibutuhkan pengembangan untuk mengubah sistem crino.id yang masih monolith dengan cara membuat atau memecah beberapa proses bisnis pada sistem yang ada menjadi web service API menggunakan arsitektur REST.

Web *service* adalah sistem perangkat lunak yang digunakan untuk mendukung interaksi dan interoperabilitas antar mesin melalui jaringan. Web *service* biasanya dimanfaatkan sebagai fasilitas untuk menyediakan layanan yang berisi informasi atau data yang dikirim melalui HTTP (*Hypertext Transfer Protocol*) sehingga bisa di akses oleh sistem lain yang berbeda *platform*, sistem operasi, maupun bahasa pemrograman[2]. REST (*Representational State Transfer*) adalah salah satu gaya arsitektur yang menerapkan konsep perpindahan antar *state*, setiap *state* mewakili satu *resource* yang ada pada server[3]. Setiap *resource* diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID[4]. Biasanya *resource* tersebut direpresentasikan dalam format JSON atau XML. Dalam penerapannya REST dibangun menggunakan protokol HTTP(*Hypertext Transfer Protocol*) dan memanfaatkan beberapa metode yang ada di dalam HTTP antara lain: *GET*, *POST*, *PUT*, dan *DELETE*[2]. REST bekerja dengan cara bernavigasi melalui link-link HTTP yang tersedia untuk melakukan aktivitas, sama seperti mengganti *state* dari halaman web, REST seakan-akan melakukan perpindahan *state* satu sama lain[4]. Sementara REST API atau RESTful API merupakan bentuk implementasi dari web *service* yang menggunakan prinsip dasar arsitektur REST[5].

Setelah melihat dan meninjau beberapa penelitian terdahulu, web service API dengan arsitektur REST dipilih karena dirasa memiliki keunggulan peforma kecepatan melakukan request dan response[5] atau melakukan proses pertukaran data[6] dibandingkan arsitektur lainnya. Kemudian adapun batasan yang dilakukan dalam pengembangan web *service* API sistem crino.id ini hanya sebatas pada penambahan fitur *customer management*, menggunakan web *framework* dari Bahasa pemrograman python yaitu Django.

II. KAJIAN PUSTAKA

Dalam penulisan karya ilmiah ini, terdapat beberapa sumber referensi penelitian terdahulu. Sumber referensi dengan kasus serupa dipilih untuk memenuhi kebutuhan sebagai landasan penulisan dan pengembangan sistem yang dapat dilihat pada tabel 1.

TABEL 1. PENELITIAN TERDAHULU

Arsitektur web service	Kelebihan dan kekurangan
SOAP [5]	Memiliki protokol HTTP, HTTPS, SMTP, FTP. Hanya memiliki format respon XML dengan spesifikasi SOAP. Aturan penulisan ketat mengikuti spesifikasi XML.
REST [6]	Memiliki protokol HTTP & HTTPS. Memiliki format respon XML, JSON, dan plain text lainnya. Tidak ada spesifikasi khusus aturan penulisan.

Berdasarkan sumber referensi penelitian terdahulu yang ada pada tabel 1, bisa disimpulkan bahwa penggunaan web service API dapat berguna banyak untuk kebutuhan pengembangan sistem perangkat lunak. Baik penggunaan web service API yang menggunakan gaya arsitektur REST maupun SOAP[5]. Pada penerapannya, web service API dengan arsitektur REST lebih mudah diimplementasi dibandingkan arsitektur SOAP, karena REST tidak memiliki spesifikasi khusus dalam penulisan, berbeda dengan SOAP yang memiliki aturan penulisan yang ketat mengikuti spesifikasi XML[6]. Selain itu, REST juga memiliki format

response yang lebih fleksibel karena memiliki pilihan seperti: JSON, XML, atau format teks lainnya, sehingga memudahkan penerima response, tidak seperti SOAP dengan XML-nya saja, sehingga sedikit sulit membaca dan memahaminya karena tidak memiliki pilihan format response lainnya[5]. Hal tersebut dapat berpengaruh juga dalam penggunaan bandwith, karena jika dalam jumlah request yang banyak, format response dengan Bahasa markup seperti XML relatif lebih boros bandwith[6]. Kemudian, berdasarkan dua penelitian sebelumnya pada tabel 1, terdapat hasil penelitian yang menguji request dan response web service API menggunakan arsitektur REST yang dibangun pada web framework FLASK memiliki performa yang lebih bagus dibandingkan SOAP[5]. Selain itu terdapat penelitian serupa yang menguji sinkronisasi 30 sampel data dengan jumlah yang bervariasi pada beberapa device smartphone berbeda, dihasilkan rata-rata waktu sinkronisasi data web service API menggunakan arsitektur REST lebih cepat dibandingkan SOAP[6]. Kedua arsitektur tersebut sebenarnya sama-sama membantu interaksi antar sistem perangkat lunak agar saling terhubung atau terintegrasi satu sama lain, sehingga dapat melakukan transaksi pertukaran data melalui web service API. Namun, berdasarkan perbandingan dari penelitian terdahulu, sepertinya REST lebih unggul dan cocok untuk diterapkan dalam pengembangan ini.

III. METODOLOGI

Dalam proses pengembangan sistem crino.id milik bhinneka ini terdapat tiga tahapan utama untuk menerapkan REST API yaitu perancangan, implementasi, dan pengujian yang mengadopsi pendekatan dari metode pengembangan agile menggunakan *framework* scrum.

A. *Perancangan REST API*

Tahap perancangan merupakan salah satu tahapan yang penting dilakukan pada tahap awal sebelum melakukan pengembangan sistem. Tahap perancangan dapat dijadikan dasar pedoman sebelum melakukan tahapan implementasi. Pada tahapan ini yang dilakukan antara lain yaitu membuat rancangan tambahan tabel basis data untuk kebutuhan mengimplentasikan fitur customer management pada sistem crino.id. Selain itu, perancangan endpoint juga dibuat untuk mempermudah pengembang dalam tahap implementasi.

B. *Implementasi REST API*

Tahap implementasi ini merupakan tahapan kedua setelah tahap perancangan. Pada tahap ini dijelaskan bagaimana mengimplementasi rancangan yang telah dibuat pada tahap sebelumnya ke dalam salah satu kerangka kerja web bahasa pemrograman Python yaitu Django dan menggunakan Django REST Framework untuk mempermudah penerapan arsitektur REST di dalam Django.

C. *Pengujian REST API*

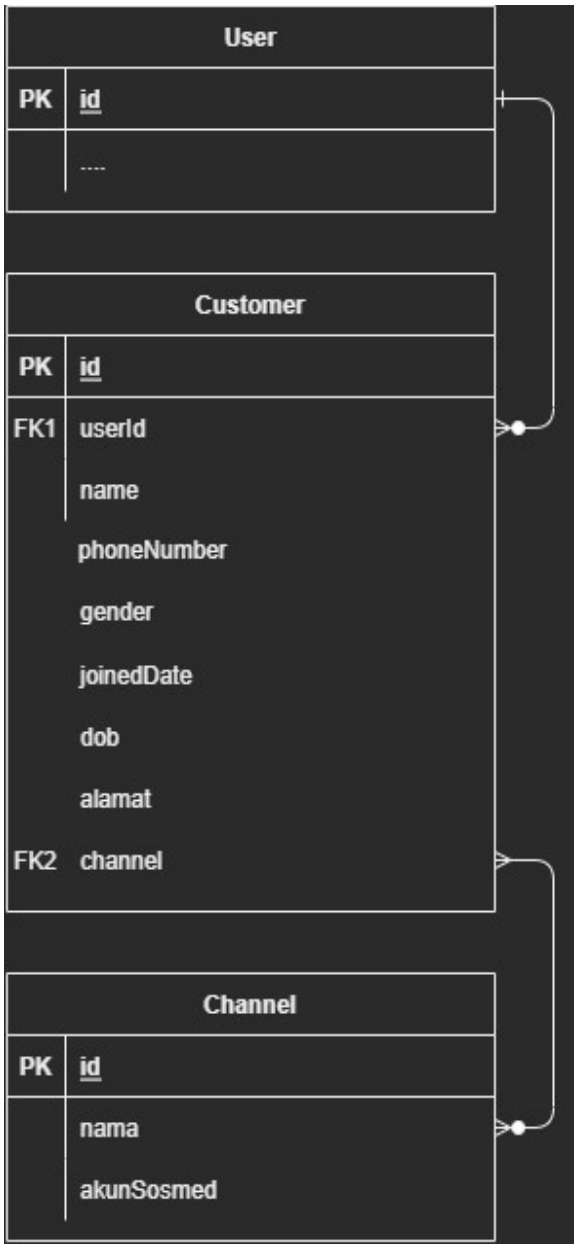
Tahap pengujian merupakan tahap ketiga atau tahap akhir. Pada tahap ini web service API di review atau diuji untuk memastikan bahwa fungsi dari setiap method telah berjalan dengan baik dan sesuai dengan rancangan. Pada tahap pengujian ini dilakukan dengan memanfaatkan perangkat lunak pengujian API bernama postman.

IV. HASIL DAN PEMBAHASAN

Pada bab ini, akan membahas hasil dari setiap tahapan yang sudah diuraikan pada bab sebelumnya.

A. *Perancangan REST API*

Sebagaimana yang telah dijelaskan pada metodologi, pada tahap ini dilakukan pembuatan rancangan tabel data tambahan dan endpoint yang akan di implementasikan ke tahap selanjutnya. Pada hasil rancangan tabel data telah ditambahkan tabel customer dan tabel channel seperti pada Gambar 1. Tabel basis data tersebut berfungsi untuk memenuhi kebutuhan pembuatan REST API customer management pada crino.id. Dengan adanya customer management diharapkan pengguna dapat memanajemen customer yang telah membeli produk nya.



Gambar 1 Rancangan Tabel Basis Data

Selain itu, terdapat rancangan endpoint yang akan diimplementasikan ke dalam web service API berbasis REST. Rancangan endpoint dibuat dengan ketentuan penamaan memiliki informasi seperti method, versi API, dan juga kata bentuk jamak agar mudah dimengerti oleh client seperti pada Tabel 2.

TABEL 2 RANCANGAN ENDPOINT

Method HTTP	Endpoint	Keterangan
GET	/api/v1/customers	Get all customer
GET	/api/v1/customers/[id]	Get detail customer
POST	/api/v1/customers	Create customer
PUT	/api/v1/customers/[id]	Update customer
DELETE	/api/v1/customers/[id]	Delete customer

B. *Implementasi REST API*

Hasil selanjutnya merupakan hasil implementasi dari tahap perancangan. Berdasarkan rancangan tabel basis data tambahan yang sudah dibuat, kemudian tabel data beserta atribut data nya diimplementasikan ke dalam kerangka kerja web dari bahasa pemrograman Python yaitu Django. Dapat dilihat model customer yang memiliki hubungan dengan model user dan channel telah ditulis ke dalam kode lengkap dengan atribut serta tipe data nya seperti pada Gambar 2. Adapun hasil implementasi pembuatan endpoint sesuai dengan rancangan seperti pada Gambar 3.

brittlestar-be > customer > models.py > Customer

```python
class Channel(models.Model):
    SOSMED = {
        ('OF', 'Offline Store'),
        ('WA', 'Whatsapp'),
        ('IG', 'Instagram'),
        ('FB', 'Facebook'),
        ('TW', 'Twitter'),
        ('LL', 'Lainnya'),
    }
    name = models.CharField(choices=SOSMED, max_length = 30)
    akunSosmed = models.CharField(max_length=30, null=True, blank=True)

    def __str__(self):
        return self.name

class Customer(models.Model):
    GENDER = {
        ('L', 'Laki-laki'),
        ('P', 'Perempuan'),
    }
    userId = models.ForeignKey(User, on_delete=models.CASCADE, blank=True, null=True)
    name = models.CharField(max_length=25)
    phoneNumber = models.CharField(max_length=20)
    gender = models.CharField(choices=GENDER, max_length=1, null=True, blank=True)
    joinedDate = models.DateTimeField(auto_now_add=True)
    dob = models.DateField(null=True, blank=True)
    alamat = models.CharField(max_length=200, null=True, blank=True)
    channel = models.ManyToManyField(Channel, related_name='channel')

    def __str__(self):
        return self.name
```

Gambar 2 Hasil Implementasi Tabel Data Customer

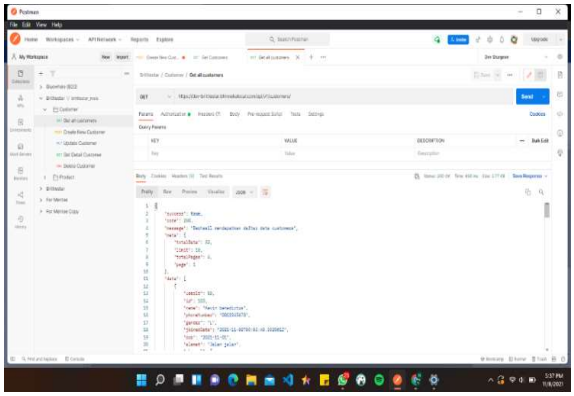
brittlestar-be > customer > urls.py > ...

```python
from django.contrib import admin
from django.urls import path, include
from django.conf.urls import url, include
from rest_framework import routers

from . import views

urlpatterns = [
    path('api/v1/customers/', views.CustomerList.as_view()),
    path('api/v1/customers/<int:pk>', views.CustomerDetail.as_view()),
]
```

Gambar 3 Hasil Implementasi Endpoint

C. *Hasil Pengujian REST API*

Setelah diimplementasi, kemudian dilakukan dan didapatkan hasil pengujian. Dari hasil pengujian ini sehingga dapat dilihat kelayakan dan keseuaian REST API yang telah dibuat. Dapat dilihat salah satu *service* yang digunakan untuk melihat semua data *customer* telah berhasil mengirim *response* melalui protokol HTTP seperti pada Gambar 4. Hasil *response* tersebut didapat dengan cara memanfaatkan aplikasi *testing* api yang bernama postman. Melalui postman yang seolah-olah menjadi rest *client* mengirimkan *request* melalui *endpoint* dan method yang telah ditetapkan, kemudian rest server akan mengirimkan response berupa *resource* dalam format data JSON. Response tersebut dapat dilihat berhasil tidaknya melalui status code 200 yang menandakan memiiki arti success.



Gambar 4 Hasil Pengujian REST API

V. KESIMPULAN

Berdasarkan hasil yang didapat mulai dari tahap perencanaan, implementasi, dan pengujian, dapat disimpulkan bahwa web *service* API customer management berbasis REST pada crino.id, telah berhasil dikembangkan dan diimplementasi. Pengembangan dilakukan dengan menggunakan kerangka kerja web bahasa pemrograman Python yaitu Django. Penerapan REST API dapat memudahkan pertukaran data atau informasi antar sistem tidak terbatas pada perbedaan teknologi apa yang digunakan. Perbedaan teknologi antar sistem dapat dimengerti oleh sistem lain dengan memanfaatkan REST yang berjalan diatas protokol HTTP sehingga dapat berkomunikasi. Selanjutnya, dengan adanya penelitian ini semoga dapat dijadikan acuan atau referensi untuk mengembangakn web *service* API menggunakan framework dan Bahasa pemrograman serta arsitektur lainnya.

REFERENSI

[1] Crinoid: Aplikasi Manajemen Marketplace Indonesia. https://www.bhinneka.com/promo/crinoid (accessed Jun. 19, 2022).

[2] R. Somya and T. M. E. Nathanael, "Pengembangan Sistem Informasi Pelatihan Berbasis Web Menggunakan Teknologi Web Service Dan Framework Laravel," *Techno Nusa Mandiri: Journal of Computing and Information Technology*, vol. 16, no. 1, pp. 51–58, Mar. 2019, doi: 10.33480/TECHNO.V16I1.164.

[3] R. Afrizal and Fitriyani, "Perancangan Web Service Berbasis REST API Untuk Aplikasi Penerimaan Peserta Didik Baru | eProsiding Teknik Informatika (PROTEKTIF)," 2021. https://eprosiding.ars.ac.id/index.php/pti/article/view/201 (accessed Jun. 13, 2022).

[4] A. Pamuji, "Rancang Bangun Web Service Menggunakan Representational State Transfer Untuk Pengolahan Data Barang - UTY Open Access," 2020. http://eprints.uty.ac.id/6287/ (accessed Jun. 13, 2022).

[5] M. G. L. Putra and M. I. A. Putera, "Analisis Perbandingan Metode Soap Dan Rest Yang Digunakan Pada Framework Flask Untuk Membangun Web Service," *SCAN - Jurnal Teknologi Informasi dan Komunikasi*, vol. 14, no. 2, pp. 1–7, Jun. 2019, doi: 10.33005/SCAN. V14I2.1480.

[6] A. Kusumaningrum, H. Sajati, and D. Anarianto, "Rest and Soap Comparison on Web Service Technology for Android Based Data Services | Kusumaningrum | Conference SENATIK STT Adisutjipto Yogyakarta." https://senatik.itda.ac.id/index.php/senatik/article/view/355 (accessed Jun. 18, 2022).