KERJA PRAKTIK - KI181330

Pengembangan Fitur Custom Data Widget Center pada Aplikasi Internal Blibli.com

PT Global Digital Niaga

Jl. Budi Kemuliaan 1 No. 1, Gambir
Jakarta Pusat, Jakarta 10110, Indonesia

Periode: 12 Juli 2021 – 03 September 2021

Oleh:
Bryan Gautama Ngo 05111840000011

Pembimbing Departemen
Ir. M.M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.
Pembimbing Lapangan
Wahyu Nur Ulil Albab, S.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

[*Halaman ini sengaja dikosongkan*]

KERJA PRAKTIK - KI181330

Pengembangan Fitur Custom Data Widget Center pada Aplikasi Internal Blibli.com

PT Global Digital Niaga

Jl. Budi Kemuliaan 1 No. 1, Gambir
Jakarta Pusat, Jakarta 10110, Indonesia

Periode: 12 Juli 2021 – 03 September 2021

Oleh:
Bryan Gautama Ngo 05111840000011

Pembimbing Departemen
Ir. M.M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.
Pembimbing Lapangan
Wahyu Nur Ulil Albab, S.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

KERJA PRAKTIK

Pengembangan Fitur Custom Data Widget Center pada Aplikasi Internal Blibli.com

Oleh:

Bryan Gautama Ngo 05111840000011

Disetujui oleh Pembimbing Kerja Praktik:

Ir. M.M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.
NIP. 197402092002121001

(Pembimbing Departemen)

Wahyu Nur Ulil Albab, S.Kom.

(Pembimbing Lapangan)

SURABAYA
7 Oktober 2021

[*Halaman ini sengaja dikosongkan*]

Pengembangan Fitur Custom Data Widget Center pada Aplikasi Internal Blibli.com

Nama	**:**	**Bryan Gautama Ngo**
NRP	**:**	**05111840000011**
Departemen	**:**	**Informatika FTEIC-ITS**
Pembimbing Depratemen	**:**	**M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.**

ABSTRAK

Tim-tim internal dalam sebuah perusahaan, termasuk Blibli.com, membuat sebuah keputusan berdasarkan data yang telah dipunyai. Pada kebanyakan perusahaan, termasuk Blibli.com, semua data terkait statistik perusahaan disimpan di dalam *database*. Namun, visualisasi data mentah pada *database* bisa dibilang kurang nyaman untuk dilihat. Blibli.com telah menyediakan sebuah *library Custom Data Widget* (CDW) yang memungkinkan pengguna internal untuk membuat *widget* untuk memvisualisasikan data yang ada. Namun dokumentasi yang ada pada *library* ini masih berupa tulisan-tulisan yang dianggap cukup sulit dimengerti oleh kebanyakan tim.

Untuk memudahkan pembelajaran dan untuk eksplorasi *library* yang telah ada, maka dikembangkanlah fitur baru *Custom Data Widget Center* untuk memuat dokumentasi yang interaktif dan memungkinkan pengguna untuk mengeksplorasi fitur *widget* yang telah disediakan oleh *library* CDW.

Fitur *Custom Data Widget Center* ini akan diimplementasikan pada sistem internal Blibli.com yang sudah berjalan, maka dari itu fitur tambahan ini harus mengikuti teknologi dan aturan yang sudah ada sehingga penambahan fitur baru ini tidak akan merusak sistem yang sudah berjalan. Hasil dari kerja praktik ini adalah berhasilnya pengimplementasian fitur *Custom Data Widget Center*.

Kata Kunci: blibli.com, custom data widget, custom data widget center.

KATA PENGANTAR

Puji dan syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena atas berkat limpahan rahmat dan lindungan-Nya penulis dapat melaksanakan salah satu kewajiban sebagai mahasiswa Teknik Informatika ITS yaitu Kerja Praktik (KP).

Penulis menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan kerja praktik maupun penyusunan buku laporan ini, namun kami berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan penulisan buku laporan ini.

Melalui laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada kepada orang-orang yang telah membantu dalam pelaksanaan kerja praktik hingga penyusunan laporan kerja praktik baik secara langsung maupun tidak langsung. Orang-orang tersebut antara lain adalah:

1. Orang tua penulis.
2. Bapak Ir. M.M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil., selaku dosen pembimbing kerja praktik yang telah membimbing penulis selama kerja praktik berlangsung.
3. Bapak Wahyu Nur Ulil Albab, S.Kom., selaku pembimbing lapangan selama kerja praktik yang telah memberikan bimbingan serta ilmunya kepada penulis.

Surabaya, Oktober 2021

Penulis

[*Halaman ini sengaja dikosongkan*]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dari waktu ke waktu, setiap perusahaan selalu melakukan perubahan untuk menjadi lebih baik. Pengambilan keputusan pada perubahan tadi tentunya didasari oleh data yang ada.

Blibli.com sebagai salah satu *e-commerce* terbesar di Indonesia yang telah beroperasi sejak tahun 2010, tentunya memiliki banyak tim internal yang membutuhkan data untuk pengambilan sebuah keputusan. Blibli.com menyimpan semua data di dalam *database* (basis data). Blibli.com telah menyediakan *library* bernama *Custom Data Widget*untuk membuat antar muka yang digunakan untuk memvisualisasikan data sehingga semua tim dapat membaca dan menganalisis dengan lebih mudah. Namun, dokumentasi pada *library* tersebut masih sangat minim, sehingga masih banyak tim lain yang kesulitan dalam menggunakannya.

Fitur baru yang akan ditambahkan adalah fitur bernama *Custom Data Widget Center* yang mana merupakan sebuah fitur yang diharapkan mampu membantu tim lain untuk mempelajari dan menggunakan fitur *Custom Data Widget*.

1.2 Tujuan

Tujuan Kerja praktik kali ini adalah mengimplementasikan fitur *Custom Data Widget Center*. Tujuan dari pengimplementasian tersebut antara lain:

1. Membuat fitur yang memungkinkan pengguna untuk mempelajari cara penggunaan *library Custom Data Widget*.
2. Membuat fitur yang memungkinkan pengguna untuk mengeksplorasi contoh penggunaan *Custom Data Widget*.

1.3 Manfaat

Berikut manfaat yang diperoleh melalui kerja praktik dalam pembuatan fitur *Custom Data Widget Center* pada aplikasi internal blibli.com:

1. Dapat meningkatkan kenyamanan pengguna dalam mempelajari dan menggunakan *library Custom Data Widget*.
2. Dapat memberikan sebuah platform untuk mempelajari penggunaan *library Custom Data Widget*.
3. Dapat merasakan suasana kerja disertai bimbingan dari pihak yang bertindak sebagai supervisor.

1.4 Rumusan Permasalahan

Berikut rumusan masalah dalam pelaksanaan kerja praktik pembuatan fitur *Custom Data Widget Center* pada blibli.com:

- Bagaimana membangun fitur *library Custom Data Widget* pada aplikasi internal blibli.com khususnya di bagian *back end*?

1.5 Lokasi dan Waktu kerja Praktik

Kerja praktik kali ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi	:	Daring
Alamat	:	Rumah Penulis
Waktu	:	12 Juli 2021 – 03 September 2021
Hari Kerja	:	Senin–Jumat
Jam Kerja	:	08.00–17.00 (*Full time*)

1.6 Metodologi Kerja Praktik

1.6.1 Perumusan Masalah

Untuk mengetahui domain dan fungsionalitas, dijelaskan secara rinci bagaimana sistem yang harus dibuat. Penjelasan oleh

pembimbing lapangan kerja praktik kali ini menghasilkan beberapa catatan mengenai gambaran secara garis besar tentang sistem berbasis *website* yang sebelumnya telah diterapkan. Setelah mendapatkan gambaran sistem, diskusi lebih lanjut dilakukan guna menentukan rancangan serta *tools* pendukung pembuatan sistem.

1.6.2 Studi Literatur

Pada tahap ini, setelah ditentukannya rancangan *database*, bahasa pemrograman sampai dengan teknologi beserta *tools* tambahan yang digunakan, dilakukan studi literatur lanjut mengenai bagaimana penggunaannya dalam membangun sistem sesuai yang diharapkan. Dikarenakan aplikasi yang akan dibuat merupakan bagian dari sistem yang sudah terbangun, maka secara garis besar *tools* yang digunakan juga tidak jauh berbeda. Bahasa pemrograman yang digunakan Java untuk *backend* dan *web server*, dengan bantuan kerangka kerja (*framework*) *Spring Framework*.

1.6.3 Analisis dan Perancangan Sistem

Langkah ini meliputi penjelasan awal tentang sistem. Bagaimana cara kerja sistem dengan skenario tertentu. Dari penjelasan awal telah didapatkan beberapa kebutuhan fungsional secara garis besar. Kemudian dilanjutkan dengan memperjelas dan menspesifikkan kebutuhan- kebutuhan tersebut. Dibuatlah sebuah diagram kasus penggunaan yang mewakili skenario-skenario untuk penggunaan fitur *Custom Data Widget Center* pada blibli.com.

1.6.4 Implementasi Sistem

Implementasi sistem didasarkan oleh perancangan dan analisis sebelumnya. Semua didasari pada rancangan *usecase* yang sudah ada sebelumnya dan penentuan *tools* yang telah dilakukan sebelumnya. Penentuan tipe data saat modifikasi sebuah entitas pada *collection* yang telah ada pada *database* disesuaikan juga dengan

kebutuhan. Pengerjaan dilakukan dengan progres setiap hari, dengan setiap harinya menargetkan perkembangan dari hari sebelumnya. Progres penyelesaian aplikasi terus dipantau oleh pihak Blibli melalui Pembimbing Lapangan.

1.6.5 Pengujian dan Evaluasi

Pengujian dilakukan oleh pembimbing lapangan setiap bagian dari fitur telah selesai dikerjakan untuk memberikan evaluasi ketika ada yang tidak sesuai, dan persetujuan apabila sudah sesuai.

1.7 Sistematika Laporan

Laporan kerja praktik ini terdiri dari 7 bab dengan rincian sebagai berikut:

1.7.1 Bab I: Pendahuluan

Bab ini berisi tentang latar belakan masalah, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2 Bab II: Profil Instansi

Bab ini berisi sekilas tentang profil PT Global Digital Niaga.

1.7.3 Bab III: Tinjauan Pustaka

Dalam bab ini dibahas mengenai konsep-konsep pembuatan aplikasi, dasar teori, teknologi yang dipakai dalam pembuatan aplikasi.

1.7.4 Bab IV: Analisis dan Perancangan Sistem

Dalam bab ini dibahas tentang proses analisa kebutuhan berdasarkan kondisi yang sesungguhnya dan perancangannya yang me-

liputi desain aplikasi yang akan dikembangkan. Proses analisa dan desain aplikasi menghasilkan daftar fitur, dan diagram alur aplikasi.

1.7.5 Bab V: Implementasi Sistem

Dalam bab ini dibahas tentang lapisan antarmuka, lapisan kontrol, lapisan data, dan antarmuka pengguna.

1.7.6 Bab VI: Pengujian dan Evaluasi

Dalam bab ini dibahas tentang lapisan antarmuka, lapisan kontrol, lapisan data, dan antarmuka pengguna.

1.7.7 Bab VII: Kesimpulan dan Saran

Bab ini berisi tentang kesimpulan dan saran yang didapatkan dari tugas selama kerja praktik.

[*Halaman ini sengaja dikosongkan*]

BAB II
PROFIL PERUSAHAAN

2.1 Profil Perusahaan

Blibli.com (PT. Global Digital Niaga) adalah perusahaan *e-commerce* di Indonesia yang bergerak dengan model bisnis B2B, B2C, dan B2B2C (*business to business to customer*).

Sebagai salah satu mal *online* terbesar dan terpercaya, Blibli menawarkan berbagai pilihan produk berkualitas yang disediakan oleh lebih dari 100.000 mitra usaha, mulai dari kebutuhan primer, produk elektronik termasuk gadget, kebutuhan sehari-hari hingga produk untuk keperluan gaya hidup. Kecepatan pengiriman di Blibli didukung oleh armada BES dan 15 mitra logistik, serta memiliki 20 gudang, dan 32 hub yang tersebar di kota-kota besar di Indonesia.

Dengan visi menjadi *e-commerce* nomor satu yang memiliki jumlah pelanggan setia terbanyak di Indonesia, Blibli berkomitmen untuk memberikan kenyamanan dan kepuasan belanja melalui 24/7 layanan *Customer Care*, kepastian 15 hari pengembalian produk serta pilihan pembayaran yang lengkap dan aman kepada pelanggan.

2.2 Logo Perusahaan

Logo perusahaan Blibli.com dapat dilihat pada Gambar 2.1.

Gambar 2.1 Logo Blibli.com

2.3 Lokasi Instansi

Penampakan depan kantor Blibli, gedung Sarana Jaya dapat dilihat pada Gambar 2.2.

Gambar 2.2 Foto kantor pusat Blibli.com

BAB III
TINJAUAN PUSTAKA

3.1 Aplikasi Web

Dalam rekayasa perangkat lunak, suatu aplikasi web adalah suatu aplikasi yang diakses menggunakan penjelajah web melalui suatu jaringan seperti Internet atau intranet. Ia juga merupakan suatu aplikasi perangkat lunak komputer yang dikodekan dalam bahasa yang didukung penjelajah web dan bergantung pada penjelajah tersebut untuk menampilkan aplikasi.

3.2 Java

Java adalah bahasa pemrograman berorientasi objek tingkat tinggi, berbasis kelas, yang dirancang untuk memiliki dependensi implementasi sesedikit mungkin. Jawa adalah bahasa pemrograman tujuan umum yang dimaksudkan untuk memungkinkan penulis program menulis sekali dan dapat berjalan di mana saja, yang berarti bahwa kode Java yang dikompilasi dapat berjalan di semua platform yang mendukung Java tanpa perlu kompilasi ulang.

Aplikasi Java biasanya dikompilasi ke *bytecode* yang dapat berjalan di *Java Virtual Machine* (JVM) apa pun terlepas dari arsitektur komputer yang mendasarinya. Java *runtime* menyediakan kemampuan dinamis (seperti refleksi dan modifikasi kode *runtime*) yang pada umumnya tidak tersedia dalam bahasa kompilasi tradisional. Java dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

3.3 Spring Framework

Spring Framework adalah kerangka kerja aplikasi dan wadah inversi kontrol untuk platform Java. Fitur inti kerangka kerja dapat

digunakan oleh aplikasi Java apa pun, tetapi terdapat ekstensi untuk membangun aplikasi web di atas platform Java. Meskipun kerangka tidak memaksakan model pemrograman tertentu, telah menjadi populer di komunitas Java sebagai tambahan untuk model Enterprise JavaBeans (EJB). Spring Framework bersifat *open source*.

3.4 Rest API

REST API atau biasa dikenal sebagai RESTful API adalah antarmuka pemrograman aplikasi (API atau *web* API) yang sesuai dengan batasan gaya arsitektur REST dan memungkinkan interaksi dengan layanan *web* RESTful. REST adalah singkatan dari *representational state transfer* dan diciptakan oleh ilmuwan komputer Roy Fielding.

3.5 Web Server

Web Server adalah sebuah perangkat lunak server yang berfungsi menerima permintaan HTTP atau HTTPS dari klien yang dikenal dengan *web browser* dan mengirimkan kembali hasilnya dalam halaman-halaman web yang umumnya berbentuk dokumen HTML.

3.6 NoSQL

Basis data NoSQL (*Not Only SQL*) adalah database non-tabular dan menyimpan data secara berbeda dari tabel relasional. Basis data NoSQL datang dalam berbagai jenis berdasarkan model datanya. Jenis utama adalah dokumen, nilai kunci, kolom lebar, dan grafik. Basis data NoSQL menyediakan skema dan skala yang fleksibel dengan mudah dengan sejumlah besar data dan beban pengguna yang tinggi.

3.7 MongoDB

MongoDB adalah program basis data berorientasi dokumen lintas platform yang bersifat *source-available*. MongoDB Diklasifikasikan sebagai salah satu program basis data NoSQL, MongoDB menggunakan dokumen mirip JSON dengan skema opsional. MongoDB dikembangkan oleh MongoDB Inc. dan dilisensikan di bawah Server Side Public License (SSPL).

3.8 Intellij IDEA

IntelliJ IDEA adalah lingkungan pengembangan terintegrasi (IDE) yang ditulis dalam Java untuk mengembangkan perangkat lunak komputer. Ini dikembangkan oleh JetBrains (sebelumnya dikenal sebagai IntelliJ), dan tersedia sebagai edisi komunitas Apache 2 Licensed, dan dalam edisi komersial berpemilik. Keduanya dapat digunakan untuk pengembangan komersial. Intellij IDEA sendiri pertama kali dirilis pada awal 2001 dan merupakan salah satu IDE Java pertama yang memiliki fitur navigas kode canggih dan kemampuan *refactoring* kode terintegrasi.

3.9 Widget

Widget adalah elemen antarmuka pengguna grafis (GUI) yang menampilkan informasi atau menyediakan cara khusus bagi pengguna untuk berinteraksi dengan sistem operasi atau aplikasi. Widget termasuk ikon, tombol, formulir, grafik dan sebagainya yang menampilkan informasi untuk pengguna.

3.10 Dokumentasi Perangkat Lunak

Dokumentasi perangkat lunak adalah teks atau ilustrasi tertulis yang menyertai perangkat lunak komputer, disertakan dalam kode sumber, atau dituliskan pada sebuah dokumen terpisah. Dokumentasi menjelaskan bagaimana perangkat lunak beroperasi atau

bagaimana menggunakannya, dan mungkin memiliki arti yang berbeda bagi orang-orang dalam peran yang berbeda. Dokumentasi digunakan agar pengguna dari perangkat lunak mampu memahami bagaimana cara penggunaan dari perangkat lunak tersebut.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisis Sistem

4.1.1 Definisi Umum Fitur

Secara umum fitur *Custom Data Widget Center* ini merupakan fitur tambahan pada aplikasi internal blibli.com yang memungkinkan tim internal untuk melihat, mempelajari, dan menggunakan *library Custom Data Widget* yang telah ada dengan tujuan untuk memudahkan visualisasi data yang ada. *Custom Data Widget Center* akan menyediakan tempat untuk eksplorasi *widget* yang telah dibuat dan menyediakan contoh-contoh *widget* yang dapat dipelajari oleh pengguna.

4.1.2 Analisis Kebutuhan Fungsional yang Ditambahkan

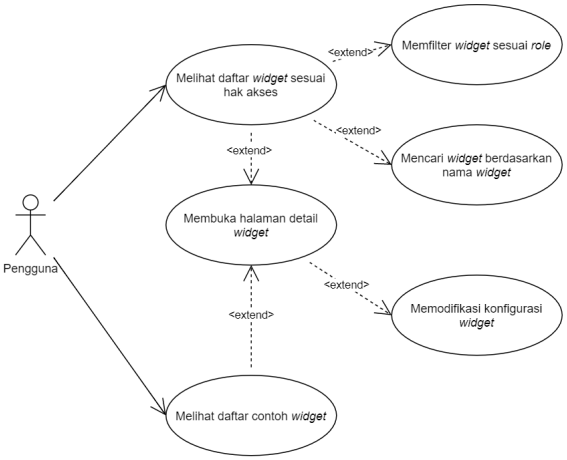
Beberapa kebutuhan fungsional yang perlu ditambahkan pada aplikasi internal blibli.com dapat dilihat pada Tabel 4.1.

Tabel 4.1 Kebutuhan Fungsional

Kode Kebutuhan	Deskripsi Kebutuhan
FR-001	Melihat daftar *widget* sesuai hak akses
FR-002	Membuka halaman detail *widget*
FR-003	Memodifikasi konfigurasi *widget*
FR-004	Memfilter *widget* sesuai *role*
FR-005	Mencari *widget* berdasarkan nama *widget*
FR-006	Melihat daftar contoh *widget*

4.2 Diagram Kasus Penggunaan

Pembahasan dengan pembimbing lapangan tentang fitur-fitur yang perlu ditambahkan pada aplikasi internal blibli.com untuk mengakomodir fitur *Custom Data Widget Center* menghasilkan beberapa fitur yang yang dijadikan diagram kasus penggunaan (*Use Case Diagram*) sehingga memudahkan untuk dipahami. *Use Case Diagram* yang telah dibuat dapat dilihat pada Gambar 4.1.



Gambar 4.1 Diagram *Use Case* dari fitur *Custom Data Widget Center* pada blibli.com

4.3 Spesifikasi Kasus Penggunaan

Pada subbab ini akan dijabarkan lebih dalam terkait kasus penggunaan yang terdapat pada diagram kasus penggunaan yang telah dibahas pada subbab 4.2.

4.3.1 Kasus Penggunaan 1: Melihat Daftar Widget Sesuai Hak Akses

Pada subbab ini akan dijabarkan spesifikasi dari kasus penggunaan 1 yaitu melihat daftar widget sesuai hak akses. Kasus penggunaan ini akan memenuhi kebutuhan FR-001 yang dijabarkan pada Tabel 4.1. Spesifikasi dari kasus penggunaan ini dapat dilihat pada Tabel 4.2.

Tabel 4.2 Tabel Kasus Penggunaan Melihat Daftar *Widget* Sesuai Hak Akses

Kode	UC001
Nama	Melihat daftar *widget* sesuai hak akses
Kebutuhan Fungsional	FR-001
Aktor	Pengguna
Deskripsi	Aktor dapat melihat daftar *widget* sesuai hak akses yang dimiliki aktor
Tipe	Fungsional
Relasi	-
Kondisi Awal	Aktor belum mengetahui daftar *widget* yang dapat diakses
Kondisi Akhir	Aktor mengetahui daftar *widget* yang dapat diakses
Alur Kejadian Normal	1. Aktor membuka halaman daftar *widget* 2. Sistem menampilkan daftar *widget* sesuai hak akses yang dimiliki aktor
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.2 Kasus Penggunaan 2: Membuka Halaman Detail Widget

Pada subbab ini akan dijabarkan spesifikasi dari kasus penggunaan 2 yaitu membuka halaman detail *widget*. Kasus penggunaan ini akan memenuhi kebutuhan FR-002 yang dijabarkan pada Tabel 4.1. Spesifikasi dari kasus penggunaan ini dapat dilihat pada Tabel 4.3.

Tabel 4.3 Tabel Kasus Penggunaan Membuka Halaman Detail *widget*

Kode	UC002
Nama	Membuka halaman detail *widget*
Kebutuhan Fungsional	FR-002
Aktor	Pengguna
Deskripsi	Aktor dapat melihat detail dari *widget* yang dipilih aktor
Tipe	Fungsional
Relasi	extend dari UC001 Melihat daftar *widget* sesuai hak akses dan UC006 Melihat daftar contoh *widget*
Kondisi Awal	Aktor belum mengetahui detail dari *widget* yang dipilih
Kondisi Akhir	Aktor mengetahui daftar *widget* yang dapat diakses
Alur Kejadian Normal	1. Aktor membuka halaman daftar *widget* 2. Sistem menampilkan daftar *widget* sesuai hak akses yang dimiliki aktor 3. Aktor menekan salah satu *widget* pada halaman daftar *widget* 4. Sistem menampilkan detail *widget* dari *widget* yang dipilih
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.3 Kasus Penggunaan 3: Memodifikasi Konfigurasi Widget

Pada subbab ini akan dijabarkan spesifikasi dari kasus penggunaan 3 yaitu memodifikasi konfigurasi *widget*. Kasus penggunaan ini akan memenuhi kebutuhan FR-003 yang dijabarkan pada Tabel 4.1. Spesifikasi dari kasus penggunaan ini dapat dilihat pada Tabel 4.4.

Tabel 4.4 Tabel Kasus Penggunaan Memodifikasi Konfigurasi *widget*

Kode	UC003
Nama	Memodifikasi konfigurasi *widget*
Kebutuhan Fungsional	FR-003
Aktor	Pengguna
Deskripsi	Aktor dapat memodifikasi konfigurasi pada *widget* yang dipilih aktor
Tipe	Fungsional
Relasi	extend dari UC002 Membuka halaman detail *widget*
Kondisi Awal	Aktor belum mengetahui detail dari *widget* yang dipilih
Kondisi Akhir	Aktor mengetahui daftar *widget* yang dapat diakses
Alur Kejadian Normal	1. Aktor membuka halaman daftar *widget* 2. Sistem menampilkan daftar *widget* sesuai hak akses yang dimiliki aktor 3. Aktor menekan salah satu *widget* pada halaman daftar *widget* 4. Sistem menampilkan detail *widget* dari *widget* yang dipilih 5. Aktor mengganti konfigurasi *widget* yang disediakan oleh sistem 6. Sistem menampilkan *widget* sesuai konfigurasi yang diberikan aktor
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.4 Kasus Penggunaan 4: Memfilter Widget Sesuai Role

Pada subbab ini akan dijabarkan spesifikasi dari kasus penggunaan 4 yaitu memfilter *widget* sesuai *role*. Kasus penggunaan ini akan memenuhi kebutuhan FR-004 yang dijabarkan pada Tabel 4.1. Spesifikasi dari kasus penggunaan ini dapat dilihat pada Tabel 4.5.

Tabel 4.5 Tabel Kasus Penggunaan Memfilter *Widget* Sesuai *Role*

Kode	UC004
Nama	Memfilter *widget* sesuai *role*
Kebutuhan Fungsional	FR-004
Aktor	Pengguna
Deskripsi	Aktor dapat memfilter *widget* sesuai *role* yang dimiliki aktor
Tipe	Fungsional
Relasi	extend dari UC001 Melihat daftar *widget* sesuai hak akses
Kondisi Awal	Daftar *widget* belum difilter berdasarkan *role* yang diinginkan aktor
Kondisi Akhir	Daftar *widget* telah difilter berdasarkan *role* yang diinginkan aktor
Alur Kejadian Normal	1. Aktor membuka halaman daftar *widget* 2. Sistem menampilkan daftar *widget* sesuai hak akses yang dimiliki aktor 3. Aktor menekan *drop down role* 4. Sistem menampilkan daftar *role* yang dimiliki aktor 5. Aktor memilih salah satu *role* dari *dropdown* 6. Sistem menampilkan daftar *widget* sesuai *role* yang dimiliki aktor
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.5 Kasus Penggunaan 5: Mencari Widget Berdasarkan Nama Widget

Pada subbab ini akan dijabarkan spesifikasi dari kasus penggunaan 5 yaitu mencari *widget* berdasarkan nama *widget*. Kasus penggunaan ini akan memenuhi kebutuhan FR-005 yang dijabarkan pada Tabel 4.1. Spesifikasi dari kasus penggunaan ini dapat dilihat pada Tabel 4.6.

Tabel 4.6 Tabel Kasus Penggunaan Mencari *Widget* berdasarkan nama *widget*

Kode	UC005
Nama	Mencari *widget* berdasarkan nama *widget*
Kebutuhan Fungsional	FR-005
Aktor	Pengguna
Deskripsi	Aktor dapat mencari *widget* berdasarkan nama *widget*
Tipe	Fungsional
Relasi	extend dari UC001 Melihat daftar *widget* sesuai hak akses
Kondisi Awal	Daftar *widget* belum difilter berdasarkan nama *widget* yang diinginkan aktor
Kondisi Akhir	Daftar *widget* telah difilter berdasarkan nama *widget* yang diinginkan aktor
Alur Kejadian Normal	1. Aktor membuka halaman daftar *widget* 2. Sistem menampilkan daftar *widget* sesuai hak akses yang dimiliki aktor 3. Aktor mengisi *text box* pada bar pencarian 4. Aktor menekan tombol *search* 5. Sistem menampilkan daftar *widget* nama *widget* yang dimasukkan aktor
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.6 Kasus Penggunaan 6: Melihat Daftar Contoh Widget

Pada subbab ini akan dijabarkan spesifikasi dari kasus penggunaan 6 yaitu melihat daftar contoh *widget*. Kasus penggunaan ini akan memenuhi kebutuhan FR-006 yang dijabarkan pada Tabel 4.1. Spesifikasi dari kasus penggunaan ini dapat dilihat pada Tabel 4.7.

Tabel 4.7 Tabel Kasus Penggunaan Melihat Daftar Contoh *Widget*

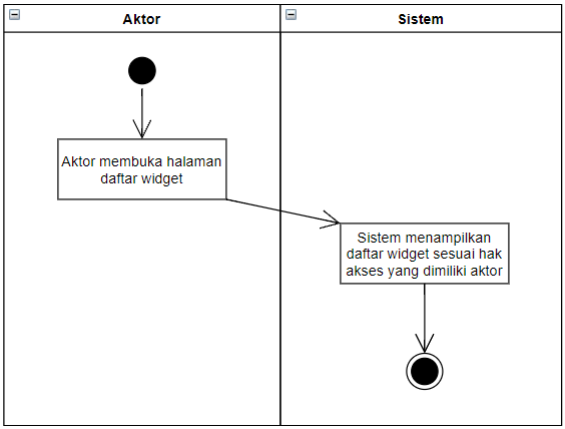
Kode	UC006
Nama	Melihat daftar contoh *widget*
Kebutuhan Fungsional	FR-006
Aktor	Pengguna
Deskripsi	Aktor dapat mencari *widget* berdasarkan nama *widget*
Tipe	Fungsional
Relasi	-
Kondisi Awal	Aktor belum mengetahui daftar contoh *widget* yang dapat diakses
Kondisi Akhir	Aktor mengetahui daftar contoh *widget* yang dapat diakses
Alur Kejadian Normal	1. Aktor membuka halaman daftar *widget* 2. Sistem menampilkan daftar *widget* sesuai hak akses yang dimiliki aktor 3. Aktor Menekan salah satu tipe *widget* pada menu bagian kiri 4. Sistem menampilkan daftar contoh *widget* dengan tipe yang dipilih oleh aktor
Alur Kejadian Alternatif	-
Pengecualian	-

4.4 Diagram Aktivitas

Pada subbab ini akan dijabarkan alur kejadian dari setiap kasus penggunaan yang telah dispesifikasikan pada subbab 4.3. Alur kejadian ini akan digambarkan dalam bentuk diagram.

4.4.1 Kasus Penggunaan 1: Melihat Daftar Widget Sesuai Hak Akses

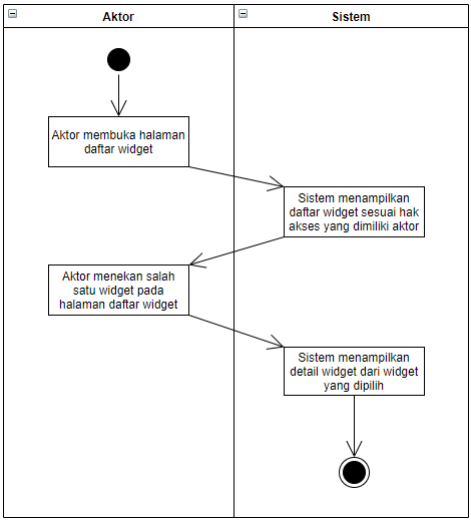
Pada subbab ini akan dijabarkan alur kejadian dari kasus penggunaan 1 yaitu melihat daftar *widget* sesuai hak akses. Diagram ini akan digambarkan berdasarkan alur kejadian pada spesifikasi kasus penggunaan pada Tabel 4.2. Diagram aktivitas pada kasus penggunaan ini dapat dilihat pada Gambar 4.2.



Gambar 4.2 Diagram Aktivitas Melihat Daftar *Widget* Sesuai Hak Akses

4.4.2 Kasus Penggunaan 2: Membuka Halaman Detail Widget

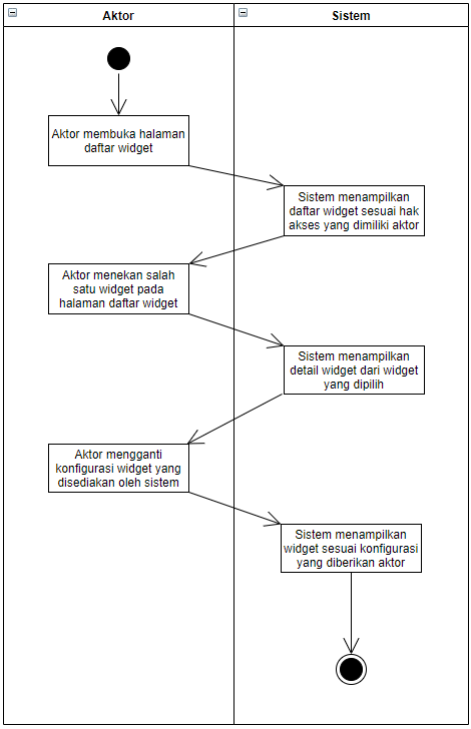
Pada subbab ini akan dijabarkan alur kejadian dari kasus penggunaan 2 yaitu membuka halaman detail *widget*. Diagram ini akan digambarkan berdasarkan alur kejadian pada spesifikasi kasus penggunaan pada Tabel 4.3. Diagram aktivitas pada kasus penggunaan ini dapat dilihat pada Gambar 4.3.



Gambar 4.3 Diagram Aktivitas Membuka Halaman Detail *widget*

4.4.3 Kasus Penggunaan 3: Memodifikasi Konfigurasi Widget

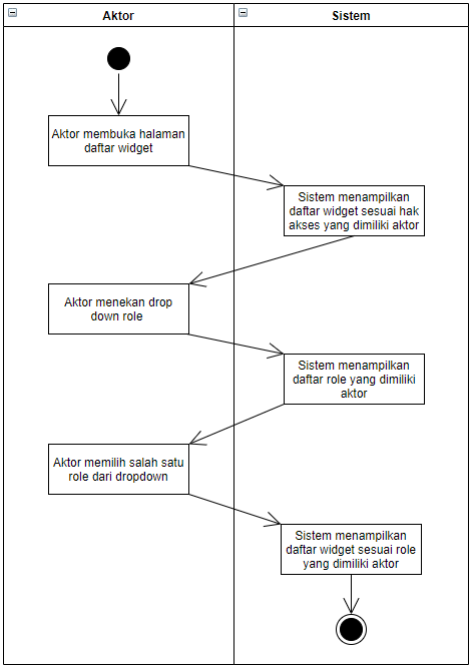
Pada subbab ini akan dijabarkan alur kejadian dari kasus penggunaan 3 yaitu memodifikasi konfigurasi *widget*. Diagram ini akan digambarkan berdasarkan alur kejadian pada spesifikasi kasus penggunaan pada Tabel 4.4. Diagram aktivitas pada kasus penggunaan ini dapat dilihat pada Gambar 4.4.



Gambar 4.4 Diagram Aktivitas Memodifikasi Konfigurasi *widget*

4.4.4 Kasus Penggunaan 4: Memfilter Widget Sesuai Role

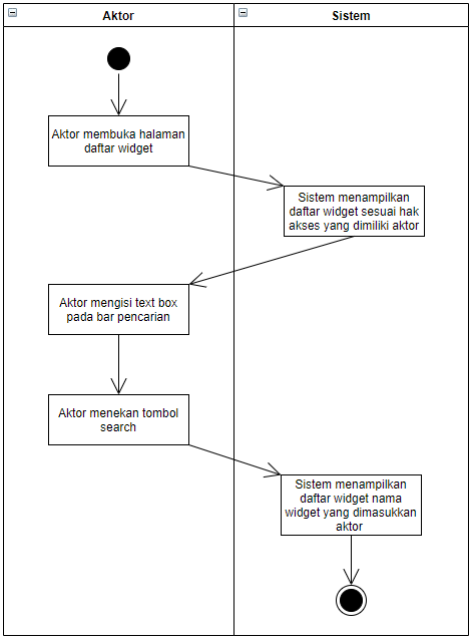
Pada subbab ini akan dijabarkan alur kejadian dari kasus penggunaan 4 memfilter *widget* sesuai *role*. Diagram ini akan digambarkan berdasarkan alur kejadian pada spesifikasi kasus penggunaan pada Tabel 4.5. Diagram aktivitas pada kasus penggunaan ini dapat dilihat pada Gambar 4.5.



Gambar 4.5 Diagram Aktivitas Memfilter *Widget* Sesuai *Role*

4.4.5 Kasus Penggunaan 5: Mencari Widget Berdasarkan Nama Widget

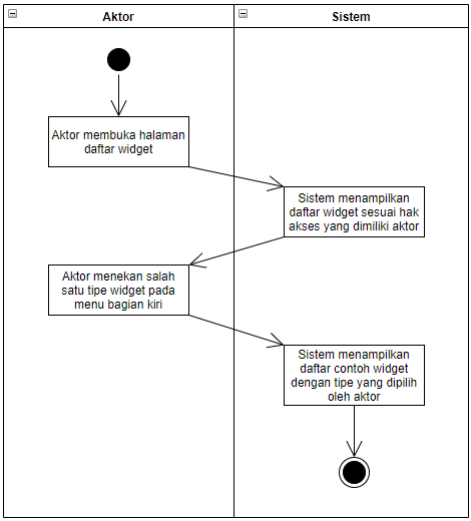
Pada subbab ini akan dijabarkan alur kejadian dari kasus penggunaan 5 yaitu mencari *widget* berdasarkan nama *widget*. Diagram ini akan digambarkan berdasarkan alur kejadian pada spesifikasi kasus penggunaan pada Tabel 4.6. Diagram aktivitas pada kasus penggunaan ini dapat dilihat pada Gambar 4.6.



Gambar 4.6 Diagram Aktivitas Mencari *Widget* Berdasarkan Nama *widget*

4.4.6 Kasus Penggunaan 6: Melihat Daftar Contoh Widget

Pada subbab ini akan dijabarkan alur kejadian dari kasus penggunaan 6 yaitu melihat daftar contoh *widget*. Diagram ini akan digambarkan berdasarkan alur kejadian pada spesifikasi kasus penggunaan pada Tabel 4.7. Diagram aktivitas pada kasus penggunaan ini dapat dilihat pada Gambar 4.7.



Gambar 4.7 Diagram Aktivitas Melihat Daftar Contoh *Widget*

BAB V
IMPLEMENTASI SISTEM

Pada bab ini dijelaskan tentang implementasi dari perancangan fitur dan pengaplikasian fitur.

5.1 Implementasi Sistem

Fitur yang dibuat merupakan tambahan fitur dari aplikasi internal blibli.com yang diimplementasikan pada bagian *backend*. Bagian *backend* diimplementasikan menggunakan *Spring Framework* di mana sebagian besar menggunakan Java sebagai bahasa utamanya.

Fitur untuk kebutuhan *Custom Data Widget Center* ini membutuhkan koneksi dengan basis data. Aplikasi yang telah ada menggunakan spesifikasi REST API. Sehingga diperlukan untuk membuat beberapa API baru untuk mengembangkan fitur ini, yaitu adalah API untuk melakukan listing *widget* sesuai dengan hak akses yang dimiliki pengguna, listing *widget* sesuai dengan *role* yang dipilih pengguna, listing dari *widget* sesuai dengan *widget name* yang dimasukkan pengguna, listing *role* yang dimiliki pengguna, melihat detail dari sebuah *widget*. Untuk memodifikasi konfigurasi dari sebuah *widget*, tidak ada API yang dipanggil dari backend ini, karena modifikasi konfigurasi tidak akan memengaruhi konfigurasi yang telah ada di basis data, karena modifikasi konfigurasi ditunjukkan untuk kebutuhan pembelajaran. Untuk kebutuhan listing dari *widget example* juga tidak ada API yang dipanggil dari backend, karena semua contoh dari *widget* akan dibuat di luar aplikasi sehingga listing dan navigasinya akan di *hard coded* di bagian frontend.

Fitur yang ditambahkan ini akan memungkinkan pengguna internal blibli.com untuk mempelajari cara menggunakan *library custom data widget* yang telah dibuat sebelumnya dan untuk mengeksplorasi semua fitur yang ditawarkan *library* ini. Dengan adanya

Custom Data Widget Center ini, diharapkan memudahkan semua pengguna internal blibli.com untuk memvisualisasikan data yang dibutuhkan sehingga kebutuhan tim internal blibli dapat terpenuhi.

5.2 Implementasi Widget Listing Serta Filter

Implementasi akan dimulai dari bagian *controller*, yang mana untuk setiap API untuk fitur ini nantinya akan berawalan ”/api/center”. Untuk kebutuhan *widget listing*, API ini juga akan menggunakan *pagination*. Karena akan digabungkan dengan kebutuhan *filter* baik dari *role* (di aplikasi ini nantinya *role* akan disebut sebagai *app name*) atau nama *widget* (di aplikasi ini nantinya nama *widget* akan disebut sebagai *widget id*), maka API ini akan membutuhkan beberapa *request parameter* seperti ”page” dan ”size” untuk kebutuhan paginasi, ”search” untuk kebutuhan memfilter berdasarkan widgetId, dan ”app-name” untuk kebutuhan memfilter berdasarkan appName.

API ini nantinya akan memanggil fungsi *execute* pada *Command* bernama *WidgetPreviewListCommand* yang membutuhkan parameter berupa *WidgetListPagingCommandRequest*. Kelas *WidgetListPagingCommandRequest* didefinisikan pada Kode Sumber 5.1.

```
1 public class WidgetListPagingCommandRequest {
2   private String key = "";
3   private String appName = "";
4   private int page;
5   private int size;
6 }
```

Kode Sumber 5.1 Kelas *WidgetPreviewListCommand*

Sehingga *controller* untuk kebutuhan implementasi *widget listing* beserta dengan *filter* dapat dilihat pada Kode Sumber 5.2.

```
@GetMapping("/widgets")
public
      Mono<Response<List<WidgetPreviewResponse>>>
      getWidgetPreviewList(@RequestParam(value =
      "search", required = false) String searchKey,
      @RequestParam("page") int page,
      @RequestParam("size") int size,
      @RequestParam(value = "app-name", required =
      false) String appName){
WidgetListPagingCommandRequest
      pagingCommandRequest =
      WidgetListPagingCommandRequest
   .builder()
   .page(page)
   .size(size)
   .key(searchKey == null ? "" :
         searchKey.toLowerCase())
   .appName(appName)
   .build();
return
      commandExecutor.execute(WidgetPreviewListComman
d.class, pagingCommandRequest)
   .subscribeOn(scheduler);
}
```

Kode Sumber 5.2 *Controller* untuk *widget listing*

Kemudian alur akan dilanjutkan dengan pemanggilan fungsi *execute* pada *WidgetPreviewListCommand*. Pada fungsi ini, akan dilakukan pemanggilan kueri untuk mencari *widget* sesuai dengan spesifikasi yang diberikan oleh *request* pada *repository*. Kemudian hasil kueri akan diubah bentuknya menjadi *web response* sesuai standard yang ada di blibli.com. Fungsi *execute* dapat dilihat pada Kode Sumber 5.3.

```
1 public
      Mono<Response<List<WidgetPreviewResponse>>>
      execute(WidgetListPagingCommandRequest
      request) {
2 PageRequest pageRequest =
      PageRequest.of(request.getPage(),
      request.getSize());
3
4 return Mono.fromCallable(() ->
      widgetPropertiesRepository.findByIdLikeAnd
5 ClientIdLikeAndComponentAuthorizationFunctionsIn
      (request.getKey(), request.getAppName(),
      currentUserHelper.getRoles(),
      pageRequest)).map(res ->
      toWebResponse(request, res.getContent(),
      res.getTotalElements()));
6
7 }
```

Kode Sumber 5.3 Fungsi *execute* untuk *widget listing*

Terlihat bahwa fungsi *execute* memanggil fungsi dari *widgetPropertiesRepository* yang mana merupakan *repository* dari *widget*. Fungsi findByIdLikeAndClientIdLikeAndComponentAuthorizationFunctionsIn bertujuan untuk melakukan kueri terhadap semua *widget* yang memiliki widgetId yang mengandung nama *widget* yang dimasukkan pengguna, memiliki hak akses yang sesuai dengan yang dipilih/dimiliki pengguna. Fungsi pada *repository* ini akan memanggil sebuah *native query* yang dapat dilihat pada Kode Sumber 5.4.

```
1 public
      Mono<Response<List<WidgetPreviewResponse>>>
      execute(WidgetListPagingCommandRequest
      request) {
2 @Query(value = "{ $and: [ { 'id': {$regex: ?0} },
      { 'component.authorizationFunctions' : { $in
      : ?2} }, { $or : [ { $expr: { $eq: ['?1',
      'null'] } } , { 'appName' : ?1 } ] }] }")
3   Page<WidgetProperties>
        findByIdLikeAndClientIdLikeAndComponent
4   AuthorizationFunctionsIn (String key, String
        appName, List<String> roles, Pageable
        pageable);
```

Kode Sumber 5.4 Fungsi *findByIdLikeAndClientIdLikeAndComponentAuthorizationFunctionsIn* pada *WidgetPropertiesRepository*

Setelah mendapatkan *widget list*, berikutnya dilakukan konversi ke *web response* sesuai standar Blibli.com dan menambahkan konfigurasi untuk *pagination*, pengkonversian ini akan ditangani oleh fungsi *toWebResponse* yang sebelumnya telah dipanggil oleh fungsi *execute* pada Kode Sumber 5.5.

Balikan dari *response* ini adalah *WidgetPreviewResponse* yang berisi widgetId dam appName dari *widget* yang sesuai dengan permintaan dari *request*. Kelas *WidgetPreviewResponse* dapat dilihat pada Kode Sumber 5.6.

```
1 private Response<List<WidgetPreviewResponse>>
      toWebResponse(WidgetListPagingCommandRequest
      pagingCommandRequest, List<WidgetProperties>
      data, Long count) {
2 int totalItem = count.intValue();
3 Paging paging = Paging.builder()
4   .page(pagingCommandRequest.getPage())
5   .totalPage(Math.floorDiv(totalItem,
        pagingCommandRequest.getSize())+1)
6   .itemPerPage(pagingCommandRequest.getSize())
7   .totalItem(totalItem)
8   .build();
9
10 List<WidgetPreviewResponse> responseList = new
      ArrayList<>();
11 data.forEach(e -> {
12   WidgetPreviewResponse widget =
        WidgetPreviewResponse
13     .builder()
14     .appName(e.getAppName())
15     .widgetId(e.getId())
16     .build();
17   responseList.add(widget);
18 });
19
20 Response<List<WidgetPreviewResponse>> response =
      ResponseHelper.ok(responseList);
21 response.setPaging(paging);
22
23 return response;
24 }
```

Kode Sumber 5.5 Fungsi *toWebResponse*

```
1 public class WidgetPreviewResponse {
2   private String widgetId;
3   private String appName;
4 }
```

Kode Sumber 5.6 Kelas *WidgetPreviewResponse*

5.3 Implementasi Widget Detail

Implementasi akan dimulai dari bagian *controller*. Untuk kebutuhan *widget detail*, karena diharapkan untuk mendapatkan detail dari sebuah *widget* pilihan pengguna, maka API ini juga akan meminta sebuah *path variable* yang dapat diisi dengan widgetId dari *widget* yang ingin dilihat detailnya.

API ini nantinya akan memanggil fungsi *execute* pada *Command* bernama *WidgetPropertiesCommand* yang membutuhkan parameter berupa *GetWidgetPropertiesCommandRequest*. Kelas *GetWidgetPropertiesCommandRequest* didefinisikan pada Kode Sumber 5.7.

```
1 public class GetWidgetPropertiesCommandRequest {
2   String widgetId;
3 }
```

Kode Sumber 5.7 Kelas *GetWidgetPropertiesCommandRequest*

Sehingga *controller* untuk kebutuhan implementasi *widget detail* dapat dilihat pada Kode Sumber 5.8.

```
1 @GetMapping("/widgets/{widgetId}")
2 public
      Mono<Response<GetWidgetPropertiesResponse>>
      getWidgetPropertiesCenter(
3 @PathVariable String widgetId) {
4     GetWidgetPropertiesCommandRequest
          commandRequest =
          GetWidgetPropertiesCommandRequest.builder()
5       .widgetId(widgetId)
6       .build();
7     return commandExecutor.execute
          (WidgetPropertiesCommand.class,
          commandRequest)
8       .map(ResponseHelper::ok)
9       .subscribeOn(scheduler);
10 }
```

Kode Sumber 5.8 *Controller* untuk *widget detail*

Kemudian alur akan dilanjutkan dengan pemanggilan fungsi *execute* pada *WidgetPropertiesCommand*. Pada fungsi ini, akan dilakukan yang pertama adalah pengecekkan hak akses apakah pengguna memiliki otoritas untuk mengakses *widget* ini dan pemanggilan kueri untuk mencari *widget* sesuai dengan widgetId yang terdapat di *path variable* pada *repository*. Kemudian hasil kueri akan diubah bentuknya menjadi *web response* sesuai standard yang ada di blibli.com. Fungsi *execute* dapat dilihat pada Kode Sumber 5.9.

```
1 public Mono<GetWidgetPropertiesResponse>
      execute(GetWidgetPropertiesCommandRequest
      request) {
2     return Mono.fromCallable(() ->
          widgetPropertiesRepository.findById
          (request.getWidgetId())
3         .orElseThrow(() -> new
              CdwRuntimeException("Failed to find
              widget properties " +
              request.getWidgetId())))
4         .flatMap(this.authHelper::
              verifyWidgetPropertiesRole)
5         .flatMap(this::toWebResponse);
6 }
```

Kode Sumber 5.9 Fungsi *execute* untuk *widget detail*

Terlihat bahwa fungsi *execute* memanggil fungsi dari *widgetPropertiesRepository* yang mana merupakan *repository* dari *widget*. Fungsi *findById* bertujuan untuk melakukan kueri terhadap semua *widget* yang memiliki widgetId yang sama persis dengan apa yang diminta oleh pengguna. Fungsi pada *repository* ini telah didefinisikan oleh Spring Framework secara *default* sehingga tidak perlu dilakukan pengkodean untuk kueri ini.

Setelah mendapatkan *widget*, berikutnya dilakukan konversi ke *web response* sesuai standar Blibli.com dan menambahkan konfigurasi untuk *pagination*, pengkonversian ini akan ditangani oleh fungsi *toWebResponse* yang sebelumnya telah dipanggil oleh fungsi

execute pada Kode Sumber 5.10.

```
 1 private Mono<GetWidgetPropertiesResponse>
       toWebResponse(WidgetProperties properties) {
 2
 3   return Mono.just(properties)
 4     .zipWhen(props ->
           Mono.just(mapper.convertValue(props,
           GetWidgetPropertiesResponse.class)))
 5     .doOnNext(tuple -> {
 6       GetWidgetPropertiesResponse.Component
             component =
             tuple.getT2().getComponent();
 7       component.setApiPath(ComponentType.fromValue
             (component.getType()).getPath());
 8     })
 9     .zipWhen(tuple ->
           dynamicSelectionHelper.getDynamicSelection
           (properties).collectList(),
10       (prev, opts) -> Tuples.of(prev.getT2(),
             opts))
11     .doOnNext(tuple ->
12       IntStream.range(0, tuple.getT2().size())
13         .forEach(i ->
               tuple.getT1().getQuerySchema()
               .getFilters().get(i).setOptions
               (tuple.getT2().get(i)))
14     ).map(tuple -> tuple.getT1());
15 }
```

Kode Sumber 5.10 Fungsi *toWebResponse*

Balikan dari *response* ini adalah *GetWidgetPropertiesResponse* yang berisi properti dari *widget* yang sesuai dengan permintaan dari pengguna. Kelas *GetWidgetPropertiesResponse* dapat dilihat pada Kode Sumber 5.11.

```
 1 public class GetWidgetPropertiesResponse {
 2   private String id;
 3   private String appName;
 4   private Component component;
 5   private QuerySchema querySchema;
 6
 7   public static class Component{
 8     private String name;
 9     private String type;
10     private String tooltip;
11     private Map<String, Object> properties;
12     private String apiPath;
13   }
14   public static class QuerySchema{
15     private List<Field> fields;
16     private List<Filter> filters;
17     private List<Setting> settings;
18   }
19   public static class Field {
20     private String displayName;
21     private String tooltip;
22     private FieldType type;
23     private Boolean sortable;
24     private Map<String, Object> properties;
25   }
26   public static class Filter {
27     private String ref;
28     private String displayName;
29     private String type;
30     private String visibility;
31     private Map<String, Object> properties;
32     private List<String> options;
33   }
34   public static class Setting {
35     private String type;
36     private Map<String, Object> properties;
37   }
38 }
```

Kode Sumber 5.11 Kelas *GetWidgetPropertiesResponse*

5.4 Implementasi Role Listing Pengguna

Implementasi akan dimulai dari bagian *controller*. Untuk kebutuhan *role listing* pengguna. API diharapkan untuk mendapatkan daftar dari *role* pengguna. API ini tidak akan meminta masukkan sama sekali, karena *role* dari pengguna selalu dikirimkan bersama dengan *request*, sehingga mendapatkan *role* dari sebuah pengguna dapat dilakukan tanpa masukkan tambahan dari API.

API ini nantinya akan memanggil fungsi *execute* pada *Command* bernama *GetAppNameCommand* yang tidak membutuhkan parameter spesifik. Sehingga *controller* untuk kebutuhan implementasi *role listing* pengguna dapat dilihat pada Kode Sumber 5.12.

```
1 @GetMapping("clients/app-names")
2 public Mono<Response<List<String>>>
      getClientIds() {
3   return commandExecutor.execute
        (GetAppNameCommand.class, "")
4     .map(ResponseHelper::ok)
5     .subscribeOn(scheduler);
6 }
```

Kode Sumber 5.12 *Controller* untuk *role listing* pengguna

Kemudian alur akan dilanjutkan dengan pemanggilan fungsi *execute* pada *GetAppNameCommand*. Pada fungsi ini, akan dilakukan pemanggilan kueri untuk mencari *appName* dari pengguna sesuai hak akses yang telah ada pada *request* pada *repository*. Hasil kueri ini tidak akan dikonversi menjadi *web response* secara manual karena balikannya adalah hanya *list of string* yang mana dapat dikonversi secara otomatis dengan *library* blibli.com yang digunakan.Fungsi *execute* dapat dilihat pada Kode Sumber 5.13.

```
1 public Mono<List<String>> execute(Object request)
      {
2    return Mono.fromCallable(() ->
         widgetPropertiesCustomRepository.
3      findAppNameDistinctByComponentAuthorizationsIn
           (currentUserHelper.getRoles()));
4 }
```

Kode Sumber 5.13 Fungsi *execute* untuk *role listing* pengguna

Terlihat bahwa fungsi *execute* memanggil fungsi dari *widgetPropertiesCustomRepository* yang mana merupakan *repository* turunan dari *WidgetPropertiesRepository*. *Custom repository* ini dibuat karena ada beberapa fitur yang hanya didukung oleh *library mongo template*. Fungsi findAppNameDistinctByComponentAuthorizationsIn bertujuan untuk melakukan kueri terhadap semua *appNames* yang dimiliki oleh *role* pengguna. Fungsi pada *repository* ini akan menggunakan *mongoTemplate* yang dapat dilihat pada Kode Sumber 5.14.

```
1 public List<String>
      findAppNameDistinctByComponentAuthorizationsIn
      (List<String> roles) {
2    Criteria criteria = new
         Criteria("component.authorizationFunctions")
         .in(roles);
3    Query query = new Query();
4    query.addCriteria(criteria);
5    return mongoTemplate.findDistinct(query,
         "appName", WidgetProperties.class,
         String.class);
6 }
```

Kode Sumber 5.14 Fungsi *findAppNameDistinctByComponentAuthorizationsIn* pada *widgetPropertiesCustomRepository*

5.5 Implementasi Modifikasi Konfigurasi Widget

Modifikasi konfigurasi pada sebuah *widget* tidak akan memanggil sebuah API pada backend karena tidak diinginkan untuk mengubah konfigurasi *widget* tersebut di basis data. Sehingga untuk implementasi modifikasi konfigurasi ini akan dilakukan pada sisi Javascript. Pengguna nantinya harus menekan sebuah tombol untuk menerapkan konfigurasi yang telah dimodifikasi pada *widget* yang akan memanggil fungsi bernama *updateConfigurations*. Konfigurasi yang dapat dimodifikasi oleh pengguna antara lain adalah tinggi dan lebar dari *widget*, *widgetConfig*, *widgetFilters*, dan *widgetSettings*. Kode sumber untuk fungsi *updateConfigurations* dapat dilihat pada Kode Sumber 5.15.

```
1 updateConfigurations() {
2     this.checkRefErrorOnClick()
3     if (this.isValidForm) {
4         this.updateWidthAndHeight()
5         this.broadcastConfig({content:
              this.widgetConfig, ...this.widgetIds})
6         this.broadcastFilter({content:
              this.widgetFilter, ...this.widgetIds})
7         this.broadcastSetting({content:
              this.widgetSetting,
              ...this.widgetIds})
8     }
9 }
```

Kode Sumber 5.15 Fungsi *updateConfigurations*

Karena *widgetConfig* akan bertipe JSON, maka dilakukan pengecekkan apakah JSON tersebut telah sesuai format yang ada, sehingga dilakukan pengecekkan untuk data-data yang dimasukkan terlebih dahulu. Kemudian, jika semua data yang dimasukkan telah memenuhi, maka dipangil fungsi-fungsi untuk melakukan *update* pada *widget* sesuai dengan konfigurasi yang dimasukkan yaitu fungsi *updateWidthAndHeight*, *broadcastConfig*, *broadcastFilter*, *broadcastSetting*. Fungsi *updateWidthAndHeight* bertanggung ja-

wab untuk melakukan pembaruan terhadap tinggi dan lebar dari *widget*. Fungsi *updateWidthAndHeight* dapat dilihat pada Kode Sumber 5.16.

```
1  updateWidthAndHeight() {
2      if(this.form.component.apiPath ===
           'area-chart' && (this.height < 300 ||
           this.height > 300)) {
3          return this.toast('The height for this
               widget cannot be configurated cause
               the type of this widget is area chart
               and the height must be equals 300')
4      } else if(this.form.component.apiPath ===
           'area-chart' && this.height === 300 &&
           (this.width >= 300 && this.width <=
           1300)) {
5          this.updateWidth = this.width
6          this.updateHeight = this.height
7      } else if((this.width < 300 || this.height <
           300) || (this.width > 1300 || this.height
           > 499)) {
8          return this.toast('Invalid value sizes,
               because the minimum size must be more
               then equals 300 px and the maximum
               size width is 1300 px and height is
               499 px')
9      }
10     this.updateWidth = this.width
11     this.updateHeight = this.height
12 }
```

Kode Sumber 5.16 Fungsi *updateWidthAndHeight*

Fungsi *broadcastConfig*, *broadcastFilter* dan *broadcastSetting* adalah fungsi yang hanya akan memanggil fungsi bawaan dari *library* yang dibuat oleh blibli.com bernama *widgetClient*, sehingga implementasi untuk fungsi *broadcastConfig*, *broadcastFilter* dan *broadcastSetting* dapat dilihat pada Kode Sumber 5.17.

```
broadcastFilter(params = {content: []}) {
  this.widgetClient.broadcastFilter(params)
},
broadcastSetting(params = {content: []}) {
  this.widgetClient.broadcastSetting(params)
},
broadcastConfig(params = {content: {}}) {
  this.widgetClient.broadcastConfig(params)
}
```

Kode Sumber 5.17 Fungsi *broadcastConfig, broadcastFilter, broadcastSetting*

5.6 Implementasi Widget Example Listing

Fitur untuk *Widget Example Listing* tidak akan memanggil API ke backend, melainkan akan di *hard coded* sehingga semua pengguna memiliki daftar *widget example* yang sama. Daftar *widget example* akan disimpan pada *file* Javascript bernama widget-example-list-constant.js yang nantinya dapat dibaca oleh frontend. Karena sangat panjang, penulis hanya akan menyertakan sebagian dari *file* widget-example-list-constant.js ini yang dapat dilihat pada Kode Sumber 5.18.

```
export default {
  examples:
  {
    'data-table': {
      listExample: [
        {
          name: 'Widget Title and Tooltip',
          description: 'Explain about how to set
              the name and tooltip, show it on UI
              and hide it',
          id:
              'widget-title-and-tooltip-data-table'
        },
        {
          name: 'Response for Unauthorized User',
          description: 'Create a widget with
              random Function, and explain about
              the response of the CDW UI',
          id:
              'response-for-unauthorized-user-data
              -table'
        },
        {
          name: 'Table Pagination',
          description: 'Explain about pagination,
              threshold, paginationDefault, and
              secondarySort',
          id: 'table-pagination-data-table'
        },
        {
          name: 'Table Limit',
          description: 'Explain about limit',
          id: 'table-limit-data-table'
        }
      ]
    }
  }
}
```

Kode Sumber 5.18 widget-example-list-constant.js

BAB VI
PENGUJIAN DAN EVALUASI

6.1 Skenario Pengujian

Pengujian akan dilakukan melalui *localhost* dikarenakan fitur masih belum di *deploy* hingga waktu berakhirnya masa kerja praktik penulis.

6.1.1 Membuka Halaman Widget List

Skenario pengujian aplikasi adalah sebagai berikut:

1. Membuka localhost:3001 pada browser.
2. Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu terdapat daftar *widget* sesuai hak akses yang dimiliki oleh pengguna, dan terdapat *pagination*.

6.1.2 Membuka Halaman Widget Detail

Skenario pengujian aplikasi adalah sebagai berikut:

1. Membuka localhost:3001 pada browser.
2. Menekan tombol *detail* di salah satu *widget*.
3. Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu terdapat properti dari *widget* seperti *widget id, app name, widget type, widget config, widget filters, dan widget settings*.

6.1.3 Memodifikasi Konfigurasi pada Widget Detail

Skenario pengujian aplikasi adalah sebagai berikut:

1. Membuka localhost:3001 pada browser.
2. Menekan tombol *detail* di salah satu *widget*.

3. Mengganti salah satu konfigurasi, misal panjang dari *widget*.
4. Menekan tombol *apply*.
5. Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu *widget* akan di perbarui dengan konfigurasi yang baru.

6.1.4 Melakukan Filter Berdasarkan App Name

Skenario pengujian aplikasi adalah sebagai berikut:

1. Membuka localhost:3001 pada browser.
2. Menekan *drop down* ”App Name”.
3. Menekan salah satu appName pada *drop down* ”App Name”.
4. Menekan tombol *search* yang ditandai dengan gambar kaca pembesar.
5. Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu terdapat daftar *widget* sesuai appName yang dipilih pengguna, dan terdapat *pagination*.

6.1.5 Melakukan Filter Berdasarkan Widget Id

Skenario pengujian aplikasi adalah sebagai berikut:

1. Membuka localhost:3001 pada browser.
2. Mengisi *text box* dengan *placeholder* ”WidgetId” dengan widgetId yang diinginkan.
3. Menekan tombol *search* yang ditandai dengan gambar kaca pembesar.
4. Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu terdapat daftar *widget* sesuai widgetId yang dimasukkan pengguna, dan terdapat *pagination*.

6.1.6 Membuka Halaman Widget Example List

Skenario pengujian aplikasi adalah sebagai berikut:

1. Membuka localhost:3001 pada browser.
2. Menekan salah satu tipe *widget* yang berada di bawah ”Widget Example” pada *left bar*.
3. Memastikan kebutuhan-kebutuhan berikut sudah berfungsi sesuai prosedur yang disepakati yaitu terdapat daftar *widget example* sesuai data yang telah di *hard code*.

6.2 Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Hasil evaluasi pengujian dapat dilihat pada Tabel 6.1 dan Tabel 6.2

Tabel 6.1 Tabel evaluasi pengujian aplikasi sesuai kebutuhan

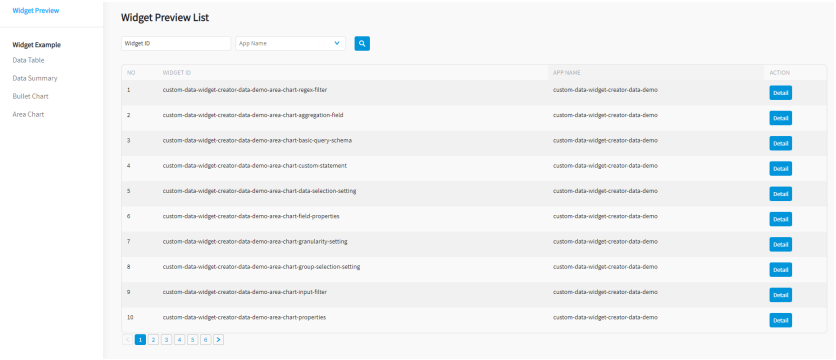
No.	Kebutuhan	Uji Coba	Status
UC001	Melihat daftar *widget* sesuai hak akses	Pengguna membuka halaman *widget list*. Pengguna dapat melihat daftar *widget* sesuai hak akses yang dimiliki oleh pengguna.	Berhasil
UC002	Membuka halaman detail *widget*	Pengguna membuka halaman *widget list*. Pengguna dapat melihat daftar *widget* sesuai hak akses yang dimiliki oleh pengguna. Pengguna menekan tombol *detail* pada salah satu *widget*. Pengguna dapat melihat halaman detail dari *widget*	Berhasil
UC003	Memodifikasi konfigurasi *widget*	Pengguna membuka halaman *widget list*. Pengguna dapat melihat daftar *widget* sesuai hak akses yang dimiliki oleh pengguna. Pengguna menekan tombol *detail* pada salah satu *widget*. Pengguna mengganti salah satu konfigurasi, misal panjang dari *widget*. Pengguna menekan tombol *apply*. Pengguna dapat melihat *widget* telah diperbarui sesuai dengan konfigurasi yang dimasukkan	Berhasil

Tabel 6.2 Tabel evaluasi pengujian aplikasi sesuai kebutuhan (lanjutan)

No.	Kebutuhan	Uji Coba	Status
UC004	Memfilter *widget* sesuai *role*	Pengguna membuka halaman *widget list*. Pengguna menekan *drop down* ”App Name”.Pengguna menekan salah satu appName pada *drop down* ”App Name”. Pengguna menekan tombol *search* yang ditandai dengan gambar kaca pembesar. Pengguna dapat melihat daftar *widget* sesuai appName yang dipilih pengguna.	Berhasil
UC005	Melakukan *Filter* Berdasarkan *Widget Id*	Pengguna membuka halaman *widget list*. Pengguna mengisi *text box* dengan *placeholder* ”WidgetId” dengan widgetId yang diinginkan. Pengguna menekan tombol *search* yang ditandai dengan gambar kaca pembesar. Pengguna dapat melihat daftar *widget* sesuai appName yang dipilih pengguna.	Berhasil
UC006	Membuka Halaman *Widget Example List*	Pengguna membuka halaman *widget list*. Pengguna menekan salah satu tipe *widget* yang berada di bawah ”Widget Example” pada *left bar*. Pengguna dapat melihat daftar daftar *widget example*.	Berhasil

6.2.1 Membuka Halaman Widget List

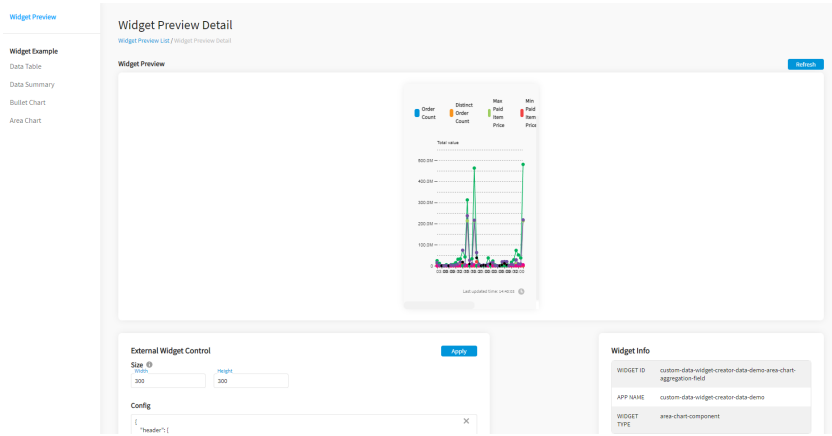
Pengujian dilakukan dengan cara membuka localhost:3001 pada browser, lalu muncullah daftar *widget* sesuai hak akses yang dimiliki oleh pengguna dengan *pagination*. Selain itu terdapat juga *text box* dengan *placeholder* "Widget Id", *dropdown* "App Name", dan tombol dengan gambar kaca pembesar untuk kebutuhan *filter* seperti pada Gambar 6.1.



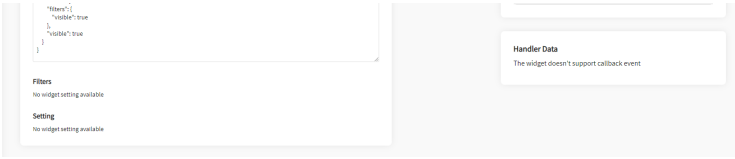
Gambar 6.1 Tampilan hasil uji coba membuka halaman *widget list*

6.2.2 Membuka Halaman Widget Detail

Pengujian dilakukan dengan cara membuka localhost:3001 pada browser, lalu muncullah daftar *widget* sesuai hak akses yang dimiliki oleh pengguna dengan *pagination* lalu menekan tombol "Detail" pada salah satu *widget*. Halaman detail dari *widget* menampilkan *widget preview*, properti dari *widget*, *handler data*, dan beberapa konfigurasi yang dapat dimodifikasi oleh pengguna seperti pada Gambar 6.2-6.3.



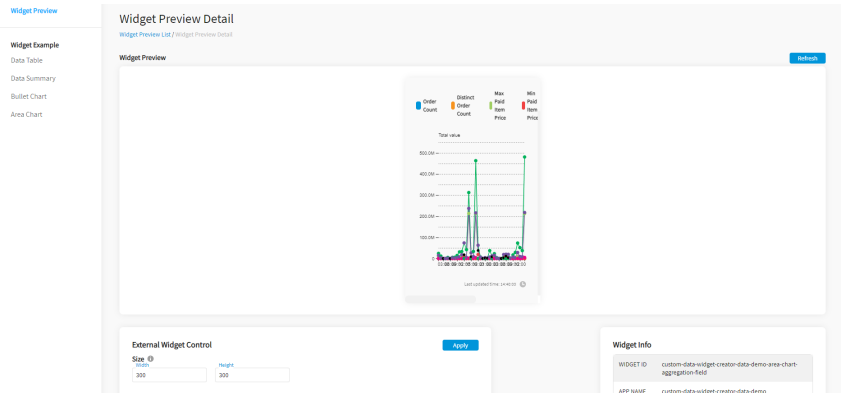
Gambar 6.2 Tampilan hasil uji coba membuka halaman *widget detail*



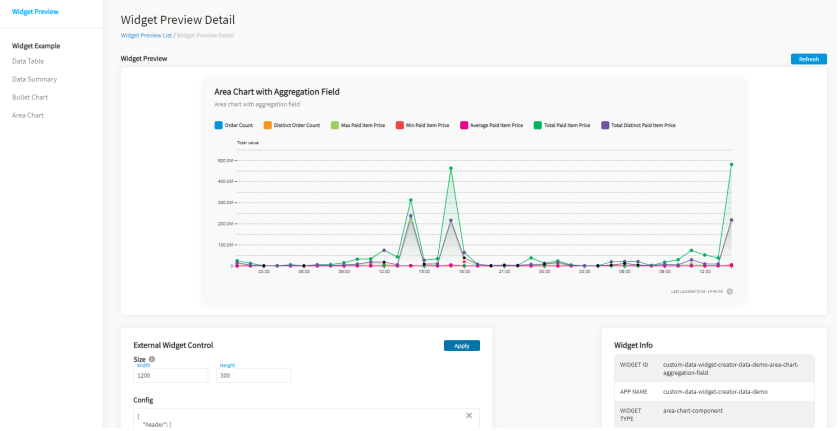
Gambar 6.3 Tampilan hasil uji coba membuka halaman *widget detail* (lanjutan)

6.2.3 Memodifikasi Konfigurasi pada Widget Detail

Pengujian dilakukan dengan cara membuka localhost:3001 pada browser, lalu muncullah daftar *widget* sesuai hak akses yang dimiliki oleh pengguna dengan *pagination* lalu menekan tombol ”Detail” pada salah satu *widget*. Lalu dapat dilakukan modifikasi, misal *width* dari *widget* diganti yang awalnya adalah 300 penjadi 1200 kemudian tekan tombol ”Apply”, maka panjang *widget* di *widget preview* akan bertambah panjang. Gambar 6.4 menunjukkan *widget* sebelum dimodifikasi dan Gambar 6.5 menunjukkan *widget* setelah dikonfigurasi.



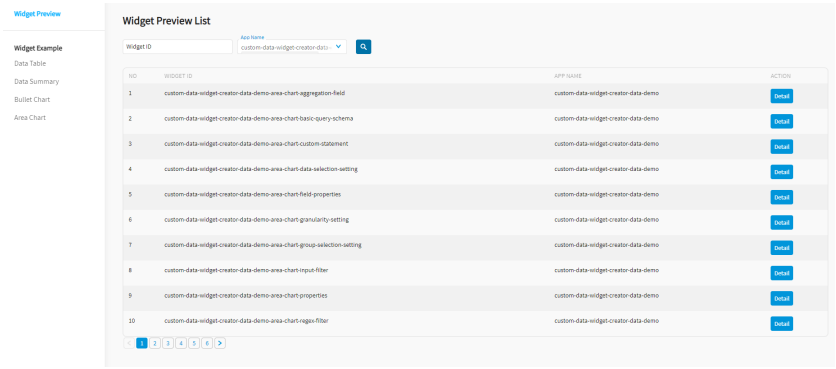
Gambar 6.4 Tampilan hasil uji coba sebelum memodifikasi konfigurasi pada detil *widget*



Gambar 6.5 Tampilan hasil uji coba setelah memodifikasi konfigurasi pada detil *widget*

6.2.4 Melakukan Filter Berdasarkan App Name

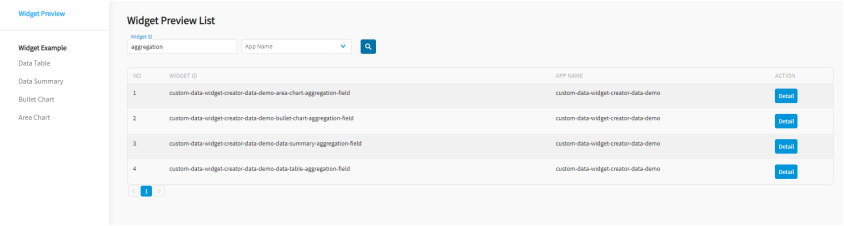
Pengujian dilakukan dengan cara membuka localhost:3001 pada browser, lalu muncullah daftar *widget* sesuai hak akses yang dimiliki oleh pengguna dengan *pagination* lalu tekan *drop down* ”App Name” dan pilih misal adalah ”custom-data-widget-creator-data-demo”. Lalu tekan tombol dengan gambar kaca pembesar, maka terlihat beberapa *widget* yang *App Name* nya adalah ”custom-data-widget-creator-data-demo” seperti pada Gambar 6.6.



Gambar 6.6 Tampilan hasil uji coba *filter* berdasarkan *App Name*

6.2.5 Melakukan Filter Berdasarkan Widget Id

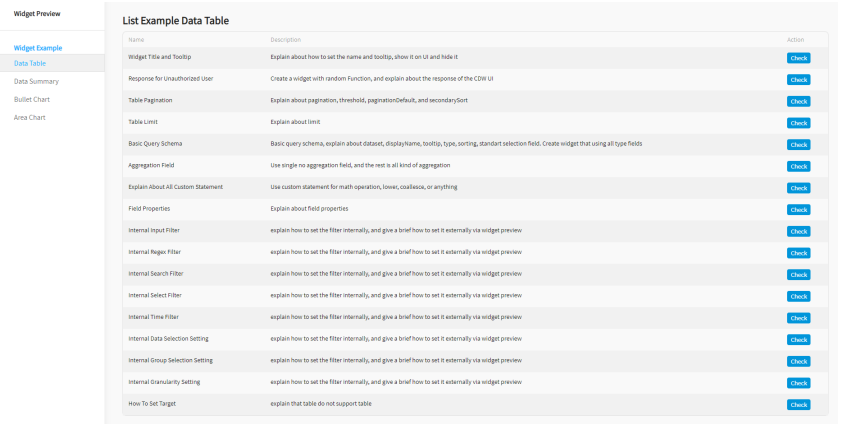
Pengujian dilakukan dengan cara membuka localhost:3001 pada browser, lalu muncullah daftar *widget* sesuai hak akses yang dimiliki oleh pengguna dengan *pagination* lalu ketikkan kata pada *text box* ”WidgetId” misal ”aggregation”. Lalu tekan tombol dengan gambar kaca pembesar, maka terlihat beberapa *widget* yang widgetId nya mengandung kata ”aggregation” seperti pada Gambar 6.7.



Gambar 6.7 Tampilan hasil uji coba *filter* berdasarkan widgetId

6.2.6 Membuka Halaman Widget Example List

Pengujian dilakukan dengan cara membuka localhost:3001 pada browser, lalu tekan salah satu tipe *widget* pada *left bar* misal ”Data Table”, maka akan muncul daftar *widget example* yang telah dibuat seperti pada Gambar 6.8.



Gambar 6.8 Tampilan hasil uji coba membuka halaman *widget example*

BAB VII
KESIMPULAN DAN SARAN

Setelah melakukan pengujian, didapatkan beberapa kesimpulan dan saran yang didapatkan untuk *Pengembangan Fitur Custom Data Widget Center pada Aplikasi Internal Blibli.com*.

7.1 Kesimpulan

1. *Custom Data Widget Center* dapat digunakan sebagai dokumentasi untuk mempelajari cara penggunaan *library Custom Data Widget* yang merupakan *library* yang disediakan Blibli.com untuk kebutuhan visualisasi data.
2. *Custom Data Widget Center* dapat digunakan untuk melakukan pengecekkan dan modifikasi terhadap *widget* yang telah dibuat oleh pengguna.
3. Fitur *Custom Data Widget Center* yang diharapkan dapat menjadi tempat untuk mempelajari dan mengeksplorasi fitur yang ditawarkan oleh *library Custom Data Widget* telah berhasil diimplementasikan dan dapat berjalan sesuai yang diharapkan.

7.2 Saran

1. Perlu dilakukan optimasi pada pengkonversian hasil kueri menjadi *response* yang memenuhi standar blibli.com.
2. Daftar *widget example* dapat dikembangkan dari *hard coded* menjadi sebuah API, sehingga lebih fleksibel.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] Blibli. (2011). “Tentang Blibli - Pusat bantuan, ”**url**: https://www.blibli.com/faq/tentang-blibli/tentang-blibli-com/.

[2] MDN. (2021). “What is a web server?, ”**url**: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server.

[3] MongoDB. (2021). “MongoDB, ”**url**: https://www.mongodb.com/.

[4] ——, (2021). “What is a NoSQL?, ”**url**: https://www.mongodb.com/nosql-explained.

[5] REST. (2021). “What is REST, ”**url**: https://restfulapi.net/.

[6] spring. (2021). “Why Spring?, ”**url**: https://spring.io/why-spring.

[7] Wikipedia. (2020). “Aplikasi web, ”**url**: https://id.wikipedia.org/wiki/Aplikasi_web.

[8] ——, (2021). “Intellij IDEA, ”**url**: https://en.wikipedia.org/wiki/IntelliJ_IDEA.

[9] ——, (2021). “Java (programming language), ”**url**: https://en.wikipedia.org/wiki/Java_(programming_language).

[10] ——, (2021). “Software widget, ”**url**: https://en.wikipedia.org/wiki/Software_widget.

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Bryan Gautama Ngo, lahir di Balikpapan tanggal 26 Januari 2001. Penulis merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh pendidikan formal di SD Latihan YBBSU Balikpapan (2007-2013), SMP Negeri 1 Balikpapan (2013-2015), SMA Negeri 1 Balikpapan (2015-2018). Penulis melanjutkan studi dengan berkuliah pada program sarjana (S-1) di Departemen Teknik Informatika ITS. Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Algoritma dan Pemrograman (AP). Selama menempuh perkuliahan, penulis mendapatkan beasiswa dari Beasiswa Unggulan yang diadakan oleh Kementrian Pendidikan dan Kebudayaan. Selain itu, penulis pernah menjadi asisten dosen pada mata kuliah Dasar Pemrograman. Penulis juga memiliki ketertarikan pada pemrograman perangkat bergerak yang mendorong penulis mengikuti dan menjadi finalis pada kompetisi pengembangan aplikasi perangkat bergerak Hology 2.0. Penulis juga aktif mengikuti organisasi kemahasiswaan, yaitu sebagai staf ahli Departemen Media dan Informasi di Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS dan staf ahli Departemen Media dan Informasi di Tim Pembina Kerohanian Buddha. Selain aktif mengikuti organisasi, penulis juga aktif dalam kegiatan kepanitiaan Schematics, yaitu sebagai staf NLC pada tahun 2019. Saat ini penulis sedang mengikuti program Future Progam Batch 5.0 di blibli.com. Penulis dapat dihubungi melalui surel di gautamabryan@gmail.com.