

Slack Form App

Slack Form App adalah aplikasi **Slack™** untuk keperluan pengisian formulir digital melalui platform tersebut. **Slack Form App** mengurus mekanisme mulai dari pengisian formulir oleh pengguna sampai dengan penyimpanan data pengisian formulir. Semua hal tersebut dapat berjalan hanya dengan memasukkan perintah `/form` pada kolom *chat* di sebuah *channel*. Dokumen ini menjelaskan lebih lanjut detail spesifik terhadap implementasi, pengembangan, dan hal lainnya yang terkait dengan **Slack Form App** mengikuti urutan judul berikut:

1. [Pengunduhan dan Instalasi](#)

Membahas tentang cara mengunduh dan menginstalasi **Slack Form App** melalui *container Docker*.

2. [Integrasi API Slack](#)

Membahas tentang cara membuat dan mengkonfigurasi **aplikasi Slack**.

3. [Implementasi Sisi Pengguna](#)

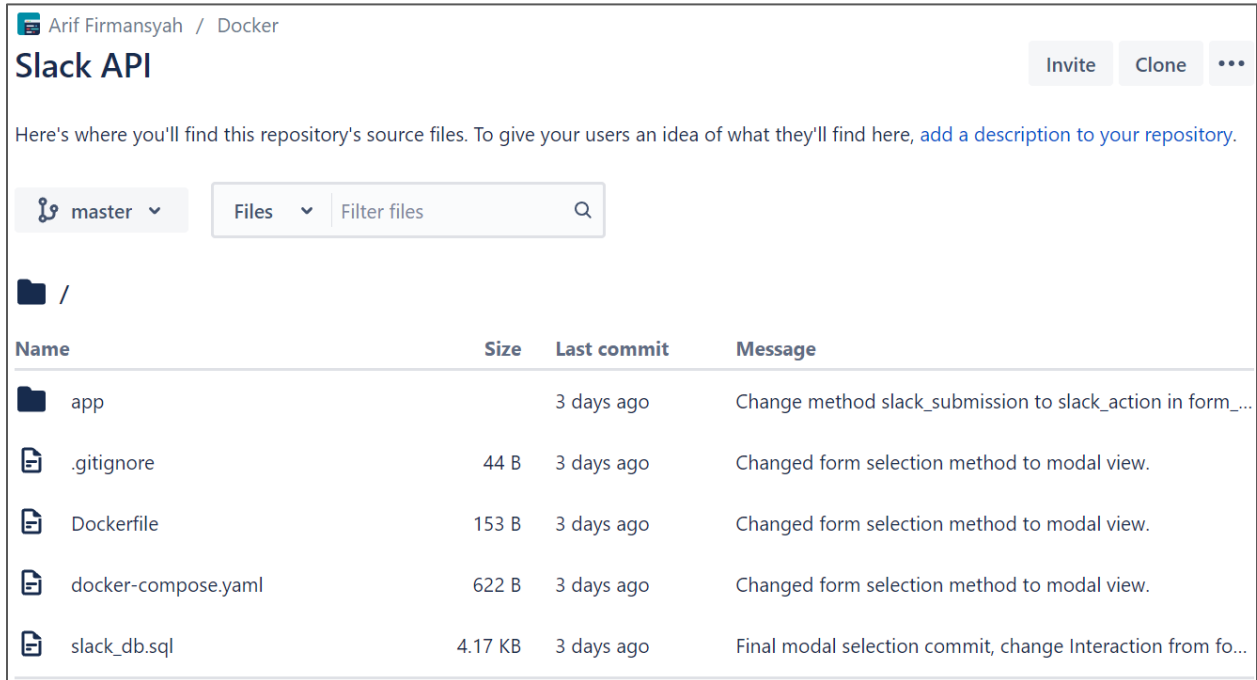
Membahas alur penggunaan **Slack Form App** dan proses yang terjadi dari sisi **backend**.

4. [Pengaksesan Database](#)

Membahas cara pengaksesan *database* dan interaksinya, seperti penambahan *template* dan membaca data yang terkumpul.

1. Pengunduhan dan Instalasi

Slack Form App dapat berjalan pada *operating system* manapun dengan menggunakan *container docker*. *Script*, *dump*, dan *file-file* terkait lainnya tersimpan bersama *file Dockerfile* dan *file docker-compose.yaml* pada repositori [BitBucket](#).



Arif Firmansyah / Docker

Slack API

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository](#).

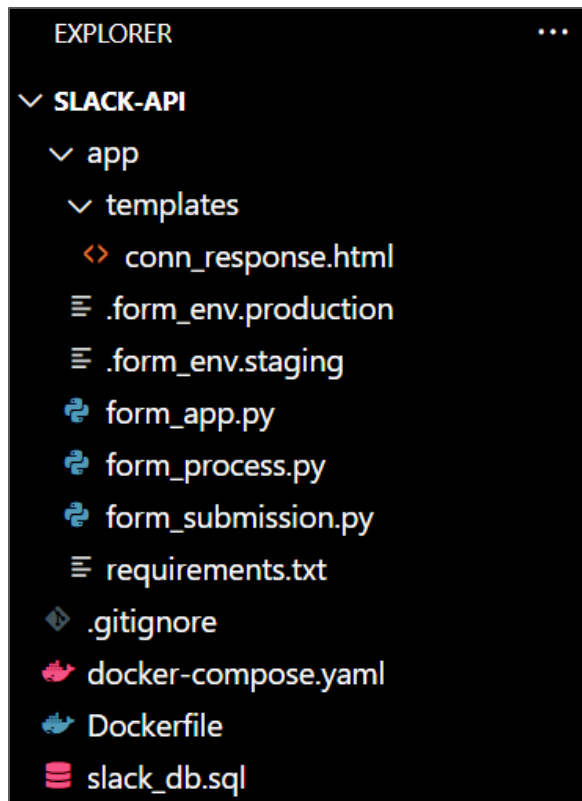
master Files Filter files

Name	Size	Last commit	Message
app		3 days ago	Change method slack_submission to slack_action in form_...
.gitignore	44 B	3 days ago	Changed form selection method to modal view.
Dockerfile	153 B	3 days ago	Changed form selection method to modal view.
docker-compose.yaml	622 B	3 days ago	Changed form selection method to modal view.
slack_db.sql	4.17 KB	3 days ago	Final modal selection commit, change Interaction from fo...

Salin repositori tersebut ke komputer Anda dengan menggunakan perintah **git clone**:

```
git clone git@bitbucket.org:arif_firmansyah_biznet/slack-api.git
```

Buka *file manager* atau **Visual Studio Code** dan akses *folder* salinan dari **BitBucket** tersebut.



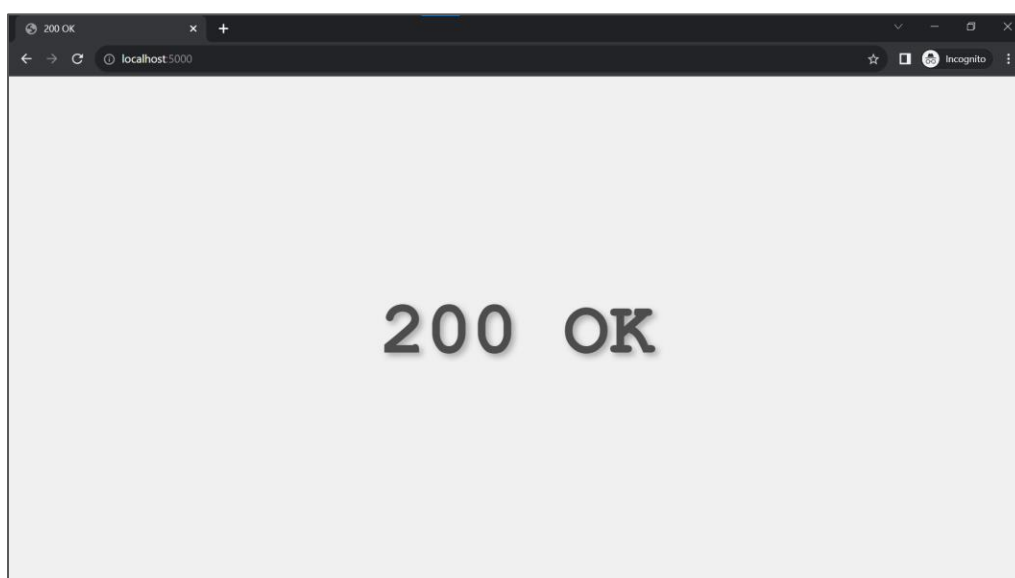
Direktori salinan akan berisi *file-file* yang dibutuhkan untuk membuat *container* **Slack Form App** menggunakan perintah *command line* **Docker**. Untuk itu, mohon pastikan **Docker** telah [terinstalasi](#) dan siap digunakan pada komputer Anda agar **Slack Form App** dapat berjalan tanpa memerlukan instalasi dependensi lainnya.



Pembuatan *container* **Slack Form App** dapat dibuat cukup dengan menggunakan perintah:

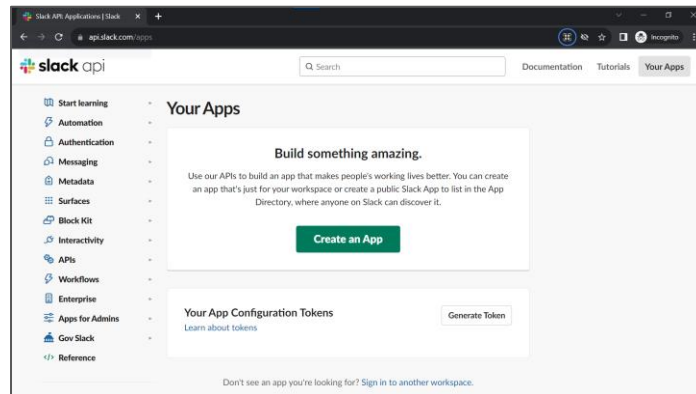
```
docker-compose up -d
```

agar pembuatan *container* terlepas dari *terminal* atau CLI yang digunakan. Buka *browser* dan ketik **localhost:5000** untuk memeriksa jika *container* telah berhasil berjalan dengan menampilkan laman respon berikut:



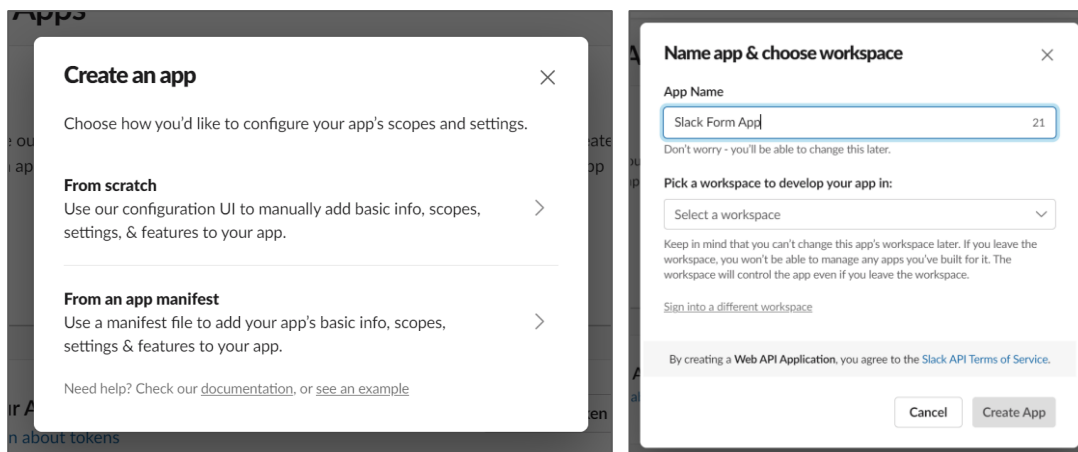
2. Integrasi API Slack

Slack Form App memanfaatkan *Application Programming Interface* (API) milik **Slack™** untuk dapat memakai fitur-fitur seperti mengirim pesan, mengambil data, dan sebagainya. Pengaksesan API memerlukan adanya pembuatan **aplikasi Slack** melalui [website API Slack™](https://api.slack.com/apps).



a. Pembuatan Aplikasi Slack

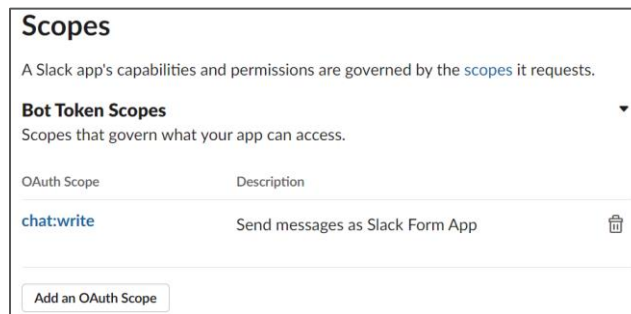
Buat **aplikasi Slack** baru dengan klik tombol “*Create and App*” dan pilih “*From scratch*”. Isi nama aplikasi dan pilih *workspace* tempat aplikasi berjalan, lalu klik tombol “*Create App*”.



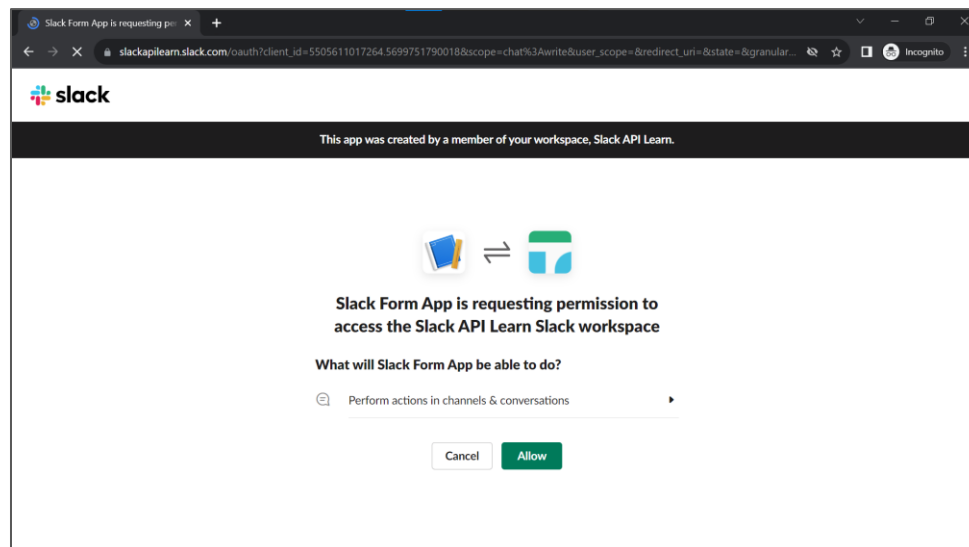
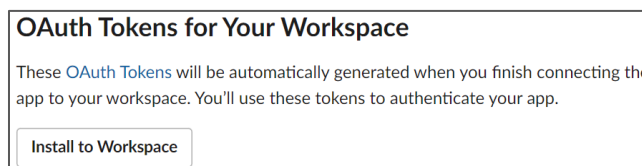
b. Penambahan Scope

Setelah **aplikasi Slack** berhasil dibuat, cari menu “*OAuth & Permissions*” pada *sidebar* di sebelah kiri laman. Cari bagian **scope** dan klik tombol “*Add an OAuth Scope*” untuk menambahkan fungsi **aplikasi Slack**. **Scope** adalah jangkauan akses API dari sebuah

aplikasi Slack. Untuk itu, ketik “*chat:write*” pada kolom pencarian agar aplikasi dapat menulis pesan pada *chat* di mana **aplikasi Slack** tersebut ada.



Jangkauan akses API yang berlaku untuk sebuah **aplikasi Slack** hanya mengikuti **scope** yang telah ditambahkan. Sebuah **aplikasi Slack** juga boleh memiliki semua **scope**. Namun, penambahan **scope** apapun perlu diinstalasikan ke **aplikasi Slack** pada *workspace*. Untuk melakukan instalasi, klik tombol “*Install to Workspace*” di bagian “*OAuth Token for Your Workspace*” dan klik “*Allow*”.



c. Bot Token & Signing Secret

Jika baru pertama kali, aksi instalasi **aplikasi Slack** ke *workspace* akan menghasilkan **bot token**. **Bot token** adalah kode unik yang perlu ada sebagai nilai parameter ketika akan mengakses *method-method* API milik **Slack™**. **Bot token** adalah kode yang memungkinkan

workspace agar dapat mengenal **aplikasi Slack**. Kode ini tidak untuk disebarluaskan karena akan beresiko terhadap penyalahgunaan aplikasi.

OAuth Tokens for Your Workspace

These tokens were automatically generated when you installed the app to your team. You can use these to authenticate your app. [Learn more.](#)

Bot User OAuth Token

xoxb-

Copy

Access Level: Workspace

Reinstall to Workspace

Selain **bot token**, kode yang sama pentingnya adalah **signing secret**. Kode ini berfungsi untuk memverifikasi tiap permintaan yang datang dari **Slack™**. Sifat dari kode ini juga tidak untuk disebarluaskan. Untuk meningkatkan keamanan, **signing secret** juga dapat diperbarui dengan klik tombol “*Regenerate*”. Pengaksesan kode **signing secret** dapat melalui menu “*Basic Information*” yang terletak pada *sidebar* di sebelah kiri layar.

Signing Secret

.....

ShowRegenerate

Slack signs the requests we send you using this secret. Confirm that each request comes from Slack by verifying its unique signature.

Slack Form App telah terintegrasi dengan *framework* **Slack Bolt** yang mengurus *response* terhadap *request* yang berasal dari *workspace*. Kedua nilai **bot token** dan **signing secret** menjadi nilai penting pada konfigurasi **Slack Bolt**. Pada *file* **form_app.py** yang berfungsi sebagai **controller**, **Slack Bolt** dikonfigurasi pada bagian **App Initialization**.

```
form_app.py
15  ...
16
17  #  =====
18  #  || Config Loading      ||
19  #  =====
20  config = dotenv_values(".form_env")
21
22
23
24  #  =====
25  #  || App Initialization ||
26  #  =====
27
28  current_form = None
29  app = Flask(__name__)
30  slack = App(
```

31	token = config["SLACK_BOT_TOKEN"],
32	signing_secret = config["SLACK_SIGNING_SECRET"]
33)
34	slack_handler = SlackRequestHandler(slack)
35	
36	...

Nilai **bot token** dan **signing secret** pada *file* tersebut tersimpan di dalam **environment variable**. Nilai-nilai tersebut berada di *file* konfigurasi **.form_env** yakni *file* yang menyimpan nilai-nilai konfigurasi aplikasi dan menyimpannya ke dalam **environment variable**. Nilai **bot token** dan **signing secret** yang telah didapatkan perlu dimasukkan ke dalam *file* ini agar **Slack Bolt** dapat terkonfigurasi dengan benar. Jangan masukkan kedua nilai tersebut langsung ke dalam *file* **form_app.py** untuk menjaga integritas data dan mempermudah proses pengembangan aplikasi.

.form_env
<pre>#Ganti nilai dengan SLACK_BOT_TOKEN dengan nilai bot token SLACK_BOT_TOKEN=xoxb-xxxxxxx-xxxxxxx-xxxxxxx #Ganti nilai dengan SLACK_SIGNING_SECRET dengan nilai signing secret SLACK_SIGNING_SECRET=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</pre>

File **.form_env** juga dapat diganti dengan *file* konfigurasi sejenis. Pastikan nilai argumen pada fungsi **dotenv_values()** pada *file* **form_app.py** adalah *path* dari *file* konfigurasi tersebut.

d. Endpoint

Slack Form App dapat menjalankan fungsi-fungsinya setelah mengirim *response* terhadap *request* dari *workspace*. **Slack™** mengirimkan *request* dari *workspace* ke **aplikasi Slack** melalui **endpoint**. **Endpoint** adalah *path* yang dispesifikkan pada *setting* di [website API Slack™](#) dan **route** yang terdapat di *file* **form_app.py**.

form_app.py
<pre>36 ... 37 38 # ===== 39 # Route Section 40 # ===== 41 42 # -- Connection check. -- 43 @app.route('/', methods = ['GET'])</pre>

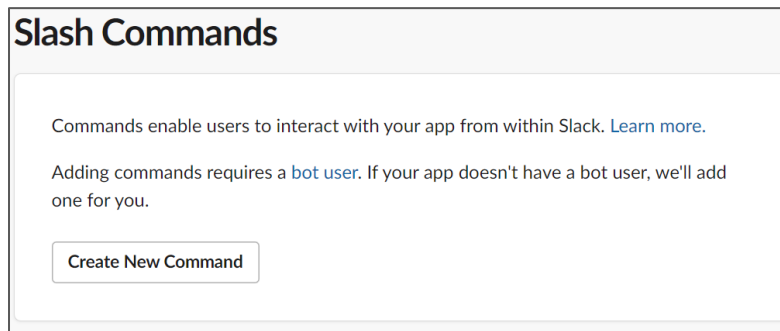
```

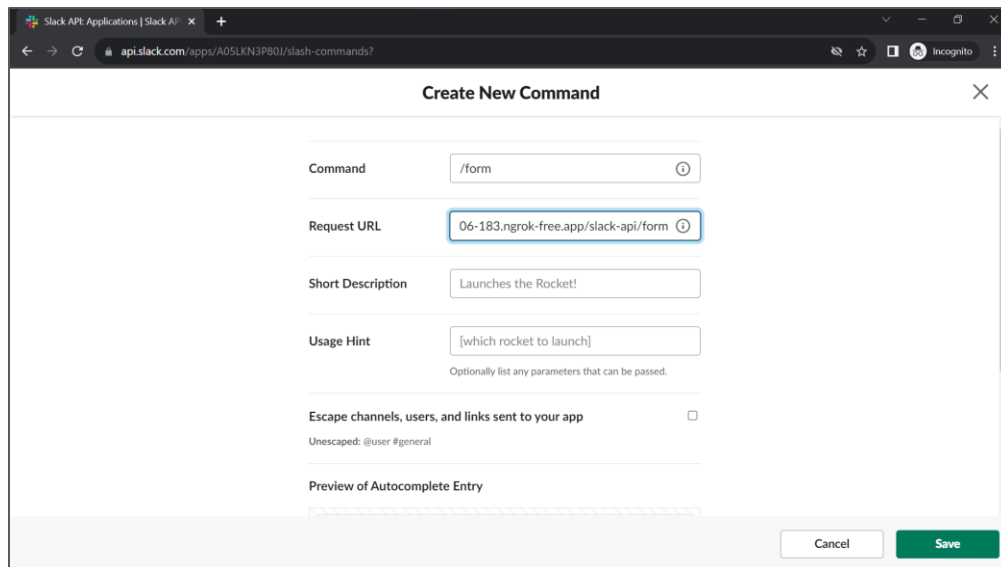
44 def slash_root():
45     return render_template("conn_response.html")
46
47 # -- Form view call route. --
48 @app.route('/slack-api/form', methods = ['POST'])
49 def slack_form():
50     response = slack_handler.handle(request)
51     return response
52
53 # -- Form submit post route --
54 @app.route('/slack-api/action', methods = ['POST'])
55 def slack_action():
56     response = slack_handler.handle(request)
57     return response
58
59 ...

```

Slack Form App hanya memanfaatkan dua **endpoint**, yakni untuk perintah `/form` dan untuk aksi interaktif, seperti tombol “*submit*” pada formulir dan pemilihan *template* formulir. **Endpoint** membutuhkan adanya **alamat IP publik** sebagai alamat penerima *request* dari *workspace*. Salah satu *tools* yang dapat dipakai sebagai sarana **alamat IP publik** adalah [Ngrok](#).

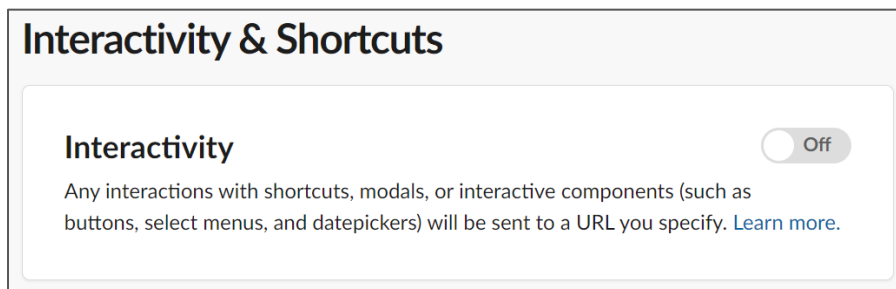
Penambahan **endpoint** untuk perintah `/form`, akses menu “*Slash Commands*” dilakukan melalui *sidebar* di sebelah kiri layar. **Slash command** adalah perintah yang diberikan melalui *chat* pada sebuah *channel* di mana penulisannya diawali dengan garis miring (/). `/form` merupakan salah satu contoh dari *slash command*, yang akan menjalankan fungsi dari **Slack Form App**. Untuk membuat perintah `/form`, klik tombol “*Create New Command*”, lalu isi nama **slash command** (dalam hal ini `/form`) pada kolom “*Command*” dan nilai **endpoint** pada kolom “*Request URL*”. Selain itu, tambahkan deskripsi singkat dari **slash command** tersebut pada kolom “*Short Description*”.



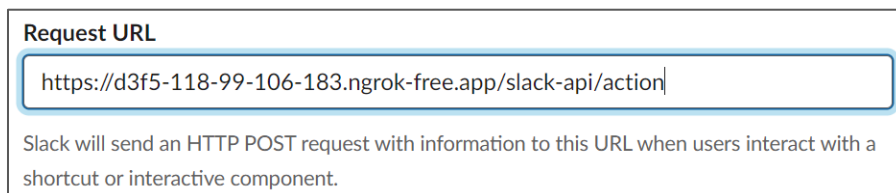


Bentuk nilai **endpoint** nilai **Alamat IP publik** ditambah dengan nilai *path* pada **route** di *file form_app.py*. Bentuk keseluruhannya dapat dilihat pada foto di atas. Setelah nama **slash command** dan **endpoint** telah terisi, klik tombol “Save” untuk menyimpan **slash command** tersebut. Lanjutkan dengan melakukan instalasi ulang **aplikasi Slack** ke *workspace* agar **slash command** dapat tersedia. Penambahan **slash command** lainnya dapat dilakukan dengan cara yang sama, namun nilai **endpoint** perlu berbeda antar **slash command** dan ditambahkan sebagai **route** baru pada *file form_app.py*.

Penambahan **endpoint** untuk aksi-aksi seperti tombol pada formulir dilakukan melalui menu “*Interactivity & Shortcuts*”, dapat diakses melalui *sidebar* di sebelah kiri layar. Nyalakan fitur **Interactivity** terlebih dahulu agar dapat diintegrasikan ke **Slack Form App**.



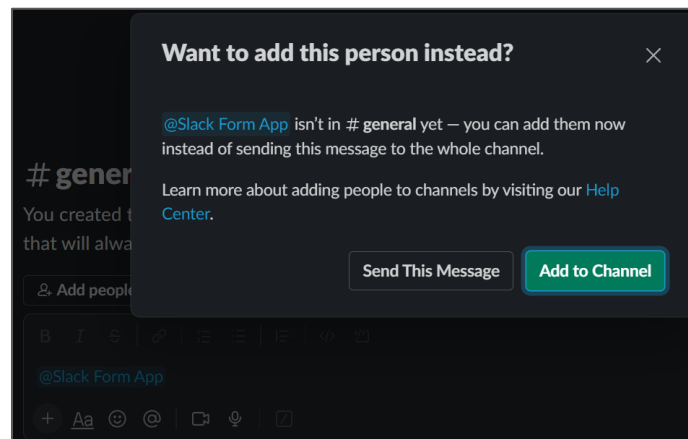
Setelah **Interactivity** dinyalakan masukkan **endpoint** dengan bentuk yang sama seperti pada **slash command**. Bentuk keseluruhannya dapat dilihat pada foto berikut:



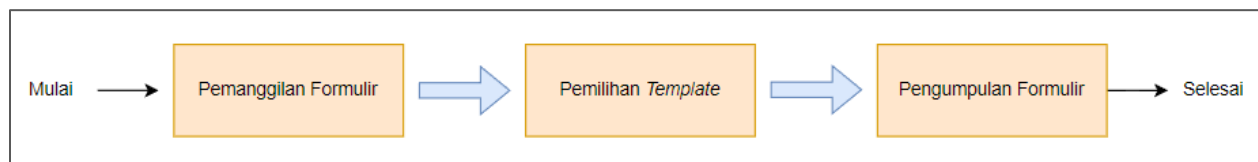
Simpan **endpoint** dengan melakukan klik tombol “*Save Changes*” dan **endpoint** akan secara otomatis terintegrasi ke **Slack Form App**.

3. Implementasi Sisi Pengguna

Slack Form App hadir dalam *workspace* sebagai **aplikasi Slack**. **Slack Form App** perlu diundang terlebih dahulu ke *channel* yang ingin dituju sehingga pengguna dapat berinteraksi dengan **Slack Form App**. Untuk mengundang **Slack Form App** ke sebuah *channel*, cukup *mention* **Slack Form App** di kolom *chat* dan klik tombol “Add to Channel”.



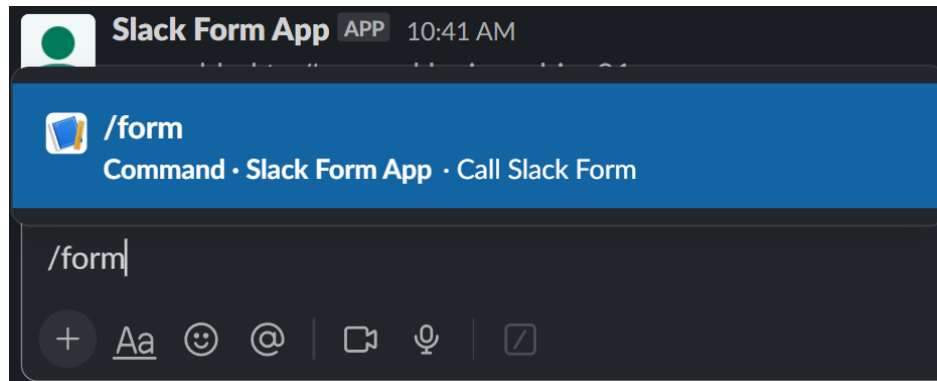
Setelah ini, segala **slash command** yang ada pada **Slack Form App** sudah bisa digunakan. Keseluruhan fungsi **Slack Form App** sudah bisa bekerja. Alur penggunaan **Slack Form App** dalam suatu *workspace* dapat dirangkum seperti yang tergambar pada diagram berikut:



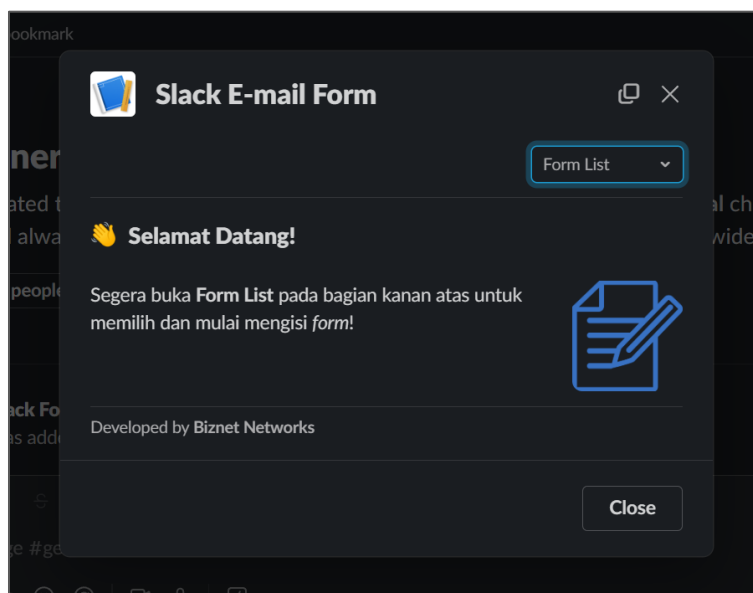
Terdapat tiga interaksi pengguna/aksi utama yakni **pemanggilan formulir**, **pemilihan *template***, dan **pengumpulan formulir**. Ketiga aksi ini berinteraksi dengan **controller *form_app.py*** untuk mengambil *template* formulir atau mengumpulkan data. Aksi-aksi lainnya merupakan bagian dari salah satu dari ketiga aksi tersebut.

a. Pemanggilan Formulir

Pengguna langsung memanggil formulir dengan mengirim perintah `/form` pada kolom *chat*. Dengan begitu, *workspace* akan mengirimkan **request** ke **Slack Form App**, spesifiknya ke **endpoint** dengan *path* `/slack-api/form` seperti yang telah di bahas sebelumnya di bagian **Integrasi API Slack**.



Slack Form App akan merespon dengan memunculkan laman depan formulir. Detail proses pemanggilan formulir dapat diperiksa pada file **form_app.py**, pada bagian **Event Section**. Di bagian tersebut, terdapat satu fungsi yang bertugas untuk memunculkan laman depan. Fungsi tersebut memakai fungsi API milik **Slack™** yakni [client.views.open\(\)](#) untuk menampilkan **view** bertipe **modal**. Selengkapnya dapat dilihat di dokumentasi **Slack™** tentang [modal view](#).



form_app.py	
59	...
60	
61	# =====
62	# Event Section
63	# =====
64	
65	# -- Form Initialize Event --
66	@slack.command("/form")
67	def init(ack, body, client):
68	ack()

69	
70	global current_form
71	current_form = FormProcess()
72	client.views_open(
73	trigger_id = body['trigger_id'],
74	view = current_form.create_form('init')
75)
76	
77	...

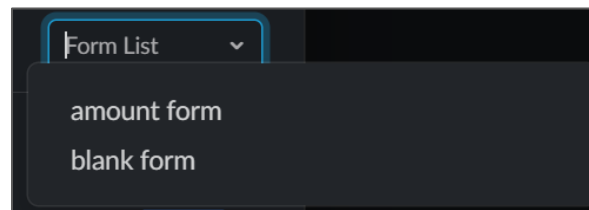
Fungsi tersebut terhubung dengan *file* **form_template.py** yang merupakan model yang terhubung ke **database**. Model ini terhubung ke tabel **form_template** yakni tabel yang menyimpan data format *template*. Dengan memanggil fungsi **create_form()** dan memasukkan nilai argumen “*init*”, *template* laman depan formulir akan dikembalikan dari **database**.

form_template.py	
80	...
81	
82	def create_form(self, command):
83	view = copy.deepcopy(self.template)
84	
85	# -- Open Database Connection. --
86	conn = mysql.connect()
87	cursor = conn.cursor()
88	
89	# -- Execute SQL Commands --
90	# cursor = mysql.connection.cursor()
91	query = f"SELECT type, template FROM form_templates
92	WHERE type = '{command}';"
93	cursor.execute(query)
94	data_row = cursor.fetchone()
95	
96	# -- Close Database Connection. --
97	cursor.close()
98	conn.close()
99	
100	self.type = data_row[0]
101	
102	if command != "init":
103	view["submit"] = {
104	"type": "plain_text",
105	"text": "Submit Form",
106	"emoji": True
107	}
108	view["close"] = {
	"type": "plain_text",

109	<code>"text": "Cancel",</code>
110	<code>"emoji": True</code>
111	<code>}</code>
112	<code>for block in json.loads(data_row[1]):</code>
113	<code>view['blocks'].append(block)</code>
114	<code>return view</code>

b. Pemilihan *Template*

Laman depan formulir berisi instruksi singkat untuk memilih *template* formulir. Pemilihan *template* dilakukan dengan membuka menu *dropdown* pada bagian kanan atas. Pengguna juga dapat mengetik langsung nama *template* pada kolom pencarian di menu *dropdown* tersebut.



Pilihan pada menu *dropdown* tersebut tersambung dengan *template* formulir yang ada pada **database**. Dengan melakukan klik terhadap salah satu pilihan, laman depan akan secara otomatis berubah ke formulir yang diinginkan. Contohnya, jika pengguna memilih “*amount form*”, laman formulir sebelumnya akan berganti ke formulir “*amount form*”.

 A screenshot of a web application interface titled 'Slack E-mail Form'. The interface has a dark theme. At the top right, there is a dropdown menu with 'amount form' selected. Below the header, there are three input fields: 'Registration ID' with the value 'ab...', 'Amount' with the value '123...789', and 'Description (optional)' with the placeholder text 'Fill your description...'. At the bottom right, there are two buttons: 'Cancel' and 'Submit Form'.

Menu *dropdown* akan tetap ada meskipun formulir berganti sehingga pengguna bebas mengganti formulir selama **Slack Form App** masih terbuka. Pergantian formulir dan pemilihan *template* ditangani oleh file **form_app.py**. Mekanismenya mirip dengan pada

tahap **Pemanggilan Formulir**. *Workspace* mengirimkan **request** ke **Slack Form App** melalui **endpoint** dengan *path* `/slack-api/action`, berbeda dengan **endpoint** milik `/form` yang merupakan **slash command**. **Slack Form App** membalas **request** tersebut dengan **response** memakai fungsi `client.views_update()` untuk mengganti tampilan formulir. Tampilan juga didapatkan dari fungsi `create_form()` pada file `form_template.py`, namun bedanya, nilai argumen adalah nama dari formulir, sesuai dengan nama-nama yang muncul pada menu *dropdown*.

```
form_app.py
59  ...
60
61  #  =====
62  #  || Event Section      ||
63  #  =====
64
65  ...
66
67
68  #  -- Form Select Event --
69  @slack.action("type_list")
70  def select_form(ack, body, client):
71      ack()
72
73      selected =
74      f"{body['actions'][0]['selected_option']['value']}"
75      client.views_update(
76          view_id = body["view"]["id"],
77          hash = body["view"]["hash"],
78          view = current_form.create_form(selected)
79      )
80
81  ...
82
83  ...
84
85  ...
86
87  ...
88
89  ...
90  ...
```

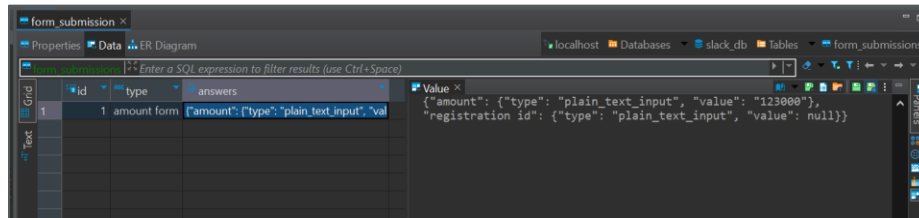
c. Pengumpulan Formulir

Pengguna dapat mengumpulkan formulir hanya dengan klik tombol “*submit*” pada bagian kanan bawah. Setelah tombol “*submit*” diklik, formulir akan tertutup dan isi formulir akan dikirim ke **Slack Form App**.



Workspace akan mengirim isi formulir dari pengguna ke **Slack Form App** sebagai bagian dari *payload* dalam **request body**. File `form-app.py` akan mengambil nilai-nilai isi formulir

tersebut dan mengirimkannya *file* **form_submission.py** yang juga merupakan **model**. *File* tersebut berfungsi memasukkan nilai isi formulir melalui fungsi **submit_form()**.



form_app.py

```

59  ...
60
61  #  =====
62  #  || Event Section      ||
63  #  =====
64
65  ...
66
67  #  -- Form Submit Event --
68  @slack.view("slack_form")
69  def submit_form(ack, view, client):
70      ack()
71
72      global current_form
73      answers = {}
74      input_blocks = view['state']['values']
75
76      for block_id, value_id in input_blocks.items():
77          if block_id == "type_menu":
78              continue
79          answers[block_id] = value_id['input']
80      print(answers)
81      results = FormSubmission(current_form, answers)
82      results.submit_form()
83      current_form = None
84
85  ...

```

form_submission.py

```

13  ...
14
15  def submit_form(self):
16      #  Format Results To JSON
17      answers_json = json.dumps(self.answers)
18      answers_sql_json = json.dumps(answers_json)

```


19	
20	# -- Open Database Connection. --
21	conn = mysql.connect()
22	cursor = conn.cursor()
23	
24	# -- Execute SQL Commands --
25	# cursor = mysql.connection.cursor()
26	query = f"INSERT INTO form_submission (type,
	answers) VALUES ('{self.form.type}', {answers_sql_json});"
27	cursor.execute(query)
28	conn.commit()
29	
30	# -- Close Database Connection. --
31	cursor.close()
32	conn.close()

4. Pengaksesan Database

a. Membuat Koneksi Database

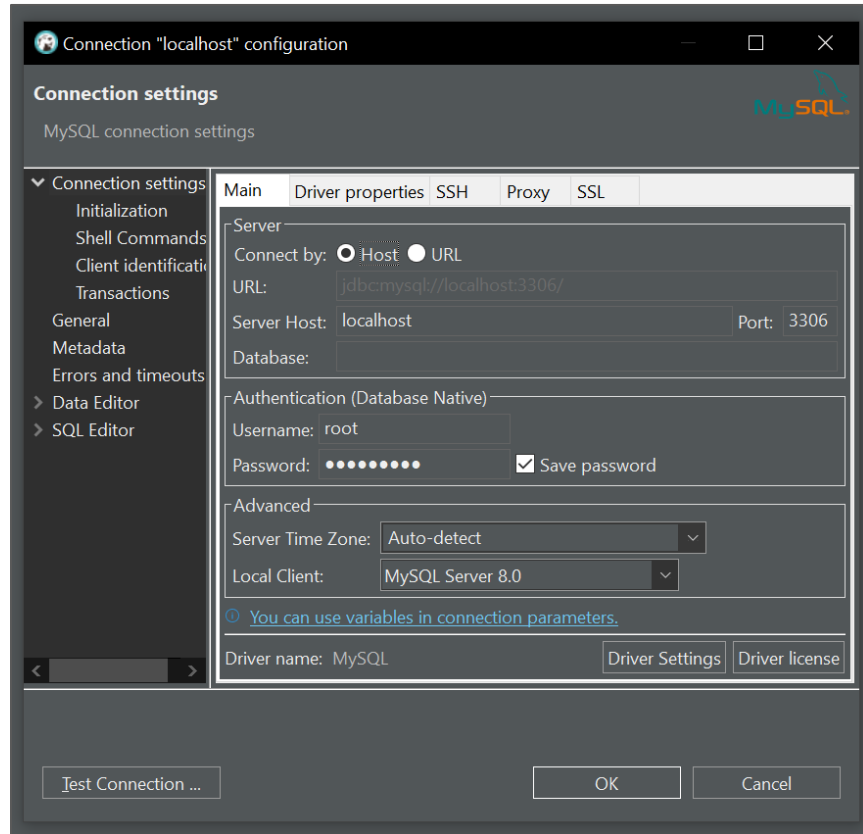
Slack Form App menggunakan **MySQL** sebagai solusi manajemen **database**. Oleh karena itu, *file* **docker-compose.yaml** juga memiliki **service** bernama **database** yang merupakan sebuah **image** dari **MySQL**, yang akan berjalan bersama dengan **Slack Form App**. Pengaksesan **database** oleh **Slack Form App** terdapat pada **model**. Kedua *file* **form_template.py** dan **form_submission.py** memiliki bagian konfigurasi agar dapat tersambung ke **database** bernama **slack_db**.

form_submission.py	
7	...
8	
9	app.config['MYSQL_DATABASE_HOST'] = config['MYSQL_LOCAL_HOST']
10	app.config['MYSQL_DATABASE_USER'] = config['MYSQL_LOCAL_USER']
11	app.config['MYSQL_DATABASE_DB'] = config['MYSQL_LOCAL_DB']
12	app.config['MYSQL_DATABASE_PASSWORD'] = config['MYSQL_LOCAL_PASSWORD']
13	mysql = MySQL(app)
14	
15	...

Nilai-nilai yang dibutuhkan untuk konfigurasi tersimpan di dalam *file* konfigurasi **.form_env** sebagai **environment variable** bersama dengan nilai **bot token** dan **signing secret**.

.form_env
#Ganti nilai dengan SLACK_BOT_TOKEN dengan nilai bot token SLACK_BOT_TOKEN=xoxb-xxxxxxx-xxxxx-xxxxxxxxxxxxxxxx- xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#Ganti nilai dengan SLACK_SIGNING_SECRET dengan nilai signing secret SLACK_SIGNING_SECRET=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#Variabel konfigurasi database slack_db. MYSQL_LOCAL_HOST=database MYSQL_LOCAL_USER=root MYSQL_LOCAL_DB=slack_db MYSQL_LOCAL_PASSWORD=mysqlroot

Pengaksesan **database** secara langsung disarankan menggunakan *tools GUI* seperti **MySQL Workbench** atau **DBeaver**. Bangun koneksi ke **database** menggunakan nilai-nilai konfigurasi seperti pada *file .form_env*. Pengaksesan **database** juga dapat melalui **script** atau program *custom* yang bisa tersambung dengan **MySQL** pada **port 3306**.

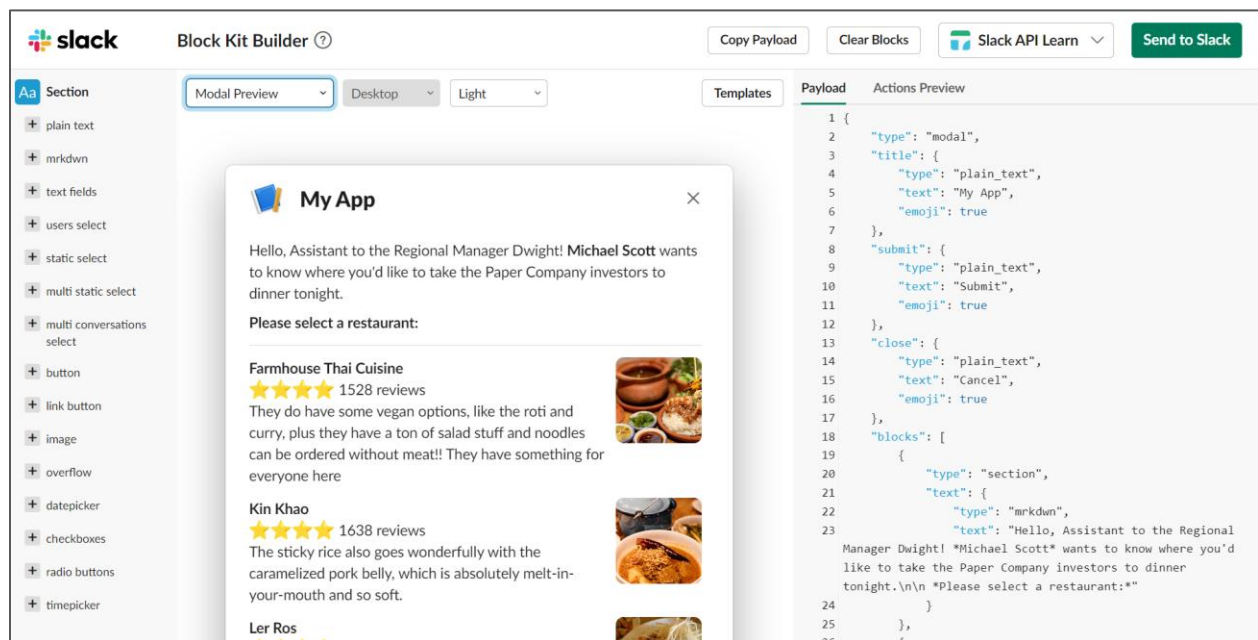


b. Menambah *Template*

Semua *template* formulir berada di tabel **form_templates**. Tabel **form_templates** terdiri dari dua kolom, yaitu kolom **type** dan kolom **template**. Kolom **type** menyimpan tipe/nama *template* yang juga akan ditampilkan pada menu *dropdown* untuk memilih formulir. Di sampingnya, kolom **template** menyimpan data berformat **JSON** yang merupakan *template* dari formulir.

*type	template	Value
1 amount form	[{"type": "input", "label": {"text": "Registration ID", "type": "plain_text", "emoji": false}, "element": {"type": "plain_text_input", "action_id": "input", "placeholder": {"text": "ab...", "type": "plain_text"}}, {"type": "input", "label": {"text": "Amount", "type": "plain_text", "emoji": false}, "element": {"type": "plain_text_input", "action_id": "input", "placeholder": {"text": "123...789", "type": "plain_text"}}, {"type": "input", "label": {"text": "Description", "type": "plain_text", "emoji": true}, "element": {"type": "plain_text_input", "action_id": "input", "multiline": true, "placeholder": {"text": "Fill your description...", "type": "plain_text"}}, {"block_id": "description", "optional": true}]	
2 blank form	[{"type": "input", "label": {"text": "Input", "type": "plain_text", "emoji": false}, "element": {"type": "plain_text_input", "action_id": "input", "placeholder": {"text": "ab...", "type": "plain_text"}}, {"type": "input", "label": {"text": "Amount", "type": "plain_text", "emoji": false}, "element": {"type": "plain_text_input", "action_id": "input", "placeholder": {"text": "123...789", "type": "plain_text"}}, {"type": "input", "label": {"text": "Description", "type": "plain_text", "emoji": true}, "element": {"type": "plain_text_input", "action_id": "input", "multiline": true, "placeholder": {"text": "Fill your description...", "type": "plain_text"}}, {"block_id": "description", "optional": true}]	
3 init	[{"type": "input", "label": {"text": "wave: Selamat Datang!", "type": "plain_text", "emoji": false}, "element": {"type": "plain_text_input", "action_id": "input", "placeholder": {"text": "ab...", "type": "plain_text"}}, {"type": "input", "label": {"text": "Amount", "type": "plain_text", "emoji": false}, "element": {"type": "plain_text_input", "action_id": "input", "placeholder": {"text": "123...789", "type": "plain_text"}}, {"type": "input", "label": {"text": "Description", "type": "plain_text", "emoji": true}, "element": {"type": "plain_text_input", "action_id": "input", "multiline": true, "placeholder": {"text": "Fill your description...", "type": "plain_text"}}, {"block_id": "description", "optional": true}]	

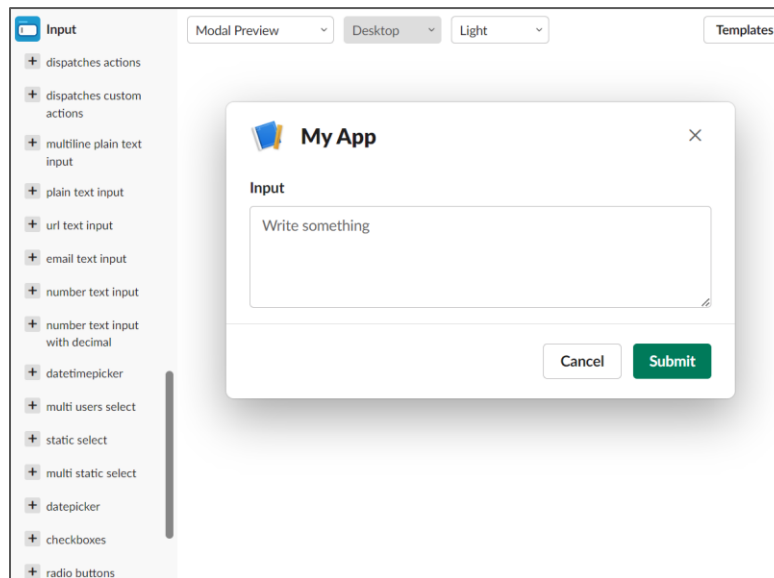
Bentuk data pada kolom *template* adalah bagian **block input** untuk *template* formulir. **Slack™** menggunakan format data **JSON** atau **dictionary** (pada bahasa pemrograman **Python**) untuk membentuk tampilan **view**, pesan, dan beberapa tampilan lainnya. Data tersebut merupakan nilai kembalian yang diberikan oleh fungsi **create_form()** pada *file form_template.py*. Untuk mempelajari lebih lanjut dan mempermudah pembuatan **block**, silahkan kunjungi [Block Kit builder](#) yang telah disediakan oleh **Slack™**, begitu juga dengan [dokumentasi tentang Block Kit](#).



Pada **Block Kit builder**, nilai yang diperlukan hanyalah nilai **item** pada **key “blocks”**. Ambil keseluruhan nilai, termasuk kurung **array** atau **list**, dan masukkan ke dalam **database** sebagai data pada kolom **template**.

```
"blocks": [
  {
    "type": "input",
    "label": {
      "text": "Input",
      "type": "plain_text",
      "emoji": true
    },
    "element": {
      "type": "plain_text_input",
      "action_id": "input",
      "multiline": true
    },
    "block_id": "input"
  }
]
```

Isi dari **key** “*blocks*” adalah **block-block** yang mewakili *input*, teks, dan bagian-bagian lainnya dari sebuah **view**. Pada **Slack Form App**, **block** yang biasanya dipakai hanyalah **block input**. *Detail* dari **block input** dapat dipelajari melalui [Block Kit builder](#) sendiri atau [dokumentasinya](#). Pastikan semua *block* yang berada di dalam **view** memiliki nilai **block_id**. Jika belum ada, **block_id** dapat ditambahkan dengan nilai apapun.



Untuk setiap **block input**, jangan lupa untuk menambahkan **action_id** di dalam **element** dengan nilai “*input*” agar **Slack Form App** dapat mengambil isi tiap **block** yang merupakan *input*. Hal ini dikarenakan fungsi pada file **form_app.py** mencari **key** dengan nilai “*input*” dan mengambil **value** dari **key** “*input*” tersebut.

```

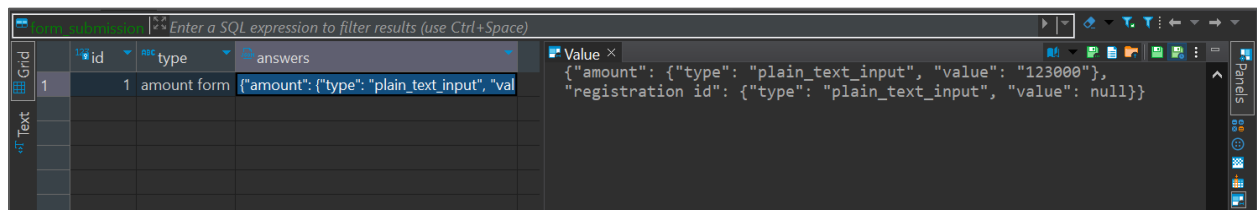
form_app.py
59     ...
60
61     #      =====
62     #      || Event Section      ||
63     #      =====
64
65     ...
66
90     #      -- Form Submit Event --
91     @slack.view("slack_form")
92     def submit_form(ack, view, client):
93         ...
94
96         answers = {}
97         input_blocks = view['state']['values']
98
99         for block_id, value_id in input_blocks.items():
100             if block_id == "type_menu":

```

101	continue
102	answers[block_id] = value_id['input']
103	
104	...

c. Membaca Isi Formulir Terkumpul

Slack Form App menyimpan semua jawaban dari formulir terkumpul dalam tabel **form_submission**. Data tersebut berada pada kolom “*answers*” di mana format data yang terkumpul adalah **JSON**. Data **JSON** disajikan dengan nilai **key** adalah **block_id** dari semua **block input**. **Key** memiliki nilai **value** berupa **value** dari **key** “*input*”, **action_id** yang dimiliki oleh semua **block input**.



id	type	answers
1	amount form	{"amount": {"type": "plain_text_input", "value": "123000"}, "registration id": {"type": "plain_text_input", "value": null}}

Nilai **value** dalam data **JSON** dapat berupa **objek** atau **array** karena **value** dapat berbeda tergantung dengan jenis **block input**. Untuk mengetahui nilai yang diisi oleh pengguna, cari nilai **key** berupa “*value*”. **Key** hanya ada satu untuk tiap **block_id** dan **value**-nya berupa data yang diisi oleh pengguna.