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](#h2_1)

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

1. [Integrasi API Slack](#h2_2)

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

1. [Implementasi Sisi Pengguna](#h2_3)

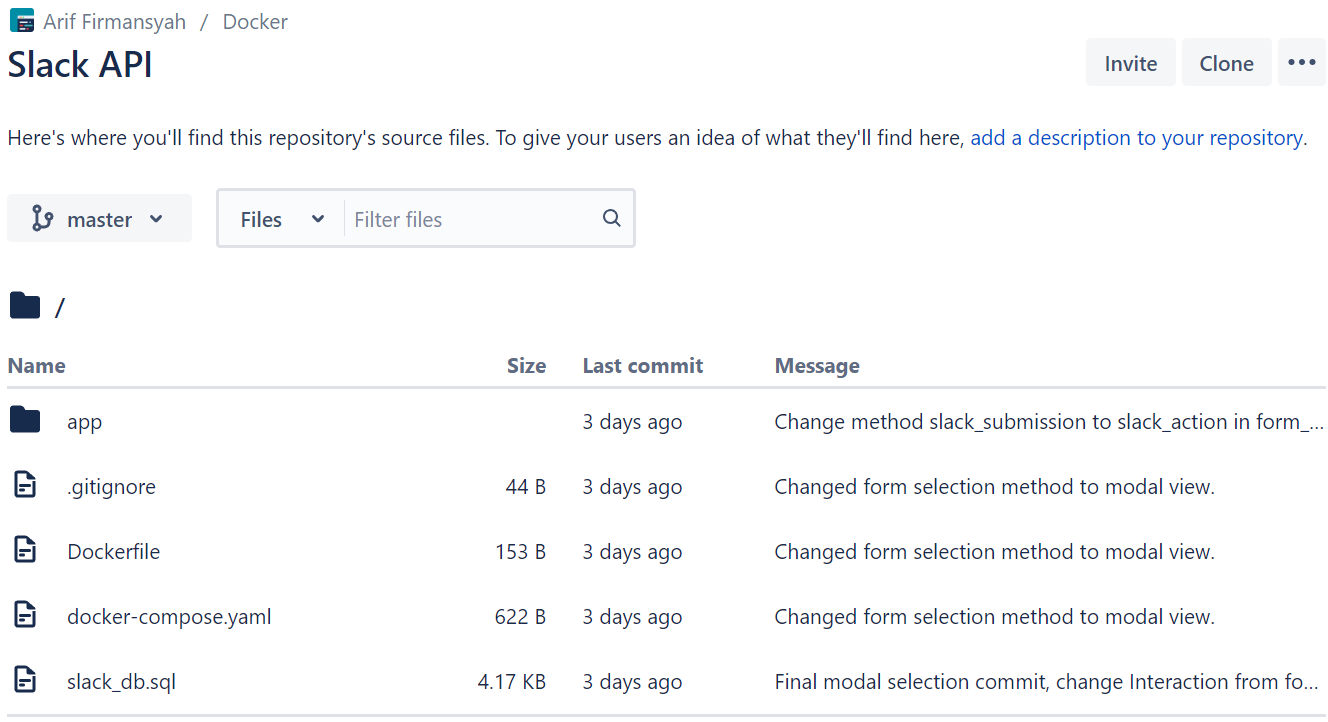
Membahas alur penggunaan **Slack Form App** dan proses yang terjadi dari sisi **backend**.

1. [Pengaksesan *Database*](#h2_4)

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

1. **Pengunduhan dan Instalasi**

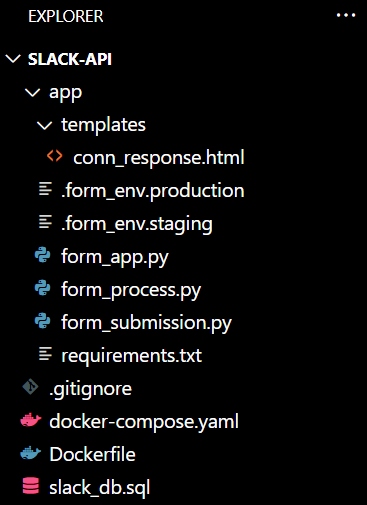
**Slack Form App** dapat berjalan pada *operating system* manapun dengan menggunakan *container docker*. *Script*, *dump*, dan *file*-*file* terkait lainnya tersimpan bersama *file* **Dockerfile** dan *file* **docker-compose.yaml** pada repositori [**BitBucket**](https://bitbucket.org/arif_firmansyah_biznet/slack-api/src/master/).



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.



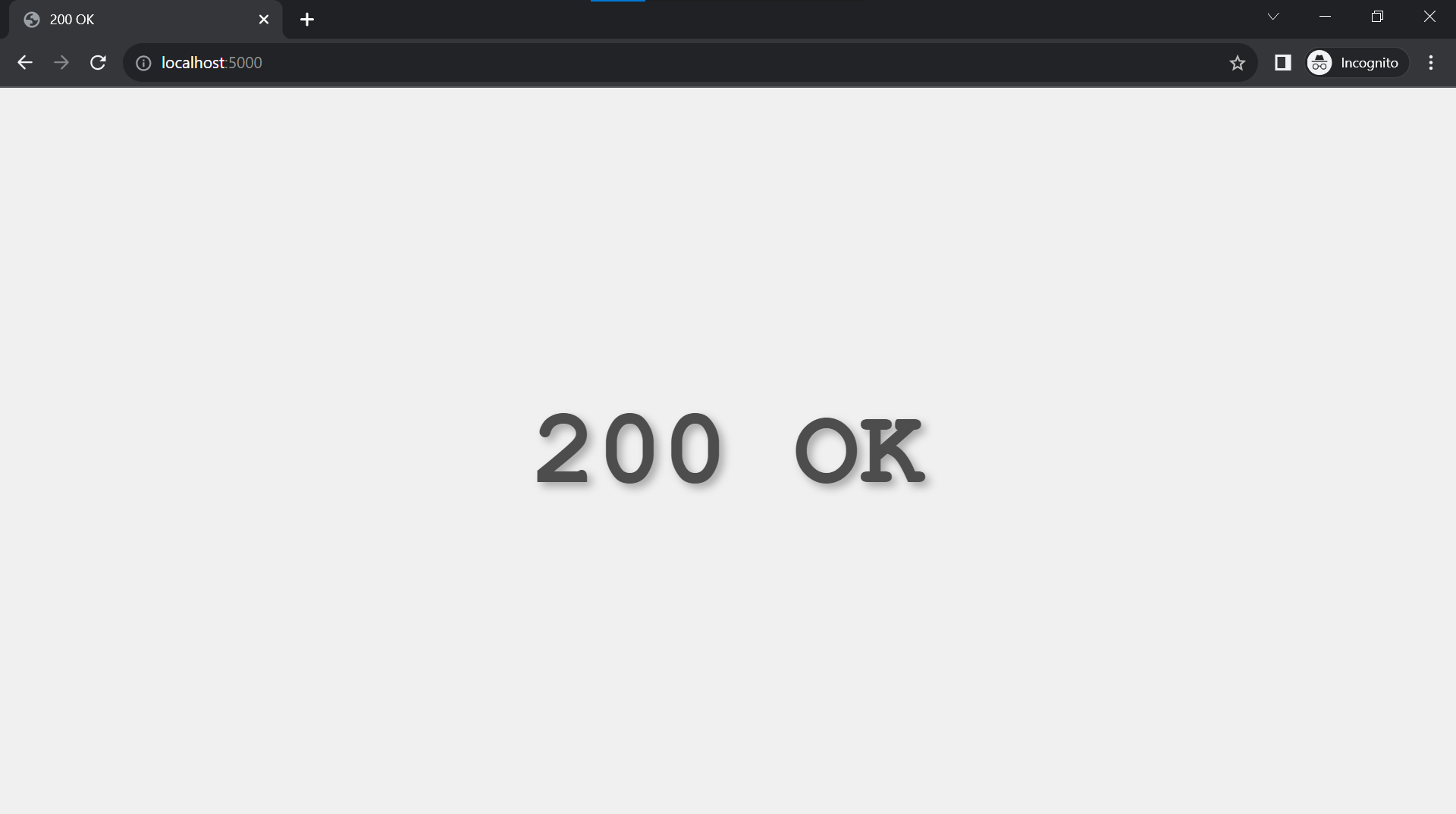
Direktori salinan akan berisi *file*-*file* yang dibutuhkan untuk membuat *container* **Slack Form App** menggunakan perintah *command line* **Docker**. Untuk itu, mohon pastikan **Docker**telah [terinstalasi](https://docs.docker.com/engine/install/) dan siap digunakan pada komputer Anda agar **Slack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



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

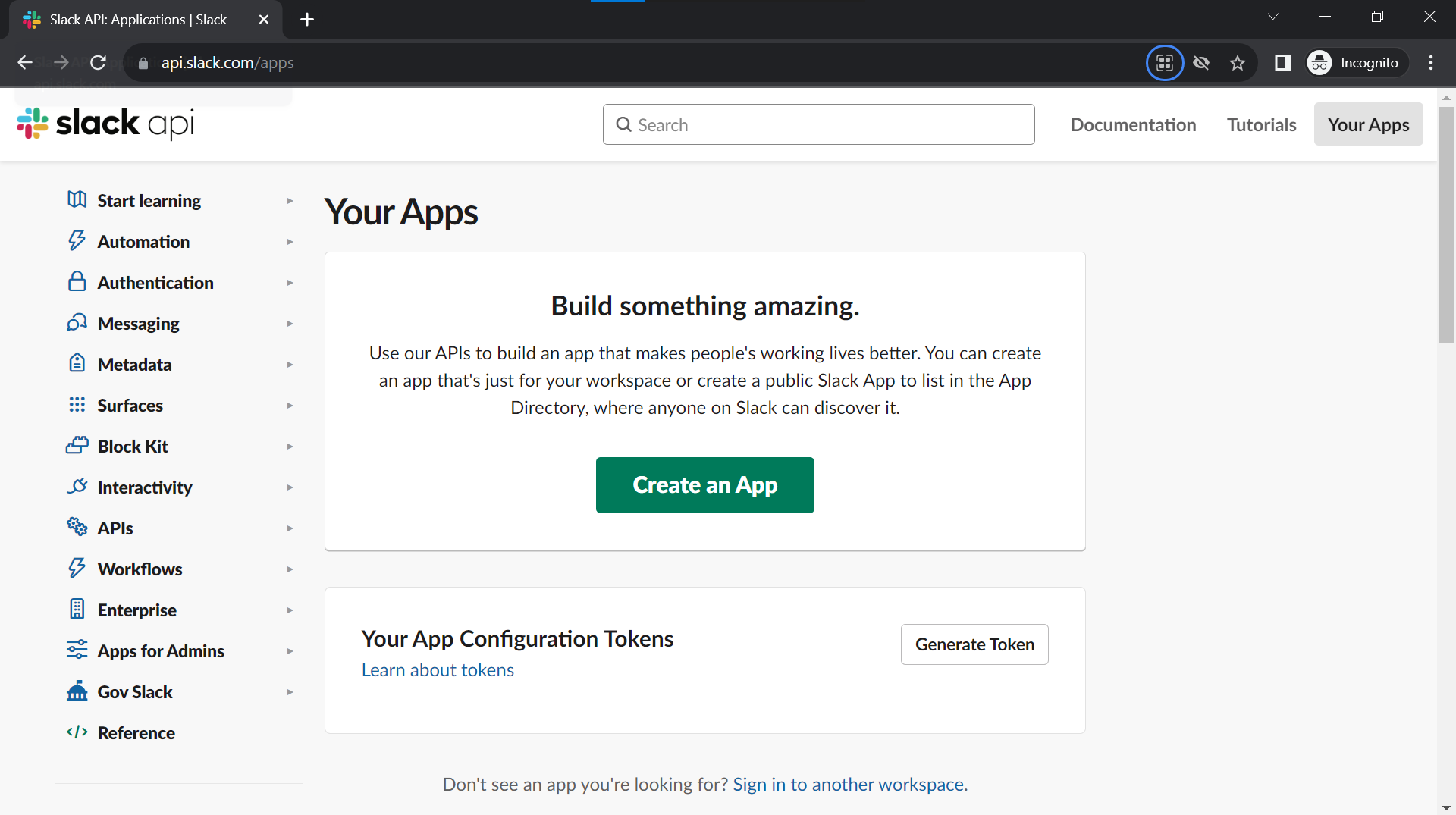
|  |
| --- |
| docker-compose up -d |

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:



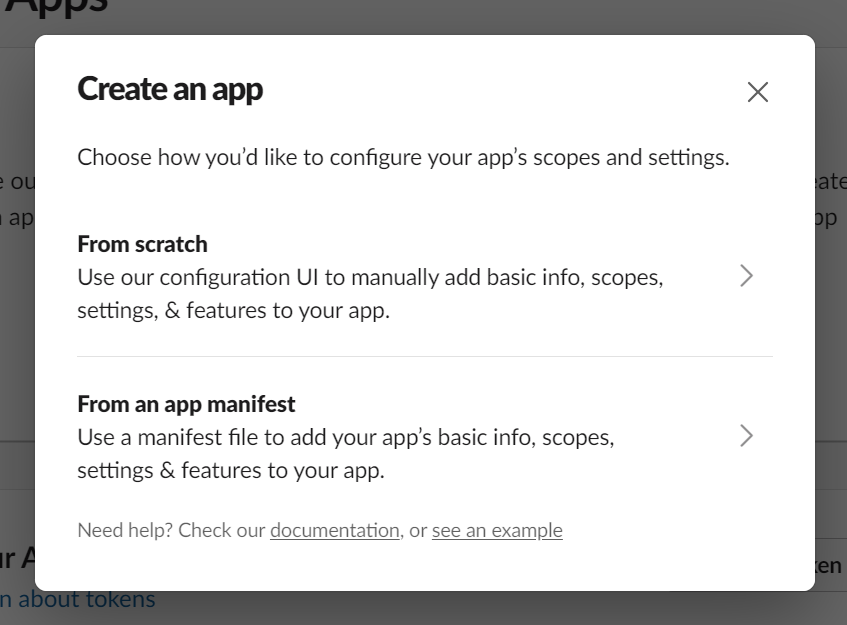
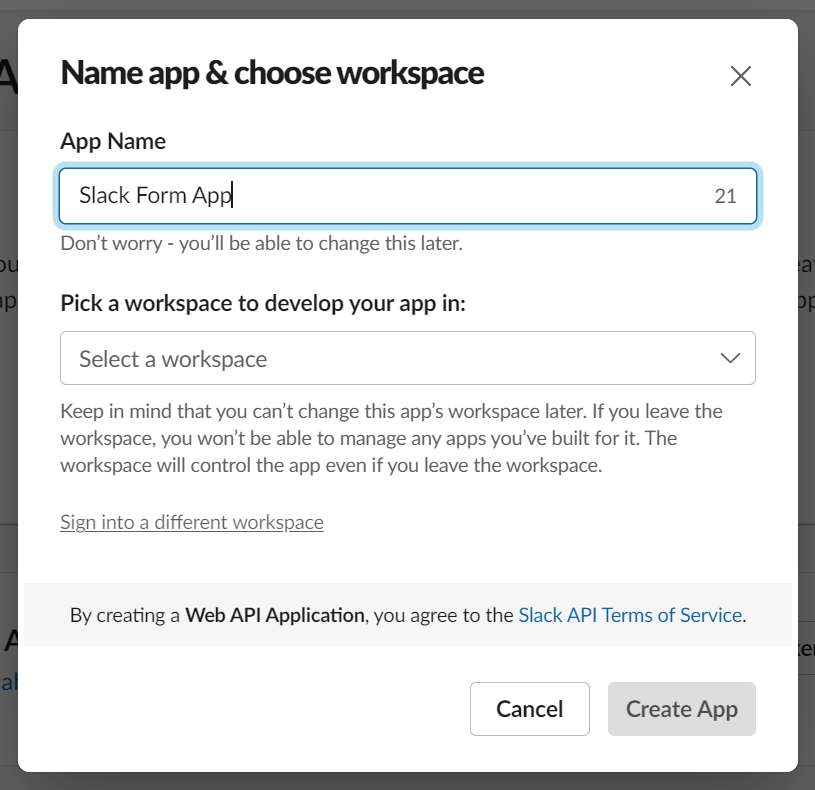
1. **Integrasi API Slack**

**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™**](website%20API%20Slack™).



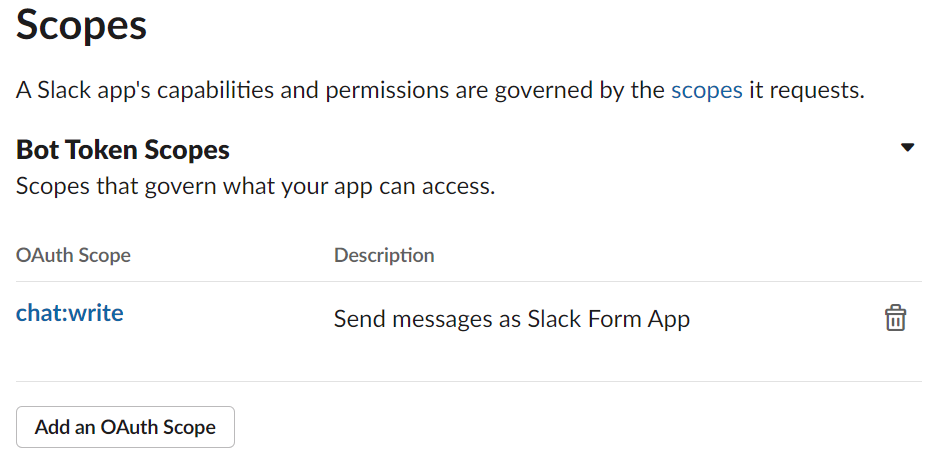
1. Pembuatan Aplikasi Slack

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*”.

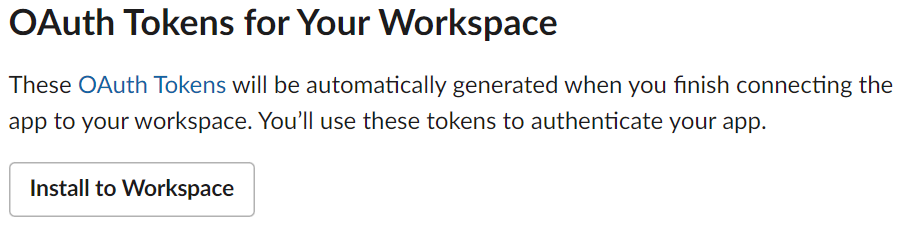
 

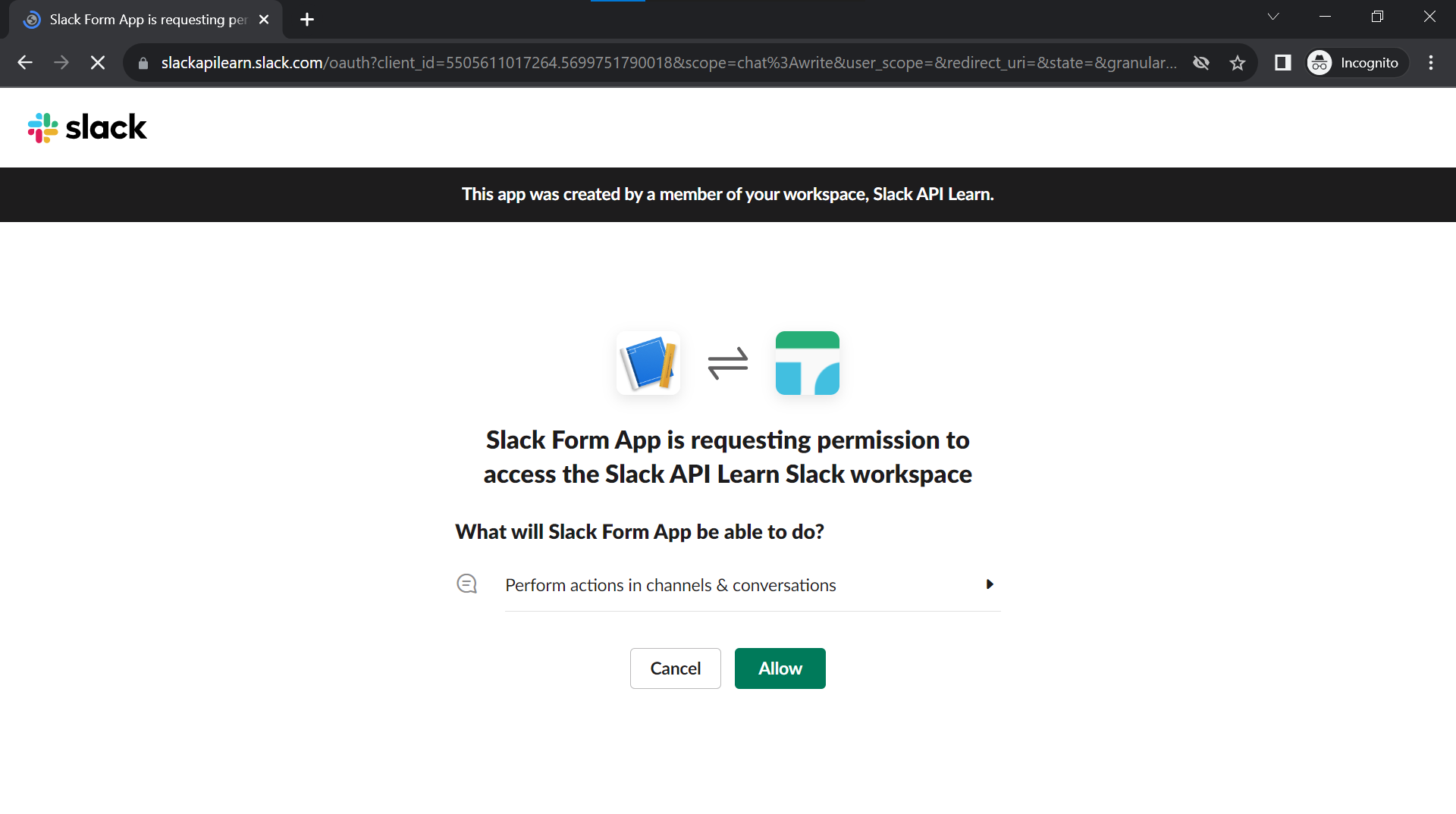
1. Penambahan Scope

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.



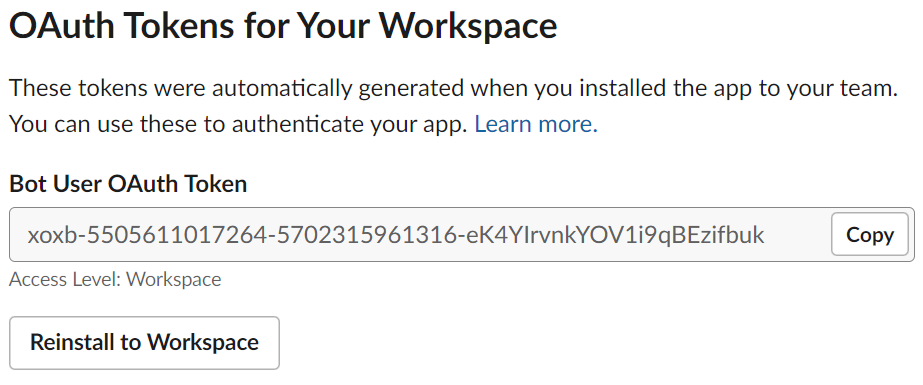
Jangkauan akses API yang berlaku untuk sebuah **aplikasi Slack** hanya mengikuti **scope** yang telah ditambahkan. Sebuah **aplikasi Slack** juga boleh memiliki semua **scope.** Namun, penambahan **scope** apapun perlu diinstalasikan ke **aplikasi Slack** pada *workspace*. Untuk melakukan instalasi,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



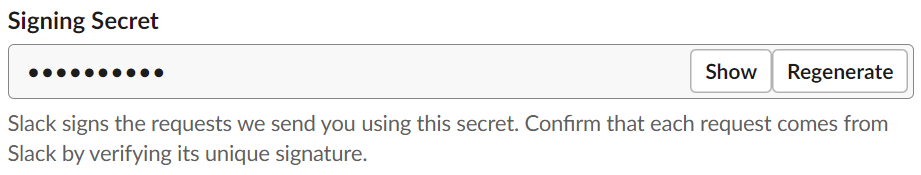


1. Bot Token & Signing Secret

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.



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.



**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** dikonfigurasikan pada bagian **App Initialization**.

|  |  |
| --- | --- |
| **form\_app.py** | |
| 15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | ...  # ========================  # || Config Loading ||  # ========================  config = dotenv\_values(".form\_env")  # ========================  # || App Initialization ||  # ========================  current\_form = None  app = Flask(\_\_name\_\_)  slack = App(  token = config["SLACK\_BOT\_TOKEN"],  signing\_secret = config["SLACK\_SIGNING\_SECRET"]  )  slack\_handler = SlackRequestHandler(slack)  ... |

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 |
| #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=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx |

*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.

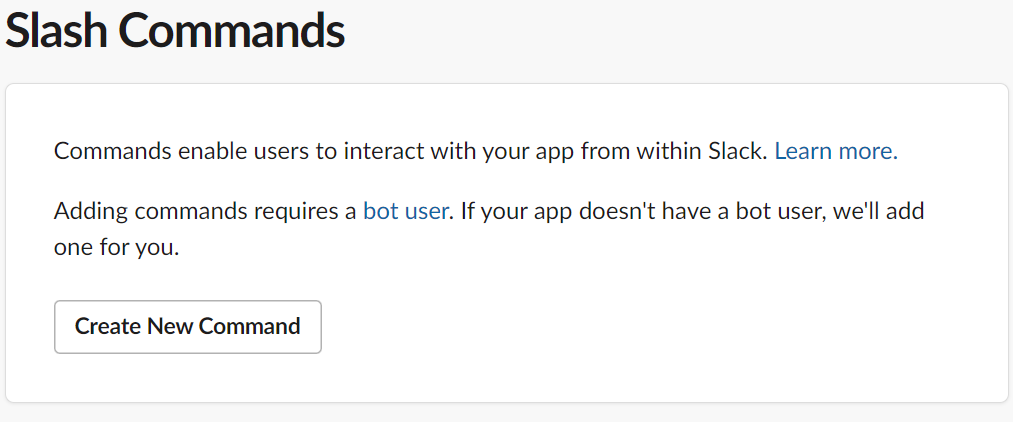
1. Endpoint

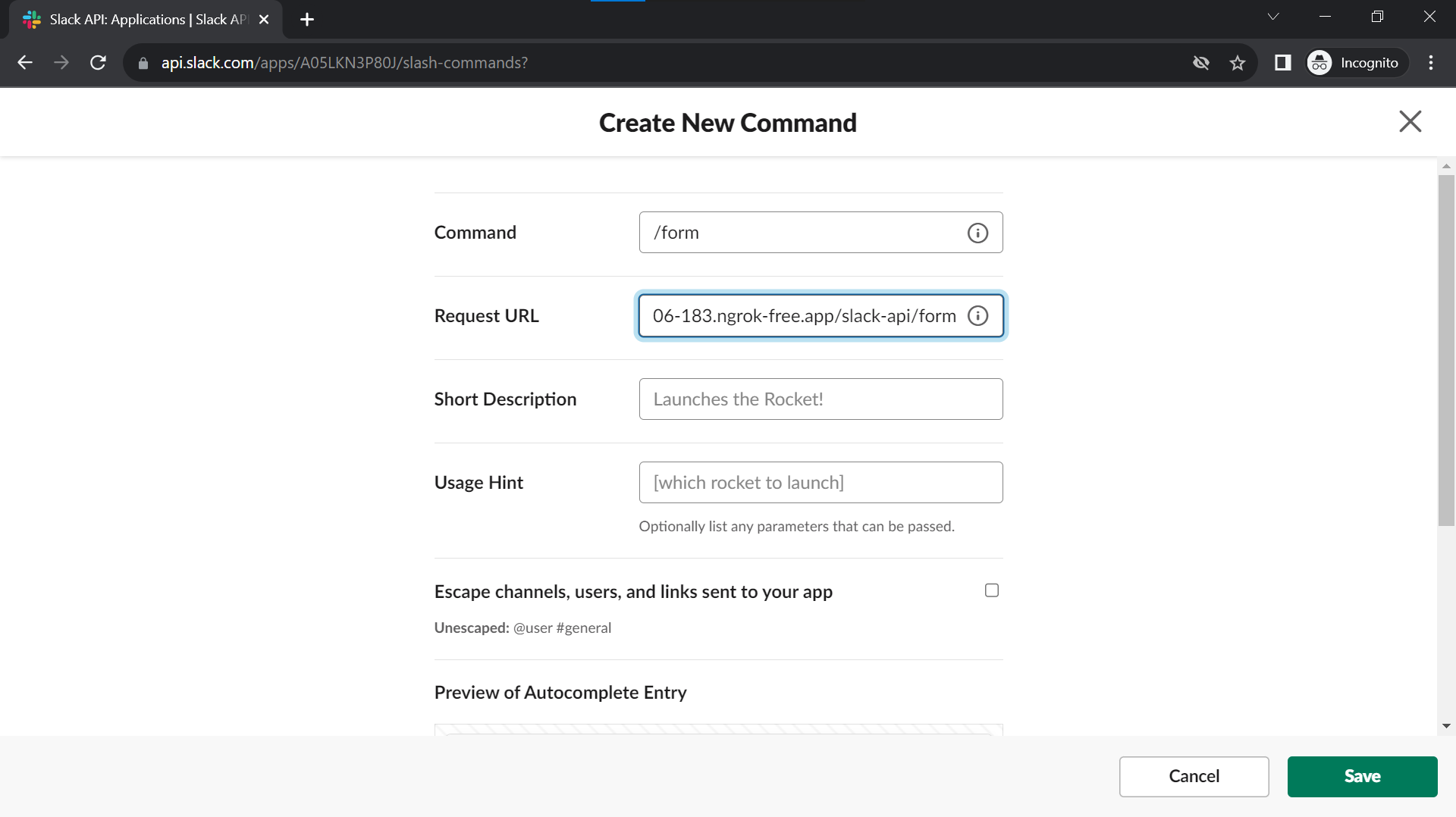
**Slack Form App** dapat menjalankan fungsi-fungsinya setelah mengirim *response* terhadap *request* dari *workspace*. **Slack™** mengirimkan *request* dari *workspace* ke **aplikasi Slack** melalui **endpoint**. **Endpoint** adalah *path* yang dispesifikkan pada *setting* di [**website API Slack™**](file:///C:\Users\PuPPet\Documents\Alat\Sistem\Notepad\Python\website%20API%20Slack™) dan **route** yang terdapat di *file* **form\_app.py**.

|  |  |
| --- | --- |
| **form\_app.py** | |
| 36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | ...  # ========================  # || Route Section ||  # ========================  # -- Connection check. --  @app.route('/', methods = ['GET'])  def slash\_root():  return render\_template("conn\_response.html")  # -- Form view call route. --  @app.route('/slack-api/form', methods = ['POST'])  def slack\_form():  response = slack\_handler.handle(request)  return response  # -- Form submit post route --  @app.route('/slack-api/action', methods = ['POST'])  def slack\_action():  response = slack\_handler.handle(request)  return response  ... |

**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](https://ngrok.com/).

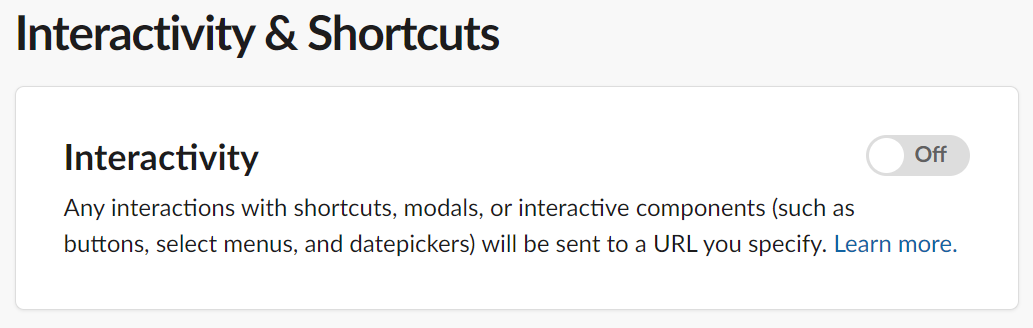
Penambahan **endpoint** untuk perintah /form, akses menu “*Slash Commands*” dilakukan melalui *sidebar* di sebelah kiri layar. **Slash command** adalah perintah yang diberikan melalui *chat* pada sebuah *channel* di mana penulisannya diawali dengan garis miring (/). /form merupakan salah satu contoh 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*”.



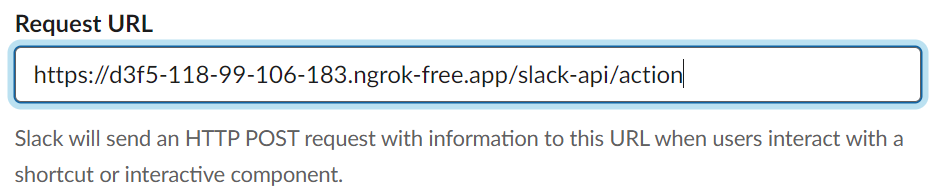


Bentuk nilai **endpoint** nilai **Alamat IP publik** ditambah dengan nilai *path* pada **route** di *file* **form\_app.py**. Bentuk keseluruhannya dapat dilihat pada foto di atas. Setelah nama **slash command** dan **endpoint** telah terisi, klik tombol “*Save*” untuk menyimpan **slash command** tersebut. 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 “*Interactivitiy & Shortcuts*”, dapat diakses melalui *sidebar* di sebelah kiri layar. Nyalakan fitur **Interactivity** terlebih dahulu agar dapat diintegrasikan ke **Slack Form App**.



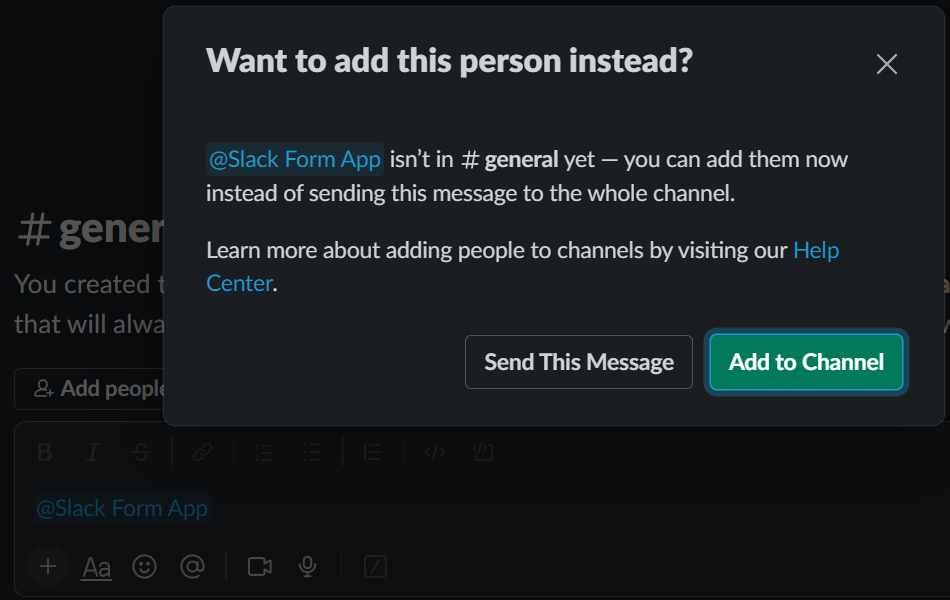
Setelah **Interactivity** dinyalakan masukkan **endpoint** dengan bentuk yang sama seperti pada **slash command**. Bentuk keseluruhannya dapat dilihat pada foto berikut:



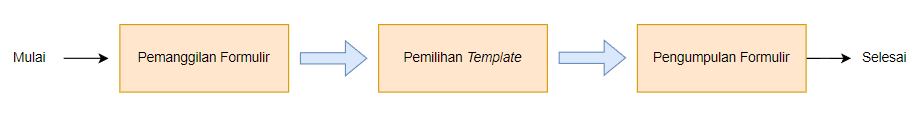
Simpan **endpoint** dengan melakukan klik tombol “*Save Changes*” dan **endpoint** akan secara otomatis terintegrasi ke **Slack Form App**.

1. **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*”.



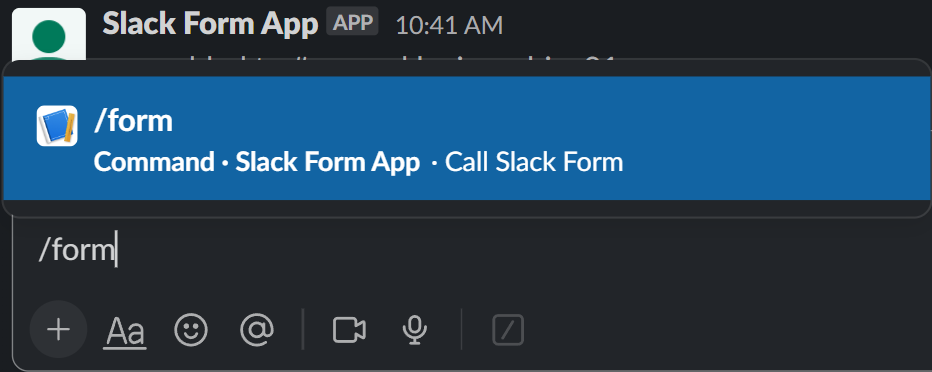
Setelah ini, segala **slash command** yang ada pada **Slack Form App** sudah bisa digunakan. Keseluruhan fungsi **Slack Form** **App** sudah bisa bekerja. Alur penggunaan **Slack Form App** dalam suatu *workspace* dapat dirangkum seperti yang tergambar pada diagram berikut:



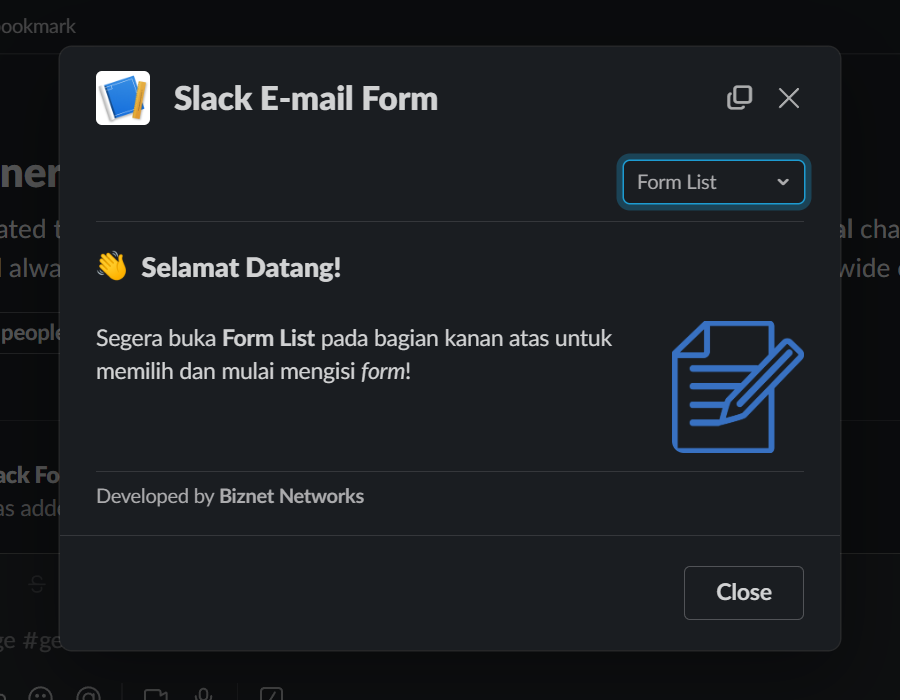
Terdapat tiga interaksi pengguna/aksi utama yakni **pemanggilan formulir**, **pemilihan *template***, dan **pengumpulan formulir**. Ketiga aksi ini berinteraksi dengan **controller** **form\_app.py** untuk mengambil *template* formulir atau mengumpulkan data. Aksi-aksi lainnya merupakan bagian dari salah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

1. 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()**](https://api.slack.com/methods/views.open) untuk menampilkan **view** bertipe **modal**. Selengkapnya dapat dilihat di dokumentasi **Slack™** tentang [**modal view**](https://api.slack.com/surfaces/modals#lifecycle).



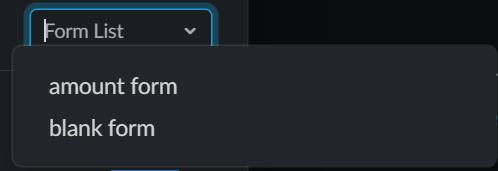
|  |  |
| --- | --- |
| **form\_app.py** | |
| 59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77 | ...  # ========================  # || Event Section ||  # ========================  # -- Form Initialize Event --  @slack.command("/form")  def init(ack, body, client):  ack()  global current\_form  current\_form = FormProcess()  client.views\_open(  trigger\_id = body['trigger\_id'],  view = current\_form.create\_form('init')  )  ... |

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**.

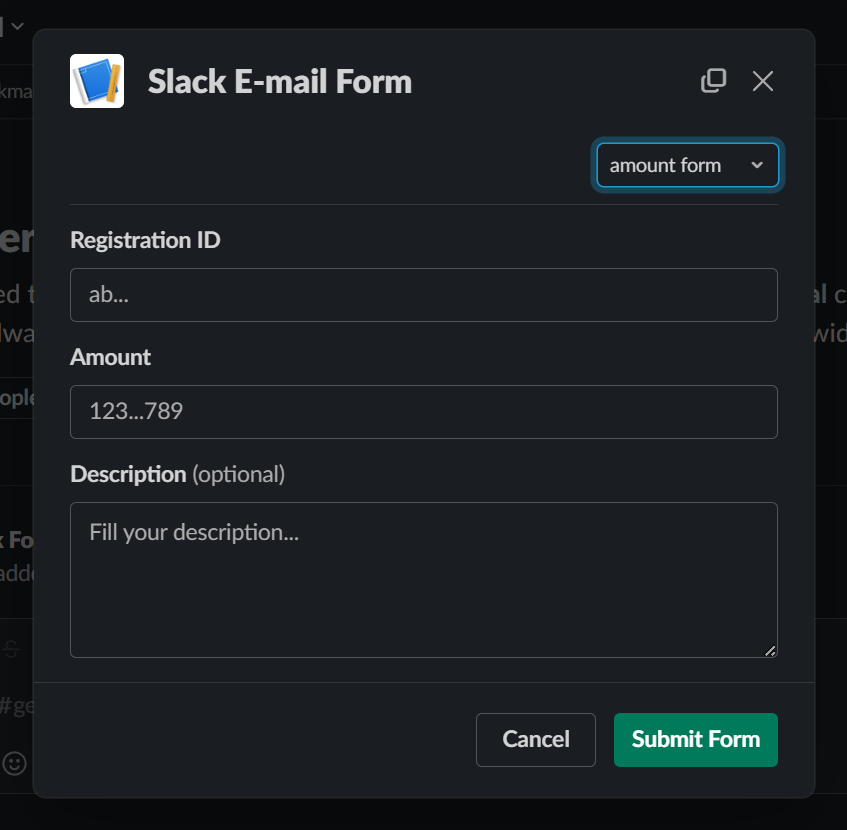
|  |  |
| --- | --- |
| **form\_template.py** | |
| 80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114 | ...  def create\_form(self, command):  view = copy.deepcopy(self.template)  # -- Open Database Connection. --  conn = mysql.connect()  cursor = conn.cursor()  # -- Execute SQL Commands --  # cursor = mysql.connection.cursor()  query = f"SELECT type, template FROM form\_templates WHERE type = '{command}';"  cursor.execute(query)  data\_row = cursor.fetchone()  # -- Close Database Connection. --  cursor.close()  conn.close()  self.type = data\_row[0]  if command != "init":  view["submit"] = {  "type": "plain\_text",  "text": "Submit Form",  "emoji": True  }  view["close"] = {  "type": "plain\_text",  "text": "Cancel",  "emoji": True  }  for block in json.loads(data\_row[1]):  view['blocks'].append(block)  return view |

1. 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*”.

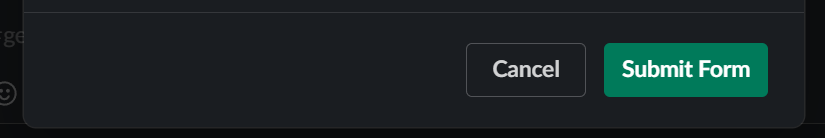


Menu *dropdown* akan tetap ada meskipun formulir berganti sehingga pengguna bebas mengganti formulir selama **Slack Form App** masih terbuka. Pergantian formulir dan pemilihan *template* ditangani oleh *file* **form\_app.py**. Mekanismenya mirip dengan pada tahap **Pemanggilan Formulir.** *Workspace* 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()**](https://api.slack.com/methods/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*.

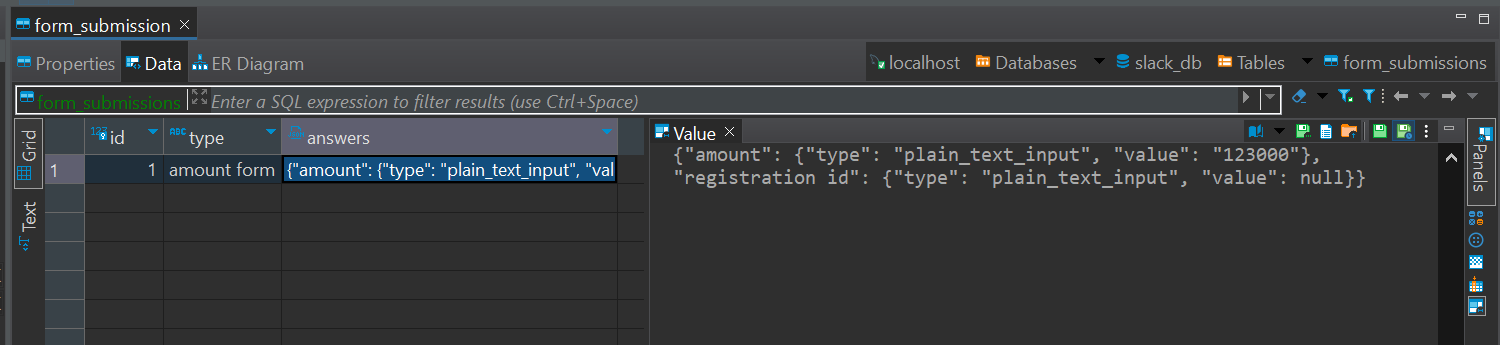
|  |  |
| --- | --- |
| **form\_app.py** | |
| 59  60  61  62  63  78  79  80  81  82  83  84  85  86  87  88  89  90 | ...  # ========================  # || Event Section ||  # ========================  ...  # -- Form Select Event --  @slack.action("type\_list")  def select\_form(ack, body, client):  ack()  selected = f"{body['actions'][0]['selected\_option']['value']}"  client.views\_update(  view\_id = body["view"]["id"],  hash = body["view"]["hash"],  view = current\_form.create\_form(selected)  )  ... |

1. 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  63  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108 | ...  # ========================  # || Event Section ||  # ========================  ...  # -- Form Submit Event --  @slack.view("slack\_form")  def submit\_form(ack, view, client):  ack()  global current\_form  answers = {}  input\_blocks = view['state']['values']    for block\_id, value\_id in input\_blocks.items():  if block\_id == "type\_menu":  continue  answers[block\_id] = value\_id['input']  print(answers)  results = FormSubmission(current\_form, answers)  results.submit\_form()  current\_form = None  ... |

|  |  |
| --- | --- |
| **form\_submission.py** | |
| 13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | ...  def submit\_form(self):  # Format Results To JSON  answers\_json = json.dumps(self.answers)  answers\_sql\_json = json.dumps(answers\_json)  # -- Open Database Connection. --  conn = mysql.connect()  cursor = conn.cursor()  # -- Execute SQL Commands --  # cursor = mysql.connection.cursor()  query = f"INSERT INTO form\_submission (type, answers) VALUES ('{self.form.type}', {answers\_sql\_json});"  cursor.execute(query)  conn.commit()  # -- Close Database Connection. --  cursor.close()  conn.close() |

1. **Pengaksesan Database**
2. Membuat Koneksi Database

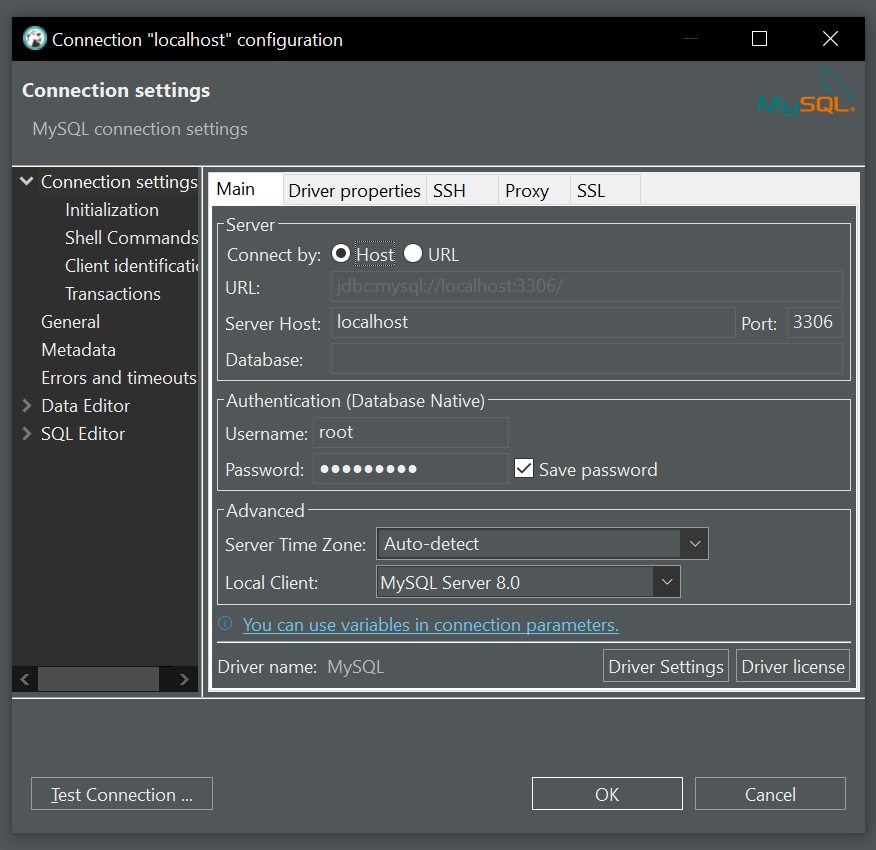
**Slack Form App** menggunakan **MySQL** sebagai solusi manajemen **database**. Oleh karena itu, *file* **docker-compose.yaml** juga memiliki **service** bernama **database** yang merupakan sebuah **image** dari **MySQL**, yang akan berjalan bersama dengan **Slack Form App**. Pengaksesan **database** 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  10  11  12  13  14  15 | ...  app.config['MYSQL\_DATABASE\_HOST'] = config['MYSQL\_LOCAL\_HOST']  app.config['MYSQL\_DATABASE\_USER'] = config['MYSQL\_LOCAL\_USER']  app.config['MYSQL\_DATABASE\_DB'] = config['MYSQL\_LOCAL\_DB']  app.config['MYSQL\_DATABASE\_PASSWORD'] = config['MYSQL\_LOCAL\_PASSWORD']  mysql = MySQL(app)  ... |

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

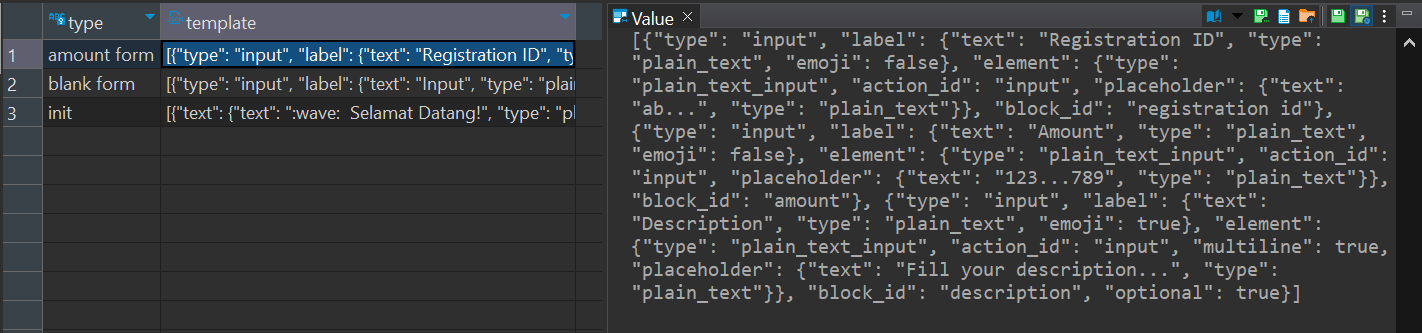
|  |
| --- |
| .form\_env |
| #Ganti nilai dengan SLACK\_BOT\_TOKEN dengan nilai bot token SLACK\_BOT\_TOKEN=xoxb-xxxxxxx-xxxxx-xxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxx  #Ganti nilai dengan SLACK\_SIGNING\_SECRET dengan nilai signing secret  SLACK\_SIGNING\_SECRET=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  #Variabel 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.

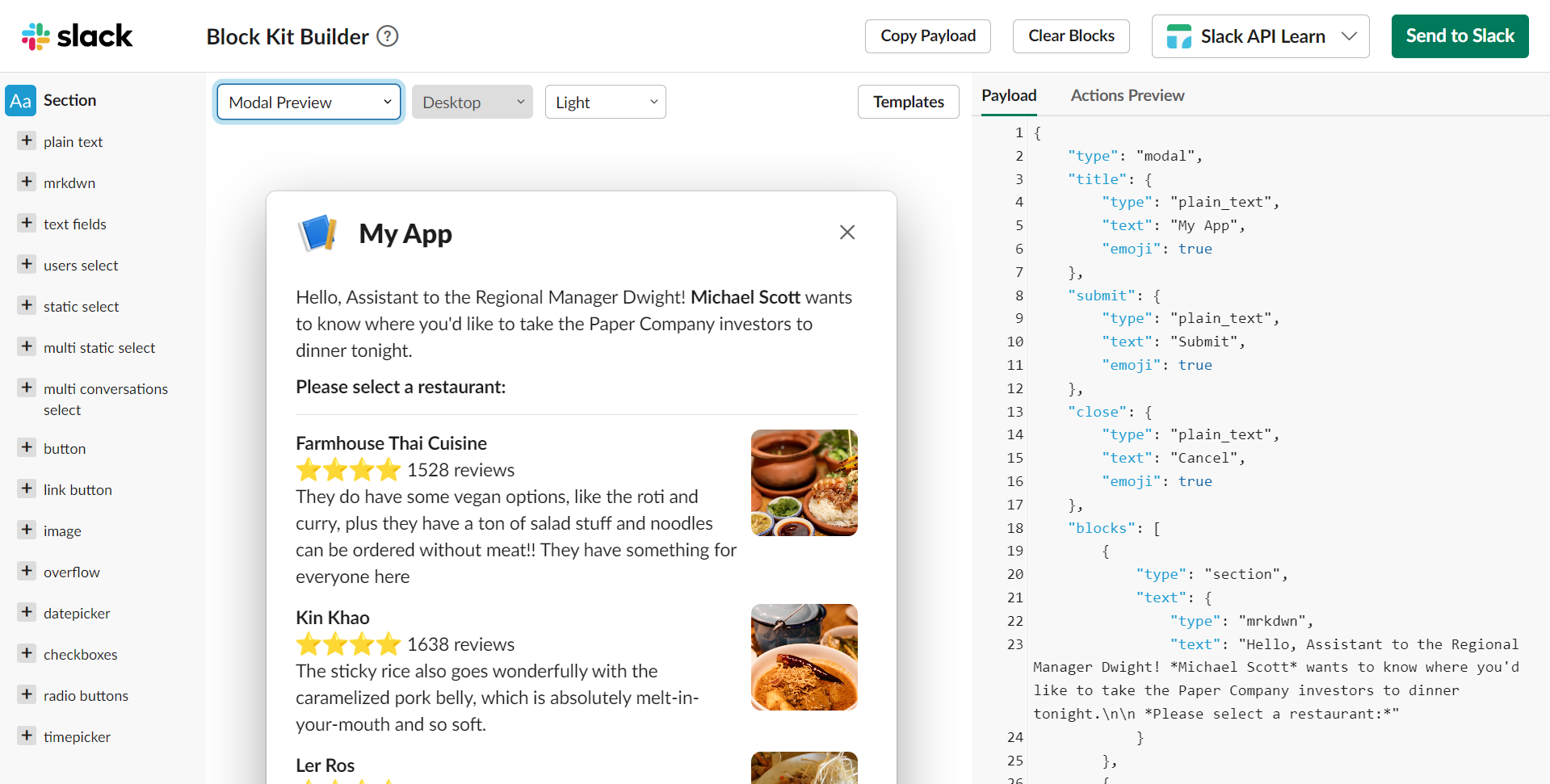


1. Menambah *Template*

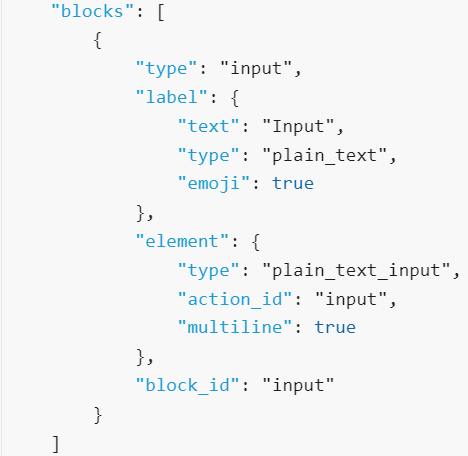
Semua *template* formulir berada di tabel **form\_templates**. Tabel **form\_templates** terdiri dari dua kolom, yaitu kolom **type** dan kolom **template**. Kolom **type** meyimpan tipe/nama *template* yang juga akan ditampilkan pada menu *dropdown* untuk memilih formulir. Di sampingnya, kolom **template**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



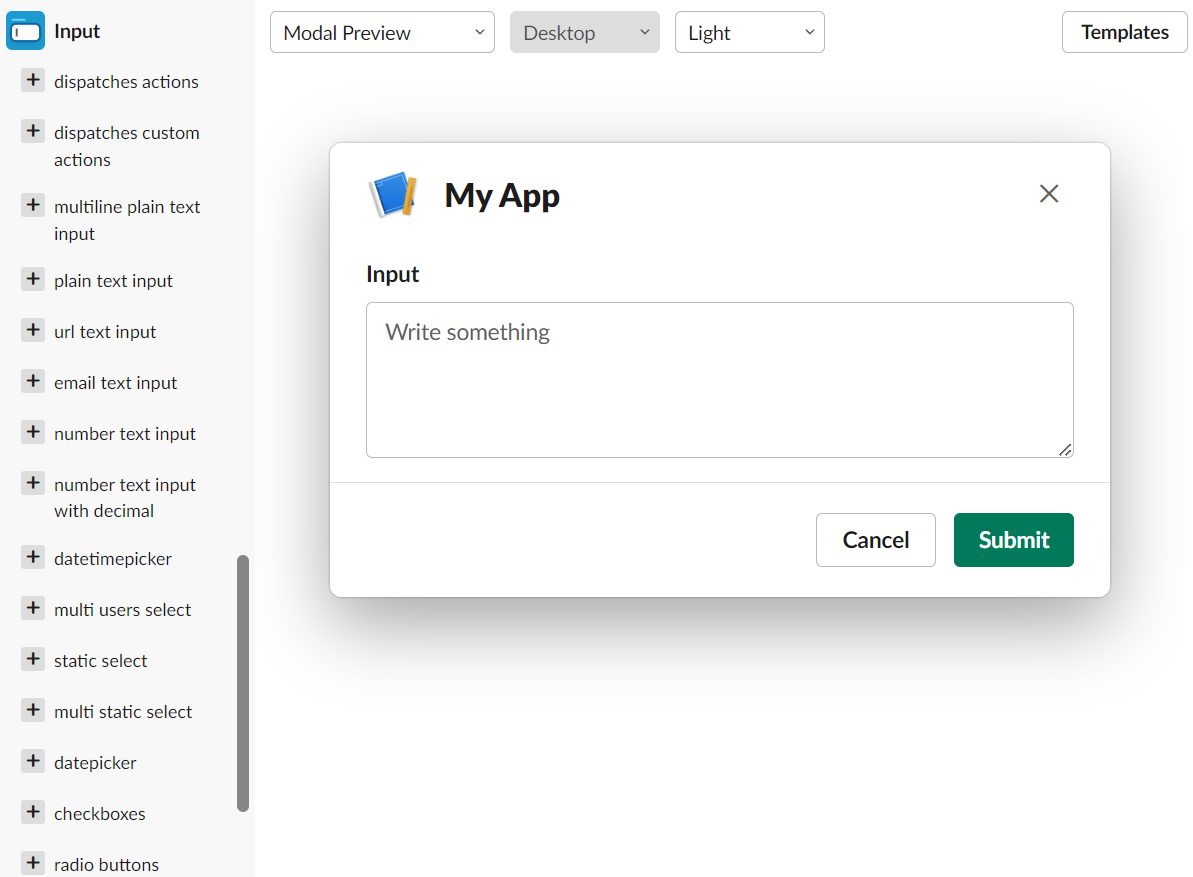
Bentuk data pada kolom *template* adalah bagian **block** *input* untuk *template* formulir. **Slack™** menggunakan format data **JSON** atau **dictionary** (pada bahasa pemrograman **Python**) untuk membentuk tampilan **view**, pesan, dan beberapa tampilan lainnya. Data tersebut merupakan nilai kembalian yang diberikan oleh fungsi **create­\_form()** pada *file* **form\_template.py**. Untuk mempelajari lebih lanjut dan mempermudah pembuatan **block**, silahkan kunjungi [**Block Kit builder**](https://app.slack.com/block-kit-builder) yang telah disediakan oleh **Slack™**, begitu juga dengan [dokumentasi tentang **Block Kit**](https://api.slack.com/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**.



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**](https://app.slack.com/block-kit-builder) sendiri atau [dokumentasinya](https://api.slack.com/reference/block-kit/blocks#input). Pastikan semua *block* yang berada di dalam **view** memiliki nilai **block\_id**. Jika belum ada, **block\_id** dapat ditambahkan dengan nilai apapun.

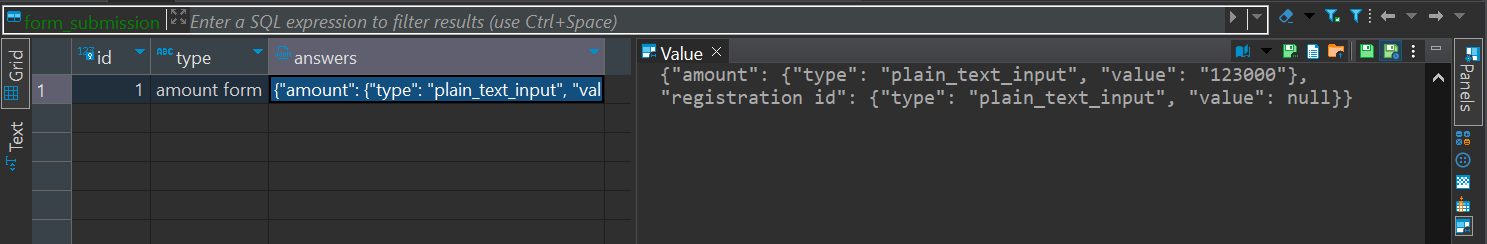


Untuk setiap **block** *input*, jangan lupa untuk menambahkan **action\_id** di dalam **element** dengan nilai “*input*” agar **Slack Form App** dapat mengambil isi tiap **block** yang merupakan *input*. Hal ini dikarenakan fungsi pada *file* **form\_app.py** mencari **key** dengan nilai “*input*” dan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

|  |  |
| --- | --- |
| **form\_app.py** | |
| 59  60  61  62  63  90  91  92  96  97  98  99  100  101  102  103  104 | ...  # ========================  # || Event Section ||  # ========================  ...  # -- Form Submit Event --  @slack.view("slack\_form")  def submit\_form(ack, view, client):  ...  answers = {}  input\_blocks = view['state']['values']    for block\_id, value\_id in input\_blocks.items():  if block\_id == "type\_menu":  continue  answers[block\_id] = value\_id['input']  ... |

1. 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*.



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.