

## Modul 5: Data Entri Parent-Child

Nama : Agi Priyono

NIM : 064002200018

### Pokok Bahasan

1. mahasiswa mampu mengimplementasi rancangan data entry dengan hubungan parentchild.
2. Mahasiswa memahami cara menyimpan data kedalam database, mengedit dan mencoretnya
3. mahasiswa mampu memvalidasi data entry dan memberikan pesan hasil validasi menggunakan JavaScript/Jquery

### Rincian Tugas Praktikum:

#### Tahap Persiapan

1. Menyiapkan virtual environment untuk membuat proyek menggunakan framework Django
2. Menyiapkan repo di github
3. Membuat proyek django dengan perintah django-admin startproject

#### Tahap Pelatihan

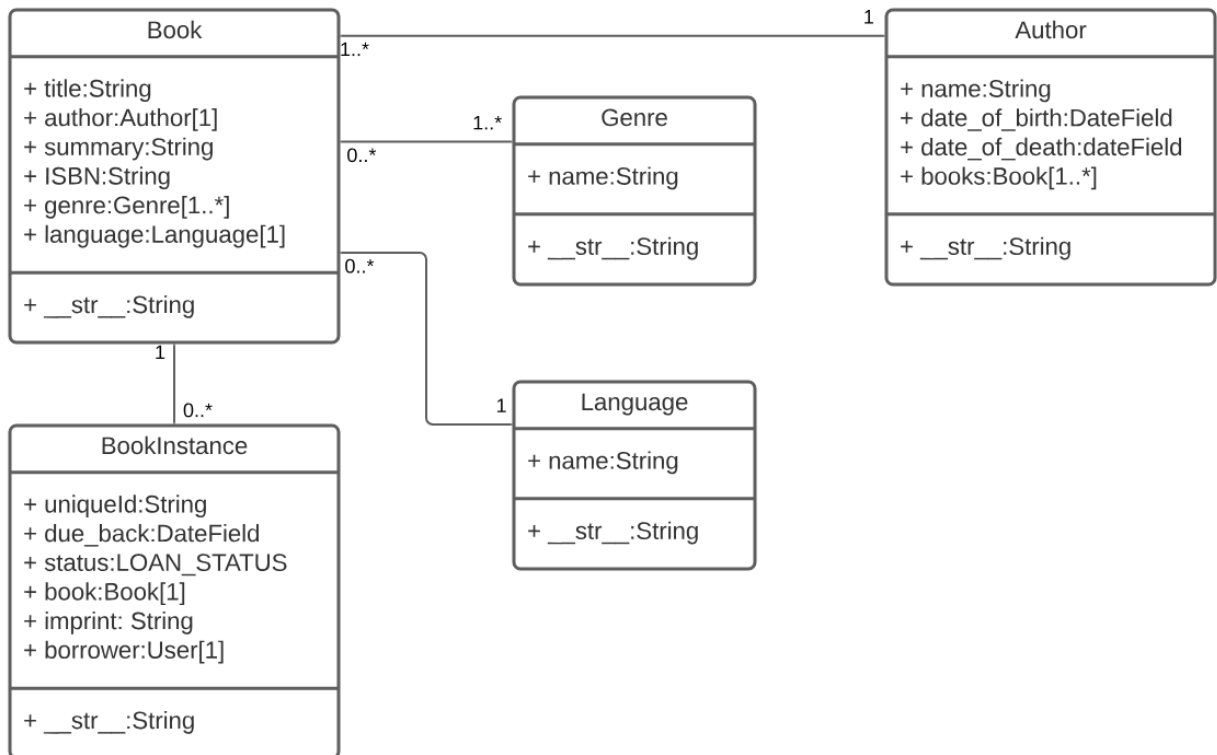
1. Membuat app dengan perintah django-admin startapp
2. Menyiapkan kelas
3. Membuat Form
4. Mendefinisikan fungsi di view untuk menampilkan form
5. Menyiapkan halaman untuk menampilkan form.
6. Melakukan pemanggilan fungsi migrate dan make migration
7. Menjalankan server dengan perintah python manage.py runserver
8. Melakukan pengujian Aplikasi

#### Tahap Penyelesaian Modul Mandiri

1. Verifikasi model yang sudah dibuat
2. Mengimplementasikan mengikuti tahapan pelatihan
3. Mengunggah hasil pekerjaan dan laporan ke github dengan git.

### Teori Singkat

Gambar 5.1 menggambarkan hubungan parent-child antara dua tabel yang terjadi antara tabel Author dan Book, Book dan Genre, Book dengan Language, dan Book dan BookInstance. Parent adalah tabel yang diacu oleh tabel child dengan key dari parent menjadi foreign-key pada tabel child.



Pada relasi buku dan author, seorang author bisa mengarang beberapa buku tetapi satu buku hanya dikarang oleh satu author (ini hanya ilustrasi untuk memudahkan pemahaman saja). Pada kenyataannya satu buku bisa dikarang oleh lebih dari satu author. Pada relasi ini, author berperan sebagai parent dan Book sebagai child. Jika dalam rancangan ditentukan ON DELETE RESTRICT maka jika ada author yang punya child buku mau di hapus maka konstrain tersebut akan memastikan bahwa jika masih ada buku dengan id pengarang tertentu sebagai foreign key maka penghapusan pengarang akan di tolak karena melanggar konstrain. Dengan kata lain, tidak boleh ada anak yang tidak punya orang tua (ON DELETE RESTRICT = You can't delete parent). Perhatikan skema lainnya dan berikan penjelasan.

Ada kondisi dimana jika orang tua dihapus maka childnya juga harus dihapus. Jika seorang penulis blog dihapus dari tabel penulis maka tulisannya juga harus ikut dihapus. Konstrain ini meneruskan perubahan pada parent ke child. ON DELETE CASCADE = Perubahan harus diterapkan di child.

### Tugas Persiapan

1. Menyiapkan virtual environment untuk membuat proyek dengan framework Django mengikuti tutorial sebelumnya.
2. Menyiapkan Repository di Github seperti tutorial sebelumnya. Beri nama repo nim\_web\_prog\_5
3. Melakukan clone di lokal komputer anda.
4. Yay...sudah siap untuk perjalanan selanjutnya...

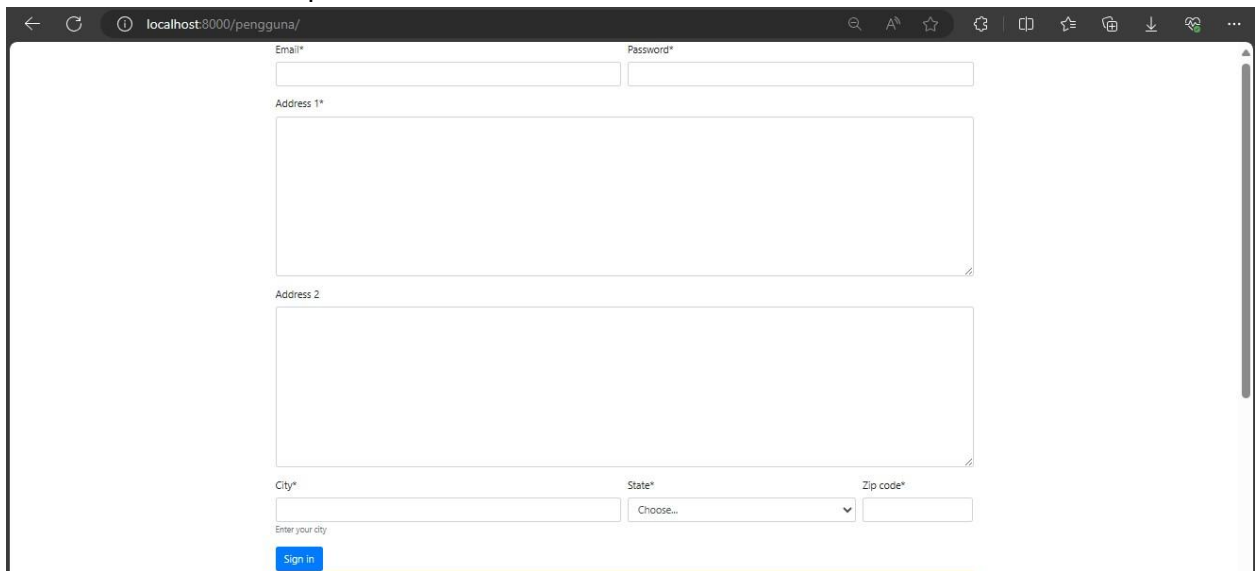
### Tahap Pelatihan

Pada bagian ini anda akan menambahkan satu entitas baru/ tabel baru yang akan menjadi child dari entitas pengguna yang dibuat pada pertemuan sebelumnya.

Menyiapkan Model kedua pada aplikasi data entry

Pada latihan ini kita akan memanfaatkan django-crispy-form untuk menyajikan form di halaman web. Tahapan :

Kita akan melakukan penyesuaian tampilan dari halaman pada praktikum sebelumnya seperti Gambar dibawah ini. Penempatan form yang dapat kita sesuaikan dengan memanfaatkan bootstrap.



The screenshot shows a web browser window with the address bar displaying "localhost:8000/pengguna/". The page contains a registration form with the following fields and elements:

- Email\***: A text input field.
- Password\***: A text input field.
- Address 1\***: A large text area for the first address line.
- Address 2**: A large text area for the second address line.
- City\***: A text input field with the placeholder text "Enter your city".
- State\***: A dropdown menu with the placeholder text "Choose...".
- Zip code\***: A text input field.
- Sign in**: A blue button located at the bottom left of the form.

1. Buat file data\_entry\_2.html dan

```

dj input_data_1.html U ×  dj content.html U  dj pengguna_detail.html
ss_sistem_cerdas > data_entry > templates > data_entry > dj input_data_1.html
Form method="post">
<div class="form-row">
  <div class="form-group col-md-6 mb-0">
    {{ form.email|as_crispy_field }}
  </div>
  <div class="form-group col-md-6 mb-0">
    {{ form.password|as_crispy_field }}
  </div>
</div>
{{ form.address_1|as_crispy_field }}
{{ form.address_2|as_crispy_field }}
<div class="form-row">
  <div class="form-group col-md-6 mb-0">
    {{ form.city|as_crispy_field }}
  </div>
  <div class="form-group col-md-4 mb-0">
    {{ form.state|as_crispy_field }}
  </div>
  <div class="form-group col-md-2 mb-0">
    {{ form.zip_code|as_crispy_field }}
  </div>
</div>
</div>

```

2. gunakan pada fungsi set\_pengguna sehingga fungsi set\_pengguna anda terlihat seperti berikut:

Mengimplementasikan fungsi untuk memfasilitasi pengguna membuat content berupa artikel Sederhananya anda akan membuat halaman yang tampilan awalnya seperti ini:

Tahapan:

1. Membuat kelas Content pada models.py yang memuat atribut : auth

Author:

Artikel:

Set view: ☐

Create

or, date\_created, set\_view (publish dan not publish) dan article yang berisi tulisan yang dibuat.

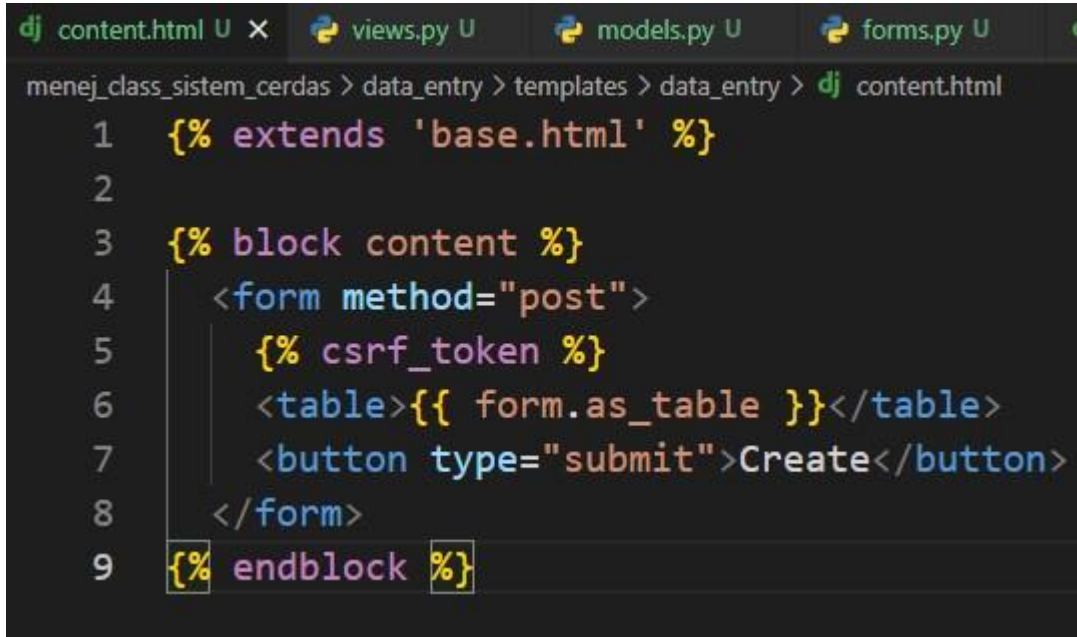
class Content adalah child dari class Pengguna yang harus mengikuti kebijakan jika user dihapus maka semua artikel dari pengguna tersebut juga harus dihapus (ON DELETE CASCADE).

Untuk menerapkan konsep parent-child, maka id kelas Pengguna harus dijadikan sebagai foreign key pada kelas Content dengan ketentuan ON DELETE CASCADE.

```
class Content(models.Model):
    author = models.ForeignKey(Pengguna, on_delete = models.CASCADE)
    date_created = models.DateField(auto_now = True)
    artikel = models.TextField()
    set_view = models.BooleanField(default=False)
```

2. Buat content.html pada folder templates/data\_entry

Hint : lakukan dengan cara yang sama dengan praktikum pertemuan 4, yaitu memanfaatkan `crispy_form`.



```
menej_class_sistem_cerdas > data_entry > templates > data_entry > dj content.html
1  {% extends 'base.html' %}
2
3  {% block content %}
4      <form method="post">
5          {% csrf_token %}
6          <table>{{ form.as_table }}</table>
7          <button type="submit">Create</button>
8      </form>
9  {% endblock %}
```

3. Definisikan fungsi `set_content` di file `views.py` untuk menampilkan form pada saat pertama kali form di load dan menyimpan hasil entry ke dalam database.

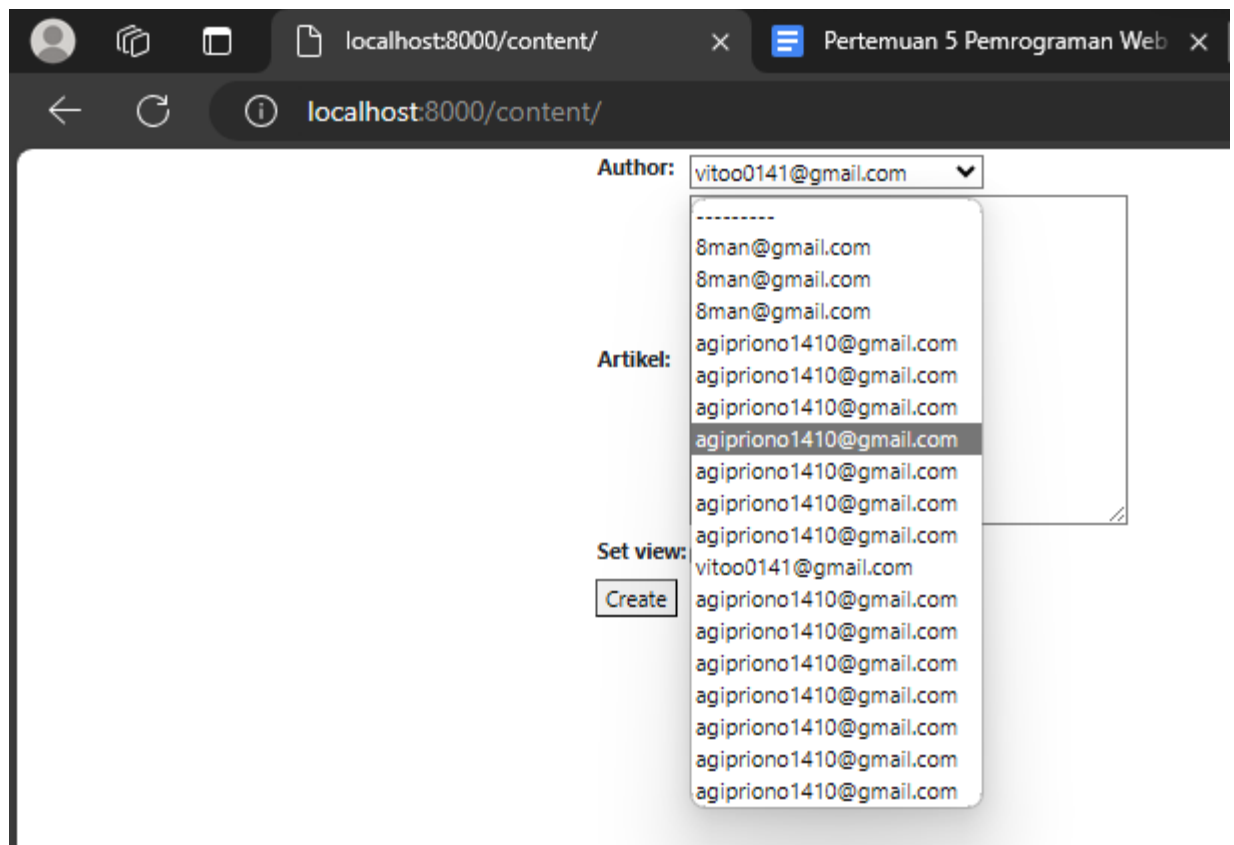
```
views.py U x  dj content.html U  models.py U  forms.py U  dj input_data_1.html U  dj penggu
menej_class_sistem_cerdas > data_entry > views.py
48 def get_pengguna_detail_api(request, user_id):
63
64 def set_content(request):
65     form = ContentForm(None)
66     if request.method == 'POST':
67         form = PenggunaForm(request.POST)
68         if form.is_valid():
69             form.save()
70             list_pengguna = Pengguna.objects.all().order_by('-id')
71             context = {
72                 "form": form,
73                 "list_pengguna": list_pengguna
74             }
75             return render('data_entry/content.html', context)
76     else :
77         context = {
78             'form': form,
79         }
80     return render(request, 'data_entry/content.html', context)
```

Tambahkan path ke urls.py di folder app

```
path('content/', views.set_content, name='set_content'),
```

```
path('content/', views.set_content, name='set_content'),
```


4. Panggil fungsi runserver dan ujicoba apakah dapat berfungsi dengan baik atau tidak. Pada saat anda gunakan anda akan menemukan kondisi seperti ini:



Daftar author yang ada didalam database sudah ada didalam list namun belum dapat dikenali. Masih perlu dilakukan pendefinisian fungsi pada class Pengguna sehingga dapat menampilkan informasi yang lebih mudah dipahami.

5. Jika anda coba isi form dan melakukan klik button create, seperti dibawah ini



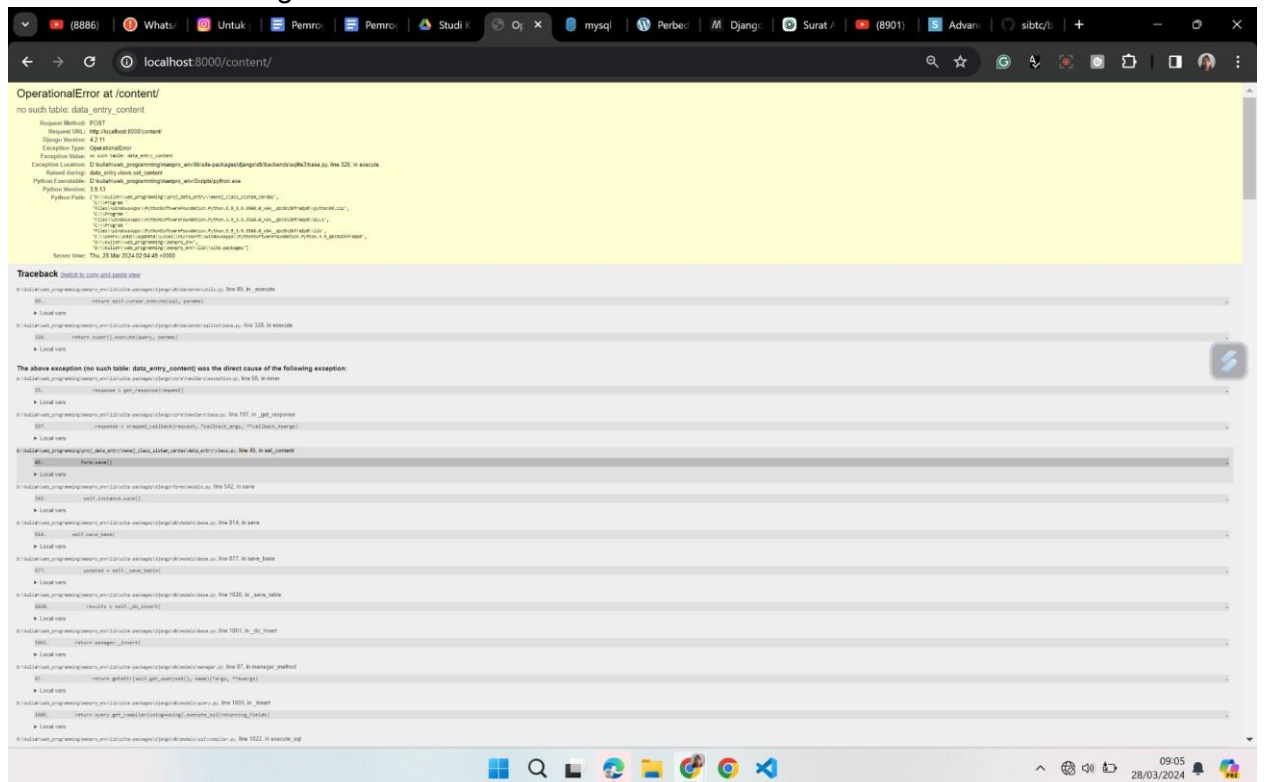
Pengguna object (1) 

ujicoba create content tanpa migrasi

**Artikel:**Set view: ☐

Create

Yaa..anda ketemu bug ...



Kita coba telusuri di bagian yang di bold ya...

The above exception (no such table: data\_entry\_content) was the direct cause of the following exception:

D:\kuliah\web\_programming\manpro\_env\lib\site-packages\django\core\handlers\exception.py, line 55, in inner

```
55.         response = get_response(request)
```

► Local vars

D:\kuliah\web\_programming\manpro\_env\lib\site-packages\django\core\handlers\base.py, line 197, in \_get\_response

```
197.         response = wrapped_callback(request, *callback_args, **callback_kwargs)
```

► Local vars

D:\kuliah\web\_programming\proj\_data\_entry\menej\_class\_sistem\_cerdas\data\_entry\views.py, line 45, in set\_content

```
45.         form.save()
```

▼ Local vars

Variable	Value
form	<ContentForm bound=True, valid=True, fields=(author;artikel;set_view)>
request	<WSGIRequest: POST '/content/'>

D:\kuliah\web\_programming\manpro\_env\lib\site-packages\django\forms\models.py, line 542, in save

Pesannya ternyata adalah no such table.

Jika kita cek di terminal, maka pesan http response yang dikirimkan kodenya 500.

```
return super().execute(query, params)
django.db.utils.OperationalError: no such table: data_entry_content
[28/Mar/2024 09:04:49] "POST /content/ HTTP/1.1" 500 150161
```

Cari tahu ya apa maksud dari kode 500 dari http respon yang diberikan dan tuliskan di laporan.

Kode status HTTP 500 menunjukkan bahwa terjadi kesalahan internal pada server yang mengakibatkan server tidak dapat memenuhi permintaan yang diajukan oleh klien (browser atau aplikasi yang membuat permintaan). Ini biasanya disebabkan oleh masalah konfigurasi server, bug dalam kode aplikasi server, atau masalah sumber daya server yang tidak mencukupi.

Penyebabnya adalah karena pembuatan class Content belum diimplementasikan ke database, sehingga muncul pesan no such table. Mari kita lakukan sedikit langkah perbaikan dengan memanggil fungsi makemigrations dan migrate. Anda masih ingetkan fungsinya?

6. Lakukan seperti di praktikum 4 ya

```
(manpro_env) C:\Users\HEISENBERG\Documents\sem4\Pemrograman web\pbw-hari5\project_data_entry\menej_class_sistem_cerdas>python manage.py makemigrations
Migrations for 'data_entry':
  data_entry\migrations\0002_content.py
  - Create model Content

(manpro_env) C:\Users\HEISENBERG\Documents\sem4\Pemrograman web\pbw-hari5\project_data_entry\menej_class_sistem_cerdas>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, data_entry, sessions
Running migrations:
  Applying data_entry.0002_content... OK
```

7. Sekarang lakukan ujicoba lagi apakah sudah berhasil. Selamat jika anda sudah berhasil melakukannya

8. Berikutnya adalah menampilkan obyek Pengguna berupa email pengguna. Lakukan dengan cara mengedit class Pengguna menjadi seperti berikut

```

9 class Pengguna (models.Model):
10     email = models.EmailField()
11     password = models.CharField(max_length=100)
12     address_1 = models.TextField()
13     address_2 = models.TextField(null=True, blank = True)
14     city = models.CharField(max_length=20, help_text='Enter your city')
15     state= models.TextField()
16     zip_code= models.CharField(max_length = 7)
17     tanggal_join = models.DateField(auto_now = True)
18     def __str__(self):
19         return f'{self.email}'

```

9. Panggil kembali fungsi runserver dan lakukan ujicoba



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/content/'. The page contains a form with the following elements:

- Author:** A dropdown menu with 'vitoo0141@gmail.com' selected.
- Artikel:** A text area containing the text 'testtt setelah migrasi'.
- Set view:** A checkbox that is checked.
- Create:** A button at the bottom of the form.

Setelah klik create sudah tidak error

#### Referensi

<https://virtualenv.pypa.io/en/latest/installation.html>

<https://getbootstrap.com/docs/4.6/components/alerts/> <https://pypi.org/project/crispy-bootstrap4/>