Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM : 223303030213
Nama : Yahya Septian Siregar
Peran : Ketua Tim

NIM : 223303030229
Nama : Nurul Safira
Peran : Anggota Tim

Sistem Informasi Resep Masakan

1. REST API

a. Framework yang digunakan : Django Rest Framework
b. Model
- Library yang digunakan :

```python
import sys
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.core.files.uploadedfile import InMemoryUploadedFile
from datetime import datetime, timedelta
from PIL import Image
from io import BytesIO
from django.utils import timezone
```

- MyUser

```python
class MyUser(AbstractUser):
    is_user = models.BooleanField(default= False)
    is_admin = models.BooleanField(default= False)
    def __str__(self):
        return str(self.username)
```

Model ini berguna untuk membuat penanda untuk user, apakah dia seorang user atau admin, tapi setelah presentasi kami memutuskan akan menghapus is_admin, karna beresiko di bajak seorang user dengan mendaftar sebagai admin.

- StatusModel

```python
class StatusModel(models.Model):
    pilihan_status = (
        ('Aktif', 'Aktif'),
        ('Tidak Aktif', 'Tidak Aktif')
    )

    nama = models.CharField(max_length=20, null=False, blank= False)
    deskripsi = models.TextField(blank=True, null=True)
    status = models.CharField(max_length= 15, choices= pilihan_status, default= 'Aktif')
    user_create = models.ForeignKey(MyUser, related_name='user_create_status_model', blank= True, null= True, on_delete=models.SET_NULL)
    user_update = models.ForeignKey(MyUser, related_name='user_update_status_model', blank= True, null= True, on_delete=models.SET_NULL)
    data_created = models.DateTimeField(auto_now_add= True)
    data_last_update = models.DateTimeField(auto_now= True)

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

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

Model ini berguna sebagai pengisi data untuk tabel tabel lainnya, yakni sebagai penanda aktif atau tidak nya sebuah model tersebut.

- Kategori

```
class Kategori(models.Model):
    nama = models.CharField(max_length= 170)
    status = models.ForeignKey(StatusModel, related_name='status_category', default= StatusModel.objects.first().pk, on_delete=models.PROT
    user_create = models.ForeignKey(MyUser, related_name='user_create_category', blank= True, null= True, on_delete=models.SET_NULL)
    user_update = models.ForeignKey(MyUser, related_name='user_update_category', blank= True, null= True, on_delete=models.SET_NULL)
    data_created = models.DateTimeField(auto_now_add= True)
    data_last_update = models.DateTimeField(auto_now= True)
```

Model ini sama seperti status model, tapi bedanya model ini akan berguna untuk filtering yang akan dilakukan di aplikasi nantinya, mencari berdasarkan kategori

- Jenis

```
class Jenis(models.Model):
    nama = models.CharField(max_length= 130)
    status = models.ForeignKey(StatusModel, related_name='status_jenis', default= StatusModel.objects.first().pk, on_delete=models.PROTECT
    user_create = models.ForeignKey(MyUser, related_name='user_create_jenis', blank= True, null= True, on_delete=models.SET_NULL)
    user_update = models.ForeignKey(MyUser, related_name='user_update_jenis', blank= True, null= True, on_delete=models.SET_NULL)
    data_created = models.DateTimeField(auto_now_add= True)
    data_last_update = models.DateTimeField(auto_now= True)

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

Model ini mirip sekali dengan kategori, yang dimana berguna sebagai filtering di masa mendatang.

- Fungsi Compress_Image

```
def compress_image(image, filename):
    curr_datetime = datetime.now().strftime('%y%m%d %H%M%S')
    im = Image.open(image)
    if im.mode != 'RGB':
        im = im.convert('RGB')
    im_io = BytesIO()
    im.save(im_io, 'jpeg', quality = 50, optimize = True)
    im.seek(0)
    new_image = InMemoryUploadedFile(im_io,'ImageField', '%' + '-' +str(filename) + str(curr_datetime) + '.jpg', 'image/jpeg', sys.getsize
    return new_image
```

Disini kami menambahkan fungsi compress_image, yang dimana berguna untuk mengkompress gambar yang dimasukan melalui django admin.

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

- DetailResepMasakan

```python
class DetailResepMasakan(models.Model):
    nama = models.CharField(max_length= 200, null= False, blank= False)
    deskripsi = models.TextField(null= True, blank=True)
    bahan = models.TextField(null=False, blank=False)
    cara_buat = models.TextField(null=False, blank= True)
    gambar = models.ImageField(default= None, upload_to= 'gambar/resep_masakan', blank= True, null= True)
    kategori = models.ForeignKey(Kategori, related_name='kategori_detail_resep_masakan', on_delete=models.PROTECT)
    jenis = models.ForeignKey(Jenis, related_name='jenis_detail_resep_masakan', on_delete=models.PROTECT)
    status = models.ForeignKey(StatusModel, related_name='status_detail_resep', default=StatusModel.objects.first().pk, on_delete=models.PROTECT)
    user_create = models.ForeignKey(MyUser, related_name='user_create_detail_resep', blank= True, null= True, on_delete=models.SET_NULL)
    user_update = models.ForeignKey(MyUser, related_name='user_update_detail_resep', blank= True, null= True, on_delete=models.SET_NULL)

    def save(self, force_insert= True, force_update= True, using= None , update_fields= None , *args, **kwargs):
        if self.id:
            try :
                this = DetailResepMasakan.objects.get(id =self.id)
                if this.gambar != self.gambar:
                    var_gambar = self.gambar
                    self.gambar = compress_image(var_gambar, 'menu')
                    this.gambar.delete()

            except: pass
            super(DetailResepMasakan, self).save(*args, **kwargs)

        else :
            if self.gambar :
                var_gambar = self.gambar
                self.gambar = compress_image (var_gambar, 'menu')
            super(DetailResepMasakan, self).save(*args, **kwargs)

    def __str__(self):
        return str(self.nama) + ' | ' + str(self.kategori) + ' | ' + str(self.jenis)
```

Model ini berguna untuk menyimpan data detail resep masakan, disini banyak atribut yang terhubung ke tabel lain, seperti kategori, jenis, status, user_create.. Disini juga kita lihat fungsi save, ketika dia dipanggil, maka akan dilakukan pergantian nama dan langsung dimasukan ke dalam folder yang sudah ditentukan dari settings.py

- Profil

```python
class Profil(models.Model):
    user = models.OneToOneField(MyUser, on_delete=models.PROTECT, related_name='user_profile')
    gambar = models.ImageField(default=None, upload_to='gambar/profile')
    bio = models.TextField()
    status = models.ForeignKey(StatusModel, related_name='status_profile', default= StatusModel.objects.first().pk, on_delete=models.PROTECT)
    user_create = models.ForeignKey(MyUser, related_name='user_create_profile', blank= True, null= True, on_delete=models.SET_NULL)
    user_update = models.ForeignKey(MyUser, related_name='user_update_profile', blank= True, null= True, on_delete=models.SET_NULL)
    data_created = models.DateTimeField(auto_now_add= True)
    data_last_update = models.DateTimeField(auto_now= True)

    def save(self, force_insert= False, force_update= False, using= None , update_fields= None , *args, **kwargs):
        if self.id:
            try :
                this = Profil.objects.get(id =self.id)
                if this.gambar != self.gambar:
                    var_gambar = self.gambar
                    self.gambar = compress_image(var_gambar, 'profile')
                    this.gambar.delete()

            except: pass
            super(Profil, self).save(*args, **kwargs)

        else :
            if self.gambar :
                var_gambar = self.gambar
                self.gambar = compress_image (var_gambar, 'profile')
            super(Profil, self).save(*args, **kwargs)

    def __str__(self):
        return str(self.user)
```

Model ini berguna untuk menyimpan data user, yang dimana ketika user membuat akun pertama kali maka model profil ini akan otomatis terisi, hal itu juga dibantu oleh signals.py

- MenuUtama

```python
class MenuUtama(models.Model):
    nama = models.CharField(max_length=100, null=False, blank=False)
    informasi = models.TextField(blank=True, null=True)
    rekomendasi = models.ManyToManyField(DetailResepMasakan, related_name='menu_utama')
    user_create = models.ForeignKey(MyUser, related_name='user_create_menu_utama', blank=True, null=True, on_delete=models.SET_NULL)
    user_update = models.ForeignKey(MyUser, related_name='user_update_menu_utama', blank=True, null=True, on_delete=models.SET_NULL)
    data_created = models.DateTimeField(auto_now_add=True)
    data_last_update = models.DateTimeField(auto_now=True)

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

Model ini berguna untuk memasukan data data detail resep masakan ke dalam 1 data untuk nantinya ditampilkan sebagai menu rekomendasi yang bisa diganti ganti.

c. Controller

- Library yang digunakan :

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status, filters
from rest_framework import permissions
from rm_app.models import (
    MyUser, DetailResepMasakan, MenuUtama, StatusModel, Kategori, Jenis, Profil
)
from api.serializers import (
    DetailResepMasakanSerializers, MenuUtamaSerializers, RegisterMyUserSerializer, LoginSerializer
)
from rest_framework import generics
from rest_framework.authtoken.models import Token
from django.contrib.auth import login as rm_login, logout as rm_logout
from django.http import HttpResponse, JsonResponse
from rest_framework.authentication import SessionAuthentication, BasicAuthentication, TokenAuthentication
from rest_framework.permissions import IsAuthenticated, AllowAny
from .paginators import CustomPagination
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework.permissions import BasePermission
from rest_framework.exceptions import ValidationError
```

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

- DetailResepMasakanListApiView

```python
class DetailResepMasakanListApiView(APIView):
    authentication_classes = [CsrfExemptSessionAuthentication, TokenAuthentication]
    permission_classes = [IsAdminOrReadOnly]

    def get(self, request, *args, **kwargs):
        detail_resep = DetailResepMasakan.objects.all()
        serializers = DetailResepMasakanSerializers(detail_resep, many=True)
        return Response(serializers.data, status=status.HTTP_200_OK)

    def post(self, request, *args, **kwargs):
        if isinstance(request.data, list):
            # Handle list of items
            responses = []
            for item in request.data:
                serializer = DetailResepMasakanSerializers(data=item)
                if serializer.is_valid():
                    serializer.save()
                    responses.append({
                        'status': status.HTTP_201_CREATED,
                        'message': 'Data created successfully...',
                        'data': serializer.data
                    })
                else:
                    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
            return Response(responses, status=status.HTTP_201_CREATED)

        elif isinstance(request.data, dict):
            # Handle single item
            serializer = DetailResepMasakanSerializers(data=request.data)
            if serializer.is_valid():
                serializer.save()
                response = {
                    'status': status.HTTP_201_CREATED,
                    'message': 'Data created successfully...',
                    'data': serializer.data
                }
                return Response(response, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

        else:
            raise ValidationError({"message": "Expected a list or a dictionary."})
```

Disini kegunaan controller ini untuk menerima request, ketika dia mendapat get, dia memberikan sesuai yang kita tetapkan, begitu juga dengan post. Dan jika terjadi error, maka akan diberikan response error dan detail nya.

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

- MenuUtamaListApi

```
78  class MenuUtamaListApiView(APIView):
79      def get(self, request, *args, **kwargs):
80          menu_utama = MenuUtama.objects.all()
81          serializers = MenuUtamaSerializers(menu_utama, many = True)
82          return Response(serializers.data, status = status.HTTP_200_OK)
83
84      def post(self, request, *args, **kwargs):
85          data = {
86              'nama' : request.data.get('nama'),
87              'informasi' : request.data.get ('informasi'),
88          }
89          rekomendasi_ids = request.data.getlist('rekomendasi')
90          if rekomendasi_ids:
91              data['rekomendasi'] = rekomendasi_ids
92
93          serializer = MenuUtamaSerializers(data = data)
94          if serializer.is_valid():
95              serializer.save()
96              response = {
97                  'status' : status.HTTP_201_CREATED,
98                  'message' : 'Data created successfully...',
99                  'data' : serializer.data
100             }
101             return Response(response, status = status.HTTP_201_CREATED)
102
103         return Response(serializer.errors, status= status.HTTP_400_BAD_REQUEST)
104
```

Disini controller menu utama juga miurip dengan detail, memliki fungsi get dan post yang dimana masing-masing memiliki response yang sudah ditetapkan.

- DetailResepMasakanApiView

```
107 class DetailResepMasakanApiView(APIView):
108     def get_object(self, id):
109         try:
110             return DetailResepMasakan.objects.get(id = id)
111         except DetailResepMasakan.DoesNotExist:
112             return None
113
114     def get(self, request, id, *args, **kwargs):
115         detail_resep_masakan_instance = self.get_object(id)
116         if not detail_resep_masakan_instance:
117             return Response(
118                 {
119                     'status' : status.HTTP_400_BAD_REQUEST,
120                     'message' : 'Data does not exists...',
121                     'data' : {}
```

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

```
                'data' : {}
            }, status= status.HTTP_400_BAD_REQUEST
        )

    serializer = DetailResepMasakanSerializers(detail_resep_masakan_instance)
    response = {
        'status' : status.HTTP_200_OK,
        'message' : 'Data retrieve successfully...',
        'data' : serializer.data
    }
    return Response(response, status = status.HTTP_200_OK)

def put(self, request, id, *args, **kwargs):
    detail_resep_masakan_instance = self.get_object(id)
    if not detail_resep_masakan_instance:
        return Response(
            {
                'status' : status.HTTP_400_BAD_REQUEST,
                'message' : 'Data does not exists....',
                'data' : {}
            }, status = status.HTTP_400_BAD_REQUEST
        )

    data = {
        'nama' : request.data.get('nama'),
        'deskripsi' : request.data.get ('deskripsi'),
        'bahan' : request.data.get('bahan'),
        'cara_buat' : request.data.get('cara_buat'),
        'kategori' : request.data.get('kategori'),
        'jenis' : request.data.get('jenis'),
    }
    serializer = DetailResepMasakanSerializers(instance = detail_resep_masakan_instance, data = data, partial = True)
    if serializer.is_valid():
        serializer.save()
```

```
            response = {
                'status' : status.HTTP_200_OK,
                'message' : 'Data created successfully...',
                'data' : serializer.data
            }
            return Response(response, status = status.HTTP_200_OK)
        return Response(serializer.errors, status = status.HTTP_400_BAD_REQUEST)

    def delete(self, request, id, *args, **kwargs):
        detail_resep_masakan_instance = self.get_object(id)
        if not detail_resep_masakan_instance:
            return Response(
                {
                    'status' : status.HTTP_400_BAD_REQUEST,
                    'message' : 'Data does not exists....',
                    'data' : {}
                }, status = status.HTTP_400_BAD_REQUEST
            )

        detail_resep_masakan_instance.delete()
        response = {
            'status' : status.HTTP_200_OK,
            'message' : 'Data deleted successfully....'
        }
        return Response(response, status= status.HTTP_200_OK)
```

Pada controller DetailResepMasakanApiView terdapat fungsi get_object yang mengambil data per id, put untuk mengupdate data, dan delete untuk mendelete data.

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

- MenuUtamaApiView

```
183  class MenuUtamaApiView(APIView):
184      def get_object(self, id):
185          try:
186              return MenuUtama.objects.get(id = id)
187          except MenuUtama.DoesNotExist:
188              return None
189
190      def get(self, request, id, *args, **kwargs):
191          menu_utama_instance = self.get_object(id)
192          if not menu_utama_instance:
193              return Response(
194                  {
195                      'status' : status.HTTP_400_BAD_REQUEST,
196                      'message' : 'Data does not exists...',
197                      'data' : {}
198                  }, status= status.HTTP_400_BAD_REQUEST
199              )
200
201          serializer = MenuUtamaSerializers(menu_utama_instance)
202          response = {
203              'status' : status.HTTP_200_OK,
204              'message' : 'Data retrieve successfully...',
205              'data' : serializer.data
206          }
207          return Response(response, status = status.HTTP_200_OK)
208
209      def put(self, request, id, *args, **kwargs):
210          menu_utama_instance = self.get_object(id)
211          if not menu_utama_instance:
212              return Response(
213                  {
214                      'status' : status.HTTP_400_BAD_REQUEST,
215                      'message' : 'Data does not exists....',
216                      'data' : {}
217                  }, status = status.HTTP_400_BAD_REQUEST
```

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

```
        )

        data = {
            'nama' : request.data.get('nama'),
            'informasi' : request.data.get ('informasi'),
        }
        rekomendasi_ids = request.data.get('rekomendasi')
        if rekomendasi_ids:
            if isinstance(rekomendasi_ids, str):
                rekomendasi_ids = rekomendasi_ids.split(',')
            data['rekomendasi'] = rekomendasi_ids

        serializer = MenuUtamaSerializers(instance = menu_utama_instance, data = data, partial = True)
        if serializer.is_valid():
            serializer.save()
            response = {
                'status' : status.HTTP_200_OK,
                'message' : 'Data created successfully...',
                'data' : serializer.data
            }
            return Response(response, status = status.HTTP_200_OK)
        return Response(serializer.errors, status = status.HTTP_400_BAD_REQUEST)

    def delete(self, request, id, *args, **kwargs):
        menu_utama_instance = self.get_object(id)
        if not menu_utama_instance:
            return Response(
                {
                    'status' : status.HTTP_400_BAD_REQUEST,
                    'message' : 'Data does not exists....',
                    'data' : {}
                }, status = status.HTTP_400_BAD_REQUEST
            )

        menu_utama_instance.delete()
```

```
            )

        menu_utama_instance.delete()
        response = {
            'status' : status.HTTP_200_OK,
            'message' : 'Data deleted successfully....'
        }
        return Response(response, status= status.HTTP_200_OK)
```

Pada controller MenuUtamaApiView, get_object, put, delete, sama persis seperti detail list api view.

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

- RegisterUserAPIView

```
259  class RegisterUserAPIView(APIView):
260      serializers_class = RegisterMyUserSerializer
261      permission_classes = [AllowAny]
262
263      def post(self, request, format = None):
264          serializers = self.serializers_class(data = request.data)
265          if serializers.is_valid():
266              serializers.save()
267              response_data = {
268                  'status' : status.HTTP_201_CREATED,
269                  'message' : 'Selamat anda telah terdaftar...',
270                  'data' : serializers.data,
271              }
272              return Response(response_data, status= status.HTTP_201_CREATED)
273          return Response({
274              'status' : status.HTTP_400_BAD_REQUEST,
275              'data' : serializers.errors
276          }, status= status.HTTP_400_BAD_REQUEST)
```

Controller ini berguna untuk meregistrasi akun, disini dapat kita lihat request yang tersedia adalah post, jadi jika di web/insomnia kita mengetik get tidak akan berhasil.

- LoginView

```
278  class LoginView(APIView):
279      serializer_class = LoginSerializer
280      permission_classes = [AllowAny]
281
282      def post(self, request):
283          serializer = LoginSerializer(data = request.data)
284          serializer.is_valid(raise_exception= True)
285          user = serializer.validated_data['user']
286          rm_login(request, user)
287          token, created = Token.objects.get_or_create(user = user)
288          return JsonResponse({
289              'status' : 200,
290              'message' : 'Selamat anda berhasil masuk...',
291              'data' : {
292                  'token' : token.key,
293                  'id' : user.id,
294                  'first_name' : user.first_name,
295                  'last_name' : user.last_name,
296                  'email': user.email,
297                  'is_active' : user.is_active,
298                  'is_admin' : user.is_admin,
299              }
300          })
```

Disini terdapat controller login, yang memvalidasi data, jika benar akan dibuatkan token, kemudian dikirim response berhasil, dan menampilkan isi data yang ada pada user.

- LogOutApiView

```
302  class LogoutAPIView(APIView):
303      permission_classes = [AllowAny]
304      def get(self, request):
305          rm_logout(request)
306          # Return success response
307          return Response({"detail": "Logout Successful"}, status=status.HTTP_200_OK)
```

Disini terdapat controller logout yang berguna untuk mengeluarkan serta menghapus token yang menempel pada webservice.

- DetailresepMasakanView

```
309  class DetailresepMasakanView(APIView):
310      authentication_class = [SessionAuthentication, BasicAuthentication]
311      permission_classes = [IsAuthenticated]
312
313      def get(self, request, *args, **kwargs):
314          detail_resep = DetailResepMasakan.objects.select_related('status').\
315              filter(status = StatusModel.objects.first())
316          serializer = DetailResepMasakanSerializers(detail_resep, many = True,)
317          response = {
318              'status' : status.HTTP_200_OK,
319              'message' : 'Pembacaan seluruh data berhasil.....',
320              'user' : str(request.user),
321              'auth' : str(request.auth),
322              'data' : serializer.data,
323          }
324          return Response(response, status= status.HTTP_200_OK)
```

Controller ini berguna untuk menampilkan data detail resep makanan yang berstatus aktif saja.

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

- MenuUtamaView

```
326  class MenuUtamaView(APIView):
327      authentication_class = [SessionAuthentication, BasicAuthentication]
328      permission_classes = [IsAuthenticated]
329
330      def get(self, request, *args, **kwargs):
331          menu_utama = MenuUtama.objects.select_related('status').\
332              filter(status = StatusModel.objects.first())
333          serializer = MenuUtamaSerializers(menu_utama, many = True,)
334          response = {
335              'status' : status.HTTP_200_OK,
336              'message' : 'Pembacaan seluruh data berhasil.....',
337              'user' : str(request.user),
338              'auth' : str(request.auth),
339              'data' : serializer.data,
340          }
341          return Response(response, status= status.HTTP_200_OK)
```

Sama seperti DetailresepMasakan yang dimana hanya menampilkan data menu yang berstatus aktif.

- DetailResepMasakanFilter

```
343  class DetailResepMasakanFilterApi(generics.ListAPIView):
344      queryset = DetailResepMasakan.objects.all()
345      serializer_class = DetailResepMasakanSerializers
346      pagination_class = CustomPagination
347      permission_classes = [permissions.IsAuthenticated]
348      filter_backends = [DjangoFilterBackend, filters.OrderingFilter]
349      filterset_fields = ['kategori__nama', 'jenis__nama',]
350      ordering_fields = ['created_on']
```

Kegunaan controller ini berguna untuk menampilkan data berdasarkan kategori dan jenis.

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

d. Routing (url)

ProjectRM/urls.py

```
The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from rm_app import views
from api import views
from django.conf import settings
from django.conf.urls.static import static


urlpatterns = [
    path('super-admin/', admin.site.urls),
    path('', include('api.urls', namespace = 'api')),
    # path('', include('api.urls', namespace= 'api')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root= settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL, document_root= settings.STATIC_ROOT)

```

Urls ini berguna untuk masuk ke dalam admin site django, dan url memasukan gambar dari admin ke dalam folder

api/urls.py

```
from django.urls import path, include
from api import views
from rest_framework.urlpatterns import format_suffix_patterns
from .views import (
    DetailResepMasakanApiView, MenuUtamaApiView, RegisterUserAPIView, LoginView, DetailresepMasakanView, LogoutAPIView,
)

app_name = 'api'
urlpatterns = [
    path('api/detail_resep_masakan', views.DetailResepMasakanListApiView.as_view()),
    path('api/menu_utama', views.MenuUtamaListApiView.as_view()),
    path('api/detail_resep_masakan/<int:id>', views.DetailResepMasakanApiView.as_view()),
    path('api/menu_utama/<int:id>', views.MenuUtamaApiView.as_view()),
    path('api/register', RegisterUserAPIView.as_view()),
    path('api/login', LoginView.as_view()),
    path('api/logout', views.LogoutAPIView.as_view()),
    path('api/detail_resep', views.DetailresepMasakanView.as_view()),
    path('api/detail_menu_filter/', views.DetailResepMasakanFilterApi.as_view()),
]
```

Urls ini berguna untuk masuk ke dalam rest api, yang dimana setiap alamat berbeda – beda sesuai controller yang sudah ada.

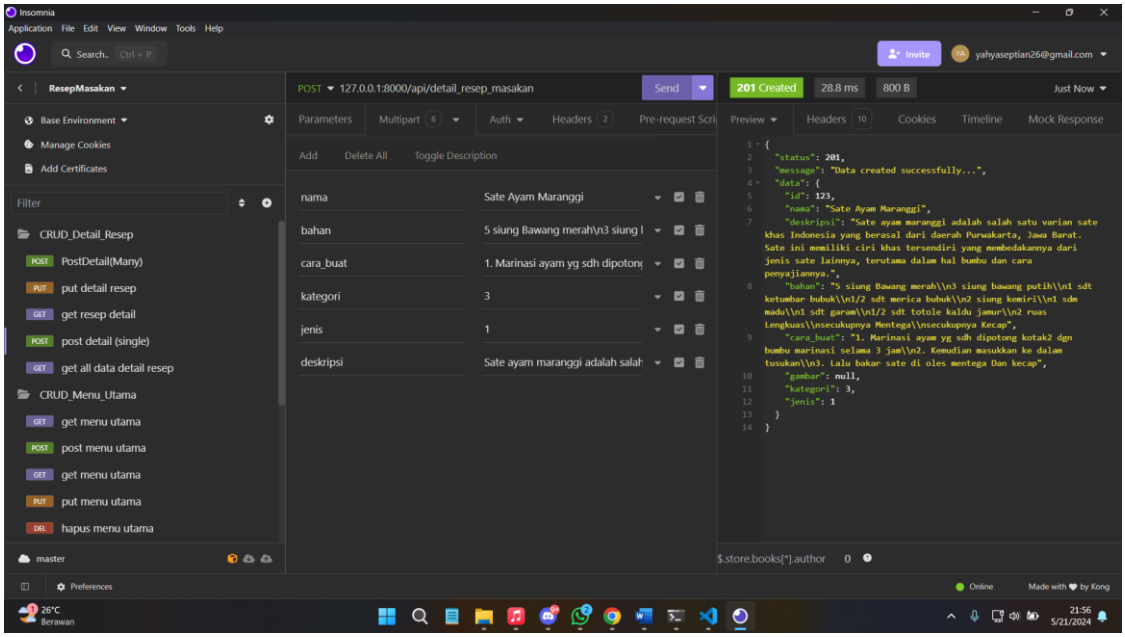
Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

e. Middleware
   Tidak Ada

f. Input - Proses/Transaksi - Output REST API (service GET, POST, PUT dan DELETE)
   Screenshot dari Input - Proses/Transaksi - Output REST API yang anda bangun dan uraikan narasi dari Input - Proses/Transaksi - Output REST API yang anda bangun. Lalu tampilkan hasil screenshot dari REST API yang dibangun menggunakan aplikasi Postman atau Insomnia *(pilih salah satu)*.

   Post data DetailResepMasakan



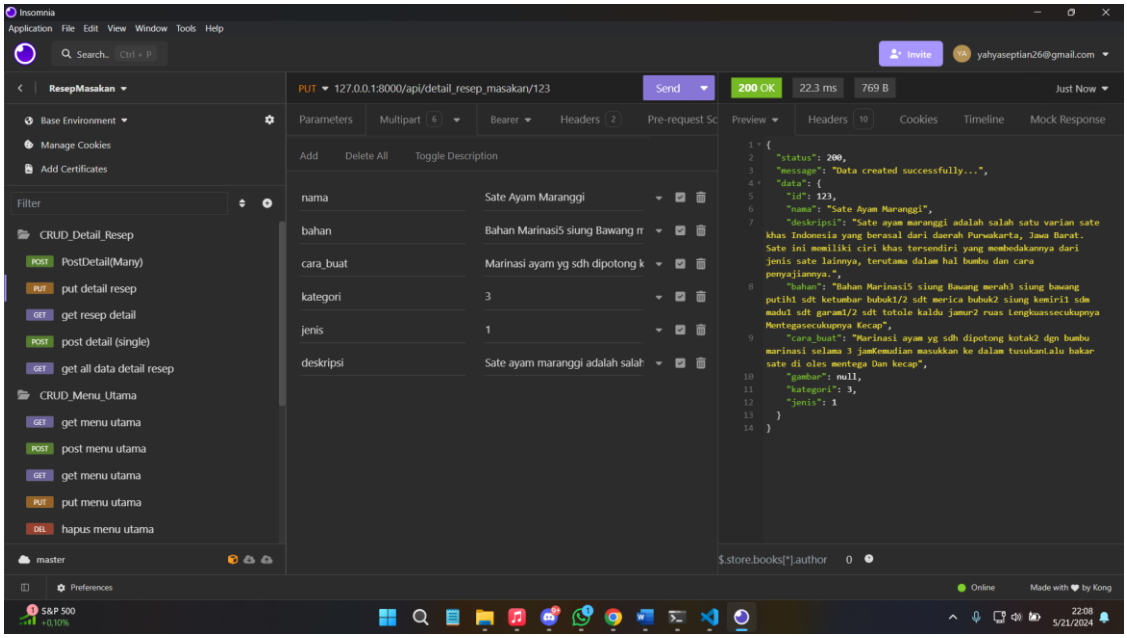
Disini kita memasukan data detailresepmasakan dengan method post, yang dimana dapat kita lihat terdapat bahan dan cara buat ada ”\n” yang nantinya bakal kita hilangkan di dalam Put (update).

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM : 223303030213
Nama : Yahya Septian Siregar
Peran : Ketua Tim

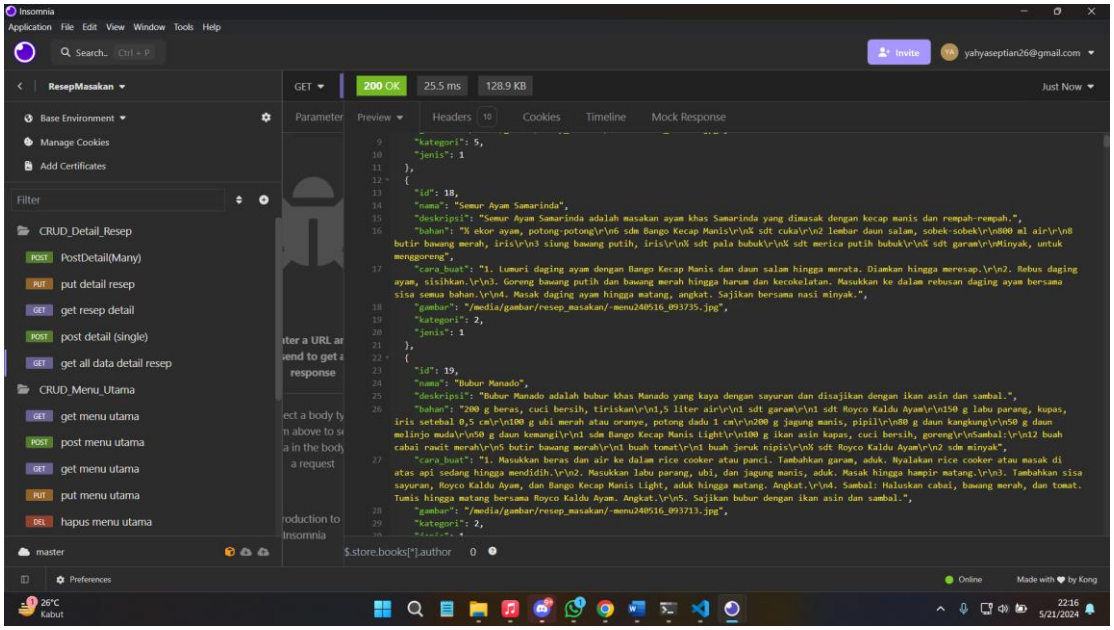
NIM : 223303030229
Nama : Nurul Safira
Peran : Anggota Tim

Put data DetailResepMasakan



Disini kita telah melakukan update data ”\n”, pada bahan dan cara buat. Disini kita lihat gambar belum dimasukan, karna kami memasukan data foto dari django-admin nya langsung.

Get All data DetailResepMasakan

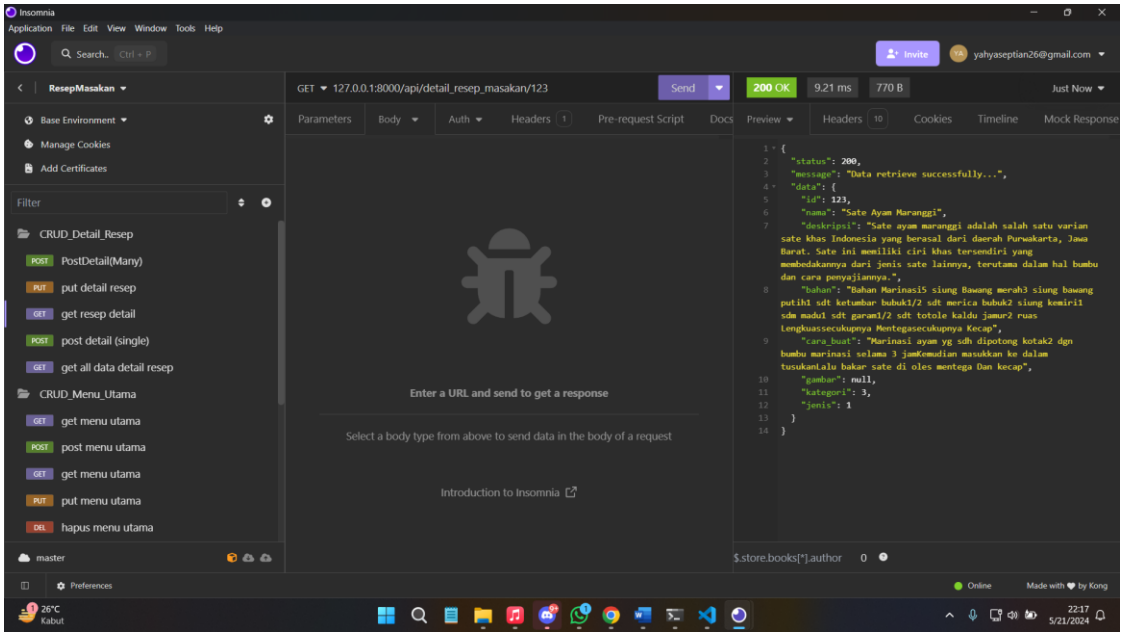


Disini ada metode getalldata, yang berguna mengambil semua data yang ada pada tabel detailresepmasakan

Nama TIM : Yahya Septian Siregar, Nurul Safira

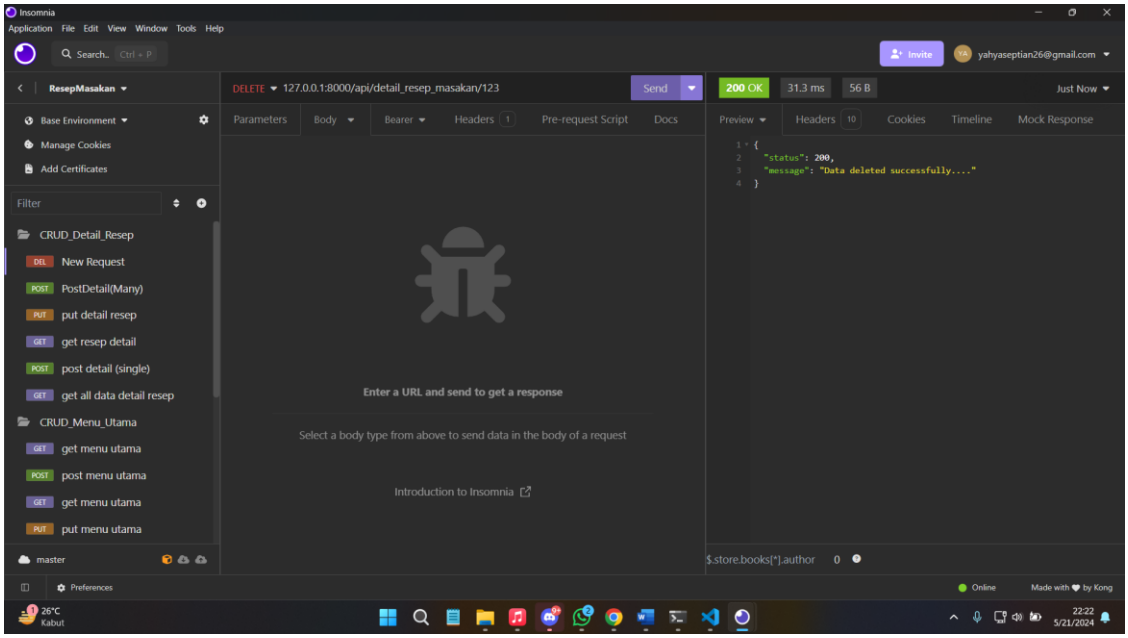
NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim

Get Single Data berdasarkan Id



Disini kita lihat kita mengambil data dengan id ke-123.

Delete data berdasarkan Id



Disini kita menghapus data berdasarkan ini, kit lihat status 200 yang berarti berhasil.

Catatan:

- *REST API tidak harus 1 (satu) halaman.*
- *Point nomor f pada REST API disesuaikan dengan topik yang telah disetujui oleh Dosen Pengampu.*
- *Format penamaan file ini : NIM_KETUA_TIM_NAMA_KETUA_TIM_ NAMA_MATA_KULIAH_KELAS, contoh : 2033030300007_Andi_Wijaya_PM_TI_4_PAGI_A.*
- *Wajib mengikuti format penaaman file diatas untuk memudahkan proses penilaian.*

Nama TIM : Yahya Septian Siregar, Nurul Safira

NIM	: 223303030213	NIM	: 223303030229
Nama	: Yahya Septian Siregar	Nama	: Nurul Safira
Peran	: Ketua Tim	Peran	: Anggota Tim