```python
""" core/api_views.py """

from rest_framework.generics import ListAPIView, RetrieveUpdateDestroyAPIView
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework.filters import SearchFilter
from rest_framework.pagination import LimitOffsetPagination
from rest_framework.permissions import IsAuthenticatedOrReadOnly
from rest_framework.authentication import SessionAuthentication, BasicAuthentication, TokenAuthentication
from rest_framework import authentication
from rest_framework import exceptions


from core.serializers import RestaurantSerializer
from core.models import Restaurant

class RestaurantsPagination(LimitOffsetPagination):
    default_limit = 10
    max_limit = 100

class RestaurantList(ListAPIView):
    queryset = Restaurant.objects.all()
    serializer_class = RestaurantSerializer
    filter_backends = (DjangoFilterBackend, SearchFilter)
    filter_fields = ('category',)
    search_fields = ('name', 'description', 'category', 'address')
    pagination_class = RestaurantsPagination

    def get_queryset(self):
        queryset = Restaurant.objects.all()
        return queryset

class RestaurantRetrieveUpdateDestroy(RetrieveUpdateDestroyAPIView):
    authentication_classes = [SessionAuthentication, BasicAuthentication, TokenAuthentication]
    permission_classes = [IsAuthenticatedOrReadOnly]
    queryset = Restaurant.objects.all()
    lookup_field = 'id'
    serializer_class = RestaurantSerializer

    def update(self, request, *args, **kwargs):
        response = super().update(request, *args, **kwargs)
        if response.status_code == 200:
            from django.core.cache import cache
            restaurant = response.data
            cache.set('restaurant_data_{}'.format(restaurant['id']), restaurant)
        return response

```

```python
""" core/models.py """

from django.db import models

class Restaurant(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=200)
    description = models.TextField()
    category = models.CharField(max_length=50)
    address = models.TextField()
    phone = models.CharField(max_length=30)
    website = models.URLField(max_length=200)
    created_at = models.DateTimeField('Created at',
    auto_now_add=True)

""" core/serializers.py """

from rest_framework import serializers

from core.models import Restaurant

class RestaurantSerializer(serializers.ModelSerializer):
    class Meta:
            model = Restaurant
            fields = (
                'id', 'name', 'description', 'category', 'address',
                'phone', 'website'
            )
```

```python
""" core/tests.py """

from rest_framework.test import APITestCase
from core.models import Restaurant

from django.contrib.auth.models import User

class AuthenticationTestCase(APITestCase):
    def setUp(self):
        user = User.objects.create(username='admin')
        user.set_password('123456')
        user.save()
        self.login_url = '/api/v1/rest-auth/login/'

    def test_login_sucess(self):
        response = self.client.post(self.login_url, {'username': 'admin','password': '123456'})
        self.assertEqual(response.status_code, 200)

    def test_login_fail(self):
        response = self.client.post(self.login_url, {'username': 'other','password': 'abcdef'})
        self.assertEqual(response.status_code, 400)

    def test_logout(self):
        self.client.login(username='admin',password='123456')
        response = self.client.post('/api/v1/rest-auth/logout/')
        self.assertEqual(response.status_code, 200)

class RestaurantListTestCase(APITestCase):
    def test_list_restaurants(self):
        restaurants_count = Restaurant.objects.count()
        response = self.client.get('/api/v1/restaurants/')
        self.assertIsNone(response.data['next'])
        self.assertIsNone(response.data['previous'])
        self.assertEqual(response.data['count'], restaurants_count)
        self.assertEqual(len(response.data['results']), restaurants_count)

class RestaurantUpdateTestCase(APITestCase):
    def setUp(self):
        restaurant = Restaurant(name='Restaurant 1',
            description='hipster, vegan',
            address='street 11',
            phone='+49 30 0000001')
        restaurant.save()

    def test_update_restaurants(self):
        user = User(username='admin', password='123456')
        user.save()
        self.client.force_authenticate(user=user)
        restaurant = Restaurant.objects.first()
        response = self.client.patch(
            '/api/v1/restaurants/{}/'.format(restaurant.id),
            {
                'name': 'New Restaurant',
                'description': 'Awesome Restaurant',
                'address': 'street 1000',
            },
            format='json',
        )
        updated = Restaurant.objects.get(id=restaurant.id)
        self.assertEqual(updated.name, 'New Restaurant')

    def test_update_restaurants_not_authenticated(self):
        restaurant = Restaurant.objects.first()
        response = self.client.patch(
            '/api/v1/restaurants/{}/'.format(restaurant.id),
            {
                'name': 'New Restaurant',
                'description': 'Awesome Restaurant',
                'address': 'street 1000',
            },
            format='json',
        )

        self.assertEqual(response.status_code, 403)
```

```python
""" restolist/urls.py """

from django.contrib import admin
from django.urls import include, path

import core.api_views

urlpatterns = [
    path('api/v1/restaurants/', core.api_views.RestaurantList.as_view()),
    path('api/v1/restaurants/<int:id>/',
        core.api_views.RestaurantRetrieveUpdateDestroy.as_view()
    ),
    path('api/v1/rest-auth/', include('rest_auth.urls')),
    path('admin/', admin.site.urls),
]


""" restolist/settings.py """

import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

SECRET_KEY = 'v6b_fsjws8pa*z!j14c_@x8t7ax)o$xgt7edj^v+4-_78j9e#a'

DEBUG = True

ALLOWED_HOSTS = []


INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    'rest_framework',
    'rest_framework.authtoken',
    'rest_auth',
    'allauth',
    'allauth.account',
    'rest_auth.registration',
    'django_filters',
    'corsheaders',
    'core'
]
```

```
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.RemoteUserMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

CORS_ORIGIN_WHITELIST = (
    'http://localhost:3000',
)

AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.RemoteUserBackend',
    'allauth.account.auth_backends.AuthenticationBackend'
)

ROOT_URLCONF = 'restolist.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'restolist.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    },
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

STATIC_URL = '/static/'

```