***En XX nos interesa tu perfil profesional. Es por eso que te enviamos esta pequeña prueba como parte del proceso de selección, para seguir conociéndote.***

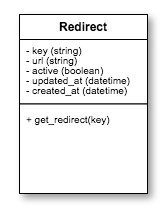
***Te pedimos que la realices y nos envíes este documento por mail. No te llevará mucho tiempo efectuarla.***

***Para la entrega, te recomendamos alojar el proyecto en alguno de los servicios web basados en Git (github.com, gitlab.com, etc)***

***Éxitos!!!***

Utilizando Python3, Django versión 2 escribir código que resuelva los siguientes puntos:

1. Representar el siguiente modelo de datos en Django utilizando MySQL como motor de base de datos.



Consideraciones:

*created\_at: representa la fecha y hora en que se creo por primera vez el registro.*

*updated\_at: representa la fecha y hora de la ultima modificación del registro.*

Ambos valores no deben ser ingresados por el usuario, sino que deben colocarse automáticamente.

1. Crear una signal asociada a las actualizaciones del modelo anterior que tome todos los datos con active=True; y los coloque en una estructura de datos en cache (preferentemente usar memcached y la estructura de datos de deja a criterio del programador)
2. Crear una view que reciba mediante GET HTTP el parámetro key y retorne la URL asociada de acuerdo con el modelo. La respuesta debe ser un Json con la siguiente estructura:

{“key”: “[el valor de la key]”, “url”: “[la url correspondiente]”}

Obtener el valor de la URL desde cache y No desde la DB.

1. Disponibilizar el Admin de Django para poder configurar la aplicación.

***RESOLUCIÓN***

Git: <https://github.com/ZitelliDZ/prueba_tecnica_django>

Mi experiencia durante la prueba técnica:

Al principio, no tuve complicaciones.

Los únicos inconvenientes fueron:

* + Instanciar la memoria caché.
  + Los tests unitarios no se podían ejecutar debido a problemas de configuración.
    - El problema principal fue la imposibilidad de importar recursos, debido a la falla de importación de módulos.
    - Otros problemas en los tests también fueron los rollbacks automáticos.
  + La incapacidad para establecer conexión con la base de datos de pruebas debido a problemas similares a "bind-address" con el contenedor de Docker.
  + Todos los obstáculos fueron superados excepto el último relacionado con la base de datos de pruebas.

Todos los obstáculos fueron alcanzados menos el último de la Base de datos de Prueba.

//README

-El proyecto incluye una carpeta Postman con un archivo para importar.

-Consta de una aplicación principal llamada Core y otra llamada Redirect.

-El archivo docker-compose define 4 contenedores: DB - DBTest - Cache - Backend (Aplicación).

**Getting Started**

* Correr "docker-compose up" en la terminal.
* Una vez creados los contenedores, refrescar el contenedor Backend.
* Correr "docker-compose exec backend sh" en la terminal.
* Realiza las migraciones con "python manage.py migrate".
* Crea un usuario administrador con "python manage.py createsuperuser".
* Correr las pruebas "pytest -W ignore ./redirects/tests/test\_redirects.py -v ".

//.env

A modo de facilitar las variables de entorno.

DEBUG=True  
  
ENGINE\_DB='django.db.backends.mysql'  
NAME\_DB='core'  
USER\_DB='root'  
PASSWORD\_DB='core'  
HOST\_DB='db'  
PORT\_DB='3306'  
  
  
NAME\_TEST\_DB='testcore'  
USER\_TEST\_DB='testcore'  
PASSWORD\_TEST\_DB='testcore'  
HOST\_TEST\_DB='dbtest'  
PORT\_TEST\_DB='3307'  
  
  
ALLOWED\_HOSTS='127.0.0.1'  
  
CORS\_ORIGIN\_ALLOW\_ALL=True  
  
SECRET\_KEY='django-insecure-q4m6jv+-zal#1e867b6k!jos\*3\*zkdlgh#nsa^ff%$u)l-ff\*p'