

CREACIÓN DE UNA API Y UNA INTERFAZ WEB PARA GENERAR UN DASHBOARD A PARTIR DE LA HERRAMIENTA PERCEVAL

Pablo Fernández Salguero

Universidad Rey Juan Carlos

10 de julio de 2017

Índice

- 1 Introducción
- 2 Tecnologías
 - Principales
 - Secundarias
- 3 Desarrollo
- 4 Producto final
 - Versión API
 - Versión WEB
- 5 Conclusiones

Objetivos

Objetivo principal




Crear una API y una interfaz WEB a partir de ella, para la creación de un dashboard, tomando como base la herramienta Perceval.

Objetivos específicos

- 1 Integración con Elasticsearch.
- 2 Integración con GitHub.
- 3 Funcionalidad para el usuario.
- 4 API.
- 5 Interfaz web.

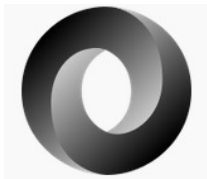
Tecnologías principales

- Perceval 
- GitHub 
- Python 

- Django 
- Elasticsearch 
- Kibana 

Tecnologías secundarias

- JSON

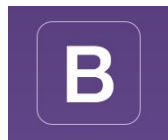


- Requests



- Python Social Auth

- Bootstrap



Iteración 1

Conocimiento de las herramientas

Se sigue el manual creado por GrimoireLab para conocer el entorno Perceval.

- Conocimiento de la herramienta Perceval.
- Creación de un dashboard en Kibana con GrimoireELK, a partir de información obtenida de Git y almacenada en Elasticsearch.
- Almacenamiento y manejo de datos con Elasticsearch. Uso de *p2o.py* y *kidash.py*.
- Script que agrupa los pasos para la creación de un dashboard, a partir de los datos proporcionado por el usuario.

Iteración 2

Primera aproximación con Django

- Aplicación Django para solicitar información al usuario.
- Integración de Elasticsearch con Django.
 - ❶ Uso de la clase *DocType* de *elasticsearch_dsl* para indexar la información en Elasticsearch.
 - ❷ Señales en Django.
 - ❸ Código para almacenar la información.
- Creación de un script para generar un dashboard a partir de la información almacenada en Elasticsearch por la aplicación Django.

Iteración 2

```
36 ...
37 class tareas(models.Model):
38     usuario=models.CharField(max_length=100)
39     repositorio=models.CharField(max_length=100)
40     estado=models.BooleanField(default=False)
41
42
43 def indexing(self):
44     obj = tareasIndex(
45         meta={'type': "doc_type", 'id': self.usuario+"-"+self.repositorio},
46         usuario=self.usuario,
47         estado=self.estado,
48     )
49     obj.save(index='tareas')
50     return obj.to_dict(include_meta=True)
51
```

Figura: Código Django para el almacenamiento de información en Elasticsearch.

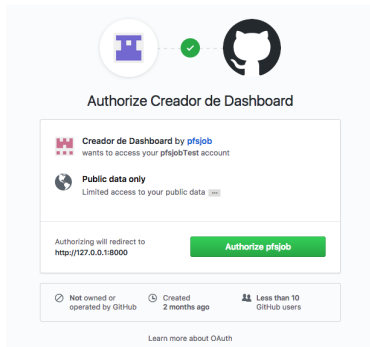
Iteración 3

Integración con GitHub

Se busca la integración del proyecto con GitHub, y para ello:

- **GitHub:**
 - 1 Aplicaciones de terceros en la plataforma.
 - 2 OAuth, protocolo de autenticación.
 - 3 API de GitHub.
- Integración de autenticación a través de GitHub en Django: Python Social Auth.
- Uso de la API de GitHub: obtener el token de usuario.

Iteración 3



(a) Página de autorización de GitHub.

2. Users are redirected back to your site by GitHub

If the user accepts your request, GitHub redirects back to your site with a temporary `code` in a `code` parameter as well as the `state` you provided in the previous step in a `state` parameter. If the states don't match, the request was created by a third party and the process should be aborted.

Exchange this `code` for an access token:

POST `https://github.com/login/oauth/access_token`

Parameters

Name	Type	Description
client_id	string	Required. The client ID you received from GitHub for your GitHub App.
client_secret	string	Required. The client secret you received from GitHub for your GitHub App.
code	string	Required. The code you received as a response to Step 1.

(b) API de GitHub.

Figura: Desarrollo de la tercera iteración.

Iteración 4

Consolidación de la aplicación

- Información adicional para el usuario:
 - ① Campos temporales.
 - ② Información sobre el creador de la tarea.
- Nuevo índice para los diferentes usuarios del sistema:
 - ① Nombre del usuario.
 - ② Token de usuario.
- Funcionalidad adicional para el usuario mediante Query.
 - ① Listado de tareas ejecutadas y pendientes del usuario activo.
 - ② Listado general de tareas del sistema.
 - ③ Listado de repositorios de un propietario dado.

Iteración 4

Consolidación de la aplicación

- Script de creación de los dashboard.
 - 1 Obtención de las tareas a ejecutar.
 - 2 Ejecución de *p2o.py* y *kidash.py* para crear los dashboard para Git y GitHub.
 - 3 Modificación de los campos temporales y de estado de Elasticsearch.

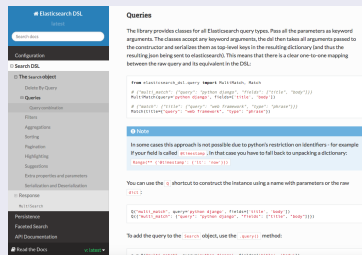


Figura: Query para Elasticsearch.

Iteración 5

Bootstrap, estadísticas y Kibana

- **Bootstrap**

- 1 Instalación y configuración.
- 2 Manejo de los elementos predefinidos por Bootstrap así como modificaciones con CSS.
- 3 Creación de una interfaz intuitiva y cómoda.

- **Estadísticas**

- 1 Estadísticas para tareas ejecutadas.
- 2 Estadísticas para tareas pendientes.

- **Kibana**

- 1 Enlace con los dashboard creados.

Arquitectura general

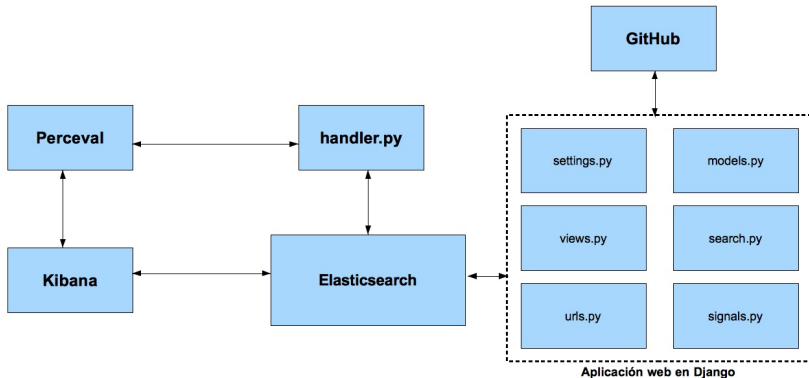


Figura: Diagrama de tecnologías del proyecto.

Versión API

Implementación

- Base para la interfaz WEB.
- Respuesta en formato JSON.

Respuesta

Ejecución de script de prueba.

Versión WEB

Implementación

- Creada a partir de Bootstrap.
- Intuitiva.

Respuesta

Ejemplo para probar la aplicación: Creador de dashboard.

Conclusiones

Consecución de objetivos

- Uso de Elasticsearch.✓
- Integración con GitHub.✓
- Funcionalidad útil.✓
- Creación de una API.✓
- Interfaz de usuario.✓

Trabajos futuros

- 1 Protección frente a errores, robustez.
- 2 Desarrollo de una interfaz de usuario dinámica.
- 3 Aumento de la funcionalidad para mejorar la experiencia de usuario.
- 4 Ampliación de las visualizaciones de los dashboard creados con Kibana.