Esquite

Comunidad Elotl

12 de julio de 2021

Índice general

Puedes comenzar con la *Instalación de Esquite* y después seguir alguno de nuestros *Tutoriales* para que configures y personalices tu propia instancia de Esquite.

Adicionalmente esta disponible nuestra imagen oficial de Docker para el *framework*.

CAPÍTULO 1

Documentación de la API

Información sobre los *endpoints* de la API de Esquite y cómo utilizarlos.

CAPÍTULO 2

¿Qué es Esquite?

Esquite es un *framework* de software libre destinado a personas que poseen corpus paralelos (textos bilingües) y que desean tener un sistema web que les permita subir, administrar realizar búsquedas de palabras o frases en sus corpus.

Este software funciona con `django` para el *backend* y utiliza `elasticsearch` como motor de búsquedas y gestión de documentos. Ejemplos de instancias del *framework* son las siguientes:

- TSUNKUA: corpus paralelo con documentos bilingües digitalizados y alineados de distintas variantes del otomí.

- KOLO: corpus paralelo con documentos bilingües digitalizados y alineados de las lenguas mixtekas.

2.1 Tabla de contenidos

2.1.1 Instalación de Esquite

Instalación manual

0. Instalar y ejecutar `elasticsearch`

Atención: El *framework* requiere de una instancia de `elasticsearch` previamente instalada. Para completar este requerimiento puedes consultar la página oficial de Elasticsearch

Truco: El asistente de configuración *`wizard()`* configurará un índice de Elasticsearch con el archivo de configuración por defecto *elastic-config.json*. Puedes modificarlo a tu gusto siguiendo la documentación de elasticsearch.

1. Clona el repositorio:

```
$ git clone https://github.com/ElotlMX/esquite
```

2. Crea un entorno virtual de `python`:

```
$ virtualenv env -p /usr/bin/python3
```

3. Activar el entorno:

```
$ source env/bin/activate
```

4. Instalar las dependencias:

```
(env)$ pip install -r requeriments.txt
```

5. Iniciar el asistente de instalación e ingresar los datos requeridos. Ver *wizard()*

```
(env)$ python wizard.py
```

Consejo: El asistente *wizard()* creara un *Archivo de configuración* llamado `env.yaml` donde se definen configuraciones de *Colores*, *Teclado*, *Datos de contacto*, entre otras.

Este archivo debe estar en la raíz del proyecto.

---

6. Aplicar migraciones de `django`:

```
(env)$ python manage.py migrate
```

7. Correr `django` en segundo plano:

```
(env)$ python manage.py runserver 0.0.0.0:8000 &
```

Imagen de Docker

Instalación de docker

Si no tienes `docker` instalado puedes ejecutar los siguientes comandos para instalarlo:

```
curl -sSL https://get.docker.com | sh
sudo service docker start
pip3 install docker-compose
```

Nota: Elasticsearch necesita la siguiente configuración en producción: El valor de `vm.max_map_count` debe ser `262144`. Para esto existen dos opciones:

a. Cambio temporal:

```
sysctl -w vm.max_map_count=262144
```

b. Cambio permanente modificando `/etc/sysctl.conf`:

```
vm.max_map_count=262144
```

Iniciando el contenedor

1. Clona e ingresa al repositorio:

```
git clone https://github.com/ElotlMX/Esquite-docker.git
cd Esquite-docker
```

2.a Usando archivo de inicialización `esquite-docker.sh`:

```
sudo ./esquite-docker.sh start
```

2.b Usando *docker-compose* directamente:

```
sudo docker-compose up -d
```

Navegando en la interfaz web

Ingresa a http://localhost. El password default para el administrador del corpus (`http://localhost/corpus-admin/`) es **elotl**.

Consejo: Puedes cambiar el password por defecto cambiandop la variable *CFG_CORPUS_ADMIN_PASS=elotl* en el archivo *docker-compose.yml*.

Nota: *sudo* es necesario ya que por default *Docker* necesita permisos de *root* para crear nuevos container. Sin embargo esto se puede cambiar si se le asigna a un usuario específico permisos para ejecturar *Docker*.

Opciones

Al ejecutar `esquite-docker.sh` aparecen las opciones disponibles:

```
##############################################
 Esquite Docker script   - Comunidad ElotlMX
----------------------------------------------
 Github: https:///github.com/elotlmx
 Web   : Elotl.mx
##############################################


[EN ] ERROR: Unknown Option: Syntax:    ./esquite-docker␣
↪(start|stop|restart|destroy|info|update|recreate)
[ES ] ERROR: Opción no valida. Sintaxis:    ./esquite-docker␣
↪(iniciar|detener|reiniciar|destruir|info|actualizar|recrear)
[NAH] TLATLACOLLI: Opción no valida. Sintaxis:    ./esquite-docker␣
↪(pehualtia|cahua|re-pehualtia|tlapoloa|tlanonotzaliztli|yancuic|tlaana)
```

Opciones de Docker compose

Opciones generales

El archivo de configuración de `docker-compose.yml` se puede personalizar para las opciones generales de Esquite.

Índice externo de Elasticsearch

Si se desea usar un indice externo de Elasticseach, solo se deben cambiar las variables `CFG_URL` y `CFG_INDEX`. Si estas opciones no se modifican, se creará un índice automáticamente en un container generado por el script de

inicialización

Actualización de versión de Esquite

Se puede habilitar la actualización de Esquite cada vez que se reinicie el container activando la opción `CFG_UPDATE_ON_BOOT` o manualmente por medio de las opciones `update` o `actualizar` o `tlanonotzaliztli` con el script `./esquite-docker.sh`

2.1.2 Tutoriales

Personalización

El archivo `env.yaml` contiene la mayoria de elementos que pueden modificarse para personalizar tu instancia de Esquite. Además en los archivos *templates/user/*.html* puedes agregar texto en `html` que se incrustará en las vistas de la interfaz web.

Colores

La personalización de colores está limitada a los fondos, texto y bordes de diversos elementos como el navbar, footer, botones, etc. Modificando la variable `COLORS` que tiene formato de diccionario se pueden alterar los colores de las vistas por elemento `HTML`. La variable tiene el siguiente formato:

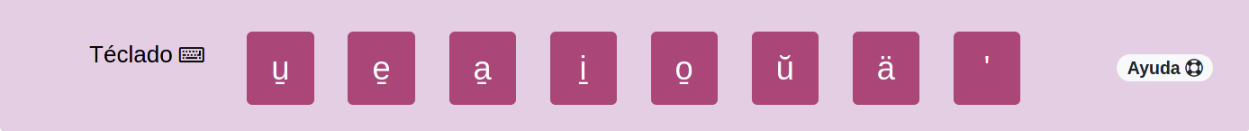
```
COLORS:
  background:
    btnhover: '#bf6492'
    button: '#aa4678'
    footer: '#e2e4f2'
    form: '#e3cee3'
    highlight: '#aa4678'
    nav: '#e3cee3'
  border:
    button: '#aa4678'
    input: '#aa4678'
  text:
    bold: '#aa4678'
    btnhover: 'white'
    button: '#e2e4f2'
    footer: '#20124d'
    form: '#20124d'
    highlight: 'white'
    hoverlinks: '#bb6a93'
    links: '#aa4678'
    nav: 'black'
    navactive: '#aa4678'
    navhover: '#bb6a93'
    result: '#20124d'
```

Se puede utilizar formato hexadecimal (`#ffffff`) o el nombre en inglés del color si es un color soportado por `css`.

Teclado

El corpus permite agregar teclas personalizadas para la lengua *l2* con caracteres que no estén en el teclado físico o que sean difíciles de obtener. Para agregar teclas personalizadas se debe modificar la variable `KEYBOARD` del archivo `env.yaml` agregando cada tecla en un renglón. A continuación se muestra un ejemplo

```
KEYBOARD:
- u
- e
- a
- i
- o
- ŭ
- ä
- "'"
```



Nota: Se debe agregar el guión antes de cada letra para que se pueda interpretar como una lista de caractéres. Tambien se puede usar una sintaxis de lista tipo python como `KEYBOARD: [u, e, a, i, o, ŭ, ä, "'"]`.

Datos de contacto

Se pueden agregar las redes sociales, datos de contacto de la organización. Además, se pueden agregar ligas a corpus similares. Dicho datos aparecen en la vista de `Ligas de ínteres` y en el *footer* de la interfaz web. Para agregar estos datos se debe modificar la variable `LINKS`. Esta variable tiene un formato de diccionario.

```
LINKS:
    social:
      site: 'https://mi_sitio.mx/'
      blog: 'https://mi_sitio.mx/blog/'
      email: 'contacto@mi_sitio.mx'
      facebook: 'https://facebook.com/mi_sitio'
      github: 'https://github.com/mi_sitio'
      twitter: 'https://twitter.com/mi_sitio'
    corpora:
      default: "https://www.default-corpus.com/search"
```

Colaboradoræs

Si el desarrollo del proyecto que estas elaborando tiene más personas involucradas es posible modificar la variable `COLABS` para agregar los nombres de estas personas. Dichos nombres se desplegaran en la vista de `participantes`.

```
COLABS:
  - Hari Seldon
  - Salvon Hardin
  - Hober Mallow
```

(continué en la próxima página)

(proviene de la página anterior)

```
  - Bayta Darrell
  - Arkady Darrell
```

Modificación del banner

El banner por defecto puede ser remplazado modificando el archivo que se encuentra en la ruta `static/img/banner.png`. El archivo **debe** llamarse `banner.png`. Se recomienda utilizar una imagen de `1260 x 270 pixeles`.

Vistas

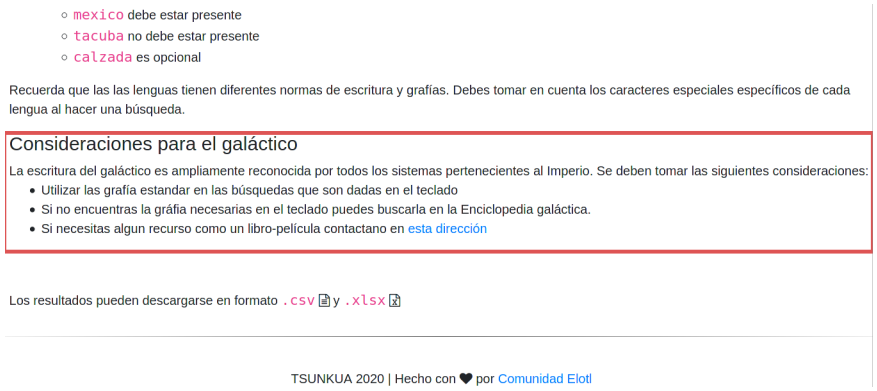
Es posible extender las vistas de `Ayuda`, `Ligas de interés`, `Acerca del Corpus` y `Participantes` con información específica del proyecto.

Para agregar información a las vistas se deben modificar los archivos que se encuentran en la ruta `tamplates/user/`. Cada archivo hace referencia a la vista que se modificará. El formato de los archivos es `html`.

Por ejemplo, si deseas extender la sección de ayuda puedes modificar el archivo `templates/user/help-user.html`. Si agregas el siguiente código `html` se obtiene el resultado de la imagen.

```
<h4>Consideraciones para el galáctico</h4>
La escritura del galáctico es ampliamente reconocida por todos los sistemas␣
↪pertenecientes al Imperio. Se deben tomar las siguientes consideraciones:

<ul>
  <li>Utilizar las grafía estandar en las búsquedas que son dadas en el teclado</li>
  <li>Si no encuentras la gráfia necesarias en el teclado puedes buscarla en la␣
↪Enciclopedia galáctica.</li>
  <li>Si necesitas algun recurso como un libro-película contactano en <a href="http://
↪www.imperio.com" target="_blank">esta dirección</a></li>
</ul>
```



Administración del corpus

La aplicación provee una interfase interna de *administración del corpus*. Para acceder al administrador se debe introducir la *URL* directamente en el navegador y es la siguiente `https://<micorpus.com>/corpus-admin/`, dónde `<micorpus.com>` dependerá de las configuraciones personales de dominio.

Nota: Si está en un entorno local `<micorpus.com>` debería ser sustituido por `localhost:8000` con lo que la *URL* será `http://localhost:8000/corpus_admin/`

Subida de documentos

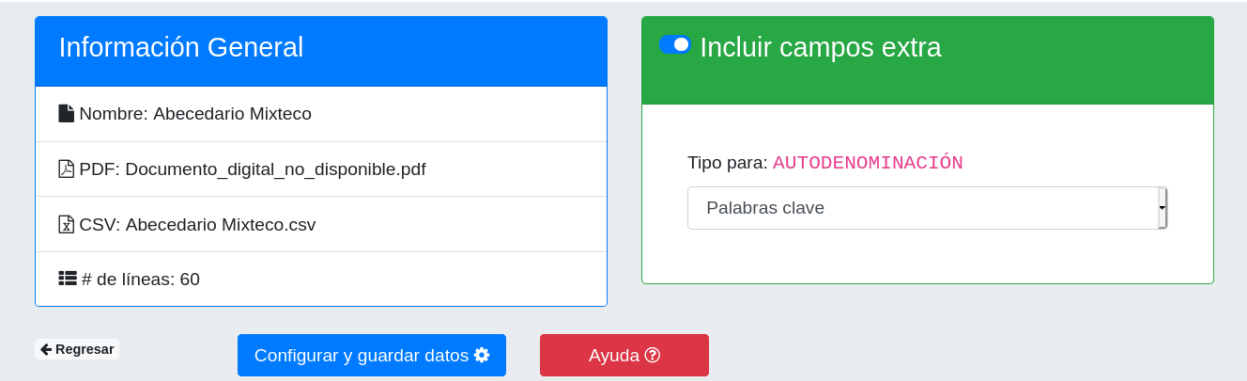
Si se quiere subir material al corpus deberá ser en formato `.csv` (**separado por comas**).

Es **indispensable** que exista la cabecera ya que con base en el nombre de las columnas se realizarán las configuraciones automaticas y el poblado de la base de datos. Actualmente se tiene la siguiente convención para la subida de nuevos documentos:

l1	l2	variant
Una vez una señora se emborrachó	nándi na ra t'ụxú bintí	Otomí del Estado de México (ots)
Luego se fue a dormir a la casa del vecino	xụtụ bimáyóhthó 'á ngŭ ra bésíno	Otomí del Estado de México (ots)
Después que se durmió	despwés ya biyóbí	Otomí del Estado de México (ots)
En la mañana se levantó y se fue a su casa	na ra nchúdi ya binántsi bimá ra ngŭ	Otomí del Estado de México (ots)
Pero había mucha neblina al irse	pewi gamáya' mihtsú mikhá ra mbangwí	Otomí del Estado de México (ots)
Y pasó por una zanja	i bithó pa na ra chái	Otomí del Estado de México (ots)
Encontró una ranota	bidót'i na ra ndụ'wẹ'chẹ	Otomí del Estado de México (ots)
Estaba muy grande	mindụgrande	Otomí del Estado de México (ots)
Luego se bajó	xụtụ bigát'ya'wi	Otomí del Estado de México (ots)
Dijo: Me voy a llevar esta rana para ir a alm	binina este máháxkhó na ra 'wẹ'pa dámá	Otomí del Estado de México (ots)
Se fue entonces	bimáya'wi	Otomí del Estado de México (ots)
Luego otra vez caminó como media parcela	xụtụ este pi biñó komo media parsela má	Otomí del Estado de México (ots)

Una columna con el nombre **l1** (en este ejemplo español), otra columna con el nombre **l2** (en este ejemplo otomí) y en la última columna con el nombre **variant** (en caso de no existir variante la columna debe existir con el nombre indicado pero vacía). Además, cada documento deberá tener asociado un archivo `.PDF` con fines ilustrativos para læs usuariæs.

Truco: Se pueden agregar más columnas. Al subir el primer documento se notificará que existen campos adicionales a los estándar (l1, l2, variant). Basta con identificar los campos como texto o palabras clave. **Agregar los campos extra es opcional**.



Nota: En caso de que la variante tenga ISO se requiere que venga entre paréntesis al final del nombre de la variante como se muestra en la tabla anterior.

Si no existe variante para el documento dicha columna **deberá** existir pero estará vacía.

Configuración

Creación del índice de elasticsearch

El *framework* requiere de un índice de `elasticsearch` configurado. Para crear el índice es necesario que una instancia de `elasticsearch` este instalada y ejecutandose. El asistente *`wizard()`* se encarga de crear un índice con configuraciones por defecto *elastic-config.json*. Adicionalemente, se puede utilizar el comando `curl` como se muestra a continuación:

```
$ curl -X PUT -H "Content-Type: application/json" -d @elastic-config.json␣
↪localhost:9200/<nombre-de-tu-indice>
```

La configuración por defecto esta optimizada para que a la lengua *l1* se le aplique un preprocesamiento asumiendo que es el idioma español para que las búsquedas tomen en cuenta las *stopwords*.

elastic-config.json

```
{
  "settings": {
    "index": {
      "number_of_shards": 1,
      "analysis": {
        "filter": {
          "spanish_stop": {
            "type":       "stop",
            "stopwords":  "_none_"
           },
          "spanish_stemmer": {
            "type": "stemmer",
            "language": "light_spanish"
          }
        },
        "analyzer": {
          "rebuild_spanish": {
            "tokenizer":  "standard",
            "filter": [
              "lowercase",
              "spanish_stop",
              "spanish_stemmer"
            ]
          }
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "pdf_file": {
        "type": "keyword"
      },
      "document_id": {
```

(continué en la próxima página)

(proviene de la página anterior)

```
        "type": "keyword"
      },
      "document_name": {
        "type": "keyword"
      },
      "l1": {
        "type": "text",
        "analyzer":"rebuild_spanish"
      },
      "l2": {
        "type": "text"
      },
      "variant": {
        "type": "keyword"
      }
    }
  }
}
```

2.1.3 Modulo `esquite`

Variables de entorno

esquite.context_processors

esquite.context_processors.**api**(*request*)
Configuraciones de los l+imites para la API

esquite.context_processors.**colors**(*request*)
Configura variables de entorno para los colores

esquite.context_processors.**google_analytics**(*request*)
Configura variables de entorno de google analytics

esquite.context_processors.**keyboard**(*request*)
Configura variable de entorno para teclaod personalizado

esquite.context_processors.**languages**(*request*)
Configura variables de entorno de las lenguas

esquite.context_processors.**project_info**(*request*)
Configura variables de entorno con información del proyecto

La información que se establece es *nombre*, *nombre de la organización*, *colaboradoras* y *redes sociales*. Las últimas dos son listas de python.

esquite.context_processors.**user_templates**(*request*)

Configuraciones generales

esquite.settings

Rutas

esquite.urls

esquite.urls.**path**(*route*, *view*, *kwargs=None*, *name=None*, *, *Pattern=<class 'django.urls.resolvers.RoutePattern'>*)

esquite.urls.**re_path**(*route*, *view*, *kwargs=None*, *name=None*, *, *Pattern=<class 'django.urls.resolvers.RegexPattern'>*)

Vistas

esquite.views

esquite.views.**about**(*request*)

Muestra la página acerca del corpus

Vista encargada de mostrar la sección de acerca del corpus motivación del corpus y descripción general de la comunidad Elotl. Además, muestra la información actual del corpus como número de documentos, parrafos por documentos y pdfs asociados.

Parámetros **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Vista de acerca del corpus

esquite.views.**ayuda**(*request*)

Muestra la página de ayuda

Vista encargada de mostrar la sección de ayuda con las operaciones soportadas, como realizar busquedas y recomendaciones generales.

Parámetros **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Vista de ayuda

esquite.views.**index**(*request*)

Muestra la vista raíz del corpus paralelo

Parámetros **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Vista principal

esquite.views.**links**(*request*)

Muestra la página de enlaces de interés

Vista encargada de mostrar la sección de ligas de interés con la página de Elotl, el blog de Elotl y el diccionario del otomi

Parámetros **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Vista de enlaces de interés

esquite.views.**participants**(*request*)

Muestra la página de participantes

Vista encargada de mostrar la sección de participantes con los nombres de los participantes y ligas de contacto con la comunidad Elotl

Parámetros **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Vista de participantes

esquite.views.**pdf_view**(*request*, *file_name*)

Muestra archivos PDF con base en el nombre

Parámetros

- **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema
- **file_name** – Nombre del PDF a mostrar

Type HttpRequest

Type str

Devuelve Archivo PDF para ser visto en el navegador

Configuración de servidor web

Advertencia: Para producción no se recomienda utilizar el servidor web que django tiene por defecto ya que este es solo para el ambiente de desarrollo

esquite.wsgi

Configuración de WSGI para el proyecto.

Para mas información consulta la documentación de django *aqui <https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/>*

2.1.4 Modulo searcher

Formularios

searcher.forms

class searcher.forms.**SearchPostForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.forms.Form

Clase encargada de generar de forma dinámica el formulario que permite hacer las búsquedas

Atributos

- **LANGUAGES: Lista de lenguas en las cuales se pueden hacer búsquedas con el formato *(KEY, VALUE)***

  type list

- **VARIANTS: Lista de variantes disponibles con el formato *(KEY, VALUE)***

  type list

- **search_placeholder: Variable que modifica el placeholder del elemento input para insertar el texto a buscar**

  type str

- **variant_label: Variable que modifica la etiqueta de las variantes del corpus**

  type str

- **idioma: Objeto de django forms que renderea un elemento <select> de html**

  type form.ChoiceField

- **busqueda: Objeto de django forms que renderea un elemento <input> de html**

  type form.CharField

- **variante** [Objeto de django forms que renderea un elemento <select> de html con multiples opciones]

  type form.MultipleChoiceField

```
LANGUAGES = [('L1', 'L1'), ('L2', 'L2')]
VARIANTS = []
base_fields = {'busqueda': <django.forms.fields.CharField object>, 'idioma': <django.f
declared_fields = {'busqueda': <django.forms.fields.CharField object>, 'idioma': <djan
property media
search_placeholder = 'Búsqueda'
variant_label = 'Variantes'
variantes = {'status': 'error'}
variants_attrs = {'class': 'form-control', 'disabled': True}
```

Funciones auxiliares

searcher.helpers

searcher.helpers.**data_processor**(*raw_data*, *idioma*, *query*)

Procesa los datos crudo de la API de Elasticsearh para devolver solo los resultados

Función que recibe una lista con los datos que prove la API de elacticsearch, procesa los datos para ignorar los metadatos del API y retorna solo los resultados de búsqueda como una lista.

Parámetros

- **raw_data** – Lista de resultados crudos del API de Elasticsearch
- **idioma** – ISO del idioma de búsqueda
- **query** – Cadena de búsqueda

Type list

Type str

Type str

Devuelve Resultados de búsqueda

Tipo del valor devuelto list

searcher.helpers.**doc_file_to_link**(*doc_name*, *doc_file*, *path*)

Función que liga el nombre de un documento con su archivo pdf

Parámetros

- **doc_name** – Nombre del documento
- **doc_file** – Nombre del archivo
- **path** – Path donde se encuentran los pdfs

Type str

Type str

Type str

Devuelve Nombre del documento con la liga al pdf

Tipo del valor devuelto str

searcher.helpers.**ethno_btn_maker**(*variante*)

Crea botones ''html'' con información ethnologue

Función encargada de construir los botones para la columna de variantes que despliegan los modals con la información obtenida de la plataforma ethnologue

Parámetros **variante** – Variante de la lengua 2

Type str

Devuelve Cadenas con los botones en `html`

Tipo del valor devuelto str

searcher.helpers.**ethno_table_maker**(*soup*)

Crea la tabla html con información de ethnologue

Con base en la variante en turno se crea, dinamicamente, una cadena que contiene una tabla html que será rendereada por un modal en la vista de busqueda.

Parámetros **soup** – Objeto con la página `html` de ethnologue

Type `BeautifulSoup Object`

Devuelve Tabla en formato `html`

Tipo del valor devuelto str

searcher.helpers.**get_variants**()

Obtiene las variantes actuales de elasticsearch

Función encargada de obtener las variantes existentes en el índice de `elasticsearch` a través del API aggregations. Se obtienen en un diccionario con el ISO de la variante como llave y el nombre de la variante como valor. Se agrega al diccionario de variantes el estatus de la consulta (success o error).

Devuelve variantes

Tipo del valor devuelto dict

searcher.helpers.**highlighter**(*hit*, *idioma*, *query*)

Resalta la búsqueda realizada por el usuarix

Función que busca el campo `highlight` en los objetos devueltos por el API de `elasticsearch` y lo utilza para reemplazar el texto del idioma en el que se realizó la búsqueda. Si la búsqueda es en español, el texto dentro del campo highlight estará preprocesado por Elasticsearch.

Parámetros

- **hit** – Resultado de búsqueda
- **idioma** – ISO del idioma de búsqueda
- **query** – Cadena de búsqueda

Type list

Type str

Type str

Devuelve Texto de los resultados resaltado

Tipo del valor devuelto list

searcher.helpers.**query_kreator**(*term*)

Crea la estructura para busquedas en Elasticsearch

Función encargada de devolver el objeto con los elementos necesarios para realizar una búsqueda en Elasticsearch. Esto beneficia la limpieza del código reemplazando una variable estática

Parámetros **term** – Consulta introducida por la usuaria desde el Frontend

Type str

searcher.helpers.**results_to_csv**(*data*, *variants*)

Función que escribe los resultados de la consulta en archivo csv

Guarda los resultados de la consulta en formato `csv` en caso de que la usuaria quiera descargarlos.

Parámetros

- **data_response** – Resultados de la consulta devueltos por el índice de `elasticsearch`
- **variants** – Variantes actuales para saber si escribir variante o dejar el campo vacío.

Type list

Type dict

Devuelve Estatus del archivo. `True` si fue escrito, `False` en caso contrario

Tipo del valor devuelto bool

searcher.helpers.**variant_to_query**(*variantes*)

Toma las variantes seleccionadas y regresa una cadena que filtra resultados de búsqueda por variantes

Parámetros **variantes** – Lista de keys de las variantes seleccionadas

Type list

Devuelve Cadena aceptada por el API de elasticsearch para filtrar

Tipo del valor devuelto str

Rutas

searcher.urls

searcher.urls.**path**(*route*, *view*, *kwargs=None*, *name=None*, ***, *Pattern=<class 'django.urls.resolvers.RoutePattern'>*)

Vistas

searcher.views

searcher.views.**download_results**(*request*)

Descarga los resultados de la busqueda actual

Vista asociada a botón que se encarga de descargar los resultados de la consulta actual

Parámetros **request** – Objeto HttpRequet para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Los resultados de busqueda en formato csv

searcher.views.**ethnologue_data**(*request*, *iso_variant*)

Búsca información de la variante en Ethnologue

Trae la información de la página de la variante de Ethnologue. Se scrappea con BeautifulSoup. Posteriormente se cra una tabla html con la función ethno_table_maker.

Parámetros **request** – Objeto HttpRequet para pasar el estado de la app a través del sistema

Type HttpRequest

Paran iso_variant ISO de la variante

Type str

Devuelve Html con la información disponible de *Ethnologue*

Tipo del valor devuelto str

searcher.views.**search**(*request*)

Realiza la búsqueda y muestra los resultados

Vista encargada de construir el archivo json que será mandado al API de Elasticsearch para ejecutar la *query*. Posteriormente preprocesa la respuesta del API y envía las variables para ser desplegadas en el template sercher.html. Además, reenvia el formulario con la información previamente introducida para nuevas búsquedas.

Parámetros **request** – Objeto HttpRequets para pasar el estado de la app a través del sistema

Type HttpRequest

Devuelve Resultados de búsqueda y formulario para nuevas búsquedas

2.1.5 Modulo corpus_admin

Formularios

corpus_admin.forms

class corpus_admin.forms.**AddDocumentDataForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.forms.Form

Clase encargada de generar de forma dinámica el formulario que permite agregar nuevos renglones a un documento particular del corpus

Atributos

- **csv: Objeto de django forms que renderea un elemento input de html**

  type form.FileField

base_fields = {'csv': <django.forms.fields.FileField object>}

declared_fields = {'csv': <django.forms.fields.FileField object>}

property media

class corpus_admin.forms.**DocumentEditForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.forms.Form

Clase encargada de generar de forma dinámica el formulario que permite modificar el nombre y el PDF de un documento existente

Atributos

- placeholder: Variable que modifica el placeholder del input para el nuevo

nombre del documento

type str

- **nombre: Objeto de django forms que renderea un elemento input de html**

  type form.CharField

- **pdf: Objeto de django forms que renderea un elemento input de html**

  type form.FileField

base_fields = {'nombre': <django.forms.fields.CharField object>, 'pdf': <django.forms.

declared_fields = {'nombre': <django.forms.fields.CharField object>, 'pdf': <django.fo

property media

placeholder = 'Ingresa el nuevo nombre del documento'

class corpus_admin.forms.**NewDocumentForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.forms.Form

Clase encargada de generar de forma dinámica el formulario que permite cargar nuevos documentos al corpus

Atributos

- **VARIANTS: Lista de variantes disponibles con el formato (KEY, VALUE)**

  type list

- **nombre: Objeto de django forms que renderea un elemento input de html**

  type form.CharField

- **csv: Objeto de django forms que renderea un elemento input de html**

  type form.FileField

- **pdf: Objeto de django forms que renderea un elemento input de html**

  type form.FileField

base_fields = {'csv': <django.forms.fields.FileField object>, 'nombre': <django.forms.

declared_fields = {'csv': <django.forms.fields.FileField object>, 'nombre': <django.fo

property media

Funciones auxiliares

corpus_admin.helpers

corpus_admin.helpers.**check_extra_fields**(*fields*, *full=False*)

Revisa si existen campos adicionales a los default

Parámetros

- **fields** – Campos del usuario presentes en la cabecera del csv
- **full** – Bandera opcional si se requieren los campos completos. Por

Type list

ejemplo cuando se sube un respaldo de la base de datos :type: bool :return: Los campos adicionales encontrados si existen :rtype: set

corpus_admin.helpers.**csv_uploader**(*csv_name*, *doc_name*, *pdf_file*, *doc_id=''*, *extra_fields=False*)

Función encargada de cargar nuevas líneas al corpus

Manipula los archivos mandados desde formulario y los carga al corpus de Tsunkua por medio del API de elasticsearch. Se espera que la primera columna del archivo csv sea el texto en español, la segunda columna sea el texto en otomí y la tercera columna sea la variante(s)

Parámetros

- **csv_name** – Nombre del archivo csv con el texto alineado

- **doc_name** – Nombre del documento a cargar
- **pdf_file** – Nombre del archivo PDF del documento

Type str

Type str

Type str

Devuelve Número de líneas cargadas al corpus

Tipo del valor devuelto int

corpus_admin.helpers.**csv_writer**(*csv_file*)

Escribe un archivo ‘‘csv‘‘ de forma temporal

Esta función escribe el `csv` en disco para posteriormente subirlo al indice de `Elasticsearch`

Parámetros

- **csv_file** – `csv` enviado por medio del objeto `request`
- **file_name** – Nombre del archivo `csv`

Type `FileField`

Type str

Devuelve `True` si se guardo correctamente

Tipo del valor devuelto bool

corpus_admin.helpers.**get_corpus_info**(*request*)

Función que obtiene la información general del corpus

Está función utiliza el framework de `Elasticsearch` llamado *aggregations* para obtener los ids del corpus. Con cada uno se obtienen los nombres de documentos, nombres de archivos y total:>>.

Devuelve El total de documentos y una lista con información de los

documentos :rtype: int, list

corpus_admin.helpers.**get_document_info**(*_id*)

Obtiene información de un documento de Elasticsearch

Función encargada de obtener el nombre de documento, nombre del archivo asociado al documento e identificador por medio del id

Parámetros **_id** – Identificador del documento

Type str

Devuelve Diccionario con nombre, archivo e identificador

Tipo del valor devuelto dict

corpus_admin.helpers.**get_index_config**()

corpus_admin.helpers.**pdf_uploader**(*file*, *name*)

Función encargada de cargar y guardar el archivo pdf de un nuevo documento

Parámetros

- **file** – `pdf` enviado por medio del objeto `request`
- **name** – Nombre del archivo PDF

Type `FileField`

Type str

Devuelve Verdadero si se pudo cargar el archivo, falso en caso contrario

Tipo del valor devuelto bool

corpus_admin.helpers.**update_config**(*config*)
Actualiza las configuraciones locales de elasticsearch

corpus_admin.helpers.**update_index_name**(*new_index_name*)

Rutas del corpus_admin

corpus_admin.urls

corpus_admin.urls.**path**(*route*, *view*, *kwargs=None*, *name=None*, ***, *Pattern=<class 'django.urls.resolvers.RoutePattern'>*)

Vistas del corpus_admin

corpus_admin.views

corpus_admin.views.**add_doc_data**(*request*, *_id*)
Vista que muestra un formulario para añadir líneas a un documento existente

- ***:param request:*** **Objeto *HttpRequets* para pasar el estado de la app a** través del sistema
- *:type: HttpRequest*
- *:param _id:* identificador del documento a visualizar
- *:type: str*
- *:return:* Formulario para agregar líneas a un documento existente

corpus_admin.views.**delete_doc**(*request*)
Vista encargada de eliminar documentos del corpus

- ***:param request:*** **Objeto *HttpRequets* para pasar el estado de la app a** través del sistema
- *:type: HttpRequest*

corpus_admin.views.**doc_edit**(*request*, *_id*)
Vista que muestra el formulario para editar el nombre y pdf de un documento

- ***:param request:*** **Objeto *HttpRequets* para pasar el estado de la app a** través del sistema
- *:type: HttpRequest*
- *:param _id:* Identificador de documento
- *:type: str*
- *:return:* Formulario para editar el nombre y pdf de un documento

corpus_admin.views.**doc_preview**(*request*, *_id*)
Vista que muestra el contenido de un documento particular

Muestra los renglones alineados que componen un documento en particular del corpus. Cada renglon tiene dos acciones, eliminar y editar.

- ***:param request:*** **Objeto *HttpRequets* para pasar el estado de la app a** través del sistema

- *:type: HttpRequest*
- *:param _id:* identificador del documento a visualizar
- *:type: str*
- *:return:* Contenido de un documento

corpus_admin.views.**export_data**(*request*)

Vista que exporta la base de datos completa del índice

Vista llamada desde el botón de *exportar* en el administrador del corpus. Se genera un respaldo de la base de datos en formato `csv` con el que se puede restaurar en otro índice de Elasticsearch.

- ***:param request:* Objeto *HttpRequets* para pasar el estado de la app a** través del sistema
- *:type: HttpRequest*

corpus_admin.views.**extra_fields**(*request*, *csv_file_name*, *document_name*, *pdf_file_name*)

Configura los campos extra detectados en un `CSV`

Parámetros

- **request** (*HttpRequest*) – Objeto `HttpRequets` de Django
- **csv_file_name** (*str*) – Nombre del archivo `csv`
- **document_name** (*str*) – Nombre del documento
- **pdf_file_name** (*str*) – Nombre del archivo `pdf`

Devuelve Redirecciona a la vista de nuevo documento

Tipo del valor devuelto None

corpus_admin.views.**list_docs**(*request*)

Esta vista muestra todos los documentos que conforman el corpus paralelo

- ***:param request:* Objeto *HttpRequets* para pasar el estado de la app a** través del sistema
- *:type: HttpRequest*
- *:return:* Lista de documentos del corpus con acciones por documento

corpus_admin.views.**new_doc**(*request*)

Vista que muestra el formulario para cargar nuevos documentos al corpus

Vista encargada de mostrar el formulario para agregar nuevo documentos al corpus. El documento se compone de archivo *CSV* alineado, PDF como portada del corpus y el nombre del archivo. En caso de éxito al subir el documento se redirige a la lista de documentos.

- ***:param request:* Objeto *HttpRequets* para pasar el estado de la app a** través del sistema
- *:type: HttpRequest*
- *:return:* Vista con formulario para nuevos documentos

2.1.6 Asistente de instalación `wizard.py`

Este programa se encarga de asistir a la usuaria a generar el archivo `env.yaml` que contiene las configuraciones generales del proyecto. El archivo mencionado es **necesario** para que el proyecto funcione correctamente.

Configuraciones

Un archivo `env.yaml` típico para el proyecto y generado por el asistente de configuración se verá de la siguiente manera:

Archivo de configuración

```
API:
  limit_results:
    anon: 10
    user: 100
  num_proxies: 0
  throttles:
    burst_anon: 20/hour
    burst_user: 50/hour
    sustain_anon: 50/day
    sustain_user: 200/day
COLABS:
  - Hari Seldon
  - Salvon Hardin
  - Hober Mallow
  - Bayta Darrell
  - Arkady Darrell
DEBUG: 'True'
KEYBOARD:
  - a
  - b
  - c
  - d
GOOGLE_ANALYTICS: 'UA-XXXXXXXXX-X'
INDEX: index-name
L1: "Español"
L2: "Galactico"
NAME: ENCICLOPEDIA GALACTICA
ORG_NAME: FUNDACION
COLORS:
  background:
    btnhover: '#69c9be'
    button: '#06a594'
    footer: '#ffffff'
    form: '#fdecb2'
    highlight: '#fdecb2'
    nav: '#fbda65'
  border:
    button: '#06a594'
    input: '#06a594'
  text:
    bold: '#06a594'
    btnhover: '#fbda65'
    button: '#ffffff'
    footer: '#000000'
    form: '#000000'
    highlight: '#048476'
    hoverlinks: '#69c9be'
    links: '#06a594'
    nav: '#06a594'
```

(continué en la próxima página)

(proviene de la página anterior)

```
    navactive: '#048476'
    navhover: '#69c9be'
    result: '#000000'
SECRET_KEY: '"<llave-secreta-autogenerada>"'
LINKS:
    social:
      site: https://example.com/
      blog: https://example.com/blog/
      email: mail@example.com
      facebook: https://www.facebook.com/fundacion/
      twitter: https://twitter.com/fundacion/
      github: https://github.com/fundacion/
    corpora:
        axolotl: "https://www.axolotl-corpus.mx/search"
        kolo: "https://kolo.elotl.mx/"
        job: "https://job.elotl.mx/"
URL: http://elasticsearch-ip:9600/
META_DESC: Corpus paralelo del Español al Galactico.
```

Advertencia: La variable llamada `DEBUG` está establecida por defecto en `True` dado que es mas conveniente. Pero, las recomendaciones de seguridad de `django` sugieren el modo `DEBUG` en `False` para un entorno de **producción**.

Sin embargo, con el modo `DEBUG` en `False` el servidor web de `django` no está habilitado, por lo que, los archivos estáticos (`js`, `css`, imagenes, entre otros) no se cargarán. Para ello se deberá configurar un servidor web externo como nginx , apache u otro.

El valor de la variable `DEBUG` en `True` es para un entorno de **desarrollo**. En este entorno se habilitará el servidor web de `django`. Además, si hubiese un error se mostraran, en el navegador, un detallado *traceback* que incluye muchos metadatos del entorno.

Recomendamos ampliamente leer la documentación sobre esta variable

Funciones del script

`wizard.`**`api_limits`**(*config*)
Establece valores de limites para la API

Se añaden límites para el consumo de la API incluyendo número de request por hora y día, resultados máximos devueltos para una consulta y el número de proxies en el server.

Parámetros **`config`** – Diccionario con la configuración

Type dict

Devuelve Configuraciones con los límites de la API

Tipo del valor devuelto dict

`wizard.`**`create_index`**(*config*)
Crea un índice de Elasticsearch con la configuración por defecto

`wizard.`**`create_user_scheme`**(*base_dir*)

`wizard.`**`set_colors`**(*config*)
Escribe los colores del proyecto

Escribe en el diccionario de configuraciones el color primario, secundario, color de los textos y color de contraste de los textos

Parámetros **config** – Diccionario con la configuración

Type dict

Devuelve Diccionario de configuraciones con los colores del proyecto

Tipo del valor devuelto dict

wizard.**set_project_info**(*config*)
Escribe información general del proyecto

Escribe en el diccionario de configuraciones el nombre de la organización, nombre del proyecto la primera y segunda lengua del corpus paralelo

Parámetros **config** – Diccionario con la configuración

Type dict

Devuelve Diccionario de configuraciones con información del proyecto

Tipo del valor devuelto dict

wizard.**set_services**(*config*)
Escribe información de los servivios

Escribe en el diccionario de configuraciones el nombre del índice y la url (ip y puerto) del servidor elasticsearch. Opcionalmente el token de Google Analytics.

Parámetros **config** – Diccionario con la configuración

Type dict

Devuelve Diccionario de configuraciones con información del de los servicios

Tipo del valor devuelto dict

wizard.**set_url**(*protocol='http'*, *ip='localhost'*, *port='9200'*)
Contruye una URL válida para el archivo de configuración

Dado el protocolo, la ip y el puerto contruye una URL válida para el archivo de configuración. Si las variables no fueron dadas por la usuaria utiliza la URL por defecto `http://localhost:9200/`

Parámetros

- **protocol** – Protocolo que debe ser `HTTP` o `HTTPS`
- **ip** – Nombre o ip del server de Elasticsearch
- **port** – Puerto del server Elasticsearch

Type str

Type str

Type str

Devuelve URL válida para el proyecto

Tipo del valor devuelto str

2.1.7 Estructura del proyecto

Esta sección está pensada para desarrolladoras y personas que desean entender como está estructurado el proyecto para contribuir con nuevo código o agregando características.

General

El proyecto esta desarrollado en `django` y por tanto sigue una estructura específica. A continuación se explicará la estructura del proyecto haciendo énfasis en la descripción general de los módulos que lo componen.

```
Esquite                     # Carpeta raíz
    ├── corpus_admin/       # Modulo de administración de archivos
    ├── env/                # Entorno virtual para python
    ├── media/              # Archivos PDFs
    ├── searcher/           # Modulo de búsqueda
    ├── static/             # Archivos estáticos (css, js, img, ...)
    ├── templates/          # Vistas en .html
    ├── esquite/            # Modulo principal del proyecto
    ├── docs/               # Carpeta que contiene la documentación
    ├── api/                # Modulo de la api
    ├── db.sqlite3          # Archivo para bases de datos
    ├── manage.py           # Command-line utility de django
    ├── __pycache__
    ├── README.md           # Archivo que describe el proyecto
    ├── requeriments.txt    # Descripción de las dependencias del proyecto
    └── wizard.py           # Asistente de instalación del proyecto
```

esquite

```
esquite
├── __init__.py
├── __pycache__/
├── settings.py              # Configuraciones globales
├── urls.py                  # URLs generales
├── views.py                 # Comportamiento de las vistas
├── context_processors.py    # Funciones para variables globales
└── wsgi.py                  # Configuraciones para servidor web
```

El modulo `esquite` contiene la base del proyecto. En este modulo se puede gestionar la **configuración global** del proyecto. También, las **urls** generales, el comportamiento de la página de inicio y la forma en que se despliegan los **pdfs**.

searcher

```
searcher
├── admin.py
├── apps.py
├── forms.py           # Formulario de búsqueda construido por django
├── helpers.py         # Funciones utilitarias para tratamiento de los datos
├── __init__.py
├── migrations/
├── models.py
├── __pycache__/
├── tests.py
├── urls.py            # Configuración de URLs para el modulo de búsqueda
└── views.py           # Comportamiendo de las vistas y manejo de consultas
```

En el modulo `searcher` se encuentra el funcionamiento de las vistas de búsqueda y manejo de las *queries* enviadas a `elasticsearch`. También, se pueden modificar las URLs del módulo `searcher` que comprenden todas las dis-

ponibles en el navbar (ayuda, links, about, participantes). De forma programática se genera el formulario de búsqueda. Este puede modificarse en `forms.py`.

corpus_admin

```
corpus_admin
├── admin.py
├── apps.py
├── forms.py      # Formularios para agregar, editar o modificar el PDF de un documento
├── helpers.py    # Funciones utilitarias
├── __init__.py
├── migrations/
├── models.py
├── __pycache__/
├── tests.py
├── urls.py       # Configuración de URLs del manejo de documentos
└── views.py      # Comportamiento de las vistas, carga de documentos, edición y␣
↪validaciones
```

El modulo `corpus_admin` se encarga de recibir, validar, cargar, editar y eliminar los documentos. Todo cambio se verá reflejado en el API de `elasticsearch`. Similar a `searcher` los formularios son creados dinámicamente y se pueden editar en `forms.py`. El archivo `views.py` es el que se encarga de la validación y subida de cualquier cambio en los documentos. Algunas funciones toman los parámetros de las URLs. La definición de los parámetros validos se puede configurar en `urls.py` así como las rutas del modulo.

templates

```
templates
├── about.html
├── base.html
├── corpus-admin/
│   ├── add-rows.html
│   ├── doc-edit.html
│   ├── doc-preview.html
│   ├── docs-list.html
│   └── new-doc.html
├── help.html
├── index.html
├── links.html
├── participants.html
├── searcher/
│   └── searcher.html
└── user/
    ├── about-user.html
    ├── help-user.html
    ├── links-user.html
    └── participants-user.html
```

En esta carpeta se encuentran las vistas `.html` que son llamadas por los archivos `views.py` de los diferentes módulos. Por convención la carpeta es llamada `templates`. Se hace uso del motor de templates de `django`. Más información del motor en la documentación. Además, se hace uso del *template inheritance* por lo que los elementos comunes (navbar, banner, footer, etc) se encuentran en el archivo `base.html` y de ahí se extienden a las diferentes vistas. Igualmente, el funcionamiento detallado de esta herramienta se puede encontrar en la documentación. Por último la carpeta `user` contiene fragmentos `.html` que brindan la posibilidad a las usuarias incrustar partes personalizadas en las vistas creadas.

static

```
static
├── css            # Estilos
├── data-tables    # Biblioteca para tablas y exportación de datos
├── img            # Imágenes del proyecto
├── js             # Scripts de bibliotecas
├── localisation   # Archivos de idiomas paralas tablas
└── fork-awesome   # Iconos decorativos del proyecto como los botones
```

Carpeta que contiene los archivos estáticos del proyecto como estilos, *scripts*, imágenes, iconos y bibliotecas utilizadas. Muchos de los estilos se encuentran en `css/main.css`. Para las tablas se utiliza la biblioteca DataTables y para las alertas select2.

CAPÍTULO 3

Indices

- genindex
- modindex
- search

Índice de Módulos Python

Índice