**INTRODUCCIÓN**

Con este trabajo se tiene el objetivo de mostrar una primera visión sobre la base de datos NoSQL CouchDB. Veremos el origen de CouchDB, cuáles son sus beneficios y profundizaremos en sus principales características. Se realizará también, un tutorial sobre como comunicarnos con la base datos y realizar operaciones básicas, tanto gráficamente como desde la consola. En estos tutoriales se asume que se tienen conocimientos básicos de HTTP. Finalmente, se ha realizado una aplicación de escritorio, para de manera ilustrativa, mostrar cómo se conectaría la base de datos y se pueden realizar distintas operaciones integradas con el lenguaje Python.

**De donde surge CouchDB**

CouchDB es un proyecto creado por Damien Katz en 2005 definido por su creador como un “sistema de almacenamiento para una base de datos objetal de gran escala”. Él mismo financió el proyecto durante dos años y lo liberó como proyecto de código abierto bajo la licencia GPL. En febrero de 2008 pasaría a ser un proyecto de Apache Incubator cambiando su licencia a Apache License. Más tarde en 2010 se publicaría la primera versión estable.

Actualmente Apache CouchDB es un gestor de base de datos de código abierto centrado en la facilidad de uso. Consiste en una base de datos NoSQL que usa JSON para almacenar los datos y Javascript como lenguaje de consulta mediante MapReduce y una API como medio para interactuar con la base de datos.

Más adelante nos centraremos en sus características principales y cómo comunicarnos o realizar operaciones básicas con la base de datos.

**Beneficios**

A continuación

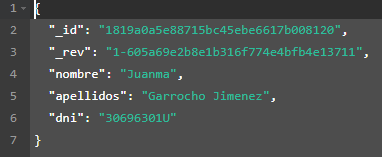
**Características**

* **Documentos**

Como ya hemos dicho anteriormente CouchDB es una base de datos NoSQL, concretamente orientada a documentos. Estos documentos serán nuestra unidad de almacenamiento. El formato utilizado por estos documentos es JSON (Javascript Object Notation), el cual es un formato sencillo para el intercambio de datos y que actualmente se encuentra bastante extendido en el mundo del desarrollo web. Además de los datos almacenados en este formato también se pueden adjuntar otros archivos a un documento de manera similar a como se hace en los correos electrónicos.

Debido al formato usado en los documentos, a diferencia de las bases de datos SQL, los datos no tendrán una esquema al almacenarse, es decir, cada documento tendrá su propio esquema sin necesidad tener unos campos específicos como se haría en las bases de datos SQL.

En estos documentos, CouchDB solo impondrá dos campos: id y rev. Estos campos respectivamente representan el identificador del documento y la versión. El identificador representa lo que en base de datos tradicionales es conocido como clave primaria y será el único atributo único. Este atributo podrá ser asignado automáticamente por CouchDB o manualmente por el desarrollador. En caso de querer que un atributo sea único en los documentos deberá asignarse como id del documento. Por otra parte, el atributo rev como ya hemos dicho anteriormente representa la versión del documento. Este atributo es usado por CouchDB para asegurar la consistencia del documento, de forma que si un documento es modificado por dos usuarios distintos a la vez se quedará con la última versión, dando lugar a un conflicto que podrá ser resuelto por el desarrollador de manera similar a lo que ocurre en git cuando dos desarrolladores “pushean” código a una misma rama.



* **Vistas**

Las vistas son algo muy importante en CouchDB, ya que una vez tenemos los datos representados en nuestros documentos tenemos que decidir cómo se van a mostrar estos documentos. Esto es una forma análoga de realizar lo que sería una query en las bases de datos SQL.

Las vistas se almacenan en un tipo de documento “especial” llamado “Design Document”, un documento también en formato JSON como cualquier otro pero con la distinción de que los datos que almacena son las vistas que han sido definidas. En un mismo “Design Document” se pueden almacenar varias vistas.



Estas vistas son escritas en Javascript usando MapReduce. La función Map nos permite filtrar y devolver la parte que queramos de nuestros documentos. Como parámetro, esta función recibe un documento ya que se ejecutará sobre todos los documentos de la base de datos y siempre retornará con la función emit() un par clave valor. Por otra parte la función Reduce nos permite realizar operaciones sobre los datos que se han filtrado. Algunas que vienen ya predefinidas son sum y count, aunque también nos ofrece la opción de realizar una función reduce personalizada.

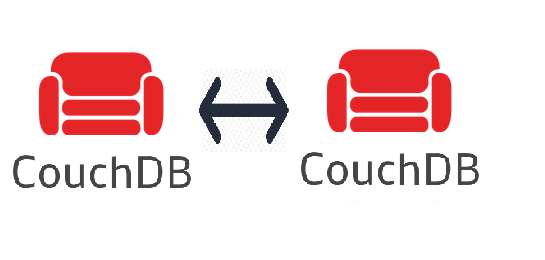
Una vez la vista es creada los resultados que se devolverán son indexados para mejorar la eficiencia de la base de datos. Cuando un documento es modificado o se crean nuevos documentos el resultado de esta vista será recalculado.

* **Replicación**

La principal característica de CouchDB es la facilidad de replicación de base de datos. Esto nos permite configurar desde un nodo que tenga una réplica continua a una copia de seguridad hasta un clúster con distintos nodos.

La replicación de bases de datos se puede configurar de forma sencilla y rápida desde entre bases de datos locales y remotas, permitiendo por ejemplo tener una réplica de la base de datos en un usuario móvil que se sincroniza con la base de datos remota cuando tiene conexión, permitiendo que el usuario pueda hacer uso de la aplicación sin que tenga conexión.

En las replicación también se pueden producir conflictos en los documentos si ambos se modifican a la vez, esto deberá ser resuelto por el desarrollador al igual que se realiza con git.



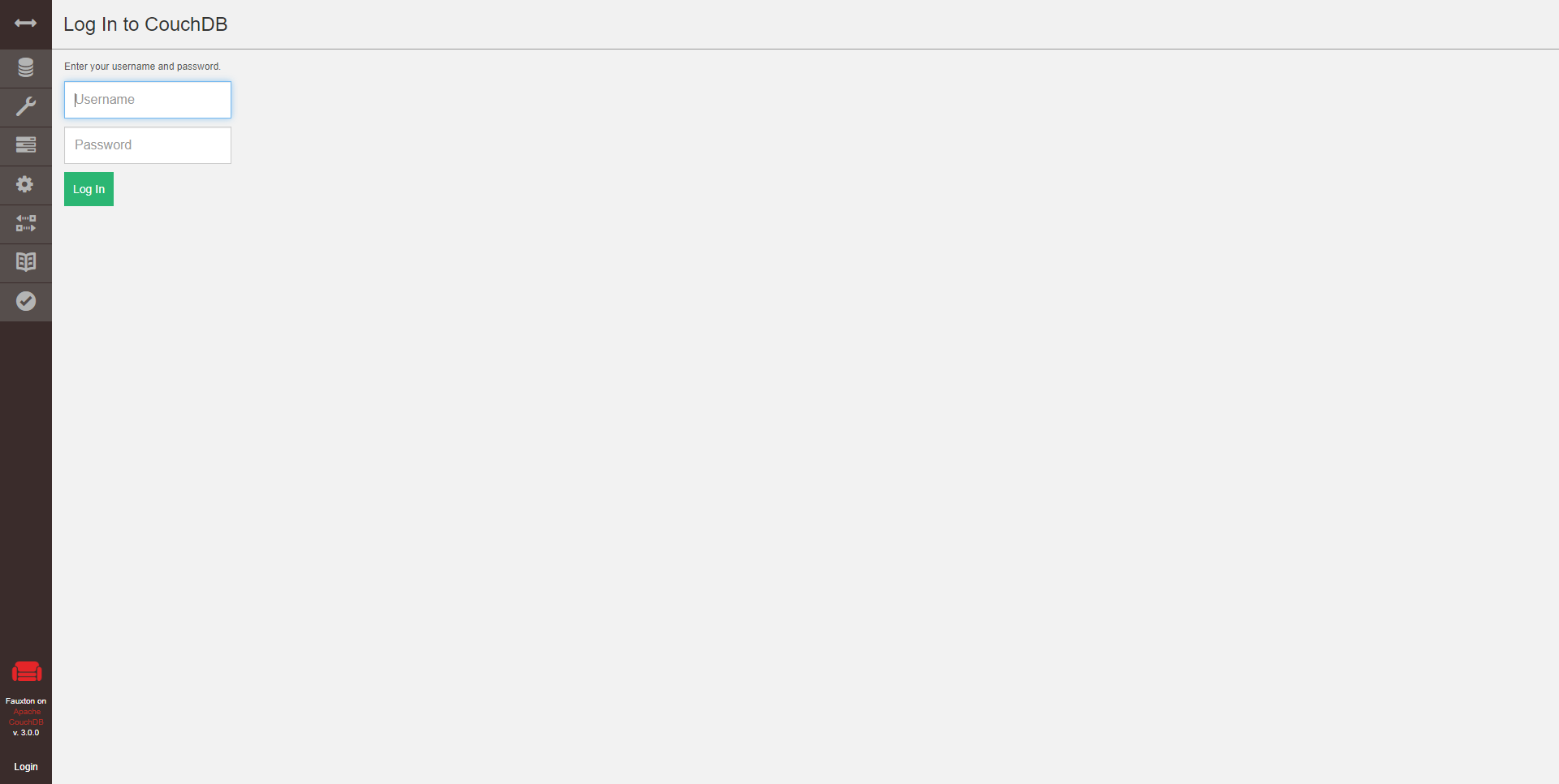
* **FAUXTON**

Fauxton es la interfaz gráfica de administración que nos ofrece en CouchDB. A continuación se presenta un tutorial de como realizar las funcionalidades básicas dentro de esta interfaz.

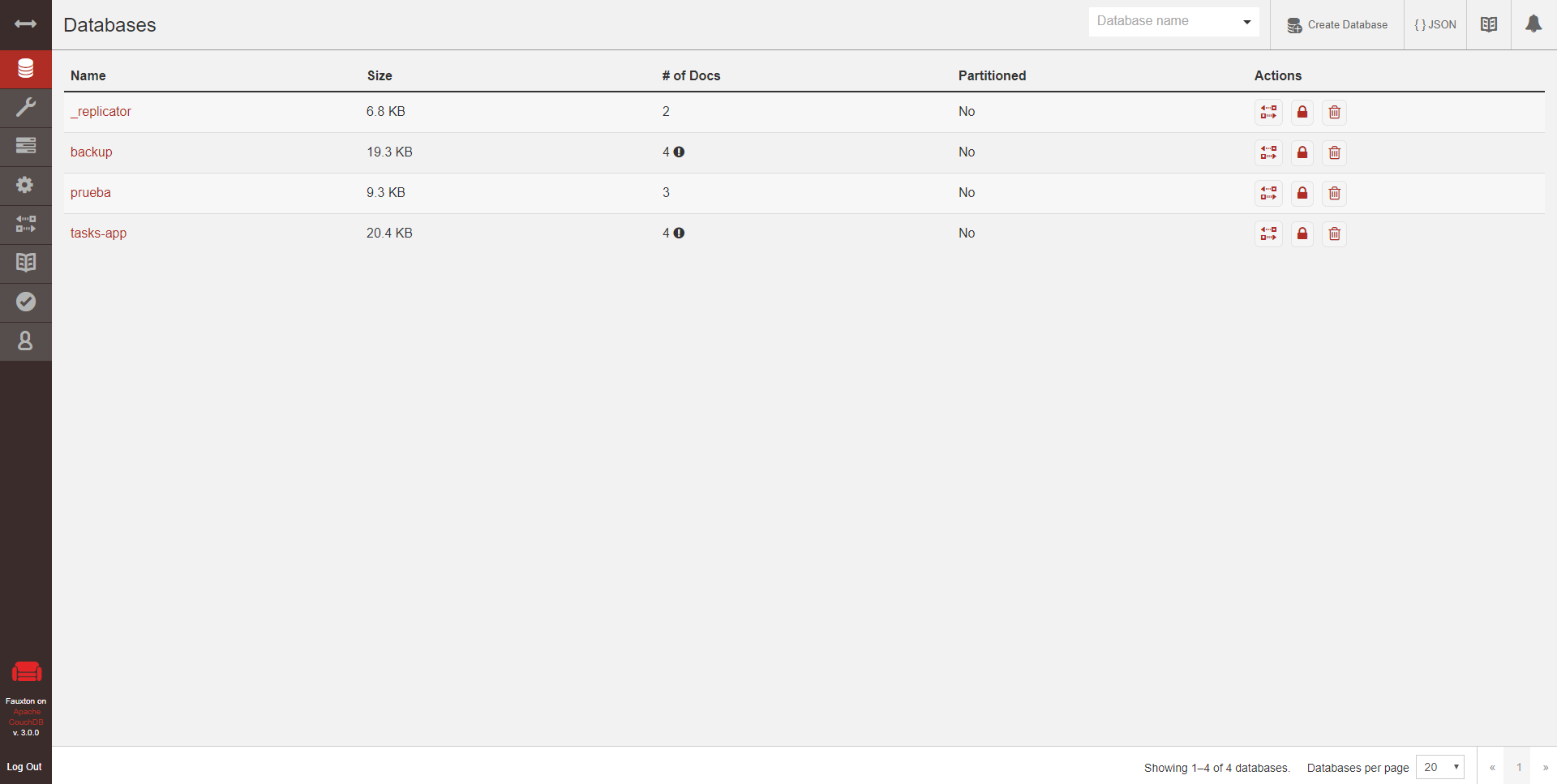
Para acceder a la interfaz basta con escribir en el navegador

<http://127.0.0.1:5984/_utils> . Una vez dentro tendremos que autenticarnos con

nuestro usuario y contraseña.

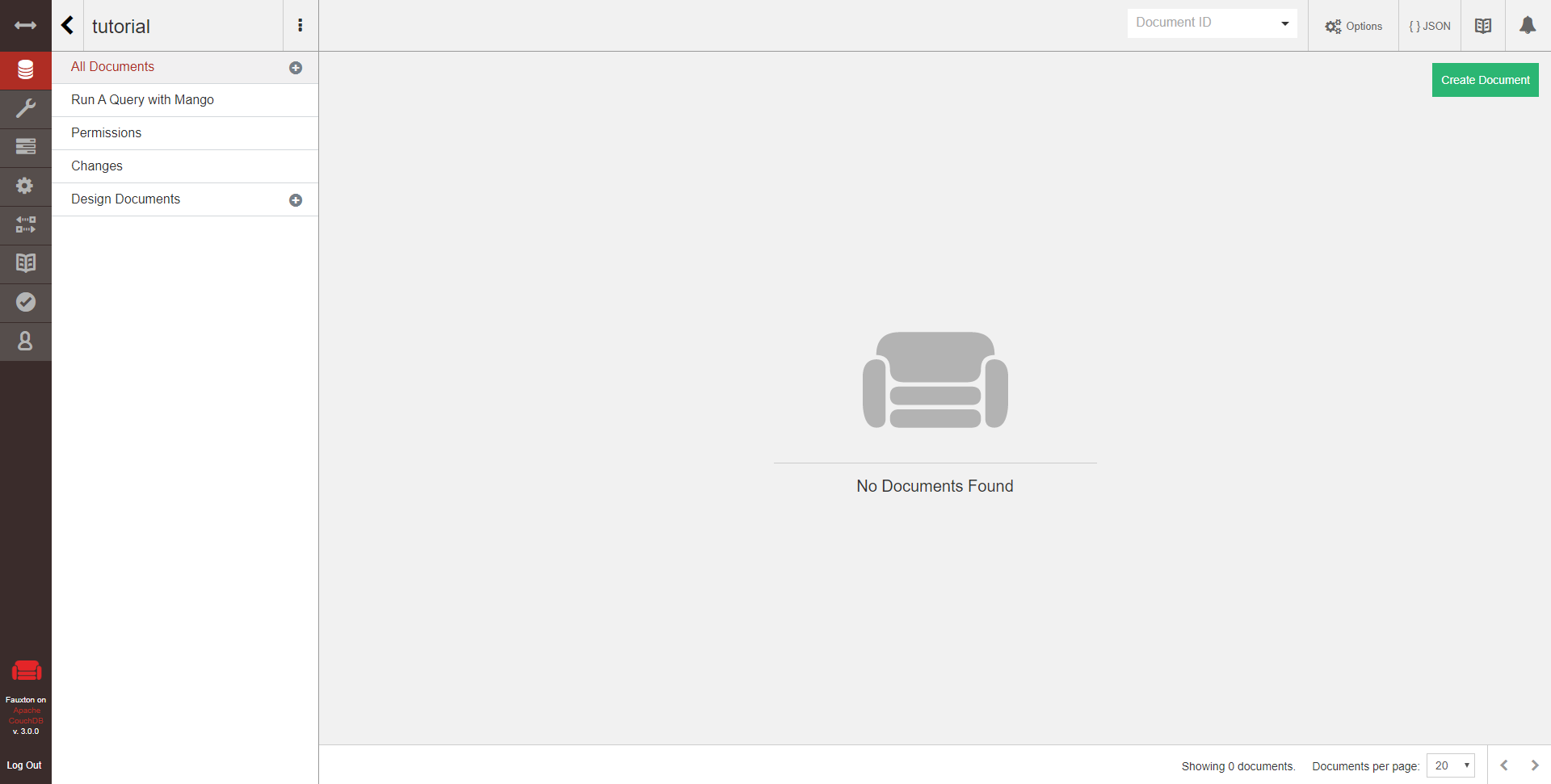


Una vez logeados se nos presenta un listado de todas las bases de datos. En la parte derecha de las bases de datos se nos muestran distintas opciones que son hacer una réplica de la base de datos, asignar permisos y eliminarla.

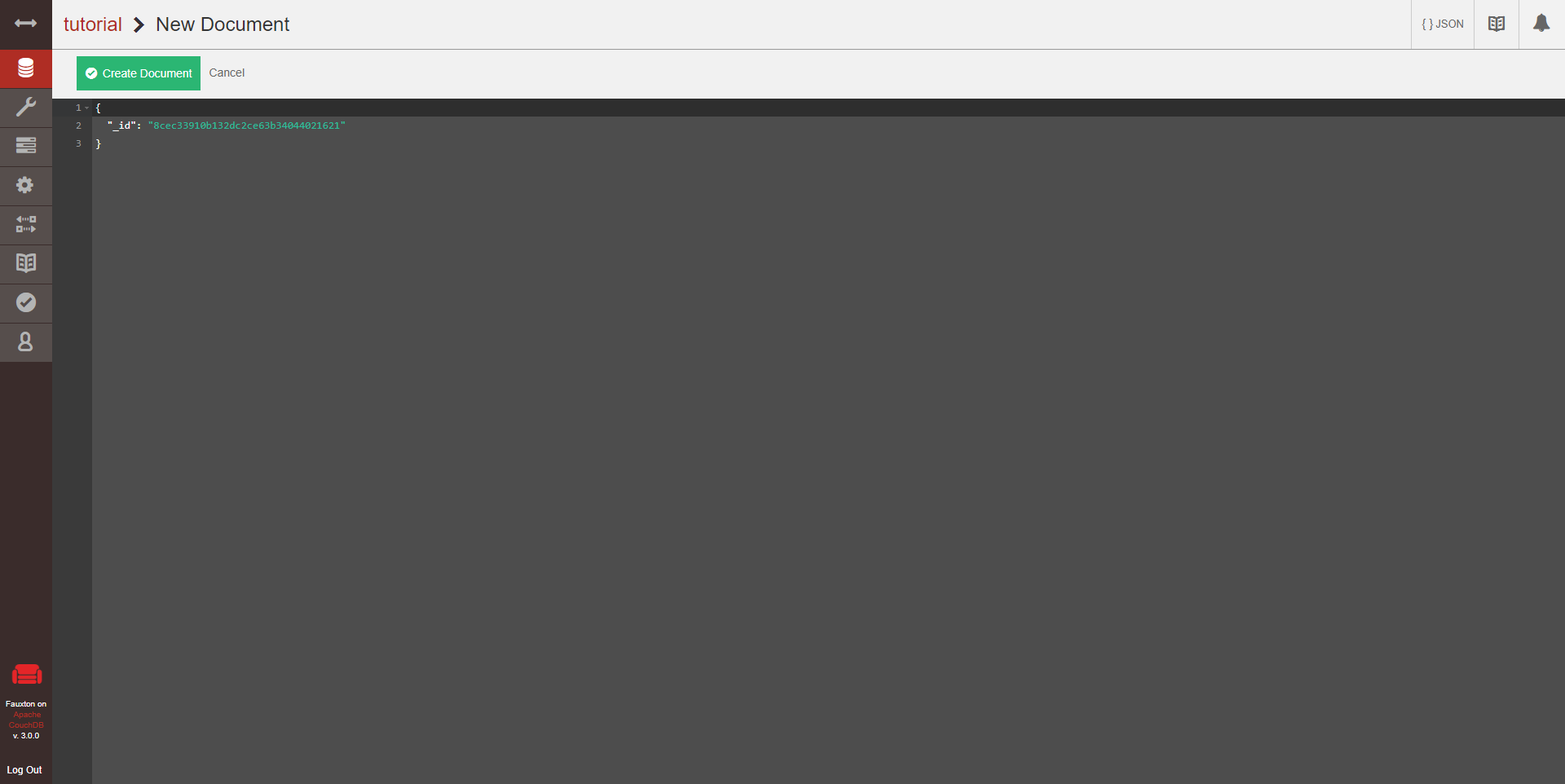


Una vez dentro de una base de datos en la parte superior derecha encontraremos un botón para crear una base de datos, nos dará también la opción de crear una base de datos particionada.

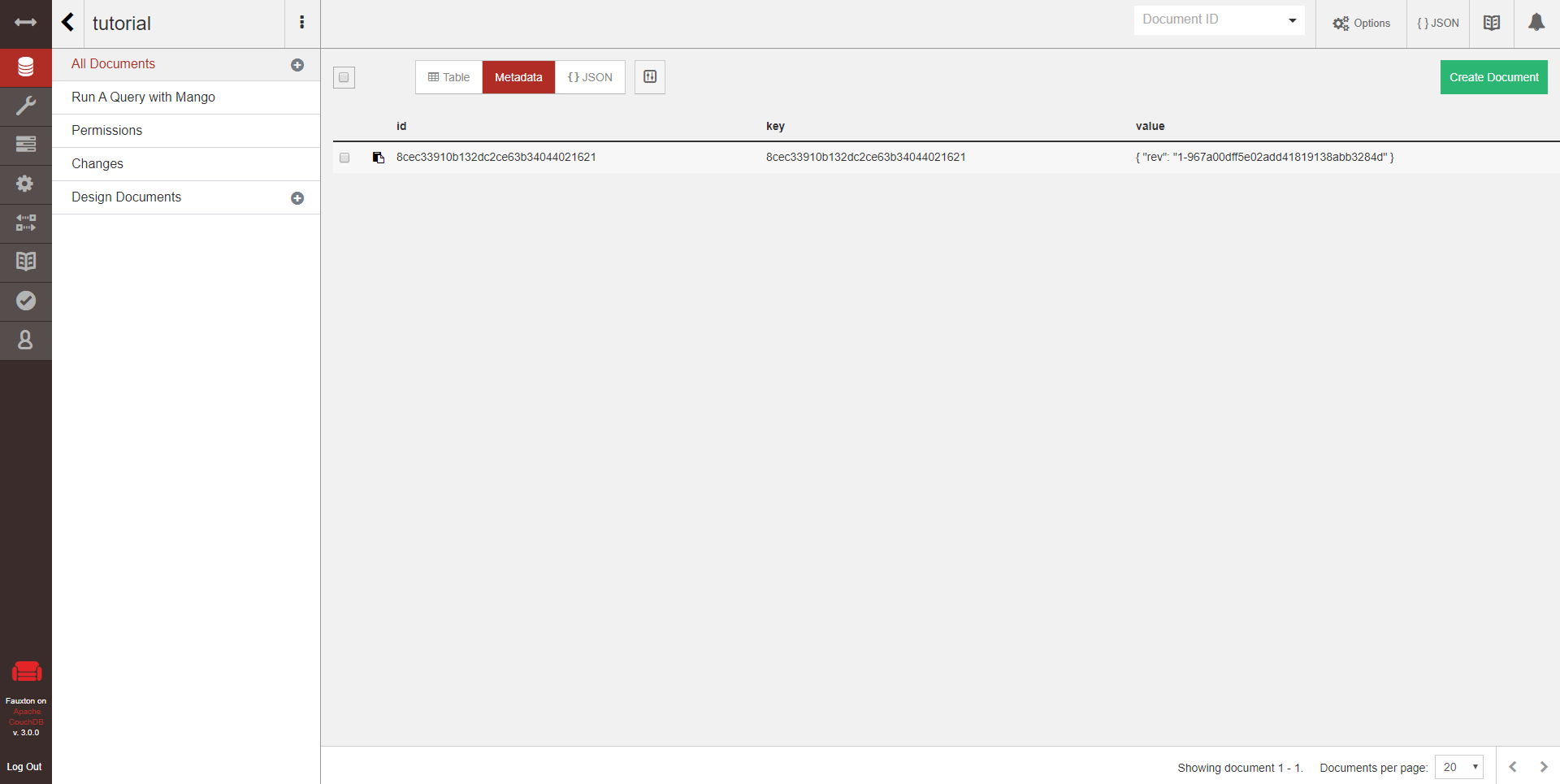
Una vez creada la base de datos nos encontramos en la vista donde irán apareciendo los documentos. Para crear un nuevo documento podremos clicar en la parte derecha en “Create Document” o en la parte izquierda en “All Documents” clicando en el botón “+” y en “New Doc”. Los documentos pueden ser visualizados en varios formatos, que aparecen en la zona superior central de la pantalla(Table, Metadata, JSON).



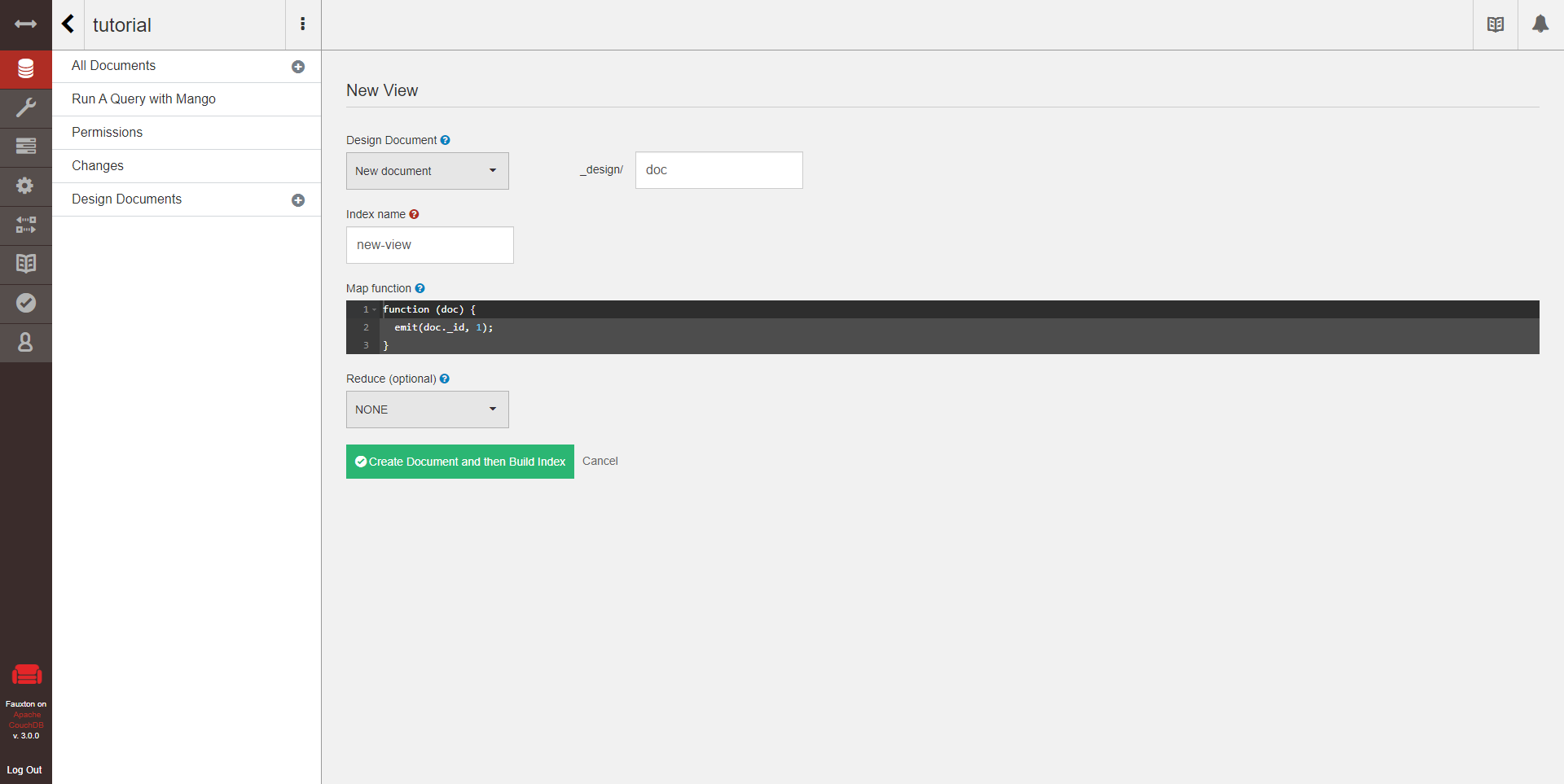
Aquí podremos crear un documento siempre siguiendo el formato JSON, si no cumplimos con este formato no nos dejará guardar el documento.



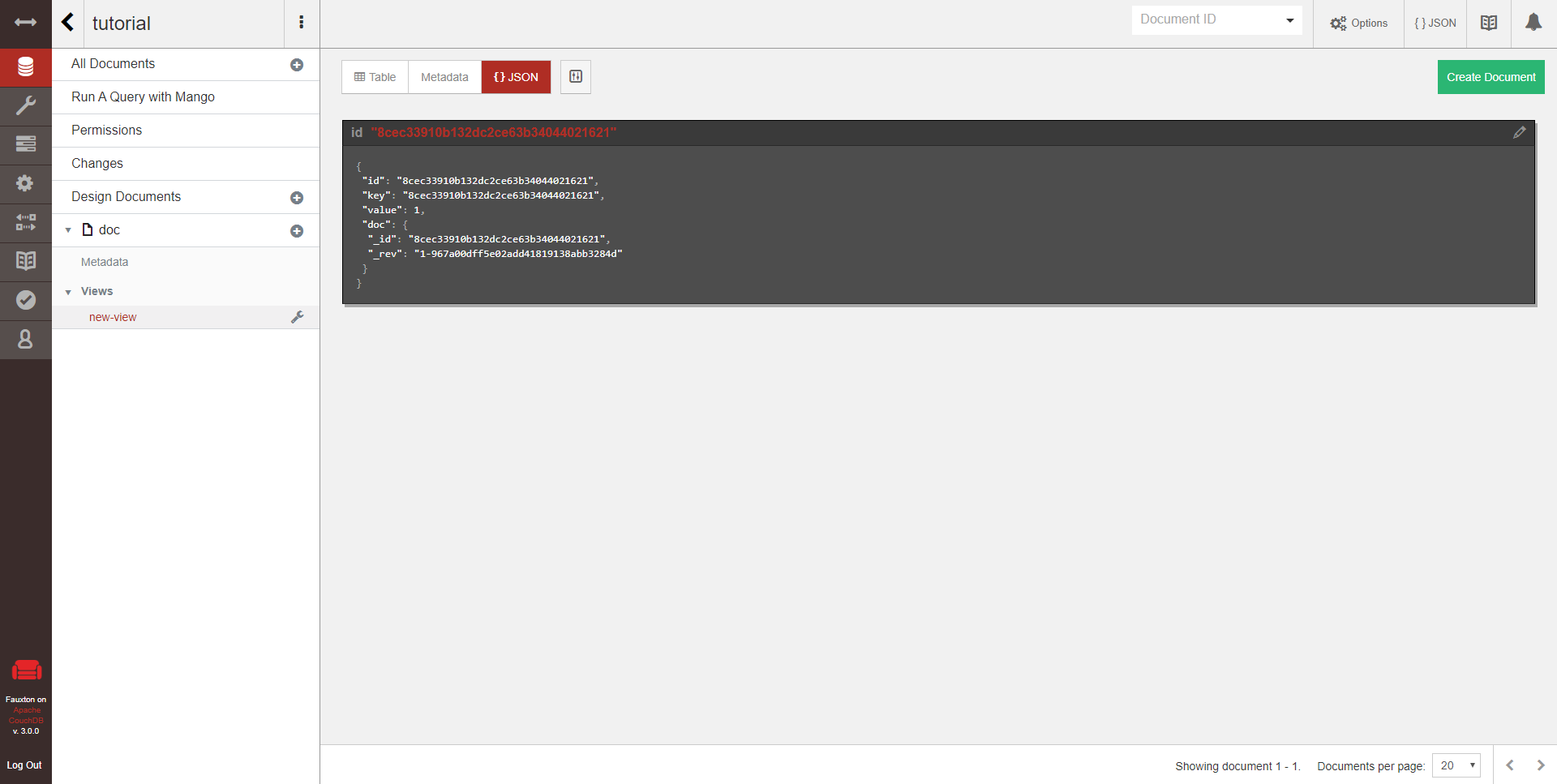
Ahora nos aparecerá el documento y bastará con clicar para poder editarlo. Dentro de la edición del documento nos permitirá eliminarlo con el botón “Delete” en la parte superior derecha. Para subir un archivo adjunto al documento también se puede hacer al editar el documento haciendo uso del botón “Upload attachment” situado cerca del botón “Delete”.



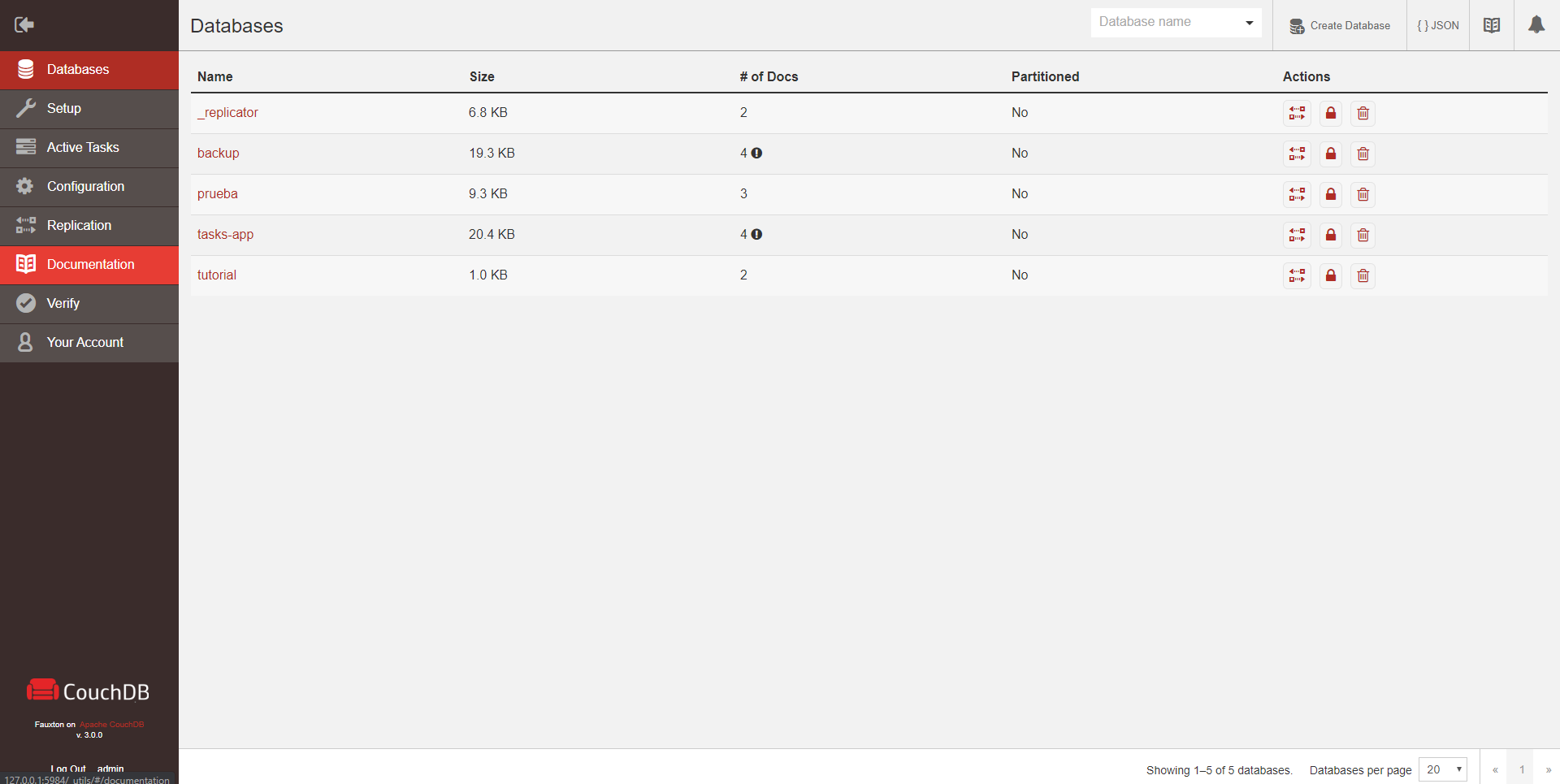
Para crear las vistas de nuestro sistema podemos clicar en el “+” de “All Documents” y seleccionar “New View”. Esto nos lleva a la vista de creación de vistas, en ella podremos ponerle nombre al “Design Document” que contendrá la vista creada y a la propia vista, el nombre de la vista será el campo “index-name”. Debajo de estos campos podremos definir en Javascript nuestras funciones MapReduce.



Ahora en la parte izquierda debajo de “Design Documents” tendremos el documento con la vista que se acaba de crear, si clicamos en él podremos clicar en la vista y ver los resultados que nos ofrecería la vista creada.

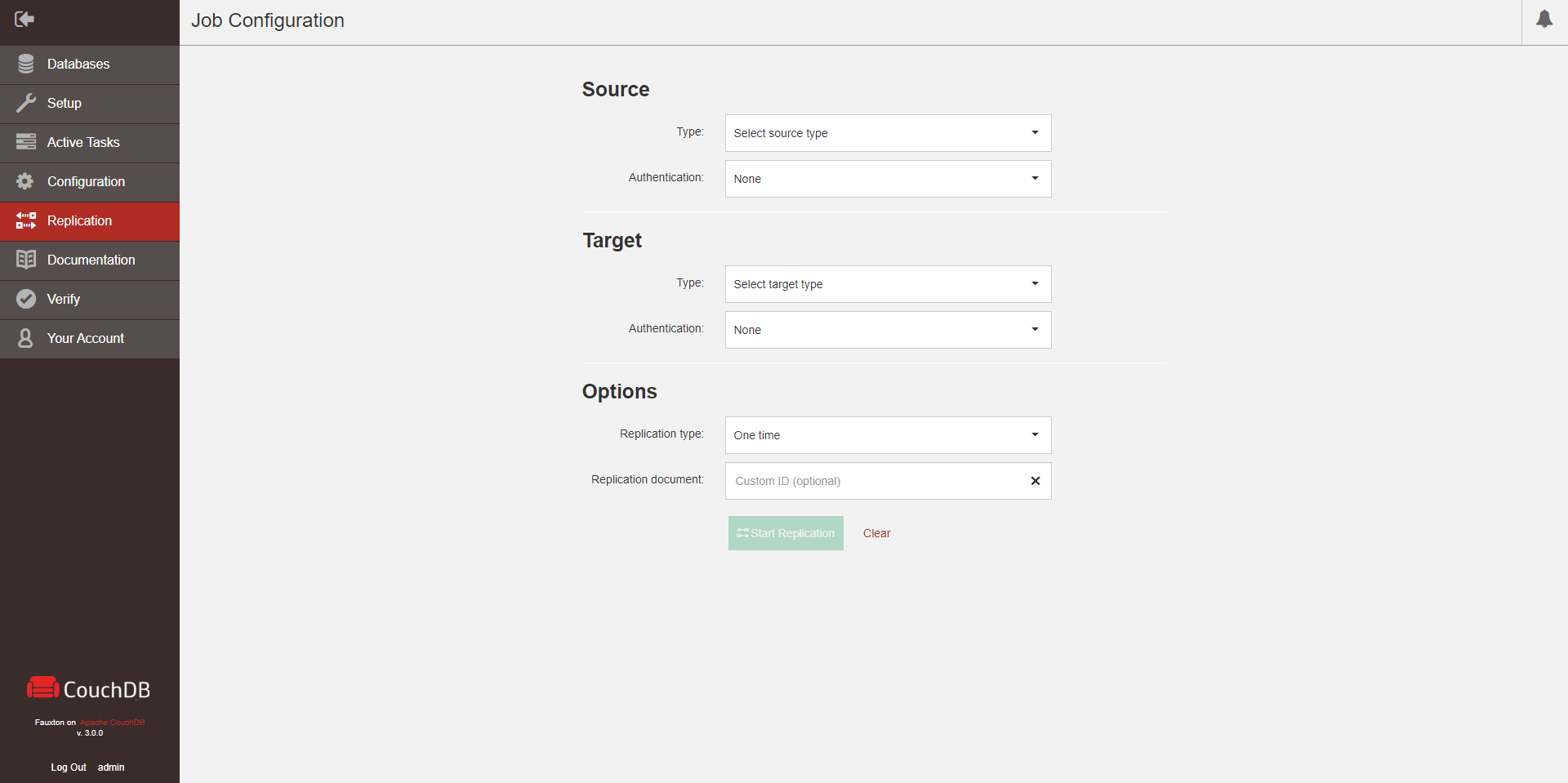


Ahora que ya hemos explicado el funcionamiento básico de las bases de datos de CouchDB en Fauxton pasaremos a ver qué otras cosas nos ofrece como gestor de bases de datos. De estas otra opciones solo se explicará cómo usar la replicación. Para esto desplegamos el menú situado en la zona izquierda de la pantalla.

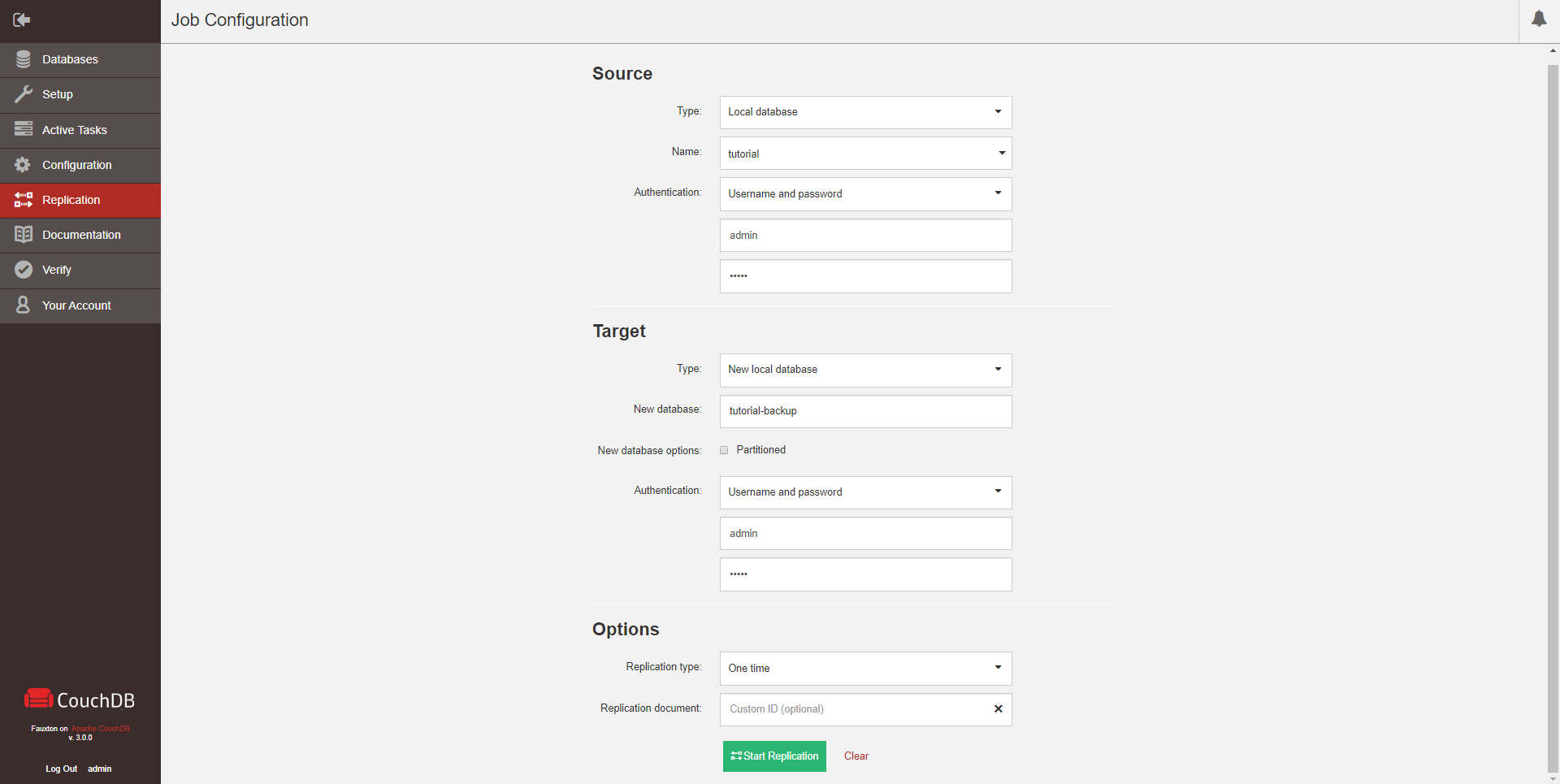


Vemos que existen distintos botones que nos ofrecen

* Setup: nos permite configurar una base de datos con un solo nodo o un clúster con varios nodos.
* Active Tasks: Aquí se nos muestran distintas tareas que se pueden programar en el gestor de base de datos como serían replicaciones o compactaciones de alguna base de datos.
* Configuration: Nos permite visualizar distintas configuraciones de CouchDB así como activar CORS(Cross-Origin Resource Sharing).
* Documentation: Nos muestra la documentación oficial de Apache CouchDB.
* Verify: Nos permite verificar que se ha realizado una instalación correcta de CouchDB.
* Your Account: Nos permite cambiar la contraseña de nuestra cuenta así como crear nuevas cuentas de usuario.
* Replication: Nos muestra las replicaciones que se han realizado o se están realizando. Si clicamos en “New Replication” podremos crear una nueva replicación de cualquier base de datos de forma sencilla y rápida.



Para crear una réplica de una base de datos sólo tendremos que configurar de qué base de datos a qué base de datos se va a realizar la replicación y si será una sola vez o continua. A continuación se muestra un ejemplo:



* **API**

Para la comunicación con la base de datos CouchDB nos ofrece una API REST por lo que podremos usar cualquier cliente HTTP, en nuestro caso para explicar el funcionamiento básico de esta API usaremos cURL.

Para obtener información de CouchDB hacemos:

**curl http://admin:admin@127.0.0.1:5984**

Y obtenemos como respuesta:

**{**

**"couchdb":"Welcome",**

**"version":"3.0.0",**

**"git\_sha":"03a77db6c",**

**"uuid":"530c3cc2a030e0c16f65d1de7ffbf6f9",**

**"features":[**

**"access-ready",**

**"partitioned",**

**"pluggable-storage-engines",**

**"reshard",**

**"scheduler"**

**],**

**"vendor":{**

**"name":"The Apache Software Foundation"**

**}**

**}**

Para obtener todas las bases de datos existentes hacemos:

**curl http://admin:admin@127.0.0.1:5984/\_all\_dbs**

Y obtenemos como respuesta(En nuestro caso tenemos varias bases de datos ya creadas):

**["\_replicator","backup","prueba","tasks-app","tutorial"]**

Si queremos crear una base de datos añadimos al final de la url el nombre que queremos para la base de datos:

**curl -X PUT http://admin:admin@127.0.0.1:5984/tutorial**

Como respuesta tendremos:

**{"ok":true}**

Si quisiéramos eliminarla bastaría cambiar el verbo HTTP PUT por DELETE quedando:

**curl -X DELETE http://admin:admin@127.0.0.1:5984/tutorial**

Para consultar datos sobre una base de datos basta con hacer un GET de la misma url que se usó para su creación:

**curl -X GET http://admin:admin@127.0.0.1:5984/tutorial**

Como respuesta se obtiene:

**{**

**"db\_name":"tutorial",**

**"purge\_seq":"0-g1AAAABXeJzLYWBgYMpgTmEQTM4vTc5ISXLIyU9OzMnILy7JAUnlsQBJhgYg9R8IshIZ8KhNZEiqhyjKAgBm5Rxs",**

**"update\_seq":"0-g1AAAABXeJzLYWBgYMpgTmEQTM4vTc5ISXLIyU9OzMnILy7JAUnlsQBJhgYg9R8IshIZ8KhNZEiqhyjKAgBm5Rxs",**

**"sizes":{**

**"file":16700,**

**"external":0,**

**"active":0**

**},**

**"props":{**

**},**

**"doc\_del\_count":0,**

**"doc\_count":0,**

**"disk\_format\_version":8,**

**"compact\_running":false,**

**"cluster":{**

**"q":2,**

**"n":1,**

**"w":1,**

**"r":1**

**},**

**"instance\_start\_time":"0"**

**}**

Para crear un documento dentro de la base de datos tendremos que hacer POST, añadir un header y escribir el json que será añadido:

**curl -X POST http://admin:admin@127.0.0.1:5984/tutorial -H "Content-Type: application/json" -d "{""nombre"":""prueba""}"**

Nota: Las dobles comillas es debido a que se está haciendo uso de cURL desde Windows.

Como respuesta obtenemos:

**{"ok":true,"id":"8cec33910b132dc2ce63b34044023863","rev":"1-1cbf9c085bd034db9424a79ce4d5db64"}**

Si queremos visualizar el documento completo que acabamos de crear basta con añadir detrás del nombre de la base de datos el id del documento:

**curl -X GET http://admin:admin@127.0.0.1:5984/tutorial/8cec33910b132dc2ce63b34044023863**

Como respuesta tenemos el documento completo:

**{"\_id":"8cec33910b132dc2ce63b34044023863","\_rev":"1-1cbf9c085bd034db9424a79ce4d5db64","nombre":"prueba"}**

Para eliminar un documento tendremos que añadir a la url el id del documento y el atributo rev de forma que quedaría:

**curl -X DELETE http://admin:admin@127.0.0.1:5984/tutorial/8cec33910b132dc2ce63b34044023863?rev=1-1cbf9c085bd034db9424a79ce4d5db64**

Como respuesta tenemos:

**{"ok":true,"id":"8cec33910b132dc2ce63b34044023863","rev":"2-bd109d977f696e3fe7cd7d3c0865b659"}**

Por último vamos a ver como obtener haciendo uso de la API el contenido devuelto por una vista. La url que hemos de usar es de la forma **http://admin:admin@127.0.0.1:5984/db/\_design/docname/\_view/viewname**

Siendo db el nombre de la base de datos docname el nombre del “Design Document” que contiene a la vista y viewname el nombre de la vista. Como ejemplo usaremos una vista creada en una base de datos llamada “prueba”, llamándose el “Design Document” “view1” y la vista “id”. La petición quedaría:

**curl -X GET http://admin:admin@127.0.0.1:5984/prueba/\_design/view1/\_view/id**

La respuesta que obtenemos en este ejemplo es:

**{**

**"total\_rows":2,**

**"offset":0,**

**"rows":[**

**{**

**"id":"1819a0a5e88715bc45ebe6617b001e57",**

**"key":"30696300V",**

**"value":{**

**"nombre":"Jesus",**

**"apellidos":"Garrocho Jimenez"**

**}**

**},**

**{**

**"id":"1819a0a5e88715bc45ebe6617b008120",**

**"key":"30696301U",**

**"value":{**

**"nombre":"Juanma",**

**"apellidos":"Garrocho Jimenez"**

**}**

**}**

**]**

**}**

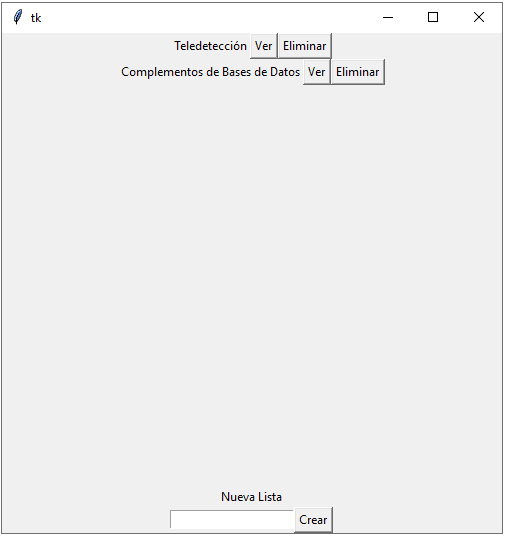
En la documentación oficial obtendremos mucha más información acerca de la API y sus capacidades, ya que a estas peticiones también se le podrán añadir parámetros que nos permitirán obtener distintas respuestas, como por ejemplo limitar el número de documentos que tendrá la respuesta.

**APLICACIÓN REALIZADA**

Para experimentar con CouchDB se ha realizado una pequeña aplicación de escritorio escrita en Python, lenguaje escogido por ambos integrantes del grupo debido a su familiaridad con él. Ésta aplicación nos da la posibilidad de crear listas de cosas a realizar. Para el desarrollo de la aplicación en lugar de usar la API ofrecida por CouchDB hemos decidido utilizar una librería de Python y así explorar una alternativa distinta a la ya conocida API. Esta librería es “cloudant” la cual permite interactuar con CouchDB por medio de objetos de Python. A continuación se muestra la aplicación y el uso de la librería.

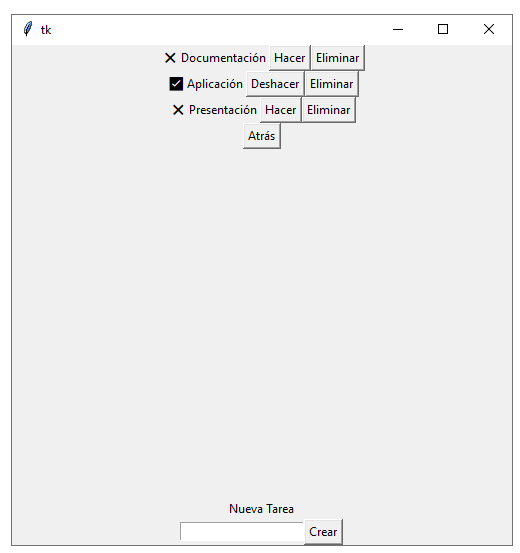
Además de cloudant también se ha usado tkinter para crear la interfaz gráfica de la aplicación.

**USO DE LA APLICACIÓN**



En la parte superior podemos ver las listas creadas y en la parte inferior podemos introducir el nombre y crear una nueva lista.

Si clicamos en ver en una lista nos mostrará las tareas a realizar, por ejemplo, clicamos en Complementos de Bases de Datos.



En la parte superior vemos las tareas a realizar mientras que en la parte inferior podemos añadir nuevas tareas, estas tareas se pueden marcar como hechas o desmarcar, también se pueden eliminar. Si clicamos en el botón de atrás volveremos a la vista anterior.

**Detalles técnicos**

A continuación hablaremos de cómo están estructurados los elementos en la base de datos y cómo interactuamos con esta mediante la librería “cloudant”.

Se ha creado una base de datos llamada “tasks-app”, dentro de esta se guardan los documentos los cuales tienen esta forma:

**{**

**"\_id": "8cec33910b132dc2ce63b34044025b50",**

**"\_rev": "5-e31d50257d6cdfd8e0d01c91c3bdae92",**

**"name": "Complementos de Bases de Datos",**

**"tasks": [**

**{**

**"name": "Documentación",**

**"status": "0"**

**},**

**{**

**"name": "Aplicación",**

**"status": "1"**

**},**

**{**

**"name": "Presentación",**

**"status": "0"**

**}**

**]**

**}**

Se ha realizado una vista llamada “nombre” dentro de un “Design Document” llamado “listas” el cual devuelve como clave el id del documento y como valor el nombre de la lista.

Para interactuar con la base de datos por medio de la librería cloudant hacemos lo siguiente:

En primer lugar importamos las clases de las que haremos uso:



Conectamos con CouchDB para ello, el objeto bdd contiene la base de datos de la aplicación:



Para la primera vista necesitaremos hacer uso de la vista “nombre”, que nos devuelve los nombres de las listas guardas y el id de cada una. Para llamar a esta vista desde python primero obtenemos el “Design Document” que contiene a esta vista, tras esto obtenemos la vista y por último accedemos a un atributo “rows” que contiene los resultados que devuelve la vista:



Al crear una nueva lista de tareas en la aplicación se debe crear un nuevo documento en la base de datos esto se hace así:



Para eliminar una lista basta con encontrar el documento mediante su id y hace uso del método delete() que ofrece la librería:



Por último el resto de interacciones que se hacen con la aplicación son para añadir/eliminar tareas o cambiar el estado de estas, esto se hace modificando los documentos de forma que recibiendo el id del documento y la tarea a modificar se cambia en la lista de tareas del documento y se guarda. Todos estos métodos se hacen de forma similar cambiando el atributo “tasks” de los documentos por tanto a continuación se muestra la creación de una nueva tarea:



El resto del código de la aplicación es código Python relacionado con la interfaz de la aplicación por lo que no se explicará en este apartado. Puede consultar este código en el archivo entregado con el código al completo de la aplicación el cual se encuentra comentado para una fácil comprensión.

Nota: Tras hacer uso de la aplicación en Fauxton aparece un icono avisando que la base de datos tiene X documentos e Y documentos borrados, esto es debido a que CouchDB no borra los documentos sino que pone un atributo oculto de los documentos llamado “\_deleted”:True de forma que este documento no aparecerá en las consultas pero sigue estando en la base de datos por si se quisiera recuperar. Para eliminar estos documentos habría que hacer lo que en couchDB se denomina como “purgar” la base de datos, esto consideramos que estaría fuera del alcance de nuestro proyecto por lo que no se explica.