Proyecto NOSQL

White Group

IFCD210 - Desarrollo aplicaciones con tecnología web
MF0491
Daniel Colomer
Abril 2018

Índice

Informe técnico de proyecto

Introducción

Con este trabajo pretendemos introducir conceptos fundamentales sobre el diseño de bases de datos NoSql usando PostgreSQL adquiriendo conocimientos teórico-prácticos en manejo de bases de datos relacionales, como son la conexión de la base de datos en el lenguaje Java, la programación en este lenguaje y las hojas de estilo CSS y sus respectivos documentos html.

Con este informe también pretendemos proporcionar la información necesaria para poder realizar un uso básico postgreSQL y Java para alguien que no tuviera conocimientos con anterioridad.

Con este trabajo también pretendemos observar las ventajas y dificultades que tienen el manejo de estos tipos de datos en comparación con otras bases de datos.

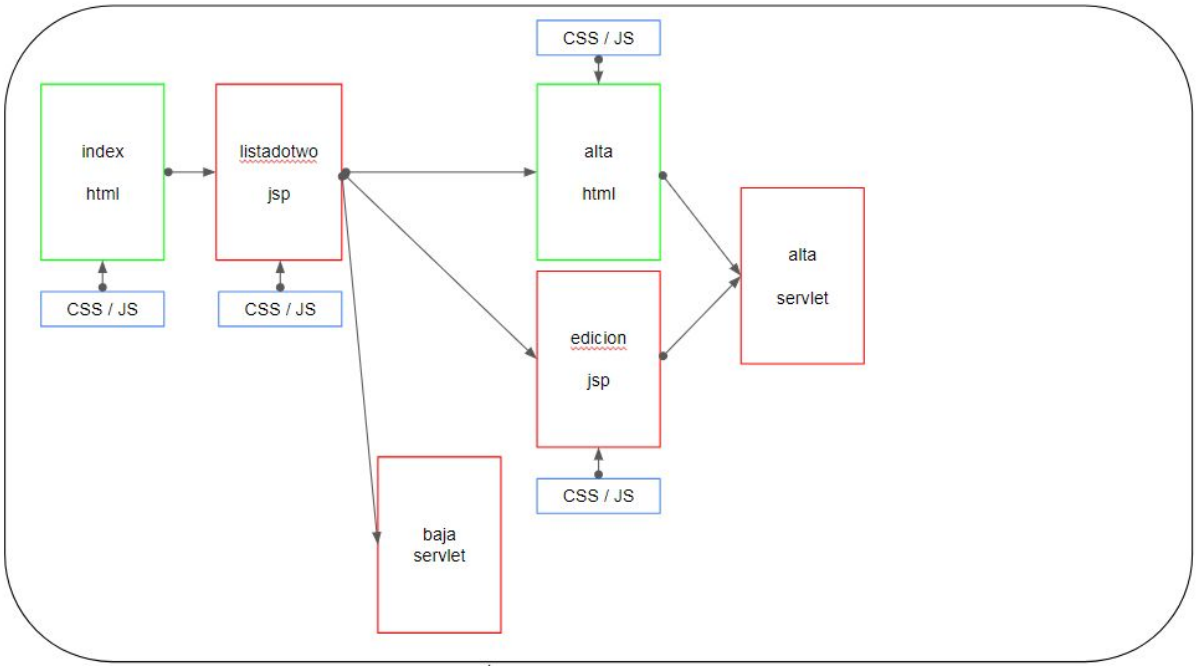
Los componentes del grupo somos:

- Ricard Alvira (FrontEnd)
- Daniel Paré (BackEnd)
- Luis Vecino (BackEnd & FrontEnd)
- Miguel Angel Torres (BackEnd)

Detalle de funcionamiento de la aplicación

Utilizamos cookies que hemos implementado con Javascript. Siguiendo las indicaciones que marca la ley.

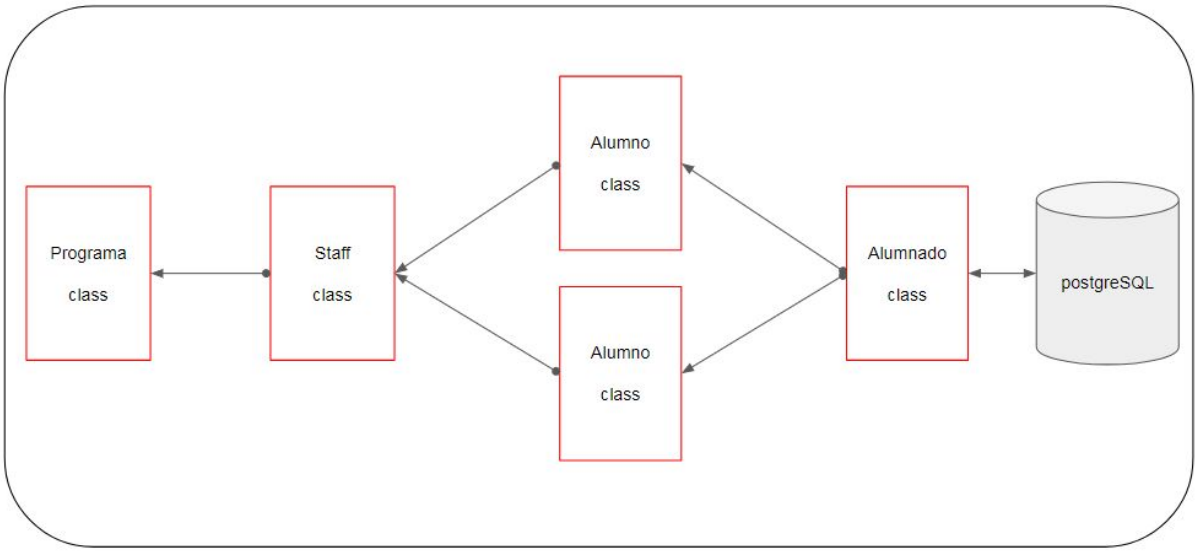
Hemos hecho uso de las diferentes tecnologías aprendidas durante el curso para el desarrollo de la misma: HTML, CSS, JS, Java, BDD



Hemos desarrollado 2 maneras de acceso: una vía consola (básicamente para desarrollo y testeo de las clases) y una vía web de cara al público.

La estructura de clases vía consola ha sido diseñada para que permita reemplazar las clases de esa parte (Programa.class, Staff.class) por la parte HTML+jsp/servlet, usando el resto de las clases para la lógica de la aplicación así como la persistencia en la base de datos.

Estructura de las clases usadas en la versión de consola (y reaprovechadas en parte para la versión WEB)



GITHUB:

https://github.com/matebcn/proyecto-nosql.git

Material utilizado en el desarrollo de la aplicación.

Trabajo realizado con la plataforma Oracle Netbeans 8.2. Utilizamos la tecnología JSP, porque nos da la posibilidad de utilizar glassfish, para poder realizar el proyecto web.

Librerías utilizadas en el proyecto

- **postgresql-42.2.2.jar** : conexión con la BDD
- **json-simple-1.1.jar** : recuperar la información de un String JSON, parseando la información y permitiéndonos trabajar con las clave-producto de nuestra tabla

Empresas que han dado su confianza a postgreSQL.

Tecnología de la base de datos

PostgreSQL tiene muchas cualidades que lo hacen ser una muy buena alternativa para instalar sistemas en empresas, universidades y una gran cantidad de otras aplicaciones.

PostgreSQL es un avanzado sistema de bases de datos relacionales Open Source, es decir, que cualquier persona puede colaborar en su desarrollo y modificación.

Una sesión en PostgreSQL consiste en la ejecución de el servidor y de la aplicación cliente:

- El servidor es el que maneja archivos de bases de datos, acepta conexiones a las aplicaciones cliente y actúa en la base de datos.
- La aplicación cliente es la que realiza operaciones en la base de datos. Estas pueden ser aplicaciones de texto en consola, gráficas, un servidor web que acceda a la base de datos para mostrar una página o herramientas especializadas para el manejo de bases de datos.

Destacables:

- Open Source.
- Multiplataforma.
- Alto volumen (Big-Data).
- Seguridad de la información.
- Facilidad de manejo.

Queries realizadas para la creación de nuestra base de datos

1. Una vez conectados a el servidor de postgreSQL crearemos la siguiente consulta para crear nuestra base de datos.

```
CREATE DATABASE alumnos
     WITH
     OWNER = postgres
     ENCODING = 'UTF8'
     CONNECTION LIMIT = -1;
```

2. Creamos las tabla “alumnos” con sus diferentes campos, incluyendo el **active*** y los atributos en características **jsonb*.**

```
CREATE TABLE alumnos (
      id serial NOT NULL,
      nombre text,
      apellido text,
      mail text,
      active integer default 1,
      caracteristicas jsonb
);
```

***Jsonb:** almacena los datos en un formato binario y además permite índices. Debido a la conversión, el formato jsonb en comparación con json puede ser un poco más lento al guardarlo, pero gracias al índice es mucho más rápido en las búsquedas.

***Active**: El campo active es por defecto 1. Esto significa que el registro está activado. Si fuera 0 significa que el usuario se ha dado de baja en la base de datos de alumnos. Gracias a este campo podremos dar de baja al alumno sin borrarlo de la base de datos.

3. Para insertar columnas y valores en la tabla que hemos creado en el paso 2 escribiremos las siguientes consultas de tipo **INSERT**:

detalle:

```
INSERT INTO alumnos (
      nombre,
      apellido,
      mail,
      active,
      caracteristicas
) VALUES (
      'Luis',
      'Vecino’,
      vecino@luis.gll ,
      1,
      '{"pelo":"poco","ojos":"marrones"}'
);
```

otros ejemplos:

```
INSERT INTO alumnos
(nombre,apellido,mail,active,caracteristicas) VALUES
('Daniel', 'Pare',
'pare@daniel.cat',1,'{"pelo":"moreno","ojos":"marrones"}');

INSERT INTO alumnos
(nombre,apellido,mail,active,caracteristicas) VALUES
('Ricardo', 'Llull',
'llull@ricardo.es',1,'{"pelo":"castaño","ojos":"verdes"}');

INSERT INTO alumnos (nombre,apellido,mail,active,
caracteristicas) VALUES ('miguel', 'mate',
'mate@miguel.es',1, '{"pelo":"rojo","ojos":"rojo"}');
```

Queries de consulta a la base de datos

1. Para hacer consultas en las tablas de la base de datos utilizaremos la sentencia SELECT como en el siguiente ejemplo:

   ```
   SELECT * from ALUMNOS;
   ```

   donde * serían todos lo campos que queremos consultar perteneciente a la tabla alumnos.

	id integer	nombre text	apellido text	mail text	active integer	caracteristicas jsonb
1	1	Luisito	Fernander	dpoijdan@kk.es	1	{"ojos":"azules","pelo":"rubio"}
2	2	Danielito	Fernander	dan@kk.es	1	{"ojos":"marrones","pelo":"moreno"}
3	3	Ricardito	Fernander	poij@kk.es	1	{"ojos":"verdes","pelo":"castaño"}
4	4	Miguelito	Fernander	dan@kk.es	1	{"ojos":"rojo","pelo":"rojo"}
5	5	Luisito	Fernander	dpoijdan@kk.es	1	{"ojos":"azules","pelo":"rubio"}
6	6	Danielito	Fernander	dan@kk.es	1	{"ojos":"marrones","pelo":"moreno"}
7	7	Ricardito	Fernander	poij@kk.es	1	{"ojos":"verdes","pelo":"castaño"}
8	8	Miguelito	Fernander	dan@kk.es	0	{"ojos":"rojo","pelo":"rojo"}

PostgreSQL devuelve un conjunto de resultados en forma de JSON y proporciona dos operadores nativos -> y ->> para ayudarnos a consultar este tipo de datos.

El operador -> devuelve el campo de objeto JSON por clave.
El operador ->> devuelve el campo de objeto JSON por texto.

Por ejemplo:

```
SELECT caracteristicas -> 'ojos' , nombre
FROM alumnos
WHERE caracteristicas ->> 'ojos' = 'marrones';
```

De esta forma seleccionamos del campo jsonb los atributos que nos interesen.

Queries realizadas para modificar la base de datos

- Actualizamos un registro cuando modificamos alguno de sus valores. Para modificar uno o varios registros utilizamos "UPDATE".

- Igual que con las consultas mySQL, utilizamos "UPDATE" junto al nombre de la tabla y "SET" junto con el campo que queremos modificar y su nuevo valor. La cláusula WHERE la utilizamos para especificar una condición.

  Por ejemplo:

  ```
  UPDATE alumnos SET nombre = 'Manolo' WHERE id = '1';
  ```

Reflexiones

- Hemos aprendido aspectos de instalación, configuración, creación y administración de sistemas de bases de datos, así como también aspectos teóricos acerca de los fundamentos de postgreSQL.
- Falta documentar el código para crear una documentación que pueda servir para su uso por otros programadores.

Conclusiones

que ha ido bien

- Enfrentarnos al reto de trabajar en grupo, aprendiendo virtudes y defectos de nuestros compañeros.

que ha ido mal

- Trabajar sobre la marcha sobre una (o varias) tecnologías que no controlas hace que se modifique el proyecto con cada cosa que aprendes al hacer frente a un problema.

cómo ha ido el trabajo en grupo

- Hemos aportado nuevas ideas e información entre los miembros del grupo y consideramos necesario el trabajo en grupo fuera de las sesiones de clase.
- Nos ha faltado coordinación / comunicación a la hora de desarrollar el proyecto, ya que aunque hemos hecho un reparto de tareas con la intención de no solapar nuestro trabajo, ha habido ocasiones en la que esto ha sucedido.
- Nos ha faltado una planificación previa más exhaustiva, para que las piezas que cada uno iba desarrollando encajasen de una manera más efectiva. Hemos invertido 2 días casi todos los recursos (humanos) en realizar este trabajo.

Referencias bibliográficas

SQL

- https://www.w3schools.com/sql/default.asp
- http://www.tutorialesprogramacionya.com
- http://www.postgresqltutorial.com/postgresql-json/

Java

- http://www.postgresqltutorial.com/postgresql-jdbc/delete/
- http://zetcode.com/java/postgresql/
- https://www.tutorialspoint.com/postgresql/postgresql_java.htm
- https://stackoverflow.com/questions/23325800/replace-the-last-occurrence-of-a-string-in-another-string
- https://stackoverflow.com/questions/46898/how-to-efficiently-iterate-over-each-entry-in-a-map
- https://stackoverflow.com/questions/5122913/java-associative-array

Java + JSON

- https://www.mkyong.com/java/json-simple-example-read-and-write-json/
- https://stackoverflow.com/questions/30757136/converting-string-to-json-object-using-json-simple/30757275
- http://theoryapp.com/parse-json-in-java/
- https://stackoverflow.com/questions/9151619/how-to-iterate-over-a-json object
- https://github.com/fangyidong/json-simple

Javascript

- http://www.joserodriguez.info/bloc/resolver-la-politica-de-cookies-en-tu-web/comment-page-1/