Instructions for a postgreSQL-compatible implementation of dbiCal

1. Download dbiCal from:

   http://kigkonsult.se/downloads

2. Extract the "dbiCal-{version}" folder contained in the zip to an administrative folder of your choosing on your web server

   (**Note:** the web-based administrative functions of dbiCal require PHP on your server; don't forget to set folder permissions for your web server, including the correct ownership...)

3. In the dbiCal administrative folder, edit "index.php"; replace the line:

   ```
   define( 'DBICALDSN', 'mysqli://dbiCal:dbiCal@localhost/dbiCal2' );
   ```

   with:

   ```
   define( 'DBICALDSN', 'pgsql://dbiCal@localhost/calendar' );
   ```

   “`calendar`” is the name of the database for your calendar data; you can choose another name if you prefer. Also, some PHP implementations do not have the postgres drivers loaded by default; you might need to do that (perhaps the package “php5-pgsql”...)

   [Note that we are presuming here that your SQL runs on the same server as your web; modify "`localhost`" on the second line to an appropriate server if that is not the case...]

4. Download (the postgres compatible) dbiCal.sql from:

   http://osfda.org/downloads

5. Upload dbiCal.sql into the "sql" subfolder of the dbiCal administrative folder you made in step (2)

   (overwrite dbiCal.sql, if necessary...)

6. Create a Group Role in your postgres server called "calendaring";
   create a Login Role in your postgres server called "dbiCal";
   make dbiCal a member of group "calendaring".

   [These accounts are configured for "least privilege"; once you get confident using dbiCal, you can change these accounts to suit by modifying the accounts referenced in the SQL script you downloaded from osfda.org and rerunning it...]

7. Create a database for the calendar, using whatever name you chose in step (3); make the owner of the database "dbiCal".

   [You can eventually have the iCalendar tables reside in a preexisting database with other tables, but experiment with a new standalone one for now!]

8. Using a SQL client tool, connect to the calendar database made in step (5) and run the SQL script "dbiCal.sql" downloaded in step (3).

9. Important security step: restrict access to the dbiCal adminstrative folder on your web server (so only you can load and update calendar data...)

   For Nginx webserver instructions, see:

   http://www.howtoforge.com/basic-http-authentication-with-nginx

   For Apache webserver instructions, see:

   https://wiki.apache.org/httpd/PasswordBasicAuth

10. Now visit the dbiCal administrative folder in your browser (log in, if necessary...)

    You should be able to browse and upload calendar files exported from iCalendar-compatible mail clients (like Thunderbird Lightning, Outlook...) Using pgadmin or another SQL administrative tool of your choosing, you will see the calendar data being populated in the tables.

11. **Elective FullCalendar JavaScript integration** (EXTREMELY cursory!):

    FullCalendar allows you to render your SQL event data (now in an iCalendar model layout thanks to the previous steps...) to a web browser using JavaScript/Jquery. It is even possible to accommodate browser-based editing of SQL calendar data using FullCalendar events (though this is by no means a trivial task...)

    FullCalendar has a "get-events.php" file in its demos subdirectory; this loads a JSON file into FullCalendar. Using this as a template (or not...), you can implement your own servlet that handles AJAX requests from FullCalendar (it could be done on the web server side using JSP, Python, PHP -whatever you prefer...)

    On the webserver servlet, your SQL query can be:

```
SELECT event_id,
       event_uid AS uid,
       event_startdatetime AS start,
       event_enddatetime AS end,
       event_summary AS title,
       event_location AS location,
       event_description AS description,
-- you could use a SQL function to enable advanced formatting,
-- "smart links", etc.in your event description text...
       attachment(event_id) AS url
-- “attachment” is a simple function that just returns a single URL
-- (iCalendar events can have binary data as attachments,
--  multiple attachments...)
       FROM event
```

[Note that the choice of these query column names is important for FullCalendar; it looks for them to render event data, particularly: “start”, “end”, “title”, “location”, “description”, and “url”...]

And in the FullCalendar configuration startup (JavaScript) object, you can use an "events:" item to indicate the URL of the servlet, for retrieving event data.

One article on working with FullCalendar can be found here:

http://www.templatemonster.com/help/js-animated-how-to-work-with-fullcalendar-plugin.html

Fortunately FullCalendar is popular these days, so web searches for answers to implementation problems usually turn up helpful advice!