Components and Connectors in a Simple Web App Architecture

Learning Objectives:

After completing this activity the students will be able to:

- Define software components and connectors
- Create an architecture for a simple web application with client, server API, server, and database components.

Prerequisites:

This activity will refer to the source code from the React Webapp Tutorial. You must complete the tutorial before starting this activity.

A. Components and Dependencies

Examine the code in `server.js` file and answer the following questions:

1. **Which statement retrieves food matches from the database?**
   Line 39, const r = db.exec(`
   select ${COLUMNS.join(', ')} from entries
   where description like '%${param}%'
   limit 100
   `);

2. **What specific columns are being retrieved from the database table?**
   const COLUMNS = [
   'carbohydrate_g',
   'protein_g',
   'fa_sat_g',
   'fa_mono_g',
   'fa_poly_g',
   'kcal',
   'description',
   ];

3. **Which specific database file is being used to retrieve the data?**
   'db/usda-nnd.sqlite3'

4. **Answer yes/no and explain your answer to the following question: Would `server.js` code continue to work if we:**

   a. **Removed column `protein_g` from the database table?**
      I think it would not work, unless there was another column added. Since the format must match in terms of # of Columns in the server.js file.
   b. **Added a column called `serving_size_g` to the database table?**
      No because we would need that column added on the server.js file.

5. **Does `server.js` depend on**
   a. **Database schema?**
      Yes
   b. **Database implementation (type of database used)?**
      Yes

c. **Database location?**
   Yes

6. **Does database implementation depend on**
   a. **Server technology?**
      No
   b. **Server location?**
      No
   c. **Server implementation?**
      No
7. **Provide a general statement describing the dependency between `server.js` and database.**
   The database does not depend on the server.js but the server depends heavily on the database.

The database and `server.js` are two of the components in our simple web application. A **component** is a logical unit of releasable code. It is up to us to decide how to split code into components. Typically, different components will represent different application concerns.

8. **What are the concerns of the server component (`server.js`) in our application?**
   Server.js is responsible for implementing the routes of our webapp using the express framework. Whenever the server gets a request on a particular route it must read the query and fetch data from the database accordingly. Then it must format this data to json and generate an HTTP response.

9. **What are the concerns of the database component (`db` directory) in our application?**
   Db directory holds the raw data that the sql client fetches for our app.
10. **What are the concerns of the client component (`client` directory) in our application?**
   Client component holds the frontend of our application. It is a react app that uses fetch to get food data from our backend and then populates the UI.

The client component is implemented with React framework, which uses JavaScript, html, and css files. These are static (non-changing) files that get loaded into your browser when you point your browser to HOST:PORT where HOST is the host that serves these static files and PORT is the port on which these static files are being served.

11. **If we change the name of the server component (`server.js`) in our application, would the client component continue to work?**
   Changing the name of server.js wouldn't affect the client component.
12. **The code where the client component interacts with the server is in the `client/src/Client.js` file. Which specific statement in that file interacts with the server?**
   The code is inside the search method:

```
fetch(`api/food?q=${query}`, {
   accept: "application/json"
})
```

13. **Provide a general statement describing the dependency between the client and the server components.**

Server requires client to make a proper HTTP request with ‘query’ variable in order to send a response. The client needs the server to return data in a format that it can understand.

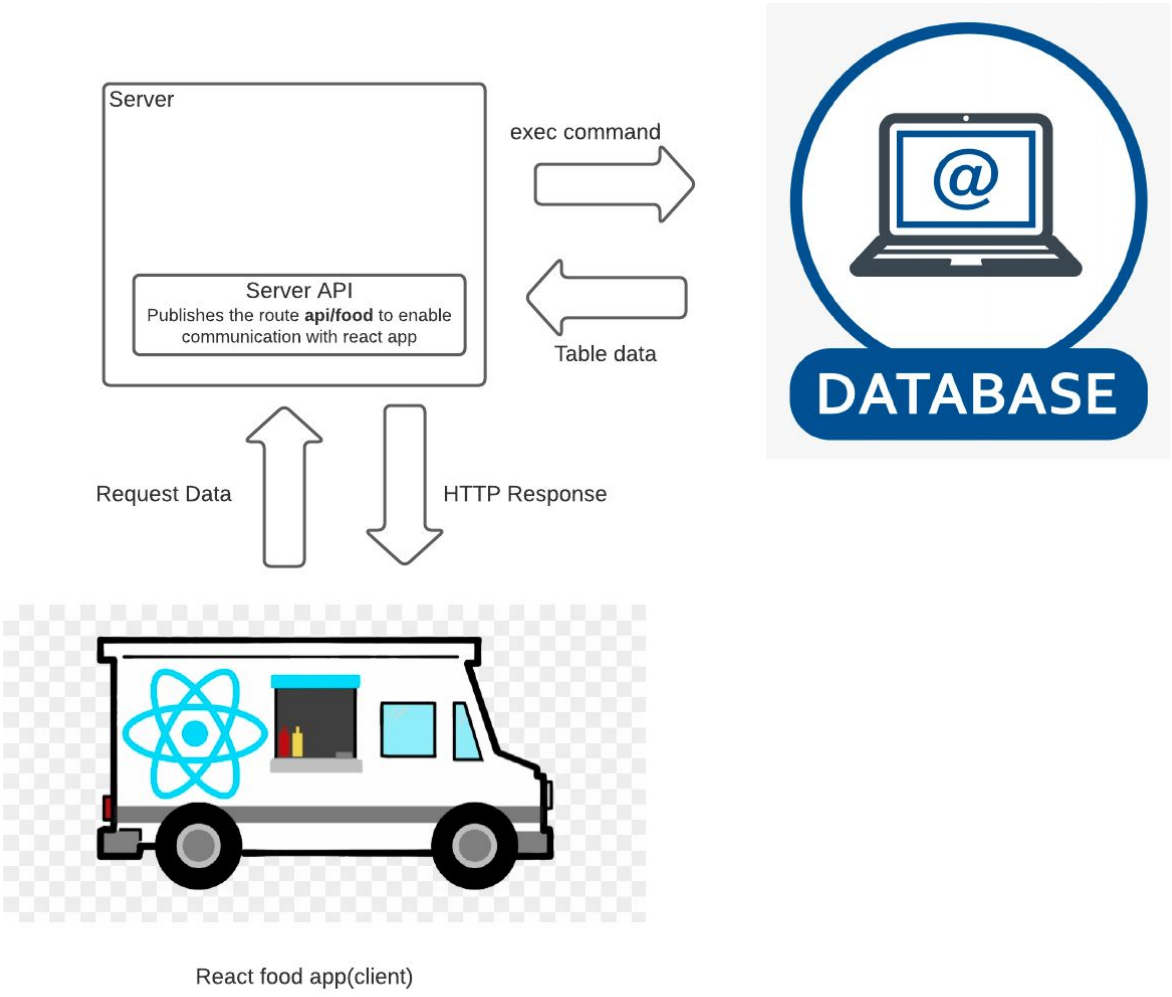
We say that the server exposes an Application Programming Interface (API) that the client component can use to retrieve the necessary data. We can think of an API as a collection of functions described by their names, inputs, and outputs. In the web API, the function name is the route the client will use to access the function, the inputs are there parameters being supplied, and the outputs is the data being returned to the client.

14. **Describe the API that `server.js` exposes in the food lookup application in terms of routes, inputs, and outputs.**

/api/food?q=aman

Above is the route which is exposed. q is the URL variable.

15. **Draw a dependency diagram between the following components of the food lookup application: client, server API, server, database. Point the arrow in the direction of dependency. For example, if the server depends on the database, you would have an arrow pointing from server to database.** ***To insert this diagram, use select Insert->Drawing->+New from the Google Docs menu.***



B. Connecting the Components

Let's consider the different ways that components interact with each other.

16. **Consider how exactly the server communicates with the database in order to retrieve the matches for a given query (you may want to review your answer to question 1). Using the following documentation of sql.js describe this communication in terms of**

    a. **Inputs and Outputs:**
    Input: URL Variables (query in terms of food-lookup, const param = req.query.q;)
    Output: (Array<QueryResults>) — An array of results (from data base). In terms of food-lookup, the column data was returned.

    b. Invocation type (explicit method call or implicit invocation through exception:
    Explicit method call, there was no Try-catch system involved with the db.exec command.

    c. **Synchronicity (synchronous - executed right away in the same thread or asynchronous - scheduled to be executed at some time in the future)**
    Synchronous

17. **Consider how exactly the client communicates with the server to get the matches on a query (you may want to review your answer to question 12). You can reference this article for the specifics of the `fetch` call. Describe the communication in terms of:**

    a. **Inputs and Outputs**
    Input: URL (in terms of food-lookup, `api/food?q=${query}`)
    Output: Front End Application(in terms of food-lookup,"application/json")

    b. **Invocation type**
    Implicit Invocation, it has a function called checkStatus in client.js to throw errors.

    c. **Synchronicity**
    Code is asynchronous. The request for data is made to the backend by the fetch command. Then the react state is updated when the json data is formatted asynchronously in the background.

18. **Explain why it is important to understand the synchronicity of a method call that communicates between components?**
    There can be a delay or stale data if components communicate in an asynchronous way.

19. **Let's revisit the answers to 16.a and 17.a.**

    a. **What is the output format used in 16.a?**

    { columns:
      ['carbohydrate_g',
        'protein_g',

```
    'fa_sat_g',
    'fa_mono_g',
    'fa_poly_g',
    'kcal',
    'description' ],
  values:
   [ [ 13.3, 0, 0, 0, 0, 51, 'Carbonated bev, crm soda' ],
     [ 0,
       0,
       null,
       null,
       null,
       0,
       'Beverages, h2o, btld, non-carbonated, dannon' ],
     [ 0,
       0,
       null,
       null,
       null,
       0,
       'Beverages, h2o, btld, non-carbonated, pepsi, aquafina' ] ] }
```


b. **The output format of 17.a is an HTTP response message with some data. What is the format of that data?**

[{"carbohydrate_g":83.85,"protein_g":6.41,"fat_g":"3.34","kcal":381,"description":"Cereals rte, quaker, king vitaman"}]

c. **Where does the data get converted from the format described in part (a) to the format described in part (b)?**
Conversion occurs in server.js, line 45. The fat columns are combined and the data is formatted rigger after SQULlite's exec method is invoked.

Software Connectors are architectural elements that model interactions between components and the rules that govern those interactions. Our food lookup application uses explicit dedicated connectors to communicate between components (`fetch` and `exec` methods). Connectors provide application-independent interaction mechanisms while components provide application specific functionality.

Converting data from one format to another is also the job of a connector. In our food food lookup application, the conversion from the format returned by the exec call to the format returned to the client is not application independent.

20. **How can we modify this code to turn it into an explicit application-independent connector?**

    If we don't have the conversion of the format in the server.js file, the connector becomes independent of the application. So extracting the exec command and the data conversion part and making it a separate module would make it an independent module that fetches the data from the database and converts it as well for direct use.

21. What is the advantage of turning the data conversion into an explicit connector?

One advantage is that you are able to dictate more where the data conversion happens. By putting the parts of the data conversion in a separate module you are able to call it in other files.