**API SPECIFICATION FOR FINANCE TRUST BANK RECIPE INTERVIEW DATASET**

**AUTHOR**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **ROLE** | **API VERSION** | **DATE** |
| **Hedwig Orieba** | **Software Developer** | **1.0.0** | **2024-06-28** |

**COMMENT**

|  |
| --- |
| **AN EXPRESS REST API TO MANIPULATE A JSON DATASET OF RECIPES** |

**RECIPE SOLUTION API SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| **Name** | **orieba\_hedwig\_ftbsln** | **Comments** |
| **Type** | RESTful | Json driven exchange. |
| **Version** | 1.0.0 |  |
| **Containerization** | Docker Ver 26.1.4 build 5650f96 |  |
| **Docker Access** | docker pull orieba/ftb-app:latest |  |
| **App Initialization** | Http://localhost:3044/api/resource\_path | The app process should be initialized using port forwarding as follows:  ***docker run -p 4098:3044 orieba\_hedwig\_ftbsln***  Any other un-reserved port can be used in place of port 4098 |
| **Base URL** | Http://localhost:3044/api/resource\_path | All resource end points following the specified URL convention. |
| **HTTP Methods** | GET |  |
| **Authentication** | NONE | Limited by interview question scope. |
| **Resources** | GET: names  GET: occurances  GET: highest  GET: match | Total: 0**4** |

**RESOURCE SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| **Resource** | names |  |
| Http Method | GET |  |
| Security | None |  |
| url/address | /api/count/uniqueRecipes/names |  |
| Description | Endpoint that counts the number of unique recipe names. |  |
| Request-header | null |  |
| Sample Request | http://localhost:4098/api/count/uniqueRecipes/names | Returns the number of unique recipe names in the dataset. |
| Sample response | {"unique\_recipe\_count":"30"} | A json file called unique\_recipe.json  With this data is created at the server in the ‘appData’ directory. |
| Response header | x-auth-token | Header contains a jwt token from the server. |
| Errors: | 500 | **Possible failure in the file input/out processing.** |

|  |  |  |
| --- | --- | --- |
| **Resource** | occurances |  |
| Http Method | GET |  |
| Security | None |  |
| url/address | /api/count/uniqueRecipes/occurances |  |
| Request-header | None |  |
| Description | Endpoint that counts the occurance of each unique recipe name |  |
| Sample Request | http://localhost:4098/api/count/uniqueRecipes/occurances |  |
| Sample Response | {"count\_per\_recipe":"[{\"recipe\":\"Cajun-Spiced Pulled Pork\",\"occurance\":\"5\"},{\"recipe\":\"Cheesy Chicken Enchilada Bake\",\"occurance\":\"2\"},{\"recipe\":\"Cherry Balsamic Pork Chops\",\"occurance\":\"3\"},{\"recipe\":\"Chicken Pineapple Quesadillas\",\"occurance\":\"1\"},{\"recipe\":\"Chicken Pineapple Qwzza\",\"occurance\":\"1\"},{\"recipe\":\"Chicken Sausage Pizzas\",\"occurance\":\"5\"},{\"recipe\":\"Creamy Dill Chicken\",\"occurance\":\"4\"},{\"recipe\":\"Creamy Shrimp Tagliatelle\",\"occurance\":\"4\"},{\"recipe\":\"Garden Quesadillas\",\"occurance\":\"2\"},{\"recipe\":\"Garlic Herb Butter Steak\",\"occurance\":\"3\"},{\"recipe\":\"Grilled Cheese and Veggie Jumble\",\"occurance\":\"4\"},{\"recipe\":\"Hearty Pork Chili\",\"occurance\":\"4\"},{\"recipe\":\"Honey Sesame Chicken\",\"occurance\":\"3\"},{\"recipe\":\"Hot Honey Barbecue Chicken Legs\",\"occurance\":\"3\"},{\"recipe\":\"Korean-Style Chicken Thighs\",\"occurance\":\"4\"},{\"recipe\":\"Meatloaf à La Mom\",\"occurance\":\"3\"},{\"recipe\":\"Mediterranean Baked Veggies\",\"occurance\":\"4\"},{\"recipe\":\"Melty Monterey Jack Burgers\",\"occurance\":\"4\"},{\"recipe\":\"Mole-Spiced Beef Tacos\",\"occurance\":\"3\"},{\"recipe\":\"One-Pan Orzo Italiano\",\"occurance\":\"2\"},{\"recipe\":\"One-Pan Orzo Italianxxx\",\"occurance\":\"1\"},{\"recipe\":\"Parmesan-Crusted Pork Tenderloin\",\"occurance\":\"3\"},{\"recipe\":\"Spanish One-Pan Chicken\",\"occurance\":\"3\"},{\"recipe\":\"Speedy Steak Fajitas\",\"occurance\":\"6\"},{\"recipe\":\"Spinach Artichoke Pasta Bake\",\"occurance\":\"8\"},{\"recipe\":\"Steakhouse-Style New York Strip\",\"occurance\":\"4\"},{\"recipe\":\"Stovetop Mac 'N' Cheese\",\"occurance\":\"1\"},{\"recipe\":\"Sweet Apple Pork Tenderloin\",\"occurance\":\"4\"},{\"recipe\":\"Tex-Mex Tilapia\",\"occurance\":\"1\"},{\"recipe\":\"Yellow Squash Flatbreads\",\"occurance\":\"5\"}]"} | A stringified version of each recipe & respective occurances. |
| Response header | null |  |
| Errors: | 500 | **Possible failure in the file input/out processing.** |

|  |  |  |
| --- | --- | --- |
| **Resource** | **highest** |  |
| Http Method | GET |  |
| Security | None |  |
| url/address | /api/postcode/deliveries/highest |  |
| Description | Endpoint that returns the postcode with the most deliveries. | If there is more than one postcode that meets a high delivery score, this endpoint returns these postcodes as an array. |
| Request-header | None |  |
| Sample Request | http://localhost:4098/api/postcode/deliveries/highest |  |
| Sample Response | {"postcode":"10216","delivery\_count":"5"} |  |
| Response header | null | Header contains a jwt token from the server. |
| Errors: | **500** | **Possible failure in the file input/out processing.** |

|  |  |  |
| --- | --- | --- |
| **Resource** | **match** |  |
| Http Method | GET |  |
| Security | None |  |
| url/address | /api/recipes/patterns/match |  |
| Description | List the recipe names  (alphabetically ordered) that contain in their name one of the following words  Potato, Veggie, Mushroom |  |
| Sample Request | http://localhost:4098/api/recipes/patterns/match |  |
| Sample Response | {"match\_by\_name":["Grilled Cheese and Veggie Jumble","Mediterranean Baked Veggies"]} |  |
| Response header | null |  |
| Errors: | 500 | **Possible failure in the file input/out processing.** |

|  |
| --- |
| **CODE INSTANCE SNIPPETS**  **function getUniqueEntityAttributes(\_\_filePath, \_entityAttribute){**  **return new Promise((resolve,reject)=>{**  **fs.readFile(\_\_filePath,'utf8', (err, data) =>{**  **if(err) return reject(err);**  **let jsonObjsArr = JSON.parse(data);**  **let namesArr = [];**  **let uniqueNameSet = new Set();**  **if(\_entityAttribute == 'recipe'){**  **jsonObjsArr.forEach(elem => namesArr.push( elem.recipe ));**  **}**    **if(\_entityAttribute == 'postcode'){**  **jsonObjsArr.forEach(elem => namesArr.push( elem.postcode ));**  **}**    **namesArr.forEach(attr => uniqueNameSet.add( attr ));**  **return resolve({elementNameSet:[...uniqueNameSet], resourceData:jsonObjsArr});**  **});**  **});**  **}**  ***Description:***  This is a helper function for creating a unique set of recipe names. The function returns a promise object that is either settled with the object {elementNameSet:[…uniqueNameSet]} or an error object ‘err’.  The method takes formal parameters, a name of the dataset to process & an attribute of an object in the dataset to be investigated for uniqueness. It effects its functionality by iterating over all the objects in the dataset while pushing them into a set data structure which is then stored into an array using a spread operator.  **async function getAttributeOccurrances(\_\_filePath,\_entityAttribute){**  **const genericResponse = await getUniqueEntityAttributes(\_\_filePath, \_entityAttribute);**  **const objsArr = genericResponse.resourceData;**  **let uniqueNameObjArr = genericResponse.elementNameSet;**  **let rawMapped = [];**  **uniqueNameObjArr = uniqueNameObjArr.sort();**  **uniqueNameObjArr.forEach(nameAttr => {**  **// internal counter variable...**  **let counter = 0;**  **objsArr.forEach( recipeObj =>{**  **/\* method was called with 'recipe' actual parameter value. Count the recipes \*/**  **if(\_entityAttribute === 'recipe'){**  **if(nameAttr == recipeObj.recipe){**  **counter++;**  **}**  **}**  **/\* method was called with 'postcode' actual parameter value. Count the postcodes. \*/**  **if(\_entityAttribute == 'postcode'){**  **if(nameAttr == recipeObj.postcode){**  **counter++;**  **}**  **}**  **});**  **/\* Conditionally initialize the respective entity array \*/**  **if(\_entityAttribute === 'recipe'){**  **rawMapped.push({recipe:`${nameAttr}`, occurance:`${counter}`});**  **}**  **/\* Conditionally initialize the respective entity array \*/**  **if(\_entityAttribute == 'postcode'){**  **rawMapped.push({postcode:`${nameAttr}`, occurance:`${counter}`});**  **}**  **});**  **return {rawMappedObjVer:rawMapped};**  **}**  ***Description:***  This is a helper function that returns an object definition of a provided object attribute[recipe or postcode] & it's number of occurances.  It gets all the unique names,  **let uniqueNameObjArr = genericResponse.elementNameSet;**  It sorts them & uses each of them to iterate through the entire original dataset while counting how many times recipe occurs thereby formulating the engine of the function. This is excerpted below:  **uniqueNameObjArr.forEach(nameAttr => {**  **// internal counter variable...**  **let counter = 0;**  **objsArr.forEach( recipeObj =>{**  **/\* method was called with 'recipe' actual parameter value. Count the recipes \*/**  **if(\_entityAttribute === 'recipe'){**  **if(nameAttr == recipeObj.recipe){**  **counter++;**  **}**  **}** |