**Basic info: ALWAYS RETURNS 200 ON SUCCESS and specific object where needed, 400 on failure and no object.**

SERVER: <https://pacific-stream-14038.herokuapp.com>

|  |  |  |
| --- | --- | --- |
| **RESOURCE: USER** | {nickname: <value>,  email: <value>,  password: <value>,  profile\_pic\_path: <value>} |  |
| **Get all users** | GET /user | REQBODY: none  Returns array of users |
| **Create a user** | POST /user | REQBODY: user object  Returns created user object |
| **Delete a user** | Delete /user/:nickname/:password | REQBODY: none  Returns only status code |
| **Update a user** | PUT /user/:nickname/:password | REQBODY: user object with fields to update set to updated values  Returns updated user.  **CAREFUL: if all the values to update are equal to those already in the DB, a 400 bad request will be returned.** |
| **Authenticate a user** | GET /user/:nickname/:password | REQBODY: none  Returns user object on successful authentication |

|  |  |  |
| --- | --- | --- |
| **RESOURCE: PERFECT NIGHT** | {id: <value> (autoincremented, you never pass this value but it gets passed back to you in response),  tag\_type: <value>,  book: <value>,  board\_game: <value>,  meal: <value>,  artist: <value>,  description: <value>,  movie\_tvSeries: <value>,  cocktail: <value>,  beer: <value>  } |  |
| **Get all nights** | GET /perfectnight | REQBODY: none  Returns array of nights |
| **Get specific night** | GET /perfectnight/:id | REQBODY: none  Returns perfectnight object on success |
| **Create a night** | POST /perfectnight/:nickname | REQBODY: perfectnight object  Returns nothing |
| **Delete a night** | DELETE /perfectnight/:id | REQBODY: none  Returns nothing |
| **Update a night** | PUT /perfectnight/:id | REQBODY: perfectnight object with fields to update set  Returns updated perfectnight object.  **CAREFUL: if all the values to update are equal to those already in the DB, a 400 bad request will be returned.** |
| **Fill feed randomly** | GET /perfectnight/feed/fillRandom | REQBODY: none  Returns array of **COMPLETE NIGHTS** in random order |
| **Get COMPLETE NIGHT VERSIONS of created, saved or upvoted nights** | GET /perfectnight/myProfile/:nickname/:mode | REQBODY: none  Returns array of **COMPLETE NIGHTS** that the user created, saved or upvoted.  **CAREFUL: mode must be “created”, “saved” or “upvoted”.** |

**Basic info: ALWAYS RETURNS 200 ON SUCCESS and specific object where needed, 400 on failure and no object.**

Where referenced as **COMPLETE NIGHTS** it means that the API will return a perfect night object where fields such as meal or cocktail are the actual meal or cocktail object, not just the id.

|  |  |  |
| --- | --- | --- |
| **RESOURCE: CREATED** | {nickname: <value>,  perfectnight\_id: <value>} |  |
| **Get all nights created by a user** | GET /created/:nickname | REQBODY: none  Returns an array of perfectnight objects |

|  |  |  |
| --- | --- | --- |
| **RESOURCE: UPVOTED** | {nickname: <value>,  perfectnight\_id: <value>} |  |
| **Insert upvote** | POST /upvoted/:nickname/:id | REQBODY: none  Returns nothing |
| **Delete upvote** | DELETE /upvoted/:nickname/:id | REQBODY: none  Returns nothing |
| **Get all upvoted nights by a user** | GET /upvoted/:nickname | REQBODY: none  Returns an array of perfectnight objects |

|  |  |  |
| --- | --- | --- |
| **RESOURCE : SAVED** | {nickname: <value>,  perfectnight\_id: <value>} |  |
| **Get all saved nights by a user** | GET /saved/:nickname | REQBODY: none  Returns an array of perfectnight objects |
| **Insert save** | POST /saved/:nickname/:id | REQBODY: none  Returns nothing |
| **Delete save** | DELETE /saved/:nickname/:id | REQBODY: none  Returns nothing |

|  |  |  |
| --- | --- | --- |
| **RESOURCE: MEAL (spoonacular version)** | {  Id: <meal id>  Image: <meal Image url>,  Title: <name of the dish>,  instructions: <instructions >,  ingredients: <meal ingredients>,  quantities  } |  |
| **GET meal list by name** | GET /meal/byName/:name/:howMany | REQBODY: object with additional parameters to filter the query. Parameters set to null won’t be considered in the query. Check below this table for a list of valid parameters.  Returns array of meals |
| **GET random meal list** | GET /meal/byRandom/:howMany | REQBODY: none  Returns array of meals totally random, no filters applied |
| **GET more specific instructions** | GET /meal/moreInfo/:id | REQBODY: none  Returns {instructions: <string>, ingredients: array} to make the dish identified by ID. |
| **GET general meal overview** | GET /meal/overview/:id | REQBODY: none  Returns {overview: string, instructions: string, ingredients: array}, a generic overview of the dish identified by ID. |

Parameters admitted to filter the query:

* “cuisine”: {African, American, British, Cajun, Caribbean, Chinese, Eastern European, European, French, German, Greek, Indian, Irish, Italian, Japanese, Jewish, Korean, Latin American, Mediterranean, Mexican, Middle Eastern, Nordic, Southern, Spanish, Thai, Vietnamese};
* “diet”: {Gluten Free, Ketogenic, Vegetarian, Lacto-Vegetarian, Ovo-Vegetarian, Vegan, Pescetarian, Paleo, Primal, Whole30};
* “excludeIngredients”: this has to be a string of comma separated ingredients;
* “intolerances”: this has to be a string of comma separated intolerances;

|  |  |  |
| --- | --- | --- |
| **RESOURCE: COCKTAIL** | {name: <name of the drink>,  id:<cocktail id>,  category: <category>,  instructions: <recipe>,  image: <image>,  ingredients: <array of ingredients>,  quantities: <array of quantities>} |  |
| **Get cocktail by name** | GET /cocktail/:name | REQBODY: none  Returns cocktail object |
| **Get multiple cocktails by name** | GET /cocktail/:name/multiple/:howMany | REQBODY: none  Returns cocktails array |
| **Get a random cocktail** | GET /cocktail/surprise/single | REQBODY: none  Returns cocktail object |
| **Get a list of random cocktails** | GET /cocktail/surprise/:howMany | REQBODY: none  Returns array of random cocktails |
| **Get a list of cocktails by ingredient** | GET /cocktail/ingredient/:howMany/:ingredient | REQBODY: none  Returns array of cocktails with common ingredient |
| **Get cocktails by category** | GET /cocktail/byCategory/:category/:howMany | REQBODY: none  Returns array of cocktails with common category  AVAILABLE CATEGORIES:  “Ordinary Drink”  “Cocktail”  “Shot”  “Homemade Liqueur”  “Beer” |
| **Get cocktail by id** | GET /cocktail/byId/:id | REQBODY: none  Returns cocktail object |

|  |  |  |
| --- | --- | --- |
| **RESOURCE: BOOK** | {volume\_id: <the one we save in the db>,  title: <book title>,  authors:< array of authors’ names >,  link:<link to the book on google books>,  image: <url to image of the book>,  categoryList: <array of categories>} |  |
| **Get a list of books by name** | GET /book/multiple/:name/:howMany | REQBODY: none  Returns array of book objects |
| **Get book by name** | GET/book/single/:name | REQBODY: none  Returns single book object, first match of the query |
| **Get book list by category** | GET /book/category/:category\_name/:howMany | REQBODY: none  Returns array of books sharing same category |
| **Get book by ID** | GET /book/byID/:id | REQBODY: none  Returns book object |
| **Get list of books by author** | GET /book/byAuthor/:author | REQBODY: none  Returns array of books sharing same author |

|  |  |  |
| --- | --- | --- |
| **RESOURCE: BOARDGAME** | {  id: id,  name:name,  minPlaytime:minPlaytime,  maxPlaytime:maxPlaytime,  minPlayers:minPlayers,  maxPlayers:maxPlayers,  minAge:minAge,  description:description,  imageUrl:imageUrl,  price:price,  averageRating:averageRating} |  |
| **Get a list of games by name** | GET /boardgame/byName/:name/:howMany | REQBODY:NONE  Returns array of boardgames objects |
| **Get a list of random board games** | GET /boardgame/byRandom/:howMany | REQBODY: NONE  Returns array of boardgames objects |
| **Get a list of games matching the filters given** | GET /boardgame/byFilters /:howMany | REQBODY: filterObj, as shown below  Returns array of boardgames objects |
| **Get a game by ID** | GET /boardgame/byId/:id | REQBODY: none  Returns boardgame object |

FilterObj = { playTime: playTime, playersNum:playersNum}

|  |  |  |
| --- | --- | --- |
| **RESOURCE: BEER** | {  name:name,  category:category,  id:id,  abv:abv,  ibu:ibu,  description:description,  image:image,  categoryDescription:description  } | **CAREFUL: there is a slight chance there’s no image provided by BreweryDB. In that case the image field will be set to “Sorry, no picture provided for this beer”. There is also a slight chance no ibu is provided, in which case the field won’t be set at all.** |
| **Get a list of random beers** | GET /beer/byRandom/:howMany | REQBODY:none  Returns array of beer objects  **CAREFUL: howMany has a maximum value of 10.** |
| **Get a list of beers by name** | GET /beer/byName/:name/:howMany | REQBODY: none  Returns array of beer objects |
| **Get a beer by ID** | GET /beer/byId/:id | REQBODY:none  Returns beer object |
| **Get a beer list with additional filters** | GET /beer/byFilters/:ibu/:abv/:howMany | REQBODY: none  Returns array of beers matching ibu & abv. To only filter by either one of those, set the other one to 0. |

|  |  |  |
| --- | --- | --- |
| **RESOURCE:MEAL (the meal db version)** | {  id:id,  name:name,  category:category,  instructions:instructions,  image:image,  cuisine:cuisine,  ingredients:array,  quantities:array  } |  |
| **Get a meal by name** | GET /meal/:name | REQBODY: none  Returns meal object |
| **Get a list of meals by name** | GET /meal/:name/multiple/:howMany | REQBODY: none  Returns array of meal objects |
| **Get a random meal** | GET /meal/surprise/single | REQBODY: none  Returns random meal object |
| **Get a list of random meals** | GET /meal/surprise/:howMany | REQBODY:none  Returns list of random meal objects |
| **Get list of meals by cuisine** | GET /meal/cuisine/:cuisine/:howMany | REQBODY: none  Returns list of meal objects |
| **Get a list of meals by category** | GET /meal/category/:category/:howMany | REQBODY: none  Returns list of meal objects |

|  |  |  |
| --- | --- | --- |
| **RESOURCE: MUSIC** | Check down below to see a description of a track object and an album object | **CAREFUL: Get 20 songs means you can get as much as 20, not exactly 20. Same reasoning applies everywhere else.** |
| **Get 20 songs by title** | GET /music/byTitle/:title | REQBODY: none  Returns array of track objects |
| **Get 30 songs by author’s name** | GET /music/byArtist/:artist | REQBODY: none  Returns array of track objects |
| **Get 15 albums by album’s title** | GET /music/album/byName/:name | REQBODY: none  Returns array of album objects |
| **Get 15 albums by album’s author** | GET /music/album/byArtist/:artist | REQBODY: none  Returns array of album objects |
| **THE FOLLOWING API ENDPOINTS REQUIRE OAUTH USER AUTHENTICATION** |  |  |
| **Login with spotify** | GET /music/auth | REQBODY: none  Redirects user first to Spotify’s login page, then to musicAdv. Browser should save the code returned in the redirection and redirect it to the second phase of the authentication. |
| **Complete authentication with spotify** | GET /music/callback?code={code} | REQBODY:none  Returns access\_token that must be provided for all future requests. |
| **Get user’s first 50 saved tracks** | GET /music/savedTracks/:access\_token | REQBODY: none  Returns first 50 saved songs of the current user |
| **Save a track for the current user** | POST /music/savedTracks/:access\_token/:song\_id | REQBODY: none  Returns 200 on success |

Track object = {

artists:array,

album\_image:image,

name:name,

release\_date:release\_date,

popularity:popularity,

id:id

preview\_url:preview\_url

}

Album object = {

artists:array,

album\_image:image,

name:name,

id:id,

release\_date:release\_date,

tracks:array,

label:label,

popularity:popularity

}