API HUB
Developing the API

45 min

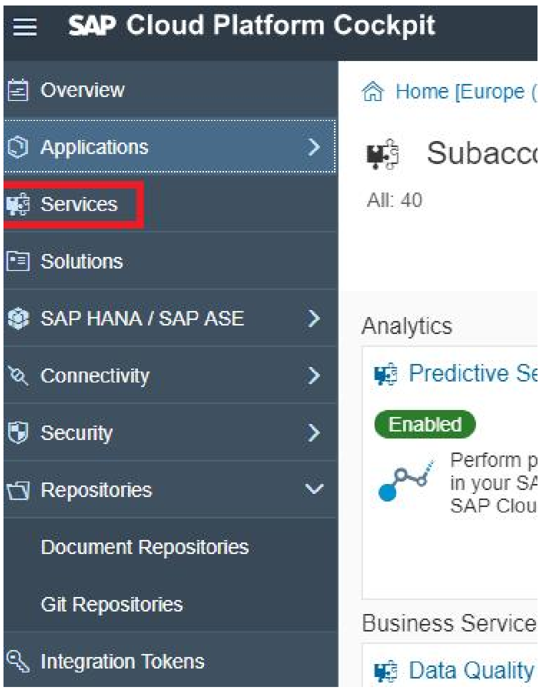
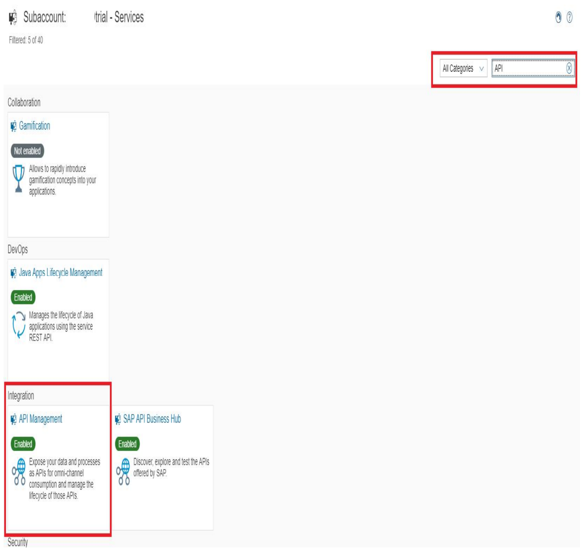
Description
In this exercise, you'll learn how to use OpenAPI and the SAP API Designer to develop and validate the APIs for the UI.
The microservice you'll create effectively combines user data from SuccessFactors with their corresponding expense reports from Concur.

Prerequisites
- Trial account on the SAP Cloud Platform (https://account.hanatrial.ondemand.com)
- Node JS version 6.12.3 or later installed (http://www.nodejs.org)
- Node Package Manager (NPM) 3.10 or later installed (should be automatically installed when you install Node JS)
- Postman REST Client version 6.0.9 or later (http://www.getpostman.com)
- Visual Studio Code (https://code.visualstudio.com/) or another code editor
- Tutorial 1

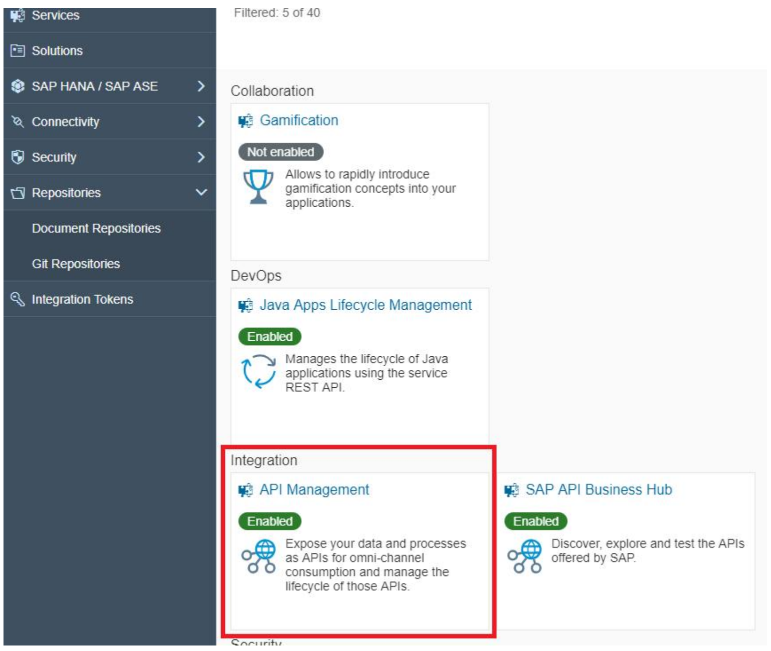
Target group
- Application developers
- People interested in the SAP Cloud Platform

Target group requirements
- Basic programming skills, ideally JavaScript

Scenario 2a: Onboarding – Building an API

Explanation	**Screenshot**
1. From your SAP Cloud Platform Neo trial account, go to **Services.** For details please refer to scenario 1a in Tutorial 1.	 SAP Cloud Platform Cockpit Overview Applications Services Solutions SAP HANA / SAP ASE Connectivity Security Repositories Document Repositories Git Repositories Integration Tokens Home [Europe (Subacco All: 40 Analytics Predictive Se Enabled Perform p in your SA SAP Clou Business Service Data Quality
2. In the Search field, type **API**	Subaccount: trial - Services Filtered: 5 of 40 All Categories API Collaboration Gamification Not enabled Allows to rapidly introduce gamification concepts into your applications. DevOps Java Apps Lifecycle Management Enabled Manages the lifecycle of Java applications using the service REST API. Integration API Management Enabled Expose your data and processes as APIs for omni-channel consumption and manage the lifecycle of those APIs. SAP API Business Hub Enabled Discover, explore and test the APIs offered by SAP. Security

3. You can see the service **API Management**
Note: If the service is not enabled, you need to enable the service to be able to use it.



Filtered: 5 of 40

Collaboration

Gamification
Not enabled
Allows to rapidly introduce gamification concepts into your applications.

DevOps

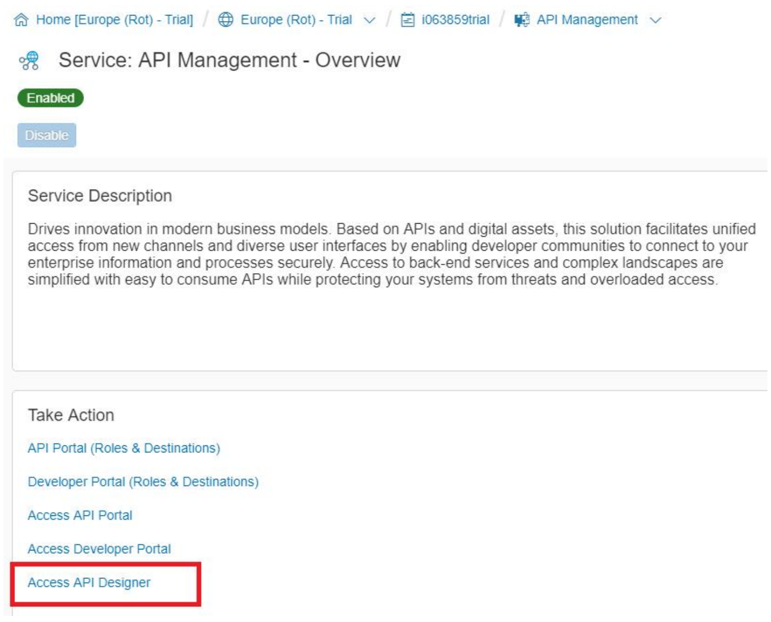
Java Apps Lifecycle Management
Enabled
Manages the lifecycle of Java applications using the service REST API.

Integration

API Management
Enabled
Expose your data and processes as APIs for omni-channel consumption and manage the lifecycle of those APIs.

SAP API Business Hub
Enabled
Discover, explore and test the APIs offered by SAP.

4. In the **Take Action** panel, click on Access **API Designer**


Home [Europe (Rot) - Trial] / Europe (Rot) - Trial / i063859trial / API Management

Service: API Management - Overview

Enabled

Disable

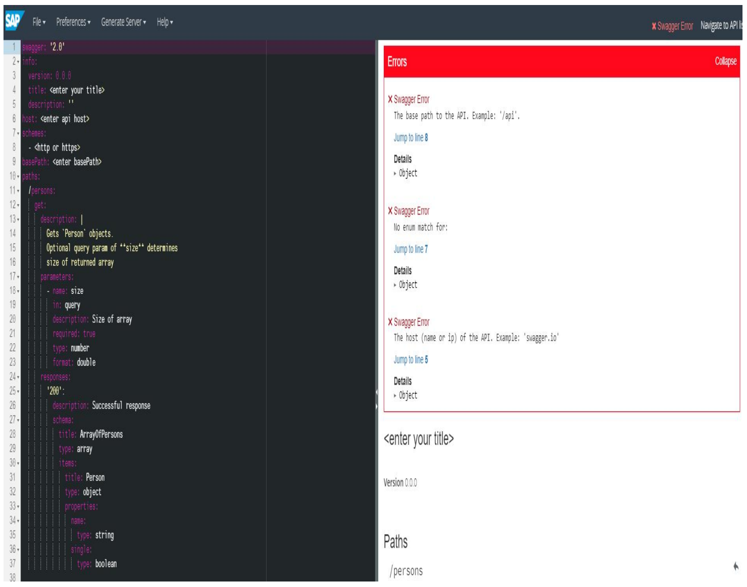
Service Description

Drives innovation in modern business models. Based on APIs and digital assets, this solution facilitates unified access from new channels and diverse user interfaces by enabling developer communities to connect to your enterprise information and processes securely. Access to back-end services and complex landscapes are simplified with easy to consume APIs while protecting your systems from threats and overloaded access.

Take Action

API Portal (Roles & Destinations)
Developer Portal (Roles & Destinations)
Access API Portal
Access Developer Portal
Access API Designer

5. You are now in the **API Designer Tool**, and should see a screen like this.



```yaml
swagger: '2.0'
info:
  version: 0.0.0
  title: <enter your title>
  description: ''
host: <enter api host>
schemes:
  - <http or https>
basePath: <enter basePath>
paths:
  /persons:
    get:
      description: |
        Gets `Person` objects.
        Optional query param of **size** determines
        size of returned array
      parameters:
        - name: size
          in: query
          description: Size of array
          required: true
          type: number
          format: double
      responses:
        '200':
          description: Successful response
          schema:
            title: ArrayOfPersons
            type: array
            items:
              title: Person
              type: object
              properties:
                name:
                  type: string
                single:
                  type: boolean
```

Errors — Collapse

X Swagger Error
The base path to the API. Example: '/api'.
Jump to line 8
Details
▸ Object

X Swagger Error
No enum match for:
Jump to line 7
Details
▸ Object

X Swagger Error
The host (name or ip) of the API. Example: 'swagger.io'
Jump to line 5
Details
▸ Object

<enter your title>

Version 0.0.0

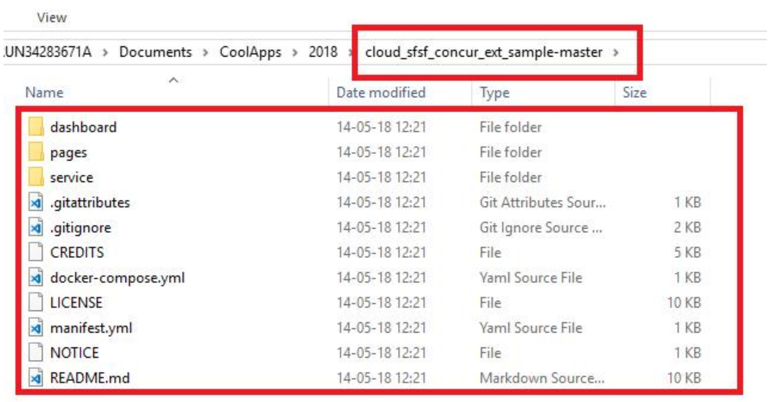
Paths

/persons

6. Download the project from the Github repository

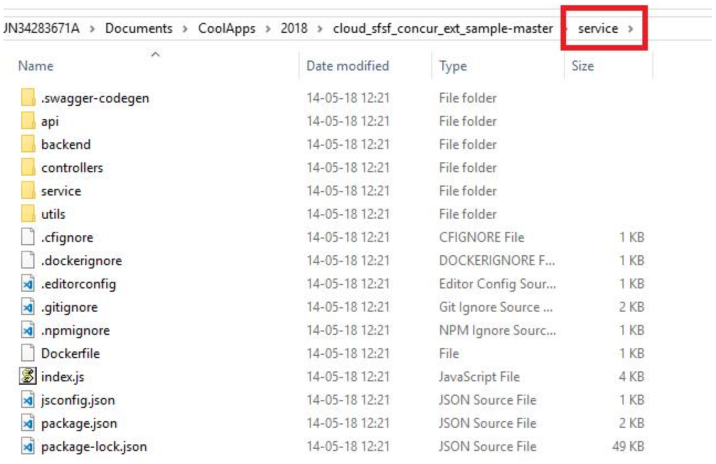
   https://github.com/SAP/cloud_sfsf_concur_ext_sample

   When project is downloaded you should see a structure like this.


View
UN34283671A › Documents › CoolApps › 2018 › cloud_sfsf_concur_ext_sample-master ›

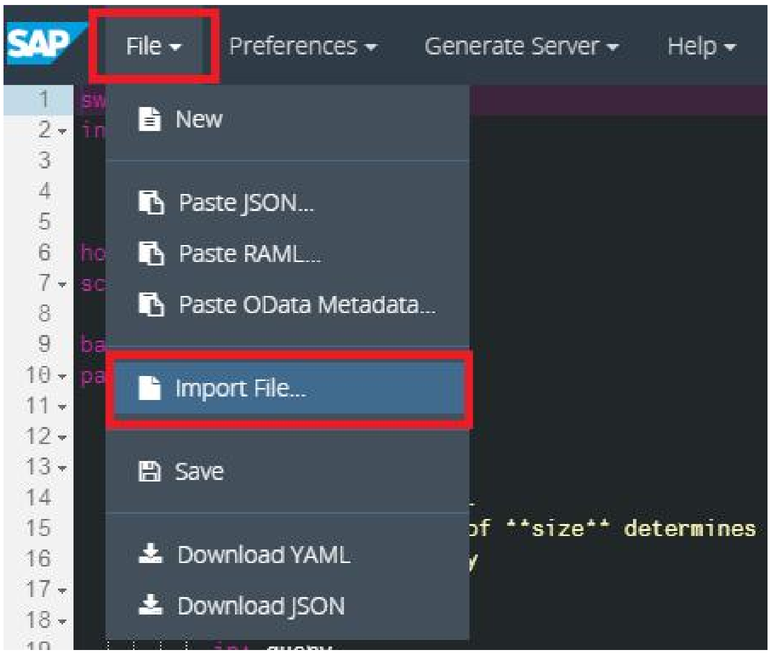
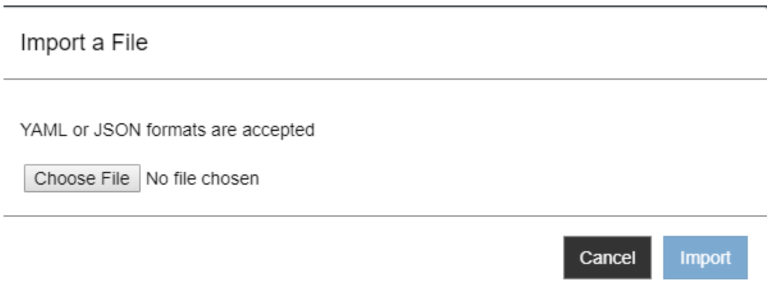
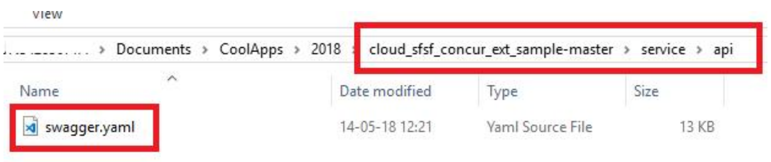
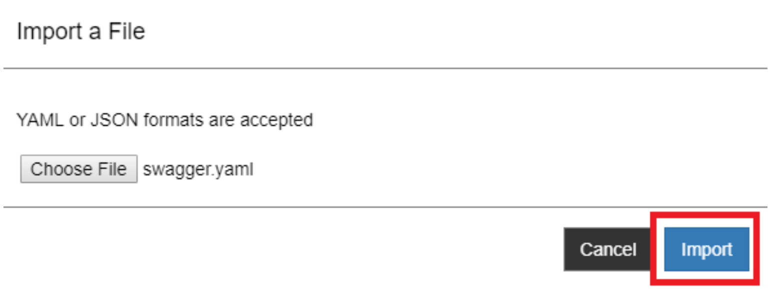
Name	Date modified	Type	Size
dashboard	14-05-18 12:21	File folder	
pages	14-05-18 12:21	File folder	
service	14-05-18 12:21	File folder	
.gitattributes	14-05-18 12:21	Git Attributes Sour...	1 KB
.gitignore	14-05-18 12:21	Git Ignore Source ...	2 KB
CREDITS	14-05-18 12:21	File	5 KB
docker-compose.yml	14-05-18 12:21	Yaml Source File	1 KB
LICENSE	14-05-18 12:21	File	10 KB
manifest.yml	14-05-18 12:21	Yaml Source File	1 KB
NOTICE	14-05-18 12:21	File	1 KB
README.md	14-05-18 12:21	Markdown Source...	10 KB

7. Go to the root folder of your project. i.e: *service* folder

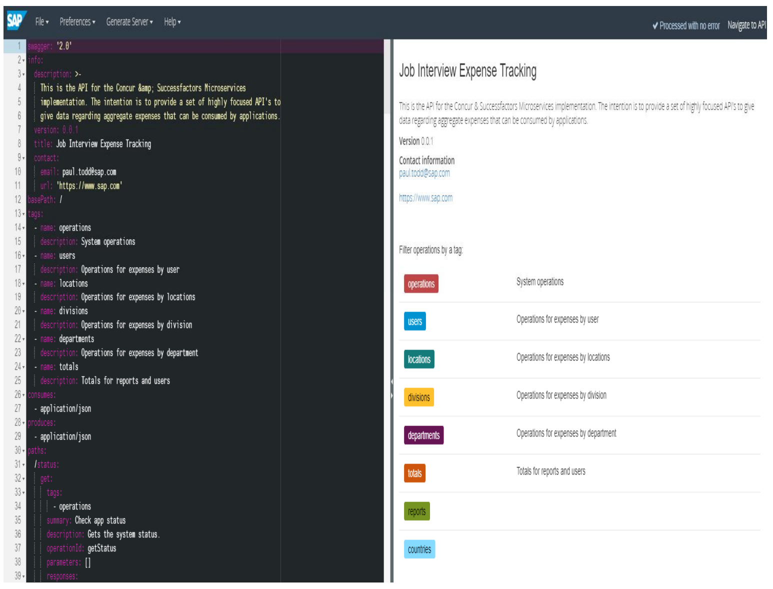

JN34283671A › Documents › CoolApps › 2018 › cloud_sfsf_concur_ext_sample-master › service ›

Name	Date modified	Type	Size
.swagger-codegen	14-05-18 12:21	File folder	
api	14-05-18 12:21	File folder	
backend	14-05-18 12:21	File folder	
controllers	14-05-18 12:21	File folder	
service	14-05-18 12:21	File folder	
utils	14-05-18 12:21	File folder	
.cfignore	14-05-18 12:21	CFIGNORE File	1 KB
.dockerignore	14-05-18 12:21	DOCKERIGNORE F...	1 KB
.editorconfig	14-05-18 12:21	Editor Config Sour...	1 KB
.gitignore	14-05-18 12:21	Git Ignore Source ...	2 KB
.npmignore	14-05-18 12:21	NPM Ignore Sourc...	1 KB
Dockerfile	14-05-18 12:21	File	1 KB
index.js	14-05-18 12:21	JavaScript File	4 KB
jsconfig.json	14-05-18 12:21	JSON Source File	1 KB
package.json	14-05-18 12:21	JSON Source File	2 KB
package-lock.json	14-05-18 12:21	JSON Source File	49 KB

8. Now you can use **API Designer** to edit the API

9. From the **File** Menu, select **Import File…**	 SAP File Preferences Generate Server Help New Paste JSON... Paste RAML... Paste OData Metadata... Import File... Save Download YAML Download JSON
10. Choose the file you want to import. *Note: Only YAML and JSON formats are accepted*	 Import a File YAML or JSON formats are accepted Choose File No file chosen
11. Go to the *api* folder and select the *swagger.yaml* file.	 Documents › CoolApps › 2018 › cloud_sfsf_concur_ext_sample-master › service › api Name · Date modified · Type · Size swagger.yaml · 14-05-18 12:21 · Yaml Source File · 13 KB
12. Click on **Import**	 Import a File YAML or JSON formats are accepted Choose File swagger.yaml

13. You have now imported the API specification, now we will explore into the API itself.



```yaml
swagger: '2.0'
info:
  description: >-
    This is the API for the Concur &amp; Successfactors Microservices
    implementation. The intention is to provide a set of highly focused API's to
    give data regarding aggregate expenses that can be consumed by applications.
  version: 0.0.1
  title: Job Interview Expense Tracking
  contact:
    email: paul.todd@sap.com
    url: 'https://www.sap.com'
basePath: /
tags:
  - name: operations
    description: System operations
  - name: users
    description: Operations for expenses by user
  - name: locations
    description: Operations for expenses by locations
  - name: divisions
    description: Operations for expenses by division
  - name: departments
    description: Operations for expenses by department
  - name: totals
    description: Totals for reports and users
consumes:
  - application/json
produces:
  - application/json
paths:
  /status:
    get:
      tags:
        - operations
      summary: Check app status
      description: Gets the system status.
      operationId: getStatus
      parameters: []
      responses:
```

Job Interview Expense Tracking

This is the API for the Concur & Successfactors Microservices implementation. The intention is to provide a set of highly focused API's to give data regarding aggregate expenses that can be consumed by applications.

Version 0.0.1

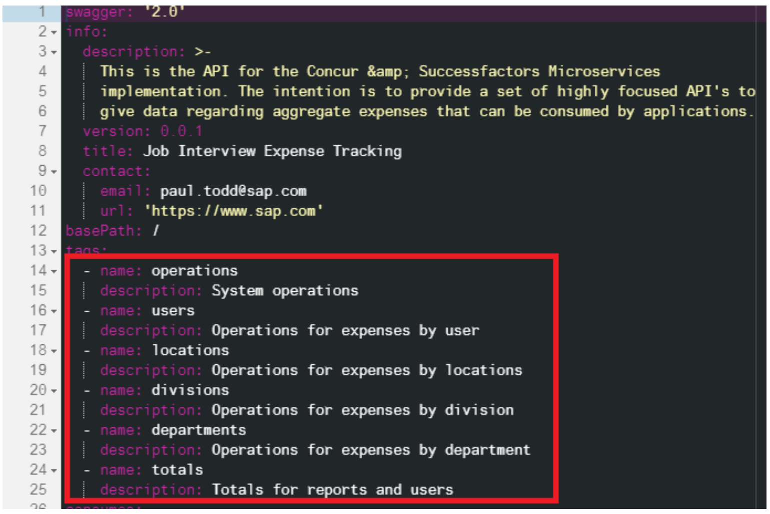
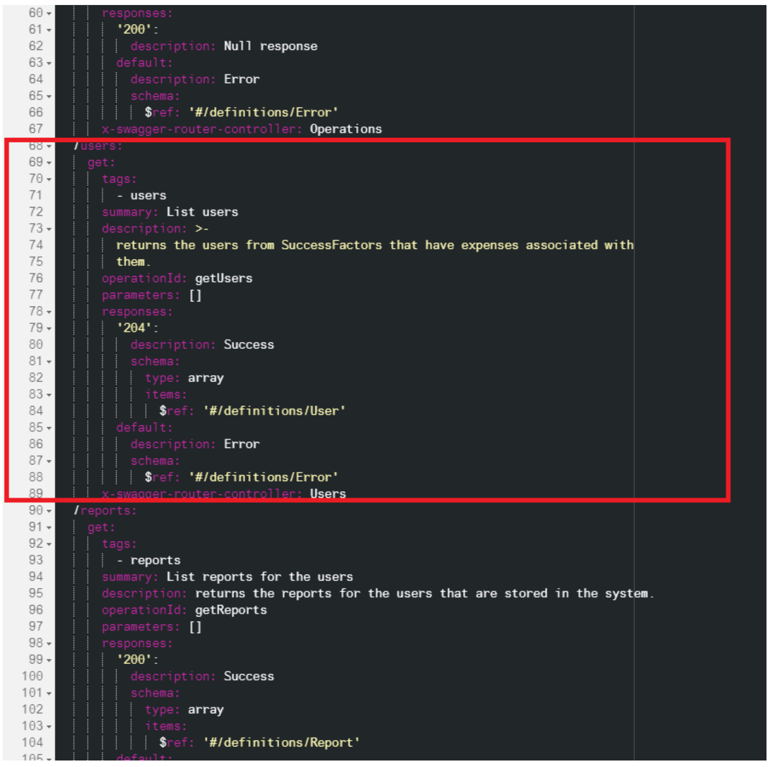
Contact information
paul.todd@sap.com

https://www.sap.com

Filter operations by a tag:

operations	System operations
users	Operations for expenses by user
locations	Operations for expenses by locations
divisions	Operations for expenses by division
departments	Operations for expenses by department
totals	Totals for reports and users
reports	
countries	

Scenario 2b: Building an API

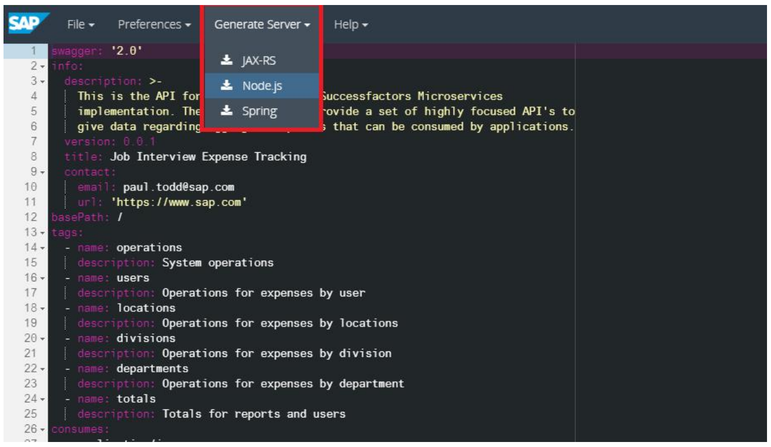
Explanation	Screenshot
1. From the **API Designer**, have a look on the descriptions.	
2. Have a look now on the **paths.** Each **Operation** is described	

```yaml
swagger: '2.0'
info:
  description: >-
    This is the API for the Concur &amp; Successfactors Microservices
    implementation. The intention is to provide a set of highly focused API's to
    give data regarding aggregate expenses that can be consumed by applications.
  version: 0.0.1
  title: Job Interview Expense Tracking
  contact:
    email: paul.todd@sap.com
    url: 'https://www.sap.com'
basePath: /
tags:
  - name: operations
    description: System operations
  - name: users
    description: Operations for expenses by user
  - name: locations
    description: Operations for expenses by locations
  - name: divisions
    description: Operations for expenses by division
  - name: departments
    description: Operations for expenses by department
  - name: totals
    description: Totals for reports and users
```

```yaml
      responses:
        '200':
          description: Null response
        default:
          description: Error
          schema:
            $ref: '#/definitions/Error'
      x-swagger-router-controller: Operations
  /users:
    get:
      tags:
        - users
      summary: List users
      description: >-
        returns the users from SuccessFactors that have expenses associated with
        them.
      operationId: getUsers
      parameters: []
      responses:
        '204':
          description: Success
          schema:
            type: array
            items:
              $ref: '#/definitions/User'
        default:
          description: Error
          schema:
            $ref: '#/definitions/Error'
      x-swagger-router-controller: Users
  /reports:
    get:
      tags:
        - reports
      summary: List reports for the users
      description: returns the reports for the users that are stored in the system.
      operationId: getReports
      parameters: []
      responses:
        '200':
          description: Success
          schema:
            type: array
            items:
              $ref: '#/definitions/Report'
```

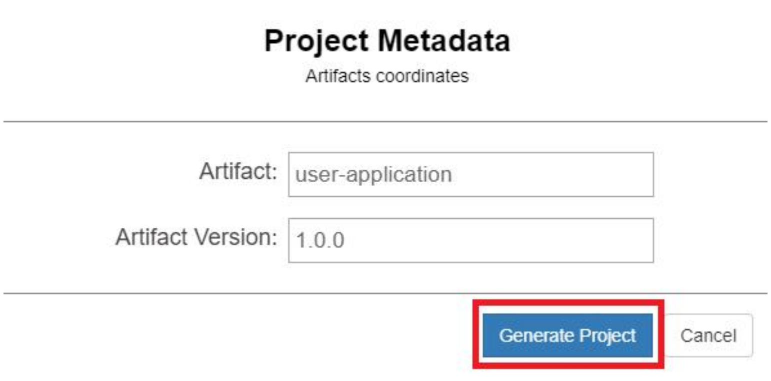
3. You can generate code from your API to node.js
   Go to Generate Server – Node.js

   Note: Our complete code for the application is added on the top of this code generation.



```
swagger: '2.0'
info:
  description: >-
    This is the API for                    Successfactors Microservices
    implementation. The                    rovide a set of highly focused API's to
    give data regarding                    that can be consumed by applications.
  version: 0.0.1
  title: Job Interview Expense Tracking
  contact:
    email: paul.todd@sap.com
    url: 'https://www.sap.com'
basePath: /
tags:
  - name: operations
    description: System operations
  - name: users
    description: Operations for expenses by user
  - name: locations
    description: Operations for expenses by locations
  - name: divisions
    description: Operations for expenses by division
  - name: departments
    description: Operations for expenses by department
  - name: totals
    description: Totals for reports and users
consumes:
```

4. You can now generate your project.
   You can specify the Artifact and Artifact version.



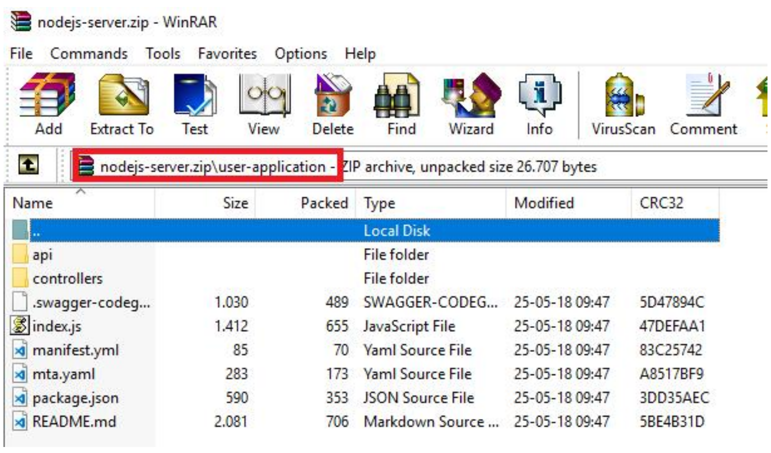
Project Metadata

Artifacts coordinates

Artifact: user-application

Artifact Version: 1.0.0

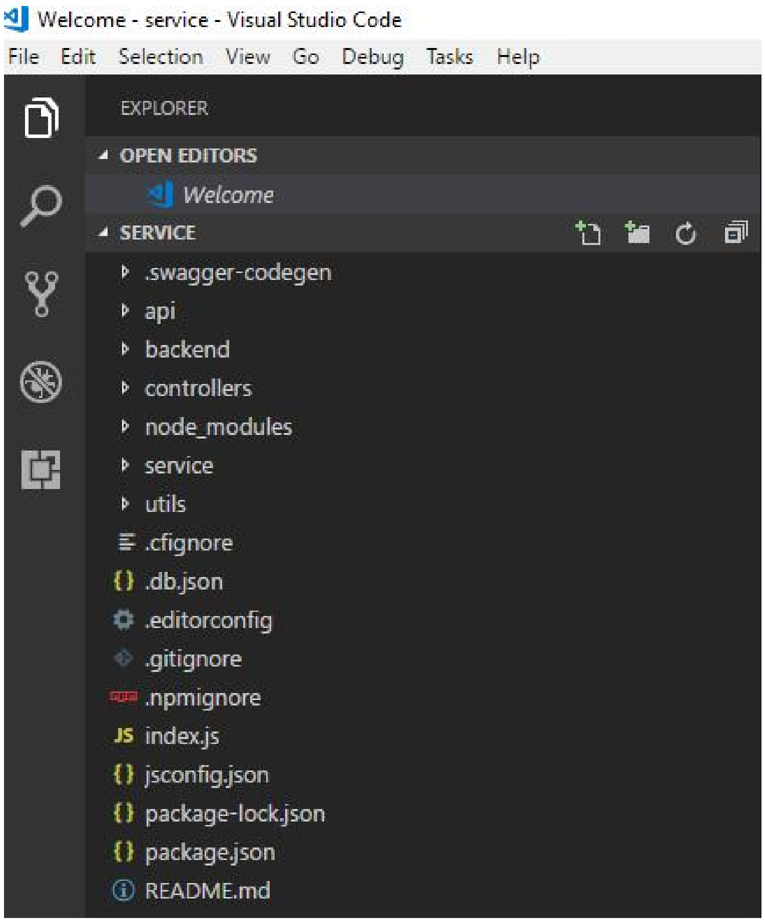
5. That will download a zip file containing your project.



nodejs-server.zip\user-application - ZIP archive, unpacked size 26.707 bytes

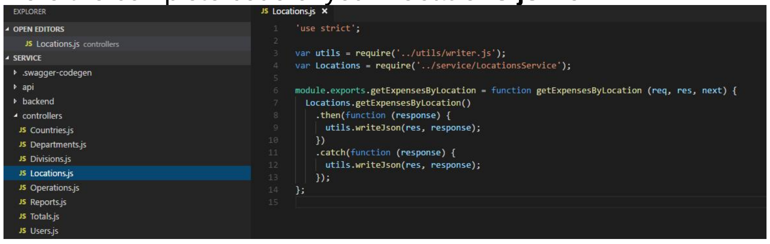
Name	Size	Packed	Type	Modified	CRC32
..			Local Disk		
api			File folder		
controllers			File folder		
.swagger-codeg...	1.030	489	SWAGGER-CODEG...	25-05-18 09:47	5D47894C
index.js	1.412	655	JavaScript File	25-05-18 09:47	47DEFAA1
manifest.yml	85	70	Yaml Source File	25-05-18 09:47	83C25742
mta.yaml	283	173	Yaml Source File	25-05-18 09:47	A8517BF9
package.json	590	353	JSON Source File	25-05-18 09:47	3DD35AEC
README.md	2.081	706	Markdown Source ...	25-05-18 09:47	5BE4B31D

6. Open your project with Visual Studio Code. (or another code editor)
   File -> Open Folder (and select the root folder of your service)



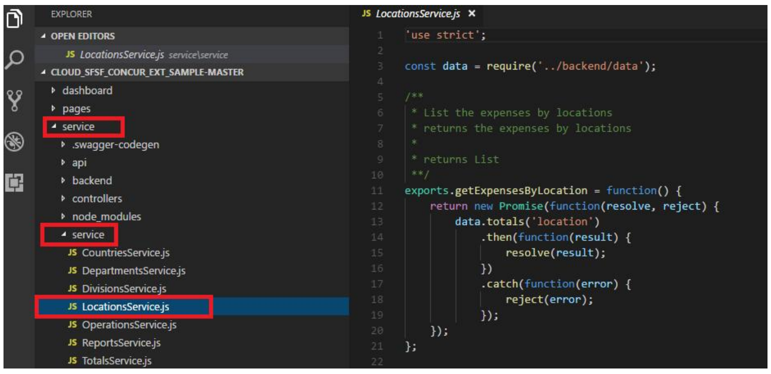
7. Have a look to the code in Locations.js file.
   As you can see, this file requires **LocationsService** in the **service** folder.

Here the complete code of your **Locations.js** file



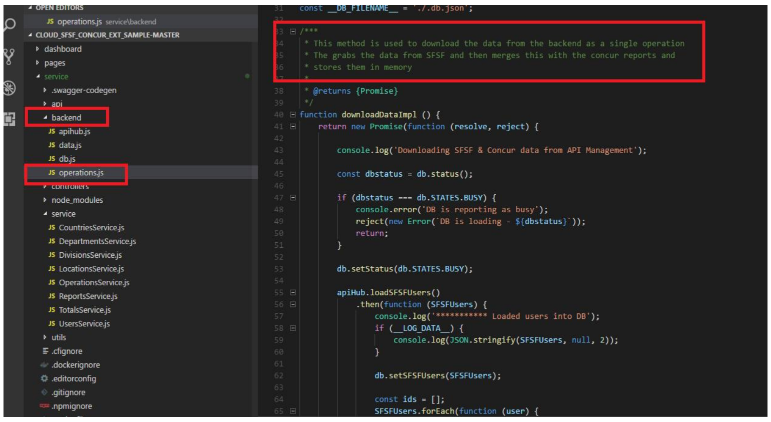
```
'use strict';

var utils = require('../utils/writer.js');
var Locations = require('../service/LocationsService');

module.exports.getExpensesByLocation = function getExpensesByLocation (req, res, next) {
  Locations.getExpensesByLocation()
    .then(function (response) {
      utils.writeJson(res, response);
    })
    .catch(function (response) {
      utils.writeJson(res, response);
    });
};
```

8. Navigate to the service folder and have a look to the file **LocationsService.js**



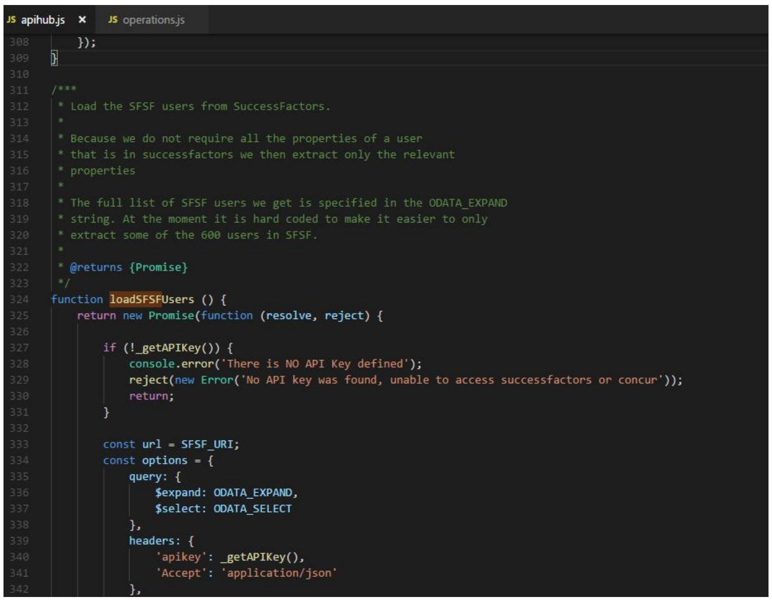
```
'use strict';

const data = require('../backend/data');

/**
 * List the expenses by locations
 * returns the expenses by locations
 *
 * returns List
 **/
exports.getExpensesByLocation = function() {
    return new Promise(function(resolve, reject) {
        data.totals('location')
            .then(function(result) {
                resolve(result);
            })
            .catch(function(error) {
                reject(error);
            });
    });
};
```

9. Open the **Operations.js** file in the **backend** folder.
   You will see the function to connect to SFSF.
   Have look on this code to understand what it's doing.



```
const __DB_FILENAME__ = './.db.json';

/***
 * This method is used to download the data from the backend as a single operation
 * The grabs the data from SFSF and then merges this with the concur reports and
 * stores them in memory
 *
 * @returns {Promise}
 */
function downloadDataImpl () {
    return new Promise(function (resolve, reject) {

        console.log('Downloading SFSF & Concur data from API Management');

        const dbstatus = db.status();

        if (dbstatus === db.STATES.BUSY) {
            console.error('DB is reporting as busy');
            reject(new Error(`DB is loading - ${dbstatus}`));
            return;
        }

        db.setStatus(db.STATES.BUSY);

        apiHub.loadSFSFUsers()
            .then(function (SFSFUsers) {
                console.log('*********** Loaded users into DB');
                if (__LOG_DATA__) {
                    console.log(JSON.stringify(SFSFUsers, null, 2));
                }

                db.setSFSFUsers(SFSFUsers);

                const ids = [];
                SFSFUsers.forEach(function (user) {
```

10. Open the **apihub.js** file and check the **loadSFSFUsers** function called in **Operations.js.**
11. Take time to check the complete code, that will show you how the application connects to the API HUB and merges the data together.

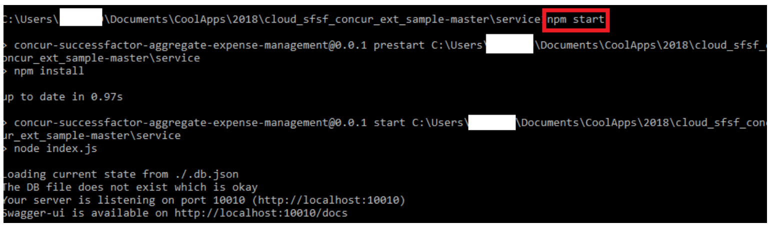


```
    });
}

/***
 * Load the SFSF users from SuccessFactors.
 *
 * Because we do not require all the properties of a user
 * that is in successfactors we then extract only the relevant
 * properties
 *
 * The full list of SFSF users we get is specified in the ODATA_EXPAND
 * string. At the moment it is hard coded to make it easier to only
 * extract some of the 600 users in SFSF.
 *
 * @returns {Promise}
 */
function loadSFSFUsers () {
    return new Promise(function (resolve, reject) {

        if (!_getAPIKey()) {
            console.error('There is NO API Key defined');
            reject(new Error('No API key was found, unable to access successfactors or concur'));
            return;
        }

        const url = SFSF_URI;
        const options = {
            query: {
                $expand: ODATA_EXPAND,
                $select: ODATA_SELECT
            },
            headers: {
                'apikey': _getAPIKey(),
                'Accept': 'application/json'
            },
```

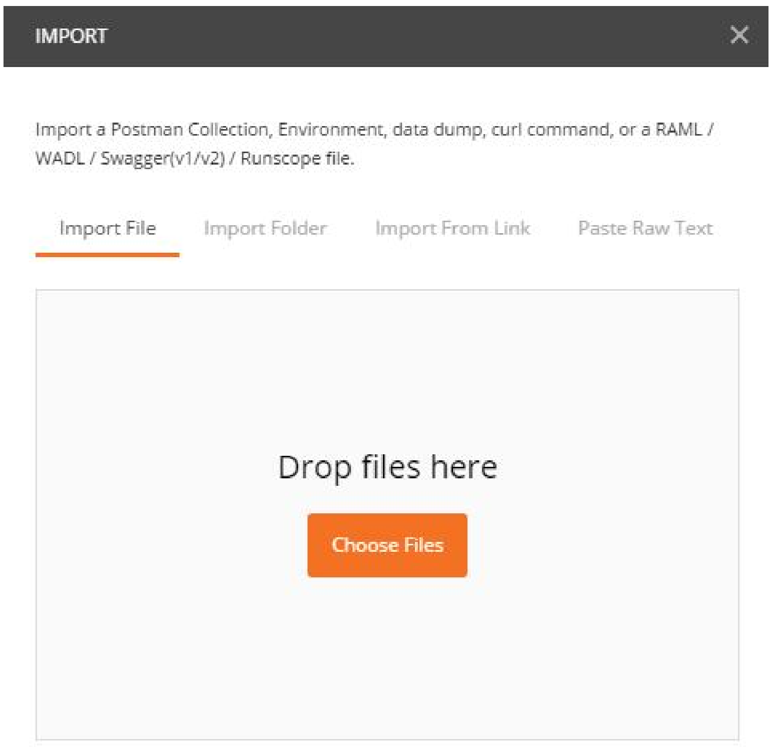
Scenario 2c: Testing the API

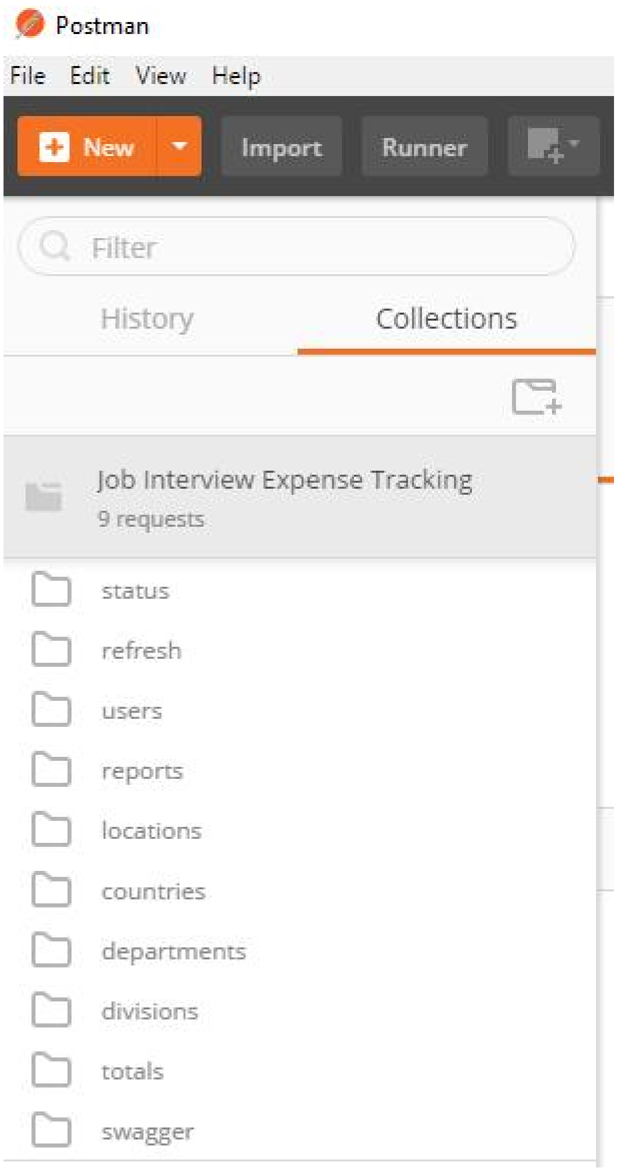
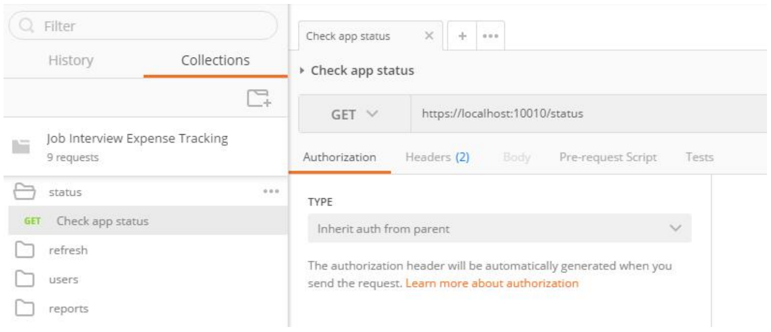
Explanation | **Screenshot**

1. Open a command line to start your node http-server.
   Go to the root folder of your service.
   use command: **npm start**
   Your local server is now started.



```
:\Users\[redacted]\Documents\CoolApps\2018\cloud_sfsf_concur_ext_sample-master\service>npm start

> concur-successfactor-aggregate-expense-management@0.0.1 prestart C:\Users\[redacted]\Documents\CoolApps\2018\cloud_sfsf_
ncur_ext_sample-master\service
> npm install

up to date in 0.97s

> concur-successfactor-aggregate-expense-management@0.0.1 start C:\Users\[redacted]\Documents\CoolApps\2018\cloud_sfsf_con
ur_ext_sample-master\service
> node index.js

Loading current state from ./.db.json
The DB file does not exist which is okay
Your server is listening on port 10010 (http://localhost:10010)
Swagger-ui is available on http://localhost:10010/docs
```

2. Open **Postman REST Client** and import **swagger.yaml** file from **File -> Import**	 IMPORT Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file. Import File · Import Folder · Import From Link · Paste Raw Text Drop files here Choose Files

3. You can see your API in the left pane.	
4. Select the **Check app status** under **status** folder to check if the server is accessible.	

The authorization header will be automatically generated when you send the request. Learn more about authorization