Accelerating Application Design With OpenAPI

Getting The Tools

Before you begin, you will need to ensure that your development environment is properly configured. To make that happen you will want certain tools installed and usable. Most of this tutorial will assume a UNIX-like environment (MacOS, Linux, WSL2).

- Install NodeJS
- Install a Java Virtual Machine

Conference Wifi Sucks!

We know, usually the conference wifi is so bad that you cannot actually download dependencies and whatnot. We've thought of that. Connect to the `REDHAT-CNAD` or `REDHAT-CNAD-5G` SSID. There is no password. When you're on that network, we are providing a NPM Proxy/Cache you can configure as follows:

```
mkdir -p /path/for/new/project
npm config -L project set registry http://nexus3.cpl:8081/repository/npm-proxy/
```

This will ensure that any projects you create starting in that directory will use our local Nexus Repository cache. We have pre-seeded all of the most important libraries for Quasar and this Vue application. If you choose to use a different toolkit, only the first copy will be downloaded and subsequent copies will come from the cache.

Bootstrap Your Project

- Create your new project. We will be creating a simple Todo list application
  - `npm init quasar`
- Change to the new project directory
  - `cd todo-quasar-openapi`
- Add Dev dependencies
  - `npm i --save-dev @openapitools/openapi-generator-cli @stoplight/prism-cli npm-watch`

- Create a basic OpenAPI contract

```yaml
openapi: 3.0.2
info:
  title: Todo
  version: 1.0.0
  description: My Application
servers:
  - url: "http://{domain}:{port}{base_path}"
    description: "API URL"
    variables:
      base_path:
        enum:
          - /
          - /api/v1
        default: /
      domain:
        enum:
          - localhost
        default: localhost
      port:
        enum:
          - '7080'
        default: '7080'
tags:
  - name: todo
paths:
  /health:
    get:
      operationId: getHealth
      tags:
        - todo
      responses:
        '200':
          description: 'OK'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Errors'
        default:
          $ref: '#/components/responses/Error'
components:
  responses:
    Error:
      description: Error
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Errors'
  schemas:
```

```
    Errors:
      type: object
      required:
        - code
      properties:
        timestamp:
          type: string
          format: date-time
        msg:
          type: string
        code:
          type: number
          format: int64
```

This is a reasonable start for any new REST API contract

- Add new scripts to `package.json`

```
"watch": "npm-watch",
"prism": "prism mock -d --cors -p 7080 openapi.yml",
"openapi": "openapi-generator-cli generate -g typescript-axios -i openapi.yml -o
src/sdk/ -p withSeparateModelsAndApi=true,apiPackage=api,modelPackage=models"
```

- Add `watch` block to `package.json` after the `scripts` block

```
"watch": {
  "openapi": {
    "patterns": [     ①
      "openapi.yml",
      "package.json"
    ],
    "inherit": true
  },
  "prism": true,      ②
  "serve": {
    "patterns": [     ③
      "package-lock.json",
      "package.json",
      "tsconfig.json",
      "quasar.config.js"
    ],
    "inherit": true
  }
},
```

① When either `openapi.yml` or `package.json` change, regenerate the OpenAPI Client code

② Ensure that the `prism` mock API server is running. It will automatically detect changes in the OpenAPI file.

③ When any of the core framework files change, restart the development web server

What did I just accomplish?

You have just created a new project using the Quasar framework for VueJS. You also added tooling which will allow you to create both a Mock API server (using Prism) but also generate the code which allows you to talk to that API automatically (using OpenAPI Generator). As we proceed, you will see that when we need a new data type or new API method, we can quickly add it to the `openapi.yml` file and the `npm-watch` tool will automatically regenerate the necessary code and restart the necessary services.

Open your project in your preferred IDE

These are IDE's I have had good luck with

- VSCode
- WebStorm

Connect our API client code to Vue

1. From the command-line in your project directory, use the Quasar CLI to create a new [Boot]() file.

   ```
   ./node_modules/.bin/quasar new boot --format=ts api
   ```

2. Fill in the logic of the boot file to be able to **provide** the API whenever it is injected.

   ```
   import { boot } from 'quasar/wrappers';
   import { TodoApi, Configuration } from '../sdk';

   export default boot((app) => {
     const config = new Configuration();
     const api = new TodoApi(config);
     app.app.provide('api', api);
   });
   ```

3. Add the new boot file to the Quasar project by editing `quasar.conf.js` and adding 'api' to the `boot` array.

   ```
   // app boot file (/src/boot)
   // --> boot files are part of "main.js"
   // https://v2.quasar.dev/quasar-cli-vite/boot-files
   boot: [
     'api',  // Add 'api' here.
     'axios',
   ],
   ```

Start Building Todo User Interface

- We know that we're going to need a Todo object type, so let's create that in the `openapi.yml`

```
components:
  schemas:
    NewTodo:
      type: object
      required:
      - title
      properties:
        title:
          type: string
          maxLength: 255
        description:
          type: string
        id:
          type: string
          format: uuid
```

- That will be a good object definition for when we are creating a new Todo item, but we also want some validation, so let's create a `Todo` type which has some required fields:

```
components:
  schemas:
    Todo:
      type: object
      required:
      - title
      - id
      allOf:
      - $ref: '#/components/schemas/NewTodo'
```

- Let's add a new endpoint to let us get the complete list of Todos

```
tags:
  - name: api
  - name: todo    ①
paths:
  /todos:
    get:
      description: Get all todos
      operationId: getAllTodos    ②
      tags:
        - todo
      responses:
        '200':
          description: 'OK'
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Todo'
```

① The tag becomes the name of the API object for this tag

② The `operationId` becomes the method name in the API object to call in order to access that endpoint

- Once we add these, save the file and start our `watch` script

```
 npm run watch

> codepalousa-todo@0.0.1 watch
> npm-watch

No task specified. Will go through all possible tasks
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older
versions may use Math.random() in certain circumstances, which is known to be
problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see
https://github.com/request/request/issues/3142

added 202 packages, and audited 661 packages in 15s

110 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
 npm run watch

> codepalousa-todo@0.0.1 watch
> npm-watch

No task specified. Will go through all possible tasks
[openapi] [nodemon] 2.0.19
[prism] [nodemon] 2.0.19
[prism] [nodemon] watching path(s): true
[nodemon] watching extensions: js,mjs,json
[prism] [nodemon] starting `npm run -s prism`
[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): package.json quasar.conf.js yarn.lock
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `npm run -s start`

 .d88888b.
d88P" "Y88b
888     888
888     888 888  888  8888b.  .d8888b   8888b.  888d888
888     888 888  888     "88b 88K          "88b 888P"
888 Y8b 888 888  888 .d888888 "Y8888b. .d888888 888
Y88b.Y8b88P Y88b 888 888  888      X88 888  888 888
 "Y888888"   "Y88888 "Y888888  88888P' "Y888888 888
       Y8b

[openapi] Download 5.1.1 ...
```

```
 » Reported at..... 8/11/2022 11:58:12 AM
 » App dir.................
/home/dphillips/Documents/RedHat/Workspace/CodepaLOUsa2022/codepalousa-todo
 » App URL................. http://localhost:9000/
                           http://172.18.12.1:9000/
                           http://192.168.1.252:9000/
                           http://10.22.10.143:9000/
 » Dev mode............... spa
 » Pkg quasar.............. v2.7.5
 » Pkg @quasar/app-vite... v1.0.5
 » Browser target......... es2019|edge88|firefox78|chrome87|safari13.1

 App  Opening default browser at http://localhost:9000/

[prism] [11:58:12 AM]  [CLI] ···  awaiting  Starting Prism···
[prism] [11:58:12 AM]  [CLI]   info      GET        http://127.0.0.1:7080/health
[prism] [11:58:12 AM]  [CLI]   start     Prism is listening on http://127.0.0.1:7080
[openapi] Downloaded 5.1.1
[openapi] [main] INFO  o.o.codegen.DefaultGenerator - Generating with dryRun=false
[openapi] [main] INFO  o.o.codegen.DefaultGenerator - OpenAPI Generator: typescript-
axios (client)
[main] INFO  o.o.codegen.DefaultGenerator - Generator 'typescript-axios' is considered
stable.
[openapi] [main] INFO  o.o.c.l.AbstractTypeScriptClientCodegen - Hint: Environment
variable 'TS_POST_PROCESS_FILE' (optional) not defined. E.g. to format the source
code, please try 'export TS_POST_PROCESS_FILE="/usr/local/bin/prettier --write"'
(Linux/Mac)
[main] INFO  o.o.c.l.AbstractTypeScriptClientCodegen - Note: To enable file post-
processing, 'enablePostProcessFile' must be set to `true` (--enable-post-process-file
for CLI).
[openapi] [main] WARN  o.o.codegen.DefaultCodegen - Unknown `format` int64 detected
for type `number`. Defaulting to `number`
[openapi] [main] WARN  o.o.codegen.DefaultCodegen - Unknown `format` int64 detected
for type `number`. Defaulting to `number`
[main] WARN  o.o.codegen.DefaultCodegen - Unknown `format` int64 detected for type
`number`. Defaulting to `number`
[openapi] [main] INFO  o.o.codegen.TemplateManager - writing file
/home/dphillips/Documents/RedHat/Workspace/CodepaLOUsa2022/codepalousa-
todo/src/sdk/models/errors.ts

// SNIP . . .

[openapi] [main] INFO  o.o.codegen.TemplateManager - writing file
/home/dphillips/Documents/RedHat/Workspace/CodepaLOUsa2022/codepalousa-
todo/src/sdk/.openapi-generator/FILES
[openapi]
################################################################################
[openapi] # Thanks for using OpenAPI Generator.
#
# Please consider donation to help us maintain this project                    #
# https://opencollective.com/openapi_generator/donate                           #
```

```
[openapi]
################################################################################
[openapi] [nodemon] clean exit - waiting for changes before restart
```

What is happening here?

By defining the `NewTodo` and `Todo` schemas along with the `/todo` **GET** operation in the `openapi.yml` file and starting the watch, prism and openapi-generator start up the mock API server and generate the client-side code for talking to the API. The API client code can be found in `src/sdk` and we will use it to talk to the mock API as we develop the user interface application.