Behavior Driven Development in Vue with Cypress and Cucumber

Pit Baum

December 2021

1 Introduction to BDD

Behavior driven development or for short BDD, is an approach to the organizational challenges of software development, that is meant to increase productivity in the market field by easing the communication between technical and non-technical stakeholders and at the same time makes documentation and testing of the produced code part of the workflow.

1.1 Explanation of the BDD workflow

Behavior Driven Development (BDD) is an agile work process in software development, designed to strengthen the enhance the communication and teamwork between the quality management and business analysis parts within a software development project. It emerged from the idea of Test Driven Development (TDD) and domain-driven design, to provide software developers and team managers, thus both technical and non-technical staff with the necessary tools and means of communicating with each other during their collaborative work.

During the BDD workflow in general, the tasks, goals and results of the Software are determined at the beginning of the Specification analysis and summed up in a text form readable to non-technical staff, which all happens in a Domain Driven Design manner. Afterwards, those specifications of the to be developed software will be translated into automated tests, which is where the TDD approach comes into play. In the end the software developers will implement the features and functionalities of the software such that they pass all the predefined tests. This has as an advantage, that both the technical and non-technical staff working on the project know at any point in time which features are included in the software and who expects what from whom. Furthermore, mostly relevant for the technical staff of the project, this approach to software development, means that the software that has to be developed is already split into smaller parts and thus easier to create, in a divide and conquer approach, which is commonly used in the IT sector, for any larger task.

1.2 BDD in context of this example

For this specific tutorial, as the title tells, we are looking into BDD in a specific framework. Hereby the Domain Driven Design part will be handled by the Cucumber pre-processor, the TDD part will be handled by the Cypress testing tool and the main software development part will be in the domain of frontend web development in Vue.

2 Installation of the setup

Before we can start and look at a concrete project idea to show the BDD process in practice, we will have to install the necessary framework and configure it's dependencies. First we start off with a package manager, make sure to have either npm or yarn commands running on your console to install the necessary packages throughout this section.

2.1 Globally install VueCli

If that is done, open the terminal of your System and run the command:

npm install -g @vue/cli ***Or*** *yarn global add @vue/cli*

This will install VueCli globally on your system, such that you can no setup new Vue projects.

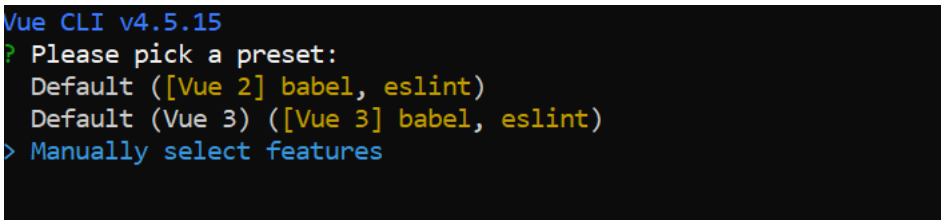
2.2 Initialize a Project (In depth)

After having installed VueCli, we can now start creating a new Vue project, with the necessary configurations to fit our BDD process with Cypress and Cucumber. Navigate in the terminal to the directory you want to create the new Vue project folder in and then start the process with the command:

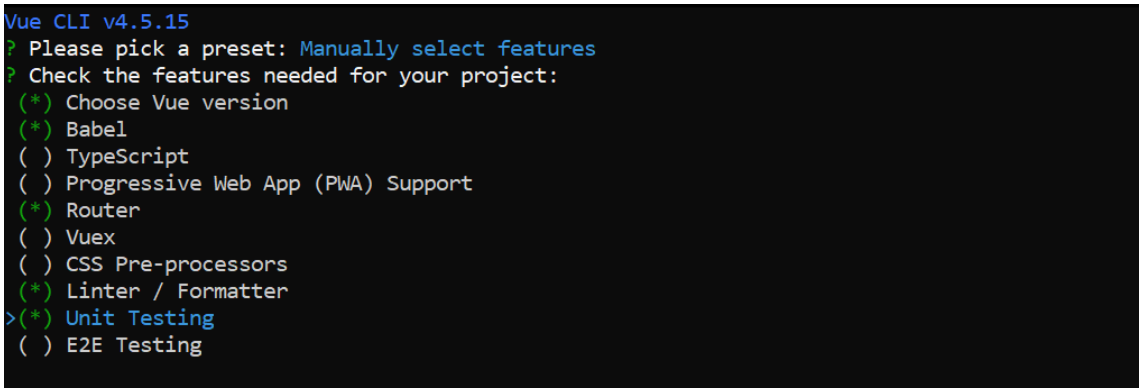
vue create appnamehere

Replace appnamehere with the name you want to give your project directory, the applications name must be completely written in lowercase.

The next thing to do now, is to pick the necessary installation settings before the installation can be started. Since none of the preset configurations include the necessary libraries, we will opt for the manual selection of features, which we select by pressing the Enter key.



Select the features necessary for our project: Tick the same options as the once shown in the image below by selecting them and pressing the Space bar and then confirming your choices after having ticked them by pressing Enter.



Next we will choose Vue version 3.x.

Enter n for Using the history mode for the router.

And select ESLint + Prettier for linter/ formater configuration.

Such as Lint on save for the additional lint features.

When you are asked to pick a Unit testing solution, we will pick Jest.

And finally all our configuration files will be put into a package.JSON, so select package.JSON.

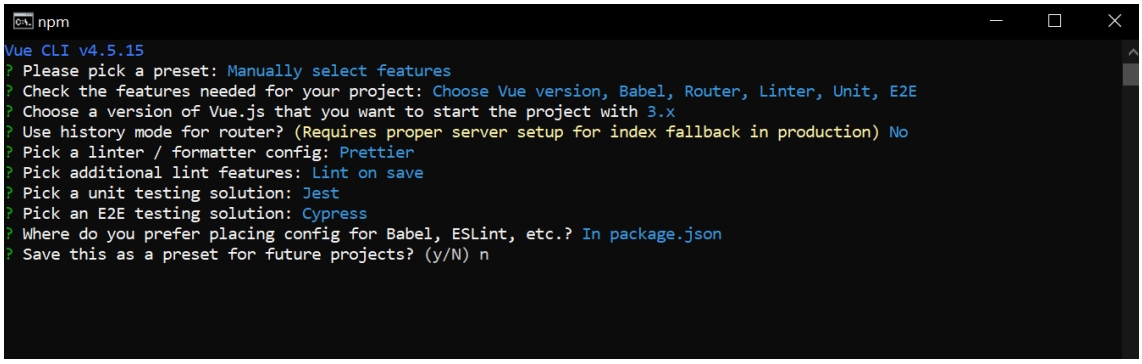
Lastly, press n and before you hit enter to not save it as future configuration, you can verify your installation configurations one last time, by making sure that the values you have selected before in the console are the same as in the image shown below in the quick start installation section.
When you are done, hit enter and wait for the installation to finish.

When the installation is done, you can check upon the correctness of the installation, by navigating into the newly created project directory and entering

the command npm run serve. Normally it should compile and if you open the localhost route now in a browser, you should be able to see the standard example application of Vue. Don't forget to close the running server again, before continuing with the configuration steps.

2.3 Initialize a Project (Quick start)

If you don't need a long walk through of the installation settings and just want to quickly look up the selected configurations again, then just run the Vue create appnamehere command in your terminal at the directory you want and imitate the settings as shown in the image of a terminal below.



2.4 Install Cucumber and Cypress

Before we start with configuring cypress and some other necessary dependencies, we are going to install the Cucumber pre-processor, which will make it possible to write our Cypress E2E testing code in a more non technical manner.

To install Cypress now, run the following commands in your projects directory:

npm install cypress
Or
yarn add cypress–dev

Once done, open cypress up, such that the necessary folders can be created. to open Cypress, use the Command:

npm install cypress

After the Cypress window has opened up, close it again and install the plugin dependencies to run cypress with cucumber.

npm install –save-dev @cucumber/cucumber
Or
yarn add –dev @cucumber/cucumber

Now we are ready to start making the configurations in our installed files. First in your projects directory, there is a file called package.json, which includes all of the dependencies that are necessary and being loaded when the program is executed. Add into that file the following line:

"cypress-cucumber-preprocessor": {"nonGlobalStepDefinitions":true}

Make sure that the syntax of the JSON file isn't vioalted by your additional line and that you add commas in front of the previous statement if necessary.

Next, also in the main directory of the project, we will edit the cypress.json file. Here we will add into the braces in the file the following lines:

"baseUrl": "http://localhost:8080",
"chromeWebSecurity": false,
"testFiles": "**/*.{feature,features}"

The commands added will set the standard route for cypress to run the tests on, to the local host and will set the feature files suffix to ".feature" or ".features".

Now we will let Cypress know that it has to render cucumber before interpreting the feature files. For that we go from the main directory of the project to the file cypress/plugins/index.js Here we will add two things, first we add the requirement for cucumber as a constant and then we apply it on the run module of cypress. This we do by adding above the already existing module.exports function the line:

const cucumber = require('cypress-cucumber-preprocessor').default

And then add inside of the module exports function the line:

on("file:preprocessor", cucumber())

In the end it should look like this. Notice that in the module exports function I have additional other returned Object assignments, those are standard directories assigned for different things you can import into Cypress, like fixtures or images. if you want to change those directories, you can do so by the assigning/adding them here.

Now we are technically all done with the configurations being ready and set to implement our first couple of features. But since we installed our project with VueCli, we might want to make sure that all the versions of our dependencies are up to date or at least match the standard we need in our project. Thus i would recommend to go back to the package.JSON file in the projects main directory and check the devDependencies and update the versions if necessary.

2.5 Clean up the example files

Since our installation comes with some already pre-installed example files and tests, we want to now clean up all those examples and create our own features, tests and Vues.

First we want to get rid of the already pre-written test from Cypress, which are located under the path cyress/integration and then delete the folders "getting started" and "advanced examples", since this article includes it's own examples. Now we want to get rid of the HelloWorld example of Vue in our frontend code. For this we go back to the main directory and under the path src/views we delete both the About.vue and the Home.vue file. Furthermore we want to delete in the src/components directory the HelloWorld.vue file.

And update our src/router/index.js file, since we have deleted the router references, we should clean them up, so we can later add new once of our own project into it. Delete in the index.js file the import home statement and the contents of the const routes = [];

If you are unsure about how a cleaned up project repository should look like more or less, you can look it up on the Githhub repository for the project example, the branch 01_Introduction, includes only the necessary files and from that gets extended into a website with a login, logout function and a dashboard.

3 A guided example project

After having gone through a brief explanation on BDD, having done the installation of the necessary tools, configured the necessary options and cleaned up all the directories from preview examples, it is time to get to see a real example of the BDD process. For that, please visit the Github repository:

https://github.com/pitbaum/BDDTutorial/tree/main

And follow the instructions and explanations made in the Readme files of the different branches.
This concludes this article, no matter if you decide to follow the practical example in the Github or just head straight into your own first project with BDD, I wish you good luck and happy coding.