# C1 - MVVM frameworks

C-DEV-121

# MVVM projects

nice 2 have

Vue.js

1.0

## Using the Vue Router

All projects must use [Vue Router] (https://www.npmjs.com/package/vue-router) (the official router associated with Vue.js).

### Why?

All projects require you to create several different pages. To navigate between these pages, a route management system is required.

### How to detect the use of a `Vue Router` in the project?

- the `package.json` file must contain the `vue-router` dependency.
- the router is usually located in `./src/router/index.js`.

---

## The purpose of dynamic routes in the application router

Like the dynamic routes offered by the server (for example: GET `/pokemons/{id}`), the application's router must offer a dynamic route to obtain information about Pokemons.

### How to detect the dynamic route?

In this file which manages the routes, a dynamic route is expressed like this:

```
{
    path: '/pokemons/:id,
    name: 'pokemon',
    component: Pokemon
}
```

Note the dynamic parameter `:id` for the `path` property (with ':', which indicates its dynamic character).

### Why?

The new version of Vue.js includes many new features, including [the Composition API] (https://v3.vuejs.org/guide/compo api-introduction.html). By using this new version, the project is therefore more open to the latest developments of the Vue.js project and is therefore a little more "future-proof" than if it were using Vue.js 2.x.

### How to detect the use of Vue.js 3 in the project?

- the `package.json` file must contain the`vue` dependency with a version number greater than `3.0.0`.
- the use of the `setup` option (typical use of the Composition API) in Vue.js components (https://v3.vuejs.org/guide/co api-introduction.html#setup-component -option).

---

## Using ES2015 (and later versions)

All projects can use modern JavaScript functionality.

### Why?

Using recent JavaScript features provides several advantages, including (among others):

- functionalities adapted to different code and algorithm issues (ex: `Object.assign`)
- better code readability (ex: [`String.prototype.includes ()`] (https://developer.mozilla.org/en-US/docs/Web/JavaSc

The adoption of modern JavaScript functionalities is all the more valuable since it is offered by default by `Vue CLI`.

### How to detect the use of ES2015 + in the project?

ES2015, ES2016, … up to ES2021 have too many features for a generic detection solution to be offered. Exhaustive lists of these features are available here: https://github.com/daumann/ECMAScript-new-features-list

---

## Using a UI-kit type framework

All projects can use a framework offering reusable graphic components (buttons, form elements, grids, …)

### Why?

UI-kit type frameworks facilitate the creation of UX/UI by offering functional graphical components.
The most common frameworks of this type are:

- Quasar
- Vuetify
- vue-material

Many other frameworks of this type exist: https://github.com/vuejs/awesome-vue#frameworks

### How to detect the use of a UI-kit framework in the project?

- the `package.json` file must contain the corresponding dependency

---

## Using a CSS pre-processor

All projects can use a CSS pre-processor to improve style sheet writing.

### Why?

CSS pre-processors allow tidier and more succinct CSS style writing, and provide advanced style sheet writing features (https://cli.vuejs.org/guide/css.html). By integrating this type of technology, the project gains flexibility and robustness in terms of style writing.
The adoption of a CSS pre-processor in the project is all the more valuable as it is offered as an option by `Vue CLI`: the main CSS pre-processors are supported by`Vue CLI`:

- [SASS or SCSS] (https://sass-lang.com/)
- [LESS] (http://lesscss.org/)
- [Stylus] (https://stylus-lang.com/)

### How to detect the use of a CSS pre-processor in the project?

- the `package.json` file must contain the corresponding development dependency ("devDependencies").
- the project includes files with the corresponding extension (.sass, .scss, .less, …).
- some components integrate the preprocessor in their style block, for example: `<style lang =" scss ">`.

---

## Using Eslint

All projects can use Eslint to ensure a consistent and regular code style across the project and to safeguard itself against syntax errors.

### Why?

Eslint allows, by means of a set of rules, to locate, and possibly correct, JavaScript syntax errors throughout the project. Via the rules, it also makes it possible to alert and correct the JS code in the event of syntax inconsistency; here are some examples of rules (all modifiable):

- JS lines of code must end with a semicolon.
- we must use the syntax `String.prototype.includes()` instead of the syntax `String.prototype.indexOf()> -1`.
- tabs should be used instead of spaces to indent the code.

The adoption of Eslint in the project is all the more valuable as it is offered as an option by `Vue CLI`, with various rule presets available.

## How to detect the use of Eslint in the project?

- the `package.json` file must contain the development dependency ("devDependencies") `eslint`.
- the `package.json` file must contain the scripts related to "linting": the`lint` script launches a `vue-cli-service lint` type command.
- the JS syntax is consistent throughout the project and no error is reported when launching the above-mentioned script.

---