**Пример структуры.**

/assets

/styles

/imgs

/icons

/fonts

/directives

directive\_name\_directive.js

/filters

filter\_name\_filter.js

/models

model\_name.js

/mixins

/mixin\_name\_mixin.js

/pages

/page\_name

page\_name\_page.vue

/components

/component\_name

componetn\_name\_component.vue

/ui

button\_ui.vue

dropdown\_ui.vue

/models

model\_name.js

/components

/component\_name

component\_name\_component.vue

/directives

directive\_name\_directive.js

/filters

filter\_name\_filter.js

/mixins

mixin\_name\_mixin.js

/ui

button\_ui.vue

dropdown\_ui.vue

/models

model\_name\_model.js

/router

router.js

/store

index\_store.js

actions\_store.js

mutations\_store.js

/modules

module\_store.js

/ui

button\_ui.vue

password\_ui.vue

App.vue

main.js

**Описание.**

Можно условно разделить проект на глобальную (верхнеуровневую) и узкоспециализированную (дочернею) структуру. В общем, верхнеуровневая структура может совпадать с вложенной в директориях components структурой. Разница в том, что в верхнеуровневых папках лежит общий функционал который можно переиспользовать в различных компонентах не связанных структурно. Дочерняя структура компонентов содержит только тот функционал, который специфичен исключительно для решения задач этих компонентов.

К именам файлов добавлять в конец тип файла (например для компонента voucher/voucher\_component.vue, для модели voucher/models/voucher\_model.js, для стора voucher/store/voucher\_store.js). С одной стороны, возможно не всегда будет необходимость задавать имена с добавлением типа (например router.js), с другой стороны это даст некое единообразие в наименовании, и в дальнейшем при только одном взгляде на название файла будет понятно какой уровень ответственности в нём выполняется. В начало названия компонента можно добавлять дополнительный идентификатор который будет однозначно указывать на то, что этот элемент разработан нами (например ntt\_voucher\_component.vue)

Возможно имеет смысл дополнительно в компонентах создавать директории в которых располагается логически сгруппированный функционал (например .../voucher/voucher\_edit и .../voucher/voucher\_view).

Входной файл должен лежать в своей собственной папке и именоваться схоже с названием папки + тип файла (например есть компонент который описывает таблицу пользователей: /pages/user/components/table\_users/table\_users\_component.vue). Если же папка общая и не предполагает дополнительной вложенности то именовать согласно компоненту к которому файл относится (например есть директория .../component/models тогда файлы можно назвать например user\_model.js, guest\_model.js, итд).

При работе над некоторыми задачами оказалось полезным использовать компоненты которые решают исключительно задачу размещения элементов в UI без какой-либо логики. Данный тип компонентов содержит только разметку в которых лежат слоты и css для унификации отображения элементов.

Нужно определиться со стилем применения компонентов в шаблонах (например </MyApp> или </my-app>).

**Используемые ресурсы**.

- Структура для VUEX: <https://vuex.vuejs.org/guide/structure.html>

- Статья для крупных приложений на VUE: <https://habr.com/ru/companies/otus/articles/571272/>