# Venus

## Requirements

* Asp.net Mvc 5 (.Net Framework).
* Azure AD for Open Id Authentication (Back Office Authentication).
* Vue JS.
* Typescript.

## Dependencies

* Vue : The view library.
* vue-class-component, vue-property-decorator: To create typescript class component.
* Vuex: The store library.
* vuex-class : To create vuex store using typescript.
* vue-router : The router library.
* Axios : A http client.
* lodash-es, @types/lodash-es : An utilities library (es6 release).
* Moment : A date time library.
* Bootstrap : The css framework.
* bootstrap-vue : A view components library based on bootstrap styles.
* vuejs-datepicker : Rich date picker component.
* @fortawesome/fontawesome-free : Icon library.
* core-js : Polyfills for old browsers (like ie11) and not supported functions (es5).

## VueJs Project Structure

```

+public

|++favicon.ico

|++index.html

+tools

|+apiMockServer.js

+src

|+ screens

||++ screen1

|||+++ components

|||+++ index.vue

||++ screen2

|||+++ components

|||+++ index.vue

||++ screen3

|||+++ components

|||+++ shared

|||+++ index.ts

||||++++ screens4

|||||+++++ components

|||||+++++ index.ts

|+ assets

||++images

||++icons

||++fonts

|+ layout

||++ index.vue

||++ components

|||+++ Sidenav.vue

|||+++ FooterPage.vue

|+ router

||++ index.ts

|+ store

||++ index.ts

||++ types.ts

||++ modules

||+++ module1

|||++++actions

|||++++getters

|||++++mutations

|||++++state

|||++++types.ts

|||++++index.ts

|+ services

||++api

|||+++ service1

||||++++index.ts

||||+++ service2

|||||++++index.ts

||++http

|||+++ index.ts

|+ theme

||++ main.scss

||++ default\_bootstrap.scss

||++ default\_theme

|||+++ default\_theme.scss

|+ shared

||++ components

||++ utils

||++ services

||++ constants

|+ mixins

|+ App.vue

|+ main.ts

|+ env.ts

```

* Public: Contains the html template.
* Tools: Contains a mock api server for tests.
* Src : Source folder :
* Main.ts: It’s the entry point, we mount here the vue application to the dom.
* App.vue: The main application view, it contains global styles (from theme folder), mixins…
* Env.ts: contains environment variables coming from the build.
* Assets: Contains application resources (images, icons, fonts…)
* Mixins: contain global/shared functions.
* Theme: The application theme includes a customization of bootstrap and the site styles.
* Layout: contains the layout view and its components (sidebar, navbar, footer…).
* Router: Contains the router configuration, we are using the “vue-router” library.
* Store: Contains the store configuration, we are using “vuex” as a store system. All store modules are stored is this folder.
* Services: Contains application services (http, api).
* Shared : Contains shared components (TextInput, CascadeDropdown, MultiSelectList…) and utilities.
* Screens: Contains all application views (ie : when clicking on a link, a view is displayed).

Imagine we have a menu with a two links : Home, About => So Home is our first view, About is our second view. Every view contains a list of components, a component is a part of the view. If a component is shared between two views (screens), it should be in the shared folder inside the top level just before the current level.

## Dotnet project structure

1- Data

- St.Venus.Data.csproj

2- Business

- St.Venus.Business.csproj

3- Presentation

- St.Venus.Web.csproj

Shared

- St.Venus.Core.csproj

Tools

- Database.sqlproj

Tests

St.Venus.Web.Tests.csproj

St.Venus.Business.Tests.csproj

St.Venus.Data.Tests.csproj

Warning : The only flow between layers :

Presentation -> Business -> Data

Only Core.csproj should be shared between layers.

## How to use ?

* Clone the remote repository.
* Build the solution.
* Change the working directory to src/St.Venus.Web/client-app
* Run : npm install
* Run the web application in any mode (Debug/Release) to expose api.
* Run the vue application : npm run serve
* Begin development, hot reloading is activated.
* You want to create a build, Run : npm run build , then run the web application.