Nova Web Application – Folder and Component structure

(based on Pavlo Pochapsky ([mpavlo.pochapsky@blueprintsys.com](mailto:mpavlo.pochapsky@blueprintsys.com)) compilation)

The directory/file structure of NOVA should resemble a feature-based hierarchy that might resemble something like a tree below.

* **src**
  + **app (modules)**
    - **shared**(contains small shared component, directives, helpers to be used by any modules and group by functionality)
      * **widgets**  
        (contains common and simple widgets like following)

bp-avatar

bp-tooltip

bp-accordion

…

bp-tree

* + - * **utils**   
        (contains helper services)
    - **core**(contains core functional component like following )
      * **configuration**  
        (contains configuration parameters and constants, paths and etc.)
      * **localization**  
        (contains localization service)
      * **authentication**  
        (including services and UI component)
      * **logger**  
        (services, interceptors and etc.)
      * **messages**  
        (contains model, service(s) and UI component to create, maintain and display all messages)
      * **models**  
        (contains emuns, classes and models descriptions to be used across all modules )
        + IItemTypePredefined
        + IPropertyTypePredefined
        + IArtifact
        + IProject
        + …
    - **editors**  
      (collection artifact editors grouped by artifact type)
      * bp-artifact-list
      * bp-storyteller
      * bp-diagram
      * bp-glossary
      * bp-artifact
    - **components**
      * bp-utility-panel
      * bp-project-explorer
      * bp-toolbar
      * …
    - **main (Main Experience**)  
      (view, services, component used by main module exclusively)
  + assets
  + styles

In addition, the project should be built around small modules, so instead of having a couple of large projects where most components should be registered, we can have a bunch of smaller modules, each of which would maintain their own components. Main chunks of the application, in turn, would simply manage lists of module dependencies, making the application a lot easier to refactor and maintain.