SIEL front-end structure:

src

     | app

               | core

               | $FEATURE-1

               | $FEATURE-2

               | shared

Core module: Eager module without routing, contains code to instantiate application and core functionality. For example: Global HTTP service, 3rd party libraries …

Feature module: Lazy module with routing, contains code related to user stories.

Shared module: Eager module without routing, contains code to use across application and feature modules.

Creating Angular files:

-          Path:

If the file should be used when instantiating application or related to 3rd party libraries, put in “core”.

If the file should be used between multiple modules, put in “shared”.

Otherwise, put into related lazy module based on domain or business context.

-          Naming: **$NAME$.$TYPE$.$EXT**

**$NAME$**

+ Search list: Use domain entity name in plural.

For example: Personne entity => personnes.component.ts

+ Detail screen: Use domain entity in singular.

For example: Persone entity => personne.component.ts

+ Others: Use Frenchnoun, related to user story

For example: User story GP-01.2 – Gestion des proprieties d’une personne => personne-proprietes.component.ts

**$TYPE$:**Related to context type. Can be: “component”, “modal”, “directive”, “pipe”, “guard”, “service”, “resolver” …

**$EXT:**Can be: “ts”, “html”, “scss”

-          Must use changeDetection: ChangeDetectionStrategy.OnPush for Angular component.

-          Do not use getter/setter in Angular component with heavy work. E.g: loading backend data.

-          Do not use NgModel for form data.

-          Use FormControl validator to validate field data. To validate combination field logic use FormGroup global validator.

-          Do not use Javascript for layout component.

-          Do not use inline style in HTML template.

-          Use decorator component instead of inheritance.

Styling:

-          Always use function “standard-size-calculator” to compute sizing (margin, padding, width, height …)

-          Use or put new variable for font-family font-size, font-weight, color in variables.scss.

-          Override style of 3rd party libraries (Bootstrap, Kendo, …) in theme-override.scss

-          Implement global custom style for application in styles.scss

-          Style that is only related to a component will be implemented in component \*.scss files.

-          Never use HTML tag as css selector for styling, please use css class.

-          Do not use child selector “>” for styling, please use css class.

-          Do not nested more than 3 levels in scss.

-          Do not use !important. Only use !important in case of overriding a 3rd party libraries !important.

-          Do not use ::ng-deep to override internal project component style.