ERP-Coin Coding Standards

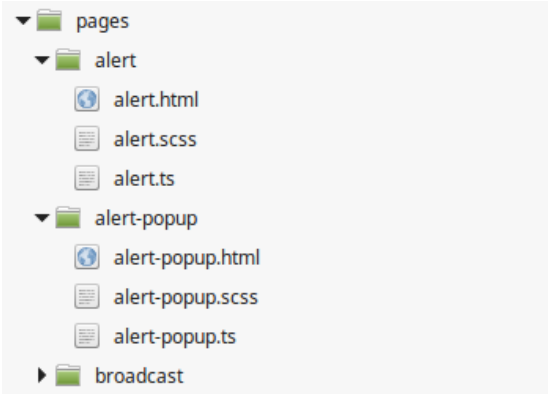
TeamTuring

April 2018

1 File Names:

Ionic:

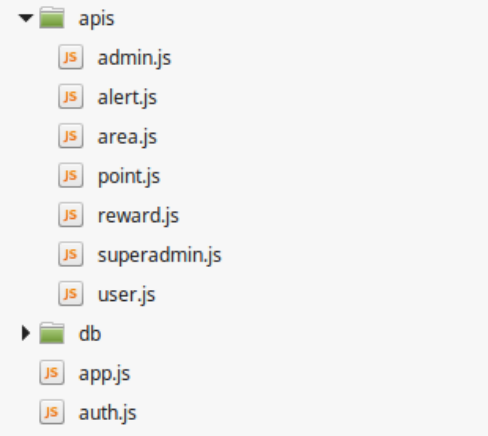
- When a new page is created in Ionic, the framework will use the page name on all accompanying files.
- The name of the page should clearly state it's purpose.
- The name should be one or two words, with no caps are used, and a '-' is used to separate words.
- Example:



Nodejs:

- For back-end files again no caps are used.
- The name of the page should clearly state the area of the system it contains API's or functions for.
- The file name again is one or two words, and a '-' is used to separate words.

- Example:



2 File Headers:

- All files should be accompanied by a header stating the author and describing all technologies used, functionality, as well as a basic description.
- The following structure should be used for all source code files generated by the user and not by the framework:

```
1  ----------------------------------------------------------------------
2  Project:        ERP-Coin
3  Version Number: 1.0.0 / 2018
4
5  Company:        EPI-USE
6
7  Author/s:       Kyle Pretorius
8                  Ulrik de Muelenaere
9                  Tristan Rothman
10                 Sewis van Wyk
11                 Richard Dixie
12                 Darius Scheepers
13
14 @Nodejs Version 10.0.2 (or Ionic version, or SQL, etc.)
15
16 Purpose Statement:      (one or two line description)
17 ----------------------------------------------------------------------
```

3 Comments:

- All files should be accompanied by a header as stipulated above.
- Comments giving information on an instruction should appear in the same line, but after the instruction.
- Comments giving information on a block of code should appear on its own line, before the block.
- If code is pushed that contains errors, the author should clearly leave a comment at the suspected block of code.

4 Indentation:

- A tab character should be used for indentation.
- Indentation is used for any sub-code, for example all code within a class should be indented, all code in a loop should be indented, and all code in a function should be indented.
- If a single instruction is wrapped to more than one, the accompanying lines should be indented.
- Indentation should not be used otherwise, to avoid confusion.
- Example:

```
this.http.post("/admin/login", jsonArr).subscribe
(
  (data) =>
  {
    var jsonResp = JSON.parse(data.text());
    if(jsonResp.success)
    {
      this.presentToast("Logged in!");
      this.navCtrl.push(TabsPage);
    }
    else
    {
      alert("Invalid Login. Try Again.");
    }
  },
```

5 Line length and Wrapping lines:

- Lines should not contain lines relatively longer than the average of the rest of the code.
- If lines are too long it should be wrapped and indented to avoid confusion, preferably after a comma or before an operator.
- Lines should contain one instruction only, new instructions should start on a new line

6 Blank Lines and Blank Spaces:

- Blank lines and blank spaces should be used extensively to increase readability
- Blank lines must be used between sections of a file, between classes and definitions, between methods, before a block or a single-line comment, and between complex sections to improve readability.
- Blank spaces should be used before and after all operators, before and after parenthesis (except in a function call), after commas, and anywhere else where the author feels readability can be improved, without sacrificing functionality.

7 Naming Conventions:

- Class names should be nouns, and the first letter must be Capitalized, together with the first accompanying word, without using spaces.
- Method names should be verbs, and start with a lower case letter, only capitalizing the first letter of the following words, without using spaces.
- Variable names should use the same convention as Methods in terms of capitalization. Underscores may be used to substitute a space.
- Constant names and variables should be in all capital letters.