UCI

Frontend training

antu@ngiro.net

Agenda

1. Pre-Requirement
2. Novicell Frontend package
3. Atom design
4. Fractal
5. Fractal component and Handlebar
6. Styles in Fractal
7. PostCSS & BEM
8. More style trick and good practices
9. Scripts in Fractal
10. Development process and deployment
11. Integration with Sitecore
12. ECOWEB case overview
13. QA

Pre-Requirement

Pre-Requirement

To attend the workshop you need good Knowledge on:

- CSS
- HTML5
- Javascript

Basic notion on:

- Git
- NPM
- CSS preprocessor
- Web development process

Pre-Requirement

Before the workshop, you need to have your local computer setup with:

- Git installed
- Optionally some git client (ex: gitkraken)
- Node.js min. v. 14.15.0 with NPM 6.14.8 installed
- Visual Code Studio installed
- Novicell frontend framework installed in a local folder. Use the following repository on master branch for it: https://github.com/agiraud/frontend-training

Novicell frontend package

framework that speeds up front-end development process and ensure frontend code quality and optimisation.

Novicell Frontend package technologies

- https://docs.npmjs.com/about-npm: software registry
- https://webpack.js.org/: assets bundle
- https://postcss.org/: css processor
- https://stylelint.io/: css style convention
- https://eslint.org/: JavaScript quality
- https://babeljs.io/: JavaScript compiler
- https://fractal.build/: component libraries & style guides
- https://handlebarsjs.com/: HTML templating

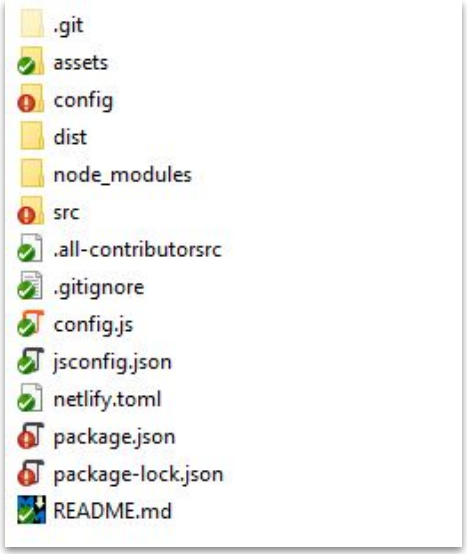
Novicell Frontend package tools

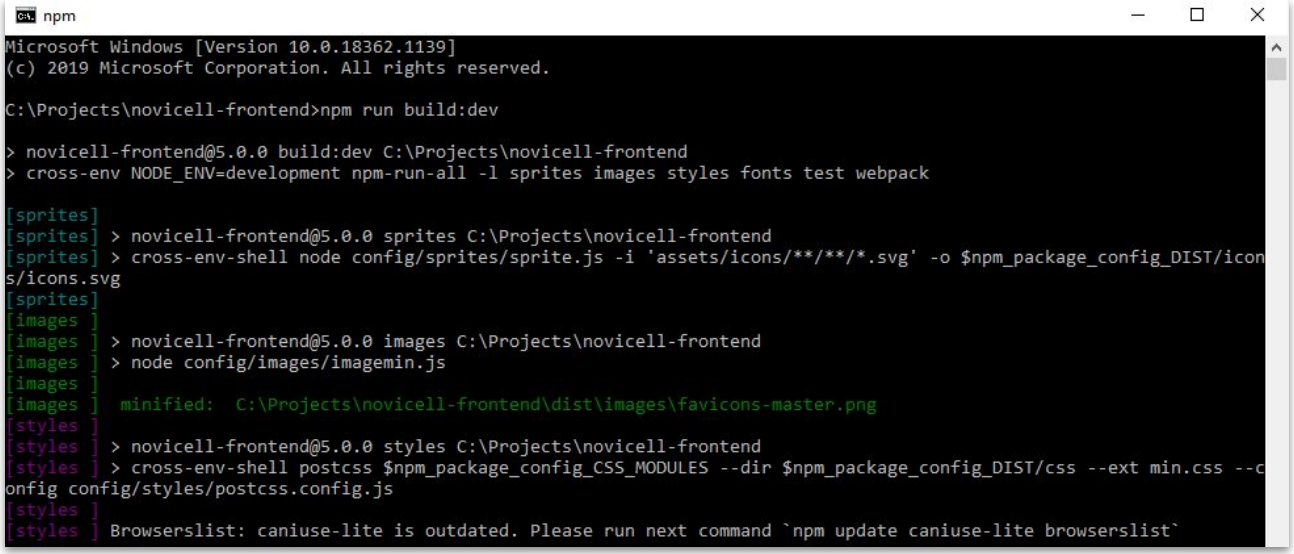
- Pre-configured libraries
- Set of NPM Scripts
- Default atom folder structure
- Component helper
- Default Grid and variable list
- SVG Sprites
- Open-source project

https://github.com/Novicell/novicell-frontend/wiki/SVG-Sprites

Novicell Frontend Setup

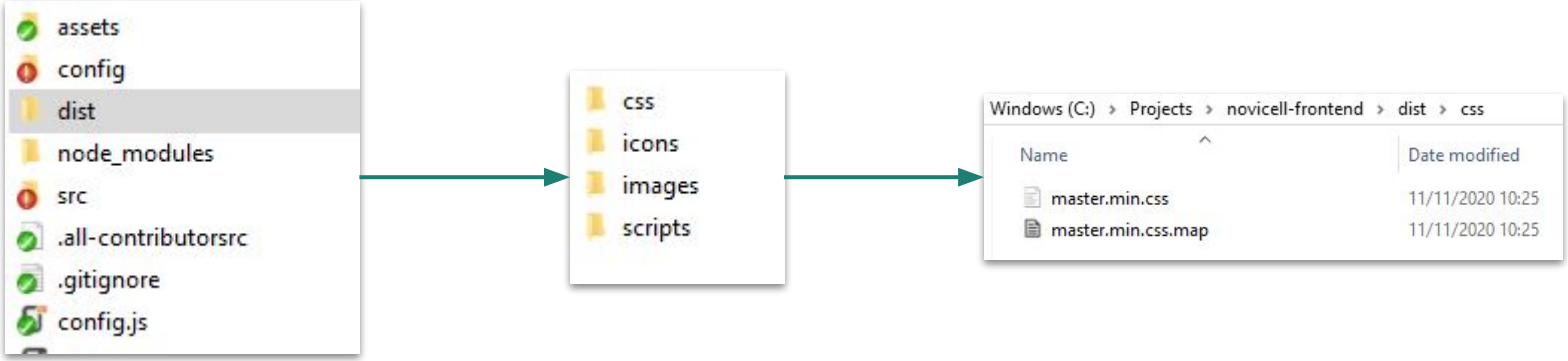
https://github.com/Novicell/novicell-frontend#setup





Exercice 1 - 5 mins

1. From Visual Studio Code, open the project with “Open Folder”
2. Open a new terminal
3. Try to run “npm run build:dev”
4. Check that the dist folder has been created and master.min.css compiled



Atomic design

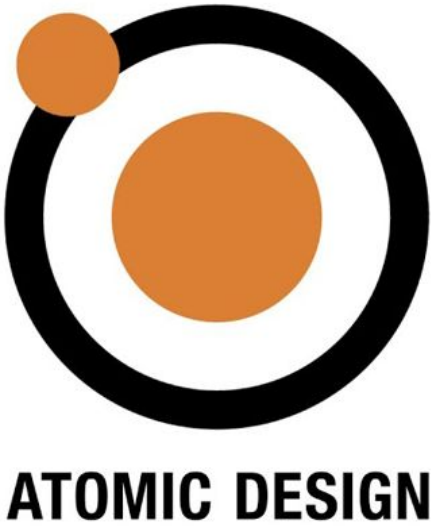
“We’re not designing pages, we’re designing systems of components” — Stephen Hay

atomic design

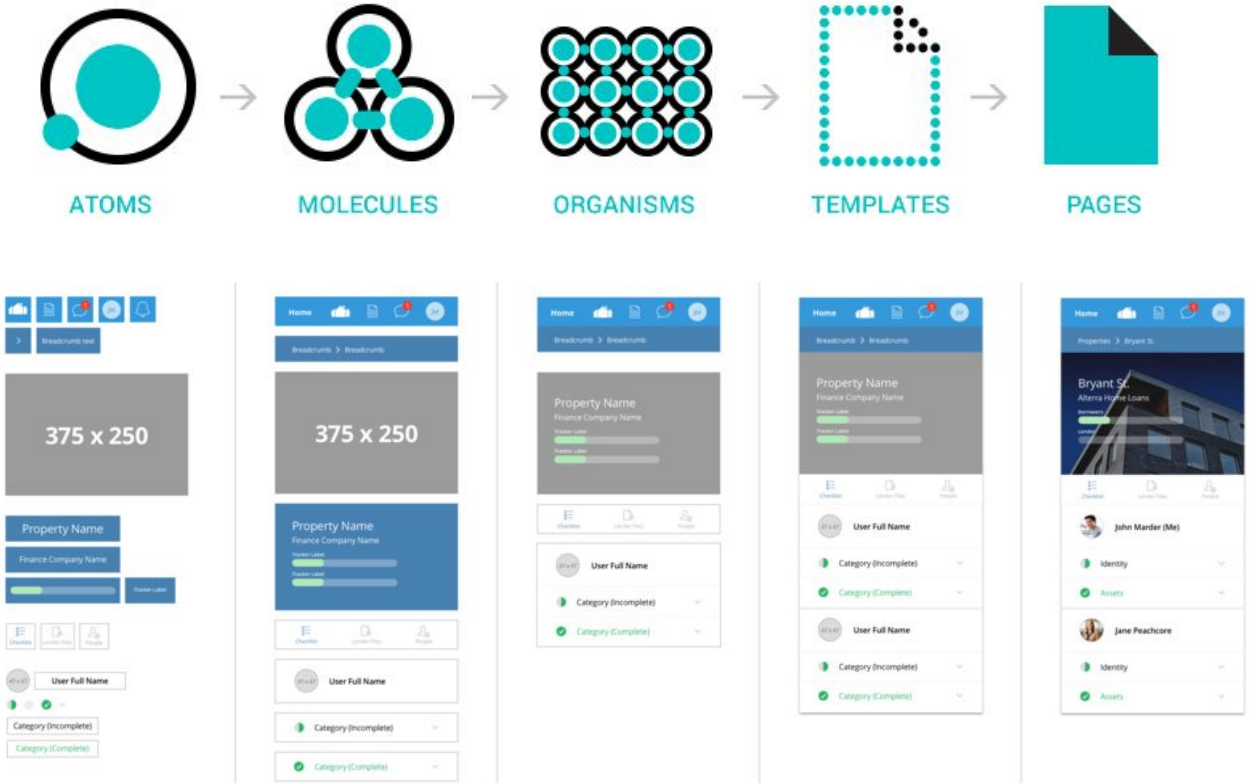
Must read:

atomic design by Brad Frost

https://atomicdesign.bradfrost.com/



atomic design


ATOMS
MOLECULES
ORGANISMS
TEMPLATES
PAGES
375 x 250
Property Name
Finance Company Name
User Full Name
Category (Incomplete)
Category (Complete)
Home
Property Name
Finance Company Name
User Full Name
Category (Incomplete)
Category (Complete)
Home
Property Name
Finance Company Name
User Full Name
Category (Incomplete)
Category (Complete)
Home
Property Name
Finance Company Name
User Full Name
Category (Incomplete)
Category (Complete)
User Full Name
Category (Incomplete)
Category (Complete)
Home
Bryant St.
John Marder (Me)
Identity
Assets
Jane Peachcore
Identity
Assets

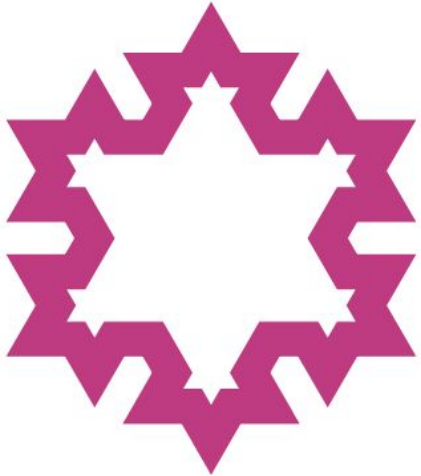
Fractal

Fractal is a tool to help you build and document web component libraries, and then integrate them into your projects.

Fractal

- Component libraries framework
- Style Guides platform
- Template & Data-driven with Handlebars
- Atom design as organisational model

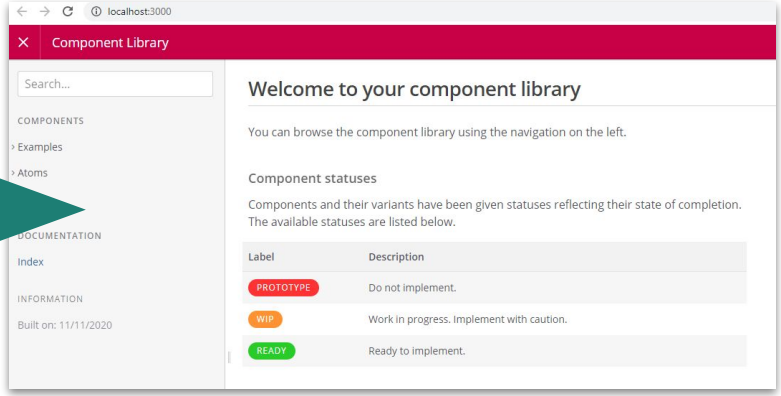
https://fractal.build/guide/



Run fractal

Run fractal as local server:
npm run fractal

```
C:\Projects\novicell-frontend [master ≡ +1 ~4 -0 !]> npm run fractal

> novicell-frontend@5.0.0 fractal C:\Projects\novicell-frontend
> fractal start --sync

Deprecated configuration file name! Rename fractal.js to fractal.config.js.

  Fractal web UI server is running!

  Local URL:       http://localhost:3000
  Network URL:     http://192.168.1.172:3000
  BrowserSync UI:  http://localhost:3002

  Use ^C to stop the server.
```



Fractal overview

Current component

Component organization

Component definition

Component Library

Search...

COMPONENTS

Examples

Atom Example

Molecule Example

Organism Example

Atoms

Docs

DOCUMENTATION

Index

INFORMATION

Built on: 11/11/2020

Organism Example

WIP

Molecule 1 heading

The text in the @molecule-example.

The text in the @atom-example.

Molecule 2 heading

The text in the @molecule-example.

The text in the @atom-example.

HTML | View | Context | Info | Notes

```
<div class="container">
    <div class="row">
        <div class="col-xs-12 col-sm-6">
            {{> '@molecule-example' moleculeData1 }}
        </div>
        <div class="col-xs-12 col-sm-6">
            {{> '@molecule-example' moleculeData2 }}
        </div>
    </div>
</div>
```

Fractal component

individual pieces of your website's UI

```
├── components
│   ├── components.config.json
│   ├── components.hbs (Handlebar)
│   ├── components.js
│   ├── README.md
│   └── components.css (postCSS)
```

```
{
    "title": "Atom Example",
    "status": "wip",
    "context": {
        "text": "This is some text for the ato
    }
}
```

```
<p class="atom-example">{{ text }}</p>
```

```
/*
 * Atom Example
 */

:root {
    --atom-example__text-color: var(--color-plum);
}

.atom-example {
    color: var(--atom-example__text-color);
}
```

https://fractal.build/guide/components/
https://github.com/Novicell/novicell-frontend/wiki/Fractal-guidelines

Fractal, component definition

HTML result

View code source

Context data

Component info

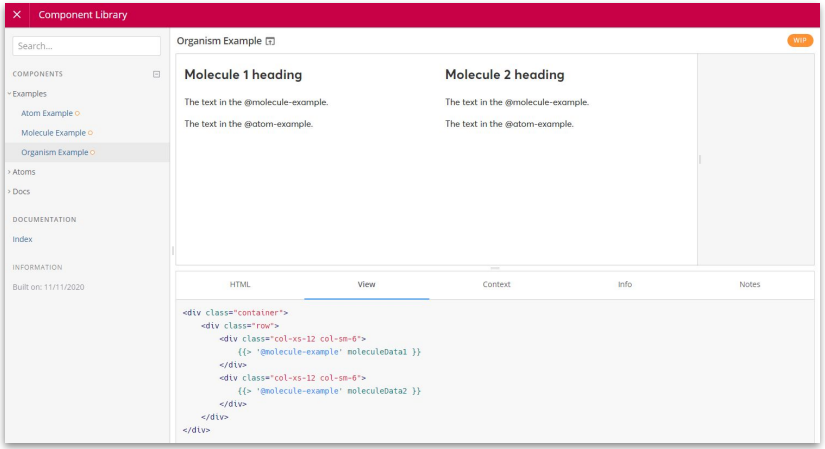
Component Notes

HTML | View | Context | Info | Notes

```
<div class="container">
    <div class="row">
        <div class="col-xs-12 col-sm-6">
            <div class="molecule-example">
                <h2 class="molecule-example__heading">Molecule 1 heading</h2>
                <div class="molecule-example__solid-pattern">
                    <p class="molecule-example__text">The text in the @molecule-example.</p>
                </div>
                <div class="molecule-example__dry-pattern">
                    <p class="atom-example">The text in the @atom-example.</p>
                </div>
            </div>
        </div>
```

Exercice 2 - 5 mins

1. Run fractal and check the fractal portal structure



Fractal component and Handlebar

Fractal, Handlebar

Handlebars is a simple templating language.

- Simple Data binding
- Nested input objects
- Evaluation context
- Partial
- …

```
{
    "title": "Molecule Example",
    "status": "wip",
    "context": {
        "heading": "Molecule example heading",
        "text": "Solid: We need a similar pattern, but want t
        "atomExample": {
            "text": "Dry: We can reuse @atom-example and don'
        }
    }
}
```

```
<div class="molecule-example">
    <h2 class="molecule-example__heading">{{ heading }}</h2>
    <div class="molecule-example__solid-pattern">
        <p class="molecule-example__text">{{ text }}</p>
    </div>
    <div class="molecule-example__dry-pattern">
        {{> '@atom-example' atomExample }}
    </div>
</div>
```

https://handlebarsjs.com/guide

Fractal, Handlebar

Helper.

- if
- unless
- each
- with
- ...

```
<ul class="people_list">
  {{#each people}}
    <li>{{this}}</li>
  {{/each}}
</ul>
```

```
{
  people: [
    "Yehuda Katz",
    "Alan Johnson",
    "Charles Jolley",
  ],
}
```

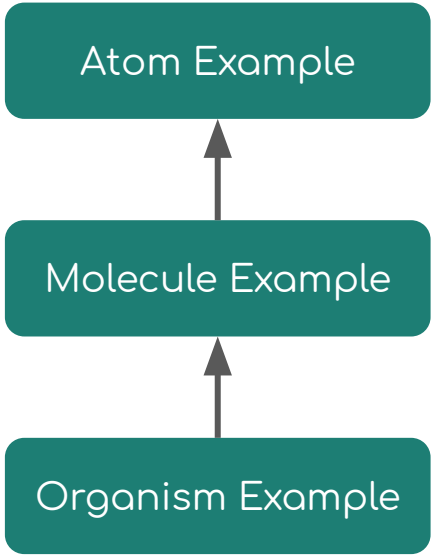
```
<ul class="people_list">
    <li>Yehuda Katz</li>
    <li>Alan Johnson</li>
    <li>Charles Jolley</li>
</ul>
```

https://handlebarsjs.com/guide

Fractal, Handlebar sub-components

- Including Sub-components
  {{> '@molecule-example' }}

- Sub-components with context data
  {{> '@molecule-example' moleculeData1 }}



https://fractal.build/guide/components/including-sub-components.html

Fractal, generate component

- createComponent -t p -n myComponent
  - a: atom
  - m: molecule
  - o: organisme
  - p: page

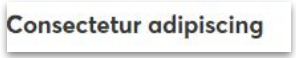
```
$ createComponent -t p -n myAwesomePage
Created folder in D:\webdev\novicell-frontend/src//04-pa
myAwesomePage.css was created
myAwesomePage.hbs was created
myAwesomePage.json was created
```

- Note: run “npm link” first

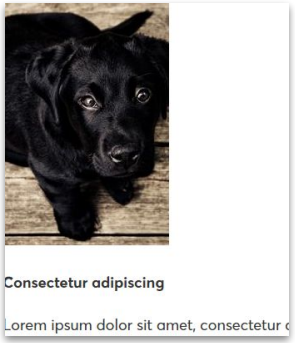
https://github.com/Novicell/novicell-frontend/wiki/Adding-new-files

Exercice 3 - 20 mins

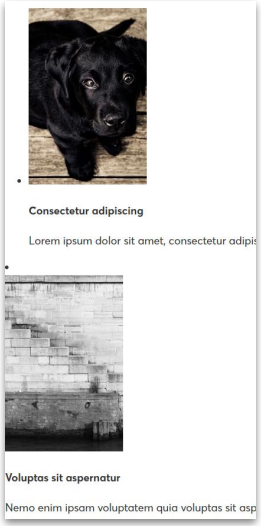
1. Create two atom components:
   a. “title” (as a <h2> element)
   b. “summary” (as a <p> element)
   c. for both component set a text as context



2. Create a molecule component “card”:
   a. With an image, a atom title, and atom molecule
   b. Set a context for the image source, title text and summary text



3. Create an organism component “last-post” which lists “card” components:
   a. Create a context object with 3 cards, use the #each instruction to list them



Exercice 3 - 20 mins - anexe

1. “title” html result
2. “summary” html result
3. “card” html result
4. “post-list” html result

```html
<h2 class="title">Consectetur adipiscing</h2>
```

```html
<p class="summary">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt</p>
```

```html
<div class="card">
  <img src="https://picsum.photos/300/175?random=1" />
  <div class="card__content">
    <h2 class="title">Consectetur adipiscing</h2>
    <p class="summary">Lorem ipsum dolor sit amet,  consectetur adipiscing elit, sed do eiusmod tempor incididunt</p>
  </div>
</div>
```

```html
<div class="last-post">
  <ul class="last-post__list">
    <li class="last-post__list__item">
      <div class="card">
        <img src="https://picsum.photos/300/175?random=1" />
        <div class="card__content">
          <h2 class="title">Consectetur adipiscing</h2>
          <p class="summary">Lorem ipsum dolor sit amet...</p>
        </div>
      </div>
    </li>
    ...
  </ul>
</div>
```

Styles in Fractal

Fractal, Add CSS files

- Generated with createComponent
- Must be imported into master.css or into a master css file into the src/Modules directory

```
@import '../04-pages/myAwesomePage/myAwesomePage.css';
```

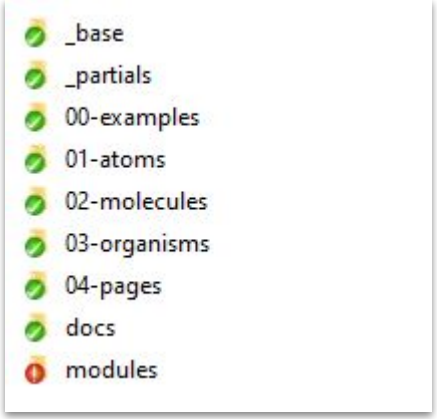
src/04-pages/myAwesomePage/myAwesomePage.css

```
.box {
    margin: 50px auto;
    width: 100px;
    height: 100px;
    background-color: blue;
}
```

https://github.com/Novicell/novicell-frontend/wiki/Adding-new-files#add-css-files

Fractal, folder structure

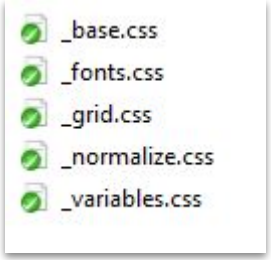
- _base
- _partials
- 01-atoms
  - component-a
  - component-b
- 02-molecules
- 03-organisms
- 04-pages
- docs
- modules



https://github.com/Novicell/novicell-frontend/wiki/Fractal-guidelines#folder-structure

Fractal, base

- _base
  - _base.css
  - _fonts.css
  - _grid.css
  - _normalize.css
  - _variables.css
- _partials
- ...



https://github.com/Novicell/novicell-frontend/wiki/Fractal-guidelines#folder-structure

Fractal, npm commands

- npm run ...
  - build:dev
  - build:prod
  - fractal
  - fractal:build
  - styles
  - watch:styles
  - ...

```
C:\Projects\novicell-frontend [master ≡ +1 ~4 -0 !]> npm run watch:styles

> novicell-frontend@5.0.0 watch:styles C:\Projects\novicell-frontend
> cross-env-shell postcss $npm_package_config_CSS_MODULES --dir $npm_package_config_DIST/css --ext min.css
 -config config/styles/postcss.config.js --watch --verbose
--config config/styles/postcss.config.js --watch --verbose

> cross-env-shell postcss $npm_package_config_CSS_MODULES --dir $npm_package_config_DIST/css --ext min.css
 --config config/styles/postcss.config.js --watch --verbose

Processing src\modules\master.css...
```

https://github.com/Novicell/novicell-frontend#npm-scripts

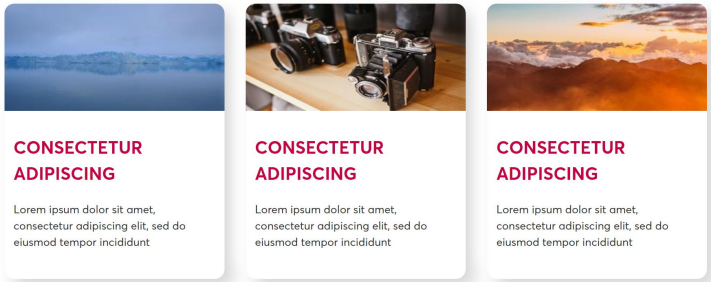
Exercice 4 - 15 mins

1. Set style for the title atom
2. Set style for the card molecule
3. Set style for the last-post module
4. Check the result for the 3 modules

```
.title {
    font-size : 1.6em;
    text-transform : uppercase ;
    color : var (--color-primary ) ;
}
```

```
.card {
    height :25em;
    width :20em;
    overflow :hidden ;
    border-radius :16px;
    position :relative ;
    display :-webkit-box;
    display :-ms-flexbox;
    display :flex;
    -webkit-box-orient :vertical ;
    -webkit-box-direction :normal ;
    -ms-flex-direction :column ;
    flex-direction :column;
    -webkit-box-shadow :  15px 15px
27px var (--color-gray-light ), -15px
-15px 27px var (--color-white ) ;
    box-shadow :  15px 15px 27px
var (--color-gray-light ), -15px -15px
27px var (--color-white ) ;
}
.card__content {
    padding : 15px;
}
```

```
.last-post ul {
    list-style : none;
    display : inline-block ;
}

.last-post ul li {
    display : inline-block ;
    margin : 0 15px
}
```



PostCSS & BEM

PostCSS aims to reinvent CSS with an ecosystem of custom plugins and tools

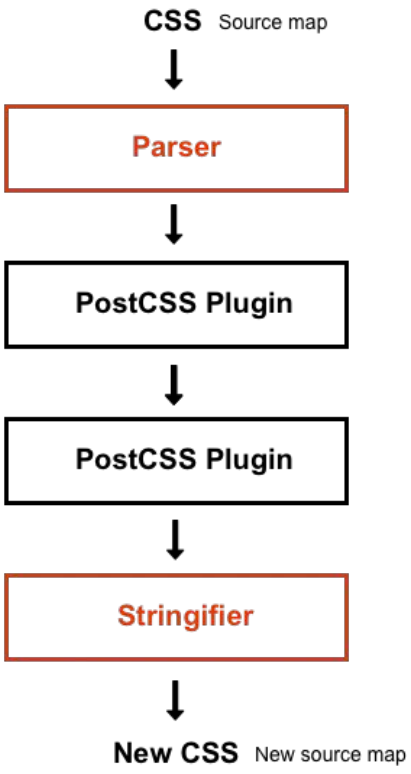
PostCSS

- PostCSS is a tool for transforming styles with JS plugins
- Working with the same principles of preprocessors such as Sass and LESS
- Ecosystem of custom plugins and tools

https://github.com/postcss/postcss

PostCSS

- Parses CSS into an abstract syntax tree (AST)
- passes that AST through any number of "plugin" functions
- passes that AST through any number of "plugin" functions



https://github.com/postcss/postcss

PostCSS - Plugin

- postcss-import
  transform @import rules by inlining content

- postcss-preset-env
  convert modern CSS into something most browsers can understand.
  This this the legacy cssnext, see feature here

- postcss-nested
  unwrap nested rules like how Sass does it

- cssnano
  optimise css code

- postcss-reporter
  Report postcss plugin operations

https://github.com/postcss/postcss

PostCSS - postcss-preset-env Plugin

.sass, .scss, .styl or .less ? nop, just traditional CSS...

postcss allow to pick up and configure how we really want to work

```
.addresses {
    &__title {
        margin-bottom: 20px;

        @media (--viewport-ms-min) {
            margin-bottom: 30px;
        }

        @media (--viewport-sm-min) {
            margin-bottom: 40px;
        }
    }

    &__offices {
        @media (--viewport-sm-min) {
            margin-bottom: -40px;
        }

        .col {
            &:last-child {
                .addresses__office {
                    @media (--viewport-ms-max) {
                        margin-bottom: 0;
                    }
                }
            }
        }
    }
```

https://github.com/postcss/postcss

PostCSS - Plugin

- postcss-import
  transform @import rules by inlining content

- postcss-preset-env
  convert modern CSS into something most browsers can understand.
  This this the legacy cssnext, see feature here

- postcss-nested
  unwrap nested rules like how Sass does it

- cssnano
  optimise css code

- postcss-reporter
  Report postcss plugin operations

https://github.com/postcss/postcss

BEM: Block Element Modifier

Methodology to create reusable and clear components in front-end development

- Easy
- Modular
- Flexible

http://getbem.com/

BEM: Block Element Modifier

Main naming concepts

- **Block**, Encapsulates a standalone entity that is meaningful on its own.

```
<div class="block">...</div>
```

```
.block__elem { color: #042; }
```

- **Element**, Parts of a block and have no standalone meaning.

```
<div class="block">
        ...
        <span class="block__elem"></span>
      </div>
```

```
.block__elem { color: #042; }
```

- **Modifier**, Flags on blocks or elements. Use them to change appearance, behavior or state.

```
<div class="block block--mod">...</div>
      <div class="block block--size-big
              block--shadow-yes">...</div>
```

```
.block--hidden { }
```

```
.block__elem--mod { }
```

Result of all of this

BEM

postcss-nested

postcss-preset-env

```
.addresses {
    &__title {
        margin-bottom: 20px;

        @media (--viewport-ms-min) {
            margin-bottom: 30px;
        }

        @media (--viewport-sm-min) {
            margin-bottom: 40px;
        }
    }

    &__offices {
        @media (--viewport-sm-min) {
            margin-bottom: -40px;
        }

        .col {
            &:last-child {
                .addresses__office {
                    @media (--viewport-ms-max) {
                        margin-bottom: 0;
                    }
                }
            }
        }
    }
```

Exercice 5 - 15 mins

1. Refactor card.css and last-post.css following the BEM rules and using nested style definition.

2. In card.css, create an alternative card version “highlight”, and apply this style to the first last-post component’s card.
   Trick: use handlebar “if” option to apply the highlight alternative style to the first card

More style trick and good practices

Grid system

Novicell Frontend framework comes with a out of the box grid system

- 12 columns
- 5 breaking point: lg, md, sm, ms, xs
- Configurable
- \src_base_grid.css

```
<div class="container">
    <div class="row">
        <div class="col-xs-12 col-ms-6 col-sm-4 col-lg-4">
            <div class="row">
                <div class="col-xs-3 col-ms-3 col-lg-3">
                    <div class="grid-example-col-content">
                        <p>1st column</p>
                    </div>
                </div>
                <div class="col-xs-3 col-ms-3 col-lg-3">
                    <div class="grid-example-col-content">
                        <p>2nd column</p>
                    </div>
                </div>
```

1st column	2nd column	3rd column	4th column	5th column	6th column	7th column	8th column	9th column	10th column	11th column	12th column

CSS style guideline

Must read:

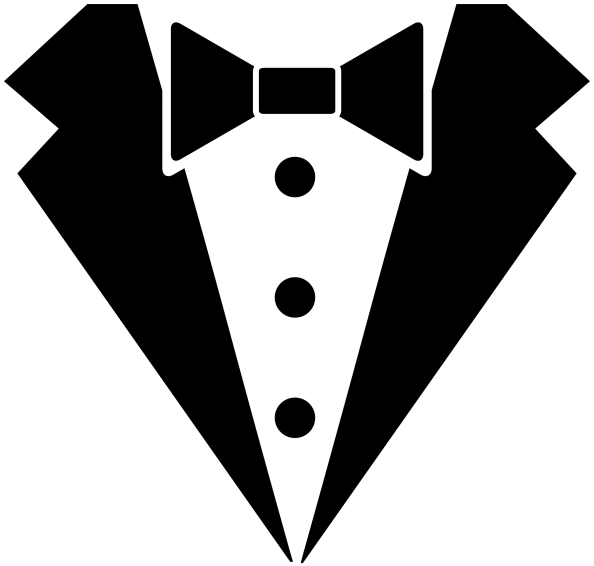
https://github.com/Novicell/novicell-frontend/wiki/CSS-(PostCSS)#Comments

stylelint.io extension

A mighty, modern linter that helps you avoid errors and enforce conventions in your styles.

https://stylelint.io/

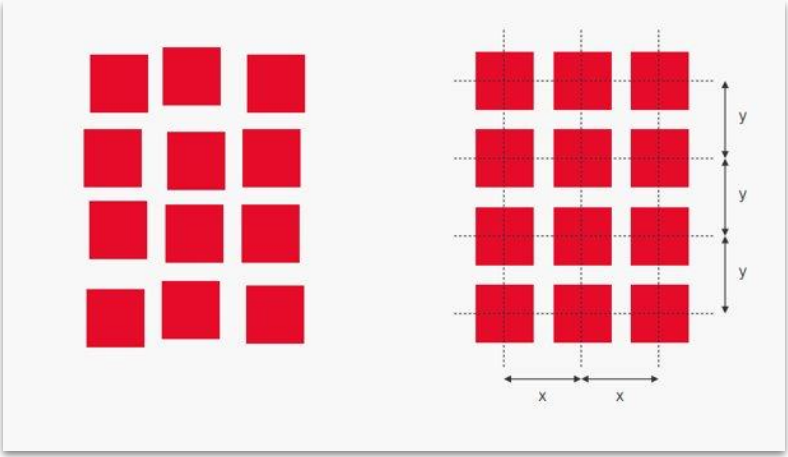
Check config in config/styles



Be pixel perfect

- A pixel-level design affinity in the encoding of HTML files and CSS styles is essential for pages to convey the stability, emotions, and concepts brought in during the design phase.

- To ensure an optimal level of quality, we can perform automated visual comparison tests with ghostinspector.com to detect any unexpected changes in graphical level.



Think mobile “first”

- Ensures an optimal user experience on mobile devices
- Favors, due to the nature of small devices, graphic creativity
- Optimize development as it is easier to scale to higher resolutions
- Ensures that content and interactions are reproduced optimally on any device
- Optimize page loading speed.

Scripts in Fractal

Add JS files

- Should be always added into the src/Modules/ directory
- Create a new folder for each component to maintain a good separate the concerns
- Add JS reference into the .hbs file

```
<div>
    <img class="lazyload lazyload-measure" data-src="https://source.unsplash.com/2Ts5HnA67k8/"/>
</div>
<script defer src="{{ path '/dist/scripts/myAwesomePage.bundle.js' }}"></script>
```

https://github.com/Novicell/novicell-frontend/wiki/Adding-new-files#add-js

ESLint extension

Find and fix problems in your JavaScript code

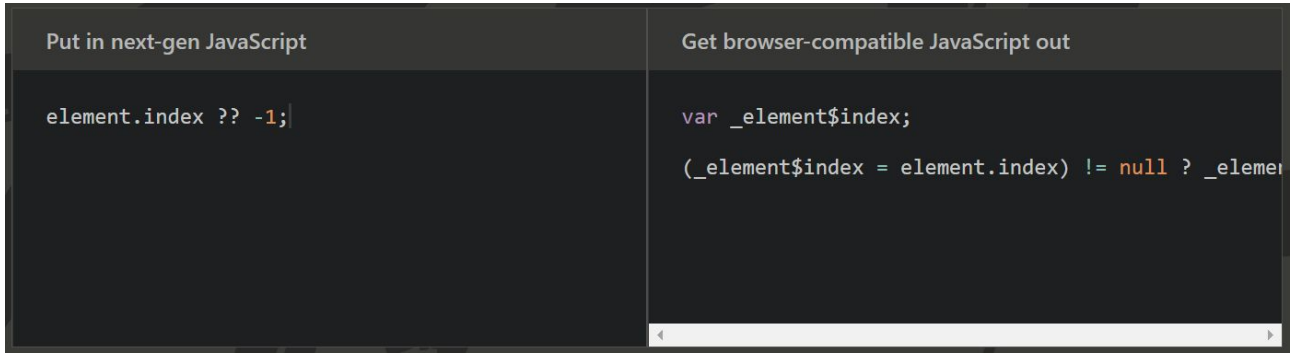
https://eslint.org/

Check config in config/script

Babel extension

JavaScript compiler, Use next generation JavaScript, today

https://babeljs.io/

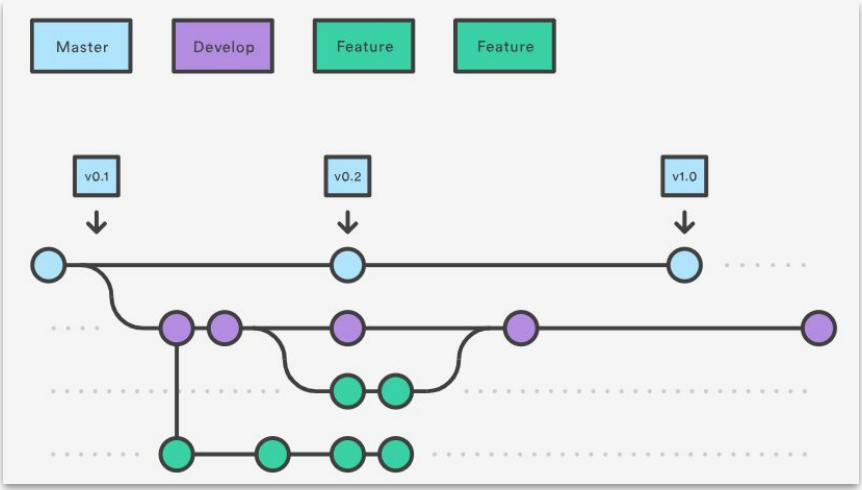


Development process and deployment

GitFlow

Standard branching model strategy

- Always create a new feature branch from develop for each User Stories
- Create sub-branches from features branches for each task and commit your changes there
- Merge you feature branches in develop by pool request
- Merge develop branch into master (or create release branches), tags each releases in Master

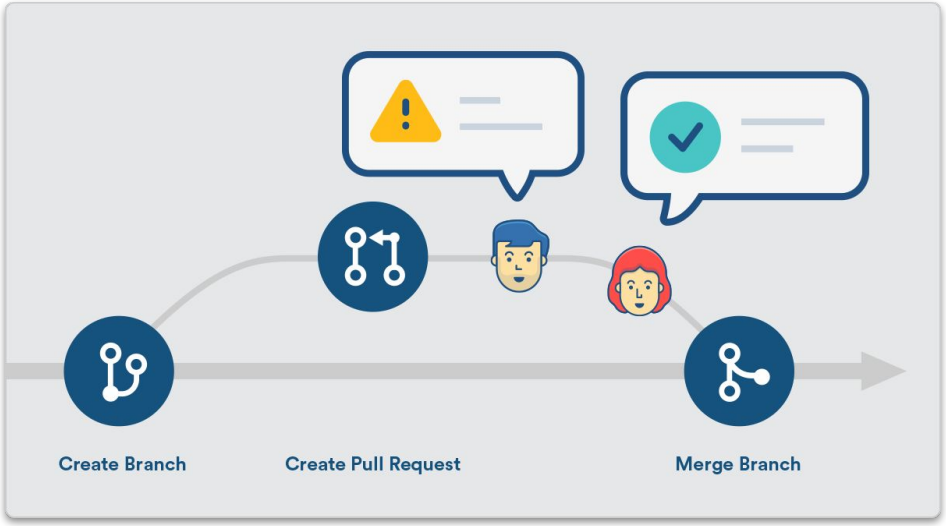


https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
https://dev.azure.com/Uci-EcowebDevops/ecoweb-app/_wiki/wikis/ecoweb-app.wiki/16/Project-Development-Process

Merging by PR process

PR: Pool request

- Testing and better stability
- Clearer responsibility
- Sharing knowledge
- Avoid conflict
- Meaningful git history



https://docs.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops

DevOps

Delivery application and services

- Project management
- Code repository
- Building pipeline
- Deploy process

https://dev.azure.com/Uci-EcowebDevops/ecoweb-app/

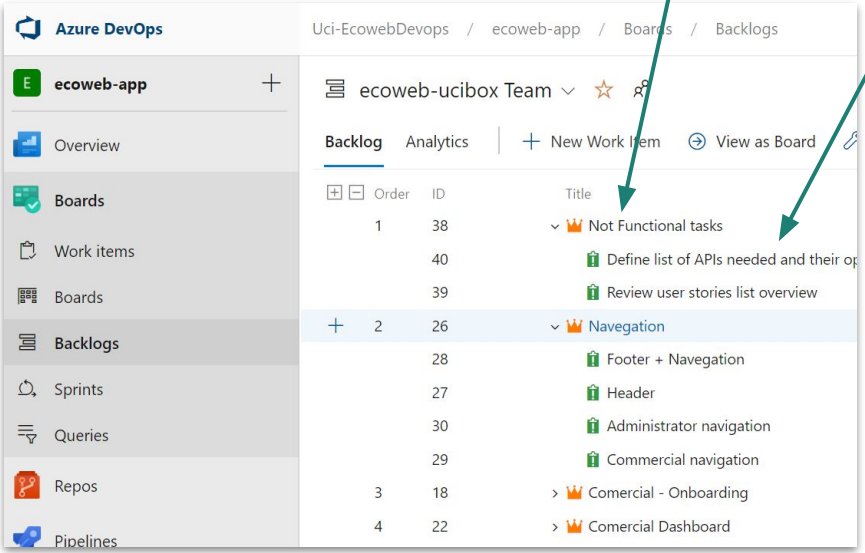
DevOps, Project management

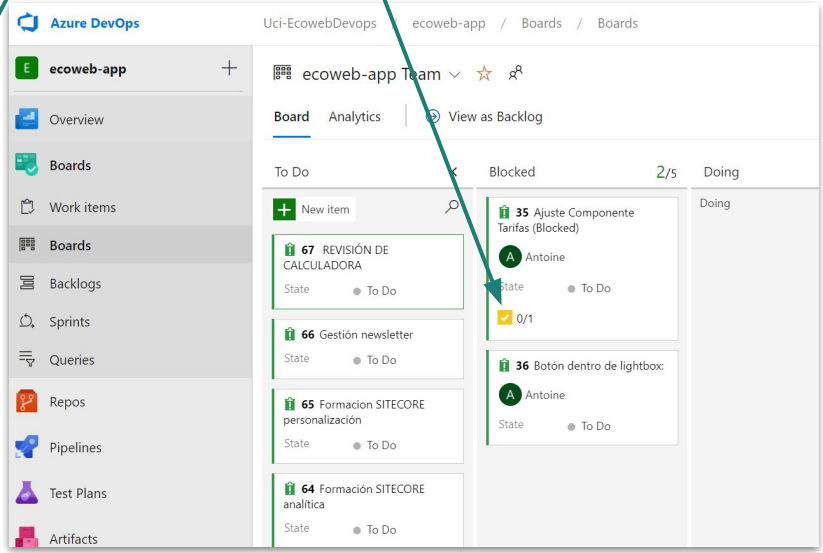
100% scrum focus

Epic

User Stories

User Stories





DevOps, Project management

100% scrum focus

Ready to be developed, could be picked by the team

I am blocked, I must mencione it on the next daily meeting

I am working on the user story, always only one in this column

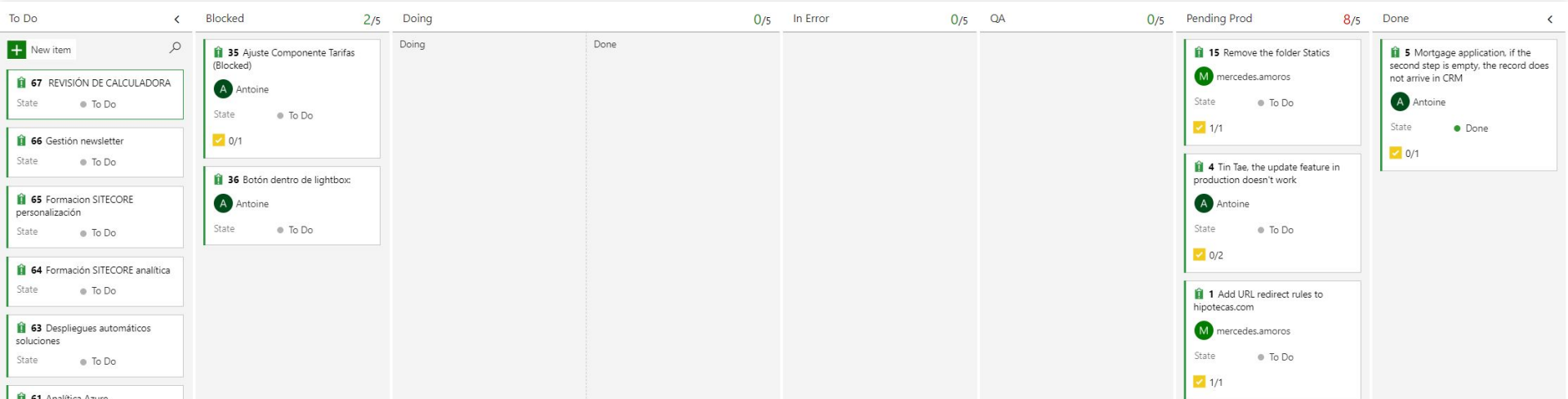
I finished the user story, it is pending to be deployed in QA

The US does not feat the acceptance criteria

The US is deployed in QA and pending of the QA team review

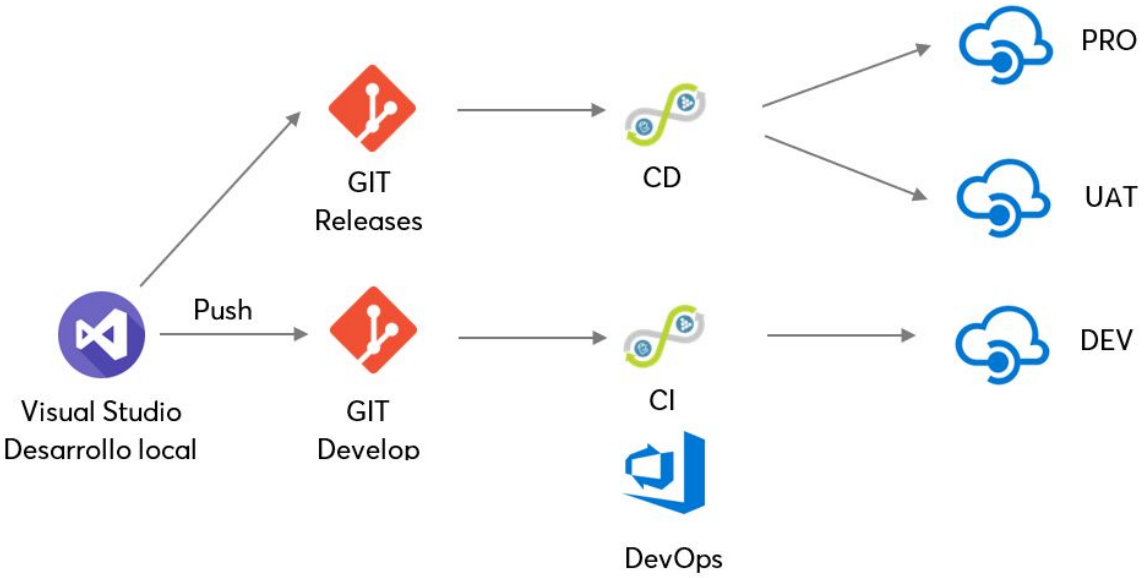
The US is approved by QA and pending to be deploy in PROD

US close and deployed in PROD



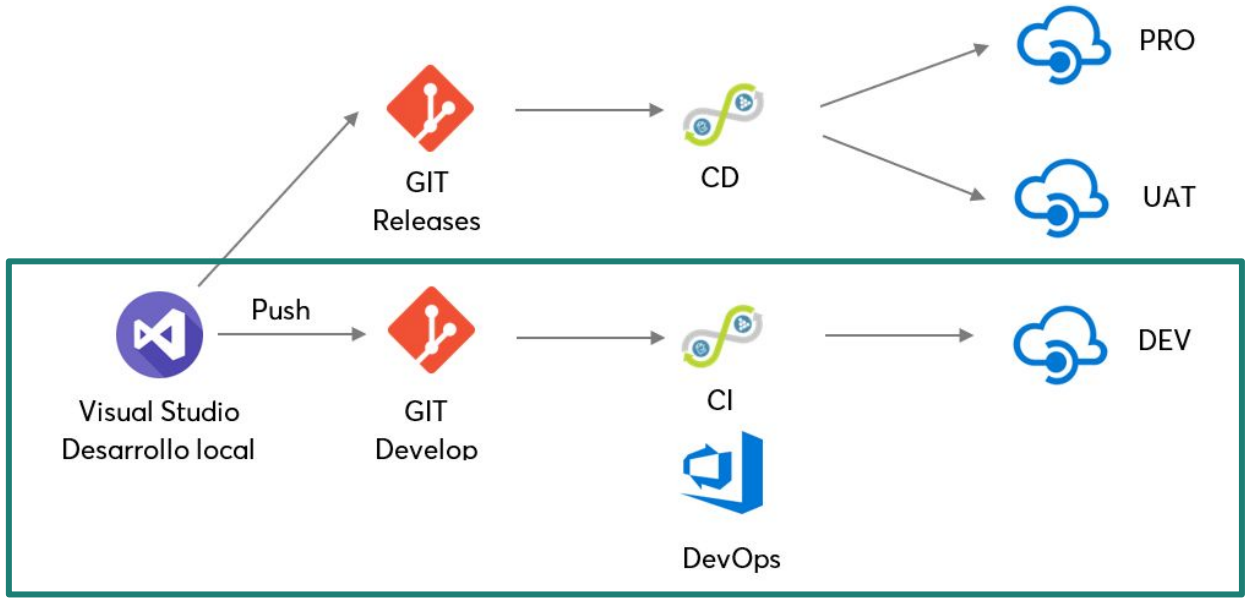
DevOps, CI / CD

Continuous Integration and Delivery Pipeline

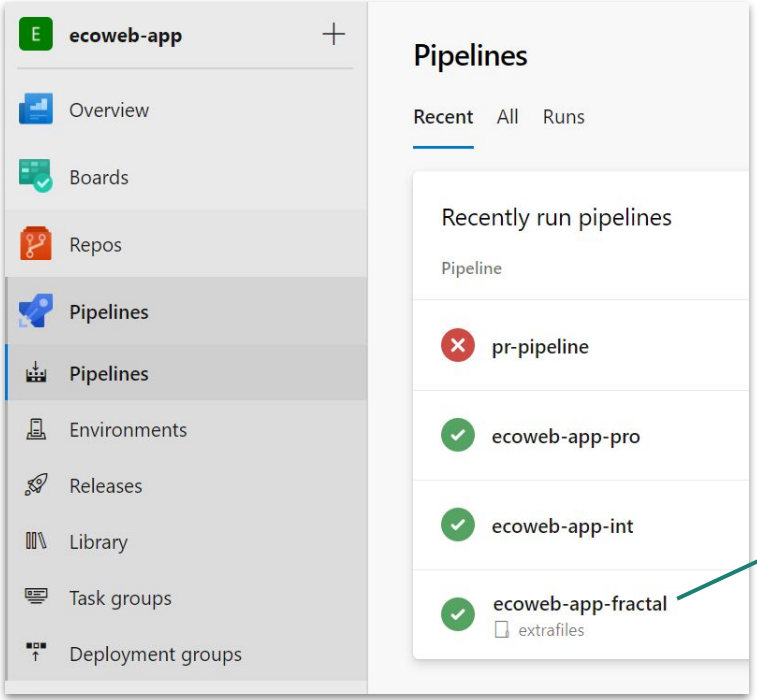


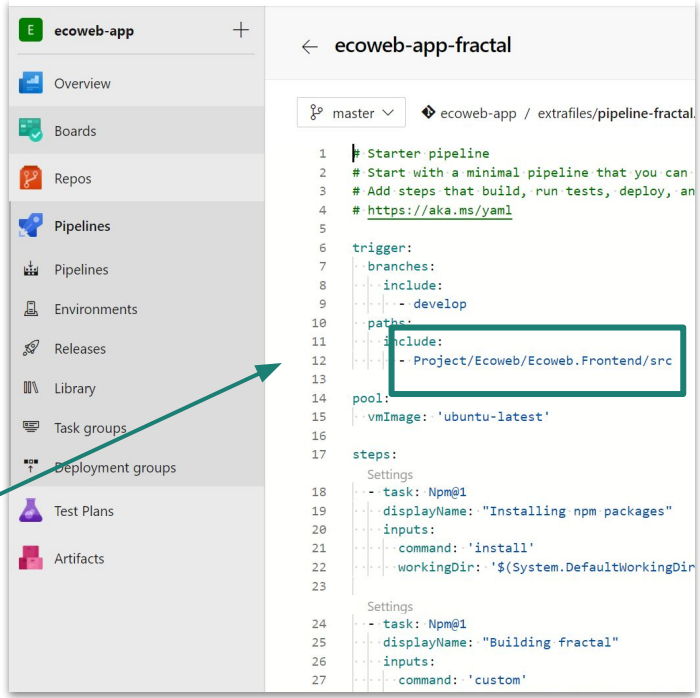
DevOps, CI / CD

Continuous Integration and Delivery Pipeline



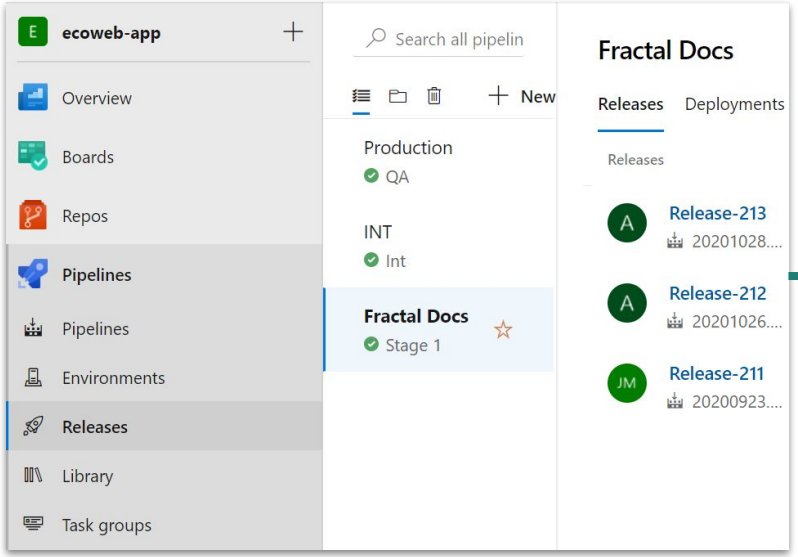
DevOps, Fractal build pipeline

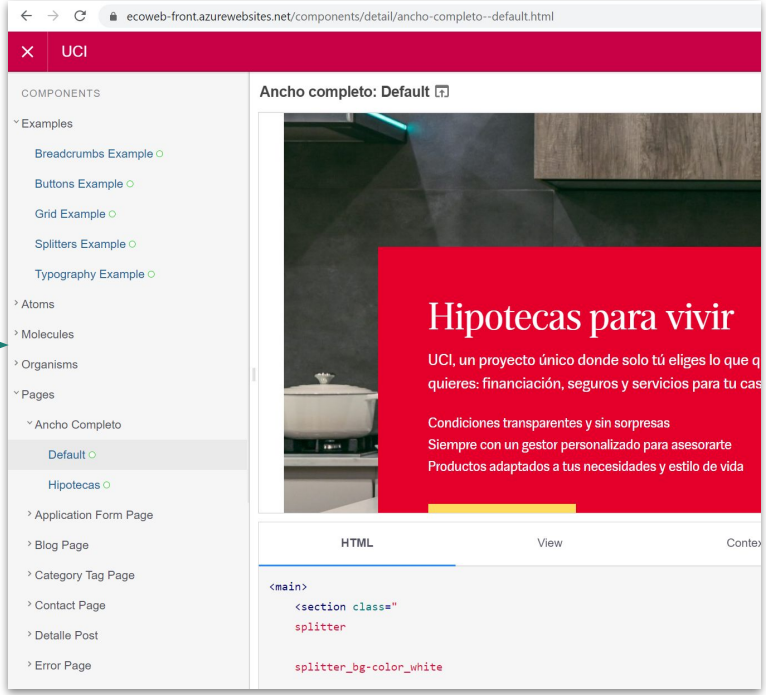




DevOps, Fractal build pipeline

https://ecoweb-front.azurewebsites.net/





Integration with Sitecore

Sitecore vs Front-end

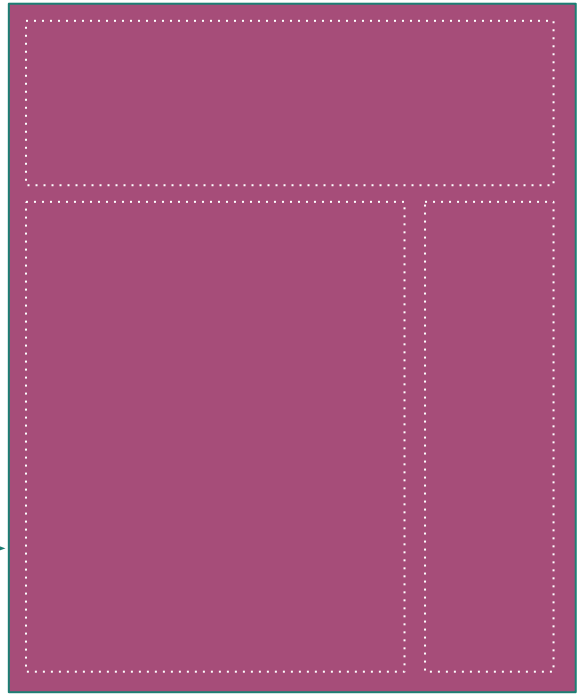
Different ways to work with Sitecore

- Standard
- SXA
- Headless

- Separation of concept front / back
- No technological dependencies
- No structural dependencies
- Front-end integration step

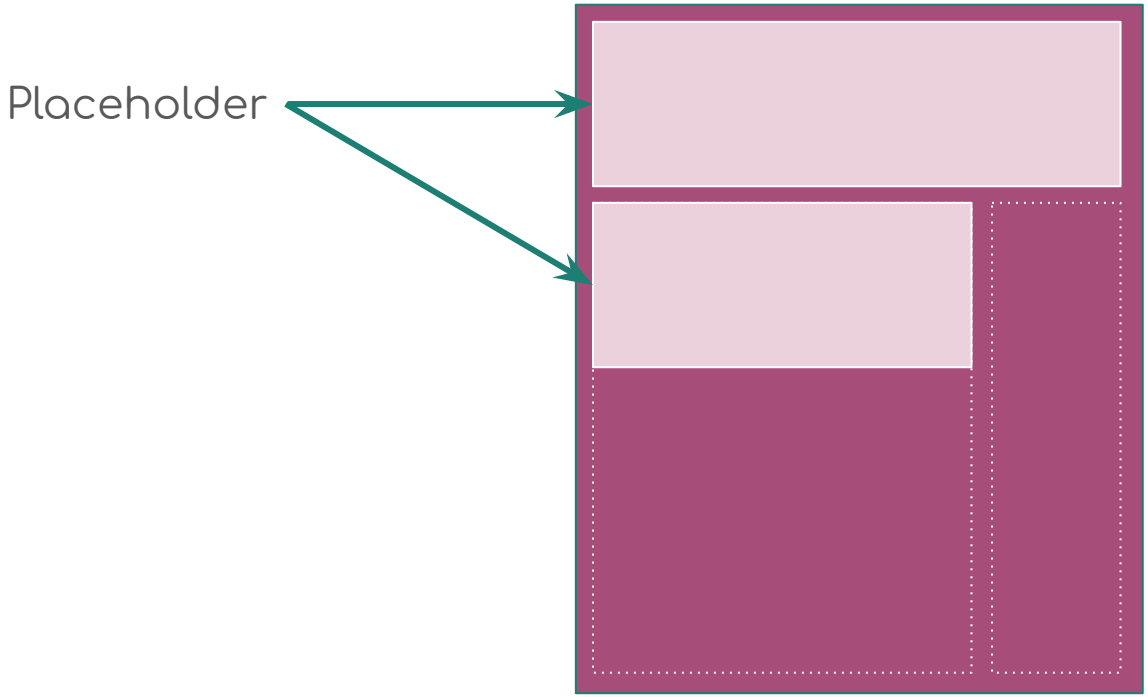
Sitecore rendering and placeholder

Page layout



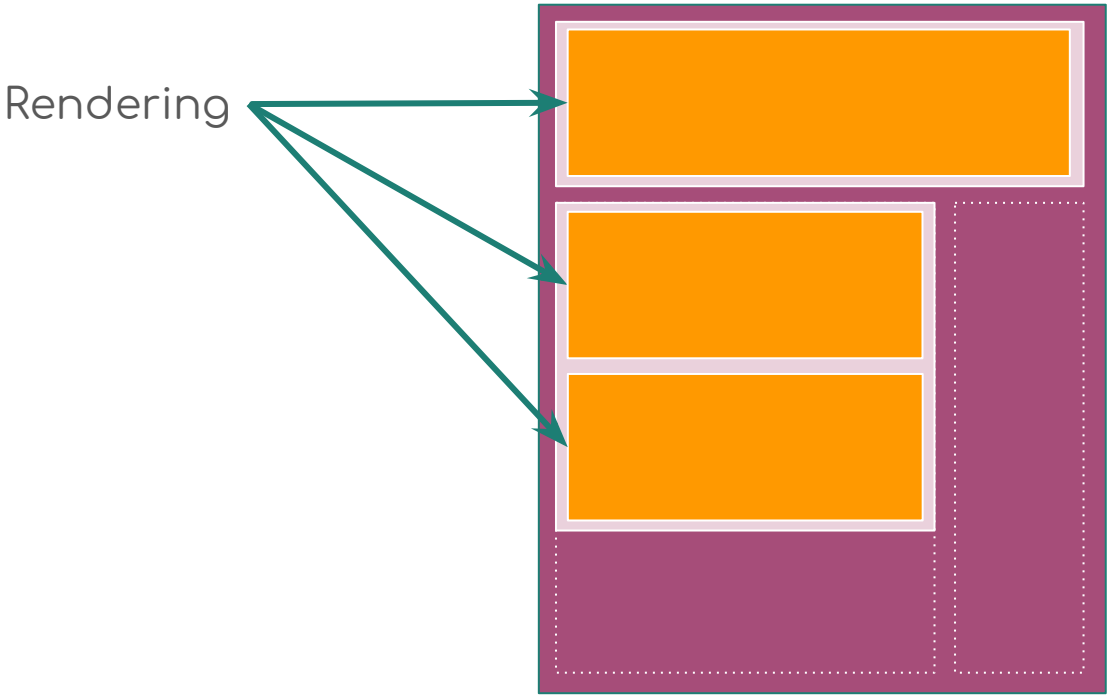
Sitecore layout defines the page structure with the outer HTML mark-up

Sitecore rendering and placeholder



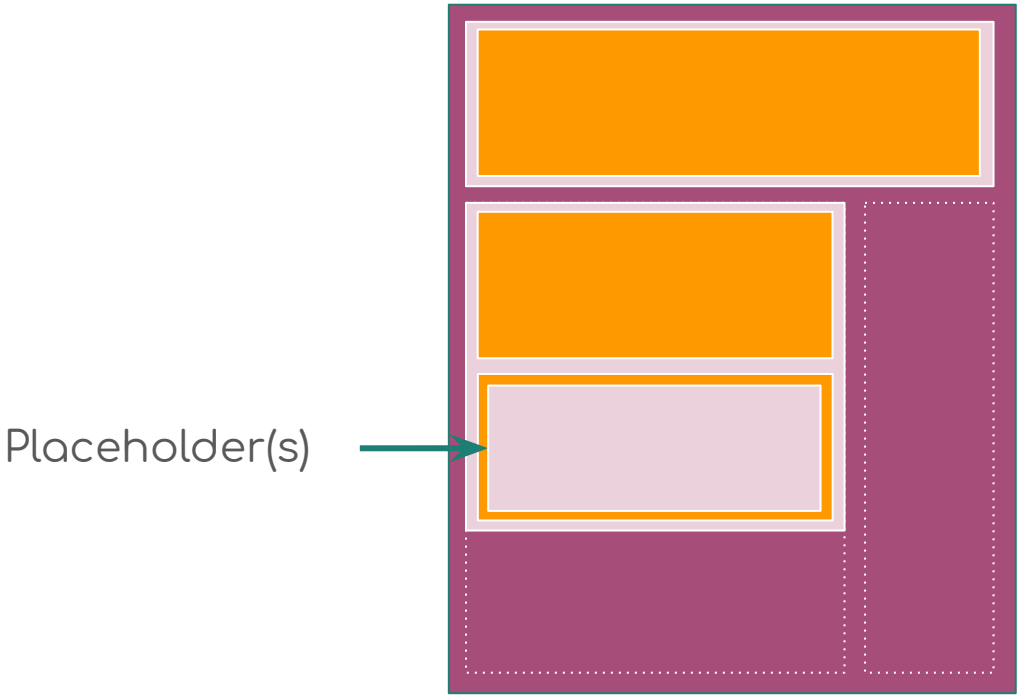
Sitecore placeholder is a place where a rendering can be bound

Sitecore rendering and placeholder



Renderings represent a specific functional block, it's a component

Sitecore rendering and placeholder



A renderings can have endebbed placeholders

Sitecore vs atomic design

Rendering (component)

Layout

Page

So, what is the integration process?

The integration process consites on integrate the frontend code generated in fractal, into the Sitecore MVC razor files.

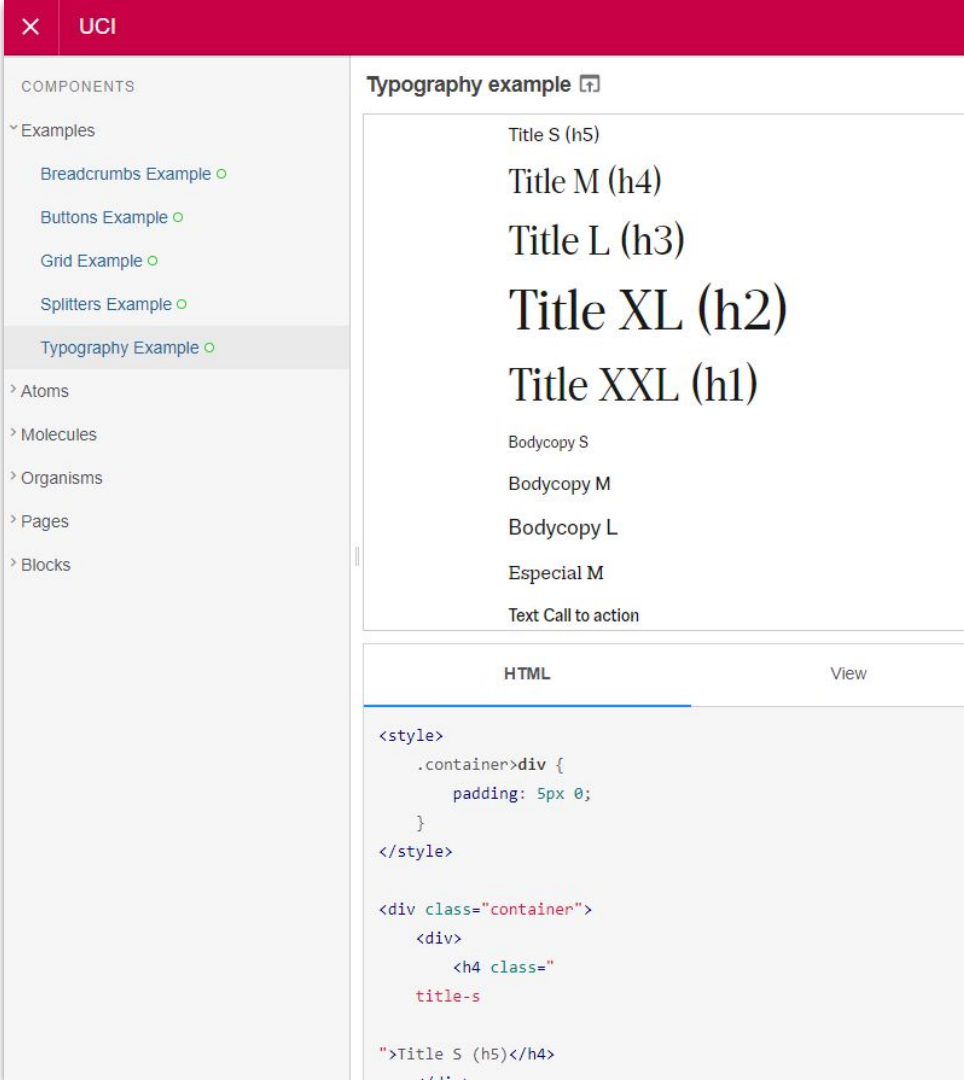
For each user stories, the process will be:

- Make a agreement on what kind of elements will be built (layout, component, etc.), their naming (default layout, last post component, etc.), and JS object format if needed.
- Create frontend sub-tasks to carry out the front-end development: html, css and javascript.
- Create a back-end sub-task to integration the frontend code into the MVC razor files. This task can't be executed before the front-end task.
- The HTML is integrated manually, the CSS, Javascript and other assets are compiled and copied automatically (npm script run during the build pipeline)
- If the userstory have both front-end and back-end subtask, the user story must assigned to the backend developer
- The user story will be close once the front-end integrated and the user story reviewed by QA in Sirecore. Once closed, any changes or correction will generate a new user story.

ECOWEB case overview

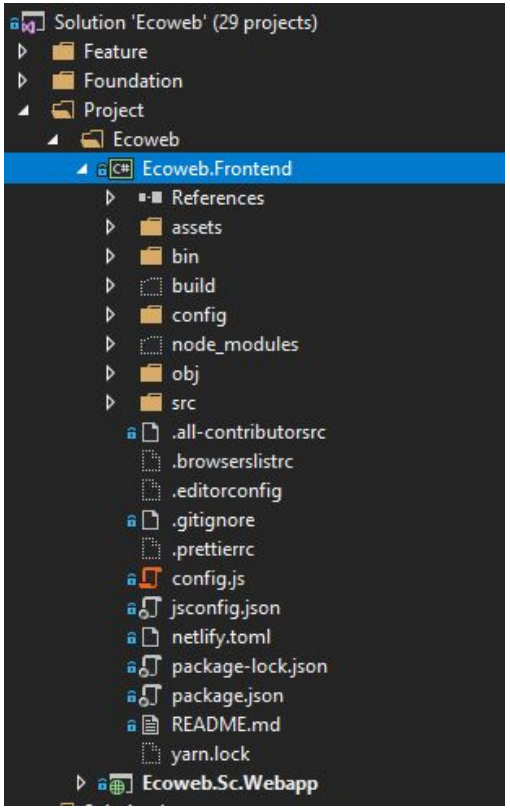
UCI fractal

- Examples
  - Buttons, Text, Grid, etc.
- Atomic structure
- No templates are used
- Example of pages
- Block section are splitter use cases

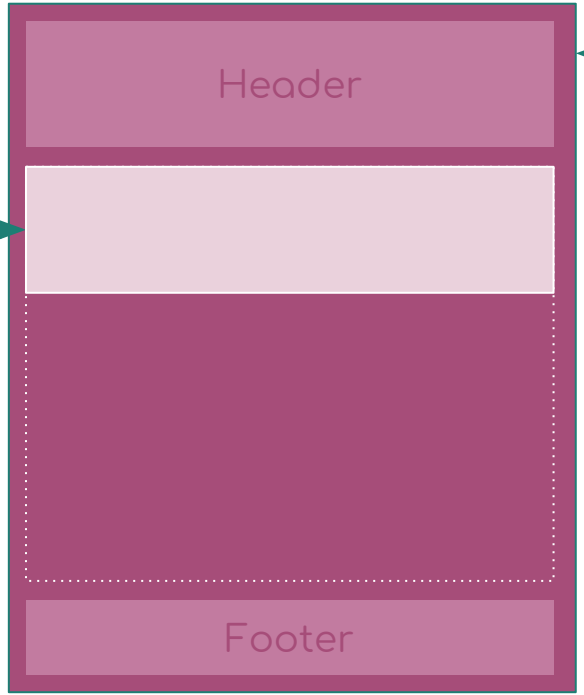


File structure

- Same code repository
- Integrated into the VS solution
- npm build script copy the dist folder into the Sitecore ECOWEB project



Sitecore UCI page structure

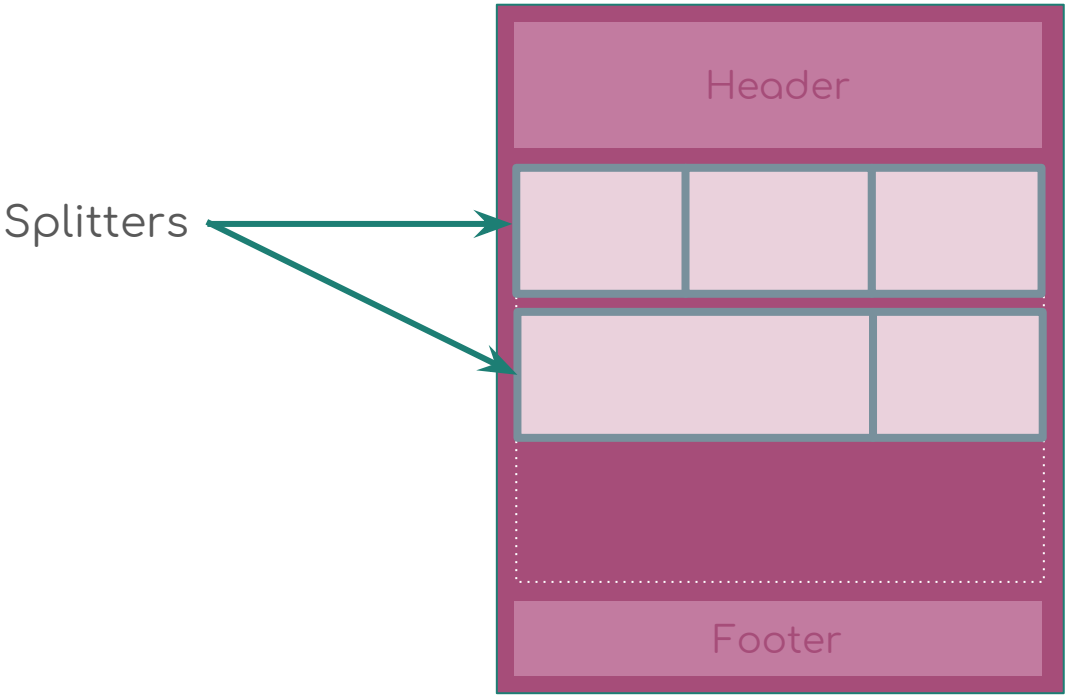


Placeholder

Layout

- Highlight Layout
- Mortgage Layout
- Reduce Layout
- etc.

Sitecore UCI page structure

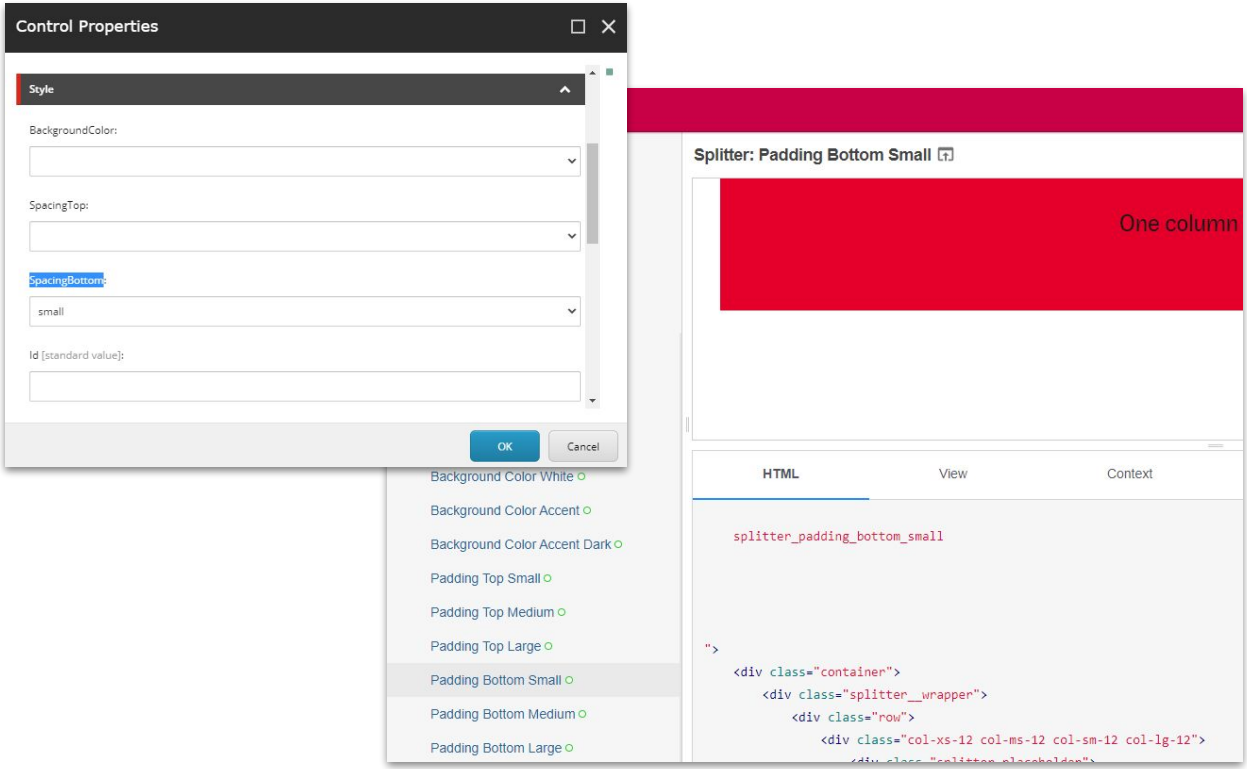


A splitter is a configurable rendering used to define page body structure on the fly

Sitecore UCI page structure

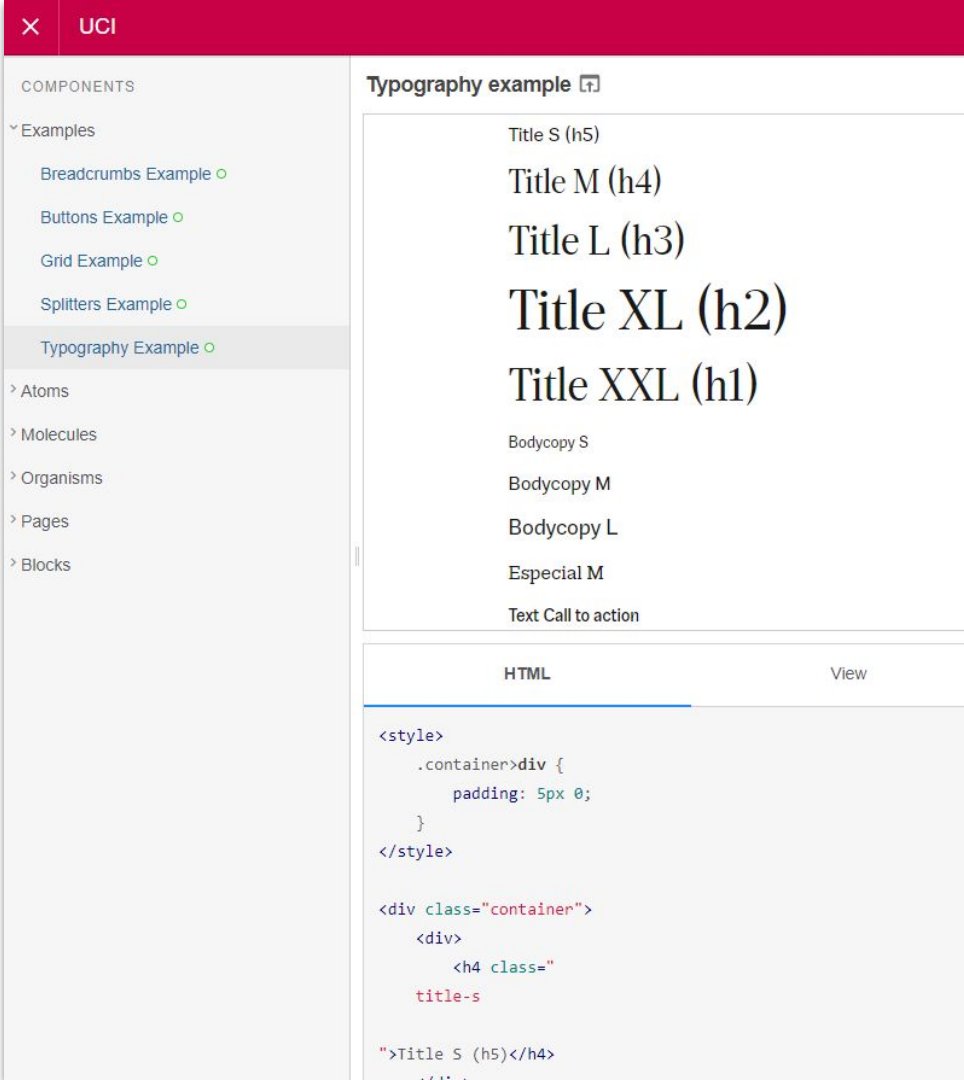
Splitter are configurable:

- BackgroundColor
- SpacingTop
- SpacingBottom
- ...



Not so good

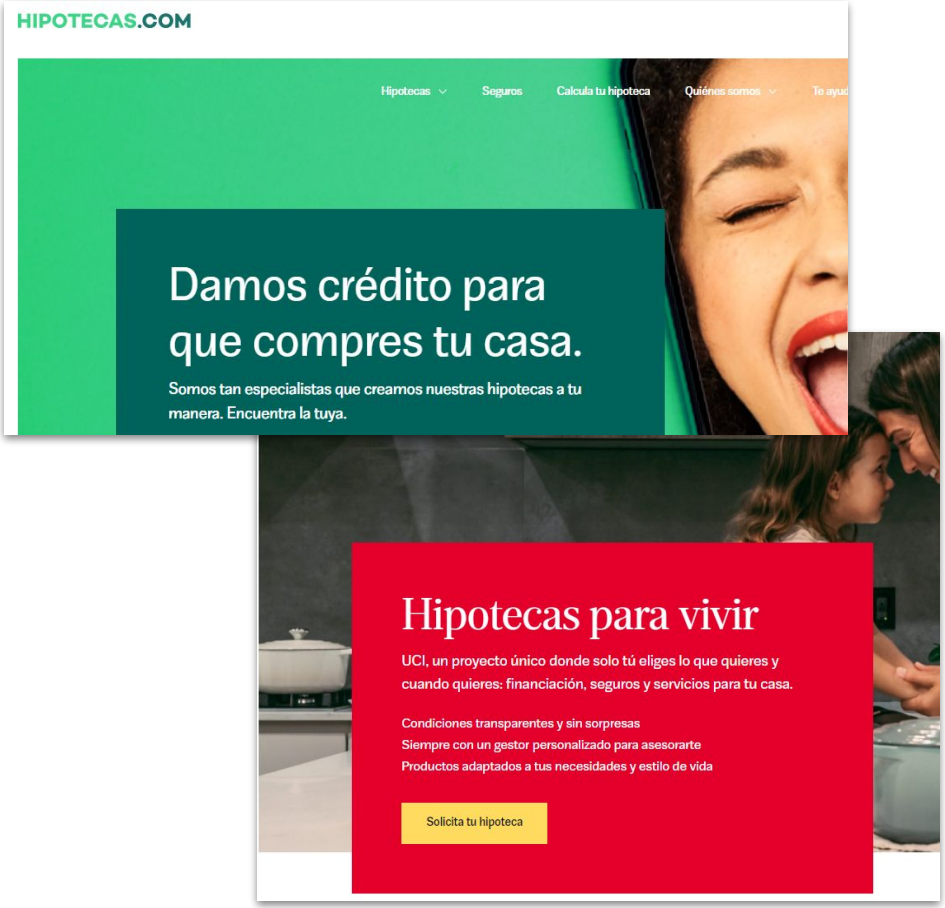
Why don’t we have template (layout) folder ?



UCI vs Hipotecas

Specific variable and template configuracion for each project:

https://novicell.atlassian.net/wiki/spaces/UCIPRO/pages/1730970042/Frontside+Development

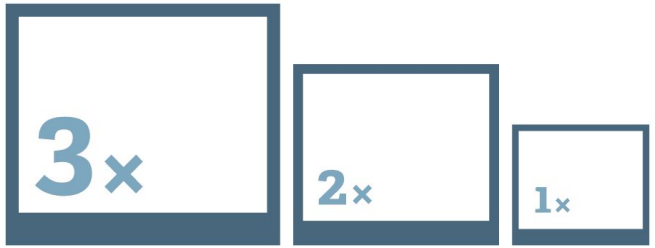


Real responsive images and lazy load

Addoc development for Sitecore real responsive images:

https://novicell.atlassian.net/wiki/spaces/UCIPRO/pages/1745158586/Lazy+loading+and+responsive+images

Sitecore extension:
\Foundation\Ecoweb.Sc.Foundation.Helpers\code




```
namespace Ecoweb.Sc.Foundation.Helpers
{
    public static class HtmlMediaHelpers
    {
#pragma warning disable CA1717 // Only FlagsAttribute enums should have plural names
        public enum Resolutions...

        public static IHtmlString BackgroundImage(this HtmlHelper _, string url, Dictiona

        public static IHtmlString ResponsiveImage(this HtmlHelper _, string url, Dictiona
    }
}
```

```
<div
    class="access-highlight lazyload"
    data-bg="/dist/images/access-highlight-bg-mob.jpg 767px,
        /dist/images/access-highlight-bg-tab.jpg 1023px,
        /dist/images/access-highlight-bg-lap.jpg 1279px,
        /dist/images/access-highlight-bg.jpg"
>
```

QA

antu@ngiro.net

Merci

antu@ngiro.net