##### Home page / (original page > <http://davidibanez.com/> )

## **4 year's experience in web design / front-end development**

##### I love to design and build clean and elegant websites. You can check out my design workflow, see how I build responsive website using Susy for Libsass, my latest AngularJS project or know more about my modular approach to styling using SASS.

##### 01 / (original page > [http://davidibanez.com/works.php?link=lsp](http://davidibanez.com/works.php?link=lsa-portail))

An AngularJS Web application

**Web Application Design and Development**

Labosalem production is an online web application for pharmaceutical wholesalers that helps them buy product and manage their orders. I designed and built the application using AngularJS and the MEAN stack.

## **From Website to Web Application**

Laboratoires Salem Production is a website that allows registered wholesalers to buy products sold by Salem Laboratory and track their orders. A previous version of the website, called **VM Report,** was already in existence for several years, but was not used by the clients, who instead were making orders by phone.

In order to bring the user back to the platform, I was tasked with the **re-design and the development** of a new version of the current website.

My first step was to identify the major problems with the existing site. What I found was the old version was extremely slow, which meant the process of buying a product was taking too long and the user had to go through too many different steps. In particular, the main menu itself had up to 10 sub-categories and even an extra level for some. Further, the interface was too complicated and it was hard to find specific information, the website was not responsive and could not be used on a smaller screen, which left some clients with no choice but to use other means to place an order.

The website was not perceived as a tool that was helping them solve their problems, but as a complicated and slow system that was making the process of buying products and managing their orders tedious and uncertain.

The future website should provide simple, intuitive, and responsive user interfaces that lets users accomplish ordering with less effort and time. The user interaction should be fast, and the process of buying new products should be easy. Ultimately, the website should feel more like a web application.

## **Information on Demand**

Simplicity is important in user interface (UI) design, the more controls that are displayed on a page, the more time the user will have to spend figuring out how to a complete a specific action. When there is less choice, the available functions become more apparent and are easier to scan.

To make the navigation less confusing, I chose to only display the main categories in the menu:

* Dashboard (A collection of the main client information)
* Order (Where the user can order products)
* My page (Where the user sees all personal information and a collection of previous activities)
* Contact

To conceal additional information available inside the page, I used a simple tab system. Additional information can then be retrieved by clicking on the action button which will open a new pop-up window.

Using this method, information, while still visible, is only fully displayed when the user needs it, resulting in a more simplified and user-friendly experience.

To focus the visitor's attention, I used color contrast and proportion differences between elements. Orders or messages are in a small white box, and information with denser data are displayed in a larger, darker box.

**Building a Fast and Responsive Front-End**

I chose AngularJS as the front-end framework for its two-way data binding abilities. This way of handling data allows for an easier and faster UI interaction.

A good example of this principle is the order page.

Thanks to the power of two-way data biding, information is updated in real-time giving instant feedback to the user.

For the server side, I used the three other parts of the four main components of the MEAN stack:

* MongoDB as the database
* Express as the web framework
* AngularJS as the front-end framework
* NodeJS as the server platform

In addition to providing all the advantages of a SPA, the MEAN stack offers other benefits. Whereas the LAMP stack is based on several different languages, the MEAN stack is a full JavaScript stack making programming MEAN-based applications significantly easier to use. NodeJS is also faster and more scalable than other server side technologies, including LAMP, due to its non-blocking architecture.

#### **MEAN stack example: The invoices**

Invoices are created each time an order is made through the application, and they are stored in the MongoDB database so they can be reviewed later. Like any parts of the application, the invoice is a module with the following structure:

-The folder view contains all of the invoice HTML templates.

-The controller.js file is in charge of fetching data from the factory and sending it to the view via its scope. The factory.js file uses the $ressource object to retrieve data from a specific URL.

-On the server side, data is requested to the MongoDB database by using Mongoose. Mongoose is an object modelling tool designed to help translate the database data into JavaScript objects, which can then be used in the application.

-Using this method, each part of the application is divided into individual modules (orders, invoices, login, notifications, etc.) making the code more scalable and maintainable. Data and business logic that are shared across multiple modules are stored inside services.

#### **The tab module**

One of the main ways to display data on the application (as seen in the design part of this post) is the tab system. Since it is used often throughout the application, and with a lot of different data, I wanted to build a modular tab system that can be added to, removed, modified, and displayed easily using a set of simple HTML attributes.

Using Angular's custom directive, the HTML code below will display one block containing three tabs: orders, stocks and messages, along with their corresponding data.

This tab module is divided into two main parts, the structure of the tab itself and the tab and the data itself. The right structure is loaded using the module directive, based on the structure argument value provided in the HTML mark-up (structure="tab"). The data that needs to be loaded inside each tab block is identified from the argument content and stored inside an array called TabsData.

The template tab displays the tab structure and names and then makes a call using ng-include to the data template.

The front-end build process includes the use of Grunt for automated tasks, like process SASS files, runs a development server, and concentrates and minimizes files for better performance.

02 / (original page > http://davidibanez.com/works.php?link=film)

## A responsive movie recommendation website build with Susy.

## **Brand & Website Design**

Ciné chez soi (or theater at home) is an ongoing personal project that combines two of my main passions; movies and web design/web development. I designed the brand and developed the website using SASS and Susy for the responsive grid.

## **Movie Recommendations**

This website is a movie recommendation platform aimed at movie-lovers that will provide movie reviews, ratings, summaries, comments, and more.

I wanted the website to be accessible on multiple devices, browsers, and screen sizes and have a clean, movie-poster-centric look.

There are desktop and mobile version of the website.

## **Responsive Design with Libsass and Susy**

The homepage was developed using **HTML** and SASS. Because my [SASS workflow](http://localhost/portfolio/www/works.php?link=film) includes multiple plug-ins like [Bourbon](http://bourbon.io/), [Vertical Rhythm](https://github.com/neoeno/libsass-compass-vertical-rhythm) and Susy, I used **Libsass** rather than Ruby to compile SASS into CSS. Libsass is a c/c++ port of SASS that is up to [4000 times faster than Ruby](https://www.solitr.com/blog/2014/01/css-preprocessor-benchmark/) and is easier to integrate into any project.

There are many tools available on the web that I have used in the past to quickly design grid for the web, from CSS framework like **Bootstrap**, **Foundation**, to sites like **Responsive Grid System**.

The problem with these solutions is that they usually add a large number of classes to my mark-up that is already getting crowded because of my adoption of the [BEM convention](http://localhost/portfolio/www/portfolio.php#code) into my workflow which can sometimes produce a bloated codebase that can impact performance.

The solution to these problems for me was [Susy](http://susy.oddbird.net/): a **SASS based grid framework** that is fast, responsive, lightweight, and enables me to create an entirely customized grid without adding anything to the mark-up.

#### **Setting up the grid**

First of all, I created default parameters for the grid in a map sorted in setting.scss that contains all of the global settings of the website. Here is how the syntax look:

In this map, several global values are defined. The container is set to 100% so that the layout extends to the full width of the page, the number of columns is set to 12 and all the blocks in the website will be a subdivision of that number, gutters value is set to 0 and the box model used is set to border-box.

Here is the layout that was implemented using the default Susy map value:

There are desktop and mobile grid version of the website.

As stated previously, each main container is a subdivision of the grids total number of columns. Thanks to the default values set up in the **settings file**, the block size will be updated automatically if the container size or the number of columns change.

#### **Implementing the grid system**

Susy depends entirely on SASS to customize the grid, so there is no need to add anything else to the HTML mark-up. The most important feature for creating grids in Susy is the span mix-in.

As you can see from the image above, the main blocks must occupy a specific space inside the 12 columns grid.

On mobile devices, all of the main blocks extend to the full width of the grid and the header is divided into 2 parts, the logo that takes 4 columns out of 12 and the menu that takes 8 columns.

Here is how the main level grid blocks were implemented with a mobile first method:

#### **Making it responsive with breakpoint**

Although the grid built with Susy is fluid at this stage, the website is still not responsive. To make the layout responsive, I used another mix-in, Susy-breakpoint. The Susy-breakpoint mix-in makes it easier to handle media queries in the layouts. It does so by simplifying the language needed to create media query breakpoints.

In my settings.css file, I defined a set of default breakpoints by devices:

Those variables are then used inside the Susy-breakpoint mixin when needed.

This is how the main blocks of the homepage layout were modified to include the breakpoints.

## **Conclusion**

Using Susy in this project made me realize how easy and fast it is to create simple or complex layouts with this grid framework without a huge amount of code. This framework

will save me time in the future, especially when designing directly in the browser or prototyping a website layout.

03 / (original page > http://davidibanez.com/works.php?link=osis)

## Brand and website design for an IT company.

## **Brand & Website Design**

Osis is a French IT service company specializing in open sources technologies. I designed their brand and website with the aim of promoting the company's work and helping them gain more exposure in the marketplace.

## **Define the Brief**

The first thing I did for this project was to determine what exactly the project goals were.

What is the purpose of the project? What problems am I trying to solve? I wrote this out in a document called **The Brief**. The goals of the project were to create an online profile to promote the company's work and obtain more exposure. It needed to show what type of services it offered, display past or ongoing work, and have a way for people to contact the company easily. At this point, I simply figured out the ultimate goal of the project. I recorded these and referred back to them in later stages of the project to ensure that it was on the right track.

#### **Research and idea generation**

After the project’s goals had been decided upon, and before beginning the design, I spent some time doing research.

I usually start by identifying the company's top five competitors. How do they emphasize their value proposition? How do their prices compare? Answering these questions helps in creating a brand that will have a better chance of being successful. Once I roughly establish a direction as to where the brand should go, I conducted more research, but this time aimed at generating creative ideas. Sometimes I might look at other sites for inspiration, but sometimes not looking at what other designers have done is best.

Instead, I like to brainstorm my own solutions to the problem at hand. All is needed at this stage is a pen, a sketchbook, and my thoughts.

##### 

## **Designing the Brand**

The brand of a company is basically the visual language that describes it and determines how others see it.

According to the brief, the brand image needed to convey a strong, memorable look and feel. The client wanted to use a white bear as the brand mascot for its strong and wild image, which fit a certain idea of open source. With that idea in mind, I started by searching for a variety of white bear images for inspiration and selected a few as my guide. I then sketched some ideas and experimented with typography and letter arrangement before deciding on the final logo (which I drew in Adobe Illustrator).

#### **Thinking about content**

When designing a website, one of the first things that needs to be decided upon is the content, because for most projects this will determine the design.

What information needs to be conveyed to the visitors? Should a given point be written as simple text, or would it make more sense as an image or a diagram? At this point in the project, I needed to think about the message and how to convey it. Therefore, I asked the client to provide me with the content early on and wrote a draft, which I broke down into six main parts: home page introduction, services description, projects, a parterres list, and the company's contact details.

#### **Wireframe**

From the content gathered earlier, it was clear that the client wanted a brochure website that outlined their services and provided potential customers with an easy way to contact them.

I decided to go with a simple one-page website broken into five sections, as seen in the sitemap above. Clicking on a link in the top navigation bar would slide the user down to the appropriate section of the page. I sketched out wireframes for each section of the site to block out the basic elements and to give me some blueprints to work from.

The approach I take to wireframes is simple but effective, and all I need at this stage is a pen and sketchbook. I first list all of the elements to include on a web page. I then group related elements, before prioritizing these groups according to importance. Once the page's elements are grouped and prioritized, arranging them on the page is much easier. More important elements are placed towards the top of the page, and white space is used to create groupings.

#### **High quality mock-up in Photoshop**

Once the desktop wireframes were sketched, I moved into Photoshop and started a mock-up of the website in more detail.

Because this project was a one-page website, I designed all of the five different sections using Photoshop, but usually I try to not spend much time in this program as it slows down the development process. Instead, I simply mock-up the main page templates, along with any other design elements and assets that will be needed. Similarly, I did not mock-up any mobile or tablet designs in Photoshop, because I find that simply coding these based on the wireframes to be quicker.

04 / (original page > <http://davidibanez.com/works.php?link=portfolio>)

Responsive, cross-browser portfolio built with SASS.

## **Brand & Website Design / Development**

My portfolio was designed and built with an emphasis on content to showcase my past and current work. The article below details the workflow I used to manage the website styles using BEM, OOCSS and SMACSS for SASS.

## **Making the Content Matter**

A professional portfolio is a necessity for anyone working in the creative industry. It is a good opportunity to show my work in a medium that actually showcases the work I really do: web design and web development. In building my portfolio, my main goal was to create a simple design that would facilitate the work of a potential employer who wishes to form a quick impression.

The portfolio site itself should fade into the background and act as a frame for the work I am presenting. Therefore, like for most websites in general, the **content should be king**, with beautiful images and texts that should be accessible regardless of the screen size, or browser it is viewed on.

## **Visual Simplicity**

I chose colors that will give the website a high contrast in order to bring focus to the content.

All of the pages of the website follow the same layout principle. They are divided into two sections; one side is shows additional information, the other, the main information of the page. This principle is then to decline on the different pages to avoid repetition.

For example, the Homepage is divided into two parts and the main content is emphasized using color contrast. On the contact page, a focal point is created by using proportion differences, the main content being larger than the additional information.

These design choices allow me to build simple, but well organized, pages that emphasize the content while also allowing for a more modular approach when developing the website.

##### 

## **A Modular Approach**

If content is king for a web designer's perspective, for a web developer, however, some may argue that **speed is king**. Unfortunately, CSS is often overlooked in this area while many front-end developers focus largely on JavaScript performances.

To help improve my CSS development process, for this project I decided to try a different approach to Styling using some of the most popular methods for writing **modular HTML and CSS;** the [BEM](http://getbem.com/naming/) method (Block, Element, Modifier), the [OOCSS](https://github.com/stubbornella/oocss/wiki) method (Object oriented CSS) and the [SMACKS](https://smacss.com/) method (Block, Element, Modifier). I used these methodologies along with SCSS.

#### **BEM**

The BEM methodology is all about modularity, each element of the page that is used more than once can become a module. To become a module, an element needs to be structured and styled in a specific way:

* Containers need to be separated from the module itself, so that it can be moved anywhere on the page without breaking the layout.
* The classes name needs to reflect the function of the element and must follow the [BEM](http://davidibanez.com/works.php?link=portfolio) naming convention.

The container wrapper\_\_lang--left position the module block Lang in the page. It is styled separately from the module itself so it can be used in different contexts. The module lang elements are named according to their function and preceded by a double underscore \_\_dropdown. The modifier of an element is marked by a double dash followed by its name --items.

Using SCSS, the module styles are then stored in a SCSS file simply called lang that is organized according to the module HTML structure.

All of the JavaScript code that affects that element is also organized in a file called lang. The module is, therefore, self-contained and can be added, removed, or modified easily without breaking the layout or the code, allowing for a more flexible, scalable, and maintainable code.

#### **OOCSS**

As with any object-based coding method, the purpose of OOCSS is to encourage code reuse and, ultimately, faster and more efficient stylesheets that are easier to add to and maintain.

OOCSS is based on one main principle, **the separation of skin and structure.** Almost every element on a styled web page has different visual features (“skins”) that are repeated in different contexts. For example, the color or the text styles. When these features are abstracted into class, they become reusable and can be applied to any element and have the same basic result. When developing my portfolio, I used this principle for many elements like the colors, the buttons, the headlines, etc.

All of the styles are contained inside a class "para", which is itself a module, all of the different font sizes are stored as an element of that module, and the color variations of a specific size are stored as a modifier. The common styles are combined into a reusable “skin” and nothing is unnecessarily repeated. I just need to apply the “skin”, for example the class para\_\_large--dark to all the elements and the result will be the same.

#### **SMACSS**

At the very core of SMACSS is a way to categorize CSS rules. For this project I used six types of categories:

The purpose of this categorization is to identify patterns that repeat themselves within the design, therefore, limiting repetition resulting in less code, easier maintenance, and greater consistency in the user experience. My workflow was powered by Grunt.js to provide automated tasks such as SCSS preprocessing, image and SVG optimization. I also used **require.JS** to organize and optimize my JavaScript files.

## **Conclusion**

Thanks to this project, I was able to explore modularity using principles like **OOCSS, SMACSS and BEM** and take advantage of **SCSS** to create better organized stylesheets. This new approach to CSS styling will be particularly useful in future large-scale projects.

05 / (original page > http://davidibanez.com/works.php?link=elsa)

An E-Commerce website designed and built with Odoo ERP/CRM.

ELSA is an E-Commerce website that sells pharmaceutical products to professionals for Salem Laboratory. I was in charge of designing and developing the website using Odoo EPR/CRM for the back-end.

#### **Website design**

I worked with the client to decide upon the site structure and content. The sitemap was purposely kept small and shallow to ensure that the website was easy to navigate.

The project had two goals: allow the visitor to buy various products sold by Salem Laboratory and introduce the company to the users. After meeting with the client, I started looking for a different kind of structure than the traditional product list/Slider used in most E-Commerce website. Rather than showing a list of products right away, I was looking for a way to show only important information, like the latest promotions or the newest products so that there would be enough space left for a proper introduction of the company.

Once the client was happy with the initial direction, I moved into Photoshop to create the high fidelity mock-up.

On the product list page, additional space was reserved to promote specific products, show the latest product and promotions, and provide direct access to the shopping cart. Products can be accessed by categories and a search bar is present at the top of the product list allowing for a fast and easy search for a specific product.

**Development**

The client was already using Odoo for other parts of its business and wanted the ELSA website to be integrated into that pre-existing infrastructure.

Therefore, the website was developed using [Odoo](https://www.odoo.com/) for the back-end and HTML and CSS/SASS for the front-end. Odoo is an open source **ERP/CRM** with E-Commerce capabilities that is based on **Python, XML, JavaScript** and the Odoo's own templating engine language **Qweb**.

06 / (original page > <http://davidibanez.com/works.php?link=lsa-portail>)

A corporate website for Salem laboratory.

## **Website Design & Development**

Labosalem Portail is Salem Laboratory's corporate website that helps inform and guide users to the company's other websites. I designed and developed it using HTML, CSS/SASS and JavaScript.

#### **The project**

I was tasked with the design of a portal website for Salem Laboratory which aimed to inform and direct users to other various websites of the group.

Customers could be basically divided into four groups; the doctors, the wholesalers, the pharmacist, and the clients. Professionals, like the pharmacists or the wholesalers, need to access the company's one-line retail platform, whereas clients and doctors should be directed to the corporate and products website.

img-project-centerTo achieve these goals, I opted for a one-page layout website divided into four sections. One of those sections allows users to choose a profession, they are then presented with specific content based on that choice.

img-project-center

07 / (original page > http://davidibanez.com/works.php?link=isucare)

A product brochure website for Salem laboratory.

## **Website Design & Development**

Isucare is a high performance blood glucose monitoring device, sold by Salem laboratory. In order to promote the product, I was tasked with the design and development of a brochure website.

## **Creating a template**

As part of the design and development team, my task was to help create the first product website of Salem Laboratory's product lineup: Isucare, a blood glucose monitoring device.

Since the website was to be one of a series of other product websites, we decided that in order to create a homogeneous visual identity and facilitate the development process, Isucare would serve as a template for all of the other product websites.

## **Planning the Website**

When designing the website, one of the first things we did was decide upon the content. What are we trying to say with the website? Once we established the purpose of the website, we set out to find the most effective way to deliver the message.

It was clear that the client wanted a brochure website that outlined the product’s benefits, line-up, and provided the potential customer with an easy way to contact them. We decided to go with a one-page website divided into three parts, as seen in the site map above. Clicking on one link will slide the user down to the right section of the page.

img-project-center

## 

## **Putting It Altogether**

The website was developed using **HTML/CSS, JavaScript** for the front-end and **PHP** for the back-end. I used the Jquery plug-in [Fullpage.js](http://alvarotrigo.com/fullPage/#secondPage) to create the full page scrolling effect.