## **Parte I: Investigación sobre Mockup**

## **¿Qué es? Y su importancia en el desarrollo Web.**

En la [manufactura](http://es.wikipedia.org/wiki/Manufactura) y [diseño](http://es.wikipedia.org/wiki/Dise%C3%B1o), un Mockup, o maqueta es un modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Un mockup es un [prototipo](http://es.wikipedia.org/wiki/Prototipo) si proporciona al menos una parte de la funcionalidad de un sistema y permite pruebas del diseño. Los mockups son utilizados por los diseñadores principalmente para la adquisición de comentarios por parte de los usuarios. Los mock-ups abordan la idea capturada en la ingeniería popular: Usted puede arreglarlo ahora en el dibujo con una goma de borrar o más tarde en la obra con un martillo.

Los wireframes son una vía de comunicación entre los llamados arquitectos de la información, los programadores, los diseñadores y los clientes que encargaron el diseño visual del sitio. Los wireframeso esquemas también pueden ser utilizados para pruebas de usuarios, para evaluar la reacción de las visitas.

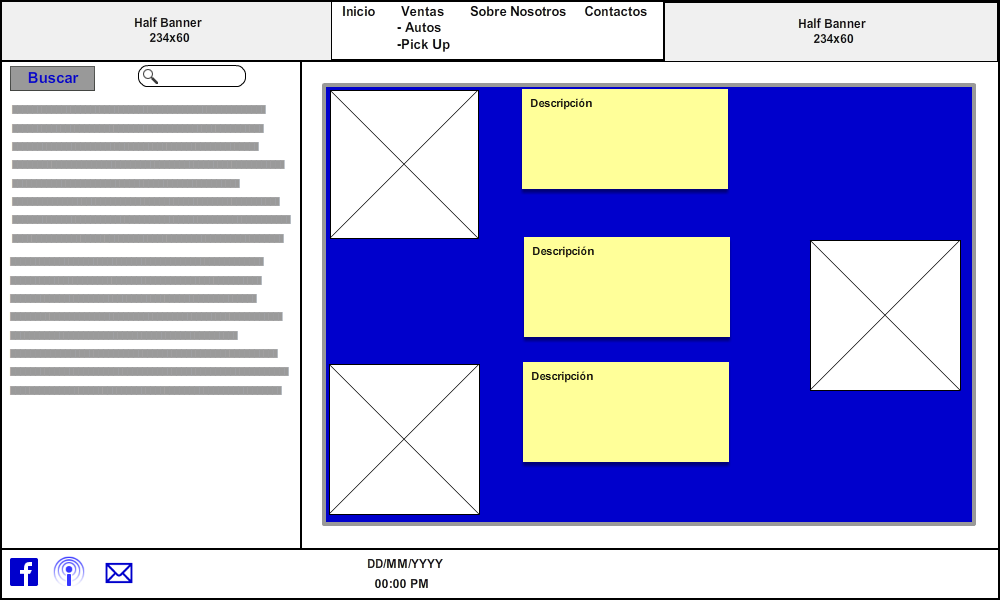
Existen diferentes ventajas: por un lado, con un wireframes el desarrollo se centra en el diseño de contenidos y no en un diseño estrictamente visual. Por el otro lado, los wireframes*o*mockups significan una reducción de riesgo de perder dinero. ¿Por qué?, porque el diseño de contenidos se evalúa antes de comenzar a ejecutarse y desarrollarse.

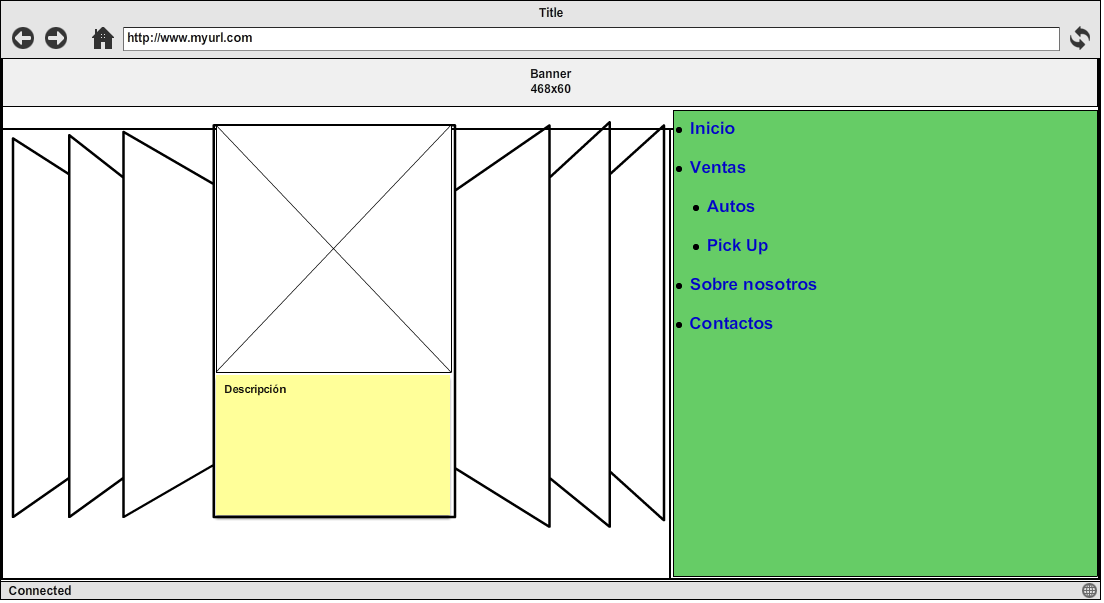
## **Herramientas libres para desarrollar Mockup**

1. [**Mockflow**](http://www.mockflow.com/): cuenta con una librería de plantillas donde podrás elegir la más adecuada para organizar tu contenido web. Si bien es gratuito, existe una versión de pago para usuarios que necesiten funciones más avanzadas.
2. [**Mockingbird**:](https://gomockingbird.com/) permite crear y compartir wireframes online. Interfaz drag and drop muy simple de utilizar, interfaz limpia y fácil de usar.
3. [**Lovely Charts**](http://lovelycharts.com/): herramienta online gratuita para crear el esquema de tu sitio web. Ideal para quienes buscan una alternativa simple para diseñar su wireframes*.*
4. [**Cacoo**](https://cacoo.com/): es una herramienta fácil de utilizar para crear wireframes completamente online Su característica más destacable es la posibilidad de colaborar entre usuarios en tiempo real.
5. [**Gliffy**](http://www.gliffy.com/): otra herramienta gratuita y online para crear Mockups de tu sitio web. Cuenta con una librería de esquemas y con una API para que los desarrolladores puedan integrar la herramienta con sus aplicaciones.

## **Elaborar un Mockup (3 vistas diferentes) para la página principal del su PTC.**

## 0 Imagen





# **Parte II: Framework**

## **Presentar de 5 a10 framework para trabajar con JavaScript y una breve descripción del mismo. Especificar si es libre o comercial.**

1. Backbone.js: es el framework preferido por muchos. En todos lados escucharás que están hablando de Backbone, muchas empresas importantes lo están utilizando.

Ventajas: La comunidad es fuerte y está creciendo bastante. Underscore.js (que se utiliza en gran medida) es también un gran framework.

Desventajas: Carece de abstracciones fuertes y deja mucho que desear. El framework enteró es sorprendentemente ligero y da lugar a que sea repetitivo. Cuanto más grande es una aplicación, más se hace evidente.

1. Ember.js: (antes Amber.js SproutCore 2.0) es uno de los nuevos contendientes. Es un intento de sacar a las características principales de SproutCore 2.0 en un framework modular compacto más adecuado para la web.

Ventajas: Extremadamente rico sistema de plantillas con vistas compuestas y los enlaces de la interfaz de usuario.

Desventajas: Relativamente nuevo. La documentación deja mucho que desear.

1. Knockout.js: es un framework MVVM que recibe destaca enlaces declarativos, interfaz de usuario y la interfaz de usuario de actualización automática.

Ventajas: Gran documentación y un sistema tutorial increíble.

Desventajas: Carece de una jerarquía de vistas de componente sólido. No es tan fácil reutilizar los componentes fácilmente. Siento que el usó de un framework MVVM es perjudicial (tal vez porque ya esté acostumbrado al MVC). Casi ninguno de estos framework es MVC, pero son de la variedad MV (MVP, MVVM, etc).

1. JavaScript MVC: es un gran framework que saca las mejores características de jQuery, así que si te gusta utilizar bastante jQuery,

Ventajas: Tiene una comunidad sólida.

Desventajas: Torpe modelo de herencia basada en cadenas. Los controladores son demasiado íntimos con vistas. El nombre es demasiado genérico - es cómo si RoR se llamará "Ruby Web Framework"

## Angular.js: es uno de los mejores framework. Desarrollado por Googlers, tiene algunas opciones de diseño muy interesantes.

## Ventajas: Las plantillas y diseño del controlador. Cuenta con un sistema de inyección de dependencia (soy un gran fan de la IOC). Tiene gran soporte de UI-Binding y sin duda su sintaxis es muy fácil de entender.

## Desventajas: El código base parece ser bastante extenso y no muy modular. Las vistas no son lo suficientemente modulares

# **Parte III: Sistema de control de versiones**

Investigue que son los sistemas de control de versiones y las ventajas que presentan en el desarrollo de proyectos de software.

## **¿Que son los sistemas de control de versiones?**

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión

Dando lugar a los llamados sistemas de control de versiones o SVC (del inglés System Versión Control).

Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas

## **Ventajas de sistemas distribuidos**

Necesita menos veces estar conectado a la red para hacer operaciones. Esto produce una mayor autonomía y una mayor rapidez.

Aunque se caiga el repositorio remoto la gente puede seguir trabajando

Al hacer los distintos repositorio una réplica local de la información de los repositorios remotos a los que se conectan, la información está muy replicada y por tanto el sistema tiene menos problemas en recuperarse si por ejemplo se quema la máquina que tiene el repositorio remoto. Por tanto hay menos necesidad de backups. Sin embargo los backups siguen siendo necesarios para resolver situaciones en las que cierta información todavía no haya sido replicada.

Permite mantener repositorios centrales más limpios en el sentido de que un usuario puede decidir que ciertos cambios realizados por él en el repositorio local, no son relevantes para el resto de usuarios y por tanto no permite que esa información sea accesible de forma pública. Por ejemplo es muy útil se pueden tener versiones inestables o en proceso de codificación o también tags propios del usuario.

El servidor remoto requiere menos recursos que los que necesitaría un servidor centralizado ya que gran parte del trabajo lo realizan los repositorios locales.

Al ser los sistemas distribuidos más recientes que los sistemas centralizados, y al tener más flexibilidad por tener un repositorio local y otro/s remotos, estos sistemas han sido diseñados para hacer fácil el uso de ramas (creación, evolución y fusión) y poder aprovechar al máximo su potencial. Por ejemplo se pueden crear ramas en el repositorio remoto para corregir errores o crear funcionalidades nuevas. Pero también se pueden crear ramas en los repositorios locales para que los usuarios puedan hacer pruebas y dependiendo de los resultados fusionarlos con el desarrollo principal o no. Las ramas dan una gran flexibilidad en la [#uso de ramas forma de trabajo].

## **Ventajas de sistemas centralizados**

En los sistemas distribuidos hay menos control a la hora de trabajar en equipo ya que no se tiene una versión centralizada de todo lo que se está haciendo en el proyecto.

En los sistemas centralizados las versiones vienen identificadas por un número de versión. Sin embargo en los sistemas de control de versiones distribuidos no hay números de versión, ya que cada repositorio tendría sus propios números de revisión dependiendo de los cambios. En lugar de eso cada versión tiene un identificador al que se le puede asociar una etiqueta (tag).