Framework para el Desarrollo de Aplicaciones

Documento de Arquitectura

Resumen:

|  |  |
| --- | --- |
| Documento: | Framework de Desarrollo de Aplicaciones - Documento de Arquitectura |
| Versión: | 1 |
| Páginas: | 6 |
| Fecha: | martes, 16 de octubre de 2012 |
| Autor: | Carlos Vazquez |
| Cargo: | Arquitecto de Sistemas |
| Contacto: | [carlos.vazquez@outlook.com](mailto:carlos.vazquez@outlook.com) |
| Destinatario: | Elías Vernieri | Paula Vedaras | Ana |

CONTENIDO

[0. Historial de Cambios 3](#_Toc338154509)

[1. Estructura lógica del Framework 3](#_Toc338154510)

[2. Implementación 4](#_Toc338154511)

[3. Productos utilizados 7](#_Toc338154512)

[4. Aspectos no provistos 8](#_Toc338154513)

[5. Algunos productos implementados con el Framework 8](#_Toc338154514)

[Anexo Técnico 15](#_Toc338154515)

### 0. Historial de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| ***Revisión*** | ***Autor*** | ***Fecha*** | ***Descripción*** |
| 1.0 | Carlos Vazquez | 26/09/2012 | Primera Versión |

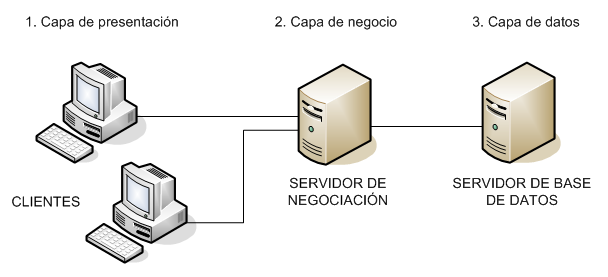
### 1. Estructura lógica del Framework

Debido a los requerimientos de la aplicación se utilizará un modelo de desarrollo en 3 capas bien definidas, capa de presentación, capa de negocios/servicios y la capa de datos.

Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio/servicios: es donde residen los [programas](http://es.wikipedia.org/wiki/Programa_(computaci%C3%B3n)) que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de [base de datos](http://es.wikipedia.org/wiki/Base_de_datos) almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.



Nota Importante: Este Framework no es un Framework en el sentido estricto de la palabra sino que es un marco de trabajo que implementa un conjunto de herramientas realizadas utilizando las mejores prácticas, probadas y aceptadas por la comunidad de desarrolladores, interconectadas entre sí que resuelven los problemas genéricos de toda aplicación web.

Junto con el Framework se provee un ejemplo de un ABM que utiliza los aspectos detallados en el documento.

### 2. Implementación

De la evaluación de los requerimientos funcionales de la aplicación se decide utilizar el patrón de diseño de aplicaciones Model-View-Controller de aquí en más (MVC) por las siguientes razones.

Testeabilidad: Debido a que existe una clara separación de tareas entre las capas promoviendo la utilización de Tests Unitarios sobre cada capa utilizando NUnit/Moq.

Escalabilidad/Extensibilidad: El diseño de la aplicación es muy escalable debido a la utilización del Framework Autofac que permite la inyección de dependencias en controladores, servicios y acceso a datos dando la posibilidad de realizar modificaciones sobre estas con un impacto mínimo.

Este patrón ha sido implementado por Microsoft en el Framework MVC. Este Framework implementa MVC en su versión 4.

El Framework definido posee las características de AOP (Aspect Oriented Programming) que son óptimas en el momento de implementar seguridad, logging, transacciones, etc. ya que el código generado en estas tareas “transversales” al problema a resolver es muy poco intrusivo.

**2.1 A.O.P**

**2.1.1 Logging**

Para el logging de información se utiliza el componente Open Source Log4Net.

Este componente se encuentra en versiones muy estables y confiables y es prácticamente el estándar al momento de realizar logueo de información.

El logging de la aplicación se realiza mediante un archivo de configuración donde se podrá registrar toda la información necesaria al medio especificado que puede ser tanto un archivo de texto como una base de datos o el envío de mail si es un error, un warning o información.

**2.1.2 Seguridad**

Se utiliza el componente ASP.NET Membership Provider de Microsoft provisto en Visual Studio 2012.

Para el manejo de roles también se utilizará el mecanismo Standard de roles por usuario provistos por el ASP.NET.

Se pueden seleccionar 2 métodos de seguridad una es la seguridad por Windows y la otra es seguridad custom implementada con la funcionalidad provista por los componentes anteriormente citados. El Framework posee clases extensibles que permiten agregar otros modelos de seguridad dentro del mismo como OpenID.

**2.2 Entity Framework 5**

Se utiliza esta librearía para la persistencia de las entidades en la base de datos. Esta librería tiene amplia aceptación en la comunidad de desarrolladores y está implementada por Microsoft.

La ventaja que ofrece es la de proveer un ORM claro y simple que permite abstraer la base de datos con la que se está trabajando, de esta manera es muy simple intercambiar bases de datos como SQL, Oracle, MySql, etc sin provocar impacto en la aplicación debido a la utilización del patrón *Repository*.

Otra de las características que tiene este ORM es ayudar a la testeabilidad de las aplicaciones (en conjunto con la inyección de dependencias mediante Autofac) debido a que la lógica de la misma no queda atada a la base de datos permitiendo de esta manera la utilización de mockups al momento de testear la capa de negocios.

Otra ventaja es la escalabilidad ya que al no estar la lógica en la base de datos –de difícil escalamiento- la capa de negocios es mucho más fácil de escalar, remitiéndose solamente al agregado de más hardware y el posterior deploy de los componentes de negocio.

**2.3 Autofac**

Autofac es un contenedor de inversión de control, este producto se utiliza para no atar las clases a una implementación particular implementando una interface, de esta manera es muy simple intercambiar componentes ayudando al desarrollo y promoviendo una muy buena testeabilidad.

**2.4 jQuery / jQuery UI**

Para la capa de presentación se utiliza como soporte las librerías jQuery. Estas librerías ya vienen incluidas en MVC 4 y constituyen un Framework de desarrollo en JavaScript simple y potente.

Se provee un set de librerías de efectos que permiten hacer efectos visuales como acordeón, carrusel, etc., así también como agregar capacidades extra a las páginas como validación de datos, customización de controles, etc. a los componentes HTML estándar ya existentes haciendo las aplicaciones altamente compatibles entre browsers debido a que este Framework está completamente desarrollado en JavaScript.

Para más información sobre los efectos disponibles dentro del Framework referirse a <http://themepixels.com/themes/demo/webpage/amanda/dashboard.html>.

Nota: El témplate de estilos detallado en la dirección web anterior viene incluido como extra dentro del Framework ofrecido.

**2.5 Tests Unitarios**

Uno de los objetivos al utilizar inyección de dependencias utilizando Autofac como container y MVC es la de facilitar la construcción de tests unitarios automáticos. Estos tests se realizarán utilizando NUnit y Moq, en la solución existen proyectos preparados para incluir el testing de los diferentes componentes.

Objetos del Dominio: Debería ser obligatorio hacer los test para objetos complejos, no es necesario hacerlos para objetos paramétricos.

Servicios: Debería ser obligatoria la confección de tests para la capa de negocios/servicios, estos serán realizados para las funcionalidades más importantes siendo lo más atómicos posibles.

### 3. Productos utilizados

Ha continuación se enumeran los productos que utiliza el Framework.

|  |  |
| --- | --- |
| https://encrypted-tbn0.google.com/images?q=tbn:ANd9GcTaaqtRpL9c74Yv1KaYNr32O9JZt0337skLDexzeYZ96QSQuGMNpg | Microsoft .NET Framework 4.5  Entity Framework 5 |
| https://encrypted-tbn3.google.com/images?q=tbn:ANd9GcT_GfqCHzsEJ3fFSza2_jxTfsUpJWJ7f9k2PjM9ku0td5s-Xm62 | Visual Studio 2012.  Lenguaje C#  MVC 4.0 |
| http://dfmoura.files.wordpress.com/2011/01/sql_server_2008_logo.png | SQL Server 2008 R2 |
| https://encrypted-tbn1.google.com/images?q=tbn:ANd9GcSSeAho0ud8KNcGSdtqCTls0M3Bos51Y1zwq0sVoEYJG7XbFyR3 | HTML 5  Autofac  jQuery / jQuery UI  Flot  Moq |

### 4. Aspectos no provistos

En esta versión actual este Framework no implementa:

* Menú con seguridad, ABM de seguridad.
* No implementa shopping cart.
* No soporta multilenguaje.

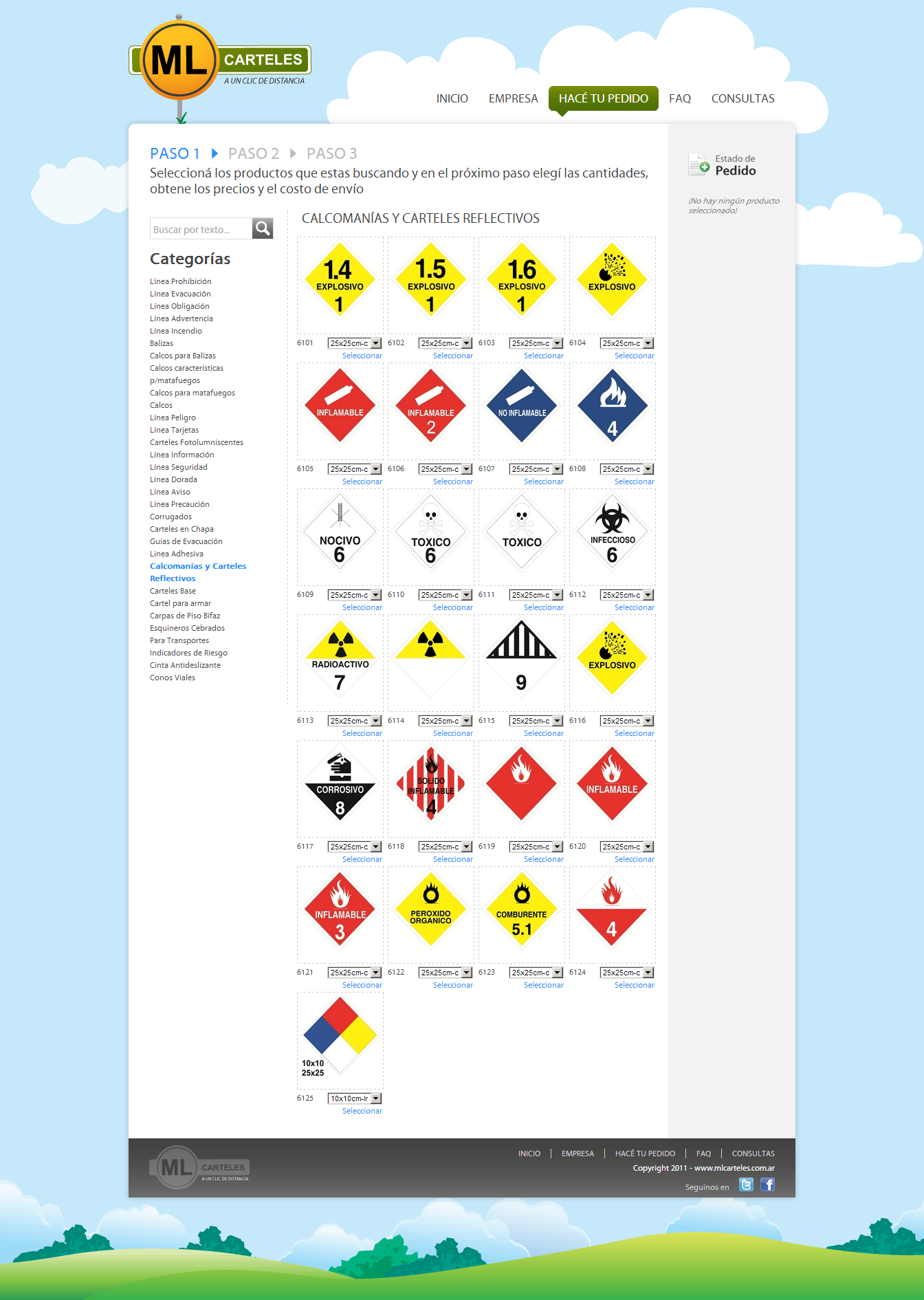
### 5. Algunos productos implementados con el Framework

**4.1 MLCarteles**

Esta aplicación es un e-commerce que permite la comercialización de cartelería de todo tipo.

Para conocer más detalles <http://www.mlcarteles.com.ar>.

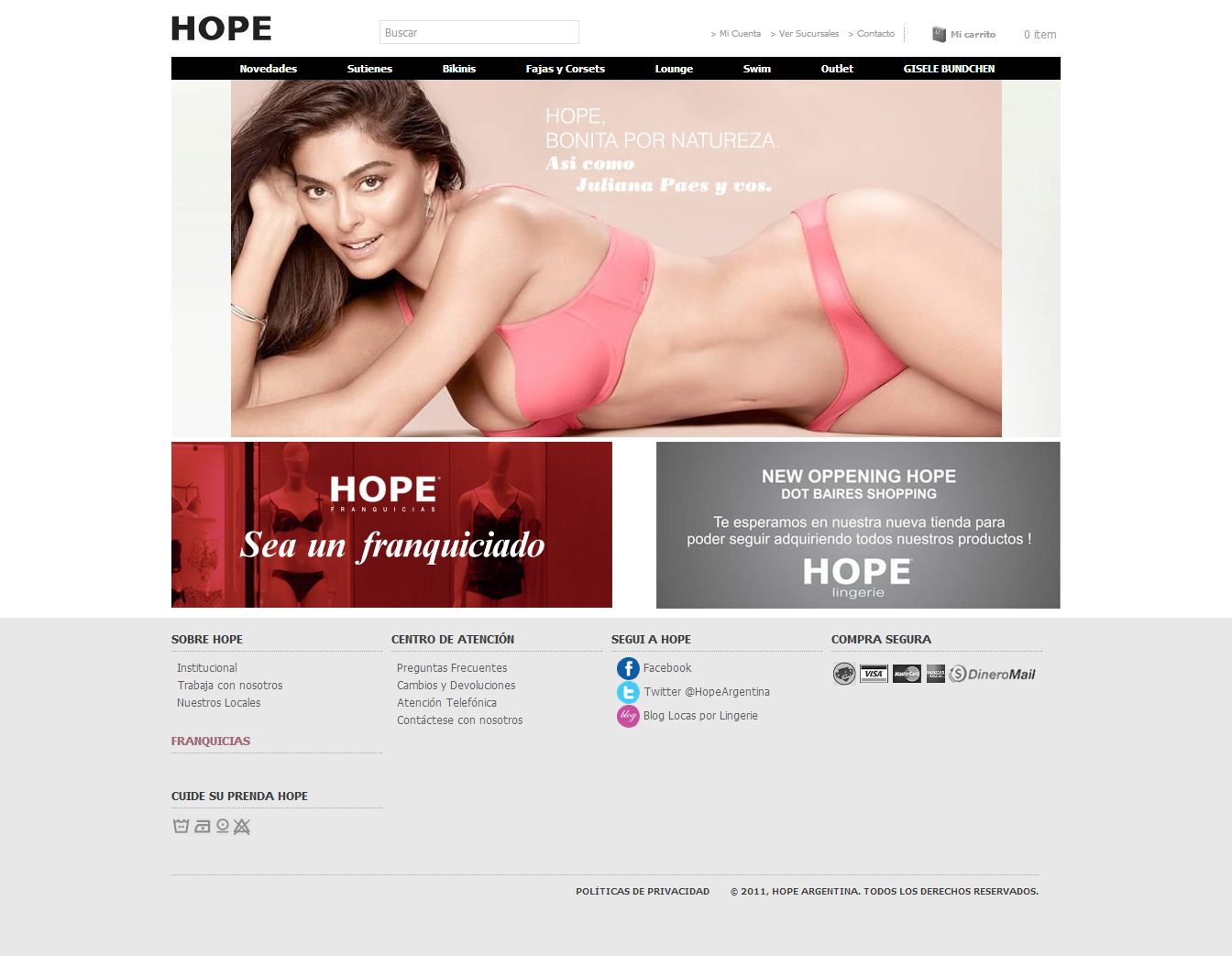


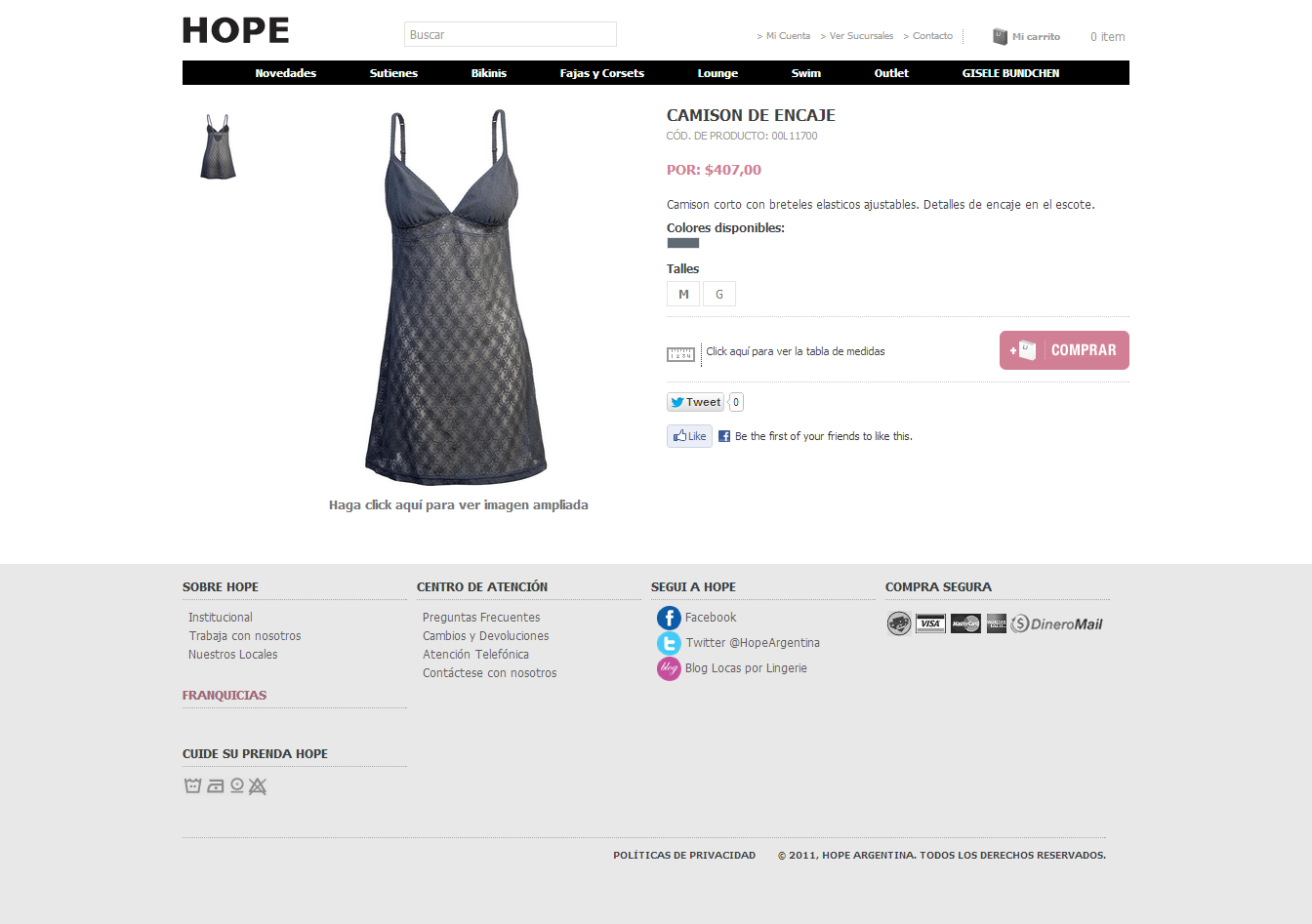


**4.2 HopeOnline**

Esta aplicación es un e-commerce que permite la comercialización de lencería. Este producto está desarrollado basado en el sitio origen de la franquicia, debido a que su costo de desarrollo debido a la utilización del Framework es de una décima parte del original.

Para conocer más detalles <http://www.hopeonline.com.ar>.

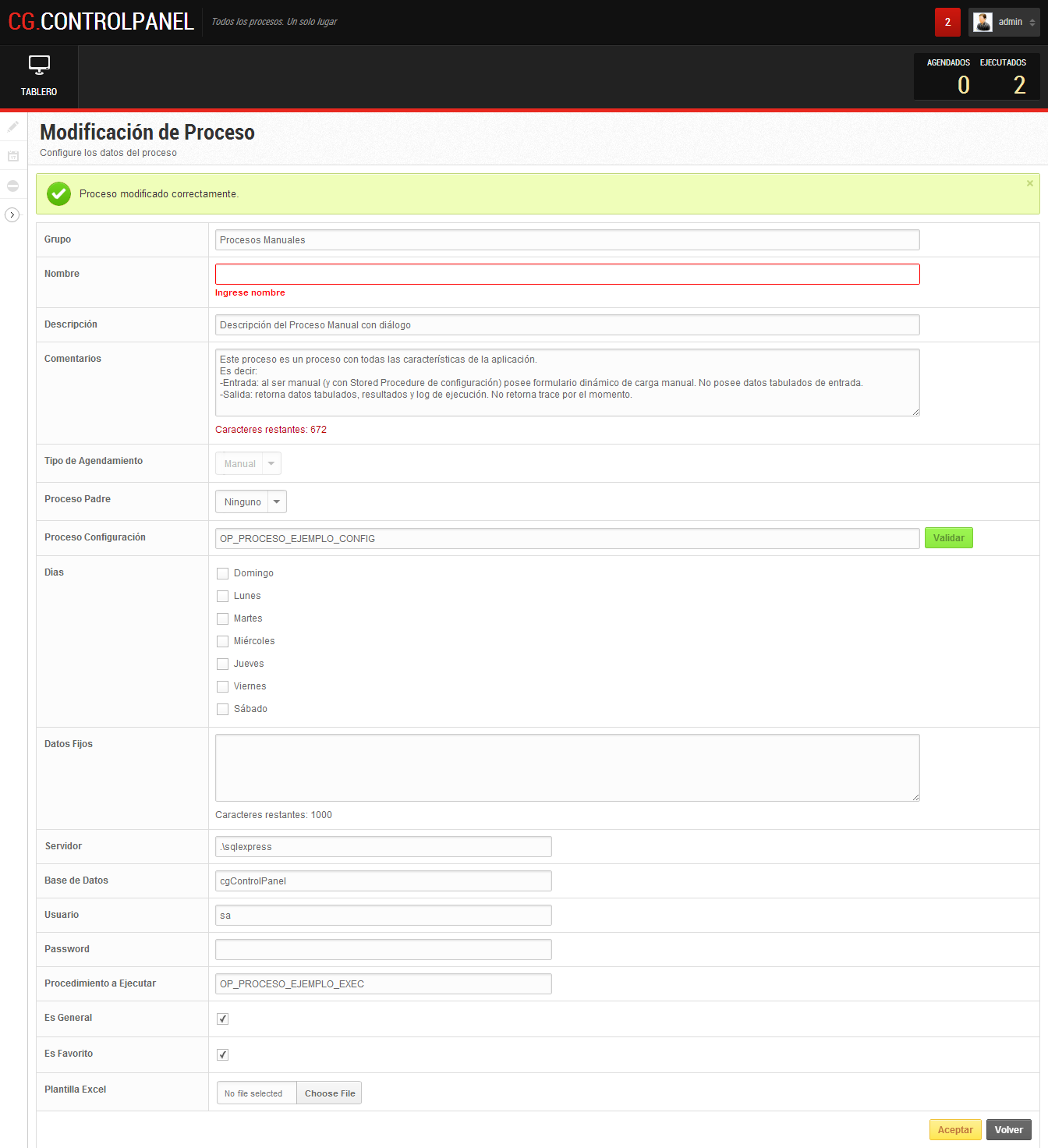


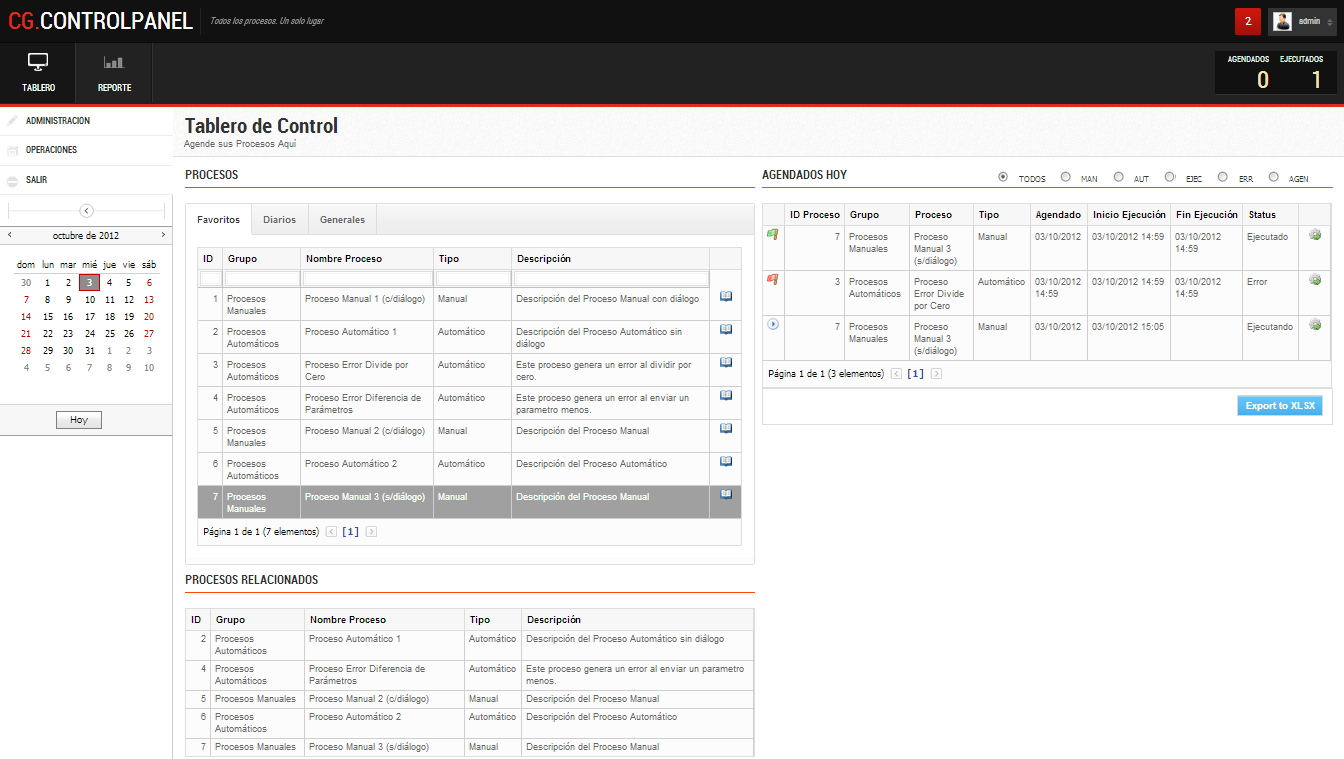


**4.3 CG.ControlPanel**

Panel de control que permite ejecutar procedimientos SQL, Oracle, o .NET desde un Dashboard simple y claro. A su vez permite la monitorización en tiempo real de los resultados obtenidos hasta el momento.

Esta es una aplicación de intranet de carácter privado por lo que no está disponible en la web.

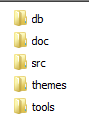




### Anexo Técnico

**1. Estructura del Proyecto**

La solución está organizada de la siguiente manera:



Detalle de cada carpeta

db: se guardan scripts de SQL y/o archivos de configuración que no pueden ser automatizados dentro del proyecto .net.

doc: en esta carpeta se almacena información inherente al proyecto.

src: código fuente del framework.

themes: esta carpeta contiene tres diseños de sites genéricos que pueden ser utilizados en el proyecto. También contienen diversos efectos y controles que agregan una excelente experiencia al usuario final.

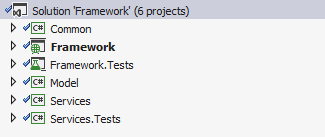
tools: herramientas que brindan soporte al framework.

* Encrypter: Esta herramienta permite encriptar claves a partir de una *PassPhrase*.

**2. Estructura de la Solución**

La solución está agrupada en proyectos, cada proyecto agrupa una determinada

* Common: Este proyecto almacena los componentes base de la aplicación como las clases base de los servicios y los controladores, también deberá contener las clases que implementen las excepciones y diferentes helpers que serán utilizados en toda la aplicación.
* Services: Esta capa es la que ofrece los servicios a la capa de presentación a algún posible wrapper que implemente web services en caso de ser necesario. Es el punto de acceso a toda la lógica de la aplicación. Se comunica a través de Interfaces lo que ayuda a la testeabilidad del sistema.
* Model: Es el dominio de la aplicación que contiene todas las clases que administran el acceso a datos, repositorios y clases del modelo.
* Services.Test, Framework.Test: Estos proyectos son utilizado para crear los tests unitarios de la aplicación. El primero deberá testear la correctitud de los servicios y el segundo permitirá testear los controladores.
* Framework: Este proyecto contiene la implementación del patrón MVC del framework.



**3. Estructura de la Solución MVC (proyecto Framework)**

La solución MVC posee determinadas características que es conveniente detallar a continuación:

**3.1 Filtros**

* HandleAndLogError: Permite loguear el error utilizando log4net previamente inyectado en la clase WebModule. Si el error proviene de una llamada Ajax entonces retorna el resultado en formato JSon.
* RoleAuthorize: Una vez logueado el usuario –autenticado- puede que el usuario tenga un rol determinado, este filtro funciona igual que Authorize pero redirecciona a una vista “Acceso Denegado” en vez de redireccionarlo al login, que es el comportamiento default de Authorize.
* UserTrackerLog: Permite hacer un trace de los datos de la acción sobre la que se agrega el filtro. En el archivo de configuración de log4net se puede determinar donde se logueará, actualmente lo hace a la base de datos.

**3.2 Global.asax**

En esta clase se configuran los aspectos más importantes de la aplicación como la configuración del inyector de dependencias, registración de áreas, minificación de estilos a través de bundles –nuevo feature de MVC 4-, etc.

AreaRegistration.RegisterAllAreas();

SetupContainer(); // Configura todas las clases del inyector de dependencias

WebApiConfig.Register(GlobalConfiguration.Configuration);

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters); // registra filtros que se aplicarán a lo largo de toda la aplicación

RouteConfig.RegisterRoutes(RouteTable.Routes); // configura patrones de ruteo

BundleConfig.RegisterBundles(BundleTable.Bundles); // minifica css y jscript

AuthConfig.RegisterAuth(); // registra seguridad mediante OpenID

XmlConfigurator.Configure(new System.IO.FileInfo(Server.MapPath(@"\\_config\log4net\log4net.xml"))); // configura el componente log4net.

La configuración citada en el párrafo anterior es aplicada sobre las clases ubicadas en App\_Start, estas clases deberán ser customizadas por el developer.

Un ejemplo de configuración es la clase AuthConfig ubicada en App\_Start que permite agrega autenticación por OpenId utilizando proveedores como Facebook o Twitter.