



Universitat Autònoma  
de Barcelona

# GENERACIÓN AUTOMÁTICA DE CUADROS DE MANDO VÍA WEB

Memòria del projecte  
d'Enginyeria Tècnica en  
Informàtica de Gestió  
realitzat per  
*Xavier Vidal Zamarreño*  
i dirigit per  
*Xavier Verge Mestre*

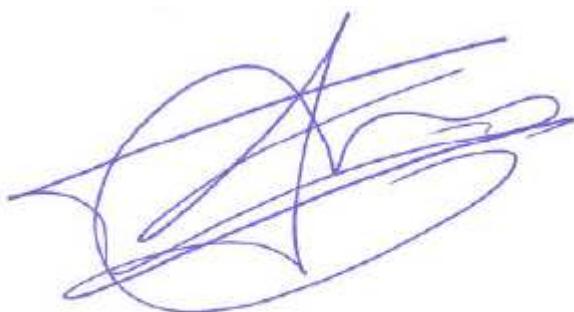
**Escola d'Enginyeria**

Sabadell, juny de 2012

[El/La] sotasignat, **Xavier Verge Mestre**,  
professor de l'Escola d'Enginyeria de la UAB,

**CERTIFICA:**

Que el treball al que correspon la present  
memòria  
ha estat realitzat sota la seva direcció per  
**Xavier Vidal Zamarreño**  
I per a que consti firma la present.  
Sabadell, **Juny de 2012**

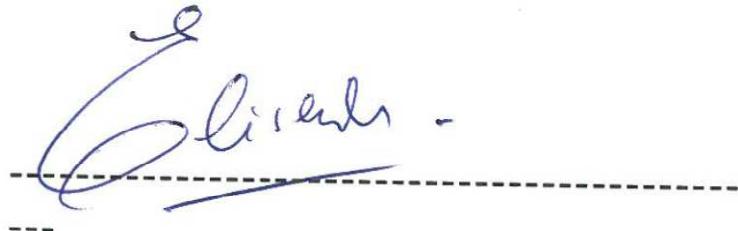


-----  
-  
Signat: **Xavier Verge Mestre**

La sotesignat, ***Elisenda Royo tutora***,  
de ***arvato services Iberia***,

**CERTIFICA:**

Que el treball al que correspon la present  
memòria  
ha estat realitzat sota la seva supervisió per  
***Xavier Vidal Zamarreño***  
I per a que consti firma la present.  
Sabadell, ***juny de 2012***

A handwritten signature in blue ink, appearing to read "Elisenda Royo", is written over a dashed horizontal line.

Signat: ***Elisenda Royo***

**FULL DE RESUM - PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA**

**Títol del projecte:** Generación automàtica de cuadros de mando via Web.

**Autor:** Xavier Vidal Zamarreño

**Data:** Juny de 2012

**Tutor:** Xavier Verge Mestre

**Titulació:** Enginyeria Tècnica en Informàtica de Gestió

**Paraules clau** (mínim 3)

- Català: Cuadro de comandament, jQuery, .NET, SQL Server, Intel·ligència de negoci
- Castellà: Cuadro de mando, jQuery, .NET, SQL Server, Inteligencia de negocio.
- Anglès: dashboards, jQuery, .NET, SQL Server, Business Intelligence

**Resum del projecte** (extensió màxima 100 paraules)

- **Català:** El projecte consta d'una aplicació web destinada a generar quadres de comandament per diferents clients d'una forma àgil i dinàmica reduint d'aquesta manera els temps d'implementació d'una eina d'aquestes característiques. Aquesta aplicació es desenvoluparà utilitzant .NET per la part del servidor i jQuery & googleMaps API per la part del client, en referència als informes i les bases de dades estaran implementats amb Microsoft SQL Server 2008 R2
- **Castellà:** El proyecto consta de una aplicación web destinada a generar cuadros de mandos para distintos clientes de una manera ágil y dinámica reduciendo así los tiempos de implementación de una herramienta de estas características. Dicha aplicación será desarrollada bajo .NET para el lado del servidor y jQuery & google Maps API para el lado cliente, por lo que hace a los informes y la base de datos todo estará implementado con Microsoft SQL Server 2008 R2.
- **Anglès:** This project is a web application used to generate dashboards for different customers in a agile and dynamic way reducing implementation of this type applications. The application will be developed using .NET for the server side and jQuery & googleMaps API for the client side. The reports and the database are implemented using Microsoft SQL Server 2008 R2.

# Tabla de contenido

1	Presentación.....	8
1.1	Antecedentes.....	8
1.2	Introducción.....	8
1.3	Motivaciones.....	9
1.4	Estructura de la memoria.....	9
2	Estudio de Viabilidad .....	11
2.1	Presentación.....	11
2.2	Descripción de la situación actual.....	11
2.3	Objetivos. ....	12
2.4	Estado del arte .....	13
2.5	Descripción del sistema a realizar .....	14
2.5.1	Descripción del escenario de ejemplo .....	15
2.5.2	Descripción de la aplicación .....	15
2.6	Planificación del proyecto .....	16
2.6.1	Modelo de desarrollo. ....	16
2.6.2	Recursos .....	16
2.7	Fases del proyecto .....	18
2.8	Análisis de coste – Beneficio .....	19
2.8.1	Costes. ....	19
2.8.2	Beneficios.....	20
2.8.3	Coste vs Beneficio .....	20
2.9	Análisis de riesgos.....	20
2.10	Conclusiones .....	21
3	Análisis .....	22
3.1	Requerimientos .....	22
3.2	Requerimientos funcionales.....	22
3.2.1	Usuarios y seguridad .....	22
3.2.2	Menú .....	22
3.2.3	Filtros .....	22
3.3	Abrir Informe.....	23
3.3.1	Abrir mapa.....	23
3.3.2	Repositorio .....	23
3.3.3	Modulo administración.....	23
3.4	Requerimientos no funcionales.....	24

3.4.1	Requerimientos de diseño .....	24
3.4.2	Requerimientos de accesibilidad .....	24
3.5	Casos de uso.....	25
3.5.1	Caso de uso principal .....	25
3.5.2	Especificación de los casos de uso .....	26
4	DISEÑO.....	38
4.1	Acceso a la aplicación .....	38
4.2	Pantalla principal .....	38
4.3	Menú Principal.....	39
4.4	Panel de filtros.....	41
4.5	Área de trabajo, Informes y Mapas .....	42
4.6	Área de administración .....	43
5	CODIFICACIÓN.....	44
5.1	Introducción .....	44
5.2	Tecnologías utilizadas. ....	44
5.2.1	.NET (c#) .....	44
5.2.2	Microsoft SQL Server 2008 R2 .....	44
5.2.3	Javascript .....	45
5.3	Descripción de los módulos .....	45
5.3.1	Pantalla principal .....	45
5.3.2	Menú Baldosas.....	46
5.3.3	Filtros.....	47
5.3.4	Generar Mapa .....	48
5.3.5	Generar Informe .....	49
5.3.6	Nuevo Usuario.....	50
5.3.7	Cambio Contraseña .....	50
6	Pruebas .....	51
7	Conclusiones y ampliaciones .....	54
7.1	Conclusiones .....	54
7.2	Desviaciones .....	55
7.3	Ampliaciones .....	56
8	Bibliografía.....	57
9	Agradecimientos .....	58

# 1 Presentación

---

## 1.1 Antecedentes

Desde hace tiempo en el departamento de Business intelligence de nuestra empresa nos vemos con la necesidad de mostrar los resultados de muchos de nuestros proyectos a los clientes con un cuadro de mando para que ellos puedan analizar los resultados obtenidos de forma rápida y precisa. Por este hecho ya hace más de 7 años que trabajamos con herramientas de *reporting* para mostrar los resultados de los proyectos. Inicialmente, para cada cliente se realizaba una herramienta única o se le entregaban los resultados finales en un Excel, pero con este sistema los costes asociados a cada proyecto se veían incrementados de una forma substancial o, en el caso de las entregas en Excel, se daba por cerrado el proyecto por lo que la vinculación con el cliente era menor.

En 2009 se decide dar un paso al frente y desarrollar una herramienta que pudiera dar salida a nuestros proyectos de *reporting* de una forma más dinámica con el fin de ahorrar costes y poder bajar así los precios dentro de un mercado cada día más duro debido a la crisis que empezaba a reinar.

En 2012 y después de años de éxitos de la anterior versión de la aplicación, se decide modernizarla y dotarla de mucha más funcionalidad. Al principio se trató de implementar pequeñas mejoras y un nuevo *front end* más moderno para adaptarla a los nuevos tiempos. Aunque, finalmente, se decide hacerla de nuevo escogiendo jQuery para dotar a la aplicación de una experiencia de usuario mucho más agradable.

## 1.2 Introducción

El proyecto desarrollado es un cuadro de mando web de generación dinámica basado en la tecnología de Microsoft Reporting Services 2008 R2. El principal objetivo del proyecto es facilitar la implementación de este en una gran variedad de clientes.

Esta aplicación consta de tres bases de datos (seguridad, *reporting* y cliente) así como de multitud de informes genéricos o personalizados para mostrar al cliente el funcionamiento de su negocio. Aunque todo el proyecto global ha sido desarrollado por el alumno en esta memoria, nos vamos a centrar sólo en el desarrollo del cuadro de mando para así adecuar los timings del proyecto a un proyecto de final de carrera.

La aplicación consta de una página de autentificación desde la cual el cliente se identificará para poder acceder a sus informes, una vez autenticado la aplicación generará la página principal para el cliente en cuestión, desde ésta, dicho cliente podrá acceder a los distintos informes organizados por apartados.

El objetivo de este desarrollo, es que una vez finalizado, para implementar un nuevo cliente o para añadir nuevos informes a un cliente existente, sólo tengamos de desarrollar el informe y una vez colgado en el reportServer el cuadro de mando se nutra de la información de esta para poder mostrarlo al cliente en la interfaz web.

## 1.3 Motivaciones

Durante los últimos 4 años hemos aumentado de forma exponencial el número de clientes al que el servicio que les ofrecemos necesita de un cuadro de mando. Hasta la fecha los cuadros de mando que habíamos implementado para nuestros clientes eran muy rígidos y se estaban quedando obsoletos, debido a esto cada vez que un cliente pedía un nuevo informe suponía un gran coste para la empresa, ya que se tenía que modificar ficheros de configuración y en algunas ocasiones incluso crear nuevas páginas **aspx** para albergar los informes. Además estas aplicaciones se empezaban a ver anticuadas y aunque los datos que proporcionábamos a los clientes eran muy buenos, resultaba difícil vender un proyecto en el que la herramienta final que veía el cliente se veía fea y anticuada.

A nivel personal he decidido entregar este proyecto como proyecto de final de carrera porque representa muy bien cuál ha sido mi trabajo desde que termine la universidad y me permite mostrar de forma clara cuales son mis conocimientos, o al menos parte de ellos, ya que en el PFC no se va a mostrar todo el proyecto y en la empresa en que trabajo no sólo me dedico a esta tipología de proyecto.

Todos estos factores motivaron el desarrollo de la nueva aplicación que se explica en esta memoria y la posterior entrega como PFC.

## 1.4 Estructura de la memoria

Esta memoria está compuesta por 7 capítulos, la estructura de los cuales es la siguiente:

### **Presentación**

En este capítulo se describe el problema que ha motivado el desarrollo del proyecto, también se explica la aplicación a grandes rasgos y se sitúa al lector en el contexto de la misma, se exponen las motivaciones que han sido claves para el desarrollo de este proyecto y la estructura que va a seguir esta memoria.

### **Estudio de Viabilidad**

En esta sección, se situará al lector en el contexto que ha llevado al desarrollo de esta aplicación. Se pretende realizar un análisis exhaustivo del proceso de desarrollo de la aplicación que nos permita saber antes de empezar si la idea que tenemos sobre la aplicación va a solucionar el problema que se plantea y si los costes y los riesgos derivados de este desarrollo son asumibles por los beneficios obtenidos una vez concluida la aplicación. Por último en este estudio se va a analizar el tiempo necesario para concluir la aplicación y si este cuadra con el tiempo que se le supone a un proyecto de final de carrera.

### **Análisis**

Para empezar este capítulo se van a exponer los requerimientos tanto funcionales como no funcionales que debe cumplir la aplicación para llegar al resultado deseado, a continuación se analizarán los distintos módulos de la aplicación y se mostrarán los diagramas de flujo que muestran el funcionamiento de los mismos.

## **Diseño**

Este capítulo mostrará el diseño final de la aplicación explicando a grandes rasgos la interfaz de usuario y su comunicación con el sistema de la aplicación y los orígenes de datos usados por ésta.

## **Codificación**

Aquí se mostrará al lector como se ha desarrollado la aplicación explicando uno por uno los distintos módulos que la forman y describiendo de forma general las funciones que los forman.

## **Pruebas**

En el capítulo de pruebas se describe la batería de pruebas que hay que realizar al finalizar el desarrollo de la aplicación para garantizar su estabilidad y buen funcionamiento.

## **Conclusiones y ampliaciones**

Para finalizar se describen las conclusiones del proyecto una vez finalizado y se analizan las posibles ampliaciones futuras, también se va evalúa la planificación del proyecto para analizar las desviaciones y dar las explicaciones pertinentes.

## **2 Estudio de Viabilidad**

---

### **2.1 Presentación**

El proyecto que nos ocupa consiste en el desarrollo de un cuadro de mando para la gestión del negocio de varios de nuestros clientes, este cuadro de mando también ejercerá de salida de muchos proyectos paralelos que pueda hacer nuestra empresa para los clientes que lo tengan.

Para conseguirlo vamos a rediseñar la herramienta que actualmente servimos a nuestros clientes con tal de adaptarla a las nuevas tecnologías y dotarla de mayor rendimiento y seguridad.

En esta sección de la memoria se va a evaluar la viabilidad de realizar dicho proyecto haciendo un análisis de los costes que supone el desarrollo y la implantación de la solución y comparándolos con los beneficios obtenidos por la empresa.

Como dentro de la empresa éste se considera un desarrollo vinculado al I+D, el estudio de viabilidad se centrará en el análisis coste beneficio para la empresa y no en el análisis coste beneficio del cliente final de la aplicación. Gracias a la experiencia obtenida en el desarrollo de las anteriores versiones de esta herramienta podemos hacer una buena estimación de las jornadas que se necesitaran para desarrollar el proyecto así como de los recursos implicados en su desarrollo. También se llevará a cabo un análisis de riesgo del desarrollo y se analizará el estado del arte para poder decidir finalmente si el proyecto es viable.

### **2.2 Descripción de la situación actual.**

Actualmente nos encontramos con una serie de clientes que tienen funcionando varias aplicaciones antecesoras de la que nos atañe. La vinculación de estos clientes con nuestra empresa es muy grande ya que la aplicación nos permite formar un vínculo de servicio permanente con estos clientes y poder venderles varios proyectos los cuales podemos dar salida mediante la aplicación. Por otro lado se están generando multitud de nuevas ofertas con nuevos clientes que van a necesitar la aplicación.

Dentro del marco de esta aplicación se les están ofertando al cliente multitud de proyectos relacionados con la gestión de su cartera de clientes y con el mayor control y conocimiento de ésta, también gracias a esta aplicación se les ayuda a controlar el resto de métricas de su negocio consiguiendo de esta manera una herramienta de control centralizada de todos los elementos de su negocio: Clientes, Productos, tiendas, Ventas, Campañas y Stocks.

Entre los proyectos de nuestra empresa que se están ofertando a partir de esta aplicación nos encontramos con la implantación de un club de clientes, el lanzamiento de campañas de marketing, el análisis avanzado de sus datos para poder realizar métricas de valor o de riesgo mediante la minería de datos, la normalización y

georeferenciación de los datos de los clientes y el enriquecimiento de los datos de estos a partir de una segmentación que tenemos de la geografía española.

Todos estos proyectos, unidos a la funcionalidad que ya de por si da un cuadro de mando, dan una herramienta muy valiosa y potente, por esto se cree indispensable cambiar el front-End de dicha herramienta para poderla dotar de una apariencia acorde con todo lo que lleva detrás. Podríamos decir que en estos momentos nos encontramos con un Ford fiesta con motor Ferrari. Los clientes ven el Ford Fiesta y sólo se dan cuenta del motor que lleva dentro cuando lo prueban, por ello necesitamos una carrocería acorde con el motor.

Por todo lo mencionado antes se decide hacer un remodelado total de la aplicación para dotarla de mayor funcionalidad y seguridad así como modernizarla y estandarizarla con el resto de aplicaciones de nuestra empresa, para dar una imagen más corporativa. Una vez desarrollada la nueva aplicación se procederá a la migración de todas las versiones anteriores a la nueva aplicación generando una acción de retención a los clientes debido a la mejora del servicio, así como una mayor facilidad de mantenimiento de todos los clientes por nuestra parte.

## 2.3 Objetivos.

Como ya se ha comentado en el anterior apartado, el objetivo primordial de este proyecto es implementar un cuadro de mando moderno y con muchas más funcionalidades del que ahora disponemos, por eso necesitamos rediseñar por completo el cuadro de mando que actualmente servimos a nuestros clientes.

Con tal de cumplir con este objetivo principal, hay unos puntos que no podemos perder de vista. Para empezar deberemos adecuar el proyecto a las nuevas tecnologías y dotarlo de una funcionalidad suprimiendo así importantes deficiencias que puede tener nuestra aplicación en frente a otras soluciones que nos proporciona el mercado. Otro un objetivo importante del proyecto es hacer que el nuevo look and feel de la aplicación se aproxime al máximo al look and feel de otras aplicaciones de la empresa para poder ofrecer a nuestros clientes una imagen corporativa más unificada y facilitar el cross-selling entre las distintas aplicaciones que tenemos a su disposición.

Para poder cumplir los objetivos que nos marcamos, la aplicación deberá cumplir una serie de requisitos que se detallan a continuación:

- Capacidad de implantación y ampliación de los proyectos para los nuevos clientes de forma fácil y rápida.
- Visualización de todos los tipos de filtros posibles en los informes.
- Capacidad para poder visualizar varios informes a la vez pudiendo así comparar datos.
- Facilitar la creación de mapas temáticos para mostrar información de carácter geográfico.
- Front End agradable a la vista y sencillo de utilizar para dotar al usuario de una mejor experiencia de uso de la aplicación.

- Modular: La aplicación tiene que poder ser ampliada en el futuro con posibles mejoras de una manera eficaz
- Repositorio de sesión. El usuario debe ser capaz de recuperar cualquier informe ejecutado durante la sesión de trabajo sin necesidad de volver a ejecutarlo de nuevo.
- Permisos. La aplicación tiene de ser capaz de gestionar distintos usuarios con distintos permisos con tal de poder diferenciar roles dentro del cliente y poder decidir que puede y que no puede hacer cada usuario.
- Gestión de usuarios. La aplicación debe tener un modulo de gestión de usuarios para facilitar las tareas del administrador.

## 2.4 Estado del arte

A nivel de datos y de integración de servicios la aplicación no tiene competencia en el mercado y tratar de compararla con otros cuadros de mando que existen en el mercado no tiene sentido ya que lo que estamos vendiendo en la empresa es un servicio en el cual el cuadro de mando viene incluido. En este apartado centrare la atención al estado de esta misma aplicación en el momento en el que se decide llevar a cabo la remodelación de la misma así como la situación actual de la empresa que tiene mucho peso en la decisión de desarrollar este proyecto.

Como ya se ha comentado con anterioridad, actualmente tenemos 3 versiones distintas de aplicaciones predecesoras a la que nos ocupa en funcionamiento en varios clientes, amén de multitud de nuevas ofertas para implementar el proyecto en curso a nuevos clientes.

La gestión y el mantenimiento de las antiguas aplicaciones es mucho más costosa que la que se pretende en la nueva aplicación. Además la funcionalidad que aportan las versiones anteriores ha quedado obsoleta dentro de un mercado con una competencia cada día más dura que nos obliga a mejorar las prestaciones y a reducir los costes para poder ser competitivos.

Internamente dentro de la empresa se está llevando a cabo la modernización de todas las soluciones web que estamos prestando a los clientes ya que como el proyecto que nos ocupa, otras soluciones de la empresa se veían en mayor o menor grado afectadas por un punto de obsolescencia.

Las predecesoras de esta aplicación están desarrolladas bajo .NET para la parte web y en SQL Server 2008 para la base de datos y el Reporting, dichas herramientas de programación fueron escogidas en su momento después de un exhaustivo análisis del mercado.

### *¿Por qué .Net?*

Nos decantamos por el .NET debido a la misma estructura de la empresa ya que todos los servidores de los que disponemos funcionan bajo Windows server con IIS y dentro la empresa tenemos el contrato Select con Microsoft de manera que podemos trabajar fácilmente y de una forma económica con tecnología Microsoft, incluyendo la última

versión de Visual Studio. Además .NET nos proporciona las mejores herramientas de comunicación con SQL Server 2008 y Reporting Services 2008.

#### *¿Por qué SQL Server 2008 & Reporting Services?*

En nuestra empresa ya habíamos trabajado anteriormente con Reporting Services en su primera versión para SQL Server 2000, aunque esa versión no nos convencía en el momento del desarrollo de nuestro primer cuadro de mando, por lo que realizamos un estudio de mercado para buscar una herramienta de reporting que cumpliera los requisitos que deseábamos. Después de analizar varias posibilidades la finalista fue Business Objects, pero antes de empezar con el desarrollo decidimos analizar la *Release pre candidate de SQL Server 2008* para ver las mejoras que aportaba. Al final la conclusión fue que Business Objects era una herramienta superior en prestaciones a Reporting Services 2008 pero esta última venía integrada ya con la licencia de base de datos al contrario de Business Objects que tenía una coste de unos 20.000 € por cada 5 licencias. Teniendo en cuenta estos costes y viendo que Reporting Services no sólo cubría nuestros requisitos sino que además era una tecnología con la que estábamos más familiarizados y que nos salía mucho más económica nos decantamos finalmente por la solución de Microsoft.

Estas elecciones tendrán un claro impacto en la nueva versión de la aplicación ya que para facilitar la migración de todos los clientes no podemos hacer un cambio radical de la tecnología utilizada, por este motivo se mantiene .Net como lenguaje de programación para la parte del servidor.

Por lo que se refiere a base de datos y reporting se decide migrar a SQL Server 2008 R2 y Reporting Services 2008 R2 los cuales son claramente superiores a sus predecesores y de fácil migración. Con esta nueva tecnología ganamos mucho dinamismo en el proceso de datos espaciales y geográficos así como un nuevo abanico de opciones para hacer nuestros informes más vistosos y potentes.

Para el lado, cliente de la aplicación la empresa a petición de los otros equipos de desarrollo proponen Silverlight de Microsoft para dotar a la aplicación de un apartado visual más moderno. Por mi parte propongo no usar dicha tecnología con la que tengo muchas reticencias técnicas y me decanto por jQuery, decisión que al final y vistos mis argumentos se acepta por parte de la dirección de la empresa y que más adelante, en el apartado dedicado a las tecnologías utilizadas en el proyecto, defenderé.

## **2.5 Descripción del sistema a realizar**

La aplicación que se presenta será una plataforma multicliente, pero para describirla necesariamente la tenemos de poner en el contexto de un cliente ya que las capas más bajas del proyecto (Base de Datos, ETL's, Informes) van estrechamente ligadas a cada cliente siendo el FrontEnd la parte común a todos.

### 2.5.1 Descripción del escenario de ejemplo

Conjuntamente con el cliente se definirán los formatos de los archivos que van a extraer de su base de datos para luego colgar en nuestra FTP y que nosotros podamos recoger para cargar en la base de datos que tendremos en hosting.

Nuestro cuadro de mando se va a nutrir de la base de datos alojada en nuestros servidores de manera que nosotros decidiremos cual es la mejor estructura de la base de datos con tal de visualizar la información del cliente de forma rápida y eficaz.

Una vez definido el diseño de la base de datos en base a los datos aportados por el cliente y a la información que quiere sacar este de los mismos, el siguiente paso es definir las categorías de informes que van a necesitar visualizar y empezar a trabajar en ellos.

El caso de ejemplo, el cliente va a tener unas pocas tiendas repartidas por la geografía Española y una base de datos de clientes de unos 50.000 clientes con un crecimiento de 2000 clientes mensuales, el cliente va a tener unos mil productos repartidos por temporadas y de momento vamos a cargar un histórico de 3 años de ventas. El cliente va a tener un par de campañas de generación automática y lanzará cada mes una campaña personalizada

Como el volumen de la información es mínimo nos bastará con generar los informes a partir de consultas SQL evitando así tener de recurrir a bases de datos OLAP.

### 2.5.2 Descripción de la aplicación

Una vez pasada la autentificación inicial, el cliente tendrá ante él un marco de trabajo desde el cual debe ser capaz de acceder a todos los informes para los que tiene permiso debidamente organizados de manera que llegue a ellos de una forma natural para él. Dicho usuario por lo general tendrá conocimientos avanzados de ofimática ya que normalmente habrá estado usando hojas de cálculo durante mucho tiempo para desempeñar la función que ahora realiza la aplicación por él.

Mediante la navegación por un menú lateral del área de trabajo el usuario escogerá el informe que quiera visualizar, una vez hecha la elección podrá controlar la información que visualizará en el informe a partir de los filtros del mismo, por eso la aplicación debe ser capaz de identificar los distintos tipos de filtro que puede contener un informe para mostrarlos al usuario.

Los informes pueden contar con 5 tipos básicos de filtro:

- **Seleccionables:** Son aquellos filtros que se deberá escoger una opción de una lista, la aplicación los mostrará mediante una lista desplegable.
- **Multi-seleccionables:** Como en el caso anterior tendremos una lista de opciones pero en esta ocasión podremos seleccionar varios elementos de la lista. Estos filtros se mostrarán mediante una lista de checkbox con la opción de seleccionar todos.
- **Fecha:** Son filtros que contienen una fecha. Se mostrarán con un input asociado a un calendario para introducir la fecha, estos filtros pueden estar

asociados en grupos de a dos para seleccionar un rango de fechas, en estos casos la aplicación debe detectarlo para usar un control de selección de rangos de fechas.

- **Texto:** Serán aquellos filtros que no tengan opciones seleccionables, se mostrarán en un cuadro de introducción de texto
- **Internos:** En un informe pueden existir filtros de carácter interno que use el informe pero que no deban ser visualizados por el usuario. La aplicación se encargará de identificarlos y no mostrarlos por pantalla.

Como se ha comentado en anteriores apartados el proyecto se va a desarrollar con Microsoft Visual Studio 2010 y SQL Server 2008 R2. Por el lado del servidor se va a utilizar el lenguaje C#, para programar la parte del cliente se utilizarán jQuery, y la API de Google Maps, para algunas funciones simples del lado de cliente también se va a usar JavaScript.

## 2.6 Planificación del proyecto

### 2.6.1 Modelo de desarrollo.

El modelo de desarrollo se dividirá en tres fases:

- Desarrollo portal web. Esta fase se llevará a cabo de forma lineal.
- Implementación nuevo cliente: En esta fase se desarrollarán las ETL's y se creará la base de datos del cliente.
- Diseño de Informes: En esta fase se crearán los informes, a petición del cliente. Esta fase puede alargarse en el tiempo con la inclusión de nuevos informes y podrá ser llevada a cabo por varias personas de forma paralela.

En esta memoria se describe la primera fase la cual dividiéremos por módulos bien diferenciados de manera que nos permita obtener una aplicación fácilmente escalable.

### 2.6.2 Recursos

#### ***Humanos:***

El proyecto será desarrollado por una única persona que se encargará del desarrollo del portal web en la primera fase, así como de las reuniones e implantación de la segunda fase. Esta persona también se encargará del inicio de la tercera fase pudiendo ceder en el futuro el mantenimiento y los futuros informes a otros recursos de la empresa. Esta persona (el estudiante) ya fue el encargado en su día de desarrollar las aplicaciones predecesoras a ésta por lo que está altamente familiarizado con las tecnologías utilizadas en el proyecto, a excepción de jQuery y Google Maps API las cuales aprenderá a usar durante el desarrollo del mismo.

### **Hardware:**

El proyecto se va a implementar utilizando una arquitectura de tres servidores, se utilizará dicha arquitectura porque es la que mejor garantiza la seguridad y la estabilidad de todo el conjunto de la aplicación, a continuación se describe el uso que se va a dar a cada uno de estos servidores:

- **Servidor BD:** En este servidor estarán alojadas las bases de datos utilizadas por la aplicación en una instancia de SQL Server 2008 R2. Para el funcionamiento de la aplicación final en cliente son necesarias 3 bases de datos:
  - **BD aplicación:** Contiene las tablas necesarias para el funcionamiento de la aplicación tales como usuarios, Roles etc....
  - **BD reporting:** Es la base de datos que usa Reporting services para guardar todos sus datos, la aplicación también se nutrirá de ella para poder generar la estructura del Front-End de forma dinámica.
  - **BD Cliente:** En dicha base de datos se alojaran los datos del cliente de los que se nutrirán los informes mostrados en la aplicación.
- **Servidor IIS:** Dicho servidor estará conectado a internet y será el encargado de alojar el portal web al que se conectarán los clientes desde donde quieran, la aplicación quedará instalada en un IIS6.
- **Servidor de aplicaciones:** En este servidor tendremos alojadas las aplicaciones utilizadas, en el caso del proyecto que nos ocupa será la aplicación de informes Microsoft Reporting Services 2008 R2.

Estos servidores son realmente máquinas virtuales de manera que si por el volumen de clientes a tratar se necesitan futuras ampliaciones de maquinaria, este cambio va ser rápido y sencillo. Todos los servidores corren bajo el sistema operativo Windows Server 2008.

### **Tecnologías:**

Como ya se ha mencionado en capítulos anteriores de la memoria, para el desarrollo de este proyecto se ha usado una mezcla de tecnologías de Microsoft y soluciones open Source existentes en el mercado. En el estado del arte ya se han dado las explicaciones pertinentes a las tecnologías Microsoft ya que vienen heredadas por la aplicación predecesora a la que nos ocupa. En este subapartado nos vamos a centrar en las nuevas tecnologías usadas en el proyecto, dichas tecnologías son jQuery y la API de googleMaps.

#### **jQuery**

Para el desarrollo del lado del cliente nos decidimos a utilizar jQuery ya que siguiendo su lema, “*write less, do more*”, nos sirve para, de una manera simple y muy natural, dotar a nuestra aplicación de un potencial y un diseño muy superior a la tecnología usada hasta la fecha con un coste de aprendizaje muy bajos. En la anterior aplicación se utilizó .Net con algunas pinceladas de javascript para la parte de cliente, eso lo conseguíamos utilizando los controles proporcionados por la librería AjaxWebtoolKit. Dicha librería funcionaba utilizando los updatePanels de .Net para la comunicación asíncrona con el servidor. Esta tecnología, aunque es de muy fácil implementación, resulta horriblemente ineficiente ya que aunque se venda una comunicación asíncrona con el servidor permitiendo un refresco parcial de la página, lo que realmente hace internamente es pasar al servidor la definición de toda la página refrescando luego sólo

la parte contenida en el updatePanel. Esto hace aumentar de una manera inaceptable el flujo de datos entre el cliente y el servidor ralentizando demasiado la aplicación. Por si esto fuera poco, como realmente se está haciendo un postback de la página, en muchas ocasiones el uso de updatePanels hacía responder de forma inestable scripts propios. Por todo esto se decide buscar una tecnología alternativa para las operaciones en el cliente y para facilitar la comunicación asíncrona con el servidor, jQuery nos proporciona una serie de funciones ajax que junto con JSON nos permiten comunicarnos con el servidor de manera fácil y eficaz, además nos proporciona un incontable número de controles creados por el mismo jQuery los cuales encontramos en las librerías de jQuery UI así como muchos otros controles creados por una comunidad en constante movimiento.

En este sentido jQuery gana, bajo mi humilde opinión, el pulso con Silverlight (herramienta deseada por otro equipo de programación) ya que esta última es una especie de Flash propietaria de Microsoft para el cual Microsoft ya ha anunciado que detiene el desarrollo de nuevas versiones y que dejará de darle soporte. Si a esto le sumamos el hecho de que las pocas cosas en las que Silverlight podría ser superior a jQuery, como el manejo de archivos multimedia, no nos sirve para nada en este proyecto, me decido finalmente por el uso de jQuery.

**API GoogleMaps.** Debido a que, tanto a nivel visual como informativo, los mapas temáticos aportan un gran valor añadido a la aplicación se decide no usar los que vienen por defecto en Reporting Services 2008 R2 ya que son de nueva implantación en la herramienta y aun están muy limitados. Por este hecho decidimos generar temáticos utilizando la tecnología que nos brinda googleMaps ya que como casi todo el mundo está muy habituado a utilizar google maps hará que la aplicación sea mucho más amigable para el usuario y le dará una potencial extra a los informes geográficos.

## JSON

Es un formato de intercambio de datos parecido a XML, será el formato utilizado en la comunicación cliente servidor. Es algo más ligero que XML aunque en este caso la elección viene dada por la mejor integración de este con jQuery.

## 2.7 Fases del proyecto

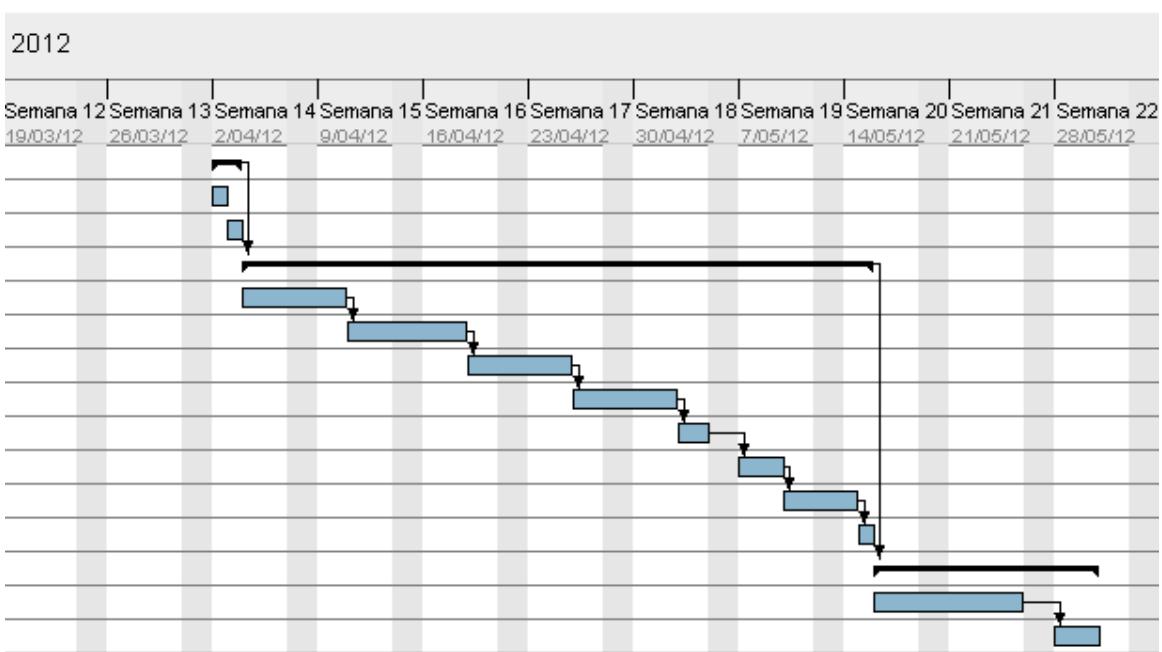
La planificación del proyecto ha sido realizada con GranttProject. Según la estimación hecha en el estudio de viabilidad nos indica que deberíamos poder realizar el proyecto en 32 jornadas.

En la siguiente tabla podemos ver las fases de desarrollo del proyecto con sus distintos *timings*, este proyecto sólo se basa en la Fase 1 de dicho diagrama, aunque aquí se muestra la información hasta la implantación de la solución en un primer cliente.

# GANTT project

Nombre	Fecha de inicio	Fecha de fin	Duración
• Inicio del proyecto	2/04/12	3/04/12	2
• Análisis de Requerimientos	2/04/12	2/04/12	1
• Estudio de Viabilidad	3/04/12	3/04/12	1
* Fase 1:	4/04/12	15/05/12	30
• Desarrollo del Menú	4/04/12	10/04/12	5
• Desarrollo de la carga/lectura de filtros	11/04/12	18/04/12	6
• Desarrollo apertura Informe	19/04/12	25/04/12	5
• Desarrollo apertura mapa	26/04/12	2/05/12	5
• Desarrollo repositorio	3/05/12	4/05/12	2
• Desarrollo Modulo Administración	7/05/12	9/05/12	3
• Batería De Pruebas/Corrección de Errores	10/05/12	14/05/12	3
• Puesta En Producción	15/05/12	15/05/12	1
• Fase 2:	16/05/12	30/05/12	11
• Diseño Base de Datos	16/05/12	25/05/12	8
• Diseño&Desarrollo ETL carga	28/05/12	30/05/12	3

En la imagen adjunta podemos ver el diagrama de Gantt del proyecto.



## 2.8 Análisis de coste – Beneficio

### 2.8.1 Costes.

En este análisis nos vamos a centrar en los costes y los beneficios que nos aporta la fase 1 ya que es la que realmente entra dentro del proyecto analizado en esta memoria.

En este sentido podemos ver que el coste del proyecto se centra en el recurso humano de la fase 1 la cual tiene una duración de 23 jornadas. Si tenemos en cuenta un sueldo estándar de 24000 € brutos para el analista que desarrolla la aplicación podemos deducir que el coste de este proyecto para la empresa será de unos 3000 € extraídos del salario del trabajador durante estas 23 jornadas más las vacaciones acumuladas, el gasto en seguridad social que tiene la empresa para este trabajador y la parte de *overflow* asociada a dicho trabajador durante estas jornadas.

### 2.8.2 Beneficios.

Los beneficios que nos aporta este proyecto son múltiples. Por una parte nos permite reducir la implantación de la herramienta en un cliente al desarrollo de la fase 2 y 3, de esta manera se reducen mucho los *timings* y los costes de desarrollo, por otra parte también tiene un impacto directo sobre los *timings* de la tercera fase que quedan reducidos de forma drástica.

A nivel indirecto la implantación de esta aplicación en los clientes genera una serie de sinergias con los clientes que la utilizan y que a la larga nos acaban contratando mas proyectos generando así un beneficio mayor.

Si tenemos en cuenta que esta aplicación va ser instalada de inicio a 5 clientes en formato de migración de la versión anterior y que tenemos 3 o 4 ofertas en curso para contratar esta aplicación, el beneficio sacado de la aplicación es evidente.

### 2.8.3 Coste vs Beneficio

Aunque por razones de confidencialidad no se han dado cifras concretas en el análisis de beneficios es evidente que los ingresos superan con creces a los costes del proyecto, siendo además incrementados por cada cliente nuevo que contrate el servicio de la aplicación.

Para tener un ejemplo cuantitativo del claro balance positivo del beneficio, sólo resta decir que la primera implementación de la aplicación predecesora a la que nos ataña tuvo una duración de 5 meses. Con la nueva aplicación reducimos la implantación en cliente a 44 días (2 meses) por lo que obtenemos un beneficio de 3 meses por proyecto o lo que es lo mismo,  $3 \text{ meses} * 3.000 = 9.000 \text{ €}$  de ahorro por cliente.

Con este ejemplo dejamos claro que el proyecto es claramente rentable.

## 2.9 Análisis de riesgos

Los riesgos asociados al desarrollo del proyecto y que podrían hacer peligrar su *timing* o su finalización son los siguientes:

**Flujo de trabajo:** Es un riesgo vinculado al hecho de que el proyecto forme parte del I+D de la empresa y por el cual muchos proyectos han caído en el olvido, se centra básicamente en la importancia que tiene en el mundo empresarial el cliente. Si durante el desarrollo del proyecto no hubiera suficientes recursos para llevar a cabo los proyectos en curso para los clientes, se corre el riesgo de desviar el recurso humano de este proyecto a un cliente, o varios, dejando así en *stand by* el proyecto en cuestión, y en el peor de los casos llegando a descartar dicho proyecto. Y aunque no se descarte el

proyecto, los parones ocasionados por trabajos paralelos a este proyecto siempre hacen aumentar las jornadas dedicadas al mismo debido al coste de retomar el proyecto.

**Incompatibilidad entre tecnologías:** Como dentro de este proyecto se están utilizando una variedad importante de tecnologías cabe la posibilidad que durante el desarrollo del mismo nos encontremos con alguna incompatibilidad que haga peligrar los *timings* del proyecto.

**Desconocimiento del lenguaje:** Aunque la mayor parte de los lenguajes utilizados para desarrollar este proyecto son ampliamente conocidos por el programador y los desconocidos son aparentemente fáciles de aprender en el momento del desarrollo, nos podemos encontrar con algún cuello de botella inesperado que nos retrase los tiempos de desarrollo.

**Legales:** Un riesgo claro en la implantación al cliente es el tema de la seguridad de datos ya que en nuestra base de datos y en los informes mostrados habrá información confidencial del cliente en sí así como de los propios clientes del cliente. En este caso el riesgo es elevado ya que los datos de los clientes del cliente están fuertemente protegidos por la Ley Integral de Protección de Datos (LOPD) pudiendo causar por una fuga o mal uso de estos datos una cuantiosa multa que pondría en peligro la viabilidad del proyecto. Este riesgo queda subsanado mediante la seguridad implementada tanto por la aplicación como por la base de datos, así como por los contratos firmados entre nuestra empresa y el cliente y entre el cliente y sus clientes.

## 2.10 Conclusiones

Después de analizar con detenimiento los análisis de costes, beneficios y riesgos que supone el desarrollo de esta aplicación llegamos a la conclusión de que los beneficios obtenidos del desarrollo superan la dupla riesgo-coste por lo que se decide seguir adelante con el desarrollo del proyecto.

A nivel docente, al tener el proyecto una duración aproximada de 250 horas más unas 40 o 50 horas para la escritura de la memoria considero que es un proyecto apto para presentar como proyecto de final de carrera.

## 3 Análisis

---

### 3.1 Requerimientos

En este capítulo se van a exponer las especificaciones necesarias para que la aplicación cumpla con su deber, así como los requisitos necesarios para el correcto funcionamiento de la aplicación. Este análisis de requerimientos viene dado por la antecesora de la aplicación ya que al ser esta una actualización de una aplicación ya existente gran parte de los requisitos y son heredados de su antecesora.

### 3.2 Requerimientos funcionales

La aplicación tiene de ser capaz de generar un cuadro de mando personalizado para cada cliente con la única ayuda del reportServer y la información aportada por el usuario autentificado.

#### 3.2.1 Usuarios y seguridad

La aplicación no debe permitir acceder a ningún usuario sin autenticar dentro de ningún esquema de los clientes que contiene.

La aplicación debe permitir únicamente el acceso al esquema al que pertenece el usuario autentificado. Dicho usuario no puede ver ni tener acceso a los informes de los otros clientes.

Se tiene de poder identificar el rol o roles del usuario de manera que sólo pueda hacer o ver lo que su rol o roles le permiten:

- **Administrador:** Los usuarios con este rol tendrán derechos para crear nuevos usuarios.
- **Central:** Este rol permite a los usuarios acceder a todos los informes del cliente.
- **Builder:** Los usuarios de este grupo podrán ver y realizar informes dinámicos.
- **Roles de secciones:** Un usuario que no pertenezca al rol central sólo podrá ver las secciones de informes cuyo rol tengan.

#### 3.2.2 Menú

Una vez el usuario se haya autenticado la aplicación debe generar el menú de navegación acorde con el esquema del cliente al que pertenece y a los roles a los que pertenece.

#### 3.2.3 Filtros

Una vez seleccionado el informe por el usuario, la aplicación debe abrir el panel lateral de selección de filtros, generando para cada filtro el control de usuario necesario para seleccionar la información pertinente en cada uno de ellos.

### 3.3 Abrir Informe

Cuando el usuario decida abrir un informe la aplicación se debe encargar de mandar los filtros al control de ReportViewer que se encuentra dentro de un iFrame que también tiene que generar la aplicación. Una vez el informe se haya renderizado la aplicación abrirá una ventana en el área de trabajo con el tamaño del informe, en el caso de que el tamaño del informe sea superior en altura o en anchura al tamaño del área de trabajo la aplicación se encargará de redimensionar la ventana de informe para que quepa dentro de ésta.

#### 3.3.1 Abrir mapa

Si un usuario abre un informe de carácter geográfico la aplicación se encargará de leer la consulta que contiene con tal de generar los polígonos y los *clusters* de puntos que mostrará mediante una interfaz de googleMaps.

#### 3.3.2 Repositorio

Cuando un usuario cierra una ventana de informe la aplicación debe ocultar ésta y generar un nuevo elemento en la sección de repositorio del menú. A su vez cuando el usuario seleccione un elemento correspondiente a un informe “eliminado” en la sección repositorio la aplicación eliminará este elemento y volverá a mostrar en su última ubicación la ventana del informe seleccionado.

#### 3.3.3 Modulo administración

Cuando el usuario haga *click* sobre el botón de cuenta de usuario la aplicación levantará una ventana modal donde mostrará las opciones de administración de la cuenta que tiene dicho usuario según los roles a los que pertenece. Los apartados que se pueden encontrar en esta ventana son:

- **Alta de usuario:** Permite al administrador crear un nuevo usuario
- **Cambiar contraseña:** Permite a cualquier usuario cambiar su contraseña
- **Informes dinámicos:** Permite a los usuarios con permisos de Builder acceder al ReportBuilder
- **Salir:** Permite a todos los usuarios cerrar su sesión.
- **Acerca de:** Proporciona a todos los usuarios información sobre la aplicación.

## 3.4 Requerimientos no funcionales

En este apartado se van a describir los requerimientos que debe cumplir la aplicación los cuales no afectan a sus funcionalidades.

### 3.4.1 Requerimientos de diseño

La aplicación deberá seguir un diseño acorde con el del resto de aplicaciones web de la empresa, dicho diseño consta de un menú lateral formado por baldosas el cual se va desplegando según profundizas en sus niveles, una área de trabajo central donde se van a ver todos los informes, y un marco de aplicación en un degradado de azules con los pantone de la empresa. En el marco inferior tenemos el logo de la empresa y el botón de la cuenta de usuario.



### 3.4.2 Requerimientos de accesibilidad

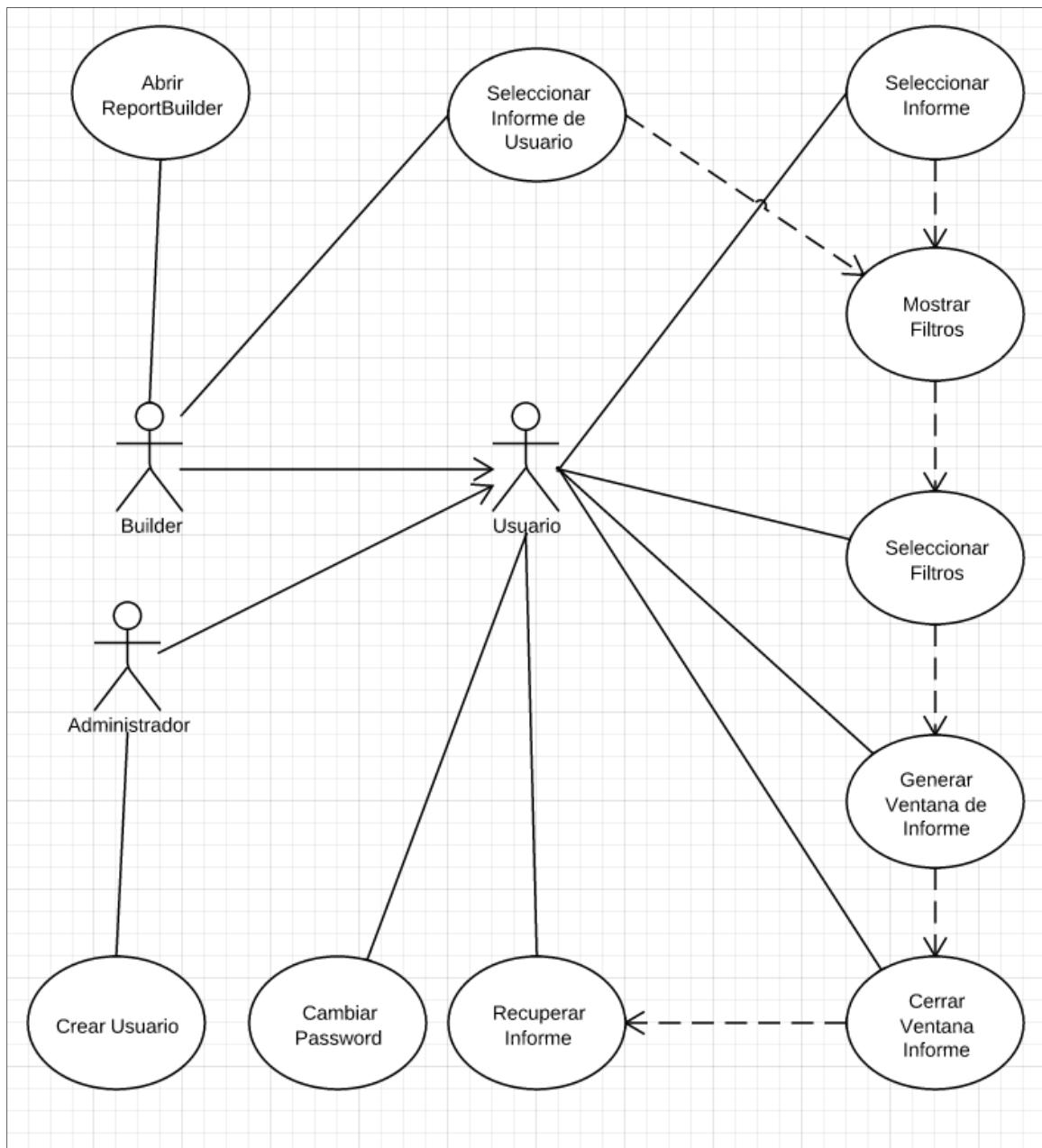
El cuadro de mando debe ser accesible desde cualquier punto y desde cualquier dispositivo. Por eso se va a desarrollar como una aplicación web, habrá que intentar hacerlo compatible con los navegadores más extendidos en el mercado.

## 3.5 Casos de uso

En este apartado se van a describir los casos de uso del proyecto.

### 3.5.1 Caso de uso principal

En esta sección podemos ver el caso de uso que muestra a grandes rasgos todo lo que pueden hacer los usuarios dentro de la aplicación



### 3.5.2 Especificación de los casos de uso

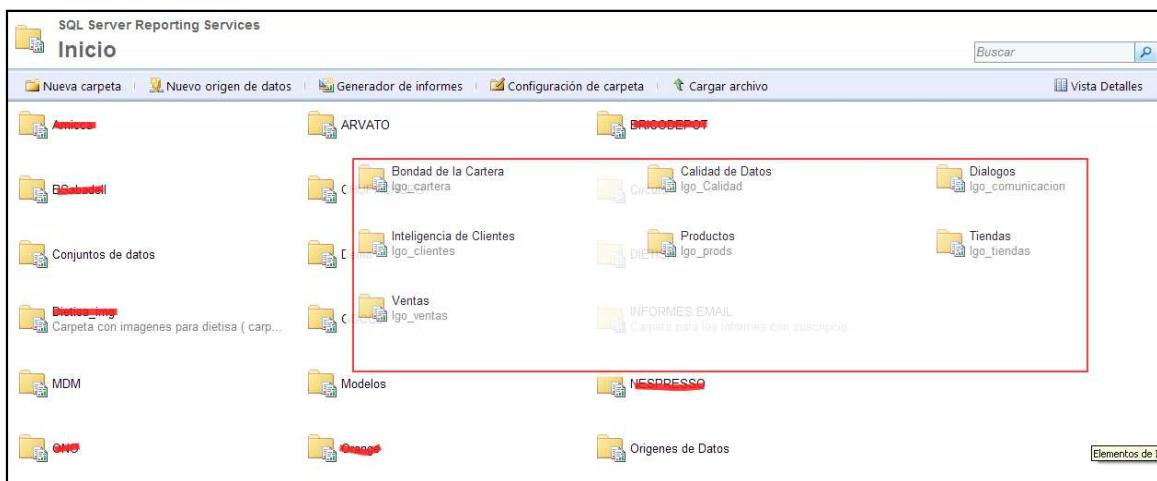
#### Generación Menú

##### Descripción

Cuando el usuario accede a la aplicación ésta se encarga en primera instancia de generar su menú de navegación en función del cliente al que pertenece y sus permisos.

#### Flujo de eventos

- La aplicación identifica el cliente al que pertenece al usuario
- Mediante una llamada al webService de reportingServices extrae el árbol de elementos del cliente.
- Se lee el primer nivel de carpetas del cliente para generar el menú principal
- Se descartan los informes de la raíz
- Se siguen leyendo los siguientes niveles para terminar de montar el árbol
- En la descripción de cada elemento encontramos el nombre del logo a utilizar por el menú
- Si un informe consta como oculto no se va a mostrar en el menú

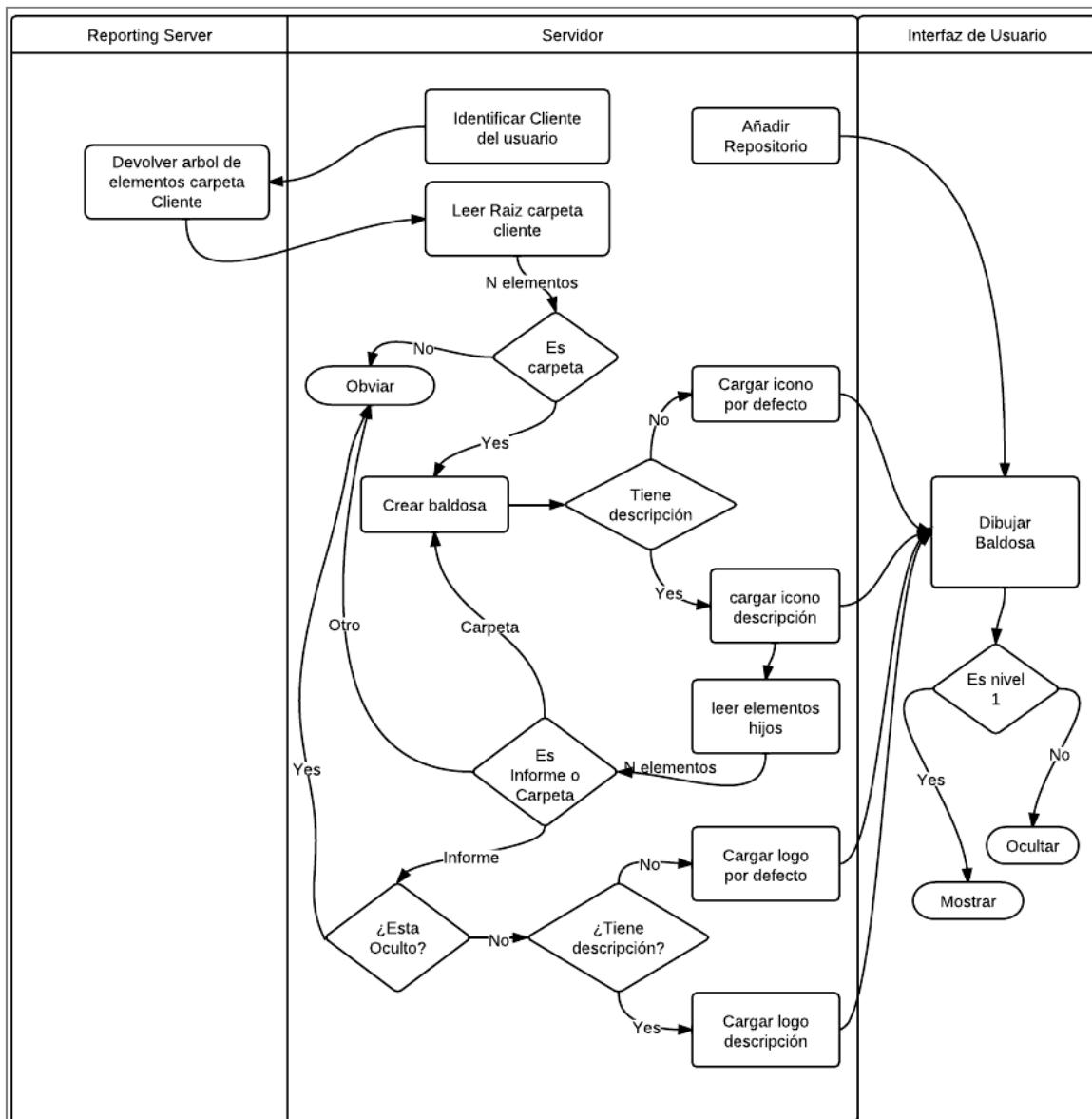


## Precondiciones

- El usuario se ha autenticado correctamente y entra a la pantalla principal de la aplicación.
- El servidor de Reporting debe estar operativo para proporcionar la información al sistema.
- Si el usuario forma parte del Grupo Builder se repetirá el mismo proceso para el servidor de Informes de usuario añadiendo una baldosa de mis informes al Menú Principal.

## Pos condiciones

- Queda creado el menú principal de la aplicación que no se volverá a crear si no se refresca la página



## *Seleccionar Informe*

### **Descripción**

El usuario se sirve de la navegación por el menú lateral de la aplicación para seleccionar el informe que desea ejecutar.

### **Flujo de eventos**

#### *Flujo Básico*

- El usuario hace *click* en un elemento del menú que contiene un subnivel.
- La aplicación abre el subnivel seleccionado.
- El usuario selecciona un elemento de informe del subnivel.
- La aplicación lanza el flujo de mostrar los filtros.

#### *Flujo alternativo 1*

- El usuario hace *click* en un elemento del menú que no forma parte del último nivel abierto.
- La aplicación cierra todos los niveles superiores y abre el subnivel seleccionado.

#### *Flujo alternativo 2*

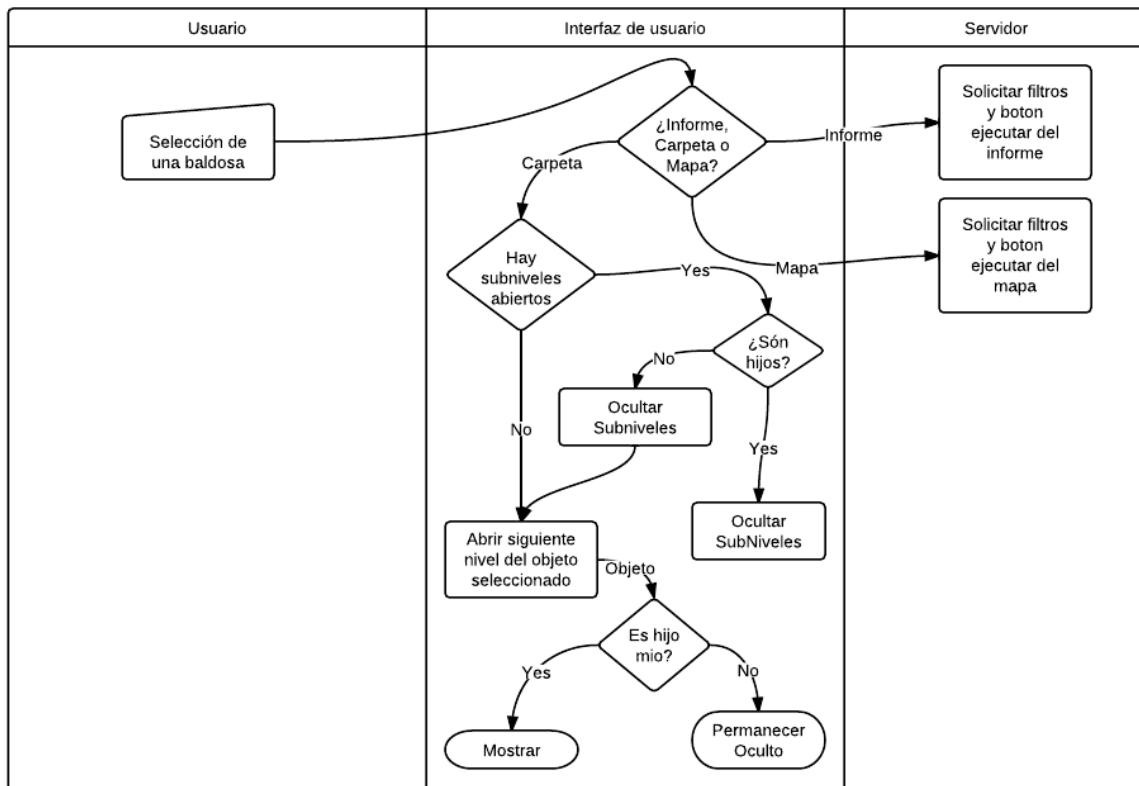
- El usuario hace *click* sobre un elemento con subniveles abiertos.
- La aplicación cierra los subniveles abiertos.

### Precondiciones

- El sistema debe haber cargado correctamente el menú.
- El usuario se sitúa sobre el área del menú y hace *click* en una baldosa

### Pos condiciones

- La interfaz de usuario envía al servidor la petición de los filtros del informe seleccionado



### *Mostrar filtros*

#### **Descripción**

Una vez seleccionado un informe por parte del usuario la aplicación se encarga de analizarlo, para extraer su colección de filtros mediante los webservices de ReportingServices y poder así mostrarlos al usuario.

#### **Flujo de eventos**

##### *Flujo Básico.*

- La interfaz de usuario envía al servidor el path del informe seleccionado.
- El servidor hace una llamada a los webServices de ReportingServices para que estos le devuelvan la colección de filtros del informe.
- El servidor comprueba si hay valores predeterminados en cuyo caso los rellena para que sean seleccionados por defecto al mostrar los filtros.
- El servidor prepara los filtros para enviarlos en formato JSON a la interfaz de usuario.
- La interfaz de usuario recibe una cadena con la colección de filtros que debe mostrar al usuario.

##### *Flujo Alternativo.*

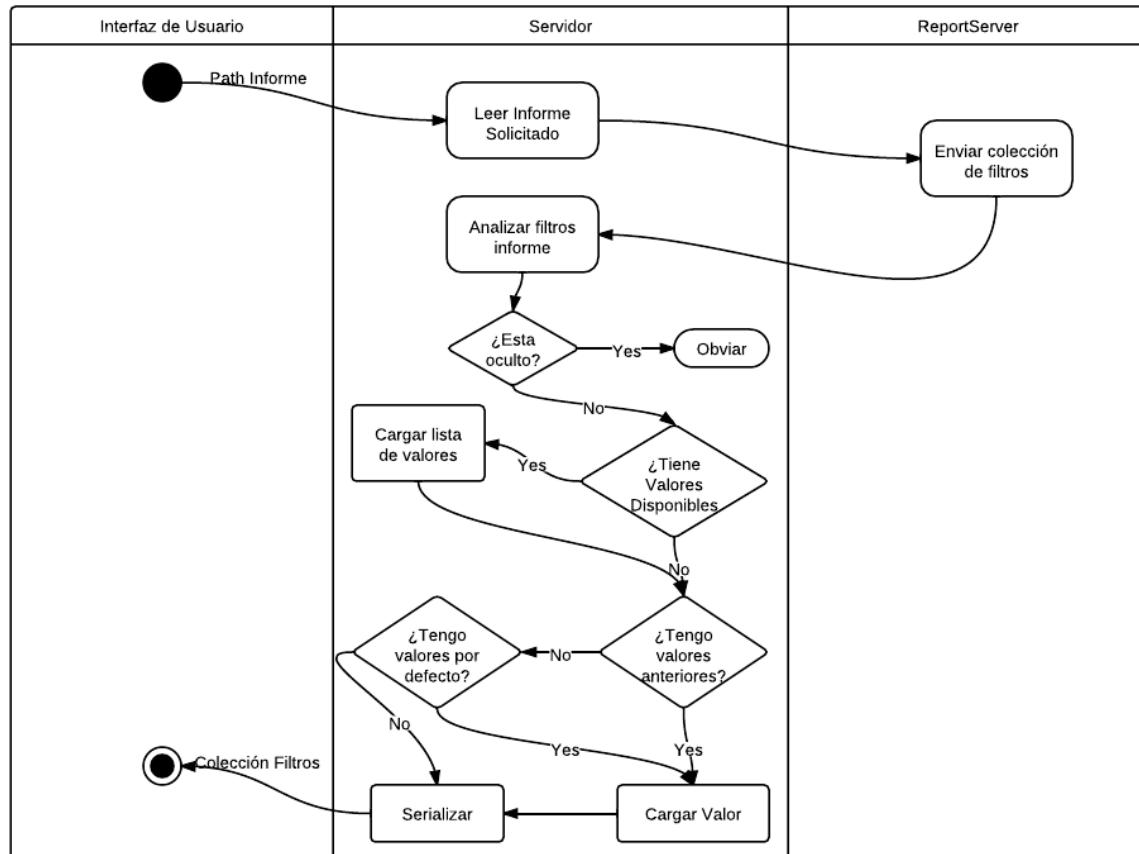
- Cuando el servidor analiza los filtros se percata de que tiene selecciones guardadas de otras ejecuciones.
- El servidor rellena como valores por defecto los valores seleccionados con anterioridad.

### Precondiciones

- La interfaz de usuario envía al servidor el path de un informe solicitando los filtros de este

### Postcondiciones

- El servidor devuelve a la interfaz de usuario la colección de filtros del informe solicitado.



### *Dibujar y Seleccionar Filtros*

#### **Descripción**

Una vez tiene los filtros en pantalla el usuario se encarga de seleccionar las opciones que crea necesarias para mostrar la información que desea ver.

## **Flujo de eventos**

### Flujo Básico

- La interfaz de usuario analiza uno a uno los filtros enviados por el servidor y genera el control de usuario necesario para cada uno de ellos.
- La interfaz de usuario rellena los filtros con valores enviados desde el servidor.
- El usuario hace los cambios necesarios en cada filtro para acotar la información que desea ver en el informe
- El usuario hace *click* en el botón de ejecutar informe.
- La interfaz de usuario carga los valores de los parámetros en la colección de filtros y los envía al servidor

### Flujo Alternativo

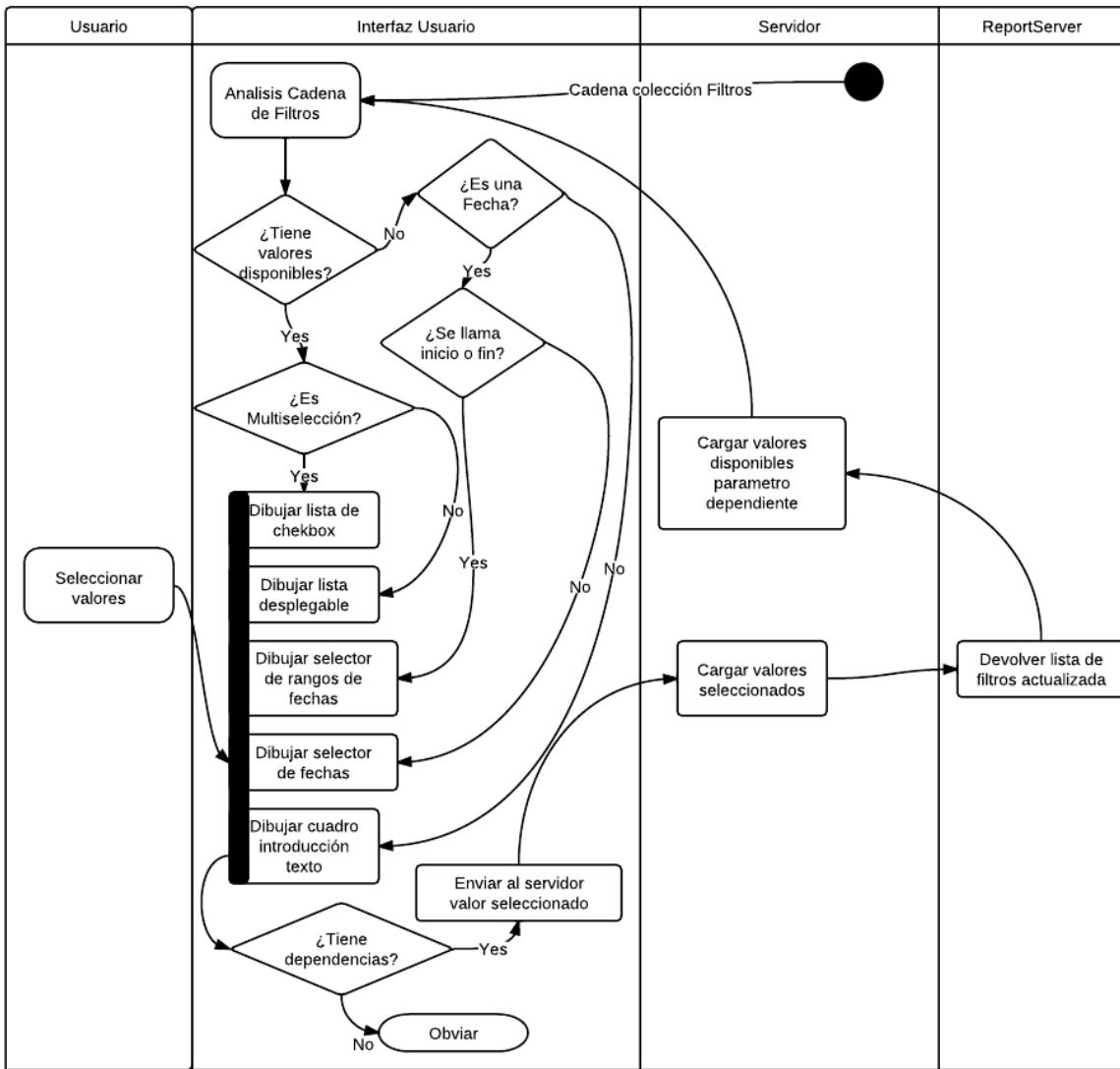
- El usuario selecciona un filtro con una dependencia
- La interfaz de usuario recoge el valor de ese filtro y lo envía al servidor
- El servidor envía al ReportServer la petición de valores pasando el filtro recibido como parámetro.
- El reportserver devuelve la lista de parámetros según los datos recibidos del parámetro enviado
- El servidor recoge los valores por defecto del parámetro dependiente y los envía a la interfaz de usuario.
- La interfaz de usuario borra las opciones del parámetro dependiente y carga en su lugar las nuevas opciones.

### Precondiciones

- La interfaz de usuario recibe la cadena con la colección de filtros que tiene que cargar

### Postcondiciones

- El usuario selecciona los filtros y hace *click* en el botón ejecutar.



## Generar Ventana de Informe

### Descripción

Cuando el usuario finalmente aprieta el botón ejecutar la aplicación se encarga de generar la ventana de informe para que este pueda visualizarlo.

### Flujo de eventos

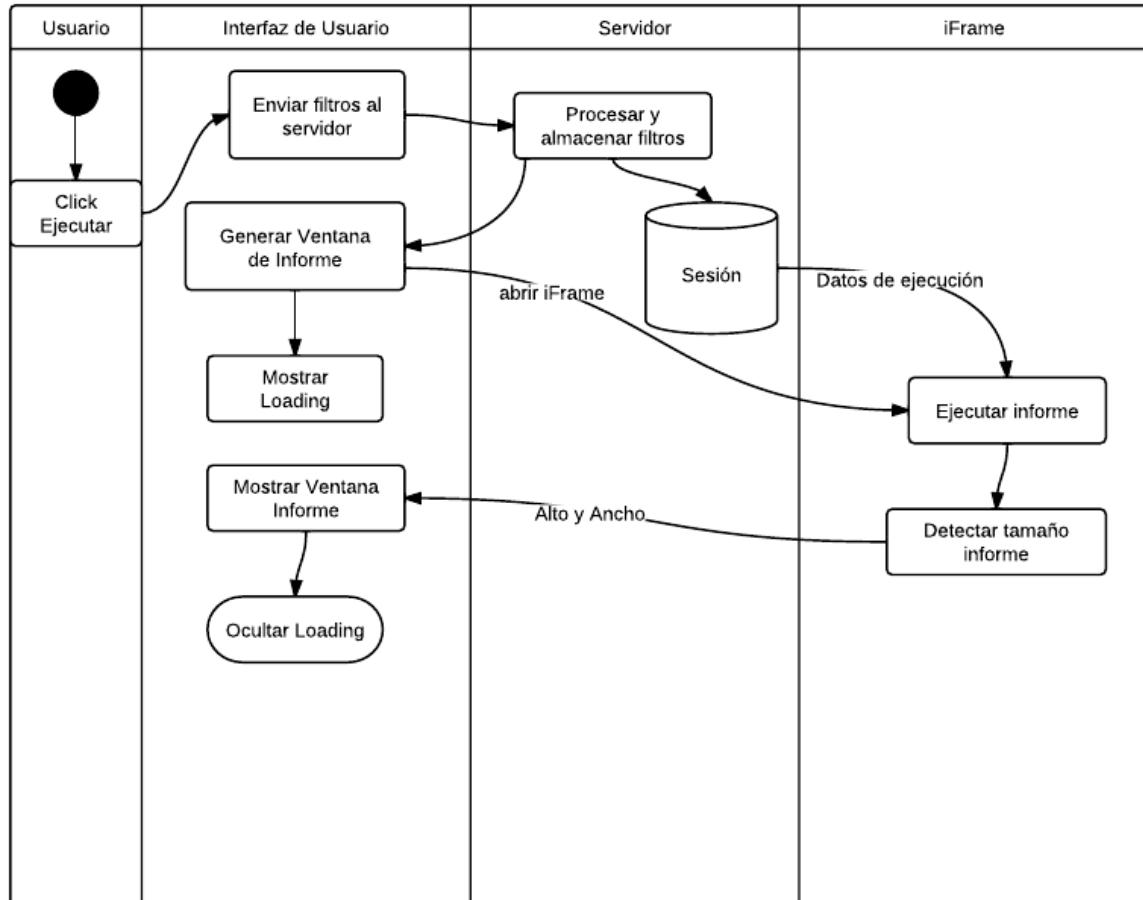
- El usuario hace *click* al botón generar Informe.
- La interfaz de usuario envía la colección de filtros con los parámetros introducidos al servidor.
- El servidor almacena los la colección de filtros con sus valores en la Sesión de trabajo.
- La interfaz web genera la ventana con un iframe en su interior que llama a la página que contiene el ReportViewer
- El ReportViewer se ejecuta usando los valores necesarios almacenados en la sesión.
- Una vez abierto el informe el iframe envía el tamaño de este a la página principal
- La página principal muestra al usuario la ventana de informe con el tamaño de este.

### Precondiciones

- Todos los filtros tienen que tener un valor asignado.

### Postcondiciones

- La interfaz de usuario mostrará por pantalla la ventana con el informe ejecutado.



### *Cerrar Informe*

#### **Descripción**

Si el usuario decide dejar de utilizar el área de trabajo para mostrar el informe este aprieta el botón de cerrar informe y la aplicación se encarga de enviarlo al repositorio.

#### **Flujo de eventos**

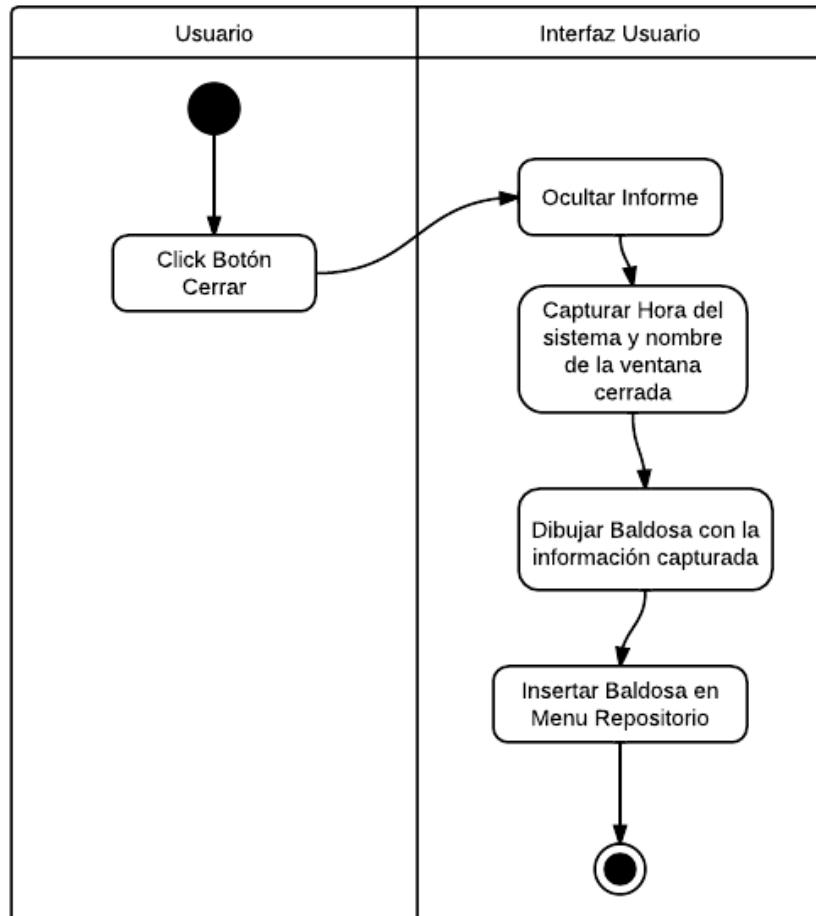
- El usuario hace *click* a la aspa de cerrar la ventana de informe.
- La interfaz web oculta la ventana de informe
- La interfaz web genera la baldosa del informe en la sección del repositorio

### Precondiciones

- La aplicación ha generado previamente la ventana de informe
- El usuario decide cerrar el informe

### Postcondiciones

- El informe permanece oculto
- Queda creada una baldosa en el repositorio para recuperar en cualquier momento de la sesión el informe cerrado.



## *Recuperar informe*

### **Descripción**

Si el usuario desea recuperar un informe previamente cerrado se dirige al repositorio y haciendo *click* sobre su baldosa la aplicación se encargará de devolver el informe al área de trabajo.

### **Flujo de eventos**

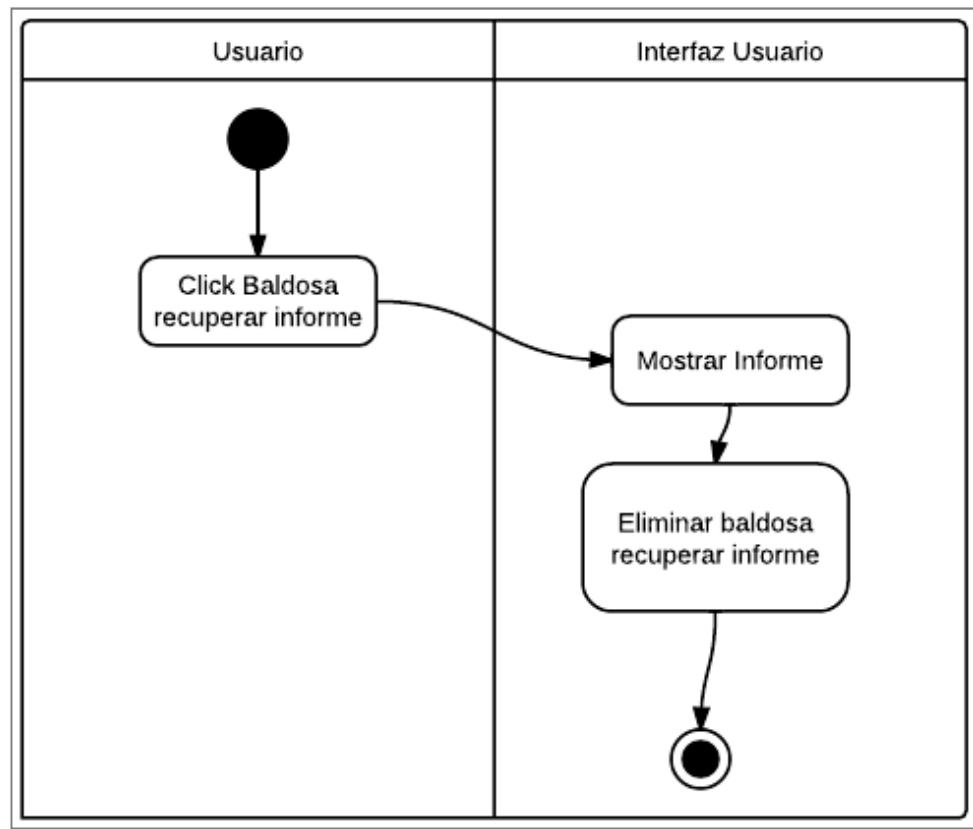
- El usuario hace *click* a la baldosa de recuperar un informe dentro del repositorio
- La interfaz web vuelve a mostrar el informe en su último emplazamiento
- La interfaz web elimina la baldosa de recuperar el informe.

### Precondiciones

- El informe ha sido previamente cerrado.

### Postcondiciones

- La baldosa de recuperación del informe queda eliminada
- El informe vuelve a ser visible.



## *Crear Nuevo Usuario*

### **Descripción**

Si el usuario tiene permisos de administrador podrá acceder al área de crear usuario para dar de alta un nuevo usuario para la aplicación.

### **Flujo de eventos**

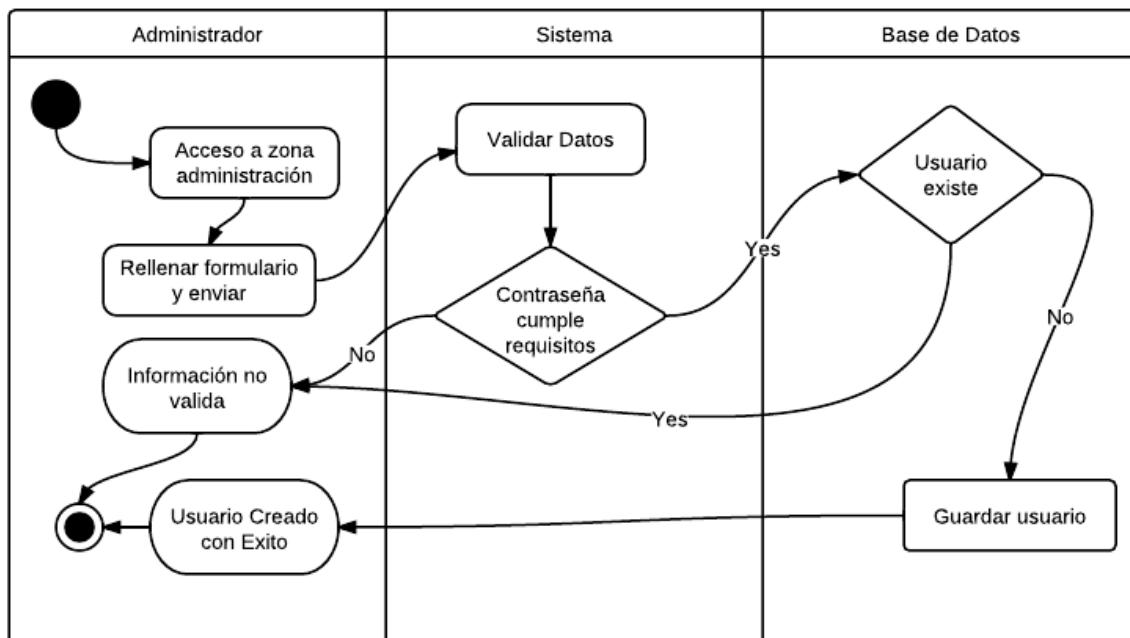
- Un administrador accede a la pestaña de alta de usuario
- Rellena el formulario de alta y lo envía
- El sistema recibe los datos del formulario valida que sean validos y los introduce en base de datos

### Precondiciones

- La sesión de usuario activa debe tener permisos de administrador

### Postcondiciones

- Quedará el usuario creado para el cliente al que pertenece el administrador
- El administrador debe ponerse en contacto con el proveedor (Nosotros) para que asignemos los permisos necesarios al usuario, de esta manera se controlan las licencias de uso.



## Cambiar contraseña

### Descripción

Si el usuario desea cambiar su contraseña podrá hacerlo en el área de cambio de contraseña.

### Flujo de eventos

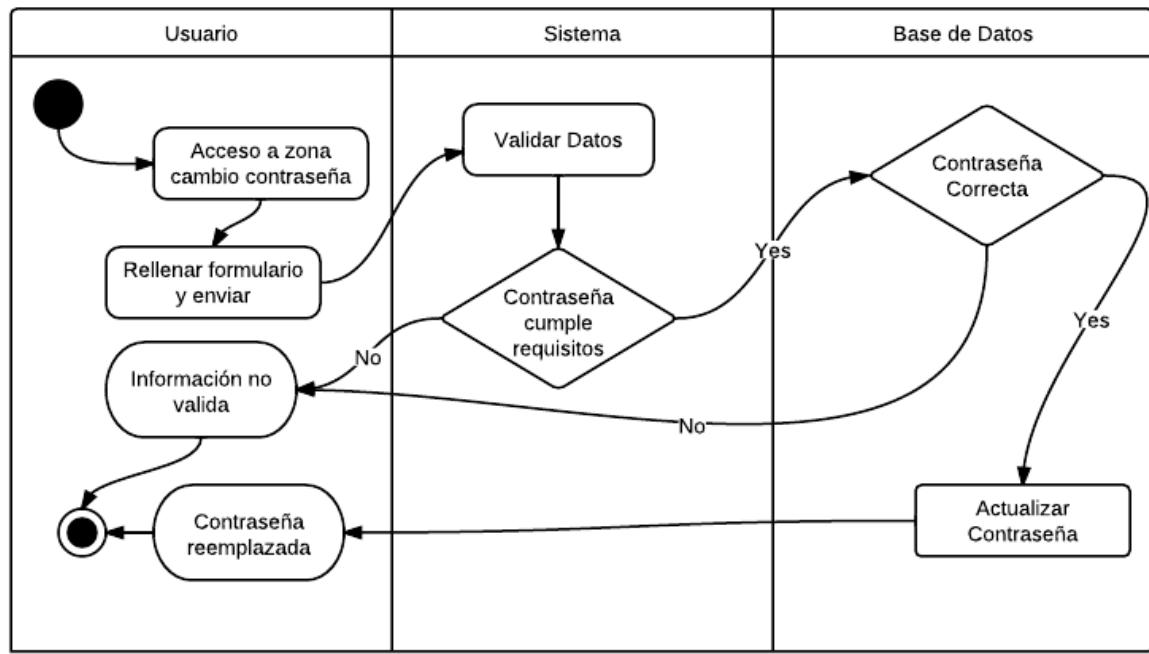
- Un usuario accede al área de cambio de contraseña.
- Rellena el formulario y lo valida.
- El sistema valida los datos y actualiza la información en base de datos.

### Precondiciones

- El usuario está autenticado en la aplicación

### Postcondiciones

- La contraseña de acceso a la aplicación se reemplaza por la introducida por el usuario.



## 4 DISEÑO

---

En este capítulo el lector podrá ver como lucirá la interfaz de usuario una vez finalizada la aplicación, se explicará cómo se relaciona ésta con el sistema para poder extraer la información de las fuentes de datos.

### 4.1 Acceso a la aplicación

El usuario accederá a la aplicación introduciendo en su navegador la dirección url de la misma. Al abrirla se encontrará con la pantalla de *login* que le obligará a autentificarse para poder seguir adelante.



En esta pantalla también encontraremos un botón que abrirá un formulario para que en caso de olvidar la contraseña la aplicación nos la mande al correo previa respuesta de una pregunta de seguridad.

### 4.2 Pantalla principal

Una vez autenticado el usuario, la aplicación abre la pantalla principal en forma de *popup* que ocupa toda la superficie de la pantalla del usuario, en esta pantalla se van a realizar todas las tareas de la aplicación. Esta pantalla como ya se ha explicado está formada por un Marco que contiene un botón de usuario en la esquina inferior derecha, el logo de la empresa en la esquina inferior derecha, el menú de navegación al lado izquierdo y los filtros y el botón de ejecución ocultos en el lado derecho.



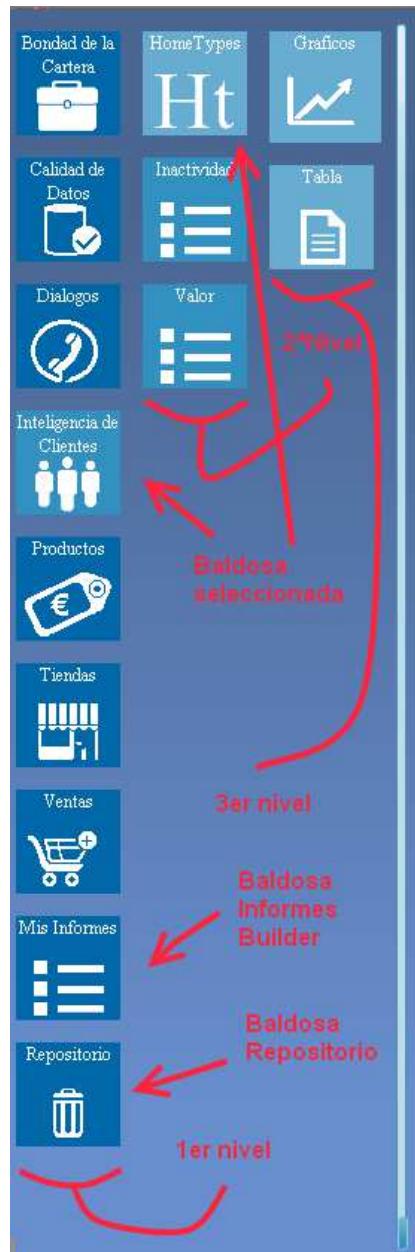
En la imagen adjunta se puede ver cómo está distribuida la pantalla principal de la aplicación tal y como se ha descrito arriba. El motivo de este diseño obedece dos premisas básicas, primero otorgar el máximo espacio posible al área de trabajo ya que es el alma de la aplicación, por otro lado como ya se ha comentado este diseño también está adaptado al diseño de las otras aplicaciones de la empresa.

#### 4.3 Menú Principal.

Para movernos por todas las opciones que nos da la aplicación se implementa el menú de baldosas mediante el cual seremos capaces de navegar a todos los informes creados para el cliente, en el caso de tener permisos de *builder* también podrá acceder a los informes creados por el mismo. Para acabar dentro del menú principal también encontraremos la baldosa de repositorio que nos permitirá acceder a los informes cerrados durante la sesión de trabajo.

El menú principal puede llegar a contar hasta con 4 niveles de agrupación de baldosas. Cada nivel tiene un tono de azul y cuando se selecciona una baldosa que contiene un nivel inferior esta cambia su tono de azul al del nivel siguiente y muestra el nivel siguiente.

A continuación se adjunta una captura para ver el aspecto final del menú de Baldosas.

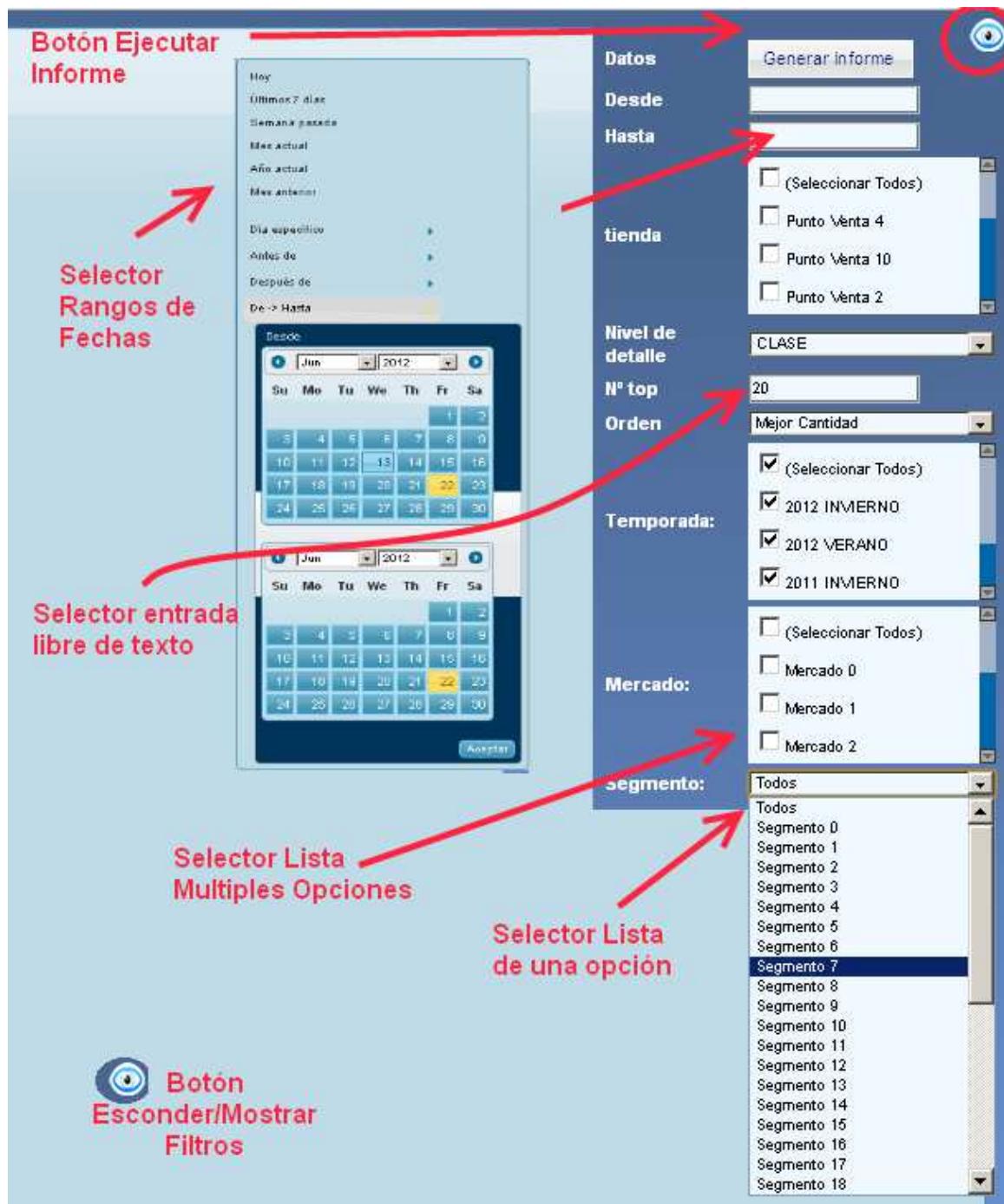


Como se puede observar en esta captura las baldosas están formadas por un título y por un ícono descriptivo. El caso del título se va a extraer del nombre de la carpeta o informe que representan. Por lo que hace al ícono pueden tener un ícono por defecto o uno definido en la descripción del informe o carpeta que representan. El menú también contará con una animación en cada baldosa para cuando el cursor pase con encima el usuario perciba interacción.

Por último cabe destacar que si el usuario no tiene el cursor dentro de la zona del menú este sólo mostrará el primer nivel, dejando así más espacio para el área de trabajo.

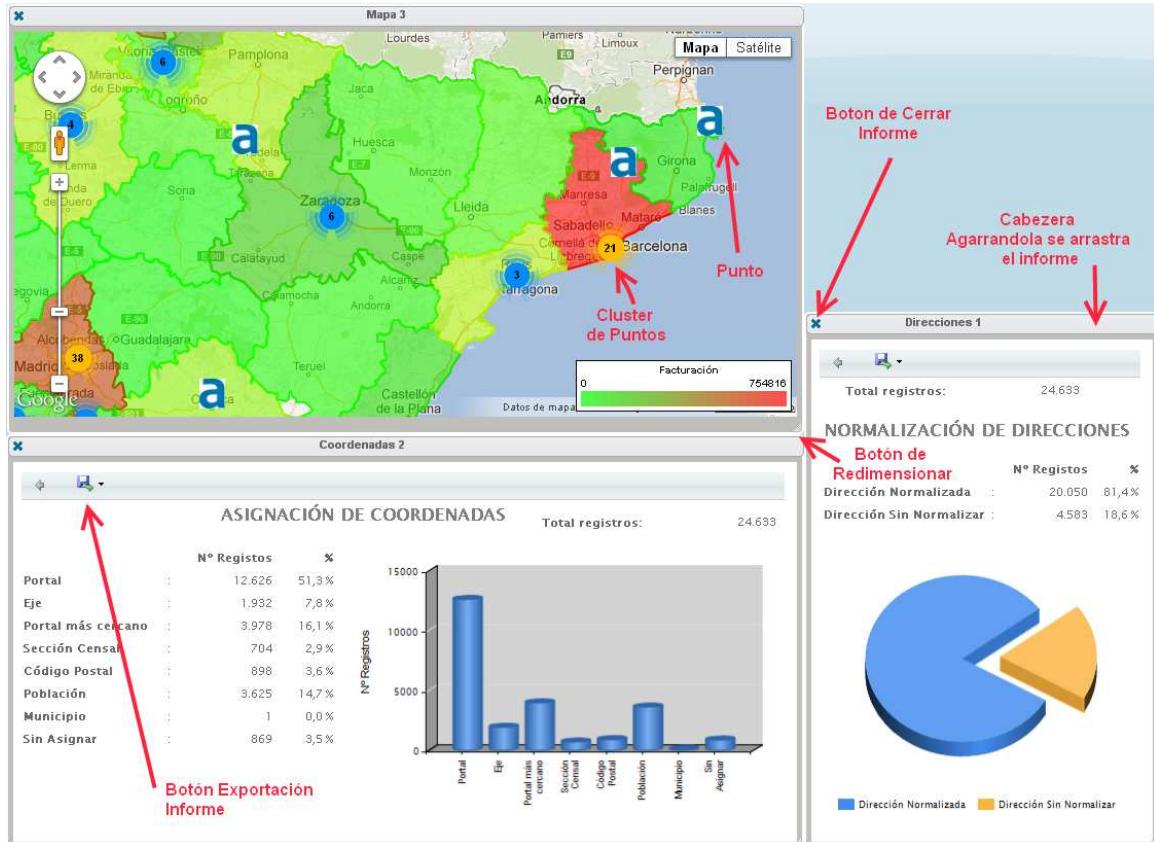
## 4.4 Panel de filtros

Cuando el usuario seleccione un informe mediante el menú de baldosas aparecerá en el lado derecho de la aplicación el panel de filtros. Desde allí podrá seleccionar los distintos parámetros de ejecución del informe y ejecutarlo.



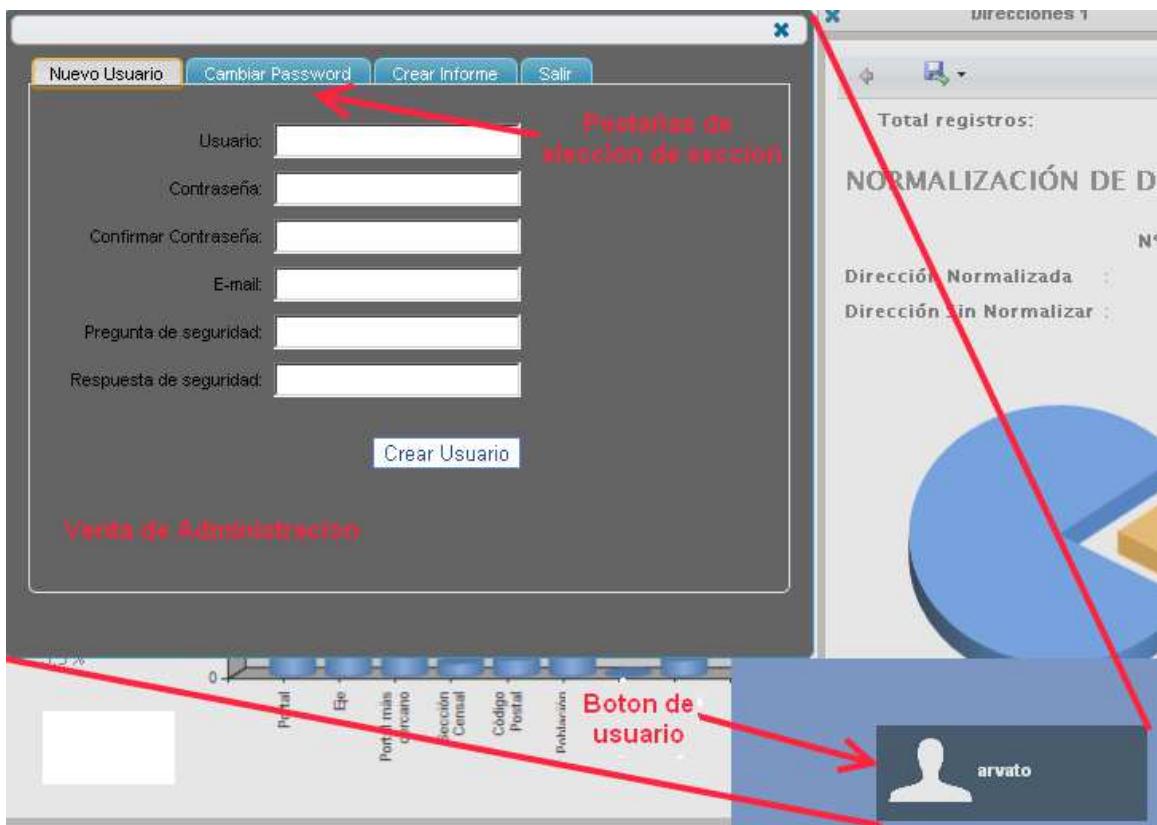
## 4.5 Área de trabajo, Informes y Mapas

En el área de trabajo es donde vamos a poder visualizar los informes y los mapas generados, estos se generarán en ventanas que se podrán cerrar, mover y redimensionar al gusto del usuario.



## 4.6 Área de administración

Cuando el usuario hace *click* en el botón de usuario se le abre una ventana modal con las distintas secciones de administración a las que tiene acceso, una vez allí podrá realizar acciones tales como: Crear usuarios, cambiar su contraseña, abrir Report Builder o abandonar la aplicación.



## 5 CODIFICACIÓN

---

### 5.1 Introducción

La aplicación está diseñada de manera que sea fácil de mantener y actualizar en el futuro, por este motivo se ha decidido separar tanto como sea posible los distintos módulos de la aplicación. Aunque el portal web donde se aloja la aplicación sólo consta de 6 páginas las funciones desarrolladas para el funcionamiento de estas están bien estructuradas y distribuidas en ficheros dependiendo de su uso de manera que permita la fácil implantación de dichas funciones en otras aplicaciones de la empresa.

A continuación en este capítulo pasaremos a describir los diferentes módulos en los que se ha separado la aplicación.

### 5.2 Tecnologías utilizadas.

En esta sección describiremos brevemente la tecnología utilizada para desarrollar este proyecto. De esta manera ubicaremos al lector en contexto en el caso de que desconozca parte de estas tecnologías.

#### 5.2.1 .NET (c#)

Es un entorno de trabajo de Microsoft que nos permite desarrollar aplicaciones de forma rápida para cualquier hardware. Dentro de este entorno de trabajo podemos escoger entre varios lenguajes de programación: C#, Visual Basic .NET, C++, F#, J# etc....

Para el desarrollo de este proyecto se ha usado C# que es un lenguaje orientado a objetos creado por Microsoft para la plataforma .NET i que deriva de C/C++ con una estructura muy similar a java.

.NET es la alternativa de Microsoft a otros entornos de programación como son PHP o Java. La decisión de usar .NET viene dada por la comodidad que siente el programador con este lenguaje y por las restricciones mencionadas en anteriores capítulos.

#### 5.2.2 Microsoft SQL Server 2008 R2

Es el sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Otras alternativas en el mercado serían Oracle, PostgreSQL o MySQL. Nos decidimos por esta solución porque ya la tenemos implantada en la empresa y porque dentro de la misma nos proporciona las bases de datos OLAP en caso de necesidad y Reporting Services para desarrollar los informes sin coste añadido.

#### *Reporting Services*

Es el modulo de gestión y creación de informes que nos proporciona Microsoft dentro de SQL Server. Los informes se definen desde una herramienta propia llamada ReportBuilder o en formato de proyectos de VisualStudio con el plugin de Business Intelligence Developer Studio que trae incorporado las herramientas de cliente de SQL Server.

### 5.2.3 Javascript

Es un lenguaje de programación interpretado utilizado para la programación en el lado cliente, dicho lenguaje es interpretable por todos los navegadores modernos por lo que es una herramienta fundamental para dotar a las páginas web de dinamismo y mejorar la interfaz de usuario de las mismas.

#### JQuery

Es un conjunto de librerías openSource de javascript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

#### API GoogleMaps

Es un conjunto de librerías javascript que nos proporciona Google para poder manejar mapas de googleMaps en nuestra página.

## 5.3 Descripción de los módulos.

### 5.3.1 Pantalla principal

Este modulo se encuentra en la página informes, dicha página hace las funciones de masterPage del proyecto y es donde se direcciona al usuario una vez autentificado. En este módulo se encuentran todas aquellas funciones utilizadas para el montaje de la página principal de forma dinámica así como todas aquellas que gestionan el uso de los otros módulos por parte del usuario. Las funciones de este módulo se encuentran todas ubicadas dentro de los mismos ficheros de código de la página, se ha tratado de separar en distintas funciones todas las acciones que realiza la página que en general suceden con la carga de la misma. La página se ha programado para que no genere refrescos por lo que una vez cargadas todas las funciones serán el resto de módulos los que se encargaran de realizar las acciones solicitadas por el usuario o por el servidor.

- **Lado Cliente:** De este lado encontramos todas las funciones destinadas al montaje de la página para adaptar está en la medida de lo posible a las distintas resoluciones y browsers que nos podemos encontrar.

#### Código de ejemplo:

```
var alto;
var ancho;

$(document).ready(function () {
    alto = screen.height - 150;
    if (alto < 570) {
        alto = 570;
    }

    $("#loadParams").hide();
    ancho = screen.width - 125;
    $("#contenedor").css("height", alto);
    $("#contenedor").css("width", ancho);
```

- **Lado Servidor:** Las funciones del lado servidor de este módulo se encargan de inicializar las conexiones tanto con las distintas bases de datos utilizadas (Cliente, seguridad, Reporting) como con el servidor de reporting. También se encargan de cargar una serie de variables en sesión que serán necesarias para el correcto funcionamiento de otros módulos de la aplicación.

**Código de ejemplo:**

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {

        SqlConnection conexion = new SqlConnection(@"Data Source=
        SqlCommand query = new SqlCommand("select logo,url,usr,pwd,domain,server,
        conexion.Open();
        SqlDataReader dr = query.ExecuteReader();

        while (dr.Read())
        {
            if (dr["rolename"].ToString() != "BUILDER")
            {

```

### 5.3.2 Menú Baldosas

En este módulo han quedado introducidas todas las funciones necesarias para el montaje del menú principal. La colección de funciones para desarrollar dicho menú se pueden diferenciar en dos partes claramente separadas.

- **Lado Cliente:** Son un conjunto de funciones desarrolladas en jQuery y almacenadas todas ellas en el fichero menú Baldosas.js que se encargan de dar al menú su aspecto y funcionalidad a partir del árbol creado por el servidor. Este conjunto de funciones son de fácil adaptación a otra aplicación ya que se encargan de transformar un árbol de <div> en el menú de baldosas que contiene la aplicación.

**Código de ejemplo:**

```
$(".baldosa").click(baldosa_Click);

$("#barraMenu").hover(
    function () {
        var anchura = open * 85;
        if ($("#barra1").slider("option", "min") < 0) {
            anchura += 20;
            $("#barra1").show();
        }
        $(this).css("width", anchura);
        for (var x = 2; x <= open; x++) $("#menu" + x).show();

    },
    function () {
        $(this).css("width", 85);
        for (var x = 2; x <= open; x++) $("#menu" + x).hide();
        $("#barra1").hide();
    });
});
```

- **Lado servidor:** En este apartado nos encontramos con la colección de funciones que partiendo de la información proporcionada por el webService de *Reporting Services* genera el árbol de *<div>* que luego utilizaran las funciones del lado del cliente para transformar en el menú. Este conjunto de funciones están muy ligadas al *reporting* por lo que no serían utilizables en otro proyecto de otro ámbito, sin embargo el hecho de que estén bien diferenciadas facilitará en el futuro posibles ampliaciones de la aplicación.

**Código de ejemplo:**

```
server = Session["builderServer"];
catalogItems06 = ((ReportingService2005)server).ListChildren(path, false);
HtmlGenericControl top = new HtmlGenericControl("div");
top.ID = "baldosa1" + x.ToString();
top.Attributes.Add("class", "baldosa firstLevel");
top.InnerHtml += "<a id=\"MisInformes\" class=\"top_link\"><span class=" +
top.InnerHtml += "<img src='../App_Themes/lgo_Fld.png' class='lgo_baldosa'>" +
llenarMenuBuilder(catalogItems06, 2, x);
cont_Menu1.Controls.Add(top);
```

### 5.3.3 Filtros.

Aquí encontraremos todas las funciones relacionadas con el montaje del panel de filtros de informe, como en el caso anterior en este modulo también tenemos las funciones diferenciadas por el lado servidor y el lado de cliente aunque en este caso están mucho más ligadas entre sí ya que mantienen una constante comunicación asíncrona mediante ajax.

- **Lado Cliente:** Del lado cliente nos encontramos con un conjunto de funciones todas ellas ubicadas en el fichero *Filtros.js* que se encargan de dibujar los controles de usuario para seleccionar los distintos filtros así como de controlarlos. Estas funciones también se encargan de comunicarse asíncronamente con el servidor para llenar los filtros con la información recibida por el servidor y para enviarle a este los datos introducidos por el cliente.

**Código de ejemplo:**

```
$.ajax({
    type: "POST",
    url: "Informes.aspx/cargaParametros",
    data: "{reportPath: '" + path + "', recarga: 'NO', tipo:'" + tipo + "'}",
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    success: function (response) {
        $("#tablaPagina").children().remove();
        parametros = response.d;
        var cadenaHtml = "<tr><td><span class='paramLabel'>" + path.substring(path.lastIndexOf("/") + 1, path.l
        if (tipo == "Map") cadenaHtml = "<tr><td><span class='paramLabel'>" + path.substring(path.lastIndexOf("/")
        $("#tablaPagina").append(cadenaHtml);
        $.each(response.d, function (index, value) {
            cadenaHtml = "<tr><td style='width:35%;'><span id='lbl" + this.controlID + "' class='paramLabel'>" +
            if (this.controlType == "checkboxList") addCheckBox(this, index, cadenaHtml);
            if (this.controlType == "dropDownList") addDropDownList(this, index, cadenaHtml);
            if (this.controlType == "fecha") addFecha(this, index, cadenaHtml);
            if (this.controlType == "text") addText(this, index, cadenaHtml);
        });
    }
});
```

- **Lado Servidor:** Por parte del servidor desarrollamos un conjunto de funciones que se encargan de comunicarse con *reporting services* para obtener la información de los filtros, también hay funciones orientadas a guardar los filtros en la sesión de trabajo para su posible uso en más de un informe.

#### Código de ejemplo:

```
[WebMethod]
public static List<ParamControl> cargaParametros(string reportPath, string recarga, string tipo)
{
    HttpContext.Current.Session["tipo"] = tipo;
    List<ParamControl> controles = new List<ParamControl>();
    string historyID = null;
    if (recarga == "NO") HttpContext.Current.Session["reportPath"] = reportPath;
    else reportPath = (string)HttpContext.Current.Session["reportPath"];
    RS2010.ParameterValue[] parametrosValues = null; // new RS2010ParameterValue[1];
    List<ParamControl> serverParams= (List<ParamControl>)(HttpContext.Current.Session["Parametros"]);
    if (recarga == "SI")
    {
```

#### 5.3.4 Generar Mapa

El módulo de generación de mapas lo encontramos en una página apartada de la página principal, esto es así porque de esta manera podemos crear varias ventanas de mapas en la pantalla principal utilizando iframe que llaman a esta página. Todas las funciones destinadas a la creación de los mapas las encontramos en los ficheros de código de esta página. En este sentido todas las funciones necesarias para crear un mapa son obligatoriamente generadas con javaScript ya que es el lenguaje sobre el que corre la api de googleMaps. De todas formas para generar el mapa también se usan funciones de la parte servidor que se encargan de crear scripts de javascript de forma dinámica.

- **Lado Cliente:** En este lado de la aplicación nos encontramos con la función que inicializa el mapa, también encontramos funciones que se generan dinámicamente en el servidor que se encargan de dibujar los polígonos y los puntos del mapa.

#### Código de ejemplo:

```
function addPolygon(Coords, color) {
    var polygon = new google.maps.Polygon({
        paths: Coords,
        strokeColor: color,
        strokeOpacity: 0.8,
        strokeWeight: 2,
        fillColor: color,
        fillOpacity: 0.35
    });
    polyArray.push(polygon);
}
```

- **Lado Servidor:** En el servidor tenemos un conjunto de funciones que se encargan de extraer la información de dentro el informe para luego poder hacer una consulta en base de datos que nos devuelva la información necesaria para dibujar los polígonos y los puntos sobre el mapa de googleMaps, una vez tenemos esta información desarrollamos una función que se encarga de escribir

de forma dinámica una función javascript con toda la información extraída de la base de datos que se ejecutará en el lado Cliente.

### Código de ejemplo:

```
StringBuilder sb = new StringBuilder();
sb.Append(@"<script type='text/javascript'>");
sb.AppendLine(@"    function initialize() {");
sb.AppendLine(@"        var myOptions = {");
sb.AppendLine(@"            zoom: 5,");
sb.AppendLine(@"            center: new google.maps.LatLng(40.40,-3.7),");
sb.AppendLine(@"            mapTypeId: google.maps.MapTypeId.ROADMAP}");
sb.AppendLine(@"        map = new google.maps.Map(document.getElementById('map_canvas'),myOptions);");
if (mapQueryPoly != "") {
    SqlCommand capas = new SqlCommand(mapQueryPoly.Replace("&gt;","<").Replace("&lt;","");
    SqlDataReader drc = capas.ExecuteReader();
    string scolor = "";
}
```

### 5.3.5 Generar Informe

El modulo que se encarga de renderizar los informes también está en una página separada de la principal. Este módulo de la aplicación está totalmente desarrollado en .Net dejando sólo una pincelada de javascript para enviar a la página principal el tamaño del informe. Dicho módulo usa el user Control ReportViewer proporcionado por .NET y que es el encargado de llamar a los informes de reporting services. Esta llamada también se podría realizar bajo el protocolo http utilizando una URL pero se hace bajo ReportViewer por motivos de seguridad ya que de esta manera podemos mantener el servidor de Reporting aislado de internet, dejando la aplicación como única ventana abierta a la red para ver los informes almacenados, e imposibilitando así el acceso desde internet a los orígenes de datos almacenados en el servidor de reporting evitando un posible hacking a nuestras bases de datos.

- **Lado Cliente:** En este lado sólo contaremos con una pequeña función que se ocupará de comunicar a la página principal el tamaño final del informe una vez este se haya renderizado por completo.

### Código de ejemplo:

```
<rsweb:ReportViewer ID="ReportViewer1" runat="server"
    DocumentMapCollapsed="True" Font-Names="Verdana"
    Font-Size="8pt" Height="100%"
    InteractiveDeviceInfos="(Collection)" ProcessingMode="Remote"
    WaitMessageFont-Names="Verdana" WaitMessageFont-Size="14pt"
    Width="100%">
    <ServerReport
        Timeout="-1" />
</rsweb:ReportViewer>
```

- **Lado Servidor:** Por este lado tendremos la función PageLoad () que se encargará de recoger los datos almacenados en sesión por la pagina principal y pasarlo al control reportViewer como parámetros de ejecución al iniciar la página.

#### Código de ejemplo:

```
Collection<Microsoft.Reporting.WebForms.ReportParameter> paramReport = new Collection<Microsoft.Reporting.WebForms.ReportParameter>();
List<ParamControl> parametros = (List<ParamControl>)Session["parametros"];
for (int i = 0; i < parametros.Count; i++)
{
    Microsoft.Reporting.WebForms.ReportParameter paramAct = new Microsoft.Reporting.WebForms.ReportParameter();
    paramAct.Name = parametros[i].controlID;
    foreach(string valor in parametros[i].selectedValues)
    {
        paramAct.Values.Add(valor);
    }
}
```

#### 5.3.6 Nuevo Usuario

El módulo de nuevo usuario también lo tenemos separado en una página aparte. Se utiliza el userControl del Framework de .NET para crear un nuevo usuario en la base de datos de seguridad de la aplicación. Únicamente se amplía la funcionalidad del userControl añadiendo al usuario creado el rol del CLIENTE al cual pertenece el administrador que lo ha dado de alta. Este módulo de la aplicación no tiene funciones por el lado cliente.

#### 5.3.7 Cambio Contraseña

Este módulo al igual que el anterior también esta implementado en una página aparte, también como el anterior usa el userControl proporcionado por .NET para cambiar la contraseña del usuario que lo solicita, en este caso no se realiza ninguna acción aislada del propio funcionamiento del control de usuario.

## 6 Pruebas

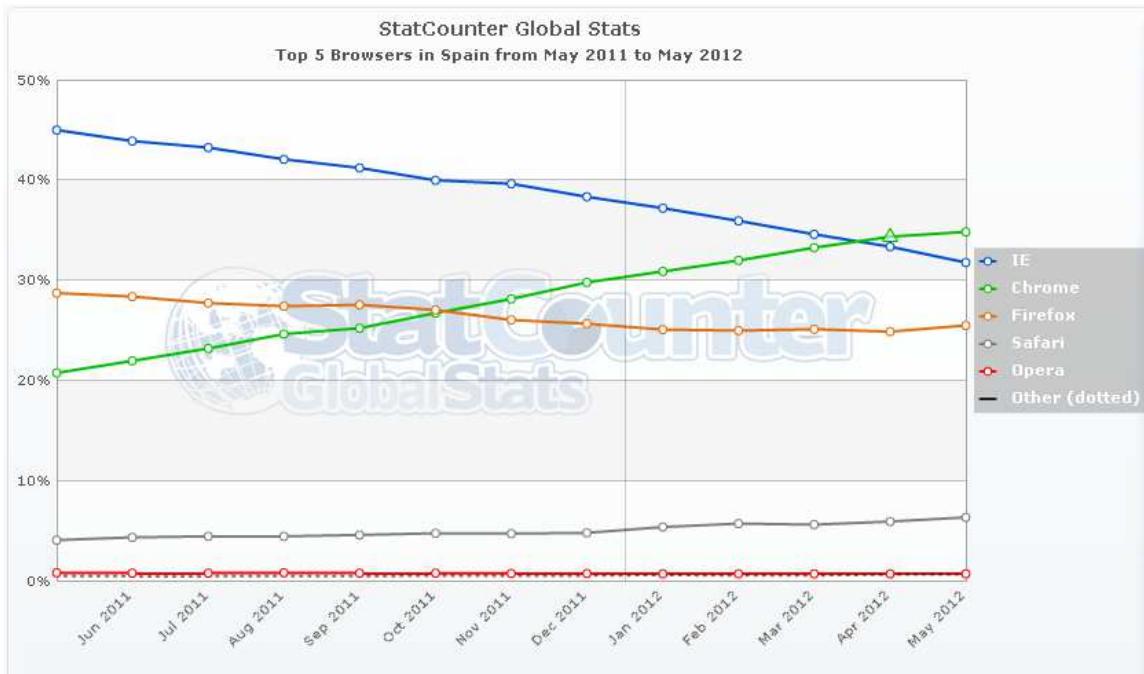
---

En este capítulo se va a describir la batería de pruebas realizada a la aplicación con el objetivo de encontrar posibles errores que no se han percibido durante el desarrollo de la misma.

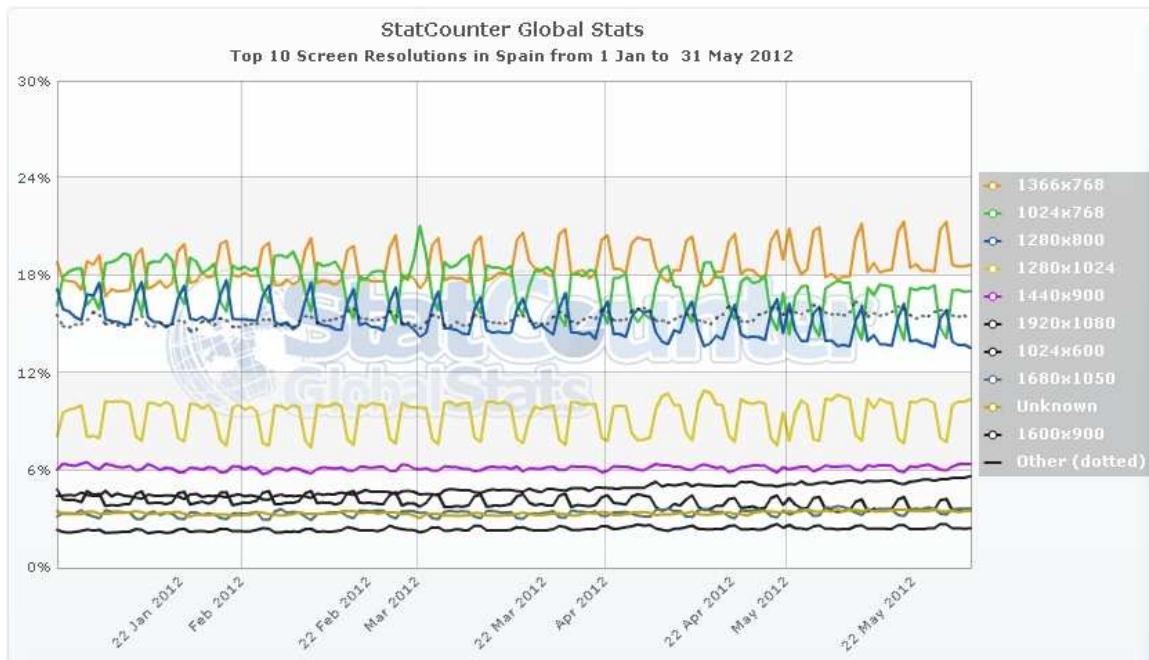
Durante el proceso de desarrollo de una aplicación por muy bueno que sea el equipo de desarrollo siempre aparecerán errores. El objetivo principal de la batería de pruebas es que una persona externa al desarrollo pruebe todos los elementos de la aplicación para así no sólo encontrar todos los errores si no también poder detectar incoherencias en la interfaz de usuario que la hagan poco intuitiva para una persona ajena al desarrollo, de esta manera cuando la aplicación llega al usuario final podemos garantizar una cierta estabilidad de la misma dejando así una buena imagen de ésta al usuario final.

Dicho esto las pruebas a realizar sobre la aplicación son las siguientes:

**Cross-Browser:** Se debe probar el funcionamiento de todos los módulos de la aplicación en los distintos navegadores del mercado de distintos dispositivos para comprobar la compatibilidad que tiene la aplicación con cada uno de ellos. Los navegadores que se probaran serán: **IE7, IE8, IE9, Chrome, FireFox, Opera, Safari(PC,IPAD)**. Dicha prueba se extenderá a las otras, es decir todas las pruebas funcionales deberían hacerse en todos los navegadores para comprobar que todos los módulos de la aplicación funcionan en todos los entornos. En el grafico adjunto se muestra la distribución del uso de los distintos navegadores en el último año en España. De aquí sacamos la importancia de tener la herramienta disponible en el máximo número de entornos posibles. Ya que como vemos en el grafico si sólo nos centramos en un navegador apenas llegaremos a un tercio de los usuarios.



**Resolución de pantalla:** También será importante hacer pruebas de visualización de la aplicación con distintas resoluciones de pantalla, así podremos asegurar una correcta visualización de la misma para la mayoría de los clientes. Las resoluciones que vamos a probar son: 1024x768, 1366x768, 1280x800, 1280x1024, IPAD. Con estas resoluciones nos aseguramos una penetración en días laborables de casi un 60%.



**Pruebas de sesión:** Se deberá probar la aplicación durante tiempo prolongado tanto de uso como de espera para establecer que los *timeout's* se cumplen según los requisitos. Estas pruebas son importantes debido a la gran cantidad de conexiones a distintos recursos que tiene la aplicación. Es necesario saber si todas las conexiones se mantienen activas durante la duración de la sesión y en el caso de caer se activen cuando sean necesarias sin producir errores o funcionamientos anómalos de la aplicación.

**Pruebas seguridad:** Hay que realizar pruebas con todos los tipos de cuentas de usuario disponibles para asegurarse que la aplicación sólo deja ver o hacer aquello por lo que el usuario tiene permisos.

- Cada usuario sólo podrá ver el esquema del cliente al que pertenece.
- Sólo los usuarios con permiso de Builder podrán ejecutar reportBuilder y ver el apartado de MisInformes, el cual debe mostrar sólo los informes del cliente al que pertenece el usuario.
- Sólo los administradores podrán crear usuarios, los usuarios creados por un administrador desde la aplicación sólo podrán acceder al cliente al que pertenece el administrador que lo ha creado.

- Todos los usuarios tienen de poder recuperar su contraseña o cambiarla cuando deseen, una vez cambiada la contraseña sólo podrán acceder con la nueva contraseña.

**Pruebas de uso:** Habrá que realizar un conjunto de pruebas de uso de la aplicación para asegurarse del buen funcionamiento de ésta:

- *Navegación Menú:* Se debe probar el uso indiscriminado del menú navegando sin sentido por el menú y analizando si este responde correctamente a nuestras órdenes:
  - Abrir filtros de informe al hacer *click* en baldosa de informe.
  - Desplegar siguiente nivel y plegar otros niveles al hacer *click* en baldosa de directorio.
  - Plegar niveles al hacer *click* en baldosa de directorio abierto.
  - Generar *Scroll* en directorio con tamaño superior al espacio del menú.
  - Ocultar subdirectorios al salir del área del menú.
  - Mostrar subdirectorios abiertos al entrar en el área del menú.
- *Funcionamiento de filtros:* Probar que todos los controles de filtros funcionan y se visionan correctamente.
- *Ejecución de Informes/Mapas:* Comprobar que los informes se ejecutan correctamente y se abren con el tamaño que deben.
- *Funcionamiento Ventanas Informe:* Probar el funcionamiento de las ventanas de informe:
  - Se deslizan bien por el área de trabajo.
  - No pueden salir del área de trabajo.
  - Se pueden redimensionar correctamente.
  - Se cierran correctamente.
  - Se recuperan sin problemas.

A parte de todas estas pruebas realizadas al concluir la aplicación, durante el desarrollo de la misma se han ido probando los distintos módulos de la aplicación para garantizar su buen funcionamiento sin embargo las pruebas que hace el desarrollador de una aplicación siempre deja algunas lagunas ya que suele hacer aquellas cosas en las que ha pensado y por lo tanto no producen errores ya que estos se suelen encontrar en las casuísticas no previstas por el programador.

## 7 Conclusiones y ampliaciones

---

### 7.1 Conclusiones

Llegados a este punto y con el proyecto que nos ocupa finalizado, ya podemos entrar a valorar el cumplimiento de las metas planeadas en el inicio del mismo.

En mi opinión se han conseguido los objetivos planteados en la fase de análisis del proyecto ya que la aplicación resultante de todo el proceso de desarrollo cumple con todas las especificaciones sugeridas durante el mismo.

Como ya se ha comentado en el capítulo de objetivos la aplicación resultante de este proyecto cumple con los requisitos marcados para dar por buenos los objetivos:

- **Implantación y ampliación de los proyectos fácil y rápida:** Gracias al estrecho vínculo creado entre reportServer y la aplicación se cumple este requisito ya que aplicando el coste de implantación de un nuevo cuadro de mando a un cliente se verá reducido a las jornadas de creación de la base de datos y a las necesarias para personalizar los informes reduciendo la creación del cuadro a prácticamente cero.
- **Visualización de todos los tipos de filtros:** Con el desarrollo de los diferentes controles de usuario se pueden gestionar sin problemas todos los tipos de filtro.
- **Capacidad para poder visualizar varios informes a la vez:** Con la nueva interfaz por ventanas conseguimos gestionar y visualizar varios informes de forma simultánea pudiendo enriquecer el análisis de un informe con información de otros.
- **Facilitar la creación de mapas:** Gracias a la programación de informes geográficos bajo googleMaps este requisito ha sido cumplido.
- **Front-End agradable a la vista y sencillo de utilizar:** El nuevo *Front-End* es simple y sigue el diseño de las nuevas tendencias.
- **Modular:** Se ha separado cada modulo de manera que reutilizar o ampliar uno solo sea posible.
- **Repositorio de sesión:** Como ya se ha visto en esta memoria ha sido desarrollado con éxito.
- **Permisos:** Utilizando la seguridad de .NET podemos administrar los permisos de cada usuario.
- **Gestión de usuarios.** Se ha programado un modulo para poder dar de alta a nuevos usuarios. El resto de operaciones son gestionadas por un consultor de la empresa para controlar las licencias.

En cuanto a los requerimientos no funcionales de la aplicación los cuales tenían que ver con el diseño de la misma también han sido realizados ya que en estos momentos la aplicación tiene un diseño totalmente acorde con el resto de aplicaciones en desarrollo en la empresa. Este punto ha sido difícil de cumplir ya que el otro equipo de desarrollo de aplicaciones de la empresa estaba ubicado en las oficinas de Madrid y estos han ido

cambiando el diseño a su gusto sin informar previamente de manera que parte de mi trabajo se ha visto duplicado debido a tener de deshacer un diseño muy avanzado para poder acercarme de nuevo al diseño corporativo.

A nivel personal gracias al desarrollo de esta aplicación he adquirido buenos conocimientos en jQuery que ha resultado ser un lenguaje de programación muy útil y funcional que me ha permitido realizar con facilidad conexiones asíncronas al servidor función que hasta la fecha realizaba mediante updatePanels que como ya he expuesto en esta memoria resultaban poco eficientes y tenía poco control sobre su funcionamiento interno, gracias a jQuery también he aprendido a mejorar la interfaz web de mi página de forma que sea mucho más agradable a la vista, en este sentido también he mejorado mis conocimientos en css. En cuanto a conocimientos adquiridos también cabe destacar el control sobre la api de googleMaps adquirido durante el desarrollo de este proyecto.

## 7.2 Desviaciones

Durante el desarrollo de este proyecto han surgido una serie de imprevistos que han afectado al *timing* inicial planteado durante el estudio de viabilidad, a continuación se van a detallar el origen de los retrasos surgidos y se va analizar el impacto que han tenido estos en la rentabilidad de la aplicación.

- **Interrupciones en el desarrollo:** Durante el desarrollo del proyecto he tenido varias interrupciones en el desarrollo producidas por necesidad de recursos en otros proyectos de la empresa. Dichas interrupciones generan un retraso en el *timing* y un aumento en las jornadas debido al coste de reiniciar la tarea detenida de nuevo. Debido a este tipo de interrupciones que durante el proyecto se han dado hasta tres veces el proyecto se ha retrasado 15 días, 11 debido al redireccionamiento de los recursos a otro proyecto y 4 debido al coste de retomar la dinámica del proyecto.
- **Incompatibilidad entre tecnologías:** Durante el desarrollo de las ventanas de informe fue imposible cargar el control de .net ReportViewer sin que afectará al funcionamiento de otros scripts de la pantalla principal, finalmente se decidió sacar todos los controles de .NET fuera de la pantalla principal y llamarlos en otras páginas que visualizaríamos mediante iFrames. Debido a este cambio se perdieron 8 jornadas de trabajo.
- **Cambios en el diseño:** Cuando la aplicación ya estaba prácticamente finalizada el otro grupo de trabajo se presentó con un diseño totalmente distinto al que seguimos hasta el momento, a la dirección de la empresa le gusto el nuevo diseño y tuve que adaptar la aplicación a ese nuevo look perdiendo en el proceso 6 jornadas más.

El daño podría haber sido mucho mayor, pero afortunadamente el tiempo perdido en los cambios en el diseño y por la incompatibilidad entre tecnologías fue muy bajo gracias al diseño modular de la aplicación, que permitió conservar el núcleo de las funcionalidades ya programadas y centrarse solo en los cambios a realizar, ya que la mayor parte del tiempo perdido en el problema de compatibilidad, fue debido a la multitud de pruebas que se hicieron hasta llegar a la conclusión de que había que separar las tecnologías en distintas páginas.

Al final el proyecto a terminado prácticamente dos meses más tarde, a nivel interno este retraso era asumible porque el proyecto pertenecía a la I+D de la empresa y no dependida de ningún plazo de entrega, por lo que hace al tiempo total de proyecto a aumentado en ocho jornadas aumentando así el coste de este. Pero como el análisis de viabilidad demostró que teníamos mucho margen de coste Beneficio dicho incremento de costes no afecta a la viabilidad del proyecto por lo que todos los retrasos sufridos son perfectamente asumibles.

### 7.3 Ampliaciones

Durante el desarrollo de la aplicación han ido surgiendo nuevas ideas para mejorar más las funcionalidades de la misma. Algunas de estas mejoras se han implementado con la aplicación y se han relatado en esta memoria debido a la fácil implementación de las mismas, un claro ejemplo de estas ampliaciones ya incluidas en el proyecto es el repositorio de datos.

Por otro lado hay ampliaciones que mejorarían la funcionalidad de la aplicación que se ha decidido aparcar para futuras ampliaciones de la misma debido a la complejidad de su desarrollo, entre las ampliaciones pensadas tenemos las siguientes:

- Vista Mosaico: Se propone crear un botón que al presionarlo muestre todos los informes en pantalla en forma de mosaico.
- Mis favoritos: Hemos pensado que sería de gran utilidad que los usuarios pudiesen guardar en sus cuentas perfiles de área de trabajo con informes prerenderizados de manera que cuando entraran pudieran abrir un perfil y aparecieran automáticamente el conjunto de informes que él quiera en pantalla distribuidos en forma y tamaño como haya establecido e incluso guardando en el repositorio posibles informes secundarios listos para abrirse.
- Añadir a la aplicación una página para poder hacer extracciones de datos para sacar targets a partir de las conclusiones sacadas del cuadro de mando.
- En el futuro habrá que migrar el servidor de informes de sql Server 2008 R2 a Sql Server 2012, es importante seguir en la cresta de la tecnología para que la aplicación no quede obsoleta.

Todas estas ampliaciones empezaran a desarrollarse en el momento en que tengamos todos los clientes migrados a esta aplicación ya que el desarrollo de la aplicación no puede pararse debido a la velocidad con la que avanza el mundo de la informática.

## 8 Bibliografía

---

Páginas web:

- <http://jquery.com/> (Marzo-Mayo 2012)
- <http://jqueryui.com/> (Marzo-Mayo 2012)
- <https://developers.google.com/maps/?hl=es> (Mayo 2012)
- <http://msdn.microsoft.com/es-es/library/ms123401> (Febrero-Mayo 2012)
- <http://gs.statcounter.com/> (Junio 2012)
- <http://mundogeek.net/archivos/2010/04/21/tutorial-rapido-de-jquery/> (Febrero 2012)
- [http://www.filamentgroup.com/lab/date\\_range\\_picker\\_using\\_jquery\\_ui\\_16\\_and\\_jquery\\_ui\\_css\\_framework/](http://www.filamentgroup.com/lab/date_range_picker_using_jquery_ui_16_and_jquery_ui_css_framework/) (Mayo 2012)
- <http://www.forosdelweb.com/f13/> (Febrero-Mayo 2012)

## 9 Agradecimientos

---

No podía terminar esta memoria sin un recuerdo especial hacia todas aquellas personas que me han ayudado de forma directa o indirecta a llevar a cabo este proyecto.

Para empezar me gustaría acordarme de mi madre, que durante todos estos años en los que no me decidía a realizar el PFC me lo ha recordado todos y cada uno de los días hasta el aburrimiento, este recordatorio constante ha sido uno de los grandes culpables de que me decidiera a dar el paso final para finalizar la carrera.

A mi padre que mientras mi madre persistía él ha sido la paciencia, la comprensión y la confianza, y poder saber que siempre está allí para todo.

También tengo un agradecimiento muy especial para Sandra, siempre escuchando mis ideas, siempre aportando su punto de vista y generando esa crítica constructiva que me permitía ir mejorando día a día el proyecto hasta llegar a su versión definitiva, sin duda sin ella esta aplicación luciría mucho menos.

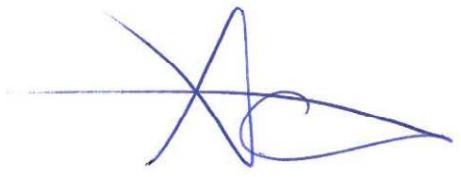
No puedo dejar de agradecer a Enrique el tiempo que me ha dado para que pudiera desarrollar el proyecto tranquilamente y permitir dar rienda suelta a mis ideas.

A Xavier Verge que como tutor de este proyecto ha creído en él desde el día en que lo vio y me ha guiado sabiamente en esta carrera para la entrega, sin sus consejos y su confianza nunca habría llegado a buen puerto.

A Eli por la confianza que siempre me ha transmitido tanto en la presencia como en la ausencia.

Un recuerdo especial para todos mis compañeros de trabajo que me soportan cada día, gracias por ayudarme siempre que lo he necesitado.

Por último el más sincero agradecimiento a Paola que siempre a mi lado me ha dado tiempo, consejos, serenidad, risas, optimismo, seriedad y todo el cariño del mundo, ha conseguido poner orden en el caos y me ha dado la fuerza necesaria para terminar este proyecto que sin ninguna duda de no ser por ella seguiría guardado en el armario de mis cosas por hacer, muchas gracias de todo corazón con ella al lado todo resulta mucho mas fácil.



**Xavier Vidal Zamarreño**