UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA
UNAN – León

UNIVERSIDAD DE ALCALÁ DE HENARES
Escuela Politécnica

TESIS DE MAESTRÍA

DESARROLLO DE UN ADDIN PARA MONODEVELOP 2.8 QUE PERMITA LA IMPLEMENTACIÓN DEL SERVICIO DE SUSCRIPCIONES DE ASP.NET 2.0 EMPLEANDO POSTGRESQL COMO GESTOR DE BASE DE DATOS

Autor: Ing. Denis Leopoldo Espinoza Hernández

Tutor: Ph.D. Francisco Javier Ceballos Sierra

LEÓN, NICARAGUA

UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA
UNAN – León

UNIVERSIDAD DE ALCALÁ DE HENARES
Escuela Politécnica

TESIS DE MAESTRÍA

DESARROLLO DE UN ADDIN PARA MONODEVELOP 2.8 QUE PERMITA LA IMPLEMENTACIÓN DEL SERVICIO DE SUSCRIPCIONES DE ASP.NET 2.0 EMPLEANDO POSTGRESQL COMO GESTOR DE BASE DE DATOS

Autor: Ing. Denis Leopoldo Espinoza Hernández
Tutor: Ph.D. Francisco Javier Ceballos Sierra

TRIBUNAL

Presidente: M.Sc. Javier de Pedro Carracedo

Secretario: M.Sc. Raúl Hermógenes Ruiz Cabrera

Vocal: M.Sc. Valeria Mercedes Medina Rodríguez

Fecha: 16 de julio de 2012

A DIOS, por estar siempre conmigo guiándome en cada uno de los momentos de mi vida indicándome siempre el camino que debo seguir.

A la Virgen María, por acompañarme y estar siempre a mi lado enseñándome a ser paciente y obediente en la misión que me ha tocado.

A mis padres, por ser para mí el mejor ejemplo de amor, entrega y esfuerzo pues de ellos aprendí el valor del estudio y la docencia.

A mi tutor y maestro Francisco Javier Ceballos, no sólo por su idea para la realización de este trabajo sino por la paciencia y consejos a lo largo del mismo.

A mis hermanos, por ser una inspiración para superarme y por su apoyo en las diferentes áreas de mi vida.

A mis alumnos, por ser para mí uno de los principales motivos para seguir estudiando y aprendiendo cada día más.

Índice de contenidos

Índice de figuras

I. Resumen

Este documento tiene como propósito explicar las etapas realizadas en el desarrollo de un addin para el IDE MonoDevelop 2.8 que permita la implementación del servicio de suscripciones de ASP.NET 2.0 empleando PostgreSql como gestor de base de datos. Este proyecto nació a partir de la necesidad de implementar el servicio de suscripciones en MonoDevelop de una forma similar a la que se emplea en Visual Studio, sin embargo dicho entorno, no poseía ninguna herramienta que nos ofreciera esa posibilidad.

A lo largo de los diferentes capítulos de este trabajo se explica el método empleado para el análisis, diseño e implementación de este addin quedando definido claramente el alcance y necesidades que se deben de cumplir para su correcto funcionamiento.

Finalmente se ofrece un manual que explica la instalación y funcionamiento de este addin, las conclusiones a las cuales se ha llegado al finalizar su desarrollo y posibles líneas de trabajo futuro.

II. Introducción

2.1 Antecedentes

La autenticación es el proceso de descubrir y comprobar la identidad de un principal mediante el examen de las credenciales del usuario y la validación de las mismas consultando a una autoridad determinada. El servicio de suscripciones de ASP.NET permite almacenar las credenciales de un usuario que, al ser empleadas junto con los controles de inicio de sesión, permiten crear un completo sistema de autenticación. Este servicio básicamente proporciona soporte para crear nuevos usuarios, almacenar su información, autenticar a los usuarios que visiten el sitio web y administrar las contraseñas. (Ceballos, Enciclopedia de Microsoft Visual C#, 2009)

De forma predeterminada, el servicio de suscripciones está activado y el proveedor de base de datos es AspNetSqlProvider por lo cual toda la información queda almacenada en una base de datos SQL Server. Sin embargo .NET permite la implementación de otros proveedores a través de la creación de un par de clases que deriven de las clases MembershipProvider para el manejo de los usuarios (Managing Users by Using Membership, 2008) y la clase RoleProvider para el manejo de los roles y reglas de acceso (Managing Authorization Using Roles, 2008). En internet se pueden encontrar implementaciones de estas clases para diversos proveedores entre los cuales tenemos:

- **Membership and Role providers for MySQL:** Permite emplear el servicio de suscripciones utilizando como proveedor de base de datos MySQL. (Membership providers for MySQL, 2006) (Role provider for MySQL, 2006).
- **Membership and Role providers for PostgreSQL:** Basado en la implementación del proveedor para MySQL, se desarrollo el proveedor para PostgreSQL empleado en este proyecto. No se coloca la referencia porque está ha sido removida.

Para la gestión del servicio de suscripciones el entorno de desarrollo Visual Studio de Microsoft provee una herramienta de administración de sitios web que al ejecutarla configura automáticamente para la aplicación el servicio de suscripciones empleando como proveedor AspNetSqlProvider. Además de esto despliega un sitio web que permite

a los desarrolladores administrar de forma sencilla todo lo referente a las cuentas de usuarios, funciones y reglas de acceso de la aplicación que se está desarrollando.

Como alternativa a la plataforma .Net de Microsoft tenemos hoy en día el proyecto Mono el cual permite crear aplicaciones multiplataforma compatibles con Microsoft .Net Framework y a diferencia de Microsoft que tiene a SQL Server como gestor de base de datos por defecto, Mono ha seleccionado a PostgreSQL como su gestor de base de datos por ser el gestor más estable y robusto dentro del software libre. La plataforma Mono soporta aplicaciones ASP.NET 2.0 y desde ella se pueden escribir este tipo de aplicaciones empleando como entorno de desarrollo MonoDevelop.

MonoDevelop es un IDE primeramente diseñado para C# y otros lenguajes soportados por .NET. Éste permite a los desarrolladores escribir rápidamente aplicaciones de escritorio y ASP.NET sobre Linux, Windows y Mac OSX. También hace que sea fácil portar a Linux aplicaciones creadas en Visual Studio (MonoDevelop, 2011). Una de las principales ventajas de MonoDevelop es que permite a los desarrolladores añadirle nuevas funcionalidades a través de la creación de extensiones mejor conocidas como addins. Actualmente, la mayoría de las funcionalidades aportadas por los desarrolladores de MonoDevelop son addins lo que ha potenciado la creación de repositorios desde los cuales se pueden instalar nuevas extensiones (Addin development basics, 2010).

Para la ejecución de aplicaciones ASP.NET sobre Mono puede hacerse de dos formas, instalando el servidor de aplicaciones XSP o instalando el módulo mod_mono de Apache. (Ceballos, Aplicaciones .NET multiplataforma, 2008)

2.2 Definición del problema

Aunque MonoDevelop permite el desarrollo de aplicaciones ASP.NET, posee ciertas limitaciones que no favorecen un desarrollo ágil. Una de las limitantes más conocidas es no poseer un diseñador gráfico para el desarrollo de las interfaces de aplicaciones web, sin embargo dicho problema es muchas veces solventado por el empleo de otras herramientas.

Con el tema de la seguridad de las aplicaciones web aunque la plataforma Mono soporta el servicio de suscripciones (manejo de usuarios, roles y reglas de acceso), MonoDevelop no aporta ninguna funcionalidad para facilitar la configuración y administración de este servicio lo que obliga al desarrollador del sitio a realizar estas operaciones de forma manual.

Actualmente para que un usuario pueda emplear el servicio de suscripciones dentro de su aplicación web en la plataforma Mono debe:

- Conocer perfectamente cómo funciona este servicio desde todos los aspectos (clases implicadas, gestor de base de datos, proveedor de acceso a la base a la datos, sintaxis del web.config para las sección de autorización, etc.) y no sólo desde el punto de vista funcional.
- Creación de la base de datos con la estructura de tablas necesarias para dar soporte a la gestión de los usuarios y roles.
- Creación de las clases Provider (Memberchip y Role) para el gestor de base de datos seleccionado.
- Creación y/o modificación del web.config para que este posea las secciones membership (para el manejo de usuarios) y roleManager (para el manejo de los roles y reglas de acceso).
- Acceder a la base de datos para agregar usuarios y roles a la aplicación así como para asociar usuarios a roles.
- Crear y/o modificar en el web.config la sección authorization para indicar los permisos de acceso a cada carpeta del sitio web.

Todo lo anterior provoca que muchos desarrolladores opten por crear ellos mismos sus mecanismos de autenticación y autorización en vez de emplear los que ASP.NET nos ofrece. Frente a la problemática que nos plantea el uso del servicio de suscripciones en la plataforma Mono y ante las facilidades que ofrece MonoDevelop para el desarrollo de extensiones y su fácil integración con PostgreSQL surgen las siguientes preguntas generales y específicas:

¿Es posible desarrollar un addin para MonoDevelop 2.8 que permita la implementación del servicio de suscripciones de ASP.NET empleando PostgreSQL como gestor de base de datos?

- ¿Qué mecanismo se puede emplear para la creación de la base de datos que dará soporte a la información de las cuentas de los usuarios y los roles?
- ¿Cómo pueden realizarse las modificaciones que deben llevarse a cabo dentro de los proyectos para que el servicio de suscripciones funcione correctamente?
- ¿Es posible desarrollar una aplicación web similar a la empleada en Visual Studio para la administración del sitio web?

2.3 Justificación

En base a los problemas antes planteados se hace necesaria la creación de un addin para MonoDevelop que permita implementar el servicio de suscripciones en las aplicaciones ASP.NET de forma parecida a como este servicio se implementa en Visual Studio, empleando PostgreSQL para el almacenamiento de la información de los usuarios y los roles de la aplicación.

2.3.1 Originalidad

Anteriormente no se ha desarrollado ningún addin o herramienta que permita solventar esta necesidad por lo que este trabajo será de gran ayuda a los desarrolladores de aplicaciones web que emplean MonoDevelop para la creación de aplicaciones ASP.NET.

2.3.2 Alcance

El addin permitirá a los desarrolladores configurar el servicio de suscripciones para sus aplicaciones web al ofrecer las siguientes posibilidades:

- Seleccionar la base de datos de PostgreSQL que se empleará para el almacenamiento de la información de los usuarios y los roles.
- Automáticamente, realizará todas las modificaciones necesarias (web.config y referencias) para que el servicio de suscripciones quede correctamente configurado.
- Permitirá visualizar una herramienta web que permitirá agregar y administrar los usuarios de la aplicación, habilitar o deshabilitar los roles de la aplicación así como administrar los ya existentes, crear y/o modificar reglas de acceso a las diferentes carpetas de la aplicación.

2.3.3 Producto

El producto entregable es un addin que incorpora al MonoDevelop un menú llamado Subscription Service que posee dos elementos Database Settings (para la configuración

del servicio de suscripciones) y Subscription Settings (para la administración del sitio web). Este addin cumplirá con las siguientes características:

Sencilla: Porque la utilización de este addin no debe de requerir muchos pasos y la necesidad y ejecución de cada uno de ellos debe estar claramente definido.

Familiar: Porque deben utilizarse interfaces similares a las empleadas en MonoDevelop cuando se requiera la visualización de algún formulario y la aplicación que se ejecuta para la administración del sitio web, debe ser similar a la empleada en Visual Studio.

Seguro y estable: Porque el addin será ampliamente probado para garantizar que su uso no represente ningún riesgo para de seguridad para los desarrolladores.

2.3.4 Impacto

Con la creación de este addin, se provee a los desarrolladores una herramienta que no solo les ayudará en la gestión de los sitios web sino que a su vez, se promueve el uso de herramientas libres cada vez mejor equipadas. También tendrá un impacto en la docencia al permitir a los docentes mostrar a sus estudiantes que en MonoDevelop se posee una herramienta similar a la de Visual Studio para la implementación del servicio de suscripciones.

2.4 Objetivos

2.4.1 Objetivo general

Desarrollar un addin para MonoDevelop 2.8 que permita la implementación del servicio de suscripciones de ASP.NET empleando PostgreSQL como gestor de base de datos.

2.4.2 Objetivos específicos

- Determinar el mecanismo a emplear para la creación de la base de datos que dará soporte a la información de las cuentas de los usuarios y los roles de la aplicación.

- Definir e implementar la lógica del addin que permita realizar en los proyectos las modificaciones necesarias para que el servicio de suscripciones funcione correctamente.

- Desarrollar una herramienta web similar a la empleada en Visual Studio para la administración del sitio web que permita la gestión de los usuarios, roles y reglas de acceso de la aplicación.

III. Recursos tecnológicos

A continuación se hace una descripción de los recursos tecnológicos empleados en el desarrollo de este proyecto.

3.1 ASP.NET

ASP.NET es una plataforma que incorpora una serie de características y utilidades para diseñar aplicaciones web: formularios web o servicios web. Las aplicaciones web son, por definición, independientes de la plataforma; es decir, los usuarios pueden interactuar con la aplicación independientemente del tipo del navegador que utilicen. (Ceballos, Aplicaciones .NET multiplataforma, 2008)

3.2 Proyecto Mono

Mono es una plataforma de software que permite a los desarrolladores crear fácilmente aplicaciones multiplataforma. Patrocinado por Xamarin, Mono es una implementación Open Source de Microsoft .NET Framework basado en los estándares del ECMA para C# y el Common Language Runtime. Una familia cada vez mayor de soluciones y una comunidad activa y entusiasta está ayudando a posicionar a Mono como la opción principal para el desarrollo de aplicaciones Linux. (Ceballos, Aplicaciones .NET multiplataforma, 2008)

3.3 MonoDevelop 2.8

MonoDevelop es un IDE primeramente diseñado para C# y otros lenguajes soportados por .NET. Éste permite a los desarrolladores escribir rápidamente aplicaciones de escritorio y ASP.NET sobre Linux, Windows y Mac OSX.

Dentro de las principales características que MonoDevelop

- Multiplataforma: Compatible con Linux, Windows y Mac OSX.

- Edición de texto avanzada: Soporte de completación de código para C# 3, plantillas de código y plegado de código.
- Área de trabajo configurable: Diseño de ventana totalmente personalizable, claves definidas por el usuario y herramientas externas.
- Soporte para múltiples lenguajes: C#, Visual Basic .NET, C/C++, Vala
- Depurador integrado: Para depurar aplicaciones Mono y nativas.
- Diseñador Visual para GTK#: Construir fácilmente aplicaciones GTL#.
- ASP.NET: Crear proyectos web con soporte completo para completación de código y pruebas sobre XSP, el servidor web de Mono.
- Otras herramientas: Control de código fuente, integración con makefile, pruebas unitarias, empaquetado y despliegue, localización.

(MonoDevelop, 2011)

3.4 MonoDevelop API

La API de MonoDevelop es extensa y en constante cambio. Dicha API se divide en 3 capas:

- **La capa de núcleo**, que proporciona servicios básicos tales como el registro, monitorización de progreso o manejo de archivos que se utilizan dentro del IDE.
- **La capa de proyectos**, implementa el modelo de objetos del proyecto. Proporciona métodos para leer, escribir y construir proyectos así como un servicio de análisis.
- **La capa superior**, es el propio IDE.

Vale la pena señalar que las capas principales y los proyectos no están vinculados a la aplicación del IDE por lo que es posible crear aplicaciones independientes que hacen uso de esos servicios.

(Lluis, 2010)

3.5 Mono.Addins

Mono.Addins es un framework genérico para la creación de aplicaciones extensibles y para crear los complementos que ampliaran dichas aplicaciones. Este framework ha sido diseñado para ser útil a una amplia gama de aplicaciones, desde aplicaciones sencillas

con necesidades de ampliación pequeña, hasta aplicaciones complejas que requieren soporte para grandes estructuras de complementos.

Las principales características de Mono.Addins son:

- Compatible con las descripciones de addins empleando atributos personalizados (para extensiones simples y comunes) o ficheros XML (para extensiones más complejas).
- Soporte para jerarquías de Addins donde los addins puedan depender de otros.
- Activación y desactivación de los addins en tiempo de ejecución.
- Permite implementación de bibliotecas extensibles.
- Soporta localización de addins.
- Proporciona una API para acceder a la descripción de los addins lo que permite la construcción de herramientas de desarrollo y documentación para su manejo.
- Además del motor básico de addins, provee una biblioteca de instalación para ser usada por las aplicaciones que deseen ofrecer a los usuarios, funciones básicas de administración de addins, tales como habilitar/deshabilitar addins, o instalación de addins desde repositorios en línea.

(slluis, 2011)

MonoDevelop ha sido desarrollado empleando este framework lo cual permitió el desarrollo del Addins para este entorno.

3.6 Servidor de aplicaciones XSP

XSP es un servidor web independiente escrito en C# que puede ser usado para correr aplicaciones ASP.NET con poco esfuerzo. XSP trabaja tanto con el runtime de Mono como el de Microsoft. (Mono-project, 2010)

3.7 PostgreSQL

Es un poderoso sistema gestor de base de datos open source. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación por su fiabilidad, integridad de datos y correcciones. (Group, 2011)

IV. Análisis de requisitos

Luego de observar la necesidad de proveer a MonoDevelop de una herramienta para la configuración y administración del servicio de suscripciones de una forma similar a la provista por Visual Studio, se ha extraído una lista de requisitos que el addin debe cumplir y los cuales se detallan en esta sección.

4.1 Alcance

El nombre con el cual el addin será conocido es Subscription Services, y este ofrecerá a los usuarios de MonoDevelop dos opciones que permitirán:

- Configurar dentro del proyecto, la base de datos en postgresql que será usada por la aplicación para almacenar la información de los usuarios y los roles que ésta utiliza. A su vez está opción configurará el web.config y agregará las referencias necesarias.
- Permitirá administrar los usuarios, roles y reglas de acceso de nuestra aplicación de una forma sencilla a través de una interfaz similar a la empleada en Visual Studio.

4.2 Modelo de casos de uso

El modelo de casos de uso es un mecanismo ampliamente empleado para la captura de requisitos de usuario ya que permite representar gráficamente, lo que el usuario espera de la aplicación. Dentro de nuestro proyecto, hemos encontrado un total de 10 casos de uso que engloban todas las opciones para las que el usuario puede emplear nuestro software:

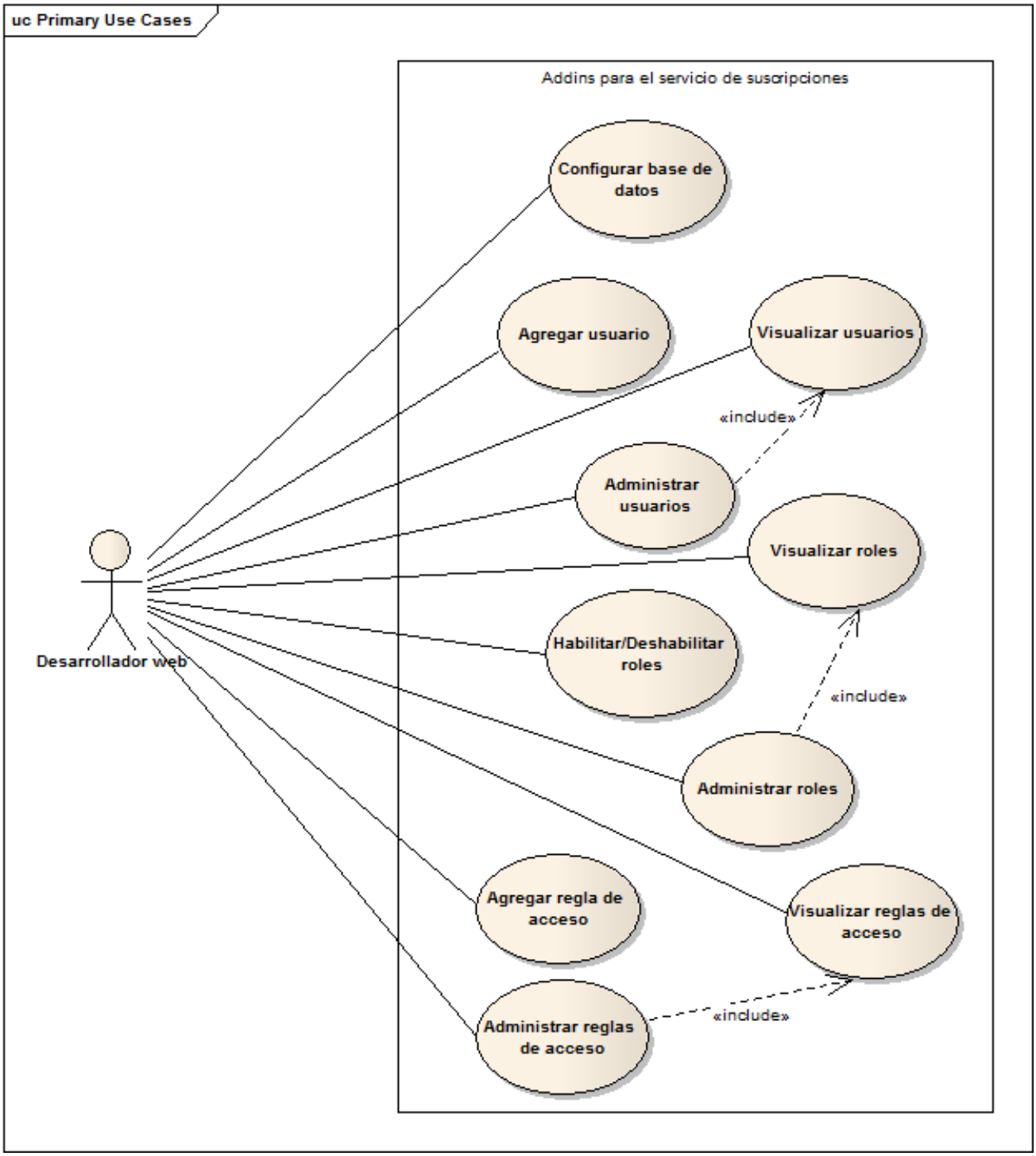

Figura 1 Diagrama de casos de uso

4.3 Descripción de actores

Nombre del actor:	Desarrollador web
Definición	Es el único actor del sistema. Es todo usuario que desarrolle aplicaciones ASP.NET en MonoDevelop y que desee agregar a su proyecto el servicio de suscripciones.
Notas:	Es capaz de acceder a todas las funcionalidades que ofrece el addin.

4.4 Descripción de casos de uso

A continuación se detallan cada uno de los casos de uso encontrados en nuestro sistema.

4.4.1 Caso de uso 1: Configurar base de datos

- **Definición:** Configura el servicio de suscripciones en la aplicación usando como base de datos para el almacenamiento de la información la indicada por el usuario.

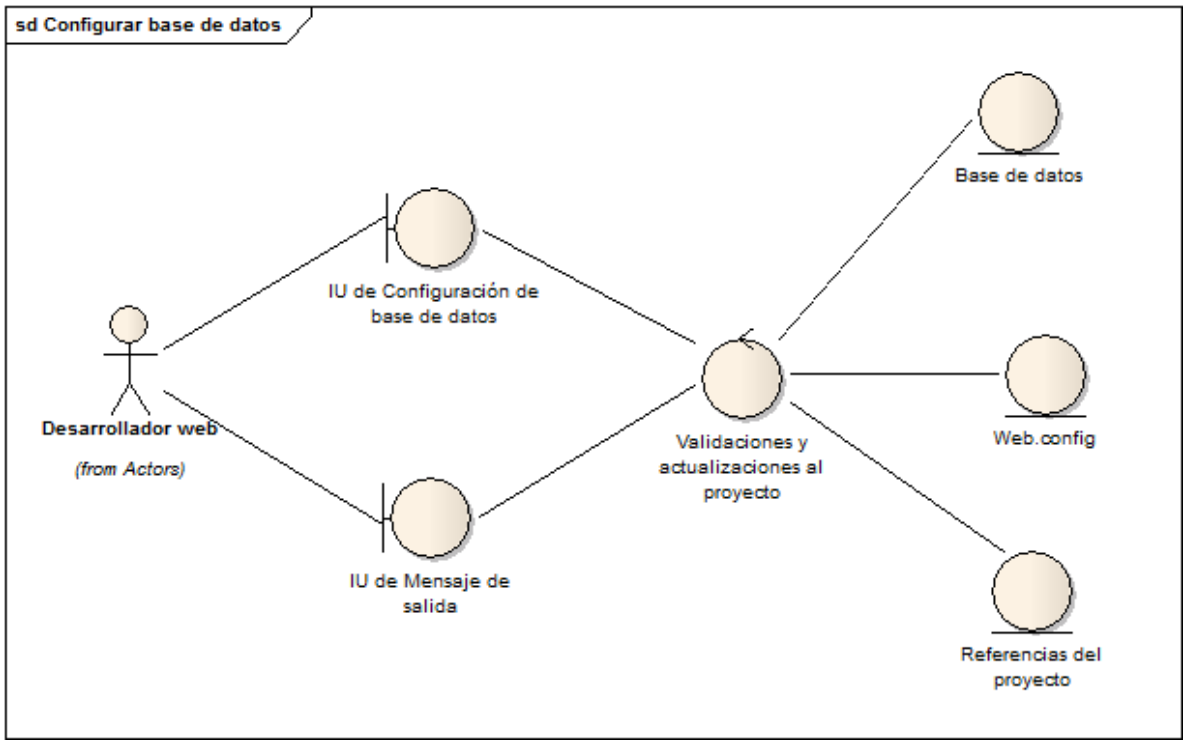

Figura 2 Modelo de análisis para el caso de uso configurar base de datos

Descripción del modelo de análisis:

1. Presentación de una interfaz a través de la cual se indica la dirección, el puerto, el usuario, la clave y la base de datos en PostgreSQL a emplear para almacenar la información de los usuarios.
2. Se realiza la validación de la información introducida y con ella se realizan las siguientes operaciones:
    a. Creación de las tablas dentro de la base de datos indicada para dar soporte a la gestión de los usuarios y los roles.
    b. Creación y/o modificación del web.config para agregar y configurar las secciones que dan soporte al servicio de suscripciones.
    c. Agregar al proyecto, las referencias necesarias.
3. Envío de un mensaje de éxito o error al usuario.

4.4.2 Caso de uso 2: Agregar usuario

- **Definición:** Permite agregar un nuevo usuario a la aplicación.

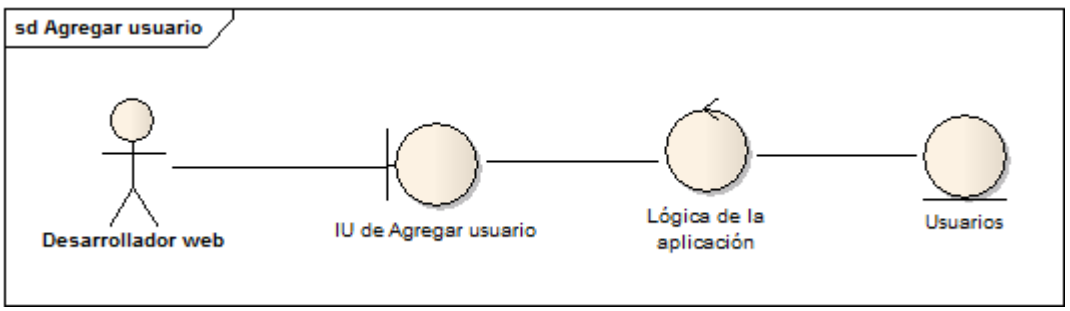

Figura 3 Modelo de análisis para el caso de uso agregar usuario

Descripción del modelo de análisis:

1. Presentación de la interfaz que permite agregar un nuevo usuario.
2. Lógica de la aplicación para realizar validaciones.
3. Agregar el usuario a la base de datos si todas las validaciones fueron correctas o enviar un mensaje de error en caso contrario.

4.4.3 Caso de uso 3: Visualizar usuarios

- **Definición:** Visualiza la lista de todos los usuarios agregados a la aplicación.

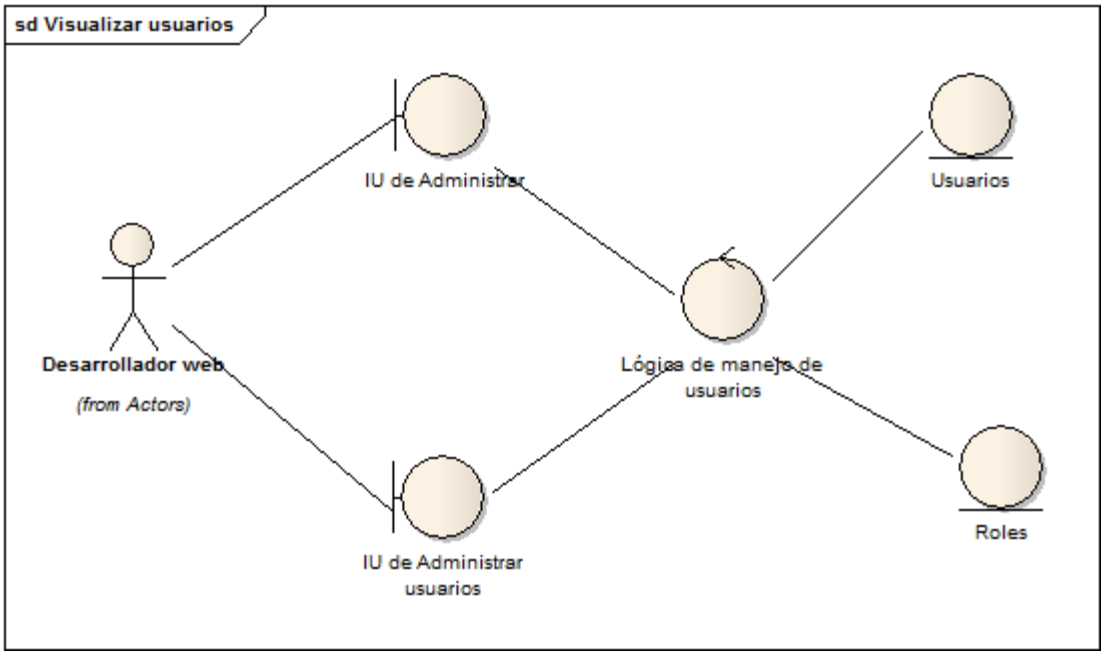

Figura 4 Modelo de análisis para el caso de uso visualizar usuarios

Descripción del modelo de análisis:

1. Presentación de la interfaz principal de la aplicación desde donde se selecciona la opción “Administrar usuarios”.
2. La lógica de la aplicación se encarga de obtener desde la base de datos, la información de usuarios y roles necesarios para la correcta visualización de la información.
3. Visualización de la interfaz “Administrar usuarios” donde se visualiza la información de los usuarios.

4.4.4 Caso de uso 4: Administrar usuarios

- **Definición:** Permite realizar operaciones sobre los usuarios almacenados en el sistema. Estas operaciones son: edición, borrado y asignación de roles.

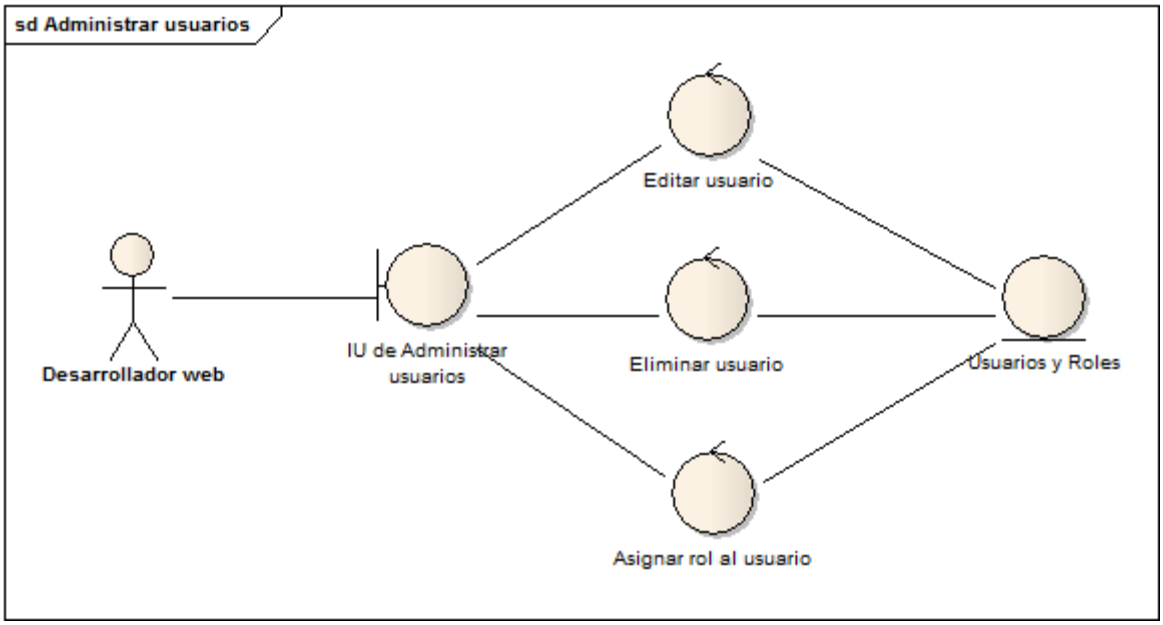

Figura 5 Modelo de análisis para el caso de uso administrar usuarios

Descripción del modelo de análisis:

1. Presentación de la interfaz “Administrar usuarios” desde la cual el usuario puede realizar las opciones antes mencionadas.
2. Ejecución de la opción seleccionada que puede ser:
    a. Editar usuario, en este caso se edita la descripción y el estado (activo o inactivo del usuario).
    b. Eliminar usuario, el usuario será eliminado de la base de datos.
    c. Asignar rol al usuario, permite asignar al usuario seleccionado uno o varios de los roles existentes en el sistema.
3. Actualizar la información de manera persistente en la base de datos.

4.4.5 Caso de uso 5: Habilitar / Deshabilitar roles

- **Definición:** Permite habilitar o deshabilitar la funcionalidad de roles en la aplicación. Esta información se encuentra almacenada dentro de la sección roleManager del web.config.

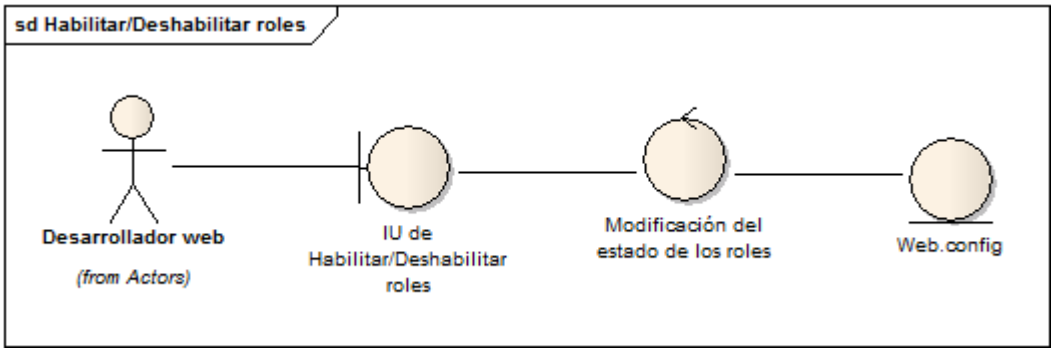

Figura 6 Modelo de análisis para el caso de uso habilitar / deshabilitar roles

Descripción del modelo de análisis:

1. Presentación de la interfaz “Habilitar/Deshabilitar roles” la cual expondrá una funcionalidad o la otra en función del estado actual.
2. Se actualizará el estado de la interfaz en función de la opción seleccionada: si la opción seleccionada fue habilitar roles, se habilitará la administración de los roles y todo lo referente al manejo de las reglas; por el contrario, si la opción fue deshabilitar roles, se deshabilitarán las opciones antes mencionadas.
3. Se actualizará el estado del web.config.

4.4.6 Caso de uso 6: Visualizar roles

- **Definición:** Visualiza la lista de todos los roles de la aplicación.

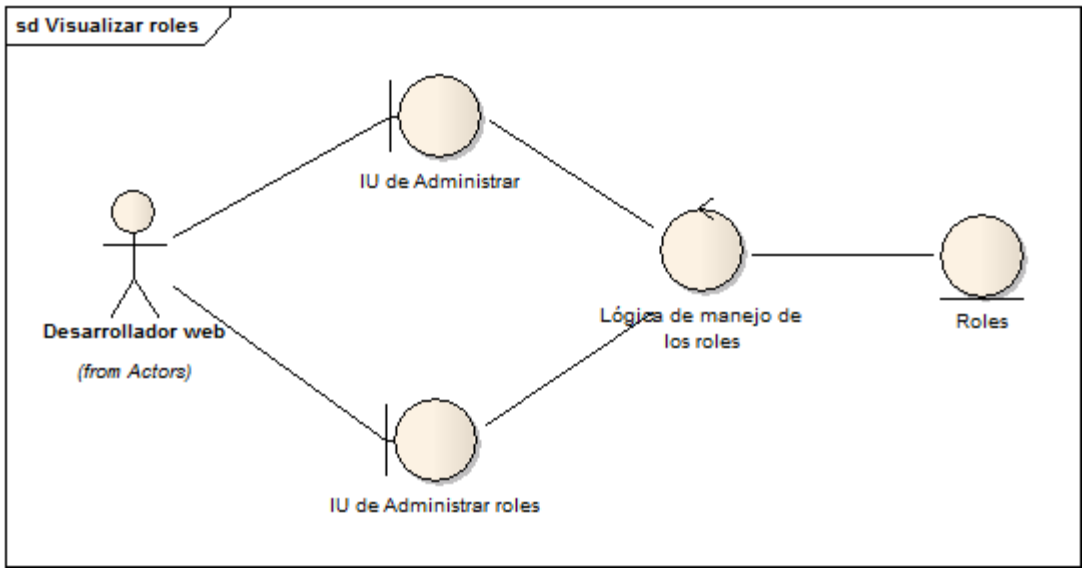

Figura 7 Modelo de análisis para el caso de uso visualizar roles

Descripción del modelo de análisis:

1. Presentación de la interfaz principal de la aplicación desde donde se selecciona la opción “Administrar roles”.
2. La lógica de la aplicación se encarga de obtener desde la base de datos, la información de los roles necesaria para la visualización.
3. Visualización de la interfaz “Administrar roles” donde se visualiza la información de los roles.

4.4.7 Caso de uso 7: Administrar roles

- **Definición:** Permite agregar un nuevo rol o eliminar uno de los roles existentes.

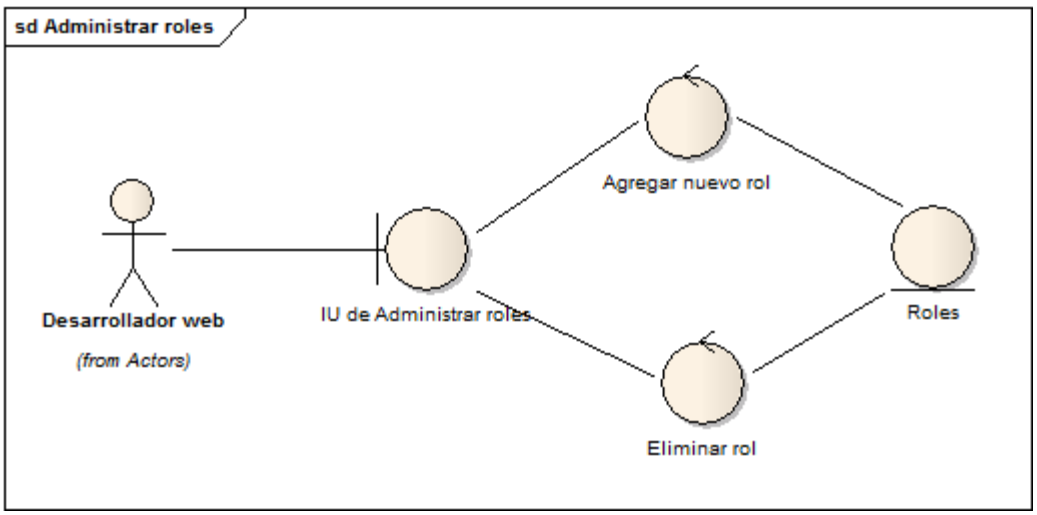

Figura 8 Modelo de análisis para el caso de uso administrar roles

Descripción del modelo de análisis:

1. Presentación de la interfaz “Administrar roles” desde la cual el usuario puede realizar las opciones antes mencionadas.
2. Ejecución de la opción seleccionada que puede ser:
    a. Agregar nuevo rol; en este caso se agrega a la base de datos el nuevo rol.
    b. Eliminar rol; el rol será eliminado de la base de datos.
3. Actualizar la información de manera persistente en la base de datos.

4.4.8 Caso de uso 8: Agregar reglas de acceso

- **Definición:** Permite agregar una regla de acceso a algún directorio del sitio web para indicar quiénes tienen permitido el acceso a dicho directorio y quiénes no.

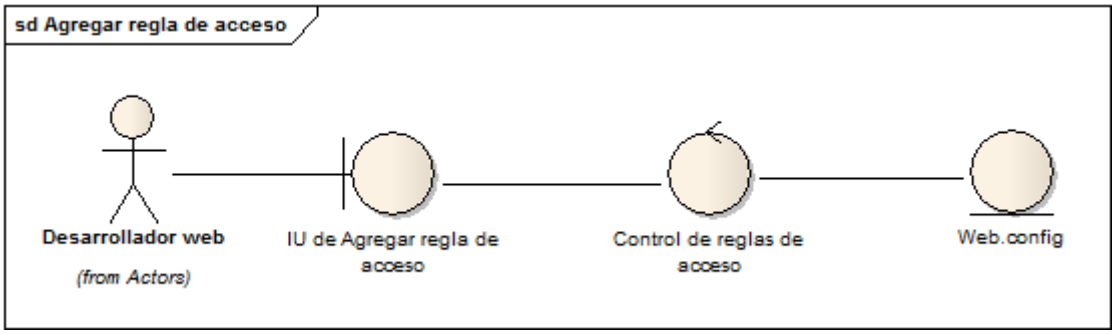

Figura 9 Modelo de análisis para el caso de uso agregar regla de acceso

Descripción del modelo de análisis:

1. Presentación de la interfaz que permite agregar una nueva regla de acceso.
2. El control de reglas de acceso se encarga de actualizar las reglas de la carpeta para la cual se está agregando la nueva regla de acceso. Si en la carpeta actual no existe el web.config, este se crea al agregar la primera regla.

4.4.9 Caso de uso 9: Visualizar reglas de acceso

- **Definición:** Visualiza las reglas de acceso para una carpeta determinada.

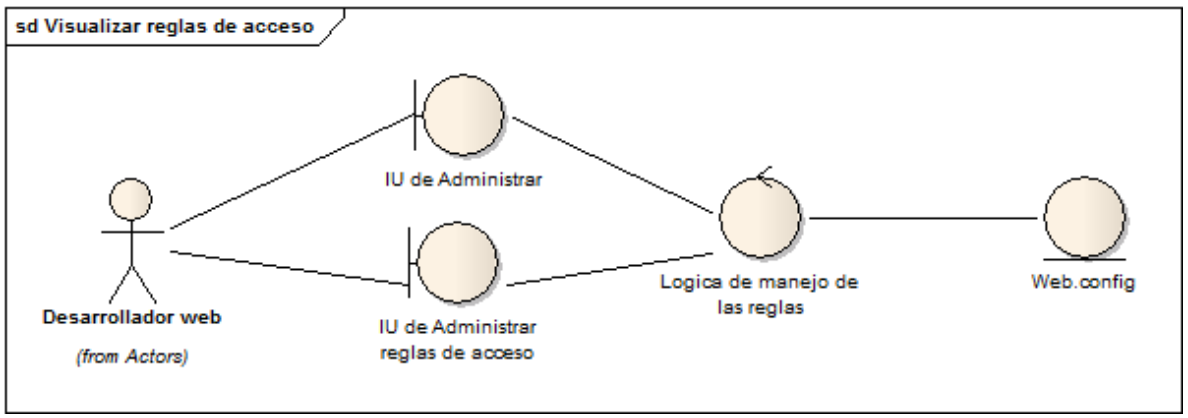

Figura 10 Modelo de análisis para el caso de uso visualizar reglas de acceso

Descripción del modelo de análisis:

1. Presentación de la interfaz principal de la aplicación desde donde se selecciona la opción “Administrar reglas de acceso”.
2. La lógica de manejo de reglas se encarga de obtener las reglas de acceso aplicadas a una carpeta determinada.
3. Visualización de la interfaz “Administrar reglas de acceso” en la cual se listarán todas las reglas aplicadas al directorio seleccionado.

4.4.10 Caso de uso 10: Administrar reglas de acceso

- **Definición:** Permite realizar las operaciones de reordenamiento y borrado de las reglas de acceso para una carpeta determinada.

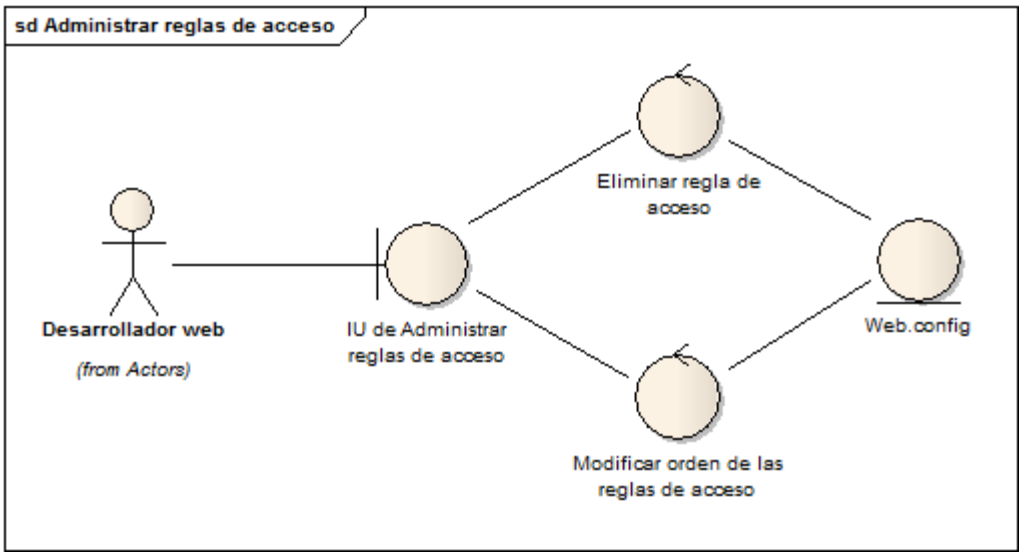

Figura 11 Modelo de análisis para el caso de uso administrar reglas de acceso

Descripción del modelo de análisis:

1. Presentación de la interfaz “Administrar reglas de acceso” desde la cual el usuario puede realizar las opciones antes mencionadas.
2. Ejecución de la opción seleccionada que puede ser:
    a. Eliminar regla de acceso: la regla seleccionada será eliminada de la lista de reglas a aplicarse al intentar acceder a la carpeta seleccionada.
    b. Modificar el orden de las reglas de acceso: permite cambiar el orden en el que serán aplicadas las reglas de acceso a la carpeta.

V. Diseño del sistema

En esta sección se explican los aspectos más relevantes dentro de la implementación del addin para MonoDevelop.

5.1 Diseño del addin

Se decidió que el addin desarrollado fuera mostrado al usuario accediendo al menú Project del MonoDevelop. Una vez instalado el addin, este agregará un submenú llamado **Subscription Settings** el cual contendrá las opciones **"Database Settings"** y **"Subscription Services"**.

5.1.1 Opción Database Settings

Esta opción realiza todo lo referente a la configuración del servicio de suscripciones para la aplicación que se esté desarrollando. Este proceso incluye varias etapas las cuales se inician al seleccionar esta opción desde MonoDevelop. Las etapas son:

a. Introducción por parte del usuario de la información de la base de datos PostgreSQL que se empleará para almacenar la información de los usuarios. Esta información es: host, puerto, usuario, clave y base de datos a usar.

b. Validación de la información introducida por el usuario.

c. Agregar la referencia al proyecto que incluye el proveedor del servicio de suscripciones para PostgreSQL.

d. Creación de las tablas para el almacenamiento de la información de los usuarios y de los roles en la base de datos indicada por el usuario.

e. Creación, si no existe, del archivo web.config.

f. Realización de las siguientes modificaciones en el archivo web.config:

- Agregar una nueva cadena de conexión llamada **postgresqlConnString** con la información de la base de datos introducida por el usuario.
- Agregar o modificar la sección authentication indicando que el tipo de autenticación para la aplicación será “Form”.
- Agregar o modificar la sección roleManeger configurándola con unos valores por defecto e indicando que el proveedor será **PostgreSqlRoleProvider**, el cual se encuentra dentro de la referencia que hemos agregado antes, y que la cadena de conexión a emplear es **postgresqlConnString** todos, la cual fue creada en el primer paso de las modificaciones realizadas al web.config.
- Agregar o modificar la sección membership configurándola con unos valores por defecto e indicando que el proveedor será **PostgreSqlMembershipProvider**, el cual se encuentra dentro de la referencia que hemos agregado antes, y que la cadena de conexión a emplear es **postgresqlConnString** todos, la cual fue creada en el primer paso de las modificaciones realizadas al web.config.

Una vez finalizados todos estos pasos, la aplicación queda lista para emplear el servicio de suscripciones empleando como proveedor PostgreSQL.

A continuación se muestra el diagrama de flujo de la opción Database Settings del addin:

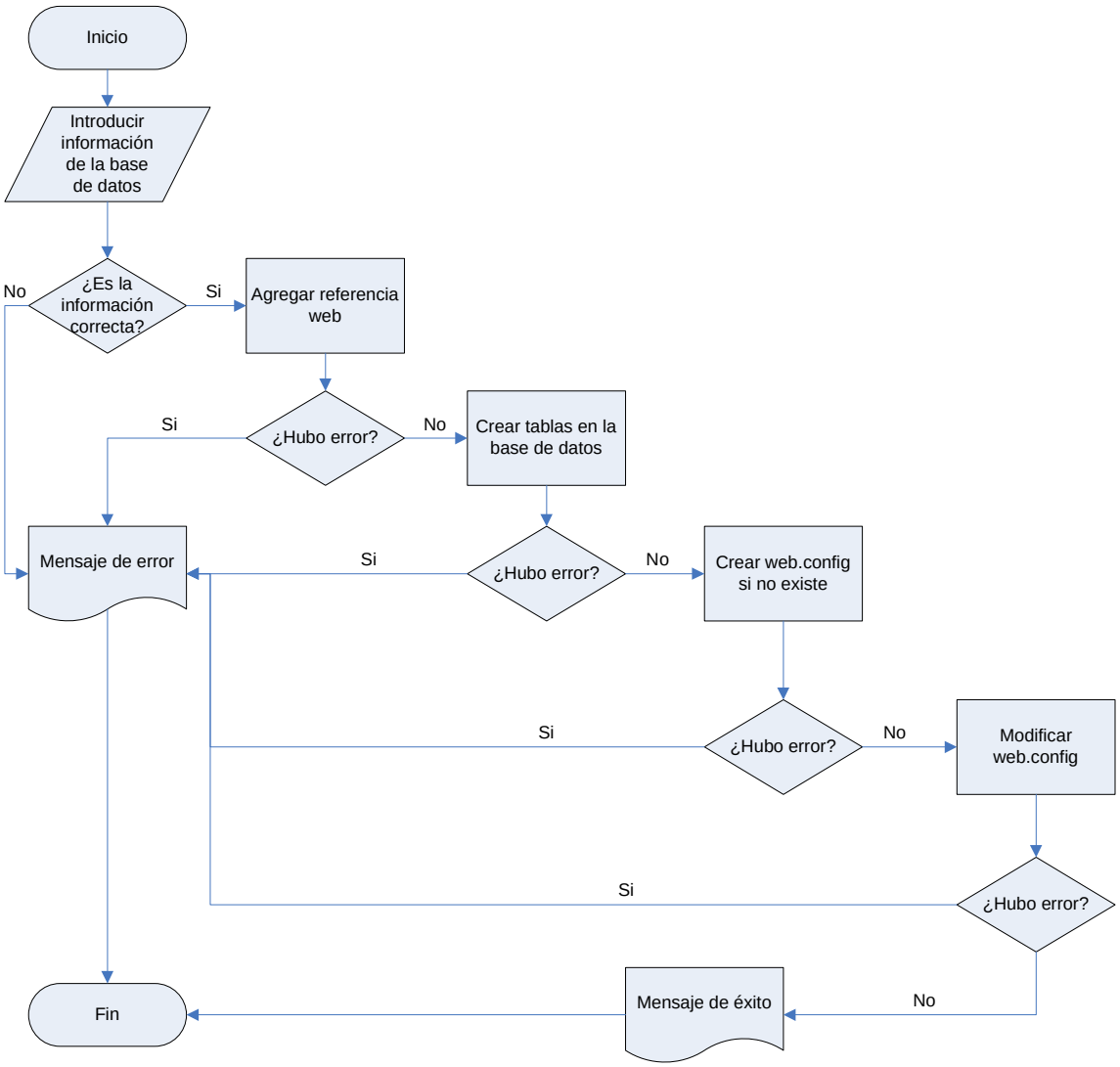

Figura 12 Diagrama de flujo de la opción Database Settings del addin

5.1.2 Opción Subscription Settings

Aunque con la primera opción del addin queda configurado el servicio de suscripciones empleando PostgreSQL como gestor de base de datos, esta segunda opción permite a los desarrolladores web, interactuar con este servicio a través de un sitio web que permite administrar los usuarios, roles y reglas de acceso que se manejan en la aplicación.

Este sitio web ha sido desarrollado con una interfaz similar a la empleada en Visual Studio para que el usuario se sienta familiarizado con la aplicación. El sitio es ejecutado empleando el servidor de aplicaciones XSP. Esta opción incluye las siguientes etapas:

a. Se verifica que la instancia actual de MonoDevelop, no posea un servidor XSP corriendo para el servicio de suscripciones. De poseerlo, se detiene la instancia.
b. Se selecciona un puerto libre para lanzar la instancia del servidor. Esto permite que distintas instancias de MonoDevelop puedan ejecutar el addin sin ningún problema.
c. Se lanza el servidor XSP en el puerto indicado.
d. Se lanza el navegador web con el sitio para la administración del servicio de suscripciones indicándole a través de la URL el sitio web que se desea administrar.

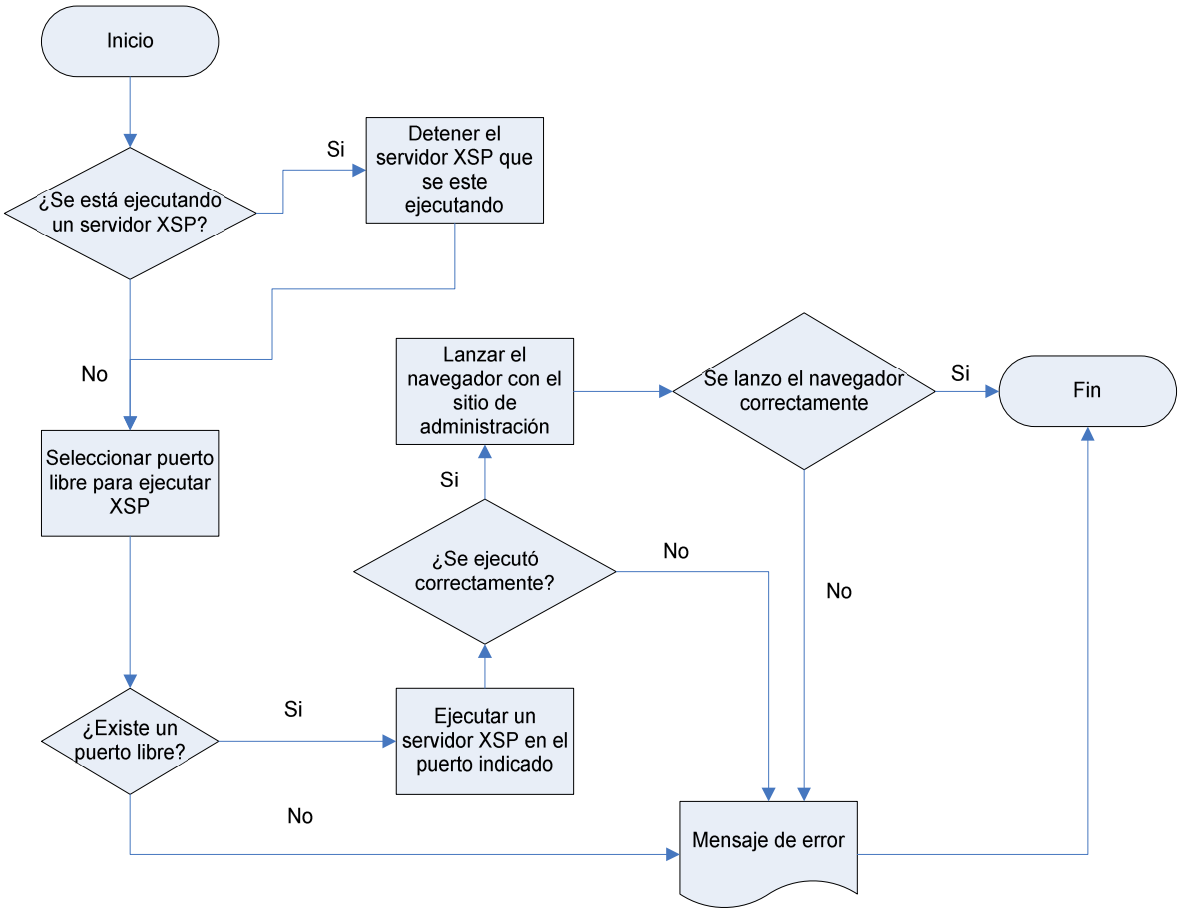

Figura 13 Diagrama de flujo de la opción Subscription Settings del addin

5.2 Agregar la opción del addins al MonoDevelop 2.8

La arquitectura addin de MonoDevelop está diseñada para que pueda ampliar cualquier parte de MonoDevelop por ejemplo el apoyo a un nuevo lenguaje o sistema de control de versiones, o comandos personalizados de edición de código fuente. En esta sección se explica el mecanismo llevado a cabo para agregar las opciones del addin desarrollado al menú Project del MonoDevelop.

MonoDevelop se construye alrededor del concepto de un árbol de extensión. Un addin es un conjunto de extensiones que se conectan a puntos de extensión definidos en otro addin, y también pueden definir nuevos puntos de extensión para otros addin que se deseen extender. MonoDevelop se construye de esta manera, así que hay muchos puntos de extensión disponibles, y los principales se describen en las referencias de MonoDevelop el cual emplea Mono.Addins como motor addins.

5.2.1 Definición del Addin

Un addin requiere un manifiesto que describe el addins y define las extensiones y puntos de extensión de éste. El manifiesto es un archivo XML con la extensión addin.xml. En nuestro caso se definió el archivo SubsciptionService.addin.xml**.**

```
<Addin namespace   = "MonoDevelop"
      id            = "SubscriptionService"
      name          = "Subscription Service"
      author        = "Denis Espinoza"
      copyright     = "MIT/X11"
      url           = "http://www.monodevelop.com"
      description = "Subscription service for ASP. NET using PostgreSQL"
      category      = "Source Editor Extensions"
      version       = "1.0">
</Addin>
```

La combinación del espacio de nombres e id debe ser única entre todos los addins de MonoDevelop. Los otros atributos son fáciles de entender, y muchos de ellos son

opcionales, especialmente si el addin no define puntos de extensión. Cualquier addin no trivial incluirá las clases necesarias para implementar la mayoría de los puntos de extensión. El manifiesto del addin puede ser embebido como un recurso incrustado dentro del ensamblado. Si el addin requiere archivos o ensamblados adicionales, éstos deben estar referenciadas en la sección Runtime, de modo que el motor addin puede asegurar que existen y se puede empaquetar todos los archivos del addin en un solo archivo mpack para su distribución. En nuestro caso fue necesaria hacer estas referencias para lograr que todos los componentes del addin estuviesen dentro del mismo empaquetado.

```
<Addin ..>
<Runtime>
                <Import assembly="Andri.Web.PostgreSQLProvider.dll" />
                <Import assembly="DatabaseConfiguration.dll" />
                <Import assembly="Npgsql.dll" />
                <Import file="subscription.sql" />
                <Import file="web.config.template" />
…..
</Runtime>
</Addin>
```

A continuación se deben declarar las dependencias que en nuestro caso fue el addin MonoDevelop.Ide pues este addin contiene muchos de los puntos de extensión y la API para el entorno de desarrollo en general. Las versiones de las dependencias deben coincidir con la versión de MonoDevelop.

```
<Addin ...>
  <Dependencies>
     <Addin id=" Ide" version="2.8"/>
  </Dependencies>
</Addin>
```

5.2.2 Agregando extensiones

Ahora que el addin está definido, podemos añadir las extensiones. Cada punto de extensión tiene un único camino, por lo que la extensión está contenida en un elemento de tipo Extensión y su ruta en un atributo llamado path. Este elemento puede contener varios nodos de extensión para conectar en el punto de extensión. El nombre del nodo y los atributos están definidos por el punto de extensión. Algunos nodos de extensión pueden tener nodos secundarios, que luego se convierten en puntos de extensión. Para agregar el addins al MonoDevelop, se crearon y agregaron los siguientes puntos de extensión:

```
<!-- Define la primera opción del addin la cual posteriormente será agregada al submenú Subscription Service que se creará dentro del menú Project del menú principal. Esta opción es Database Settings -->

<Extension path = "/MonoDevelop/Ide/Commands/Edit">
      <Command id = "SubscriptionService.SubscriptionServiceCommands.Connection"
            _label = "Database Settings"
            _description = "Configure the connection to the PostgreSQL database"
            defaultHandler = "SubscriptionService.ConnectionHandler"/>
</Extension>

<!-- Define la segunda opción del addin la cual posteriormente será agregada al submenú Subscription Service que se creará dentro del menú Project del menú principal. Esta opción es Subscription Settings -->

<Extension path = "/MonoDevelop/Ide/Commands/Edit">
      <Command id = "SubscriptionService.SubscriptionServiceCommands.Configure"
            _label = "Subscription Settings"
            _description = "Configure the subscription service for ASP. NET using PostgreSQL"
            defaultHandler = "SubscriptionService.ConfigureHandler"/>
</Extension>

<!-- Agrega al menú Project del menú principal el submenú Subscription Service que es de tipo
```

```
ListOptions lo que permite que sea un submenú y no una opción sencilla -->

<Extension path = "/MonoDevelop/Ide/MainMenu/Project">
      <ItemSet id="ListOptions"
          _label="Subscription Service"/>
</Extension>

<!—Las siguientes dos extensiones agregan dentro del submenú Subscription Service creado dentro de la extensión anterior las opciones del addin creadas al inicio -->
<Extension path = "/MonoDevelop/Ide/MainMenu/Project/ListOptions">
      <CommandItem id="SubscriptionService.SubscriptionServiceCommands.Connection" />
</Extension>

<Extension path = "/MonoDevelop/Ide/MainMenu/Project/ListOptions">
      <CommandItem id="SubscriptionService.SubscriptionServiceCommands.Configure" />
</Extension>
```

Esté archivo nos permite tener dentro del entorno de MonoDevelop las nuevas opciones sin embargo aún no poseen ninguna funcionalidad. El agregar la funcionalidad a estas opciones se aborda en el apartado siguiente.

5.3 Diseño de clases

Para una mejor organización del proyecto y lograr las funcionalidades que deben cumplir cada una de las opciones del addin se han agrupado las clases necesarias en cinco paquetes.

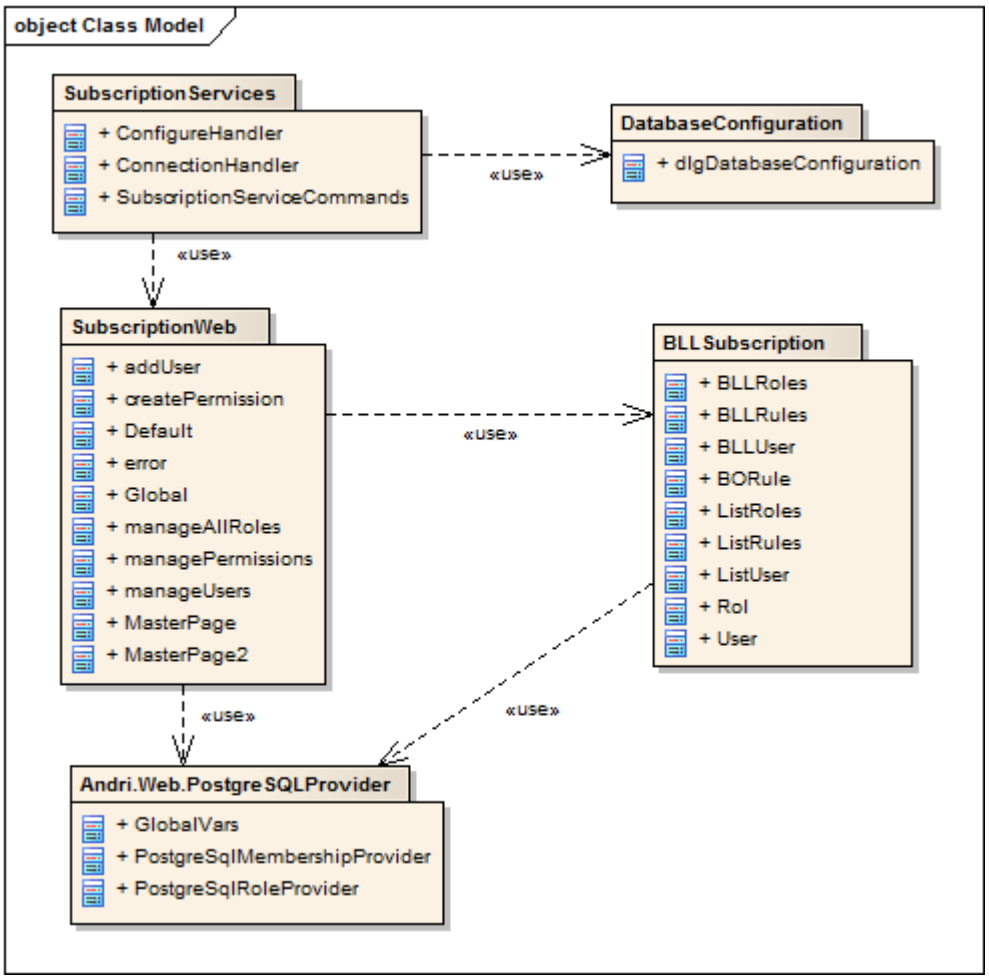

Figura 14 Relación entre paquetes de la aplicación con las clases que contienen

5.3.1 Paquete SubscriptionServices

Contiene las clases encargadas de la implementación del addin.

Clases incluidas:

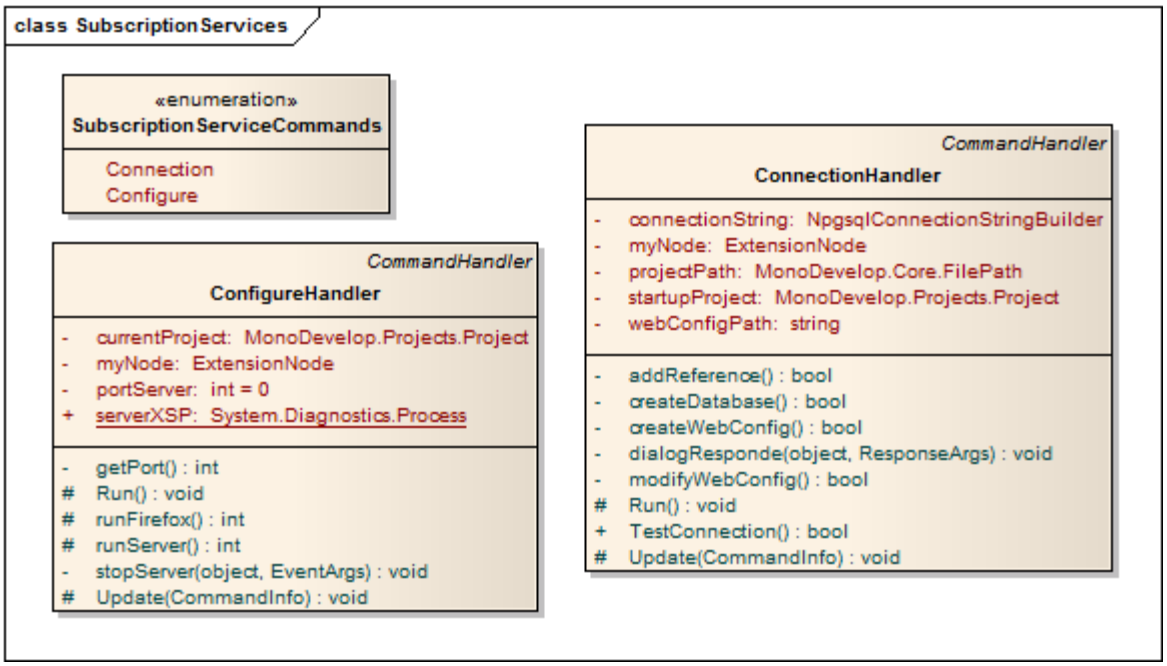

Figura 15 Detalles de las clases del paquete SubscriptionServices

- **SubscriptionServiceCommands:** Este es un enumerado que contiene las dos operaciones que el addin puede realizar. Estas opciones son “Connection” y “Configure”. De aquí toman sus nombres las otras dos clases que componen el paquete.
- **ConnectionHandler:** Esta clase encapsula la funcionalidad para solicitar la información de la base de datos a emplear para el almacenamiento de los usuarios y roles haciendo uso del paquete **DatabaseConfiguration**. Además configura el web.config de la aplicación con las opciones necesarias para el funcionamiento del servicio de suscripciones y agrega como referencia al proyecto la biblioteca generada por el paquete **Andri.Web.PostgreSQLProvider** (caso de uso 1).
- **ConfigureHandler:** Esta clase encapsula la funcionalidad que permite lanzar la aplicación web del paquete **SubscriptionWeb** desde la cual se realiza la administración de usuarios, roles y reglas (casos de uso del 2 al 10).

5.3.2 Paquete DatabaseConfiguration

Contiene una única clase llamada **dlgDatabaseConfiguration** que es la encargada de mostrar una interfaz desarrollada en GTK 2.0 empleada para solicitar la información de la base de datos a emplear. La razón por la cual se empleo GTK en lugar de Windows Form fue mantener la uniformidad en el estilo del entorno pues MonoDevelop 2.8 desarrollado empleando GTK 2.0.

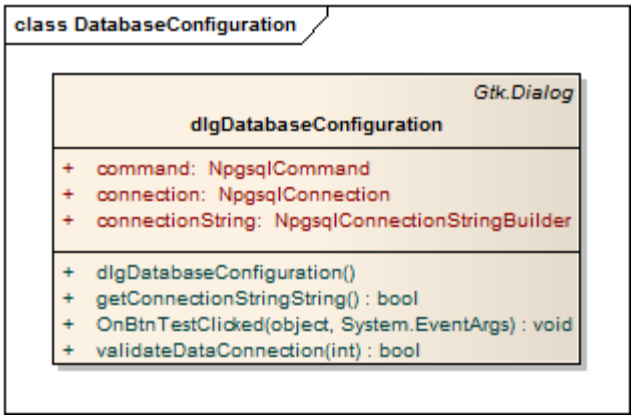

Figura 16 Detalle de las clases del paquete DatabaseConfiguration

5.3.3 Paquete Andri.Web.PostgreSQLProvider

Las clases contenidas en este paquete no han sido desarrolladas por el autor de este proyecto. Estas clases fueron hechas por un desarrollador que se identifica como Andri.Web (citado en los antecedentes de este documento). Este paquete contiene dos clases (PostgreSqlMembershipProvider y PostgreSqlRoleProvider) que implementan el proveedor del servicio de suscripciones para PostgreSQL. Estas clases fueron empleadas para realizar las primeras pruebas de este trabajo.

Como una forma de respeto a los derechos de autor, se ha mantenido el mismo espacio de nombres (Andri.Web) que coloco su desarrollador. El autor de este trabajo realizó algunas modificaciones sobre estas clases, necesarias para su correcto funcionamiento debido a que en el momento de ser obtenidas, presentaba algunos errores. Aparte de esto, se añadió la clase GlobalVars y se empaquetaron todas las clases en una dll para facilitar su utilización en los otros paquetes.

Clases incluidas:

Figura 17 Detalles de las clases del paquete Andri.Web.PostgreSQLProvider

- **GlobalVars:** Esta clase almacena dos variables estáticas, necesarias para la correcta inicialización de las dos clases siguientes cuando estas son invocadas desde el paquete SubscriptionWeb.
- **PostgreSqlMembershipProvider:** Esta clase se deriva de MembershipProvider y se encarga de implementar la lógica del servicio de subscripciones necesaria para que la gestión de los usuarios se lleve a cabo con una base de datos PostgreSQL.
- **PostgreSqlRoleProvider**: Esta clase se deriva de RoleProvider y se encarga de implementar la lógica del servicio de subscripciones necesaria para que la gestión de los roles se lleve a cabo con una base de datos PostgreSQL.

5.3.4 Paquete BLLSubscription

Contiene las clases de lógica de negocio empleadas para el manejo de usuarios, roles y reglas desde acceso.

Clases incluidas:

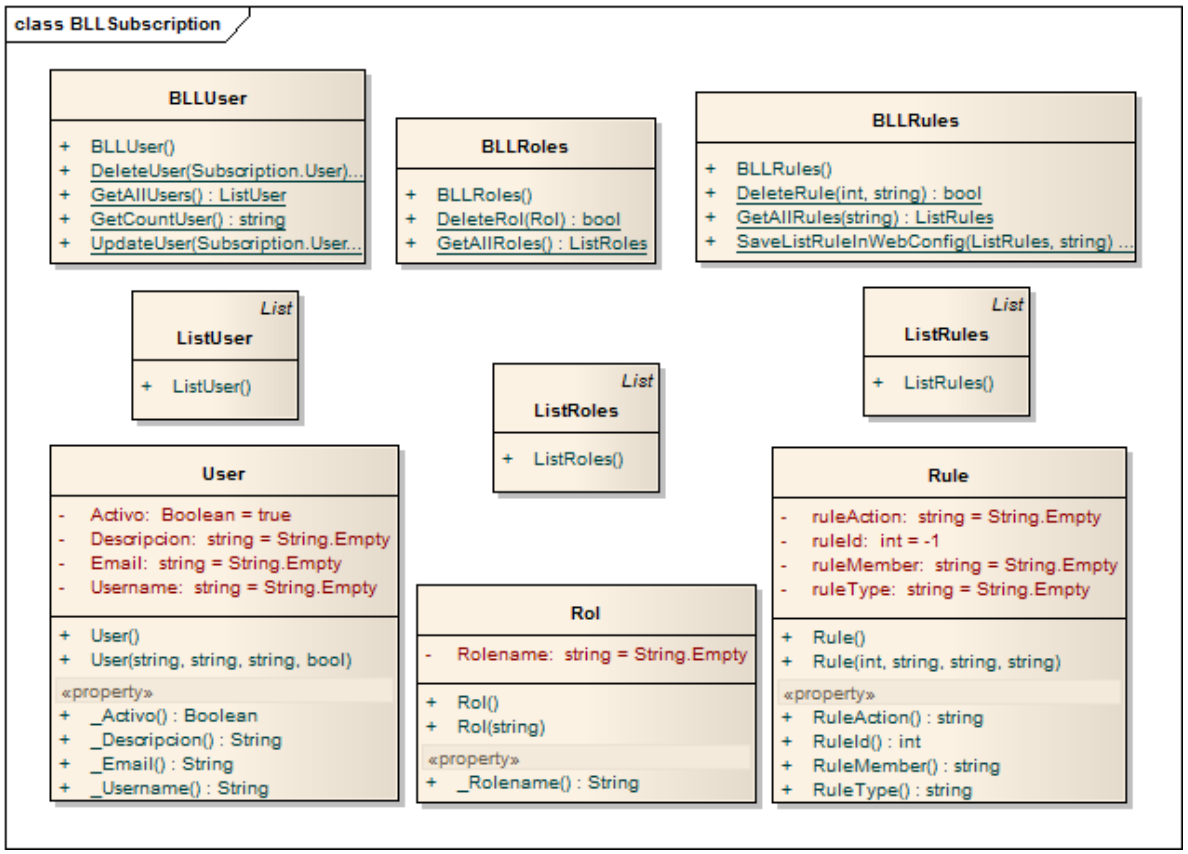

Figura 18 Clases contenidas en el paquete BLLSubscription

- **User:** Objeto de negocio que encapsula parte de la información de un usuario para ser manipulada desde la aplicación web del paquete SubscripionWeb.
- **Rol:** Objeto de negocio que encapsula la información de un rol.
- **Rule:** Objeto de negocio que encapsula la información que compone una regla de acceso.
- **ListUser:** Colección de objetos de tipo User.
- **List:Roles:** Colección de objetos de tipo Rol.
- **ListRules:** Colección de objetos de tipo Rule.
- **BLLUser:** Encapsula las operaciones a realizar sobre los usuarios creados en la aplicación.
- **BLLRoles:** Encapsula las operaciones a realizar sobre los roles creados en la aplicación.
- **BLLRules:** Encapsula las operaciones a realizar con las reglas creadas para administrar el acceso a las carpetas de la aplicación. Esta clase tiene la capacidad de acceder y modificar el fichero web.config de la carpeta sobre la que se están aplicando las reglas.

5.3.5 Paquete SubscriptionWeb

Este paquete encapsula un sitio web que ha sido desarrollado para administrar la aplicación que se esté desarrollando. Este sitio ha sido creado teniendo como modelo el empleado en Visual Studio con el objetivo de que el usuario se sienta familiarizado con su uso.

5.3.5.1 Estructura del sitio web

Página Default.aspx: Es la página principal del sitio, desde aquí se accede a las diferentes opciones para la administración de la aplicación. Para una mejor organización, la página ofrece tres secciones que agrupan las distintas operaciones que es posible realizar:

1. Sección **Users:** Maneja lo referente a los usuarios. Posee los enlaces **Add user** que visualiza la página addUser.aspx y **Managing users** que visualiza la página manageUsers.aspx.

2. Sección **Roles:** Maneja lo referente a los roles. Posee un enlace que permite habilitar o deshabilitar los roles (caso de uso 5) y otro enlace llamado **Create and manage roles** que visualiza la página manageAllRoles.aspx.
3. Sección **Access Rules:** Maneja lo referente a las reglas de acceso. Posee los enlaces **Create access rules** que visualiza la página createPermission.aspx y el **Manage Access rules** que visualiza la página managePermissions.aspx.

Página addUser.aspx: Esta página permite agregar un nuevo usuario a nuestra aplicación con lo que queda implementado el caso de uso 2.

Página manageUsers.aspx: Esta página muestra la lista de los usuarios almacenados ofreciendo la posibilidad de editar su descripción y estado. También permite borrar un usuario o asignarle los roles que uno deseo. Con esta página quedan implementados los casos de uso 3 y 4.

Página manageAllRoles.aspx: Esta página lista los roles existentes y permite agregar uno nuevo o borrar alguno. Con esta página quedan implementados los casos de uso 6 y 7.

Página createPermission.aspx: Está página permite crear reglas de acceso para cualquier carpeta dentro de la aplicación web que estemos desarrollando. Con esta página queda implementado el caso de uso 8.

Página managePermissions.aspx: Permite listar, borrar y cambiar el orden de los permisos aplicados a una carpeta de la aplicación web que estemos desarrollando. Desde esta página se tiene acceso a la página createPermission.aspx para poder agregar un permiso nuevo a la carpeta que estemos actualmente administrando. Con esta página quedan implementados los casos de uso 9 y 10.

5.3.5.2 Archivos auxiliares

Son páginas o archivos que no implementan ninguna de la lógica principal de la aplicación pero que dan soporte a su correcto funcionamiento y visualización.

Archivo MasterPage2.master: Página maestra para las páginas Default y error.

Archivo MasterPage.master: Página maestra para las demás páginas del sitio.

Página Global.asax: Contiene las configuraciones globales para todo el sitio. Es donde se especifica que para cualquier error que ocurra, se visualice la página error.aspx.

Página error.aspx: Es la página que se mostrará cada vez que se presente un error dentro de la aplicación.

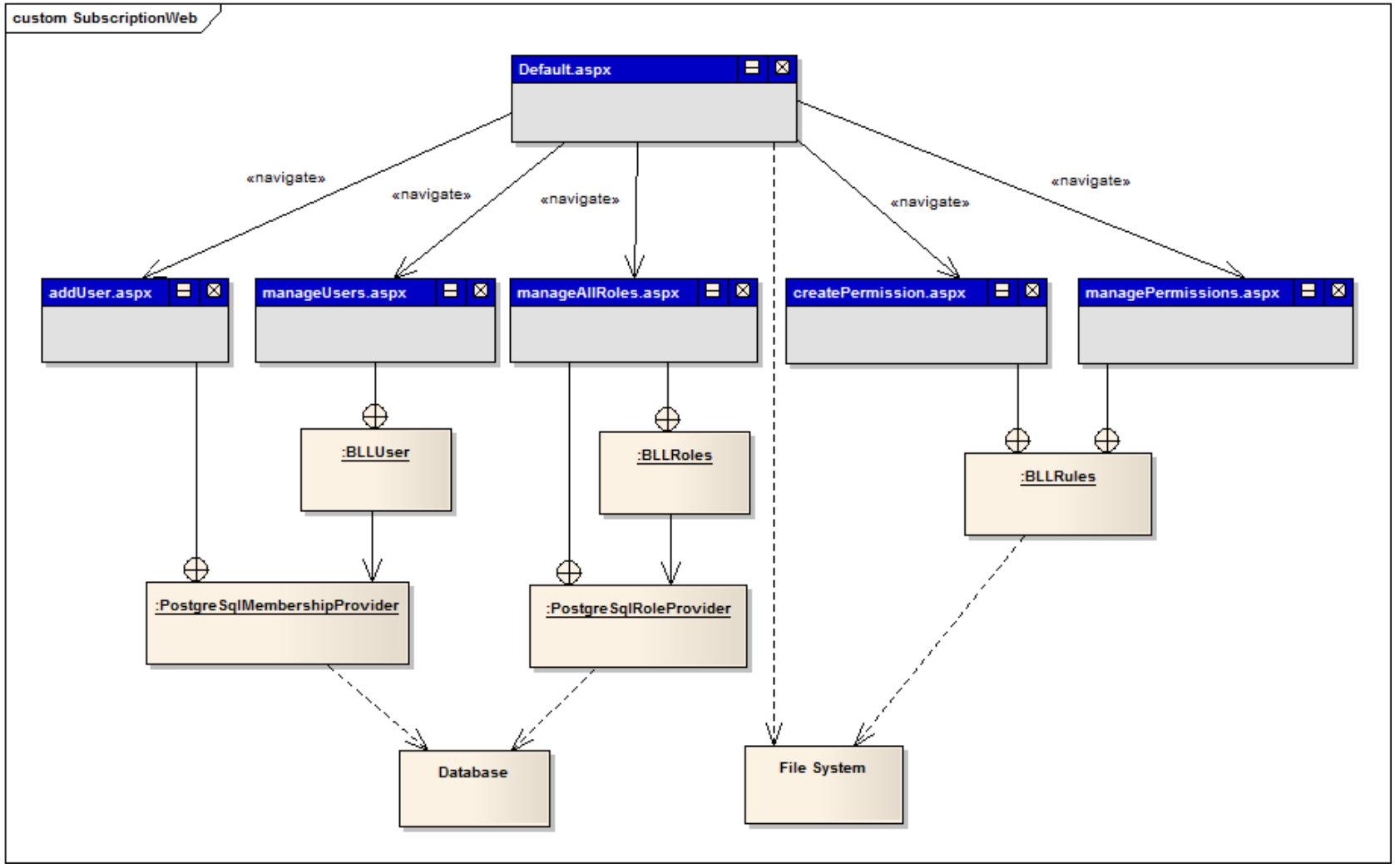

Figura 19 Estructura de la aplicación web contenida para la administración del sitio web

5.4 Diseño de la base de datos

Como se ha mencionado anteriormente, la aplicación emplea una base de datos en PostgreSQL sobre la cual se crean las tablas necesarias para el almacenamiento de la información de los usuarios y de los roles. El esquema de esta base de datos es el siguiente:

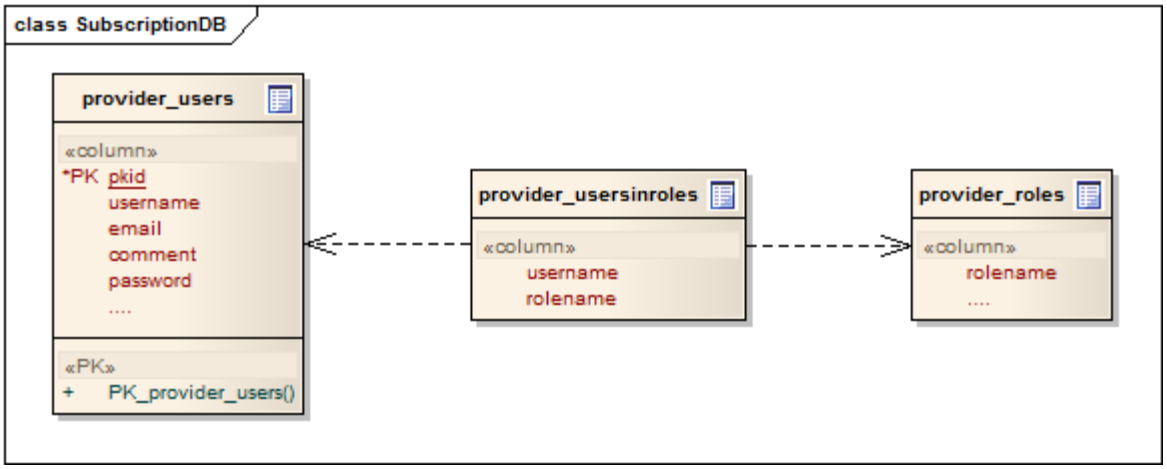

Figura 20 Esquema relacional de la base de datos empleada para proveer el servicio de suscripciones

Nos se han colocado todos los campos que poseen las tablas sino los más relevantes y empleados dentro del desarrollo de la aplicación. Las tablas son:

- **provider_users:** Está tabla almacena la información de todos los usuarios que son agregados a nuestra aplicación.
- **provider_roles:** Esta tabla almacena todos los roles que han sido creados en nuestra aplicación.
- **provider_usersinroles:** Establece relaciones entre las entradas de la tabla provider_users y entradas de la tabla provider_roles, para indicar qué usuarios se encuentran en qué roles.

Es importante destacar que la base de datos no lleva ninguna información acerca de las reglas de acceso ya que estas son manejadas a través de los ficheros web.config ubicados en cada carpeta de la aplicación web.

VI. Implementación del addin

El addin se implementa según se muestra en el esquema siguiente:

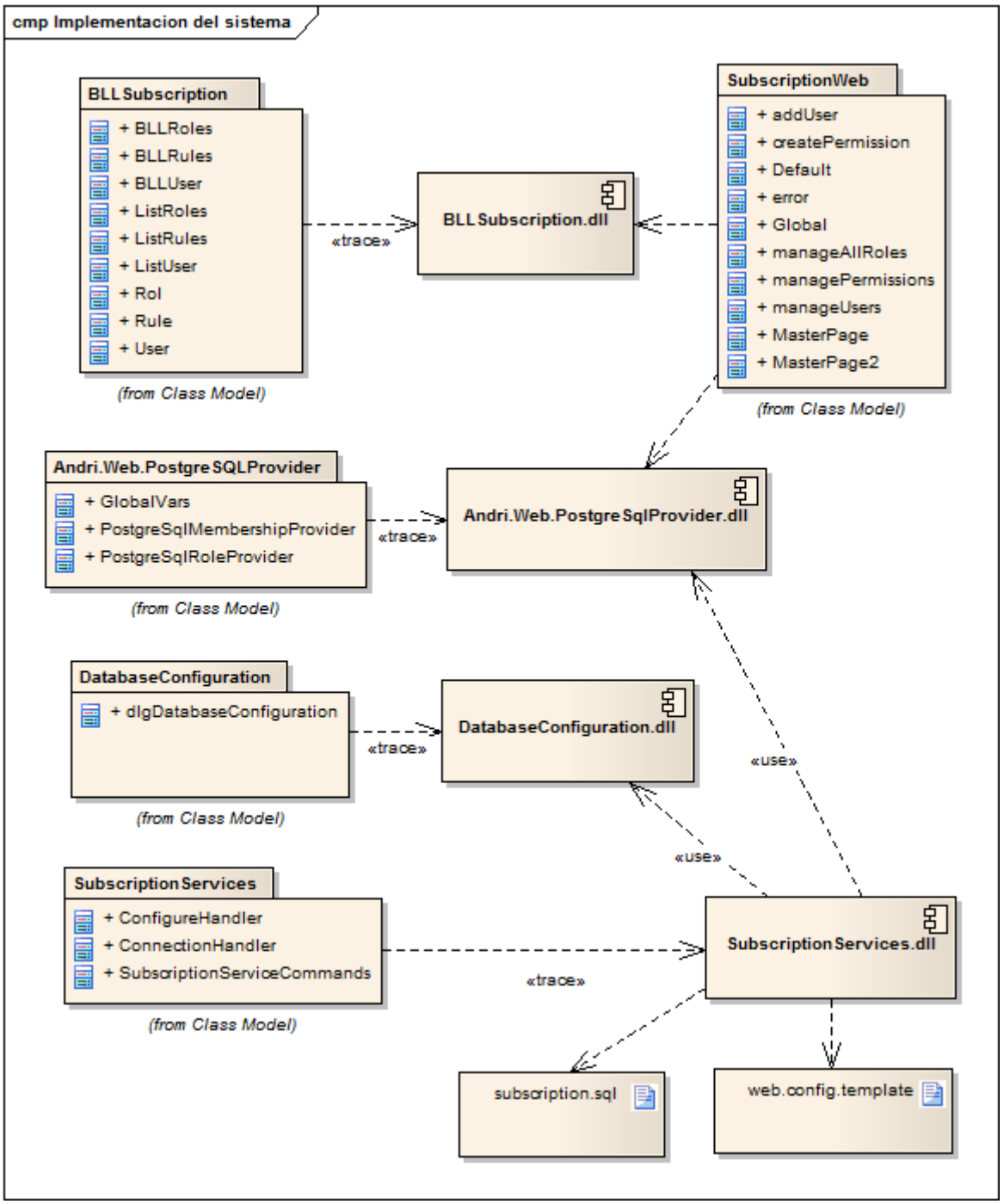

Figura 21 Diagrama de implementación del addin

Los paquetes BLLSubscription, Andri.Web.PostgreSqlProvider, DatabaseConfiguration y SubscriptionServices, generan como salida bibliotecas (archivos dll) que forman parte del addin. Como se observa en el diagrama anterior y en el diagrama de clases, existen dependencias entre estas dll. El paquete SubscriptionWeb no genera ninguna dll porque como se mencionó en el apartado anterior, este es un sitio web y, al igual que los demás, tiene dependencias de algunas de las dll creadas.

A parte de lo generado por cada uno de estos paquetes, la implementación emplea dos archivos:

- suscripción.sql, que guarda el script a partir del cual se generan las tablas de la base de datos.
- web.config.template, empleado como plantilla para crear el web.config para cualquier aplicación que no posea uno.

Todo los elementos anteriores son empaquetados en un solo archivo llamado **MonoDevelop.SubscriptionService_1.0.mpack** que es el que se emplea para la instalación del addins dentro del MonoDevelop.

VII. Instalación

La instalación de este addin es muy sencilla pero requiere tener instalados los siguientes programas:

- Mono 2.10
- MonoDevelop 2.8
- Mozilla Firefox 2.0+.
- PostgreSQL 8.4

Para instalar este addin debemos tener el empaquetado del addins llamado **MonoDevelop.SubscriptionService_1.0.mpack** el cual contiene el archivo de instalación del addins y todas sus dependencias.

Figura 22 Empaquetado del addins

Posteriormente, accedemos al administrador de add-ins de MonoDevelop el cual se encuentra en el menú **Tool -> Add-in Manager**. Esto cargará la interfaz mostrada en la Figura 23 Administrador de addins en MonoDevelop donde debemos pulsar el botón “Install from file…” que nos permitirá seleccionar el archivo **MonoDevelop.SubscriptionService_1.0.mpack**.

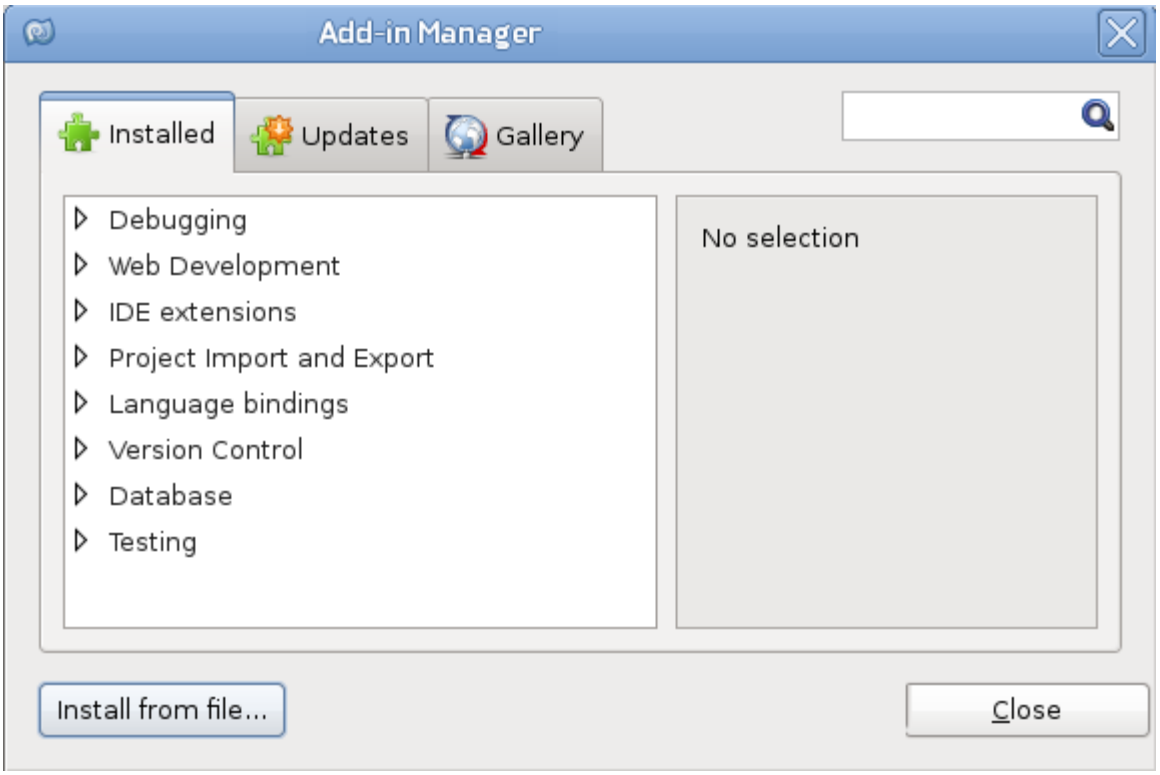

Figura 23 Administrador de addins en MonoDevelop

Una vez seleccionado el archivo, se nos presentará un diálogo que pedirá la confirmación de la instalación y finalizada la misma el nuevo add-in aparecerá en la lista de add-ins instalados tal como se muestra en la Figura 24 Add-in Subscription Service instalado

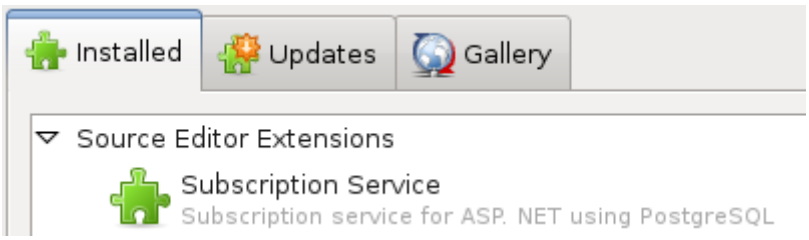

Figura 24 Add-in Subscription Service instalado

Para la desinstalación del add-in se emplea también el administrador de addins que hemos usado para la instalación.

VIII. Conclusiones y futuros trabajos

Una vez finalizada y probada la aplicación se ha llegado a las siguientes conclusiones:

1. El servicio de suscripciones que ofrece ASP.NET es una potente herramienta para la gestión de usuarios dentro de las aplicaciones web ya que facilita todos los aspectos relacionados con el manejo de las cuentas de usuario, roles y reglas de acceso de la aplicación.
2. La posibilidad de crear nuestros propios proveedores para el servicio de suscripciones de ASP.NET permite que esta funcionalidad pueda ser implementada sobre casi cualquier sistema gestor de bases de datos.
3. La tecnología Mono.Addins con la cual ha sido desarrollado el IDE MonoDevelop y la API de este, permiten a los desarrolladores crear extensiones para dotar de mayores y mejores herramientas a dicho entorno.
4. La compatibilidad de la plataforma Mono con la plataforma .NET permitió el desarrollo de una interfaz web, similar a la empleada en Visual Studio para la administración de sitios web.

Líneas de trabajos futuros

Al finalizar este trabajo se proponen las siguientes líneas de trabajo:

1. Mejorar el addin para que este pueda funcionar con más gestores de base de datos y no solo con PostgreSQL. Esto se podría realizarse empleando las clases DbProviderFactories y DbProviderFactory.
2. Agregar a la interfaz web soporte para múltiples idiomas.
3. Agregar al sitio web, una sección que permita confirmar las cuentas de usuario a través de email.
4. Revisar la página de MonoDevelop en la cual se citan una serie de addins que se pueden desarrollar y de esta forma, enriquecer las funcionalidades que ofrece MonoDevelop.

IX. Manual de la aplicación

9.1 Introducción

En este apartado se explica cómo emplear el addin para la implementación del servicio de subscripciones con ASP.NET y PostgresSQL empleando Mono Jit compiler 2.10.2 y MonoDevelop IDE 2.8.5 o superiores.

9.2 Pasos iniciales

Para emplear este addin se debe crear un proyecto de tipo **ASP.NET**. En el ejemplo se ha seleccionado como plantilla **Web Application** y se ha llamado EjemploWeb al proyecto. Sin embargo, tanto el tipo de plantilla como el nombre del proyecto los decide el usuario.

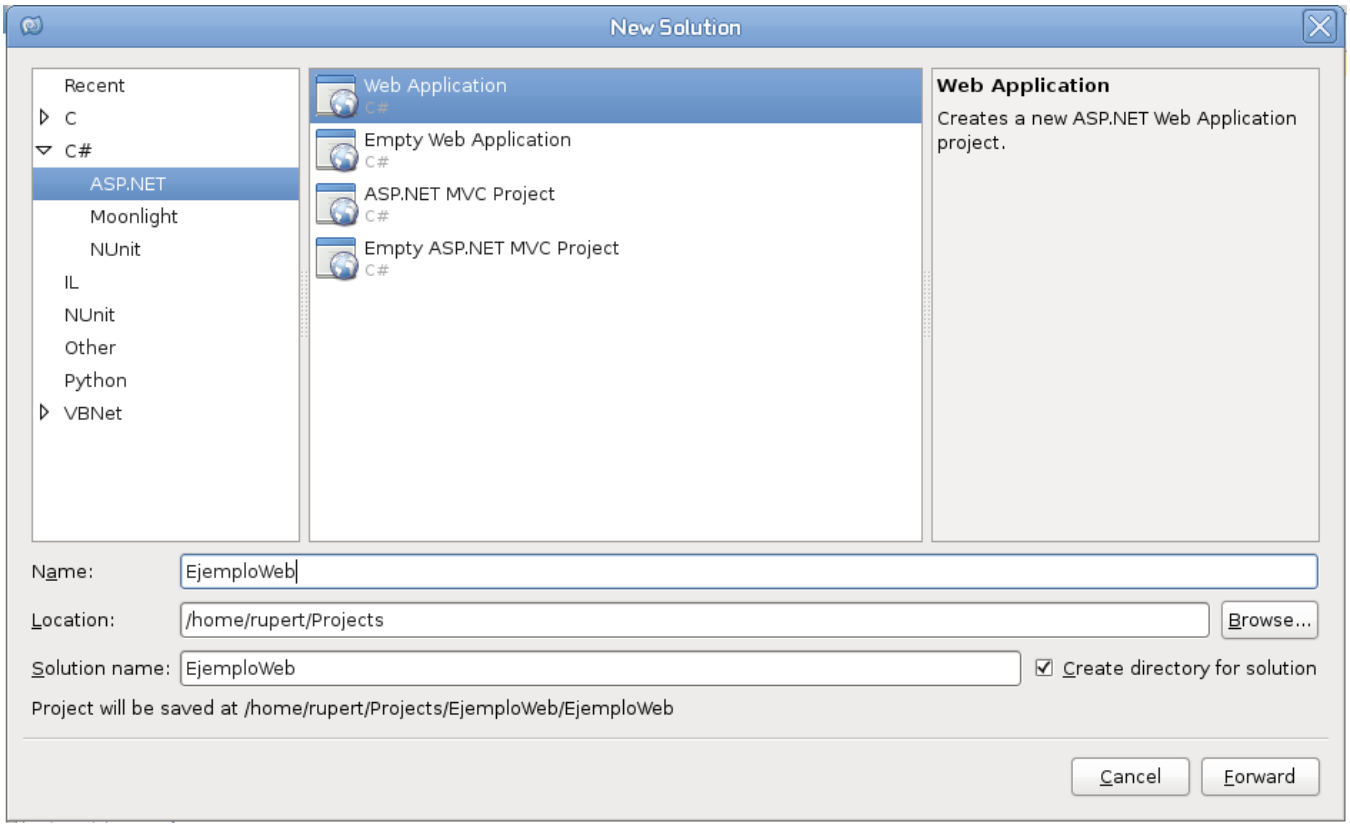

Figura 25 Creación de un proyecto ASP.NET en MonoDevelop

Una vez que el proyecto ha sido creado observamos el addin desarrollado, el cual se ubica dentro del menú **Proiect** de la barra de menús del entorno.

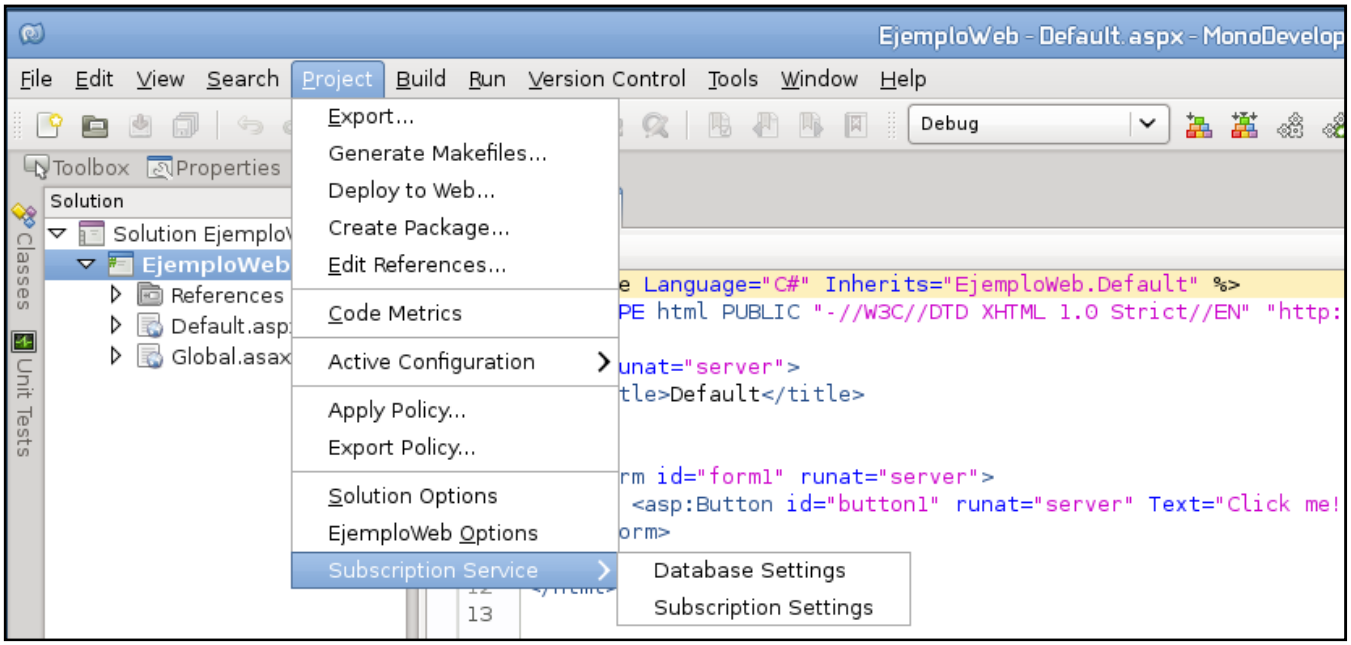

Figura 26 Visualización del addin desde el menú Project de MonoDevelop

El addin aparece con el nombre **Subscription Service** y de él se derivan dos opciones:

- Database Settings
- Subscription Settings

Es importante destacar que nuestro addin, sólo se habilitará cuando el proyecto seleccionado sea de tipo ASP.NET.

9.3 Configuración de PostgreSQL

Como se ha dicho, nuestro addin emplea como gestor de base de datos PostgreSQL, por lo cual, antes de proceder a la explicación de las dos opciones del addin, debemos realizar algunas configuraciones dentro de este gestor. Por defecto PostgreSQL permite conectarse sin la necesidad de contraseña lo que es muy inseguro. Por tal motivo, procederemos a cambiar la contraseña del usuario postgres. Para ello, desde la consola, iniciamos una sesión como el usuario postgres y allí ejecutamos pgsql.

```
> sudo su postgres
> psql
```

Una vez dentro de la consola de PostgreSQL como usuario postgres (que es un usuario administrador de PostgreSQL), cambiamos nuestra contraseña con la siguiente sentencia:

```
postgres=# alter user postgres with password ’mono’;
ALTER ROLE
postgres=# \q
```

Con esto hemos cambiado la clave del usuario postgres y hemos salido de la consola del gestor. Después de ello, debemos modificar el fichero ***/var/lib/pgsql/data/pg_hba.conf*** para dejarlo de la siguiente manera:

```
local   all     all                     password
host    all     all     127.0.0.1/32    password
host    all     all     ::0/128         password
```

Estos cambios obligarán a que los usuarios que deseen utilizar este gestor, deban autenticarse empleando un usuario y una clave. Para que los cambios tengan efecto, se debe de reiniciar el servidor de base de datos.

```
>/etc/init.d/postgresql restart
```

Probamos la conexión. Para ello utilizamos la utilidad pgsql de la siguiente forma:

```
> psql -U postgres -W
Password for user postgres:
psql (8.4.7)
Type "help" for help.

postgres=#
```

9.3.1 Creación del usuario que emplearemos para conectarse a la base de datos

Para crear el usuario que se conectará a la base de datos, desde la consola de Linux ejecutamos el comando **createuser**. Por ejemplo, los pasos para crear al usuario user1 son los siguientes:

```
> createuser -P -U postgres user1
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
Password:
```

9.3.2 Creación de la base de datos a emplear

Para crear la base de datos, abrimos la consola de administración de PostgreSQL y, desde allí, creamos la base de datos a emplear en nuestra aplicación, indicando como propietario al usuario user1 previamente creado:

```
postgres@linux:/home/rupert> psql -U postgres -W
Password for user postgres:
psql (8.4.7)
Type "help" for help.

postgres=# CREATE DATABASE db_user1 OWNER=user1;
CREATE DATABASE
```

Una vez creada la base de datos, cerramos la consola de PostgreSQL y regresamos a MonoDevelop para emplear la información siguiente:

- Base de Datos: db_user1
- Usuario: user1
- Clave: user1

9.4 Opción Database Settings

Desde MonoDevelop, accedemos a la primera de las opciones que nos ofrece el addin **Project -> Subscription Service -> Database Settings**. Esto nos mostrará un formulario que nos solicitará la información de la base de datos a emplear.

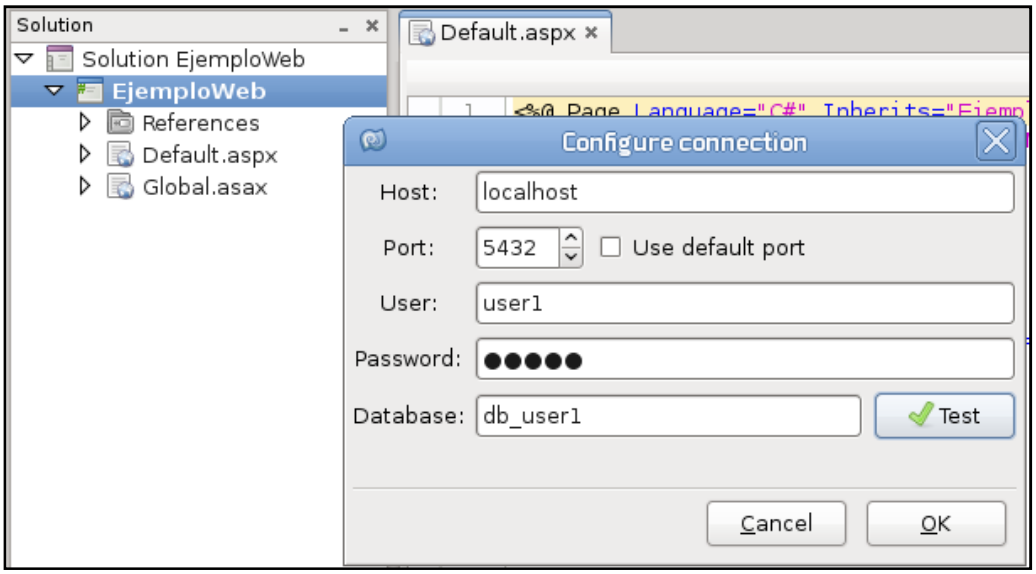

Figura 27 Formulario mostrado por la opción Database Settings del addin

Host y **Port** aparecen con valores por defecto. Los demás, los rellenamos con los valores que tenemos después de crear la base de datos a emplear. Luego, tras pulsar Test para validar los datos, procedemos a pulsar el botón OK para que se realicen todas las modificaciones respectivas. Una vez que el addin ha realizado todas las modificaciones pertinentes en el proyecto, nos enviará el mensaje **Configuration Finished**.

Tras mostrarse el mensaje indicado anteriormente, el entorno habrá realizado las siguientes operaciones en el proyecto:

1. Habrá copiado el archivo Andri.Web.PostgreSQLProvider.dll al directorio bin de la aplicación y habrá agregado la referencia correspondiente al proyecto.
2. Habrá creado las tablas necesarias para el manejo de los usuarios de la base de datos.
3. Habrá creado el archivo **web.config** si no existe.

4. Habrá modificado el archivo **web.config** introduciendo el código necesario para el funcionamiento del servicio de suscripciones.

9.4.1 Archivo web.config modificado

En el archivo web.config de la aplicación quedan registradas las configuraciones necesarias para el manejo de los usuarios y de los roles. Lo más relevante de este archivo es la cadena de conexión llamada **postgresqlConnString** que almacena la información necesaria para realizar la conexión con la base de datos de los usuarios.

```
<configuration>
<connectionStrings>
<addname="postgresqlConnString" connectionString="HOST=localhost;PORT=5432;
USER ID=user1;PASSWORD=user1;DATABASE=db_user1" />
</connectionStrings>
</configuration>
```

9.5 Opción Subscription Settings

Esta opción carga una aplicación web similar a la empleada en Visual Studio que permite gestionar los usuarios, los roles y las reglas de acceso de nuestra aplicación web. Para acceder a esta opción entramos en **Project -> Subscription Service -> Subscription Settings**.

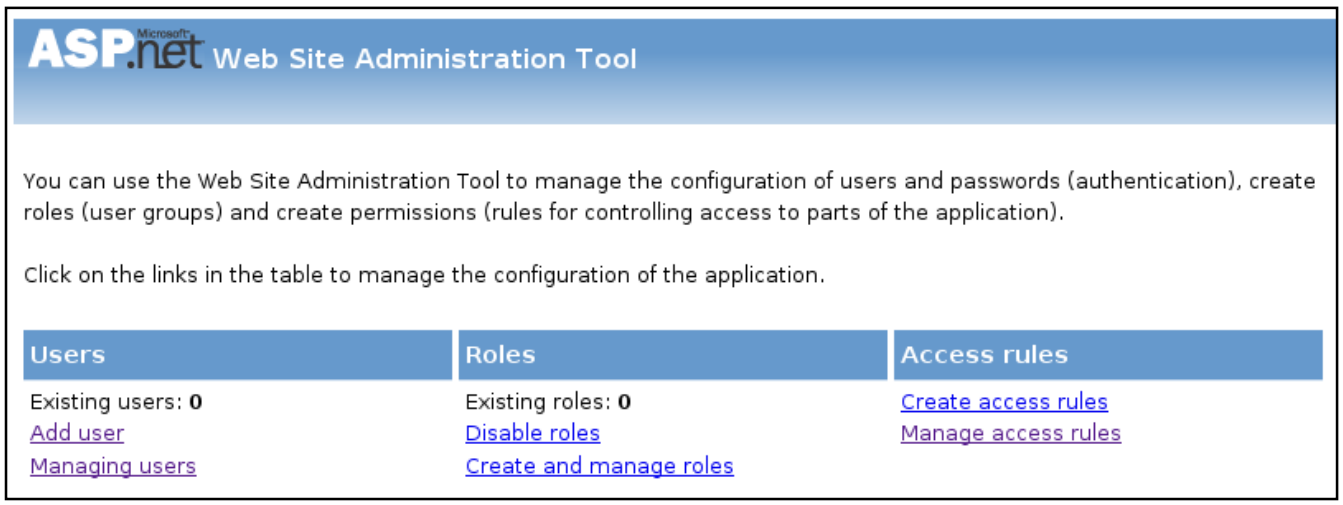

Figura 28 Interfaz principal del sitio web para la administración de nuestra aplicación

La página se divide en 3 secciones que son: Users, Roles y Access rules. Empecemos estudiando la sección Roles.

9.5.1 Roles -> Disable roles

Por defecto los roles aparecen habilitados en nuestra aplicación. Este enlace permite deshabilitar el manejo de los roles, con lo cual no se podrán emplear ni roles ni reglas de acceso.

9.5.2 Roles -> Create and manage roles

Permite crear y eliminar los roles de la aplicación que estemos desarrollando. En la imagen siguiente se muestra que hemos creado un rol llamado **Operador** y estamos creando un rol llamado **Secretaria**.

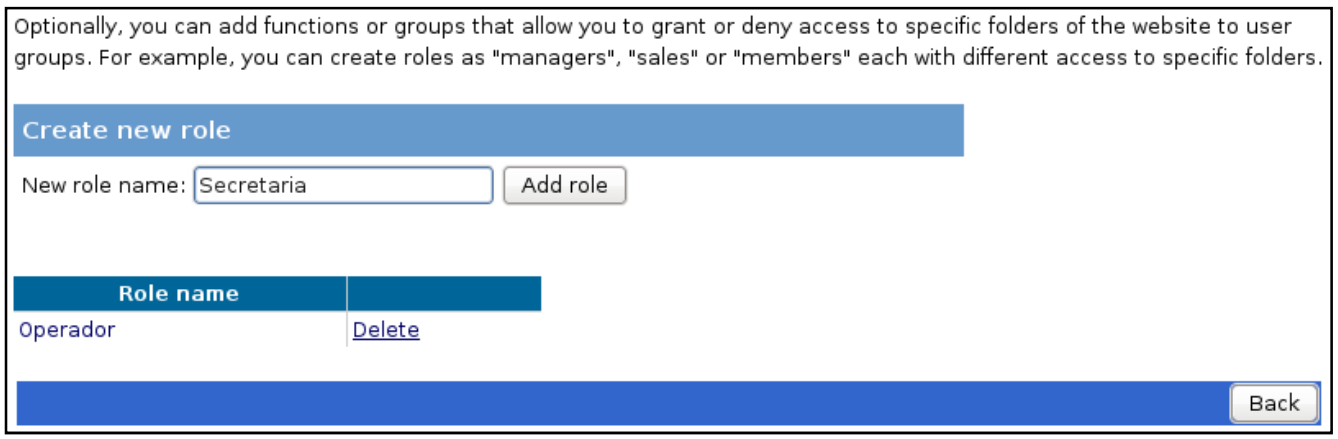

Figura 29 Interfaz para la administración de los roles de la aplicación

Esta interfaz permite también eliminar cualquiera de los roles creados. Al eliminar un rol se desasocian de este a todos los usuarios que pertenecían a él.

9.5.3 Sección User - > Add user

Dentro de la sección User tenemos primero la opción Add user que permite agregar nuevos usuarios a la aplicación. En este ejemplo estamos agregando a un usuario llamado juan y luego añadiremos a un usuario llamado pepe.

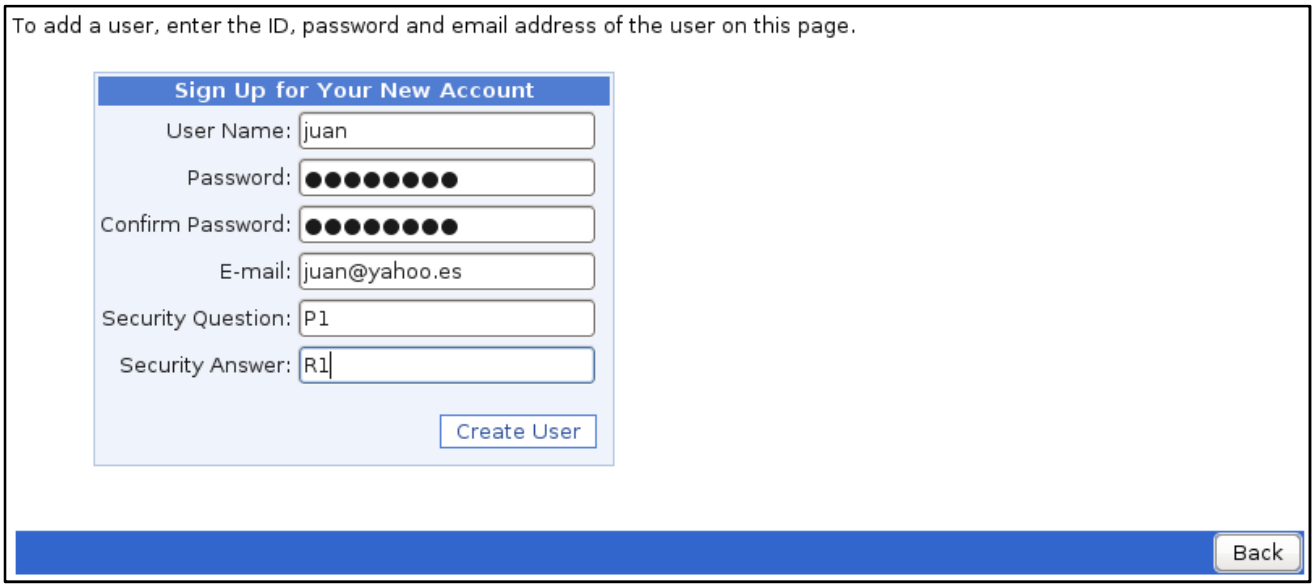

Figura 30 Interfaz para agregar un nuevo usuario a nuestra aplicación

9.5.4 Sección User - > Managing users

Esta sección permite editar la descripción y el estado de un usuario así como asignar a un usuario los roles que deseamos que posea dentro de nuestra aplicación.

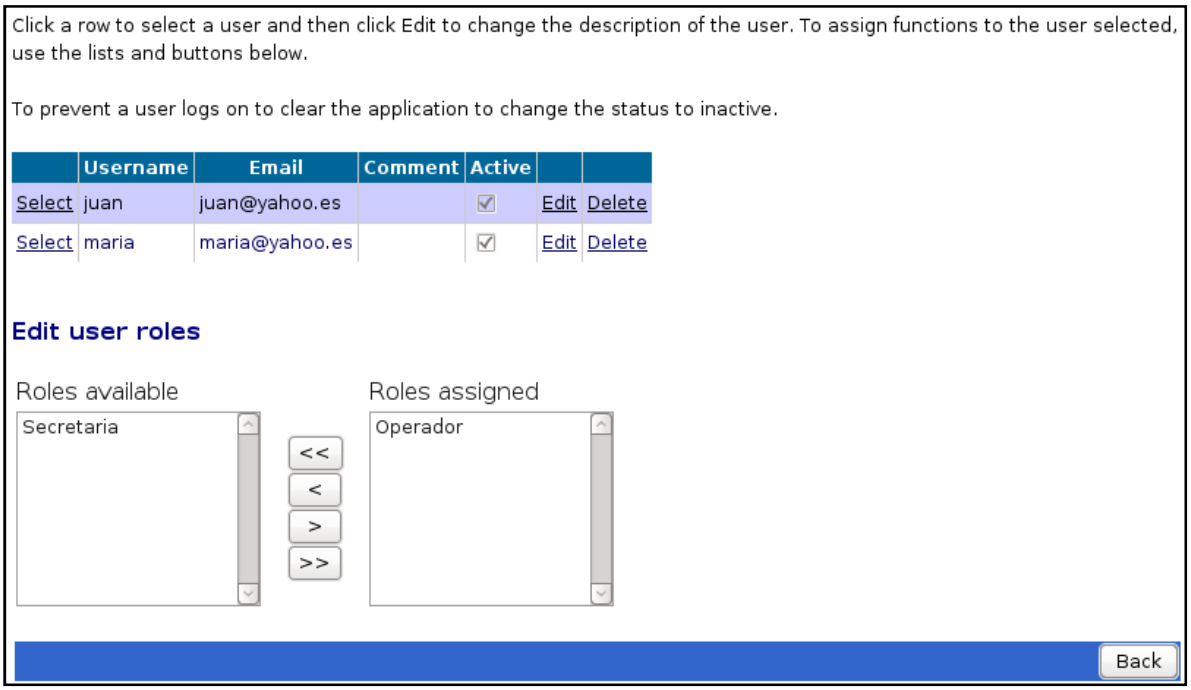

Figura 31 Interfaz administrar usuarios de la aplicación

En el ejemplo anterior, podemos observar como el usuario **juan** que antes hemos creado, pertenece al rol **Operador** pero no al rol **Secretaria.**

9.5.5 Sección Access rules

Es una de las secciones más importantes del sitio, pues permite agregar reglas de acceso para las diferentes partes de nuestro sitio web. Para probar esta parte, hemos creado la siguiente estructura de archivos en nuestra aplicación:

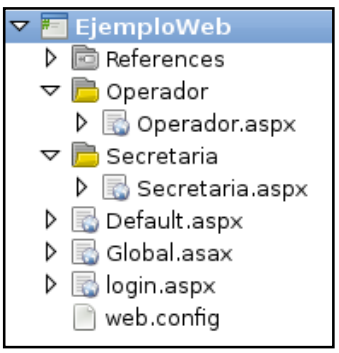

Figura 32 Árbol de directorio de la aplicación empleada en este ejemplo

Las reglas de acceso que hemos aplicado a cada una de las carpetas son:

1. A la raíz del sitio, todos los usuarios tienen acceso.
2. A la carpeta Operador que contiene la página Operador.aspx, sólo deben tener acceso aquellos usuarios que estén en el rol Operador.
3. A la carpeta Secretaria que contiene la página Secretaria.aspx, deben tener acceso aquellos usuarios que estén en el rol Secretaria.

Todo esto lo configuramos desde la primera opción de la sección Access rules. Será necesaria una página llamada login.aspx donde serán redirigidos los usuarios siempre que necesiten autenticarse para poder acceder a un recurso determinado.

9.5.6 Sección Access rules -> Create access rules

Esta opción nos permite agregar reglas de acceso a los directorios de nuestra aplicación. Para crear una regla de acceso debemos:

1. Seleccionar el directorio al que deseamos aplicar la regla de acceso desde el árbol de directorios mostrado en la parte izquierda de la página.
2. Seleccionar a quién se debe aplicar la regla (un rol, un usuario o lista de ellos separados por coma, todos los usuarios o usuarios anónimos).
3. El permiso que se debe aplicar y que puede ser: Allow o Deny (permitir o denegar).

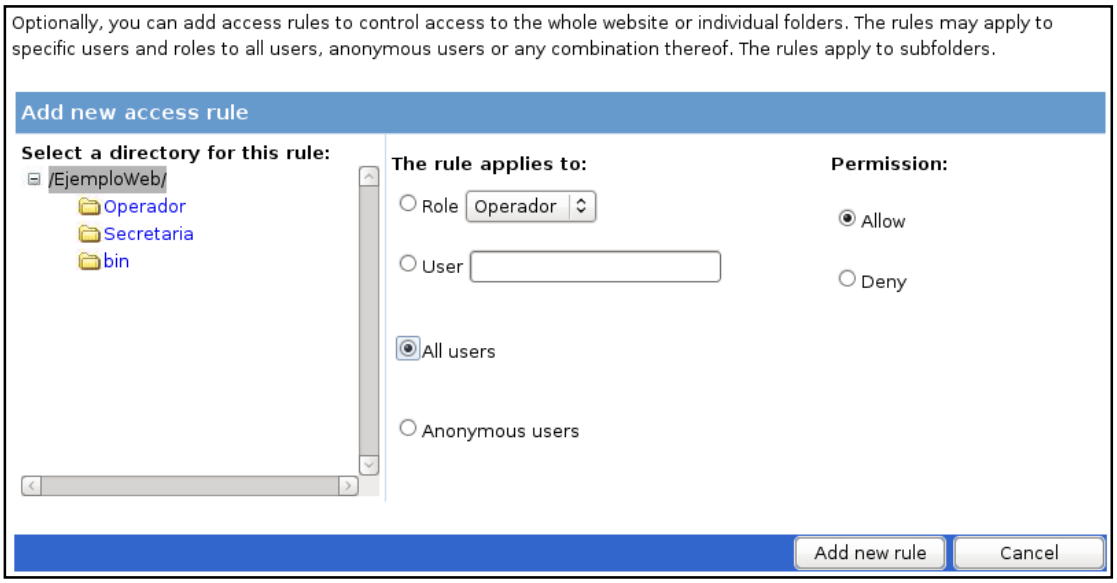

Figura 33 Interfaz para agregar una regla de acceso a una carpeta de nuestra aplicación

Al agregar una regla se realizan las siguientes operaciones:

1. Si no existe un archivo web.config en la carpeta seleccionada, se creará uno.
2. Se revisará que el web.config contenga la sección authorization; de no ser así, se agrega.
3. Se obtienen, si existen, las reglas del directorio actual y se valida que la regla que se desea agregar no se encuentra ya en el web.config.
4. Se agrega la regla al web.config.

9.5.7 Sección Access rules -> Manage access rules

Desde esta página se pueden editar las reglas de acceso que se hayan creado para cada una de las carpetas de la aplicación teniendo las posibilidades de:

- Cambiar el orden de aplicación de las reglas.
- Eliminar una regla de acceso.
- Agregar una nueva regla de acceso.

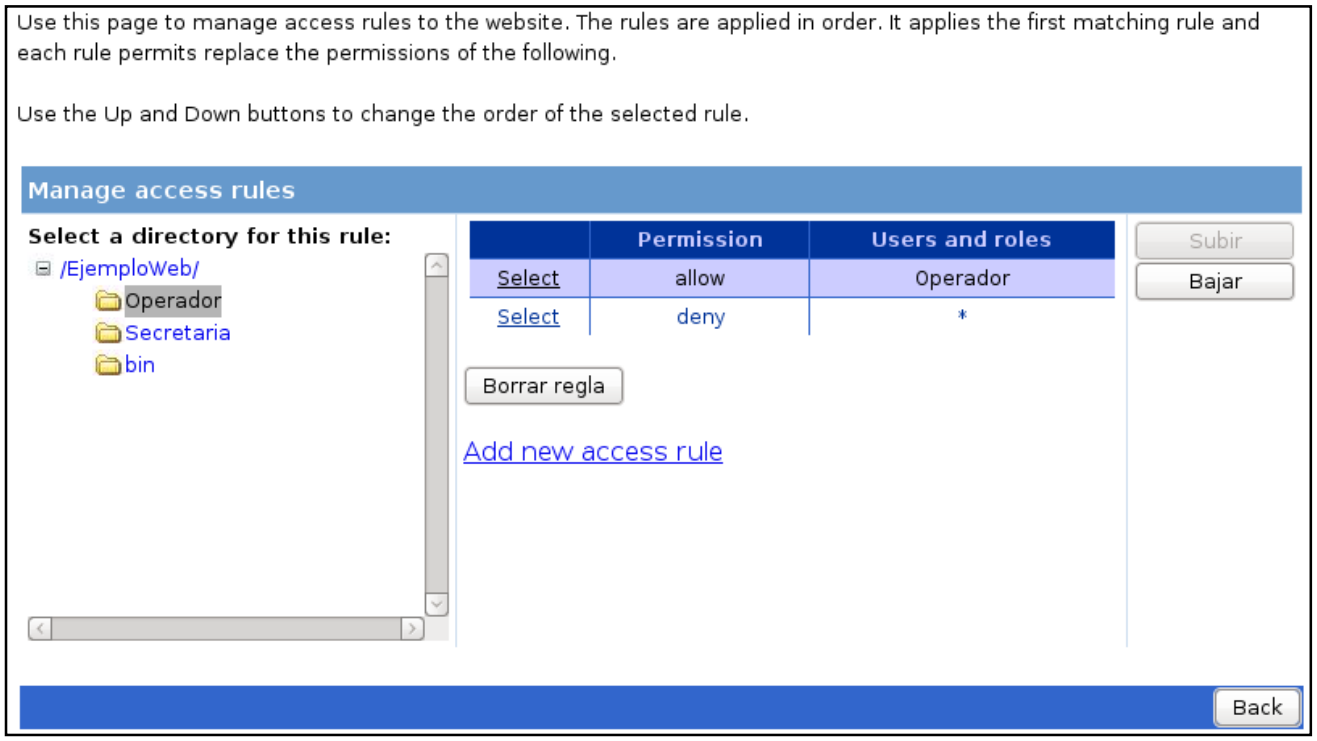

Figura 34 Interfaz para la administración de las reglas de acceso que muestra las reglas aplicadas a la carpeta Operador de la aplicación

Esta imagen nos muestra las reglas que hemos aplicado a la carpeta operador.

9.5.8 Página de error

Si se ha producido alguna modificación en el web.config que impida la carga correcta de la página, se enviará la siguiente página de error:

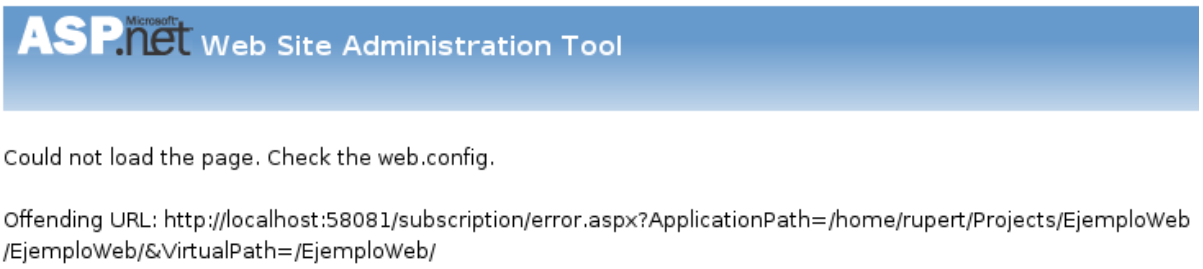

Figura 35 Página de error por defecto de la aplicación

X. Bibliografía

1. *Addin development basics*. (17 de 10 de 2010). Recuperado el 3 de 11 de 2011, de MonoDevelop: http://monodevelop.com/Developers/Articles/Addin_development_basics
2. Ceballos, F. J. (2008). *Aplicaciones .NET multiplataforma.* Madrid: RA-MA Editorial.
3. Ceballos, F. J. (2009). *Enciclopedia de Microsoft Visual C#.* Madrid: RA-MA Editorial.
4. Group, P. G. (2011). *About PostgreSQL*. Recuperado el 20 de 11 de 2011, de PostgreSQL: http://www.postgresql.org/about/
5. Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software.* Madrid: Pearson Educación, S.A.
6. Lluis. (1 de 5 de 2010). *API Overview*. Recuperado el 13 de 11 de 2011, de MonoDevelop API: http://monodevelop.com/Developers/Articles/API_Overview
7. *Managing Authorization Using Roles*. (2008). Recuperado el 2 de 11 de 2011, de MSDN: http://msdn.microsoft.com/en-us/library/9ab2fxh0.aspx
8. *Managing Users by Using Membership*. (2008). Recuperado el 2 de 11 de 2011, de MSDN: http://msdn.microsoft.com/en-us/library/tw292whz.aspx
9. *Membership providers for MySQL*. (2006). Recuperado el 24 de 1 de 2011, de Web Ref: http://www.webref.eu/asp-net-security-mysqlmembershipprovider.php
10. MonoDevelop. (13 de 12 de 2011). *Home MonoDevelop*. Recuperado el 3 de 11 de 2011, de MonoDevelop: http://monodevelop.com/
11. Mono-project. (2010). *ASP.NET*. Recuperado el 11 de 11 de 2011, de Mono-project: http://www.mono-project.com/ASP.NET
12. *Role provider for MySQL*. (2006). Recuperado el 25 de 1 de 2011, de Web Ref: http://www.webref.eu/asp-net-security-mysqlroleprovider.php
13. slluis. (24 de 1 de 2011). *Mono Addins*. Recuperado el 11 de 11 de 2011, de Mono Addins: http://monoaddins.codeplex.com/

XI. Contenido del CD

El CD que se entrega junto con este trabajo contiene los siguientes elementos:

1. Una copia de este documento en los formatos PDF y DOCX.
2. La carpeta **MonoDevelop.Tesis.SubscripcionService** que contiene la solución con los 5 proyectos empleados para el desarrollo de este addin.
3. La carpeta **SubscriptionServicesPgSQL** que es el instalador del addin.
4. La carpeta **Demostracion** que contiene la solución empleada en el ejemplo que se muestra en la sección Manual de la aplicación.