Notification Server

1. Conceptos Generales

1.1. Senders

Los *senders* son los encargados de realizar la entrega de una nofificación a su canal de destino. Todos los *Senders* implementan la interface *NotificationServer.Core.ISender,* que está definida así:

Método	**Propósito**	**Tipo de Retorno**
Send(*Notification*)	Envíar notificaciones sincrónicamente.	*void*
SendAsync(*Notification*)	Envíar notificaciones asincrónicamente.	*void*

1.1.1. El objecto *Notification*

El parámetro de entrada para los métodos *Send* y *SendAsync* es una instancia de *Notification*. Este objeto está definido así:

Propiedad	**Descripción**	**Tipo de Dato**
PartitionKey		*Guid*
TemplateName		*String*
From	Cadena de quién envía la	*String*

	notificación.	
Subject	El asunto de la notificación.	*String*
To	Colección de direcciones a la que se debe enviar la notificación.	*IList<String>*
CC	Colección de direcciones a la que se debe enviar en copia la notificación.	*IList<String>*
BCC	Colección de direcciones a la que se debe enviar en copia oculta la notificación.	*IList<String>*
ReplyTo	Colección de direcciones a las que se debe contestar la notificación.	*IList<String>*
Attachments	Colección de adjuntos que se deben enviar con la notificación.	*Collection<Attachment>*
Properties	A collection of properties that should be used to render body template.	*IList<NotificationProperty>*

Type		*String*
Tags		*String*

1.1.2. El objecto *Attachment*

Propiedad	**Descripción**	**Tipo de Dato**
ContentStream	Representa el flujo de datos del adjunto.	*System.IO.Stream*
Content	Representa el flujo de datos del adjunto representado en *bas64*.	*String*
ContentDisposition	Proporciona la información de presentación para los adjuntos.	*System.Net.Mime.Conten*
Name	MIME content type name value in the content type associated with this attachment.	*string*
NameEncoding	An System.Text.Encoding value that specifies the type of name encoding. The default value is determined from the name of the attachment.	*System.Text.Encoding*

1.1.2. El objecto *NotificationProperty*

Propiedad	Descripción	Tipo de Dato
Key	Cadena que será utilizada como el nombre del la propiedad.	*String*
Value	Valor de la propiedad de la notificación.	*Object*

1.2. Template Engines

Define el contrato requerido para implementar un servicio de motor de plantilla. Los motores de plantilla son los responsables de traducir una notificación en un mensaje que el canal de destino pueda interpretar.
Todos los *Template Engines* implementan la interface *NotificationServer.Core.ITemplateEngineService*, que está definida así:

1.3. Interceptors

Los *Interceptors* permiten leer y manipular mensajes antes y después de su transferencia mediante un sender.
Todos los *Interceptors* implementan la interface *NotificationServer.Core.INotificationInterceptor*, que está definida así:

Método	Propósito
OnNotificationSending(*NotificationSendingContext*)	Este método será llamado antes de enviar

	cada mensaje.
OnNotificationSent(*NotificationSentContext*)	Este método será llamado después de que se envía cada mensaje.

1.3.1 El objeto *NotificationSendingContext*

Es el objeto de contexto enviado en el evento OnMailSending para permitir que inspeccione la Notification subyacente antes de que se envía. Este objeto está definido así:

Propiedad	Descripción	Tipo de Dato
Sender	El *Sender* que se está usando para enviar la notificación.	*Guid*
Notification	La notificación que se está enviando.	*NotificationServer.Contract.Notification*
Parameters	Parámetros con los que la notificación	*Dictionary<String, Object>*

	fue generada.	
Cancel	Indicador que puede ser activado para evitar que se envíe la notificación.	*Boolean*

1.3.2. El objeto *NotificationSentContext*

Es el objeto de contexto enviado en el evento OnMailSent para permitir que inspeccione la Notification subyacente después de que se envía. Este objeto está definido así:

Propiedad	**Descripción**	**Tipo de Dato**
Notification	La notificación que se está enviando.	*NotificationServer.Contract.Notification*
Parameters	Parámetros con los que la notificación fue generada.	*Dictionary<String, Object>*
Response	Indicador que puede ser activado para evitar que se envíe la	*Object*

	notificación.	
HostSender	Indicador que puede ser activado para evitar que se envíe la notificación.	*String*

1.4. Host

1.5. Providers

Los *providers* son los encargados de implementar las características del *Notification Server* específicas en una tecnología. Por ejemplo, la implementación SQL Server de la persistencia. Existen los siguientes tipos de providers:

1.5.1. Settings Repository

Una vez la notificación es recibida por el *Host,* este debe resolver cada uno de los *Senders* que procesarán esa notificación y la configuración para cada uno de ellos. Todos los *Settings Repository* implementan la interface *NotificationServer.Service.Repositories.IConfigurationsRepository,* que está definida así:

Método	**Propósito**	**Tipo**
GetNotificationSpecsFor(*NotifyCommand*)	Devuelve una lista de	*IEnu*

	especificaciones de emisión, con las instrucciones de como procesar la notificación para cada uno de los *Senders* solicitados.	

1.5.2. Template Repository

Una vez la notificación es son resueltos las instrucciones de procesamiento para cada uno de los *Senders*, se resuelven las plantillas que procesaran el mensaje para cada uno de los *Senders* solicitados. Todos los *Template Repository* implementan la interface *NotificationServer.Core.ITemplateResolver*, que está definida así:

Método	**Propósito**	**Tipo de Retorno**
Resolve(*String*, IDictionary<String, Object>)	Devuelve una plantilla dada su nombre. Los paramétros son de uso opcional y pueden ser utilizados para soportar escenarios mas complejos.	*String*

1.5.3. Notifications Repository

Una vez la notificación es procesada por el *Host* esta es persistida con el fin de poder reanudar el proceso en caso de que algo falle antes de que la notificación sea entregada. Todos los *Notifications Repository*

implementan la interface *NotificationServer.Service.Repositories.INotificationsRepository,* que está definida así:

Método	**P**
Save(*NotifyCommand*)	G ta n D id d n
Get(*Guid*)	D ta n d
AddToBatch(*Guid,Guid*)	A n u p e D id p e
GetBatch(*Guid*)	D li d d tr lo
CancelBatch(*Guid*)	D ta

	n d
ReportNotificationStatus(*ReportNotificationStatusCommand*)	G e o la u n

1.5.4. Users Repository

El *Notific ation Server* dispone de un dashboard para el monitoreo de las tareas notificación recibidas. Este acceso está protegido por los usuario y clave entregados por este provider. Todos los *Users Repository* implementan la interface *NotificationServer.Service.Repositories.IUsersRepository,* que está definida así:

Método	**Propósito**	**Tipo de Retorno**
Exists(*String*)	Devuelve si un nombre de usuario existe.	*Boolean*
Get(*String*)	Devuelve un usuario dado su nombre.	*User*
Get(*Guid*)	Devuelve un usuario dado su id.	*User*
Get(*String,String*)	Devuelve un usuario dado su usuario y	*User*

	contraseña. La contraseña será enviada encriptada utilizando el *IEncryptionService* configurado.	
Insert(*User*)	Guarda un nuevo usuario.	*void*
ChangePassword(*String*,*String*)	Cambia la contraseña de un usuario. La contraseña será enviada encriptada utilizando el *IEncryptionService* configurado.	*void*

1.5. Scheduler

El *Scheduler* es quien se encarga de realizar el agendamiento y control de las tareas de notificación. Todos los *Scheduler* implementan la interface *NotificationServer.Service.Repositories.INotificationsScheduler,* que está definida así:

Método	**Propósito**	**Tipo de Retorno**
Startup(*IAppBuilder*)	Este método es llamado junto a las tareas de arranque del Notification Server. Debe ser utilizado como un sustitudo del	*void*

	Global.asax.	
Add(*NotificationRunner*)	Este método será llamado por cada tarea que se cree a partir de una notificación.	*void*

---

2. *Senders* disponibles

2.1. NotificationServer.Senders.CSScript

Ejecuta código CSScript. La plantilla configurada para este sender debe devolver una aplicación de consola clásica de C#, asi:

```
using System;
namespace HelloWorld
{
    class Hello
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Se debe evitar hacer lecturas de teclado. Todos los parámetros de configuración serán pasados como argumentos en el mismo orden en que fueron guardados en el *Settings Repository*.

2.2. NotificationServer.Senders.DnnCoreMes

Entrega la notificación vía Dotnetnuke. Los parametros de configuración de este *Sender* son:

Nombre	Descripción	Valor por Defec
Host	URL de sitio de DNN.	
Resource	URI de API de notificaciones.	/DesktopModules
User	Usuario de DNN con permisos para la API de notificaciones.	
Password	Contraseña del usuario de DNN con permisos para la API de notificaciones.	
ModuleId	Identificador de DNN para el módulo de	511

	notificación.	
TabId	Identificador de DNN para el *Tab* de notificación.	98
RequestVerificationToken	Valor de Header provisto por DNN para autenticación de la API.	
CookieRequestVerificationToken	Valor de Cookie provisto por DNN para evitar CSRF.	

2.3. NotificationServer.Senders.Mail

Entrega la notificación vía SMTP. Los parametros de configuración de este *Sender* son:

Nombre	Descripción	Valor por Defecto
Host	SMTP Host.	
Port	SMTP Port.	
Usuario	SMTP Username.	
Password	SMTP Password.	
SSL	Indicador si el SMTP requiere	

	SSL.	
From	Valor de *from* por defecto.	

2.4. NotificationServer.Senders.Powershell

Ejecuta código Powershell. La plantilla configurada para este sender debe devolver una script powershell, asi:

```
Write-Host "Hello, World!"
```

Se debe evitar hacer lecturas de teclado. Todos los parámetros de configuración serán pasados como argumentos nombrados. Solo los módulos *built-in* estan cargados por defecto.

2.5. NotificationServer.Senders.Rest

Entrega la notificación vía REST. Los parametros de configuración de este *Sender* son:

Nombre	Descripción	Valor por Defecto
Enabled	Indica si el endpoint REST está habilitado.	
Host	URL del endpoint REST.	
Port	Puerto del endpoint REST.	80

Resource	Recurso REST a consumir.	
Method	Método HTTP a invocar.	POST
ContentType	Tipo de contenido a enviar. Esta parámetro esta directamente relacionado a la forma en como la plantilla entrega el contenido.	application/json
Headers	Parámetros a enviar por cabecera.	
Query	Parámetros a enviar por QueryString.	

2.6. NotificationServer.Senders.SendGrid

Entrega la notificación vía Send Grid. Los parametros de configuración de este *Sender* son:

Nombre	Descripción	Valor por Defecto
Usuario	Usuario de API de Sendgrid.	
Password	Contraseña de API de Sendgrid.	

2.8. NotificationServer.Senders.Twitter

Entrega la notificación vía Twitter. Los parametros de configuración de

este *Sender* son:

Nombre	Descripción	Valor por Defecto
AccessToken	Información provista por Twitter API.	
AccessTokenSecret	Información provista por Twitter API.	
ConsumerKey	Información provista por Twitter API.	
ConsumerSecret	Información provista por Twitter API.	

3. *Template Engines* disponibles

3.1. Razor Template Engines

4. *Core Interceptors* disponibles

4.1. Reply Interceptor

4.2. Options Token Replacing Interceptor

5. *Providers* disponibles

5.1. SQL Server Notifications Repository Provider

5.2. SQL Server Settings Repository Provider

5.3. SQL Server Template Repository Provider

5.4. Azure Table Storage Notifications Repository Provider

5.5. Azure Table Storage Settings Repository Provider

5.6. Azure Table Storage Template Repository Provider

6. *Scheduler* disponibles

5.1. Hangfire Scheduler

7. Despliegue

7.1. Instalación de un *Sender*

7.2. Instalación de un *Template Engine*

7.3. Instalación de un *Interceptor*

7.4. Instalación de un *Notifications Repository Provider*

7.5. Instalación de un *Settings Repository Provider*

7.6. Instalación de un *Template Repository Provider*