Componente .NET para el acceso a datos de Profesionales

Servicio Andaluz de Salud
Consejería de Sanidad, Presidencia y Emergencias

Contenido

- Dirección General de Sistemas de Información y Comunicaciones
  - Áreas de Gobierno Tecnológico de SSII y del Dato
  - Introducción
  - Dependencias
  - Errores conocidos
  - Historial de cambios
  - Integración de la librería
  - Configuración
  - Uso

Resumen

- **Versión:** 1.0.0
- **Fecha publicación:** 5 ago 2020
- **Entrada en vigor desde:** 5 ago 2020

blocked URL

Cumplimiento normativo

Las normas expuestas son de obligado cumplimiento. La STIC podrá estudiar los casos excepcionales los cuales serán gestionados a través de los responsables del proyecto correspondiente y autorizados por el Área de Gobernanza de la STIC. Asimismo cualquier aspecto no recogido en estas normas deberá regirse en primera instancia por las guías técnicas correspondientes al esquema nacional de seguridad y esquema nacional de interoperabilidad según correspondencia y en su defecto a los marcos normativos y de desarrollo software establecidos por la Junta de Andalucía, debiendo ser puesto de manifiesto ante la STIC.

La STIC se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata según la planificación de cada proyecto.

En el caso de que algún actor considere conveniente y/o necesario el incumplimiento de alguna de las normas y /o recomendaciones, deberá aportar previamente la correspondiente justificación fehaciente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida por la STIC para proceder a su validación técnica.

Contacto Arquitectura: l-arquitectura.stic@juntadeandalucia.es

Histórico de cambios

Los cambios en la normativa vendrán acompañados de un registro de las modificaciones. De este modo se podrá realizar un seguimiento y consultar su evolución. Ordenándose de mas recientes a menos recientes, prestando especial cuidado a las cabezeras de la tablas dónde se indican las fechas de entrada en vigor y versión.

Versión	v02r00	**Fecha publicación**	26 de octubre de 2018	**Fecha entrada en vigor**	1 de enero de 2019
Alcance					

- Se actualizan todas las referencias al repositorio de código SVN incluyendo GIT como nuevo repositorio.
- Dentro del entorno tecnológico se incluye un enlace para la descarga de una instancia de Sonarqube 6.7.2 parametrizada con los datos de la STIC
- Características del fichero de configuración: se ha incluido una tabla con los códigos a incluir en la propiedad groupId. Esta tabla contiene una codificación de las diferentes suite de aplicaciones existentes en la CMS
- Se ha incluido cual es objeto de las PLs de "Correctivo de datos" y "Carga de datos" y cual es el contenido de los scripts que se adjuntan
- Verificación de calidad del código fuente: se indican cuales son la nuevas métricas Sonarqube a revisar, umbrales mínimo y de confianza y los criterios para la revisión de PLs en FARO
- Calidad de test unitarios: para aquellos proyectos JAVA se incluye el plugin PITEST como parte de la ejecución en JENKINS con el objeto de medir la calidad de los test unitarios realizados.
- Normativa CNO: será de obligatorio cumplimiento todas las reglas descritas en la herramienta CNO
- Ficha de Entrega en PLs: Se elimina la obligatoriedad de adjuntar la ficha de entrega en las PLs de FARO. Esta información es mantenida por el proveedor de desarrollo en GIT mediante el fichero readme.md

Versión	v03r00	**Fecha publicación**		**Fecha entrada en vigor**	
Alcance					

- Verificación de calidad del código fuente: se indican cuales son la nuevas métricas Sonarqube a revisar, umbrales mínimo y de confianza y los criterios para la revisión de PLs en FARO

Introducción

En el presente documento se presenta la librería **SAS.ProfesionalApiClient**, componente .NET Standar 2.0, para el acceso a datos de Profesionales.

Esta librería permite el acceso a los datos de Profesionales utilizando por ahora réplica de BBDD.

Dependencias

Nombre del componente	Versión	Descripción
SAS.Common.Core	1.1.0	Componente .NET común con clases e interfaces core de aplicaciones
SAS.Common.Kernel	1.1.0	Componente .NET común kernel de aplicaciones: interfaz de mapper, extensión de linq, etc
SAS.Common.OracleProvider	1.3.3	Componente .NET común para la configuración del proveedor de BBDD Oracle
EntityFramework	6.2.0	ORM de Microsoft .NET

Errores conocidos

Historial de cambios

Histórico de cambios realizados en la librería:

- 1.0.0 Versión Inicial
- 1.1.0 Se separa el Core de la api y se actualizan versiones de librería comunes (core, kernel)
- 1.2.0 Contexto no manejado
- 1.3.0 Refactorizar librería contexto no manejado
- 1.4.0 Se añade posibilidad de configurar tipo de contexto utilizado, manejado/nomanejado
- 1.4.1 Se actualiza la versión de sas.common.core a la 1.1.0 **SAS.ProfesionalApiClient.Core**
    - Boundary\Contracts\Repositories

      IProfesionalRepository

Profesional GetProfesionalByCodigo(string codigoProfesional);
IList<Profesional> GetProfesional(QueryCriteriaProfesional profesionalCriteria);

- Boundary\Contracts\Services

  IProfesionalService

  Profesional GetProfesionalByCodigo(IGenericAuthContext<T> context, string codigo);
  IList<Profesional> GetProfesional(IGenericAuthContext<T> context, QueryCriteriaProfesional queryCriteriaProfesional);

- Entity

  QueryCriteriaProfesional (CodigoMaco, CNP,Nombre, Apellido1, Apellido2)
  Profesional (CodigoMaco, CNP,Nombre, Apellido1, Apellido2)

SAS.ProfesionalApiClient

- Boundary\

  ProfesionalService: Servicio para acceder a datos de profesional desde BBDD. Implementa la interfaz IProfesionalService<object>

SAS.ProfesionalApiClient.ManagedDataAccess.Oracle

- Boundary\

  ProfesionalContextConnection. Configuración de conexiones a la api. Implementa la interfaz IConfigContextApiConnection<SchemaProfesionalType>
  ProfesionalRepository. Implementa la interfaz IProfesionalRepository

    - Control\Mapper

      Conjunto de clases que mapean la información a las entidades del sistema ProfesionalesMapper ProfesionalMapper
    - Entity\EntityFramework Maco: Entidades del sistema Maco. PersonasEntity y ProfesionalesEntity MacoConexion. Esquema de conexión de Maco no manejado MacoManejadoConnect. Esquema de conexión manejado de Maco
    - Entity\Projection Entidades proyección utilizadas en el sistema: ProfesionalProjection
    - Entity

      SchemaProfesionalType. Enumerado de esquemas permitidos.
- 1.5.0 Se amplia el método GetProfesional para que, en el caso de que el filtro tenga relleno módulo o unidad centro, devuelva profesionales filtrando por estos parámetros.
- 1.5.1 Se cambia el tipo del atributo Modulo en la entidad Profesional. De tipo string pasa a ser decimal? (puede ser nulo)
- 1.5.2 Se actualiza la versión de los componentes Oracle.ManagedDataAccess y Oracle.ManagedDataAccess. EntityFramework a la versión 19.6.0
- 1.6.0 Se incluyen OrderBy por el campo description a consultas que devuelven listado de entidades con dicho atributo.
- 1.6.1 Se regulariza la versión de MacoApiClient a 2.4.2 y BduApiClient a v1.0.3
- 1.6.2 Se corrige un error en la ordenación de profesionales por apellidos/nombre
- 1.7.0 En esta versión se añade la siguiente funcionalidad: - Filtro por CNP en el método ya existente GetProfesionalesByModulosCentro - Nuevo servicio TipoProfesionalService, que permite la consulta de tipos de profesionales por código. `TipoProfesional GetTipoProfesionalByCodigo (IGenericAuthContext<object> context, string codigo);`
- 1.7.1 Se regulariza (en el proyecto Test) la versión del componente SAS.MacoApiClient a v2.4.3, la cual permite comunicación con servicios Maco usando HTTPS

Integración de la librería

Configurar en Microsoft Visual Studio un nuevo origen de paquetes Nuget con la dirección de los artefactos .NET del SAS. Para ello, seguir las indicaciones de la página Repositorio de artefactos. Por último, instalar la librería SAS. ProfesionalApiClient.

Configuración

Debido a la versión de EntityFramework y del componente proveedor de BBDD Oracle, la librería requiere el fichero de configuración .config.

- En el fichero de configuración de la aplicación, **app.config**, se define la cadena de conexión de la BBDD de Maco, ya que en la versión actual, es el esquema en el que se encuentra los datos de profesionales.
- En el fichero **Startup.cs**, método **ConfigureServices**, se inicializa el contenedor de inyección de dependencia

  public void ConfigureServices(IServiceCollection services)

```
{
    ...
        ContainerDI.SetAsyncScopedLifestyle(); // establecer estilo de vida por defecto del contenedor de inyección. AsyncScopedLifestyle permite procesar tanto operaciones asíncronas como síncronas
    ContainerDI.RegisterServices(ref services);
    ...
}
```

ContainerDI, clase del componente común SAS.Util.Injection, el cual permite el registro por inyección de dependencia de los diferentes servicios y tipos utilizados en el sistema.

- En el fichero **Startup.cs**, método **InitializeContainer**, se ha definido las instrucciones necesarias para configurar el componente de Profesionales, así como la inyección de dependencia de los servicios/repositorios utilizados en éste componente.

```
                    ContainerDI.
Register<IConfigContextApiConnection<SchemaProfesionalType>>(() =>
       new ProfesionalContextConnection(
          new ManagedConfigConnectionBase<SchemaProfesionalType>[]
              {
                    new ManagedConfigConnectionBase<SchemaProfesionalType>
(ConfigurationManager.ConnectionStrings["MACOConnection"].ConnectionString,
SchemaProfesionalType.Maco, "REP_PRO_MACO")
              })
       , ContainerDI.LifeStyleContainer.Transient);

        ContainerDI.Register<IProfesionalRepository, ProfesionalRepository>
(ContainerDI.LifeStyleContainer.Transient);
        ContainerDI.Register<IProfesionalService<object>, ProfesionalService>
(ContainerDI.LifeStyleContainer.Transient);
```

Uso

Definir en el constructor de la clase donde se desea utilizar servicios de Profesionales la interfaz de ésta que se desee utilizar.En este caso IProfesionalService:

```
  /// <summary>
  /// Interfaz de servicio al componente de profesionales.
  /// </summary>
  private readonly IProfesionalService<object> servicioProf;

   public ProfesionalService(IProfesionalService<object> servicioProf)
      {
          this.servicioProf = servicioProf;
      }
      public Profesional GetProfesional(string codProfesional)
      {
          // Obtenemos la informacion del profesional
          ProfesionalApiClient.Core.Entity.Profesional prof = this.servicioProf.GetProfesionalByCodigo
(null, codProfesional);

          // Si no existe informacion asociada se devuelve excepción
          if (prof == null)
          {
              throw new ValidacionAccesoException("ERR-VAL-09", new object[] { codProfesional });
          }

          // Si el CNP es 0 se devuelve excepción, ya que no se permiten crear peticiones para
profesionales con CNP cero
          if (prof.CNP == "0")
          {
              throw new ValidacionAccesoException("ERR-VAL-15");
          }

          Profesional domProf = new Profesional();
          domProf.Apellido1 = prof.Apellido1;
          domProf.Apellido2 = prof.Apellido2;
          domProf.CNP = prof.CNP;
          domProf.Codigo = prof.CodigoMaco;
          domProf.Nombre = prof.Nombre;

          return domProf;
      }
```