Patrones de nombrado de servicios

A fin de unificar un poco los criterios para el nombrado de servicios e interfaces utilizaremos esta tabla con los patrones de nombres:

Objeto	**Patrón**
Servicios	[Nombre Servicio] + Service → [Class]
Request	[Nombre Servicio] + Request → [Class]
Response	[Nombre Servicio] + Response → [Class]
Esquema para modelar el objeto de negocio de un Request	[Nombre Servicio] + REQ.xsd → [File]
Esquema para modelar el objeto de negocio de un Response	[Nombre Servicio] + RES.xsd→ [File]

La idea de establecer estos patrones de nombrado no son a fin de mejorar la performance. Solo se trata de unificar criterios de desarrollo y convenciones de nombrado para poder hacer más legible y ordenada la ubicación de los componentes generados por la fábrica de software.

De esta manera, cuando se detecten algunas PROBLEMÁTICAS en el correcto funcionamiento de algún componente y quien tenga la responsabilidad de solucionarlo, no requiera de mucho esfuerzo en el entendimiento de cómo fueron construidos los servicios y de una manera intuitiva pueda ubicar sus subcomponentes con la menor transferencia de know-how tecnológico entre el personal del área.

Servicio Search:

Los servicios de búsqueda retornan más de un ítem, por ejemplo: Buscar clientes por nombre. Se dice que es un servicio de búsqueda cuando estamos frente la situación de retornar un vector, es decir un conjunto de valores.

Nomenclaturas de ejemplo

- ✓ **Search**ClientsByNameService
- ✓ **Search**ClientsByParamsService

Donde la interfaz de entrada podría ser una clase cualquiera llamada Params con atributos necesarios como tipo de búsqueda, nombre etc.-

Y la interfaz response que retorne una BE de lista de clientes

Servicio Get:

Estos servicios a diferencia de los anteriores obtienen un único elemento con mayor grado de detalle que los datos que traería un elemento de la colección de un servicio de búsqueda.

Nomenclaturas de ejemplo:

- ✓ **Get**ClientByIdService
- ✓ **Get**ProductCatalogByIdService

Donde la interfaz de entrada podría ser una clase cualquiera llamada Params con atributos necesarios como Id, guid de blocking etc.

Y la interfaz de retorno retorne una BE de clientes. (NO una lista)

Servicios de Creación y actualización

Estos servicios a diferencia de los anteriores tienen un poco de mayor complejidad en la interfaz de entrada es decir el request y por lo general no retornan datos o suelen retornar algún Id de creación en el caso de los servicios de creación.-

- ✓ **Create**CleintService
- ✓ **Update**ClientService

Servicio de eliminado o cambio de vigencia

Ejemplos descombrados:

- ✓ **Delete**ClientByIdService

Validaciones de existencia u otros:

- ✓ **Exist**ClientByParams
- ✓ **Validate**ClientAcountBalanceByParamService
- ✓ Etc.

Patrones Param/Result

En casos donde no se sepa exactamente que objeto debe enviar un Request o retornar un response, se pueden poner objetos que envuelvan la lógica del contrato de servicio.- En estos casos utilizaremos los nombres genéricos de dependiendo si es un contrato request o response:

Request → Param

Response → Result

Ejemplo:

Clase Request

```
[Serializable]
    public class SearchClientsByParamsRequest : Request<Param>
    {
        public SearchClientsByParamsRequest()
        {
            this.ServiceName = "SearchClientsByParamsService";
        }
    }

    [XmlInclude(typeof(Param)), Serializable]
    public class Param : Entity
    {
        private String _UserName;
         private String _StartDate;
        private bool _ActiveRecordsOnly;

        #endregion

        #region [Properties]
         ///Propiedades aqui
         #endregion
     }
```

Clase Response

```
[Serializable]
public class SearchClientsByParamsResponse : Response<Result>
{}

[XmlInclude(typeof(Param)), Serializable]
public class Param : Entity
 {
    private Clients _Clients;
    private ClientsLogs_HistoryLogs;
    private int _TotalClientRemoved;

        #endregion

        #region [Properties]
         ///Propiedades aqui
         #endregion
     }
```