

**Cátedra Proyecto Final**

**Sistema:** Home Safe Home

**Tema:** Seguridad en departamentos y edificios

Docentes:

* Gastañaga, Iris Nancy (Titular)
* Aquino, Francisco Alejandro (JTP)
* Arenas, Maria Silvina (JTP)
* Jaime, Maria Natalia (JTP)

Integrantes:

* Campos, Diego 57596
* Luna, Franco 55388
* Marchetti, Diego 40704
* Tavorda, Marcos 41876

Curso 5K4 - Año: 2018

Estándares, Herramientas y Tecnologías

Versión 1.0

Historial de Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 18/06/2018 | 1.0 | Primera versión del documento | Diego Campos |
|  |  |  |  |

Tabla de Contenidos

Contenido

[1. Introducción 4](#_Toc517721516)

[1.1 Propósito de este documento 4](#_Toc517721517)

[1.2 Alcance 4](#_Toc517721518)

[2. Desarrollo 4](#_Toc517721519)

[2.1 Tecnologías 4](#_Toc517721520)

[2.2 Herramientas 6](#_Toc517721521)

[2.3 Convención y estándares de codificación: 7](#_Toc517721522)

[2.3.1 Reglas Generales 7](#_Toc517721523)

[2.3.2 Buenas Practicas 7](#_Toc517721524)

[2.3.3 Base de datos 8](#_Toc517721525)

[2.3.4 Código C# 8](#_Toc517721526)

[2.3.5 Servicio Web 8](#_Toc517721527)

[2.3.6 Documentación 11](#_Toc517721528)

Estándares, Herramientas y Tecnologías

# Introducción

Este documento describe las tecnologías, herramientas y estándares que serán utilizados durante el proceso de desarrollo y modificación del Software de Home Safe Home

## Propósito de este documento

Facilitar la información de referencia necesaria a las personas implicadas en el desarrollo del sistema Home Safe Home, como así también dejar en claro las reglas y prácticas que serán utilizadas. Ante cualquier inconveniente de discusiones sobre cómo desarrollar algún modulo o herramienta nueva a utilizar quedara plasmado en este documento. Es un documento que estará presente en todo el ciclo de vida del proyecto, por ende tendrá varias modificaciones.

## Alcance

Este documento será utilizado por el grupo completo del proyecto, debido a que todos vamos a desarrollar.

# Desarrollo

## Tecnologías

En el desarrollo y despliegue del producto Home Safe Home utilizaremos las siguientes tecnologías:

* Respecto la base de datos relacional, usaremos **Oracle SQL Server**.



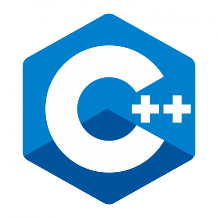
* Respecto el desarrollo del servidor web utilizaremos el Framework **ASP.NET Web Api**.



* Respecto a la aplicación tanto web como móvil utilizaremos **React-native**.



* **C++** para el desarrollo del programa en el **Arduino**.

## Herramientas

Para el desarrollo del proyecto utilizaremos las siguientes tecnologías:

* Para el manejo del repositorio vamos a utilizar **GitHub**



* Para los entornos de desarrollo vamos a utilizar PL SQL Developer, Visual Studio Comunnity y Visual Studio Code

* Para las pruebas del servicio web utilizaremos



* Para la gestión del proyecto, tanto en el seguimiento como en la organización del mismo, utilizaremos Redmine



## Convención y estándares de codificación:

### Reglas Generales

* Usar palabras entendibles y descriptivas para nombrar, no usar abreviaciones.
* Se recomienda el desarrollo empleando partes sencillas, comprensibles y con errores de fácil detección y corrección.
* Comentar lo máximo posible los métodos del código.
* Usar TAB, NO espacios.
* Mantener el formateador de Visual Studio (Ctrl+K, Ctrl+D) o Visual Code (alt+shift+F)

Nota, palabras utilizadas:

* UpperCamelCase, cuando la primera letra de cada una de las palabras es mayúscula. Ejemplo: EjemploDeUpperCamelCase.
* lowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeLowerCamelCase.

### Buenas Practicas

Seguir los patrones S.O.L.I.D. lo mejor posible:

SRP (Single Responsibility Principle):

El principio de responsabilidad única nos indica que debe existir un solo motivo por el cual la clase debe ser modificada, o sea, que la clase debe tener un solo propósito. Si una clase asume más de una responsabilidad, será más sensible al cambio, y las responsabilidades se acoplan.

OCP (Open-Closed Principle):

El principio Abierto/Cerrado indica que las clases deben estar abiertas para la extensión y cerradas para la modificación, o sea, que una clase debe poder ser extendida sin tener que modificar el código de la clase

LSP (Liskov Substitution Principle):

El principio de sustitución de Liskov indica que las clases derivadas (hijas) pueden ser sustituidas por sus clases base. Se promueve que la herencia se realice en forma transparente, o sea, no se debe implementar métodos que no existan en sus clases base ya que de lo contrario se rompe el principio.

“De OCP y LSP se deduce que las clases base (abstractas o no) modelan el aspecto general y las clases heredadas modelan el comportamiento local.”

ISP (Interface Segregation Principle):

El principio de segregación de interfaces indica muchas interfaces muy especializadas son preferibles a una interfaz general en la que se agrupen todas las interfaces

DIP (Dependency Inversion Principle):

El principio de inversión de dependencias indica que las abstracciones no deben depender de los detalles, los detalles deben depender de las abstracciones

### Base de datos

Reglas que rigen en las definiciones de objetos en la base de datos.

#### Tablas

Nombre de las tablas en plural y mayúscula, empiezan con T\_<Nombre Tabla>.

Los atributos se escriben en UpperCamelCase, además debe incluirse en todas las tablas los atributos Id<Nombre Tabla> como identificador y no habrá borrado físico de los registros todo será lógico mediante el atributo fecha baja.

#### Secuencias

Las secuencias tendrían el nombre SEQ\_<Nombre Tabla>.

#### Restricciones

* Primary key: PK\_<Nombre Tabla>
* Foreign key: FK\_<Nombre Tabla que tiene la FK>\_<Nombre Tabla a la que apunta>
* Unique: UK\_<Nombre atributo>

#### Procedimientos almacenados

Nombre de cada uno de los procedimientos básicos, necesarios para dar respuesta al Servicio Web:

* PR\_<Nombre Tabla>\_SF: se utiliza para devolver varios registros, filtrados desde el cliente
* PR\_<Nombre Tabla>\_S: se utiliza para devolver un solo registro filtrado por Id
* PR\_<Nombre Tabla>\_G: se utiliza para guardar un nuevo registro
* PR\_<Nombre Tabla>\_G: se utiliza para modificar un registro
* PR\_<Nombre Tabla>\_DL: se utiliza para eliminar un registro, es decir, modificar la fecha baja, ya que solo hay borrado lógico desde la aplicación
* PR\_<Nombre Tabla>\_SET: se utiliza para devolver varios cursores, con los datos de los diferentes combos de selección.

### Código C#

#### Clases

El nombre del archivo debe coincidir con el nombre de la clase, este último se debe escribir en singular, usando la notación UpperCamelCase.

#### Variables

Las variables se deben escribir en singular, usando la notación lowerCamelCase. Para todas las variables de las entidades se debe respetar el principio de ocultación, es decir, debe poseer propiedades tanto para obtener como modificar.

En visual escribir prop+TAB+TAB, lo escribe automáticamente.

En las variables auxiliares de métodos, se pueden utilizar abreviaciones.

En las variables booleanas usar el prefijo “es”. Ejemplo: esValido

#### Metodos

El nombre del método debe representar la acción realiza, usando la notación lowerCamelCase, deben tener entre 1 y 25 líneas de código, sino necesita refactorización.

### Servicio Web

Mantener la base de la URI simple e intuitiva, basándonos en el principio “Keep simple things simple”, esto nos permite facilitar el enrutamiento al recurso.

El versionado es mandatorio en las API, se debe dar desde la URL, no en los encabezados. Ejemplo (homesafehome.com.ar/api/v1/usuarios).

Se utilizarán el formato JSON para la transmisión de datos.

Los métodos de búsqueda con filtros si poseen datos sensibles, se deberían usar POST, ya que el body con protocolo HTTPS viaja encriptado.

#### Nombre recurso

Utilizar sustantivos en plural para los nombres, no verbos. El nombre de la clase ApiController debe respetar la notación UpperCamelCase

#### Métodos del recurso

Los métodos de las ApiController son verbos y deben respetar la notación UpperCamelCase

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recurso | Get | Post | Put | Delete |
| /Eventos | Lista todos los eventos | Crea un nuevo evento | Modifica todos los eventos | Elimina todos los eventos |
| /Eventos/1 | Muestra el evento con id 1 | Error | Si existe el evento con id 1 lo modifica sino da error | Elimina el evento con id 1 |

#### Errores

|  |  |
| --- | --- |
| Código de respuesta | Descripción |
| 200 OK | Solicitud aceptada; la respuesta contiene el resultado. Este es un código de respuesta general a cualquier solicitud. En las solicitudes GET, el recurso o datos solicitados están en el cuerpo de la respuesta. En las solicitudes PUT o DELETE, la solicitud fue satisfactoria y la información acerca del resultado (como los identificadores de recursos nuevo o los cambios en el estado del recurso) se puede encontrar en el cuerpo de la respuesta. |
| 201 CREATED | Las operaciones PUT o POST devuelven este código de respuesta e indica que se ha creado un recurso de forma satisfactoria. El cuerpo de la respuesta podría, por ejemplo, contener información acerca de un nuevo recurso o información de validación (por ejemplo, cuándo se actualiza un activo). |
| 204 NO CONTENT | Indica que se ha aceptado la solicitud, pero no había datos para devolver. Este respuesta se devuelve cuando se ha procesado la solicitud, pero no se ha devuelto ninguna información adicional acerca de los resultados. |
| 400 BAD REQUEST | La solicitud no fue válida. Este código se devuelve cuando el servidor ha intentado procesar la solicitud, pero algún aspecto de la solicitud no es válido; por ejemplo, un recurso formateado de forma incorrecta o un intento de despliegue de un proyecto de sucesos no válido en el tiempo de ejecución de sucesos. La información acerca de la solicitud se proporciona en el cuerpo de la respuesta e incluye un código de error y un mensaje de error. |
| 401 UNAUTHORIZED | El servidor de aplicaciones devuelve este código de respuesta cuando está habilitada la seguridad y faltaba la información de autorización en la solicitud. |
| 403 FORBIDDEN | Indica que el cliente ha intentado acceder a un recurso al que no tiene acceso. Podría producirse si el usuario que accede al recurso remoto no tiene privilegios suficientes; por ejemplo, con el rol WBERestApiUsers o WBERestApiPrivilegedUsers. Los usuarios que intenten acceder a proyectos de sucesos privados que son propiedad de otros podrían recibir también este error, pero solo si tienen el rol WBERestApiUsers en lugar de WBERestApiPrivilegedUsers. |
| 404 NOT FOUND | Indica que el recurso de destino no existe. Esto podría deberse a que el URI no está bien formado o a que se ha suprimido el recurso. |
| 405 METHOD NOT ALLOWED | Se produce cuando el recurso de destino no admite el método HTTP solicitado; por ejemplo, el recurso de funciones solo permite operaciones GET. |
| 406 NOT ACCEPTABLE | El recurso de destino no admite el formato de datos solicitado en la cabecera de Accept o el parámetro accept. Es decir, el cliente ha solicitado la devolución de los datos en un determinado formato, pero el servidor no puede devolver datos en ese formato. |
| 409 CONFLICT | Indica que se ha detectado un cambio conflictivo durante un intento de modificación de un recurso. El cuerpo de la respuesta contiene más información. |
| 415 UNSUPPORTED MEDIA TYPE | El recurso de destino no admite el formato de datos del cuerpo de la solicitud especificado en la cabecera de Content-Type. |
| 500 INTERNAL SERVER ERROR | Se ha producido un error interno en el servidor. Esto podría indicar un problema con la solicitud o un problema en el código del lado del servidor. Se puede encontrar información acerca del error en el cuerpo de respuesta. |
| *Tabla 1. Códigos de respuesta REST* | |

### Documentación

Agregando “///” al inicio de una clase, el entorno de desarrollo de .NET automáticamente genera una secuencia especial de comentarios sobre el código y se ve de esta manera:

|  |
| --- |
| /// <summary>  /// Pretendo construir un reloj y esta es la descripción de este objeto  /// </summary>  **public** Class Reloj  {  **private** **int** hora { **get**; **set**; }  } |

|  |
| --- |
| <summary> |

**Requerido**  
Se utiliza usualmente para describir un objeto. Es la primera línea de las etiquetas de la documentación. También es utilizado para ayudar al Intellisense cuando se muestra un tooltip sobre un objeto.

|  |
| --- |
| <remarks> |

**Opcional**  
Se utiliza para escribir una descripción detallada de un objeto o método.

|  |
| --- |
| <example> |

**Opcional**  
Se utiliza para describir ejemplos sencillos de la implementación.

|  |
| --- |
| <seealso> |

**Opcional**  
Agrega enlaces asociados al método.

|  |
| --- |
| <param> |

**Requerido**  
Describe un parámetro en un método, también es mostrado en el intellisense.

|  |
| --- |
| <typeparam> |

**Requerido para cada parámetro de tipo genérico**  
Describe un parámetro o método de tipo genérico.

|  |
| --- |
| <returns> |

**Requerido en cada método que retorna un valor**  
Se utiliza para describir el valor a retornar por un método.

|  |
| --- |
| <exception> |

**Opcional**  
Describe una excepción que podría ser lanzada por un método.

|  |
| --- |
| <permission> |

**Opcional**  
Se utiliza para describir el nivel de acceso a un método

|  |
| --- |
| <include> |

**Opcional**  
Utilizado para incluir en la documentación un archivo o parte de otro en el código

|  |
| --- |
| <para> |

**Opcional**  
Utilizado para definir un bloque, solo se utiliza a nivel del elemento de documentación.

|  |
| --- |
| <list> |

**Opcional**  
Crea una lista con uno o diferentes formatos (bullet, number, table). Se utiliza solamente a nivel del elemento de documentación.

|  |
| --- |
| <code> |

**Opcional**  
Se utiliza para especificar el formato que tendrá el código.

|  |
| --- |
| <c> |

**Opcional**  
Similar al anterior, la única diferencia es que “code” permite escribir en múltiples líneas.

|  |
| --- |
| <see> |

**Opcional**  
Se utiliza para definir un enlace a una página interna (cref) o externa (href).

|  |
| --- |
| <paramref> |

**Opcional**  
Hace referencia a un parámetro dentro de otra etiqueta

|  |
| --- |
| <typeparamref> |

**Opcional**  
Hace referencia a un tipo de parámetro genérico