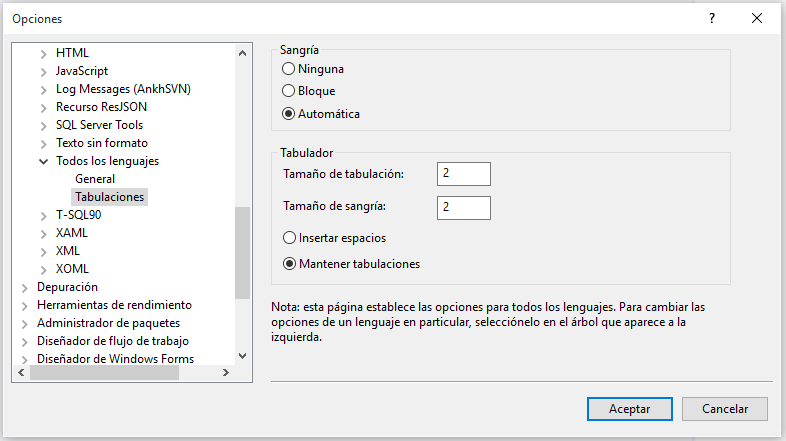
# Reglas en Ambiente de Desarrollo

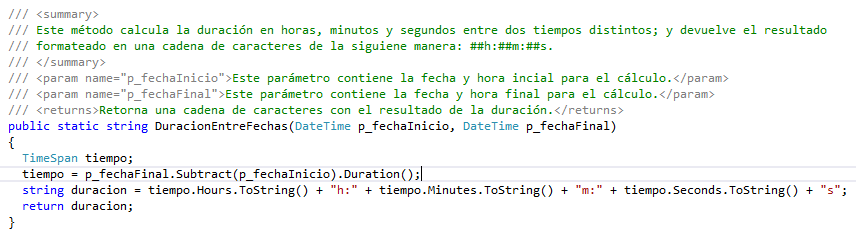
## Consejos

* Una vez que se haya seleccionado una de las actividades de las asignadas, hay que dedicarse completamente a ésta, hasta terminarla, si se la ve muy compleja, se puede requerir ayuda para simplificar su programación, pero no es aconsejable dejarla de lado para realizar otra.
* Si al momento de elaborar una actividad se encontró un problema o error en algo que ya se encontraba funcional, se deberá agregar una nota o comentario indicando la posible existencia de un error, esto se deberá adjuntar a la actividad para luego ser procesada como otra actividad en otro sprint o en el mismo.
* Es importante entender que a menudo nos tocarán tareas fáciles, que podremos realizarlas lo más rápido posible, pero como es el caso estas pueden generar una gran cantidad de errores sino se realizan las pruebas pertinentes; por tal motivo, es aconsejable probar el código y la interfaz varias veces, hasta estar satisfecho.
* Es recomendable prepararse bien según las actividades asignadas que se hayan terminado, ya que pueda ser que se le asigne para presentar estas modificaciones y avances efectuados, en las reuniones del sprint final, que incluyen a los miembros del área de desarrollo, área de QA e inclusive al cliente final.

## Editor de Código Fuente

* Se deberá trabajar únicamente con una sangría de 2 espacios, y se deberá mantener como tabulador el espacio y no como espacios en blanco.



* Si es necesario separar dos palabras hay que utilizar solamente el carácter guión bajo (\_), generalmente esto sucede en constantes.
* Toda variable de una clase deberá llevar el prefijo m\_. Por ejemplo: m\_codigoTercero.
* Toda variable local dentro de una función no deberá llevar ningún prefijo.
* Todo parámetro que se pase a una función deberá llevar el prefijo p\_. Por ejemplo: p\_codigoTercero.
* Todo nombre de una variable de una clase deberá ser escrita utilizando la nomenclatura camelCase. Por ejemplo: m\_codigoTercero.
* Todo nombre de una propiedad de una clase deberá ser escrita utilizando la nomenclatura PascalCase. Por ejemplo: CodigoTercero.
* Todo nombre de función deberá utilizar la nomenclatura PascalCase.
* Todo nombre de método o función deberá incorporar un verbo para indicar que acción se realizará.
* Es necesario que cada vez que se cree una función nueva, se escriba la documentación respectiva; utilizando la funcionalidad de Visual Studio. Por ejemplo:
* Toda constante que se cree deberá estar en mayúsculas. Por ejemplo:

private const int NUMERO\_CASAS = 10.

* Los nombres de los archivos deberán utilizar la nomenclatura PascalCase.

## Ambiente Visual

* En todas las ventanas activas, controles y clippets ya realizados, no se puede modificar los códigos o nombres (ids) de los controles; debido a que estos se utilizarán en las pruebas de automatización, y estas necesitan utilizar los ids existentes.
* Prefijos que serán utilizados según el tipo de control.

|  |  |
| --- | --- |
| **Control** | **Prefijo** |
| Border | brd |
| Button | btn |
| Canvas | cnv |
| CheckBox | chk |
| ComboBox | cmb |
| ContextMenu | cmn |
| DataGrid | dgd |
| DatePicker | dtp |
| DateTimePicker | dtp |
| DockPanel | dck |
| Expander | exp |
| Grid | grd |
| GroupBox | grb |
| Image | img |
| Label | lbl |
| ListBox | lsb |
| ListView | lvw |
| Menu | mnu |
| MenuItem | mni |
| ProgressBar | pgb |
| RadioButton | rbn |
| Rectangle | rct |
| StackPanel | stk |
| TextBlock | txb |
| TextBox | txt |
| Window | wn |
| WrapPanel | wpn |

## Actualización del Repositorio

* Una vez que se hayan hecho los cambios o se haya creado las nuevas funcionalidades descritas en las actividades asignadas, y se hayan compilado sin errores localmente, y se hayan hecho las suficientes pruebas por el desarrollador; se procederá a realizar una actualización (commit) al repositorio.
* Es importante siempre documentar los cambios realizados con buena ortografía, ya que estos circularán por toda el área de desarrollo hasta llegar al cliente final.