**TECNOLÓGICO DE COSTA RICA**

**Escuela de Ingeniería en Computación**

**Proyecto de Ingeniería de Software**

**Profesora:**

María Estrada Sánchez

**Entrega 1:**

Resumen de estilos para OCAML

y estándar implementado.

**Estudiantes:**

Christian León Guevara - 2013371982

Gabriel Ramírez Ramírez - 201020244

**Fecha de entrega:**

22-12-2018

**Período Verano**

**Cartago**

**Tabla de contenido**

[**Introducción.** 3](#_Toc535326573)

[**Lista de reglas de estilo.** 3](#_Toc535326574)

[ **Nombres** 3](#_Toc535326575)

[ **Comentarios** 3](#_Toc535326576)

[ **Espaciado** 3](#_Toc535326577)

[ **Formato** 4](#_Toc535326578)

[ **Identación** 4](#_Toc535326579)

[**Estándar utilizado** 5](#_Toc535326580)

[**Bibliografía** 6](#_Toc535326581)

# **Introducción.**

En este documento se establecen las principales reglas de estilos que se deben seguir a la hora de desarrollar código en OCAML. Son reglas establecidas por distintas fuentes, sin embargo, en este documento se establecen las que llegan a un consenso entre todas.

# **Lista de reglas de estilo.**

Las reglas de estilos es una guía de cómo se debe escribir código, en nuestro caso particular el lenguaje OCAML. Dichas reglas pueden definir convenciones para: escribir nombres, comentarios, espaciado, formato e identación.

El principal principio a la hora de escribir los programas es asegurarse de que sea sencillo y fácil de leer. Ya que la mayor la cantidad de tiempo se dedica a la lectura de código y hacerlo cumpliendo el principio permite la comprensión sencilla de lo que se ha desarrollado. Un aporte importante es que recomienda utilizar el ancho de 80 columnas para permitir la lectura del código en cualquier pantalla e imprimirlo, de ser necesario, en una fuente legible.

* **Nombres:** La regla principal es el uso de nombres significativos para hacer el código entendible.

1. La principal regla es que los nombres para identificadores de valores y tipos deben escribirse con letra minúscula y separadas con guiones bajos.
2. Los constructores deben escribirse con la primera letra en mayúscula y usar minúsculas separadas con guiones bajos.
3. Los módulos deben escribirse con la primera letra en mayúscula y pueden usar mayúsculas y minúsculas separadas con guiones bajos.

* **Comentarios.**

1. Comentar solamente cuando haya alguna dificultad.
2. Evitar los comentarios nocivos y muy largos.
3. Evitar comentar los cuerpos de las funciones, es mejor realizar un comentario al comienzo de la función.
4. Los comentarios se definen dentro de los delimitadores: **(\****Comentario***\*)**

* **Espaciado:** Es necesario siempre separar las palabras con espacios en blanco, nunca debe hacerse usando tabulaciones.
* **Formato:** A continuación, se presentan algunos casos particulares de cómo se debe escribir el código.

1. El uso de delimitadores va seguido por espacio en blanco. Por ejemplo, cuando se utilizan comas y paréntesis.
2. Se debe considerar la escritura de funciones con una altura no superior a las 70 líneas esto con el fin de que pueda visualizarse en una pantalla completa.
3. En los patrones la barra inicial no es necesario colocarla. Y si una definición es muy larga debe romperse a la siguiente línea inmediatamente después de la fecha.

* **Identación:** La identación es importante ya que permite una mejor legibilidad del código.

1. Mantenga siempre el mismo estilo de identación.
2. Considere que en OCAML se respeta el ancho de 80 columnas.
3. Lo recomendable para la identación son 1 o 2 espacios, nunca usar tabulaciones para este fin.
4. El cuerpo de una función normalmente se identa con el nombre de esta y en el cuerpo se utilizan 1 o 2 espacios como mencionamos anteriormente.
5. Existe el caso especial con los patrones, estos se introducen utilizando una barra vertical. Y las cláusulas de coincidencia del patrón se alinean a un mismo nivel de identación.

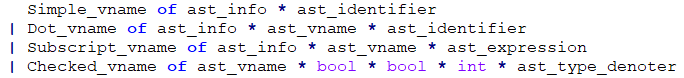
# **Estándar utilizado.**

Parte del proceso de desarrollo, verificación y validación de código del analizador contextual en OCAML implicó el uso de una serie de reglas de estilos que ayudan a la legibilidad y comprensión de este, a continuación, se presenta el estándar utilizado por nuestro equipo de desarrollo:

1. Los nombres se escriben utilizando el estilo snake\_case en lugar del comúnmente utilizado CamelCase.
2. Uso de mayúsculas y minúsculas para establecer nombres:
3. Minúsculas: se utiliza el estilo snake\_case con las iniciales en minúsculas para los identificadores.



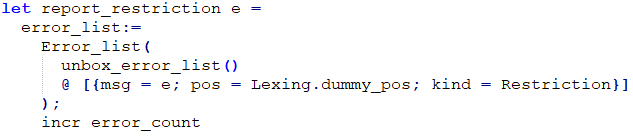
1. Mayúsculas: se utiliza el estilo snake\_case con la primera letra en mayúscula para constructures.



1. Excepción: Dentro del código se presenta una excepción con 2 identificadores: write\_XML\_errors y write\_XML\_tree. Estos siguen el estilo snake\_case sin embargo XML se escribe todo en mayúscula por ser un acrónimo técnico que facilita su comprensión.



1. Identación: El equipo estableció el nivel de la identación de 2 espacios.
2. Se intentó mantener el ancho a 80 columnas. Sin embargo, hay ciertas excepciones dentro del código donde no se pudo mantener y buscando una mejor comprensibilidad se rompió esta regla.
3. Se rompieron las línea de código muy extensas intentando cumplir con el ancho de 80 columnas.
4. Se mantuvo la identación y agrupación con el uso de paréntesis.



# **Bibliografía.**

*Computer Science.* (2018). Recuperado el 20 de Diciembre de 2010, de OCaml Style Guide: https://cs.brown.edu/courses/cs017/content/docs/ocaml-style.pdf

*Cornell CIS - Computer Science.* (s.f.). Recuperado el 20 de Diciembre de 2018, de CS 3110 OCaml Style Guide: https://www.cs.cornell.edu/courses/cs3110/2018sp/handouts/style.html

*OCaml.* (s.f.). Recuperado el 20 de Diciembre de 2018, de OCaml Programming Guidelines: https://ocaml.org/learn/tutorials/guidelines.html