

Universidad de Buenos Aires

Facultad de Ingeniería

Presentación Trabajo Final:

Cleasure IDE

Because it’s a pleasure to work with…

**Tutora:** Rosita Wachenchauzer

**Alumno:**  Juan Martín Muñoz Facorro

**Padrón N°**: 846722

Contenido

[Introducción 3](#_Toc327307140)

[¿Qué es Clojure? 3](#_Toc327307141)

[¿Por qué Clojure? 3](#_Toc327307142)

[¿Qué es un IDE? 3](#_Toc327307143)

[¿Por qué un IDE para Clojure? 3](#_Toc327307144)

[Desafíos 4](#_Toc327307145)

[Ventajas 4](#_Toc327307146)

[Desventajas 4](#_Toc327307147)

[Conclusión 4](#_Toc327307148)

# Introducción

Por medio del siguiente informe se pretende presentar una propuesta para un **Trabajo Final** de la carrera de **Ingeniería Informática** de la **Universidad de Buenos Aires**. Dicho **Trabajo Final** consiste en implementar un **IDE** (Integrated Development Environment) pensado principalmente para trabajar con el lenguaje **Clojure** (aunque extensible a otros lenguaje), que ofrezca una **interfaz** (UI) y **experiencia** **de usuario** (UX) accesible y que fomente la productividad en un entorno gráfico moderno y familiar.

# ¿Qué es Clojure?

**Clojure** es un dialecto de **Lisp**, el cual es a su vez un lenguaje de programación diseñado por John McCarthy en 1957 [REF]. Desde su creación en ese año han surgido un gran número de implementaciones de dicho lenguaje, siendo las más conocidas al día de hoy **Common Lisp** y **Scheme** [REF]. Una de las razones por la cual existe un gran número de implementaciones de este lenguaje, es la simplicidad de su sintaxis, la cual se basa en la utilización de paréntesis y notación prefija [REF] como elemento de expresión.

Lisp is not the right language for any particular problem. Rather, Lisp encourages one to attack a new problem by implementing new languages tailored to that problem.

—“Lisp: A Language for Stratified Design” (Abelson 1988)

**Lisp**, a pesar de ser un lenguaje con gran poder de expresión, por muchos años fue considerado como un lenguaje de uso académico y relacionado con la rama de investigación en Inteligencia Artificial[REF]. No obstante también se lo utilizó para el desarrollo de algunas aplicaciones comerciales como **Viaweb**, uno de los primeros sitios en ofrecer una plataforma web de e-commerce y que luego fue comprado por **Yahoo** (Graham, 2001).

**Clojure** fue creado por Rick Hickey en el 2008 [REF], su implementación le permitía correr sobre la **JVM** (Java Virutal Machine) y diseñado para soportar interoperabilidad con las librerías de existentes Java [REF], pero la comunidad open-source extendió las plataformas donde se puede utilizar este lenguaje creando soporte para el **CLR** (Common Language Runtime) de **Microsoft.NET** y un compilador a **JavaScript** llamado **ClojureScript**.

# ¿Por qué Clojure?

Desde su aparición hace unos años, **Clojure** fue ganando terreno en las comunidades de programación, tanto por su poder de expresión como por la posibilidad de reutilización de librerías existentes de Java.

# ¿Qué es un IDE?

Un **IDE** es una aplicación que ofrece un conjunto de herramientas que facilita el desarrollo de aplicaciones informáticas de distintos tipos, en uno o varios lenguajes determinados. Existen hoy en día **IDE**s tanto de carácter propietario como open-source, siendo los más conocidos el **Microsoft Visual Studio**, utilizado para desarrolar en la plataforma **.NET**, y el **Eclipse** orientado principalmente a desarrollos en **Java**.

El objetivo de todo **IDE** es facilitar la tarea de crear software, de forma tal que se logre aumentar la productividad del usuario y minimizar las operaciones rutinarias o que no aportan valor al producto que se quiere generar.

# ¿Por qué un IDE para Clojure?

La documentación oficial de **Clojure** (Clojure, 2012) lista una seria de opciones para trabajar con este lenguaje. Estas son en su gran mayoría complementos para **IDE**s existentes y otras son editores de texto de consola típicamente utilizados por la comunidad de programación:

* **Counter-Clockwise**: complemento para **Eclipse**.
* **La Clojure**: complemento para **IntelliJ**.
* **Enclojure**: complemento para **Netbeans**.
* **Clooj**: editor básico de Clojure.
* **Emacs**.
* **Vim**.

Históricamente la comunidad de **Lisp** recomienda para programar en este lenguaje o cualquiera de sus dialectos (como lo es **Clojure**), alguna aplicación que pertenezca a la familia **Emacs**. En particular se recomienda **GNU Emacs**, la cual ofrece una alta flexibilidad por su capacidad de extender sus comandos utilizando **Emacs Lisp**, una variante de **Lisp**.

**Emacs**, a pesar de ser una herramienta muy poderosa, sufre la desventaja de ser muy poco amigable, lo que tiene como consecuencia una curva de aprendizaje muy lenta. Entre algunas de las cosas que producen esto, se puede mencionar el hecho que no ofrece una interfaz de usuario familiar, dado que muestra al usuario ventanas de puro texto sin incluir al mouse como un elemento de interacción. Sumado a esto, para poder manejar **Emacs** con naturalidad, se deben incorporar algunos conceptos especiales que esta aplicación utiliza para trabajar, como son los **buffers**, mientras que también se deben aprender algunas definiciones y nombres nuevos, como las operaciones de texto **kill** y **yank**, entre otras (Carper, 2010).

A pesar de apuntar a todos los **IDE**s en general, la siguiente imagen trata de captar la naturaleza de lo expuesto anteriormente (Stanislav, 2012).



Por lo expuesto anteriormente se considera un aporte de gran valor y a la vez un desafío, el desarrollo de un **IDE**, concebido desde el origen para ser utilizado con el lenguaje **Clojure** y extensible a otros lenguajes de programación; utilizando para su construcción el mismo lenguaje **Clojure**.

# Desafíos

**UI/UX**

Como se mencionó en una sección anterior, el principal objetivo de un **IDE** es darle al usuario una herramienta que facilite su trabajo y pueda concentrar sus esfuerzos en aquellas tareas que generan valor. Es por esto que la interfaz del usuario y la experiencia de éste con la aplicación son factores clave para el éxito del presente proyecto.

**Extensibilidad**

Si bien el lenguaje principal para el cual se piensa utilizar el **IDE** propuesto es **Clojure**, se cree que la posibilidad de extender o adaptar la herramienta a diferentes necesidades del usuario, es una característica deseable dado que sin ésta la aplicación tendría un carácter rígido que no se condice con la naturaleza del ámbito tecnológico. Esto también permite darle al usuario que lo desee una herramienta para la customización de su ambiente y herramientas de trabajo.

**Facilidad de Uso**

# Funcionalidades

A continuación se presenta dos listas de funcionalidades que tendrá el IDE propuesto. La primera lista consta de aquellas que no pueden faltar, dado que hacen a la esencia de una aplicación de este estilo. La segunda está compuesta por funcionalidades deseables, las cuales podrían llegar a incluirse como extensiones en el caso que se deseen.

**Requeridas**

**Deseables**

# Desventajas

# Conclusión

# Bibliografía

Carper, B. (Junio de 2010). *Emacs isn't for everyone*. Obtenido de briancarpet.net: http://briancarper.net/blog/534/emacs-isnt-for-everyone

Clojure. (Mayo de 2012). *Getting Started*. Obtenido de Clojure Documentation: http://dev.clojure.org/display/doc/Getting+Started

Graham, P. (Abril de 2001). *Beating the Averages*. Obtenido de Paul Graham: http://www.paulgraham.com/avg.html

Stanislav. (March de 2012). *Loper OS: Programmer's Editors, Illustrated.* Obtenido de Loper OS.