# Propuesta de Trabajo de Inserción Profesional

*Título:*
**Una implementación de Gobstones utilizando editores proyectivos**

*Alumno:*
Ariel Alvarez

*Director:*
Ing. Nicolás Passerini

*Codirector:*
Ing. Javier Fernández

*Carrera:*
Tecnicatura Universitaria en Programación Informática

# Propuesta de Trabajo de Inserción Profesional

11 de octubre de 2015

**Resumen**

El lenguaje Gobstones posee una secuencia didáctica bien definida que ha demostrado ser eficaz, pero el entorno de programación usado actualmente, aún cumpliendo las funciones de edición para las cuales fue creado, no refuerza activamente esta didáctica ni suple las necesidades de una comunidad creciente de usuarios. El presente trabajo busca desarrollar un *Entorno Integral de Aprendizaje* centrado en un *Editor Proyectivo*, haciendo uso de sus cualidades intrínsecas para facilitarle al alumno la comunicación de soluciones en términos de conceptos en lugar de trabajar sobre texto plano.

## 1. Introducción

En esta sección se detallan brevemente las características del lenguaje Gobstones y de los editores proyectivos. Luego se presenta un caso particular de ambiente de desarrollo de lenguajes basados en tecnología proyectiva, que será utilizado para implementar el entorno propuesto.

### 1.1. El lenguaje Gobstones

Gobstones[2] [TODO: hablar sobre gobstones]

### 1.2. Editores Proyectivos

El término Editor Proyectivo fue acuñado por Martin Fowler en el año 2005[4], al intentar plantear un ambiente de desarrollo donde el programador pudiera expresar sus ideas en términos de conceptos en lugar de texto. Lo que vemos como texto pasaría entonces a constituir una representación editable del concepto al que hace referencia (y al cual Fowler llama *representación abstracta*).

De esta manera, los conceptos del lenguaje son el dominio de los editores proyectivos, y decimos que un programa es una *representación abstracta* construida utilizando dichos conceptos. Para modificar esta representación el programador interactúa con una interfaz de usuario, llamada *representación editable*, sobre la cual la *representación abstracta* se *proyecta* en forma de texto[3].

A su vez, la *representación abstracta* puede persistirse de diferentes maneras a diferentes soportes, con lo cual se introduce la idea de *representación persistida* para hablar del formato en que guardará el programa, ya sea en una base de datos, un archivo binario, un texto con formato XML, etc.

Esto tiene varias consecuencias:

- Deja de necesitarse un parser para el lenguaje, volviéndolo más sencillo de extender.
- El programador trabaja más cerca de los conceptos que quiere expresar.
- Reduce drásticamente los posibles errores de sintaxis.

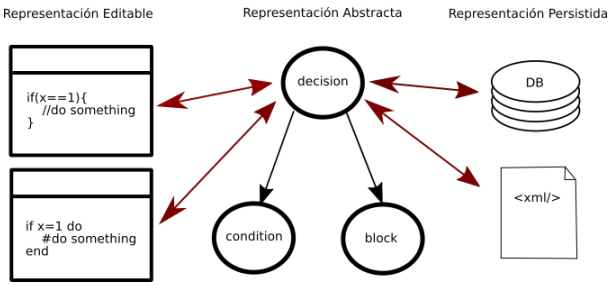


Figura 1: Representaciones de un programa en un editor proyectivo

- El editor trabaja directamente con las instancias de los conceptos, con lo cual:
  - es más sencillo analizar el programa
  - se simplifica la construcción de herramientas (ej: refactors, migrado de versiones de lenguaje, intentions, etc)
  - se mejora la performance del editor al eliminarse la etapa de parseo.

### 1.3. MPS Workbench

MPS (Meta programming System)[7] es un ambiente de trabajo orientado al desarrollo de lenguajes específicos de dominio, de la empresa JetBrains, basado en las ideas de Fowler; hoy en día se posiciona como uno de los referentes más importantes en editores proyectivos[1]. Entre los productos desarrollados utilizando MPS se encuentran: *MetaR* [8], un ambiente especializado en biomedicina computacional; *mbeddr* [6], un ambiente orientado a software embebido.

## 2. Motivación

En el libro *"Las bases conceptuales de la Programación. Una nueva forma de aprender a programar"*[2] se presenta una secuencia didáctica bien detallada junto con el entorno *PyGobstones*[5], pero notamos que en el marco de una adopción cada vez mayor de esta didáctica es necesario:

- disponer de un ambiente que sea fácilmente extensible y posea una arquitectura que agilice la colaboración de la comunidad de Gobstones.
- que el ambiente sea un refuerzo de la se secuencia didáctica, y no sólo una herramienta de edición.
- utilizar tecnologías que posean un buen soporte y comunidad.
- que el entorno sea sencillo de instalar

## 3. Propuesta

Se propone utilizar *MPS Workbench* para implementar un *Entorno Integral de Aprendizaje*, supliendo las necesidades antes mencionadas, siendo que:

- soporta tanto extensión de lenguaje como extensión del entorno mediante la creación de plugins que pueden no depender entre sí.
- los features de lenguaje pueden ser activados y desactivados durante el uso del entorno para cumplir con la secuencia didáctica.
- es soportado por JetBrains, uno de los líderes mundiales en desarrollo de entornos de programación.
- el entorno sólo necesita Java Runtime Environment (siendo que se trabajará sobre la implementación de un intérprete Gobstones, con lo cual se vuelve innecesario contar con un compilador).

A su vez, la naturaleza proyectiva del entorno le permite al alumno concentrarse en el concepto que desea expresar al evitarle errores comunes de sintaxis. Además, el programa siempre se presenta al alumno con un formato estándar (bien indentado, secciones organizadas, etc), ayudándolo a incorporar más rápidamente estas buenas prácticas al reconocerlas como algo natural. Esto es así porque al principio el alumno no posee una noción de estilos *correctos*, sino que se va formando a medida que es expuesto a ellos.

En particular, esta implementación busca:

- que el lenguaje refuerce la necesidad de buenos comentarios y la especificación del propósito y precondición de funciones y procedimientos

- el uso de estilos sutiles que faciliten el reconocimiento estructuras repetidas; por ejemplo, coloreando nodos similares al que se encuentre debajo del cursor. Esto se vuelve útil a la hora de enseñar determinados conceptos, como la parametrización de procedimientos.

Todo esto siguiendo la filosofía de Gobstones y buscando mantener las virtudes de la implementación original (identidad gráfica, multiplataforma, interfaz de usuario minimalista, etc).

# 4. Plan de trabajo

[TODO: transcribir los sprints planificados en waffle]

# Referencias


[1] Markus Voelter y col. «Language modularity with the MPS language workbench». En: *Software Engineering (ICSE), 2012 34th International Conference* (1905). DOI: 10.1109/ICSE.2012.6227070.

[2] Pablo E. Martínez López. *Las bases conceptuales de la Programación: Una nueva forma de aprender a programar*. La Plata, Buenos Aires, Argentina, 2013. ISBN: 978-987-33-4081-9. URL: http://www.gobstones.org/bibliografia/Libros/BasesConceptualesProg.pdf.

[3] Markus Voelter y col. «Towards User-Friendly Projectional Editors». En: *7th International Conference on Software Language Engineering (SLE)*. 2014.

[4] M. Fowler. *Language Workbenches: The Killer-App for Domain Specific Languages?* URL: http://martinfowler.com/articles/languageWorkbench.html.

[5] Gobstones. *Link de descarga a PyGobstones*. URL: http://www.gobstones.org/?page_id=30.

[6] itemis y fortiss. *mbeddr*. URL: http://mbeddr.com/.

[7] JetBrains. *MPW Workbench*. URL: https://www.jetbrains.com/mps/.

[8] Campagne Laboratory. *MetaR*. URL: http://metaR.campagnelab.org.