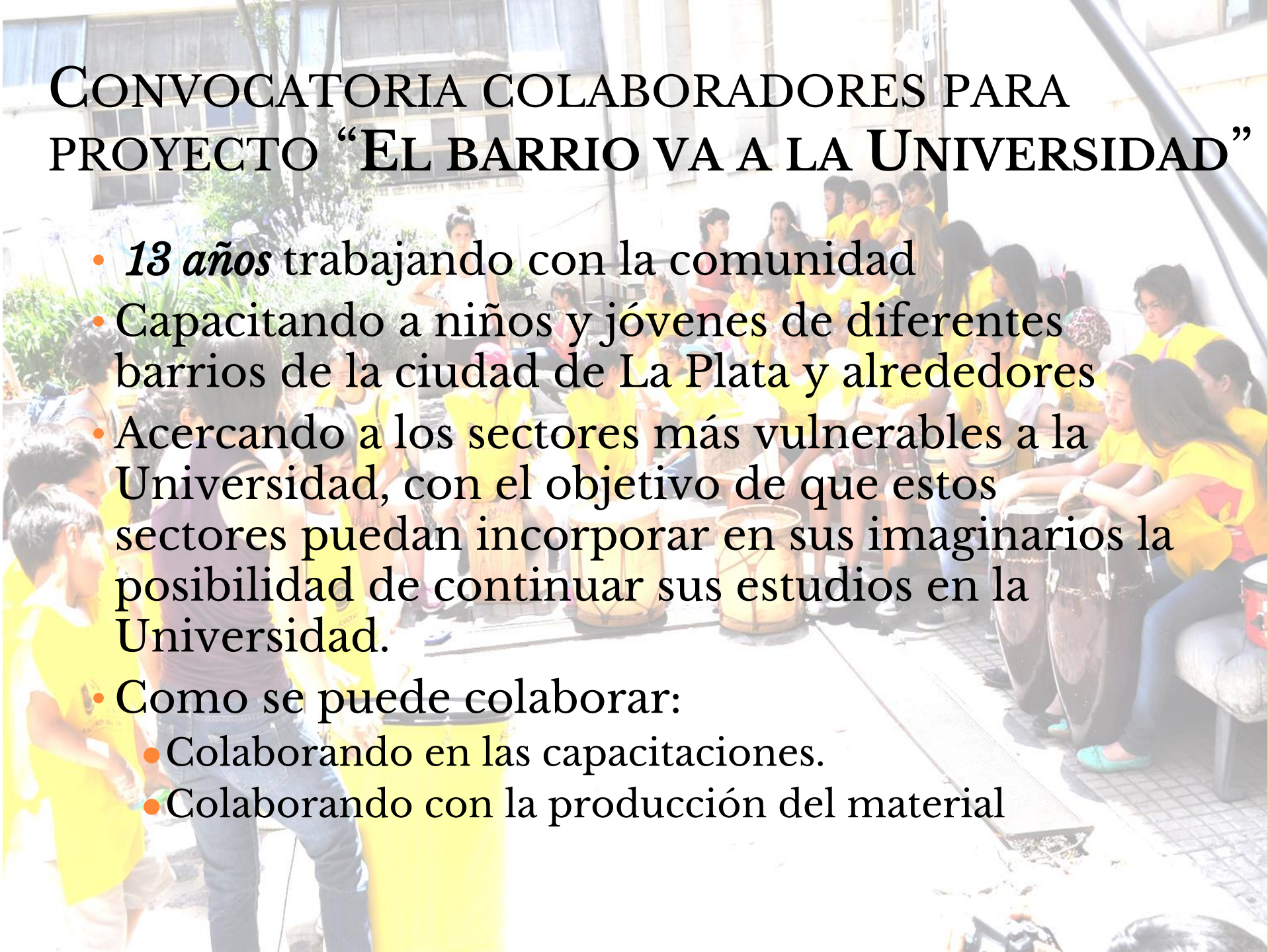


CONVOCATORIA COLABORADORES PARA PROYECTO “EL BARRIO VA A LA UNIVERSIDAD”

- **13 años** trabajando con la comunidad
- Capacitando a niños y jóvenes de diferentes barrios de la ciudad de La Plata y alrededores
- Acercando a los sectores más vulnerables a la Universidad, con el objetivo de que estos sectores puedan incorporar en sus imaginarios la posibilidad de continuar sus estudios en la Universidad.
- Como se puede colaborar:
 - Colaborando en las capacitaciones.
 - Colaborando con la producción del material





CONCEPTOS Y PARADIGMAS DE LENGUAJES DE PROGRAMACIÓN

INTRODUCCION Y EVALUACION DE
LENGUAJES



INTRODUCCIÓN

**Los lenguajes de Programación son el corazón de la Ciencia de la Infomática.
Son herramientas que usamos para comunicarnos con las máquinas y también con las personas.**



CUAL ES LA IDEA?

*“El valor de un lenguaje o de un concepto se debe juzgar según la forma en que **afecta la producción de Software** y a la facilidad con la que puede **integrarse** a otras herramientas”*

- Introducir, analizar y evaluar los conceptos más importantes de los lenguajes de programación.



QUÉ CONSEGUIREMOS

- Adquirir habilidad de **apreciar y evaluar** lenguajes, identificando los **conceptos** más importantes de cada uno de ellos y sus **límites y posibilidades**
- Habilidad para **elegir, para diseñar, implementar o utilizar** un lenguaje
- Enfatizar la **abstracción** como la mejor forma de manejar la complejidad de objetos y fenómenos



PARA QUÉ ESTUDIAR CONCEPTOS DE LENGUAJES

- Aumentar la capacidad para producir software.
- Mejorar el uso del lenguaje
- Elegir mejor un lenguaje
- Facilitar el aprendizaje de nuevos lenguajes
- Facilitar el diseño e implementación de lenguajes



CRITERIOS PARA EVALUAR LOS LENGUAJES DE PROGRAMACION

Para poder evaluar los lenguajes necesitamos establecer criterios de evaluación.



OBJETIVOS DE DISEÑO

- Simplicidad y legibilidad
- Claridad en los bindings
- Confiabilidad
- Soporte
- Abstracción
- Ortogonalidad
- Eficiencia

Analicemos algunas respuestas de docentes respecto al lenguaje que enseñan..



CRITERIOS PARA EVALUAR LOS LENGUAJES DE PROGRAMACIÓN

Textos obtenidos de trabajos finales de promoción

Pregunta a docentes de Cátedra de Lenguaje C:

Suele observarse que el código de C no es **legible**. Cree que esto se debe a la naturaleza del lenguaje o a malos hábitos de programación? ¿Cuál cree que sea la causa?

Rta 1:

*La sintaxis de C permite que pueda escribirse código muy compacto. Los alumnos deberían salir preparados para leer código de otros sin problemas. Muchos de los proyectos más grandes escritos en C tienen estrictas normas de estilo y esto permite que a pesar de ser compacto, el código sea suficientemente **expresivo** para todos.*

Rta 2:

A la naturaleza del lenguaje sobre todo. El permitir código conciso y.....

Pregunta a docentes de Cátedra lenguaje Ruby:

¿Qué opina de la sintaxis de Ruby?

Rta:

*La sintaxis de Ruby es, a mi parecer, **simple** y elegante. y por otro lado es muy fácil desarrollar con él porque es muy conciso y preciso a la hora de implementar cualquier clase de funcionalidad*

CRITERIOS PARA EVALUAR LOS LENGUAJES DE PROGRAMACIÓN

Pregunta a docentes de Cátedra lenguaje Ruby:

¿Usted considera que Ruby es un buen ejemplo de un lenguaje que presenta **Ortogonalidad**? ¿A qué se debe?

Rta:

A mi parecer sí. El simple hecho de que toda sentencia del lenguaje sea una expresión (incluso las estructuras de control) permite realizar combinaciones y/o composiciones sin ninguna clase de limitación. Otro factor que favorece la ortogonalidad es que casi todo en este lenguaje son objetos; no recuerdo fácilmente casos en los cuales haya tenido problemas por encontrarme con algo que no sea/se comporte como un objeto.

Pregunta a docentes de Cátedra lenguaje Python:

¿Piensa que el lenguaje Python es indicado para iniciar a los alumnos en la programación?

Rta:

*Si, Python es un lenguaje **simple** y fácil de enseñar. Es un lenguaje de tipado dinámico pero es fuertemente tipado, Es un lenguaje **ortogonal** porque se pueden combinar sus componentes. Su código **es legible**, **confiable** por ser fuertemente tipado y proveer manejo de excepciones,*

En las respuestas se pueden observar criterios de evaluación aplicados, relacionados con: **Simplicidad, Ortogonalidad, Confiabilidad, Expresividad, etc.**



SIMPLICIDAD Y LEGIBILIDAD

- Los lenguajes de programación deberían:
 - Poder producir programas fáciles de escribir y de leer.
 - Resultar fáciles a la hora de aprenderlo o enseñarlo
- Ejemplo de cuestiones que atentan contra esto:
 - Muchas componentes elementales
 - Conocer subconjuntos de componentes
 - El mismo concepto semántico – distinta sintaxis
 - Distintos conceptos semánticos - la misma notación sintáctica
 - Abuso de operadores sobrecargados



CLARIDAD EN LOS BINDINGS

- Los elementos de los lenguajes de programación pueden ligarse a sus atributos o propiedades en diferentes momentos:
 - Definición del lenguaje
 - Implementación del lenguaje
 - En escritura del programa
 - Compilación
 - Cargado del programa
 - En ejecución
- La ligadura en cualquier caso debe ser clara



CONFIABILIDAD

- La confiabilidad está relacionada con la seguridad
 - Chequeo de tipos
 - Cuanto antes se encuentren errores menos costoso resulta realizar los arreglos que se requieran.
 - Manejo de excepciones
 - La habilidad para interceptar errores en tiempo de ejecución, tomar medidas correctivas y continuar.



SOPORTE

- Debería ser accesible para cualquiera que quiera usarlo o instalarlo
 - Lo ideal sería que su compilador o intérprete sea de dominio público
- Debería poder ser implementado en diferentes plataformas
- Deberían existir diferentes medios para poder familiarizarse con el lenguaje: tutoriales, cursos textos, etc.



ABSTRACCIÓN

- Capacidad de definir y usar estructuras u operaciones complicadas de manera que sea posible ignorar muchos de los detalles.
 - Abstracción de procesos y de datos



ORTOGONALIDAD

- Significa que un conjunto pequeño de constructores primitivos, puede ser combinado en número relativamente pequeño a la hora de construir estructuras de control y datos. Cada combinación es legal y con sentido.
 - **El usuario comprende mejor si tiene un pequeño número de primitivas y un conjunto consistente de reglas de combinación.**

Pascal y ADA **NO** son ortogonales, Rubyl **SI**

- En **Pascal**, por ejemplo: los proc y las funciones pueden ser pasadas por parámetro pero solo pueden pasar parámetros por valor, las funciones SOLO pueden devolver datos elementales, NO punteros, ni arreglos, etc.
- En **ADA**, por ejemplo: los parámetros por valor pasados solo pueden ser elementales, por referencia: arreglos, etc.

EFICIENCIA

- Tiempo y Espacio
- Esfuerzo humano
- Optimizable

