A History of Haskell: Being Lazy With Class

- Durante finales de los años 70 se había dado un avance en los lenguajes de programación funcionales (como lo fue *Lisp*) tomándolos más como una herramienta práctica de programación que sólo una curiosidad matemática. Tomando consideraciones importantes, como lo fueron basarse en el cálculo lambda para modelar dichos lenguajes y añadiendo la casa de patrones.
  Posteriormente a principios de los años 80's se comenzó a analizar el término de evaluación perezosa (llamada por necesidad), el cuál fue definido en tres distintas versiones por:
  - Dan Friedman y David Wise
  - Peter Henderson y James H. Morris Jr
  - David Turner
- Al mismo tiempo se realizaban grandes maneras de implementar éstos lenguajes perezosos:
  1. Utilizando reducción gráfica mediante el uso de combinadores SK.
  2. Y el importante uso de éstas técnicas usadas en arquitecturas distintas a la *Von Neumann*.

  A lo largo de los años 80's se celebraron distintas conferencias dónde se hablaba de la importancia de éste término y su aplicación en los lenguajes ya existentes de la época. Contando con numerosos artículos de personajes reconocidos en el campo, como *Darlington*, *Turner*, *Dijsktra*, aumentando la euforia por desarrollar más el tema y nutriéndolo poco a poco.
- Como resultado de toda la actividad realizada, en la mitad de los 80's ya había bastantes investigadores en el campo incluyendo los que ya trabajaban ahí, y se encaminaron a realizar distintos artículos conforme a sus investigaciones, contribuyendo así, al primitivo desarrollo de los lenguajes perezosos, un proceso que iba lento, pero con avances increíbles, no visibles para esa época.
  - Se promueve la modificación de Miranda, para añadir un fuerte polimorfismo y type inference.
  - Sale a la luz la Lazy ML, la cuál tuvo una gran influencia para el desarrollo de la *G-machine*, la cual podía compilar aplicaciones de lenguajes perezosos funcionales.
  - Orwell es creado, por Wadler, utilizando una notación más matemática.
  - Alfl es creado, por Hudak, el cual tenía un intérprete basado en combinadores.
  - Clean, un lenguaje perezoso basado explícitamente en la reducción gráfica.
  - Entre otros...

  Con ésto surgieron grandes ideas sobre el desarrollo, pero también la especulación de cuál lenguaje era superior a otro, y una falta de estandarización , pues cada uno tenía su propia definición, en comparación con *Scheme* o *ML* los cuales ya tenían sus propios estándares.

- Para el año 1987 en la ***Functional Programming and Computer Architechture Conference (FPCA)*** coincidieron Peyton Jones y Hudak, marcando el inicio del diseño de Haskell y aunque no tenían el nombre ni una manera para diseñarlo, concordaron que debía comenzar con un lenguaje existente y evolucionarlo en la dirección que ellos querían. Y el más maduro, y por tanto, elegido fue Miranda desarrollado por David Turner, pues era puro, bien diseñado y una implementación robusta. Sin embargo, al tener una reunión con Turner, declinó aceptar que su lenguaje fuése utilizado, ambas partes tenían espectativas diferentes. Jones y Hudak, querían un lenguaje que pudiése ser utilizado para otras características, en particular, la posibilidad de extender y modificar el lenguaje, para crear y distribuir una implementación. Por otra parte, Turner quería mantener un estándar de su lenguaje, sin tener más de un dialecto para Miranda. Y además declinó la invitación a se parte del comité.
  Ésto marcó una nueva ruta, dónde había que desarrollar un lenguaje desde cero, y aún con ésto, Haskell posee muchas ideas inspiradas en Miranda, pues fue un buen modelo a seguir.
  Además comenzó una discusión sobre el tema en el correo ***fplangc@cs.ucl.ac.uk*** donde nació el ***FPLang Committee***.

- Después de ésto, se comenzaron a realizar varias reuniones donde comenzó el proceso del diseño e implementación, la primera fue en Yale (Enero 9 - 12, 1988) dónde se establecieron las metas a seguir para el lenguaje, las cuales fueron:

  1. Debía ser adecuado para enseñar, investigar y aplicaciones (incluyendo construir largos sistemas)
  2. Debía ser completamante descrito en la publicación para una sintáxis y semántica formal.
  3. Debía ser libremente disponible
  4. Debía ser usable como base para futuras investigaciones
  5. Debía ser basado en ideas que disfrutaran un amplio consenso.
  6. Debía reducir diversidades inecesarias en los lenguajes de programación funcionales.

  Con las dos últimas metas se prentendía que fuese un lenguaje un tanto conservativo, en vez de uno radicalmente distinto.
  Así mismo algunas de éstas metas no fueron realizadas.
  Y fue entonces donde se comenzó el acuerdo para trabajar en el proyecto, con los siguientes acuerdos:

  1. Decidir temas para discutir y assignar un *líder* a cada tema.
  2. Los líderes comienzan la discusión por medio de los conflictos para el tema.
  3. Se alentaría tomar descansos, discusiones laterales y literatura si fuese necesario.
  4. Algunos conflictos no serán resueltos. Pero se darían acciones para su eventual resolución.
  5. No se comenzaría hasta que no se tuviera un nombre para el lenguaje (Aunque pareciese tonto).
  6. La actitud sería importante: un espiritu de cooperación y compromiso.

- El hecho de decidir un nombre para el lenguaje fue una cosa de suma importancia. Entre los nombres propuestos resaltan: *Semla, Haskell, Vivaldi, Mozart, CFL(Common Functional Language), Funl 88, Semlor , Candle,(Common Applicative Notation for Denoting Lambda Expressions), Fun, David, ...*
  Después de una larga discusión quedó sólo un nombre, *"Curry"*, en honor a un lógico-matemático llamado Haskell B. Curry. Sin embargo, también existía una personalidad llamada Tim Curry, una máquina abstracta de Jon Fairbarin y Tim Curry era famoso por jugar de líder en Rocky Horror Picture Show; lo que hizo que un día después, el nombre cambiara a *"Haskell"* para evitar confusiones. Así se solicitó el permiso a la viuda de Curry para adoptar el nombre de su marido dentro del proyecto, a lo cuál accedió y añadió, *"A Haskell nunca le gustó su nombre"*.

- Durante **The Glasgow Meeting, Abril 6-9, 1988** se tomaron varias decisiones clave en el diseño e implementación del ya nombrado Haskell. Y quedando Hudak y Wadler como los editores del primer reporte de Haskell, *Report on the Programming Language Haskell, A Non-strict, Purely Functional Language*. Y durante ése tiempo comenzó el desarrollo de Haskell, ya con todo en orden y con metas por realizar.

  - **1/Abril/1990** Se reporta Haskell 1.0
  - **Agosto 1991** Se libera Haskell 1.1 dónde se incluyeron las expresiones *let*.
  - **Marzo 1992** Se libera Haskell 1.2 incluyendo cambios menores.
  - **1994** Haskell cobra presencia en Internet al subir el dominio haskell.org
  - **Mayo 1996** Se libera Haskell 1.3 y fue el release más importante desde Haskell 1.0
    - Se incluyó un reporte de bibliotecas.
    - Aparece el I/O con mónadas, incluyendo "do" sintáxis.
    - Se introducen los constructores de clase.
    - Se extienden los tipos de datos algebráicos.
  - **Abril 1997** Se libera Haskell 1.4 donde se generalizaron las listas de comprensión en mónadas arbitrarias.
  - **Febrero 1997** ***The Haskell 98 Report: Language and Libraries*** fue lanzado.

- Una curiosidad es que *Haskell Report* fue publicado el 1 de Abril, 1990, coincidiendo con el *April Fool's Day (Día de los inocentes)* y, evidentemente, Haskell no era una broma aquel entonces, sin embargo, ésta situación se prestó para distintas bromas relacionadas con la fecha. Las cuales fueron incluidas dentro del mismo sitio de Haskell, *haskell.org/humor* y entre ellas podemos encontrar las siguientes:

  1. El 1 de Abril de 1993, Will Partain escribió un increíble anuncio acerca de una extensión de Haskell llamada *Haskerl*, donde se combinaba lo mejor de los lenguajes Haskell y Perl, un documento tan detallado y profesional que es bastante creíble.
  2. También se publicó la siguiente frase alusiva:
     "Recientemente Haskell ha sido utilizado en un experimento en la escuela de medicina de Yale. Fue utilizado para reemplazar un programa desarrollado

en C el cuál controlaba una máquina que ayudaba a los pulmones y corazón. En los seis meses que estuvo en operación, el hospital estimó que al menos una docena de vidas fueron salvadas porque el programa resultó más robusto que el desarrollado en C, el cuál solía crashear y mataba a los pacientes."

3. Equivalentemente:
   "Recientemente, un hospital local sufrió varias negligencias debido a una falla en el software utilizado en la máquina de rayos X. Así que decidieron reescribir dicho software en Haskell para más confiabilidad. Ahora las negligencias se han reducido a cero. La razón es que aún no han tomado ningun rayo X nuevo (Seguimos compilando el Standard Prelude)."
4. El 1 de Abril de 1998, John Peterson escribió un artículo donde anunciaba que debido a la demanda de Sun Microsystems sobre Microsoft por el uso de Java, Microsoft había adoptado Haskell como su princpial lenguaje de desarrollo.