# ¿Cuan ágil es el seguimiento que realizas?

Christian Nahuel Balsamo, *Alumno de la Universidad de Palermo*

***Abstract*—En una época en la que la inmensa mayoría de los proyectos de desarrollo se realizan utilizando metodologías ágiles, existe un pequeño pero importante espacio, el cual pocos se toman el trabajo de revisar, que es, la adecuada utilización de las herramientas de seguimiento.**

**No basta solo con elegir la herramienta más adecuada al negocio, o la más popular, sino que se debe realizar y describir un proceso de utilización de la misma que acompañe al desarrollo, que aporte un cierto valor al proyecto, sin que esto consiga volverlo menos ágil.**

**Este documento brinda una serie de recomendaciones para sacar el mejor provecho de este tipo de herramientas en ambientes ágiles.**

***Palabras claves*--Ágil, Desarrollo, Jira, Metodologías, Recomendaciones, Scrum, Seguimiento.**

## I. Nomenclatura

*Scrum*: Metodología ágil comúnmente utilizada para el desarrollo de software.
*Jira*: Herramienta de seguimiento de errores y de proyectos desarrollada por Atlassian.
*Confluence*: Herramienta de documentación (estilo wiki) desarrollada por Atlassian.
*Tracking*: Seguimiento de tareas, errores, defectos y nuevas funcionalidades.
*Tickets*: Unidad de trabajo en las herramientas de seguimiento de errores.
*Points*: Unidad de medida del esfuerzo necesario para realizar la tarea en comparación al esfuerzo necesario para llevar a cabo tareas catalogadas.
*Daily Scrum / Stand-Up Meeting*: Reunión diaria utilizada en la metodología Scrum para conocer en que se encuentra trabajando cada integrante del equipo y que impedimentos tiene para realizar su trabajo.
*Story / User Story*: Texto que describe la necesidad de la realización de cierta funcionalidad en el proyecto en el que se trabaja.
*Task*: Unidad de trabajo a realizar.
*Backlog / Product Backlog / Sprint Backlog*: Paquetes de funcionalidades / Users Stories que se necesitan desarrollar para completar el producto en general o en una iteración dada.
*Sprint / Iteración*: Ventana de tiempo definida en Scrum en la cual el equipo se compromete a terminar una cierta cantidad de tareas.
*Cards*: Tajetas, posts-it o papeles en donde se escriben las User Stories y que se colocan en tableros encolumnados para representar el estado de cada tarea a través del proceso de desarrollo del mismo.

## II. Introducción

UNA inmensa cantidad de proyectos dicen de estar trabajando con metodologías ágiles. La mayoría de ellos asume que con solo utilizar Scrum uno ya esta desarrollando de manera ágil. Pero esto no es realmente cierto.

Existen varios principios dentro de los cuales se basan las metodologías ágiles que suelen generar conflicto en el desarrollo de proyectos de software muy a menudo. En particular uno de ellos y en el cual se basa este paper es el siguiente: **"El software que funciona es la principal medida del progreso".**

Desde mi punto de vista y mi experiencia sobre los diversos proyectos en los que he estado puedo asegurar que en muchas ocasiones existe un excesivo uso de las herramientas de seguimiento. Muchas veces por ejemplo, el "Jira" del proyecto se encuentra mucho mas completo y detallado que el propio software que se esta realizando.

## III. Casos de estudio

En este paper se proponen 3 escenarios reales sobre los cuales se intentaran dar ciertas recomendaciones para poder distinguir cuando se esta siendo ágil en el seguimiento y cuales son las cosas que se deben evitar.

### *A. Escenario 1*

En una conocida empresa multinacional extranjera, donde tuve la posibilidad de trabajar por algunos meses en un pequeño equipo de desarrollo que contaba con 4 personas más un líder técnico en la locación donde yo me encontraba, se utilizaba Scrum como metodología de trabajo y todo el seguimiento de errores y nuevos requerimientos eran incluidos en la herramienta Jira.

Las tareas no solían ser demasiado grandes ni necesariamente tenían algo en común con las demás, y los productos principales de la empresa eran sus sitios webs desde donde se promocionaban los programas de televisión que manejaba la empresa.

En un Sprint que duraba 2 semanas, se realizaba, al inicio, una reunión de planeamiento en donde todos los jefes de las áreas de marketing y producción en conjunto con el equipo de desarrollo repasaban todas los tickets cargados en el backlog para conocer el alcance de cada tarea y darles una prioridad dentro del mismo backlog.

Una vez concluida la reunión, el equipo de desarrollo, conociendo su capacidad de trabajo, decidía y luego

comentaba a los interesados cuales eran las tareas que iban a integrar el sprint backlog.

En general, todas las funcionalidades necesarias y los pedidos se cargaban en Jira y luego se repartían a medida que los desarrolladores iban terminando sus tareas. Prácticamente toda la comunicación se realizaba a través de Jira, incluso la comunicación por mail era copiada y pegada en los comentarios de cada ticket para dejar sentado todas las definiciones posibles que se hayan pensado.

Diariamente se realizaba la Stand-Up Meeting donde se repasaban todas las tareas incluidas en el Sprint backlog y el estado de cada una, sin importar si alguien de desarrollo estaba trabajando en ellas, ya que muchas veces las tareas dependían de la entrega de material externo para realizarse, o algunas otras tareas podían quedar bloqueadas, con lo cual diariamente se confirmaba que tarea iba a realizar cada persona en el día.

Cada vez que se finalizaba el desarrollo de alguna tarea, se notificaba en Jira la cantidad de horas que demandó el desarrollo de la misma.

### *B. Escenario 2*

En otra conocida empresa multinacional de origen nacional, donde participe del equipo de desarrollo por 4 años, estaba integrada por más de 200 personas entre lideres, managers, desarrolladores y testers.

También utilizaban Scrum como metodología de desarrollo y el seguimiento de errores y nuevos requerimientos eran volcados en las herramientas de Atlassian (Jira, Confluence, etc).

Cada nuevo proyecto se creaba en Jira, se armaba el backlog correspondiente incluyendo las tareas de mayor nivel y por cada iteración se iban desmenuzando esas tareas en Stories más pequeñas que se priorizaban en las reuniones de planeamiento y se les asignaban sus puntos para medir la velocidad el equipo.

Los sprints solían durar 3 semanas incluyendo una reunión de planeamiento, las reuniones diarias y una reunión de retrospectiva al final de cada iteración.

En general, la comunicación y el intercambio de información relativa a las tareas se solía realizar a través de mails, entre el equipo y con el equipo de Marketing quienes eran los encargados de realizar los pedidos y evaluar las funcionalidades requeridas una vez finalizado el proyecto o a medida que se iban completando las tareas de mayor nivel.

Al ser el cliente interno, el control de avance del proyecto se realizaba diariamente ya que existía una comunicación verbal con el equipo de marketing muy fluida. Esto también llevaba a que no exista la necesidad de cargar las horas trabajadas.

### *C. Escenario 3*

En otra consultora en la que tuve la oportunidad de trabajar por más de 1 año, también se utilizaba Scrum como metodología de trabajo. En este caso, la consultora proporcionaba un equipo de trabajo para un cliente externo extranjero.

Tanto el equipo local de desarrollo como el equipo del cliente eran de más de 200 personas entre los cuales se encontraban desarrolladores, testers y lideres técnicos del lado de la consultora, y un equipo similar incluyendo también a los managers, arquitectura y dueños de procesos del lado del cliente.

Diariamente se realizaban las reuniones de scrum para informar el avance con el equipo local de desarrollo pero muy pocas veces se contaba con la presencia del manager del lado del cliente. Luego de esta reunión diaria, el líder técnico armaba un mail resumiendo la reunión y era enviado al manager.

Al ser un único y gran proyecto, en Jira lo que se cargaban eran los módulos de nuevas funcionalidades con una estructura jerárquica basada en el tamaño de las tareas. Primero se creaba una gran tarea que abarcaba una funcionalidad completa. Luego se creaban varias tareas de nivel inferior para dividir las partes en las que se iban a trabajar para completar la funcionalidad.

Cada pequeña tarea del último nivel de la jerarquía era una tarea que podía ser estimada y realizada por el equipo y sobre la cual se trabajaba. Estas tareas tenían así mismo, un set de sub-tareas que especificaban cada paso del desarrollo que se debía realizar o revisar las cuales eran compartidas por todas las tareas similares de último nivel jerárquico.

A su vez, cada funcionalidad tenia una serie de entradas en Confluence. Estas entradas podían ser donde se definían los bocetos y la interacción necesaria para la tarea, entradas donde se especificaban las guías de diseño o varias entradas donde se especificaban los casos de testeo.

Existía también una gran serie de procesos de validación para empezar, analizar y completar cada una de las tareas, que iban desde la revisión de los criterios de testeo, la revisión del seguimiento de los estándares para el código involucrado en el desarrollo de la tarea y por ultimo procesos de verificación y aceptación del funcionamiento por parte de los dueños de los módulos.

Las tareas en general eran medidas con puntos equivalentes a la cantidad de esfuerzo necesario para realizarse, pero estos puntos venían otorgados o validados por los propios dueños de los módulos, ya que estos estaban directamente relacionados con el presupuesto que tenían asignados para utilizar.

Al ser una consultora externa al cliente, existía además una herramienta propietaria externa a Jira donde se realizaba el seguimiento de la cantidad de horas dedicadas al trabajo que luego eran facturadas directamente al cliente.

## IV. MARCO TEÓRICO

En el marco teórico de esta investigación incluimos algunas definiciones para poder comprender los temas de los que vamos a estar hablando.

### *A. Metodologías Ágiles*

Se conoce como metodologías agiles a aquellos métodos de llevar a cabo procesos que se diferencian de los métodos formales que son considerados excesivamente “pesados” y rígidos por su carácter normativo y su fuerte dependencia de planificaciones detalladas previas al desarrollo. Algunos de sus valores más importantes definidos son los que se mencionan a continuación.

*1) Valorar más a los individuos y su interacción que a los procesos y las herramientas.*

Los procesos son una guía de operación. Los procesos se

vuelven peligrosos cuando los trabajos necesitan creatividad e innovación.

*2) Valorar más el software que funciona que la documentación exhaustiva.*

Los prototipos ofrecen una retro-alimentación que genera ideas imposibles de concebir en un primer momento. La documentación permite la transferencia del conocimiento pero aportan menos valor al producto.

*3) Valorar más la colaboración con el cliente que la negociación contractual.*

Las prácticas ágiles son indicadas para productos difíciles de definir con detalle en el principio. Un contrato no aporta valor al producto. El cliente es un miembro más del equipo.

*4) Valorar más la respuesta al cambio que el seguimiento de un plan.*

Es más valiosa la capacidad de respuesta que la de seguimiento. La anticipación y la adaptación. [1]

### *B. Scrum*

Scrum es una de las metodologías ágiles más utilizadas en el ambiente del desarrollo de software. Es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental.

El modelo de referencia de Scrum define un conjunto de practicas y roles que pueden tomarse como punto de partida para definir el proceso de desarrollo de un proyecto. Dentro de los roles que define scrum, se encuentra el ScrumMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (interesados externos o internos), y el Team que incluye a los desarrolladores.

Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar.

Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.

Existen varias implementaciones de sistemas para gestionar el proceso de Scrum, que van desde notas amarillas "post-it" y pizarras hasta paquetes de software. Una de las mayores ventajas de Scrum es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar.[2]

### *C. Jira*

Uno de las herramientas de software paga más comúnmente utilizadas en los desarrollos de software para llevar un seguimiento de las tareas realizadas por el equipo o para medir el avance del proyecto es Jira.

Jira fue concebida como una aplicación web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos. También fue pensada como apoyo para la gestión de requisitos, seguimiento del estado del proyecto y seguimiento de errores.[3][4]

### *D. Bugzilla*

Es otra herramienta basada en Web de seguimiento de errores originalmente desarrollada y usada por el proyecto Mozilla. Lanzada como software de código abierto por Netscape en 1998. Bugzilla ha sido adoptada por una variedad de organizaciones para su empleo en el seguimiento de defectos, tanto para software libre como para software propietario.

Al igual que Jira, permite organizar en múltiples formas los defectos de software, permitiendo el seguimiento de múltiples productos con diferentes versiones, a su vez compuestos de múltiples componentes. Permite además categorizar los defectos de software de acuerdo a su prioridad y severidad, así como asignarles versiones para su solución.[5][6]

### *E. TrackStudio*

Otra herramienta paga para el seguimiento de errores y usada por varias empresas es TrackStudio. En comparación con Jira, TrackStudio no esta limitado solo al management de las tareas, sino también a los archivos y los documentos. Estos pueden ser organizados como árboles jerárquicos dentro de la herramienta, otorgar ciertos permisos de acceso a los usuarios deseados o filtrar la lista de documentos de alguna persona.

Otra de las diferencias claves con Jira es el precio del producto y los términos de utilización del mismo, el cual es ciertamente menor en costos que la herramienta de Atlassian.[7]

### *F. Usando jira en ambientes ágiles*

Los mismos desarrolladores de Jira y Confluence (Wiki) se tomaron el trabajo de conocer e investigar el uso que se les estaba dando a sus propias herramientas en el proceso de desarrollo ágil. De esta investigación obtuvieron el balance correcto en el uso de las herramientas dadas las diversas formas que tenían los equipos de utilizarlas, y finalmente llegaron a la definición de un proceso que funcionaba realmente bien para ellos.

Para este proceso, se definieron ciertas reglas de utilización de Jira y Confluence para que el desarrollo se lleve a cabo de la manera más ágil posible.

*1) Regla 1: Todo va en Jira*

Jira es la fuente de la verdad sobre lo que se realizo en el Proyecto y que es lo que queda por hacer. Cuando se arman las notas sobre lo lanzado en cada nueva versión del proyecto se consulta Jira para saber cuales cambios integran esa versión. Cuando un usuario se queja por un bug o una funcionalidad faltante, Jira es en el único lugar donde se puede consultar por mas información.

Los votos, los seguidores y los comentarios de los tickets son grandes indicadores de la prioridad de cualquiera de las tareas de desarrollo y se les presta atención.

Se utiliza también Jira para programar los lanzamientos de las versiones con los arreglos de bugs. Todos los tickets que son candidatos a ser incluidos en la próxima versión se los

incluye dentro del paquete para la próxima versión y son realizados en orden de prioridad, impacto, facilidad de resolución, etc. Aquellos tickets que no llegan a incluirse en la versión al momento del lanzamiento, son pasados a la siguiente versión.

*2) Regla 2: El planeamiento va en Confluence*

Las funcionalidades planeadas o potencialmente para planear tienen una pagina propia en Confluence. Utilizan el termino de "user stories" para estas páginas, pero generalmente son mucho mas completas que una "User Story" de Scrum.

Cada página describe una funcionalidad entera. Se empiezan bastante simples como una pequeña descripción similar a una "User Story" de Scrum, con algunos links a tickets/bugs en Jira o alguna otra referencia previamente existente. Luego, a medida que surgen más ideas sobre la funcionalidad son agregadas a la página y luego discutidas en los comentarios lo cual permite tener grabadas todas las conversaciones y el intercambio de ideas que se tuvieron al respecto de la funcionalidad tiempo atrás.

Tener todo documentado en Jira y Confluence significa que cualquiera puede encontrar una "funcionalidad" y plasmar sus ideas en ella en cualquier momento y lugar, sin importar si esa funcionalidad va a ser llevada a cabo en el momento más próximo.

Cuando se realiza una reunión de planeamiento en cada nueva iteración se arma una página en Confluence con los detalles administrativos y los links a aquellas tareas de Jira que se proponen llevar a cabo.

También Confluence es un perfecto lugar para ubicar el estado del proyecto adjuntando los gráficos sobre el avance en la propia página.

*3) Regla 3: Las stories van en "cartas"*

Una vez definidas las tareas que se incluirán en la iteración, se escriben esas stories en Cartas, se estiman, se ubican en la iteración y se las pega en la pared. Una Story de una página en Confluence puede ser transformada en 5 o 10 diferentes cartas de diferentes estimaciones, lo cual brinda más flexibilidad para incluir o quitar cosas del alcance a medida que el planeamiento avanza.

La ventaja de las cartas es su inmediatez y su presencia física lo que las convierte en muy buenas representaciones sobre lo que se esta trabajando actualmente.

*4) Regla 4: Las pruebas van en archivos CVS*

Todas los casos de prueba que se desean verificar con las pruebas automatizadas están en archivos CVS. Estos archivos son las fuentes de entrada con los parámetros con los que se alimentan las pruebas.[8]

## V. Desarrollo

Cada escenario presenta diversas situaciones y diversas cosas a favor o en contra. Se intentará dar recomendaciones sobre las cosas negativas utilizando como base los conocimientos referidos en el marco teórico y luego sobre estas recomendaciones se armará una guía general de utilización de las herramientas de desarrollo que favorezca al desarrollo ágil de los proyectos.

### *A. Escenario 1*

*1) Pros*

- Se reducen los malos entendidos al incluir toda la información relevante a la tarea en la herramienta de seguimiento.
- Los gerentes tienen una clara idea del esfuerzo realizado por el equipo al estar detallado la cantidad de horas dedicadas a cada tarea. Se puede medir el desempeño y la capacidad de trabajo de cada individuo.

*2) Contras*

- La única fuente de información válida es Jira. No se encuentra separada la información referida al ticket de la información de la funcionalidad y esto puede traer confusiones.
- Existe poca interacción entre personas cara a cara. Se prefiere que la gente este atenta a los tickets y no tanto a los mails ni a las reuniones.

*3) Recomendaciones*

- Seria conveniente separar la información útil para realizar las funcionalidades en una herramienta como Confluence o una wiki. Con esto Jira quedaría mucho mas prolijo para identificar las tareas que deben ser realizadas en el sprint, el estado actual y la carga horaria que llevan las mismas.
- Respetar e incentivar las reuniones que propone Scrum o la metodología de desarrollo utilizada, para ordenar y revisar las tareas.

### *B. Escenario 2*

*1) Pros*

- Se valoran más las relaciones humanas. Se respetan las reuniones propuestas y se tiene una correcta interacción entre las partes involucradas en el proyecto.
- No se dedica tiempo a las tareas que no aportan ningún o poco valor al producto. Se prioriza el desarrollo del producto y el avance del proyecto por sobre todas las cosas.

*2) Contras*

- Jira no refleja el estado actual del proyecto. No se mantiene correctamente actualizado u ordenado para facilitar la planificación de las tareas.
- Nadie confía en la información de Jira. Por lo mencionado anteriormente, a veces se realizan varias verificaciones sobre la misma tarea o se comienza a trabajar duplicando el esfuerzo por no tener la información correcta sobre el estado de la tarea.
- Jira es utilizado únicamente como tablero digital para la organización del equipo. No se esta aprovechando el potencial de la herramienta de seguimiento. Seria fácilmente reemplazable con la funcionalidad que brindan los post-its.

*3) Recomendaciones*

- Dedicar un tiempo y esfuerzo para organizar las tareas en Jira. Solamente incluir en estas la información

necesaria para poder tener una visión clara sobre las tareas que se deben realizar, en que iteración y actualizar el estado de las mismas.

- Separar e incluir en Confluence o una herramienta tipo wiki la información necesaria para realizar la tarea o funcionalidad, donde también se incluyan los comentarios y los cambios o decisiones tomadas sobre cada tema.

### C. Escenario 3

#### 1) Pros

- Conocimiento exacto de las tareas de desarrollo. Esto se debe a su detallado nivel de división de tareas y su complejo sistema de estados en los que pasa cada tarea incluida en Jira.
- Se divide correctamente la información referente a la tarea de la información de la funcionalidad. Esto se logra con las entradas que se manejan en Confluence.

#### 2) Contras

- Excesiva carga de información en Jira que resta mucho tiempo al desarrollo de la tarea en sí. El gran nivel de detalles y de separación de tareas conlleva a un continuo acceso a la herramienta para reflejar el estado de cada pequeña parte involucrada, y el burocrático proceso de verificación de cada acción realizada restan una importantísima cantidad de tiempo para realizar la tarea en sí.
- El estado oficial de la tarea es el que dice Jira (Se basan únicamente en esto para medir el avance y el cumplimiento de las tareas). A veces las herramientas no pueden expresar los motivos por los cuales se encuentran bloqueadas o atrasadas las tareas, o los motivos por los cuales no fueron aprobadas en los procesos de verificación, dando una mala impresión sobre el avance y el esfuerzo realizado por el equipo.

#### 3) Recomendaciones

- Incluir solo lo necesario en Jira para poder conocer el estado de las tareas en general. No agrega valor y resta una gran cantidad de tiempo y esfuerzo que podría ser dedicado al propio desarrollo de las tareas.
- Promover el mejor aprovechamiento de las reuniones adecuadas e incentivar la inclusión de las personas involucradas para conocer realmente el estado del proyecto y mejorar las relaciones entre las personas.

## VI. Conclusión

En general notamos que para que un proyecto pueda afirmar que esta siendo en verdad ágil no alcanza con seleccionar una metodología y conseguir las herramientas adecuadas, si en verdad no se utilizan de la mejor forma posible. Innumerables proyectos pierden la oportunidad de ser lanzados en el momento indicado o permanecen en un estado cíclico siendo corregidos una y otra vez sobre la misma base del error por la falta de gestión precisa y útil de su seguimiento.

Las siguientes recomendaciones son una recopilación de buenas prácticas observadas y evaluadas en diversos proyectos con muchos errores en común.

- No olvidarse de la importancia y las bases de las metodologías ágiles que incentivan que el desarrollo tiene que ser ágil para que el proyecto también lo sea.
- Entender el propósito de cada reunión y explotar su potencial para las funciones asignadas, y no realizarlas por el mero hecho de justificar la elección de la metodología.
- Realizar un planeamiento previo sobre como se desea utilizar las herramientas con su justificación basada en las recomendaciones de uso.
- Las herramientas de seguimiento deben contener información precisa y concreta para conocer el estado del proyecto, las necesidades funcionales de la aplicación, los errores encontrados y para ordenar los lanzamientos de las diferentes versiones.
- La información del seguimiento debe ser accesible y de conocimiento de todo el equipo, y siempre debe reflejar lo más posible la realidad del avance.
- Tratar de no sobre cargar de información a las herramientas de seguimiento para poder darle el mayor tiempo posible al equipo de realizar las tareas que realmente son importantes.
- El planeamiento, las discusiones sobre las funcionalidades y toda la documentación funcional del proyecto debe estar cargado en las herramientas de información colectiva (Confluence, Wiki, etc.)
- El planeamiento, las tareas actuales del equipo y las asignaciones de las mismas deben estar plasmadas con herramientas físicas y prácticas de fácil visibilidad y manejo que sean útiles para el equipo como son los post-its o los tableros de Scrum.

## VII. Líneas futuras de investigación

A. Implementaciones personalizadas para los distintos tipos de proyectos.

B. Investigación y desarrollo de herramientas integrales de las diferentes necesidades para el seguimiento de los proyectos.

## VIII. Referencias


[1] Manifiesto Ágil, http://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil
[2] Scrum, http://es.wikipedia.org/wiki/Scrum
[3] Jira, http://es.wikipedia.org/wiki/JIRA
[4] Oficial de Jira, http://www.atlassian.com/software/jira/
[5] Bugzilla, http://es.wikipedia.org/wiki/Bugzilla
[6] Sitio Oficial de Bugzilla, http://www.bugzilla.org/
[7] Sitio Oficial de TrackStudio, http://www.trackstudio.com/
[8] Usando Jira en Desarrollos Ágiles, http://blogs.atlassian.com/2006/01/how_we_use_jira_and_confluence/


## IX. Curriculum Vitae


**Christian Nahuel Balsamo**.

**Alumno de la Universidad de Palermo de la carrera de Ingeniería en Informática.**

**Egresado de la Escuela Técnica Nº 3 María Sánchez de Thompson.**