BotBasic

Escenarios y perfiles de gente

Resultado de la ejecución de la tarea 1b de la Pila de Tareas

Gorka G LLona

Contenido

Revisiones

- 02.05.2017 Primera versión del documento.

Notas

- Se recomienda leer el documento “veni-botbasic concepto marketing” de fecha agosto/2016 antes de proceder con este.

Objetivo

Analizar varios escenarios de operación, definir los perfiles del recurso humano que se requerirá en cada uno y algunos habilitadores tecnológicos.

Los escenarios no son excluyentes entre sí.

Los escenarios son presentados en orden de complejidad de gestión:

1. BotFactory
2. Cloud
3. Open Source
4. The Toolbox

BotFactory

En este escenario Venicua es un generador de negocios por cuenta propia o en sociedad con terceros. Para cada uno de esos negocios Venicua genera una solución tecnológica en forma de una BBapp. El hosting de las BBapps reside en uno o más clusters dentro de las instalaciones de Venicua.

El uso y desarrollo de BotBasic (BB) se circunscribe al ciclo de vida de las BBapps asociadas a los negocios de Venicua. Estas BBapps plantean necesidades, hacia BB, de los siguientes tipos:

1. **Desarrollo**: actualmente están planificadas las siguientes capacidades adicionales a las existentes a la fecha, a las que se agregan otras de naturaleza continua:
    a. Capacidades multimedia (actividad 2 de la pila de tareas (PT-2)).
    b. Integrar con BB con webchat (nueva actividad PT-2b).
    c. Archivos ICS (PT-3).
    d. App de autoconfiguración de BBapps (PT-6a).
    e. Otras mejoras para autonomía de programadores BB (PT-6b).
    f. Push notifications (PT-7).
    g. Nuevas chatapps (PT-2b = webchat, PT-8, PT-9).
    h. Detección, diagnóstico y corrección de bugs (prioritario).
    i. Documentación.

2. **Plataforma**: construcción, administración y adecuación continua de la infraestructura de hardware/software sobre la que corre BB:
    a. Suministro eléctrico y UPS's.
    b. Enlaces de Internet y routers.

c. Clusters: hardware, Linux y servicios base.
d. Bases de datos de BB y BBapps.
e. Actualizaciones a nuevas versiones de BB.
f. Servicios contratados: nombres de dominios, certificados SSL, túneles de acceso a back-end, API's (pasarelas de pago, ...)
g. Revisión diaria de logs.
h. Seguridad física.
i. Documentación.
j. Containerización (con Docker).
k. Administración de la seguridad informática.

3. **BBapps**: dentro del ciclo de vida de cada una se trabaja en:
a. Análisis y diseño desde el punto de vista del negocio.
b. Análisis y diseño desde el punto de vista técnico.
c. Creación de bases de datos (BD's).
d. Codificación en BB y pruebas alfa.
e. Codificación en PHP (bizmodel) y pruebas alfa.
f. Autoconfiguración de bots (@Telegram).
g. Pruebas beta.
h. Pase a producción.
i. Revisión diaria de logs (bot de monitoreo).
j. Detección, diagnóstico y corrección de bugs.
k. Mejoramiento continuo.
l. Documentación.

Los literales que detallan cada una de las necesidades enumeradas conforman, conjuntamente, las responsabilidades del equipo técnico que trabaja como back-end de Venicua en este escenario.

El escenario BotFactory es el más fácil de lograr y el más inmediato. La descripción del personal está proyectada desde la situación actual, en la que Gorka trabaja en Boquete y el cluster estará en Ciudad de Panamá.

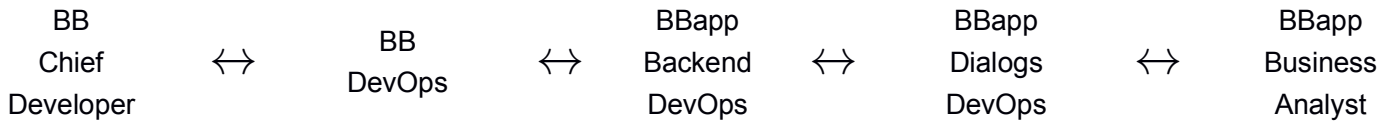
Hay, a grandes rasgos, 5 perfiles en el equipo humano que debe asumir estas responsabilidades. Se enuncian aquí junto con las responsabilidades técnicas asociadas:

A. BB Chief Developer	1(a-i)	
B. BB DevOps	2(a-k)	
C. BBapp Business Analyst	3(a,g)	
D. BBapp Dialogs DevOps	3(b,d,f,g,h,i,j,k,l)	Codifica en BotBasic
E. BBapp Backend DevOps	3(b,c,e,g,h,j,k,l)	Codifica en SQL y PHP

El esquema general de funcionamiento es el siguiente: el BB Chief Developer mantiene a BotBasic funcionando y lo mejora. Las versiones de BotBasic son probadas en su equipo de desarrollo; una vez que pasan las pruebas, son enviadas al BB DevOps, quien las instala en los clusters (actualización de previas versiones de BB). EL BB Devops

monitorea los logs de manera continua y envía reportes de posibles bugs al BB Chief Developer, quien los trabaja. Eso es la operación técnica desde el núcleo del back-office, que incluye también el mantenimiento de la infraestructura por parte del BB DevOps. Desde el lado de los negocios, el BBapp Business Analyst, como parte del desarrollo de un negocio concreto, un test-drive o un MVP, genera diagramas de modelos de negocio e historias conversacionales que transfiere al BBapp Dialogs DevOps, el cual define qué se puede hacer en lenguaje BB y qué debe hacerse en PHP (primitivas, magicvars); estos últimos requerimientos los transfiere al BBapp Backend DevOps, el cual las implementa y las pone a su servicio. El BBapp Backend DevOps interactuará con el BB DevOps para la instalación de los componentes PHP del bizmodel en los clusters.

Este diseño con estos roles tiene la ventaja de que implementa bajo acoplamiento; es decir, no todos hablan con todos, sino que se minimiza la cantidad de nexos entre los diferentes roles, contribuyendo a la disciplina organizacional:



En una práctica inicial del "hoy mismo", hay varios perfiles fusionados:

- BB Chief Developer, BB DevOps, BBapp Backend DevOps y BBapp Dialogs DevOps: en Gorka.
- BBapp Business Analyst y BBapp Dialogs DevOps (para test-drives): en Jorge y Nicole.

Pero la única forma previsible de escalar el escenario BotFactory, tanto en tiempo (rapidez de avances - time to market) como en magnitud (cantidad de negocios soportados) es si el equipo humano asigan la siguiente cantidad de recursos por rol:

A. BB Chief Developer	1 persona (desde ya)
B. BB DevOps	1 persona (a partir del 1er negocio comercial en funcionamiento)
C. BBapp Business Analyst	Según demanda de negocios
D. BBapp Dialogs DevOps	Según demanda de negocios
E. BBapp Backend DevOps	Según demanda de negocios

A esta composición debería llegarse en forma progresiva.

Una consideración importante es que es posible pensar en una estructura donde los roles "D" y "E" estén fusionados en un único "BBapp Integral DevOps", un programador que entienda el negocio, codifique en BB y en PHP a la vez, y mantenga la operación técnica del ciclo de vida de las BBapps. Sin embargo, se considera difícil conseguir este perfil en el mercado a precios razonables. Por ese motivo es mejor separar los roles, especializar y apostar por lograr una buena dinámica de trabajo en equipo.

Adicional a las responsabilidades de cada perfil, hay otras relacionadas con el trabajo en equipo o competencias "blandas".

A. BB Chief Developer	disciplina, calidad, comunicación efectiva, autoinvestigación
B. BB DevOps	disciplina, trabajo bajo presión, calidad, comunicación efectiva
C. BBapp Business Analyst	pensamiento estructurado, comunicación efectiva
D. BBapp Dialogs DevOps	disciplina, trabajo bajo presión, buena redacción, comunicación efectiva
E. BBapp Backend DevOps	disciplina, trabajo bajo presión, comunicación efectiva

Cloud

El objetivo de este escenario es distinto a BotFactory. Con Cloud, Venicua ofrece y administra una "nube" en la que corre BotBasic. Personas de cualquier parte del mundo pueden inscribirse como usuarios de la nube para ser BBapp Developers, aprender BotBasic por medio de cursos online, opcionalmente certificarse, y generar sus propias BBapps, las cuales corren en la nube ofrecida por Venicua con bots de Telegram registrados por los propios Developers.

El modelo de negocios no está definido aún, pero debe tomar en consideración los siguientes elementos:

1. La formación y crecimiento de una comunidad BotBasic externa a Venicua es interesante pues contribuye a posicionar BotBasic como una solución que permite crear chatbots de complejidad conversacional mucho mayor a las soluciones ofrecidas actualmente en la red.
2. El posicionamiento público de BotBasic influye directamente en el posicionamiento comercial de Venicua, tanto en prestigio como en visibilidad.
3. Puede implementarse un modelo Freemium en el cual algunas características sean cobradas a los BBapps Developers.
4. A todo tipo de BBapp Developer puede cobrársele por el uso de primitivas que conformen una "librería estándar ampliada de primitivas" que ofrezca funciones, acceso a API's específicos de Internet, etc.

El modelo Freemium puede ser diseñado como un modelo de acceso gratuito que ofrezca la mayoría de las características de BotBasic que estarán disponibles en el escenario BotFactory: el BotBasic actual más las reseñadas como 1(a,c,d,e,f). Mientras que otras características pueden ofrecerse en la versión premium: 1(b,g) y otras como espacio ampliado en BD, tráfico ilimitado, etc.

Una "librería estándar de primitivas" (Common Library, CL) debe ser diseñada y desarrollada para incluir allí una completa colección de rutinas útiles, de propósito general, que permitan a los Developers disfrutar de un lenguaje completo, apto tanto para las necesidades de las BBapps como para los diferentes estilos de programación que pueden desarrollar. La librería estándar deberá ser segmentada en varias colecciones, para aportar organización y a la vez permitir la implementación del modelo Freemium (si no se va a implementar este modelo, también es necesaria).

Para lograr un lenguaje completo, la CL debe incluir rutinas que favorezcan la implementación de BBapps poderosas sin necesidad de tener que desarrollar un bizmodel en PHP. Esto se debe a que el escenario Cloud no

debe incluir la posibilidad de que los Developers implementen BBapps con bizmodels en PHP, ya que la inserción de código "extraño" en un cluster puede ocasionar anomalías (por mala programación o mala intención) que pueden degradar el funcionamiento del cluster de manera importante. Características de este tipo de código son: fallas de parseo de PHP por errores de sintaxis, acceso a rutinas que manipulan el sistema de archivos del servidor, ciclos infinitos, alocación de memoria excesiva o descontrolada. Cabe destacar que, por el contrario, una BBapp que esté implementada sólo como código en BB no permite quiebres de la estabilidad el cluster, dado que los ciclos infinitos están controlados por duración de tiempo (sólo se prevé que tenga que fortalecerse el datahelper -instrucciones DATA SET y DATA GET- para evitar llenado artificial de la BD común de la VM de BB).

Entre las colecciones que deben ser implementadas como parte de la CL están:

1. Funciones matemáticas.
2. Funciones de manejo de strings.
3. Funciones de acceso a bases de datos SQL.
4. Funciones de fecha y hora.
5. Funciones relativas a geolocalización.
6. Funciones de procesamiento de imágenes, videos, audios y otros formatos multimedia importantes (incluye ICS).
7. Funciones de acceso a web services externos al cluster (entre otras cosas, permite emular primitivas de BB).
8. Accesos directos a API's importantes (ya implementados: envío de email por Gmail, envío de SMS por Messente; otros).

Además de la CL, es importante completar características de BB que forman parte, como estándar, de los lenguajes de programación de alto nivel. Sin ellas, no tiene sentido ofrecer BB a la comunidad de desarrolladores, ya que tendrá baja aceptación a pesar que por medio de técnicas de programación se puedan lograr los mismos resultados usando otras combinaciones de instrucciones. Estas características o extensiones son:

1. Juego completo de estructuras de control (FOR, LOOP, WHILE, REPEAT-UNTIL, BREAK, CONTINUE).
2. Operadores booleanos adicionales (AND, OR, XOR).
3. Soporte a tipo de dato arreglo (colecciones de elementos bajo un mismo nombre, a semejanza del OPTIONS actual).
4. Soporte de localización (a idiomas) para otras lenguas occidentales adicionales a español e inglés.
5. Eliminación de la capacidad de implementar MENU's con extensión a PHP (no ha sido probado y ha demostrado no ser necesario).

Dado que se trata de un escenario que requiere un desarrollo adicional importante, no está de más completar las pruebas de las partes de BB que están implementadas pero que no han sido puestas en producción aún, y que son:

1. BBapps con múltiples canales para un mismo bot/rol.
2. Transferencia de control/interacción automática entre diferentes ChatMedia para una misma BBapp como producto del cambio en las preferencias de interacción del usuario de la BBapp.

Para habilitar la nube, BotBasic requiere la implementación de una interfaz de administración (Dev Admin Toolkit, DAT). Actualmente se puede considerar que el núcleo ya implementado del DAT es la interfaz web del parser y el reflejo de los eventos de bitácora en el bot de monitoreo de cada BBapp. Los elementos que formarán parte del DAT son:

1. Parser.
2. Bot de monitoreo de cada BBapp.
3. App de autoconfiguración de bots, que en el escenario BotFactory es 1(d).
4. Dashboard operacional con estadísticas e interruptores maestros de control de las BBapps.
5. Herramienta de soporte a BBapps Developers, a ofrecer por la comunidad y por Venicua.
6. Base de conocimiento que incluya, entre otros contenidos, materiales educativos.
7. Dashboard de servicios que permite administrar el plan (free, premium) al que está suscrito el Developer.

Los componentes que están atados a la VM de BB deben implementarse de la manera más opaca posible. Una de las claves es no revelar el nombre del servidor (accesible públicamente desde Internet) que identifica al cluster, a fin de minimizar posibles ataques. La única forma de implementación opaca disponible es a través de bots de BB. Se requiere, entonces, un trabajo de adaptación para el parser, dado que actualmente funciona como una interfaz web.

Dado que es previsible que el escenario Cloud requiera la instalación de más de un cluster, debido a la demanda de los BBapp Developers, sería necesario implementar una herramienta de administración del cluster que consolide la administración de los diferentes servicios que son conjugados en la operación de un cluster. Esto puede ser tan sencillo como una colección de shellscripts a los que se accede por linea de comandos (CLI) y a la vez por medio de una interfaz web de acceso privado.

Por último, sería una especie de "seguro de operación" el diseño e implementación de un "Health Monitor" (HM), que a modo de proceso permanente en el cluster, monitoree el comportamiento del cluster, de BB y de las BBapps para detectar anomalías y condiciones sospechosas. Esto puede garantizar la escalabilidad de la nube en relación al tiempo requerido por los administradores para detectar y resolver problemas, pero a la vez es el componente más difícil de pensar, diseñar y desarrollar. Siendo conservadores con respecto a la prestación de servicios de calidad, es mejor incluirlo e implementar al menos una versión inicial simple y de alcance limitado que pueda ir siendo mejorada con el tiempo a partir de los problemas que vayan siendo detectados.

A partir de todo lo anterior, puede observarse que las necesidades planteadas por el escenario Cloud son una extensión de las del escenario BotFactory, y que la experiencia que sea obtenida dentro del escenario BotFactory, junto con sus activos de información, son los que constituyen el escalón de arranque para apalancar la construcción del escenario Cloud.

Las necesidades que plantea el escenario Cloud son:

1. **Desarrollo**: cada componente comprende diseño, desarrollo, pruebas, puesta en producción, documentación, mejoras y mantenimiento:
   a. Del escenario BotFactory: 1(a-i).
   b. Common Library (CL).

c. Extensiones al lenguaje BB.
d. Pruebas y puesta en producción de funcionalidad actual del código fuente de BB.
e. BBapp parser.
f. BBapp de autoconfiguración de bots.
g. BBapp dashboard operacional.
h. Web app de soporte a Developers.
i. Web app base de conocimiento (KB).
j. Web app panel de control de servicios y billing.
k. Contenidos instruccionales y para la KB.

2. **Plataforma**:
   a. Del escenario BotFactory: 2(a-k).
   b. Replicación de clusters para operación comercial.
   c. Replicación de clusters para laboratorio (BB Chief Developer, BB DevOps).
   d. Seguridad y redundancia de electricidad y enlaces de datos.

3. **Clientes**:
   a. Soporte a BBapp Developers.

Los perfiles de gente que se requieren para este escenario son:

A. BB Chief Developer	1(a-g)
B. BB DevOps	2(a-d)
C. Customer Support Developer	1(h-k)
D. Customer Support Agent	1(k), 3(a)

La cantidad de recursos que deben ser asignados a cada perfil es:

A. BB Chief Developer	1 persona
B. BB DevOps	1 persona; luego adicionales según demanda en función de cantidad de clusters en operación
C. Customer Support Developer	1 persona
D. Customer Support Agent	1 persona

Es importante destacar que el escenario Cloud no admite una operación inicial donde se fusionen perfiles diferentes en una misma persona, o una modalidad en la que sean manejadas en forma paralela, por la misma persona, las responsabilidades de perfiles similares de los escenarios BotFactory y Cloud cuando estos corren en forma paralela. Si se opta por esto, simplemente, la capacidad de trabajo real será excedida por la demanda de actividades a realizar y no se podrá garantizar una mínima calidad de servicio, que es vital en negocios como los que se plantea con este escenario. Esta observación aplica para los siguientes escenarios descritos en este documento.

Adicional a las responsabilidades de cada perfil, hay otras relacionadas con el trabajo en equipo o competencias blandas. Sólo se muestran las de los roles que han sido agregados con este escenario; para las otras consultar el

escenario BotFactory:

C. Customer Support Developer	disciplina, trabajo bajo presión, calidad, comunicación efectiva, orientación al cliente
D. Customer Support Agent	disciplina, comunicación efectiva, orientación al cliente

Comentarios adicionales:

- El escenario Cloud, a semejanza de los siguientes que son expuestos en este documento, requieren hacer visible y público el nombre BotBasic. Debe estudiarse cuidadosamente este asunto, ya que existe en el mercado actualmente una solución de software llamada BotBasic, cosa que se puede verificar fácilmente en Internet (ver https://github.com/kaiateic/BotSpine/wiki/BotBasic-An-Introduction). El nombre BotBasic podría tener que ser cambiado. Una posibilidad es "ChatBasic" que al parecer está disponible (un producto de software existía con ese nombre pero ya no está disponible; el dominio chatbasic.com está comprado por alguien pero no tiene contenido). La desventaja de este nombre es que se parece a "CheatBasic".
- En función del rendimiento y la competencia con otros productos, vale la pena posicionar físicamente los clusters que conformarán la nube en lugares en los que la latencia en las conexiones troncales-de-Internet $\leftrightarrow$ cluster sea lo más baja posible. Lo mejor es colocar al menos un cluster o banco de clusters con conexión directa por fibra óptica a backbones ubicados en USA, y hacer lo mismo en un punto de Europa.

Open Souce

El objetivo de este escenario es la liberación al mercado (la "comunidad" de nuevos "desarrolladores") del código fuente de BB, y a la vez la generación de algún modelo de negocios que permite la subsistencia de Venicua y un adecuado retorno sobre la inversión.

La liberación del código fuente de BB tiene la ventajas y desventajas. Las ventajas son:

- Desde 2016 hay un consenso general de que la industria del open source "ganó la batalla" que libraba por años. Continuamente empresas de open source se establecen con posiciones sólidas en un sector de la industria que siempre había sido dominado por soluciones propietarias. La fama del software open source es que es un producto de mejor calidad que la del software propietario. El open-source se considera actualmente como la clave de la innovación en software, produciendo desarrollos más rápidos y ágiles, time-to-market acelerado e interoperabilidad con otras soluciones. Es un modelo con una "madurez joven" de 20 años.
- Es un modelo coherente con la infraestructura que usa BB, que es 100% open source (Linux, Mysql, Apache, Php, Unirest, PhpMailer) y las que usará (Docker).
- La apropiación del código por parte de la comunidad produce una mejora de la calidad del código mismo y del producto, bien sea por causa de detección y corrección de bugs, extensiones funcionales y no funcionales, refactorización, interoperabilidad, eficiencia incluyendo reimplementación sobre lenguajes más eficientes, arquitectura de servicios y de clusters, etc. La calidad de las soluciones está reportado como la tercera ventaja del uso de software libre en 2016.

- El codigo actual (PHP) tiene un nivel de calidad suficiente como para que sea aceptado y hecho evolucionar por la comunidad de desarrolladores.
- Se habilita una modalidad en la cual personas externas a Venicua pueden implementar primitivas (en PHP) en forma irrestricta e ilimitada sobre sus propios servidores; en otras palabras, los clientes pueden disfrutar del 100% del potencial que disfruta Venicua.
- No hay lock-in por parte de los desarrolladores a Venicua, lo cual es un factor que favorece la aceptación y apropiación de BB por parte de ellos. Este factor está reportado como la segunda ventaja del uso de software libre en 2016.
- Este es el único escenario que permite escalar el negocio de BB en forma más que directamente proporcional a la inversión o a los costos; en otras palabras, es la forma más rápida de posicionar BB a nivel mundial sin inversiones o crecimiento de gran escala.
- No excluye ninguno de los otros escenarios.
- Permite la habilitación de una variante del escenario The Toolbox sin los inconvenientes de la encriptación del código de BB.
- Se contribuye significativamente al crecimiento de la industria de chatbots a nivel mundial, cosa que es beneficiosa en sí misma y arroja sobre Venicua un nivel de prestigio que no puede alcanzar con ninguno de los otros escenarios; sobre este prestigio puede apalancar otros negocios.

Las desventajas son:

- El porcentaje de empresas que liberan software al mercado como open source y tienen éxito no es alto. Por otra parte, se puede decir que el porcentaje de startups basadas en software que tiene éxito tampoco es alto. Por lo tanto esto pudiera no ser considerado una desventaja.
- Si bien el open source está suficientemente posicionado, no lo está el hecho de que las empresas de open source consigan inversionistas con la facilidad que -se podría pensar- se derivaría de ese posicionamiento (investments of $3.5 billion into open-source startups between 2012 and 2015; $7 billion has been invested in open source but only $1 billion of that has been returned to investors).
- Requiere una logística de evolución, control e integración de código fuente a partir de los requerimientos y aportes de la comunidad, que puede representar un problema de escala en la medida en que la comunidad de "modificadores del código fuente" crezca. Típicamente se utiliza la herramienta git para la administración participativa del código.
- Debido a que el código fuente estará disponible, el análisis de fallas por parte de terceros podría conllevar riesgos para la operación comercial del escenario Cloud o incluso del escenario BotFactory (uso de bugs no descubiertos por Venicua para afectar BBapps existentes).
- Es previsible que surjan variantes de BB incompatibles entre sí; es decir, "forks" de github que implementen extensiones de BB que habiliten la construcción de BBapps que corran sobre una variante de BB, pero no sobre otra o sobre el BB original. La concreción de este riesgo debilita la portabilidad del código entre proveedores de clouds de BB.
- No hay lock-in por parte de los desarrolladores a Venicua, lo cual puede afectar al modelo de negocios de la empresa debido a la competencia por parte de empresas fundadas por inversionistas adinerados que se apalancan sobre los activos producidos por Venicua. Para contrarrestar esto, la captación de inversionistas propios que permitan obtener una velocidad de mercado adecuada es un factor clave.

El balance entre ventajas y desventajas es un juego delicado e incierto, que debe generar un modelo de negocios que esté claro desde el inicio. Precisamente se considera esto, tener un modelo de negocios claro desde el arranque del proyecto open source, un factor clave para evitar el fracaso dentro de esta subindustria.

El licenciamiento del sofwtare el el elemento central del modelo open source y en general de todo modelo que, como BB, entrega al cliente el código fuente de sus activos.

> "Licenses range from permissive (Apache) to restrictive (General Public License, or GPL, variants, with Affero GPL the most restrictive), and everything in between. There are 71 different flavors, to be exact, all listed on opensource.org. More permissive licenses make it easier to build large ecosystems. If any open-source project can incorporate yours, you can get a tailwind by being included in, or connected to, other successful projects. Looser licenses also may make it easier to solicit contributions to the product, and they make users more comfortable that the project is less dependent on a small startup ... For open-source entrepreneurs who believe their product can gain wide adoption without having to piggyback on the success of other open-source projects, a more restrictive license may be better in the longer run. But if you go that route, be prepared to invest heavily in building community in the early days". (Techcrunch)

No es lo mismo software libre que open source. Una licencia de software libre debe garantizar al usuario las llamadas "4 libertades", que son:

> "0: la libertad de usar el programa, con cualquier propósito (uso).
> 1: la libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a las propias necesidades (estudio).
> 2: la libertad de distribuir copias del programa, con lo cual se puede ayudar a otros usuarios (distribución).
> 3: la libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie (mejora)." (Wikipedia)

Por su lado, una licencia open source define la entrega al cliente (developer, en el caso de BB) del código fuente, pero ninguna otra condición. Las licencias de software libre se consideran más restrictivas que varias open source pero menos que licencias propietarias de código cerrado.

La dificultad de generar un modelo de negocios a la vez viable y de gran escala ha hecho que surja recientemente la Fair Source License:

> "Why did you write a new license? Why didn’t you use GPL/MIT/BSD/Apache?
> Open source licenses are excellent. They have helped advance technological progress, and we love them for that. However, open source doesn’t make sense for all code. If a company open sources code that represents its core business value, it would generally fail as a business—and it couldn’t continue to build great software for you.
> By creating the Fair Source License, we’re offering the world a new choice in between open source and closed source. Fair Source enables companies to share source code with the public and still drive revenue. It’s the best of both worlds—transparency and progress without sacrificing our businesses." (fair.io)

Esta licencia permite ofrecer, por ejemplo, el software open source sin costo a particulares y a organizaciones con hasta X usuarios de BB. Por ejemplo, una "Fair Source 3 License" permite 3 usuarios gratis y a partir del cuarto usuario la organización cliente debe pagar las tarifas que Venicua defina. Este modelo, que se define a él mismo como de licencia propietaria open source, se muestra útil para Venicua (cuando "X" es bajo) porque:

1. Permite el surgimiento de la comunidad de developers, compuesta tanto por particulares como por empresas pequeñas, con los beneficios que eso representa.
2. No canibaliza el escenario Cloud y lo hace compatible con éste, de modo que si surgen otros proveedores Cloud, estarán pagando por la licencia, ya que probablemente los licenciatarios requieran más de "X" usuarios internos para gestionar su nube. Esto puede ser conveniente mientras Venicua se posiciona y logra obtener inversionistas.
3. Permite iniciar con un modelo de entrega del código fuente restrictivo, y a medida que pasa el tiempo evaluar si se debe migrar a un modelo más permisivo que ofrezca mayores libertades a los desarrolladores. Si se comienza al reves, por un modelo de licencia de software libre clásico o uno aún más permisivo como la licencia MIT, será muy difícil restringir a futuro la modalidad a un modelo basasdo en la Fair Source License.
4. Favorece el esquema de pensamiento del cliente "try-before-buy".

La otra modalidad es diseñar una licencia dual: bajo unas condiciones asociadas al cliente, se entrega el producto con una licencia de software libre o similar. Bajo otras, se entrega bajo una licencia comercial. Este modelo era el que se usaba antes de la aparición de la Fair Source License y no es tan recomendable, pues resulta bastante complejo tanto para Venicua como para sus clientes.

El modelo de negocios que entrega el código fuente y que resulta apropiado para BB, independientemente del esquema de licenciamiento, se basa en una combinación de producto y servicios, y se deriva de la propia evolución del negocio. No hay mucha información disponible sobre modelos de negocios open source exitosos más que las que se refieren a las pocas empresas open source públicas que han tenido éxito, entre las cuales despunta Red Hat Software, cuyo modelo de negocios obedece a circunstancias históricas diferentes a las actuales.

Uno de los esquemas de negocios open source actuales es:

> "According to Accel (they are one of the most prominent venture capital investors and were early investors in companies like Facebook, Dropbox, Slack, Etsy, Atlassian, Lynda.com, Kayak etc.) open software companies are constructed differently and pass through 3 stages, the 3P's. Companies that nail all three, can build a large business.
>
> 1. The Project phase - developing an open source project and a vibrant community around it.
> 2. The Product phase - laying the groundwork for packaging and deploying a monetizable version of the project.
> 3. The Profit phase - doubling down on a core customer-base and builds the business to scale." (Drupal)
>
> "At the end of the project phase, an open company might be 80% services and 20% product. By the end of the product phase, this should flip." (Accel@Medium)

Con respecto al producto, conviene separar de la versión "community" algunas características que pueden ir únicamente en la "premium"; sin embargo, esto puede representar costos elevados en time-to-market, y por tanto puede ser considerado opcional:

1. Performance and business analytics (por desarrollar).
2. Cluster management integrated tool (por desarrollar).
3. Your-own-cloud toolkit (kit de herramientas de software producto del desarrollo del escenario Cloud).
4. Better auditing, security, monitoring.
5. Integración extensa con otros softwares (API's, ...) (por desarrollar)
6. El parser lee archivos nativos de Excel y LibreOffice, y no sólo CSV (por desarrollar).
7. Procesamiendo de lenguaje natural (por conceptualizar y desarrollar).

Independientemente de cómo se diferencie, a nivel de producto, la versión community de la premium, lo cierto es que es conveniente que exista esa diferenciación. De otro modo, no se puede capitalizar por producto y no se puede aspirar a la relación 80-20 indicada arriba. La versión premium no es más que la community, con la misma licencia y con un esquema de cobros distintos (en caso de usar la licencia Fair Source), agregando a ello la serie de componentes adicionales de software que se entregan con una licencia estrictamente comercial.

En materia de servicios ofrecidos que generen retorno financiero a Venicua, pueden ser considerados:

- SaaS / PaaS: implementar en paralelo el escenario Cloud.
- Servicios de customización de BB a necesidades específicas de clientes.
- Soporte directo, por diferentes canales. Por su lado, las licencias community dependen de foros web para el soporte de los developers.
- Training: online o presencial, con posibilidad de certificado en el caso online.
- Empaquetamiento HW+SW: construcción y venta de clusters basados en Odroid's.
- Una BBapp pequeña, gratis, desarrollada a la medida por Venicua, como beneficio incluido dentro del pago de la licencia. Esto puede apalancar al escenario BotFactory.
- Suscripciones: "we BotFactor and you host yourself". Se puede cobrar uan suscripción anual por un número determinado de horas de desarrollo por parte de Venicua.

Uno de los riesgos del escenario open source es que developers avanzados pueden crear extensiones a BB y pueden surgir variantes incompatibles entre sí. Como resultado, en el mediano y largo plazo, pueden surgir nubes comerciales de BB donde cada una soporta un dialecto distinto. Esto impediría la portabilidad de BBapps entre proveedores de nubes y generaría lock-in de los BB developers a sus proveedores particulares.

Tal situación va a suceder, pero hay que evitar al máximo la fragmentación del lenguaje. Para ello, sería conveniente, desde los inicios, favorecer la fundación de un "BotBasic Group" que incluya a todos los desarrolladores (particulares o empresas) interesados en mejorar el lenguaje. Este grupo generaría una dinámica de trabajo colectiva hacia la estandarización de BB y la preservación de un "main branch" de la especificación, que va avanzando a partir de BotBasic 1.00 hacia sucesivas versiones. Esta modalidad de organización ya hay sido probada con éxito para otros lenguajes.

La habilitación de este escenario requiere efectuar un conjunto de actividades, algunas previas que lleven a BB a un nivel mínimo de entrada dentro del mundo open source, y otras durante la misma operación:

1. **Desarrollo**:
    a. Del escenario Cloud: 1(a-g).
    b. Del escenario Cloud: 1(h-k).
    c. Implementar notación infija adicional a la prefija para expresiones aritméticas, con parentización (ej 1: "SET c a : SUM c b" puede expresarse también como "LET c = a + b") (ej 2: "SET u x : SUM u y : MUL u 2" puede expresarse también como "LET u = 2 * (x + y)").
    d. Traducir todos los comentarios del código fuente de español a inglés.
    e. Generar documentación adicional (la documentación tipo API se genera automáticamente con phpdoc).
    f. Generar un documento formal de especificación del lenguaje BotBasic 1.00.
    g. Web de auto-soporte para developers (único recurso disponible para developers de licencias no pagas, junto con la base de conocimiento).
    h. Detección, diagnóstico y corrección de bugs (detección interna y externa).
    i. Implementación de características requeridas (requerimientos internos y externos).
    j. Integración de aportes externos (git management).
    k. Co-definición periódica de las siguientes versiones de BB "main branch" (especificación de lenguaje).
    l. Componente de envío opcional de estadísticas anónimas de uso (de BB por parte de los implementadores) a Venicua.

2. **Plataforma**:
    a. Del escenario Cloud: 2(a,c).
    b. Consolidar clusterización con Docker (refinación de 2(j) del escenario BotFactory).
    c. Pruebas de instalación en nubes públicas a partir de containers Docker de BB.
    d. Administración de clusters de laboratorio.
    e. Administración de una instalación sobre un servicio en la nube de Internet (laboratorio).

3. **Community management** (clientes):
    a. Promoción (a nivel técnico) de BB entre la comunidad de software libre.
    b. Soporte a clientes/developers del modelo free.
    c. Soporte a clientes/developers del modelo premium.
    d. Administración del BotBasic Group.

Los perfiles de gente que se requieren para este escenario son:

A. BB Chief Developer	1(a,c-f,h-l) y 3(d)
B. BB DevOps	2(a-e)
C. Community Support Developer	1(b,g) y 3(a-d)

La cantidad de recursos que deben ser asignados a cada perfil es:

A. BB Chief Developer	1 persona
B. BB DevOps	1 persona
C. Community Support Developer	1 persona inicialmente; luego adicionales según crecimiento de la comunidad

Adicional a las responsabilidades de cada perfil, hay otras relacionadas con el trabajo en equipo o competencias blandas. Sólo se muestran las de los roles que han sido agregados con este escenario; para las otras consultar los escenarios anteriores:

C. Community Support Agent	disciplina, comunicación efectiva, orientación al cliente, liderazgo, resolución de conflictos

Referencias:

Sobre datos del mercado:

- http://www.northbridge.com/2016-future-open-source-survey-results
- http://www.zdnet.com/article/its-an-open-source-world-78-percent-of-companies-run-open-source-software/
- https://techcrunch.com/2017/04/07/tracking-the-explosive-growth-of-open-source-software/

Sobre modelos de negocio:

- https://techcrunch.com/2016/02/09/the-money-in-open-source-software/
- https://berlinbuzzwords.de/16/session/community-commercialization-how-build-open-source-company-2016
- https://medium.com/accel-insights/the-rise-of-open-innovation-the-3p-s-for-building-a-durable-open-software-company-3bc6e0ec6fa7
- https://siliconangle.com/blog/2017/05/06/open-source-software-startups-still-struggling-reach-escape-velocity/
- https://en.wikipedia.org/wiki/Business_models_for_open-source_software

Sobre licencias:

- https://ipg.host.cs.st-andrews.ac.uk/monty/open-source-licensing-andrews.pdf
- https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses
- https://es.wikipedia.org/wiki/Software_libre
- https://www.wired.com/2016/03/former-open-sourcers-ask-companies-pay-fair-share/
- https://fair.io/

The ToolBox

El objetivo de este escenario es la penetración en el mercado corporativo bajo un modelo "tradicional" de negocios de IT basado en una combinación de licenciamiento y prestación de servicios.

La prestación de servicios al mercado corporativo por medio del modelo Cloud o una derivación del modelo Cloud adaptado específicamente a empresas tiene las siguientes debilidades:

1. Típicamente las empresas no quieren alojar data sensitiva en ubicaciones externas a la red de datos propia de la empresa. Cuando lo hacen, buscan proveedores de alrta reputación.
2. El nivel de servicio que hay que prestar a empresas (SLA) es superior al del mercado promedio del escenario Cloud. El "uptime" del servicio debe ubicarse en niveles normales de la industria, por lo que debe superar 99% (tiempo disponible / tiempo total) o incluso más.

Para superar (1) es obligatorio instalar un cluster dentro de las instalaciones de la empresa. En caso de que se logre una implementación tipo Cloud, para (2) se requiere demostrar redundancia en suministro eléctrico, enlaces de datos y en el cluster en sí mismo. Los dos primeros elementos pueden obtenerse por medio de un nivel adecuado de inversión e incluso ser certificados (lo cual ayuda a startups como Venicua, que no disponen inicialmente de reputación); el tercero es más difícil, pues implica un rediseño de la arquitectura de BB en función de habilitar clusters de respaldo con funcionamiento tipo "failback en caliente", con las consiguientes implicaciones en tiempos de adecuación de funcionalidades ya existentes, pruebas y estabilización (el área de redundancia de infraestructura es una de las más delicadas y complejas y tiende cada vez más a asegurarse por medio de servicios en la nube y containerización).

Las dificultades mencionadas obligan a apuntar este escenario hacia suministrar a las empresas soluciones on-premise (servidores físicos ubicados en las instalaciones del cliente).

Un modelo de negocios para este escenario debe estar basado en la segmentación del mercado en:

1. "Entry": empresas pequeñas y empresas medianas con uso periférico de BB. A estos clientes se les puede ofrecer una solución basada en el escenario Cloud, para el cual Venicua aprovisione un cluster dirigido específicamente al segmento. Las BBapps, que serían desarrolladas llave en mano por Venicua, podrían incluir bizmodels (PHP) corriendo en el cluster, lo cual no representa un problema de seguridad dado que el desarrollo es interno. Uno o más clusters de respaldo con "failback en frío" deben estar disponibles para casos de falla.
2. "Pro": empresas medianas con uso medular de BB y empresas grandes. A estos clientes se les dota con un cluster dedicado, que on-premise será instalado junto con los otros servidores del cliente.

El segmento Entry puede ser considerado como una extensión natural del escenario Cloud y no se considerará en lo subsiguiente dentro de esta sección. Baste decir que para habilitarlo será necesario implementar prevaiemnet la web app de soporte a clientes empresariales que se requiere para el segmento Pro.

Un elemento central del segmento Pro es la "entrega" del código fuente de BB. Al estar BB implementado en PHP, el código fuente es la versión final (como archivos) del programa ejecutable (esto no sucede con lenguajes como C++ o Java, pero sí con Python o Javascript/node.js). Aquí se plantean las siguientes opciones:

1. Entregar el código fuente original y confiar en la ética del cliente para que no lo distribuya fuera de las fronteras de su organización.
2. Entregar una versión modificada del código, "obfuscada" o, en otras palabras, ilegible para un programador (todos los símbolos -nombres de variables, constantes, funciones, métodos, clases- son

sustituidos por nombres aleatorios, las cadenas de texto son codificadas, los comentarios son eliminados; entre otras técnicas). El alto nivel de complejidad del código de BB permite presumir que una tarea de ingeniería inversa a partir del producto de una obfuscación de código de buena calidad, no sea factible según su costo-beneficio.

El espionaje industrial es una constante tanto dentro como fuera de la industria del software. El hacking no-ético es también una constante en la actualidad. Una proporción grande de los problemas de seguridad de las empresas provienen de insiders. Es difícil apostar por la primera opción de las enumeradas, a menos que este escenario sea desarrollado en paralelo o con posterioridad al escenario Open Source. Así, los riesgos cambiarían radicalmente, ya que la distribución y publicación del código es inherente al modelo de negocios.

Si no se considera la conjunción con el escenario Open Source, sólo queda la opción de obfuscar el código de BB. Para ello, hace falta elegir una herramienta de protección de código y obfuscación. Hasta la actualidad, lo que se ha investigado identifica a PHP Encoder de IonCube.com como la mejor opción disponible. Se puede optar entre las versiones Pro y Cerberus (las licencias cuestan $299 y $399 respectivamente y son perennes) y se requiere inicialmente sólo una licencia (si la cartera de clientes corporativos se va incrementando, será necesario comprar una segunda licencia con un descuento de 25%).

Es importante tomar en cuenta que, independientemente de que funcione, el uso de PHP Encoder u otra similar ocasionará una pérdida de rendimiento apreciable (tanto por la codificación como por no aprovechar el engine v7 de PHP); lo cual supondrá un impacto en la velocidad de respuesta del cluster: (a) mínimo cuando el cluster carece de carga; (b) apreciable cuando la carga supera el 50% de la carga máxima funcional de un cluster no-protegido equivalente. Esto obliga a reducir las expectativas del hardware del cluster, o sencillamente a agregar más nodos/Odroids tipo "Apache servers" al cluster.

De funcionar, PHP Encoder aporta fácilmente la solución para la parte "producto" del combo producto+servicio del modelo closed-source. Una facilidad que incluye esta solución de protección de código es la posibilidad de generar versiones obfuscadas que funcionan hasta una fecha determinada, con lo que se puede implementar la garantía de un modelo de negocios basado en suscripción anual, por ejemplo. La versión Cerberus permite anclar el código que se ejecuta a la dirección MAC del nodo del cluster, por lo que para el cliente no sería factible copiar el código de BB para correrlo en otro dispositivo o aumentar por su cuenta la capacidad del cluster por medio de la adición de nodos Apache.

Una versión de evaluación de PHP Encoder está disponible, por lo que es posible probar antes de comprar para asegurar la factibilidad y medir el rendimiento del código obfuscado (velocidad y consumo de memoria), con el objeto de redimensionar la capacidad nominal de un "cluster protegido" (como se llamará a esta variante de cluster desde ahora).

Pero ¿cuál es el modelo más creible para los clientes corporativos, en la actualidad? Definitivamente, un modelo basado en código abierto y auditable. Apalancar el escenario The Toolbox sobre el escenario Open Source permite obtener credibilidad ante las empresas. No sólo se les puede demostrar que existen clientes (aunque sean particulares) usando BB, sino la misma característica de código abierto garantiza mejor calidad en el mediano y largo plazo cuando el proyecto está bien administrado y a partir de la colaboración de la comunidad (y no estaría

bien administrado si no se llega al escenario The Toolbox basado en Open Source). Por lo tanto, así como el escenario Open Source es el que permite obtener el impulso requerido para manejar las escalas de negocio más atractivas, The Toolbox sobre Open Source (TT/OS) es también el que permite entrar a las empresas con impulso y escala -siempre que se aúne a una organización comercial de abordaje a los potenciales clientes y soporte a ellos que responda a esta valoración del crecimiento-. Por otra parte, la liberación de código implícita en Open Source hace suponer que surgirán empresas que usen BB para abordar a las empresas, cosa que también puede hacer, en forma más natural, Venicua.

Por otra parte, la variante closed-source (TT/CS) no debe ser automáticamente excluida a partir de las consideraciones del párrafo anterior. Podría ser considerada, por ejemplo, como un escalón previo en el tiempo al escenario TT/OS, orientado a satisfacer los requerimientos de clientes específicos que sean capturados como producto colateral del desarrollo de los escenarios BotFactory y Cloud, y podrían permitir a Venicua conocer el mercado corporativo y desarrollar progresivamente su abordaje comercial. Este enfoque "en escalera" es lo que se considera para el resto de esta sección.

Dentro de la dualidad producto+servicios del modelo de negocios, el producto es BB en forma de "The Box" como plataforma corporativa de funcionamiento de bots. Específicamente:

1. Dotación e instalación de "The Box", el cual consiste en una "caja" de formato servidor de rack que contiene un cluster completo con excepción del router y el UPS.
2. Suministro y configuración de un conjunto de BBapps genéricas, tales como SimpleEDMS e IQcheck.
3. Desarrollo de una BBapp pequeña o mediana, a elegir por el cliente.
4. Taller inicial de desarrollo de BBapps para una o dos personas de la organización cliente.

En relación a los servicios, estaría incluido en una suscripción anual:

1. Actualizaciones al software de The Box por parte de Venicua, en forma remota (sistema operativo, servicios, BB, BBapps genéricas).
2. Servicio de monitoreo y control remoto del nivel de calidad de The Box (modalidad premium).
3. Soporte in-situ para fallas que no puedan ser corregidas en forma remota.
4. Soporte a BBapps developers del cliente (horas mensuales determinadas).
5. Taller semestral de desarrollo de BBapps o refuerzo de conocimientos.
6. Horas adicionales de soporte o entrenamiento remoto (facturables por cada hora).
7. Service Level Agreement (SLA, básico y premium).

Lo anterior conforma una matriz clásica de suministro de plataformas de IT. Sin embargo, variados modelos de negocios son válidos, y deben ser esbozados caso a caso. Lo cierto es que la variante TT/OS debe escalar bien, por lo cual para ello se requiere un modelo estándar sobre el cual se pueda efectuar personalización de última milla.

Las actividades que deben ser desarrolladas para habilitar este escenario son (adicionalmente la variante TT/OS requiere la satisfacción de todas las actividades del escenario Open Source):

1. **Desarrollo**:
    a. Del escenario Cloud: 1(a-g).
    b. Del escenario Cloud: 1(h-k).
    c. Del escenario Open Source: 1(e,f,l).
    d. Del escenario Open Source: 1(g).
    e. Componente de instalación "fácil" y validación de correctitud de los bizmodels (PHP) de las BBapps.
    f. Componente de configuración de recursos externos (bases de datos y directorios de filesystems).
    g. Componente de conexión con servidor de directorio corporativo (LDAP, ...)
    h. Desacoplamiento de bizmodels (PHP) hacia la forma de web services (robustece la estabilidad del cluster).
    i. Gestor de calidad proactivo.
    j. Web app de soporte, manejo de licencias y billing para clientes (extensión de 1(j) del escenario Cloud).
    k. Sistema de actualización de sistema operativo y servicios de los nodos del cluster.
    l. Sistema de actualización de código fuente de componentes de BB (codificados o no codificados).
    m. Sistema de actualización remota de BBapps genéricas.

2. **Plataforma**:
    a. Del escenario Cloud: 2(a,b,c).
    b. Del escenario Open Source: 2(d).
    c. Diseño y construcción del cluster para suministro comercial (un enclosure/case que puede ser usado como base puede encontrarse por $100 en https://www.amazon.com/Rosewill-Rackmount-Computer-Pre-Installed-RSV-R4000/dp/B0091IZ1ZG).
    d. Monitoreo remoto de clusters de clientes premium.

3. **BBapps**:
    a. Del escenario BotFactory: 3(a-l).

4. **Clientes**:
    a. Maquetación de metodología refinada para el desarrollo de BBapps.
    b. Maquetación del training corporativo.
    c. Soporte a clientes (BBapp Developers, administrativos).

Los perfiles de gente que se requieren para este escenario son:

A. BB Chief Developer	1(a,c,e-i,k-m), 4(a)
B. BB DevOps	2(a-d)
C. Customer Support Developer	1(b,d,j)
D. BB Business Consultant	3(a), 4(a-c)

La cantidad de recursos que deben ser asignados a cada perfil es:

A. BB Chief Developer	1 persona
B. BB DevOps	1 persona; luego adicionales según demanda en función de cantidad de clusters en clientes

C. Customer Support Developer	1 persona
D. BB Business Consultant	1 persona; luego adicionales según demanda en función de cantidad y tipo de clientes

Por último, las competencias genéricas de cada perfil. Sólo se muestran las de los roles que han sido agregados con este escenario:

C. Customer Support Developer	disciplina, trabajo bajo presión, calidad, comunicación efectiva, orientación al cliente
D. BB Business Consultant	disciplina, comunicación efectiva, trabajo bajo presión, calidad, orientación al cliente, buena redacción, liderazgo, resolución de conflictos, pensamiento estructurado

Perfiles del personal técnico para los escenarios

En el **Anexo 1** se presenta una **tabla de responsabilidades** en la que, para cada perfil de cada escenario, se reseñan las actividades, responsabilidades y logros asociadas a ese perfil.

En el **Anexo 2** se presenta una **tabla de competencias** en la que, para cada perfil de cada escenario, se detallan las competencias técnicas y genéricas que corresponden a ese perfil.

Se ha dejado por fuera del alcance de este documento la definición de un rango de ingresos mensual/anual para cada uno de los perfiles.

Herramienta para selección de personal

Para acelerar la selección del personal técnico es conveniente utilizar una herramienta de screening online después de la selección por CV y antes de la entrevista. Por $299, interviewmocha.com ofrece un paquete que permite aplicar 100 tests en un plazo máximo de 6 meses.

Una idea más refinada sería aplicar varios tests en forma sucesiva a cada candidato, donde la aplicación de un test implica la aprobación de los anteriores, y en combinación con pasos de entrevista. Por ejemplo:

1. Preselección por CV.
2. Primer test.
3. Segundo test (opcional).
4. Primera entrevista (no técnica).
5. Tercer test.
6. Cuarto test (opcional).
7. Segunda entrevista (técnica).