Cómo transformar una idea en una startup

Julián Mayorga

Cómo transformar una idea en una startup

Julián Mayorga

©2013 - 2014 Julián Mayorga

Índice general

Desarrollo de Ingeniería

Para completar los conceptos sintetizados en la sección anterior, en las siguientes hojas detallaré cómo usar técnicas de Programación Extrema y de Desarrollo de Clientes para transformar una idea inicial en una startup.

Utilizaré el proceso descripto en la síntesis del Marco Teórico, aclarando con ejemplos cómo conseguir los objetivos de las dos grandes etapas de una startup: Descubrimiento de Clientes y Validación de Clientes.

El esfuerzo de ingeniería de software es grande para llevar adelante una startup innovadora. El equipo de ingeniería debe desarrollar y mantener una infraestructura que lograr dos objetivos: lograr una fácil adaptación a los cambios y realizar entregas frecuentes del producto/servicio. Para lograr la capacidad de adaptarse a cambios utilizaré una **arquitectura modular**, **orientada a servicios**; y para realizar entregas frecuentes del producto usaré **Programación Extrema**.

Diseñaré una arquitectura orientada a servicios para lograr un bajo acoplamiento entre cliente y servidor, y así mejorar la capacidad de adaptación del producto desarrollado. Utilizaré la arquitectura cliente-servidor, con una API REST como servidor y una aplicación con tecnologías modernas como cliente.

Para lograr las entregas frecuentes que requiere una startup utilizaré técnicas de XP. Configuraré el desarrollo para posibilitar una puesta en producción continua, y utilizaré Desarrollo Guiado por Pruebas (TDD en inglés), escribiendo pruebas automatizadas tanto unitarias como de aplicación.

El Desarrollo de Ingeniería comienzará con el desarrollo práctico de la etapa Descubrimiento de Clientes.

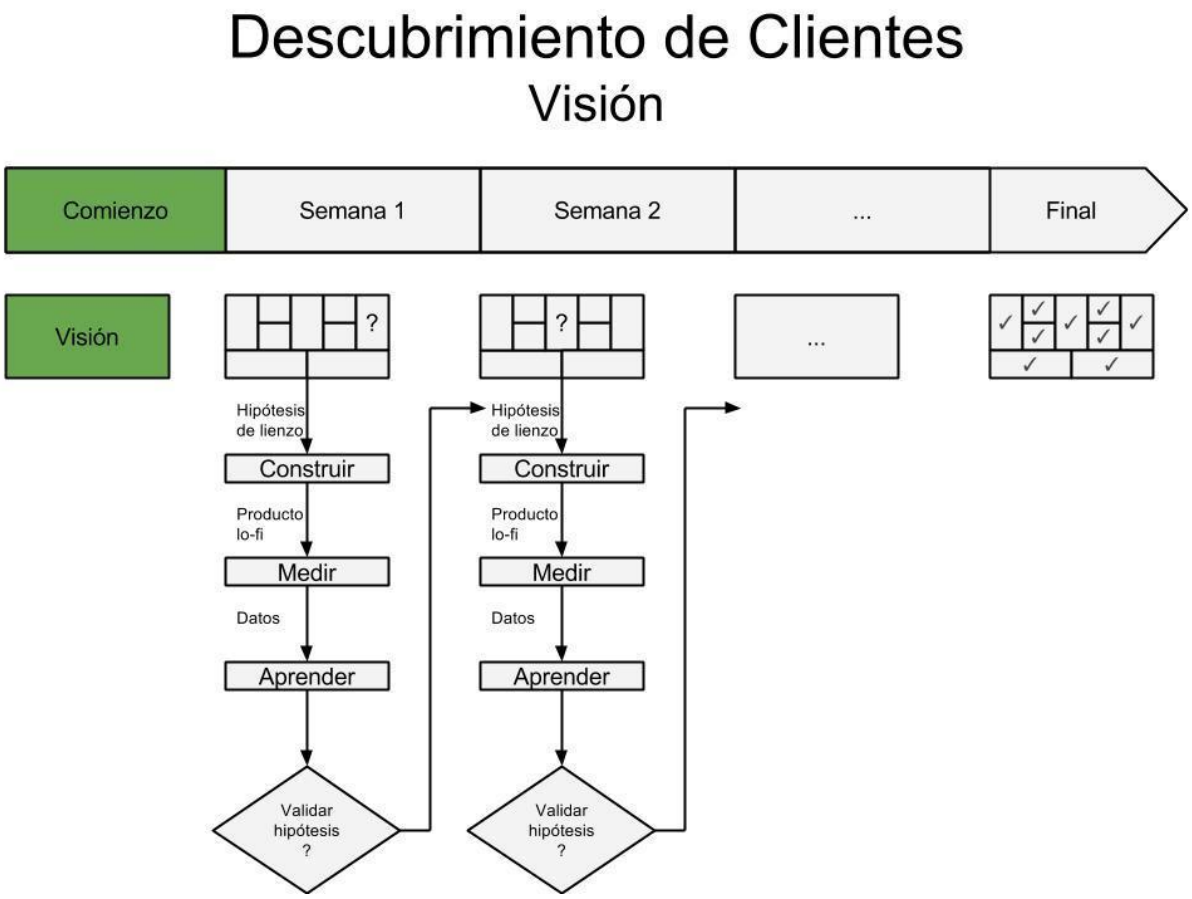
Descubrimiento de Clientes

Ahora comienza la etapa de Descubrimiento de Clientes, en la que se busca validar un modelo de negocios. En esta fase hay que realizar experimento tras experimento hasta llevar a cero la gran lista de incógnitas con la que comienzan todas las startups.

Por cuestiones de tiempo y alcance del presente trabajo realizaré dos iteraciones de Descubrimiento de Clientes, las suficientes para aclarar cada elemento del proceso de creación de startups. En la realidad pueden llegar a ser necesarios meses o años llevar a cabo los experimentos necesarios para conseguir un modelo de negocios estable.

A continuación introduciré la idea sobre la cual construir una startup, y formularé una visión a partir de ella.

Visión



Vision

En este capítulo determinaré la visión que guiará el camino de la startup a partir de una idea. Empezaré por el planteo del problema a resolver, después construiré una visión a partir de lo planteado y sólo al final propondré una solución.

El proceso de planteo de visión propuesto contrasta con la forma de pensar que tienen en su mayoría los ingenieros. Al ser personas encargadas de crear soluciones, generalmente

empezamos la creación de startups con una solución ideal en nuestra mente. El problema con este método es, irónicamente, que no haya problema que se ajuste a nuestra solución.

A continuación describiré el problema a resolver.

Problema

Siempre he pensado que para ser un buen programador no sólo hay que escribir mucho código, sino también leer mucho código. Los buenos escritores se forjan tanto escribiendo obras propias como leyendo obras ajenas, entonces ¿por qué no forjar buenos programadores de la misma manera? Pienso que leer código de calidad es una parte importante del proceso de aprendizaje de programación.

Creo que mejorar la forma de leer código puede significar beneficios en ámbitos laborales, no sólo de aprendizaje. Si bien es cierto que es raro que un programador lea código ajeno en sus momentos ociosos con el único objetivo de leer código de calidad, hay muchas situaciones laborales que pueden ser beneficiadas por herramientas que faciliten la lectura, como por ejemplo:

- Aprender sobre librerías ajenas, lo cual puede ser necesario debido a mala documentación de las mismas
- Encontrar bugs, debido que todos usamos código escrito por terceros en nuestros proyectos
- Agregar features a proyectos ajenos
- Volver a trabajar sobre proyectos propios antiguos, debido a que al pasar el tiempo puede ser dificil comprender hasta el código propio
- Hacer Code Review
- Buscar material para reusar en código ajeno

Sin embargo, una cosa es segura: **Leer código es dificil**. Por eso voy a encarar este problema como inicio para la startup que detallaré en este Trabajo Final.

Visión

Una visión tiene que ser amplia porque todavía no hay nada definido en el modelo de negocios. Si definimos un objetivo muy específico seguramente el mismo va a cambiar en el ciclo de vida de una startup, sin embargo una visión a largo plazo seguirá vigente.

La visión es la siguiente:

> *Aumentar la capacidad de entender código*

También, después de incontables semanas de brainstroming, elegí el nombre de la startup que trataré por el resto del Trabajo Final:

[sonido de redoble de tambor]

DIP

Hay que empezar por un objetivo más alcanzable que la visión propuesta debido a la magnitud de la visión y al carácter iterativo de la metodología de Desarrollo de Clientes. Por este motivo a continuación mostraré cómo deconstruir la visión en una solución al problema de lectura de código.

Solución

En el planteo de problema enumeré motivos por el cuál leer código. Para proponer una solución al problema planteado voy a pensar en cómo se lee código actualmente. Finalmente enumeraré los features que la solución ideal debería tener.

A la hora de pensar cómo leer codigo, se puede armar una extensa lista de posibilidades. Siempre depende del objetivo que cada uno tenga para leer código, pero a nivel general algunas formas de leer son:

- Empezar la exploración por el punto de partida del programa, por ejemplo un archivo main
- Pensar en qué se quiere aprender, buscarlo con herramienta tipo grep y usarlo como punto de partida de la exploración del código
- Buscar palabras clave del lenguaje en el que el código está escrito
- Buscar declaraciones de funciones u objetos interesantes
- Ver estructura del código
- Encontrar comentarios que ayuden a entender fragmentos
- Analizar el código con un profiler y ver donde tarda mas, y usarlo como punto de partida

Teniendo en cuenta la forma en la que se lee el código actualmente, a continuación daré una lista de historias de usuario para la solución propuesta.

Como usuario, quiero:

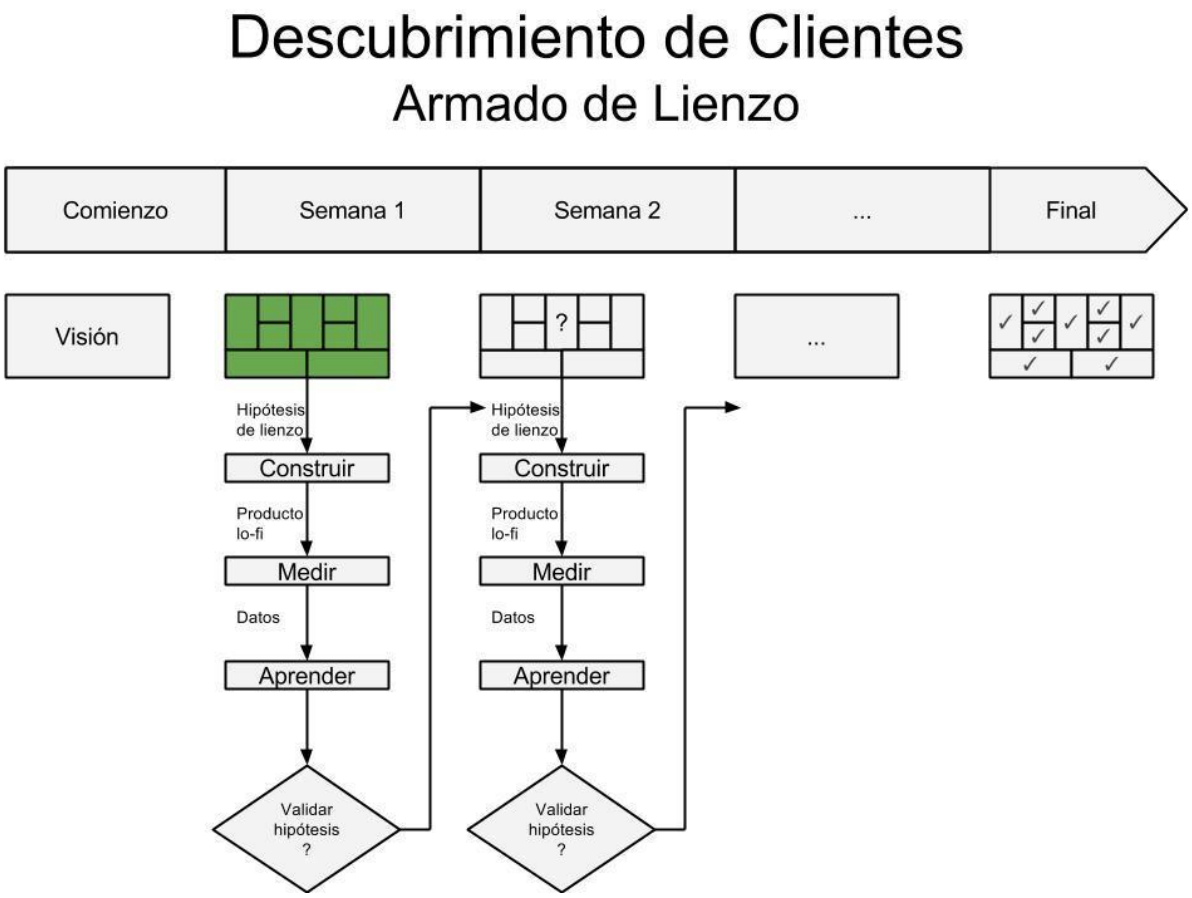
- Seleccionar código y hacer preguntas sobre él en stack overflow
- Subrayar código que me parezca importante
- Tomar notas sobre fragmentos de código y hacerlas visibles para los otros usuarios
- Hacer diagramas sobre fragmentos de código
- Correr fragmentos de código en intérpretes del mismo y ver el resultado
- Buscar el uso de palabra clave en el código
- Seleccionar código y ver la documentación asociada
- Hacer guías paso a paso sobre el código para que otros vean
- Escribir tests para fragmentos del código
- Refactorizar fragmentos de código
- Armar versiones simplificadas del código

- Remover una implementación y reimplementarla hasta que los tests pasen de nuevo

Es oportuno aclarar que las previas historias de usuario son para la versión ideal de la aplicación. Esta solución ideal se plasmará en la sección de Propuesta de Valor del Lienzo de Modelos de Negocios que se construirá en la próxima sección, para después ser validada o rechazada a través del feedback de clientes.

Una vez que esté determinada la visión, problema y solución de la startup hay que realizar la construcción del Lienzo de Modelos de Negocios, lo cual es el objetivo del siguiente capítulo.

Lienzo



Lienzo

Después de hablar sobre el Lienzo de Modelos de Negocios finalmente llegó la hora de dar un ejemplo de construcción de lienzo. En este capítulo deconstruiré la solución planteada anteriormente en un modelo de negocios, descripto en las próximas páginas a través de sumario sobre cada elemento que lo compone.

Diseñaré el modelo de negocios partiendo del público objetivo. Para esto usaré una herramienta llamada Empathy Map, desarrollado por la compañía XPLANE [1] y referenciada en el libro "Business Model Generation" [2]. Partir del público objetivo permite construir un modelo de

[1] http://xplane.com/

[2] http://www.businessmodelgeneration.com/

negocios mas fuerte porque el perfil del cliente guía el diseño de mejores Propuestas de Valor, Canales para llegar a él, y mejores Relaciones con Cliente.

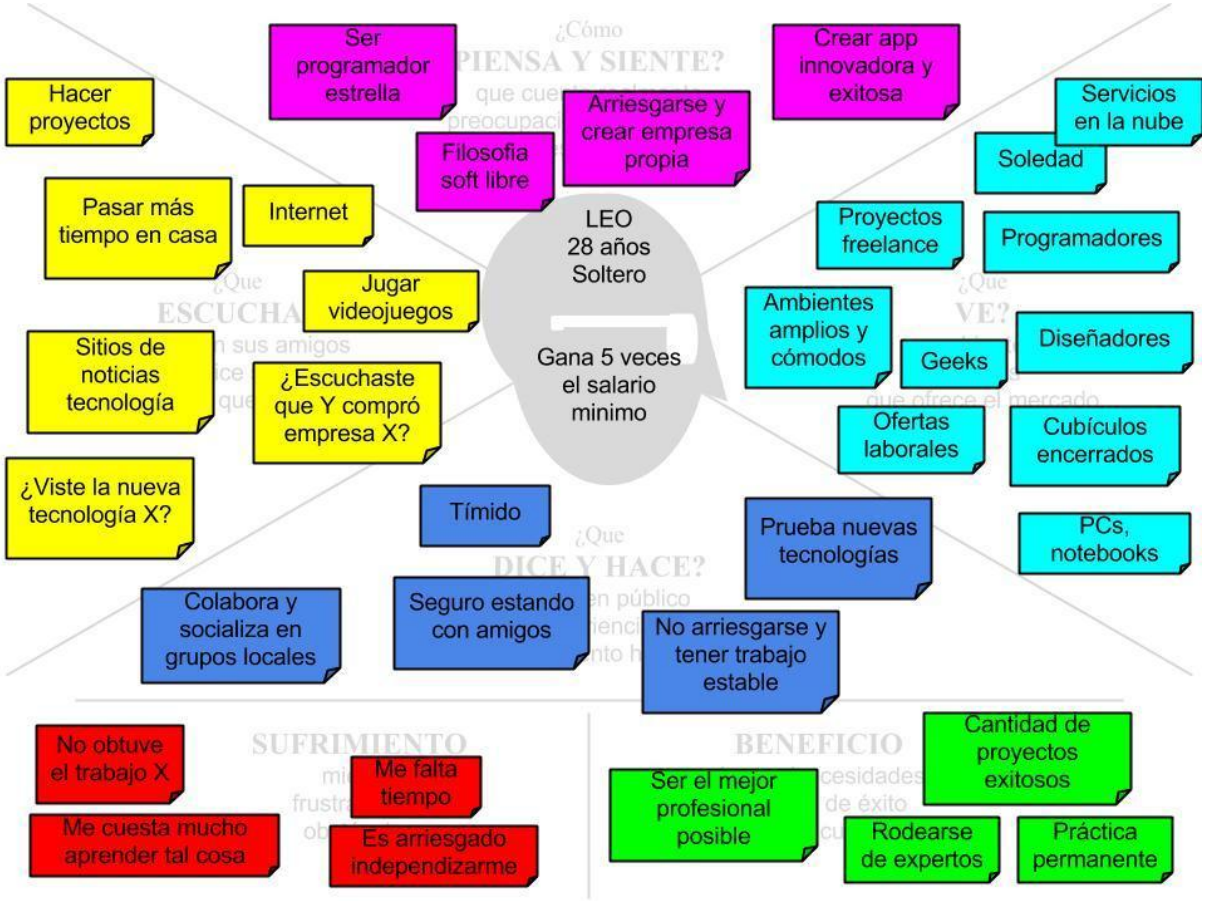
La forma de diseñar el mapa de empatía de nuestro cliente objetivo es la siguiente:

1. Dar al cliente un nombre y características demográficas, como salario, estado civil, etc.
2. Refiriéndose al siguiente diagrama, usar un pizarrón o cartulina para construir un perfil del cliente recientemente nombrado preguntando y respondiendo las preguntas del diagrama.

Mapa de Empatía

Mapa de empatía de DIP

El mapa de empatía realizado para DIP quedó asi:



Mapa de Empatía DIP

Conocé tus usuarios como a vos mismo

Lo mejor que puede pasar a la hora de diseñar partiendo del público objetivo es ser parte del mismo. Por este y muchos otros motivos es fácil darse cuenta por qué es beneficioso construir una startup y a la vez ser el usuario objetivo de la misma.

Llegó el momento de describir los nueve bloques del modelo de negocios de DIP.

Segmento de Clientes

DIP está orientado a programadores web, entre 18 y 30 años que quieran mejorar continuamente como profesionales. La mayoría de las personas a las que apunto son solteros, están trabajando y ganan 5 veces el salario mínimo.

Propuesta de Valor

Usando DIP van a comprender fácilmente proyectos open source, así pueden contribuir a los mismos. Ofreceremos este entendimiento a través de guías digitales llamados “dips” que analizan código fuente de proyectos open source, y guían al usuario a través del código.

Canales

La forma de distribución del servicio es a través de apps para smartphones y tablets, y a través del sitio dip.io.

Relaciones con Clientes

Mantendremos la relación con nuestros clientes a través de un blog técnico que sea de interés para ellos, y también con la creación de comunidades de contribuidores de proyectos libres.

Flujos de Ganancias

Utilizaré un modelo de suscripciones mensuales, el cual dará a los usuarios acceso completo a todos los dips.

Recursos Clave

Los elementos claves que hacen funcionar al negocio son tres: los usuarios, los dips que les proveen valor y el hecho de que todo el software utilizado por la startup sea código libre.

Actividades Clave

El armado de dips por especialistas, idealmente expertos en el código; y el desarrollo de las apps son las actividades clave.

Sociedades Clave

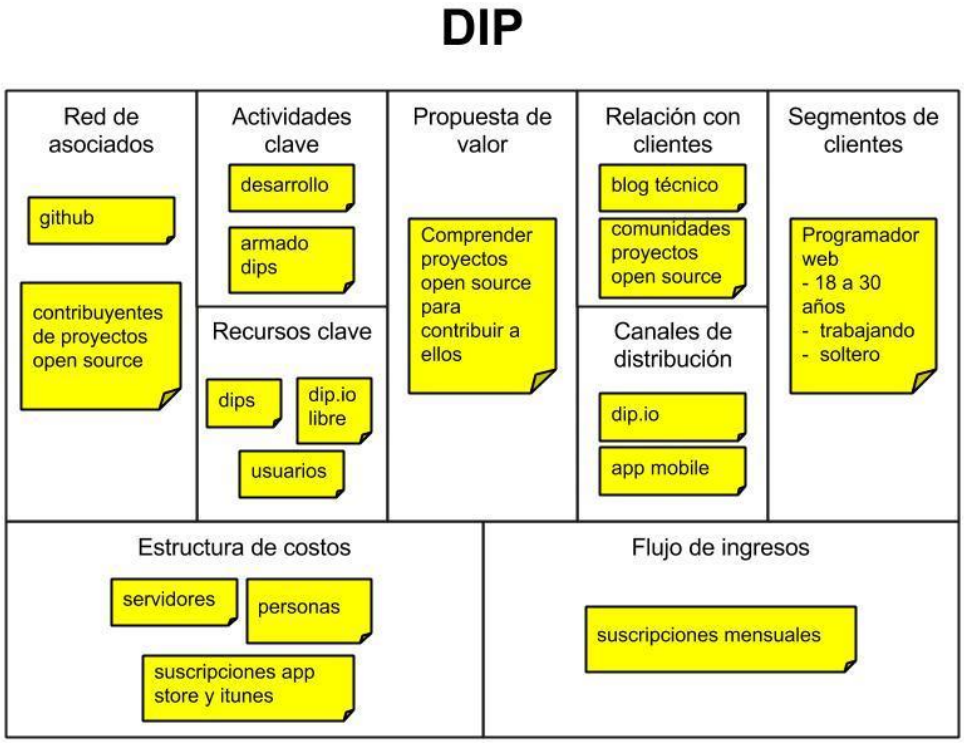
Github es un socio clave porque prácticamente todo proyecto open source está hosteado allí, y proveen una API para interactuar con ellos.

Estructura de Costos

La startup cuenta con costos típicos de una empresa web/mobile: servidores, suscripción a la Play Store de Google y a iTunes de Apple. También hay que incluir a los salarios de personas y suscripciones a servicios y herramientas de desarrollo.

Lienzo de DIP

Sintetizando el modelo de negocios de DIP en una sola imagen:



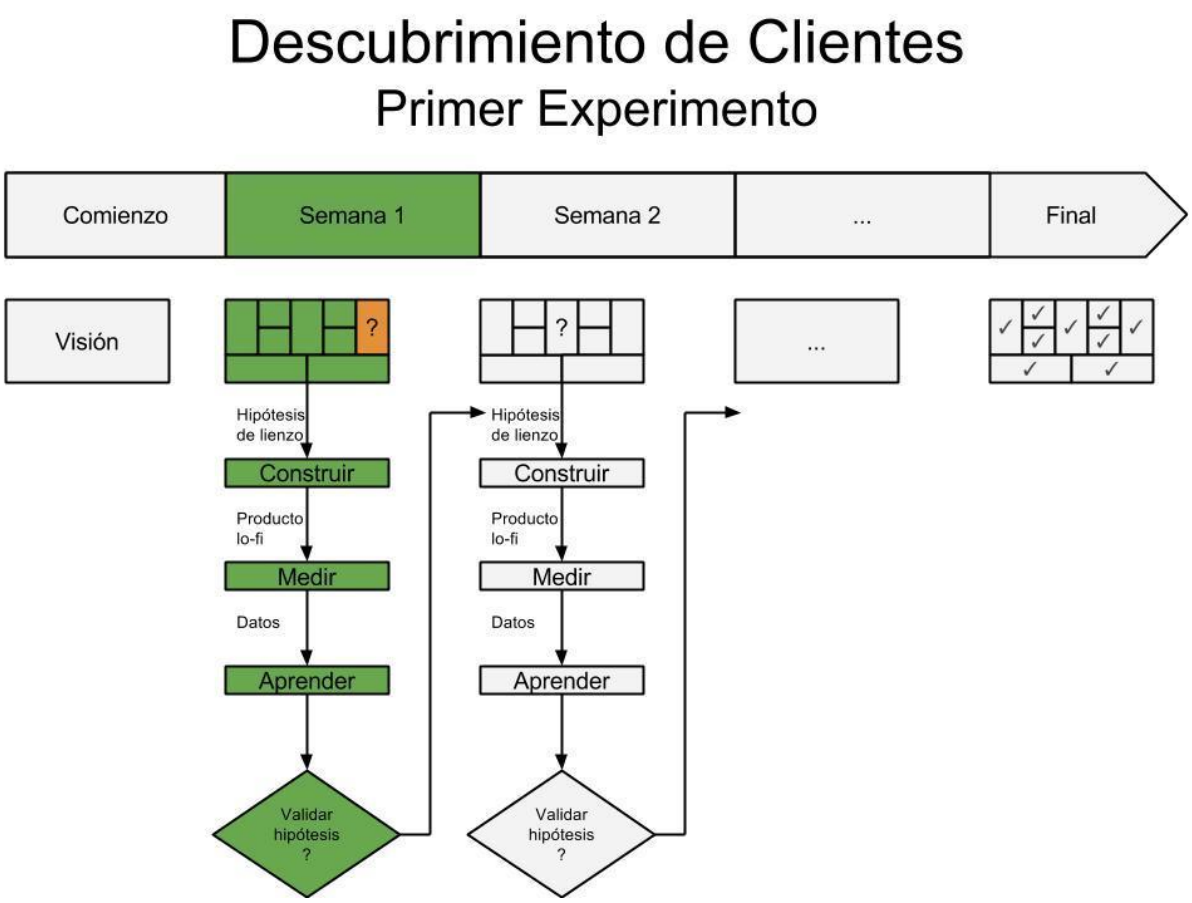
Lienzo dip

Es importante recordar que este modelo de negocios es solo una primera aproximación llena de suposiciones, que pueden ser acertadas o erradas, y sólo sabremos eso con el Desarrollo de Clientes.

Resúmen

Con una visión formada y un lienzo de modelo de negocios listo, no queda nada que impida el comienzo de la parte mas emocionante: armar el producto y salir del edificio. En el próximo capítulo comenzaré con la primera fase de construcción.

Primer Experimento



Primer experimento

Finalmente llegó el momento de empezar la validación del modelo de negocios, y convertir todas las suposiciones en certezas. Este proceso generalmente va a ser largo y laborioso, a menos que seas parte de la minoría que saca la lotería de usuarios (Facebook, Instagram, Twitter, etc).

Hipótesis

¿Por dónde empezar? Tenemos un lienzo de negocio lleno de hipótesis sin validar, sin embargo generalmente el punto de partida será siempre el mismo: hay que tener certeza que la startup encara un problema real de los clientes.

Por lo tanto la primer hipótesis en la que trabajaré va a ser la del Segmento de Clientes:

> Hay programadores web que quieren contribuir a proyectos open source, pero les resulta difícil comprender el código de los mismos.

Construir

¿Cómo validar esta hipótesis? Llegó la hora de construir un producto que valide la hipótesis planteada. Pero es importante tener en cuenta que se construirá un MVP—un producto que tenga la funcionalidad mínima y necesaria para probar la hipótesis—no hay que desperdiciar recursos en productos con funcionalidad innecesaria por ahora.

Generalmente los ingenieros tenemos la tendencia de construir soluciones y que éstas sean perfectas hasta el más mínimo detalle. Esta forma de pensar nos puede llevar a construir soluciones perfectas que nadie usa porque no solucionan ningún problema real.

Sin embargo no todo es programar, es posible que haya etapas de Construcción que no involucren código, por ejemplo:

- podríamos armar un video que plantee el problema a solucionar y después medir su éxito
- podríamos relizar entrevistas con posibles usuarios y detectar si realmente cuentan con el problema planteado en la hipótesis
- podríamos hablar con potenciales clientes y así entender qué necesitan realmente

También está la opción de escribir sólo el código necesario, esto puede ser una simple pero efectiva página web que plantee el problema a validar y mida el éxito de la misma para validar o no la hipótesis. Ése es el MVP que voy a construir en esta etapa: una página de marketing.

MVP Construiré una página que despierte el interés del público objetivo y recolecte emails de interesados.

A pesar de la simpleza de la solución planteada, hay que asegurar que la recolección de emails funcione perfectamente. Para esto armaré un conjunto de pruebas funcionales y unitarias que verifiquen el óptimo funcionamiento de la página.

A continuación describiré el proceso de construcción del MVP.

Construcción de MVP

La arquitectura de la página es un simple y efectivo stack de javascript en el cliente, servidor y base de datos:

- Cliente desarrollado con framework AngularJS
  - hosteado en github gratis
- Servidor de API en NodeJS
  - desacoplado del cliente para mejorar modularidad
  - hosteado en heroku gratis
- Base de datos MongoDB, elegida por
  - javascript
  - libre de esquemas, lo cual es ideal para una startup que siempre cambia
- Servidor CI
  - Travis, gratis para open source

Arquitectura cliente

- Controlador
- Vista
- Modelo

Arquitectura servidor

- REST

Desarrollo TDD

- E2E
- Unitarios

Medir

- emails
- cuantos considerariamos exito?

Aprender

- analizar resultados
  - validar
  - seguir iterando

Paso siguiente

- Conexión con experimento 2: por fin solucion!