laSalle
UNIVERSITAT RAMON LLULL

Escola Tècnica Superior d'Enginyeria La Salle

Trabajo Final de Máster

Máster Universitario en Ingeniería de Telecomunicación

Kynda. El lanzamiento de una Startup: idea, equipo, ejecución y validación.

Nombre Alumno

André F. Ribeiro Caçador

Nombre Profesor Ponente

Miguel Ramírez

ACTA DEL EXAMEN
DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en la fecha indicada, el alumno

D. André Filipo Ribeiro Caçador

expuso su Trabajo Final de Máster, titulado:

Kynda. El lanzamiento de una Startup: idea, equipo, ejecución y validación.

Acabada la exposición y contestadas por parte del alumno las objeciones formuladas por los Sres. miembros del tribunal, éste valoró dicho Trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

EL LANZAMIENTO DE UNA STARTUP: IDEA, EQUIPO, EJECUCIÓN Y VALIDACIÓN

MET
LA SALLE, UNIVERSIDAD RAMON LLULL

BARCELONA, ENERO 2018

Titulación: Máster Oficial en Ingeniería de Telecomunicaciones
Autor: André Ribeiro Caçador
Tutor: Miguel Ramirez

Resumen

Kynda es una startup que ha creado un marketplace móvil y web que permite a los usuarios reservar servicios de bienestar y belleza en un domicilio, hotel o apartamento turístico.

Tenemos una red de profesionales autónomos seleccionados que se adaptarán a las necesidades del usuario, nuestros profesionales trabajan en función de la demanda siendo mucho más productivos que el modelo tradicional de jornada laboral (9h-20h).

La aplicación se ha lanzado en la ciudad de Barcelona.

Se explicará todo el proceso que se ha seguido para crear y ejecutar el proyecto, se explicará desde el origen de la idea hasta etapa actual de lanzamiento y crecimiento con métricas de facturación, usuarios, retención, etc.

Me centraré principalmente en el producto tecnológico detallando todas las decisiones tecnológicas y su posterior ejecución ya que ha sido mi principal responsabilidad en el proyecto.

También explicaré aspectos a nivel organizativo, y de cómo fomentar la creatividad y motivación en la startup.

Palabras clave

startup, aplicación, web, móvil, programación, backend, frontend, servidor, ios, android, swift, golang, javascript, vuejs, base de datos, mysql, producto, lean startup, agile, kanban, marketing digital, métricas, KPIs

Índice

1. Introducción

Actualmente Barcelona es uno de los ecosistemas más atractivos de startups de Europa, especialmente en el campo de las nuevas tecnologías de internet como e-commerce, marketplaces, big data, eHealth, automoción y finanzas entre otros. Gracias a diversos factores que otorgan a la ciudad con más de cientos de startups con una visión ambiciosa y global.

En este momento el ecosistema cuenta con más de 10000 profesionales repartidos en las más de 1000 startups existentes, la ciudad se ha convertido en un punto de referencia en Europa en la creación de nuevas compañías de base tecnológica.

Los elementos que han contribuidos a este crecimiento han sido infraestructuras locales que se benefician y ofrecen recursos a los emprendedores, ayudas públicas y privadas del Estado, emprendedores de éxito que han ayudado a crear un nicho de aceleradoras, incubadoras y vehículos de inversión para las nuevas compañías.
La presencia de grandes corporaciones y organizaciones mundiales también a ayudado como el Mobile World Congress, 4YFN, Smart City Expo, IOT, etc.

Este auge lo he vivido desde dentro ya que en enero de 2017 decidí fundar una compañía en Barcelona. Se trata de Kynda, un marketplace móvil y web de servicios de belleza y bienestar a domicilio. Esta startup se creó dentro de una factoría de startups llamada Venture Builder (VB). El concepto VB es bastante novedoso, los primeros se crearon hará unos 7 años atrás. Su función es identificar y analizar modelos de negocio que estén funcionando en en otros mercados para replicarlos en otros donde no existen. Kynda es una copia de un modelo de negocio que está funcionando en Estados Unidos (Glamsquad) y Londres (gopriv) que han sido capaces de atraer mucha inversión y se ha demostrado que funcionan.

A nivel Europeo, el VB más conocido es Rocket Internet, un grupo de Berlín del que han salido compañías de éxito como Just Eat, Delivery Hero, Food Panda, etc… Han creado más de 100 compañías en todo el mundo que suman una valoración de 15Bn de €.
En España, el más conocido es Antai VB, mucha gente no lo sabe pero ellos están detrás de importantes startups como Wallapop, LetsBonus, Cornerjob, Glovo, Carnovo, etc.
De media tardan unos 3 meses en crear una nueva startup, son expertos en captar grandes montos de inversión en estas etapas tan iniciales.

Kynda se ha creado dentro de la estructura de Nuclio, un VB muy reciente creado por el emprendedor de éxito Carlos Blanco después de vender su anterior compañía, Akamon, por varios millones de €.

Hay diferencias en la metodología que ejecución de cada Venture Builder. En el modelo de Antai un mismo equipo es el que crea el primer producto de la startup, y una vez lo han

probado en el mercado consiguiendo buenas métricas captan talento para que continúe con la startup. En cambio, Nuclio (el VB de Kynda) selecciona las ideas y organiza un evento de fin de semana para formar esos equipos de talento que las ejecutarán en paralelo. En la conclusión detallaré mi opinión sobre estos modelos ya que he sido parte de uno de ellos, los pros y contras de cada modelo.

2. Objetivos

A lo largo del proyecto los objetivos han ido cambiando como es normal en una startup. Al inicio se decidió crear una primera versión del producto centrada en una vertical de servicios, la de fisioterapeutas. Pero en marzo, sin aún haber lanzado el producto decidimos pivotar a un marketplace multicategoría de servicios de estética, peluquería, nutrición, entrenadores, entre otros. Se tomó esta decisión porque el concepto no era atractivo para inversión después de varias reuniones con inversores privados para conocer su interés en el proyecto.

A partir de abril, debido al cambio de producto, tuvimos que cambiar en bastantes aspectos por lo que se alargó la fecha de lanzamiento hasta septiembre, después del verano.
En mis anteriores trabajos aprendí muchísimo a nivel técnico para poder afrontar este reto, quería crear un negocio digital desde cero y aquí se me presentaba la oportunidad. A nivel de lógica de servidor, sabía crear una API que proveyese de datos a un cliente, sea web o móvil. También aprendí a construir el cliente, conocía tecnologías WEB y iOS.

Mi objetivo profesional dentro de la startup era crear un producto tecnológico fantástico en 4 o 5 meses, dado mis conocimientos y nuestro target de cliente decidimos para el primer prototipo realizar una aplicación para iOS. También necesitábamos crear otra aplicación para nuestros proveedores, los profesionales necesitaban gestionar su agenda y las citas con los clientes.

El objetivo a nivel equipo era lanzar las aplicaciones en el mercado locales de Barcelona para obtener las primeras métricas y validar nuestro modelo de negocio para captar más inversión y poder seguir creciendo hacia otras ciudades.

3. Alcance

En este trabajo se detalla todo el trabajo que se ha tenido que realizar para ejecutar el proyecto de Kynda centrándome más a nivel de producto y técnico ya que ha sido mi principal responsabilidad en el proyecto.
También se explicará la metodología, organización y la estrategia de marketing. Se explicará todo el aprendizaje a nivel personal, hablaré sobre los aciertos y los errores cometidos.
Intentaré profundizar en todos los factores que intervienen en la creación de un negocio digital, y de como todos son importantes para el éxito de la startup.

4. Idea

Como propuesta de proyecto, se quiere crear un marketplace[1] de servicios de belleza y bienestar a domicilio, sea en casa, hotel u oficina. Gracias a la geolocalización, proponemos al usuario de nuestra aplicación servicios de profesionales que estén cerca, disponibles y puede desplazarse hasta su domicilio en un corto periodo de tiempo. Se ofrece un gran abanico de servicios como fisioterapia, peluquería, maquillaje, estética, entrenamiento personal, etc.

Como en cualquier marketplace, debíamos equilibrar la oferta y la demanda, esto se traducía en que en realidad teníamos dos tipos de cliente, el consumidor y el proveedor.

Nos debíamos inspirar en modelos de negocio similares que estaban funcionando en Estados Unidos y Londres, además empezaban a aparecer en España alguna startup queriendo hacer lo mismo que nosotros en Madrid, asique el tiempo de lanzamiento era fundamental.

Incluso la idea inicial iba mucho más allá, nos dimos cuenta de que la forma de consumir estos servicios era ineficiente para el consumidor. La mayor parte de centros están cerrados cuando el posible cliente tiene tiempo para ir, se solapan las horas de trabajo con las horas de abertura de un centro. Los centros tienen picos muy grandes de trabajo los fin de semana donde es necesario contratar más personal para dar abasto, y durante la semana los centros y salones de belleza están desiertos.
Se concibió Kynda pensando en crear la red de profesionales de belleza y bienestar más grande de Europa siendo capaz de proveer profesionales en horas de máxima afluencia tanto a centros como a personas (B2B y B2C).

También era una propuesta de valor para nuestros profesionales, para poder conciliar la vida familiar con la profesional pudiendo aprovechar de manera óptima su tiempo.

La plataforma no supondrá ningún coste fijo para el profesional, nuestro modelo de negocio se basaba en la aplicación de una comisión sobre el servicio contratado. Dependiendo de la valía del profesional, esta comisión variaba del 20% al 10%.

(1) Sitio en Internet donde se llevan a cabo interacciones comerciales entre diferentes empresas o personas.

5. Análisis

En este apartado realizaremos un estudio fundamental para conocer la viabilidad del proyecto que toda persona debería hacer en el momento que tiene una idea, es importante entrar en profundidad en aspectos como el tamaño del mercado, conocer tu competencia, socios estratégicos, capital necesario, etc. No debe ser un plan de negocio, ya que tenemos muchas hipótesis que resolver, lo que se suele utilizar es un modelo de negocio con una serie de preguntas que debemos resolver para determinar si seguir adelante con la ejecución.

5.1. Modelo de negocio

1. Propuesta de valor
    a. ¿Qué valor estamos entregando a los clientes?
    La posibilidad de solicitar un servicio de belleza o bienestar profesional a su casa, oficina u hotel.
    Conocer a los mejores profesionales de su ciudad, gracias a opiniones de otros clientes, además todos los profesionales han sido previamente seleccionados y validados.
    b. ¿Qué problema estamos ayudando a resolver?
    Hay personas que no disponen de tiempo para reservar, hacer colas de espera o desplazarse hasta el centro.
    Muchas personas no encuentran centros abiertos por incompatibilidades con su jornada laboral.
    Otras prefieren que el profesional vaya a casa, sea por pereza o por problemas en su condición física.
    c. ¿Qué necesidad estamos satisfaciendo?
    Los clientes necesitan encontrar un sitio de confianza dónde contratar profesionales cualificados que realicen servicios a domicilio.
    d. ¿Qué paquete de productos/servicios estamos ofreciendo a cada segmento de clientes?
    Tenemos 9 categorías diferenciadas: fisioterapia, manicura, maquillaje, tratamientos corporales, tratamientos faciales, peluquería, nutrición, entrenadores personales y yoga.

2. Actividades clave
    a. ¿Para nuestra propuesta de valor?
    Creación de plataforma digital que facilite la intermediación entre clientes y profesionales.
    b. ¿Para nuestros canales?
    Para el canal digital, diseñar una buena estrategia de marketing con un equipo con conocimiento del sector.
    c. ¿Para nuestras relaciones con los clientes?

Conseguir una excelente experiencia de usuario durante todo el proceso, desde que se instala la aplicación hasta que acaba el servicio realizado por el profesional.

d. ¿Para nuestras fuentes de ingresos?
Captar clientes para que prueben el servicio, y más importante, que repitan.

3. Recursos clave
   a. ¿Para nuestra propuesta de valor?
   Reclutar profesionales cualificados y un equipo que desarrolle un buen producto.
   Debemos equilibrar la oferta y la demanda procurando tener siempre profesionales disponibles a las horas de más demanda.
   b. ¿Para nuestros canales?
   Reclutar un equipo de marketing con experiencia en el sector.
   c. ¿Para nuestras relaciones con los clientes?
   Comprar material de trabajo estándar de buena calidad para nuestros profesionales.
   d. ¿Para nuestras fuentes de ingresos?
   La plataforma tecnológica tiene que estar bien implementada y el servicio debe estar operativo siempre.

4. Socios clave
   a. ¿Quiénes son nuestros socios clave?
   Hoteles, apartamentos turísticos y empresas.
   Tiendas de cosmética, proveedores de productos de belleza,
   b. ¿Quiénes son nuestros proveedores clave?
   Profesionales autónomos y centros que quieran ofrecer sus servicios a domicilio, además debemos contratar personal en plantilla para cubrir la demanda.
   c. ¿Qué recursos clave estamos adquiriendo de nuestros socios clave?
   La oferta de la plataforma.
   d. ¿Qué actividades realizan nuestros socios clave?
   Son lugares idóneos donde sus clientes pueden pedir servicios en nuestra plataforma.
   También forman parte del sector bienestar.

5. Relación con clientes
   a. ¿Qué tipo de relación espera que establezcamos y mantengamos cada uno de nuestros segmentos de clientes?
   Debemos crear esa relación en entornos de confianza ya que el cliente debe permitir que una persona desconocida realice el servicio en su domicilio.
   b. ¿Qué relaciones establecemos?
   Se quiere crear eventos relacionados con el bienestar y el cuidado del cuerpo donde se realizarán servicios para promocionar Kynda y sus profesionales.

Se promocionará Kynda en lugares de confianza para el cliente como tiendas de cosmética, hoteles, eventos, etc.

c. ¿Cuán costosas serán?
Tendrán un coste de 1000€ a 3000€ en algunos casos pero es fundamental hacerlo porque debemos generar confianza al principio.
d. ¿Cómo se integran con el resto de nuestro modelo de negocio?
Las acciones de marketing digital que se realicen deben ir acompañadas de una acciones homólogas offline. La partida destinada a marketing de Kynda debe ser grande, es un modelo de negocio que lo necesita.

6. Canales
   a. ¿A través de qué canales nuestros segmentos de clientes pueden ser alcanzados?
   Digitales
   Facebook Ads
   Google Adwords
   ASO, SEO, SEM

   Offline
   Flyers y eventos

7. Segmentos de clientes
   a. ¿Para quién estamos creando valor?
   Distinguimos 4 segmentos:
      1. Healthy
      2. Acomodada sin tiempo
      3. Chica que le encanta la cosmética y belleza.
      4. Persona con movilidad reducida.

   b. ¿Quiénes son nuestros clientes más importantes?
   Los dos primeros por nivel adquisitivo, gastarán más.

8. Estructura de costes
   a. ¿Cuáles son los costes más importantes en nuestro modelo de negocio?
   Producto y marketing.
   b. ¿Qué recursos clave son los más costosos?
   Campañas de marketing.
   c. ¿Qué actividades son las más costosas?
   Captar clientes.

9. Fuentes de ingresos
   a. ¿Por cuál valor nuestros clientes están dispuestos a pagar?

Calidad, buenos profesionales, precios competitivos, e inmediatez del servicio.

b. ¿Actualmente por qué se paga?
   Calidad del servicio, precio y proximidad del centro.
c. ¿Cómo prefieren pagar?
   Cada vez los usuarios están más acostumbrados a pagar a través de una plataforma digital.
d. ¿Cuánto aporta cada fuente de ingresos?
   A priori, nos centramos en el modelo B2C (business to customer) pero también se quiere trabajar el modelo B2B (business to business) con empresas y hoteles para que ofrezcan nuestros servicios.

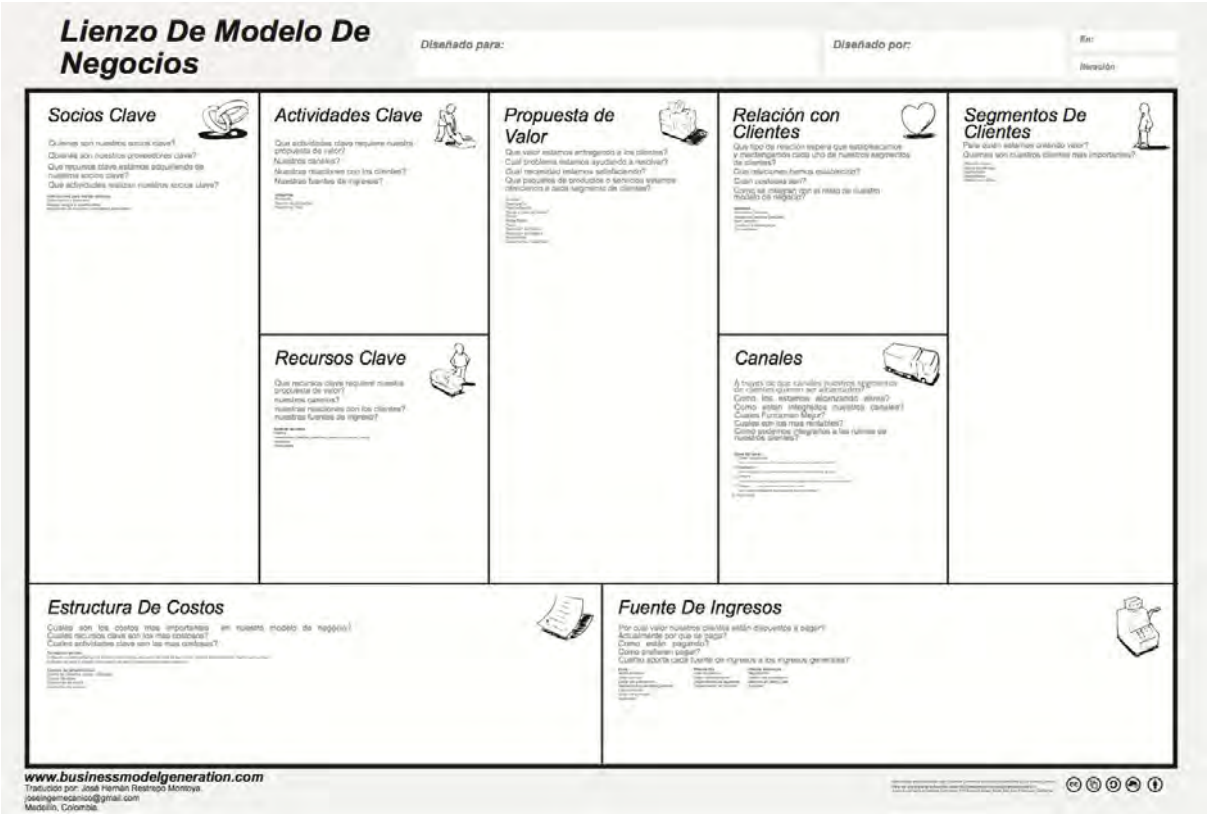


Tabla típica de generación del modelo de negocio

5.2. Tamaño de mercado

El mercado global de la belleza y bienestar tiene un tamaño de 3.7 trillones de dólares (2015). Además, España es un país en el que el sector del turismo es muy importante, su economía se basa en el sector terciario por lo que es un buen mercado para debutar con nuestra propuesta de valor.

5.3. Competencia

Directa

1.1. Internacional
- 1.1.1. Glamsquad, USA
- 1.1.2. Gopriv, London

1.2. Nacional
- 1.2.1. Styleprive, Madrid
- 1.2.2. Urvan, Madrid
- 1.2.3. Curlerapp, Barcelona
- 1.2.4. Yatepeino, Madrid

Indirecta

Plataformas de reserva en un centro de belleza con gran penetración de mercado en España.

a. Treatwell
b. Bucmi

Durante todo el proceso de creación de la startup se debe tener monitorizada la competencia y estar atentos al más mínimo cambio tanto en producto, como en comunicación o marketing.

6. Equipo

El equipo fundador de una startup debe ser multidisciplinar. Ni deben ser todos ingenieros, ni todos financieros. Para mí todas las funciones sea producto, marketing, desarrollo de negocio,... son fundamentales para el éxito de la compañía.

Al principio, en kynda se cometió el error de crear un equipo sobredimensionado y con solapamientos en aptitudes. El equipo era formado por Nuclio, a raíz del evento que organizan para presentar sus proyectos y seleccionar a las personas que lo ejecutarán, con lo que es difícil formar equipo con personas que jamás has trabajado en un entorno real y no conoces.

Como consecuencia de ese error, el abril después de 3 meses de trabajo en equipo con muchas cambios organizativos y de modelo de negocio, Nuclio que es era socio mayoritario decide prescindir de todos los fundadores menos de mí.

A partir de mayo estoy liderando el equipo, en ese momento se incorporó Ekaterina para realizar la función de desarrollo de negocio para empezar a captar los profesionales que necesitábamos en la plataforma. También incorporamos a Alex, un desarrollador junior que nos ayudó mucho en el desarrollo tecnológico.

A nivel de marketing contábamos con Quim y Carlota, pero eran recursos limitados ya que estaban en diversos proyectos de Nuclio.
Coral como diseñadora era un recurso que teníamos disponible.

7. Metodología

1. Lean Startup

Es un proyecto de estas características, para la producción de la primer prototipo debemos aplicar metodologías startup, Lean startup, para el éxito de la primera iteración del marketplace.
Lean startup ofrece una propuesta para crear y gestionar startups consiguiendo hacer entregables de un producto de una manera rápida a los clientes. Los métodos de Lean startup enseñan como dirigir una startup, a hacer crecer el negocio con la máxima aceleración.

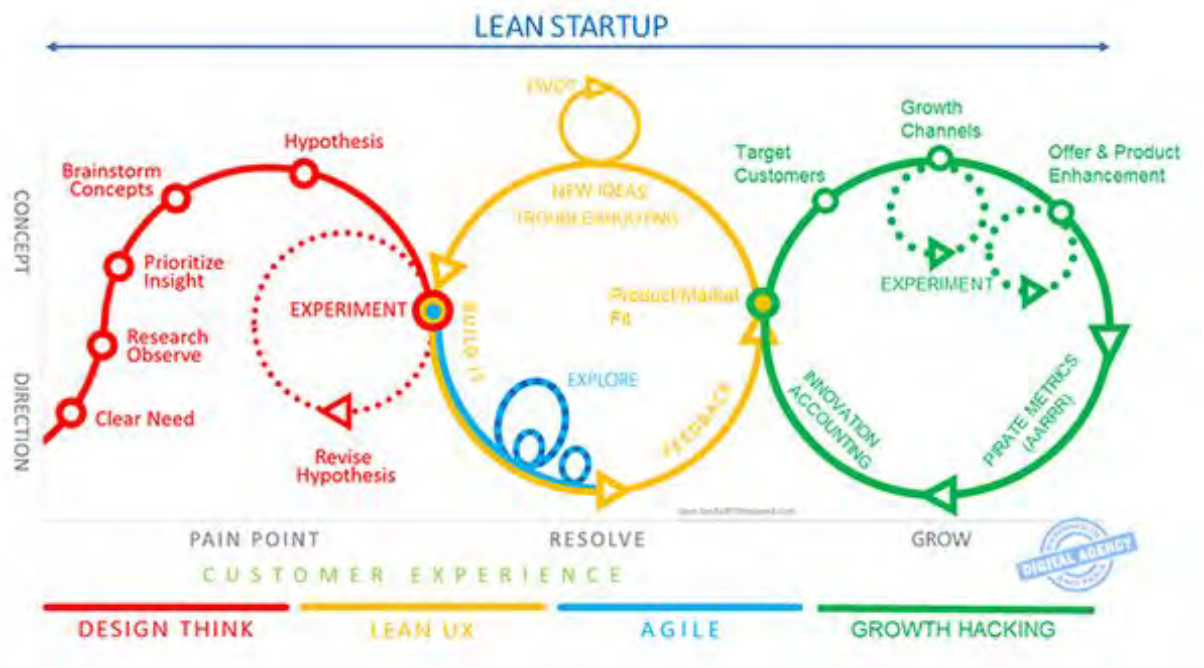
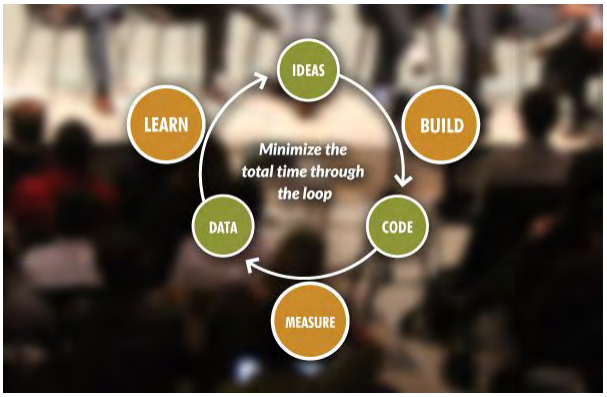


Ilustración de todo el proceso Lean Startup



Se basa en el aprendizaje validado, experimentación científica e iteración en los lanzamientos del producto, lo utilizaremos para acortar los ciclos de desarrollo, medir el progreso y ganar valiosa retroalimentación de los clientes.
En definitiva,

crear - medir - aprender

Este proceso tiene muchas ventajas, primeramente en la etapa de creación se somete el equipo a un trabajo creativo dónde tienen que dar rienda suelta a su imaginación y talento para crear un producto con las características suficientes para darlo a conocer en el mercado, se le llama el producto mínimo viable (MVP, en sus siglas en inglés)
En esta etapa, en referencia al desarrollo de software, la aplicación de Lean startup nos llevará a crear un entorno de producción continua donde todo el código que se desarrolla se despliega en producción de manera inmediata. En este proyecto se ha aplicado, en la parte de cliente móvil no es bien así porque la aplicación necesita pasar por los procesos de revisión de apple store donde se publica pero en la parte de servidor veremos en lo siguiente apartados como en cuestión de minutos el código desarrollado puede estar en un entorno de producción.

2. Producto Mínimo Viable

Para empezar, he realizado un proceso muy recomendado a la hora de definir un producto mínimo viable, se trata del *Product Thinking.*

Básicamente responde a una serie de cuestiones que deben ser planteadas antes de realizar cualquier proyecto.
Las cuestiones son:
1. ¿Qué problema queremos resolver?
2. ¿Para quién estamos haciendo esto?
3. ¿Porqué lo hacemos?
4. ¿Cómo lo vamos a hacer?
5. ¿Qué objetivos queremos lograr?
6. ¿Qué haremos?

Muchas personas cometen el error de definir un producto comenzando por la sexta pregunta: ¿Qué haremos? sin tener en cuenta la gran importancia de las anteriores.

Procedamos a responderlas:

- ¿Qué problema queremos resolver?
  El consumidor no sabe como encontrar profesionales de confianza que se desplacen hasta su posición y de manera inmediata.

- ¿Para quién estamos haciendo esto?
  Para personas que se cuidan mucho, con poco tiempo o con movilidad reducida.

- ¿Porqué lo hacemos?
  Porque creemos que hay una necesidad en el mercado existente.

- ¿Cómo lo vamos a hacer?
  Creando una plataforma digital que tenga un red de profesionales validados para que el cliente pueda elegir.

- ¿Qué objetivos queremos lograr?
  Dar primeros servicios.

- ¿Qué haremos?
  Aplicación para reserva de manera rápida. Modelo transaccional.

8. Ejecución

La producción de una aplicación móvil conlleva la creación de una arquitectura de software compleja, tendremos estructuras con funcionalidades muy concretas.
En este apartado veremos qué tecnologías se han utilizado, las decisiones que se han tomado y su porqué.

1. Requerimientos mínimos

Para el primer prototipo se requiere crear un marketplace móvil similar a los competidores internacionales.
Toda las aplicaciones serán mobile-first dado que los accesos son cada vez mayores por móvil.

Dados mis conocimientos se optó por crear una aplicación nativa para clientes en iOS para validar el modelo de negocio. Era la manera más rápida de crear un producto funcional y testearlo con un grupo de usuarios crítico para continuar iterando hacia una propuesta de valor sólida y coherente. También el usuario de iOS según datos tiene más poder adquisitivo y está más acostumbrado a pagar vía app.

Nuestros “otros” clientes, los profesionales, eran muy importantes para nosotros, necesitábamos mucha oferta en la plataforma y debíamos conseguir que fuera lo más fácil posible para ellos formar parte de nuestra comunidad, por ese motivo se decidió crear una aplicación en Android para profesionales. De esta manera teníamos cubiertos tanto los que tenían dispositivos iOS como Android.

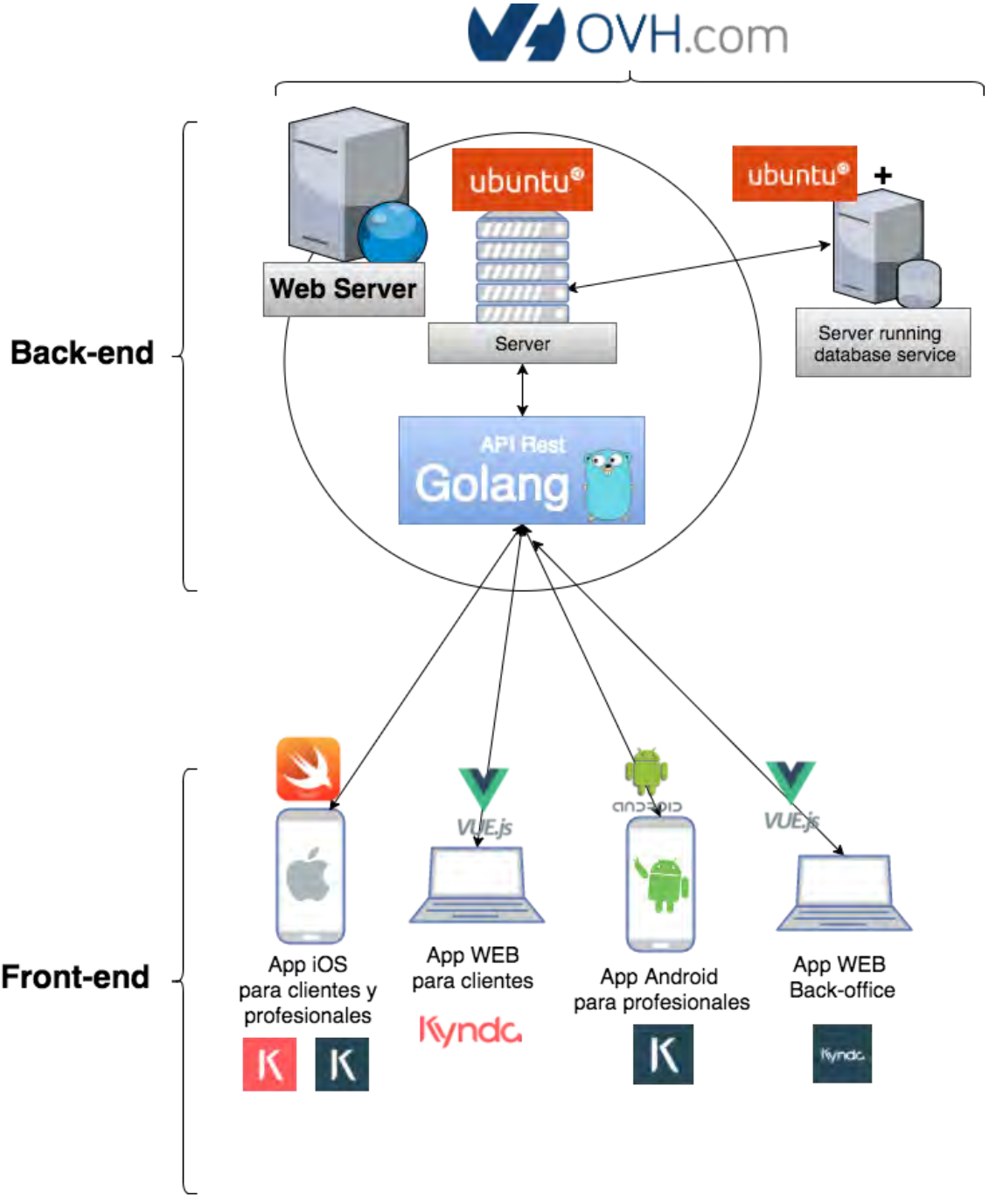
También debíamos crear una herramienta para gestionar todos los datos de la plataforma, un back-office dónde poder consultar, crear o eliminar datos relacionados con los usuarios, profesionales, servicios, pagos, etc.

Una vez lanzada la aplicación de iOS en la App Store, también quisimos realizar una web-app para permitir reservar desde el navegador o dispositivos Android, además era una manera más fácil de acceder a nuestros servicios sin tener que realizar una instalación previa de nuestra aplicación.

2. Diseño

2.1. Arquitectura del sistema

Normalmente la arquitectura de software se separa en front-end y back-end, es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. El front-end es la parte del software que interactúa con los usuarios y el back-end es la parte que procesa la entrada desde el front-end.



2.2. Back-end, la parte de servidor

Se compone por:

- Un servidor web con una API Rest hecha en GOLANG.
- Otro servidor que sirve datos de una base de datos a la API Rest

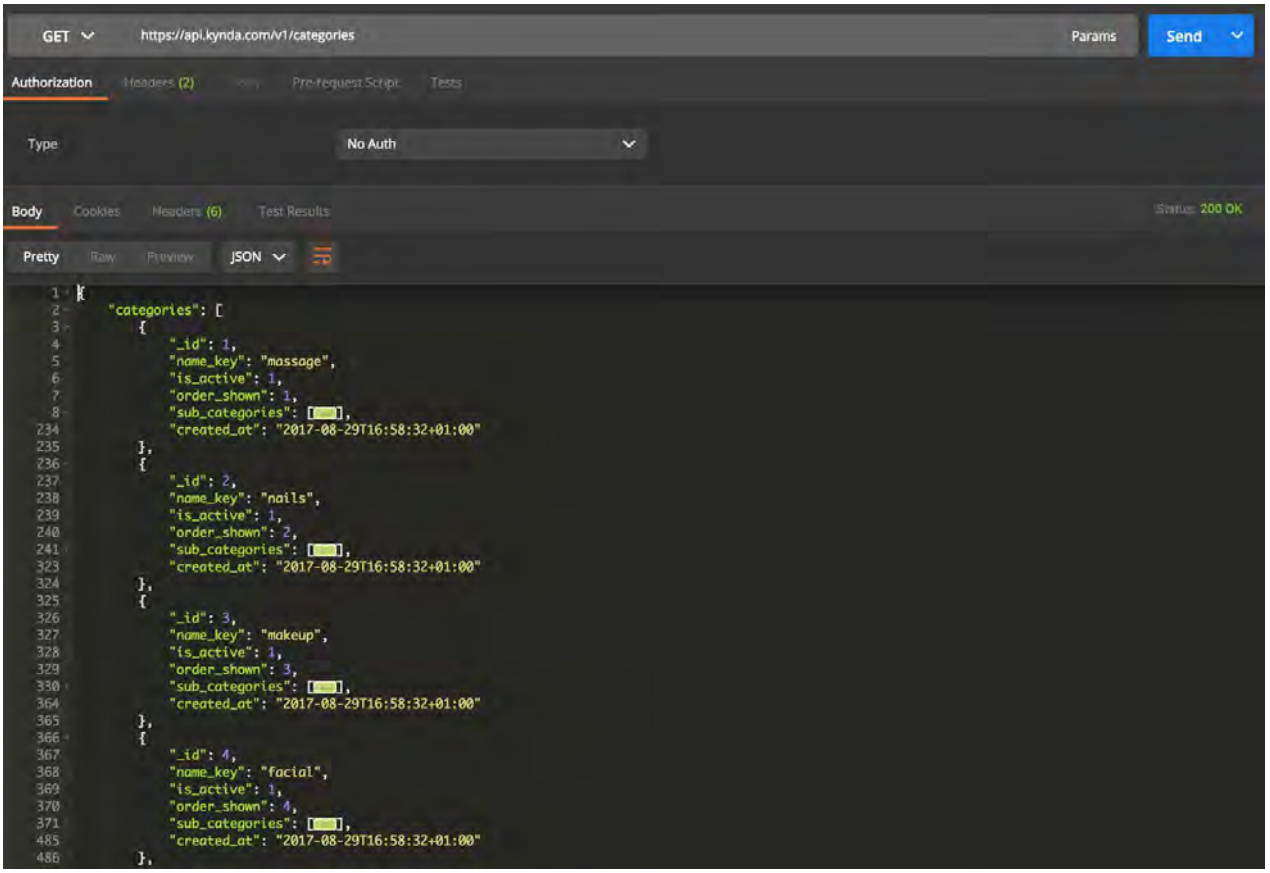
Servidor Web

En nuestro caso, la infraestructura se ha alojado en OVH Cloud Computing.
OVH es una plataforma de computación en la nube. Ofrece servicios a nivel de aplicación (SaaS), servicios a nivel de plataforma (PaaS) y servicios a nivel de infraestructura (IaaS). Nosotros utilizaremos los servicios de plataforma e infraestructura para desplegar el servidor web y base de datos.

Prefiero otras opciones como AWS o Google Cloud, pero se decidió OVH porque nuestro DevOps tenía experiencia con ellos y dominaba la plataforma.

Para servir datos a los clientes, se ha implementado una aplicación web que sirve datos bajo una Application Programming Interface (API) de tipo REST (REpresentational State Transfer) que es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP, nos permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo y actualmente es la más utilizada por las grandes compañías.
Es una arquitectura orientada a recursos, es la información que queremos acceder, modificar o borrar. Esto será importante en nuestras aplicaciones, nuestros recursos son los usuarios, categorias, servicios, pagos, etc..
Las URIs *(Uniform Resource Identifier)* nos permiten acceder a estos recursos, por ejemplo, para acceder a la información sobre el perfil de un usuario haremos la siguiente petición con la siguiente estructura:

{{protocolo}}:// {{hostname}}:{{puerto}}/{{ruta del recurso}}
{{ruta del recurso}} = users/{userID}/profile



POSTMAN. Cliente web para realizar peticiones HTTP. Se podría utilizar cURL también.

En en ejemplo, la API Rest nos devuelve como respuesta una estructura de datos en formato **JSON** (JavaScript Object Notation) con los datos asociados a las categorías del marketplace, el estado de la petición que ha sido un 200 OK. En el caso que se creara un recurso, se haría una petición de tipo POST y recibiríamos en la respuesta un 201 CREATED. Otros métodos que se pueden utilizar son UPDATE o DELETE.

Veremos como se han implementado todas estas funcionalidades en la sección de implementación.
La aplicación que sirve la API Rest se ha desarrollado en GOLANG.

El lenguaje de programación Go es un proyecto de código abierto para hacer programadores más productivos, está inspirado en la sintaxis de C. Ha sido desarrollado por Google y sus diseñadores iniciales son Robert Griesemer, Rob Pike y Ken Thompson.

Go es expresivo, conciso, limpio y eficiente. Sus mecanismos de concurrencia facilitan la escritura de programas consiguiendo lo mejor de máquinas multinúcleo y de la red, mientras su novel sistema de tipos permite la construcción de programas flexibles y modulares. Go compila rápidamente a código máquina aún con la comodidad de la recolección de basura y el poder de reflexión en tiempo de ejecución. Es un lenguaje tipado estáticamente, compilado y por lo tanto rápido, que se siente como un lenguaje interpretado y tipado dinámicamente.

Los motivos de elección de GOLANG han sido los siguientes:

- En cada petición que realiza el cliente a la API Rest se envía un token para identificarlo, es mucho más óptimo que abrir una sesión y mantenerla abierta, las peticiones son independientes y concurrentes por lo que Golang es la mejor opción para dar servicio gracias a la capacidad de resolver las peticiones rápidamente. Golang y Node.js son las mejores opciones actualmente.
- Rápido de programar y desplegar en un entorno de producción.
- Tenía mucho interés por aprender este lenguaje.
- Lenguaje de programación que cada vez más se orienta hacia la programación funcional, la cual me encanta.

Para acelerar la implementación de la API Rest se ha utilizado un framework llamado BEEGO. Es conocido por su alto rendimiento y facilidad para desplegar una aplicación en Golang en poco tiempo.

Tiene soporte RESTFul, sigue el patrón MVC, y contiene una tool llamada bee que te ayuda en el desarrollo: compila cuando detecta un cambio en el código, testeo automático, packing automático y despliegue en un entorno de ejecución.

También monitoriza las QPS (Queries Per Second), utilización de memoria y CPU, y el estado de las goroutines (concurrencia).

Beego es modular ya que contiene módulos de control de sesión, caché, logging, parseo de configuración, supervisor de rendimiento, contexto, soporte ORM (Object Relational Mapping) y simulación de peticiones.

En el apartado de implementación veremos algunos de estos módulos.

Base de datos

La decisión sobre qué sistema de base de datos utilizar ha sido muy complicado. Tenia dudas si utilizar una SQL como MySQL o PostgreSQL, o irme hacia una de tipo NoSQL (not only SQL) como MongoDB ante el creciente éxito que está teniendo.
Estudiemos las dos opciones:

MySQL es el más popular, un sistema de gestión de base de datos relacional que guarda los datos en tablas y utiliza SQL *(structured query language)* para el acceso a esos datos. Debes predefinir el esquema de la base de datos indicando las relaciones entre los campos de las tablas, la información relacionada puede ser almacenada en tablas separadas

Y se asocian esos datos mediante JOINS, la duplicación de datos se minimiza.

MongoDB en cambio, guarda los datos en documentos de tipo JSON que pueden variar de estructura, la información relacionada es guardada conjuntamente para un rápido acceso mediante el lenguaje de query de MongoDB.

A diferencia de MySQL, utiliza esquemas de datos dinámicos por lo que cada estructura de datos puede variar. Por ejemplo, en la aplicación un servicio podría tener más atributos que otro.

Casos de uso

Se debe usar MySQL cuando se requiere de un sistema complejo de transacciones, un conjunto de operaciones que deben ejecutarse como una unidad sobre la base de datos, por ejemplo, una plataforma de reserva online como la nuestra.
Los casos de unos más habituales de una MongoDB son en internet de las cosas (IoT), juegos móvil, analíticas real-time, gestión de contenidos, etc..
Finalmente, la base de datos elegida ha sido MySQL.

2.3. Front-end, la parte de cliente

Disponemos de 4 aplicaciones en el front-end:

1. App iOS para clientes y profesionales.
2. App WEB para clientes.
3. Cliente Android para profesionales.
4. App WEB de back-office para nosotros.

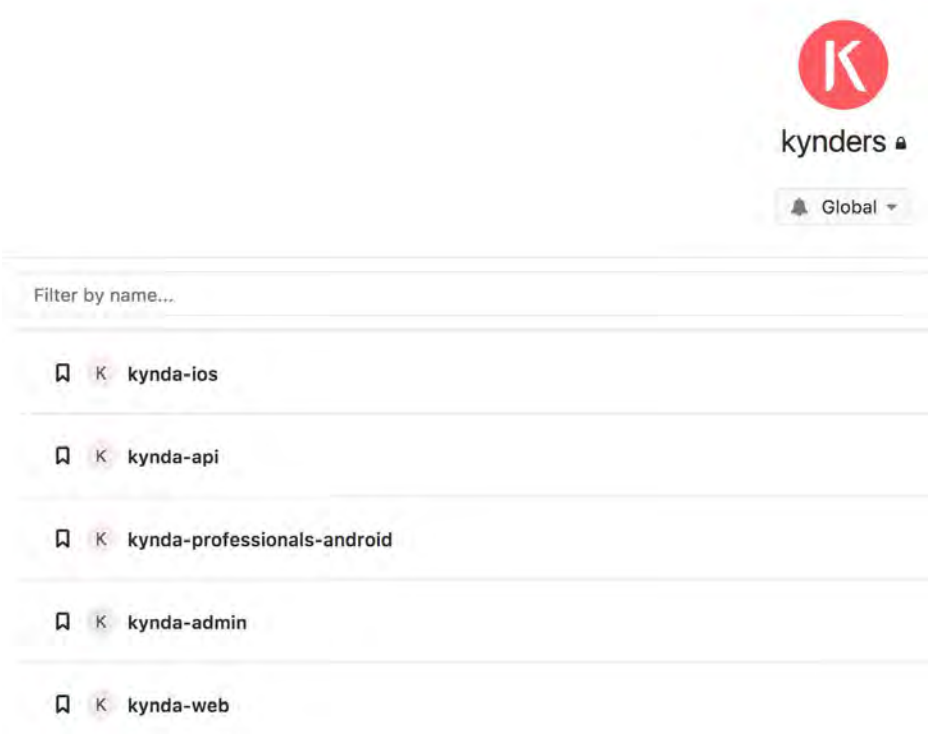
3. Implementación

3.1. Aplicación de metodologías ágiles: integración continua

En este proyecto se ha aplicado el concepto de integración continua. Es una práctica orientada a equipos pero como son muy buenas prácticas en desarrollo de software, algunas las he aplicado aquí.
Por definición es una práctica de desarrollo de software donde los miembros del equipo integran su trabajo frecuentemente, al menos una vez al día. Cada integración se verifica con un build automático (que incluye la ejecución de pruebas) para detectar errores de integración tan pronto como sea posible.

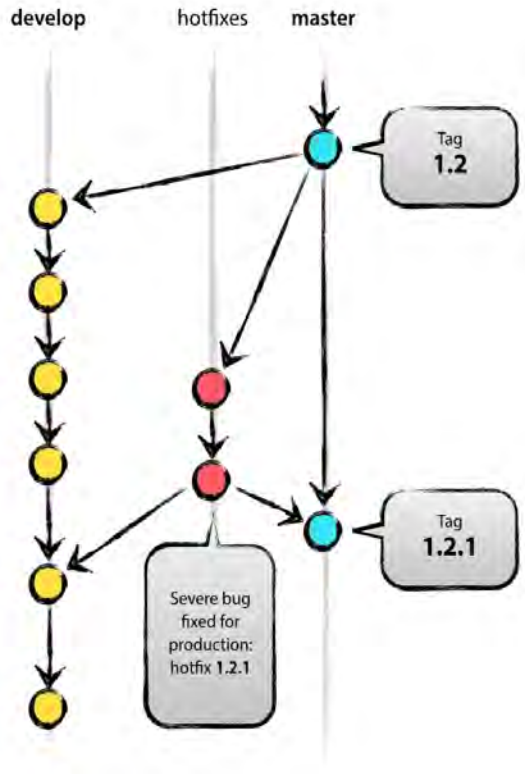
Control de versionado
Esta integración se realiza con un sistema de control de versiones distribuido, en mi caso, he utilizado GIT y como repositorio del proyecto, Gitlab.
Se han creado 5 proyectos en Gitlab:

- **kynda-api:** La aplicación BEEGO (Golang) que sirve una API Rest
- **kynda-ios:** El cliente iOS desarrollado en Swift
- **kynda-professionals-android:** El cliente ANDROID para profesionales
- **kynda-web:** El cliente WEB desarrollado en VueJS (JavaScript)
- **kynda-admin:** El cliente WEB de back-office



Repositorios privados de Kynda en Gitlab

Ramificación en los repositorios



La manera de trabajar con estos repositorios es mediante el manejo de ramas de desarrollo. El código se va ramificando para que cada miembro del equipo pueda trabajar por separado y al terminar se unifican para obtener como resultado una rama con todo el código desarrollado.
Normalmente se crean estas ramas en un repositorio:

Master: Contiene todo el código funcional y testeado que se despliega en producción. Siempre debe estar sin errores para poder desplegar en cualquier momento en producción.

Develop: Rama dónde se desarrolla. Una vez testeado y probado se fusiona con máster
Independiente de estas ramas, se van creando otras para desarrollar funcionalidades concretas de la aplicación.

Entornos

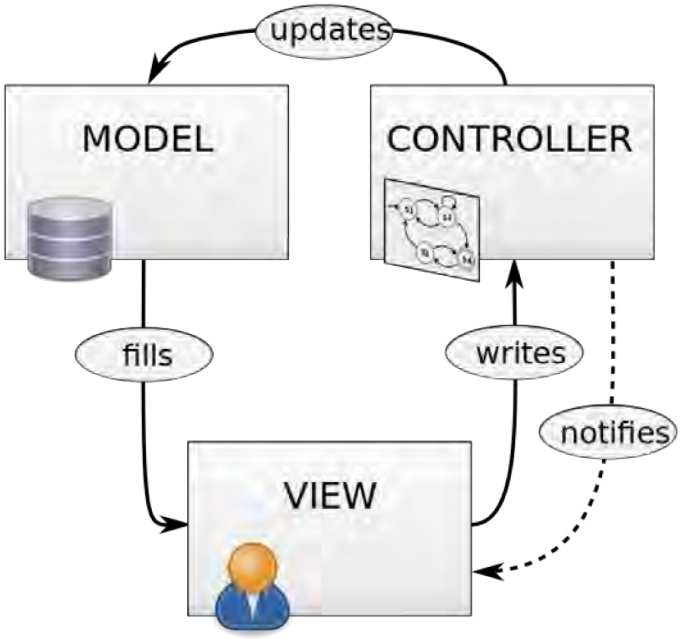
- Entorno de producción: Entorno en el que se despliega la aplicación de la rama master, la aplicación estará disponible en api.kynda.com
- Entorno de pre-producción: Es un entorno lo más parecido a producción (máquinas iguales, bases de datos, etc..) dónde prueban otras ramas a testear, en este caso la aplicación estará disponible en pre.kynda.com
- Entorno de desarrollo: En mi caso es mi ordenador personal, un MacBook Pro dónde tengo un servidor web y base de datos para desarrollar: localhost:8888

Para acabar, comentar que una parte muy importante de un entorno de integración continua es la programación de tests: Test-driven Development (TDD)
Implementar unit testing, integration testing, ... te permite protegerte ante errores en el desarrollo que por falta de tiempo no se han podido llevar a cabo.

Para acceder al código, los repositorios dónde se aloja son privados ya que no seria una buena práctica publicar el trabajo realizado para que otros puedan servirse de ello. Para acceder a ellos no duden en contactarme a mi e-mail y os daré acceso de inmediato.

3.2. Back-end: API Rest

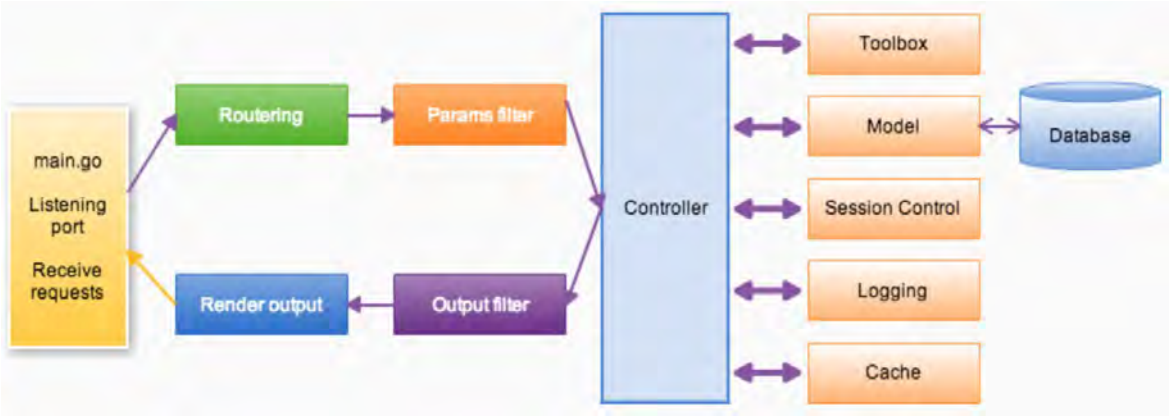
Para desarrollar la API Rest se ha utilizado el framework de Golang llamado Beego. Este framework sigue el patrón de diseño de software modelo - vista - controlador (MVC)



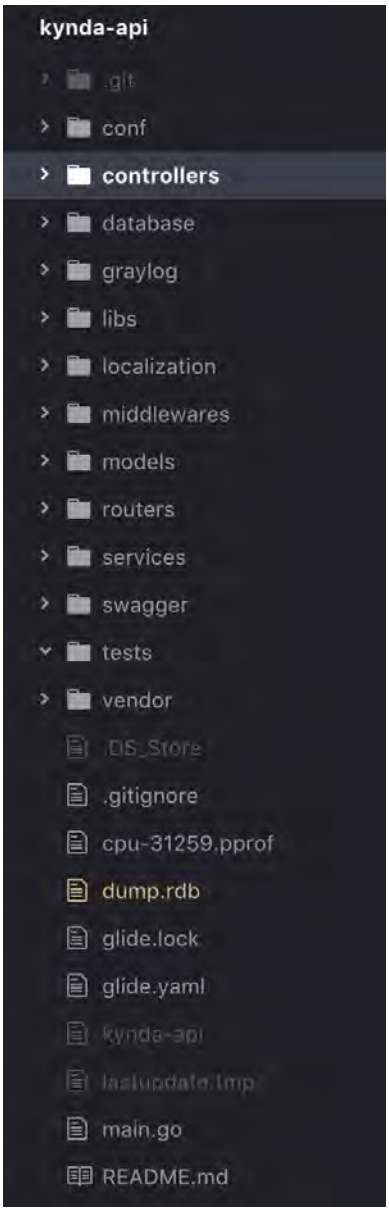
Patrón de software Model-View-Controller (MVC)

Para instalar el framework se debe antes haber instalado Golang. Una vez instalada toda la paquetería de beego también debemos instalar la herramienta para línia de comandos que nos facilitará la creación de todo el proyecto.

```
$ go get -u github.com/astaxie/beego
$ go get -u github.com/beego/bee
```



Arquitectura de una aplicación de beego



Directorio raíz de la aplicación de beego

Como en cualquier aplicación de Golang, el archivo **main.go** contiene la función main() que será el punto de entrada para la ejecución de nuestro programa.

```
package main

import (
  "gitlab.com/kynders/kynda-api/database"
  "gitlab.com/kynders/kynda-api/graylog"
  "gitlab.com/kynders/kynda-api/localization"
  _ "gitlab.com/kynders/kynda-api/routers"

  "github.com/astaxie/beego"
)

func init() {}

func main() {
  if beego.BConfig.RunMode == "dev" {
    beego.BConfig.WebConfig.DirectoryIndex = true
    beego.BConfig.WebConfig.StaticDir["/swagger"] = "swagger"
  }

  graylog.Init()
  db.Init()
  localization.Init()
  beego.Run()
}
```

main.go

Vemos que importamos las librerías database, graylog, localization, etc. para poder llamar a sus respectivas funciones de inicialización en el main().

Como podemos observar, se importa el paquete de routers que concuerda con el diagrama de la arquitectura de una aplicación de beego. La lógica desarrollada en el paquete analiza y filtra las peticiones entrantes, según el recurso pedido redirigirá la responsabilidad hacia un controlador.

En los controladores es donde se alberga gran parte de la lógica de negocio desarrollada para la API Rest.

En el directorio **vendor** se guardan todas las librerías dependientes del proyecto.

El directorio de **conf/** es muy importante en el proyecto. En el se definen conexiones a bases de datos, entornos de ejecución, claves, etc.

Routers/: Aquí se definen las rutas de la API Rest. Se enlaza ruta y controlador. Cuando se envía una petición, la aplicación sabe qué controlador debe recibirla y resolverla. Por ejemplo, una petición GET de una lista de servicios:

```
beego.NSNamespace("/services",
        beego.NSRouter("/",&controllers.ServiceController{}, "get:GetAllServices")
),
```

```
func init() {
    ns := beego.NewNamespace("/v1",

        beego.NSNamespace("/auth",
            beego.NSRouter("/provider", &controllers.AuthController{}, "post:TokenByProvider"),
            beego.NSRouter("/register", &controllers.AuthController{}, "post:RegisterByEmail"),
            beego.NSRouter("/login", &controllers.AuthController{}, "*:AuthByEmail"),
            beego.NSRouter("/forgot_password", &controllers.AuthController{}, "post:ForgotPassword"),
        ),

        beego.NSNamespace("/users",
            beego.NSRouter("/profile", &controllers.UserController{}, "get:GetProfile"),
            beego.NSRouter("/profile", &controllers.UserController{}, "put:UpdateUser"),
            beego.NSRouter("/name", &controllers.UserController{}, "put:UpdateFullname"),
            beego.NSRouter("/email", &controllers.UserController{}, "put:UpdateEmail"),
            beego.NSRouter("/password", &controllers.UserController{}, "put:UpdatePassword"),
            beego.NSRouter("/phone_number", &controllers.UserController{}, "put:UpdatePhoneNumber"),
            beego.NSRouter("/firebase_token", &controllers.UserController{}, "get:GetFirebaseToken"),
            beego.NSRouter("/device_token", &controllers.UserController{}, "post:CreateOrUpdateDeviceToken"),
        ),

        beego.NSNamespace("/professionals",
            beego.NSRouter("/", &controllers.ProfessionalController{}, "get:GetProfessional"),
            beego.NSRouter("/all", &controllers.ProfessionalController{}, "get:GetProfessionalList"),
            beego.NSRouter("/:id", &controllers.ProfessionalController{}, "get:GetProfessionalById"),
            beego.NSRouter("/:id/services", &controllers.ProfessionalController{}, "get:GetServicesByProfessionalId"),
            beego.NSRouter("/last_added", &controllers.ProfessionalController{}, "get:GetLastAddedProfessionals"),
            beego.NSRouter("/register", &controllers.AuthController{}, "post:RegisterProfessionalByEmail"),
            beego.NSRouter("/calendar", &controllers.CalendarController{}, "get:GetProfessionalCalendar"),
            beego.NSRouter("/calendar", &controllers.CalendarController{}, "post:CreateTimeSlotOnCalendar"),
            beego.NSRouter("/calendar", &controllers.CalendarController{}, "put:DisableTimeSlotStatusOnCalendar"),
            beego.NSRouter(":id/ratings", &controllers.ProfessionalController{}, "get:GetRatingsFromProfessional"),
            beego.NSRouter("/available", &controllers.ProfessionalController{}, "post:GetAvailableProfessionalsForService"),
            beego.NSRouter(":id/available_time_slots", &controllers.ProfessionalController{}, "get:GetAvailableTimeSlotsForProfessional"),
            beego.NSRouter("/appointments", &controllers.AppointmentController{}, "get:GetProfessionalAppointments"),
            beego.NSRouter("/appointments/:id", &controllers.AppointmentController{}, "get:GetProfessionalAppointment"),
            beego.NSRouter("/appointments/status", &controllers.AppointmentController{}, "put:UpdateAppointmentStatus"),
            beego.NSRouter("/services", &controllers.ProfessionalController{}, "get:GetProfessionalServices"),
            beego.NSRouter("/channels/own", &controllers.ChannelController{}, "get:GetProfessionalChannels"),
            beego.NSRouter("/channels/user/:userId", &controllers.ChannelController{}, "post:CreateOrFindChannelWithUser"),
        ),

        beego.NSNamespace("/locations",
            beego.NSRouter("/me", &controllers.LocationController{}, "get:GetUserLocations"),
            beego.NSInclude(
                &controllers.LocationController{},
            ),
        ),

        beego.NSNamespace("/categories",
            beego.NSInclude(
                &controllers.CategoryController{},
            ),
        ),
```

Definición de las rutas de la API Rest. Fuente: routers/router.go

En los directorios de **models/** y **controllers/** del patrón MVC con la lógica de negocio de la aplicación.
Se han creado los siguientes modelos de datos:

AccountProvider, Appointment, Area, Auth, Calendar, Card, Category, Channel, City, Country, Customer, DeviceToken, Location, Payment, Product, ProfessionalService, Professional, Rating, Referral, ServiceDetails, Service, SessionAttendant, SessionGuest, SessionProfessional, Session, SubCategory, TimeSlot, User.

Veamos un ejemplo, el usuario se ha modelado de esta manera:

```
type User struct {
    Id              int64           `orm:"column(id);pk;auto" json:"_id"`
    UId             string          `orm:"column(uid);size(255);unique" json:"uid"`
    Fullname        string          `orm:"column(fullname);size(255);null" json:"fullname"`
    Email           string          `orm:"column(email);size(255);unique" json:"-"`
    Birthday        time.Time       `orm:"column(birthday);type(date)" json:"-"`
    Gender          string          `orm:"column(gender);size(40);null" json:"-"`
    Picture         string          `orm:"column(picture);size(255)" json:"picture"`
    Password        string          `orm:"column(password);size(255);null" json:"-"`
    Salt            string          `orm:"column(salt);size(255);null" json:"-"`
    PhoneNumber     int             `orm:"column(phone_number);null" json:"-"`
    Type            int8            `orm:"column(type);default(1)" json:"type"`
    IsEnabled       int8            `orm:"column(is_enabled);default(1)" json:"-"`
    AccountProvider *AccountProvider `orm:"column(account_provider_id);rel(fk);null" json:"account_provider"`
    DeviceTokens    []*DeviceToken  `orm:"reverse(many)" json:"-"`
    Locations       []*Location     `orm:"reverse(many)" json:"-"`
    Channels        []*Channel      `orm:"reverse(many)" json:"channels"`
    PreferredLanguage string        `orm:"column(preferred_language);size(20);null" json:"-"`
    Country         *Country        `orm:"column(country_id);rel(fk);null" json:"-"`
    CreatedAt       time.Time       `orm:"column(created_at);type(datetime);null;auto_now_add" json:"created_at"`
    UpdatedAt       time.Time       `orm:"column(updated_at);type(datetime);null;auto_now" json:"-"`
    DeletedAt       time.Time       `orm:"column(deleted_at);type(datetime);null" json:"-"`
}
```

Como podemos observar, dentro de la struct definimos los atributos más típicos de un usuario y sus relaciones con otros modelos. El ORM de beego nos permite indicar la columna relacionada en la tabla de users de la base de datos para ese atributo. También indicamos el nombre de los campos en el caso que devolvamos un objeto user dentro de un JSON.

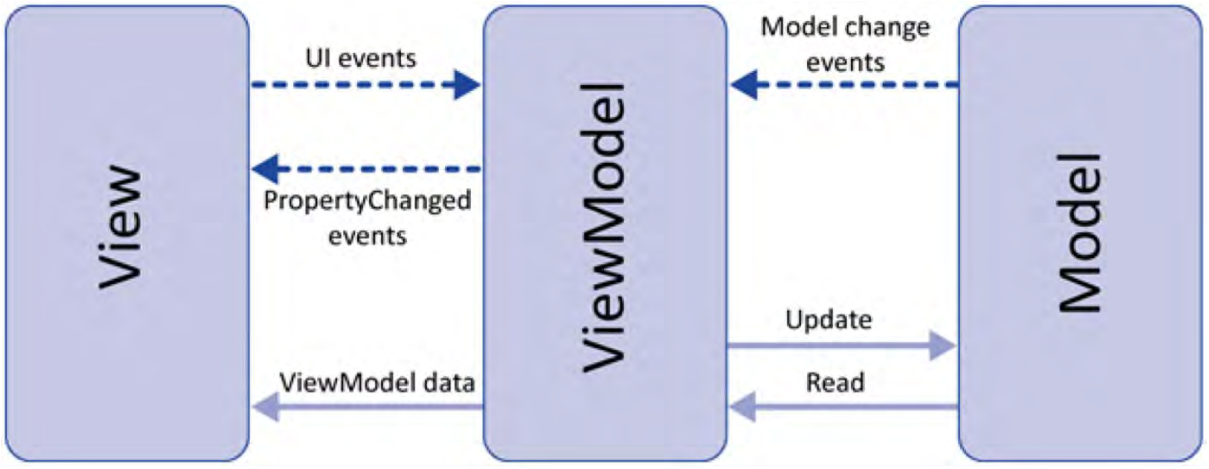
La lógica desarrollada en los modelos se basa principalmente la obtención, creación o actualización del modelo en cuestión. En cambio, en los controladores es donde se encuentra la lógica de negocio.

Se implementaron servicios de notificaciones push, registro con terceros como facebook, mensajería utilizando Firebase, pagos utilizando Stripe, etc.

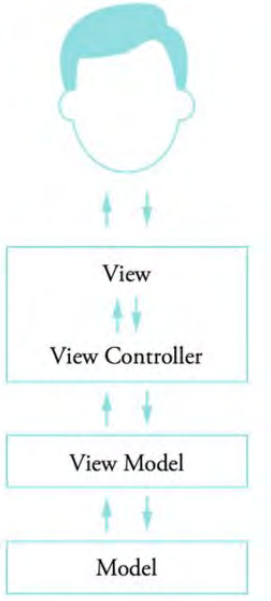
3.3. Front-end: Cliente iOS

Arquitectura del cliente iOS en Swift

La arquitectura de software del cliente en Swift sigue el patrón modelo - vista - modelo de vista (MVVM). Es el patrón por excelencia que se utiliza en programación iOS.



Patrón de software Model-View-ViewModel (MVVM)

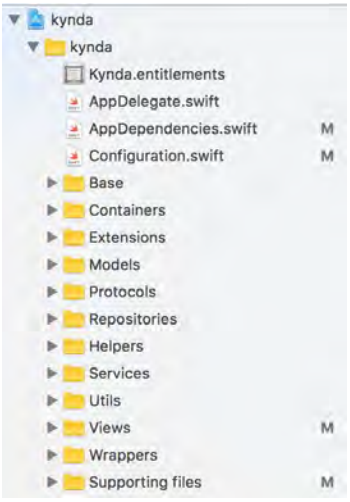


Usuario en un MVVM

Teniendo en cuenta las interacciones del usuario con el dispositivo, en la imagen se ve el flujo de datos entre las diferentes capas:
Este modelo tiene las siguientes restricciones:

- Los modelos no pueden hablar con cualquiera
- Los modelos de vista solo hablan con los modelos
- Los controladores de vista no pueden hablar directamente con los modelos, solo pueden interaccionar con los modelos de vista y las vistas.
- Las vistas solo pueden hablar con los controladores de vista, notificándolos de eventos que se producen.

Procedamos a explicar como está estructurado el proyecto de swift:



AppDelegate.swift: Contiene applicationDidFinishLaunching que es el primer método llama el sistema operativo (OS) cuando la aplicación ha acabado de lanzarse. Allí es dónde se le indica a OS qué controladores y vistas tiene que lanzar a continuación.
Configuration.swift: Su responsabilidad es cargar un archivo de configuración que he creado dónde figuran los escenarios en los que se puede lanzar la app: desarrollo y preproducción.
Por ejemplo, si es desarrollo la app hará las peticiones a mi máquina.

Base/: En este directorio se implementan clases y métodos que serán heredados o extendidos por otras vistas de la aplicación. Por ejemplo, la barra de navegación es compartida por todas las vista.

Containers/: Se implementa un MainContainerController que es el que contiene el menú principal, las demás vistas se montan sobre este contenedor consiguiendo así que la navegación sea muy fluida y pudiendo lanzar vistas desde este contenedor, como podría ser una vista que informara que no hay conexión a internet. De esta manera, solo el contenedor tiene esta responsabilidad.

Extensions/: Las extensiones en Swift sirven para extender clases propias o nativas del lenguaje para añadir funcionalidad que necesitemos. Por ejemplo, Swift tiene una clase nativa UIColor con los colores principales, en el proyecto se ha extendido esa clase para añadirle más colores que se necesitaba, o incluso he añadido métodos a la clase nativa UIView para poder insertar un background en una vista.

```swift
//
//  ColorExtension.swift
//  kynda
//
//  Created by André Ribeiro Caçador on 29/11/16.
//  Copyright © 2016 Kynda. All rights reserved.
//

import Foundation
import UIKit

extension UIColor {

    static func kyn_downy() -> UIColor {
        return UIColor(red: 0.47, green: 0.79, blue: 0.76, alpha: 1.0)
    }

    static func kyn_defaultPrimaryColor() -> UIColor {
        return self.kyn_carnationRed()
    }

    static func kyn_defaultSecondaryColor() -> UIColor {
        return self.kyn_tealBlue()
    }
```

Extensión de UIView

De esta manera se puede utilizar este método en todas las views sin tener que repetir código siguiendo el principio de desarrollo de software, DRY (don't repeat yourself)

Models/: Se implementan los modelos de datos de la aplicación, la mayor parte son comunes con los modelados en el servidor.

Protocols/: Los protocolos en Swift son las clases abstractas de Java. En un protocolo defines los métodos, propiedades, tareas o funcionalidades que deben contener las clases o estructuras que se ajusten a ese protocolo. En nuestro proyecto tenemos por ejemplo el protocolo Navigator que define que toda clase que sea navegable (puedas ir de una vista a otra) debe implementar los siguientes métodos:

```swift
//
//  Navigator.swift
//  kynda
//
//  Created by André Ribeiro Caçador on 9/1/17.
//  Copyright © 2017 Kynda. All rights reserved.
//

import Foundation
import UIKit

protocol Navigator {
    func pushViewController(_ viewController: UIViewController, animated: Bool)
    func setViewController(_ viewController: UIViewController, animated: Bool)
    func present(viewController: UIViewController, animated: Bool)
    func popToRootViewController(animated: Bool)
    func popViewController(animated: Bool)
    func transitionToPublicContainer()
}
```

Protocolo Navigator

Repositories/: La responsabilidad de un repositorio es la de realizar peticiones al servidor, gestionar las respuestas y modelar los datos del JSON devuelto. Cada entidad tiene su propio repositorio, por ejemplo, esta es la implementación del ServiceRepository.

```swift
//
//  ServiceRepository.swift
//  kynda
//
//  Created by André Ribeiro Caçador on 24/2/17.
//  Copyright © 2017 Kynda. All rights reserved.
//

import Foundation
import ObjectMapper
import BrightFutures

class ServiceRepository {
    lazy var apiDatasource = APIDataSource(networkMode: .network)

    func retrieveByCategory(_ categoryID: Int) -> Future<[Service]?, NSError> {

        let serviceAPI = ServiceAPI.retrieveByCategory(categoryID: categoryID)

        let request = APIRequest(resource: serviceAPI)
        let requestFuture = self.apiDatasource.processRequest(request)
        let responsePromise = Promise<[Service]?, NSError>()

        requestFuture.onSuccess(callback: { (json) in
            let node = request.resource.rootNode
            let servicesJSON = json[node] as! [JSON]
            let services = Mapper<Service>().mapArray(JSONObject: servicesJSON)
            responsePromise.success(services)
        }).onFailure { (error) in
            responsePromise.failure(error)
        }

        return responsePromise.future
    }
```

ServiceRepository.swift

Services/: Este es quizás el concepto que más me gusta del proyecto Swift. Aquí están servicios que pueden ser llamados desde cualquier clase de la aplicación. Esto implica gestión de instancias, singletons, estados, etc... He dedicado mucho tiempo a implementar el servicio de red, es uno de los servicios que más me ha costado crear y que procederé a explicar:

Se basa fundamentalmente en 5 clases: APIResource, APIRequest, APIDataSource, NetworkService y AlamofireNetworkService.

- **APIResource**

Modela los atributos que debe tener un recurso para ser pedido a la API: Un recurso tiene asociado un método, un path (ruta) y puede tener parámetros (querystrings) de filtro, ids, etc..

- **APIRequest**

Una APIRequest se inicializa con un APIResource, y tiene la responsabilidad de formatear los datos necesarios para realizar la petición.

- **NetworkService**

NetworkService es un protocolo que define que todo NetworkService debe tener un método que procese la petición.

- **AlamofireNetworkService**

la clase AlamofireNetworkService se ajusta al protocolo NetworkService por lo que implementa el método processRequest() que utiliza métodos de la librería de Alamofire, una librería de red HTTP.

- **APIDataSource**

Finalmente, esta clase instancia al servicio AlamofireNetworkService pasándole la URL del servidor y gestiona las respuestas de la red.

Views/: Contienen los modelos de vista, controladores de vista y vistas del proyecto.

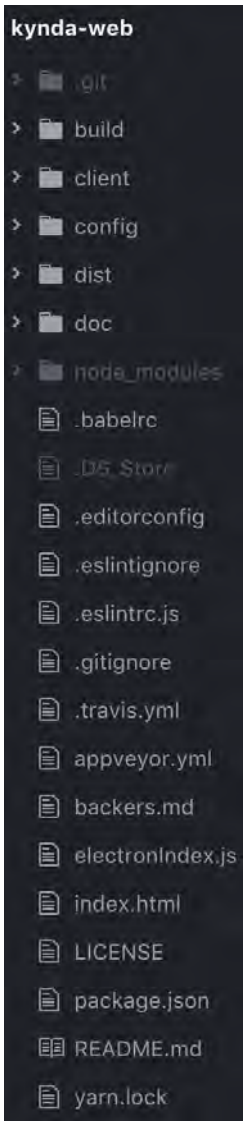
Mencionar que la aplicación es Reactiva, esto significa que toda la gestión de señales las realiza el framework ReactiveCocoa. Un framework inspirado en programación funcional-reactiva que te permite poner “observadores” a variables que cuando cambian envían una señal avisando de dicho cambio, consiguiendo así que no haya incoherencia de datos ni que tengas que preocuparte de implementar tu esas señales.

A grandes rasgos he explicado con ejemplos algunos conceptos interesantes del proyecto en Swift. Es un proyecto muy extenso y que ha absorbido la mayor parte del tiempo, es difícil relatar aquí toda la arquitectura diseñada e implementada.

3.4. Front-end: Cliente WEB y BACK-OFFICE

Javascript es un lenguaje de programación que se ha popularizado en los últimos años. Es liviano, interpretado y de scripting para los navegadores pero también se utiliza para otros entornos como en lógica de servidor. Es un lenguaje de tipado dinámico, orientado a objetos, imperativo y declarativo. Es el lenguaje más popular actualmente.

Los clientes web se han creado utilizando un framework Javascript llamado VueJS, se le conoce como un framework progresivo ideal para crear interfaces de usuario. Al contrario de otros frameworks monolíticos, Vue está diseñado para ser integrado progresivamente. La librería core está enfocada solo en la capa de vista y es de fácil integración con otras librerías y proyectos existentes.



Directorio raíz webapp

La aplicación webapp de cliente se ha estructura de la siguiente manera:

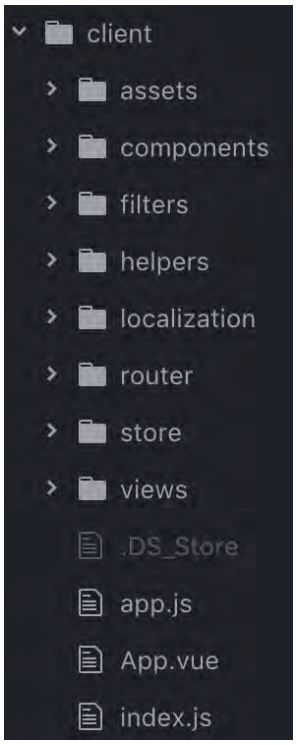
build/: Se generan los archivos html que serán consumidos por el servidor web en el momento de servir el código fuente de la aplicación. Se genera una build cada vez que desplegamos una nueva versión en el entorno en cuestión (producción, pre o local).

client/: Este directorio lo veremos más detallado a continuación ya que es el más importante porque contiene toda la lógica de la aplicación.

config/: Variables globales dependiendo del entorno.

dist/: Archivos estáticos como imágenes, estilos, fuentes, etc.

En el directorio **node_modules** se guardan todas las librerías dependientes del proyecto.



App.js es el punto de entrada a la aplicación, donde importamos librerías dependientes e inicializamos módulos como el router, red, localización, etc.

En **App.vue** se monta la raíz de la plantilla de la aplicación. De aquí se irán incluyendo más vistas. En nuestro caso:

```
<template>
  <div id="app">
    <nprogress-container></nprogress-container>
    <navbar :show="true"></navbar>
    <app-main></app-main>
    <footer-bar></footer-bar>
  </div>
</template>
```

En lenguaje que se utiliza en los archivos .vue es característico del framework.
En el directorio **views/** encontraremos todas las plantillas de la aplicación.

En el **router** indicamos para cada componente que se ejecuta desde el navegador cual debe ser la vist que debe mostrarse.

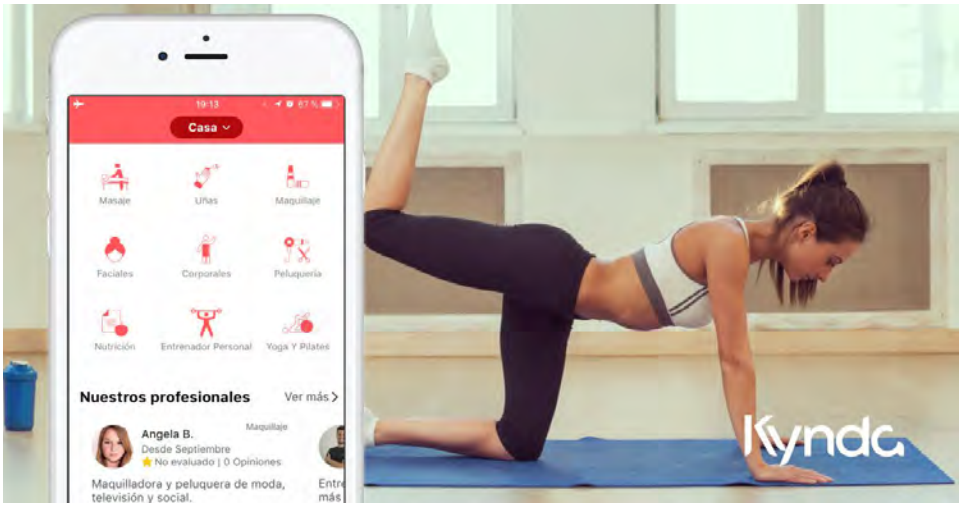
En la **store** gestionamos los diferentes estados de la aplicación ya que conforme va creciendo diferentes componentes deben comunicarse entre sí, por ese motivo Vue tiene su propia librería de gestión de estados: vuex. También declaramos constantes y datos estructurados referentes a la lógica de aplicación.

En **localization/** tenemos los archivos con el diccionario de traducciones de tipo .json.

3.5. Front-end: Cliente ANDROID

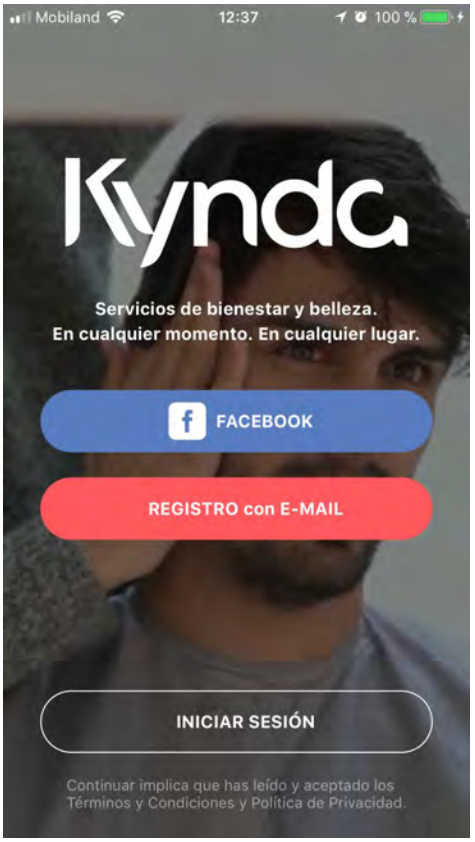
La creación del cliente de Android ha sido supervisada y dirigida por mi pero ejecutada por otro miembro de nuestro equipo.

4. Aplicaciones

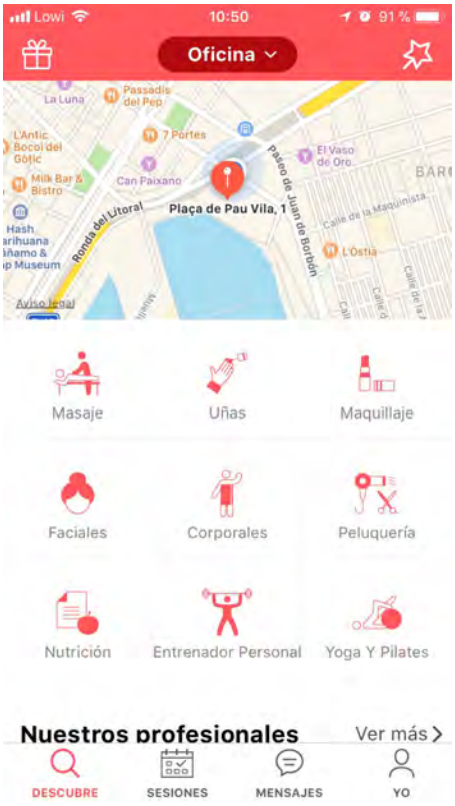


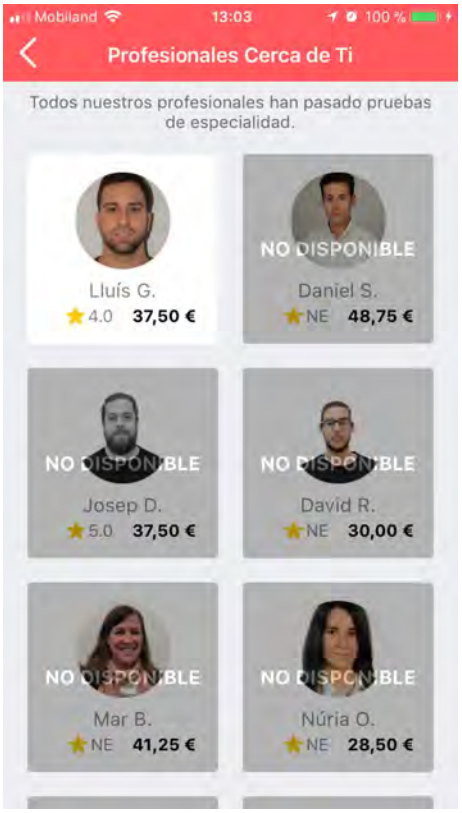
1. Cliente IOS

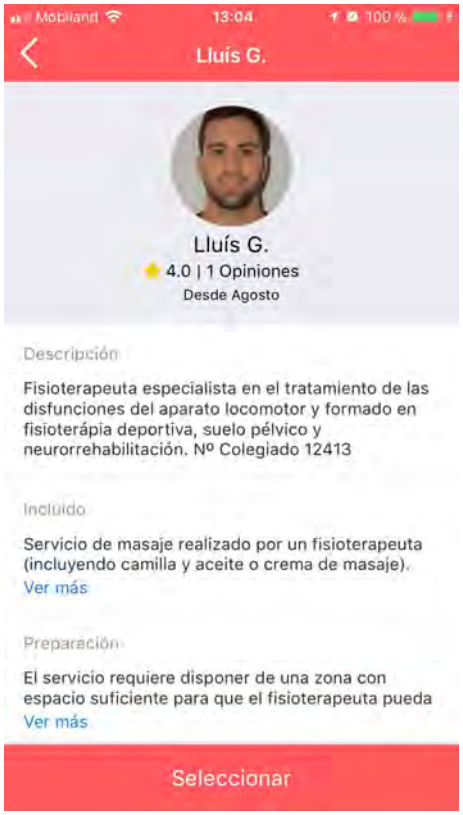
Login:

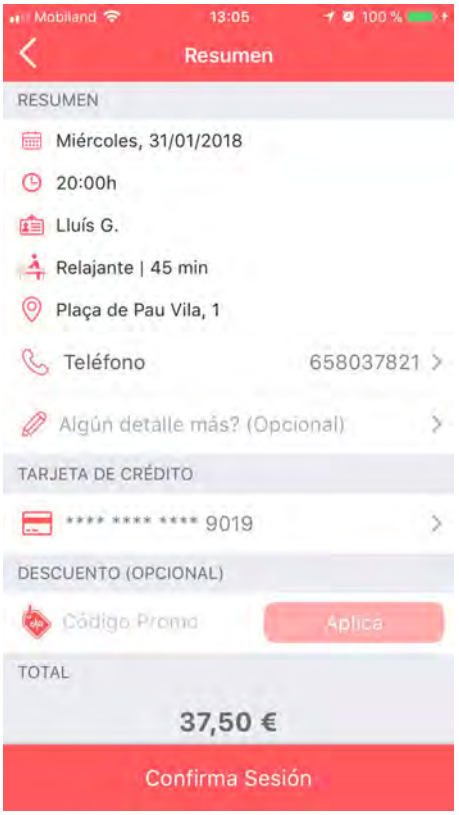


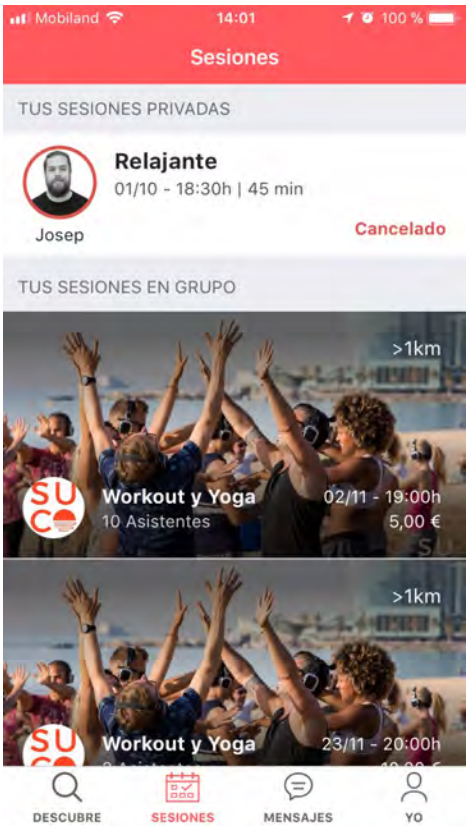
Proceso de reserva:

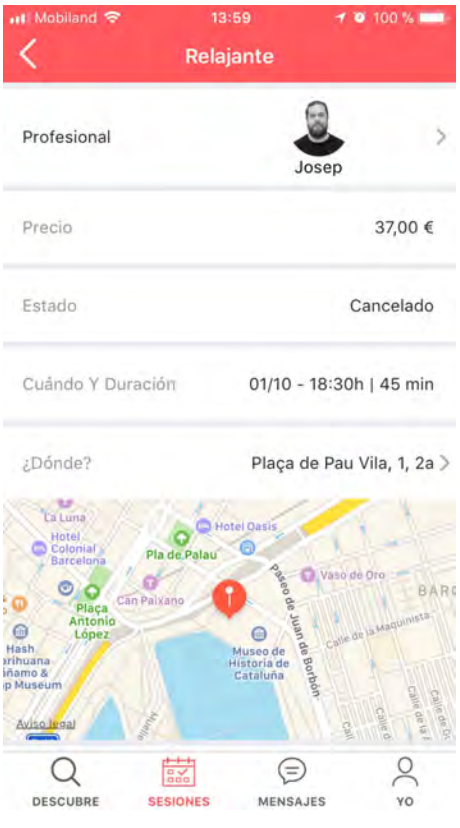


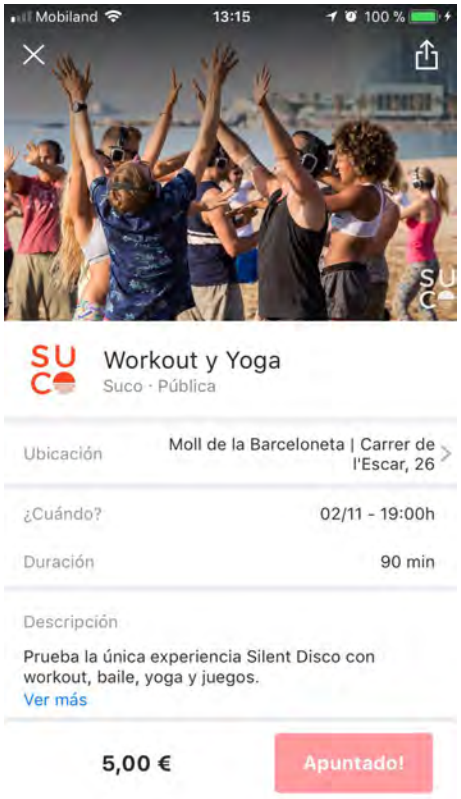



Mobiland 13:04 100 %
Lluis G.
Lluís G.
4.0 | 1 Opiniones
Desde Agosto
Descripción
Fisioterapeuta especialista en el tratamiento de las disfunciones del aparato locomotor y formado en fisioterápia deportiva, suelo pélvico y neurorrehabilitación. Nº Colegiado 12413
Incluido
Servicio de masaje realizado por un fisioterapeuta (incluyendo camilla y aceite o crema de masaje).
Ver más
Preparación
El servicio requiere disponer de una zona con espacio suficiente para que el fisioterapeuta pueda
Ver más
Seleccionar


Mobiland 13:05 100 %
Resumen
RESUMEN
Miércoles, 31/01/2018
20:00h
Lluís G.
Relajante | 45 min
Plaça de Pau Vila, 1
Teléfono 658037821
Algún detalle más? (Opcional)
TARJETA DE CRÉDITO
**** **** **** 9019
DESCUENTO (OPCIONAL)
Código Promo
Aplica
TOTAL
37,50 €
Confirma Sesión

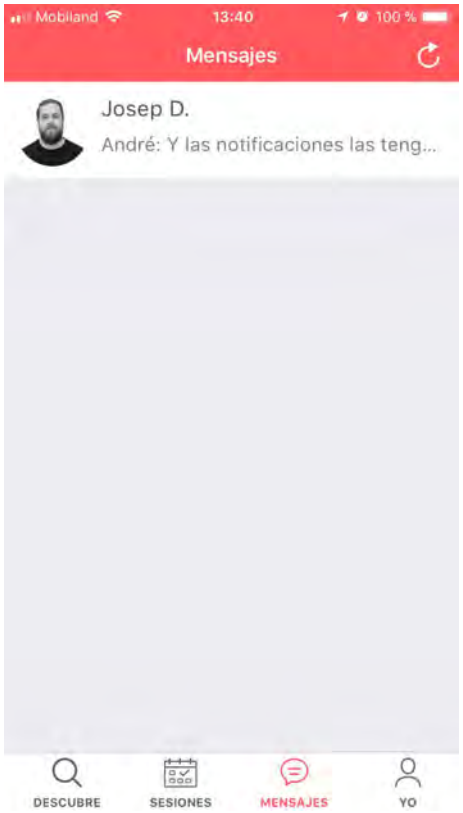

Mobiland 14:01 100 %
Sesiones
TUS SESIONES PRIVADAS
Relajante
01/10 - 18:30h | 45 min
Josep
Cancelado
TUS SESIONES EN GROUPO
>1km
Workout y Yoga
10 Asistentes
02/11 - 19:00h
5,00 €
>1km
Workout y Yoga
23/11 - 20:00h
DESCUBRE
SESIONES
MENSAJES
YO


Mobiland 13:59 100 %
Relajante
Profesional
Josep
Precio 37,00 €
Estado Cancelado
Cuándo Y Duración 01/10 - 18:30h | 45 min
¿Dónde? Plaça de Pau Vila, 1, 2a
Hotel Oasis
Pla de Palau
Vaso de Oro
Plaça Antonio López
Can Paixano
Museo de Historia de Cataluña
DESCUBRE
SESIONES
MENSAJES
YO



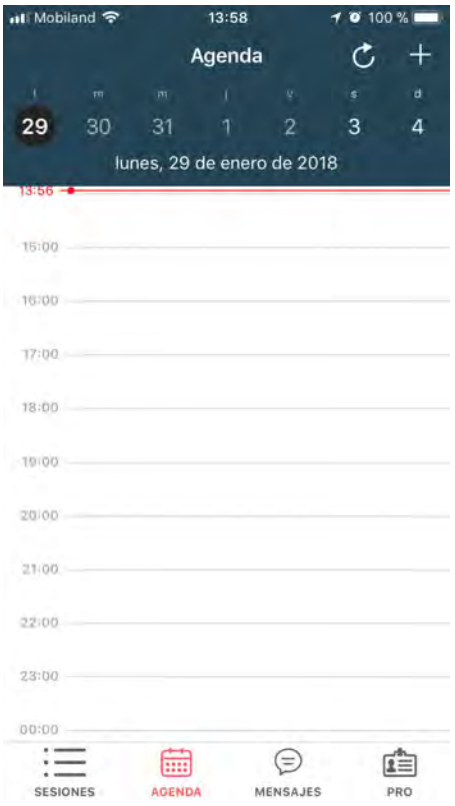
Sesiones en grupo
También organizamos sesiones en grupo en diferentes puntos de tu ciudad con nuestros mejores profesionales: un workout intenso en la playa, sesión de yoga en tu parque más cercano, manicuras en un hotel, etc.

Mensajería con el profesional:

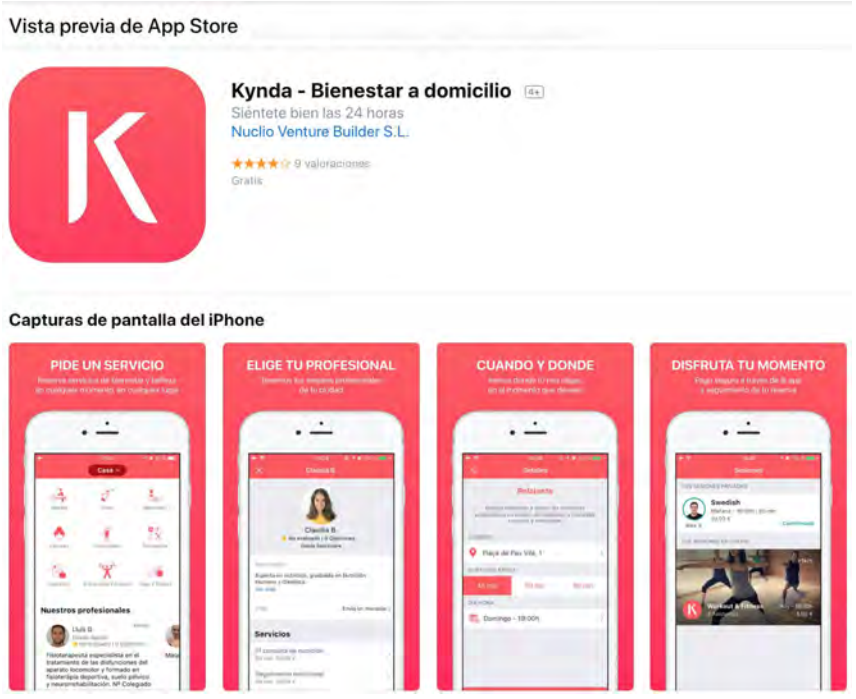


Perfil de usuario:

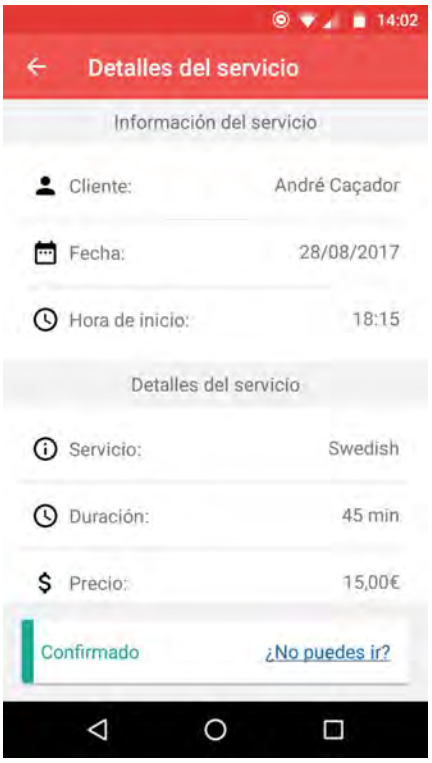
Si el usuario es profesional puede acceder a su área desde el perfil con información sobre sus citas, su calendario con las disponibilidades, servicios realizados, precios, etc.

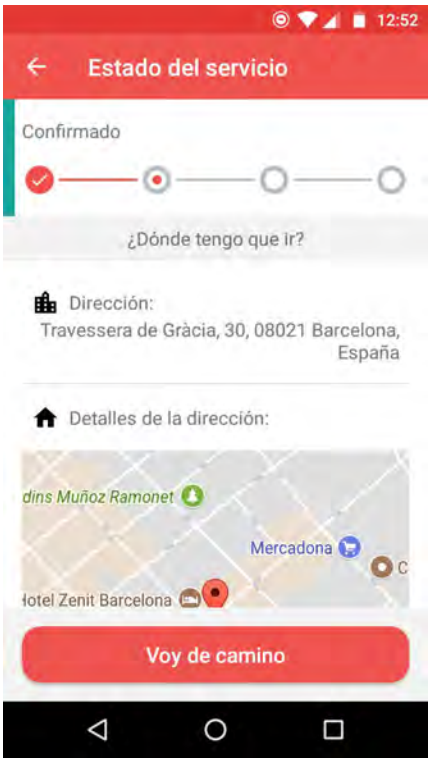


Se puede descargar la aplicación a través de itunes en https://itunes.apple.com/ES/app/id1269490824?mt=8



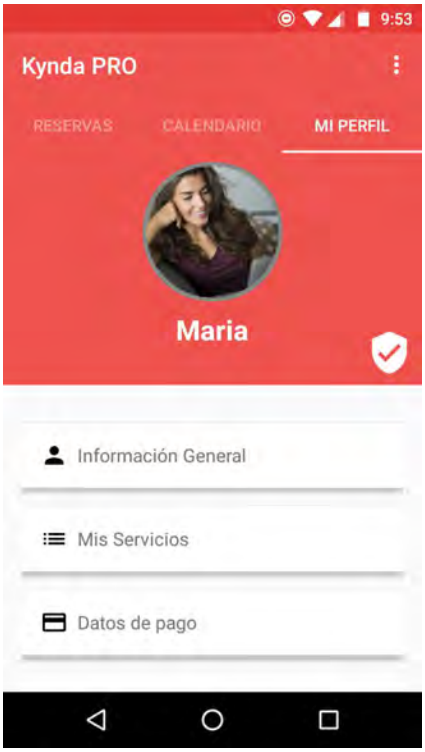
Cliente Android (solo para profesionales):



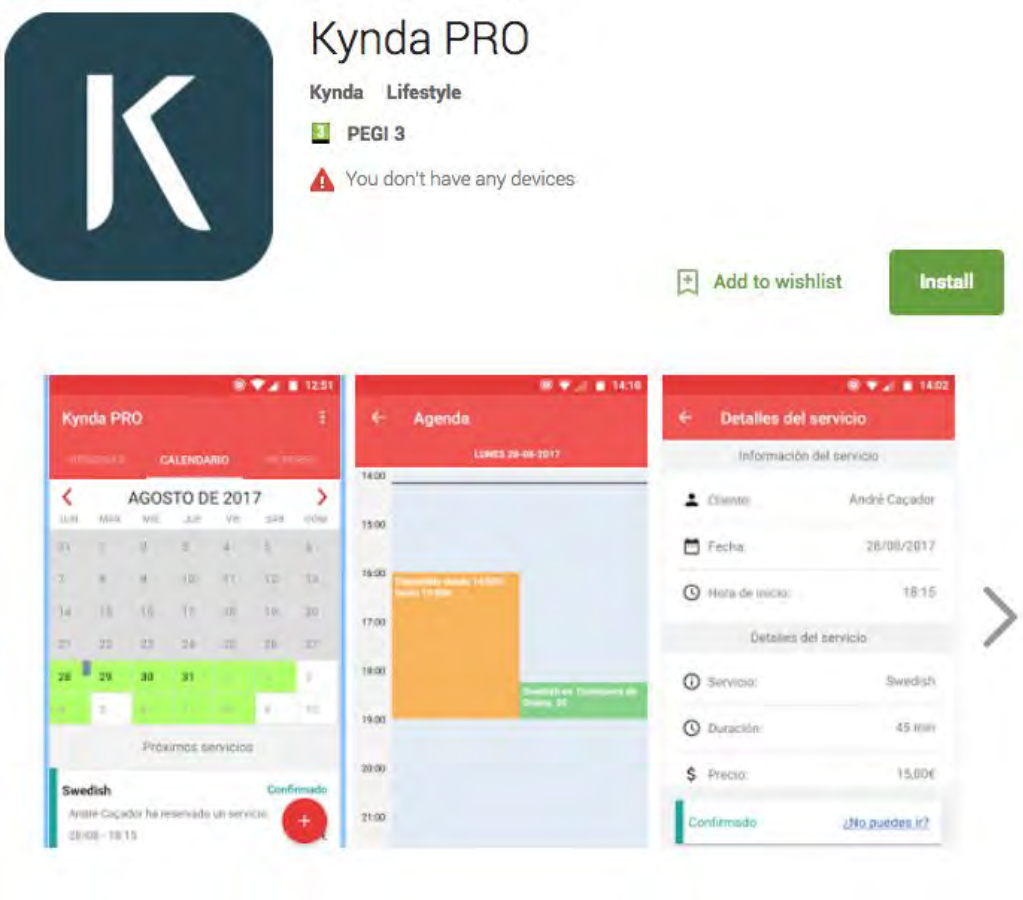



9:53
Kynda PRO
RESERVAS
CALENDARIO
MI PERFIL
JUNIO DE 2017
LUN. MAR. MIÉ. JUE. VIE. SÁB. DOM.

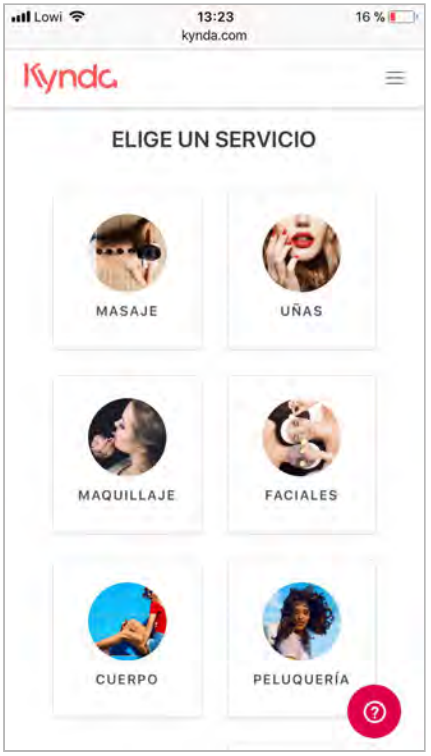

14:10
Agenda
LUNES 28-08-2017
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
Swedish en Travessera de Gracia, 30

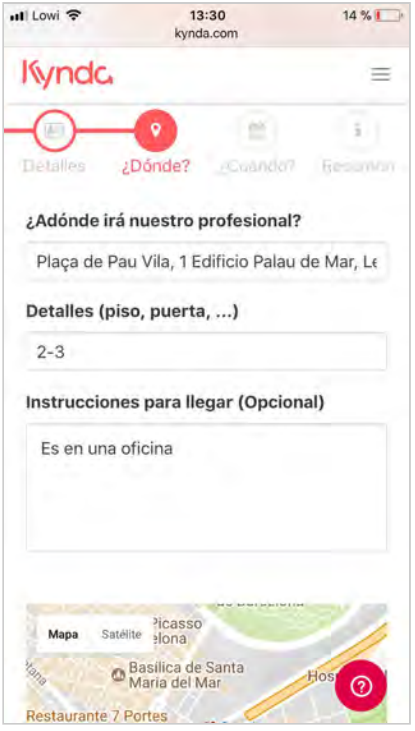

9:53
Kynda PRO
RESERVAS
CALENDARIO
MI PERFIL
Maria
Información General
Mis Servicios
Datos de pago

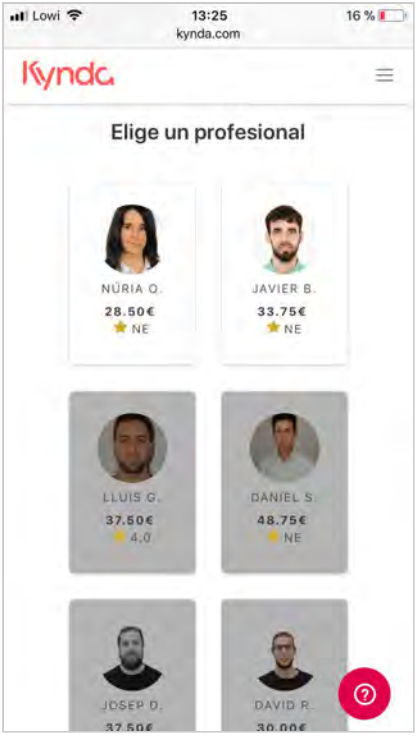
Se puede descargar la aplicación a través de la play store en
https://play.google.com/store/apps/details?id=com.kynda.professionals



Cliente WEB accediendo a www.kynda.com:

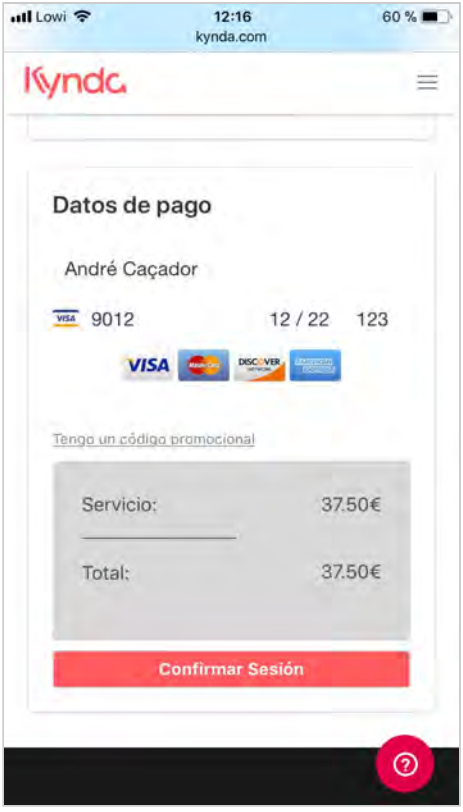






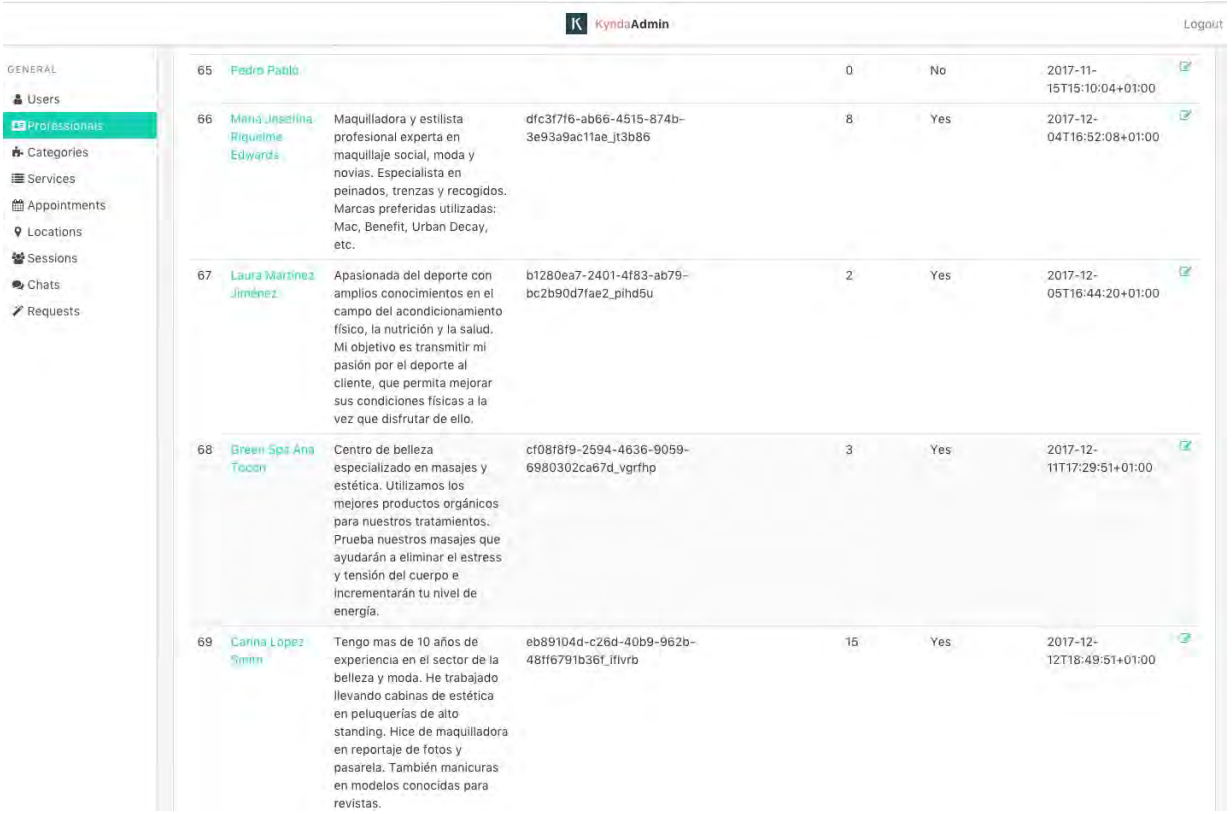

Lowi
11:57
kynda.com
70 %
Kynda
Lluís G.
4.0 | 37.50€
Fisioterapeuta especialista en el tratamiento de las disfunciones del aparato locomotor y formado en fisioterápia deportiva, suelo pélvico y neurorrehabilitación. Nº Colegiado 12413
Descripcion
Enfocado a la relajación de la musculatura, aumento del retorno venoso, eliminación de productos de desecho y disminución de presión sanguinea. Calma el cuerpo y la mente proporcionando sensación de bienes
Esconder


Lowi
12:13
kynda.com
61 %
Kynda
Detalles
¿Dónde?
¿Cuando?
Resumen
Relajante
(45min aprox)
01-02-2018
09:00h
Plaça de Pau Vila, 1 Edificio Palau de Mar, Letra A, Planta 2, 08039 Barcelona, España
2-4
Tu profesional
Lluís G.
4.0
Explicanos más sobre ti (Opcional)

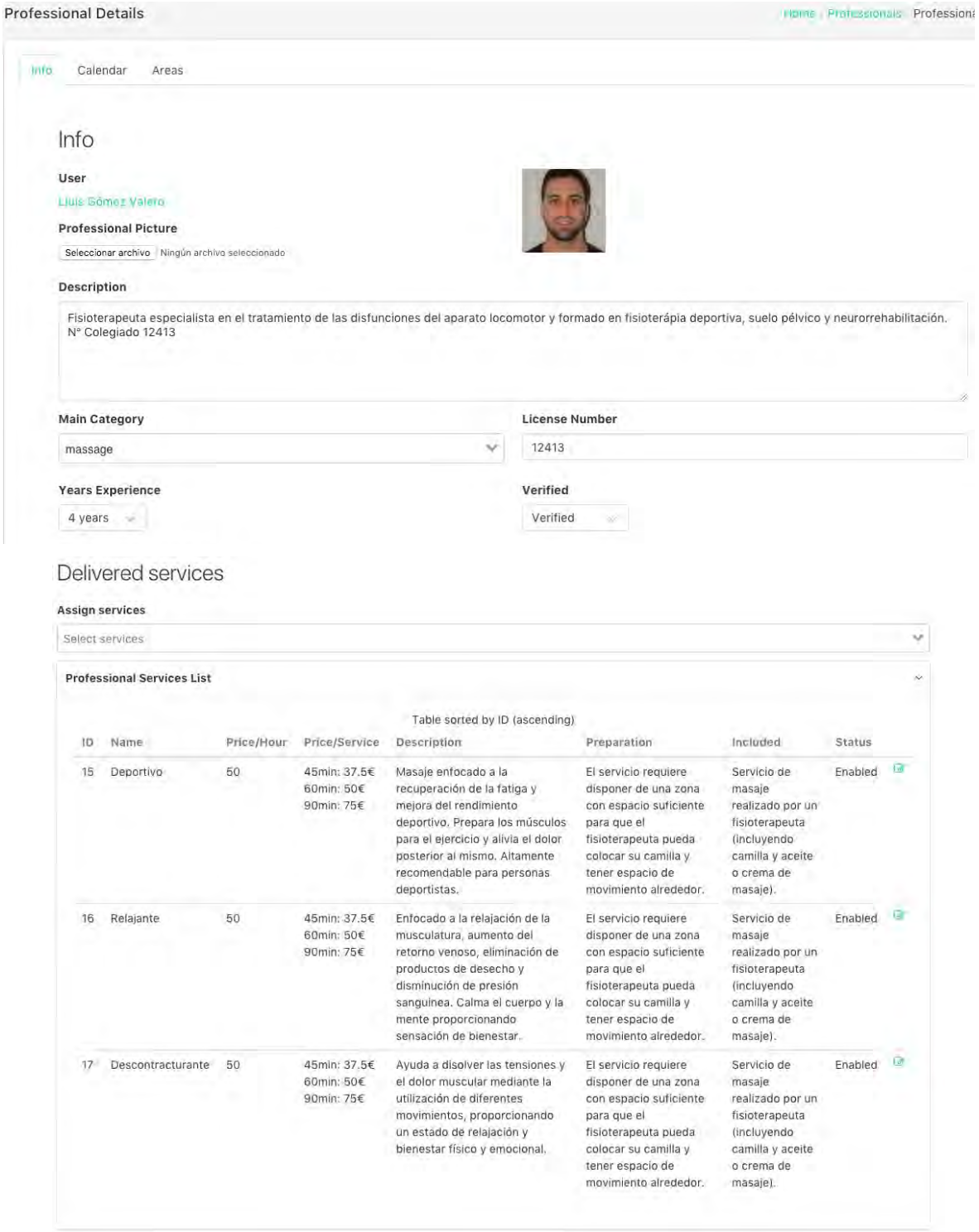

Lowi
12:16
kynda.com
60 %
Kynda
Datos de pago
André Caçador
9012
12 / 22
123
Tengo un código promocional
Servicio:
37.50€
Total:
37.50€
Confirmar Sesión

El BACK-OFFICE accediendo a admin.kynda.com (debes ser admin para poder acceder):

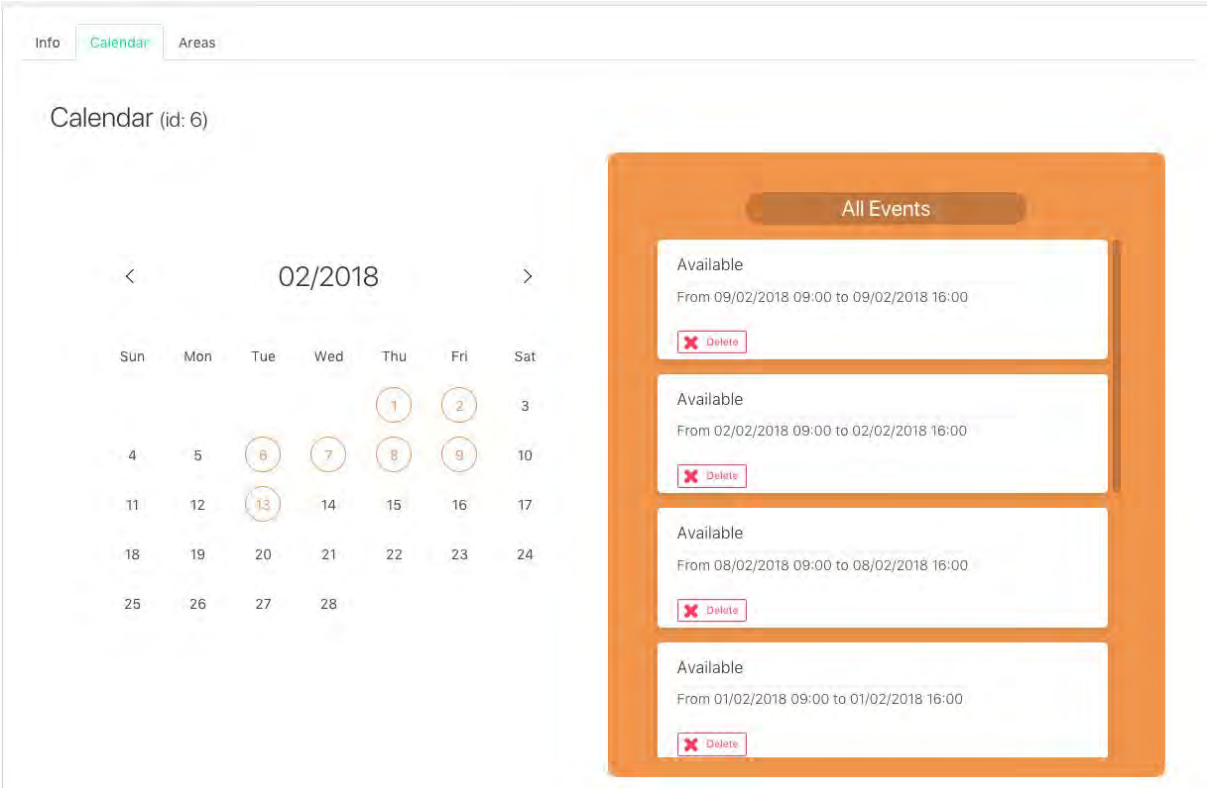
Profesionales:



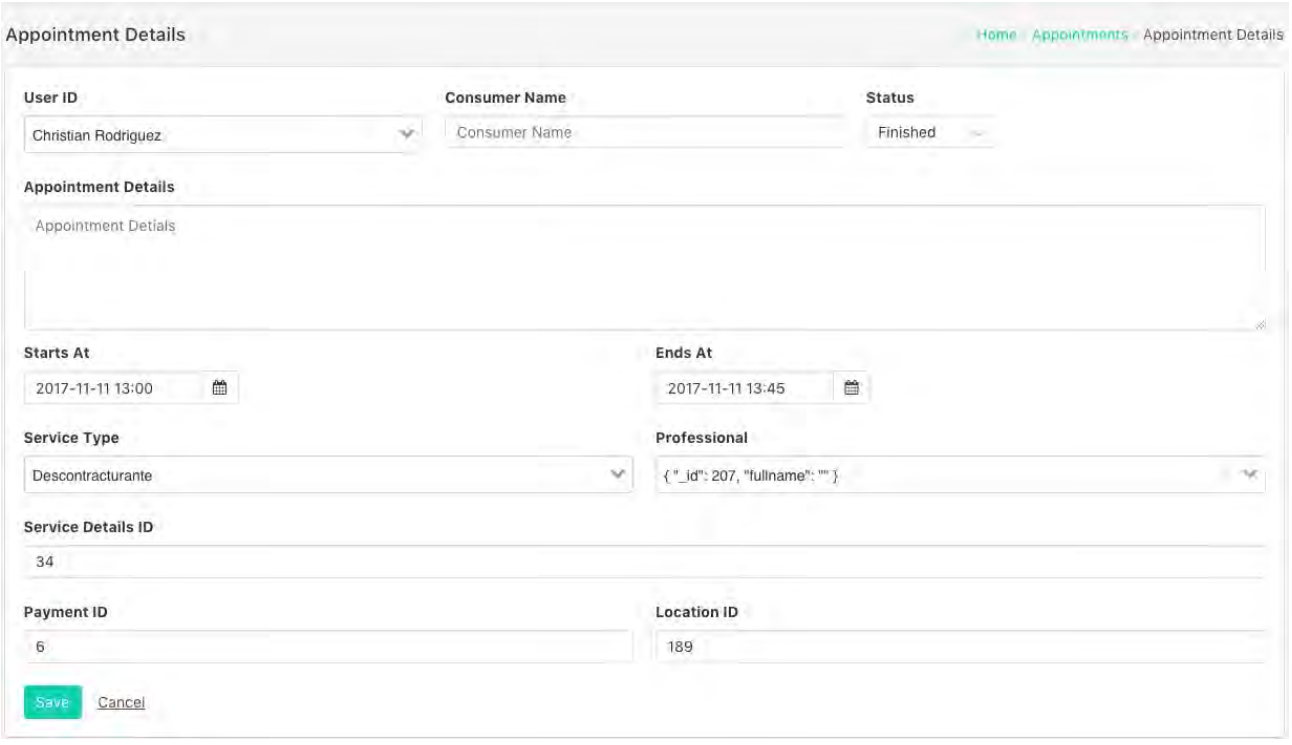
Ficha del profesional:



El calendario del profesional:



Detalles de una reserva:



9. Marketing digital

La importancia de una buena estrategia de marketing tanto digital como offline es muy importante en la promoción de un producto o servicio tecnológico, más si el servicio innovador como en nuestro caso. Kynda propone una experiencia inexistente en el mercado, además necesita de crear marca a su alrededor para generar confianza ya que el cliente confiar en la plataforma para dejar entrar a uno de nuestros profesionales en su domicilio.

Durante todo el proceso de creación de la startup el equipo nunca fue consciente de nuestra futura necesidad en ese sentido. Además entre los recursos que disponíamos no contábamos con una persona experta en adquisición de usuarios, y los mejores “marketers” como se le conoce en el mundillo startup están trabajando para startups ya consolidadas por lo que es un perfil complicado de encontrar.

Una vez finalizada la aplicación iOS, la primera versión del producto para testear en entorno de producción debíamos empezar a conseguir métricas para validar el modelo de negocio e ir mejorando.

La plataforma contaba en ese momento con unos 40 profesionales en el conjunto de categorías del marketplace, debíamos empezar a obtener los primeros clientes para que nuestros profesionales no abandonaran la plataforma.

La estrategia que se diseñó fue totalmente digital dado el poco presupuesto que contábamos para los siguientes tres meses (octubre-noviembre-diciembre) que era de unos 5.000€.
Se decidió crear campañas de Facebook Ads para conseguir las primeras instalaciones de la aplicación en iOS. Sus herramientas son muy potentes, nos permitieron acotar y definir nuestro potencial cliente de forma eficiente. Se crearon Ads para cada vertical del marketplace, acotando a mujeres con dispositivo iOS con más de 24 años de la ciudad de Barcelona.

Durante los meses de octubre y noviembre se invirtieron alrededor de 900€ en Facebook Ads:

- 750€ en adquisición de usuarios.
- 150€ en adquisición de profesionales.

Asi podríamos obtener nuestras primeras métricas referidas a clientes:

- Coste por click, CPC
- Coste por instalación, CPI
- Coste de adquisición, CAC

- Tasa de abandono
- Ticket medio neto
- Recurrencia
- Lifetime value, LTV

Las más importantes son:
El CAC es muy importante porque nos indican lo que debemos invertir para conseguir que un usuario pague por uno de nuestros servicios.
Con el ticket medio sabremos nuestra ganancia media por cliente y servicio.
Finalmente con el LTV haremos una aproximación sobre nuestras ganancias con un cliente en el tiempo que un cliente esté con nosotros gracias a la recurrencia.
Es complicado realizar una buena aproximación para el LTV y retención con solo dos meses de métricas, por ese motivo no hemos podido obtenerlas.

A continuación, añado algunos ejemplos de las creatividades que se realizaron para crear las campañas de ads. Las imágenes utilizadas son extraídas de repositorios de libre utilización.

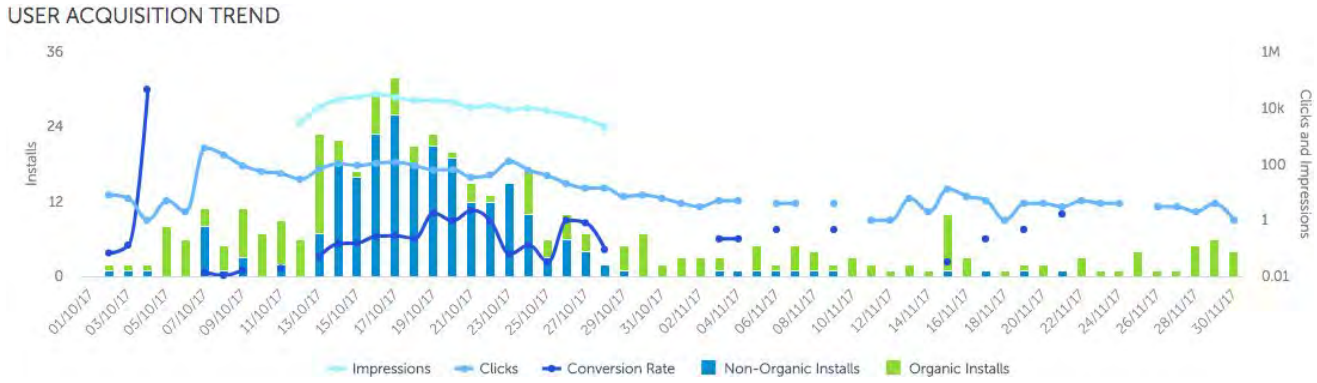
Captación de usuarios:

Captación de profesionales:

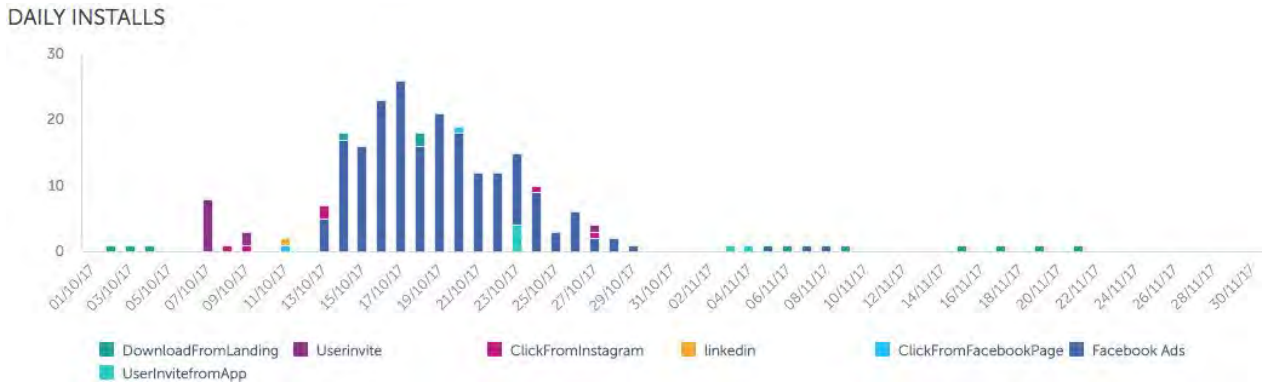
Resultados

Impresiones: 228.121
Clicks: 2.033
Ratio de conversión a instalación: 11.90%
Usuarios totales: ~500
Servicios realizados: 4
CAC: ~180€

Como observamos en la gráfica inferior, durante el mes de octubre ha sido cuando más dinero se ha invertido el adquisición de usuarios, por ese motivo tenemos muchas instalaciones no orgánicas.

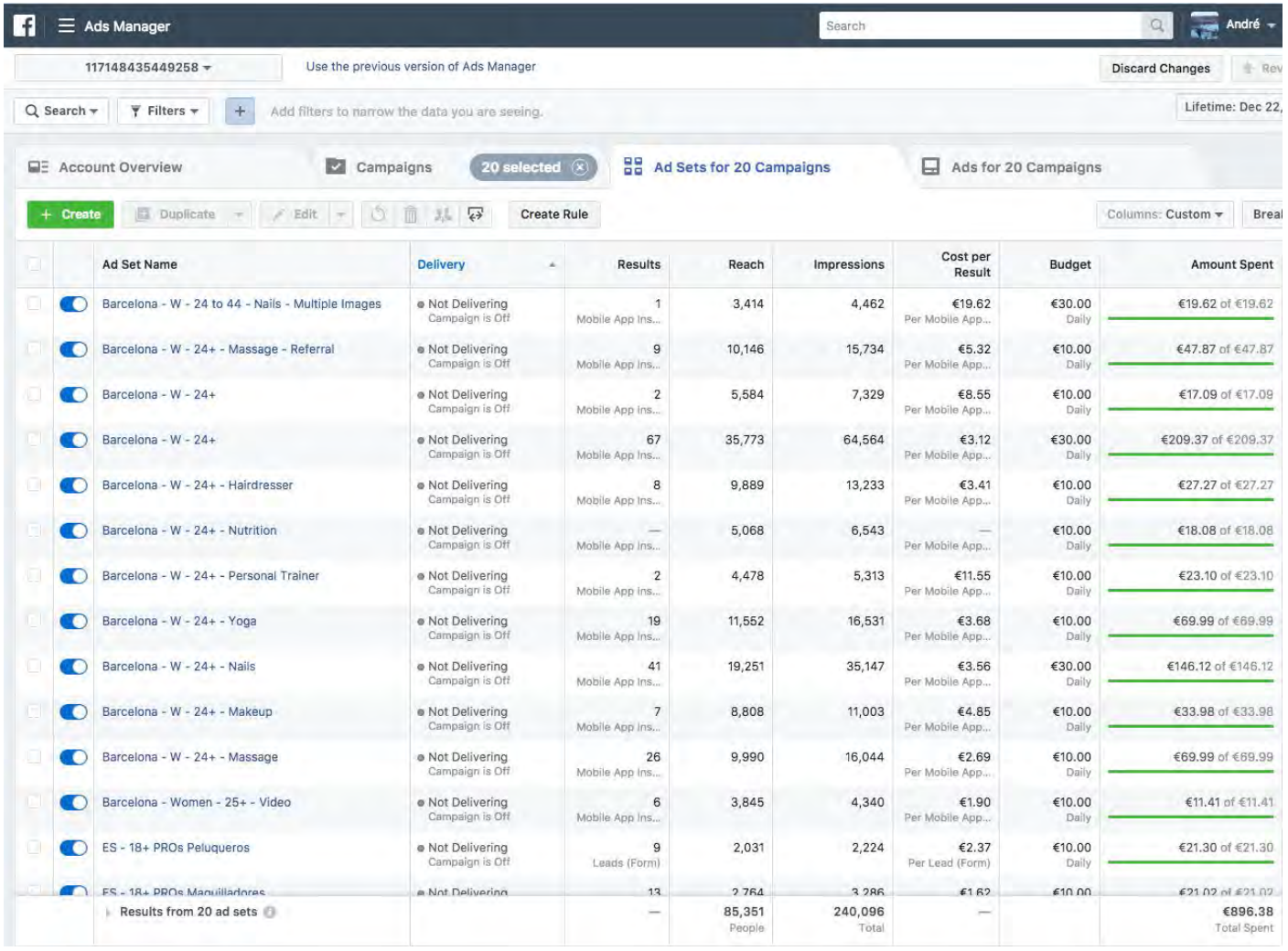


Tendencia de adquisición de usuarios, oct-nov (Appsflyer)



Instalaciones diarias, oct-nov (Appsflyer)

Panel de Facebook Ads con las respectivas campañas:



Ad Set Name	Delivery	Results	Reach	Impressions	Cost per Result	Budget	Amount Spent
Barcelona - W - 24 to 44 - Nails - Multiple Images	Not Delivering Campaign is Off	1 Mobile App Ins...	3,414	4,462	€19.62 Per Mobile App...	€30.00 Daily	€19.62 of €19.62
Barcelona - W - 24+ - Massage - Referral	Not Delivering Campaign is Off	9 Mobile App Ins...	10,146	15,734	€5.32 Per Mobile App...	€10.00 Daily	€47.87 of €47.87
Barcelona - W - 24+	Not Delivering Campaign is Off	2 Mobile App Ins...	5,584	7,329	€8.55 Per Mobile App...	€10.00 Daily	€17.09 of €17.09
Barcelona - W - 24+	Not Delivering Campaign is Off	67 Mobile App Ins...	35,773	64,564	€3.12 Per Mobile App...	€30.00 Daily	€209.37 of €209.37
Barcelona - W - 24+ - Hairdresser	Not Delivering Campaign is Off	8 Mobile App Ins...	9,889	13,233	€3.41 Per Mobile App...	€10.00 Daily	€27.27 of €27.27
Barcelona - W - 24+ - Nutrition	Not Delivering Campaign is Off	— Mobile App Ins...	5,068	6,543	— Per Mobile App...	€10.00 Daily	€18.08 of €18.08
Barcelona - W - 24+ - Personal Trainer	Not Delivering Campaign is Off	2 Mobile App Ins...	4,478	5,313	€11.55 Per Mobile App...	€10.00 Daily	€23.10 of €23.10
Barcelona - W - 24+ - Yoga	Not Delivering Campaign is Off	19 Mobile App Ins...	11,552	16,531	€3.68 Per Mobile App...	€10.00 Daily	€69.99 of €69.99
Barcelona - W - 24+ - Nails	Not Delivering Campaign is Off	41 Mobile App Ins...	19,251	35,147	€3.56 Per Mobile App...	€30.00 Daily	€146.12 of €146.12
Barcelona - W - 24+ - Makeup	Not Delivering Campaign is Off	7 Mobile App Ins...	8,808	11,003	€4.85 Per Mobile App...	€10.00 Daily	€33.98 of €33.98
Barcelona - W - 24+ - Massage	Not Delivering Campaign is Off	26 Mobile App Ins...	9,990	16,044	€2.69 Per Mobile App...	€10.00 Daily	€69.99 of €69.99
Barcelona - Women - 25+ - Video	Not Delivering Campaign is Off	6 Mobile App Ins...	3,845	4,340	€1.90 Per Mobile App...	€10.00 Daily	€11.41 of €11.41
ES - 18+ PROs Peluqueros	Not Delivering Campaign is Off	9 Leads (Form)	2,031	2,224	€2.37 Per Lead (Form)	€10.00 Daily	€21.30 of €21.30
ES - 18+ PROs Maquilladores	Not Delivering	13	2,764	3,286	€1.62	€10.00	€21.02 of €21.02
Results from 20 ad sets		—	85,351 People	240,096 Total	—		€896.38 Total Spent

10. Validación

Una vez empezamos a obtener los primeros resultados, nos dimos cuenta de que nos costaba mucho conseguir que el usuario acabará pagando por nuestros servicios. En general, el coste por instalación era bajo, de media unos 3€ e incluso alguna campaña llegamos a tener 2.69€ por instalación.

Pero el coste de adquisición era muy alto, analizando los motivos con grupos de usuarios llegamos a la conclusión de dos hechos importantes:

1. Precios demasiado altos
   Como el marketplace no tenía mucha oferta, no había competencia entre profesionales haciendo que los precios fueran más altos que en un centro. Además, el profesional dado que se desplazaba incluía ese desplazamiento en el precio.
   La solución era aumentar la oferta contratando a personal para la empresa que estuviera con nosotros todo el día, así podríamos bajar los precios y conseguir convertir más fácilmente.

2. Falta de confianza
   Necesitábamos hacer campañas de branding offline para que las personas probaran nuestros servicios. Al final, éramos una plataforma online pero con prolongación a contacto directo entre personas desconocidas y debíamos trabajar eso. La solución pasaba por inyectar más dinero en marketing para generar branding realizando eventos en la ciudad y consiguiendo que la gente nos conozca y pruebe.

Una vez llegado a este punto, en diciembre, expusimos al resto de socio la necesidad de inyectar más capital en la startup para dedicarlo a marketing además de la necesidad de contratar una persona con experiencia en marketing y en el sector belleza y bienestar.

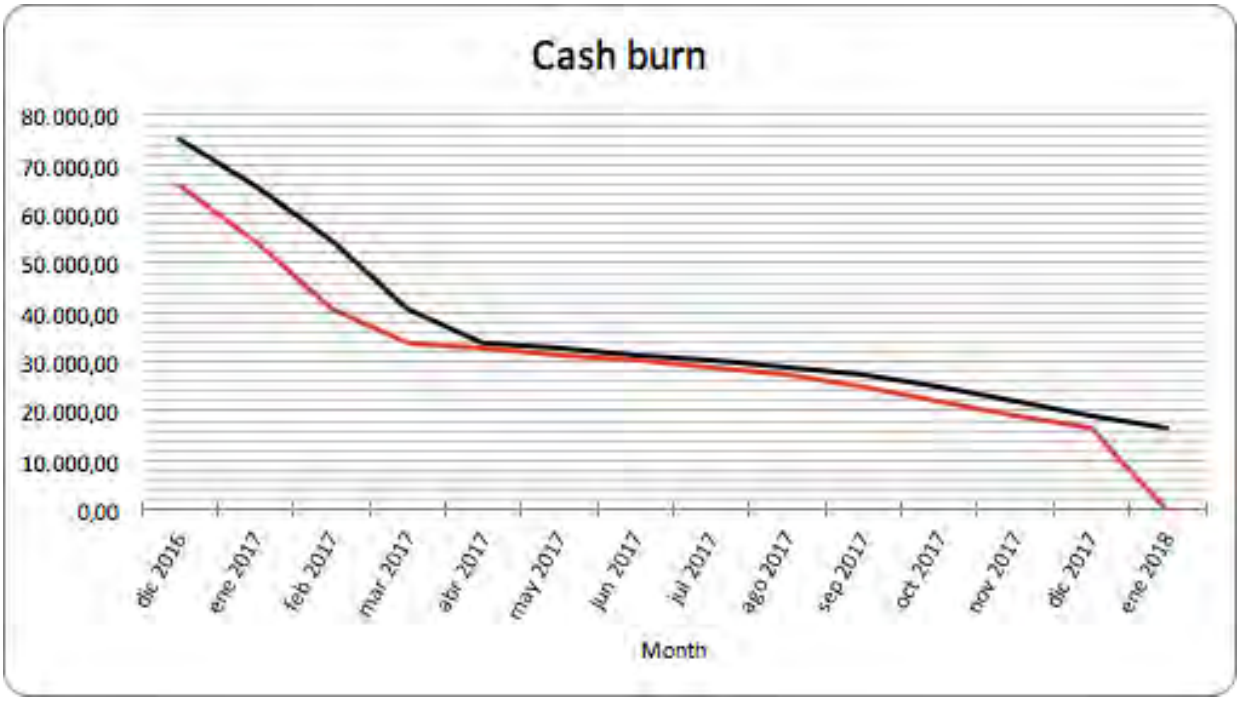
La propuesta fue descartada, con lo que como socio fundador de la startup decidí abandonar el proyecto. Tenía varios motivos como pérdida de ilusión, poca ayuda del socio mayoritario, pocos recursos disponibles, etc.

11. Costes asociados

La startup ha contado con un presupuesto de 75.000€ aportados por Nuclio Venture Builder. Como podemos observar se consumió gran parte de ese capital en los 4 primeros meses dado que se tenía que pagar las nóminas de un equipo sobredimensionado de 4 fundadores y 2 becarios.

	dic 2016	ene 2017	feb 2017	mar 2017	abr 2017	may 2017	jun 2017
Capital	75.000,00	65.620,80	54.618,80	40.606,37	33.916,17	32.578,81	31.419,81
Rest	65.620,80	54.618,80	40.606,37	33.916,17	32.578,81	31.419,81	30.160,81
REVENUES	0,00	6,61	67,77	0,00	0,00	0,00	0,00
Gross Sales	*0,00*	*40,00*	*82,00*	*180,00*	*0,00*	*0,00*	*0,00*
Net Sales	*0,00*	*33,06*	*67,77*	*148,76*	*0,00*	*0,00*	*0,00*
Take Rate	*0,2*	*0,2*	*1*	*0*	*0,2*	*0,2*	*0,2*
EXPENSES	9.379,20	11.008,61	14.080,20	6.690,20	1.337,36	1.159,00	1.259,00
Hr	7.974,20	8.374,20	11.374,20	5.884,20	900,00	900,00	900,00
*Employees***	*7.974,20*	*7.974,20*	*7.974,20*	*4.984,20*	*0,00*	*0,00*	*0,00*
*Interns***	*0,00*	*400,00*	*3.400,00*	*900,00*	*900,00*	*900,00*	*900,00*
Facilities	30,00	294,08	206,00	306,00	187,36	59,00	59,00
Phone	*30,00*	*30,00*	*30,00*	*30,00*	*30,00*	*30,00*	*30,00*
*Softwares market***		*240,08*	*147,00*	*147,00*	*128,36*	*0,00*	*0,00*
*Softwares Tech***		*24,00*	*29,00*	*129,00*	*29,00*	*29,00*	*29,00*
Marketing	0,00	497,29	2.000,00	0,00	0,00	0,00	0,00
Offline	*0,00*	*131,74*	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*
Flyers		*78,30*					
Others		*53,44*					
Online	*0,00*	*365,55*	*2.000,00*	*0,00*	*0,00*	*0,00*	*0,00*
Facebook	*0,00*	*245,55*	*2.000,00*	*0,00*	*0,00*	*0,00*	*0,00*
Adwords		*120,00*					
Instagram							
Twitter							
Others							
Biz Dev	0,00	405,54	0,00	0,00	0,00	0,00	0,00
Transport		*205,89*					
Massages		*130,00*					
Food		*15,00*					
Accomodation		*54,65*					
Central Services	1.375,00	1.437,50	500,00	500,00	250,00	200,00	300,00
Design	*900,00*	*762,50*	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*
Hours	*36,00*	*30,50*					
Price / hours	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*
HR	*0,00*	*175,00*	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*
Hours		*7,00*					
Price / hours	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*
Tech	*75,00*	*0,00*	*0,00*	*0,00*	*50,00*	*0,00*	*0,00*
Hours	*3,00*				*2,00*		
Price / hours	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*
Offices	*400,00*	*500,00*	*500,00*	*500,00*	*200,00*	*200,00*	*300,00*

jul 2017	ago 2017	sept 2017	oct 2017	nov 2017	dic 2017	ene 2018
30.160,81	28.831,81	27.266,98	24.810,31	21.882,99	19.138,33	16.393,66
28.831,81	27.266,98	24.810,31	21.882,99	19.138,33	16.393,66	
0,00	0,00	0,00	351,43	0,00	0,00	
0,00	*0,00*	*0,00*	*2.126,15*	*0,00*	*0,00*	
0,00	*0,00*	*0,00*	*1.757,15*	*0,00*	*0,00*	
0,2	*0,2*	*0,2*	*0,2*	*0,2*	*0,2*	
1.329,00	1.564,83	2.456,67	3.278,75	2.744,67	2.744,67	
500,00	1.170,83	1.841,67	1.841,67	2.241,67	2.241,67	
0,00	*920,83*	*1.841,67*	*1.841,67*	*1.841,67*	*1.841,67*	
500,00	*250,00*	*0,00*	*0,00*	*400,00*	*400,00*	
29,00	44,00	70,00	114,50	103,00	103,00	
0,00	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	
0,00	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	
29,00	*44,00*	*70,00*	*114,50*	*103,00*	*103,00*	
0,00	0,00	120,00	1.022,58	0,00	0,00	
0,00	*0,00*	*120,00*	*62,58*	*0,00*	*0,00*	
		60,00				
		60,00	*62,58*			
0,00	*0,00*	*0,00*	*960,00*	*0,00*	*0,00*	
0,00	*0,00*	*0,00*	*900,00*	*0,00*	*0,00*	
			60,00			
0,00	0,00	0,00	0,00	0,00	0,00	
800,00	350,00	425,00	300,00	400,00	400,00	
500,00	*0,00*	*125,00*	*0,00*	*0,00*	*0,00*	
20,00		*5,00*				
25,00	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	
0,00	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	
25,00	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	
0,00	*50,00*	*0,00*	*0,00*	*0,00*	*0,00*	
	2,00					
25,00	*25,00*	*25,00*	*25,00*	*25,00*	*25,00*	
300,00	*300,00*	*300,00*	*300,00*	*400,00*	*400,00*	



12. Conclusiones

Realmente, la creación de Kynda me ha puesto a prueba. No ha sido fácil compaginar 8h laborales con la realización de este máster pero he aprendido muchísimo. Siempre había querido crear un producto tecnológico de inicio a fin: identificar una idea, elaborar los diseños, implementar el código y desplegarlo en un entorno de producción.

La aplicación de Lean Startup y sus metodologías ágiles ha sido un gran acierto, el mundo startup me apasiona y quería probarlas en este proyecto. Es una metodología que se debería empezar a introducir en empresas tradicionales, sé que es un proceso complicado pero les aportará mucho valor.
Lean es el camino a seguir en el desarrollo de cualquier proyecto tecnológico, si no sabes qué problema estás resolviendo, las horas que habrás dedicado a programar no habrán servido para nada. Es fundamental ir prototipando, y realizando tests en usuarios.

La realización de este proyecto también ha hecho que me de cuenta de lo importante que es el trabajo en equipo, lo importante de tener un equipo multidisciplinar y capaz de afrontar situaciones límite. La formación del equipo es una actividad de recursos humanos muy complicada, es clave conocer bien a los miembros, tanto sus actitudes como aptitudes y evaluar si ese perfil es adecuado para comenzar un proyecto. Hay profesionales muy buenos en el mundo corporate que no funcionan en una startup y viceversa.

Después de la experiencia en mi anterior empresa, Kerad Games, donde estaba inmerso en el equipo técnico, allí aprendí a crear software y producto tecnológico de alto nivel aplicando metodologías ágiles, me di cuenta que necesitaba una aventura para conocer otros campos como el marketing digital o operaciones, sabía que en una startup tendría esa oportunidad y así ha sido.

Una vez terminada mi etapa en Kynda, tengo claro que quiero continuar creando nuevos negocios digitales ya que es lo que me apasiona verdaderamente.

Referente a la elaboración de este documento, me ha costado sintetizar y resumir un proyecto de tal dimensión. Hay muchísimas cosas que no he explicado aquí, ha sido difícil llevarlo a papel e hacer que se entienda, tenia muchas dudas sobre qué conceptos explicar y cuáles no. Al final he optado por hacer una explicación general de todo lo que se ha realizado sin entrar el mucho detalle.

Quería agradecer a Miguel Ramirez por aceptar mi proyecto de Kynda como trabajo final ya que con la carga de trabajo no hubiera podido compaginarlo con otro.

Gracias por todo!

13. Bibliografía

https://golang.org
https://nachopacheco.gitbooks.io/go-es/content/doc/index.html
https://www.facebook.com/business/products/ads
https://beego.me/docs/intro/
https://vuejs.org/v2/guide/
https://stackoverflow.com/