Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

ADVERSPOT:
APP en Transporte Público

Proyecto Final de Carrera

Ingeniería Informática

Autor: Jaime Sierra Rodríguez

Director: Patricio Letelier

Septiembre 2015

Dedicatoria

Dedicado a Patricio, mi motivador profesor y tutor por ayudarme a abordar este proyecto. A mis compañeros y amigos de este gran e ilusoria aventura en el que locamente nos hemos embarcado, David, Javier e Ignacio. A mi madre por darme la vida y apoyarme en todo lo que he hecho aunque estuviese equivocado. A mi hermana porque es mi ojito derecho y mi debilidad. A Santiago porque eres un gran amigo. Y por supuesto a Paige, *my hope and my heart*. Gracias a todos por estar en mi vida.


TAXI
ADVERSPOT

Resumen

El siguiente documento relataremos nuestra experiencia al desarrollar una APP para dispositivo móvil incoporada en un transporte público, como puede ser un taxi. Consta de los siguientes capítulos: Idea (Lean Canvas, DAFO, Contexto...), Producto (Diseño, Opciones Descartadas, Funcionalidad, Arquitectura...), Plan de Negocio (Ingresos, Gastos...) y Conclusiones.

Desarrollaremos nuestra idea de negocio para comprobar y comparar que funcionalidades ofrecen otros productos del mercado, y que, aunque no serían competencia directa, sí podrían ser complementarias a nuestra idea. Capítulo 1.

Además de realizar un Canvas y un MoSCoW con las funcionalidades críticas, necesarias, opcionales y descartadas. Detallaremos en este apartado cada una de las funcionalidades que ofrece el producto al usuario (pasajero) de los vehículos. En este apartado explicaremos el reto tanto tecnológico como físico (soporte e infraestructura) al que nos hemos enfrentado. Capítulo 2.

En el siguiente apartado analizaremos la viabilidad del proyecto a la hora de implantarlo y que sea rentable realizando una balanza de gastos y de ingresos. El modelo de negocio será financiado mediante publicidad implantada dentro de la propia APP que manejará el dispositivo en todo momento. Capítulo 3.

Tras todo esto, realizaremos un apartado de conclusiones donde detallaremos el resultado final del producto y las impresiones percibidas al realizar el despliegue real en varios vehículos para comprobar el comportamiento del sistema tanto a nivel funcional como la aceptación por parte del sector del transporte público y de los usuarios. Capítulo 4.

Índice general

Capítulo 1

De la Idea al Producto

En este capítulo analizaremos las características del producto desde el punto de vista de la calidad de la idea, es decir, que tan bueno es el producto y cual es la viabilidad de nuestra StartUp [1].

[1] StartUp: es un término utilizado actualmente en el mundo empresarial el cual busca arrancar, emprender o montar un nuevo negocio y hace referencia a ideas de negocios que están empezando o están en construcción, es decir son empresas emergentes apoyadas en la tecnología.

1.1 Descripción de la Idea

Para explicar nuestra idea de negocio debemos explicar el contexto en el que surgió:

Todo empezó durante una conversación distendida entre amigos cuando uno de ellos pensó en voz alta una idea que le quitaba el sueño. Él había trabajado en el sector del taxi durante unos años. Además, es consciente del auge del turismo en España durante los últimos años que representa ya el *10 %* del **PIB**.

Nos esbozó su idea en la que básicamente quería incorporar un dispositivo que consiguiera reproducir publicidad para fomentar lo negocios locales y además que fuese interactiva para el usuario del Taxi para dotarlo de funcionalidades que le ayudasen a moverse por la ciudad. Rápidamente pensamos en una tableta o *tablet*.

Poco a poco la idea fue tomando forma enumerando ciertas funcionalidades de las cuales podría disponer la `APP móvil` y como se abordarían tecnológicamente, que detallaremos más adelante.

1.2 Adverspot Lean Canvas

En esta sección vamos a explicar la metodología que se sigue cuando se desea crear una StartUp, es decir, una empresa con una idea de negocio en el sector tecnológico y que. por ello, puede tener un crecimiento exponencial. Una de los requisitos recomendables es la creación de un Lean Canvas o lienzo Lean, el cuál nos dará una visión global de si nuestra idea es potente para la creación de un plan de negocio. Explicaremos los puntos o parcelas dentro de este lienzo, figura 1.1:

- ***Problema***: Nuestro problema a resolver consiste en la dificultad que los turistas tienen a la hora de comunicarse con los taxistas cuando visitan una ciudad, así como buscar el acceso a internet y el conocimientos de lugares de interés y negocios cercanos.
- ***Solución***: La solución pasa por un dispositivo con una APP interactiva que les permita comunicarse con el taxista y que tenga una lista de lugares de interés a los que puedan acudir.
- ***Métricas Clave***: Las métricas clave para nosotros son: el número de personas que suben a un taxi, el número de dispositivos colocados en taxis o el número de taxis operativos para nosotros y el número de anuncios que se reproducen en ellos.
- ***Ventaja Injusta***: Nuestra idea que marca la diferencia y no es fácil de copiar es que la publicidad es dinámica y se descargará una lista de videos actualizada, además de un balanceo entre los dispositivos con déficit de tiempo reproducido y los que tienen un ritmo aceptable de reproduciones.
- ***Canales***: nuestro camino hacia el consumidor son el cara a cara mediante las cooperativas de taxi, la que realicen los hoteles ofreciendo servicio de transporte a los turistas y el de nuestro dispositivo cuando la gente suba al taxi.
- ***Segmento de clientes***: Nuestros segmento de clientes está enfocado a turistas que utilizan el taxi.
- ***Características principales de los clientes***: Qué sean turistas y en edades comprendidas entre los 18 y los 40 (son los usuarios que más fácilmente utilizan *smartphones*).
- ***Coste de la estructura***: Nuestros costes fijos serían: servidor, dominio web y Seguro de responsabilidad civil. Los variables dependientes del número de taxis en los que se instale el producto: tablet, cableado, soporte y SIM móvil.
- ***Las fuentes de ingresos***: Este será la publicidad que se reproduzca en los dispositivos.

- ***Única frase de valor***: Es un servicio para turistas que puedan consultar una guía sobre la ciudad, sitios interesantes sobre ella y el acceso a internet.
- ***Conceptos de alto nivel***: Para nosotros hay dos cálculos importantes. "1 segundo = 1 céntimo", "X céntimos = Y segundos * Z reproducciones * V taxis".

Para más información puede consultar nuestro Anexo A, que explica el origen y uso del Lean Canvas.

Adverspot - Main

PROBLEM	SOLUTION	UNIQUE VALUE PROPOSITION	UNFAIR ADVANTAGE	CUSTOMER SEGMENTS
No Mobile Network Service Foreign Language Tourist References	Translator on APP Device with internet Guide with lots of language	A service for a tourists to visit city and some interesting places inside.	Video synchronization between the server and clients	People who going up to our Taxis, especially who are tourist
EXISTING ALTERNATIVES	**KEY METRICS**	**HIGH-LEVEL CONCEPT**	**CHANNELS**	**EARLY ADOPTERS**
Papper Guides Roaming Service Dictionaries or Translator APPs(with roaming)	Number of devices installed in Taxis Number of companies advertised Number of persons who going up to our Taxis	1 second = 1 cent x cents = y seconds * z replays * v taxis	When they going up to Taxi When they ticket in a Hotel Taxi Cooperatives	Tourist Age between 18 to 40 years old
COST STRUCTURE			**REVENUE STREAMS**	
Tablet Mobile SIM Wires 3D printed support Liability insurance Web Domain Server			Publicly	

Lean Canvas is adapted from The Business Model Canvas (BusinessModelGeneration.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License.

Figura 1.1: Adverspot Lean Canvas

1.3 DAFO

Según las características que tiene nuestra empresa deberíamos adoptar una estrategia óptima para el desarrollo y supervivencia de esta. El DAFO ayuda a plantearnos las acciones que deberíamos poner en marcha para aprovechar las oportunidades detectadas y eliminar o preparar a la empresa contra las amenazas, teniendo conciencia de nuestras debilidades y fortalezas.

Fijados los objetivos –que deben ser jerarquizados, cuantificados, reales y consistentes–, elegiremos la estrategia para llegar a ellos mediante acciones de marketing. Repasemos las posibles estrategias con ejemplos:

- **Defensiva**: La empresa está preparada para enfrentarse a las amenazas.Si tu producto ya no se considera líder, resalta lo que te diferencia de la competencia. Cuando baje la cuota de mercado, busca clientes que te sean más rentables y protégelos.
- **Ofensiva**: La empresa debe adoptar estrategias de crecimiento. Cuando tus fortalezas son reconocidas por los clientes, puedes atacar a la competencia para exaltar tus ventajas (por ejemplo: el 83 % prefieren el producto X) . Cuando el mercado está maduro, puedes tratar de robar clientes lanzando nuevos modelos.
- **Supervivencia**: Te enfrentas a amenazas externas sin las fuerzas internas necesarias para luchar contra la competencia. Deja las cosas como están hasta que se asienten los cambios que se producen (por ejemplo: observa la internetización del entorno antes de lanzarte a la red).
- **Reorientación**: Se te abren oportunidades que puedes aprovechar, pero careces de la preparación adecuada. Cambia de política o de productos porque los actuales no están dando los resultados deseados.

El DAFO que construimos para el producto Adverspot fue el de la figura 1.2.

- **Análisis Externo**: La organización no existe ni puede existir fuera de un entorno, fuera de ese entorno que le rodea; así que el análisis externo permite fijar las oportunidades y amenazas que el contexto puede presentarle a una organización.

  El proceso para determinar esas oportunidades o amenazas se puede realizar de la siguiente manera:

  - ***Oportunidades***: Las oportunidades son aquellos factores, positivos, que se generan en el entorno y que, una vez identificados, pueden ser aprovechados.

    Las oportunidades de nuestro producto son la falta de competencia pues es un producto innovador, el sector del Taxi desea introducir las TIC en su sector, el sector del Taxi está interesado en potenciar los servicios turísticos que ofrece el servicio, un apoyo institucional que apuesta por

ANÁLISIS INTERNO	ANÁLISIS EXTERNO
Debilidades	**Anemazas**
Falta de fondos	Sector del Taxi en crisis (Blablacar, ...)
Falta de experiencia gerencial	Sector del Taxi poco confiado
Falta de inversores	
Fortalezas	**Oportunidades**
Un atractivo o incentivo al uso del taxi	No existe competencia a nivel local
Buen equipo técnico	Deseo de introducir las TIC en transporte público
Producto mínimo funcional terminado	Simbiosis con el gremio de taxis para publicitar el producto
	Insertar un producto novedoso en el sector
	Apoyo institucional hacia el turismo
	Sector servicios en auge

Figura 1.2: Adverspot DAFO

el turismo como motor económico en el país y un sector servicios en auge que ya supone más del 10 % PIB.

- ***Amenazas***: Las amenazas son situaciones negativas, externas al programa o proyecto, que pueden atentar contra éste, por lo que llegado al caso, puede ser necesario diseñar una estrategia adecuada para poder sortearlas.

  Como amenazas para nuestro producto está la crisis que atraviesa el sector de transporte público, en particular el Taxi a causa de la competencia desleal de conductores sin licencia utilizando programas de contacto y petición de vehículos privados que los transporte y el otro punto es que el sector del Taxi ya está escarmentado con ciertos productos con poco éxito e que no cumplian las expectativas o que simplemente fueron estafas.

- **Análisis Interno**: Los elementos internos que se deben analizar durante el análisis DAFO corresponden a las fortalezas y debilidades que se tienen respecto a la disponibilidad de recursos de capital, personal, activos, calidad de producto, estructura interna y de mercado, percepción de los consumidores, entre otros.

  El análisis interno permite fijar las fortalezas y debilidades de la organización, realizando un estudio que permite conocer la cantidad y calidad de los recursos y procesos con que cuenta el ente.

  - ***Fortalezas***: Las fortalezas son todos aquellos elementos internos y positivos que diferencian al programa o proyecto de otros de igual clase.

    Para las fortalezas diremos que el sector Taxi está ilusionado con implantar un producto como el nuestro para el uso en sus vehículos, potenciando la comunicación con el cliente y ofreciendo unos servicios de guía que este podría solicitar/utilizar. El otro punto es que ya tenemos un producto mínimo funcional, incluso algunas funcionalidades extra que lo hacen atractivo como es el traductor con `Speech` y que puede ser usado de forma hablada, y para acabar, nuestro equipo técnico tiene una larga trayectoria en desarrollo de software en distintas plataformas.

  - ***Debilidades***: Las debilidades se refieren a todos aquellos elementos, recursos de energía, habilidades y actitudes que la empresa ya tiene y que constituyen barreras para lograr la buena marcha de la organización. También se pueden clasificar: aspectos del servicio que se brinda, aspectos financieros, aspectos de mercado, aspectos organizativos, aspectos de control.

    Por contra partida, las debilidades de nuestra organización son la falta de experiencia en el punto de vista de la gerencia de una empresa, otro punto sería la falta de financiación para el proyecto y muy relacionada con esta la falta de inversores que quieran apostar por nuestro producto.

1.4 Competidores

Uno de los puntos a estudiar eran los posibles competidores que habrían en el sector. Quizá parezca una idea innovadora pues nunca se ha intentado algo tan ambicioso como nuestro producto pero si podemos encontrar empresas que se dedican a realizarlo con otros métodos pues la publicidad en tanto en el interior o como exterior del taxi existe desde prácticamente la historia de este.

Uno de los métodos es colocar la publicidad en pegatinas. Podemos verlo la siguiente figura 1.3.

Figura 1.3: Publicidad Taxi Pegatina.

Con un aspecto más que aceptable la pegatina llama la vista de los viandantes que fortuitamente quieran dedicarle unos momentos a mirar la superficie del taxi. Siempre que además se encuentre parado si no, será prácticamente imposible enterarse de algo.

Otro de los métodos utilizados, aunque menos frecuente, es colocar publicidad en un cartel, incluso algunos rotatorios. Al igual que el método de la pegatina, este tiene el inconveniente de que son los usuarios de fuera del taxi los que reciben la publicidad y el vehículo ha de estar parado. Por otro lado parece ser que es más fácil de instalar pero en contra partida la superficie es más limitada que el propio coche.

Podemos ver uno en la figura 1.4.

Pero ambos métodos tienen un inconveniente. La publicidad es visualmente estática y limitada en superficie.

La legislación debe de limitar el tamaño de estas instalaciones.

Figura 1.4: Publicidad Taxi Cartel.

Pero por supuesto lo que nos ha llamado la atención, y quizá sea un paso anterior al nuestro, es digitalizar la publicidad. Están apareciendo compañías que instalan marcos electrónicos en el reposacabezas incrustado. Esto permite tener una parrilla de anuncios reproducciéndose constantemente.

Tenemos el enlace de una de estas empresas en la cita **Luarstudio** Si nos fijamos en la figura 1.5, que está sacada de su página web, podemos fijarnos en que efectivamente utilizan estos dispositivos para publicitar a sus clientes.

Esto tiene ventajas sobre la publicidad estática y exterior mencionadas anteriormente, al igual que si la comparamos con la de nuestro producto.

Figura 1.5: Publicidad Taxi Marco Digital.

Comparándola con `Adverspot`, tenemos que:

- la publicidad no se puede sincronizar en el instante. Se debe sacar la unidad de almacenamiento para poder actualizar la publicidad.
- el dispositivo es una "*caja tonta*", es decir, no se puede interactuar con ella. Para el usuario no ofrece ningún tipo de servicio ni para el taxista ninguna ventaja frente a la competencia.
- el taxista ha de cambiar su reposacabezas estándar para incorporar el dispositivo.
- acceso a navegador web para los usuarios.

- traducción en varios idiomas para la comprensión tanto de publicidad mediante los subtítulos como la elección de idiomas, mencionando también el traductor.

  Y fijándonos en los costes de este tipo de dispositivos comerciales se equiparan al coste monetario que debemos invertir por cada unidad de nuestro producto.

Capítulo 2

Producto

En este capítulo analizaremos las características del producto desde el punto de vista de sus características técnicas, tanto en diseño, arquitectura, funcionalidad y un punto sobre ideas analizadas y desechadas para realizar un producto atractivo pero también mínimo viable sin derrochar el esfuerzo que supondría implementar todas las buenas ideas que tuvimos desde el principio para nuestro producto.

2.1 Diseño

Cuando nos planteamos el producto desde la idea supimos que deberíamos afrontar ciertos retos o al menos intentábamos cosas que, aun siendo una quimera entre varias que ya existían, no se habían planteado en este contexto, por lo que cualquier referente anterior de poco nos servía.Así que le dimos rienda suelta a nuestra imaginación.

Lo primero fue pensar en el soporte físico y las prestaciones que necesitaba para poder llevar a cabo nuestro ambicioso plan: pantalla táctil y de dimensiones considerables, 3G, *GPS*, cámara aceptable, ... hablamos de ello en la **sección 2.1.1**.

Después tuvimos que pensar en como alimentaríamos el dispositivo puesto que al ir integrado en un vehículo, este debería proveerlo de energía, es decir, debemos enchufar nuestra *tablet* a la batería del taxi. Este se explica en la **sección 2.1.2**.

Debíamos hacer una interfaz que permitiera la buena visualización de los anuncios además de una navegación óptima e intuitiva para el usuario. Todo ello se detalla en la **sección 2.1.3**.

Además debíamos proteger el dispositivo de posibles daños tanto por parte de los usuarios hacia este como desde el dispositivo a los usuarios. Al final optamos por una solución óptima entre calidad y precio. Todo ello en la **sección 2.1.4**.

2.1.1 Hardware

Cuando pensamos en un dispositivo que no solo reprodujese publicidad de forma ininterrumpida rápidamente pensamos en una *tablet* puesto que además de un tamaño considerable para la visualización del anuncio también dispondremos de espacio para los menús de las distintas funcionalidades que se le ofrece al cliente.

Por sus características y su precio al final nos decantamos por el modelo `BTPC-1015QC-3G` de la marca `Brigmton`, ver **figuras 2.1**, **2.2** y **2.3**, ya que disponía de 3G para ser conectada a internet en cualquier parte necesario para varias de sus funcionalidades, que explicaremos más detalladamente en la sección 2.3, y su bajo coste.

Sus *Especificaciones* son:

- Procesador MT8382 QUAD CORE 1.5GHZ Cortex A7
- Android 4.2.2
- Memoria RAM 1GB DDR3

Figura 2.1: Brigmton BTPC-1015QC-3G Delantera.

- Memoria Interna 8GB
- Pantalla 10.1" 1024x600 multitáctil capacitiva
- Controlador Grafico MALI400-MP1 OpenGL ES2.0
- Formatos Soportados
  - Formatos de vídeo: AVI, MEPG 1/2/4, H.263/H.264, RMVB, WMV/VC-1, MVC, AVS, etc.
  - Formatos de audio: MP3, WMA, MP2, OGG, AAC, MA4, FLAC, APE, 3GP, WAV
  - Formatos imagen: BMP, JPG, JPEG, GIF,PNG
  - Función Ebook: TXT, EPUB, PDF, WORD
- Conectividad
  - 3G: GSM 850/900/1800/1900 - WCDMA 2100
  - Doble entrada de tajetas SIM
  - Función teléfono

Figura 2.2: Brigmton BTPC-1015QC-3G Trasera.

  - Wifi: 802.11b/g/n
  - Bluetooth integrado V4.0
- Cámara
- Cámara frontal 0.3Mp
- Cámara trasera 2Mp
- Micrófono
- Altavoz Altavoz incorporado: 8/1W
- Batería de litio recargable: 6000mAh
- Conexiones
  - Entrada de auriculares jack 3.5 mm
  - Jack con función llamada de teléfono
  - Entrada USB OTG

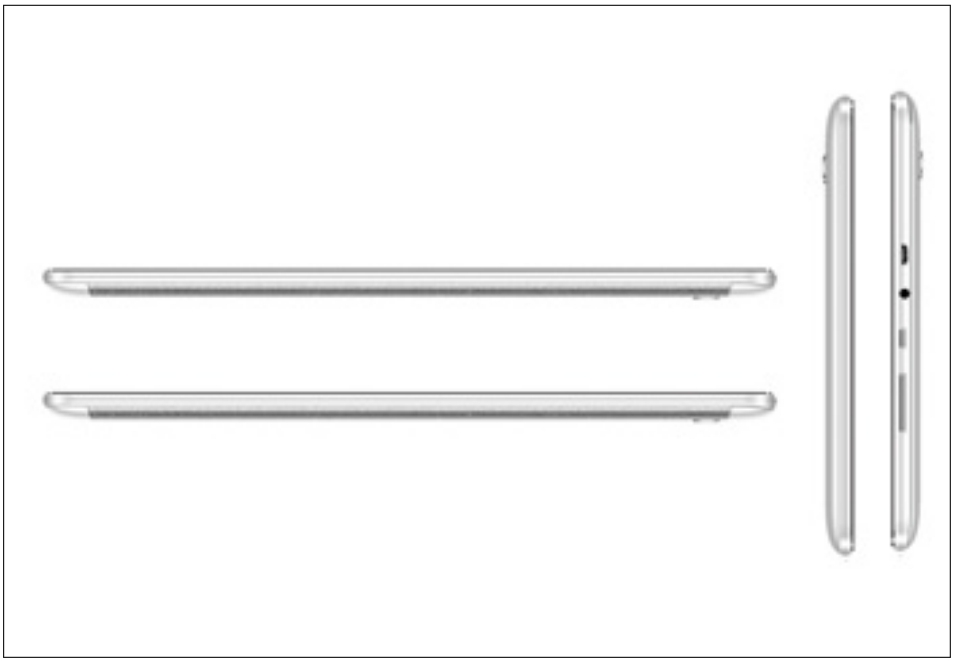

Figura 2.3: Brigmton BTPC-1015QC-3G Lateral.

- Sistema operativo Android 4.2.2 Jelly Bean
- Dimensiones 260x165x10.5mm.
- Peso 605 gr.
- Color Blanca

Más información en la cita **Brigmton**

2.1.2 Alimentación: Conexiones & Cableado

Después de mucho pensar en cual sería la mejor forma de que cables serían los óptimos, para que la *tablet* no sufriese un apagón inoportuno por falta de alimentación, pensamos en un que lo primero que necesitábamos era un cable en ángulo que permita sacarlo desde la `orejera` del soporte hacia abajo de este por lo que no nos servía un cable *micro-USB* convencional debía ser en ángulo de 90º y además izquierdo porque la interfaz USB no es simétrica, figura 2.4.

Sus características son:

- **longitud**: 1 metro.
- **versión**: 2.0.
- **interfaces**: microUSB macho - USB macho.

Si desea obtener este artículo 2.4 puede encontrarlo en el siguiente enlace **LeftAngled**

Figura 2.4: Cable Micro USB de ángulo izquierdo 90º.

Si la *tablet* iba a ser desenchufada, transportada y guardada cada vez que el conductor acababa su jornada, por motivos de seguridad (recordemos en figura 2.9 que el soporte tenía un candado para separar el `sobre` del brazo que lo sujeta al reposacabezas), entonces debíamos de disponer de un cable que hiciese puente entre el cable de 90º grados, figura 2.4, y el enchufe del mechero, figura 2.6. Además calculando la distancia desde el reposacabezas y metiendo los cables por dentro del asiento del copiloto por el suelo y hasta llegar al conector de mechero del coche el cálculo era de entre 2 y 3 metros por lo que este alargador USB, figura 2.5 era necesario.

Sus características son:

- **longitud**: 5 metros.
- **versión**: 2.0.
- **interfaces**: USB hembra - USB macho.

Si desea obtener este artículo 2.5 puede encontrarlo en el siguiente enlace **USBExtender**

Para finalizar, pensamos en que no debíamos perjudicar los conectores de mechero del conductor y si necesitaba un acceso a la alimentación para otro

Figura 2.5: Cable alargador USB.

dispositivo como puede ser un teléfono móvil, un *GPS* o cualquier otro que ya utilizaba debíamos proporcionarle un enchufe extra para este. Optamos entonces por un "`ladrón` o cargador DUAL para USB figura 2.6.

Sus características son:

- **versión**: 2.0.
- **interfaces**: dos USB hembra.
- **Intensidad**: 2.1 A.
- **Voltaje**: 5 V.

Si desea obtener este artículo 2.6 puede encontrarlo en el siguiente enlace **DualUSBCharger**

Figura 2.6: Cargador Dual USB para coche.

2.1.3 Interfaz Gráfica de Usuario

Cuando nos pusimos a diseñar la interfaz teníamos clara una cosa, la publicidad nunca debía desaparecer de la visión del usuario, por lo que pensamos en posicionar la *tablet* con vista apaisada (o *landscape*. Por lo que podríamos dividir la pantalla entre publicidad e interfaz interactiva, véase figura 2.7.

Figura 2.7: Pantalla principal de Adverspot.

Cuando arrancamos la aplicación tenemos esta pantalla principal que divide la vista en 2 `layouts`. El izquierdo contiene la publicidad que se controla vía servidor mediante la descarga de una lista de reproducción. El derecho contiene toda la funcionalidad que el usuario puede manejar. Si pulsamos sobre cualquiera de los iconos centrales la pantalla se desliza mediante una animación abarcando la mitad de la pantalla. Por lo que el panel izquierdo también abarcará la mitad de la pantalla reduciéndose la superficie del vídeo.

Toda la navegación por los distintos paneles se produce con animaciones de deslizado. En la parte de arriba tenemos la hora actual en un reloj digital. En la parte inferior una lista con banderas que indican el idioma al que se desea configurar la aplicación. El idioma puede ser cambiado en cualquier momento. Para más información diríjase a la sección 2.3.

Y una cuestión a resaltar es que si el usuario no interactúa con el dispositivo este entrará en *standby*, es decir, se lanzará un fondo de pantalla posible-

mente con el `logo` personalizado o si no está disponible, uno estándar de nuestro producto. En este estado se entiende que la publicidad no realizará impactos por lo que no se reproducirán anuncios publicitarios.

Si queremos interactuar con el dispositivo tan solo debemos pulsar la pantalla y la publicidad se reanudará. Otra posibilidad es que el usuario sea detectado mediante su rostro. Para saber más acerca de esto consulte la sección 2.3.7.

Puede consultar también la sección de publicidad en 2.3.8.

2.1.4 Soporte

Después de barajar muchas posibilidades nuestra decisión se decantó por el de impresión 3D. Con la ayuda de un ingeniero industrial y una impresora 3D, se realizó un modelo 3D, que puede ser observado en las figuras 2.8 y 2.9, mediante un programa informático ajutado a las medidas de la tablet con ciertos detalles.

El soporte está compuesto por 7 piezas, impresas a medida para nuestra *tablet*, fabricadas con un plástico de resina resistente:

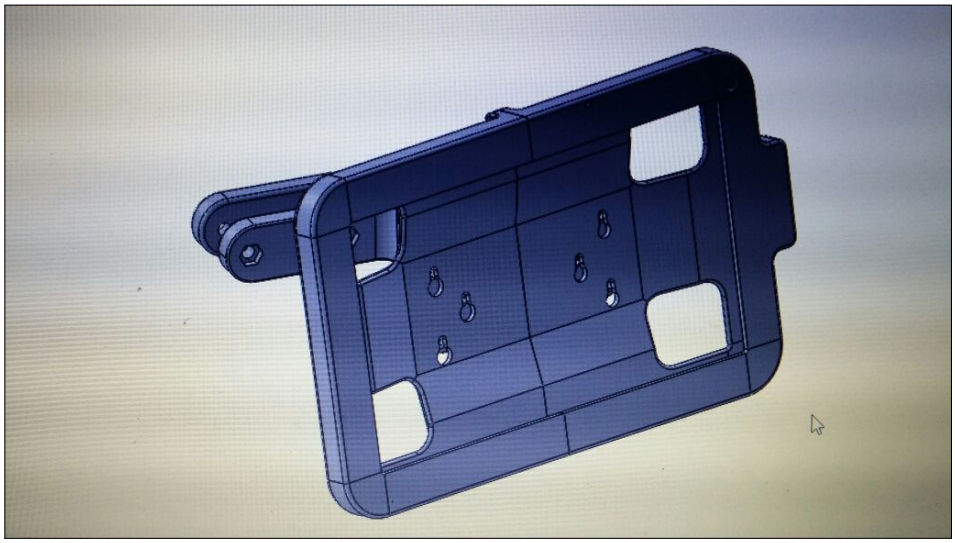

Figura 2.8: Soporte 3D.

1. Compuesto por 2 para el "`sobre`"que contendrán la *tablet* protegida salvo por la parte frontal par que el usuario pueda manipularla. Cada una de estas piezas contiene 3 agujeros en forma de lágrima que se usarán para encajar los tornillos que permitirán unirlas con la otra pieza en forma de brazo. Además una de las piezas contiene un saliente u "`orejera`"que permitirá tener el cable enchufado o apagarla/encenderla en caso de emergencia.
2. Otras 2 piezas están atravesadas por 3 tornillos con rosca que son los que permiten acoplar el brazo a la carcasa o sobre de la *tablet* tan solo encajando y deslizándolos por los agujeros en forma de lágrima y estás a su vez están sujetas y unidas por una pieza central que contiene un

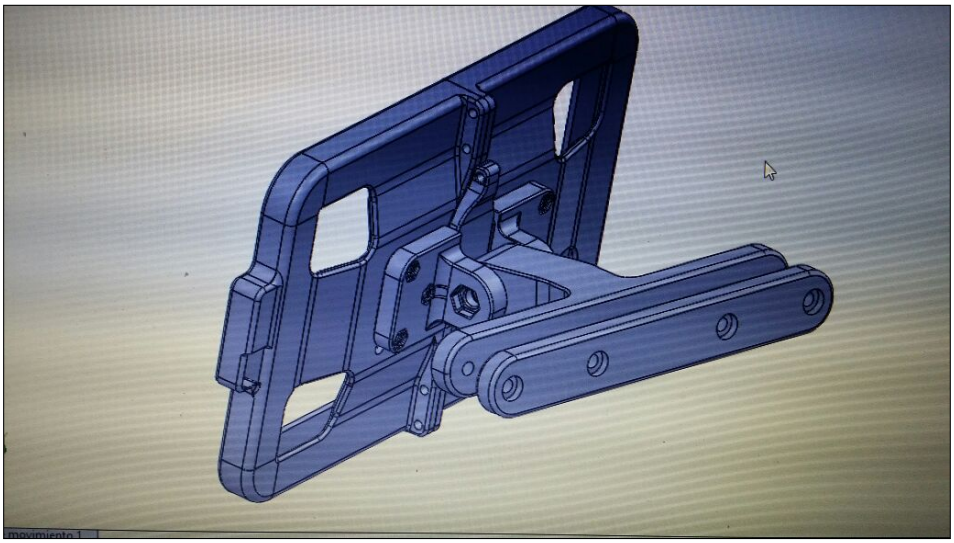

Figura 2.9: Soporte 3D Back.

pasador con un tornillo más grande y que forma el brazo del soporte para ser sujetado sobre el reposa-cabezas.

3. Esta pieza en forma de **T** central que contiene un gran tornillo con cabeza hexagonal puede ser aflojado para permitir una rotación de inclinación (arriba-abajo) aproximada de 180º. Esta pieza contiene 4 agujeros por los que se pasarán unos tornillos con rosca que junto con otra pieza se encargará de sujetar el soporte al reposa-cabezas del coche.

4. Una pieza de idéntica al extremo de la anterior encaja los 4 tornillos antes mencionados para que el soporte quede bien sujeto en el reposa-cabezas.

5. Por último mencionamos la pieza más pequeña pero no menos importante que se coloca en el centro del soporte, justo detrás del "`sobre`ÿ entre las dos piezas que unen el brazo con este y que sirve como medida de seguridad pues sobre unos agujeros perforados sobre el y sobre las piezas del "`sobre`"se puede pasar un candado, facilitando también la separación de este y el brazo, con lo que el taxista podrá llevarse el "`sobre`"que contiene la *tablet* a un lugar seguro, dejando tan solo la parte del brazo.

El tiempo de disponibilidad es de 2 días para fabricar las piezas. Se han de fabricar las piezas con una impresora 3D por separado y después lijarlas para pulir su superficie.

Después de haberlo presentado podemos enumerar algunas características positivas sobre el soporte:

- Su diseño está adaptado a nuestro dispositivo al mm.
- Su coste es muy reducido, el material utilizado cuesta menos de 10.
- Se puede corregir y personalizar el modelo 3D tantas veces como queramos, color, pintar, añadir dibujos sobre la superficie, ...
- Es ligerísimo y poco invasivo.

Y como nada es perfecto también tenemos algunos inconvenientes o aspectos negativos del soporte, realmente solo hemos encontrado uno evidente:

- El material no es tan resistente como el metal por lo que podría partirse sobretodo en su punto más débil, la unión entre el sobre y el brazo.

2.2 Opciones Descartadas

En esta sección trataremos de enumerar ciertas cuestiones sobre el diseño del producto como sobre funcionalidades que se han deshechado por ser muy laboriosas para un `mínimo producto viable`, por haber encontrado una alternativa mejor o por ser inviable.

Primero hablaremos acerca de una cuestión de diseño del producto, en concreto de un soporte metálico el cual pedimos desde China para testearlo. Para más información consultar sección 2.2.1.

Después hablaremos sobre una funcionalidad muy seductora como es utilizar el dispositivo como un `HotSpot` móvil, esto es, un punto de acceso público a internet. Se detalla en la sección 2.2.2.

Otra de las funcionalidades descartadas para el producto que supondría ingresos mediante la publicidad es la de utilizar nuestro `HotSpot` como una plataforma de publicidad dentro de los navegadores de los dispositivos de nuestros clientes cuando se conectasen a él para utilizar nuestro servicio de internet. Vea más detalles sobre esto en la sección 2.2.3.

Para terminar también tuvimos en mente implementar otra funcionalidad que es la de añadir un *pide taxi*. Esto supondría crear también una aplicación paralelo para los usuarios del taxi, y que pudiesen pedir mediante sus dispositivos un taxi. Consulta sección 2.2.4.

2.2.1 Soporte Metálico

El primer prototipo de soporte para el dispositivo fue este hecho en metal, ver figuras 2.10 y 2.11.

Sus características que se pueden destacar como positivas son:

- la robustez del material, por lo que es practicamente imposible romperlo.
- Una protección casi completa del dispositivo. Salvo lo que se quiere visualizar.
- Bastante *customizable*, se puede perforar para pasar cables, para la cámara, etc...
- Tiene una cerradura de llave circular lo que la hace muy segura y permite llevarse el dispositivo fácilmente para guardarlo cuando acaba la jornada.
- tiene un brazo que lo permite girar 180º arriba y a bajo y unos 90º aproximadamente de izquierda a derecha.

Figura 2.10: Soporte Metálico frontal.

También mencionar que se ha de pedir al extranjero por lo que para bien o para mal esto supone que no estará disponible con celeridad, debemos de esperar al menos 1 semana.

Pero por otro lado tenemos algunos inconvenientes o aspectos negativos del soporte:

- El soporte no está diseñado exclusivamente para nuestro dispositivo. Se necesitan almohadillas interiores para fijarla sobre el marco.
- Aún siendo de aluminio, su peso es excesivo y su precio superior a 100.
- Es bastante grande lo que lo hace demasiado invasivo para el reducido espacio del taxi.

Figura 2.11: Soporte Metálico trasera.

2.2.2 Portal Cautivo

Un reto ambicioso fue la de utilizar una *tablet* con **SO Android** como `Portal Cautivo` lo que el dispositivo daría servicio de `HotSpot`.

Primero deberemos de explicar que es un *HotSpot*: En el contexto de las comunicaciones inalámbricas, es un lugar que ofrece acceso a Internet a través de una red inalámbrica y un enrutador conectado a un proveedor de servicios de Internet.

Usualmente, los "hotspots"son zonas de alta demanda de tráfico, y que por tanto el dimensionamiento de su cobertura está condicionado a cubrir esta demanda por parte de un *punto de acceso*[1] o varios. Este servicio se suele cubrir con `Wi-Fi`.

Dicho esto, pasaremos a explicar que es un `Portal Cautivo` y porque es tan importante si queremos dar servicio a Internet a los usuarios del taxi mediante el *Wi-Fi*.

Un *Portal Cautivo* (o captivo) es un programa o máquina de una red informática que vigila el tráfico `HTTP` y fuerza a los usuarios a pasar por una página especial si quieren navegar por Internet de forma normal.

El programa intercepta todo el tráfico *HTTP* hasta que el usuario se autentifica. El portal se encargará de hacer que esta sesión caduque al cabo de un tiempo. También puede empezar a controlar el ancho de banda usado por cada cliente (haciendo lo que se llama Calidad de Servicio) o limitar su uso durante un intervalo de tiempo.

Es decir, podemos controlar el acceso que realizará el usuario a Internet para que, por ejemplo, no consuma toda la tarifa de datos del dispositivo, filtrar las páginas, obtener información de datos y gustos del usuario, o implementar funcionalidad extra como podría ser incrustar publicidad en el `HTML` de su navegador. Más sobre esto en la sección 2.2.3.

Y ahora viene el inconveniente. No se conoce un *Portal Cautivo* para dispositivos móviles por lo que deberíamos fabricar el nuestro. Un par de ideas de como construir un sistema así en el capítulo 4 en la sección **??**.

[1] **Un punto de acceso**, en una red de computadoras, es un dispositivo de red que interconecta equipos de comunicación alámbrica para formar una red inalámbrica que interconecta dispositivos móviles o con tarjetas de red inalámbricas.

2.2.3 Publicidad en wifi

En la sección 2.2.2 hablamos de implementar un *Portal Cautivo* en nuestro dispositivo *Android*. Esto abre la posibilidad a implementar una funcionalidad nueva que generaría ingresos adicionales a la empresa.

Hablamos de colocar anuncios en el navegador del usuario periódicamente mientras navega, un comportamiento que realizan muchos *HotSpot* comerciales para financiar sus despliegues. Podemos hablar de unas cuantas ideas acerca de como abordarlo suponiendo que disponemos de un *Portal Cautivo* en nuestro dispositivo en el capítulo 4, la sección **??**.

2.2.4 Pide Taxi

Otra idea plausible era la de incorporar un pide taxi al dispositivo pero eso provocaría la necesidad de desarrollar otra, la del usuario del taxi para poder solicitarlo.

Además otro inconveniente que le vimos a esta idea fue que el mercado ya estaba copado. Ya existen múltiples aplicaciones que realizan esto. Algunas de ellas están facturando miles de millones en todo el mundo como `Uber`, para más información visitar su página *Web* en **Uber**

Aunque *Uber* no es un pide taxi al uso, se utiliza para particulares existen algunas otras que si lo son, algunos ejemplos: `MyTaxi`, `HAILO`, `By-Taxi`, `Taxible` o `TaxiClick`.

2.3 Funcionalidad

Quizá esta sea la parte más interesante de la publicación. Después de mucho deliberar, debíamos decidir el `producto viable mínimo` [2], y quizá este objetivo no lo conseguimos, pues siendo ambiciosos desarrollamos muchas más funcionalidades de las necesarias para ser lanzado.

Pero por otro lado le da un punto atractivo al usuario para que quiera utilizarlo, pues si solo lanzamos un producto que reproduce publicidad pronto el usuario perderá el interés, así que le agregamos todas estas fantásticas cosas.

Quizá una herramienta por defecto en nuestros dispositivos sea el **Navegador**, ver sección 2.3.1, con el que el usuario podrá visitar sus portales, *blogs* y buscadores favoritos.

También tenemos un básico buscador de trayectorias que con la ayuda de un **Mapa** podremos calcular el tiempo y distancia que tiene nuestro viaje, ver en sección 2.3.2.

Una herramienta útil para los turistas será la **Guía**, con la que podrá visualizar fotos y escuchar las descripciones de monumentos y demás lugares de interés, detallaremos más su funcionalidad en la sección 2.3.3.

Otra funcionalidad muy útil para turistas y relacionada con la anterior son las **Rutas**, con la que al activar una, el turista podrá disfrutar de las fotos y descripciones de los lugares de interés dentro de un circuito cerrado, un servicio que será ofrecido junto con los taxistas, más detalles en la sección 2.3.4.

Quizá es la funcionalidad que más impresiona, la del **Traductor**, mediante la cual podremos hablar con el taxista puesto que todos los idiomas predefinidos en la aplicación, ver sección 2.3.6, nuestras palabras serán traducidas al castellano, todo el detalle de este proceso en la sección 2.3.5.

Otra funcionalidad indispensable para el turista sea la elección de **Idioma** predefinidos, con el cuál toda interfaz, texto y audio de la aplicación serán traducidos a la lengua que se haya elegido. Se detalla en la sección 2.3.6.

Una funcionalidad que nos conquistará es la **Detección de rostros**, mediante la cámara y el procesado de imágenes. Detallado en la sección 2.3.7.

La última, y no menos importante, es la funcionalidad de la **Publicidad**, que será la base de nuestros ingresos. Este proceso se detalla en la sección 2.3.8.

[2] **producto viable mínimo** o **MVP** (del inglés *Minimum Viable Product*) es la versión de un nuevo producto que permite a un equipo recolectar, con el menor esfuerzo posible, la máxima cantidad de conocimiento validado sobre sus potenciales clientes.

2.3.1 Navegador Web

Cuando pensábamos en la aplicación, rápidamente nos dimos cuenta que no podría faltar un navegador, la aplicación por excelencia dentro de cualquier dispositivo conectado a internet.

Básicamente tenemos el navegador incrustado en el *layout* derecho y arriba un panel de control, véase la figura 2.12.

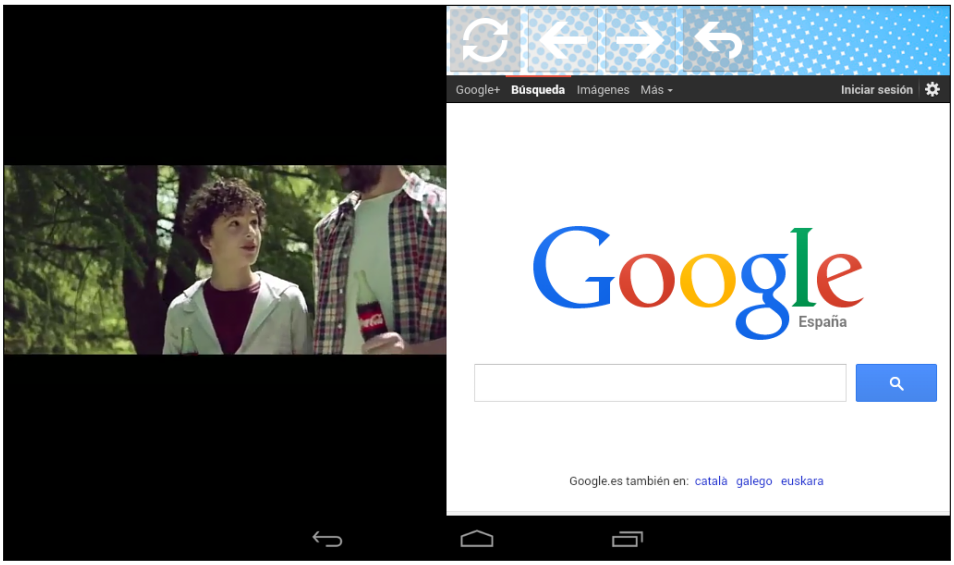

Figura 2.12: Navegador Web Adverspot.

En el tenemos de izquierda a derecha:

- el **botón de actualizar**, que tiene una animación de rotación hasta que la página se ha cargado.
- el **botón atrás** y **botón adelante** para la navegación del historial entre página anterior y página posterior. Si no existe una página anterior o posterior visitada anteriormente estos botones estarán desactivados.
- y el **botón de volver** al menú principal.

2.3.2 Mapa

Al pulsar el botón de mapa nos aparece esta ventana, figura 2.13, donde se desplegará un mapa haciendo zoom sobre tu ubicación actual.

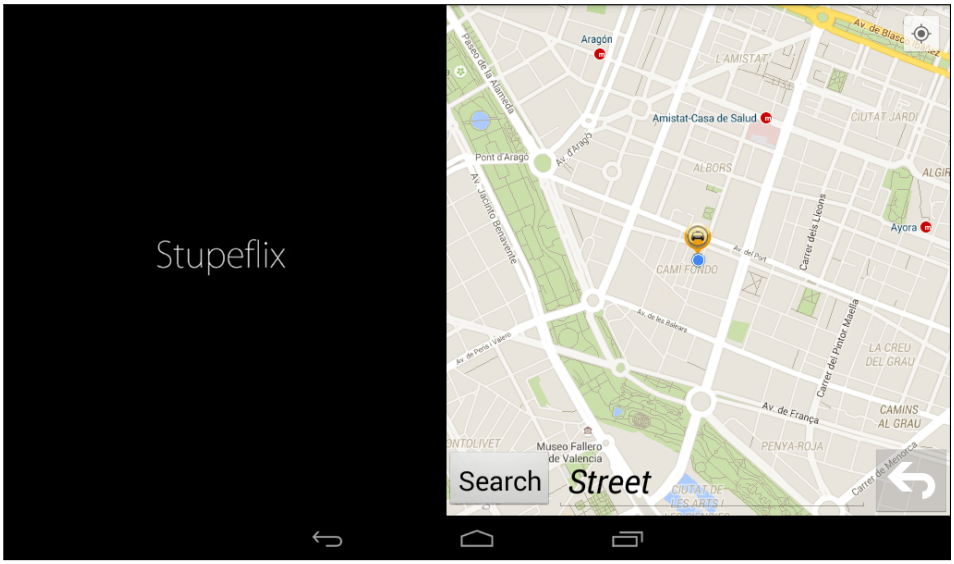

Figura 2.13: Mapa Adverspot.

También dispone de un panel inferior que se utiliza para calcular nuestra ruta.

Mediante el `Mapa de Google` podemos navegar por él pulsando y arrastrando el dedo. Tiene dos funcionalidades fundamentales: con un dedo podremos movernos por el mapa y con dos dedos hacia adentro y hacia afuera podremos alejar y acercar el zoom.

Si nos hemos perdido por el mundo y queremos volver a nuestra ubicación actual tan solo debemos pulsar el pequeño botón del margen derecho superior.

Lo primero que debemos hacer es pulsar sobre el mapa donde queramos viajar. Se lanza un proceso el cual busca el punto pulsado sobre el mapa y nos escribe la localización sobre el campo de texto, ver figura 2.14.

La localización es bastante precisa, la cadena de texto devuelve el nombre de la calle, el número, la provincia y el país.

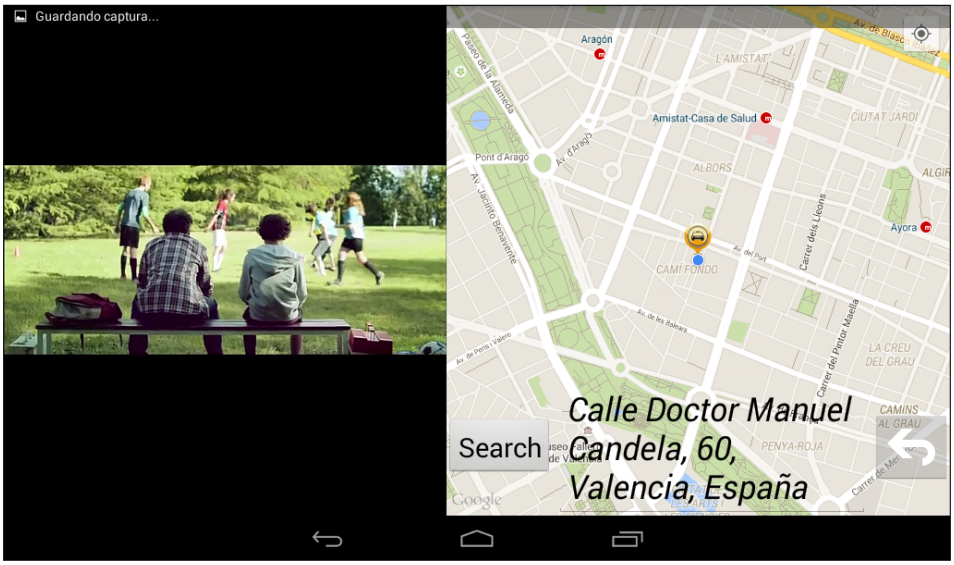

Figura 2.14: Calle Mapa Adverspot.

Una vez que se ha escrito el nombre de la calle podemos pulsar sobre el **botón buscar**, el cual provoca que se trace la ruta con una linea roja entre la posición actual y la que hemos decidido.

Se lanza un proceso que calcula todos los puntos mediante la librería de `Google Maps`. Construye una lista de puntos *latitud* y *longitud* y dibuja rectas entre los puntos construyendo un camino por las calles entre la ubicación actual y el destino deseado.

Se podrá visualizar tanto tiempo como distancia, ver figura 2.15. Comprensiblemente, el tiempo es estimado y depende de la densidad de tráfico y de los semáforos.

.

Si queremos volver al menú principal, figura 2.7, debemos pulsar el **botón volver**.

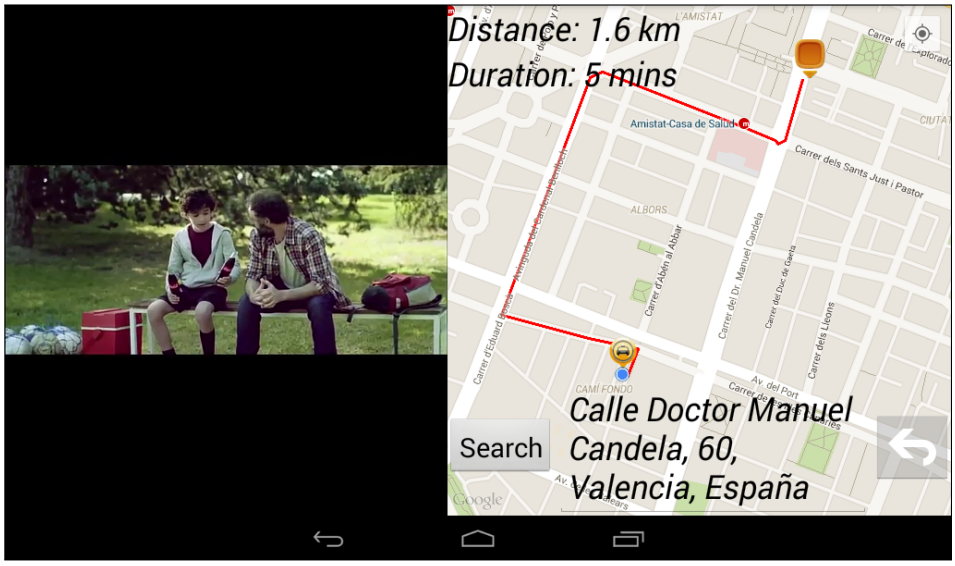

Figura 2.15: Trazo Mapa Adverspot.

2.3.3 Guía

En esta sección detallamos la funcionalidad de la guía. Tenemos que recordar que el texto de la interfaz se cambia al seleccionar el idioma. Para más información diríjase a la sección 2.3.6.

Figura 2.16: Principal Guía Adverspot.

.

Mediante la librería de `Google Speech` podremos escuchar el texto de las interfaces de forma hablada. El idioma seleccionado en la aplicación determinará la voz utilizada para ello. Para más información acerca de esta librería diríjase al enlace **googlespeech**

También disponemos de un botón para parar la reproducción con el icono de altavoz y otro para volver al menú principal, figura 2.7, o podemos deslizar nuestro dedo de izquierda a derecha sobre el panel también para hacerlo.

En el menú principal de la guía disponemos de 5 categorías principales que dependen de la temática. Estos son:

- **Eventos**: en esta categoría disponemos de un número de acontecimientos particulares como son conciertos, ferias, eventos comerciales, ...
- **Monumentos**: disponemos de construcciones singulares y bellos en la ciudad, aquí podremos encontrarlos.

- **Lugares**: ubicaciones de ocio como son grandes superficies comerciales o parques.
- **Museos**: galerías de arte y edificios de interés cultural.
- **Actividades**: la oferta de actividades que tenemos en la ciudad como son deportes de riesgo, paseos en barca, bici, contratación de servicios de ocio, cines, ...

Para acceder a cualquiera de las categorías debemos pulsar sobre una de ellas.

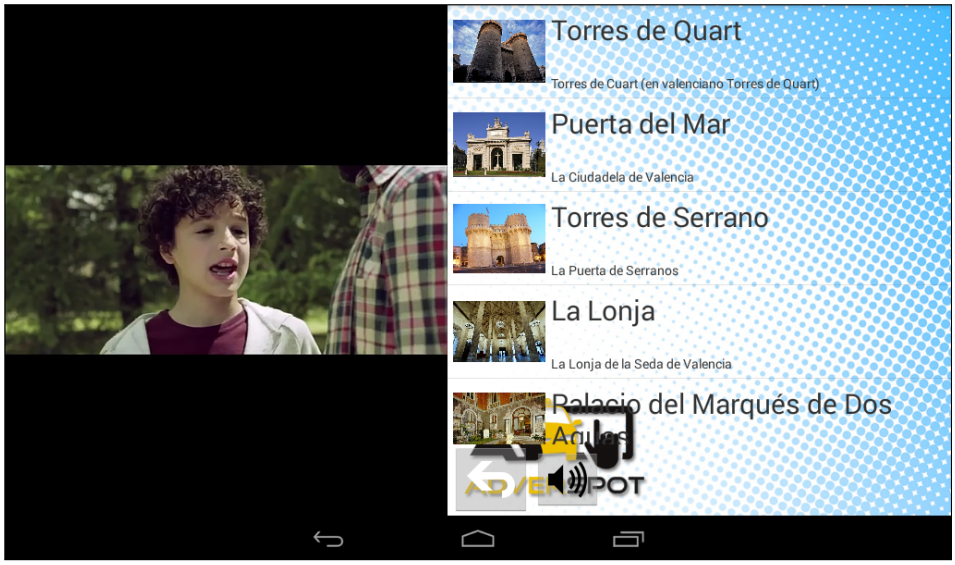

Figura 2.17: Categoría Guía Adverspot.

.

En esta nueva pantalla tenemos una lista de la categoría seleccionada, ver figura 2.17, en este caso de monumentos de la ciudad de Valencia, con una imagen de presentación, el nombre del monumento y su descripción breve.

Todo ello volverá a ser hablado por el `Google Text-To-Speech` nuevamente. Recordar la funcionalidad de los botones de parada y volver a reproducir el audio del texto del menú y el botón de atrás.

Además de el deslizamiento de dedo de izquierda a derecha para volver atrás y de arriba a abajo y viceversa para recorrer la lista de monumentos.

Si queremos ver una descripción más detalla debemos pulsar sobre él.

Figura 2.18: Punto de Interés Guía Adverspot.

.

En esta nueva pantalla tenemos la descripción extensa del elemento que hayamos seleccionado.

Arriba aparece su nombre y justamente debajo tenemos una lista con una galería de fotos del elemento. Si pulsamos sobre alguna de ellas, se redimensionará la foto para poder visualizarla mejor. Ver figura 2.19. Si queremos cerrar la ventana tan solo debemos pulsar sobre la foto nuevamente.

.

También podemos volver atrás deslizando el dedo de izquierda a derecha. Si la descripción es demasiado larga y no cabe en la pantalla se podrá deslizar su texto de arriba a abajo para poder navegar por él.

Recordamos que siguen existiendo las funcionalidad del `Text-To-Speech`. Los botones de atrás y audio, y uno nuevo, el botón de *Maps* para saber donde se ubica el elemento en nuestra ciudad.

Si hemos pulsado sobre el botón de mapa navegaremos hacia el `Google Maps` de nuevo, pero esta vez no podremos elegir destino. Tan solo aparecerá la ruta desde nuestra ubicación actual hasta la del elemento seleccionado, nue-

Figura 2.19: Foto Galería Guía Adverspot.

vamente con la información de distancia y tiempo estimado por si queremos visitarlo, ver figura 2.20.

Seguimos teniendo la funcionalidad básica del mapa de navegar por él, zoom y volver la vista a nuestra ubicación actual.

.

Si queremos volver a la pantalla de descripción tendremos que pulsar sobre el botón atrás.

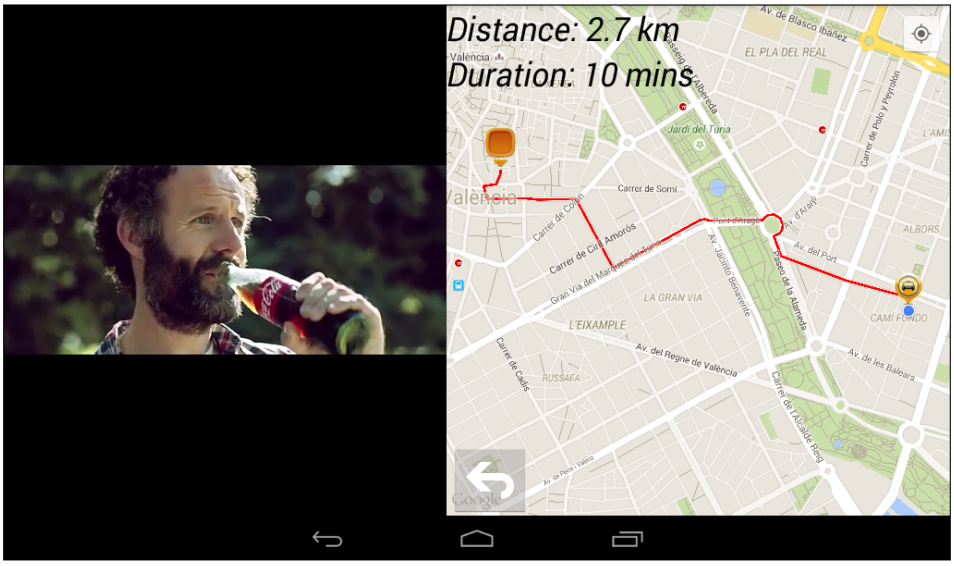

Figura 2.20: Mapa Guía Adverspot.

2.3.4 Ruta

Esta funcionalidad es parecida a la guía pero la cooperativa de taxi nos la pidió expresamente. En ella podremos elegir una lista de rutas que están configuradas para cada *tablet*.

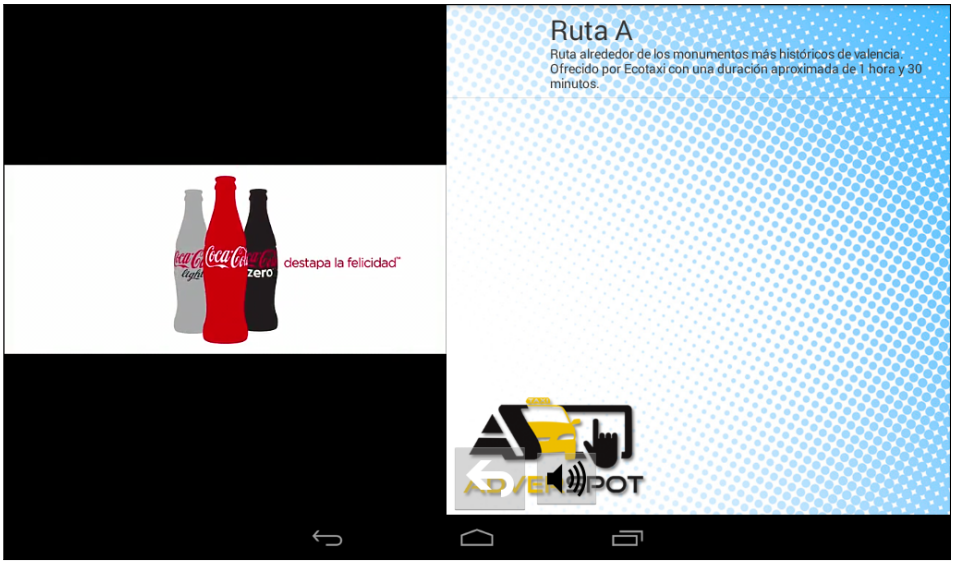

Figura 2.21: Ruta Adverspot.

.

Si nos fijamos en la figura 2.21, cada ruta tienen un nombre y una breve descripción.

Nuevamente tenemos el `Google Text-To-Speech` que nos hablará acerca de las rutas disponibles. Para parar o reanudar su reproducción podemos utilizar el botón del icono de altavoz.

Para volver atrás las dos opciones estándar. El botón atrás o deslizando el dedo por la pantalla de izquierda a derecha.

Para seleccionar una ruta pulsamos sobre ella. Al pulsarla nos aparecerá una lista de *items*. Lugares que se irán visitando a lo largo del recorrido con su foto de presentación, su nombre y su texto. Ver figura 2.22.

.

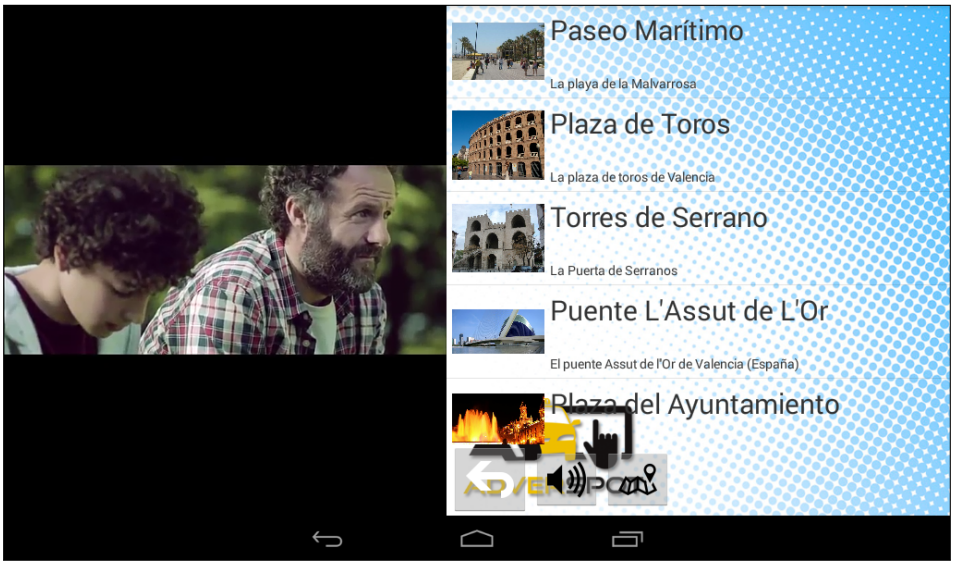

Figura 2.22: Lista Ruta Adverspot.

Como explicamos en la sección 2.3.3, tenemos las mismas funcionalidades como texto hablado, volver atrás y el botón de parar o reanudar audio. Salvo una singularidad, si presionamos sobre alguno no ocurrirá nada.

Para hacerla funcionar debemos presionar el nuevo icono. Entonces aparecerá una barra de carga. Ver figura 2.23. Esto lanzará un proceso comparando la ubicación actual con la de los puntos de interés para obtener el punto de interés más cercano.

Este proceso es sencillo. Básicamente se suman las distancias entre la resta de sus *latitudes* y *longitudes* de los puntos.

La razón de esto es que el taxi realizará la ruta desde el punto de interés más cercano, y si el cliente lo desea la completará. Es entonces, un servicio que podrá ofrecer el taxi al usuario.

.

Cuando acabe el proceso, se navegará hacia un nuevo panel idéntico al de la guía, ver figura 2.18. Pero el proceso no se detiene. Duerme durante unos pocos minutos y después vuelve a lanzarse para encontrar el punto de interés más cercano.

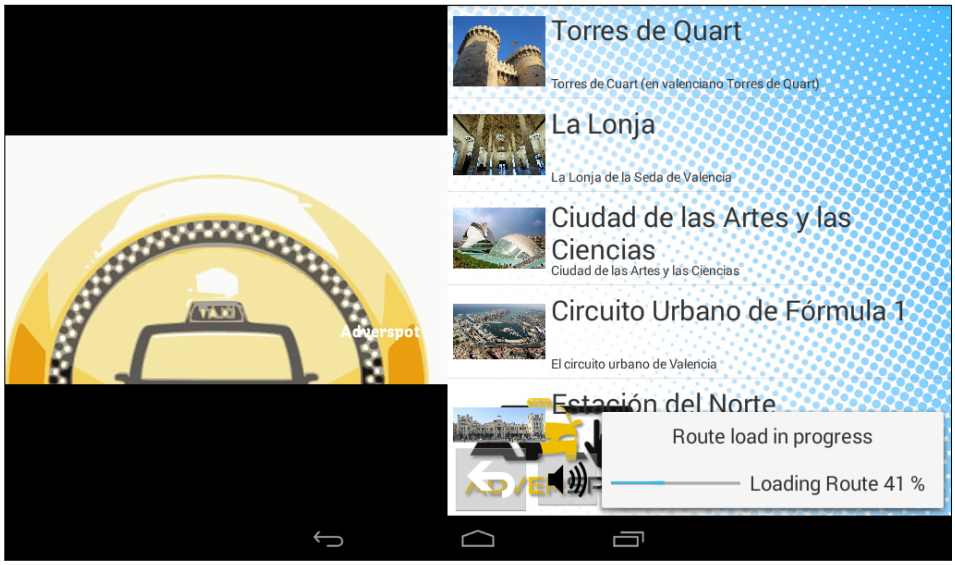

Figura 2.23: Carga Ruta Adverspot.

Posteriores veces a la primera vez, tenemos un pequeño margen de error para evitar posibles transiciones entre puntos de interés equidistantes. Esto provoca que el usuario no tenga que hacer nada para que la pantalla se refresque con el nuevo punto de interés.

Y por supuesto, tenemos la opción de visualizar la ubicación del punto de interés desde el mapa. Ver figura 2.20. La única diferencia esta vez es que no podremos elegir el destino. Tan solo aparecerá la información de la ubicación actual hasta el punto de interés, su distancia y su tiempo estimado hasta llegar a él. No podremos modificarlo. Tan solo volver a la pantalla de información.

Si deseamos parar la ruta debemos navegar hacia atrás dos paneles. Esto nos permite repasar la lista de puntos de interés si lo deseamos, figura 2.22.

Lo que ocurre si deseamos volver a la lista de rutas entonces saltará una ventana de confirmación. Ver figura 2.24. Esto detendrá definitivamente el proceso de búsqueda de puntos de interés para esa ruta. Y nos llevará nuevamente a la vista de lista de rutas. Figura 2.21.

.

Figura 2.24: Parar Ruta Adverspot.

2.3.5 Traductor

Quizá sea la diferencia de idioma la mayor barrera que un turista puede tener en un país extranjero. Por eso, la funcionalidad más potente la comunicación entre taxista y usuario extranjero sea el traductor.

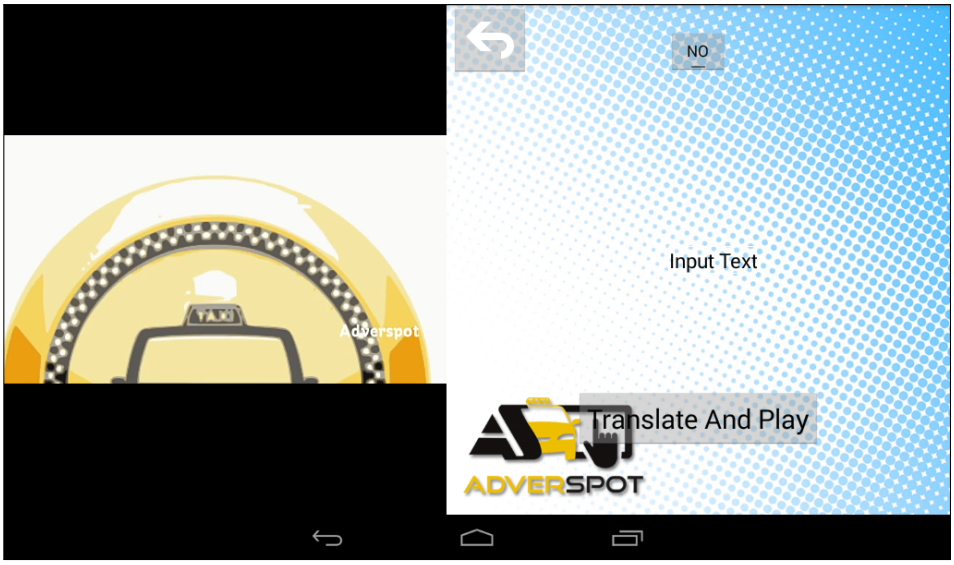

Figura 2.25: Translator Adverspot.

.

En el panel principal del traductor podemos observar que tenemos un botón en la parte superior central para poder procesar nuestra voz y convertirla a texto mediante nuevamente las librerías de `Google Speech`. Ver figura 2.25.

Si lo pulsamos, se lanzará un proceso de escucha para que nosotros podamos hablar y grabar la frase que deseamos decirle al taxista. Este texto se escribe sobre un campo de texto para que si fuese necesario podamos corregirlo a mano.

Después debemos pulsar el botón **Translate And Play** el cual lanza otro proceso para la traducción del texto. La aplicación se comunica con un *Servicio Web* de traducción de textos.

Cuando nos devuelve la cadena traducida, se lanza otro proceso que usa una vez más `Google Speech` para que nos hable la frase en castellano. Antes de

eso, bajamos el volumen de la publicidad para que el taxista pueda escucharlo adecuadamente.

Se piensa en desarrollar una aplicación cliente-servidor para que la comunicación se pueda realizar entre dispositivos `Android`. El taxista se podría comunicar bidireccionalmente con el turista extranjero. Hablaremos de ello en el capítulo 4, sección **??**.

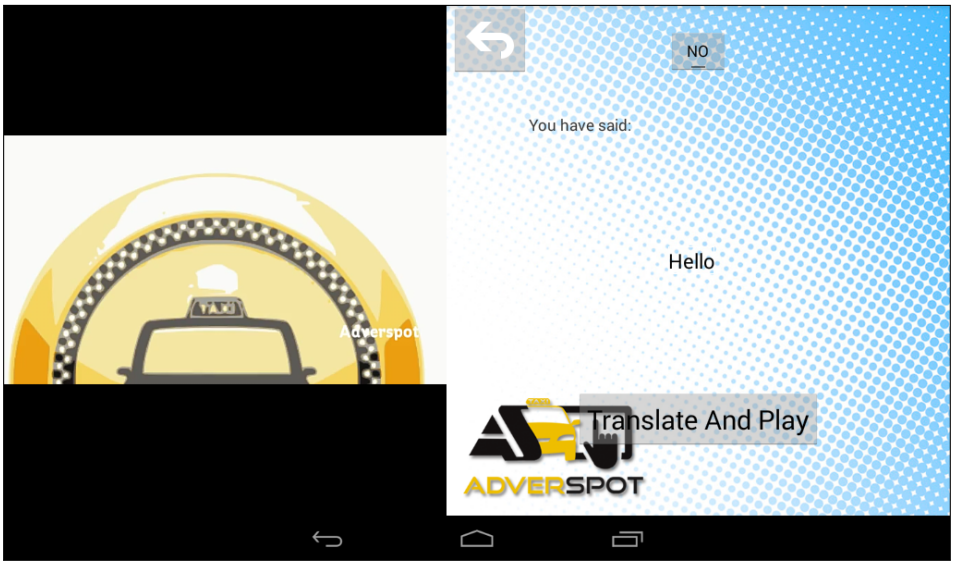

Figura 2.26: Translator Adverspot.

.

Para volver al menú principal, figura 2.7, podemos pulsar el botón atrás del margen superior izquierda.

2.3.6 Elección de idioma

Esta funcionalidad básicamente elige el idioma en la que queremos la interfaz de la aplicación `Adverspot`. Toda la configuración de `Google Speech` se basará en el idioma que hayamos elegido, por lo que es importante elegir el que deseemos antes de realizar cualquier otra acción.

También la publicidad se subtitulará con el idioma seleccionado. Más sobre esto en la sección 2.3.8.

Para poder elegirlo simplemente debemos pulsar sobre la lista de idiomas de la parte inferior del panel principal la bandera del país con el idioma que prefiramos. Ver figura 2.7. Podremos deslizar la lista de banderas lateralmente para poder los idiomas predefinidos.

2.3.7 Detección de caras mediante cámara fotográfica

Una de las cosas que pensamos a la hora de diseñar la aplicación es que debíamos controlar cuando el usuario del taxi entraba o salía del taxi. Tras deliberaciones decidimos que la mejor manera, aunque no la más fácil, sería detectar a una persona detrás del habitáculo el vehículo, por lo que la aplicación permanecería sin reproducir publicidad hasta que este evento sucediese.

Para ello, utilizamos la cámara que realiza un *Preview* o foto en un `buffer de bytes`. Simplemente utilizando un algoritmo de detección de rostros proporcionado por las librerías `Android` sobre esa foto nos indica el número de personas que aparecen en ella. Si el número es mayor que 0. Entonces se activa la reproducción de la publicidad.

Esto ocurre cada pocos segundos por lo que si el anuncio acaba y no se ha vuelto a detectar tanto el rostro de la persona como la interacción con la pantalla del dispositivo, entonces la *tablet* volverá a parar la reproducción de la publicidad. Esto hace que la contabilización de las reproducciones sea veraz, garantía a los clientes que contraten parrillas de publicidad a tenido impactos.

2.3.8 Publicidad

Esta última funcionalidad es la única que no está pensada para el usuario pero no por ello menos importante.

Su utilidad es la de financiar el producto. Su funcionamiento es sencillo. El dispositivo descargará unas listas de reproducción e irá lanzando vídeos hasta cumplir con su parrilla.

Todo vídeo contiene el audio subtitulado para que el idioma no sea una limitación para entender la publicidad. Observe la figura **??**.

Las listas, que son configuradas desde el servidor, tienen un rango de fechas y de horas. Esto permite, por ejemplo, configurar distintas listas dependiendo del horario al que se quiera exponer la publicidad. La ley impide publicitar cierto tipo de publicidad durante el horario diurno o considerado como "*infantil*"por lo que es necesario controlar este tipo de publicidad que por otro lado, puede reportar grandes beneficios.

Cabe destacar que cuando el usuario está interactuando con el dispositivo, como hemos podido observar, la publicidad tan solo ocupará la mitad de la pantalla por lo que la superficie del vídeo también disminuye. Cuando no se interactúa con ella su superficie es aproximadamente 2/3 del total.

2.4 Arquitectura

En esta sección se detallan los detalles de la arquitectura que conforman el sistema.

Primero se habla de la arquitectura del sistema en aspectos generales y la diferencia de roles que existen entre las máquinas. Sección 2.4.1.

Después se detalla la navegación del cliente por la interfaz gráfica entre las distintas funcionalidades y transiciones de pantalla. Sección 2.4.2.

El siguiente punto es hablar sobre las especificaciones del servidor que se configura para atender la petición de multitud de clientes. Sección 2.4.3.

Acto seguido se explica el modelo de datos que se ha desarrollado para el sistema. Cliente-Servidor comparten este modelo salvo que el cliente no tiene conocimiento del resto de clientes. Sección 2.4.4.

Por último explicaremos los procesos de sincronización que realiza el cliente mediante las peticiones al servidor. Sección 2.4.5.

2.4.1 Arquitectura Sistema

En los inicios de la informática, la programación se consideraba un arte y se desarrollaba como tal, debido a la dificultad que entrañaba para la mayoría de las personas, pero con el tiempo se han ido descubriendo y desarrollando formas y guías generales, con base a las cuales se puedan resolver los problemas. A estas, se les ha denominado `Arquitectura de Software`, porque, a semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software.

Una de las estrategias que utiliza la *Arquitectura del Software* son los llamados `patrones de diseño` que son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Por tanto, nuestra arquitectura utilizada es la llamada `Arquitectura en tres capas` que es una especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente, y

en nuestro caso, la capa de persistencia se divide entre el cliente y el servidor. Ver figura 2.27.

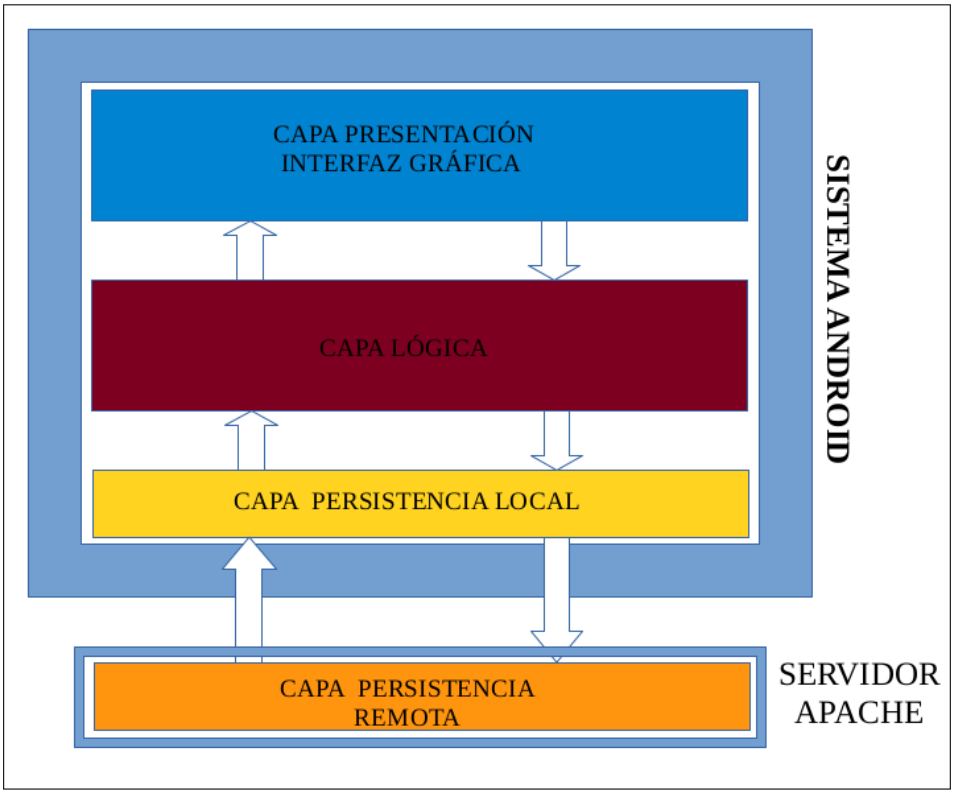

Figura 2.27: Arquitectura Sistema Adverspot.

.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo se ataca al nivel requerido sin tener que revisar entre código mezclado. Es decir, muy probablemente la capa de presentación no necesite de la de persistencia local y en raras ocasiones la de persistencia local lo hará frente a la de persistencia remota.

De hecho, tenemos 2 procesos por debajo en los que se intercambian datos entre la capa de persistencia local-remota. Para más información ver sección 2.4.5

2.4.2 Navegación Cliente

Aquí se explica la navegación entre las funcionalidades del producto que se explican en la sección 2.3 del capítulo 2. Este sería el diagrama de navegación, ver figura 2.28.

.

En él se distinguen los nodos y las transiciones que son bidireccionales, de arriba a abajo, desde el **Menú Principal** hacia las funcionalidades principales: **Traductor** sección 2.3.5, **Ruta** sección 2.3.4, **Mapa** sección 2.3.2, **Guía** sección 2.3.3, **Idioma** sección 2.3.6 y **Navegador Web** sección 2.3.1, todos ellos en el capítulo 2.

Se ha de mencionar que todo estado, incluso el **Menú Principal** tiene un *timeout* predefinido en el que si el conductor no interactúa con la pantalla del dispositivo o no se detecta su rostro, ver sección 2.3.7 para más información, entonces aparecerá un fondo de pantalla de la marca de `Adverspot` o algún *logo* de la cooperativa de taxis y no se reproducirá el siguiente anuncio hasta que ocurran tales eventos.

Cuando esto ocurre volveremos a la pantalla principal, figura 2.7 y el volveremos al estado inicial.

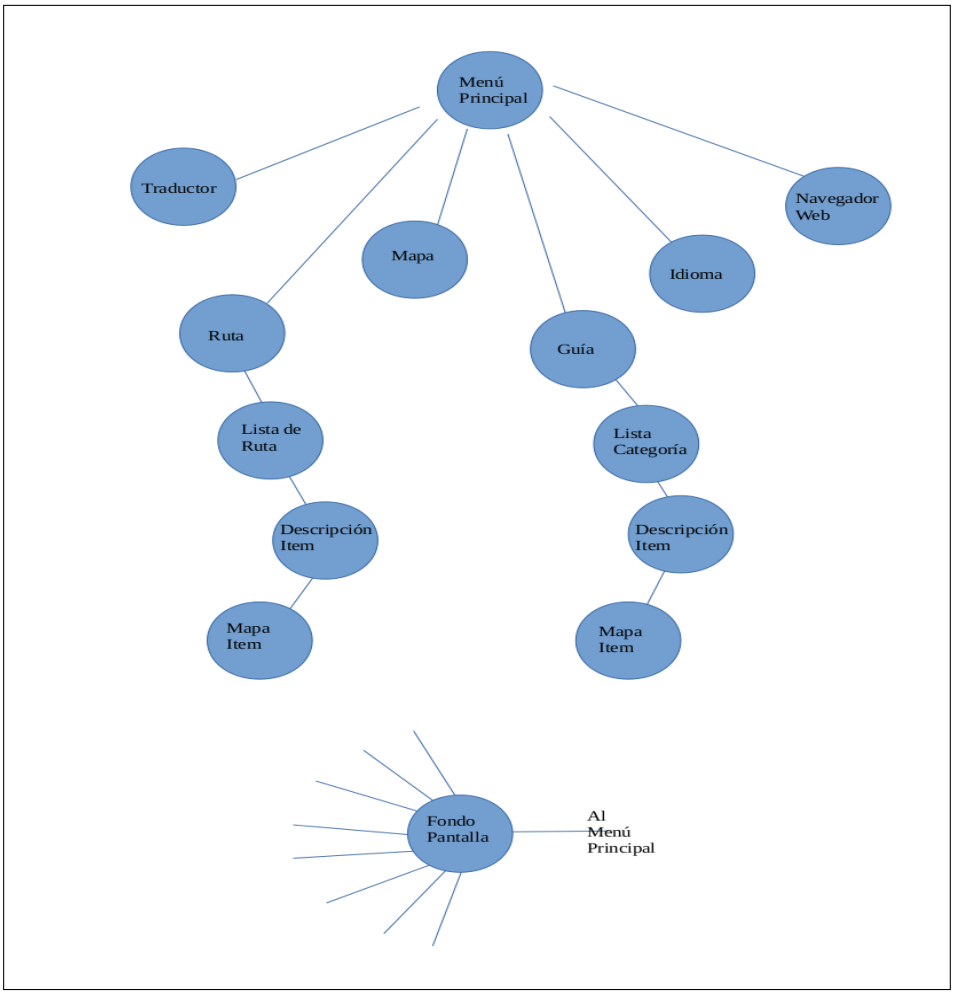

Figura 2.28: Navegación Adverspot.

2.4.3 Servidor

Ya dijimos en sección 2.4.1, nuestro sistema está construido mediante una arquitectura cliente-servidor. La parte del servidor por tanto está separada de la máquina que realiza el rol de cliente pero que es accesible desde internet.

Este requisito hizo necesario la adquisición de una máquina. Existen infinidad de empresas que contratan estos servicios. Se ha optado por uno ofrecido por la compañía **Kimsufi**, por su calidad/precio. El enlace en **Kimsufi** El modelo según las prestaciones necesarias para el proyecto `Adverspot` fue el **KS2** y tiene las siguientes características:

- **Procesador**: Atom^{TM}N2800
- **Núcleos/Hilos**: 2c / 4t
- **Frec. Procesador**: 1,8Ghz+
- **RAM**: 4GB
- **Disco**: 1TB
- **Red**: 100Mbps
- **IPv6**: /128
- **Precio**: 9,99 + IVA

Figura 2.29: Servidor Kimsufi.

.

Se estima que para 50 clientes el servidor debería soportar bien la carga de trabajo. Para una carga mayor se deberá contratar un servicio más potente. Hemos calculado que con buena conexión a internet la sincronización del cliente-servidor se realiza entre 2-5 minutos dependiendo del contenido descargado: vídeos, imágenes, audio, subtítulos, texto, ... calculando unos pocos *GBs* para un solo cliente.

2.4.4 Modelo de Datos

Ya que nuestra capa de presentación está dividada en dos subcapas cliente-servidor, debemos utilizar una interfaz común para los dos, puesto que la **Base de Datos** local es *relacional* y la del servidor es *objeto-relacional*, sin sentencias `SQL` como si estuviésemos manipulando la capa lógica, es decir, objetos aunque se aprovecha la arquitectura relacional de las bases de datos ya implementadas `Oracle`, `MySQL`, `PostgreSQL`, ...

Para el cliente utilizamos las librerías nativas de *Android* que utilizan `SQLite` mientras que el servidor está desarrollado en `Sailsjs` que está basado en `Nodejs`, un *framework* para el desarrollo de servicios *Web* y de propósito general. Este entorno de desarrollo para servidores utiliza `Waterline` que es un **ORM** que permite "*mapear*"las estructuras relacionales en objetos de la capa lógica por lo que no se manipula directamente la capa de persistencia, se crea un envoltorio genérico para manipularla. Más información de todo esto en el anexo C.

Después de esta introducción compleja de las tecnologías en las que se basa nuestro modelo de datos de la base de datos del servidor es que aparece en la figura 2.30.

.

Como se puede apreciar, la entidad principal es la *tablet* que contendrá un vehículo, que a su vez puede tener varios conductores a su servicio y una serie de listas.

Estas listas contendrán varios anuncios, estos anuncios podrán tener uno o varios vídeos a reproducir, como una campaña publicitaria y este a su vez tendrá varios subtítulos, uno por cada idioma. Detallado en la sección de publicidad 2.3.8 para vídeos y en la sección idiomas 2.3.6 para subtítulos.

Estos anuncios estarán sujetos a un contrato que se hará con el cliente en el que se especificará ciertas características como ingreso, dinero, documento adjunto, ... y este a su vez la forma de pago con sus datos bancarios, domiciliación de los recibos o pago electrónico.

Un contrato esta relacionado con una o varias empresas y esta a su vez tiene varias personas de contacto.

Por otra lado, tenemos una lista de *PINs*, puntos de interés, para la guía con sus imágenes y sus descripciones escrita en los idiomas disponibles. La ruta también utiliza una serie de estos *PINs* que también contiene sus descripciones en distintos idiomas. Más información de la **Guía** en la sección 2.3.3 y de las **Rutas** en la sección 2.3.4.

Por último tenemos un *historial* con el número de reproducciones de cada anuncio que ha reproducido una *tablet* en una fecha concreta.

Todas las entidades que se reflejan de color azul, las pertenecientes a la publicidad y las de verde, las pertenecientes a las rutas y guía, son las percibe el dispositivo en su base de datos local. Del resto no tiene constancia. Las de color morado son las encargadas de la logística del producto. Las entidades de rojo pertenecen a la parte empresarial del producto.

Por el contrario, para el cliente cada vez que realiza una petición sincroniza su lista de reproducción, el servidor le manda sus listas, sus anuncios, sus vídeos y sus subtítulos. Y por otro lado los *PINs*, sus imágenes y descripciones, las rutas y sus descripciones. Este proceso de sincronización y el de historial se detallan en la sección **Proceso Sincronización** 2.4.5.

Por lo tanto la capa de persistencia local es muy similar a la remota solo, quizá un microcosmos de ella misma pues no contiene información del resto de dispositivos ni de sus listas, tampoco de contratos, ni actores.

Otra diferencia no mencionada antes es que al cliente se le ha implementado su propio *ORM* al no disponer de uno estádar para la plataforma `Android`. Esto es la no necesidad de manipular código `SQL` directamente desde la capa de negocio, sino que se construyen objetos que luego serán consultados, manipulados y actualizados con la capa de persistencia.

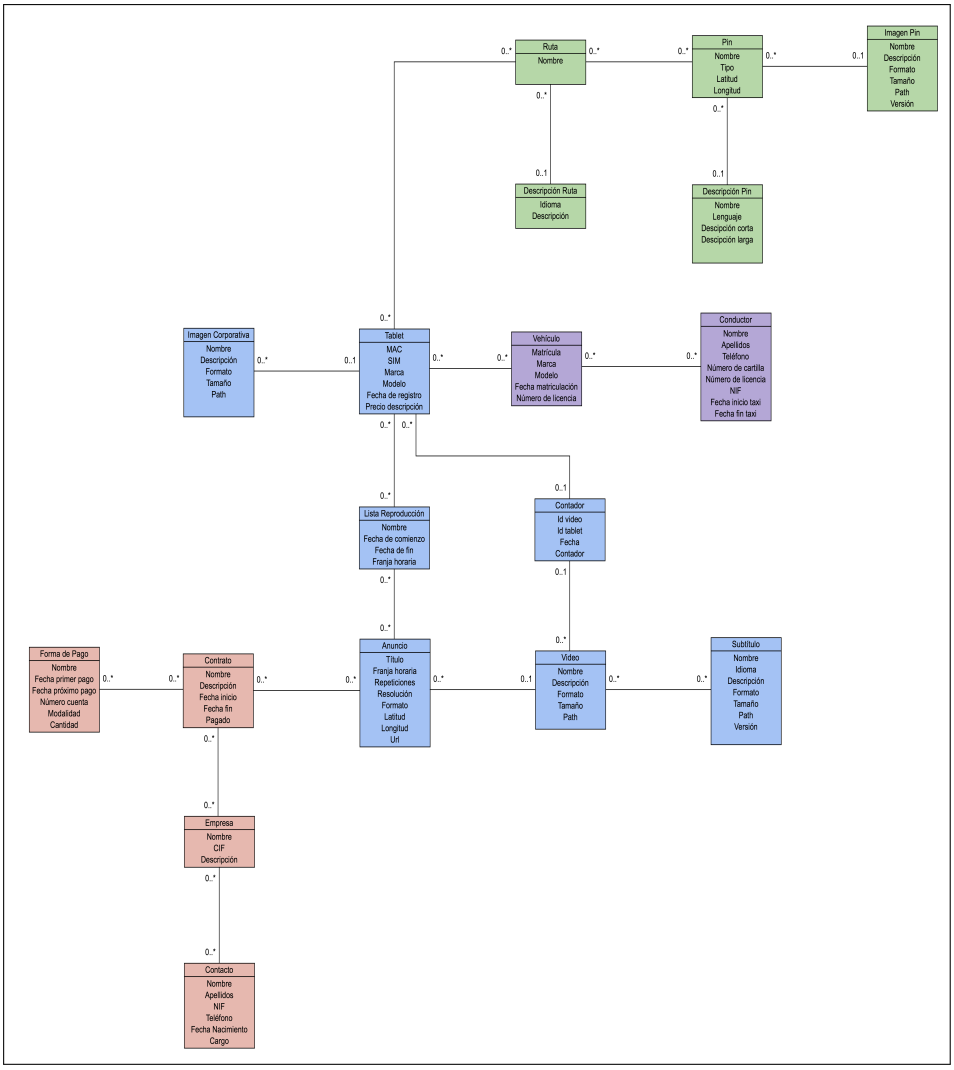

Figura 2.30: Modelo de Datos Servidor.

2.4.5 Proceso Sincronización

El proceso de sincronización se realiza mediante peticiones **JSON**. Básicamente es un formato en texto plano que utiliza una sintaxis estructurando el contenido con `clave-valor`.

Tenemos dos procesos de sincronización principales:

El primero es el más importante y se lanza desde el inicio de la aplicación para obtener todo el contenido que se va a reproducir. Durante este periodo de varios minutos la aplicación se mantiene bloqueada. En el se descargan las listas de vídeos, los puntos de interés con sus descripciones e imágenes y las rutas. Es un hilo que cuando acaba el proceso de sincronización se manda a dormir un tiempo definido y cuando despierta se vuelve a comparar con la información del servidor. Si no es necesario actualizar el contenido finaliza muy rápidamente.

En la figura se puede observar como la pantalla de carga inicial que bloquea la interfaz del dispositivo hasta que ha terminado. Ver figura 2.31.

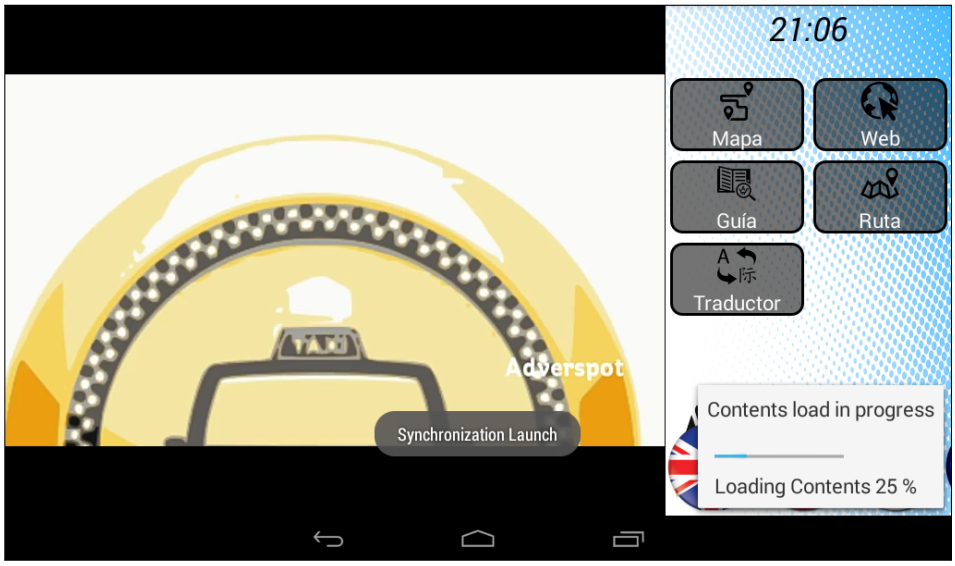

Figura 2.31: Sincronización de Contenidos.

.

El segundo es otro hilo que se encarga de mandar periódicamente el historial de reproducciones cada varios minutos pues es la contabilización de los impactos publicitarios el que determine nuestros ingresos.

Por lo tanto, estos dos procesos de intercambio de información dentro de la capa de persistencia cliente-servidor son críticos para el sistema.

Capítulo 3

Plan de Negocio

En este capítulo analizaremos lo que exige montar una empresa a lo hora de embarcarse en la aventura de emprender. Regular en el marco de la legislación vigente la idea del producto y materializarla en un negocio rentable que al fin y al cabo es el objetivo de toda empresa, poder sobrevivir a su nacimiento y a los cambios y dificultades por las que atravesará.

En la sección 3.1 se explica la forma jurídica de nuestra empresa y una visión general del mundo del emprendimiento.

En la sección 3.2 se detalla cual es la fórmula utilizada para poder obtener beneficios de la operatividad de la empresa.

En la sección 3.3 se tabla y explica cuales han sido los gastos tanto inciales como los que se generarán con la actividad de la empresa.

3.1 Introducción

En nuestro país existen varias de formas jurídicas de una empresa. Existen **SL**, **SA**, **Autónomos**, ... No se detalla aquí cada una de ellas, pertenecen al ámbito jurídico, pero nosotros nos decantamos por la *SL* por diferentes motivos.

- Se quiere ver reflejado como entidad la actividad de diferentes personas como una marca o nombre.
- La inversión de capital inicial ha de ser mínima de 3.000 para ello.
- Si los ingresos superan una franja de facturación anual los beneficios fiscales son amplios.
- Es posible vender una porción de la empresa en acciones a terceras personas o empresas que quieran financiar el proyecto.

El registro mercantil de la empresa se hace además con el **logo** de la empresa. Figura 3.1.

Figura 3.1: Logo Graphene Code.

No es necesario formar una empresa para emprender pero si será necesario en un futuro si en algún momento queremos cobrar dinero regularizado y declarado a la hacienda pública por nuestro producto o servicio, como se dijo antes, es solo una forma de formalizar y transformar una idea de *emprendimiento* en un negocio rentable y su estrategia es lo que se conoce como *plan de empresa* o *plan de negocio*.

3.2 Ingresos

Prácticamente los ingresos van a ser procedentes de la publicidad de nuestros clientes. Para eso llevaremos a cabo un cálculo de facturación o de precios que le cobraríamos al cliente que se quiera publicitar.

El valor por segundo emitido seria de 2 céntimos esto quiere decir que para calcular el precio de un de anuncio deberíamos saber los siguientes factores:

1. **X**: Vehículos en los que se tiene que repetir.
2. **Y**: Cuantas repeticiones al día por vehículo.
3. **Z**: Cuantos días teniendo en cuenta que los taxis trabajan 22 días al mes.
4. **W**: Precio por segundo.
5. **U**: Duración del anuncio.

$$X * Y * Z * W * U = T \tag{3.1}$$

Así pues se podría poner como ejemplo lo siguiente. Supongamos que tenemos:

1. 30 vehículos.
2. 10 repeticiones al día por vehículo.
3. 22 días al mes.
4. 2 céntimos el segundo que son 0.02.
5. Y finalmente 15 segundos de anuncio.

La operación seria la siguiente:

$$30 * 10 * 22 * 0,02 * 15 = 1980 \tag{3.2}$$

En este ejemplo 1980 seria el precio de dicho anuncio

De esta cantidad el comercial se llevaría un 10 %, calculamos la cantidad percibida por él:

$$1980 * 10\,\% = 198 \tag{3.3}$$

Ahora bien bajo las mismas cantidades vamos a poner como ejemplo lo que se llevaría él conductor por el visionado del anuncio. Al ser solo un vehículo la operación seria:

- 10 repeticiones al día por vehículo

- 22 días al mes.
- 2 céntimos el segundo que son 0.02
- Y finalmente 15 segundos de anuncio.

$$10 * 22 * 0{,}02 * 15 = 66 \tag{3.4}$$

Al taxista le correspondería el 10 % del visionado de este anuncio

$$66 * 0{,}1 = 6{,}6 \tag{3.5}$$

6.6 al mes por un único anuncio. Lógicamente cuantos mas anuncios visionados más dinero sacaría al mes. Vamos a poner otro ejemplo:

- Calculamos que de las 12 horas que trabaja un taxista solo 4 sean productivas.
- En 4 horas tenemos un total de 14400 segundos si el taxista reproduce el total de dichos.
- Segundos a 2 céntimos el segundo su diez por ciento seria esta cantidad:

$$14400 * 0{,}02 * 0{,}1 = 28{,}8 \tag{3.6}$$

28.8 al día si lo multiplicáramos por 22 días hábiles sería:

$$28{,}8 * 22 = 633{,}6 \tag{3.7}$$

Así pues el taxista tendría un pago de 633.6 al mes por 4 horas de reproducción de anuncios en 22 días.

3.3 Gastos

En esta sección podremos calcular cuales son los gastos iniciales y mensuales de nuestro producto y de la infraestructura de la empresa y valorar el riesgo para nuestro patrimonio.

Producto	soporte	tablet	experimento cabezal	diseño logo	Soporte 3D + diseño (prototipo)
Precio	158	129	30	120	100

Figura 3.2: Tabla inversión inicial.

cable en L	cable extensión USB 5m	cargador coche usb	diseño logo graphene	Servidor OVH kimsufi	Dominio	Total
6,6	16,08	2,88	80	157,14	8,13	807,83

Figura 3.3: Tabla inversión inicial 2.

Como se puede apreciar en la figura 3.2 y figura 3.3 quedan reflejado los gastos que hemos tenido que realizar antes incluso de formalizar la empresa.

Muchos de ellos han servido para decantarnos por ideas a partir de prototipos como es el soporte tanto metálico como por impresión 3D. Incluso tuvimos la idea de comprar un reposacabezas e incrustar en *poliespan* el dispositivo.

También encargamos a un diseñador el diseño de nuestro logo tanto de empresa como de producto. Pedimos una cantidad de cables y una *tablet* para el prototipo y la programación y hacer pruebas sobre el terreno. Por supuesto la adquisición de un servidor para testear y el dominio donde se alojará la página web de nuestra empresa.

En la siguiente tabla, figura 3.4 aparece desglosado los gastos previstos para los primeros 6 meses de la vida de la empresa.

Se calcula que el dispositivo viene a costar unos 220 aproximadamente por unidad, siempre dependiendo del tamaño del lote, con *tablet* tarjeta SD almacenamiento 32GB, tarjeta SIM con tarifa mensual, cables, soporte con logos incluidos y pintado.

Después realizamos una un cálculo de la inversión inicial, desde el día de la implantación de la empresa, basada en los pagos únicos incluidos el seguro

	Precio U.	Precio	1 mes	6 meses	Total
Tablet 5 Unidades	137	685	-	-	
Soporte 5 U.	40	200	-	-	
Internet 5 U.	12	-	60	300	
Autónomo	-	-	356	1780	
Enchufes/cables 5 U.	5,112	25,56	-	-	
Seguro	200	1000	-	-	
Abogado	-	-	75	450	
Gestor	-	-	34	200	
Tarjeta SD 5U.	13	65	-	-	
Registro de marca	-	150	-	-	
total	420,112	2125,56	525	2730	4855,56

Figura 3.4: Tabla de pronóstico de gastos 6 meses.

anual de resposabilidad civil, el registro de la marca y los 5 dispositivos con los que se empezará a operar.

Otro cálculo es el de gastos mensuales que deberemos afrontar como es la cuota de autónomo, el abogado, el gestor de la empresa, además de la tarifa de teléfono para los dispositivos con la compañía. Esto supone una permanencia de 1 año y medio.

Otro de los cálculos es el precio que deberemos afrontar durante 6 meses de gastos mensuales antes mencionados. Y la última el coste total de la empresa estimado en 6 meses.

6 meses es lo que podemos estimar que podríamos llegar a tardar, como máximo, en afrontar que no entrasen ingresos de ningún tipo en la empresa y replantearnos estrategias.

Capítulo 4

Conclusiones

A día de hoy el dispositivo ha sido probado en un vehículo real pero no en un taxi. Su respuesta es mejor de lo que esperábamos y su percepción muy amigable.

Debemos de enfrentarnos a algunas cuestiones técnicas como si la tarifa de datos y la tarjeta de almacenamiento serán suficientemente óptimas para poner en compromiso el perfecto funcionamiento del sistema. Nada que la inversión en una tarifa de datos más amplia o más almacenamiento en el dispositivo no pudiese solucionar.

Otra cuestión si el taxista se comprometerá a cuidar del dispositivo y de que si la batería del automóvil funciona con el motor apagado. De no ser así, si el dispositivo se quedara sin batería y se apagara se deberá hacer cargo de arrancar el dispositivo una vez el coche esté en marcha y la fuente de alimentación carga la batería. En esta misma línea deberá abrir el candado y trasladar el dispositivo a un lugar seguro al acabar la jornada y traerlo de vuelta cuando comience la siguiente. Son aparatos muy golosos.

Constituida la empresa como **Graphene Code SL.** estamos a la espera de llegar a un acuerdo comercial con la cooperativa **ECOVLCTAXI** para implantar 5 dispositivos de prueba.

Llegados a este punto, de realizar *feedback* mediante las impresiones de los usuarios, clientes publicitarios y conductores de los taxis, refinaremos el producto para sacar una remesa de unos 20-30 dispositivos con los cuales podremos realizar los contratos publicitarios pertinentes.

También para depurar los posibles *BUGs* que el código pudiese tener. ¡¡El comportamiento del usuario es impredecible!!

Como trabajos futuros se podrían implementar muchas funcionalidades para el producto:

- **Pide Taxi**: Se podría implementar una funcionalidad extra de cálculo de rutas en la que desde que el cliente sube al taxi con su móvil con el taxi pedido, esta calculará el trayecto. Los datos de posicionamiento podrían recogerse y almacenarse para futuros estudios.
- **Pagos NFC**: Mediante la aplicación Pide Taxi que hubiese una opción de monedero electrónico en el que al pasar el móvil se pudiese pagar mediante tecnología NFC y algún portal de pago electrónico como `PayPal`.

  HotSpot: Mediante la implementación de un portal cautivo se podría crear un punto de acceso público a a internet y controlar su acceso a través de cuentas personales. Esto podría usarse para introducir publicidad en el navegador del cliente.

Para concluir puedo decir que esta experiencia me ha hecho crecer tanto profesional como humanamente. Teniendo en cuenta que hemos tenido que organizar una empresa con el trabajo que conlleva realizar un plan de empresa viable. Testear y pivotar en ciertos aspectos del producto además de calcular costes de tiempo entre las funcionalidades y decidir cuales eran las más imporantes, es decir, gestionar un proyecto de desarrollo de software.

En conclusión, una experiencia enriquecedora que me hace animar al resto de estudiantes de Ingeniería Informática y resto de titulaciones en general para emprender y llevar a cabo sus ideas.

Apéndice A

Lean Canvas

A.1 Contexto Lean Canvas

Muchos autores y gurús del mundo del emprendimiento han hablado sobre diferentes métodos que recomiendan para analizar y preparar un modelo de negocio apropiado. Sin embargo, un joven emprendedor que intentó utilizar muchos de ellos, fue más allá y creó su propio proceso, adaptando éstos métodos a su experiencia personal cada vez que se chocaba contra una pared o llegaba a un callejón sin salida.

Así fue como *Ash Maurya*, autor del libro `"Running Lean"`(Cómo iterar del Plan A a un Plan que funciona) creó el **Lean Canvas** o lienzo lean.

Pero para explicar su modelo, primero debemos detenernos en el **Business Model Canvas**.

A.1.1 Business Model Canvas

Maurya fue expuesto al `Business Model Canvas` (lienzo para modelos de negocio) por primera vez tras leer el libro `"Business Model Generation"`de **Alex Osterwalder**. A pesar de que el libro le parecía muy bien ilustrado, rechazó el enfoque del lienzo debido a que lo encontraba "muy simple". La mayoría de los ejemplos del libro mostraban modelos de negocios de grandes compañías como **Apple** y **Skype** después de ser exitosas, pero Maurya estaba mucho más interesado en aprender cómo éstas compañías alcanzaron al éxito.

Sin embargo el Business Model Canvas le sirvió como base para crear su propio lienzo. El canvas de Osterwalder es considerado una herramienta estratégica empresarial y de gestión que permite describir, diseñar, retar, inventar y

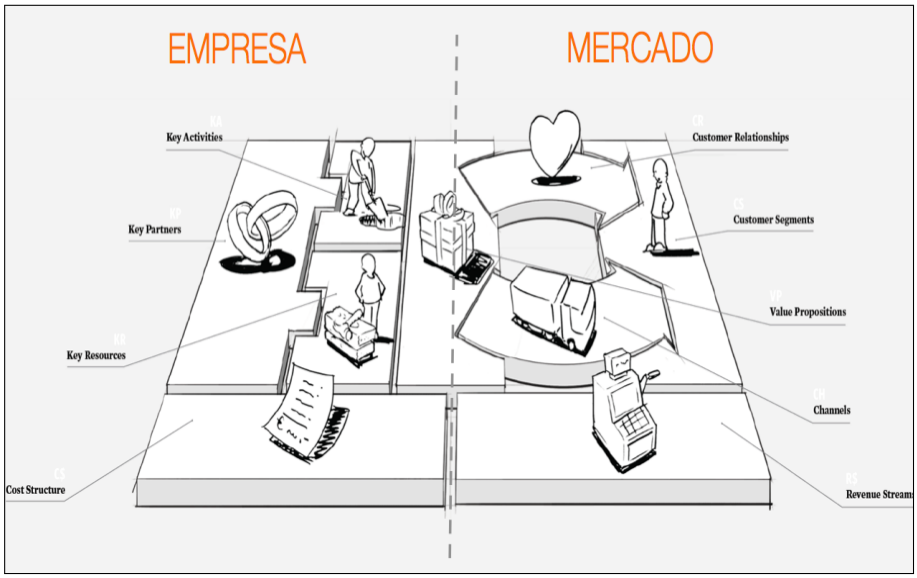

Figura A.1: Gráfico Business Model Canvas

pivotar nuevos modelos de negocio. Se basa en nueve pilares fundamentales, organizados en un lienzo pre-estructurado de 9 casillas, con el cual puedes hacer un mapa completo de tu modelo de negocio en una sola imagen. Éstos pilares son:

1. El segmento de mercado que comprende a todas las personas u organizaciones para los cuales estás creando valor (esto incluye usuarios simples y clientes que pagan).
2. Se debe tener una propuesta de valor única para cada segmento. Esta propuesta está conformada por los productos y servicios que crean valor para tus clientes.
3. Los canales o puntos a través de los cuales tienes contacto con tus clientes y les entregas las propuestas de valor.
4. Las relaciones que estableces con tus clientes.
5. Las fuentes de ingreso que generas (cómo y a través de qué mecanismos está generando valor tu modelo de negocios).
6. Los recursos clave son los activos indispensable para tu modelo de negocio, es decir, la infraestructura necesaria para crear, entregar y capturar valor.

7. Las actividades clave que requieres para generar ingresos, es decir, aquella cosa en la cual tu compañía realmente debe tener un buen desempeño.
8. Los socios clave que son los que te pueden ayudar a impulsar tu modelo de negocio, debido a que es probable que no seas dueño de todos los recursos clave que necesites, ni que puedas realizar todas las actividades claves tú solo.
9. Una vez comprendas la infraestructura de tu modelo de negocio, también tendrás una idea de la estructura de costes.

A.1.2 Lean Canvas

Después de intentar utilizar las hojas de trabajo de Steve Blank, propuestas en su libro "The Four Steps to Epiphany", y de echarle un vistazo a la variación que hizo su colega Rob Fitzpatrick's de ellas (The Startup Toolkit), Maurya decidió tomar el modelo de Osterwalder y optimizarlo para start-ups, creando el Lean Canvas. Lo probó exitosamente con varias start-ups en los talleres que impartía y lo posteó en su blog recibiendo mucho feedback positivo de otros emprendedores.

Su objetivo principal era que el Lean Canvas fuera lo más viable y práctico posible, sin perder el enfoque empresarial; crear una especie de guía o mapa que ayudara a los emprendedores a navegar por el proceso, desde el nacimiento de la idea hasta la creación de la start-up. Basándose en los principios de Lean Startup y en las palabras de Eric Ries, "las start-ups operan bajo condiciones de extrema incertidumbre", su idea era capturar aquello que era más incierto o más arriesgado. De acuerdo con Douglas Hubbard, en su libro "How to Measure Anything: Finding the Value of 'Intangibles' in Business", entendemos por incierto "la existencia de más de una probabilidad" y por arriesgado "un estado de incertidumbre donde una posibilidad puede involucrar una pérdida o un resultado no deseado".

Había algo en el Business Model Canvas original que no convencía del todo a Maurya: dentro de los nueve pilares, faltaban elementos de alto riesgo y algunos de los que estaban enumerados le parecían muy poco riesgosos. Así que decidió modificarlo eliminando unos y agregando otros. Agregó cuatro elementos:

1. Problema: es importante conocer bien el problema para poder encontrar la solución. "Un problema claramente señalado es un problema mitad-resuelto" –Charles Kettering.
2. Solución: una vez definido el problema, se puede ofrecer una posible solución. Ésta es una de las cajas más pequeñas del lienzo, adrede, con

la idea de limitar a los emprendedores y mantener el concepto de MVP o producto viable mínimo.

3. Métricas Clave: Según Noah Kagan, emprendedor y fundador de Appsumo –una página de ofertas de productos y servicios digitales-, "Una sartup sólo puede enfocarse en una métrica, por lo que debes decidir cuál es, e ignorar todo lo demás". El riesgo está en no saber identificar la métrica correcta y perder recursos correteando la meta equivocada.
4. Ventaja Injusta: También conocido como la ventaja competitiva o barreras de entrada que se suelen encontrar en un el de negocio. Una start-up nueva que está recién empezando, no tendría una ventaja injusta, por lo que esta casilla en principio estará en blanco. La idea de esta caja es motivar a los emprendedores a encontrar o construir su ventaja injusta. Una vez que la start-up consiga un nivel inicial de éxito, será inevitable la aparición de competidores y personas que copien la idea. Si no tienes una defensa contra ellos, corres el riesgo de que te extingan.

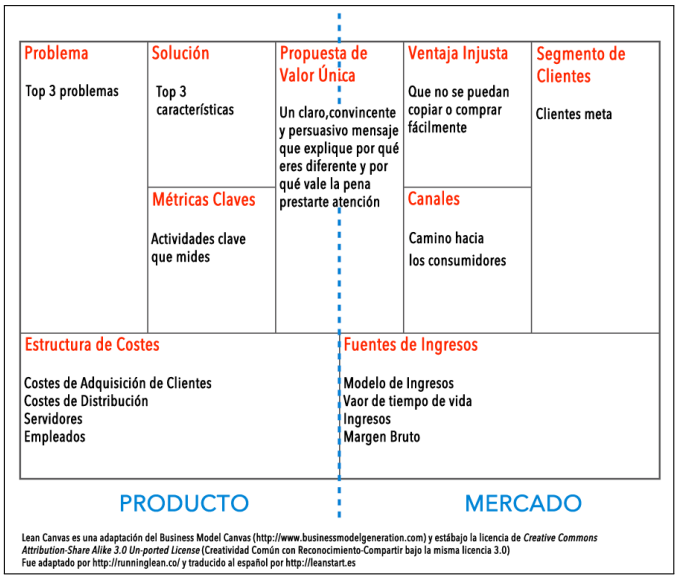

Figura A.2: Gráfico Lean Canvas

Para incorporar estos cuatro elementos al canvas, Maurya tuvo que quitar otros cuatro elementos:

1. Actividades y recursos claves: Para Maurya, estas casillas ayudaban a personas externas a la compañía a entender lo que hacía la start-up,

pero no ayudaban a los emprendedores a enfocarse. Las actividades clave deberían derivarse de la casilla Solución, una vez el MVP fuera testado y evaluado inicialmente. En cuanto a los recursos clave, construir y lanzar productos al mercado hoy en día no requiere de tantos recursos físicos gracias a Internet, la computación en nube, el software abierto, y la globalización. En ese caso, los recursos claves van más alineados con la ventaja injusta; sin embargo, a pesar de que los recursos clave pueden ser una ventaja injusta, no toda ventaja injusta es un recurso clave.

2. Relaciones con los clientes: Maurya cree fielmente en comenzar la creación de cualquier producto con una relación directa con los consumidores a través de entrevistas y observación, y así poder identificar el camino correcto para llegarles, según la Solución que se ha encontrado para ese Segmento de Mercado específico. Esto se presenta mejor en la casilla Canales.

3. Socios Clave: Para la creación de algunos productos, como por ejemplo una plataforma de energía solar, puede ser importante contar con socios claves desde el comienzo. Pero "para una start-up desconocida con un producto no testado, buscar socios claves desde el día uno, puede ser una forma de derroche o malgasto", dice Maurya. Con el tiempo, los socios pueden ser cruciales para optimizar una empresa, pero el riesgo real de una start-up no está en la falta de socios, sino en las deficiencias en la Estructura de Costes y los Canales de Distribución.

   Para más información visitar enlace LeanStack, la web donde aprender esta metodología, su autor es **Ash Maurya**.

Apéndice B

Modelo de Datos

En este anexo se detalla qué es un modelo de datos. Ver sección **??**.

Después se explica un tipo concreo de modelo de datos como es el modelo relacional que manipulan las llamadas base de datos relacionales en la sección B.1.1.

Por último se expone una solución a la diferencia de sistemas de tipos entre una base de datos relacional y un lenguaje orientado a objetos. Se ubica en la sección B.1.2

B.1 Introducción al Modelo de Datos

Un modelo de datos es un lenguaje orientado a hablar de una Base de Datos. Típicamente un modelo de datos permite describir:

- Las estructuras de datos de la base: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejan realidad deseada.
- Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

Otro enfoque es pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí.

No hay que perder de vista que una Base de Datos siempre está orientada a resolver un problema determinado, por lo que los dos enfoques propuestos son necesarios en cualquier desarrollo de software.

Un modelo de datos es un lenguaje que, típicamente, tiene dos sublenguajes:

- Un Lenguaje de Definición de Datos o DDL (Data Definition Language), orientado a describir de una forma abstracta las estructuras de datos y las restricciones de integridad.
- Un Lenguaje de Manipulación de Datos o DML (Data Manipulation Language), orientado a describir las operaciones de manipulación de los datos.

Una opción bastante usada a la hora de clasificar los modelos de datos es hacerlo de acuerdo al nivel de abstracción que presentan:

- **Modelos de Datos Conceptuales** Son los orientados a la descripción de estructuras de datos y restricciones de integridad. Se usan fundamentalmente durante la etapa de Análisis de un problema dado y están orientados a representar los elementos que intervienen en ese problema y sus relaciones. El ejemplo más típico es el Modelo Entidad-Relación.
- **Modelos de Datos Lógicos** Son orientados a las operaciones más que a la descripción de una realidad. Usualmente están implementados en algún Manejador de Base de Datos. El ejemplo más típico es el Modelo Relacional, que cuenta con la particularidad de contar también con buenas características conceptuales (Normalización de bases de datos).
- **Modelos de Datos Físicos** Son estructuras de datos a bajo nivel implementadas dentro del propio manejador. Ejemplos típicos de estas estructuras son los Árboles B+, las estructuras de Hash, etc.

A la parte del **DML** orientada a la recuperación de datos, usualmente se le llama *Lenguaje de Consulta* o**QL** (*Query Language*).

B.1.1 Modelo Relacional

El **Modelo Relacional**, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos.

Tras ser postuladas sus bases en 1970 por `Edgar Frank Codd`, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Pese a que esta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, pensando en cada relación como si fuese una *tabla* que está compuesta por *registros* (cada fila de la tabla sería un registro o "*tupla*") y *columnas* (también llamadas "*campos*").

Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

Para más información sobre el modelo relacional consultar la cita **ModeloRelacional**

B.1.2 Modelo Objeto-Relacional

El mapeo **Objeto-Relacional** (más conocido por su nombre en inglés, *Object-Relational mapping*, o sus siglas `O/RM`, `ORM`, y `O/R mapping`), ver figura B.1, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas `ORM`.

Para más información sobre el modelo relacional consultar la cita **ModeloObjetoRelacional**

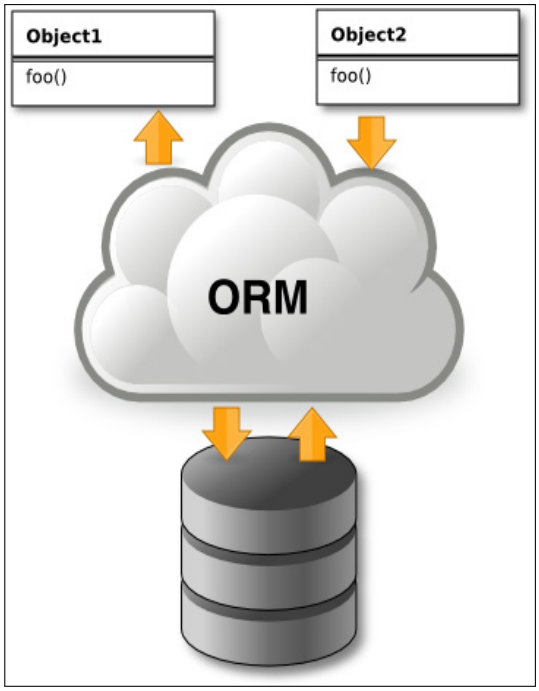

Figura B.1: ORM.

Apéndice C

Sails

En este anexo se introduce una breve explicación de que es **Nodejs**, el motor sobre el que corre y sus características principales. Ver sección C.1.

Y después una breve introducción de **Sailsjs** y un poco de manejo sobre él. Ver sección C.2

C.1 Introducción a Node

Para explicar que es **Sailsjs** primero se debe explicar que es **Nodejs**.

`Node.js`®es un *framework* en tiempo de ejecución de **JavaScript** construido sobre el motor *JavaScript V8* de `Chrome` . `Node.js` utiliza una , el bloqueo no- modelo orientado a eventos de E/S que lo hace ligero y eficiente. El paquete ecosistema `Node.js`, la `NGP`, es el mayor ecosistema de las bibliotecas de código abierto en el mundo.

Como un marco orientado a eventos asíncronos, `Node.js` está diseñado para construir aplicaciones de red escalables. En el siguiente ejemplo "*hola mundo*",se pueden manejar muchas conexiones al mismo tiempo. Tras cada conexión la devolución de llamada se dispara , pero si no hay trabajo por hacer Nodo duerme.

```
1 var http = require('http');
2
3 http.createServer(function (req, res) {
4 res.writeHead(200, {'Content-Type': 'text/plain'});
5 res.end('Hello World\n');
6 }).listen(1337, "127.0.0.1");
7
8 console.log('Server running at http://127.0.0.1:1337/');
```

Esto está en contraste con los modelos de concurrencia más comunes cuando se emplean hilos en los *SO*. Los hilos basados en conexión de redes son relativamente ineficientes y muy difíciles de usar. Por otra parte, los usuarios de Nodo están libres de preocupaciones de bloqueos por muerte del proceso pues no hay bloqueos. Casi ninguna función en el nodo realiza directamente peticiones de **E/S**, por lo que el proceso nunca se bloqueará. Ya que nada bloquea los procesos, cualquier programador no experto es capaz de desarrollar sistemas escalables.

Nodo es un diseño similar e influenciado por los sistemas como el *Evento de* ***Ruby*** o **Twisted**, la *Máquina de Python*. Nodo utiliza el modelo de eventos un poco más allá, se presenta el ciclo de eventos como una construcción del lenguaje en lugar de como una biblioteca. En otros sistemas siempre hay una llamada de bloqueo para iniciar el bucle de eventos. Normalmente se define devoluciones de llamada a través de un comportamiento en el comienzo de un *script* y al final se inicia con el servidor a través de una llamada de bloqueo como *EventMachine :: run ()*. En Nodo no hay tal llamada puesta en el bucle de lanzamiento. Nodo simplemente entra en el bucle de eventos después de ejecutar la entrada del *script*. Nodo sale del bucle de eventos cuando no hay más devoluciones de llamada. Este comportamiento se realiza como en un navegador `JavaScript` - el bucle de eventos está oculto para el usuario. Se realiza de forma transparente.

El protocolo **HTTP** es prioridad en `Node.js`, diseñado con *streaming* y pensado para una baja latencia de recursos. `Node` es muy adecuado para la creación de bibliotecas web o entornos de desarrollo.

No porque `Node` esté diseñado sin hilos, significa que no pueda tomar ventaja de los multi-procesadores. Puede generar procesos con hijos los cuales son fáciles de comunicarse entre si usando nuestra **API** $child_process.fork$. Construido sobre lo que es el mismo grupo de módulo de interfaz, permite compartir *sockets* entre procesos para el equilibrio de carga sobre sus núcleos.

C.1.1 Motor V8 de google

Como dijimos en la sección C.1, **Node.js** esta basado en el *motor V8 de JavaScript de Google*. Este motor está diseñado para correr en un navegador y ejecutar el código de *JavaScript* de una forma extremadamente rápida.

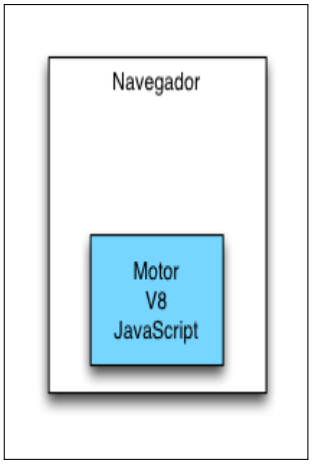

Figura C.1: Motor V8 Google

La tecnología que está detrás de `Node.js` permite ejecutar este motor en el lado del servidor abriendo un nuevo abanico de posibilidades en cuanto al mundo de desarrollo se refiere.

Para ello el entorno de `Node.js` ha desarrollado un conjunto amplio de librerías que no tienen nada que envidiar a otras plataformas. Por otro lado se han eliminado algunas funcionalidades que en el entorno de servidor no tenían sentido como por ejemplo el uso de *Document Object Model*.

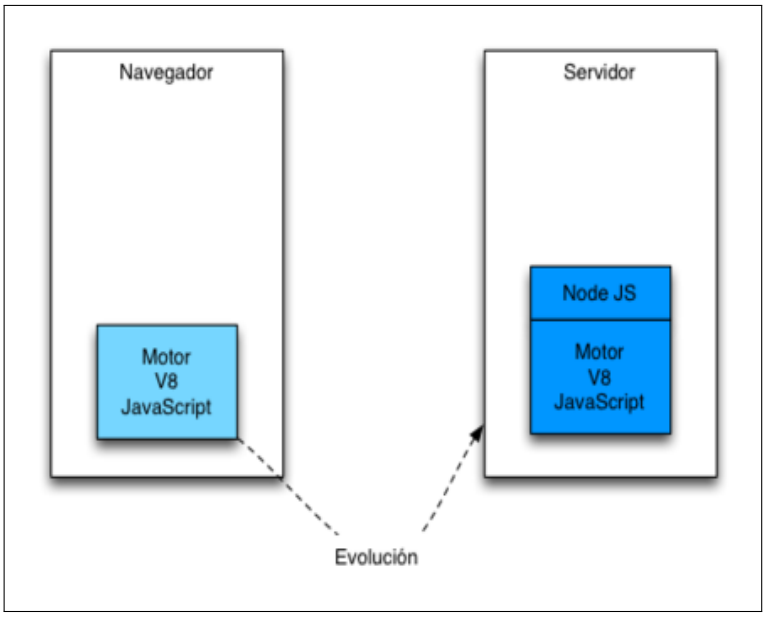

Figura C.2: Motor V8 Google + Nodejs

C.1.2 Node e Hilos: Programación Orientada a Eventos

`Node.js` trabaja con un único hilo de ejecución que es el encargado de organizar todo el flujo de trabajo que se deba realizar.

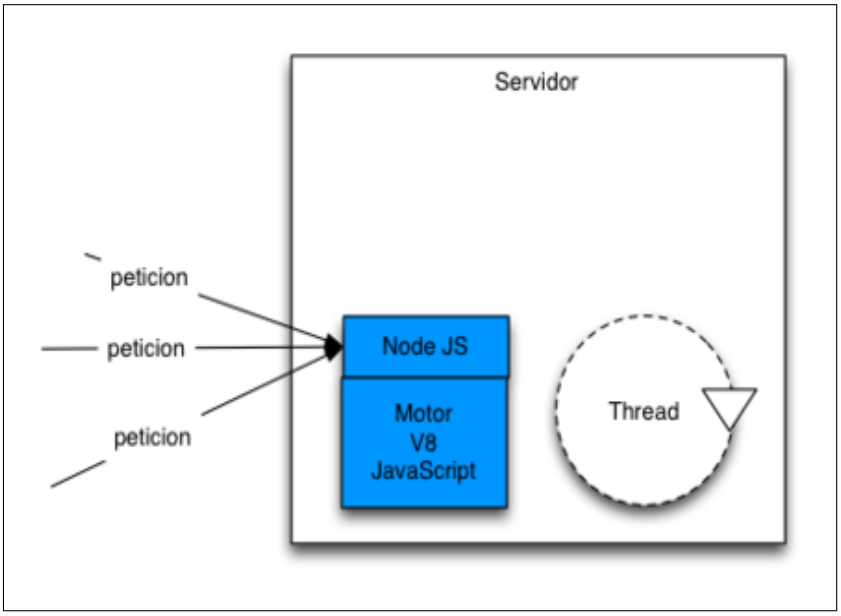

Figura C.3: Nodejs Thread

La primera duda que nos surge es: ¿Qué sucede si ese hilo se pone a realizar una tarea que lleva X tiempo y queda bloqueado?. Esa es una buena pregunta y la respuesta es que `Node.js` gestiona todas sus tareas de una forma *asíncrona*. Esto nos puede sonar mucho a **AJAX**. Ahora bien ¿Cómo funcionan esas peticiones asíncronas exactamente?.

Para poder trabajar de una forma óptima `Node.js` delega todo el trabajo en una cola de hilos ("*pool de threads*". Esta cola de hilos esta construido con la librería **libuv**. Esta librería dispone de su propio entorno multithread asíncrono. `Node.js` envía el trabajo que hay que realizar a la cola.

Libuv realizará a través de alguno de sus hilos el trabajo encomendados. Una vez que el trabajo haya sido completado `libuv` emitirá un evento que será recibido por `Node.js`.

Recibido el evento una función de *callback* se encargará de terminar de procesarlo. Por eso cuando trabajamos con `Node.js` prácticamente toda la pro-

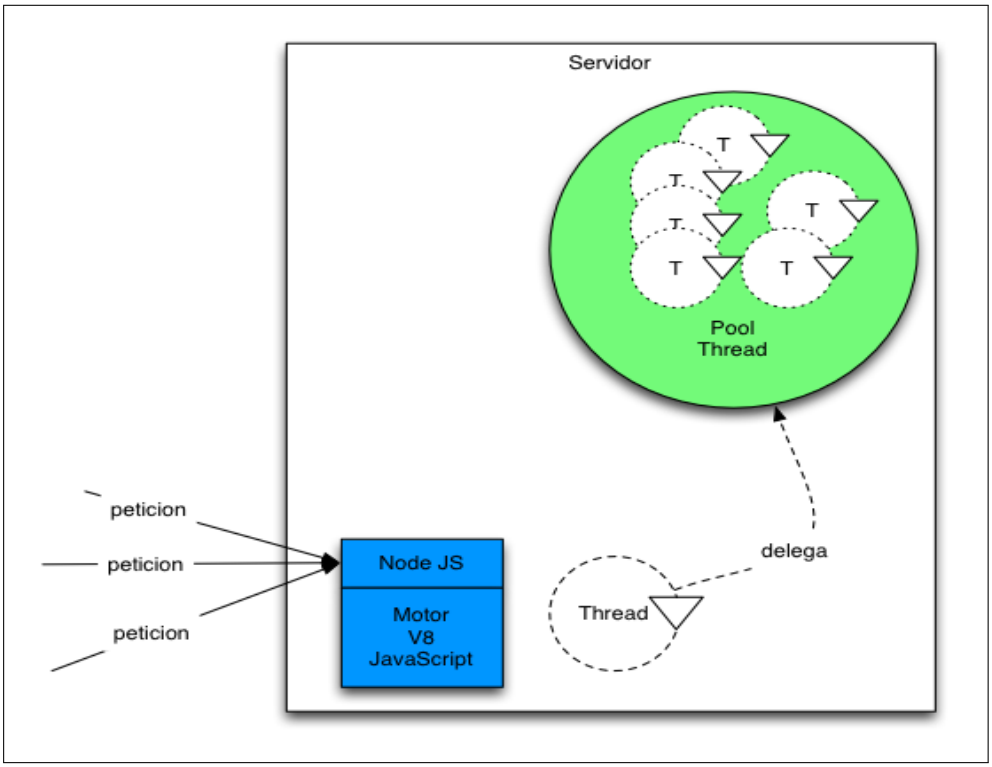

Figura C.4: Nodejs Thread Pool

gramación es asíncrona y se parece tanto a las clásicas llamadas **AJAX** que realizamos. Vamos a ver un sencillo ejemplo de código en el cual utilizamos `Node.js` para leer un fichero:

```
1 fs = require('fs')
2 fs.readFile('fichero.txt', 'utf8', function (err,datos) {
3   if (err) {
4     return console.log(err);
5   }
6   console.log(datos);
7 });
```

Como podemos ver incluso algo tan sencillo como leer un fichero se realiza de forma asíncrona y dispone de una función de *callback* ya que prácticamente todo el trabajo se delega.

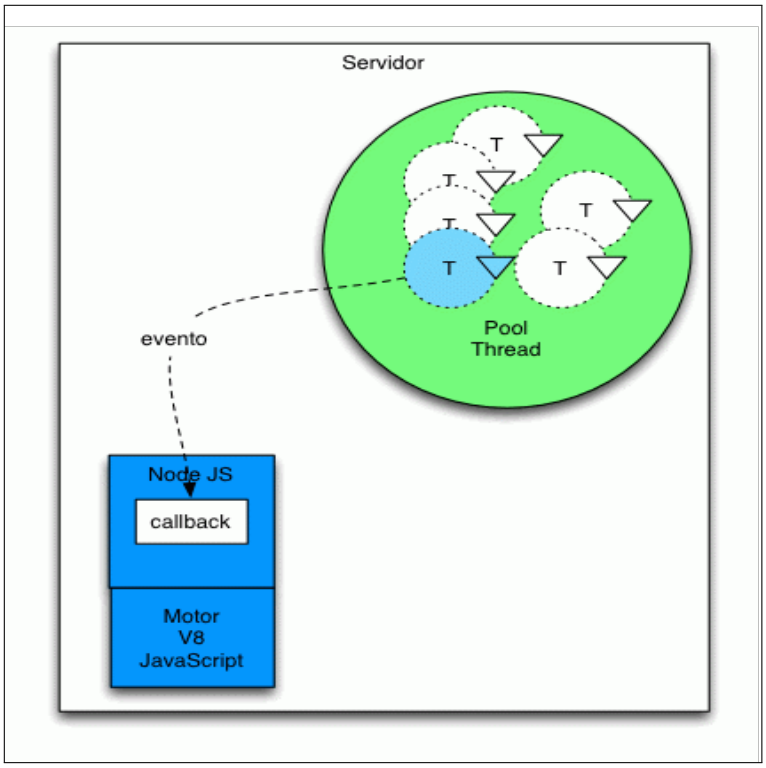

Figura C.5: Nodejs Callbacks

Para más información visite cita **Nodejs** **NodejsWikipedia** **NodejsGenbetadev** y **NodejsGenbetadev**

C.2 Introducción a Sails

Sails hace que sea fácil de construir de forma "*custom*", aplicaciones para una gran la empresa mediante Node.js . Está diseñado para emular el conocido como patrón de marcos **MVC** como `Ruby on Rails`, pero con el apoyo a los requisitos de las aplicaciones modernas : Estas son *APIs* basadas en datos con un servicio orientado a una arquitectura escalable. Es especialmente bueno para la construcción de chat, tableros de control en tiempo real (monitores), o juegos multi-jugador; pero se puede utilizar para cualquier proyecto de aplicación web - pasando por todas las capas de la arquitectura.

Una de las cosas que nos permite `Sails.js`, entre muchas otras, es el desarrollo de forma sencilla y rápida de un **API REST**. Por eso considero que puede ser una herramienta muy importante del stack de software que comentamos en el artículo desarrollo de aplicaciones web modernas.

C.2.1 Creando API en Sails

Ejecutando únicamente un par de instrucciones e insertando una única línea de código conseguimos desarrollar el mismo API que desarrollamos en el artículo desarrollo de un API REST con MongoDB, NodeJS y Express, pero con más funcionalidades:

```
1
2 npm install sails@beta -g
3 sails generate new MyApi --no-front-end
4 cd MyApi
5 npm install
6 sails generate api message
```

Como vemos, en primer lugar generamos nuestro proyecto mediante el comando sails generate new MyApi –no-front-end y después generamos nuestro API para la entidad Message mediante el comando sails generate api message.

Finalmente tenemos que definir las propiedades de nuestra entidad, la cual se compone únicamente de la propiedad text. Por lo tanto, abrimos el fichero api/models/Message.js y definimos sus propiedades:

```
1
2 module.exports = {
3   attributes: {
4     text: { type: 'string'}
5   }
6 };
```

C.2.2 Waterline, el ORM de Sails.js

`Sails.js` utiliza el `ORM` `Waterline`, que es desarrollado y mantenido por el mismo equipo de desarrollo de `Sails.js`.

`Waterline` dispone de adaptadores para conectarse a varias bases de datos (`MySQL`, `PostgreSQL` y `MongoDB`, entre otras). También podemos hacer que almacene los datos en disco o en memoria (no recomendado para producción), con lo cual podríamos probar nuestras aplicaciones sin necesidad de tener instalado un gestor de base de datos.

Por defecto, `Sails.js` está configurado para utilizar el local *disk adapter* (almacena los datos en un directorio temporal de nuestro sistema de ficheros). En el fichero *config/connections.js* están definidas nuestras conexiones a base de datos y en el fichero `config/models.js` se define qué conexión queremos que nuestras entidades utilicen por defecto. Cuando definimos cada una de nuestras entidades podemos especificar qué conexión queremos que utilice. Si no especificamos ninguna entonces `Sails.js` utilizará aquella que esté definida en *config/models.js*.

C.2.3 Prueba de Sails.js

Para probar nuestro API previamente tenemos que arrancar nuestra aplicación ejecutando la siguiente instrucción:

```
1
2 sails lift
```

Una vez arrancado el servidor podemos utilizar la extensión *POSTMAN* de `Chrome` para hacer peticiones a nuestro **API**.

Si queremos obtener todos los mensajes, lanzaremos una petición **GET** a la *URL http://localhost:1337/message*. Ver figura C.6.

C.2.4 Conclusiones Sails.js

En este artículo hemos visto que el desarrollo de un **API REST** con `Sails.js` es muy sencillo y muy rápido. Simplemente ejecutando un par de instrucciones de `Sails.js` podemos crear la estructura de nuestra aplicación y un **API**. A partir ahí sólo tenemos que definir las propiedades de nuestras entidades y sus relaciones.

Además, `Sails.js` ofrece muchas más características de las que hemos visto en este artículo, como notificaciones en tiempo real, gestión de la seguridad y un **ORM** que nos permite trabajar indistintamente con varios gestores de bases de datos, entre muchas otras características.

Para más información visite cita **Sails** y **SailsMonteagudo**

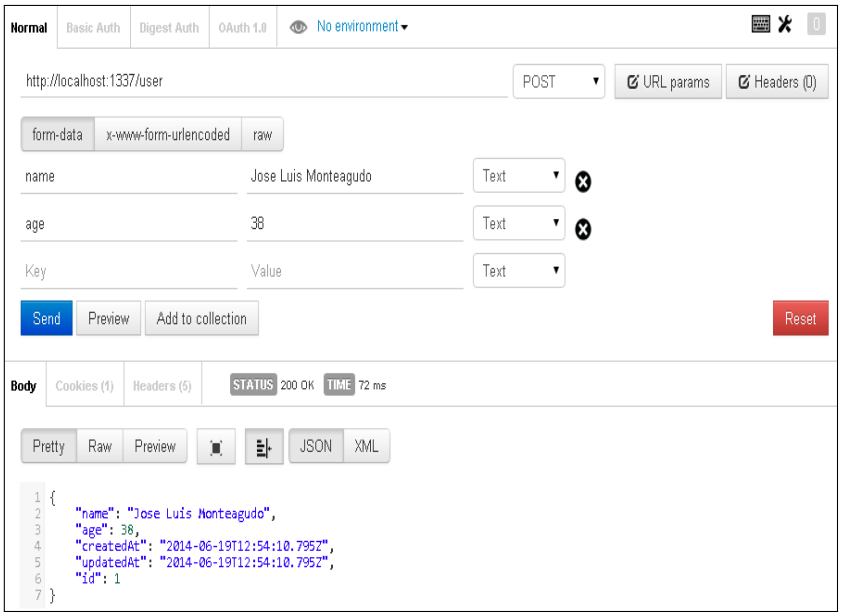

Figura C.6: Sails get

Referencias

[1] Ash Maurya: Running Lean: Iterate from Plan A to a Plan That Works. The Lean Series. Ed. O'Reilly. https://s3.amazonaws.com/runninglean/book/Running_Lean_Second_Edition.pdf

[2] Leanstart Lean Canvas: http://www.leanstart.es/lean-canvas/

[3] Emprenderalia Lean Canvas: http://www.emprenderalia.com/lean-canvas-aprende-a-disenar-modelos-de-negocio/

[4] Analizatuidea Lean Canvas: http://www.analizatuidea.com/como-y-porque-usar-lean-canvas-para-el-analisis-de-tu-idea\de-negocio-en-internet/

[5] Advenio Lean Canvas: http://advenio.es/lean-canvas-una-fusion-entre-el-lienzo-del-modelo-de \negocio-y-lean-startup/

[6] LeanStack Lean Canvas: https://app.leanstack.com

[7] Wikipedia DAFO: https://es.wikipedia.org/wiki/An%C3%A1lisis_DAFO

[8] Emprendedores DAFO: http://www.emprendedores.es/gestion/como-hacer-un-dafo

[9] BlogSalmon DAFO: http://www.elblogsalmon.com/conceptos-de-economia/que-es-el-dafo-y \cual-es-su-valia-como-herramienta-analitica

[10] Guia Calidad DAFO: http://www.guiadelacalidad.com/modelo-efqm/analisis-dafo

[11] Strategic development and SWOT analysis at the University of Warwick: http://web.archive.org/web/20120913032253/http://www2.egi.ua.pt/cursos/files/sad/strategic%20development%20and%20SWOT%20analysis.pdf

[12] Brigmton BTPC-1015GC-3G http://www.pccomponentes.com/brigmton_btpc_1015qc_10_1__8gb_3g_blanca.html

[13] Cable Micro USB de ángulo izquierdo 90º http://es.aliexpress.com/store/product/Left-Angled-90-Degree-Micro-USB-Male-to-USB-Data \ Charge-Cable-for-Cell-phone-for/1420178_32269362173.html.

[14] Cable alargador USB http://es.aliexpress.com/store/product/Black-5M-USB-2-0-Extension-Cable-Type \ A-Male-to-Type-A-Female-Extender-Cable/505394_32371836660.html.

[15] Cargador Dual USB para coche http://www.aliexpress.com/snapshot/6755312313.html?orderId=67940509762839.

[16] Google Text-to-Speech https://play.google.com/store/apps/details?id=com.google.android.tts&hl=es

[17] Uber España https://www.uber.com/es-ES/

[18] Luarstudio http://www.luarstudio.es/publicidad-digital-en-taxis.html

[19] Kimsufi http://www.kimsufi.com/es/

[20] Modelo de Datos https://es.wikipedia.org/wiki/Modelo_de_datos

[21] Modelo Relacional https://es.wikipedia.org/wiki/Modelo_relacional

[22] Mapeo Objeto Relacional https://es.wikipedia.org/wiki/Mapeo_objeto-relacional

[23] Nodejs https://nodejs.org/en/

[24] NodeJs Wikipedia https://es.wikipedia.org/wiki/Node.js

[25] NodeJs Genbetadev http://www.genbetadev.com/frameworks/como-funciona-node-js

[26] NodeJs Genbetadev http://blogs.lanacion.com.ar/freeware/packs/backup-en-tiempo-real/

[27] http://sailsjs.org/

[28] Sails Monteagudo http://www.jlmonteagudo.com/2014/06/desarollo-de-un-api-rest-con-sails-js/