

DESARROLLO DE UNA APLICACIÓN PROTOTIPO PARA LA LOCALIZACIÓN DE
PARQUEADEROS EN LA PLATAFORMA iOS

MIGUEL DARIO ROJAS CORTES

UNIVERSIDAD EAN
FACULTAD DE INGENIERIA
INGENIERIA DE SISTEMAS
BOGOTÁ D.C.

2013

DESARROLLO DE UNA APLICACIÓN PROTOTIPO PARA LA LOCALIZACIÓN DE
PARQUEADEROS EN LA PLATAFORMA iOS

Autor:

MIGUEL DARIO ROJAS CORTES

Tesis de grado presentada como requisito para optar por el título de:

INGENIERÍA DE SISTEMAS

Modalidad:

MONOGRAFÍA

Asesor:

INGENIERO RUBEN DORADO

DOCENTE ACADÉMICO

UNIVERSIDAD EAN

FACULTAD DE INGENIERIA

INGENIERIA DE SISTEMAS

BOGOTÁ D.C.

2013

AGRADECIMIENTOS

A mi madre que con su amor y dedicación fundó las bases de un ser humano correcto.

A mi tío Ernesto, que un día cualquiera, me ofreció (sin yo saberlo) mejorar mi calidad de vida.

A mi familia en general, que siempre estuvo pendiente de mi progreso.

A las buenas amistades adquiridas durante el proceso educativo.

A la parte docente de la facultad de Ingeniería, siempre atenta y dispuesta a apoyar mi labor.

ÍNDICE

1. TITULO.....	12
2. PLANTEAMIENTO DEL PROBLEMA.....	13
<i>2.1 Descripción</i>	<i>13</i>
3. FORMULACION DEL PROBLEMA	14
4. OBJETIVOS	15
<i>4.1 Objetivo General</i>	<i>15</i>
<i>4.2 Objetivos Específicos.....</i>	<i>15</i>
5. JUSTIFICACIÓN.....	16
6. ALCANCE.....	19
7. MARCO DE REFERENCIA.....	20
<i>7.1 Marco Teórico.....</i>	<i>20</i>
<i>7.1.1 Introducción</i>	<i>20</i>
<i>7.1.2 Desarrollo de aplicaciones móviles.....</i>	<i>25</i>
<i>7.1.3 MinTIC</i>	<i>27</i>
<i>7.1.4 Desarrollo Nativo.....</i>	<i>29</i>

7.1.5 Desarrollo No Nativo.....	33
7.1.6 Ciclo de vida.....	39
7.1.7 PhoneGap	41
7.1.8 Metodologías.....	45
7.1.8.1 Planeación.....	46
7.1.8.2 Diseño.....	47
7.1.8.3 Codificación	47
7.1.8.4 Pruebas.....	48
7.1.9 Análisis de Requerimientos.....	48
7.1.9.1 Requerimientos Funcionales y No Funcionales.....	49
7.1.10 Interfaz Humana	49
7.1.11 Georeferenciación	54
7.2 Marco Conceptual	57
7.2.1 Xcode.....	58
7.2.2 Interface Builder.....	58
7.2.3 Frameworks.....	59

7.2.4 <i>HTML</i>	59
7.2.5 <i>CSS</i>	60
7.2.6 <i>JavaScript</i>	60
7.2.7 <i>JQuery</i>	61
7.2.8 <i>DOM</i>	61
7.2.9 <i>Archivo Binario</i>	61
7.2.10 <i>Plugin</i>	62
7.2.11 <i>Lenguaje de programación</i>	62
7.2.12 <i>SPI</i>	63
7.2.13 <i>API</i>	63
7.2.14 <i>GPS</i>	63
7.2.15 <i>Base de datos</i>	64
8. ANÁLISIS DE REQUERIMIENTOS	65
8.1 <i>Descripción de Requerimientos Funcionales</i>	69
8.2 <i>Descripción de Requerimientos No Funcionales</i>	71
9. MODELAMIENTO DEL PROTOTIPO	73

9.1 Casos de uso.....	73
9.2 Modelo Entidad-Relación.....	81
9.3 Arquitectura del Prototipo.....	87
9.3.1 Arquitectura Web.....	88
9.3.2 Arquitectura Nativa (PhoneGap).....	91
9.4 Diseño de Interfaz Gráfica.....	93
9.5 Diagramas de secuencia.....	102
9.5.1 Localización de parqueadero.....	103
9.5.2 Clasificación de parqueaderos.....	105
9.5.3 Calcular costo de parqueo.....	106
10. IMPLEMENTACIÓN DE LA APLICACIÓN.....	108
10.1 Vistas Prototipo.....	110
10.1.1 Localización Parqueaderos.....	113
10.1.2 Clasificación de Parqueaderos.....	117
10.1.3 Cálculo de Costo de Parqueo.....	118
10.2 Delimitación del área de parqueaderos públicos.....	119

<i>10.2.1 EMAUS</i>	120
<i>10.2.2 QUINTA CAMACHO</i>	121
<i>10.2.3 CHAPINERO NORTE</i>	122
<i>10.3 Migración a entorno nativo</i>	123
11. PRUEBAS	124
<i>11.1 Pruebas de funcionalidad</i>	124
<i>11.2 Pruebas de usabilidad</i>	132
12. CONCLUSIONES	134
13. BIBLIOGRAFIA	138

LISTA DE FIGURAS

Figura 1 Estadísticas de sistemas operativos móviles en Colombia18

Figura 2 Conocimiento y herramientas para el desarrollo de apps nativas.....30

Figura 3 Conocimiento Base para el Desarrollo de una Web App.....35

Figura 4 Arquitectura PhoneGap.....43

Figura 5 Arquitectura PhoneGap Aplicada a la Plataforma iOS.....44

Figura 6 Requerimientos Funcionales.....68

Figura 7 Requerimientos No Funcionales.....68

Figura 8 Primer Requerimiento Funcional.....69

Figura 9 Segundo Requerimiento Funcional.....70

Figura 10 Tercer Requerimiento Funcional.....70

Figura 11 Cuarto Requerimiento Funcional.....70

Figura 12 Primer Requerimiento No Funcional.....71

Figura 13 Segundo Requerimiento No Funcional.....71

Figura 14 Tercer Requerimiento No Funcional.....72

Figura 15 Descripción Primer Caso de Uso.....74

Figura 16 Descripción Segundo Caso de Uso.....77

Figura 17 Descripción Tercer Caso de Uso.....79

Figura 18 Modelo Entidad-Relación iParqueo.....86

Figura 19 Arquitectura Prototipo - Web App.....90

Figura 20 Arquitectura Prototipo - PhoneGap.....92

Figura 21 Plantilla iPhone - Dimensiones.....93

Figura 22 Plantilla iPhone - Mockup Icono.....94

Figura 23 Plantilla iPhone - Splash Screen.....95

Figura 24 Plantilla iPhone - Mockup Vista Inicial.....96

Figura 25 Plantilla iPhone - Mockup Vista Localización.....97

Figura 26 Plantilla iPhone - Mockup Vista Clasificación Parqueaderos.....98

Figura 27 Plantilla iPhone - Mockup Vista Cálculo Costo Parqueadero.....99

Figura 28 Descripción de Controles de Interfaz Gráfica.....101

Figura 29 Diagrama De Secuencia - Localización de Parqueadero.....104

Figura 30 Diagrama De Secuencia - Clasificación de Parqueaderos.....105

Figura 31 Diagrama De Secuencia - Calcular Costo de Parqueo.....107

Figura 32 Splash Screen para Prototipo.....109

Figura 33 Formulario web para el ingreso de nuevos parqueaderos a iParqueo.....112

Figura 34 Vista inicial de prototipo.....113

Figura 35 Escenarios en respuesta a la petición de usuario.....114

Figura 36 Vista Localización prototipo iParqueo.....115

Figura 37 Trazo de ruta - Vista Localización Prototipo.....116

Figura 38 Vista Clasificación De Parqueaderos.....117

Figura 39 Vista Cálculo de Costo.....118

Figura 40 Barrio Emaús.....120

Figura 41 Barrio Quinta Camacho.....121

Figura 42 Barrio Chapinero Norte.....122

LISTA DE TABLAS

Tabla 1 Descripción de ciclos de vida empleados en el desarrollo de aplicaciones móviles.....40

Tabla 2 Dimensiones y resoluciones de pantalla por dispositivo.....52

Tabla 3 Componentes gráficos de acuerdo a dispositivo iOS.....53

Tabla 4 Comparación entre iconos principales del prototipo.....108

Tabla 5 Prueba tiempo de respuesta - Localización de Parqueaderos.....126

Tabla 6 Prueba tiempo de respuesta - Obtener Información Parqueaderos.....128

Tabla 7 Prueba tiempo de respuesta - Trazar ruta usuario - Parqueadero.....130

Tabla 8 Resumen pruebas de funcionalidad.....131

1. TITULO

“Desarrollo de aplicación prototipo para localización de parqueaderos en plataforma iOS”

2. PLANTEAMIENTO DEL PROBLEMA

2.1 Descripción

En la actualidad, la ubicación de un parqueadero en la ciudad de Bogotá no es tarea fácil, fenómeno generado por el atraso vial que caracteriza a la capital, sumado a la pésima gestión de las obras para mejorar el estado de las vías, las facilidades para la adquisición de vehículo, los cambios de nomenclatura en las direcciones y las políticas establecidas para hacer uso del automóvil como medio de transporte las cuales, en ciertos horarios, generan congestiones vehiculares de grandes dimensiones entre otros aspectos.

Los dispositivos móviles se presentan como la herramienta idónea para obtener de forma práctica la localización de un estacionamiento cercano. La constante evolución de las plataformas móviles y el boom de las “apps” encaja a la perfección en el contexto de esta situación. En particular, la plataforma iOS se muestra especialmente interesante dada la gran acogida que ha tenido desde su aparición allá por 2007. Su tienda de aplicaciones es todo un éxito y hay poca competencia para el problema mencionado, pues las aplicaciones disponibles están muy por debajo de lo que se puede ofrecer a un usuario, especialmente en términos de la información que se le presenta al usuario, como por ejemplo el valor del minuto, valor del día, horarios, capacidad y tipos de vehículos aceptados (motos, automóviles).

3. FORMULACION DEL PROBLEMA

¿Cómo facilitar la localización de un parqueadero en la localidad de Chapinero por medio de un dispositivo móvil con iOS?

4. OBJETIVOS

4.1 Objetivo General

Desarrollar una aplicación prototipo denominada iParqueo, por medio de la cual el usuario de un dispositivo iOS podrá ubicar el estacionamiento más cercano y económico disponible dentro de un área específica en la localidad de Chapinero.

4.2 Objetivos Específicos

- Definir los requerimientos de los usuarios de una aplicación de búsqueda automática de parqueaderos.
- Diseñar un modelo de aplicación para construir un software de manejo y búsqueda de parqueaderos.
- Implementar un prototipo de aplicación en plataforma móvil (iOS) de búsqueda de parqueaderos por zona.
- Evaluar el prototipo mediante un conjunto de pruebas

5. JUSTIFICACIÓN

Según el observatorio ambiental de movilidad, más de 1.2 millones de vehículos particulares circulan a diario por la ciudad de Bogotá, cifra que se eleva cada vez más dadas las facilidades actuales para adquisición de vehículo. De acuerdo a esta situación, y teniendo en cuenta la delimitación de la zona sobre la cual se desea que el producto ofrezca el servicio, (cuyo radio comprende el centro financiero de la ciudad, grandes instituciones académicas y una buena cadena de restaurantes entre otros importantes puntos concurridos de la ciudad), existe una base interesante de los clientes potenciales producto de un filtro realizado dado el campo de acción lógico a abordar, el cual, como bien se mencionó anteriormente, no es otro que el terreno de las aplicaciones móviles dado el contexto implícito de movilidad.

Dado el auge que presenta en la actualidad este sector, desde aquella revolución generada en la industria de la telefonía móvil celular con la presentación del iPhone de Apple en 2007, el abanico de opciones que actualmente presenta el mercado para las distintas plataformas, se hace cada vez mas grande, ampliando las posibilidades para los desarrolladores y, por otro lado, forzando a los operadores de telefonía móvil a ofrecer una mejor calidad de servicio, que, dentro de un contexto técnico, obedece a reforzar la infraestructura que hace posible al usuario obtener una calidad decente en términos de conectividad en casi cualquier zona en la que se encuentre prestando cobertura el operador y así, garantizar un correcto aprovechamiento de las características de los dispositivos móviles.

De acuerdo a las estadísticas facilitadas por el Ministerio de las Tecnologías de la Información y Telecomunicaciones (MINTIC), y con base en el informe del 4to trimestre del año 2012, existen aproximadamente 1'925,435 usuarios de telefonos móviles afiliados a un plan de datos, distribuidos en terminales 2G y 3G.

La plataforma elegida para llevar a cabo este proyecto es iOS, sistema operativo de los dispositivos móviles de Apple, que al día de hoy, se encuentra presente en el iPad, iPod Touch y iPhone. Los argumentos para construir una aplicación de este tipo, y de elegir dicha plataforma para este desarrollo, son entre otros, la gran aceptación del público frente a los productos de esta compañía a pesar de no ser muy competitivos en materia económica, pues de acuerdo a StatCounter, a la fecha, iOS es el segundo sistema operativo móvil más usado en Colombia con el 25.77% del mercado (ver Figura 1); así mismo, resaltan la integración software/hardware, la estabilidad de su sistema operativo como consecuencia del punto anterior, la relativa homogeneidad en las dimensiones de sus pantallas, las aproximaciones que pueden hacer los desarrolladores ya que no están sujetos a un único lenguaje de programación para poder dar vida a una aplicación y muchas otras cualidades enmarcadas en su tienda de aplicaciones que a la fecha, ha alcanzado más de 50 billones de descargas en poco menos de 5 años de existencia.

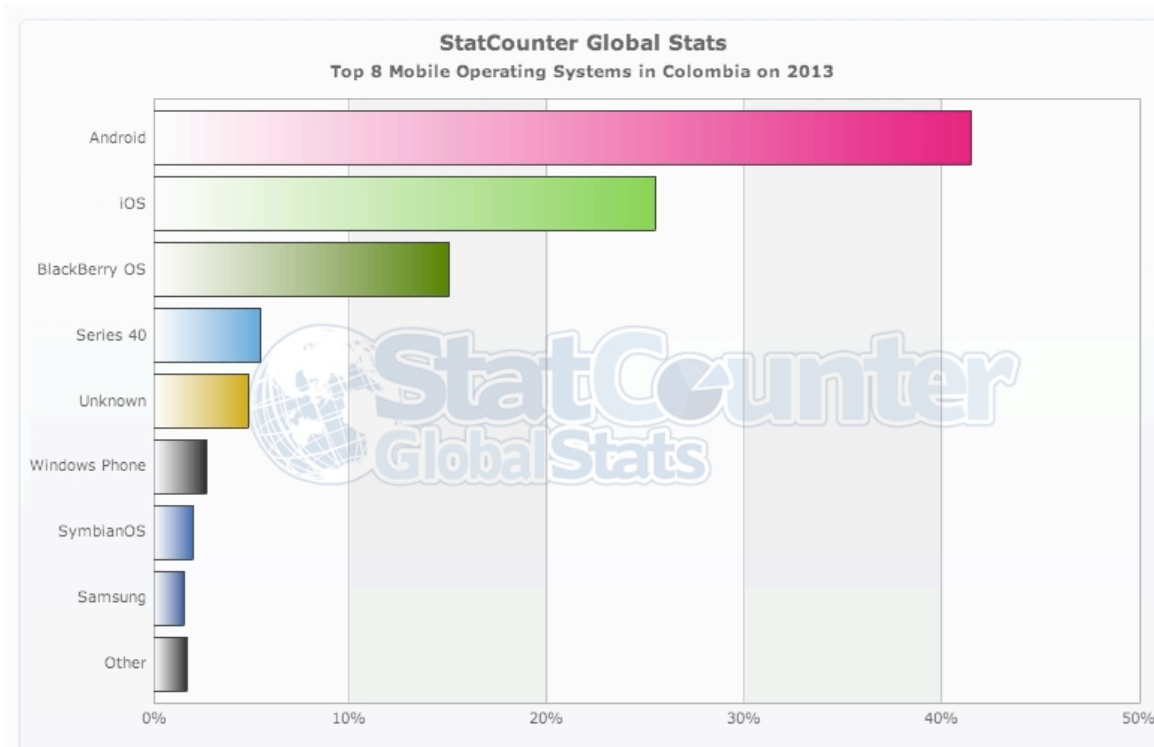


Figura 1. Estadísticas de sistemas operativos móviles en Colombia (2013), recuperado de <http://gs.statcounter.com/>

En lo que concierne a la aplicación misma, cabe resaltar que, de las más de 775.000 aplicaciones disponibles en la App Store, a la fecha existe una sola que busca el mismo objetivo que la idea propuesta y que a pesar de ofrecer una interfaz “limpia” y muy bien pensada, tiene bastante margen de mejora en aspectos como la información desplegada para cada parqueadero de la ciudad.

6. ALCANCE

La construcción de un cliente del lado del dispositivo móvil con plataforma iOS y la construcción de un proceso del lado del servidor que resolverá las peticiones de localización de parqueaderos públicos que el móvil realice para los barrios Chapinero Norte, Quinta Camacho y Emáus, ubicados hacia el oriente de la Ciudad de Bogotá. Adicionalmente, se habilitará un formulario web para ingresar nuevos parqueaderos a la base de datos de iParqueo en un marco administrativo.

7. MARCO DE REFERENCIA

7.1 Marco Teórico

7.1.1 Introducción

En la actualidad, debido al inevitable avance de la tecnología, la sociedad ha adoptado el hábito de tener todo a mano: personas, información, herramientas de trabajo, y todo aquello que podamos enmarcar como recurso para poder llevar a cabo las funciones de nuestro día a día tanto en el ámbito profesional como en el personal. De esta forma, el elemento primordial dentro del cual consignamos todo aquello que consideramos relevante para nuestro accionar, es el teléfono móvil, el cual, gracias a la fuerte evolución que ha venido adquiriendo sobre los últimos años, ha dado un vuelco enorme en nuestras vidas optimizando la forma en la que realizamos nuestras tareas a tal punto de obligarnos a replantear nuestras actividades para poder ser más productivos y competitivos dentro de la sociedad. La cantidad de posibilidades que ofrecen los móviles hoy en día son innumerables, pero no siempre fué así, es por eso que se precisa hacer un breve repaso del punto de inflexión que permitió llegar a lo que en este momento tenemos.

El terreno de la telefonía móvil celular ha sufrido un cambio drástico en su evolución, marcado por la exitosa incursión del iPhone en el mercado, el smartphone que Apple incorporó a mediados del 2007, rompiendo todo tipo de esquemas, creencias, y modelos que sus competidores presentaban sobre la fecha.

Se puede hablar de un antes y un después de este producto dada la calidad de hardware y software empleado para su concepción, ya que impulsó y obligó a toda la industria a replantear aquello que venían haciendo durante años para poder sacar productos de calidad similar y así intentar no perder su lugar en el mercado. Citando un ejemplo claro, mucho antes de que Apple sacara el iPhone al mercado, Microsoft sostuvo por varias versiones sin cambios drásticos, su sistema operativo Windows Mobile, el cual se incorporaba en equipos con pantalla táctil que en la época se denominaban “PDAs” por sus siglas en inglés Personal Digital Assistant, pero que, viéndolo desde la actualidad, parecería una broma pues se podría definir como una versión de escritorio miniaturizada que pretendía cumplir con la labor de PC de bolsillo, algo que claramente al día de hoy no cabría dentro de la industria. Es aquí en donde resalta la labor realizada por Apple, la cual sigue la premisa de entender de verdad aquello que realmente demanda el usuario, a veces sin que éste último lo sepa.

Sin lugar a dudas, el mayor pilar de Apple en la concepción de su dispositivo móvil fue, en principio, aplicar la filosofía de fabricación de su línea de ordenadores mac, es decir, la construcción de su propio hardware y software basado en su sistema operativo de escritorio OS X. El nivel de integración obtenido en su plataforma, conocida como iOS o iPhone Operative System en aquel entonces y la abolición de un teclado físico a cambio de incorporar una pantalla con tecnología multitáctil, fueron el complemento perfecto que marcó el inicio del camino del éxito para la compañía.

La simbiosis generada entre el software y el hardware del dispositivo, genera como resultado una experiencia de usuario muy superior con respecto a aquella que pueden ofrecer los dispositivos que son fabricados por una compañía y alimentados con el software de otra, pues bien decía hace unos años Carolina Milanesi (Vicepresidente de investigación en el equipo dispositivos de consumo de Gartner) en el texto de Reed (2009), que para ese entonces, tecnológicamente podían existir muchos dispositivos que en características de hardware superaban al iPhone, pero en ese momento, no había nada en el mercado que lograra la superioridad en la experiencia de usuario producto de la integración previamente mencionada; si bien fué algo que ella mencionó hace casi 4 años, su apreciación bien puede trasladarse al presente, pues a la fecha, sigue sin existir una compañía dentro del contexto móvil que logre integrar su propio hardware-software en beneficio del usuario de esta forma tan exitosa.

Particularmente, se puede decir que son 2 las compañías que, dentro del contexto de software, dominan actualmente la escena del mercado móvil: Apple y Google. Según Gartner, finalizado el 3er cuatrimestre del 2012, estas dos compañías comparten el 86.3% del mercado a nivel mundial, con Android encabezando las estadísticas.

Está claro que la cuota de mercado de Apple en este momento, es mucho menor a la de Google con su sistema operativo para móviles denominado Android. Según su web oficial, “Android es un software de código abierto creado para teléfonos móviles y otros dispositivos”. Esto quiere decir que cualquier fabricante de dispositivos móviles puede adoptar Android como su plataforma base, siendo esta una buena razón por la cual el mercado se encuentra atestado de

móviles con el sistema creado por Google. A la pregunta de por qué los fabricantes eligen Android y no otro sistema, se pueden decir varias cosas, para empezar, el gran parecido en ciertos aspectos que tiene ésta con iOS, un sistema fácil de usar y estéticamente agradable al usuario, la constante evolución que ha venido adquiriendo la plataforma adelantándose por encima de sus competidores al margen de Apple, la libertad del usuario al momento de configurar el sistema a su gusto pudiendo personalizar desde su apariencia hasta su funcionamiento y tal vez la más importante de todas, el ecosistema de Google integrado a lo largo y ancho de la plataforma ofreciendo la mejor búsqueda del mercado, quizás para muchos el mejor servicio de mensajería con Gmail, Google Docs, Google Maps y Google Earth por mencionar los más destacados.

Es por esto y más, que Android domina la escena de las plataformas móviles frente a la competencia, pues existe una gran cantidad de fabricantes que adoptan el software creado por Google como base para sus dispositivos móviles, en contraste con Apple, que incorpora iOS únicamente en los dispositivos que ellos mismos fabrican. El resto de la competencia, BlackBerry, conocida antes como Research In Motion (RIM), Microsoft y los demás fabricantes que hacen parte de esta industria, se ven opacados como consecuencia de su tardía respuesta al fenómeno iPhone, dada la difícil tarea de fabricar equipos competentes en términos de calidad y la complejidad de los distintos modelos de negocio llevados a cabo por cada una de éstas.

Ahora, si bien es importante tener en cuenta la cantidad de gadgets en el mercado discriminados por plataforma, es aún más relevante conocer cuál de éstas genera más beneficios

económicos para los desarrolladores, sus características técnicas y destacar los avances en materia de la tecnología que traen los móviles hoy día, para obtener un contexto mucho más completo del mercado que se pretende atacar.

Con la masificación de distintas tecnologías que vienen incorporándose en estos equipos tales como acelerómetros, giroscopios y GPS entre otros, las posibilidades que se tienen al momento de fabricar una aplicación se expanden permitiéndo dar vida a productos de altísima calidad que se pueden implementar de forma práctica en el día a día del usuario. La georeferenciación es una tecnología cada vez mejor optimizada y masificada tanto por los fabricantes de equipos como por las compañías que proveen los servicios que permiten explotar las bondades de ésta. Su aprovechamiento en las aplicaciones móviles ha generado resultados interesantes, y día a día, siguen apareciendo áreas en las cuales ésta encaja a la perfección.

El uso de esta tecnología en una aplicación de iOS aplicada a un problema en particular, es el propósito del presente estudio, mediante el cual se pretende dar un vistazo a la creación de apps y entregar información de las distintas aproximaciones que se pueden tomar para conseguir el objetivo.

7.1.2 Desarrollo de aplicaciones móviles

Dentro del campo de desarrollo de aplicaciones para móviles, confluyen algunos puntos importantes a tener en cuenta con respecto a un desarrollo tradicional de software. Para empezar, el tiempo de desarrollo de una aplicación móvil es bastante menor, en promedio, que el de un producto que no está siendo elaborado para tal, la filosofía con la que se abordan estos tipos de implementaciones debe estar basada en la practicidad y la sencillez para ofrecer la mejor experiencia de usuario posible, algo que desde luego, también se puede implementar en una aplicación de escritorio claro está, pero el punto clave dado un dispositivo de pequeñas dimensiones, es abstraer de la mayor complejidad posible al usuario, evitando entregar un producto con demasiados controles para cumplir n funciones y en contraparte, procurar que la aplicación haga una tarea concreta de forma eficiente y con la menor cantidad de intervenciones posibles.

Ahora, dado el auge que han generado los principales competidores dentro de esta materia, refirámonos a ellos citando plataformas (iOS, Android, Blackberry, Windows Phone) es de vital importancia mantener activa la atención del usuario, es decir, la vida de una aplicación no concluye con su publicación, esto es apenas el principio. La elaboración de actualizaciones con cierta regularidad, que además de subsanar bugs y temas de estabilidad del producto amplíen el funcionamiento del mismo entregándole al usuario más y mejores formas de aprovechar su dispositivo, es un punto neurálgico si se quiere trascender y tener éxito en este campo de acción. La competencia en este sector se hace cada vez más fuerte y, dadas las facilidades de acceso que plantea este “mundo móvil” se hace muy sencillo flaquear en un esquema en el que la copia de

ideas, es el pan de cada día. Por esta y muchas otras razones, es necesario ser cautos con el desarrollo que se lleve a cabo y esforzarse por buscar mecanismos que capten y mantengan la atención de los potenciales usuarios.

Existen unos lineamientos básicos para iniciar con el desarrollo de una aplicación en iOS, con el fin de orientar un desarrollo acorde a la calidad de la plataforma y de proveer estrategias que permitan alcanzar el objetivo propuesto; éstos, bajo una descripción generalizada, pasan por tener claro cual es el propósito principal de la aplicación a desarrollar y a que tipo de público se pretende llegar con esta, acto seguido, se puede realizar un listado de las características que caben dentro del objeto que se plantea tenga la aplicación y por último, asegurarse de ofrecer una experiencia de usuario agradable aprovechando correctamente las características de los dispositivos sobre los cuales se genere el desarrollo.

Superada esta etapa, se hace necesario elegir las herramientas de trabajo para iniciar con la construcción de la aplicación. Dichas herramientas definirán el camino a tomar que marca la diferencia entre un desarrollo nativo, o uno no nativo.

Desde el punto de vista comercial, será determinante identificar cuál es la plataforma cuya tienda de aplicaciones genera mejores beneficios económicos para el desarrollador, en el caso de iOS, se conoce que el porcentaje de ganancia del desarrollador es del 70% quedándose Apple con el 30% restante para costear temas de mantenimiento y hosting de la aplicación entre otros.

7.1.3 MinTIC

El Ministerio de Tecnologías de la Información y las Comunicaciones viene apostándole de forma fuerte a la formación de emprendedores en la construcción de aplicaciones móviles, con el fin de generar oportunidades de negocio y hacer realidad las buenas ideas que en muchas ocasiones se ven opacadas, en gran parte, por la falta de conocimiento técnico y estratégico entre otros aspectos. Es así como Apps.co ofrece de forma muy completa, una oportunidad real de materializar una idea de negocio relacionada con el empleo de las tecnologías de la información y las comunicaciones, en cualquier etapa que ésta se encuentre.

Básicamente, la iniciativa consta de 3 fases que buscan cubrir los distintos contextos dentro de los cuales puede estar ubicado el emprendedor: Ideación, Aceleración y Consolidación.

La primera fase, es ideal para aquellas personas que tienen una buena idea de negocio, pero carecen de las distintas habilidades necesarias para poderle dar vida al proyecto. Es así como los emprendedores que se acomodan a estas características, reciben formación en tecnología, negocios y mercadeo, para sentar las bases de su proyecto de forma integral.

En la fase de aceleración, se ubican aquellas empresas relativamente jóvenes (constitución mayor o igual a un año) en adelante, cuyo potencial es interesante y tienen perspectivas de crecimiento a corto plazo. El acompañamiento aquí va enfocado a un entorno

más organizacional con el fin de solidificar la estructura de la compañía y fortalecerla en cada una de sus áreas.

Para la última fase, entran aquellos emprendedores cuyo modelo de negocio se encuentra validado y buscan consolidarlo para traducirlo en un negocio rentable. La ayuda brindada en este frente se fundamenta en impulsar la parte técnica y de negocio.

De igual forma, existe una etapa básica de aprendizaje en la cual los interesados en entrar en éste campo de acción, pueden formarse en el aspecto técnico, aprendiendo sobre las distintas plataformas y los lenguajes de programación requeridos para desarrollar aplicaciones móviles.

Las convocatorias realizadas por Apps.co se realizan de forma regular, en lapsos de 2 meses para cada fase, repitiéndose éstas en un tiempo estimado de 4 meses aproximadamente.

7.1.4 Desarrollo Nativo

El panorama actual de desarrollo de aplicaciones para móviles, abarca un entorno nativo y otro no nativo. Centrándonos en la plataforma iOS entendemos por desarrollo nativo, que las herramientas empleadas para tal fin son proporcionadas directamente por el fabricante de la plataforma (en este caso Apple), garantizando desde un punto de vista técnico, la construcción de aplicaciones de altísima calidad, con un desempeño superior y un acceso no restringido a los componentes de hardware del dispositivo sobre el cual se esté implementando la solución, gracias a las interfaces de programación suministradas (APIs).

Para la plataforma *iOS*, Apple entrega a los desarrolladores un conjunto de herramientas muy interesantes *SDK* que traduce Software Development Kit, (ver Figura 2) para materializar sus ideas en aplicaciones tan complejas (en términos de robustez) como las que se pueden encontrar en programas de escritorio. Estas aplicaciones, luego de pasar por un proceso previo de aprobación por parte de la compañía de la manzana, en el cual verifican que la app se ha construido respetando las políticas de Apple tanto a nivel técnico como a nivel comercial, se cargan a su tienda virtual de aplicaciones denominada “App Store” que demanda sencillez, usabilidad y un gran catálogo de productos de diversas categorías para todos sus dispositivos móviles.

Según Michael Davis la razón principal para desarrollar una aplicación de forma nativa, debe ser la necesidad de acceso específico a la funcionalidad que ofrece el dispositivo, como por

ejemplo acceder al acelerómetro, la cámara o el GPS, como también, menciona, la necesidad que puede tener la aplicación para beneficiarse de la integración con otras aplicaciones nativas entre otros aspectos (Davis, 2011).



Figura 2. Conocimiento y herramientas para el desarrollo de apps nativas (Adaptada por el autor)

Siguiendo por la línea de los argumentos o criterios para elegir el camino nativo, según el artículo “Web Vs. Native Development: There's No Winner” publicado en la revista InformationWeek por un autor anónimo, (Anonymous, 2011) las aplicaciones nativas se ejecutan más rápido en muchos escenarios y por obvias razones, tienen el privilegio de ser las primeras en aprovechar las ventajas de las mejoras introducidas en las actualizaciones realizadas a las plataformas.

El artículo también menciona que las herramientas de desarrollo empleadas para optimizar el rendimiento de los dispositivos móviles en entornos no nativos, siempre están un paso atrás de las herramientas nativas imposibilitando alcanzar de forma sencilla el desempeño de las que son concebidas con la ayuda de éstas últimas. Ahora, un punto muy importante dada la alta demanda de este sector en estos momentos es, como bien lo dice el artículo en una comparación con

Google, que la comercialización de aplicaciones nativas a través de la App Store simplemente funciona, algo fácilmente comprobable a través de las estadísticas que prestan diversos portales web y que de acuerdo a Distimo, para Abril de 2013, la rentabilidad de la App Store a nivel mundial llega al 73% por encima de su más directa rival, Google Play, que ronda el 27% del mercado. (Distimo, 2013).

A pesar de los magníficos productos de software que se pueden llevar a cabo haciendo uso del enfoque nativo, existen razones de peso para considerar otras aproximaciones cuando se trata de desarrollar para iOS, entre ellas, las condiciones que conlleva, como el hecho de que se debe pagar una licencia anual que a la fecha cuesta USD\$99 la cual otorga acceso a documentación técnica, la posibilidad de probar la aplicación en un dispositivo real y como último, permite publicar la aplicación en la tienda de aplicaciones “App Store” facilitando su distribución. Siguiendo por la línea de condiciones que plantea un enfoque nativo, la aplicación que se construya está sujeta a un proceso de aprobación en la compañía de la manzana, se debe utilizar un único lenguaje de programación (Objective-C) y entre otras, el desarrollo debe hacerse en una mac, pues el SDK sólo se encuentra disponible para el sistema operativo de ésta, OS X. (Stark, 2010)

Por otro lado, encontramos la dificultad de los desarrolladores para adaptarse al lenguaje de programación empleado en su plataforma móvil, Objective-C, ya que a pesar de que su paradigma de programación orientado a objetos se encuentra ampliamente difundido y aplicado, entre muchas otras razones, su sintaxis no se asemeja a los lenguajes habituales o populares

sobre los cuales se encuentran enfocados la gran mayoría de profesionales dedicados a este oficio y que en gran parte carecen de experiencia desarrollando para OS X.

Dada esta situación, la curva de aprendizaje de esta alternativa no es apropiada para aquellos que buscan desplegar una aplicación en contra del tiempo, y es éste precisamente uno de los problemas que rodean este panorama, pues aquellos que abordan la construcción de una app con cero experiencia en este enfoque, terminan desaprovechando las herramientas suministradas y obteniendo como resultado apps de bajísima calidad, llenas de agujeros, de inconsistencias y de bajo rendimiento que, pueden desencadenar en 2 posibles escenarios, la no publicación en la App Store por no cumplir con las condiciones técnicas de desarrollo o tener suerte a la hora de ser verificada, pasar satisfactoriamente el proceso y ser publicada llegando un público que muy seguramente la calificará de forma negativa dañando la reputación del desarrollador y haciendo más difícil el camino a figurar de forma positiva en medio de la competencia.

Ahora, retomando un poco del proceso de verificación y pruebas llevado a cabo para mantener un estándar de alta calidad en el catálogo de la tienda de aplicaciones, hay que decir que éste suele prolongar el proceso de publicación, razón por la cual, es altamente recomendable depurar al máximo la solución y realizar todas las pruebas en los dispositivos bajo distintas condiciones con el fin de que una vez Apple apruebe y publique la app, el desarrollador o equipo involucrado se evite dolores de cabeza teniendo que verificar y corregir una aplicación con errores que en el momento, se encuentra disponible al público y que mientras se trabaja en una

actualización para remitirla de nuevo a la tienda, pierde adeptos y le facilita el camino a la competencia.

Ahora, si vamos un poco más allá de las restricciones que implica esta aproximación, existe la desventaja de que una vez finalizado el producto, no es posible migrarlo a otras plataformas si existiera la necesidad de expandirse hacia otros mercados más allá de la App Store. De esta forma, si por citar un ejemplo, se pretende ingresar a Google Play, la tienda de aplicaciones de la plataforma Android, el software debe escribirse desde cero haciendo uso de las herramientas que Google facilite para su desarrollo. Teniendo en cuenta lo anterior, es fundamental la definición del camino a elegir para el desarrollo, pasando por un proceso previo de diseño de la app a materializar, para así ir obteniendo claridad sobre el producto que se pretende lanzar y actuar de forma apropiada basándose en las condiciones bajo las cuales esté pensada la aplicación.

7.1.5 Desarrollo No Nativo

El enfoque no nativo comprende dos alternativas; por un lado están las APIs generadas por terceros que buscan cubrir la mayor cantidad de funcionalidades de la plataforma, pero que dada su concepción sobre otras tecnologías de ámbito general, es decir, no enfocadas ni relacionadas estrechamente con la plataforma objeto, carecen de ciertas funciones que si están soportadas por el enfoque nativo, tales como temas de acceso a elementos característicos de cada dispositivo.

La segunda alternativa es una de las aproximaciones más habituales dentro de este campo: Las Web apps, aplicaciones creadas a partir de tecnologías web transversales a las plataformas que dominan la escena del software para móviles. Es así que con HTML, CSS y Javascript es posible construir una aplicación que trabaje en múltiples sistemas operativos móviles (ver Figura 3). Uno de los puntos más interesantes de este enfoque no nativo, es la posibilidad de migrar su código a cualquier plataforma convirtiéndose la aplicación en nativa por medio de frameworks como PhoneGap.

Se puede hablar de PhoneGap en si, como un framework que además de permitir generar aplicaciones con las herramientas anteriormente seleccionadas actúa como puente entre las web apps y los dispositivos móviles. (Stark, 2010). En el caso particular del objeto de estudio (iOS) PhoneGap permitirá acceder a las características nativas del dispositivo móvil y también proveerá los medios necesarios para publicar la aplicación en la tienda de aplicaciones de Apple. De esta forma, el producto se escribirá una sola vez y podrá ser migrado a otros sistemas como Android, Windows Phone o Blackberry por citar algunas de las posibilidades, evitando tener que ser codificado de cero para cada plataforma, situación a la que obliga un desarrollo nativo.

Elegir este camino también tiene sus ventajas. Hace poco tiempo, Microsoft dejaba en claro que apostaba con bastante fuerza al desarrollo en HTML 5 para Windows 8, mientras que Steve Jobs, en su conocida disputa con la tecnología “flash” de Adobe, proclamaba que los nuevos estándares abiertos creados en la era móvil, (refiriéndose a HTML5), vencerían en los dispositivos móviles” (Anonymous, 2011).

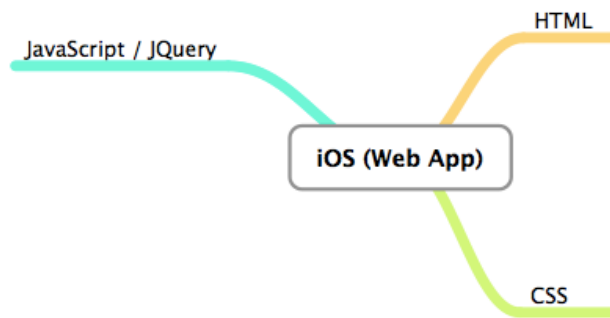


Figura 3. Conocimiento base para el desarrollo de una Web App
(Adaptada por el autor)

Continuando con las bondades de esta elección, como bien acotan en el artículo (Anonymous, 2011) se tiene el hecho de que actualmente, la guerra de patentes en el entorno nativo es de dimensiones exorbitantes, pues de acuerdo al EFF (Electronic Frontier Foundation), un desarrollador puede recibir hasta cientos de reclamaciones por patentes y agregan que, aunque el panorama Web no se encuentra totalmente ajeno a este tipo de situaciones, el W3C (World Wide Web Consortium), organismo que desarrolla y supervisa los estándares Web, posee acuerdos con más de 50 compañías que hacen parte del HTML Working Group en donde se establece que su tecnología se encuentra disponible sin derechos de autor, y en caso de que se lleguen a presentar reclamaciones, el W3C interviene de forma activa recopilando evidencia para invalidar dichas solicitudes.

Pero tal vez uno de los puntos a favor más fuertes que encierra este tipo de desarrollo, es el hecho que plantea (Davis, 2011), citando una frase en particular promulgada por los proveedores de APIs para desarrollo no nativo “Un código fuente, todo teléfono inteligente”, pues una vez se adquiere una de estas soluciones para crear aplicaciones de forma veloz y técnicamente de

manera más sencilla, ésta puede ser migrada a casi cualquier sistema operativo móvil. Para dar un poco más de claridad sobre esa facilidad “extra” concerniente a la parte técnica, es apropiado mencionar que las tecnologías empleadas para este fin, son mucho más populares que las concernientes a un solo sistema en concreto, por lo cual su implementación se hace más sencilla dada la rápida adaptación por parte de los desarrolladores a la misma, y por ende, el despliegue de la aplicación se consigue en un plazo más corto.

Tan o más importante que el punto anteriormente mencionado, es la reusabilidad de módulos de fuentes fiables, algo que permite la aceleración de un proyecto e incrementa su confiabilidad. Sin embargo, debe verificarse con detenimiento la calidad del código implementado con el fin de garantizar la seguridad en la integración del módulo a la solución (Davis, 2011).

Bajo cualquier implementación no nativa que se emplee con el fin de generar aplicaciones para iOS, se ha de saber que en caso de no ser migrada para poder ser publicada en la App Store, su comercialización puede ser un tema bastante complejo en términos de visibilidad de la aplicación y en el esfuerzo que supone poner ésta a disposición del público por medios externos a la App Store teniendo que hacer uso de un sistema de pago de un tercero como Paypal.

A lo largo del desarrollo de una app pueden aparecer tanto obstáculos como alicientes, el tema es que se debe estar preparado para hacer frente a estas situaciones y en lo posible anticiparlas, pero lo más importante es haber elegido la ruta correcta para abordar la solución al problema o necesidad planteada y así enfocar los esfuerzos a seleccionar, dentro de una serie de

tecnologías comunes a la alternativa escogida, cuál se adapta de mejor forma a lo que se pretende en la implementación de una app.

La aproximación que se va a adoptar en la elaboración del prototipo para iOS será ésta última, las web apps junto con PhoneGap, obteniendo así lo mejor de dos mundos, las bien conocidas tecnologías web en constante evolución y de relativo fácil aprendizaje, y el framework más conocido y estable actualmente empleado para migrar este tipo de aplicaciones. La idea entonces es escribir una sola aplicación personalizada para el iPhone y el iPad, con la ventaja de poder utilizar el mismo código fuente y por medio de pequeños ajustes, ampliar los horizontes de la misma y lograr posteriormente su distribución en otros ecosistemas sin necesidad de reescribirla ni de pasar por un denso proceso de aprendizaje de otros lenguajes de programación específicos para lograrlo.

En resumen, entre las ventajas más destacadas que supone elegir esta alternativa, están: la posibilidad de trabajar con tecnologías ampliamente conocidas, cero limitaciones a la hora de elegir la máquina desde la cual se desea trabajar, la ejecución en un contexto multiplataforma, pues éstas corren sobre el navegador del dispositivo que las ejecute, la corrección de errores en tiempo real y un ciclo de desarrollo veloz. (Stark, 2010)

Para hacer posible una experiencia de usuario agradable, se procederá a hacer uso de plugins o librerías de Javascript que permitirán simplificar el trabajo en la parte visual de la aplicación,

garantizando animaciones y componentes gráficos acordes a la calidad de la plataforma y permitiéndolo centrar los esfuerzos en la lógica de la misma.

Entrando en contexto, y dadas las circunstancias anteriormente mencionadas, es indispensable tener claro el alcance que va a tener la aplicación a desarrollar, pues una mala decisión a la hora de abordar este tema, puede generar consecuencias nefastas en el proceso de construcción del proyecto tanto así como para tener que abortarlo e iniciar su desarrollo nuevamente desde cero. El primer paso se basa en la selección del ciclo de vida de la aplicación.

7.1.6 Ciclo de vida

El ciclo de vida, en términos de la ingeniería del software, de acuerdo a INTECO (2009) se define como “el conjunto de fases por las que pasa el sistema que se está desarrollando desde que nace la idea inicial hasta que el software es retirado o reemplazado”.(INTECO, 2009). De acuerdo a la necesidad que plantea la construcción de un software en particular, existen los llamados modelos de ciclo de vida, cuya variedad, radica en las condiciones de alcance, contenido y estructura del proyecto. Es así, como un modelo de ciclo de vida debe contener las siguientes etapas para una correcta implementación según (INTECO 2009):

- ▶ Describir las fases principales de desarrollo de software.
- ▶ Definir las fases primarias esperadas de ser ejecutadas durante esas fases.
- ▶ Ayudar a administrar el progreso del desarrollo.
- ▶ Proveer un espacio de trabajo para la definición de un proceso detallado de desarrollo de software.

Teniendo en cuenta que las aplicaciones móviles están sujetas a un entorno cambiante, tanto por optimización como por actualización y que las metodologías empleadas en su mayoría buscan generar un producto de forma rápida, los modelos de ciclo de vida que mejor se adaptan a éstas se describen a continuación. (ver Tabla 1)

Modelo	Definición	Características
Modelo Iterativo	Se basa en la iteración de varios ciclos de vida en cascada	<ul style="list-style-type: none"> • Los requerimientos pueden no estar completamente claros, lo cual también puede jugar en contra del desarrollo en términos de la arquitectura de la aplicación • Cada iteración comprende un ciclo en cascada completo, lo cual da lugar a varias versiones • Cada versión es evaluada por el cliente el cual define los cambios a realizar para la próxima iteración hasta llegar a una versión final
Modelo de desarrollo incremental	Fusiona elementos del modelo en cascada con la intención de construir prototipos que incrementan su funcionalidad a través del tiempo	<ul style="list-style-type: none"> • Parte de lo simple a lo complejo • Cada prototipo es un producto usable con cierta funcionalidad de tendencia incremental • Por su filosofía, permite obtener entregas tempranas con funcionalidad parcial pero operativas • Simplifica la complejidad en la elaboración de pruebas • La experiencia juega un rol importante en el éxito o fracaso de esta implementación
Modelo por prototipos	Se construyen prototipos funcionales completos basados en una continua retroalimentación con el cliente	<ul style="list-style-type: none"> • Es ideal para ser usado en casos de incertidumbre técnica, o de la idea de negocio • Permite una rápida entrega de versiones completas preliminares del producto final • Responde de forma positiva a cambios de los requerimientos • En abuso, puede desencadenar en todo lo contrario a un desarrollo rápido • Los costes se pueden incrementar de acuerdo a la cantidad de prototipos desechados

Tabla 1. Descripción de ciclos de vida empleados en el desarrollo de aplicaciones móviles (Adaptada por el autor)

7.1.7 PhoneGap

PhoneGap se trata, de acuerdo a (Trice, 2012) de una tecnología contenedora de aplicaciones que permite la creación de éstas de forma nativa haciendo uso de HTML, CSS y Javascript. Se puede interpretar esta definición de la siguiente forma: PhoneGap básicamente consiste en un navegador simplificado que contiene todo el código no nativo de nuestra aplicación y, adicionalmente, provee una interfaz de programación de aplicaciones que a través de Javascript, un lenguaje de programación interpretado, es decir, leído línea a línea por el navegador, permite acceder a las características nativas del sistema operativo móvil. Ahora, la verdadera magia de PhoneGap radica en que a pesar de que la aplicación se haya desarrollado con las tecnologías web previamente mencionadas, el producto final es un archivo de aplicación binario que se puede exportar a cualquiera de los sistemas operativos móviles actualmente soportados. A la fecha tales sistemas son: Android, Blackberry, iOS, Symbian, WebOS, Windows Phone 7, Windows Phone 8, Windows 8, Bada y Tizen de acuerdo a la documentación oficial del proyecto.

De esta forma se obtienen los mismos ficheros generados por desarrollos con un enfoque nativo, y por ende éstos archivos pueden ser distribuidos a su correspondiente tienda de aplicaciones de acuerdo al ecosistema elegido. Dentro de un contexto un poco más técnico, cabe mencionar la arquitectura con la que PhoneGap trabaja a nivel general y específico a iOS.

Según (Trice, 2012) la arquitectura que se recomienda utilizar del lado del cliente, es conocida como SPI (Single Page Interface) la cual se fundamenta en incluir todo el contenido de la aplicación en una sola página, de tal forma que todos los archivos necesarios para el funcionamiento de la misma, se carguen en el inicio, evitando requerir archivos adicionales durante las intervenciones que realice el usuario, permitiendo una experiencia de usuario más fluida y aliviando la innecesaria transmisión de datos al servidor, entre las características más destacadas. (Wikipedia, 2013a). A continuación se describe, en términos generales, la forma en la que PhoneGap sirve de puente entre el sistema operativo móvil y una Web App.

Claramente se aprecia que PhoneGap actúa como contenedor de la Web App dentro del cual encontramos 3 conjuntos de componentes: lógica del negocio (Web App), archivos javascript que permiten el acceso a las características de hardware del dispositivo (PhoneGap Plug-ins) y un motor de renderización HTML (navegador web simplificado) que permite exponer la Web App al usuario como si de una aplicación nativa se tratara.

Estos componentes identificados en color verde y cuya comunicación se muestra de forma bidireccional a través de API's (flechas blancas) que ejercen de puentes permitiéndolo el acceso a los controladores nativos como el WebView y las características de hardware del dispositivo móvil como el acelerómetro, el GPS o la cámara, encierran el objetivo de adaptarse a la mayoría de plataformas móviles disponibles en el mercado, siendo accedidas por las propias librerías de cada uno de los distintos ecosistemas. (Figura 4).

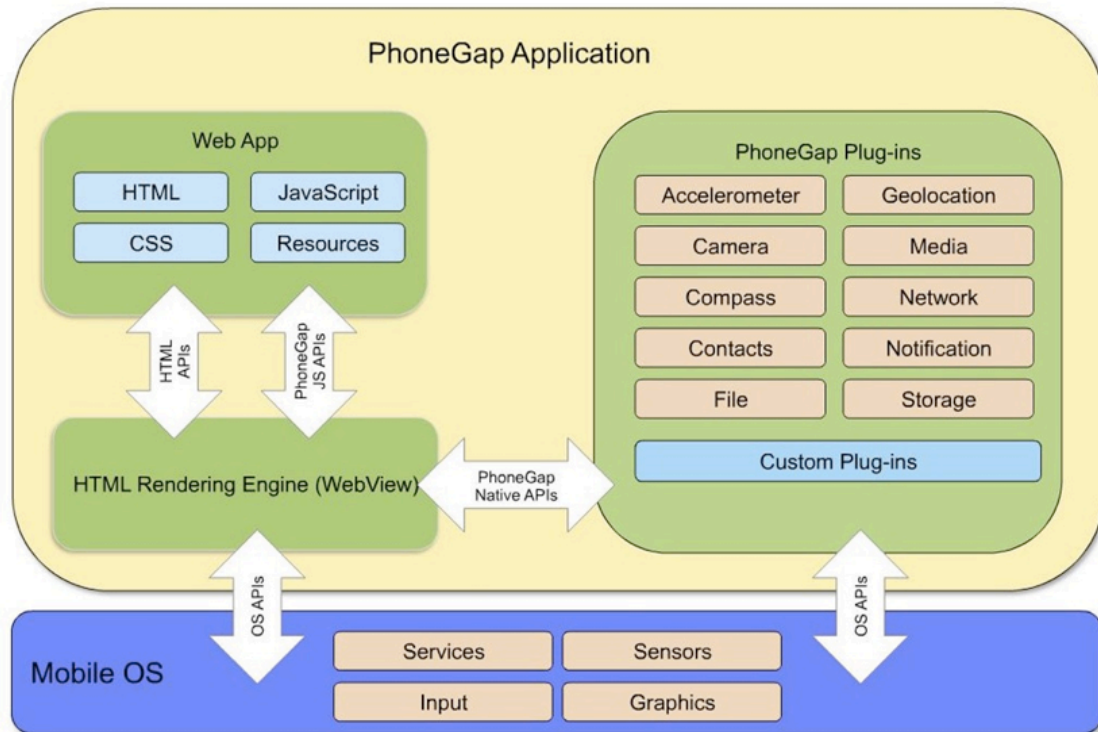


Figura 4. Arquitectura PhoneGap (2011), tomada de <http://html5hu.wordpress.com/2011/10/05/a-mobil-fejlesztis-leghatkonyabb-html5javascript-ben/>

Para el caso de estudio en particular, encontramos que para iOS existen 3 puentes que vinculan y permiten el intercambio de información entre la web app y dicha plataforma. (Figura 5)

De acuerdo a la interpretación de (Puthraya, 2012), la comunicación ejercida entre el controlador WebView y la parte nativa se realiza a través de un iFrame. Un iFrame es un elemento (es decir una etiqueta HTML) que permite insertar un fichero HTML dentro de otro, siendo éste último el principal, o el que recibe los documentos que la aplicación requiera de acuerdo al uso de la app.

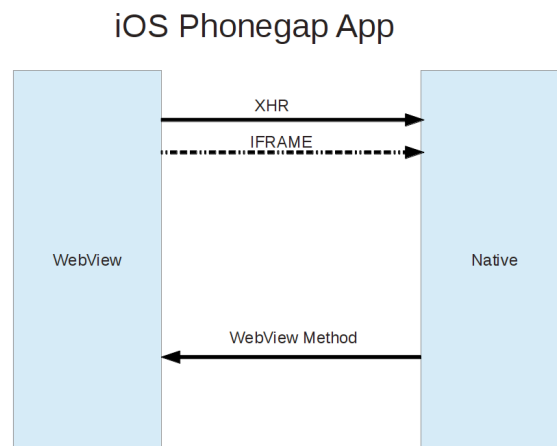


Figura 5. Arquitectura PhoneGap aplicada a la plataforma iOS (2012), tomada de <http://agiliq.com/blog/2012/09/dissecting-phonegaps-architecture/>

Por otro lado, el puente XHR (XMLHttpRequest), menciona Puthraya, se encarga de realizar llamados hacia una URL falsa; éstos llamados se realizan por medio de un lenguaje de scripting como lo es javascript, y son almacenados en una cola (estructura de datos), que a su vez es leída y ejecutada por el componente nativo cuyos comandos se localizan en la cabecera de la petición; éstos comandos son interceptados, serializados y ejecutados. Para el puente que permite comunicar la parte nativa que contiene la Web App con el controlador encargado de mostrar en pantalla la app, el WebView, la comunicación es generada de forma nativa usando Objective-C, por medio de un método que se conoce como *stringByEvaluatingJavascriptFromString* de tipo **UIWebView**, es decir, la clase dentro de la cual se accede al mismo.

7.1.8 Metodologías

Existe una cantidad considerable de metodologías a emplear para llevar a cabo un desarrollo de software; como bien menciona (Rodríguez, 2011) existen ciertas peculiaridades en torno a la creación de aplicaciones móviles en contraste con las soluciones que se implementan fuera de esta industria, como por ejemplo los cortos tiempos de desarrollo, la velocidad de cambio en las plataformas y la diversidad de hardware entre otros aspectos, que son pieza clave para proceder a elegir la metodología apropiada en la construcción de una app. Dentro de éstas, existe un grupo que se acopla mejor que el resto al contexto de aplicaciones para dispositivos móviles, y se conoce como desarrollo ágil de software, pues ha sido concebido según (Darwish, 2011) pensando en la gente, es decir, enfocado a la comunicación, con la flexibilidad necesaria para acoplarse a los cambios que se presenten en cualquier momento y permitiéndolo la mejora constante durante y después del proceso de construcción del software.

El desarrollo ágil de software, de acuerdo al artículo de (Darwish, 2011) es un enfoque iterativo e incremental, que se ejecuta de forma altamente colaborativa, para producir software de alta calidad que a su vez se acopla a las necesidades cambiantes de las partes interesadas. Dentro de las metodologías pertenecientes al desarrollo ágil, se encuentran las siguientes: XP (Extreme Programming), Scrum, Crystal, Feature Driven Development (FDD), Dynamic System Development Methodology (DSDM) y Adaptive Software Development (ASD) (Darwish, 2011). Para el objeto de estudio, se empleará XP bajo la siguiente premisa:

“Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto y aplicarlo de manera dinámica durante el ciclo de vida del software.” (INTECO, 2009).

El proceso de XP está caracterizado por ciclos de desarrollo cortos, planeación incremental, retroalimentación continua y dependencia en la comunicación y diseño evolutivo. Está diseñado para ser usado en pequeños grupos de trabajo que requieren desarrollar software de forma rápida y dentro de un entorno muy cambiante en cuanto a requerimientos se refiere. (Darwish, 2011). La implementación de esta metodología consta de 4 actividades primarias en su desarrollo; Planeación, Diseño, Codificación y Pruebas. (Roger, 2005)

7.1.8.1 Planeación

Se centra en conversaciones entabladas con el usuario, que hacen las veces de lo que en otras metodologías se conoce como levantamiento de requerimientos. Cada uno de estos relatos debe ser breve (25-50 palabras) aclarando aquello que debe ser llevado a cabo en el proyecto, dándole a éste el enfoque necesario y permitiéndole al equipo de desarrollo fijar los objetivos y sentar las bases de las métricas de estimación.

7.1.8.2 Diseño

El mantra principal del diseño en XP es “mantenlo simple”, y dado que un proyecto bajo ésta metodología está basado en iteraciones a lo largo de todo su ciclo de vida, es importante manejar su complejidad diseñando y desplegando en principio, sólo aquello que es estrictamente necesario para dar luz verde al proyecto y de esta forma, diferir toda aquella dificultad adicional en iteraciones posteriores. En la mayoría de proyectos de programación, el equipo humano se enfrenta a situaciones con las que no se encuentra completamente familiarizado convirtiéndose estos en verdaderos desafíos. Es por esto que la metodología XP dispone de una estrategia que ataca directamente esta situación. Dicha estrategia se entiende como una “prueba concepto” en la que una parte del equipo de desarrolladores aborda la situación conflictiva y la restante mantiene la totalidad del proyecto en curso.

7.1.8.3 Codificación

XP contiene buenas prácticas para escribir código, muchas de ellas predecibles como por ejemplo, lograr antes que nada la funcionalidad de la aplicación y posteriormente centrarse en su optimización. Quizás la tarea más ‘extrema’ y controvertida en esta fase, es la que propone Beck, (Creador de esta metodología) al sugerir que la programación sea llevada a cabo al mando de dos programadores por cada terminal, ejerciendo uno de “piloto” y el otro de “navegador” e intercambiando roles, algo con lo que muchos no están de acuerdo. Se deben realizar la

construcción de pruebas como parte integral del proyecto y su correspondiente aplicación en cada ciclo de iteración a medida que aumenta la funcionalidad dentro del proyecto.

7.1.8.4 Pruebas

XP exige que todo el código generado tenga sus unidades de prueba y las supere. Esta fase no se puede obviar ni reemplazar de ninguna forma. Sin duda, el obstáculo más grande que se vislumbra en el horizonte es el hecho de que las pruebas tienen un tiempo límite para ser implementadas. Dada la regularidad con la que se deben ejecutar éstas, es recomendable ver esta fase como un proceso monótono que no conlleva problema alguno para facilitar la adaptación a ellas. La automatización de pruebas implementadas a lo largo de la vida del proyecto, son la mejor métrica para encontrar y eliminar errores en el sistema. Una frase que justifica la exigencia de esta fase dentro de esta metodología reza: “Entre más difícil de escribir sea la prueba, mayor es la necesidad de implementarla, pues mejores beneficios traerá”.

7.1.9 Análisis de Requerimientos

Dentro de la metodología XP, el levantamiento de información o la etapa en la que se establecen los requerimientos de usuario (funcionales y no funcionales) se realiza por medio de las historias de usuario: Según (INTECO, 2009), se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan

implementarlas en el correr de unas semanas. Las historias de usuario se descomponen en tareas de programación y se asignan a los programadores para ser implementadas durante una iteración.

7.1.9.1 Requerimientos Funcionales y No Funcionales

Los requerimientos funcionales según (Westfall, 2005) se basan en aquello que el software debe **hacer** para agregar valor a sus stakeholders y así definir las capacidades del producto. De la misma manera, los requerimientos no-funcionales toman como parámetro aquello que el software debe **ser**, definiendo características y propiedades, que en general, identifican al software en términos de su funcionamiento.

7.1.10 Interfaz Humana

Desde la irrupción de las pantallas táctiles en los dispositivos móviles, las interfaces humanas han dado un vuelco en cuanto a la aproximación que se tenía para construirlas; Los primeros “teléfonos inteligentes” contaban con un teclado físico, que si bien variaba en su concepción de acuerdo al fabricante, la gran mayoría mostraba una distribución bastante extensa de botones que pretendía emular un teclado de escritorio miniaturizándolo para encajar en un dispositivo móvil.

Si bien por aquellos tiempos, la demanda y calidad de aplicaciones era ridícula (en minoría) comparada con la actualidad, existían bastantes limitaciones en cuanto al control que los

desarrolladores podían ofrecer al usuario para interactuar con su aplicación, pues estaban restringidos a hacer uso de las forma de interacción que la industria presentaba en el momento.

Las interfaces táctiles ofrecen un escenario completamente distinto tanto para productores como para consumidores, pues, los equipos de desarrollo tienen la ventaja de generar su propia interfaz de tal forma que la integración con la lógica de la aplicación sea completa. Es así como, ahora, el desarrollador no se encuentra sujeto a las características físicas de un equipo para desarrollar su interfaz, al margen del correcto aprovechamiento de sus dimensiones.

De esta manera, y siguiendo los principios de la interfaz humana propuestos por Apple para iOS, la clave de una interfaz de usuario exitosa, se basa en centrar la atención en la forma en que los usuarios piensan y desempeñan sus labores y construir funcionalidad alrededor del negocio. Otro de los puntos claves para lograr este cometido, radica en que la apariencia de la aplicación debe dejar claro el propósito funcional de la misma, algo que se conoce como integridad estética. La consistencia, concepto que se refiere a la homogeneidad de los diseños que se rigen de acuerdo a los estándares y los patrones de la plataforma, para que así, los usuarios tengan la capacidad de familiarizarse fácilmente con las distintas aplicaciones, es una más de las recomendaciones que realiza la compañía cuando de diseñar la interfaz humana se trata.

Finalmente, se deben explotar las posibilidades que ofrece la pantalla multitáctil transmitiéndole al usuario la sensación de dominio sobre la app, y no de ésta sobre el usuario,

apoyándose en la interacción con objetos por ejemplo, que obedezcan de forma adecuada e inmediata a los gestos que el usuario realice. (Apple, 2012c)

Entrando en materia, los dispositivos con iOS, comprenden varios puntos a evaluar y a destacar de acuerdo a las ventajas que presentan con respecto a la competencia. Para empezar, las dimensiones estándar de la pantalla del iPod Touch y del iPhone en los modelos previos a la versión 5 (2g, 3g, 3gs, 4, 4s) siendo éstas de 3,5 pulgadas, y la compatibilidad que ofrece Apple con el modelo actual de 4 pulgadas, supone un alivio al no tener que considerar la infinidad de dimensiones existentes en los teléfonos de otros fabricantes, caso puntual: Android, que al ser un sistema operativo que se distribuye desde Google a éstos, permite una variedad abundante en términos de diseño y especificaciones que buscan abarcar distintos nichos de mercado, lo cual hace la labor de desarrollo un poco más compleja; con respecto a las tabletas de Apple, sus dimensiones son de 7.9 pulgadas para las iPad mini y de 9.7 pulgadas para el iPad. La resolución de los dispositivos también ha de tenerse en cuenta al momento de diseñar los elementos gráficos que van a hacer parte de la aplicación.

Para tener una idea más clara de estas características, se ha consignado la información provista por la página oficial de Apple en la que se puede apreciar cada dispositivo con iOS junto con sus dimensiones y resolución (ver Tabla 2).

Dispositivo	Dimensiones pantalla	Resolución
<i>iPad 3ra - 4ta generación (Retina Display)</i>	9.7 pulgadas	2048 x 1536 pixeles 264 pixeles x pulgada
<i>iPad 1ra - 2da generación</i>	9.7 pulgadas	1024 x 768 pixeles 132 pixeles x pulgada
<i>iPad mini</i>	7.9 pulgadas	1024 x 768 pixeles 163 pixeles x pulgada
<i>iPod Touch 5ta generación / iPhone 5 (Retina Display)</i>	4 pulgadas	1136 x 640 pixeles 326 pixeles x pulgada
<i>iPod Touch 4ta generación / iPhone 4/4s (Retina Display)</i>	3.5 pulgadas	960 x 640 pixeles 326 pixeles x pulgada
<i>iPod Touch 1ra - 3ra generación iPhone 2g/3g/3gs</i>	3.5 pulgadas	480 x 320 pixeles 163 pixeles x pulgada

Tabla 2. Dimensiones y resoluciones de pantalla por dispositivo, Adaptada por el autor

Más allá de los componentes gráficos que conforman la aplicación, existen unas regulaciones en cuanto a la creación de los elementos básicos que ésta requiere para ser desplegada correctamente por iOS, de igual forma, se clasifican los distintos elementos gráficos que componen la aplicación de acuerdo a cada dispositivo móvil (ver Tabla 3).

Elemento gráfico (Formato PNG)	Dimensiones por dispositivo (en pixeles)				
	<i>iPod Touch 5ta generación</i> <i>iPhone 5 (Retina Display)</i>	<i>iPod Touch 4ta generación</i> <i>iPhone 4/4s (Retina Display)</i>	<i>iPod Touch 1ra - 3ra generación</i> <i>iPhone 2g/3g/3gs</i>	<i>iPad 3ra - 4ta generación (Retina Display)</i>	<i>iPad 1ra - 2da generación</i>
*Icono de la aplicación	114 x 114	114 x 114	57 x 57	144 x 144	72 x 72
*Icono de la aplicación para la App Store	1024 x 1024	1024 x 1024	512 x 512	1024 x 1024	512 x 512
*Imagen de ejecución de la aplicación	640 x 1136	640 x 960	320 x 480	1536 x 2008 (vertical) 2048 x 1496 (horizontal)	768 x 1004 (vertical) 1024 x 748 (horizontal)
Icono pequeño para resultados de la búsqueda de Spotlight y ajustes	58 x 58	58 x 58	29 x 29	100 x 100 (búsqueda) 58 x 58 (ajustes)	50 x 50 (búsqueda) 29 x 29 (ajustes)
Icono web de acceso directo	114 x 114	114 x 114	57 x 57	144 x 144	72 x 72
Icono de barra de navegación	40 x 40 aprox.	40 x 40 aprox.	20 x 20 aprox.	40 x 40 aprox.	20 x 20 aprox.
Icono de pestaña de navegación	60 x 60 aprox.	60 x 60 aprox.	30 x 30 aprox.	60 x 60 aprox.	30 x 30 aprox.

Tabla 3. Componentes gráficos de acuerdo a dispositivo iOS (2012), tomado de (2012), http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/IconsImages/IconsImages.html#//apple_ref/doc/uid/TP40006556-CH14-SW1

* Elementos requeridos de acuerdo a las políticas de Apple.

7.1.11 Georeferenciación

La Georeferenciación, de acuerdo a (Wikipedia, 2013b) está definida como la identificación de la ubicación geográfica real de un objeto, como por ejemplo, un radar, un teléfono móvil o un equipo de cómputo que se encuentra conectado a la red. Para que esto sea posible, el objeto a localizar debe tener un chip GPS integrado que, según comenta Ionescu (2010), hace uso de información satelital basada en la longitud y latitud del mismo, para calcular su posición exacta, la cual posteriormente puede ser usada y graficada por servicios de terceros como Google con Google Maps. Cuando el equipo desde el cual se solicita la ubicación tiene conectividad a una red de datos, y por alguna razón no le fué posible entregar su localización haciendo uso del GPS, el servicio entonces, de acuerdo a sus prestaciones y limitaciones, procede a intentar recuperarla por medio de otros métodos como por ejemplo, triangular la posición por medio de las torres de comunicación empleadas por el operador de telefonía móvil celular que presta el servicio.

Claramente, esta tecnología se encuentra compuesta tanto por una pieza de hardware como por una de software, aquella que permite rastrear la ubicación actual de un equipo, y la que explota el potencial de esa pequeña porción de información suministrada. Dentro de los servicios más populares y completos que se pueden encontrar para poder hacer uso de los datos de localización recuperados, se encuentran Google Maps, Yahoo! Maps, Bing Maps, MapQuest, OpenStreetMap, Tom Tom, Waze y Nokia Here. Cada uno de estos servicios ofrece una funcionalidad condicionada a sus propios proveedores de información cartográfica, tipos de mapas (vistas satelitales, vista de carreteras, vistas de trafico, vistas híbridas), navegadores web

soportados, antigüedad de las imágenes, disponibilidad de funcionalidades extra de acuerdo al área, idiomas soportados, integración al GPS, integración de voz, localización basada en triangulación celular, localización basada en redes Wi-Fi y la más importante de todas, la disponibilidad de APIs. (Wikipedia, 2013b).

Las APIs de los servicios de mapas, permiten implementar el producto rápidamente en una solución personalizada de acuerdo a las necesidades de la misma. Para la app que se construirá, se hará uso del servicio más popular y completo disponible hasta el momento: Google Maps. Existen varias razones para elegir este servicio, por un lado, encontramos la experiencia de este gigante en el negocio, la integridad y precisión del contenido provisto, y su constante esfuerzo por mejorar la herramienta de forma regular, algo que se puede ver evidenciado en la más reciente publicación de su API para aplicaciones web, su versión 3, la cual se encuentra diseñada específicamente para garantizar un mejor rendimiento y aplicabilidad en dispositivos móviles de acuerdo a su sitio oficial.

La utilización de la API denominada Google Maps JavaScript API V3, como bien lo dice su nombre, se realiza a través del lenguaje de programación Javascript, el cual es el lenguaje de scripting más popular de la web. Se deben tener nociones claras del paradigma de programación orientado a objetos, ya que esta API hace un intensivo uso del mismo.

En cuanto al apartado de licenciamiento, Google permite un máximo de 25,000 cargas de mapas por día, siendo una carga técnicamente definida como aquella en la cual el mapa se

inicializa en una página web, es decir, aparece, descartando cualquier otra acción que el usuario realiza sobre éste después de que ha cargado en su totalidad. Dado que existen sitios web que experimentan picos de audiencia (súbitos aumentos en tráfico de red) Google no tiene en cuenta las “violaciones” a éste límite sino hasta cuando la situación se presenta de forma reiterada durante el curso de más de 90 días consecutivos.

Una vez superada esta barrera, se debe proceder a elegir entre dos alternativas: realizar el pago de esas cargas adicionales, las cuales son calculadas de forma diaria y cobradas de forma mensual, o, si el volumen de usuarios es realmente alto, adquirir una licencia de negocios la cual es mucho más apropiada para éste último escenario dentro del contexto económico.

En el esfuerzo que supone aplicar esta tecnología de georeferenciación al desarrollo del prototipo, el protagonista restante a las herramientas a emplear mencionadas anteriormente, es el navegador. En iOS, el navegador por defecto se conoce como Safari, a través de este se obtendrán las coordenadas de ubicación del usuario que serán aprovechadas por Google Maps para llevar a cabo el objetivo propuesto.

En el esfuerzo que supone aplicar esta tecnología de georeferenciación al desarrollo del prototipo, el protagonista restante a las herramientas a emplear mencionadas anteriormente, es el navegador. En iOS, el navegador por defecto se conoce como Safari, a través de este se obtendrán las coordenadas de ubicación del usuario que serán aprovechadas por Google Maps para llevar a cabo el objetivo propuesto.

7.2 Marco Conceptual

La plataforma *iOS*, originalmente conocida como iPhone Operative System (puesto que hizo su primera aparición en este dispositivo), es el sistema operativo actual de todos los dispositivos móviles de Apple, (iPhone, iPod Touch, iPad, Apple Tv) que en principio, careció de los medios necesarios para aprovechar su integración de hardware/software por medio de la construcción de aplicaciones que pudieran llevar más allá su potencial como smartphone.

Para el lanzamiento de la segunda versión del teléfono de Apple, se implementa la tienda de aplicaciones de la compañía denominada “App store” en respuesta a un cúmulo de aspectos presentes en la escena de los smartphones como lo fueron la demanda de los desarrolladores de una herramienta nativa, dada la carencia de explotar al máximo el potencial de la plataforma con el método empleado en esa época como alternativa para el desarrollo de aplicaciones: las *Web Apps*, o aplicaciones móviles que en ese momento no contaban con el suficiente desarrollo de las tecnologías web para obtener un producto de calidad respetable y además carecían de la posibilidad de acceder a toda la funcionalidad del dispositivo, algo que sí permitió posteriormente, el lanzamiento del *SDK* de Apple.

Un *SDK* se entiende como “Kit de Desarrollo de Software”, cuya versión actual incluye un conjunto de herramientas que hacen mucho más completa y plácida la construcción de aplicaciones bajo el lenguaje de programación Objective-C. Dentro de estas herramientas se encuentran las que siguen a continuación:

7.2.1 Xcode

Xcode es un Entorno de Desarrollo Integrado (*IDE*), empleado para desarrollar aplicaciones tanto para el sistema operativo de ordenadores mac (*OS X*), como para *iOS*. Entre sus características encontramos un editor de texto, soporte para autocompletamiento, análisis de código, navegador de archivos y una buena variedad de herramientas para testear rendimiento y realizar depuración. Fue diseñado desde cero para poder exprimir al máximo las últimas tecnologías generadas por Apple y su versión actual es la cuarta. (Apple, 2012b)

7.2.2 Interface Builder

Interface Builder es una aplicación implícita dentro de Xcode, que en el pasado funcionaba de forma externa al IDE, la cual permite construir la interfaz gráfica para una app de forma visual facilitando la incursión en el desarrollo nativo. Controles tales como botones, barras de estado, deslizadores y etiquetas pueden ser fácilmente arrastrados y configurados por medio de paneles para la creación de la interfaz. También permite manejar los eventos, realizando la conexión entre los objetos y su respectiva acción. Esta herramienta es una buena forma de familiarizarse con la interfaz gráfica de iOS pero es altamente recomendable entender lo que *UIKit* genera automáticamente para comprender la creación desde cero de una interfaz gráfica. (Apple, 2012a)

7.2.3 Frameworks

La herramienta más importante de todas; permite entre otras crear interfaces de usuario, escribir código de “networking”, encriptar información importante, dibujar gráficas, ejecutar pistas de audio y video, almacenar información y contraseñas, tomar fotografías, renderizar páginas web, etc. Los Frameworks que provee Apple, permiten aprovechar desde un principio un conjunto de comandos y herramientas bastante robustas para construir aplicaciones de forma mucho más sencilla optimizando el tiempo de desarrollo.(Apple, 2012a)

7.2.4 HTML

Hyper Text Markup Language, según (King, 2000) le dice a un servidor Web cómo debe dar formato al documento como página Web y cómo un navegador debe mostrarlo en pantalla. En palabras más sencillas, se dice que el lenguaje de etiquetado universal se encarga de dar *estructura* a la aplicación, por medio del apropiado empleo de estas según las características que se requieran. El empleo del estándar en su versión más reciente, *HTML5*, permite ampliar las posibilidades multimedia de la aplicación y realizar la implementación de una tecnología absolutamente indispensable en el objeto del proyecto: Georeferenciación.

7.2.5 CSS

Cascade Style Sheets, de acuerdo a (Langley) permite separar la presentación del contenido de una página web, ofreciendo compatibilidad a través de las distintas plataformas; su funcionamiento se basa en el llamado a las etiquetas construídas con HTML, denominados en el contexto web como selectores, con el fin de aplicar estilos, es decir, especificar el *diseño* que va a tener nuestra aplicación, dándole la apariencia que percibirá el usuario final.

7.2.6 JavaScript

Definido por (Gibbs, 1997) es un lenguaje liviano interpretado, con funciones básicas enmarcadas en un paradigma orientado a objetos. Cuando Gibbs se refiere a “lenguaje interpretado” explica que los programas son escritos usando texto plano y estos interpretados textualmente en tiempo de ejecución; en otras palabras se puede decir que se trata de líneas de código que van siendo traducidas una a una por el navegador y cuya finalidad es proveer interacción entre el usuario y la app.

7.2.7 JQuery

Es una gran librería de JavaScript que simplifica la escritura de código permitiendo obtener más y mejores resultados, esto es, escribir menos código y obtener mucho más en cuanto a funcionalidad se refiere. Dentro de sus grandes ventajas está el hecho de que permite generar efectos visuales de forma más sencilla que mediante el uso directo de Javascript.

7.2.8 DOM

El DOM, cuya sigla significa Document Object Model, se basa en la jerarquía de elementos HTML que van siendo desplegados en una página web. En otras palabras, se puede hablar de cada elemento como un nodo que va conformando una estructura en forma de árbol la cual se va distribuyendo en orden jerárquico. Finalmente cabe mencionar, que cada uno de estos nodos es tomado como un objeto, característica que permite ejercer manipulación sobre cada uno de estos haciendo uso de javascript.

7.2.9 Archivo Binario

Un archivo binario es un fichero en lenguaje máquina, que el cerebro del computador, mejor conocido como CPU, es capaz de leer para ejecutar las instrucciones consignadas en él. En el contexto de desarrollo de software, estos archivos son producto de la labor del compilador, que

toma los archivos que el programador ha escrito en x lenguaje de programación y los convierte a un lenguaje que el ordenador puede entender y ejecutar.

7.2.10 Plugin

Un plugin dentro del contexto de programación, puede considerarse como un conjunto de instrucciones reutilizables, que contienen cierta funcionalidad. Estos archivos permiten agilizar el trabajo del desarrollador evitando emplear tiempo innecesario en generar código ya existente.

7.2.11 Lenguaje de programación

Un lenguaje de programación se basa en una sintaxis definida la cual es empleada para dar instrucciones a la CPU. Los lenguajes de programación son categorizados de acuerdo a su nivel. Los llamados lenguajes “de alto nivel” son aquellos cuyas instrucciones son dadas en un lenguaje familiar al empleado por los seres humanos, mientras los lenguajes conocidos como “de bajo nivel” se aproximan más en su sintaxis al lenguaje máquina.

7.2.12 SPI

Sigla que se lee como Single Page Interface, y que se refiere a que todos los ficheros que comprenden un sitio o aplicación web, están contenidos dentro de un solo archivo con el objetivo de simplificar la transmisión de datos e incrementar la respuesta a usuario haciendo más liviana la navegación.

7.2.13 API

Una API (Application Programming Interface), define la forma adecuada en la que un desarrollador solicita servicios de un programa en particular. Estas son fundamentales para la construcción de una solución y su uso está sujeto a las condiciones y políticas del proveedor que las suministre. (Orenstein, 2000). Normalmente, se pone a disposición del desarrollador la documentación correspondiente a una API en particular con el fin de facilitar su implementación y garantizar su funcionamiento.

7.2.14 GPS

Conocido como Global Positioning System, es un servicio de localización satelital que permite conocer la ubicación geográfica de un dispositivo, mediante sus coordenadas de latitud y longitud registradas.

7.2.15 Base de datos

Una base de datos es básicamente, un conjunto de datos organizado. Dado que éstos datos se encuentran estructurados, se procede a realizar consultas por lo general, a través de un lenguaje estructurado de consultas conocido como SQL (Structured Query Language), aunque existen varias alternativas afuera del mundo SQL. El objetivo es que los datos almacenados tengan la menor redundancia posible y se encuentren organizados de forma coherente para, mediante la manipulación de los mismos, obtener información.

8. ANÁLISIS DE REQUERIMIENTOS

Para la consecución de este primer objetivo, se ha procedido a recolectar información por medio de charlas sostenidas con usuarios de la plataforma iOS de acuerdo a lo planteado previamente en la metodología. Las charlas y notas diligenciadas dieron como resultado un total de 19 funcionalidades deseadas que se listan a continuación:

1. Facilidad de uso.
2. Demarcación de la ruta desde la ubicación del usuario hasta el parqueadero.
3. Instrucciones de parqueo asistidas por voz.
4. Información de horarios disponibles.
5. Tarifas de hora y día para conocer el parqueadero más económico.
6. Permitir conocer cuál es el parqueadero más cercano.
7. Calcular el costo a pagar.
8. Conocer si el parqueadero tiene lugares disponibles.
9. Mostrar en tiempo real cómo va quedando parqueado el auto.
10. La aplicación debe ser rápida.
11. Informar si el parqueadero es cubierto o descubierto.
12. Mostrar si tiene convenio con algún establecimiento para aprovechar las promociones.
13. Saber que tipo de vehículos se pueden parquear.
14. La interfaz de la aplicación debe ser llamativa.
15. Permitir reservar un parqueadero sobre la marcha.

16. Poder ver los datos completos del parqueadero, como dirección y teléfono.
17. Permitir agregar parqueaderos favoritos.
18. Entregar la opción de hacer comentarios sobre el parqueadero.
19. La aplicación debe ser práctica.

Una vez definidas las funcionalidades que los usuarios quisieran tener en la aplicación, se procedió a realizar un filtro empezando por aquellas que de acuerdo al objetivo principal, son inviábiles o comprenden un desarrollo mucho más complejo y de momento innecesario. Dichas funcionalidades fueron:

- a. Instrucciones de parqueo asistidas por voz.
- b. Conocer si el parqueadero tiene lugares disponibles.
- c. Mostrar en tiempo real cómo va quedando parqueado el auto.
- d. Informar si el parqueadero es cubierto o descubierto.
- e. Permitir reservar un parqueadero sobre la marcha.

Las características **a** y **c** fueron descartadas dado que son funcionalidades que van mucho más allá del objetivo primordial de esta app y que demandan un gran conocimiento técnico y mucho más tiempo. Para hacer posibles **b** y **e** se requeriría de un sistema del lado de los parqueaderos que se sincronizara con la app para poder establecer en tiempo real los cambios que se generen en estos y así el usuario pudiera intervenir, una idea genial pero que de momento no se encuentra dentro del horizonte del proyecto. Por último, la opción de implementar **d** es

bastante viable técnicamente pues es simplemente un dato más anexo a la información del parqueadero, pero dado que sólo 1 persona de las indagadas la postuló, y lo que aporta no afecta para bien ni para mal el desarrollo, se tomó la decisión de dejarla al margen de los requerimientos.

Así las cosas, se determinó que las características restantes eran viables a nivel técnico y funcional, pero bajo la premisa de imponer simplicidad antes que complejidad desde el punto de vista del usuario y de dar prioridad a cumplir con el objetivo final del prototipo, se posterga la implementación de las siguientes funcionalidades y se centra la atención en el diseño y construcción de las restantes:

- Mostrar si tiene convenio con algún establecimiento para aprovechar las promociones.
- Permitir agregar parqueaderos favoritos.
- Entregar la opción de hacer comentarios sobre el parqueadero.

Dada la similitud de ciertos requerimientos, se realiza una agrupación acorde y se lleva a cabo una categorización de requerimientos funcionales y no funcionales (como se puede observar en la Figura 6 y Figura 7).

Requerimientos Funcionales
Localizar parqueaderos relativos a la ubicación del usuario.
Brindar información completa por parqueadero: horarios, tarifas, tipo de vehículo
Clasificar parqueaderos por tarifa minuto, ubicación y tipo de vehículo.
Calcular el tiempo de parqueo y el valor a pagar.

Figura 6. Requerimientos Funcionales
(Adaptada por el autor)

Requerimientos No Funcionales
Aplicación fácil de usar e intuitiva
Desempeño veloz
Interfaz llamativa

Figura 7. Requerimientos No Funcionales
(Adaptada por el autor)

8.1 Descripción de Requerimientos Funcionales

A continuación se presenta una breve descripción por cada requerimiento funcional, basada en un nombre que identifica la funcionalidad, un resumen que explica su función, unas entradas que representan la forma en la que el usuario activa el requerimiento descrito y unos resultados producto de esa interacción usuario - sistema (resultados apreciables en la Figura 8, Figura 9, Figura 10 y Figura 11).

Nombre	R1 - Localizar parqueaderos relativos a la ubicación del usuario.
Resumen	Ubica en un mapa de Google los parqueaderos cercanos a la posición del usuario
Entradas	
Elemento deslizable	
Resultados	
Despliega en un mapa provisto por Google, los parqueaderos relativos a la localización del usuario	

Figura 8. Requerimiento Funcional
(Adaptada por el autor)

Nombre	R2 - Brindar información completa por parqueadero: horarios, tarifas, tipo de vehículo
Resumen	Despliega la información apropiada por cada parqueadero
Entradas	
Parqueadero seleccionado	
Resultados	
Cuadro de diálogo desplegando la información consignada	

Figura 9. Requerimiento Funcional
(Adaptada por el autor)

Nombre	R3 - Clasificar parqueaderos por tarifa minuto, tarifa hora y ubicación.
Resumen	Listar parqueaderos encontrados de acuerdo al criterio seleccionado por el usuario
Entradas	
Opción seleccionada en el criterio de búsqueda (tarifa minuto / tarifa hora / ubicación)	
Resultados	
Listado de parqueaderos de acuerdo al criterio seleccionado	

Figura 10. Requerimiento Funcional
(Adaptada por el autor)

Nombre	R4 - Calcular el tiempo de parqueo y el valor a pagar.
Resumen	De acuerdo al parqueadero elegido por el usuario se calcula el monto a pagar
Entradas	
Valor del minuto, tiempo transcurrido	
Resultados	
Valor a cancelar en pesos colombianos	

Figura 11. Requerimiento Funcional
(Adaptada por el autor)

8.2 Descripción de Requerimientos No Funcionales

Para la descripción de los requerimientos no funcionales, se elabora una representación compuesta de los elementos: tipo, el cual busca identificar la característica en la que el requerimiento se fundamenta, es decir, cómo funciona cada requerimiento; nombre, que identifica el requerimiento como tal y descripción el cual contiene las premisas bajo las cuales el requerimiento ha sido construido. (como se muestra en la Figura 12, Figura 13 y Figura 14)

Tipo	Usabilidad
Nombre	Aplicación fácil de usar e intuitiva
Descripción	
<ul style="list-style-type: none"> ▸ Se parte de un supuesto en el que el usuario debe intervenir de forma mínima para obtener respuesta de la aplicación. ▸ Se construye una interfaz agradable, sencilla y clara. 	

Figura 12. Requerimiento No Funcional
(Adaptada por el autor)

Tipo	Usabilidad
Nombre	Desempeño veloz
Descripción	
<ul style="list-style-type: none"> ▸ La aplicación se optimizará por medio de tecnologías como Ajax y el código se minificará para ofrecer un mejor rendimiento. ▸ La consulta de los parqueaderos disponibles se hará en el momento inicial, incluso con los que se encuentren fuera del perímetro para que cuando la posición del usuario cambie, la respuesta en pantalla sea ágil evitando realizar peticiones recurrentes al servidor. 	

Figura 13. Requerimiento No Funcional
(Adaptada por el autor)

Tipo	Interfaz de Usuario
Nombre	Aplicación llamativa
Descripción	
<ul style="list-style-type: none"> ▸ La aplicación tendrá un diseño sencillo pero llamativo para los usuarios. ▸ Todos los elementos gráficos de la app se diseñaran y construirán teniendo en cuenta las dimensiones y resoluciones de los dispositivos iOS. ▸ Se hará uso de la libreria jQtouch para darle a la aplicación un acabado profesional y agradable. 	

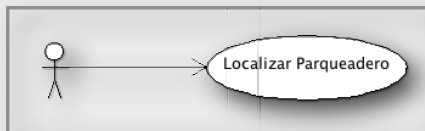
Figura 14. Requerimiento No Funcional
(Adaptada por el autor)

9. MODELAMIENTO DEL PROTOTIPO

9.1 Casos de uso

Partiendo de la base en la que una interacción usuario-sistema se traduce, desde el punto de vista del modelado de aplicativos de software, como un caso de uso, se realiza la correspondiente descripción de cada uno (ver Figura 15, Figura 16 y Figura 17), en una tabla que consigna las características de éstos indicando la identificación de cada caso de uso, los actores involucrados (usuarios, sistema), las precondiciones a nivel técnico y de contexto que deben existir para que el requerimiento pueda ser satisfecho, algo denominado “flujo normal” que establece paso a paso lo que sucede durante la ejecución ideal del requerimiento y un “flujo excepcional” que de igual forma, marca lo que sucede en un escenario alternativo dentro del cual el requerimiento no se ejecuta de forma idónea.

Finalmente se muestran unas post condiciones en las cuales se registra lo que el sistema realiza una vez finalizada la ejecución del caso de uso y un último espacio dedicado a plasmar los posibles fallos que se pueden llegar a tener durante el proceso junto con sus posibles soluciones.



1. IDENTIFICACIÓN DE CASO DE USO

1.1 ID Caso	1,0
1.2 Nombre	Localizar Parqueadero

2. HISTORICO DE CASO DE USO

2.1 Autor	Miguel Rojas Cortés
2.2 Fecha de Creación	25 Noviembre de 2012
2.3 Última Actualización	19 Junio de 2013

3. DEFINICIÓN DE CASO DE USO

3.1 Descripción

El usuario desliza un control que permite realizar la consulta de los parqueaderos relativos a su posición.

3.2 Actores

- Usuario
- Servidor
- Base de datos

3.3 Precondiciones

- El equipo del usuario debe tener acceso a la red, bien sea por medio de datos o wifi.
- El usuario debe tener habilitado el GPS de su equipo.
- El usuario debe estar ubicado en una de 3 zonas específicas de Chapinero (Emaús /Chapinero Norte / Quinta Camacho).

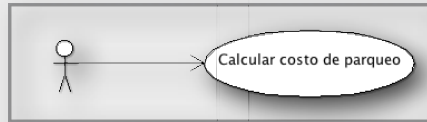
3.4 Flujo Normal

Paso	Actor	Sistema
1	El usuario desliza en forma vertical el control con forma de mira hasta que este quede completamente sobre el objetivo representado por medio de un icono de parqueadero que se encuentra ubicado en la parte inferior de la pantalla.	
2		El evento "drag" es lanzado a medida que el usuario desplaza el control hacia el objetivo (el logo del parqueadero), a la vez que se evalúa la posición actual del control en su recorrido.
3		El evento "drop" se activa sobre el objetivo, habilitando una clase css que permite resaltar el lugar al que la mira debe ser dirigida por el usuario.
4	El fondo del objetivo se torna blanco como señal indicativa al usuario del elemento que esta desplazando.	
5		Se envía una imagen animada a la pantalla del usuario, indicando el progreso del proceso de consulta.
6	El usuario recibe en su pantalla una imagen indicando que una consulta se encuentra en proceso.	

7		Si el recorrido realizado por el control es de exactamente 380 pixeles, momento en el cual la mira se encuentra completamente llena por el objetivo, se capturan las coordenadas geográficas actuales del dispositivo móvil.
8		Se realiza una comparación de las coordenadas del usuario respecto al área sobre la cual se encuentran los parqueaderos, si el usuario está ubicado por fuera de este rango, simplemente se genera una alerta informando al usuario que la información de la aplicación se encuentra restringida a las localidades de Chapinero alto, Emaús y Quinta Camacho.
9		Por medio de una consulta Ajax, el dispositivo envía las coordenadas geográficas a un servicio php almacenado en un servidor local, éste servicio realiza una consulta a la base de datos de acuerdo a los datos de ubicación enviados desde el dispositivo móvil y recupera los más cercanos a la misma.
10		Los datos recuperados son almacenados en un arreglo y codificados en json para ser devueltos al script que generó la consulta.
11		Se genera un mapa usando la API de Google Maps y marcadores para asociar los elementos almacenados en el arreglo generado por el servicio php
12	El usuario recibe respuesta a su consulta por medio de un mapa que se despliega en la pantalla del móvil mostrando los parqueaderos cercanos relativos a su posición.	
3.5 Flujo Excepcional		
#	Actor	Sistema
1	El usuario desliza hacia abajo el control que aparece una vez se lanza la aplicación, pero éste no llena la mira con el logo de parqueadero que se encuentra ubicado en la parte inferior de la pantalla.	
2		No hay respuesta pues el recorrido del control no ha sido igual a 380 pixeles.
3.6 Post Condiciones		
Ninguna		
Posibles Fallos		Solución
Imposibilidad de establecer comunicación con el servidor.		Mensaje a usuario indicando el error y solicitando intentar nuevamente.
Imposibilidad en el servidor de establecer comunicación con la base de datos.		Mensaje a usuario indicando el error y solicitando intentar nuevamente.
Imposible acceder al GPS del equipo.		Mensaje a usuario indicando que encienda el GPS de su equipo y solicitando intentar nuevamente.

Figura 15. Descripción Caso de Uso
(Adaptada por el autor)

Como se puede apreciar, la descripción del caso de uso “Localizar Parqueaderos” se torna extenso dada la cantidad de procesos involucrados que deben ser correctamente detallados. Es clave el cumplimiento de las precondiciones establecidas para que el denominado “flujo normal” pueda ser ejecutado de forma satisfactoria evitando inconvenientes en el desarrollo de la funcionalidad. También es importante resaltar el hecho de que el “grosso” de la aplicación se encuentra en este requerimiento, técnicamente hablando, pues intervienen los distintos servicios proporcionados por la API de Google que permiten traducir un par de coordenadas en una aplicación funcional.



1. IDENTIFICACIÓN DE CASO DE USO

1.1 Id Caso	2,0
1.2 Nombre	Calcular costo de parqueo

2. HISTORICO DE CASO DE USO

2.1 Autor	Miguel Rojas Cortés
2.2 Fecha de Creación	Noviembre 25 de 2012
2.3 Última Actualización	Junio 19 de 2013

3. DEFINICIÓN DE CASO DE USO

3.1 Descripción

Se calcula el valor a cancelar de acuerdo al parqueadero que haya sido elegido una vez hecha la localización.

3.2 Actores

Usuario

3.3 Precondiciones

El usuario ha seleccionado uno de los parqueaderos encontrados.

3.4 Flujo Normal

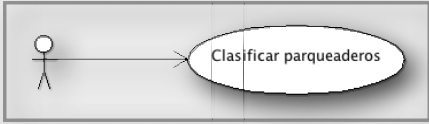
Paso	Actor	Sistema
1	El usuario presiona el botón calcular costo de parqueo ubicado en la parte superior derecha de la pantalla.	
2		Realiza la carga de la vista que contiene los controles para ejecutar el cálculo del costo de parqueo.
3	El usuario ingresa el valor del minuto y activa el temporizador, mediante el botón "iniciar"	
4		Se limpian todas las variables y se genera un temporizador cuya finalidad es almacenar la cantidad de minutos transcurridos desde que el usuario parquea hasta que regresa.
5		Se despliega un cronómetro que permite ver el tiempo de parqueo que va transcurriendo, y se actualiza el botón "iniciar".
6	El usuario recibe una actualización en la vista de "costo" que le permite ver el tiempo de parqueo que va transcurriendo, y el cambio de estado del botón que presiono, que ahora se encuentra rojo con el texto "detener".	
7	El usuario detiene el temporizador, mediante el botón "detener".	

8		Se toma el total de minutos transcurridos partiendo de la base en la que si el temporizador se detiene antes de que se cumpla un minuto, éste se escala y se toma como cumplido. Se procede a calcular el valor del tiempo de parqueo transcurrido de acuerdo al valor del minuto ingresado por el usuario.
9		El botón para iniciar y detener el temporizador se actualiza de nuevo permitiendo realizar de nuevo el proceso cuando el usuario así lo requiera. El temporizador se oculta hasta que se genere otro proceso.
10	El usuario recibe en pantalla el valor a cancelar en pesos colombianos. El temporizador desaparece y el botón para activarlo se actualiza de nuevo permitiendo al usuario iniciar un nuevo proceso.	
3.5 Flujo Excepcional		
#	Actor	Sistema
1	El usuario activa el temporizador sin ingresar el valor del minuto.	
2		Se despliega un mensaje indicando que se debe ingresar el valor del minuto.
3	El usuario ingresa el valor del minuto.	
4		Se detecta un cambio en el campo de texto que recibe el minuto, y se procede a calcular el costo del parqueo.
5	El usuario recibe en pantalla el valor a cancelar en pesos colombianos. El temporizador desaparece y el botón para activarlo se actualiza de nuevo permitiendo al usuario iniciar un nuevo proceso.	
3.6 Post Condiciones		
Una vez detenido el contador y realizado el cálculo, los valores se reestablecen a cero.		
Posibles Fallos		Solución
Ninguno		Ninguno

Figura 16. Descripción Caso de Uso
(Adaptada por el autor)

Este caso de uso, si bien se toma como un requerimiento secundario, pues es un complemento agradable pero no determinante en el objetivo primordial de la aplicación, ha sido pensado de tal forma que el usuario encuentre practicidad en la ejecución de la funcionalidad con el fin de mantener la filosofía de “usabilidad” que se destaca a lo largo y ancho de la aplicación. Es importante resaltar que como siempre, existe margen de mejora para ir un poco más allá en la experiencia de usuario, pero dada la necesidad de profundizar de forma mucho más marcada en

las distintas variables que se involucran en este aspecto, se llega a una solución intermedia plasmada en la tabla anterior.

		
1. IDENTIFICACIÓN DE CASO DE USO		
1.1 Id Caso	3,0	
1.2 Nombre	Clasificar parqueaderos	
2. HISTORICO DE CASO DE USO		
2.1 Autor	Miguel Rojas Cortés	
2.2 Fecha de Creación	Noviembre 26 de 2012	
2.3 Última Actualización	Junio 19 de 2013	
3. DEFINICIÓN DE CASO DE USO		
3.1 Descripción		
Lista los parqueaderos encontrados según el criterio elegido por el usuario (tarifa minuto, tarifa hora, cercanía).		
3.2 Actores		
Usuario		
3.3 Precondiciones		
El usuario ha localizado los parqueaderos disponibles próximos a su ubicación.		
3.4 Flujo Normal		
Paso	Actor	Sistema
1	El usuario presiona el botón <i>clasificar parqueaderos</i> ubicado en la parte superior izquierda de la pantalla.	
2		Se despliegan 3 opciones para que el usuario seleccione el criterio de organización por el cual se registrará la lista de parqueaderos a construir: tarifa minuto, tarifa hora, cercanía.
3	El usuario selecciona una de las 3 opciones disponibles para clasificar los parqueaderos encontrados: tarifa minuto, tarifa hora, cercanía.	
4		Se genera una nueva vista para el usuario en la que se listan los parqueaderos localizados teniendo en cuenta el criterio elegido por el usuario previamente.
5	El usuario recibe los parqueaderos listados de acuerdo al criterio seleccionado anteriormente.	

3.5 Flujo Excepcional		
#	Actor	Sistema
1	El usuario no selecciona ninguna de las opciones disponibles quedando por defecto el texto "clasificar".	
2		Identifica que no se ha elegido ninguno de los criterios de selección, por tanto no ejecuta ninguna función.
3	La vista de localización permanece sin cambio alguno, pues no se solicitó ordenar los parqueaderos bajo ningún criterio.	
3.6 Post Condiciones		
Ninguna.		
Posibles Fallos		Solución
Ninguno		Ninguno

Figura 17. Descripción Caso de Uso
(Adaptada por el autor)

El anterior caso de uso es quizás el más sencillo de implementar siendo su fuerte el hecho de que la información ya se encuentra disponible del lado del cliente, por lo cual la respuesta a la petición del usuario es casi inmediata pues no existe la necesidad de solicitar información adicional al servidor cada vez que el usuario elija un criterio de clasificación distinto. El objetivo específico es entregar la mayor cantidad de información disponible, organizada de acuerdo a la selección del usuario.

9.2 Modelo Entidad-Relación

El siguiente modelo entidad-relación permite entender la forma en la que la aplicación hace uso del almacenamiento persistente a través de una base de datos MySQL. Siguiendo con la filosofía de la simplicidad en el prototipo a elaborar, el modelo de la base de datos consta de tan sólo 2 tablas: Tabla “parkings” y la tabla “owners”.

El objetivo de la tabla “parkings” es almacenar nuevos parqueaderos ingresados por medio de un módulo administrador y retornar los parqueaderos disponibles cuando un usuario realiza una consulta desde su dispositivo iOS. Dado que cada tabla se compone de campos que almacenan la información, se han dispuesto 13 de estos que contienen la información necesaria para poder realizar las consultas mencionadas. Cada uno de estos campos, se crea indicando el tipo de dato que va a ser almacenado, (numérico, texto, fecha, etc...) seguido de la longitud, en número de caracteres, que podrá recibir:

id_parking: Es un campo numérico que comprende una característica denominada “Auto Incremento”, es decir, cada vez que un parqueadero sea agregado por medio del módulo administrador, éste quedara identificado por medio de éste campo y automáticamente incrementará su valor en una unidad, llegando hasta un longitud máxima de 11 caracteres. El tipo de dato empleado en este campo es conocido como “INT”, que permite salvar valores enteros o exactos en un rango determinado. También se atribuye la característica de llave

principal a este campo, lo cual permite identificar de manera inequívoca un registro completo dentro de la tabla.

address: Campo destinado a almacenar la dirección de un parqueadero. El tipo de dato que emplea es denominado como “VARCHAR” el cual, de acuerdo a la documentación oficial de MySQL, permite guardar cadenas de texto, cuyo espacio en cada columna varia de acuerdo a la longitud de la palabra almacenada. La longitud de este campo ha sido establecida en un máximo de 50 caracteres con el fin de tener suficiente espacio para guardar direcciones extensas.

phone: Este campo permite almacenar el número telefónico de cada establecimiento. Su tipo de dato es INT y la máxima longitud permitida en cuanto a cantidad de dígitos es de 12. Este límite se ha establecido contemplando números de celulares y dejando un pequeño margen por seguridad.

schedulePark: Campo dentro del cual se almacena la información concerniente a los horarios de un parqueadero, su tipo de dato es VARCHAR y su longitud se ha estimado en 60 caracteres dada la variabilidad de tiempos entre distintos parqueaderos, que algunas veces, hacen un poco más extensa la descripción del horario.

latitude: Para poder almacenar las coordenadas de latitud de cada parqueadero, se ha dispuesto este campo cuyo tipo de dato corresponde a DOUBLE, que de acuerdo a la documentación, se trata de un dato numérico con punto decimal cuya precisión supera a la del

tipo de dato FLOAT usado para el mismo fin. No se ha especificado la longitud de este campo para garantizar que cada coordenada tome el espacio que requiera y evite afectar la precisión de localización.

longitude: Campo que recibe las coordenadas de longitud para un parqueadero en particular. Cumple con las mismas características del campo anterior: tipo de dato DOUBLE y longitud no especificada.

minCar_value: Campo que recibe el costo por minuto de parqueo para un vehículo. El tipo de dato elegido ha sido INT y su longitud ha sido establecida en un máximo de 3 dígitos dado que en la actualidad, este valor se encuentra en un rango de 2 dígitos, pero que a futuro, fácilmente puede llegar a cubrir 1 más.

minBike_value: Campo que de igual forma al anterior, se compone de un tipo de dato INT y una longitud de 3 dígitos como máximo para almacenar el costo por minuto de parqueo para una moto.

hour_value: Campo cuyo tipo de dato es un INT. La longitud del mismo tiene una extensión de máximo 4 dígitos. Su función es almacenar el costo de cada hora de parqueo de cualquier vehículo.

day_value: Campo que recibe el valor correspondiente al costo de un día de parqueadero para cualquier tipo de vehículo. Su tipo de dato es un INT, y su longitud es de 5 dígitos.

month_value: Campo que recibe el valor correspondiente al costo de un mes de parqueadero para cualquier tipo de vehículo. Su tipo de dato es un INT, y su longitud es de 6 dígitos.

type_of_vehicle: Campo de texto que recibe el tipo de vehículo al que un parqueadero presta servicio. Su tipo de dato es VARCHAR y su longitud es de 10 caracteres dado que las dos únicas opciones posibles son: Autos y Todos. Se deja un margen de 5 caracteres para posibles variaciones en el futuro.

owners: Este campo se encuentra vinculado a la tabla “owners” y se conoce como una clave foránea o llave externa. Su objetivo es dar a conocer a que compañía pertenece un parqueadero en particular.

La tabla “owners” se ha definido con el fin de agrupar a los propietarios de los parqueaderos y se han estimado los siguientes campos:

id_owner: Al igual que para la tabla “parkings”, se define este campo como el identificador de cada uno de los propietarios de los parqueaderos y se establece como llave

principal. Su tipo de dato es entero y su longitud comprende 11 digitos automáticamente generados por la característica AUTO_INCREMENT aplicada.

owner_name: Campo que contiene el nombre del propietario para determinado parqueadero. Su tipo de dato es VARCHAR y la longitud de éste es de 80 caracteres en procura de poder abarcar nombres extremadamente extensos.

Por último, dado que las tablas de una base de datos se pueden relacionar entre sí con el fin de mantener la integridad de los datos en ella, se puede apreciar una relación desde la tabla “owners” hacia la tabla “parkings” que se define como “uno a varios”. Esto último quiere decir, que para cada propietario en la tabla de los propietarios, pueden existir muchos parqueaderos, pero no en sentido contrario. En otras palabras un propietario puede tener muchos parqueaderos, pero un parqueadero no puede tener muchos propietarios. Es determinante dejar claro ese aspecto y tener en cuenta el sentido en el que va la relación.

Dejando esto claro, obtenemos el siguiente diagrama mejor conocido técnicamente como “schema” el cual representa lo previamente mencionado. (ver Figura 18)

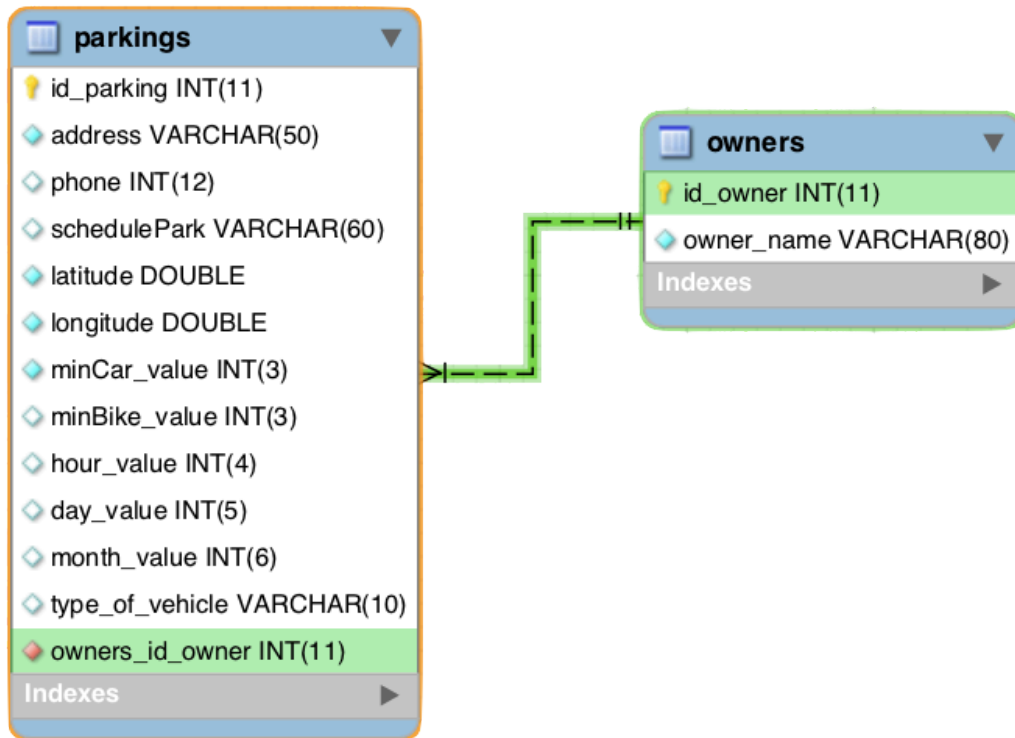


Figura 18. Modelo Entidad-Relación iParqueo
(Adaptada por el autor)

9.3 Arquitectura del Prototipo

Partiendo del hecho en el cual el prototipo ha sido construido haciendo uso de tecnologías web y posteriormente migrado a un entorno nativo, se hace necesario presentar ambas perspectivas por medio de un diagrama individual, que entregue un poco más de claridad sobre el funcionamiento del mismo.

De esta forma se logra una mejor comprensión del proceso y se deja evidencia de cómo se implementa PhoneGap para poder obtener una aplicación nativa por medio de una descripción gráfica en la que se muestran los componentes que intervienen en el proceso y la forma en la que éstos se comunican para solucionar las distintas peticiones. De igual forma, se pueden apreciar fácilmente las distintas tecnologías aplicadas en la elaboración de iParqueo y el contexto dentro del cual cada una ejerce su rol para dar vida al prototipo.

9.3.1 Arquitectura Web

El diagrama que representa ésta arquitectura, (ver Figura 19) persigue el objetivo de comunicar de forma clara el funcionamiento del prototipo iParqueo en un contexto web.

Todo parte del navegador web, que en iOS se conoce como Safari. En él, se ingresa la dirección web que conduce al lugar en el que se encuentra alojado el prototipo. Esta comunicación se realiza a través de algo que se conoce como un protocolo que en este caso se denomina http (Hypertext Transfer Protocol) y que permite recuperar el lugar de la web que contiene iParqueo. A partir de este momento, es clave saber diferenciar entre dos aspectos puntuales: el cliente y el servidor. Cuando a través del protocolo http se genera la solicitud para acceder al sitio web, safari obtiene los archivos del prototipo los cuales se ejecutan directamente en el dispositivo del usuario dentro del navegador, es decir, desde el lado del cliente.

El lado del cliente contiene archivos html, css y javascript que, en su orden, definen la estructura, presentación e interactividad del prototipo. Así mismo, existen plugins basados en javascript que permiten realizar, en su mayoría, las interacciones usuario-sistema, es decir, aquellas acciones para las que el prototipo se encuentra programado a resolver. Del lado del servidor se suelen encontrar las labores más pesadas, en otras palabras, aquellos procesos que requieren de una máquina con mejores características en términos de hardware o de potencia de cómputo.

Es así como se puede apreciar como el cliente se comunica con el servidor por medio de una tecnología conocida como AJAX (Asynchronous Javascript and XML) la cual permite obtener información puntual del lado del servidor con el fin de agilizar la respuesta del lado del cliente y permitir una experiencia de usuario más agradable. De la misma forma que existen tecnologías del lado del cliente, del lado del servidor se cuenta con una conocida como PHP la cual es empleada para realizar la solicitud de parqueaderos a la base de datos y retornar la información solicitada por medio de un eficiente formato de transferencia de datos conocido como JSON (Javascript Object Notation). Lo que sucede a continuación del lado del cliente tiene que ver con la manipulación de la información retornada y su correspondiente actualización en pantalla.

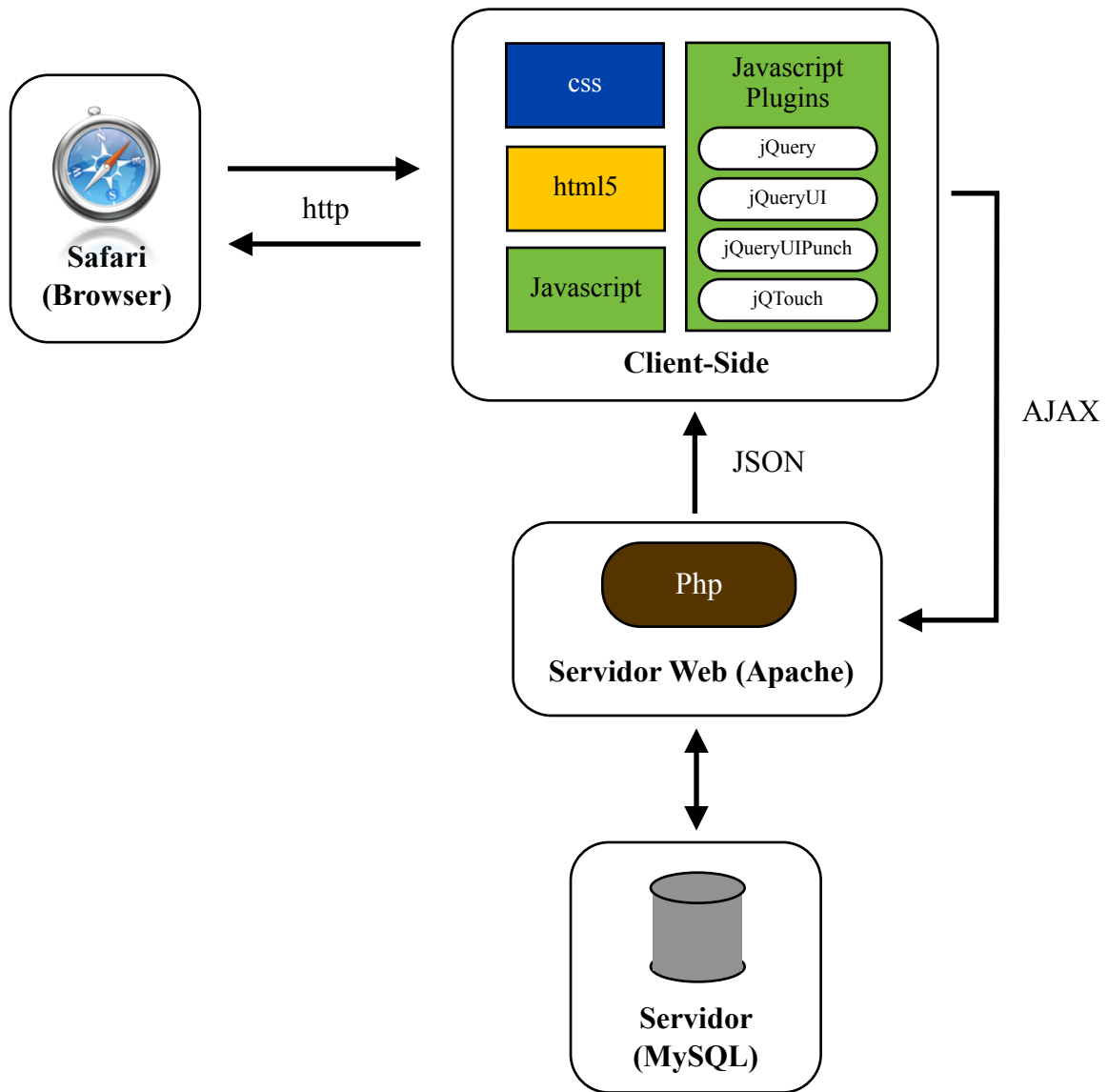


Figura 19. Arquitectura prototipo - Web App
(Adaptada por el autor)

9.3.2 Arquitectura Nativa (PhoneGap)

Por medio de esta arquitectura, (ver Figura 20) se puede apreciar la forma en la que PhoneGap toma la aplicación generada con tecnologías web y la “empaqueta” dentro de un navegador que contiene los plugins necesarios para poder acceder a las características de hardware del dispositivo móvil.

El escenario respecto a la arquitectura web cambia de forma sutil, pues ahora no hace falta emplear safari e ingresar la dirección web en la que se encuentra la app, sino que por el contrario, se accede a la misma como a cualquier otra aplicación nativa de iOS, tocando el icono que la identifica, el cual una vez se activa, vale la pena aclarar, realmente abre un navegador que contiene y ejecuta los archivos web generados desde un principio. A pesar de que ahora la app se encuentra “instalada” en el dispositivo móvil, se mantiene la filosofía cliente-servidor pues ésta continúa solicitando el proceso al archivo PHP de la misma forma en la que se detallaba con la arquitectura anterior, recuperando los parqueaderos de la base de datos para dar respuesta a la aplicación, sólo que ahora, el proceso es mucho más transparente para el usuario, pues la app no se está ejecutando desde el navegador web sino de forma nativa.

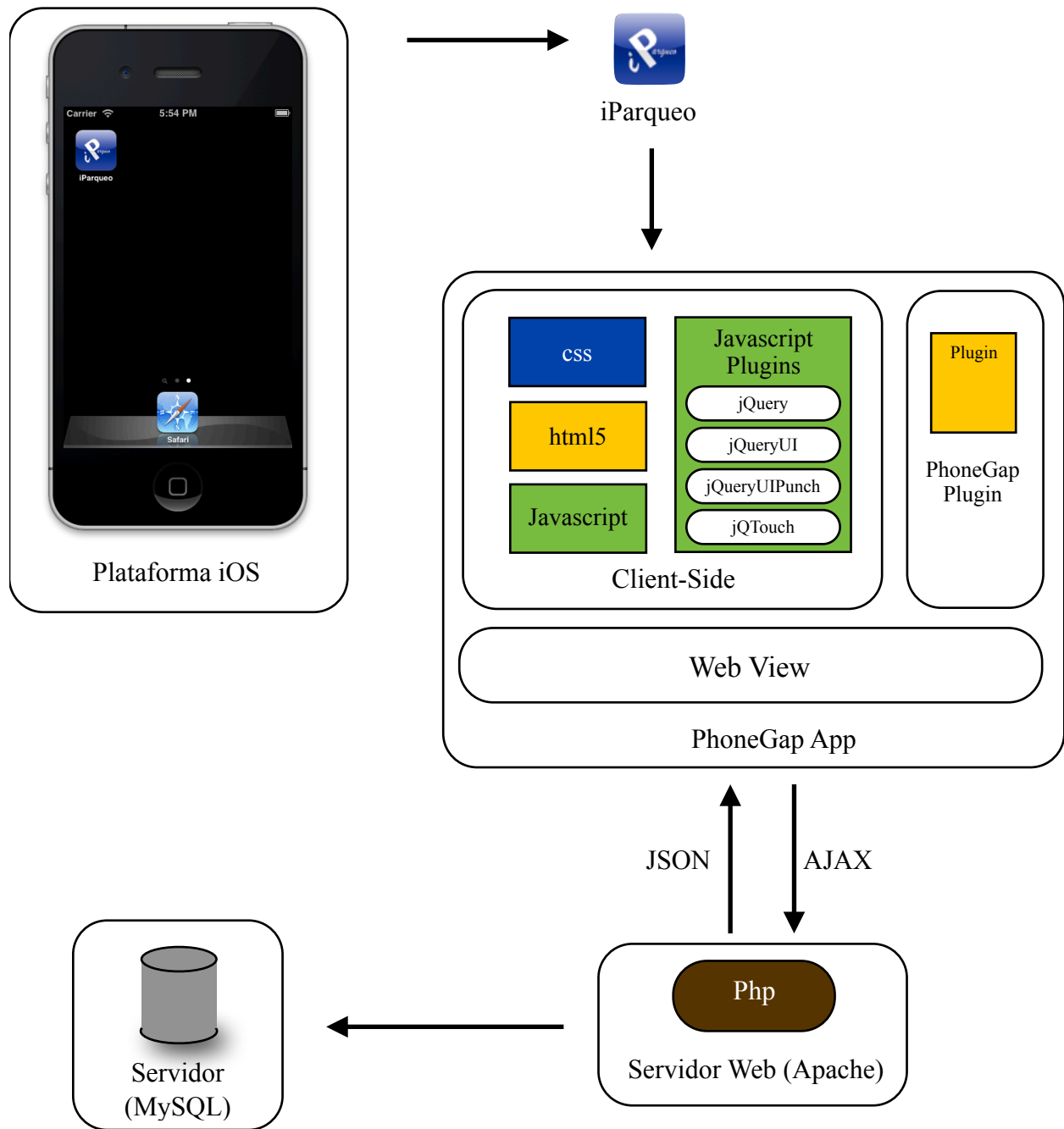


Figura 20. Arquitectura prototipo - PhoneGap
(Adaptada por el autor)

9.4 Diseño de Interfaz Gráfica

Para el diseño de la interfaz gráfica, se tienen en cuenta las dimensiones del iPhone incluido el último modelo, (iPhone 5) y las dimensiones del iPad mini y el iPad tradicional. Así mismo, se toman las resoluciones de las distintas pantallas para poder generar los componentes gráficos apropiados y cuidar la creación de la interfaz gráfica de la aplicación. (ver Figura 21)

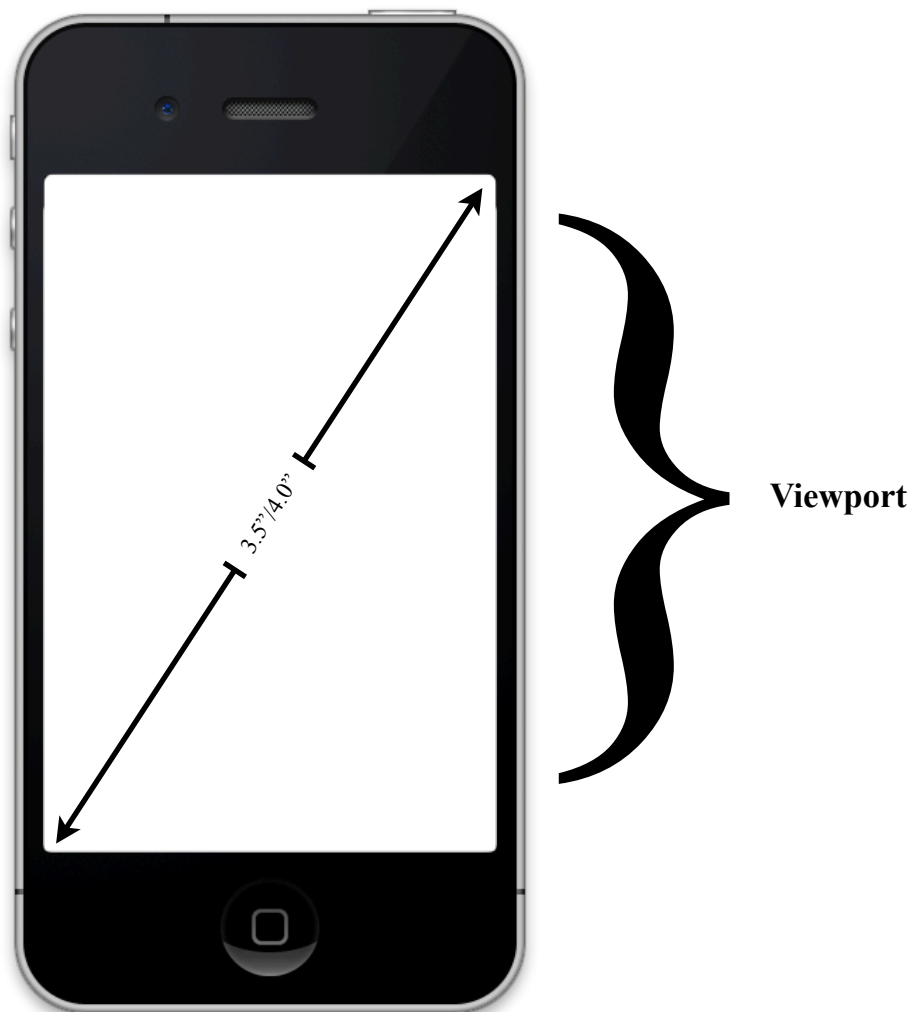


Figura 21. Plantilla iPhone - Dimensiones |
(Adaptada por el autor)

Uno de los puntos más importantes dentro del contexto gráfico de la app, es el icono que identifica a la misma. Se ha de tener especial cuidado en la construcción del mismo, ya que además de ser comunicativo y dar a conocer una idea sobre qué hace la aplicación, éste debe ser lo suficientemente llamativo para atraer la atención del usuario y destacar sobre la competencia en la App Store. (ver Figura 22)

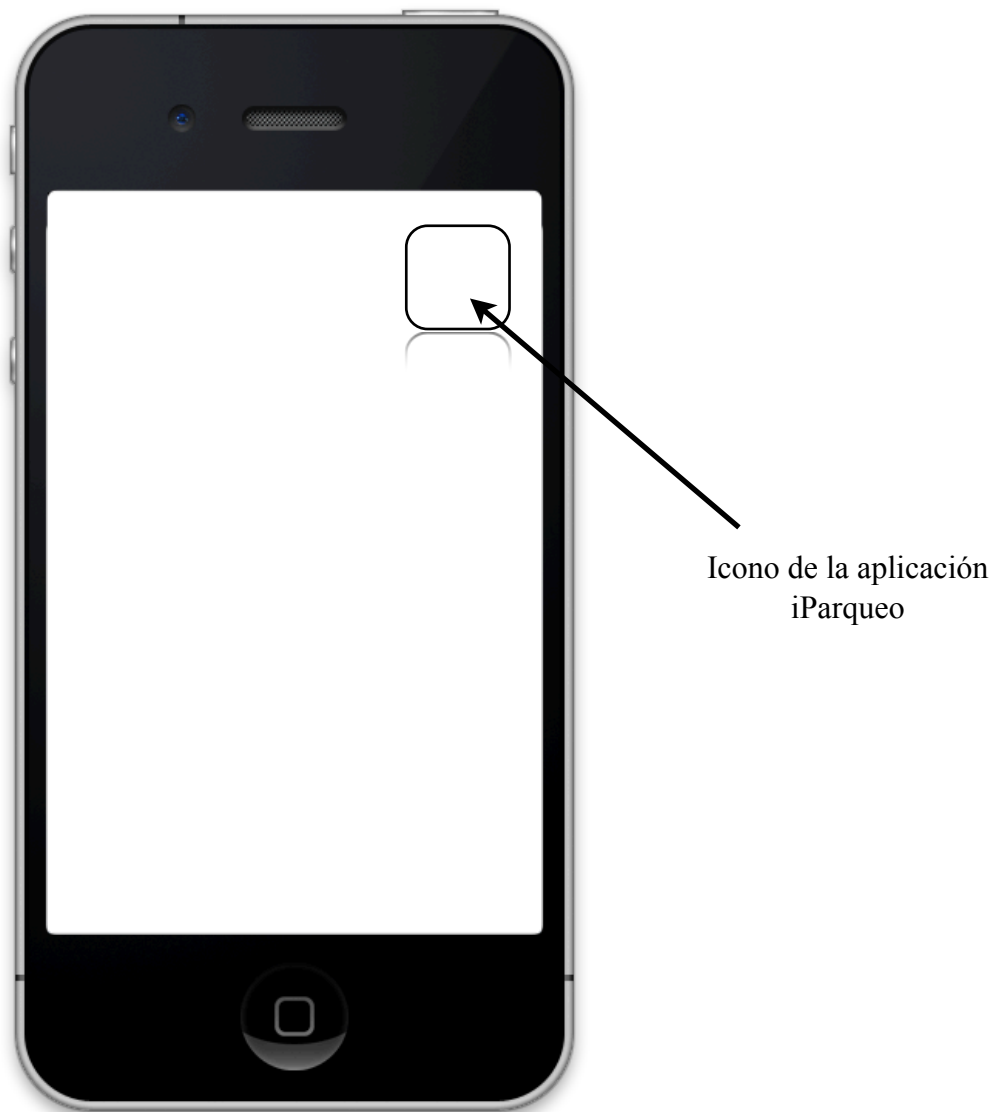


Figura 22. Plantilla iPhone - Mockup icono
(Adaptada por el autor)

Existe algo conocido como “Splash Screen” y consiste en una imagen que se muestra al usuario justo en el momento en el que la aplicación se ejecuta e inicia su carga. Si bien el propósito de la misma es “timar” al usuario haciéndole creer que la aplicación carga de forma instantánea, ésta, por lo general, es utilizada para mostrar el logotipo de la compañía desarrolladora, o para ampliar la información de su funcionamiento. En el caso particular, ampliaremos la información sobre iParqueo. (ver Figura 23)

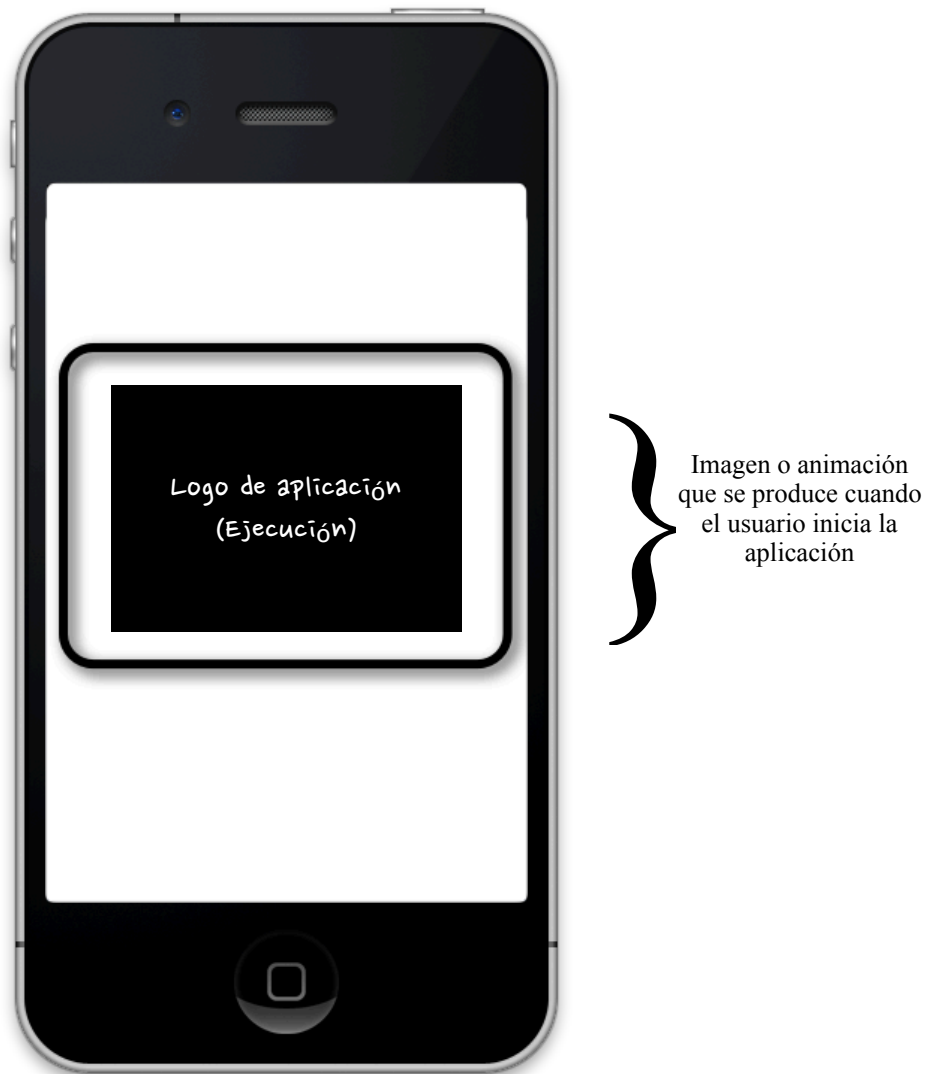


Figura 23. Plantilla iPhone - Splash Screen (Adaptada por el autor)

Tomando como base los requerimientos no funcionales previamente mencionados, se realiza un diseño de aplicación que cumpla con las expectativas de usuario mediante una interfaz gráfica sencilla y componentes gráficos llamativos e intuitivos en su manipulación. Adicionalmente, y para evitar confusiones en la comprensión que obtenga el usuario respecto al funcionamiento de la app, se incorpora texto explicativo dejando explícita la operación de la misma. (ver Figura 24)

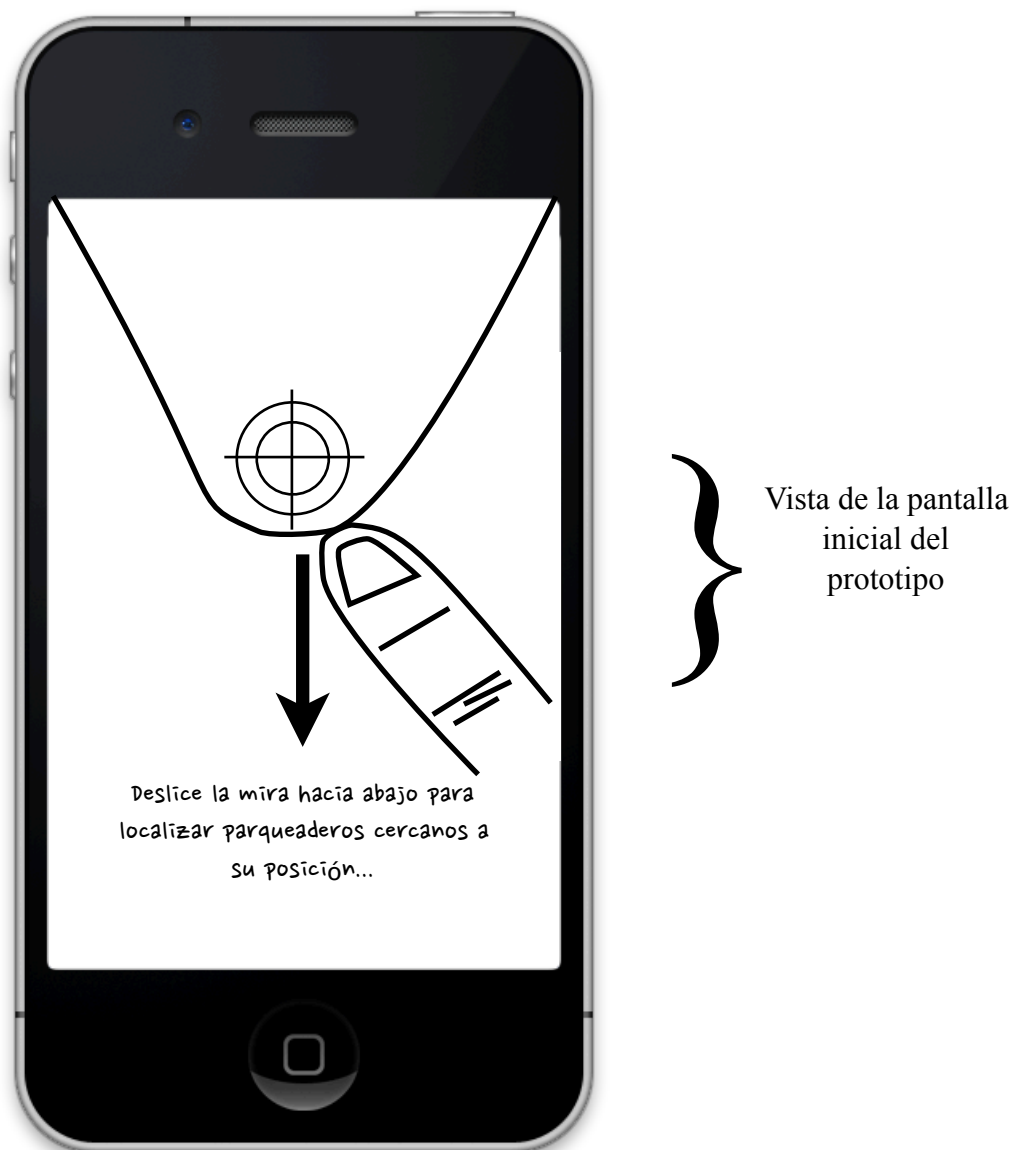


Figura 24. Plantilla iPhone - Mockup Vista Inicial
(Adaptada por el autor)

La vista en la que se muestran los parqueaderos cercanos a la ubicación del usuario debe ocupar la mayoría del espacio en pantalla, permitiéndolo una navegación agradable a través del mapa. Se habilitan controles en la parte superior de la pantalla con el fin de ofrecer un poco más de flexibilidad al momento de hacer uso de la app. (ver Figura 25)



Figura 25. Plantilla iPhone - Mockup Vista Localización
(Adaptada por el autor)

Una de las alternativas presentadas al usuario es la clasificación de los parqueaderos ubicados de acuerdo a distintos criterios útiles de búsqueda. Se busca representar esta vista de forma limpia entregando la información completa para que el usuario tenga argumentos suficientes para tomar una decisión. Es así como se toma la decisión de mostrar en una lista cada uno de los parqueaderos junto con su información anexa. (ver Figura 26)



Figura 26. Plantilla iPhone - Mockup Vista Clasificación Parqueaderos
(Adaptada por el autor)

Otra de las alternativas previstas se basa en el cálculo de costo del parqueadero elegido. Se pretende generar esa información de forma práctica de tal manera, que el usuario sólo tenga que ingresar el valor del minuto del parqueadero e iniciar el cronómetro a partir del momento en que el usuario abandona el parqueadero, hasta el instante en el que se encuentra de vuelta en el lugar para retirar su vehículo. (ver Figura 27)



Figura 27. Plantilla iPhone - Mockup Vista Cálculo Costo de Parqueo (Adaptada por el autor)

De acuerdo al diseño previamente detallado, se hace una descripción de los controles que tendrá la aplicación (ver Figura 28) basados en los siguientes campos:

ID: Identifica el control dentro de los bosquejos de diseño.

Control: Identifica el tipo de control con el que interactúa el usuario.

Función: Define la funcionalidad que tendrá el control.

Descripción: Especifica detalladamente el proceso realizado una vez se activa el control.

Descripción de los controles de la aplicación			
ID	Control	Función	Descripción
a	Imagen	Localizar parqueaderos	Al deslizarse en sentido vertical hasta abarcar 380 pixeles dipara los métodos correspondientes para localizar parqueaderos cercanos a la posición del usuario.
b	Botón	Retroceder a la vista anterior	Al presionarse, devuelve al usuario a la pantalla inmediatamente anterior.
c	Combo seleccionable	Clasificar parqueaderos	Despliega una lista con los criterios mencionados a continuación, y una vez se realiza la selección de uno de éstos, se clasifican y muestran los parqueaderos listados: <ul style="list-style-type: none"> • Tarifa minuto • Tarifa hora • Cercanía
d	Botón	Avanzar a vista de cálculo de consumo en parqueadero	Conduce a la vista que contiene los controles necesarios para calcular el costo de parqueo.
e	Campo de texto	Recibir valor minuto de parqueadero	Campo de texto que recibe el valor del minuto para calcular el costo de parqueo. Detecta cambios en el contenido o información ingresada.
f	Elemento de lista	Mostrar tiempo transcurrido	Elemento que se activa mostrando un temporizador una vez el usuario ha pulsado el botón “Iniciar/Detener”. El objetivo es indicar al usuario el tiempo que va transcurriendo desde el momento en el que éste parquea.
g	Elemento de lista	Mostrar el costo de parqueo	Elemento que se actualiza de acuerdo al valor del minuto ingresado por el usuario y al tiempo transcurrido de parqueo. Se activa una vez el usuario detiene el temporizador por medio del botón “Iniciar/Detener”.
h	Botón	Calcular costo de parqueo	Bóton cuya función varia de acuerdo al estado en el que se encuentre: “Iniciar”: Limpia todas las variables involucradas en el cálculo del costo de parqueadero y activa el temporizador. “Detener”: Detiene el cronómetro y dispara la función que calcula el costo de parqueo validando que el usuario haya ingresado el valor del minuto correspondiente.

Figura 28. Descripción de controles de interfaz gráfica
(Adaptada por el autor)

9.5 Diagramas de secuencia

Los diagramas de secuencia ejemplifican los pasos que son llevados a cabo en un orden lógico, para plasmar en detalle el comportamiento de cada una de las funcionalidades del prototipo.

Cada línea vertical representa lo que se conoce como un actor, el cual ejecuta o cumple una acción dentro del proceso que se está llevando a cabo. De cada actor se desprende una línea horizontal que tiene un sentido el cual indica la forma en la que viaja el flujo de la actividad y es acompañada por un texto explicativo de acuerdo al paso que se esté realizando.

Estos diagramas creados en UML, permiten una comprensión a fondo de cada caso de uso sobre un escenario perfecto, dentro del cual todo funciona de acuerdo a lo planificado. A continuación se presentan los 3 diagramas de secuencia elaborados para cada caso de uso.

9.5.1 Localización de parqueadero

El diagrama de secuencia para la localización de parqueaderos comprende 4 actores fundamentales: el usuario, el cliente (el prototipo iParqueo), el servidor y la base de datos. La comunicación parte, obviamente, del usuario que al deslizar la mira de forma vertical, está comunicándose con el cliente, solicitándole los parqueaderos relativos a su ubicación. Más allá de lo que el usuario recibe en pantalla, de fondo, tienen lugar varios procesos previos a la entrega de la información. Para empezar, existe una validación del espacio, en cantidad de pixeles, que ha recorrido la mira para corroborar que ésta ha sido ubicada justo encima del objetivo denotado con una P en alusión a un parqueadero.

Una vez verificado esto, se capturan las coordenadas de localización del dispositivo desde el cual se está solicitando la petición, y se comparan con las áreas dentro de las cuales presta cobertura la aplicación, (Emaús, Quinta Camacho y Chapinero Norte) para así comunicar al usuario la respuesta apropiada. Posteriormente, se procede a solicitarle al servidor los parqueaderos relativos al área por medio de un petición GET, método que se emplea cuando no se pretende modificar información del lado del servidor, sino simplemente obtener ciertos datos puntuales; el servidor, realiza las validaciones pertinentes y ejecuta una consulta a la base de datos obteniendo los parqueaderos. Aplicando un formato para transportar la información, el servidor entrega respuesta al cliente y éste la formatea permitiendo desplegarla visualmente en un mapa que finalmente es mostrado al usuario. (ver Figura 29)

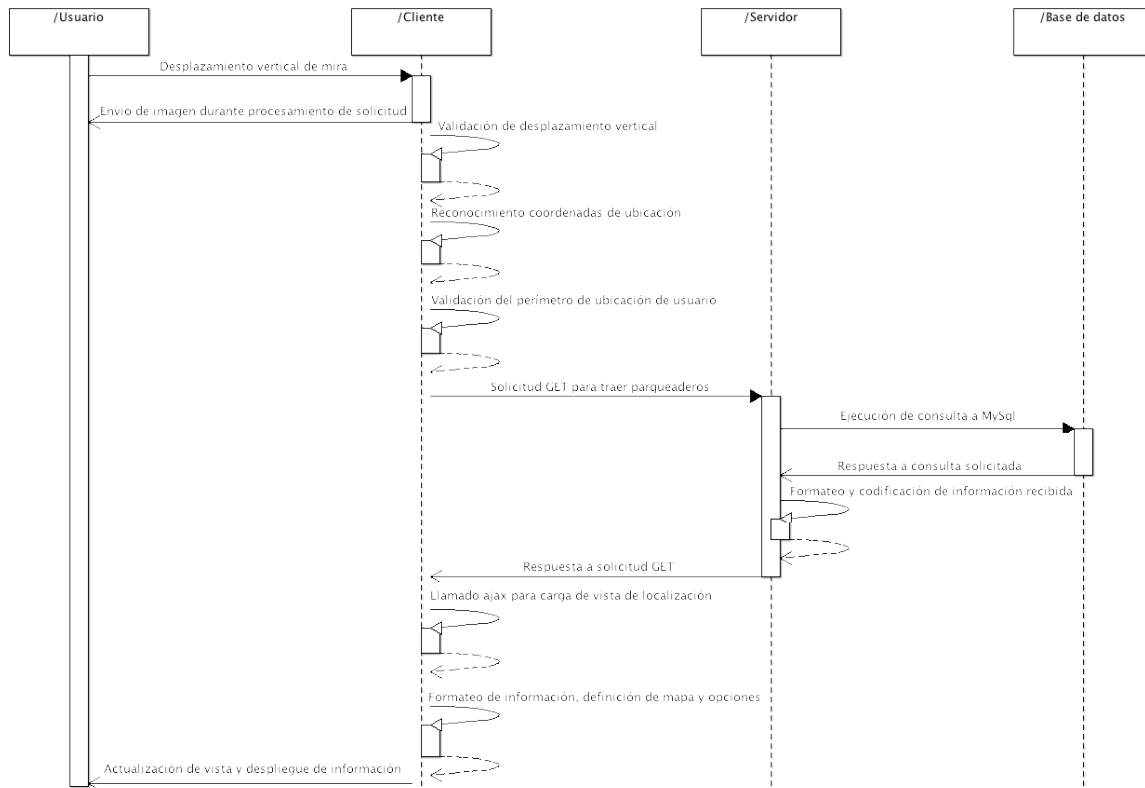


Figura 29. Diagrama De Secuencia - Localización de Parqueadero (Adaptada por el autor)

9.5.2 Clasificación de parqueaderos

El diagrama de secuencia concerniente a la clasificación de los parqueaderos obtenidos, tiene un flujo lógico que se comporta de la siguiente forma: Identificados los 2 actores que intervienen en esta funcionalidad, (Usuario y Cliente) una vez el usuario toca el botón “clasificar”, el cliente informa las alternativas de organización que pueden ser aplicadas a los parqueaderos, éstas son valor minuto, valor hora y por último cercanía. El usuario elige el criterio de clasificación que se acomode a sus necesidades y cierra la ventana que despliega las opciones. El cliente detecta que el usuario ha elegido una opción y abandonado la ventana de alternativas, e inmediatamente, procede a organizar los parqueaderos que lista en una vista diferente a la que actualmente tiene desplegada en pantalla el usuario para finalmente, mostrarla solucionando la petición del usuario. (ver Figura 30)

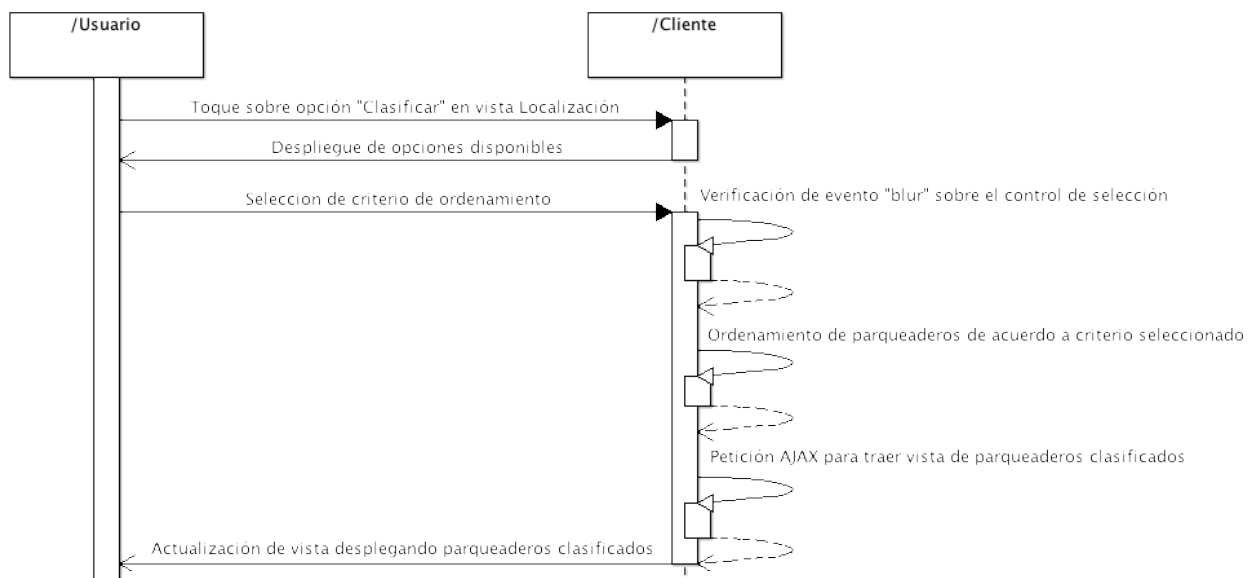


Figura 30. Diagrama De Secuencia - Clasificación de Parqueaderos (Adaptada por el autor)

9.5.3 Calcular costo de parqueo

La funcionalidad que comprende calcular el costo de parqueo, se ve reflejada en este diagrama de secuencia de la siguiente forma: De nuevo, como en el caso anterior, intervienen 2 únicos actores, el usuario y el cliente. El usuario toca el botón “costo” en la vista que contiene el mapa y los parqueaderos desplegados, y el cliente conduce a una nueva vista en la que el usuario puede apreciar un campo en el cual se ingresa el valor del minuto para el parqueadero elegido y un botón que activa y desactiva un cronómetro cuyo objetivo es contabilizar el tiempo que transcurre durante el tiempo de parqueo. El usuario ingresa los datos solicitados y toca el botón denominado “Iniciar”, en ese momento, el cliente captura la información y descubre un contador de minutos y segundos que se van actualizando conforme va avanzando el tiempo.

Cuando el usuario se dispone a retirarse del parqueadero, toca de nuevo el botón que ahora se denomina “Detener”, acción por la que el cliente se ve notificado, realizando un proceso de verificación sobre los datos suministrados y desplegando la información del valor a pagar en pesos colombianos de acuerdo al tiempo transcurrido. (ver Figura 31)

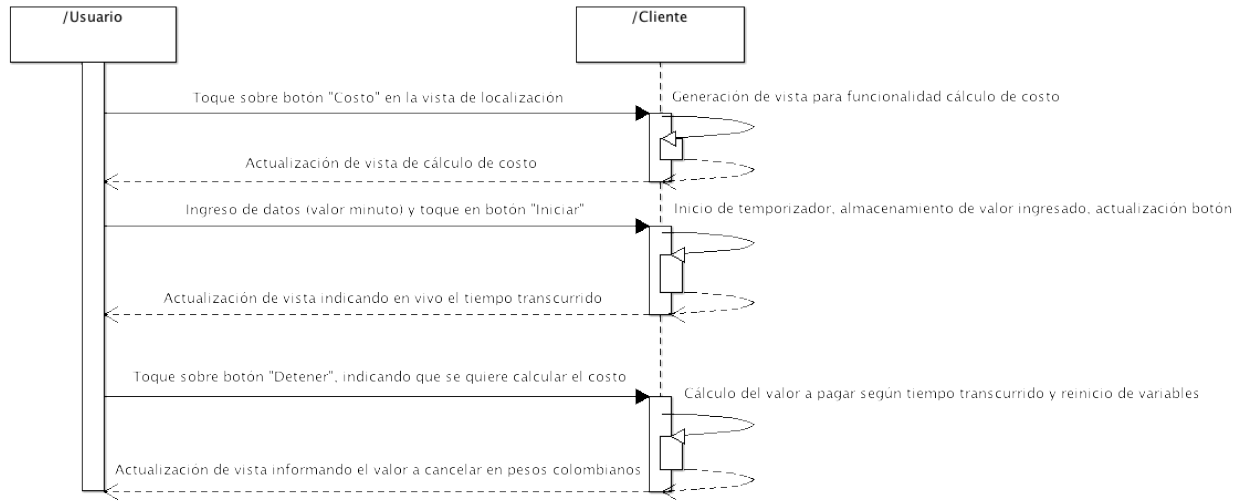


Figura 31. Diagrama De Secuencia - Calcular Costo De Parqueo
(Adaptada por el autor)

10. IMPLEMENTACIÓN DE LA APLICACIÓN

La construcción de la aplicación inició en su parte gráfica con la creación del icono de la misma, el cual se generó a partir de un editor de imágenes llamado Pixelmator. El objetivo es que el icono comunique de qué se trata la aplicación, motivo por el cual sobresale una P mayúscula como el estándar de señalización de un parqueadero, que hace parte del nombre que se eligió para la misma: iParqueo. El color elegido se fundamenta en el mismo contexto de señalización previamente mencionado.

El reto técnico entonces, es la creación de un icono que cumpla con las especificaciones de cada uno de los dispositivos iOS; para ello, se genera un único componente que se escala, variando en sus dimensiones y resoluciones, para dar soporte a los dispositivos mostrados relacionados a continuación. (ver Tabla 4)

iPhone / iPod Resolución estándar		iPhone / iPod Retina Display / iPhone 5		iPad Retina Display	
Dimensiones	57 x 57	Dimensiones	114 x 114	Dimensiones	144 x 144
Resolución (DPI)	162	Resolución (DPI)	326	Resolución (DPI)	264

Tabla 4. Comparación entre iconos principales del prototipo
(Adaptada por el autor)

De igual forma, se generan dos archivos de estilos (css) para cubrir las distintas resoluciones y tamaños de pantalla del iphone, iPad mini, y iPad. Siguiendo esta línea, se procede a crear la imagen de ejecución de la aplicación, la cual realza un poco más el significado de la misma, dando paso a la vista de inicio. De igual forma, se genera una única imagen que cada dispositivo escalará automáticamente a conveniencia. (ver Figura 32)



Figura 32. “Splash Screen” para prototipo
(Adaptada por el autor)

10.1 Vistas Prototipo

Las vistas del prototipo son producto del trabajo realizado en el diseño de la interfaz gráfica, que en su construcción, se pretende sean lo más parecido posible a lo planeado con el fin de mantener una misma línea de trabajo. Para cumplir con este objetivo, se hace uso de una herramienta de diseño gráfico conocida como Pixelmator que, junto con la ayuda de varios plugins de jQuery, permiten simplificar la labor y como valor agregado, centrar la atención en la lógica del prototipo.

Con Pixelmator se contruyen los componentes gráficos de la app, mientras que los plugins de jQuery permiten generar otro tipo de componentes enfocados a la navegación entre las distintas vistas y el comportamiento de éstos relacionado al uso del usuario.

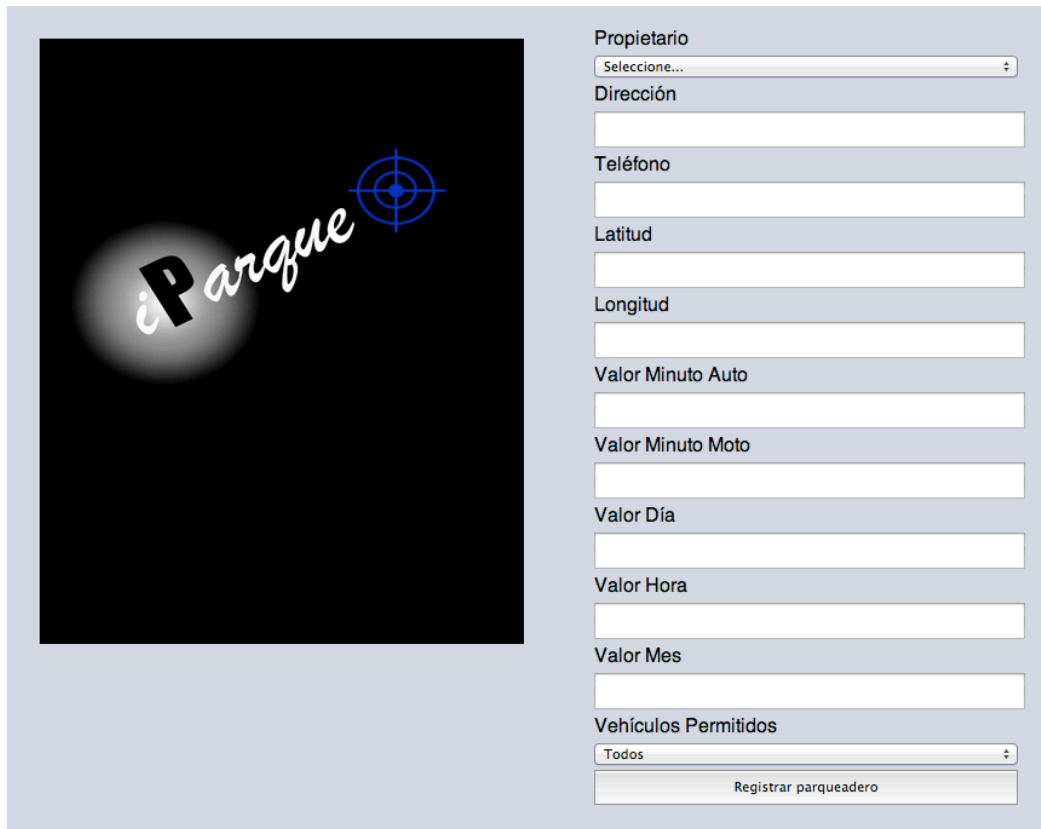
Dentro de las grandes bondades que aporta hacer uso de plugins javascript, en particular jQTouch, encontramos:

- Gestión automática de la interfaz en el cambio de orientación del dispositivo. (El diseño de la app se ajusta a los cambios en las dimensiones en pantalla).
- Simplicidad, practicidad y buena calidad en la creación de los distintos componentes gráficos como botones, formularios, listas, etc.
- Fácil implementación de animaciones que se asemejan con gran precisión a aquellas que se pueden obtener con el enfoque nativo.

- Métodos simplificados para realizar peticiones AJAX de forma programática. (Es muy sencillo obtener información concreta del servidor).

Dado que se debe suministrar una interfaz gráfica para alimentar la aplicación por medio de un módulo administrador, se ha resuelto hacer uso del tradicional formulario web (Ver Figura 33) para ingresar cada uno de los datos necesarios y así cumplir con las exigencias de los usuarios frente a la información que desean tener de cada parqueadero. El formulario comprende los siguientes campos autoexplicativos:

- **Propietario:** El nombre de la compañía propietaria de la cadena de parqueaderos.
- **Dirección:** La dirección exacta del parqueadero.
- **Teléfono:** El teléfono de contacto del parqueadero.
- **Latitud:** Las coordenadas de latitud del parqueadero.
- **Longitud:** Las coordenadas de longitud del parqueadero.
- **Valor Minuto Auto:** El costo en números del minuto de parqueo para automóviles.
- **Valor Minuto Moto:** El costo en números del minuto de parqueo para motos.
- **Valor Día:** El costo de parqueo para un día completo.
- **Valor Hora:** El costo de parqueo para una hora.
- **Valor Mes:** El costo de parqueo para un mes completo.
- **Vehículos Permitidos:** El tipo de vehículo que el parqueadero permite ingresar, sólo pueden existir dos opciones, “Todos” que comprende automóviles y motos o “Autos”, que restringe el ingreso a las motos.



The image shows a web form for registering a parking spot. On the left is a black square with the 'iParqueo' logo in white and a blue target icon. On the right is a light blue form with the following fields:

- Propietario: Seleccione... (dropdown)
- Dirección: [input field]
- Teléfono: [input field]
- Latitud: [input field]
- Longitud: [input field]
- Valor Minuto Auto: [input field]
- Valor Minuto Moto: [input field]
- Valor Día: [input field]
- Valor Hora: [input field]
- Valor Mes: [input field]
- Vehículos Permitidos: Todos (dropdown)
- Registrar parqueadero (button)

Figura 33. Formulario web para el ingreso de nuevos parqueaderos a iParqueo
(Adaptada por el autor)

Se ha dispuesto la validación de cada uno de los campos asociados a los datos requeridos para que el usuario los diligencie en su totalidad. De esta forma, aquel campo que el usuario deje vacío en medio del proceso, mostrará un mensaje no invasivo a la derecha del mismo, con el fin de alertar la necesidad de no dejar ningún espacio en blanco. En el momento en el que se presione el botón denominado “Registrar parqueadero” se realizará una nueva validación campo por campo para garantizar que la base de datos reciba la información adecuada.

10.1.1 Localización Parqueaderos

Es la vista inicial de la aplicación, se compone de una mira que se desliza en sentido vertical con el fin de alcanzar el objetivo representado con una “P” que simboliza un parqueadero. En la parte inferior se observa un cuadro de texto dentro del cual se indica el procedimiento que debe realizar el usuario. Con prototipos iniciales se identificó una falta de comprensión en la acción a realizar, por lo cual se agregó una pequeña “animación” en la que aparece un dedo índice que simula una pulsación sobre la mira a deslizar, dando un poco más de claridad. (ver Figura 34)



Figura 34. Vista inicial de prototipo
(Adaptada por el autor)

En el escenario perfecto, la aplicación debe solicitar permiso para hacer uso de la ubicación del usuario (básicamente permitir el acceso al GPS) y así proceder a la siguiente vista. Caso contrario arrojará mensajes de error que comunicarán al usuario la razón por la cual no se ha logrado proceder con la petición. En el caso inicial, se desplegará una alerta con dos opciones (No permitir/Ok) en las que el usuario determina que hacer. Para el segundo, (como se puede apreciar en la Figura 35) se muestra un texto en la parte inferior de la vista describiendo la razón por la cual no es posible realizar el proceso de solicitar parqueaderos.



Figura 35. Escenarios en respuesta a la petición de usuario
(Adaptada por el autor)

Una vez obtenidas las coordenadas de usuario, la aplicación procede a mostrar la vista de localización en la cual se marca la ubicación del usuario y los parqueaderos cercanos simbolizados con una “P”. Cada icono representando al parqueadero, despliega la información del mismo al ser tocado por el usuario. Así mismo, se ponen a disposición del usuario los controles para clasificar los parqueaderos y calcular el costo de parqueo. (ver Figura 36)



Figura 36. Vista Localización prototipo iParqueo (Adaptada por el autor)

Como característica adicional, se ha incluido la ruta representada por un trazo de color azul, que se marca desde la posición actual del usuario, hasta el parqueadero seleccionado facilitando la aproximación a éste (Nótese la Figura 37). Esta característica se encuentra disponible de forma automática una vez el usuario cierra el cuadro de diálogo (parte superior derecha) que despliega cada uno de los parqueaderos.



Figura 37. Trazo de ruta - Vista Localización prototipo (Adaptada por el autor)

10.1.2 Clasificación de Parqueaderos

Para la clasificación de parqueaderos se han determinado 3 posibles criterios de filtro basados en las necesidades del usuario: **tarifa minuto**, **tarifa hora** y **cercanía**. Una vez el usuario selecciona una de estas opciones, se muestran en orden descendente los parqueaderos organizados de acuerdo a la selección con su respectiva información. (ver Figura 38)



Figura 38. Vista Clasificación De Parqueaderos (Adaptada por el autor)

10.1.3 Cálculo de Costo de Parqueo

La construcción del último requerimiento se concibe de la siguiente forma: el usuario activa un cronómetro indicando el valor del minuto de acuerdo al parqueadero en el que se encuentre. En el momento en el que procede a retirarse del mismo, éste detiene el cronómetro y revisa la información en pantalla permitiéndole conocer el valor aproximado a cancelar. (ver Figura 39)

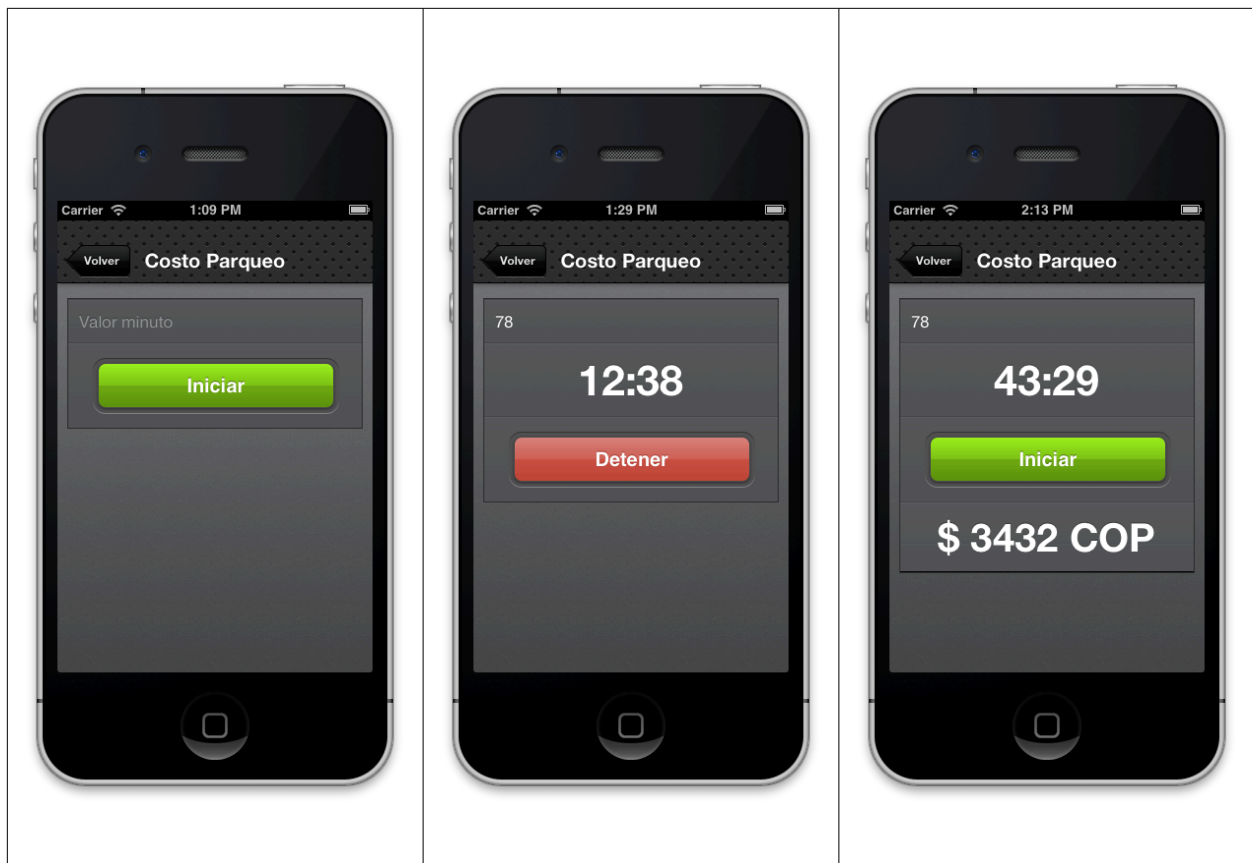


Figura 39. Vista Cálculo de costo
(Adaptada por el autor)

10.2 Delimitación del área de parqueaderos públicos

Dada la extensión de la ciudad de Bogotá, y partiendo del hecho en el que se está elaborando un prototipo para identificar una oportunidad dentro del mercado de aplicaciones móviles, se toma la decisión de limitar la cobertura de la aplicación a 3 barrios ubicados en la localidad de Chapinero. La selección de estos barrios, se fundamenta con el argumento de que esta zona comprende el centro financiero de la ciudad, pues dentro de sus inmediaciones se encuentran la bolsa de valores y grandes compañías multinacionales que hacen de esta área un lugar muy concurrido con elevados flujos de tránsito vehicular.

Esta elevada cifra de vehículos transitando sobre el área señalada, presenta una gran alternativa para el comercio y por ende ofrece una oportunidad para el prototipo tratado, a causa de los lugares disponibles para parquear los distintos vehículos que llegan a la zona.

10.2.1 EMAUS



Figura 40. Barrio Emaús
 (Tomada de <http://beta.mapas.com.co/>)

Barrio ubicado en el área oriental de la ciudad de Bogotá, (Ver Figura 40) que limita al noroccidente con la carrera 9na en su cruce con la calle 72, al nororiente entre la carrera 5ta y 4ta con calle 73, al suroriente entre la carrera 1ra y 2da con la calle 69A y al suroccidente entre la carrera 6ta y 5ta con calle 67.

10.2.2 QUINTA CAMACHO

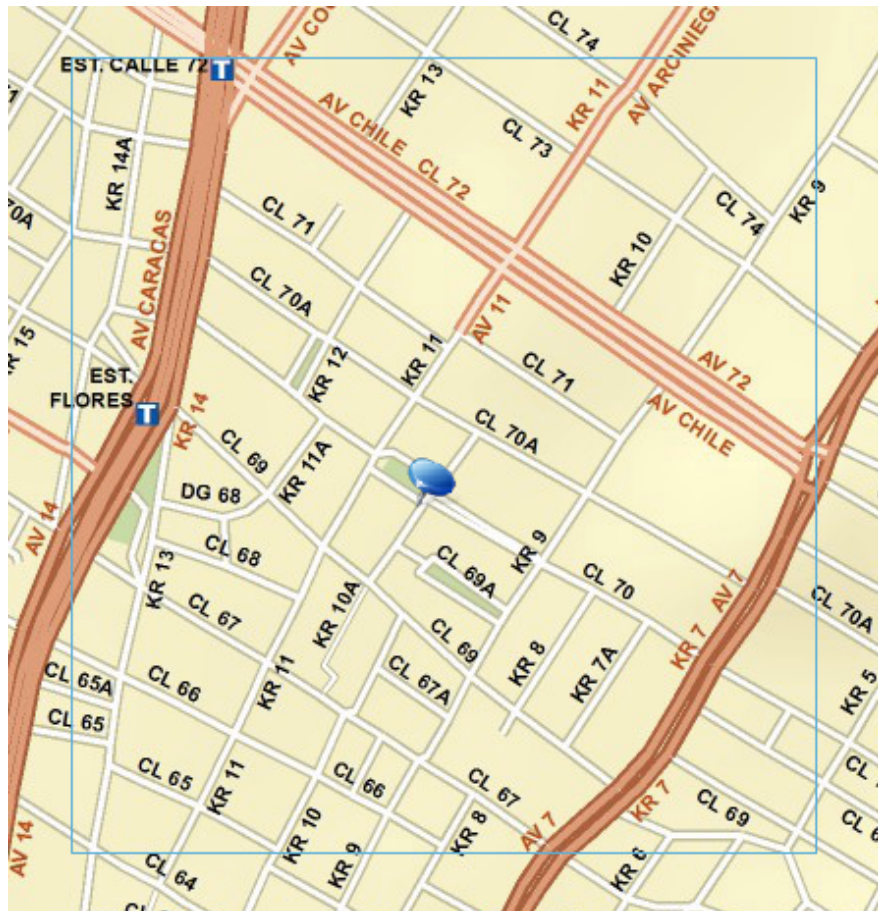


Figura 41. Barrio Quinta Camacho
 (Tomada de <http://beta.mapas.com.co/>)

Barrio ubicado en el área oriental de la ciudad de Bogotá que limita al noroccidente con la carrera 15 en su cruce con la calle 71, al nororiente entre la carrera 9na y 10ma con calle 75, al suroriente con la carrera 5ta en su cruce con la calle 69 y al suroccidente con la carrera 12 con calle 64. (Ver Figura 41)

10.2.3 CHAPINERO NORTE

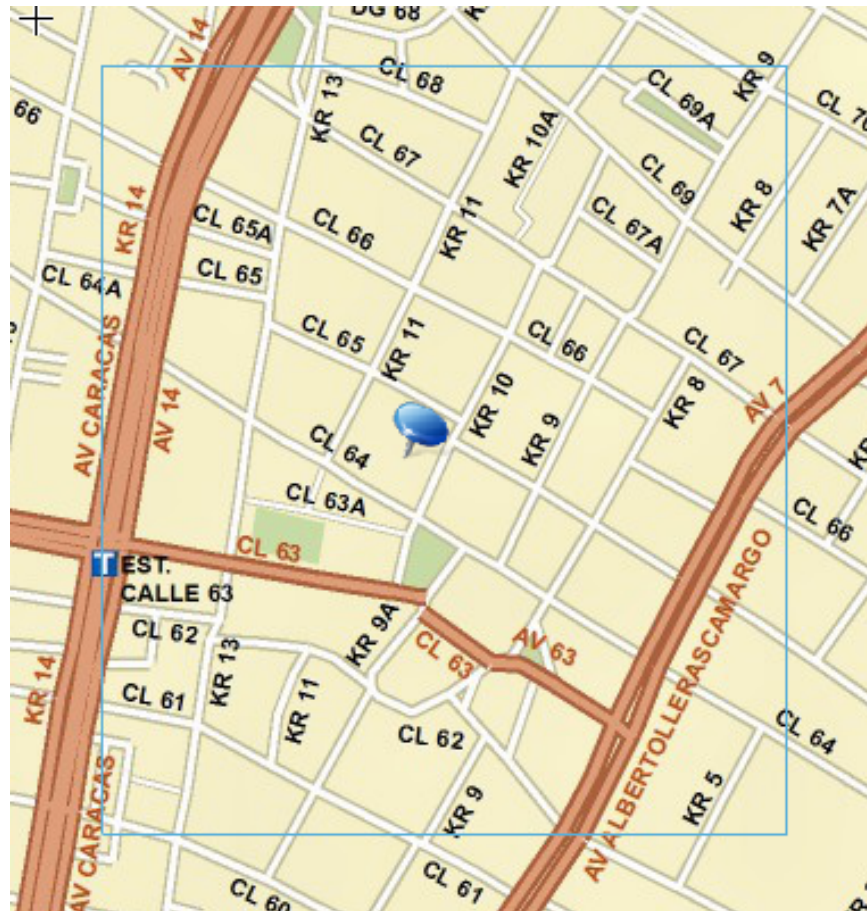


Figura 42. Barrio Chapinero Norte
(Tomada de <http://beta.mapas.com.co/>)

Barrio ubicado en el área oriental de la ciudad de Bogotá que limita al noroccidente con la carrera 15 en su cruce con la calle 66A, al nororiente con la carrera 9na en su cruce con la calle 70, al suroriente entre la carrera 5ta y 3ra en su cruce con la calle 63 y al suroccidente con la carrera 14 con calle 60. (Ver Figura 42)

10.3 Migración a entorno nativo

Como bien se mencionó previamente, es posible tomar una aplicación construida con herramientas no proporcionadas por Apple y migrala a un entorno nativo por medio de distintas tecnologías como PhoneGap.

El proceso es bastante sencillo. Básicamente, se trata de empaquetar toda la aplicación web dentro de un navegador generado en Objective-C por medio de XCode, el entorno de desarrollo provisto por Apple para el desarrollo de aplicaciones nativas. Para el caso de iParqueo, el procedimiento consistió en descargar la versión 2.6.0, descomprimirla y localizar la estructura para la plataforma iOS. Posteriormente, y siguiendo la guía provista por el sitio del desarrollador, se procedió a crear un proyecto que toma como base la estructura del directorio previamente mencionado y crea una copia en limpio alimentada por argumentos sencillos como el nombre de la aplicación y el nombre del paquete del proyecto.

Una vez dispuesta la estructura, se procede a ubicar la carpeta denominada “www” y a reemplazar su contenido con los archivos que conforman la web app. Hecho esto, el paso final se basa en elegir el proyecto creado, accederlo haciendo uso de XCode, y elegir el simulador de acuerdo al dispositivo en hardware y software (versión) que se pretenda soportar.

11. PRUEBAS

A lo largo del proyecto se realizaron distintas pruebas con el fin de garantizar que el comportamiento del prototipo fuera el adecuado en términos tanto de funcionalidad como de estabilidad.

11.1 Pruebas de funcionalidad

Tomando como base cada uno de los casos de uso, se procedió a evaluar la respuesta del prototipo de acuerdo a distintos escenarios de conectividad.

Los escenarios de conectividad disponibles actualmente en Bogotá son:

- **Edge**
- **3G**
- **Wifi**

Para todos los casos, se descarta el uso de la generación Edge dada la pobre respuesta obtenida por parte de ésta en términos de calidad, pues rompe con el propósito de entregar una experiencia de usuario agradable y fluida.

Dentro del escenario de pruebas de velocidad, se consideraron las 3 funcionalidades principales de acuerdo a los requerimientos de usuario anteriormente mencionados:

- **Localizar parqueadero**
- **Desplegar información de parqueadero**
- **Clasificar parqueaderos**

El proceso realizado para éstas, consiste en ejecutar una función que tome el tiempo a partir del momento en el que el usuario interviene activando la funcionalidad deseada, hasta que la aplicación finaliza la ejecución de la petición y actualiza la vista para que el usuario pueda recibir la información solicitada.

Servidor: Host Webfreehosting.net

Ancho de banda/Ping 4 Mbps / 43ms

Equipo/OS iPhone 4s / iOS 6.1.3

Navegador: Safari Móvil

Localización de parqueaderos			
En la vista inicial, el usuario desliza la imagen que contiene la mira en sentido vertical hasta llenar el objetivo representado con una "P" y obtiene un mapa con su localización y los parqueaderos disponibles en el área.			
# Prueba	Tiempo de respuesta (ms)		Respuesta
	wifi	3G	
1	4443	7454	OK
2	3587	6430	OK
3	2903	4352	OK
4	2764	2564	OK
5	2917	3714	OK
6	1012	2708	OK
7	1328	3564	OK
8	2637	1654	OK
9	1384	3659	OK
10	2342	1023	OK
11	1223	775	OK
12	2954	1071	OK
13	1881	7017	OK
14	1566	1076	OK
15	1660	963	OK
16	3658	1068	OK
17	1382	997	OK
18	716	1861	OK
19	855	1024	OK
20	1637	1019	OK
21	2180	1020	OK
22	969	2109	OK
23	1888	975	OK
24	1323	849	OK
25	754	1121	OK

Tabla 5. Prueba tiempo de respuesta - Localización de Parqueaderos (Adaptada por el autor)

Para esta serie de pruebas se puede concluir que la velocidad promedio de respuesta en un entorno wifi es de 1998,52 milisegundos, es decir que regularmente, y bajo unas condiciones de funcionamiento ideales dentro de un entorno de conexión no masivo, el prototipo tardará un poco menos de 2 segundos en procesar la solicitud del usuario y actualizar en pantalla los parqueaderos localizados.

En cuanto al tiempo que tarda el mismo proceso pero en el marco de una conexión 3G provista por un operador de telefonía móvil celular, y bajo una zona de buena cobertura, encontramos que el resultado es de 2402,68 milisegundos, un poco menos de 2 segundos y medio, algo que no está nada mal para un prototipo no nativo. Finalmente, como resultado de que cada una de las pruebas finalizó de forma exitosa, el porcentaje de éxito en general fué del 100%, un número muy positivo, aunque debe verse con precaución, pues en un escenario de consulta masivo, será determinante adecuar el proceso servidor para obtener una alta disponibilidad de servicio.

Obtener información de parqueadero			
Una vez se han localizado los parqueaderos y éstos se encuentran desplegados en el mapa, el usuario toca el icono q identifica a cada parqueadero (P) y espera que la vista se actualice hasta que se muestre un cuadro de diálogo con la información del establecimiento.			
# Prueba	Tiempo de respuesta (ms)		Respuesta
	wifi	3G	
1	236	572	OK
2	376	398	OK
3	459	429	OK
4	361	313	OK
5	198	548	OK
6	432	186	OK
7	366	439	OK
8	567	320	OK
9	452	419	OK
10	172	378	OK
11	330	379	OK
12	332	256	OK
13	431	423	OK
14	258	137	OK
15	352	482	OK
16	113	592	OK
17	342	619	OK
18	256	422	OK
19	278	490	OK
20	325	763	OK
21	324	236	OK
22	315	187	OK
23	284	409	OK
24	367	327	OK
25	387	549	OK

Tabla 6. Prueba tiempo de respuesta - Obtener información de Parqueaderos (Adaptada por el autor)

Los resultados para la 2da prueba, que consiste en conocer los tiempos de respuesta del prototipo en el momento en el que se quiere obtener la información de un parqueadero desplegado en el mapa, son de igual forma satisfactorios a nivel general. Bajo un ambiente wifi liviano, y luego de 25 evaluaciones, se obtiene un valor de 332,52 milisegundos transcurridos para dar respuesta a la petición, un poco menos de medio segundo. En una cobertura 3G, este valor se eleva hasta 410,92 milisegundos, manteniéndose por debajo del medio segundo para cumplir con el mismo objetivo.

En ambos escenarios de conectividad, la respuesta es medianamente satisfactoria, pues a pesar de que el usuario está esperando una respuesta instantánea, estar abajo del medio segundo con tecnologías web no es nada menospreciable.

El índice de éxito en esta operación ha sido del 100%, una cifra reproducible a grande escala en términos de usuarios.

Trazar ruta Usuario - Parqueadero			
Se ejecuta en el momento en el que el usuario toca sobre la "x" en la esquina superior derecha del cuadro de diálogo que se encuentra desplegando la información del parqueadero seleccionado, generando que se marque la ruta desde la ubicación del dispositivo hasta el parqueadero elegido.			
# Prueba	Tiempo de respuesta (ms)		Respuesta
	wifi	3G	
1	360	5434	OK
2	435	572	OK
3	387	572	OK
4	915	320	OK
5	1204	60	OK
6	246	66	OK
7	378	97	OK
8	296	1126	OK
9	214	71	OK
10	410	63	OK
11	436	46	OK
12	110	117	OK
13	306	632	OK
14	476	434	OK
15	285	68	OK
16	304	116	OK
17	317	491	OK
18	538	91	OK
19	429	51	OK
20	35	127	OK
21	31	47	OK
22	33	502	OK
23	34	301	OK
24	63	80	OK
25	62	33	OK

Tabla 7. Prueba tiempo de respuesta - Trazar ruta usuario - parqueadero (Adaptada por el autor)

La 3era prueba que consiste en conocer el tiempo que tarda el prototipo en dibujar la ruta más adecuada desde la ubicación del usuario hasta un parqueadero en particular, entrega resultados bastante cambiantes. Si bien el promedio de respuesta a esta petición está por debajo del medio segundo tanto para un ambiente wifi (332,16 milisegundos) como para una cobertura 3G (460,68 milisegundos), existen tiempos de respuesta demasiado elevados y otros en contraparte extremadamente bajos, lo cual en gran medida, se debe a la complejidad del servicio provisto por Google en el que a partir de las coordenadas que posicionan tanto al usuario como al parqueadero se calcula, teniendo en cuenta el contexto geográfico, el recorrido apropiado para moverse de un punto al otro. El índice de éxito llega al 100% para una prueba con una variabilidad muy marcada en tiempo de respuesta.

A continuación (Ver Tabla 8) se presenta una compilación de las pruebas realizadas junto con los resultados obtenidos, como imagen global que refleja la respuesta en las funcionalidades cuyos procesos se componen de la relación Cliente-Servidor.

Compilación de Pruebas de Funcionalidad Realizadas				
Descripción	# de Iteraciones	Promedio Respuesta 3G	Promedio Respuesta Wifi	Porcentaje de Éxito
Localizar Parqueadero	25	2402,68	1998,52	100%
Obtener Información Parqueadero	25	410,92	332,52	100%
Trazar Ruta Usuario - Parqueadero	25	460,68	332,16	100%

Tabla 8. Resumen pruebas de funcionalidad (Adaptada por el autor)

11.2 Pruebas de usabilidad

Para las pruebas de usabilidad se habilitó el prototipo en un host y se solicitó a personas aleatorias que ingresaran por medio del navegador safari a iParqueo.

El objetivo era determinar si la interfaz gráfica cumplía con su propósito y si la funcionalidad prestada dejaba satisfechos los requerimientos planteados, además de proporcionar información adicional que pudiera ser aprovechada. De inmediato se localizaron puntos sensibles de mejora, empezando con la vista inicial, en la cual se identificó que si el usuario se distraía por un momento durante el tiempo de carga de la aplicación, se encontraba un poco desorientado en aquello que tenía que realizar para ejecutar la consulta de parqueaderos disponibles. La razón es que si bien se crearon componentes gráficos suficientemente entendibles, el texto que apoyaba la descripción de la acción a realizar no permanecía en pantalla, aparecía y se desvanecía al cabo de 5 segundos.

En un principio esto se implementó con el fin de no saturar la pantalla con muchos elementos, pero dadas las reacciones de los usuarios, fue necesario hacer que el texto permaneciera en pantalla y adicionalmente, mostrar una pequeña animación que simulara la pulsación de un dedo índice sobre la mira para dejar claro el movimiento a realizar.

Así mismo se identificó que aquellos dispositivos que tienen realizado el popular Jailbreak presentan incoherencias gráficas al momento de ejecutar la aplicación, dejando en evidencia que

este método de ampliar las posibilidades del dispositivo, compromete la estabilidad del sistema. El caso que se presentó, en concreto, fue un recorte en el despliegue del mapa para la vista de localización, dentro de la cual, el lugar ocupado en pantalla era un poco menor al 50% del espacio total.

Dentro del objetivo de realizar estas pruebas de funcionalidad, también se encontraba la posibilidad de incorporar nuevas características para complementar la experiencia de usuario, un propósito que dió fruto tanto en la información arrojada por parqueadero como en la idea de marcar la ruta desde la ubicación del usuario hasta el destino elegido.

Como punto final, este proceso fué muy útil para encontrar comportamientos anómalos en la aplicación, conocidos como “bugs”. Fué así como se descubrió una alteración en el cronómetro para el cálculo del costo de parqueo, y se procedió a solucionar este aspecto en la funcionalidad mencionada.

De igual forma se corrigió un problema que entraba en escena en el momento en el que el equipo cambiaba de orientación en el sentido “portrait” (vertical) a “landscape”(horizontal).

12. CONCLUSIONES

El desarrollo de aplicaciones móviles es un escenario cuyo protagonismo crece a pasos agigantados dada la calidad del software que es posible construir, gracias a la variedad de herramientas disponibles y a la evolución tanto de la industria móvil como del usuario.

El levantamiento de información en establecimientos públicos para la construcción de una aplicación móvil de interés general, es un proceso dispendioso que encierra factores adicionales como la dificultad que plantea, para algunas personas, suministrar datos sobre su negocio como medida preventiva para evitar un aumento de la competencia.

Se puede decir que el desarrollo nativo de aplicaciones móviles permite generar software al tope de calidad y rendimiento, facilitando el acceso a las características de hardware del dispositivo (cámara, GPS, acelerómetro), pero que en una aproximación temprana, conlleva un tiempo de aprendizaje mayor y limita su distribución a una sola plataforma.

El desarrollo nativo para iOS es el mejor camino para explotar al máximo el rendimiento de los dispositivos de la manzana, en contra parte, el desarrollador está sujeto a los cambios de implementaciones de las APIs y a las limitaciones que presenta el no tener adquirida una licencia anual, lo cual impide realizar pruebas de sus aplicaciones en dispositivos reales y publicar éstas en la App Store.

Elegir un ciclo de vida acorde al contexto, permite asimilar de forma sencilla los cambios de requerimientos sobre la marcha en la producción de una aplicación, una constante en el mundo de las “apps”.

La tienda de aplicaciones de la plataforma iOS, denominada “App Store”, ofrece una rentabilidad al desarrollador del 70% por aplicación vendida, lo cual la ubica por encima de las demás alternativas del mercado.

Se debe considerar una buena estrategia para destacar dentro de la “App Store” dado el alto número de aplicaciones disponibles: más de 775,000 a la fecha.

Las oportunidades de negocio materializables por medio de la tecnología de la georeferenciación son bastante numerosas si tomamos en cuenta la variedad de comercios y los relacionamos con las distintas necesidades de la sociedad.

Reparar con detalle en la documentación de las APIs provistas por terceros para la implementación de cierta característica o servicio, es fundamental para garantizar el correcto desempeño de la funcionalidad deseada, aún más cuando esta información proviene de gigantes como Google.

Una “web app” es una gran alternativa para iniciarse en el mundo de las aplicaciones móviles; dentro de sus ventajas destacan la velocidad de desarrollo que se puede lograr gracias a

la masificación de las distintas tecnologías web disponibles, la creciente cantidad de plugins pensados en optimizar la funcionalidad del usuario, la constante evolución en la documentación que sirve como base y como guía en la búsqueda de un propósito y la cómoda posibilidad de hacer uso de un sólo código para llegar a los distintos ecosistemas del mercado por medio de herramientas de migración como PhoneGap.

Plantearse la creación de una aplicación para móviles envuelve varios aspectos importantes, pero quizás el más determinante, es elegir el nicho que se pretende abarcar teniendo en cuenta la participación en el mercado de las plataformas que dominan la escena móvil y las versiones de éstas que es posible soportar, dado que esto desencadenará en una correcta elección de las herramientas a usar en el desarrollo de la idea y facilitará el camino en la consecución del objetivo.

Dada la naturaleza de su construcción, es factible realizar de forma “sencilla” la migración de la aplicación a la plataforma con mayor porcentaje de usuarios en el mercado: Android. Una vez realizado ese porte, se procedería a poner sobre la balanza las dos plataformas para, de acuerdo a la estadística, establecer si existe predilección de una plataforma sobre la otra con el objetivo de optimizarla dentro de las posibilidades técnicas.

Dado que el desempeño de una aplicación nativa es sustancialmente mejor que el de una aplicación web, puede ser interesante reescribir la aplicación de forma totalmente nativa siempre y cuando la recepción del público haya sido lo suficientemente importante para embarcarse en

dicha labor e incurrir en los gastos de licenciamiento anual necesarios para poder publicar en la app store.

13. BIBLIOGRAFIA

Ministerio de Tecnologías de la Información y las Comunicaciones. (2012) Boletín trimestral de las TIC, cifras cuarto trimestre de 2012 http://www.mintic.gov.co/images/documentos/cifras_del_sector/boletin_4t_banda_ancha_vive_digital_2012.pdf

StatCounter Global Stats - Browser, OS, Search Engine including Mobile Market Share. (2013). *StatCounter Global Stats - Browser, OS, Search Engine including Mobile Market Share*. Tomado de <http://gs.statcounter.com/>

A Granular App Level Look at Revenues: Google Play vs. Apple App Store | Blog | Distimo. (2013). *App Analytics, Conversion Tracking & Market Data | Distimo*. Tomado de http://www.distimo.com/blog/2013_05_a-granular-app-level-look-at-revenues-google-play-vs-apple-app-store/

Anonymous. (2011). Web Vs. Native Development: There's No Winner. ProQuest Computing. Tomado de United Business Media LLC <http://search.proquest.com/docview/884269606?accountid=34925>

Apple. (2012a). Developer Tools. *Developer*. 2012. Tomado de <https://developer.apple.com/programs/ios/develop.html>

Apple. (2012b). iOS Developer Program. Tomado de <https://developer.apple.com/programs/ios/develop.html>

Apple (2012c). Human Interface Principles. iOS Human Interface Guidelines. Tomado de https://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Principles/Principles.html#//apple_ref/doc/uid/TP40006556-CH5-SW1

Darwish, Nagy Ramadan. (2011). Improving the Quality of Applying eXtreme Programming (XP) Approach. ProQuest. Tomado de LJS Publishing <http://search.proquest.com/docview/928762799?accountid=34925>

Mapa - Mapas.CO. (2013). *Mapa - Mapas.CO*. Tomado de <http://beta.mapas.com.co/>

Davis, Michael A. (2011). Take Charge Of Your Mobile Apps. ProQuest Computing. Tomado de United Business Media LLC <http://search.proquest.com/docview/898806820?accountid=34925>

Gibbs, Mark. (1997). JavaScript: Dressing up Web pages with ease. *Network World*, 14, 2.

INTECO, Laboratorio Nacional de Calidad del Software de. (2009). INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA (pp. 83): INTECO.

King, Nelson. (2000). HTML: The lingua franca of the Web. *ComputerUser. Twin Cities*, 19, 2.

Langley, Nick. Cascading Style Sheets separate presentation from website content. *Computer Weekly*, 1.

Rodríguez, Txema. (2011). Métodos aplicables para el desarrollo de aplicaciones móviles. *Desarrollo Aplicaciones Móviles*. Tomado de <http://www.genbetadev.com/desarrollo-aplicaciones-moviles/metodos-aplicables-para-el-desarrollo-de-aplicaciones-moviles>

Roger, Pence. (2005). Zen and the Art of EXTREME PROGRAMMING (General Information).

ProQuest. Tomado de Penton Business Media, Inc. and Penton Media Inc. <http://search.proquest.com/docview/219596436?accountid=34925>

Westfall, L. (2005). Software requirements engineering: What, why, who, when, and how. *Software Quality Professional*, 7(4), 17-26. Tomado de <http://search.proquest.com/docview/214073024?accountid=34925>

Stark, J. (2010). Building iPhone apps with HTML, CSS, and JavaScript. Beijing: O'Reilly.

Reed, B. (2009). Why the iPhone can't be 'killed'. Network World (Online). Tomado de <http://search.proquest.com/docview/223743554?accountid=34925>

About the Android Open Source Project | Android Open Source. (2013). Welcome to Android | Android Open Source. Tomado de <http://source.android.com/about/index.htm>

Technology Research | Gartner Inc. (2012) Tomado de <http://www.gartner.com/it/page.jsp?id=2237315>

Ionescu, D. (2010). Geolocation 101: How It Works, the Apps, and Your Privacy | PCWorld. PCWorld - News, tips and reviews from the experts on PCs, Windows, and more. Tomado de <http://www.pcworld.com/article/192803/geolo.html>

Single-page application - Wikipedia, the free encyclopedia. (2013a). Tomado de http://en.wikipedia.org/wiki/Single-page_application

Geolocation - Wikipedia, the free encyclopedia. (2013b). Tomado de <http://en.wikipedia.org/wiki/Geolocation>

Ionescu, D. (2010). Geolocation 101: How It Works, the Apps, and Your Privacy | PCWorld. PCWorld - News, tips and reviews from the experts on PCs, Windows, and more. Tomado de <http://www.pcworld.com/article/192803/geolo.html>

Comparison of web map services - Wikipedia, the free encyclopedia. (2012). Wikipedia, the free encyclopedia. Tomado de http://en.wikipedia.org/wiki/Comparison_of_web_map_services

Orenstein, D. (2000). Application programming interface. Computerworld, 34(2), 66-66. Tomado de <http://search.proquest.com/docview/216071140?accountid=34925>

Trice, A. (2012) | PhoneGap Explained Visually. PhoneGap | Home. Tomado de [http://
phonegap.com/2012/05/02/phonegap-explained-visually/](http://phonegap.com/2012/05/02/phonegap-explained-visually/)

Puthraya, T. (2012). Dissecting Phonegap's architecture - Agiliq Blog | Django web app development . Agiliq - Building Amazing Apps | Django Web Development | Mobile App Development. Tomado de <http://agiliq.com/blog/2012/09/dissecting-phonegaps-architecture/>