UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

SOLUCIÓN DE MARKETING Y GESTIÓN PUBLICITARIA APLICADA A CONTENIDO DIGITAL EN PANTALLAS CONECTADAS A DISPOSITIVOS ANDROID TV BOX

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela
Por los bachilleres:

Br. Marín, Enrique.
Br. Franco, Richard.

para optar al título de
Licenciado en Computación

Tutor:

Prof. Sandoval, Franklin

Caracas, Agosto de 2018



Universidad Central de Venezuela.
Facultad de Ciencias
Escuela de Computación

SOLUCIÓN DE MARKETING Y GESTIÓN PUBLICITARIA APLICADA A CONTENIDO DIGITAL EN PANTALLAS CONECTADAS A DISPOSITIVOS ANDROID TV BOX

Autores:

Enrique Marín.

Richard Franco.

Fecha: Agosto de 2018.

RESUMEN

Actualmente la sociedad se encuentra en una era donde la tecnología tiene un crecimiento acelerado y exponencial, donde existen infinidades de temas y áreas desarrolladas con el mismo o similar fin; para tener una mejora evolutiva en una sociedad o país, como la visión de administrar mejor el consumo de tiempo y energía, es necesario hacer la integración de nuevas tecnologías que ayuden a la automatización y eficiencia en la realización de las tareas diarias. En este trabajo de grado se toma como propósito el desarrollo de una solución orientada a mejorar la manera en que se implementa el marketing digital en la actualidad. El objetivo general fue desarrollar una solución de marketing y gestión publicitaria que permitan la visualización de contenido digital a través del desarrollo de dos aplicaciones. Así mismo, esta solución utilizara dispositivos *Android tv box* conectados a pantallas desplegadas en todo el territorio nacional para mostrar publicidad de los comercios afiliados donde podrán enviar contenido digital en tiempo real. Durante el desarrollo se recogió la información de desarrollo del proyecto, en el cual se aplicaron fundamentos y herramientas aptas para el desarrollo adecuado de una solución que integrara una aplicación móvil para los clientes que se encuentren cerca de las pantallas y una aplicación web para la administración de contenido digital por parte de los comercios afiliados.

Palabras Claves: Aplicaciones móviles, marketing digital, Cartelera digital, tecnologías, *SmartCity*.

Universidad Central de Venezuela.
Facultad de Ciencias
Escuela de Computación

ACTA

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado con el título "**SOLUCIÓN DE MARKETING Y GESTIÓN PUBLICITARIA APLICADA A CONTENIDO DIGITAL EN PANTALLAS CONECTADAS A DISPOSITIVOS ANDROID TV BOX**", el cual es presentado por el Br. Enrique Marín, de Cédula de Identidad 17.055.903 y el Br Richard Franco, de Cédula de Identidad 17.958.603, a los fines de optar al título de Licenciado en Computación, dejamos en constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 15 de octubre de 2018, a las 7:30 am en el aula PA3, para que los autores lo defendieran en forma pública, mediante una presentación oral de su contenido, luego de lo cual se respondió las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de 18 puntos.

En fe de lo cual se levanta la presente acta, en Caracas a los 15 días del mes de Octubre del año 2018.

Prof. Franklin Sandoval
Tutor

Prof. Franky Uscátegui
Jurado

Prof. Carlos Acosta
Jurado

DEDICATORIAS

En un principio a Dios por todas las oportunidades brindadas, en ese mismo orden de prioridad al único familiar que tuve en durante la mayor parte de mi vida, a mi difunta abuela Carmen Torres de Marin que se encargó de cuidarme desde mi infancia hasta su ultimo día, a Mariana Bastidas por estar allí durante tanto tiempo en mi vida, por representar aquella persona que me ayudo sin condiciones en todo lo que pudo como pareja y como amiga. A mi Madre que aun cuando no estuvo mucho tiempo presente en mi vida, lo poco que estuvo me cuido con mucho amor.

A Richard Franco, por seguir allí durante tanto tiempo como amigo, por apoyar la idea de un último esfuerzo, que, aunque no parecía estar clara se concretó al final. A su familia que siempre está pendiente de cómo pueden ayudar la gente que lo necesite, entre esas mi persona.

Enrique Leonel Marin Martínez.

Primeramente, quiero agradecer a mi familia y especialmente a mi Madre y a mi Padre, que lo son todo para mí, sin ellos no habría cabida para esta meta y gracias a ellos mismos hoy tengo la oportunidad de alguna forma retribuirles todo lo que han hecho por mí. A mi segunda madre, mi hermana Laymar, ejemplo de vida, actitud y profesionalismo que en todo momento estuvo y está pendiente de mí y hasta más de una vez me ayudó en varios proyectos de la Universidad. A mis amigos y compañeros de trabajo que de alguna u otra forma me ayudaron cuando los necesitaba sin pedirme nada a cambio. A Francisco y Siremla que siempre estuvieron pendiente de que sacara mi título y en todo momento me ofrecieron su apoyo para lograrlo.

Enrique Marín, mi preparador de Orga I que se terminó convirtiendo en mi gran hermano y socio, quien me hizo creer que si se podía. Gracias por reclamar ese último esfuerzo cuando parecía muy lejano. Aunque parezca un fin, esto apenas comienza y yo sé que valdrá la pena.

Richard J. Franco R.

AGRADECIMIENTOS

Para la culminación de este Trabajo Especial de Grado, muchas personas sirvieron de apoyo y ayuda para que el mismo se realizara con total éxito, por tal motivo queremos agradecerles el hecho de habernos ayudado, así sea en un mínimo instante o durante toda la carrera.

Le agradecemos principalmente a dios que nos da sus bendiciones y fuerza diariamente, así como también a todos nuestros familiares por su preocupación, apoyo y motivación durante la realización de este trabajo especial de grado. En especial a nuestras madres y padres, ya que con la educación y los principios que nos impartieron pudimos crecer y convertirnos en los profesionales que somos actualmente.

Gracias a nuestros amigos por acompañarnos durante esta aventura, por estar ahí en los momentos más difíciles y cuando eran más necesarios. El impulso que nos dieron fue absolutamente vital para culminar nuestra carrera exitosamente.

Gracias a nuestro tutor, Franklin Sandoval, por ser no solo nuestro profesor y amigo durante nuestra carrera y proyecto de grado, sino también excelente persona y profesional, futuro colega si dios nos da licencia. Gracias por estar con nosotros en todo momento y por impulsarnos en este emprendimiento. Por abrirnos puertas que nosotros no pensábamos tocar aun y por darnos la fuerza y la motivación de saber que siempre después del final de un camino, existe el inicio de otro. De verdad gracias.

Agradecemos a los profesores de la Facultad de Ciencias de la Universidad Central de Venezuela y a la Universidad, por brindarnos los conocimientos, las metodologías y, sobre todo, la estructura de pensamiento y resolución de problemas que son esenciales para ser un profesional exitoso en el área de la computación, además de impulsarnos con cada enseñanza a ser cada día mejores.

ÍNDICE

ACTA ..... ERROR! BOOKMARK NOT DEFINED.
DEDICATORIAS **4**
AGRADECIMIENTOS **5**
ÍNDICE **6**
ÍNDICE DE TABLAS **10**
ÍNDICE DE FIGURAS **11**
INTRODUCCIÓN **13**
CAPÍTULO 1 PLANTEAMIENTO DEL PROBLEMA **15**
1.1. OBJETIVOS GENERALES 16
1.2. OBJETIVOS ESPECÍFICOS DEL TEG 16
1.2.1. Principales requerimientos *17*
1.2.2. Principales funcionalidades *17*
1.3. ARQUITECTURA DE LA SOLUCIÓN 17
CAPÍTULO 2 MARCO CONCEPTUAL **21**
2.1. SISTEMAS DE INFORMACIÓN 21
2.2. TRANSFORMACIÓN DIGITAL 233
2.3. DISPOSITIVOS MÓVILES 234
2.4. SISTEMAS OPERATIVOS MÓVILES 24
2.5. ARQUITECTURA CLIENTE/SERVIDOR 25
2.6. APLICACIONES WEB 26
2.7. APLICACIONES MÓVILES 26
2.8. TECNOLOGÍAS DE DESARROLLO 28

2.9. ANTECEDENTES DE DESARROLLO 45

CAPÍTULO 3 MARCO METODOLÓGICO 49

3.1. METODOLOGÍAS TRADICIONALES 49

3.2. METODOLOGÍAS AGILES 50

3.2.1. Programación Extrema (XP) 51

3.2.2. Metodología AgilUs 53

3.2.3. Metodología Ágil: Scrum 55

3.3. HISTORIAS DE USUARIO 60

CAPÍTULO 4 MODELO CANVAS 61

4.1. APLICACIÓN DEL MODELO CANVAS A LA SOLUCIÓN CARTELERA DIGITAL 61

4.1.1. Segmentos de mercado 62

4.1.2. Propuesta de valor 63

4.1.3. Canales de distribución 65

4.1.4. Relaciones con los clientes 67

4.1.5. Fuentes de ingresos 68

4.1.6. Recursos clave 69

4.1.7. Actividades clave 70

4.1.8. Asociaciones clave 71

4.1.9. Estructura de costes 72

CAPÍTULO 5 MARCO APLICATIVO 74

5.1. METODOLOGÍA IMPLEMENTADA 74

5.2. HERRAMIENTAS DE SOLUCIÓN 75

5.2.1. Lenguajes de programación *75*

5.2.2. Herramientas de soporte para la metodología Scrum *76*

5.2.3. Herramientas de desarrollo y administración del sistema *78*

5.3. DISEÑO DEL SISTEMA 79

5.4. DISEÑO DE LA SOLUCIÓN 85

5.4.1. Diagrama de clases *86*

5.4.2. Modelo de Base de Datos *87*

5.5. DISEÑO DEL API 89

5.5.1. Estructura de directorios *91*

5.5.2. Documentación del API *93*

5.6. IMPLEMENTACION Y DISEÑO DE PROTOTIPO DE APLICACIÓN WEB 93

5.6.1. Primefaces y Java Server Faces *94*

5.6.2. CouchDB y Ektorp *96*

5.6.3. Desarrollo en Java Server Faces *98*

5.7. IMPLEMENTACION Y DISEÑO DE PROTOTIPO FUNCIONAL DE APLICACIÓN MÓVIL 99

5.7.1. Apache Cordova *99*

5.7.2. AngularJS *100*

5.7.3. Ionic *102*

5.7.4. PouchDB *103*

5.7.5. Desarrollo de aplicaciones móviles híbridas en AngularJS *104*

5.8. PRUEBAS FUNCIONALES Y RESULTADOS 105

5.8.1. Pruebas unitarias 105

5.8.2. Pruebas de funcionalidad 111

5.8.3. Pruebas de aceptación 118

CONCLUSIONES Y RECOMENDACIONES 122

REFERENCIAS BIBLIOGRÁFICAS Y DÍGITALES 124

ANEXOS 127

Índice de tablas

Tabla 1 - Sistemas operativos más usados en Venezuela 25
Tabla 2 - Tipos de aplicaciones móviles 27
Tabla 3 - Ventajas y desventajas de bases de datos 41
Tabla 4 - Diferencias entre metodologías ágiles y tradicionales 50
Tabla 5 - Historia de usuario #1 - Sprint 0 80
Tabla 6 - Historia de usuario #2 - Sprint 0 82
Tabla 7 - Historia de usuario #3 - Sprint 1 83
Tabla 8 - Historia de usuario #4 - Sprint 1 83
Tabla 9 - Historia de usuario #5 – Sprint 2 84
Tabla 10 – Historia de usuario #6 – Sprint 2 84
Tabla 11 – Historia de usuario #7 – Sprint 2 81
Tabla 12 – Historia de usuario #8 – Sprint 3 85
Tabla 13 – Historia de usuario #9 – Sprint 3 85
Tabla 14 – Prueba Unitaria 1 106
Tabla 15 – Prueba Unitaria 2 106
Tabla 16 – Prueba Unitaria 3 1086
Tabla 17 – Prueba Unitaria 4 107
Tabla 18 – Prueba Unitaria 5 107
Tabla 19 – Prueba Unitaria 6 107
Tabla 20 – Prueba Unitaria 7 108
Tabla 21 – Prueba Unitaria 8 108
Tabla 22 – Prueba Unitaria 9 108
Tabla 23 – Prueba Unitaria 10 109
Tabla 24 – Prueba Unitaria 11 109
Tabla 25 – Prueba Unitaria 12 109
Tabla 26 – Prueba Unitaria 13 109
Tabla 27 – Prueba Unitaria 14 110
Tabla 28 – Prueba Unitaria 15 110
Tabla 29 – Prueba Unitaria 16 110

Índice de ilustraciones

Ilustración 1 - Arquitectura de la solución 18
Ilustración 2 - Modelo Sistema de Comunicación 23
Ilustración 3 - Modelo de Cliente/Servidor 26
Ilustración 4 - Android tv Box 29
Ilustración 5 - Como funcionan los Android Tv Box 31
Ilustración 6 - Funcionamiento de la Máquina Virtual de Java 37
Ilustración 7 - Arquitectura del modelo vista-controlado 39
Ilustración 8 - Diseño Arquitectónico Framework 39
Ilustración 9 - Plantillas Digital Signage 1 45
Ilustración 10 - Plantillas Digital Signage 2 46
Ilustración 11 - Dashboard Digital Signage 46
Ilustración 12 - App Móvil Digital Signage 47
Ilustración 13 - Tellysignage Digital Signage 48
Ilustración 14 - Ciclo de vida XP 52
Ilustración 15 - Ciclo de vida AgilUs 55
Ilustración 16 - Ciclo de vida de eventos dentro de un Sprint 59
Ilustración 17 - Lienzo del modelo de negocio 61
Ilustración 18 - Fases de los canales de distribución 65
Ilustración 19 - Tablero de ciclo de vida las tareas 77
Ilustración 20 - Git en Visual Studio Code 77
Ilustración 21 - Google Cloud - Administración 79
Ilustración 22 - Diagrama de clases de la solución 86
Ilustración 23 - Ejemplo de llamado a una operación en JSF 89
Ilustración 24 - Ejemplo de llamado al API HTTP de Couchdb desde Java 90
Ilustración 25 - Instancia del conector de CouchDB desde Java 90
Ilustración 26 - Estructura Proyecto Web Modulo Middleware 91
Ilustración 27 - Estructura Proyecto Web Modulo Front-End 92
Ilustración 28 - Estructura Proyecto Movil 92
Ilustración 29 - Integración Primefaces con Java Server Faces 95
Ilustración 30 - Ejemplo Primefaces 96
Ilustración 31 - Ejemplo de un documento CouchDB 97
Ilustración 32 - Configuración de CouchDB con Ektorp 98
Ilustración 33 - Ejecución de aplicación con Apache Cordova 100
Ilustración 34 - Configuración de la app con AngularJS 101
Ilustración 35 - Consulta del api desde AngularJS 101
Ilustración 36 - Ejemplo de AngularJS 102
Ilustración 37 - Ejemplo de IONIC 103
Ilustración 38 - Configuración de PouchDB 104

Ilustración 39 - Se muestra la bandeja imágenes vacia 111
Ilustración 40 - Presionando Nueva imagen/video 111
Ilustración 41 - Carga exitosa de multiple imágenes/videos 112
Ilustración 42 - Seleccionando archivo a eliminar 112
Ilustración 43 - Se muestra mensaje confirmando que el borrado fue exitoso 112
Ilustración 44 - Se muestra la tabla dispositivo 111
Ilustración 45 - Formulario de entrada para los datos del dispositivo 111
Ilustración 46 - Dispositivo fue creado exitosamente 112
Ilustración 47 - Se muestra la bandeja de galerías vacia 113
Ilustración 48 - Presionando Nueva Galeria y colocando nombre de la galeria 113
Ilustración 49 - Se seleccionan las imágenes/videos para asociar a la galeria 113
Ilustración 50 - Galería (Gaeria PC) fue creada exitosamente 116
Ilustración 51 - Se muestra el calendario de parrillas vacia 116
Ilustración 52 - Se le asigna un nombre a la parrilla 117
Ilustración 53 - Se seleccionan las galerías que se quieren asociar a esta parrilla 117
Ilustración 54 - Se muestra el mensaje que la galería fue creada exitosamente 117

INTRODUCCIÓN

Los consumidores se han vuelto más exigentes, las comunicaciones unidireccionales ya no tienen el mismo efecto que solían tener, con la llegada de la web 2.0 ahora todo gira en torno a la interacción, a la capacidad que tiene el consumidor de poder recibir directamente la información y generar una retroalimentación inmediata.

Así mismo la alta competitividad que hay entre las empresas, marcas y productos, además de la necesidad de buscar vías alternas para darse a conocer, ha traído como resultado el nacimiento de nuevas formas de hacer publicidad, conocidas como publicidad no convencional. Para los anunciantes en la actualidad no es suficiente ofrecer los productos que puedan satisfacer las necesidades de los consumidores. Deben apostar no sólo a aumentar sus ventas, sino a generar recuerdo, posicionamiento y a estar dentro del *top of mind* pero ¿cómo logran esto? Claro, haciendo publicidad, y para ello deben hacer uso de los nuevos recursos que la tecnología pone en sus manos.

El *marketing* digital de bienes y servicios a través de medios electrónicos es una actividad bastante común actualmente. El uso de la tecnología ofrece una plataforma de fácil y rápido acceso para una gran cantidad de personas en casi todo el mundo y debido a esto cada vez son más los negocios que surgen con el objetivo de utilizar dicha plataforma como mercado.

En la actualidad, pequeñas y grandes empresas invierten recursos monetarios y talento humano para la realización de publicidad y marketing, aplicando diversas estrategias de promoción o comunicación como; entrega de volantes, invitaciones a locales, anuncios promocionales, muestras de productos, entre otros. Con la finalidad de adquirir la atención de la mayor cantidad de consumidores posibles

El medio electrónico contiene una gran diversidad de dispositivos y plataformas, que permiten el acceso a los servicios de marketing digital. Pero se presenta la ausencia de una plataforma que le brinde a los usuarios la opción de poder autogestionar el contenido digital que almacenen en estas plataformas, mucho menos de poder enviar

dicho contenido a pantallas digitales desplegadas en cualquier parte del mundo, estas causas justifican la presentación de esta propuesta de emprendimiento, la cual supone además una solución que puede ser adaptada a cualquier rubro que se desee publicitar por el medio electrónico y en consecuencia una solución potencial para los comerciantes y para la mejora de la economía mundial.

Capítulo 1. En este capítulo se plantea el contexto del trabajo de grado y la descripción del planteamiento del problema. También se describe la justificación de la investigación, objetivo general, objetivos específicos, principales requerimientos, arquitectura de la solución y propuesta de la solución.

Capítulo 2. En este capítulo se define el marco conceptual de esta investigación donde se pueden encontrar los conceptos bases referentes a dispositivos móviles, aplicaciones móviles y web, dispositivos Android Tv Box y tecnologías de desarrollo.

Capítulo 3. En este capítulo se despliega la información que llevan de la mano las metodologías para el desarrollo. En el cual se definirá esta metodología como propia de esta investigación y su implementación en el proyecto final.

Capítulo 4. En este capítulo se presenta un análisis sobre el propósito de este emprendimiento a través del modelo Canvas.

Capítulo 5. En este capítulo de despliega principalmente el marco aplicativo para el trabajo especial de grado, así como las herramientas utilizadas y procesos realizados para la culminación del mismo.

Finalmente se plantearán las conclusiones y recomendaciones, anexos y referencias bibliográficas.

CAPÍTULO 1
PLANTEAMIENTO DEL PROBLEMA

Todos los tejidos de la sociedad han venido evolucionando, unos más que otros, a pasos acelerados como es el caso de la tecnología. La sociedad se ha convertido en un constante dar y recibir información, y el proceso de la comunicación, dentro del cual se encuentra la publicidad, ha tenido que evolucionar a la par para poder adaptarse. Así mismo la alta competitividad que hay entre las empresas, marcas y productos y la necesidad de buscar vías alternas para darse a conocer, ha traído corno resultado el nacimiento de nuevas formas de hacer publicidad, conocidas corno publicidad no convencional.

Los consumidores se han vuelto más exigentes, las comunicaciones unidireccionales ya no tienen el mismo efecto que solían tener, con la llegada de la web 2.0 ahora todo gira en torno a la interacción, a la capacidad que tiene el consumidor de poder recibir directamente la información y generar una retroalimentación inmediata.

Este emprendimiento surge de la dificultad encontrada para captar la atención de clientes que se encuentren transitando en las ciudades en un determinado momento. En la actualidad, pequeñas y grandes empresas invierten recursos monetarios y talento humano para lograr captar la atención de posibles clientes que están cercanos a sus locales y para llegar a clientes remotos tienen que recurrir a la publicidad a través de sus redes sociales.

Las empresas que operan actualmente en Venezuela, no ofrecen un servicio de publicidad o marketing en tiempo real en pantallas o vallas electrónicas dentro del territorio nacional, esto requiere que la información de sus productos sea visualizada directamente en una tienda física o publicada en alguna red social.

Debido a la problemática mencionada anteriormente, surge la siguiente interrogante: ¿Qué mecanismo o estrategia permite acercar a los clientes a los productos y servicios de una manera innovadora?

La forma de usar la información y de conducir los negocios está cambiando a nivel mundial gracias al uso de teléfonos inteligentes y dispositivos que son cada día más móviles. Estos cambios están impulsando nuevas formas de hacer negocio y ofrecer servicios, logrando con esto disminuir los espacios y tiempos entre clientes internos, clientes externos y proveedores.

Es gracias a la evolución en las tecnologías de telecomunicaciones que las empresas pueden mantenerse en línea todo el tiempo y les permite conectarse con su audiencia, quienes se convierten en clientes potenciales. El uso de soluciones móviles consiste en proveer flexibilidad e inmediatez, no solo ofreciendo soporte directo a sus clientes sino también utilizando eficientemente los recursos, así como acceso a la información de interés en tiempo real para distintos tipos de audiencia.

Esta solución busca perseguir la satisfacción del cliente ofreciendo información de los productos sin límite geográfico, sin necesidad de que él mismo se encuentre dentro del comercio así poder abarcar la máxima audiencia posible haciendo uso de pantallas, vallas publicitarias electrónicas conectadas a dispositivos Android Tv Box. Este prototipo funcional de solución estará compuesto por dos aplicaciones; una móvil orientada a los clientes cercanos a las pantallas y vallas publicitarias y una aplicación *web* orientada a los dueños de los comercios ya que esta se encargará de la administración del contenido digital a publicitar.

1.1. Objetivos Generales

Desarrollar una solución de marketing y gestión publicitaria que permita la visualización de contenido digital en pantallas conectadas a dispositivos Android Tv Box a través del desarrollo de dos aplicaciones.

1.2. Objetivos Específicos del TEG

- Definir los requerimientos funcionales y no funcionales en el desarrollo de una aplicación web y móvil que controlen el envío de contenido digital a los dispositivos Android Tv Box.

- Diseñar y desarrollar una aplicación web que permita la carga de contenido multimedia.

- Diseñar y desarrollar una aplicación móvil hibrida que permita la sincronización y reproducción del contenido multimedia cargado desde la aplicación web.

- Realizar un conjunto de pruebas de carga, rendimiento, usabilidad y aceptación con el fin de evaluar los puntos de posibles mejoras para las aplicaciones web y móvil.

1.2.1. Principales requerimientos

- Pantalla con puerto HDMI.
- Dispositivo *Android tv box*.
- Acceso a Internet.
- Servidor para el alojamiento de la aplicación web y base de datos de la solución.

1.2.2. Principales funcionalidades

- Registro de usuarios.
- Carga, visualización y borrado de imágenes y videos.
- Gestión de dispositivos lógicos.
- Gestión de galerías.
- Programación de parrillas digitales.

1.3. Arquitectura de la solución

La arquitectura de la propuesta de solución a desarrollar en la aplicación será como se muestra en la figura 1.

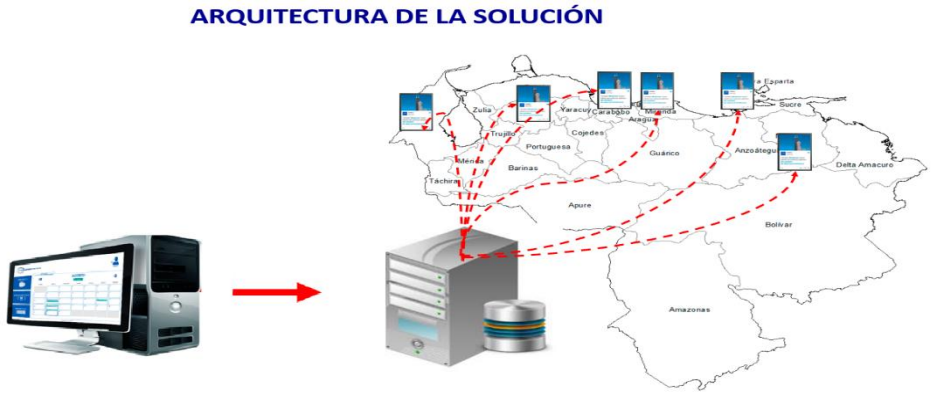


Ilustración 1 - Arquitectura de la solución
Fuente: Los Autores

La solución estará conformada por:

- Modulo cliente administrador, estará conformada por la aplicación web.
- Modulo cliente usuario, estará conformada por la aplicación móvil.
- Modulo almacenamiento y sincronización de datos, será el componente encargado de actualizar en tiempo real todos los dispositivos que tengan instalada la aplicación móvil con las configuraciones realzadas en la aplicación *web*

Para esto se proyecta usar las siguientes herramientas y tecnologías en la implementación de la solución:

Aplicación *Web*

- **Java** como lenguaje de programación para la implementación de la aplicación Web por la sencillez, robustez, flexibilidad y extensibilidad de los *APIS* que provee y por la vasta experiencia que poseen los autores que agilizará el desarrollo de la misma

- ***Java Server Faces*** como *Frameworks* de aplicaciones java basadas en web que simplifica el desarrollo de interfaces de usuario, manejo de eventos, definición de esquemas de navegación y accesibilidad.

- ***Primefaces*** como librería de componentes visuales que facilita la creación de interfaces de usuario ya que posee un interesante conjunto de componentes como editores *HTML*, autocompletado, graficas etc. Es muy ligero, sencillo de instalar y posee una amplia documentación.

- ***Spring framework*** como framework de desarrollo que facilita la creación y el consumo de Servicios *Web RESTful* que nutrirán la aplicación móvil

- ***CouchDB*** como base de datos no relacional que por su característica principal de *replication* entre múltiples bases de datos, permite la actualización en tiempo real de todos los dispositivos que tengan instalada la aplicación móvil

Aplicación Móvil

- ***Ionic*** como plataforma de desarrollo de aplicaciones híbridas basadas en HTML, CCS y JavaScript.

- ***AngularJS*** como *framework front-end* de aplicaciones web basadas en JavaScript cuyos componentes son utilizados para el desarrollo de aplicaciones móviles multiplataformas

- ***PouchDB*** como base de datos JavaScript orientada a trabajar perfectamente con o sin conexión a internet. Permite a las aplicaciones almacenar localmente los datos mientras está *offline* y sincronizarlos con CouchDB cuando la aplicación esté nuevamente *online*.

CAPÍTULO 2
MARCO CONCEPTUAL

En este capítulo se describen teorías, conceptos, procesos y plataforma tecnológica relacionados con el problema a resolver y con la solución planteada. Se muestra una reseña a las plataformas que brindan una solución análoga a la aquí mencionada y se describe los puntos en los cuales por qué, esta presenta una diferencia que representa una mejora.

2.1. Sistemas de información

La Enciclopedia Británica (2018), lo define como un conjunto integrado de componentes para recopilar, almacenar y procesar datos y para proporcionar información, conocimiento y productos digitales. Es sabido que las empresas y otras organizaciones confían en los sistemas de información para llevar a cabo y administrar sus operaciones, interactuar con sus clientes y proveedores, y competir en el mercado. Los sistemas de información se utilizan para ejecutar cadenas de suministro interorganizacionales y mercados electrónicos, entre los tipos de sistemas de información se mencionan:

- Sistema de Procesos de Control (*Process Control Systems*)

- Sistema de Manejo de Información (*Management Information Systems*)

- Sistema de Manejo de Conocimiento (*Knowledge Management Systems*)

- Sistema de Ventas y Mercadeo (*Sales and Marketing Information Systems*)

- Sistema de Proceso de Transacciones (*Transaction Processing Systems*)

El trabajo de investigación se enfocó en describir el concepto de Sistemas de Información dado que en el mundo en el que actualmente habitamos se basa justamente en los principios de la comunicación; si analizamos los distintos avances tecnológicos

que se fueron sucediendo a lo largo de la historia encontramos que la mayoría de ellos están vinculados a la comunicación y a tornar la vida del hombre un poco más sencilla

Sistema de Información: Se entiende como el sistema que posee los Métodos, Procesos, Medios y Acciones que, con Enfoque Sistémico y Regular, aseguren el Flujo de Información en todas las Direcciones con Calidad y Oportunidad.

A continuación, los Componentes de un Sistema de Información y Comunicación:

Emisor: Persona que emite o produce el mensaje en el acto de la comunicación

Receptor: Persona que recibe o produce el mensaje en el acto de la comunicación

Mensaje: En la teoría de la comunicación, información que el emisor transmite al receptor mensaje acústico, escrito o visual.

Canal: Vía o medio utilizado para comunicar un mensaje o distribuir un producto

En la figura 2, se visualizan los componentes de un Sistema de Información interactuando entre sí:

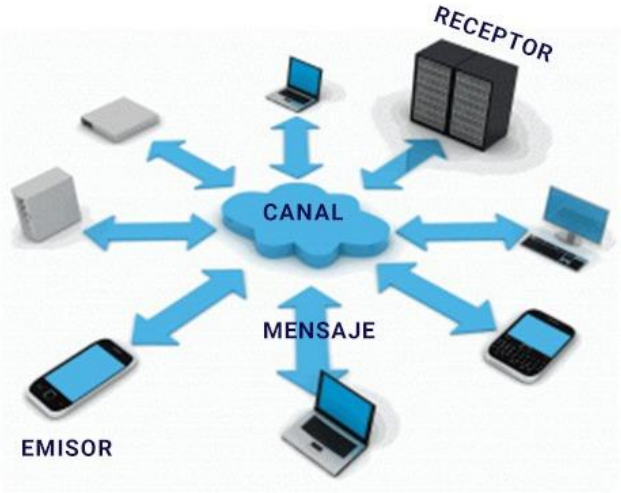


Ilustración 2 - Modelo Sistema de Comunicación.
Fuente: http://thomasinnovation.com/unified-communication-system/

2.2. Transformación Digital

La transformación digital es un proceso que integra la tecnología digital en todos los aspectos del negocio y que requiere de cambios fundamentales en el ámbito de la tecnología, la cultura, las operaciones y la entrega de valor. Para aprovechar mejor las tecnologías emergentes y su rápida expansión en las actividades humanas, las empresas deben reinventarse y transformar radicalmente todos sus procesos y modelos. La transformación digital requiere un cambio de enfoque al perímetro de la empresa y centros de datos más ágiles que puedan de respaldar ese entorno. También implica mudar la tecnología anterior, que puede ser costosa de mantener para la empresa, y modificar la cultura empresarial de manera que respalde la aceleración que trae consigo la transformación digital.

La transformación digital tiene como objetivo la integración de tecnología digital en todas las áreas de una empresa, cambiando fundamentalmente la forma en que opera y brinda valor a sus clientes. También supone un cambio cultural que

requiere que las organizaciones desafíen constantemente sus límites para plantear nuevas formar de hace negocios viendo el mercado desde diversos puntos de vista.

Por un lado, la transformación digital la ha impuesto la creciente demanda de los consumidores por obtener experiencias digitales simples e integradas a través de los diversos canales de servicio. Por otro lado, esta transformación ahora es posible gracias a las diversas revoluciones tecnológicas que permiten a las empresas digitalizar todos los procesos de sus negocios.

La convergencia de la nube informática, las redes sociales corporativas, la movilidad, la inteligencia artificial y los productos conectados establece la plataforma tecnológica necesaria para emprender el camino hacia la transformación digital. Esta "tormenta perfecta" entre las demandas de los consumidores y la viabilidad tecnológica hace que la transformación digital sea uno de los temas de mayor interés del momento.

2.3. Dispositivos móviles

Según Porto (2011), es un aparato o mecanismo que desarrolla determinadas acciones. Su nombre está vinculado a que dicho artificio está dispuesto para cumplir con su objetivo". En otras palabras, un dispositivo no es más que una pieza o conjunto de piezas o elementos preparados para realizar una función determinada y que generalmente forman parte de un conjunto más complejo. Ahora bien, cuando se habla de "Dispositivos Móviles" se hace referencia a dispositivos que poseen ciertas características como; pequeño tamaño, algunas capacidades de procesamiento, conexión permanente o intermitente a una red, memoria limitada, diseñados para una determinada función pero que pudiesen llevar a cabo funciones más complejas.

2.4. Sistemas operativos móviles

Un sistema operativo móvil, al igual que un sistema operativo común y corriente, es un programa o conjunto de programas encargado de gestionar recursos de hardware y proveer servicios a los programas de aplicación de software con las diferencias de que estos son mucho más simples y están orientados a ciertas características como: la

conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de interacción entre el usuario y el dispositivo móvil. Android es el sistema operativo más usado en Venezuela. De acuerdo con datos descritos por Emen acerca de Conatel en el diario El Mundo se construye la tabla 1.

Tabla 1 - Sistemas operativos más usados en Venezuela

Sistema Operativo	**Empresa Propietaria**	**Última Versión Estable**	**Porcentaje de usuarios en Venezuela**
IOS	Apple INC	9.3.3	20,99%
Android	Google INC	6.0.1	45,25%

Fuente: **http://www.elmundo.com.ve/noticias/tecnologia/dispositivos/android-domina-el-mercado-venezolano-de-telefonos-.aspx**

2.5. Arquitectura Cliente/Servidor

Es un patrón utilizado para aislar la lógica empresarial de la interfaz de usuario. Usando MVC, el Modelo representa la información (los datos) de la aplicación y las reglas comerciales utilizadas para manipular los datos, la Vista corresponde a elementos de la interfaz de usuario como texto, elementos de casillas de verificación, etc., y el Controlador gestiona detalles involucrando la comunicación entre el modelo y la vista. El controlador maneja las acciones del usuario, como las pulsaciones de teclas y los movimientos del mouse, y las canaliza al modelo o a la vista según sea necesario.

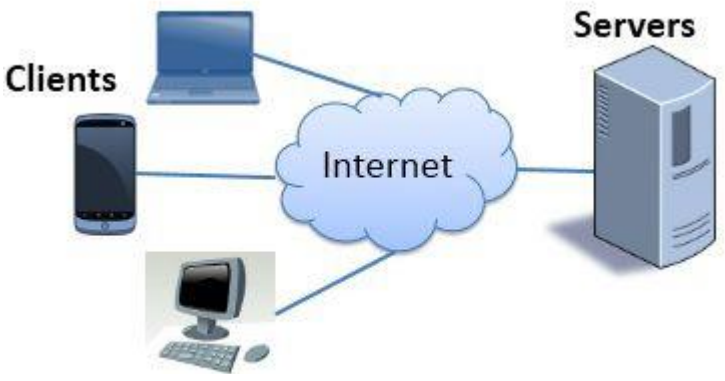


Ilustración 3 - Modelo de Cliente/Servidor
Fuente: https://techdifferences.com/difference-between-client-server-and-peer-to-peer-network.html

2.6. Aplicaciones web

Basados en AlFedaghi (2011) se definen como las aplicaciones que hacen referencia a procesos o funciones que son accedidos a través de un navegador web mediante través de una red y que se desarrollan utilizando lenguajes soportados por el navegador (por ejemplo, HTML, JavaScript). Para su funcionamiento, las aplicaciones web dependen de los navegadores web e incluyen muchas funciones conocidas como aplicaciones de ventas en línea para minoristas, servicios de atención al cliente y correo electrónico. Las aplicaciones web son necesarias en el área de la interacción empresarial. Debido a la complejidad de los sistemas de servicio, el análisis de cada componente y subsistema se vuelve más desafiante.

Desde el punto de vista del usuario, una aplicación web puede proporcionar una interfaz de usuario más consistente en múltiples plataformas porque la apariencia depende del navegador y no del sistema operativo. Además, los datos que ingresa en una aplicación web se procesan y guardan de forma remota. Esto le permite acceder a los mismos datos desde múltiples dispositivos, en lugar de transferir archivos entre sistemas informáticos.

2.7. Aplicaciones móviles

Según Techopedia (2018) Una aplicación móvil, más comúnmente conocida como aplicación (*APP*), es un tipo de aplicación de software diseñada para ejecutarse en un dispositivo móvil, como un teléfono inteligente o tableta. Las aplicaciones móviles a menudo sirven para proporcionar a los usuarios servicios similares a los accedidos en las PC. Las aplicaciones generalmente son pequeñas, unidades de software individuales con funciones limitadas.

Estas se desarrollan pensando en arquitectura base Cliente/Servidor y es utilizado para facilitar al usuario la realización de diversos trabajos, ejecutada desde una

plataforma móvil, pudiendo encontrarse en cualquier lugar o zona geográfica por medio de la utilización de tecnología de conexión de datos (H, 3G, EDGE, entre otros), el uso de esta señal permite establecer una conexión, consulta de información y almacenamiento de la misma a una base de datos externa.

Actualmente se conocen tres tipos principales de aplicaciones móviles, en la tabla 2 se pueden ver tanto concepto como las ventajas y desventajas de cada una de ellas.

Tabla 2 - Tipos de aplicaciones móviles

Tipos de aplicaciones	**Concepto**	**Ventajas**	**Desventajas**
App Nativa	Desarrollada en el lenguaje nativo del dispositivo.	- Acceso completo al dispositivo. - Presentan rendimiento optimizado	- Código no reutilizable entre plataformas. - Constantes actualizaciones.
App Web	Una web app es una versión de la página web optimizada y adaptable a cualquier dispositivo móvil.	- Código base reutilizable. - Proceso de desarrollo sencillo y económico. - Reutilización de sitios responsivos.	- Requiere conexión a internet. - Acceso muy limitado a hardware del dispositivo. - Tiempo de respuesta.
App Hibrida	Combinación de App Nativa y Web, esta recoge los mejor de ellas. Desarrolladas en HTML, CSS, java y JavaScript, pero provee la facilidad de acceder a	- Multiplataforma. - Instalación nativa pero construida en HTML, JavaScript y CSS - Acceso a parte del hardware del dispositivo.	- Experiencia del usuario más propia de la aplicación web que de la app nativa. - No tiene full acceso a los recursos del teléfono. - Visualmente, no son tan atractivas como las nativas

	hardware del dispositivo.	- Puedes visualizarlas en cualquier teléfono móvil. - Permite la reutilización de código ahorrando bastante tiempo a los desarrolladores	

Fuente: Los Autores / https://www.vexsoluciones.com/apps-moviles/apps-nativas-vs-hibridas/

2.8. Tecnologías de desarrollo

Los modelos y tecnologías de desarrollo web han evolucionado mucho en la última década, existen multitud de aplicaciones, frameworks, librerías, arquitecturas y sistemas de publicación en diferentes versiones que a su vez reciben cambios o mejoran con el tiempo. El progreso también ha tenido lugar en lo relacionado con la administración de sistemas, herramientas de publicidad, técnicas de marketing digital.

Android Tv Box

Es un aparato con sistema operativo *Android* en su interior, que conectado a un televisor mediante el puerto HDMI, logra que se puedan usar todas las aplicaciones y conseguir toda la conectividad con nuestra pantalla. Dicho de una manera más simple, un A*ndroid tv box* es como tener una *tablet* en vuestro televisor, con las mismas funciones y posibilidades que vuestra *tablet* o smartphone tiene.

Este dispositivo presenta más recursos de hardware que de la mayoría de los dispositivos móviles potentes existentes en el mercado con la vital diferencia de que, todo el hardware está enfocado en rendimiento y calidad de la imagen, además cuenta con una alta capacidad de memoria interna.

Ilustración 4 – Android tv Box
Fuente: https://www.google.com

Características principales

Dentro de las características de esta tecnología, *Android Tv Box* (2018) menciona las siguientes:

- *CPU:* Amlogic S950X QuadCore de la Corteza A53 2.0 GHz 64Bit.
- *RAM:* 2GB.
- *Memoria Interna:* 16GB
- *Lectores de Tarjetas:* 2 SD/SDHC/mmc
- *Video:* Apoyo 4K * 2K Super HD Video
- *Wifi:* IEEE 802.11b/g/n 2.4G
- *Usb:* 2.0 *3
- *Blueooth: 4.0*

Características Generales

Una de las cosas para las que sirve un *Android tv box* es para conectar todos los aparatos electrónicos entre sí. Basta con imaginar poder entrar al disco duro de la computadora y poder copiar alguna película o serie y verla en algún televisor del hogar conectado, o pasar las fotos del móvil a la televisión de manera rápida y fácil. Todo

esto y mucho más es lo que podrás hacer con un *Android tv box*, como si fuera un centro multimedia para vuestro salón, pero mucho más barato.

Otra de las cosas para las que sirve es para ver todo tipo de contenido multimedia, desde vídeos de YouTube hasta películas o series en todos los formatos posibles. Además, gracias al uso de aplicaciones como XBMC (Kodi) podréis ver películas, series y todo lo que queráis, a la carta, sin esperas ni publicidad.

Para jugar a todos los juegos *Android* que hay en la Play Store. Además, podrás usar controles de alguna consola con puerto USB y jugar, convirtiendo el televisor en un potente centro multimedia capaz de hacer muchas más cosas que cualquier Smart tv del mercado. El aspecto social de tener un cuadro de Android es otra característica popular. Los usuarios pueden, por ejemplo, compartir sitios web interesantes e historias de noticias directamente desde el televisor, consultar el correo electrónico, chatear en línea a través de varios servicios, y más. Las cajas de Android cambian la forma en que se usan los televisores. La conveniencia de poder usar Internet desde el televisor es una característica muy popular. Como la pantalla del televisor se usa en lugar de la pantalla del teléfono, la tableta o la computadora portátil, todo es más fácil de ver. Eso significa que es una experiencia más agradable ver videos, compartir fotos y más en una pantalla grande.

Aplicaciones y usos

El aspecto social de tener un cuadro de Android es otra característica popular. Los usuarios pueden, por ejemplo, compartir sitios web interesantes e historias de noticias directamente desde el televisor, consultar el correo electrónico, chatear en línea a través de varios servicios, y más. Las cajas de Android cambian la forma en que se usan los televisores. La conveniencia de poder usar Internet desde el televisor es una característica muy popular. Como la pantalla del televisor se usa en lugar de la pantalla

del teléfono, la tableta o la computadora portátil, todo es más fácil de ver. Eso significa que es una experiencia más agradable ver videos, compartir fotos y más en una pantalla grande.

- ***Multimedia***: capacidad de reproducir contenido multimedia en HD.

- *Reproduce Netflix*.

- ***Marketing***: se convierte en un canal de comunicación dentro de la ciudad uniendo el comercio con otros sectores como turismo, cultura y ocio. Además, permite promover y administrar publicidad de forma totalmente remota.

- **Bajos costo**: el hardware que contiene el dispositivo con respecto al precio presenta una relación altamente a favor de generar poco gasto al usuario.

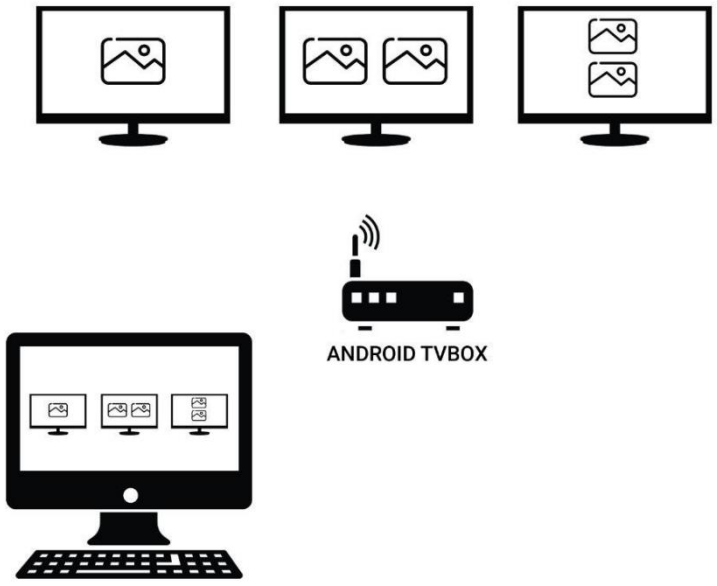


Ilustración 5 - Como funcionan los Android Tv Box

Fuente: Los Autores

REST (*Representational State Transfer*)

Es un tipo de arquitectura de desarrollo que se apoya totalmente en el estándar HTTP. En el estilo arquitectónico REST, los datos y la funcionalidad se consideran recursos y se accede a ellos utilizando identificadores uniformes de recursos (URI), generalmente enlaces en la Web, este enfoque es simple debido a que permite crear servicios y aplicaciones que puedan ser usadas por cualquier dispositivo que entienda HTTP. Márquez (2013). Este tipo de arquitectura define un conjunto de reglas que deben seguirse estrictamente para conseguir su aplicación de forma correcta, cuando esto ocurre se conoce como el nombre de arquitectura de desarrollo *Restful*.

Se actúa sobre los recursos utilizando un conjunto de operaciones simples y bien definidas, típicamente HTTP. En el estilo de arquitectura REST, los clientes y servidores intercambian representaciones de recursos mediante el uso de una interfaz y protocolo estandarizados.

Los siguientes principios alientan a las aplicaciones RESTful a ser simples, livianas y rápidas:

• **Identificación de recursos a través de URI**: un servicio web RESTful expone un conjunto de recursos que identifican los objetivos de la interacción con sus clientes. Los recursos se identifican mediante URI, que proporcionan un espacio de direccionamiento global para el descubrimiento de recursos y servicios. Consulte Las plantillas de anotación @Path y URI Path para obtener más información.

• **Interfaz uniforme**: los recursos se manipulan utilizando un conjunto fijo de cuatro operaciones de creación, lectura, actualización y eliminación: PUT, GET, POST y DELETE. PUT crea un nuevo recurso, que luego puede eliminarse utilizando DELETE. GET recupera el estado actual de un recurso en alguna representación. POST transfiere un nuevo estado a un recurso. Consulte Cómo responder a los métodos y solicitudes HTTP para obtener más información

Lenguajes de programación

Lenguajes de programación hay en gran cantidad, algunos han evolucionado a lo largo del tiempo y siguen vigentes en el transcurso de muchos años, mientras que otros han sido operativos durante un período más o menos largo y actualmente no se usan.

Se entiende que son los diversos conjuntos de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. Cada lenguaje posee sus propias sintaxis.

Dada esta gran variedad de lenguajes, no se pretende dar una visión de todos, sino una clasificación en diversos tipos y concretar algunos de ellos. En general un lenguaje es un método conveniente y sencillo de describir las estructuras de información y las secuencias de acciones necesarias para ejecutar una tarea concreta

Front End

El Front-end es considerado el componente de desarrollo de software que interactúa directamente con el usuario en la capa de presentación y es el encargado de presentar los datos al usuario, este es desarrollado generalmente con los siguientes lenguajes:

- **Lenguaje de Marcado Extensible (XML):** Quin (2015) comenta que el Lenguaje de Marcado Extensible es un simple y muy flexible formato de texto derivado de SGML (ISO 8879). XML juega cada vez más un papel muy importante en el intercambio amplio de data en la Web y en otros medios. XML se convirtió en un estándar el 10 de febrero de 1998 y fue diseñado principalmente para ser autodescriptivo, guardar y transportar data; el mismo es recomendación de la Word Wide Web Consortium (W3C).

- **Lenguaje de marcado de hipertexto (HTML):** (*HyperText Markup Language)* se usa para definir y describir semánticamente el contenido (marcado) de una página web en un formato bien estructurado. HTML proporciona un medio para crear documentos estructurados formados por bloques llamados elementos HTML

delineados por etiquetas, escritos con corchetes angulares. Algunos introducen contenido en la página directamente, otros proporcionan información sobre el texto del documento y pueden incluir otras etiquetas como subelementos. Obviamente, los navegadores no los muestran, ya que se utilizan para interpretar el contenido de la página.

- **Hoja de estilos de cascada (CCS):** Una hoja de estilos en cascada (CSS) contiene definiciones de estilo que se aplican a elementos en un documento HTML. Los estilos CSS definen cómo se muestran los elementos y dónde se ubican en su página. En lugar de asignar atributos a cada elemento en su página individualmente, puede crear una regla general que aplique atributos siempre que un navegador web encuentre una instancia de un elemento, o un elemento asignado a un cierto estilo CLASS. Los estilos CSS pueden colocarse en línea dentro de un único elemento HTML, agruparse en un bloque <STYLE> en la sección HEAD de una página web, o importarse de un archivo de hoja de estilo CSS por separado.

- El mismo archivo de hoja de estilo externo se puede vincular a muchas páginas web, dando así una apariencia común a todo un sitio web. Para usar reglas de estilo CSS en el Diseñador HTML, la propiedad targetSchema de su documento HTML debe configurarse en un navegador web que admita HTML.

- **JavaScript**

JavaScript: es un lenguaje de scripting orientado a objetos y multiplataforma utilizado para hacer que las páginas web sean interactivas (e.x. tener animaciones complejas, botones seleccionables, menús emergentes, etc.). También hay versiones de *JavaScript* más avanzadas del lado del servidor, como *Node.js*, que le permiten agregar más funciones a un sitio web que simplemente descargar archivos (como la colaboración en tiempo real entre varias computadoras). Dentro de un entorno de host (por ejemplo, un navegador web), JavaScript se puede conectar a los objetos de su entorno para proporcionar un control programático sobre ellos.

JQuery: jQuery es una biblioteca de JavaScript rápida, pequeña y rica en funciones. Hace cosas como el recorrido y manipulación de documentos HTML,

manejo de eventos, animación, y Ajax mucho más simple con una API fácil de usar que funciona en una multitud de navegadores. Con una combinación de versatilidad y extensibilidad, jQuery ha cambiado la forma en que millones de personas escriben JavaScript.

Back End

Dentro del desarrollo de *software*, el *Backend* forma parte de un subsistema que permite crear abstracción al momento de consultar y modificar datos relacionados al sistema que lo contiene. Es fundamental a la hora de desarrollar el *Backend* de cualquier *software*, conocer los diferentes tipos de lenguajes de programación que existen, para seleccionar el que mejor se adapte a los requerimientos del sistema en cuestión. A continuación, se mencionan dos de los lenguajes de programación más utilizados al momento de construir el *Backend* de un *software*.

- **RUBY**

Según Ruby, (2016)

> "Un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla."

Desde su liberación pública en 1995, Ruby ha atraído devotos desarrolladores de todo el mundo. En el 2006, Ruby alcanzó reconocimiento masivo, formándose grupos de usuarios activos en las ciudades más importantes del mundo y llenando las capacidades de las conferencias relacionadas a Ruby. Según el índice TIOBE (2016), que mide el crecimiento de los lenguajes de programación, ubica a Ruby en la posición #11 del *ranking* mundial en Julio del año 2016 mientras que en Julio del año 2015 se encontraba en el puesto #15.

Ruby es considerado un lenguaje flexible, ya que permite a sus usuarios alterarlo libremente. Las partes esenciales de Ruby pueden ser quitadas o redefinidas. Se puede agregar funcionalidad a partes ya existentes. Ruby intenta no restringir al desarrollador. Por ejemplo, la suma se realiza con el operador suma (+). Pero si prefieres usar la palabra sumar, puedes agregar un método llamado sumar a la clase *Numeric* que viene incorporada. **Véase** Figura 9.

- **PYTHON**

Python es un lenguaje de programación interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipado dinámico, tipos de datos dinámicos de muy alto nivel y clases. Python combina una potencia notable con una sintaxis muy clara. Tiene interfaces para muchas llamadas y bibliotecas del sistema, así como para varios sistemas de ventanas, y es extensible en C o C ++. También se puede usar como un lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Finalmente, Python es portátil: se ejecuta en muchas variantes de Unix, en la Mac y en Windows 2000 y versiones posteriores.

El intérprete de Python se extiende fácilmente con nuevas funciones y tipos de datos implementados en C o C ++ (u otros idiomas que se pueden llamar desde C). Python también es adecuado como un lenguaje de extensión para aplicaciones personalizables

- **JAVA**

Arnold y Gosling (2001) definen que Java es un lenguaje de programación de propósito general, marca registrada, y como tal es válido para realizar todo tipo de aplicaciones profesionales. Una de las características más importantes es que los programas ejecutables creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. (p. 12). Como características principales se tiene:

- Orientado a objeto.
- Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes. En particular C++.
- Manejo de memoria gestionada por el propio lenguaje y no el programador.
- Incorpora multihilos (multi-threading) para permitir la ejecución de tareas concurrentes dentro de un mismo programa.

En el lenguaje de programación Java, todo el código fuente se escribe primero en archivos de texto plano que terminan con la extensión .java. Esos archivos fuente luego son compilados en archivos *.class* por el compilador *javac*. Un archivo *.class* no contiene código nativo de su procesador; en su lugar contiene *bytecodes*: el lenguaje de máquina de Java Virtual Machine1 (Java VM). La herramienta java launcher luego ejecuta su aplicación con una instancia de Java Virtual Machine.

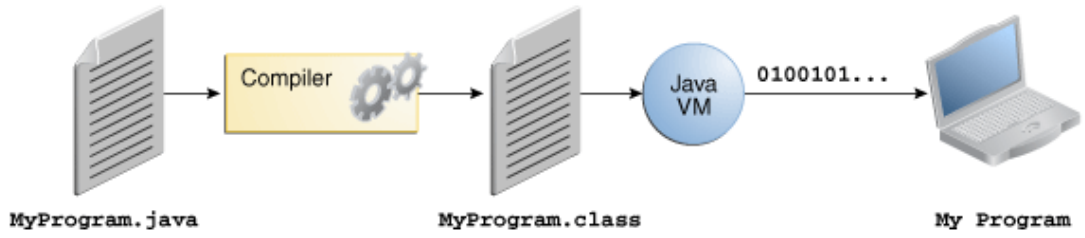


Ilustración 6 – Funcionamiento de la Máquina Virtual de Java

Fuente: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

Arquitectura MVC (Modelo Vista Controlador)

En el paradigma MVC, la entrada del usuario, el modelado del mundo externo y la retroalimentación visual para el usuario se dividen explícitamente y son manejados por tres tipos de objetos, cada uno especializado para su tarea. La vista administra la salida gráfica y / o textual a la parte de la visualización de mapa de bits que está asignada a su aplicación.

El controlador interpreta las entradas de mouse y teclado del usuario, ordenando al modelo y / o a la vista que cambien según corresponda. Finalmente, el modelo maneja el comportamiento y los datos del dominio de la aplicación, responde a las solicitudes de información sobre su estado (generalmente desde la vista) y responde a las instrucciones para cambiar el estado (generalmente desde el controlador). La separación formal de estas tres tareas es una noción importante que se adapta particularmente a Smalltalk-80, donde el comportamiento básico puede incorporarse en objetos abstractos: Vista, Controlador, Modelo y Objeto. El comportamiento de MVC se hereda, se agrega y se modifica según sea necesario para proporcionar un sistema flexible y potente.

Su objetivo era reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples. Tiene como principal característica, que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; con esto se logra que cualquier cambio que se produzca en el Modelo se refleje automáticamente en cada una de las Vistas.

El componente Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. Este componente no sabe que hay internamente en los otros componentes, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo. Referente al componente Vista es el objeto que maneja la presentación visual de los datos representados por el componente Modelo. Genera una representación visual del

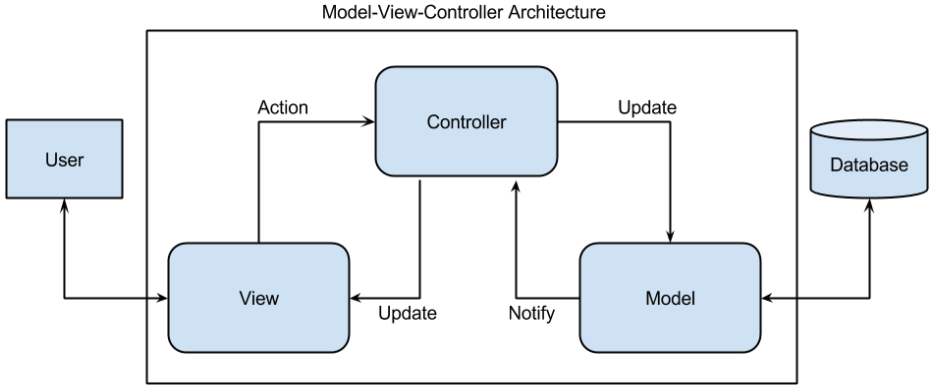


Ilustración 7 – Arquitectura del modelo vista-controlado

Fuente: http://www.patricksoftwareblog.com/overview-of-model-view-controller-mvc/

Modelo y muestra los datos al usuario. Como tercer componente está el Controlador que es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando hay cambios se activa, bien sea por cambios en la información del Modelo o por alteraciones de la Vista.

Frameworks MVC

Es un activo clave en el desarrollo de software orientado a objetos a gran escala. Librerías que brindan facilidad de trabajo para una mayor productividad, tiempos de desarrollo más cortos y una mayor calidad de las aplicaciones. Para cumplir con esto, los *framework* deben estar diseñados de tal manera que puedan evolucionar, ser reutilizados, adaptados y configurados fácilmente. Aprovechando la experiencia en proyectos de banca industrial a gran escala, presenta conceptos y técnicas para la creación de particiones de dominios, capas de marcos y construcción de marcos. En particular, se discutió cómo los aspectos de la solución se relacionan con la utilidad del Frameworks.

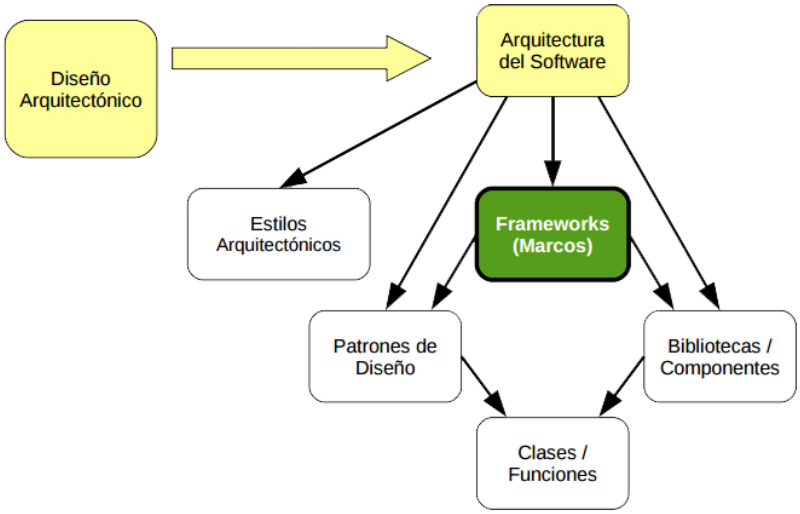


Ilustración 8 - Diseño Arquitectónico Framework

Fuente: Gutiérrez (2010)

Es, en definitiva, un modelo de trabajo que facilita la creación de aplicaciones complejas. Se les conoce también con el nombre de plantillas. A continuación, se mencionan algunos de los *frameworks* utilizados para el desarrollo de aplicaciones.

Base de Datos

Una Base de Datos (BD) es un repositorio centralizado de datos que permite almacenar y organizar hechos o eventos y restituirlos a demanda del usuario para producir información. (Date, 2001). Se puede decir entonces que una BD es un repositorio de datos, que se relacionan entre sí, cuyos datos corresponden a un hecho o evento y que gracias a que existen métodos de extracción para ellos, se puede producir información.

Date (2001) caracteriza a las bases de datos de la siguiente manera:

- Integridad de los datos.
- Acceso concurrente a los datos.
- Facilidad para el cambio de hardware o software.
- Independencia de los datos.
- Control centralizado de los datos.
- Minimización de la redundancia.
- Costo mínimo de almacenamiento y mantenimiento.
- Versatilidad para la representación de relaciones.
- Establecimiento de medidas de seguridad.

Ventajas y desventajas

Date (2001) explica en su libro de Introducción a los sistemas de bases de datos señala diferentes ventajas y desventajas que se están desplegadas en la tabla 4 a continuación:

Tabla 3 - Ventajas y desventajas de bases de datos

Ventajas	Desventajas
Los datos pueden compartirse, haciendo posible la integración con distintas aplicaciones.	Vulnerabilidad si no se controla adecuadamente.
Redundancia controlada, debido al sistema tradicional de archivos independientes.	Falta de integridad si no se controla la BD.
Se logra consistencia de los datos, gracias al control de la redundancia.	Pérdida de tiempo al recuperar una BD.
Manejo de transacciones, es una unidad de trabajo lógica.	Dependiendo de la cantidad de datos se ve afectado el rendimiento.
Integridad, validación de condiciones al introducir datos	Complejidad en relaciones entre los datos.
Seguridad, el ABD al tener control central de los datos.	
Cumplimiento de estándares, se pueden estandarizas procesos, formas, nombres de datos.	

Fuente: Los Autores

Sistema de Base de Datos

Un Sistema de Base de Datos (SBD) es un sistema diseñado para manejar grandes cantidades de datos y producir información. Un SBD es básicamente un sistema computarizado cuyo propósito general es mantener información y hacer que esté disponible cuando se solicite. La información puede ser cualquier cosa que se considere importante para el individuo o la organización a la cual debe servir el sistema. (Date, 2001).

Sistema Manejador de Base de Datos (SMBD)

Un Sistema Manejador de Base de Datos (SMBD) es colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. (Silberschatz, Korth y Sudarshan, 2002). Un SMBD se apoya en la tecnología para cumplir con el propósito de permitir, a las personas, la manipulación (consultas,

actualizaciones, eliminaciones e inserciones) de su información. Se puede decir que un SMBD es un sistema que se encarga de almacenar información y que las personas pueden manipularlas mediante operaciones sobre ellos.

Funciones de SMBD

Date (2001) establece las siguientes funciones:

- Definición de los datos: Definición de datos: debe ser capaz de aceptar definiciones de datos en versión fuente y convertirlas en la versión objeto apropiado. Para hacer esta conversión el SMBD debe incluir un procesador DDL (Lenguajes de definición de datos) como uno de sus componentes.
- Manipulación de datos: debe estar capacitado para atender las solicitudes del usuario para extraer, actualizar, o ingresar nuevos datos. Para atender estas solicitudes debe tener un procesador DML como otro de sus componentes.
- Optimización y ejecución: las peticiones del DML, deben ser procesadas por un componente optimizador, cuya finalidad es determinar una forma eficiente de implementar una petición.
- Seguridad e integridad de los datos: debe controlar las solicitudes de los usuarios para rechazar aquellas solicitudes no permitidas para que los datos estén seguros e íntegros.
- Recuperación y concurrencia de los datos: el SMBD debe tener un componente que sea un monitor de procesamientos de transacciones, que debe cuidar del cumplimiento de ciertos controles de recuperación y concurrencia.
- Diccionario de Datos: debe incluir una función de diccionario de datos, en este caso existen los metadatos que son datos acerca de los datos.
- Rendimiento: debe ejecutar todas las funciones recién identificadas en la forma más eficiente posible.

Existen diferentes manejadores de base de datos que se encuentran en el mercado y dentro de los cuales existen algunos que se inclinan por la filosofía de software libre. A continuación, se describen dos de ellos:

Base de Datos No Relacionales

Una base de datos relacional es un formato de bases de datos muy estructurado basado en una tabla, como MySQL u Oracle. Las bases de datos NoSQL están orientadas a los documentos y le permiten almacenar y recuperar datos en formatos que no sean tablas. Algunas de las plataformas NoSQL más conocidas son MongoDB, CouchDB@ y Cassandra®.

Una base de datos no relacional es cualquier base de datos que no sigue el modelo relacional proporcionado por los sistemas de administración de bases de datos relacionales tradicionales. Esta categoría de bases de datos, también conocidas como bases de datos NoSQL, ha tenido un crecimiento de adopción constante en los últimos años con el aumento de las aplicaciones de Big Data.

Las aplicaciones modernas usan y generan tipos de datos complejos y que cambian constantemente, y las bases de datos relacionales no fueron diseñadas para gestionar este tipo de almacenamiento y recuperación de datos. Las bases de datos NoSQL son más flexibles y escalables. Al trabajar con una base de datos NoSQL, usted puede agregar datos nuevos, sin tener que definirlos previamente en el esquema de la base de datos, lo que le permite procesar rápidamente grandes volúmenes de datos sin estructura, semiestructurados y estructurados. El esquema dinámico de bases de datos NoSQL permite realizar desarrollos ágiles, que requieren iteraciones rápidas y significativas y durante los que no puede haber tiempo de inactividad.

MONGODB

Se trata de una base de datos creada por 10gen del tipo orientada a documentos, de esquema libre, es decir, que cada entrada puede tener un esquema de datos diferente que nada tenga que ver con el resto dé registros almacenados. Es bastante rápido a la hora de ejecutar sus operaciones ya que está escrito en lenguaje C++. Para el almacenamiento de la información, utiliza un sistema propio de documento conocido con el nombre BSON, que es una evolución del conocido JSON pero con la peculiaridad de que puede almacenar datos binarios.

En poco tiempo, MongoDB se ha convertido en una de las bases de datos NoSQL favoritas por los desarrolladores.

CASSANDRA

Dispone de un lenguaje propio para realizar consultas CQL (Cassandra Query Language). Cassandra es una aplicación Java por lo que puede correr en cualquier plataforma que cuente con la JVM. Y cuenta con las siguientes características

• **Distribuido y descentralizado**: ningún nodo tiene el control, por lo que no hay un solo punto de falla.

• **Tolerante a fallas**: los datos se replican, por lo que la falla se puede manejar con gracia.

La configuración habitual de esta BD es tener n nodos cada uno actuando como parte del clúster. Se coordinan entre sí y deciden cómo particionar los datos para que se distribuyan uniformemente y sean redundantes. Un cliente puede conectarse a cualquiera de ellos para realizar consultas. Si se tienen muchos nodos de bases de datos independientes. Cada trabajador puede leer / escribir en un nodo de Cassandra para distribuir la carga. El trabajador y la base de datos podrían incluso estar en la misma máquina

COUCHDB

CouchDB funciona con datos autocontenidos que tienen conexiones sueltas o ad-hoc. Es un modelo que se adapta a muchos elementos del mundo real, como contactos, facturas y recibos, pero descubrirá que esta base de datos puede manejar fácilmente datos de cualquier tipo. Con este libro, aprenderá a trabajar con CouchDB a través de su interfaz web RESTful, y se familiarizará con las funciones clave, como el simple documento CRUD (crear, leer, actualizar, eliminar), MapReduce avanzado, la optimización de la implementación y más.

Se trata de un sistema creado por Apache y escrito en lenguaje Erlang que funciona en la mayoría de sistemas POSIX, incluyendo GNU/LINUX y OSX, pero no así en sistemas Windows. Como características más importantes cabe destacar el uso

de Restfull HTTP API como interfaz y JavaScript como principal lenguaje de interacción. Para el almacenamiento de los datos se utiliza archivos JSON. Permite la creación de vistas, que son el mecanismo que permite la combinación de documentos para retornar valores de varios documentos, es decir, CouchDB permite la realización de las operaciones JOIN típicas de SQL.

2.9. Antecedentes de desarrollo

En términos generales, es en Estados Unidos y en Europa donde la evolución de este tipo de aplicaciones, ayuda a generar un mercado usando una plataforma electrónica, la cual se encuentra en una fase de mayor desarrollo con respecto a Venezuela. Según la investigación realizada se determina que actualmente en Venezuela no existen aplicaciones que se basen en esta tecnología, y esto hace que esta investigación se vuelva un emprendimiento innovador para el país. Tomando en cuenta que existe una cantidad considerable de empresas que ya implementan la solución de cartelera digital, de explica de forma general el planteamiento de una ya existentes en el mercado. En la siguiente sección se habla de empresas que brindan un servicio de carteleras digitales:

Digitalsignage:

Planteamiento (http://www.digitalsignage.com/)

- Ofrece servicio de carteleras al usuario pequeño o mediano (Gratuito).

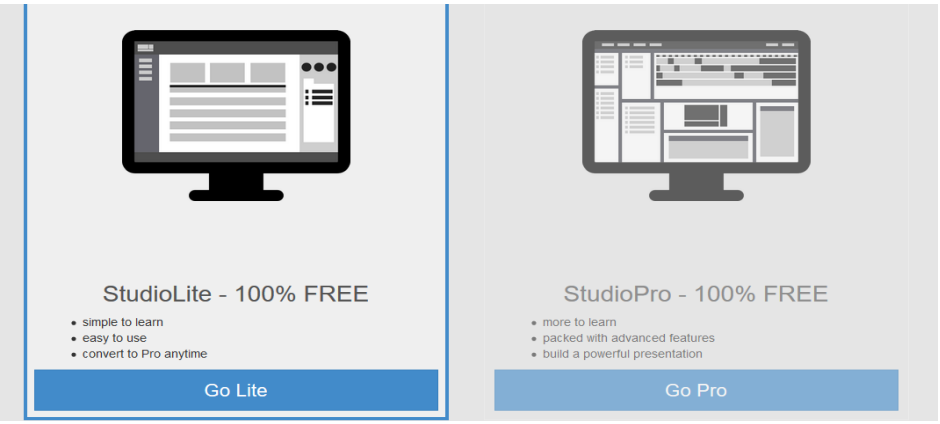


Ilustración 9 - Plantillas Digital Signage 1
Fuente: http://www.digitalsignage.com

- Para las presentaciones cuenta con plantillas ya prediseñadas con temas generales con la idea de brindar adaptación al usuario.

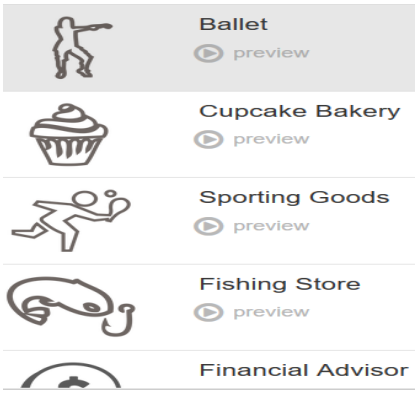


Ilustración 10 - Plantillas Digital Signage 2

Fuente: http://www.digitalsignage.com

- Ofrecen un dashboard (tablero de control) que permite controlar el contenido digital a mostrarse en que pantalla (estación)

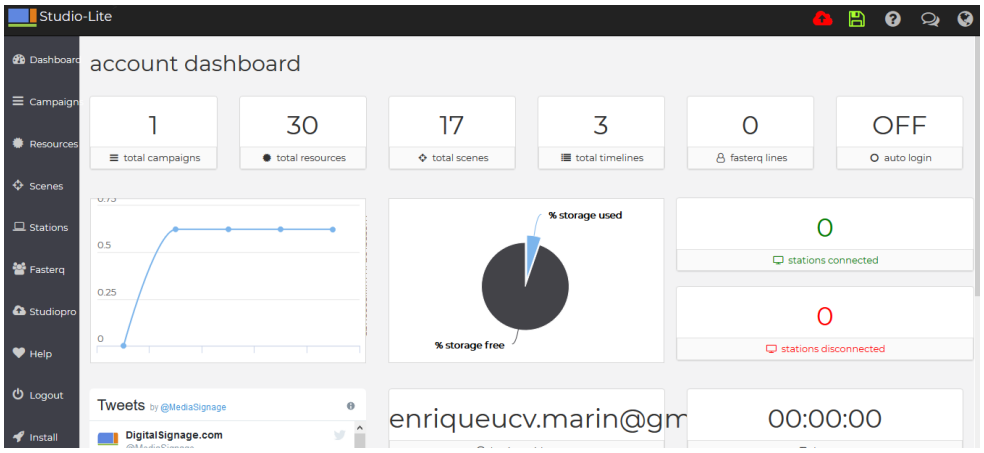


Ilustración 11 - Dashboard Digital Signage

Fuente: http://www.digitalsignage.com

Mobile

Ilustración 12 - App Móvil Digital Signage

Fuente: http://www.digitalsignage.com

Resumen

Como idea base de la aplicación esta empresa ofrece la oportunidad de que el usuario puedo registrarse y crear una cartelera, el registro y administración parecen estar gratuitos hasta cierto punto, el dashboard es complejo y no intuitivo para que al tener que pedir soporte técnico se presenten los pagos. Complementando su app en Android es prácticamente inexistente dado que solo cuenta con pocas pantallas, login y la de administración de que grupo de imágenes se verán en que servidor (estación).

Cuentan con una buena idea de negocio, pero su ejecución no es muy cómoda, a pesar de que cuentan con una buena plataforma de hosting (MediaServer) y usan MedaCloud para almacenar la información, los cuales brindan un buen soporte técnico, toma mucho tiempo adaptarse a la aplicación y su administración.

Tellysignage:

Las carteleras digitales les permiten a los supervisores muestras no solo mensajes de texto, sino también imágenes junto con videos interesantes sobre sus productos. La configuración de carteles digitales mediante el uso de plantillas personalízalas y listas para usarse en ayuda al gerente del negocio. Las plantillas base

del menú ofrecen bastantes diseños. La pantalla le permite al gerente mostrar simultáneamente una mayor variedad de productos. Esta tecnología simplificará su servicio e influirá en sus usuarios, a la vez que ahorrará tiempo y dinero que se hubiera gastado en actualizar manualmente los paneles de menú.

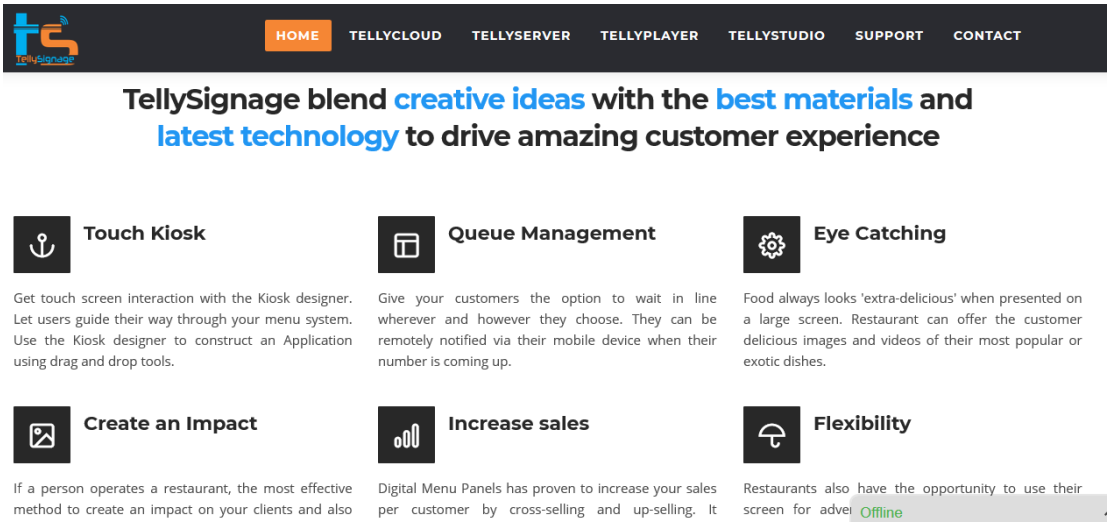


Ilustración 13- Tellysignage Digital Signage

Fuente: http:// www.tellysignage.com

Resumen

Esta plataforma representa el comportamiento de la mayoría de las aplicaciones que brindan el servicio de carteleras digitales, se basan en ofrecer la posibilidad de usar las pantallas para publicar información de sus productos a su vez brindan soporte al usuario para el control y manejo de la aplicación. Tomando en cuenta que estas compañías ofrecen servicios de alta calidad a sus clientes y que tienen personal dedicado a guiar al usuario en el aprendizaje, tiene sentido que no tengan versión gratuita de la plataforma.

CAPÍTULO 3
MARCO METODOLÓGICO

Vasilis (2011) señala que una metodología de software es un proceso de software detallado y completo, y ésta puede basarse en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, espiral, entre otros). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo. Habitualmente se utiliza el término "método" para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño.

Por otra parte, Blanco (2008) dice que una metodología de desarrollo es una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software. Es importante desatacar que existen metodologías tradicionales y agiles las cuales se describen a continuación.

3.1. Metodologías tradicionales

Abrahamsson (2003) define a las metodologías no ágiles son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo; llamadas también metodologías tradicionales o clásicas, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema. Todas las propuestas metodológicas antes indicadas pueden considerarse como metodologías tradicionales. Aunque en el caso particular de RUP, por el especial énfasis que presenta en cuanto a su adaptación a las condiciones del proyecto (mediante su configuración previa a aplicarse), realizando una configuración adecuada, podría considerarse Ágil (Abrahamsson, 2003).

3.2. Metodologías Agiles

Sommerville (2002) relata que un proceso es ágil cuando el desarrollo de software es incremental (entregas pequeñas de software, con ciclos rápidos), cooperativo (cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación), sencillo (el método en sí mismo es fácil de aprender y modificar, bien documentado), y adaptable (permite realizar cambios de último momento). En la tabla 6 describe algunas diferencias entre las metodologías Agiles y las metodologías Tradicionales:

Tabla 4 - Diferencias entre metodologías ágiles y tradicionales

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Más roles.	La arquitectura del software es esencial y se expresa mediante modelos.

Fuente: Sommerville (2002)

Canós (2003) sugiere que cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. A continuación, se mencionan algunas metodologías ágiles. La mayoría de ellas ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento:

- *Extreme Programming.*
- *SCRUM.*
- *Crystal Methodologies.*
- *Dynamic Systems Development Method (DSDM).*
- *Adaptive Software Development (ASD).*
- *Feature -Driven Development (FDD).*
- *Lean Development (LD).*

A continuación, se presentarán en detalle tres de estas metodologías para tener una idea de cómo funcionan las metodologías agiles y se destacarán aspectos como características, ciclo de vida, roles, ventajas, desventajas, entre otros.

3.2.1. Programación Extrema (XP)

Según Beck (1999) XP (*Extreme Programming*, por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Ciclo de vida

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. Estas fases se presentan de acuerdo con Beck (1999):

En **la fase de exploración**, los clientes escriben las historias de usuario que desean para ser incluidas en la primera versión. Cada historia describe una

característica que se añade en el programa. **La fase de planificación** establece el orden de prioridad para las historias y se acuerda el contenido de la primera versión. Los programadores primero estiman cuánto esfuerzo requiere cada historia y se acuerda el calendario. **La fase de iteraciones** incluye varias iteraciones de los sistemas antes la primera versión. El cliente decide que historias serán seleccionarán para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Al final de la última iteración el sistema está listo para la producción. **La fase de producción** requiere pruebas adicionales y el control del rendimiento del sistema antes de que el sistema pueda ser entregado al cliente. Después de que se produce la primera versión para uso del cliente, el proyecto XP debe mantener el sistema en funcionamiento mientras que se producen nuevas iteraciones. Con el fin de hacer esto, **la Fase de mantenimiento** requiere un mayor esfuerzo también para satisfacer las tareas cliente. Por lo tanto, la velocidad de desarrollo puede desacelerarse después de que sistema está en producción. **La fase de muerte** está cerca cuando el cliente ya no tiene más historias para implementar. Esto requiere que el sistema satisfaga las necesidades cliente también en otros aspectos (por ejemplo, en relación con el rendimiento y fiabilidad). En la Figura 12 se muestra el ciclo de vida de XP.

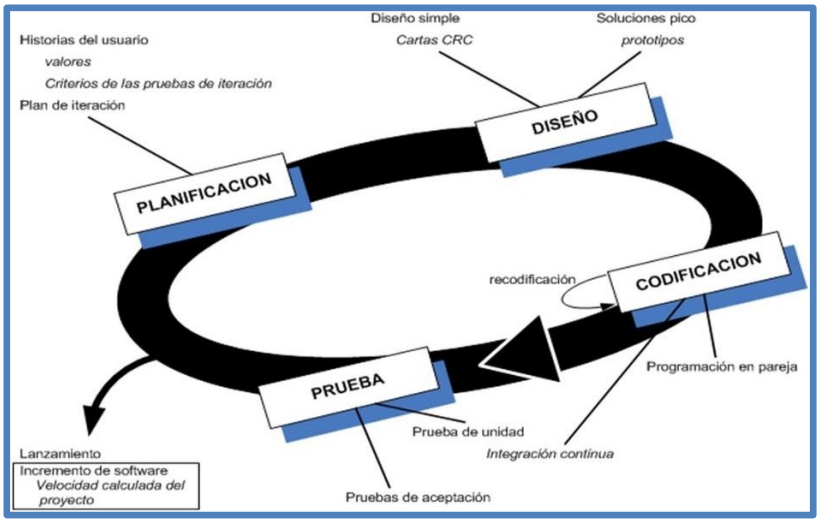


Ilustración 14 - Ciclo de vida XP.
Fuente: Espinoza (2007). (p.55).

Roles y responsabilidades

Hay diferentes roles en XP para diferentes tareas y propósitos durante el proceso y sus prácticas. A continuación, estos papeles se presentan según Beck (1999).

- **Programador:** es el encargado del desarrollo de la solución, arquitecto y responsable de la codificación.

- **Cliente**: El cliente escribe las historias y las pruebas funcionales, y decide cuando cada requisito se cumple. El cliente establece la prioridad de ejecución de los requisitos.

- **Encargado de pruebas (*Tester*)**: corren las pruebas funcionales regularmente, transmiten los resultados de la prueba y mantienen las herramientas de prueba.

- **Encargado de seguimiento (*Tracker*):** encargado de seguimiento da retroalimentación en XP.

- **Entrenador (*Coach*):** responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- **Consultor**: El consultor es un miembro externo que posee los conocimientos técnicos específicos necesarios. El consultor guía al equipo en la solución de sus problemas específicos.

- **Gestor (*Big Boss*):** El gestor toma las decisiones. Con el fin de ser capaz de distinguir cualquier dificultad o deficiencia en el proceso.

3.2.2. Metodología AgilUs

Según Acosta (2012) AgilUs es un método de desarrollo iterativo e incremental que pone el mayor peso del desarrollo en la consecución de la usabilidad. Se centra en que la construcción y desarrollo de las interfaces de usuario no debe ser una adición estética que se da al final del desarrollo del sistema sino, muy por el contrario, el desarrollo de interfaces de usuario debe guiar las decisiones en Ingeniería de Software.

En AgilUs son los usuarios, y no el cliente ni los programadores quienes guían el desarrollo del proyecto.

Ciclo de vida

El ciclo de vida de AgilUs definido por (Acosta 2012) hace énfasis en la importancia del usuario y sus evaluaciones. Está basado en el desarrollo iterativo e incremental de prototipos de alta fidelidad hasta que se convierten en el producto final para entrega. Este producto final puede ser posteriormente modificado a través de un mantenimiento correctivo y/o evolutivo, que no está contemplado como parte del método.

Se busca proporcionar una manera de proceder organizadamente para construir la usabilidad durante el desarrollo de un producto. El ciclo de vida engloba la definición de requisitos, análisis, prototipaje y entrega.

A continuación, se describen las etapas de este método:

- **Requisitos:** Se realiza el análisis global del problema a solucionar, se estudian productos similares existes, se genera un perfil de usuario, y se define la lista de requerimientos a desarrollar. Esta etapa es importante en el desarrollo del software, ya que un mal análisis de requisitos traería como consecuencia un software que no cumple con las necesidades del usuario.

- **Análisis:** Se lleva a cabo el análisis de la solución a desarrollar, se emplean diagramas de casos de uso y modelo de objetos del dominio, siguiendo la notación UML, para definir las funcionalidades que tendrá el producto a desarrollar.

- **Prototipaje:** Se implementa un prototipo rápido de la interfaz de usuario a partir de los patrones de interacción, el cual va evolucionando hasta convertirse en el producto final, se genera la guía de estilo, y se realizan evaluaciones de usabilidad apropiadas a esta etapa: las evaluaciones heurísticas y las listas de comprobación.

- **Entrega:** Se aplican las pruebas al sistema para certificar que la aplicación desarrollada sea un software usable y sin errores, finalmente se pone en producción la aplicación.

Ilustración 15 - Ciclo de vida AgilUs.

Fuente: Acosta (2012).

3.2.3. Metodología Ágil: Scrum

Marco de trabajo por el cual las personas pueden acometer problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente. Scrum es:

- Ligero.
- Fácil de entender.
- Extremadamente difícil de llegar a dominar.

Según Schwaber y Sutherland (2014), Scrum es un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios

de los años 90. Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos. Scrum muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo, de modo que se pueda mejorar. (p.4). El marco de trabajo Scrum consiste en los Equipos Scrum, roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso. Las reglas de Scrum relacionan los eventos, roles y artefactos, gobernando las relaciones e interacciones entre ellos.

Schwaber y Sutherland (2014) dicen,

> "Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo." (p. 4).

Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación, los cuales son:

- **Transparencia**: Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La transparencia requiere que dichos aspectos sean definidos por un estándar común, de tal modo que los observadores compartan un entendimiento común de lo que se está viendo. Por ejemplo: lenguaje común, definición de terminado.

- **Inspección**: Los usuarios Scrum deben inspeccionar los artefactos y progreso frecuentemente, para detectar variaciones. Esto no debe interferir en el flujo de trabajo.

- **Adaptación**: Si se detecta una variación con respecto a las desviaciones de límites aceptables de un proceso, y que el producto resultante no será aceptable, el proceso o material debe ser ajustado. Dicho ajuste debe realizarse cuando antes para minimizar desviaciones mayores y re trabajos.

Scrum prescribe cuatro eventos formales para estas inspecciones y adaptaciones:

- Reunión de planificación del sprint (Sprint Planning Meeting).
- Scrum diario (Daily Scrum).

- Revision del sprint (Sprint Review).
- Retrospectiva del sprint (Sprint Retrospective).

El equipo Scrum

El equipo Scrum consiste en un Dueño del producto (*Product Owner*), el Equipo de Desarrollo (*Development Team*) y *Scrum Master*. Los equipos Scrum son auto organizados y multifuncionales; esto quiere decir que estos eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas eternas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo de Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad (Schwaber y Sutherland, 2013). (p .6). Estos equipos entregan productos de forma iterativa e incrementar, maximizando las oportunidades de obtener retroalimentación. Las entregas incrementales de producto "Terminado" aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto.

- **El Dueño del Producto (*Product Owner*):** El Dueño de Producto es el responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo. El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto (*Product Backlog*). El Dueño de Producto es una única persona, no un comité. El Dueño de Producto podría representar los deseos de un comité en la Lista del Producto, pero aquellos que quieran cambiar la prioridad de un elemento de la Lista deben hacerlo a través del Dueño de Producto. Para que el Dueño de Producto pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones. Las decisiones del Dueño de Producto se reflejan en el contenido y en la priorización de la Lista del Producto.

- **El Equipo de Desarrollo (*Development Team*):** El Equipo de Desarrollo consiste en profesionales que desempeñan el trabajo de entregar un incremento de producto "Terminado". Son los participantes y responsables en la creación del producto, así como también, son empoderados para organizar y gestionar su propio trabajo. El tamaño de estos equipos debe ser lo suficientemente pequeños como para

mantener la agilidad, pero a su vez lo suficientemente grandes como para completar una cantidad de trabajo significativa.

- **El Scrum Master:** Recurso responsable de asegurar que Scrum es entendido, aplicado y adoptado. Líder que está al servicio del Equipo Scrum, encargado de modificar las iteraciones para maximizar el valor creado por el Equipo Scrum. Las actividades principales del Scrum Master hacia la metodología y demás roles los se puede definir como entender la planificación del producto, guiar al equipo de desarrollo para ser auto organizados y multifuncionales, eliminar impedimentos para el progreso del equipo, liderar y guiar la adopción de Scrum.

Eventos de Scrum

Para conllevar una mejor organización y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (time-boxes), de modo que tienen una duración máxima. Una vez que se planifica y se comienza un Sprint, su duración no debe ser modificada, así como tampoco debe ser modificado su alcance, esto último referente a nuevos requerimientos en el transcurso del Sprint o modificación de las historias de usuario definidas. A continuación, se verá los eventos que se definen en el proceso de implementación de dicha metodología:

- **Sprint:** Es considerado el evento más importante y la base de esta metodología, se debe planificar en periodos de tiempo (time-box) no mayores a un mes y en el cual se crea un incremento de producto "Terminado". Este contiene diferentes reuniones que deben ser llevadas a cabo para la buena práctica del Scrum, las cuales son: Reunión de planificación de Sprint (*Sprint Planning Meeting*), Scrums Diarios (*Daily Scrum*), Revisión de Sprint (*Sprint Review*), Retrospectiva del Sprint (*Sprint Retrospective*). En la Figura14se puede ver el ciclo de vida de los eventos de Scrum para cada Sprint planificado.

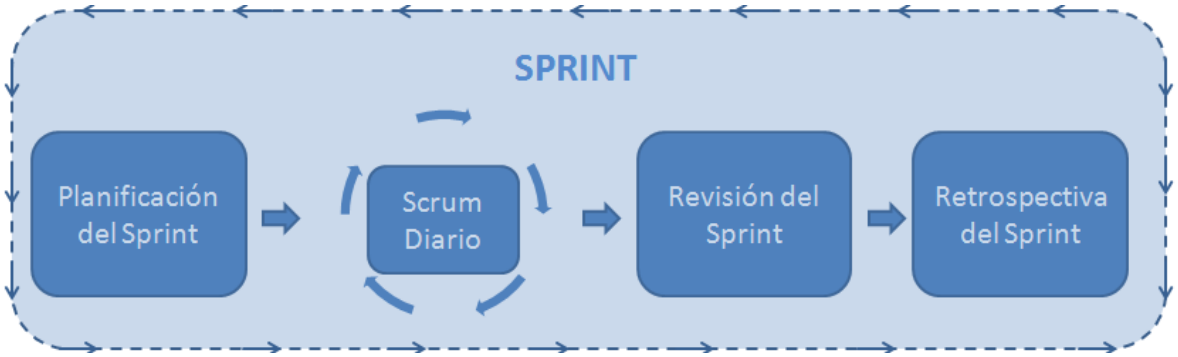


Ilustración 16 - Ciclo de vida de eventos dentro de un Sprint.

Fuente: Vila (2015)

Artefactos de Scrum

- **Lista de producto (*Product Backlog*):** Lista ordenada prioritariamente en la cual se despliegan todas las actividades necesarias para el desarrollo del producto y se convierte en la única fuente de requerimientos. El dueño del producto (*Product Owner*) es el responsable de mantener esta lista de producto. Dicha lista es evolutiva ya que va creciendo a medida que el producto y el entorno también lo hace, cambia constantemente para identificar lo que el producto necesita para ser adecuado. Estas listas pueden ser armadas a raíz de otros artefactos externos como, por ejemplo. Historias de usuario, en las que se define una actividad con valor para el negocio y de ella se despliegan tareas que son tomadas por el Equipo de Desarrollo.

- **Lista de pendientes (*Sprint Backlog*):** Lista que posee un conjunto de elementos de la lista de producto que no se encuentran en estado "Terminado", esta es generada por el Equipo de Desarrollo y se toma como el incremento para el siguiente Sprint dando así más valor al mismo.

- **Definición de "Terminado" (*Definitionof "Done"*):** Es definición para que un elemento de la lista de producto sea "Terminado" debe ser la misma para todo el equipo Scrum, debe cumplir con estándares específicos y ser un incremento utilizable para el Dueño del Producto. A medida que los equipos de Scrum maduran, se espera que su definición de "Terminado" se amplíe para incluir criterios más rigurosos para una mayor calidad en las actividades que se desarrollan.

3.3. Historias de usuario

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación). Cada historia de usuario debe ser limitada, ésta debería poderse escribir sobre una nota adhesiva pequeña. Dentro de la metodología XP las historias de usuario deben ser escritas por los clientes.

CAPÍTULO 4
MODELO CANVAS

El Método Canvas o como se le conoce mundialmente el "*Business Model Canvas*" fue creado inicialmente como tesis del doctorado de Alex Osterwalder en el 2010, El modelo de Osterwalder organiza la operativa donde las empresas crean valor. Una de las ventajas es que las fortalezas y debilidades son plasmadas en un" lienzo" de manera visual. Proporciona una metodología válida para diferentes empresas sin importar la fase de madurez que se encuentren o el sector al que se dirige. Tal y como explica Alexander "la mejor manera de describir un modelo de negocio es dividirlo en 9 módulos interrelacionados entre sí en los que son mostrados gráficamente y pueden moldearse. Estos módulos cubren las cuatro áreas principales de un negocio: clientes, oferta, infraestructuras y viabilidad económica.

Ilustración 17 – Lienzo del modelo de negocio.
Fuente: Osterwalder (2010)

4.1. Aplicación del modelo canvas a la solución Cartelera Digital

Primeramente, se describe los nueve (9) módulos en la cual se centró la solución Cartelera Digital, tal y como ya sabemos estos bloques están interrelacionas entre sí por tanto todos y cada uno de ellos son importantes para desarrollar un modelo de negocio claro y consistente.

4.1.1. Segmentos de mercado

En este módulo se analiza el mercado objetivo al que está enfocado el modelo de negocio, es decir, a cada uno de los diferentes mercados de clientes a los que se atenderá. En este emprendimiento se ha decidido elegir entre dos (2): Por un lado, serían aquellos locales y comercios que dispongan de televisores o pantallas para publicitar sus productos y servicios con la finalidad de captar la atención de posibles clientes que estén transitando en las cercanías de dichas pantallas o empresas que deseen transmitir información de interés a sus empleados ubicados en diferentes pisos u oficinas. Por otro se consideran aquellas empresas que buscan abarcar la máxima audiencia posible y deseen publicitar sus productos y servicios en pantallas propias que estarán desplegadas en calles, avenidas y sitios estratégicos alrededor del mundo

En primera instancia se ha decidido centrarse en el primer grupo, es decir, locales, comercios o empresas que poseen pantallas y las quieren utilizar para llamar la atención de posibles clientes o hacer llegar información de interés a sus empleados o visitantes. Por lo tanto, se define las características de estos y se describen sus necesidades:

- Al mercado que se dirige la solución estaría dentro de lo que es el mercado de masa ya que no se centra en el público en general. Tal y como se ha indicado anteriormente, se centra primeramente en un grupo.
- Tampoco tendrá en cuenta las plataformas multilaterales (o mercado multilateral), ya que la solución no está dirigida ni a 2 o más segmentos.

- Se centrará en un nicho de mercado, es decir, a una porción bien definida de un segmento de mercado mayor. Dentro del nicho de mercado los clientes poseen características y necesidades similares. Se elige un nicho de mercado y se busca en él las necesidades que no estén siendo satisfechas.
- El principal cliente tal y como se ha explicado, son los comercios o empresas que posean pantallas las cuales desean utilizar para promocionar sus productos, servicios o mostrar información de interés. Clientes que por lo general terminan recurriendo a un prestador de servicio que administre la información que se muestra en sus pantallas y terminan dependiendo de dicho prestador de servicio para gestionar el contenido a mostrar y en casos de fallos deben esperar mucho tiempo para que el prestador del servicio les solucione los problemas.
- Clientes que deseen tener el control de gestión del contenido a mostrar y el tiempo que desean mostrarlo. Utilizando una herramienta muy sencilla de gestión y donde las actualizaciones de contenido se vean reflejadas inmediatamente sin necesidad de un tercero que les de soporte.
- Cada cliente tiene su particularidad, tienen gustos distintos que generalmente los servicios que existen son muy estáticos y son intolerables a adaptarse a las necesidades específicas de los clientes. Con esta solución, cada cliente tendrá control total del contenido a mostrar.

En definitiva, los clientes serán comercios o empresas que adquieran el servicio a través de las redes sociales, anuncios publicitarios o por referencia y ellos mismos tendrán control total de la solución donde podrán adaptarla a sus necesidades.

4.1.2. Propuesta de valor

La propuesta de valor es el factor que hace que un cliente se decante por una u otra empresa; su finalidad es solucionar un problema o satisfacer una necesidad del cliente. Las propuestas de valor son un conjunto de productos o servicios que satisfacen los requisitos de un segmento de mercado determinado. En este sentido, la propuesta de valor constituye una serie de ventajas que una empresa ofrece a los

clientes. Algunas propuestas de valor pueden ser innovadoras y presentar una oferta nueva, mientras que otras pueden ser parecidas a ofertas ya existentes e incluir alguna característica o atributo adicional.

Este módulo se centra en el producto que el cliente debe prestar atención y para persuadirlo se debe entender por qué el cliente necesita la solución. Por tanto, se debe tener una buena propuesta de valor, tanto cuantitativa (precio, velocidad del servicio) o cualitativa (diseño, experiencia del cliente). Además, dicha propuesta de valor puede ser usada como base de la estrategia de *marketing*.

En Cartelera Digital, tal como se ha indicado en el segmento de mercado, la propuesta de valor es la **autogestión del contenido digital** de cada cliente. A esto se hace la siguiente explicación:

- La personalización de contenidos se encuentra dentro del ámbito tecnológico, marketing, publicidad y la comunicación audiovisual en general.

Se debe destacar para los clientes, que el producto es **novedoso** en el ámbito tecnológico por la utilización de los dispositivos Android Tv Box y sincronización inmediata de contenido entre el servidor y la aplicación móvil

Es **diferente** porque la gestión del contenido la hace el mismo cliente, puede gestionar qué, cuándo y dónde se muestra el contenido digital deseado y no necesita de terceros ni depender de sus tiempos de respuesta para actualizar cualquiera de las pantallas.

Les permite a los clientes **reducir costos** porque no tiene que contratar empresas de servicio de mantenimiento ni de soporte de la solución y dado la sencillez en la utilización de las aplicaciones *web* y móvil, cualquier empleado del comercio puede hacer actualizaciones y no se necesita un agente experto en el manejo de la herramienta.

Ayuda a **reducir riesgos** en lugares de poca o baja conexión a internet porque la descarga del contenido solamente se realiza una vez y luego puede trabajar completamente *offline* según la programación que se le haga a dicho contenido.

La solución es intuitiva, **sencilla y cómoda de utilizar** incluso para personas ajenas al área de tecnología, no requiere de conocimiento especial para la instalación de los dispositivos en las pantallas y una vez iniciada la sesión en la app móvil, esta recuerda la última configuración que tuvo antes de cerrarse o apagarse el dispositivo y solo necesita abrirse y continuará reproduciendo el contenido digital sin tener que hacer configuraciones adicionales.

4.1.3. Canales de distribución

En este módulo se explica el modo en que una empresa se comunica con los diferentes segmentos de mercado para llegar a ellos y proporcionarles una propuesta de valor. Los canales de comunicación, distribución y venta establecen el contacto entre la empresa y los clientes. Son puntos de contacto con el cliente que desempeñan un papel primordial en su experiencia. Los canales tienen, entre otras, las funciones siguientes:

- Dar a conocer a los clientes los productos y servicios de una empresa.
- Ayudar a los clientes a evaluar la propuesta de valor de una empresa
- Permitir que los clientes compren productos y servicios específicos
- Proporcionar a los clientes una propuesta de valor
- Ofrecer a los clientes un servicio de atención posventa.

Tipos de canal			Fases de canal				
Propio	Directo	*Equipo comercial*	**1. Información** ¿Cómo damos a conocer los productos y servicios de nuestra empresa?	**2. Evaluación** ¿Cómo ayudamos a los clientes a evaluar nuestra propuesta de valor?	**3. Compra** ¿Cómo pueden comprar los clientes nuestros productos y servicios?	**4. Entrega** ¿Cómo entregamos a los clientes nuestra propuesta de valor?	**5. Posventa** ¿Qué servicio de atención posventa ofrecemos?
		Ventas en internet					
	Indirecto	*Tiendas propias*					
Socio		*Tiendas de socios*					
		Mayorista					

Ilustración 18– Fases de los canales de distribución.
Fuente: Business Model Generation escrito por Alexander Osterwalder

El *marketing* de Cartelera Digital es el mismo que en cualquier otro *marketing*. Al ser por internet, sus características son más específicas y las técnicas siempre están evolucionando. El marketing digital es la integración de estrategias simultáneas en la web, a través de un proceso y metodología específica, en busca de objetivos claros usando diversas herramientas, plataformas y medios sociales. Los canales de *marketing* digital son las tácticas que podemos disponer para elaborar nuestra estrategia en internet.

Existen muchos canales digitales como: creación de páginas *web* comerciales orientada a la venta, *marketing* por correo, publicidad en buscadores, *banners*, *blogs*, redes sociales, *marketing* viral, etc. En el siguiente apartado se explica algunos de estos canales que se han decidido utilizar para la solución de Cartelera Digital.

- **Página web comercial**: una página web comercial debe ser fácil de usar y el usuario debe tener siempre visible su compra. Hay que trabajar con el usuario y su navegación dentro de la página, donde la primera impresión o el *landing page* de la página, le presente inmediatamente una propuesta de valor que lo lleve a seguir navegando para conocer más a detalle la solución. Se le debe presentar en todo momento la posibilidad de contactos o una sección donde los posibles clientes puedan dejar su correo electrónico con alguna sugerencia o inquietud para estrechar la distancia con los clientes. No menos importante, una vez se tenga casos de éxito con otros clientes, publicarlos en la página para dar sentido de confianza a aquellos clientes que tengan incertidumbre de utilizar la solución.
- **Publicidad**: la publicidad se centra en internet a través de anuncios constantes en los buscadores, utilizando mayormente el servicio de *Google*. Se harán constantemente anuncios de las ofertas de los productos y se mostrarán imágenes para que impacte más en el consumidor. Se publicará dentro de otros sitios web un *banner* publicitario. Aunque esta fuente de tráfico tiene una baja tasa de conversión, proporciona una buena visibilidad en la web.
- **Mailing**: Se debe llevar a cabo un *marketing* digital y para ello se propuso enviar correos electrónicos al cliente que haya dejado su correo contacto en la

página comercial. Esto se realizará poco antes de hacer la compra con preguntas, como la gama de servicios que le interesa y si desea estar informado mediante este canal de comunicación. Con estos correos se da la oportunidad de que el cliente pueda enviar ideas que le gustaría que existiesen. Ya que esto promueve el *feedback* inmediato, se establece una relación con el cliente más fuerte y permite saber qué es lo que más interesa en el mercado y qué más se puede hacer para complacer al consumidor.

- **Banners**: Un banner es una forma de publicidad por internet, consiste en incluir una pieza publicitaria dentro de una página *web*. Su objetivo principal es atraer el tráfico hacia el sitio web del anunciante que paga por su inclusión. Esta es una fórmula de publicidad digital que trata de despertar la atención del usuario y que promueve también la interacción.
- **Redes sociales**: Es uno de los medios más utilizado en la actualidad, las redes sociales representan cada día más una excelente fuente de tráfico. Te permiten dar a conocer la marca por todo el mundo y crear una real interacción con los clientes ya que tienen un fuerte impacto en la imagen de la empresa.

4.1.4. Relaciones con los clientes

En este módulo se describen los diferentes tipos de relaciones que establece una empresa con determinados segmentos de mercado. Las empresas deben definir el tipo de relación que desean establecer con cada segmento de mercado. La relación puede ser personal o automatizada, pueden estar basadas en los siguientes fundamentos:

• Captación de clientes.

• Fidelización de clientes.

• Estimulación de las ventas (venta sugestiva).

Para este emprendimiento, las relaciones con los clientes serian de servicios automáticos, en el que combina una forma más complicada de autoservicio con

procesos automáticos. Estos servicios automatizados reconocen a los diferentes clientes y sus características para ofréceles información relativa a sus necesidades.

Respecto a la captación de clientes tal y como se ha explicado en el módulo anterior, se utilizan los banners y se intenta aparecer en los buscadores. Para estimular las ventas, se decide sentarse en los canales como página *web* comercial, envío de correos constantes con ofertas, promociones, actualizaciones novedosas, en las redes sociales publicitando continuamente la solución. También se tiene pensado una pequeña encuesta, donde el cliente indique que servicios desea si no está disponible e ideas nuevas para la solución, de esta forma puede puntuar el producto y poder mejorar el servicio lo máximo posible. Para la comodidad del cliente en la página, habrá un video que facilitará el uso de la solución.

4.1.5. Fuentes de ingresos

El presente módulo se refiere al flujo de caja que genera una empresa en los diferentes segmentos de mercado. Cada fuente de ingresos puede tener un mecanismo de fijación de precios diferente: lista de precios fijos, negociaciones, subastas, según mercado, según volumen o gestión de la rentabilidad. Un modelo de negocio puede implicar dos tipos diferentes de fuentes de ingresos:

- Ingresos por transacciones derivados de pagos puntuales de clientes.
- Ingresos recurrentes derivados de pagos periódicos realizados a cambio del suministro de una propuesta de valor o del servicio posventa de atención al cliente.

Cada fuente de ingresos puede tener un mecanismo de fijación de precios diferente, lo que puede determinar cuantitativamente los ingresos generados. Existen dos mecanismos de fijación de precios principales: fijo y dinámico. Las distintas fuentes de ingresos son:

- Venta de activos.
- Cuota por uso.

- Cuota de suscripción.
- Préstamo / alquiler.
- Concesión de licencias.
- Gastos de corretaje.
- Publicidad.

Este emprendimiento tiene pensado que las fuentes de ingreso sean o por **cuota por uso** o por **cuota por suscripción**. Una vez cerrado un negocio con un cliente del segmento de locales o comercios que disponen de sus pantallas, se le espera generar una suscripción ya sea mensual, semestral o anual sobre la cantidad de dispositivos *Android tv Box* que solicite, es decir, dependiendo de la cantidad de pantallas que el cliente desea utilizar, se le proporcionará la cantidad de dispositivos y se le cobrará una suscripción por dispositivos en uso y la utilización de la aplicación web.

Para el segundo segmento de mercado, donde el negocio provee las pantallas o vayas electrónicas publicitarias, se les cobraría a los clientes por tiempo de transmisión de su contenido en dichas pantallas. Se espera que, dependiendo de la ubicación de las pantallas, tráfico de personas y horario se tendrán diferentes tarifas para generar competencia y a futuro proveer de suscripciones VIP con horarios y ubicaciones preferenciales.

4.1.6. Recursos clave

En este módulo se describen los activos más importantes para que un modelo de negocio funcione. Todos los modelos de negocio requieren recursos clave que permiten a las empresas crear y ofrecer una propuesta de valor, llegar a los mercados, establecer relaciones con segmentos de mercado y percibir ingresos. Los recursos clave pueden ser:

- Físicos.
- Económicos.
- Intelectuales
- Humanos.

Además, la empresa puede tenerlos en propiedad, alquilarlos u obtenerlos de sus socios clave.

Este emprendimiento tiene como recurso clave la solución de Cartelera digital, es decir, la aplicación web alojada en un servidor y la aplicación móvil que se instalará en los dispositivos *Android Tv Box*.

- **Físicos**: Los dispositivos Android Tv Box que se conectarán a las pantallas o televisores provistos por los clientes.
- **Humanos:** En este apartado se debe analizar qué trabajadores se necesitan, en un principio los socios del emprendimiento llevarían el negocio, pero a medida que vaya creciendo y vaya teniendo más clientela, quizá se necesite más personas. Es necesario establecer las personas de aquí a tres años como mínimo.

4.1.7. Actividades clave

Todos los modelos de negocio requieren una serie de actividades clave. Estas actividades son las acciones más importantes que debe emprender una empresa para tener éxito, y al igual que los recursos clave, son necesarias para crear y ofrecer una propuesta de valor, llegar a los mercados, establecer relaciones con clientes y percibir ingresos. Además, las actividades también varían en función del modelo de negocio.

Las actividades clave se pueden dividir en las siguientes categorías:

- Producción.
- Resolución de problemas.
- Plataforma / red.

Dentro de las categorías de las actividades clave que nombra Alexander Osterwalder, Cartelera digital se encuentra en la categoría Plataforma / red, ya que el modelo de negocio está diseñado con una plataforma que su recurso clave está subordinado a las actividades calve relacionadas con la red. Por tanto, las redes pueden funcionar como una plataforma.

En cuanto a la solución de Cartelera Digital, se tienen principalmente las siguientes actividades clave:

- **Posicionamiento**: por una parte, se puede posicionar la solución con publicidades en buscadores, a través de blogs, y por las redes sociales. Se debe hacer publicaciones y actualizaciones constantes de la solución para estar conectados en todo momento con la audiencia.
- **Contacto**: a través del envío personalizado de correos con aquellos clientes que dejen sus sugerencias o inquietudes en la página *web* comercial y así tener una atención personalizada con cada cliente.
- **Instalación de la solución**: en algunos casos se necesita de algún recurso humano que vaya a las locales de los clientes y haga la instalación con un breve tutorial de uso de la solución.
- **Servicio post venta eficaz**: estar en contacto con los clientes para ofrecerle actualizaciones, mejoras o escuchar sugerencias y siempre poder mejorar el servicio.

4.1.8. Asociaciones clave

En este módulo se describe la red de proveedores y socios que contribuyen al funcionamiento de un modelo de negocio. Las empresas se asocian por múltiples motivos y estas asociaciones son cada vez más importantes para muchos modelos de negocio. Las empresas crean alianzas para optimizar sus modelos de negocio, reducir riesgos o adquirir recursos. Se puede mencionar cuatro tipos de asociaciones:

- Alianzas estratégicas entre empresas no competidoras.
- Asaciones estratégicas entre empresas competidoras.
- Joint-ventures (empresas conjuntas) para crear nuevos negocios.
- Relaciones cliente-proveedor para garantizar la fiabilidad de los suministros.

Entre las alianzas que se tienen en mente, serían los clientes ya que una relación comercial basada en la confianza, la transparencia y la lealtad conlleva recompensas

mutuas y ayuda a las empresas a disfrutar de una ventaja competitiva en el mercado gracias a la contribución del cliente en cuanto al diseño del producto o la utilización de una nueva tecnología, la mejora de la calidad, la reducción de costes, etc.

4.1.9. Estructura de costes

En este módulo se describen los principales costes en los que se incurre al trabajar con un modelo de negocio determinado. Tanto la creación y la entrega de valor como el mantenimiento de las relaciones con los clientes o la generación de ingresos tienen un coste. Estos costes son relativamente fáciles de calcular una vez que se han definido los recursos clave, las actividades clave y las asociaciones clave. No obstante, algunos modelos de negocio implican más costes que otros. Las compañías aéreas de bajo coste, por ejemplo, han desarrollado modelos de negocio completamente centrados en estructuras de costes reducidos.

Obviamente, los costes deben minimizarse en todos los modelos de negocio. No obstante, las estructuras de bajo coste son más importantes en algunos modelos que en otros, por lo que puede resultar de utilidad distinguir entre dos amplias clases de estructuras de costes:

- Según costes.
- Según valor.

Muchos emprendedores tienen la idea errónea de que el comercio electrónico no tiene gastos y que es suficiente con crear una página *web* comercial para que lleguen las ventas. Pero se equivocan. Al igual que en un local físico, por ejemplo, se debe pagar alquileres, suministros o seguros, una *web* necesita alojamiento, mantenimiento, unas actualizaciones etc. Además, la *web* necesitará un desempeño de horario concreto, un servicio de atención al cliente que consume tiempo y recursos y un procesamiento propio de envío de correos. También se requiere de la compra de los dispositivos *Android Tv Box* y seguramente se necesite hacerle algún tiempo de *root* o *flash* para dejar el software lo más limpio posible para la ejecución de la aplicación móvil y esto requiere manipulación física.

Para el objetivo de determinar los gastos fijos mensuales, se debe estimar adecuadamente los costes además de cuantificarlos y presupuestarlos. Se estudiará la estructura de costes en la que se analiza tanto los costes fijos como los variables.

Los costes variables en los que puede a incurrir son:

- Compra de dispositivos *Android Tv Box*.
- Servicios subcontratados, gastos de transporte de recursos humanos para instalaciones de la solución.
- Reparación de dispositivos *Android Tv Box, root* y *flash* de los mismos.

Y en la parte de los costes fijos que incurre la empresa estarían los gastos de personal y otros gastos de explotación como:

- Alquiler de oficina, donde estarían los empleados encargados de actualizaciones de las aplicaciones web y móvil.
- Publicidad.
- Suministros de la oficina (luz, agua, internet, etc.).
- Sueldos y salarios.
- Seguridad social de empleados.
- Otros gastos (diseñados de página web, programadores, dominios, servidores, etc.)

En cuanto a las economías de escala, en caso de que tenga éxito el negocio, una solución sería contratar más empleados, obtener alianzas con otras empresas, hacerla más potente.

CAPÍTULO 5
MARCO APLICATIVO

En este capítulo se describe en detalle la realización de los objetivos que componen el presente Trabajo de Grado en función de la metodología seleccionada (expuesta en el capítulo anterior) para su cumplimiento.

5.1. Metodología implementada

Para el proceso de desarrollo de la solución se implementó la metodología de desarrollo ágil Scrum, sin embargo, existen aspectos definidos en diversas fuentes sobre este método de desarrollo de software que no se adaptaron fácilmente a las características de este proyecto por lo que fueron modificados u omitidos. La metodología se implementó siguiendo las siguientes fases:

- Durante la **fase inicial de planificación** se realizó la captura de requerimientos del *producto Owner* (Equipo conformado por: Franklin Sandoval, Enrique Marin, Richard Franco) y se definió una arquitectura para la solución. Ésta estuvo sujeta a cambios, pero fue necesaria para dar comienzo a la siguiente etapa.
- Luego se procedió a la **fase de desarrollo**, en esta el objetivo fue completar el proyecto de manera incremental por medio de una serie de sub fases llamadas *sprints*, cada una de éstas tiene una corta duración, generalmente de una a dos semanas, en este intervalo de tiempo se culminan las tareas asignadas para ese *sprint*. Las tareas consideradas más complejas y que conllevan más tiempo se subdividen en tareas más pequeñas y/o simples para poder cumplir los objetivos de cada sprint.
- Las tareas a cumplir son organizadas por prioridad en una lista llamada *producto backlog*, ésta lista es revisada y se actualizan las prioridades de cada tarea antes de cada *sprint*. Una parte resaltante de esta metodología es que plantea realizar reuniones del equipo diariamente, en donde se rinde cuenta de los avances realizados, que obstáculos se presentan en la actividad actual y que cosas se planean cumplir antes

de la siguiente reunión, así como también que tarea se asumirá al culminar la que se posee en curso.

- Luego de dar por culminada la fase de desarrollo se procedió a la **fase de pruebas**. Se realizaron pruebas para medir el rendimiento de la aplicación en cuanto a funcionalidad y usabilidad y por último se realizaron las pruebas de aceptación con usuarios reales donde utilizaron la aplicación y dieron su feedback sobre la misma.

5.2. Herramientas de solución

En el desarrollo de la solución se utilizaron varias herramientas en conjunto, entre ellas se encuentran lenguajes de programación, herramientas de soporte para la metodología Scrum, herramientas de desarrollo del sistema, y los frameworks usados tanto para el desarrollo del API como de la aplicación. A continuación, se explica cada una de estas herramientas y sus características más resaltantes por la cual fueron seleccionadas.

5.2.1. Lenguajes de programación

La aplicación móvil se desarrolló en JavaScript utilizando los frameworks de desarrollo AngularJS y IONIC. Ambos frameworks especialistas en desarrollo de aplicaciones móviles híbridas, los cuales proveen todos los elementos de ftont-end y back-end necesarios para la aplicación. Se utilizó Apache Cordova como entorno de desarrollo que permite la utilización de tecnologías web para el desarrollo de aplicaciones móviles multiplataforma.

El API de consumo fue desarrollado bajo el lenguaje Java, Según Martínez (2015), Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos desarrollado por Sun Microsystems y ahora a cargo de Oracle, uno de sus objetivos principales fue proporcionar portabilidad en el software desarrollado.

La aplicación Web fue desarrollada en *Java Server Faces* ya que está especializado en el desarrollo de interfaces de usuario mediante el uso de componentes ya predefinidos, de este lenguaje se usó el framework Primefaces, como librería para los elementos visuales ya que posee un interesante conjunto de componentes planteados en XHTML, autocompletado, graficas etc. Es muy ligero, sencillo de instalar y posee una amplia documentación.

5.2.2. Herramientas de soporte para la metodología Scrum

Durante todo el proceso de desarrollo se usaron un conjunto de herramientas que dieron soporte a la metodología ágil Scrum. Estas herramientas facilitaron la organización de tareas en el proyecto y el mantenimiento del código de la solución.

Para mantener un orden, priorizar y establecer fechas límites se creó una cartelera de tareas en Redmine. Redmine es una página web que provee una interfaz para manejar proyectos mediante tareas (*Task*). Las tareas se crearon con estado "Nueva", se pasaban a estado "En progreso" y finalmente al culminarse se establecía el estado "Resuelta". Las tareas se corresponden con requerimientos, funcionalidades y errores a corregir dentro de la aplicación. También se manejó el progreso de la documentación haciendo uso de esta herramienta y la trazabilidad de la actividad: que especifica el desarrollo de la tarea.

#	Tracker	Status	Priority	Subject	Assignee
772977	Task	New	High	Desarrollo login en la mobil	Richard Franco
772976	Task	In Progress	High	Desarrollo modulos para transmitir el contenido digital segun la confugacion de la web	Richard Franco
772974	Task	Resolved	Urgent	Mejora Listener de la configuracion en parrilleras	Richard Franco
772973	Task	Resolved	High	Desarollo servicio cosulta y listener de la configuracion de las parrilleras en la web	Richard Franco
772972	Task	Resolved	Normal	Plantear estructura base proyecto Mobil	Richard Franco
772970	Task	Resolved	High	Desarrollo modulo login user	Enrique Marin
772969	Task	Resolved	Urgent	Desarrollo Modulo Parrillera en Web	Enrique Marin
772968	Task	Resolved	Normal	Desarrollo servicios web consulta y carga de imagenes	Enrique Marin

Ilustración 19 - Tablero de ciclo de vida las tareas.
Fuente: Los autores

Para llevar el control de los cambios sobre el código de la aplicación se utilizó un repositorio de código Git. Ya que permite almacenar y manejar proyectos y repositorios Git en la nube, sin costo para cinco o menos usuarios. Esta herramienta se utilizó en conjunto con el plug-in incluido en Visual Studio Code para manejar copias locales del proyecto y desplazar los cambios realizados al repositorio Git en la nube.

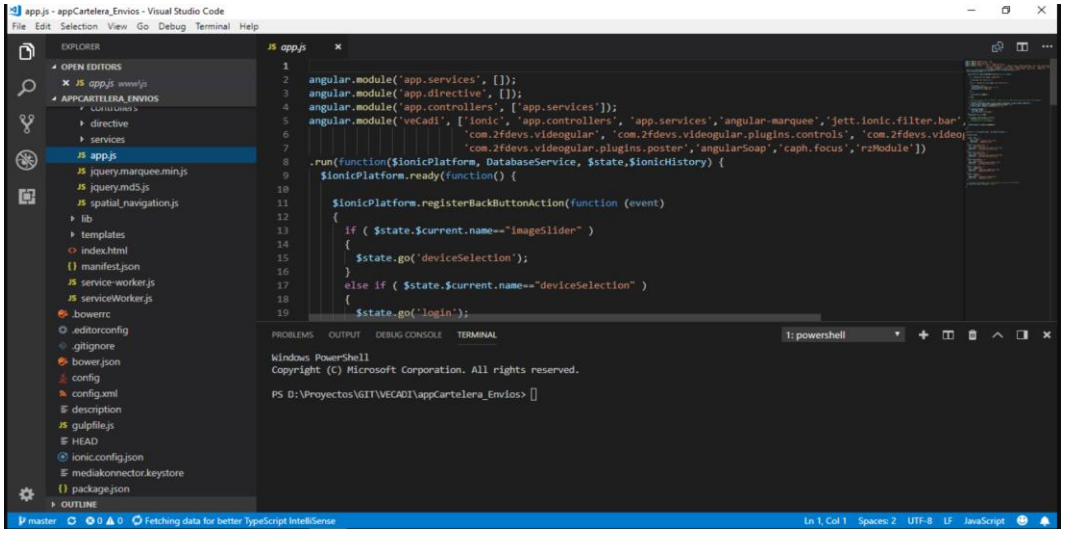


Ilustración 20 - Git en Visual Studio Code.
Fuente: Los Autores

Así mismo las sesiones diarias (*Daily's*) fueron programadas acorde a las disponibilidad del día, la mayoría eran alrededor de las 9:00 PM, con una duración aproximada de 10-15min vía *Skype* o por tlf, herramienta de software que permite comunicaciones de texto, voz y video sobre internet (VoIP). En las cuales se respondían tres preguntas básicas, ¿Que estás haciendo ahora? ¿Qué impedimentos se tienen hasta los momentos? Y ¿Que tarea se atenderá luego de culminar la actual? Para estas sesiones diarias se hizo uso de una de las herramientas de chat desktop más potentes en la actualidad, *Skype Desktop*.

5.2.3. Herramientas de desarrollo y administración del sistema

Google Cloud: es una plataforma que ofrece Google, que es utilizada para crear ciertos tipos de soluciones a través de la tecnología almacenada en la nube. *Google Cloud* se refiere al espacio virtual a través del cual se puede realizar una serie de tareas que antes requerían de *hardware* y *software* y ahora utilizan la nube de Google como única forma de acceso, almacenamiento y gestión de datos. Se utilizó esta plataforma como alojamiento del servidor de aplicaciones en el cual se configuró una máquina virtual con sistema operativo Linux, apache tomcat como servidor de despliegue y CouchDB como servicio de base de datos.

Ilustración 21 – Google Cloud - Administración y publicación de la aplicación web.

Fuente: https://console.cloud.google.com

5.3. Diseño del sistema

Luego que se definieron los requerimientos funcionales del sistema se procedió al diseño de cada uno de los componentes que se desarrollaron y su integración entre la aplicación *web* administrativa y la aplicación móvil enfocada al usuario final. Para esto se hizo uso del artefacto historias de usuario para el desarrollo del sistema.

Se definieron nueve (9) historias de usuario las cuales se dividen en dos (2) o tres (3) historias por *sprint*. El *sprint* 0 o llamado *sprint* inicial involucró reuniones de planificación y preparación de ambientes de desarrollo. Los *sprints* 1, 2 y 3 involucraron a su vez las historias correspondientes al desarrollo de los prototipos funcionales de aplicaciones. Todas estas historias de usuario fueron definidas con el equipo que representa el *product owner* del proyecto (Franklin Sandoval, Enrique Marin, Richard Franco).

Sprint 0 – Planificación y preparación de ambientes

En este *Sprint* se definieron una serie de historias de usuario para la preparación de los ambientes de desarrollo tanto móvil como *web* y de esta manera lograr la realización de los *sprints (1, 2 y 3)* de requerimientos funcionales siguientes sin inconvenientes.

Tabla 5 - Historia de usuario #1 - Sprint 0

		Historia de Usuario – App Móvil
Número: 1	**Usuario:** Desarrollador	
Nombre historia: Construcción y configuración de ambiente de desarrollo móvil		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 4	**Estimación:** 40 horas	**Iteración asignada: 0**
Programador responsable: Richard Franco		
Descripción: Instalar las herramientas y aplicaciones necesarias para iniciar el desarrollo y construcción de la aplicación móvil que forma parte de la solución propuesta.		

Fuente: Los Autores

Tabla 6 - Historia de usuario #2 - Sprint 0

		Historia de Usuario – App Web
Número: 2	**Usuario:** Desarrollador	
Nombre historia: Construcción y configuración de ambiente de desarrollo web		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 4	**Estimación:** 40 horas	**Iteración asignada: 0**
Programador responsable: Enrique Marin		
Descripción: Instalar las herramientas y aplicaciones necesarias para iniciar el desarrollo y construcción de la aplicación web que forma parte de la solución propuesta.		

Fuente: Los Autores

Sprint 1 – Carga de contenido digital y gestión de dispositivos

En el Sprint uno (1) para la aplicación web se definió un módulo con la funcionalidad de Cargar contenido digital (imágenes, videos, etc.) y un módulo con la funcionalidad de gestionar dispositivos lógicos a los cuales se le asociará dicho contenido. Aquí se definió un tiempo de 80 horas para dicho Sprint.

Tabla 7 - Historia de usuario #3 - Sprint 1

Historia de Usuario – App Web - Rol Administrador		
Número: 3	**Usuario:** Desarrollador	
Nombre historia: Carga Contenido Digital		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Estimación: 8**0 horas	**Iteración asignada: 1**
Programador responsable: Enrique Marin		
Descripción: Se desarrollará un módulo en la web que tenga la funcionalidad de cargar contenido digital (Imágenes, Videos, Gif, etc.).		

Fuente: Los Autores

Tabla 8 - Historia de usuario #4 - Sprint 1

Historia de Usuario – App Web - Rol Administrador		
Número: 4	**Usuario:** Desarrollador	
Nombre historia: Gestión de dispositivos		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Estimación: 8**0 horas	**Iteración asignada: 1**
Programador responsable: Richard Franco		
Descripción: Se desarrollará un módulo en la aplicación web que tenga la funcionalidad de crear, consultar, editar y eliminar dispositivos a los cuales se le asociará el contenido digital deseado.		

Fuente: Los Autores

Sprint 2 – Gestión de Galerías e Inicio de sesión

En el *Sprint* dos (2) para la aplicación web se definió un módulo con la funcionalidad de gestionar galerías de imágenes y/o videos y en la app móvil la funcionalidad de inicio y cierre de sesión en la app móvil y selección de los dispositivos cargados.

Tabla 9 - Historia de usuario #5 - Sprint 2

Historia de Usuario – App Web – Rol Administrador		
Número: 5	**Usuario:** Desarrollador	
Nombre historia: Modulo Galerías		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Estimación:** 80 horas	**Iteración asignada: 2**
Programador responsable: Enrique Marin		
Descripción: Se desarrollará un módulo en la aplicación web que permita crear, consultar, editar y eliminar galerías donde se podrá agrupar contenido digital previamente cargado.		

Fuente: Los Autores

Tabla 10 - Historia de usuario #6 - Sprint 2

Historia de Usuario – App Móvil – Rol Administrador		
Número: 6	**Usuario:** Desarrollador	
Nombre historia: Login		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 4	**Estimación:** 40 horas	**Iteración asignada: 2**
Programador responsable: Richard Franco		
Descripción: Se desarrollará un módulo en la app móvil que permita iniciar y cerrar sesión en la misma.		

Fuente: Los Autores

Tabla 11 - Historia de usuario #7 - Sprint 2

Historia de Usuario – App Móvil – Rol Administrador		
Número: 7	**Usuario:** Desarrollador	
Nombre historia: Selección de dispositivos		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 4	**Estimación:** 40 horas	**Iteración asignada: 2**
Programador responsable: Richard Franco		
Descripción: Se desarrollará un módulo en la app móvil que permita seleccionar de entre todos los dispositivos cargados en la aplicación web, aquel que se desee reproducir el contenido digital asignado a él.		

Fuente: Los Autores

Sprint 3 – Gestión de parrillas digitales y Reproducción de contenido digital

En el Sprint tres (3) para la aplicación web se definió un módulo con la funcionalidad de programar parrillas digitales y para la app móvil se plantea reproducir las parrillas digitales programadas.

Tabla 12- Historia de usuario #8 - Sprint 3

Historia de Usuario – App Web – Rol Administrador		
Número: 8	**Usuario:** Desarrollador	
Nombre historia: Gestión de Parrillas Digitales		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Estimación:** 80 horas	**Iteración asignada: 3**
Programador responsable: Enrique Marin		
Descripción: Se desarrollará un módulo en la aplicación web que permita programar parrillas digitales asignando galerías existentes a distintos dispositivos y asociándoles fechas de reproducción.		

Fuente: Los Autores

Tabla 33 - Historia de usuario #9 - Sprint 3

Historia de Usuario – App Móvil – Rol Administrador		
Número: 9	**Usuario:** Desarrollador	
Nombre historia: Reproducción de parrillas digitales		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alta
Puntos estimados: 8	**Estimación:** 80 horas	**Iteración asignada: 3**
Programador responsable: Richard Franco		
Descripción: En las pantallas conectadas a los dispositivos se debe reproducir el contenido digital programado según el dispositivo seleccionado y las galerías asignadas a él.		

Fuente: Los Autores

5.4. Diseño de la solución

La solución propuesta compone el desarrollo de dos aplicaciones (*Web* administrativa, móvil) el esquema de diseño que se impone se divide en dos secciones. **El Esquema de diseño móvil** impone el desarrollo de una aplicación híbrida utilizando frameworks que facilitan el desarrollo de aplicaciones móviles utilizando tecnologías *web* que llevó a diseñar la aplicación como un flujo de pantallas, cada una con funciones específicas y comunes a la vez. Se define un modelo sencillo en el que el usuario necesita estar logeado en la aplicación para el uso del prototipo funcional de solución

El enfoque de solución de Cartelera Digital llevo a desarrollar la aplicación *web* administrativa basándose en El **Esquema de diseño web** que incluye un único rol de administrador de comercios, el api *Restful* para el inicio de sesión de la app móvil, y el api *HTTP* proporcionado por *CoouchDB* para la gestión de los datos y envió de estos

desde el servidor de base de datos a cada uno de los dispositivos *Android Tv Box* que tengan corriendo la aplicación móvil.

5.4.1. Diagrama de clases

En la siguiente Figura, se presenta el diagrama de clases de la aplicación, en este se muestran las asociaciones entre los diferentes documentos que se definieron dentro del sistema, así como también los métodos principales para cada documento definido.

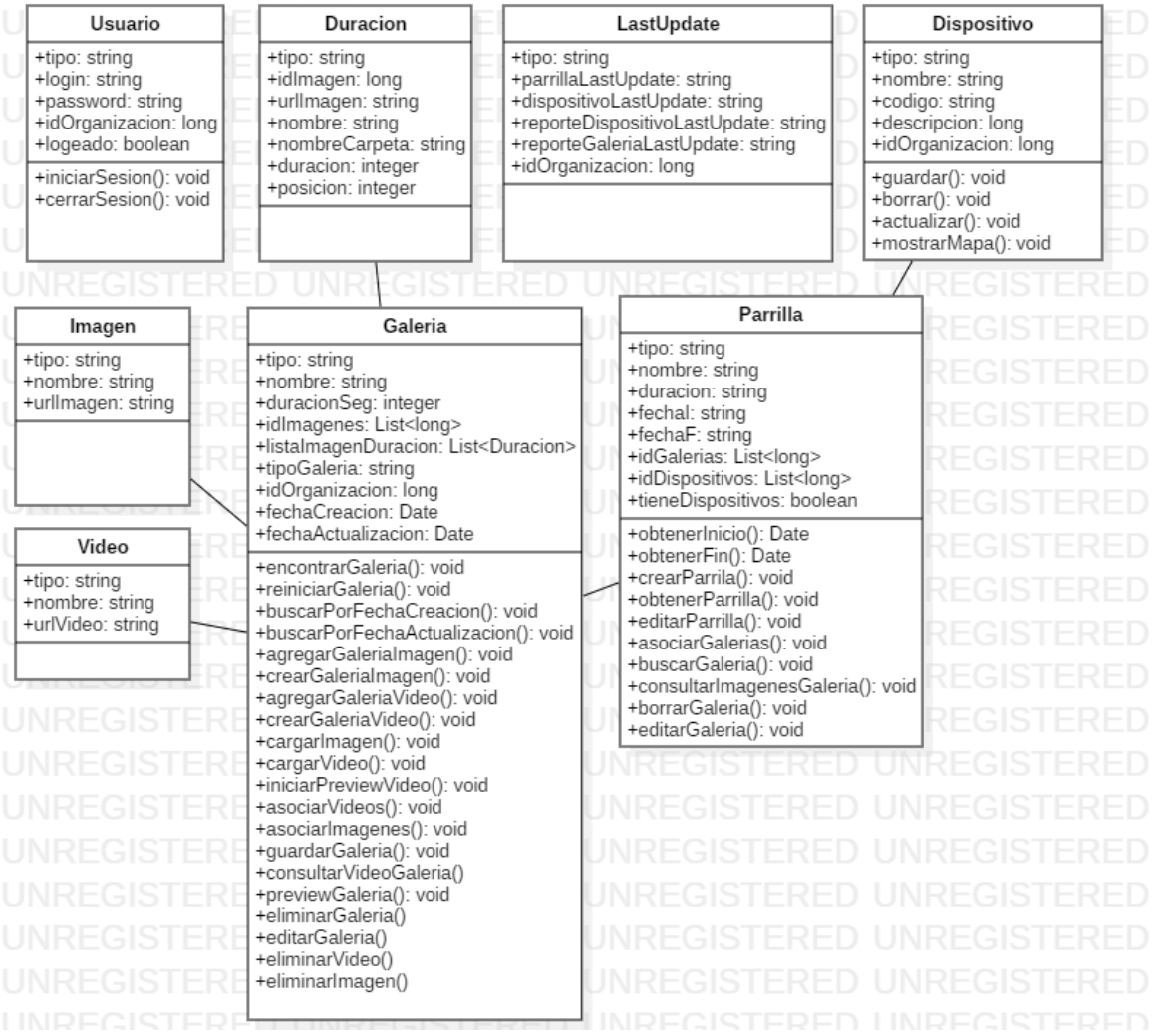


Ilustración 22 – Diagrama de clases de la solución.
Fuente: Los Autores

5.4.2. Modelo de Base de Datos

Se muestra una representación del diseño planteado en los documentos, estos almacenando la data en formato Json. Los documentos son la estructura de almacenamiento para las bases de datos no relaciones, por eso no presentan una dependencia o relación entre ellos, en este planteamiento existen unos documentos que comparten una asociación.

Dispositivo

```
{ "_id": 1,
“rev” : “adfgtr”
 "tipo": “Dispositivo”,
 "nombre": "La Ciudadela",
 "codigo": "La Ciudadela",
 "descripcion": "Descripcion",
 "coOperation": "U",
 "daUpdate": "2018-04-10",
 "idOrganizacion": "1"
```

Usuario

```
{ "_id": "1",
 "_rev": "abcd",
 "tipo": "Usuario",
 "login": "edurne",
 "password": "****",
 "email": "tesis@ucv.com",
 "idOrganizacion": "ucv",
 "logeado": true,
 "coOperation": "I"
}
```

Duracion

```
{ “tipo”: “duracion”
   "idImagen": "12",
   "urlImagen": "url/Imagen",
   "nombre": "img_20180301_114957",
   "nombreCarpeta": "carpeta",
   "duracion": 10,
   "posicion": 0
 }
```

Imagen


```
{  "_id": "1",
 "_rev": "abcd
 "nombre": "img",
 "tipo": "Imagen",
 "urlImagen": "url/Imagen",
 "urlImagenComp": "url/ImagenComp",
 "_attachments": {
  "Imagen": {
            "content_type": "image/jpeg",
             "revpos": 2 }
```


Galeria


```
{  "_id": "1",
 "_rev": "abcd",
 "tipo": "Galeria",
 "nombre": "3 spot del futuro",
  "duracionSeg": 30,
 "idImagenes": ["12" ],
  "idVideos": ["11" ],
 "listaImagenDuracion": [ {
    "idImagen": "12",
    "urlImagen": "url/Imagen",
    "nombre": "img_20180301_114957",
    "nombreCarpeta": "carpeta",
    "duracion": 10,
    "posicion": 0
  }
 ]
```


Parrilla


```
{  "_id": "1",
 "_rev": "abcd",
 "tipo": "Parrilla",
 "nombre": "1 spot",
 "duracion": "00 : 11 : 48",
 "fechaI": "20/03/2018",
 "fechaF": "20/03/2018",
 "coOperation": "I",
 "daUpdate": "2018-06-16",
 "idGalerias": [ "1" ],
 "idDispositivos": {   "id": 1  },
 "tieneDispositivos": true,
```


Parrilla contiene una lista de los id de las Galerias

Video


```
{  "_id": "1",
 "_rev": "abcd
 "nombre": "video",
 "tipo": "Video",
 "urlVideo": "url/mp4",
 "urlVideoComp": "url/ImagenComp",
 "_attachments": {
  "Imagen": {
            "content_type": "video/mp4",
             "revpos": 2 }
```


Galeria contiene una lista de los id de las Imágenes / Video

5.5. Diseño del API

El API implementado está dividido entre los desarrollados en java y los ofrecidos por el couchDB como recurso HTTP. Ambas plataformas usan el formato JSON como entrada y salida de los servicios, nuestra solución no tiene el enfoque de un API REST como normalmente se conoce, es decir, una url que accede a un recurso en formato JSON. Este desarrollo se basó en las peticiones HTTP que la base de datos de forma directa, el *Front-End* accede a los métodos por referencia directa en *Java Server Faces*, a su vez las operaciones en Middleware internamente acceden al repositorio de instrucciones ofrecidas por el framework.

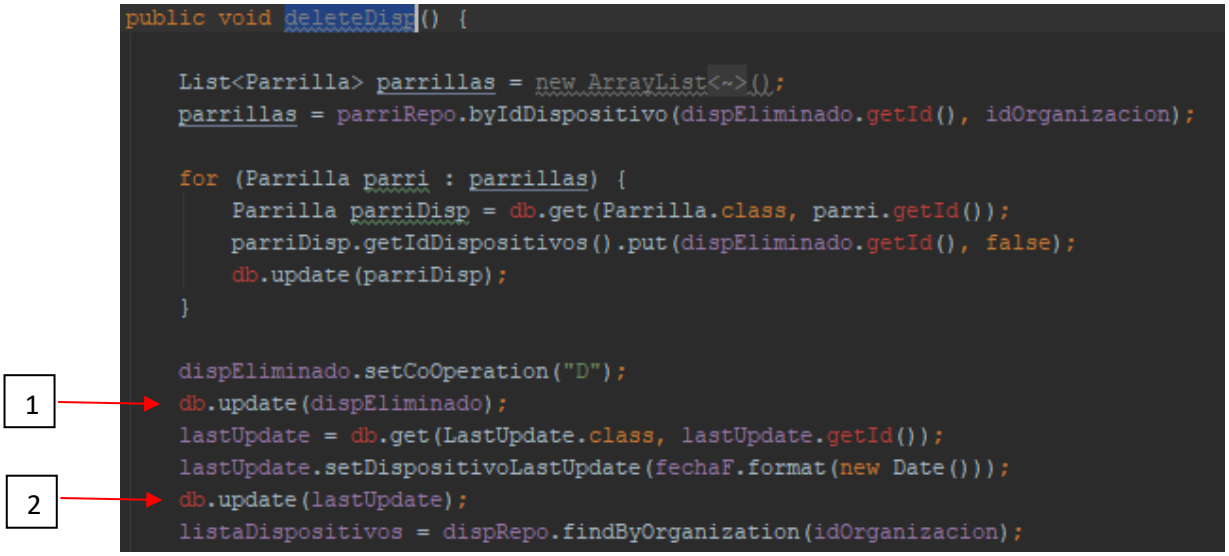
Planteando un caso se tiene que en la vista dispositivos.xhtml, se encuentra la operación borrarDispositivo, en código JSF seria:

```
<p:commandButton id="deleteButton" value="#{msgs.eliminar}"
    styleClass="botonVerde botonGris btn-format aceptar"
    style="..."
    action="#{dispositivoBean.borrarDispositivo()}" update="formDisp">
</p:commandButton>
```

Ilustración 23 – Ejemplo de llamado a una operación en JSF.

Fuente: Los Autores

Aquí se invoca al método sin necesidad de acceder a una url, esta es la forma de invocar las operaciones desde la web. Ahora en Middleware para realizar la tarea se debe invocar la petición HTTP, seria:




```
public void deleteDisp() {

    List<Parrilla> parrillas = new ArrayList<~>();
    parrillas = parriRepo.byIdDispositivo(dispEliminado.getId(), idOrganizacion);

    for (Parrilla parri : parrillas) {
        Parrilla parriDisp = db.get(Parrilla.class, parri.getId());
        parriDisp.getIdDispositivos().put(dispEliminado.getId(), false);
        db.update(parriDisp);
    }

    dispEliminado.setCoOperation("D");
    db.update(dispEliminado);
    lastUpdate = db.get(LastUpdate.class, lastUpdate.getId());
    lastUpdate.setDispositivoLastUpdate(fechaF.format(new Date()));
    db.update(lastUpdate);
    listaDispositivos = dispRepo.findByOrganization(idOrganizacion);
```

Ilustración 24 – Ejemplo de llamado al API HTTP de Couchdb desde Java.

Fuente: Los Autores

Se resalta en la Figura 22 las líneas (1 - db. update(dispEliminado)) y (2 - db. update(lastUpdate)) para indicar que la variable db es una instancia de la conexión a la base datos.

```
private CouchDbConnector db = null;
```

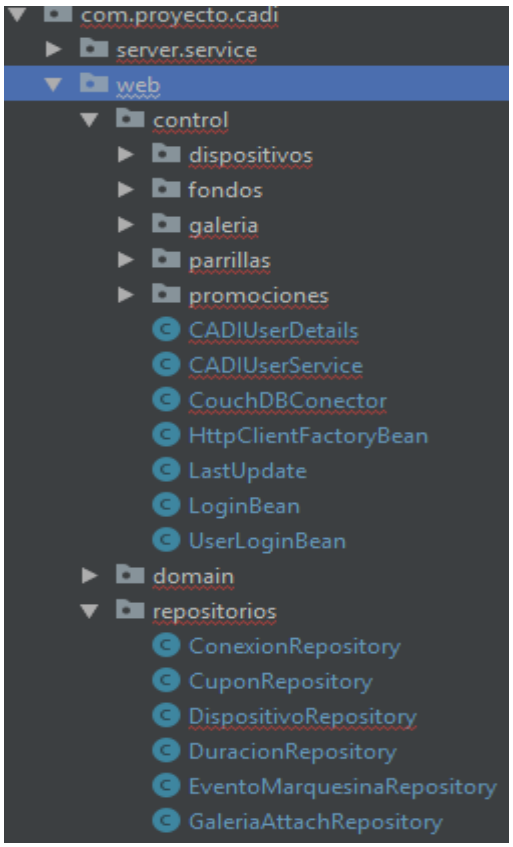
Ilustración 25 – Instancia del conector de CouchDB desde Java.

Fuente: Los Autores

Esta ofrece los métodos (create, delete, update, get) que representan las peticiones HTTP para acceder a los recursos con el comportamiento API estándar. En el caso de móvil se acceden a las url directamente sin necesidad de un framework que tramite los request.

5.5.1. Estructura de directorios

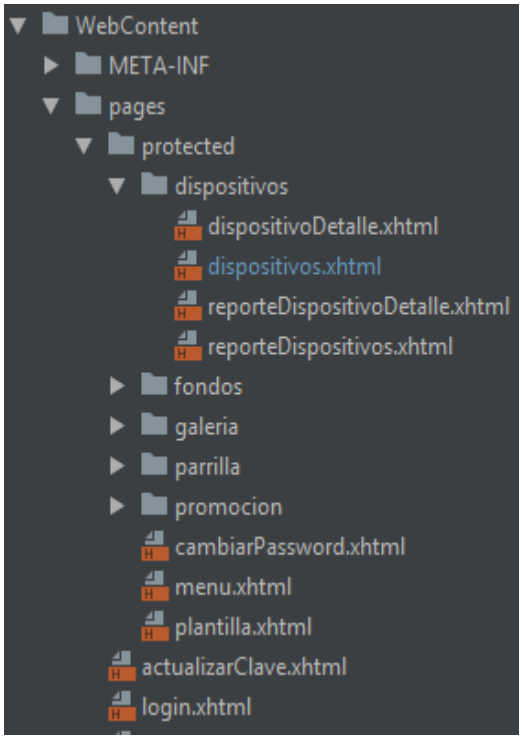
Al plantear la estructura base de proyecto WEB basados en el framework de *spring*, este genera una organización estándar para el uso correcto de carpetas y archivos de configuración. Además, que contiene una carpeta *Webcontent* donde internamente contaba con la distribución de carpetas correcta, para contener las paginas en JSF necesarias. Se adaptó a esta organización dado que logró cumplir con todas las necesidades que se presentaron en Middleware y Front-End. Se tiene que:



Middleware: aquí se observa como los servicios están separados de los controladores de forma ordenada y comprensible, que los repositorios están todos ubicados en una sola carpeta, que los controladores están separados en carpetas acorde a los recursos que respresentan.

Ilustración 26 – Estructura Proyecto Web Modulo Middleware.

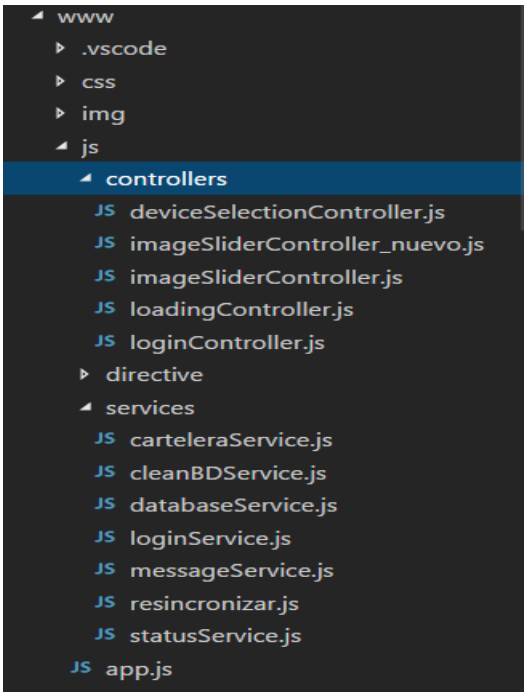
Fuente: Los Autores



Front-End: aquí se agrupó las vistas en carpetas basados en la funcionalidad o recurso que representan, permitiendo asi tener una estructura praticamente igual que middleware lo que permite una fácil comprensión de la estructura del proyecto en general.

Ilustración 27 – Estructura Proyecto Web Modulo Front-End.

Fuente: Los Autores



App móvil: aquí se agrupó las vistas, los controladores y las llamadas a los servicios en carpetas basados en la funcionalidad o recurso que representan, permitiendo asi tener una estructura praticamente igual al de la app web lo que permite una fácil comprensión de la estructura del proyecto en general.

Ilustración 28 – Estructura Proyecto Móvil.

Fuente: Los Autores

5.5.2. Documentación del API

Como se ha mencionado antes, el planteamiento no es un API estándar, se contó con dos (2) servicios para el recurso Usuario que cumplen con esa estructura pero en la mayoría del desarrollo se utilizó las urls ofrecidas por el la base de datos CouchDB, teniendo eso en cuenta se explican las rutas de la Base de Datos de forma genérica con solamente un ejemplo ya que es la misma estructura de ruta para todos los recursos, se tiene como api:

Método	Ruta	Descripción
POST	/userService/register	registra un usuario
POST	/userService/login	usuario accede a la aplicación
CouchDB API		
GET	/db/document	obtiene un document
PUT	/db/document	actualiza un documento
DELETE	/db/document	borra un documento
POST	/db/document	registra un documento, cabe señalar que en el body se debe especificar el tipo.
GET	/cadi_bd/_design/Galeria/_view/all	obtiene todas las galerias
POST	/cadi_bd/new	registra un documento, cabe señalar que en el body se debe especificar el tipo.

5.6. Implementación y diseño de prototipo de aplicación web

Java Server Faces es un *framework* MVC (Modelo-Vista-Controlador) bastante robusto, el cual permite que exista una abstracción entre la aplicación y el acceso a los datos. *JSF* ofrece la flexibilidad de que se puede integrar con distintos manejadores de base de datos incluso sistemas de bases de datos no relacionales y la integración con

una amplia gama de *frameworks* para el desarrollo de los componentes de interfaz de usuario.

Java Server Faces tiene como objetivos de diseño:

1. Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.
2. Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.
3. Proporcionar un modelo de JavaBeans para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.
4. Definir APIs para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.
5. Especificar un modelo para la internacionalización y localización de la interfaz de usuario.
6. Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

5.6.1. Primefaces y Java Server Faces

Primefaces es considerada la más completa biblioteca de componentes de interfaz de usuario para *Java Server Faces*. Es de código abierto y cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web.

Primefaces Provee un conjunto de más de cien (100+) componentes de interfaz de usuario muy sencillos de utilizar que pueden adaptarse a cualquier tipo de requerimiento.

Todos los componentes son de código abierto y libres para su uso.

Ayuda en la productividad haciendo que el tiempo de desarrollo se invierta en la lógica de negocios y no en la creación de interfaces de usuario complejas.

Proporciona una gran variedad de temas y plantillas adaptables para el diseño y *look and feel* de las aplicaciones.

Ofrece excelentes experiencias de usuario móviles al proveer elementos con diseño *responsive* optimizado.

```
<dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>6.2</version>
</dependency>
```

```
xmlns:p="http://primefaces.org/ui"
```

Ilustración 29 - Integración Primefaces con Java Server Faces.

Fuente: Los Autores

En la Figura 25 se puede ver una sección donde se hizo uso de componentes de Primefaces

Ilustración 30 – Ejemplo Primefaces.

Fuente: Los Autores

5.6.2. CouchDB y Ektorp

CouchDB es un gestor de bases de datos de código abierto que se enfoca en su facilidad de uso y de poseer una arquitectura escalable. *CoudhDB* tiene una arquitectura no relacional (*NoSQL*) orientada a documentos. Utiliza *JavaScript Object Notation* (*JSON*) para el almacenamiento de los datos, *JavaScript* como lenguaje de consulta y *Hypertext Transfer Protocol* (*HTTP*) como *API*.

Una de las características principales por las cuales se decidió utilizar CouchDB como motor para el almacenamiento fue por su arquitectura distribuida con replicación. CouchDB fue diseñado tomando en cuenta la replicación bidireccional o **sincronización** y la operación *offline*. Con esto, se logró sincronización inmediata del contenido digital que es gestionado desde la aplicación web y reproducido por la aplicación móvil.

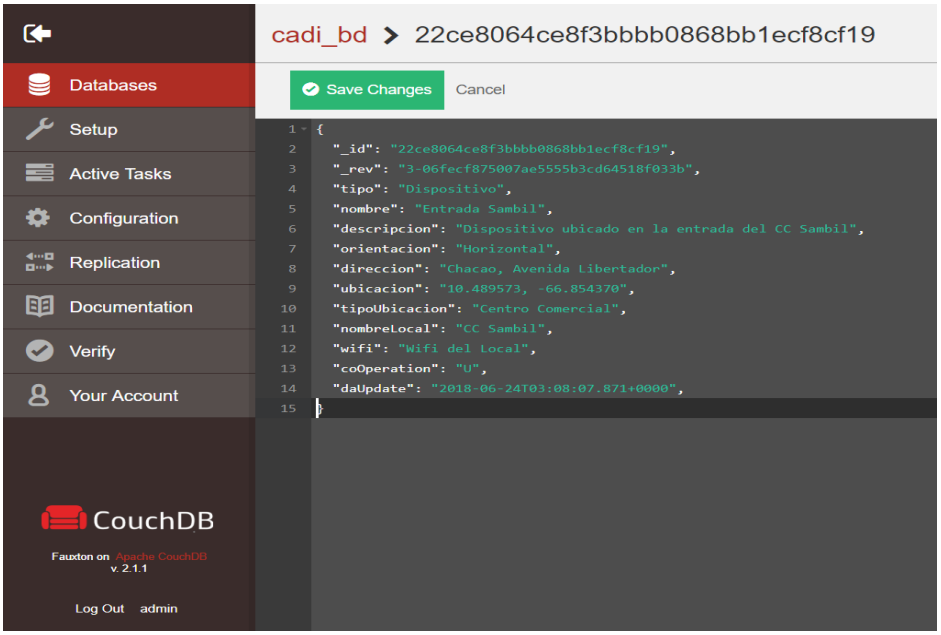


Ilustración 31 – Ejemplo de un documento CouchDB.

Fuente: Los Autores

Ektorp es un *API Java* de persistencia que utiliza CouchDB como motor de almacenamiento. El objetivo principal de Ektorp es combinar las funcionalidades ofrecidas por *Java Persistence API* (JPA) con la simplicidad y flexibilidad que CouchDB provee.

Dentro de las bondades que Ektorp ofrece, tiene un gran manejo de objetos de dominio. Gracias al poderoso mapeo de objeto-JSON (*JSON-Object Mapping*) que provee la clase *Jackson*, se pueden crear objetos Java a partir de objetos JSON de manera muy sencilla. También trabaja perfectamente con sistemas de bases de datos sin esquema. Como CouchDB por ser NoSQL no tiene un esquema definido, las configuraciones y ajustes son transparentes y automáticos lo que permite ahorrar tiempos valiosos en el desarrollo de las aplicaciones.

```
57      public CouchDbConnector createConector() {
58          HttpClient httpClient;
59          CouchDbConnector db = null;
60          try {
61              httpClient = new StdHttpClient.Builder().url("http://localhost:5984").username("admin").password("Varsovia")
62                      .build();
63              CouchDbInstance dbInstance = new StdCouchDbInstance(httpClient);
64              db = dbInstance.createConnector("cadi_bd", true);
65          } catch (MalformedURLException e) {
66              // TODO Auto-generated catch block
67              e.printStackTrace();
68          }
69          return db;
70      }
71
72      public CouchDbConnector createConectorAttach() {
73          HttpClient httpClient;
74          CouchDbConnector db = null;
75          try {
76              httpClient = new StdHttpClient.Builder().url("http://localhost:5984").username("admin").password("Varsovia")
77                      .build();
78              CouchDbInstance dbInstance = new StdCouchDbInstance(httpClient);
79              db = dbInstance.createConnector("cadi_bd_attach", true);
80          } catch (MalformedURLException e) {
81              // TODO Auto-generated catch block
82              e.printStackTrace();
83          }
84          return db;
85      }
86
87  }
88
```

Ilustración 32 – Configuración de CouchDB con Ektorp.

Fuente: Los Autores

5.6.3. Desarrollo en Java Server Faces

El código de la aplicación consiste en un conjunto de vistas, modelos y controladores, haciendo así que las vistas representen cada pantalla en un archivo XHTML que define la disposición, de la forma y color de los elementos de interfaz de usuario como botones, campos de datos y títulos. Dicho esto, a continuación, se explica por pantalla y por funcionalidad el flujo del prototipo funcional de aplicación web desarrollada en un manual de usuario para el uso de la aplicación que se adjunta en los anexos del trabajo de grado. Manual de usuario de app web.

5.7. Implementación y diseño de prototipo funcional de aplicación móvil

Por la sencillez de las vistas de la aplicación móvil, las cuales básicamente consisten en dos formularios: uno para el inicio de sesión de la aplicación y otro para la selección del dispositivo del cual se desee reproducir el contenido digital programado, y una vista para el reproductor de contenido multimedia, se decidió desarrollarla como una aplicación móvil híbrida. Por ser una aplicación hibrida se desarrolló en su totalidad en lenguaje de programación JavaScript. Se utilizaron *frameworks* que facilitan el desarrollo de aplicaciones móviles hibridas como Apache Cordova que compila código JavaScript, HTML y CCS y lo transforma en código nativo para las distintas plataformas móviles como Android y IOS. El *backend* de la aplicación se desarrolló en AngularJS, poderoso framework JavaScript que permite que vistas HTML estáticas se conviertan en vistas dinámicas por sus directivas *ng-model* y *ng-bind.* Los componentes de interfaz de usuario fueron desarrollados utilizando el framework IONIC fuertemente compatible con Cordova y AngularJS. Por último, para el manejo de los datos se utilizó el plugin PouchDB que permite que cada dispositivo que tenga instalada la aplicación sea un nodo cliente del servidor CoudhDB y con él se puede hacer la sincronización en vivo (*live replication*) de los datos entre cliente y servidor.

5.7.1. Apache Cordova

Cordova es un framework de desarrollo de aplicaciones móviles el cual permite construir aplicaciones para múltiples plataformas móviles como *Android*, *IOS* y *Windows Phone* utilizando código HTML5, CCS3 y JavaScript en lugar de *Api's* específicos para cada plataforma. Por la naturaleza de la aplicación, la cual no necesita

de usar componentes específicos de los dispositivos sino está basada em almacenar y reproducir contenido multimedia, Cordova se adapta completamente al requerimiento porque permite desarrollar la aplicación con tecnologías sencillas de utilizar como HTML, JavaScript y CCS, y obtener los resultados esperados en muy poco tiempo. El mismo código fuente puede ser utilizado para múltiples plataformas móviles y además provee gran cantidad de plugins para la utilización de algunos componentes de los dispositivos móviles como la cámara, el calendario, memorias de almacenamiento entre otros.

```
$ npm install -g cordova

$ cordova create MyApp

$ cd MyApp
$ cordova platform add browser
```

Ilustración 33 – Ejecución de aplicación con Apache Cordova.

Fuente: Los Autores

5.7.2. AngularJS

AngularJS es un *framework* JavaScript que permite que código HTML estático pueda extenderse mediante el uso de directivas y asignar (*bind*) el valor de los componentes HTML (formularios, áreas de texto, selectores, etc.) a los datos de la aplicación y viceversa logrando tener vistas dinámicas. Se decidió utilizar AngularJS porque su sencillez lo que se convirtió en un rápido desarrollo logrando tener resultados extraordinarios en poco tiempo.

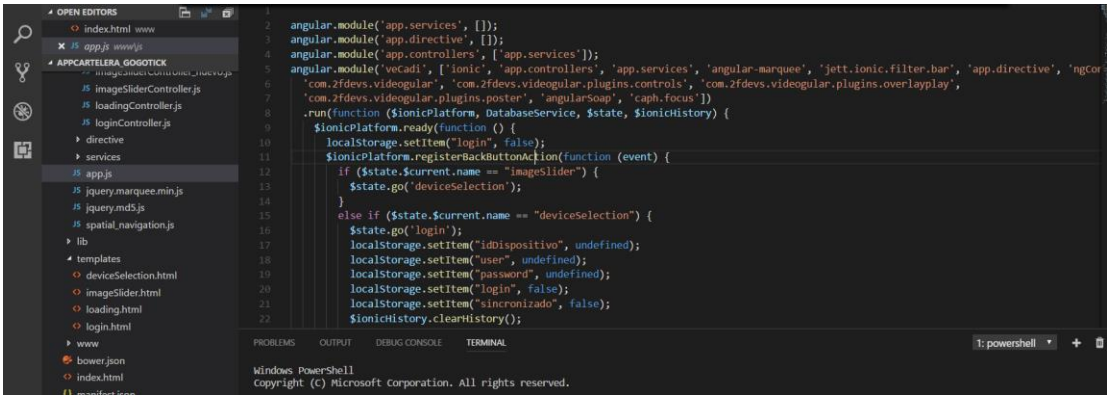


Ilustración 34 – Configuración de la app con AngularJS

Fuente: Los Autores

Por la flexibilidad y extensibilidad de AngularJS se pudo integrar fácilmente con otros API'S como IONIC y PoudchDB lo que permitió manejar fácilmente toda la lógica de la aplicación móvil. Al iniciar la aplicación se ejecutan todas las funciones que se necesitan como la sincronización de los datos del servidor y la visualización de los elementos de las vistas. También se ejecutan las llamadas al API desarrollado para hacer el inicio de sesión de la aplicación.

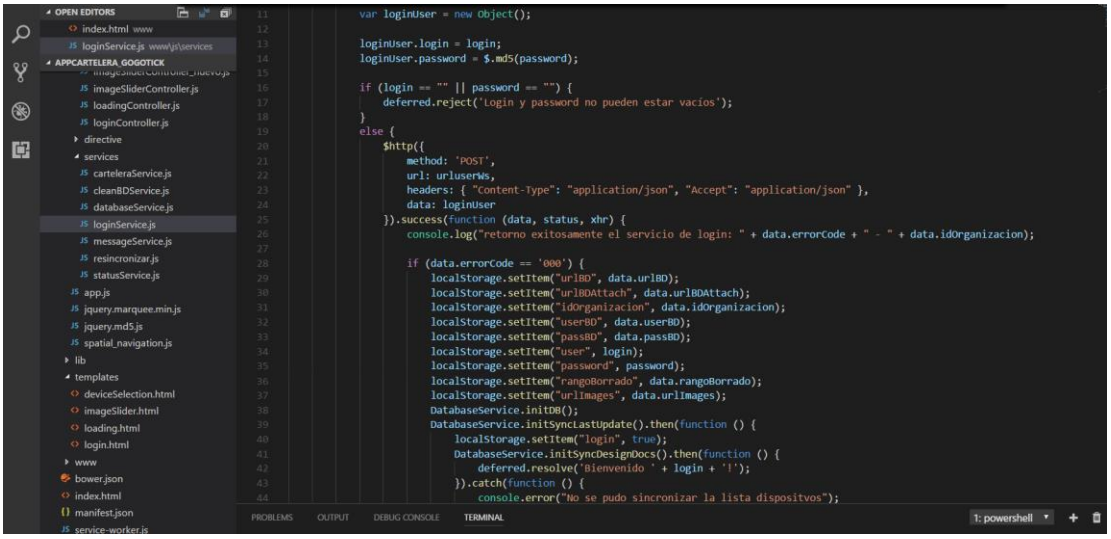


Ilustración 35 – Consulta del api desde AngularJS

Fuente: Los Autores

Una vez se obtiene toda la información programada desde la app web, AngularJS provee una amplia variedad de reproductores de contenido multimedia para la visualización de las publicidades. Por último y muy importante la actualización del contenido a mostrar, una vez las programaciones son editadas y sincronizadas, AngularJS se encarga de actualizar el reproductor de contenido multimedia gracias a sus directivas.

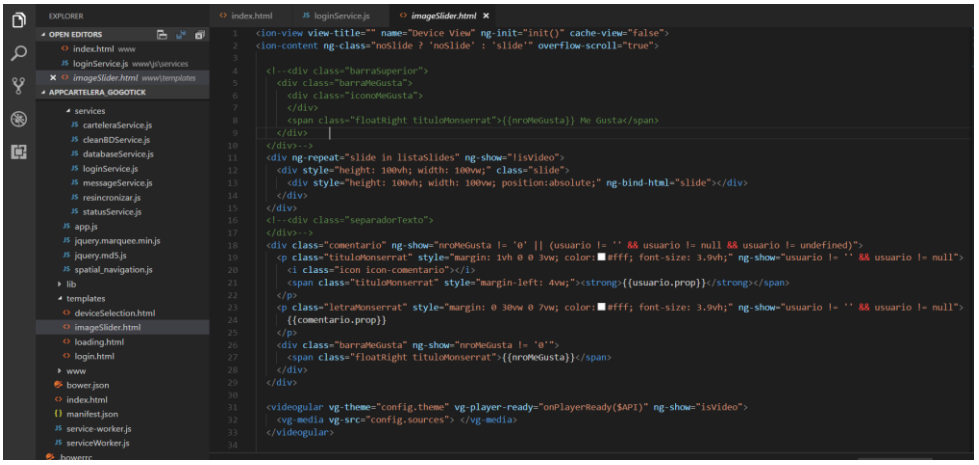


Ilustración 36 – Ejemplo de AngularJS.

Fuente: Los Autores

5.7.3. Ionic

Ionic es un completo SDK (*Software Development Kit)* de código abierto que provee herramientas y servicios para la construcción de aplicaciones móviles híbridas utilizando tecnologías *web* como HTML5, CCS y SaSS. Está diseñado sobre AngularJS y Apache Cordova lo que lo hizo el framework más adecuando para la construcción de las interfaces de usuario de la aplicación.

Con la utilización de Angular, Ionic provee múltiples componentes configurables y métodos para interactuar con ellos, como por ejemplo hacer *scroll* entre listas de miles de *ítems* sin que el rendimiento de la aplicación se vea comprometido.

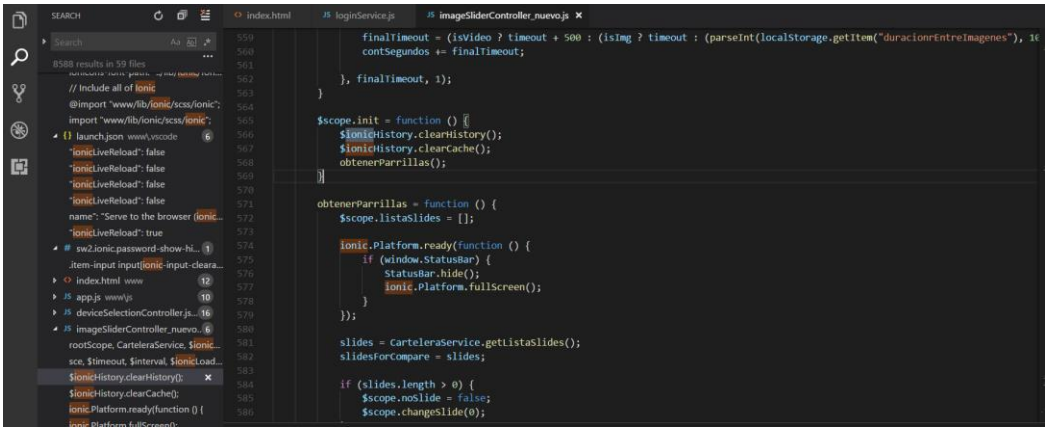


Ilustración 37– Ejemplo de IONIC.

Fuente: Los Autores

5.7.4. PouchDB

PouchDB conocida como “La base de datos que Sincroniza”, es una base de datos basada en JavaScript de código abierto que fue creada con la intención de ayudar a los desarrolladores a construir aplicaciones para que trabajen en modo sin conexión (*offline*) tan bien como lo harían con conexión a internet (*online*). PouchDB permite que las aplicaciones almacenen los datos localmente mientras están *offline* y luego los sincroniza con el servidor de CouchDB cuando la aplicación vuelva a estar *online* y con esto se logra que las aplicaciones estén operativas en todo momento.

Se utilizó PouchDB para almacenar y sincronizar todo el contenido multimedia que se cargue desde la aplicación web. Una vez se programe una parrilla digital desde la web y se gestiona a que dispositivos se quiere reproducir dicho contenido, desde la app móvil se elige alguno de esos dispositivos donde comienza todo el proceso de sincronización de contenido. Una vez descargado, se empieza la reproducción según la programación fijada desde la web. Además, PouchDB queda escuchando cualquier

cambio que se realice en el servidor de CouchDB y de ocurrir, es inmediatamente sincronizado y se actualiza automáticamente la reproducción.

PoduchDB permitió que una vez descargado el contenido y sus programaciones de reproducción, la aplicación no necesite de internet para mostrar el contenido lo cual es muy conveniente para lugares con poca o mala señal y solo se necesite conexión a internet para actualizar el contenido de haber sido modificado desde la *web*. De esta manera los clientes pueden planificar el contenido multimedia que deseen mostrar y el tiempo (días, semanas, meses) que quieran reproducirlo y estar completamente seguros que sus publicidades van a ser transmitidas según deseen.

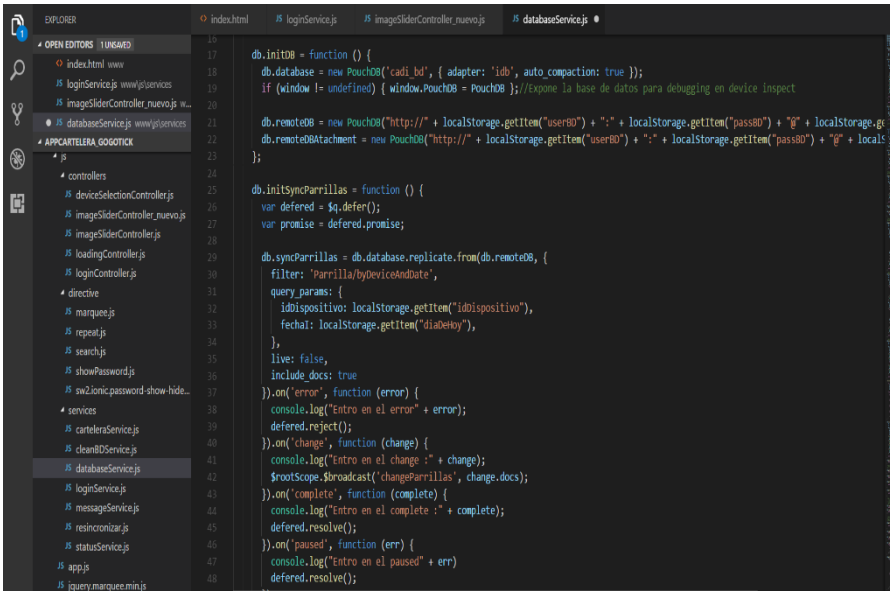


Ilustración 38 – Configuración de PouchDB.

Fuente: Los Autores

5.7.5. Desarrollo de aplicaciones móviles híbridas en AngularJS

Por ser desarrollada con tecnologías web, el código de la aplicación es muy parecido a la lógica de la aplicación web donde se tiene un conjunto de vistas, modelos

y controladores. Las vistas representan cada pantalla donde se muestran los elementos de interfaz de usuario como botones, campos de datos y reproductor multimedia. Dicho esto, a continuación, se explica por pantalla y por funcionalidad el flujo del prototipo funcional de aplicación móvil desarrollada, en un manual de usuario para el uso de la aplicación que se encuentra en los anexos del trabajo de grado. Manual de usuario móvil.

5.8. Pruebas funcionales y resultados

Según Oré (2009), se denominan pruebas funcionales a las pruebas de software que tienen por objeto probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de calidad con apoyo de algunos usuarios finales de la aplicación, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción. También se les denomina pruebas de comportamiento o pruebas caja negra, ya que los analistas se enfocan en el análisis de los datos de entrada y su correspondencia contra los parámetros de salida.

Las pruebas realizadas a la solución se dividen en tres (3) grupos: las pruebas unitarias, realizadas sobre el código durante el desarrollo de la aplicación, las pruebas funcionales, realizadas al considerarse como completo el desarrollo de un *sprint* y las pruebas de usuario final, realizadas entre usuarios afines al área de comercio y publicidad con el fin de conocer las opiniones generales de la aplicación.

5.8.1. Pruebas unitarias

En cada iteración del desarrollo las funciones y procedimientos generados fueron sometidos a pruebas unitarias, para asegurar su correcto funcionamiento y la validez de sus valores de retorno.

P.U. 1. Módulo de Imágenes (Web): Cargar File (Imagen/Gif/Video)
Entrada: Se selecciona un archivo de tipo (Imagen/Gif/Video).
Pruebas: Comprobar que el archivo exista en la base de datos.
Salida: Confirmar que el archivo sea visible desde la web.

Tabla 14 – Prueba Unitaria 1

Fuente: Los Autores

P.U. 2. Módulo de Imágenes (Web): (Eliminar Archivo)
Entrada: Se selecciona un archivo.
Pruebas: Se eliminar el archivo seleccionado.
Salida: Confirmar que al consultar las galerías no salga el archivo eliminado.

Tabla 15 – Prueba Unitaria 2

Fuente: Los Autores

P.U. 4. Módulo de Imágenes (Web): (Crear Dispositivo)
Entrada: Formulario de datos de entrada.
Pruebas: Al insertar todos los campos en el formulario se debe registrar el dispositivo en base de datos.
Salida: Al consultar los dispositivos debe aparecer el agregado.

Tabla 16 – Prueba Unitaria 3

Fuente: Los Autores

P.U. 5. Módulo de Imágenes (Web): (Consultar Dispositivos)
Entrada: Se selecciona la empresa a consultarle los dispositivos.
Pruebas: Se consultan todos los dispositivos asociados a la empresa.
Salida: Se muestran todos los dispositivos asociados a la empresa.

Tabla 17 – Prueba Unitaria 4

Fuente: Los Autores

P.U. 6. Módulo de Imágenes (Web): (Editar Dispositivo)
Entrada: Se selecciona un dispositivo.
Pruebas: Se elimina el dispositivo seleccionado.
Salida: Al agregar o consultar los dispositivos muestre los datos nuevos.

Tabla 18 – Prueba Unitaria 5

Fuente: Los Autores

P.U. 7. Módulo de Imágenes (Web): (Eliminar Dispositivo)
Entrada: Se selecciona un dispositivo.
Pruebas: Comprobar que el dispositivo cambio de estatus en base de datos (tenga status D).
Salida: Confirmar que al consultar los dispositivos no se muestre el eliminado.

Tabla 19 – Prueba Unitaria 6

Fuente: Los Autores

P.U. 8. Módulo de Imágenes (Web): (Crear Galería)

Entrada: Formulario entrada.
Pruebas: Se insertan los datos del formulario (nombre, imágenes, datos cliente) y se almacena en base de datos.
Salida: Confirmar que exista la galería con las imágenes asociada en base de datos.

Tabla 20 – Prueba Unitaria 7

Fuente: Los Autores

P.U. 9. Módulo de Imágenes (Web): (Consultar Galería)
Entrada: Se selecciona la empresa a consultarle las galerias.
Pruebas: Se consultan todas las galerías asociados a la empresa.
Salida: Se muestran todas los galerias asociados a la empresa.

Tabla 21 – Prueba Unitaria 8

Fuente: Los Autores

P.U. 10. Módulo de Imágenes (Web): (Editar Galería)
Entrada: Se selecciona una galería.
Pruebas: Se modifican los datos de la galería y se almacena en base de datos.
Salida: Al consultar las galerías se muestre con los datos nuevos.

Tabla 22 – Prueba Unitaria 9

Fuente: Los Autores

P.U. 11. Módulo de Imágenes (Web): (Eliminar Galería)
Entrada: Se selecciona una galería.
Pruebas: Se elimina la galería seleccionada.

Salida: Al consultar las galerías no se muestre la eliminada.

Tabla 23 – Prueba Unitaria 10

Fuente: Los Autores

P.U. 12. Módulo de Imágenes (Móvil): (Iniciar Sesión)
Entrada: Se insertan los campos de entrada.
Pruebas: Se verifican los campos de entrada y se inicia sesión.
Salida: Se muestra una pantalla con los dispositivos asociados.

Tabla 24 – Prueba Unitaria 11

Fuente: Los Autores

P.U. 13. Módulo de Imágenes (Móvil): (Cerrar Sesión)
Entrada: Se ejecuta la acción de cerrar sesión.
Pruebas: Se presiona el botón para salir de la sesión.
Salida: Se muestra la pantalla de inicio de sesión.

Tabla 25 – Prueba Unitaria 12

Fuente: Los Autores

P.U. 13. Módulo de Imágenes (Móvil): (Cerrar Sesión)
Entrada: Se ejecuta la acción de cerrar sesión.
Pruebas: Se presiona el botón para salir de la sesión.
Salida: Se muestra la pantalla de inicio de sesión.

Tabla 26 – Prueba Unitaria 13

Fuente: Los Autores

P.U. 14. Módulo de Imágenes (Móvil): (Reproducir Contenido Digital)
Entrada: Se selecciona un dispositivo.
Pruebas: Para el dispositivo seleccionado consultar las galerías con el contenido digital que se desea reproducir.
Salida: Se reproduce el contenido digital de la galería seleccionada en el dispositivo.

Tabla 27 – Prueba Unitaria 14

Fuente: Los Autores

P.U. 15. Módulo de Imágenes (Web): (Gestión de Parrillas Digitales)
Entrada: Se insertan los campos del formulario (galería, dispositivo, fecha).
Pruebas: Completados los campos del formulario se registra en base de datos la parrilla con las galerías y dispositivos asociados para la fecha seleccionada.
Salida: Confirmar que exista la parrilla con las galerías asociadas en base de datos.

Tabla 28 – Prueba Unitaria 15

Fuente: Los Autores

P.U. 16. Módulo de Imágenes (Móvil): (Gestión de Parrillas Digitales)
Entrada: Datos de la parrilla, dispositivos y fechas a mostrar.
Pruebas: Se consultan las fechas de la parrilla y en que dispositivos asociados debe reproducirse.
Salida: En el dispositivo seleccionado para la fecha insertada debe mostrar la galería con el contenido digital cargado.

Tabla 29 – Prueba Unitaria 16

Fuente: Los Autores

5.8.2. Pruebas de funcionalidad

Carga de Archivo

Ilustración 39 – Se muestra la bandeja imágenes vacía.

Fuente: Los Autores

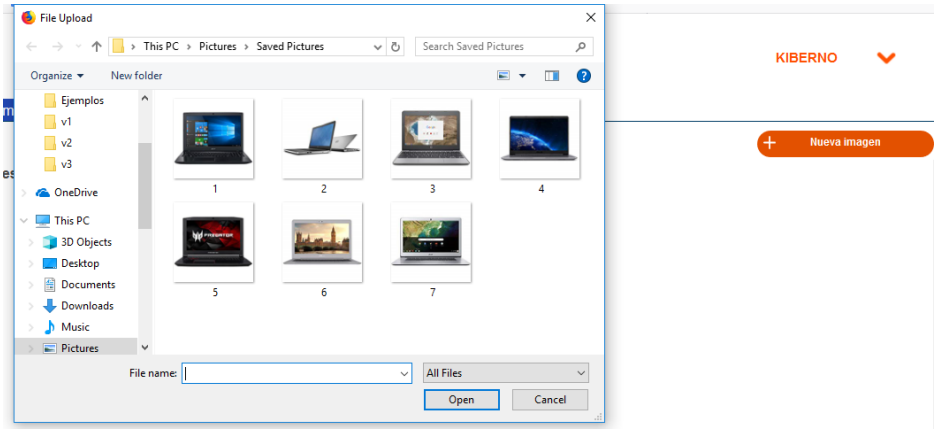


Ilustración 40 – Presionando Nueva imagen/video se muestra un explorador para seleccionar uno o varios archivos a cargar.

Fuente: Los Autores

Ilustración 41 – Carga exitosa de múltiples imágenes/videos.

Fuente: Los Autores

Eliminar Archivo

Ilustración 42 – Seleccionando archivo a eliminar.

Fuente: Los Autores

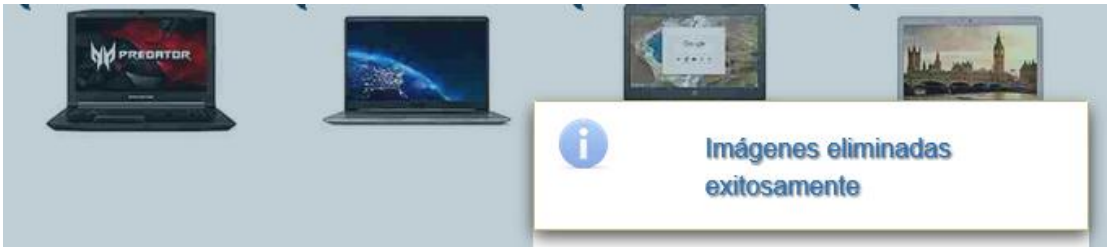


Ilustración 43 – Se muestra mensaje confirmando que el borrado fue exitoso.

Fuente: Los Autores

Creando Dispositivo

Ilustración 44 – Se muestra la tabla dispositiva.

Fuente: Los Autores

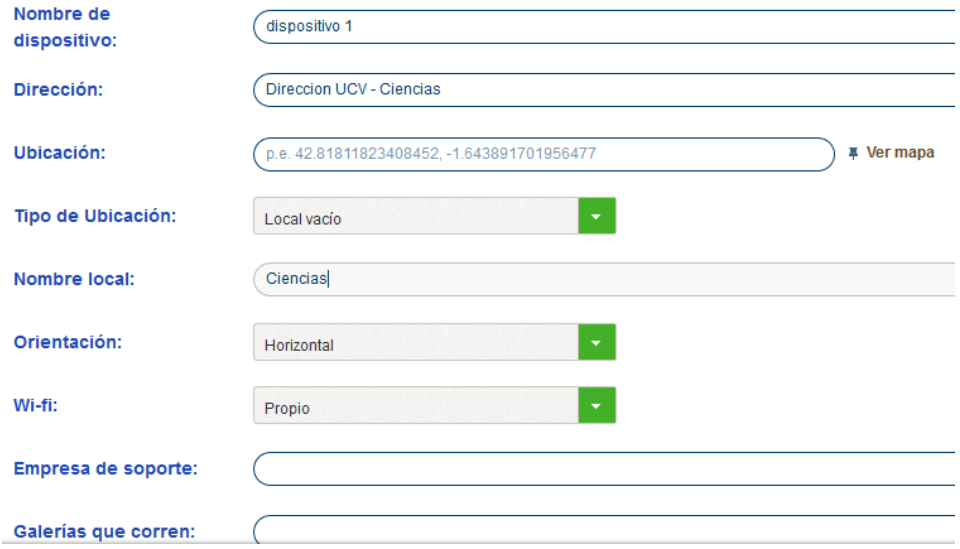


Ilustración 45 – Formulario de entrada para los datos del dispositivo.

Fuente: Los Autores

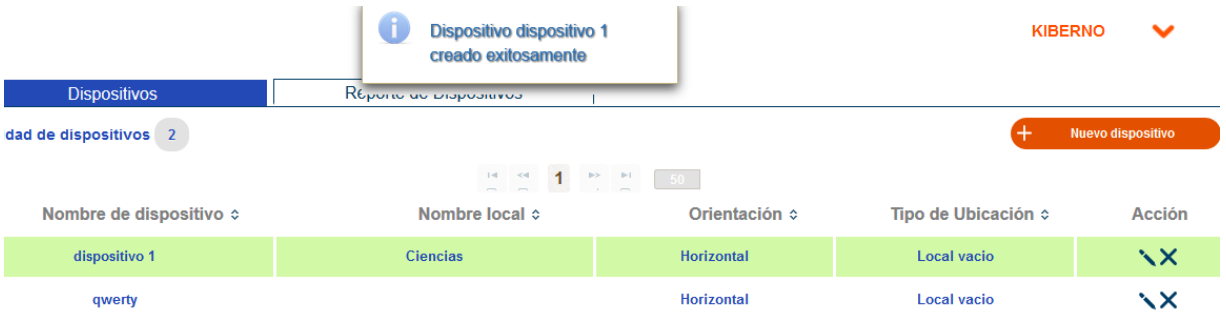


Ilustración 46 – Se muestra el mensaje que el dispositivo fue creado exitosamente.

Fuente: Los Autores

Creando Galería

Ilustración 47 – Se muestra la bandeja de galerías vacía.

Fuente: Los Autores

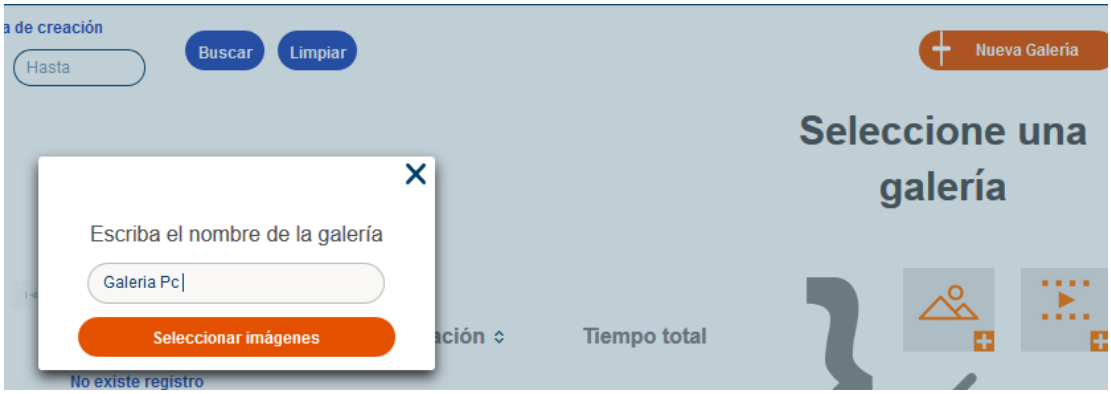


Ilustración 48 – Presionando Nueva Galería y colocando nombre de la galería.

Fuente: Los Autores

Ilustración 49 – Se seleccionan las imágenes/videos para asociar a la galería.

Fuente: Los Autores

Ilustración 50 – Se muestra el mensaje que la galería (Galería PC) fue creada exitosamente.

Fuente: Los Autores

Creando Parrilla de Imágenes/Video

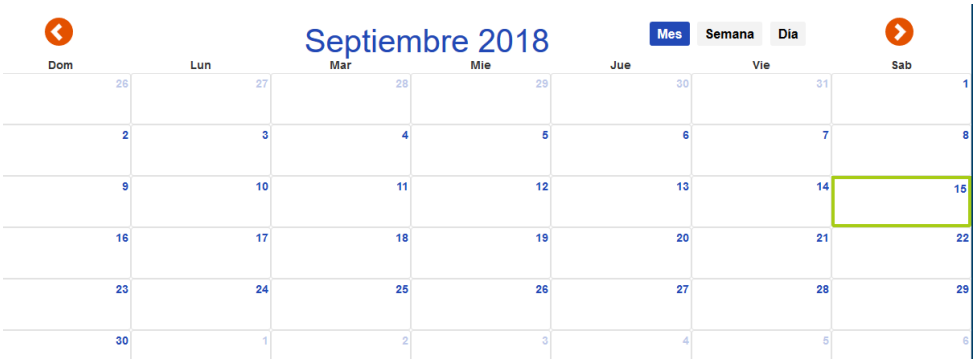


Ilustración 51 – Se muestra el calendario de parrillas vacía.

Fuente: Los Autores

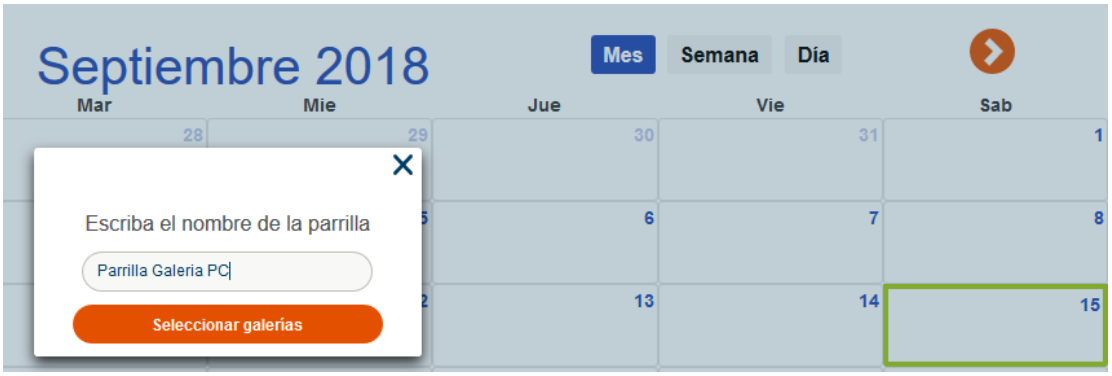


Ilustración 52 – Se le asigna un nombre a la parrilla.

Fuente: Los Autores

Ilustración 53 – Se seleccionan las galerías que se quieren asociar a esta parrilla.

Fuente: Los Autores.

Ilustración 54 – Se muestra el mensaje que la galería fue creada exitosamente con las imágenes asociadas.

Fuente: Los Autores

5.8.3. Pruebas de aceptación

Estas pruebas fueron realizadas en 10 personas que cuentan con el perfil de posibles clientes de la aplicación. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas se realizaron sobre el prototipo funcional. Las preguntas contestadas por los usuarios fueron:

1 - ¿Que tan intuitiva le pareció la aplicación?

2 - ¿Le agrado la interfaz de la aplicación?

3 - ¿Le parece buena para transmitir información?

4 - ¿Le parecieron óptimos los tiempos de respuesta?

5 - ¿Usaría usted esta app para su comercio?

6 - ¿Le pareció interesante la app?

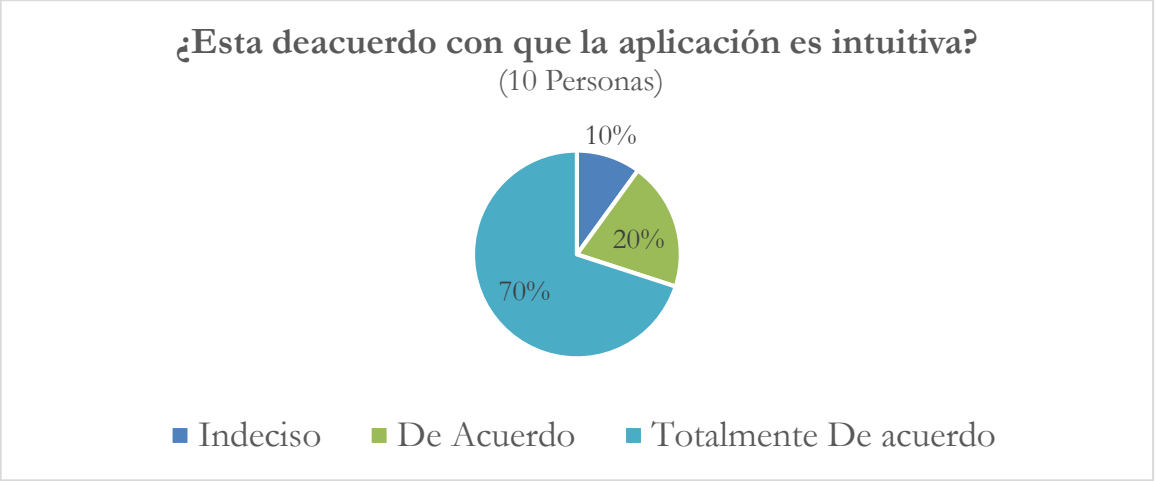
7 - ¿Q tan innovadora le parece la app?

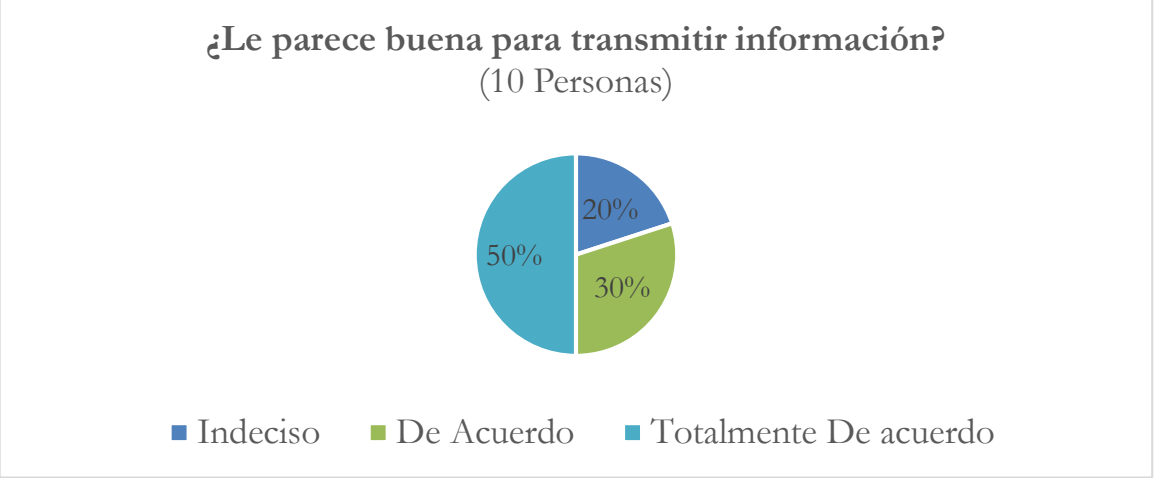
8 - ¿En términos generales que puntaje le da a la aplicación?

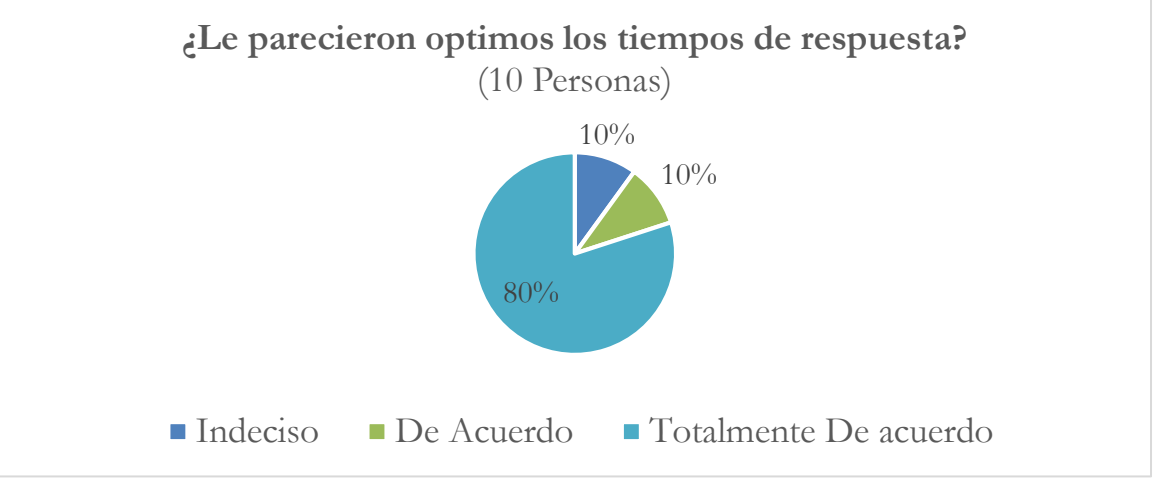
La escala de valoración para las respuestas fue planteada cuando la escala de Likert:

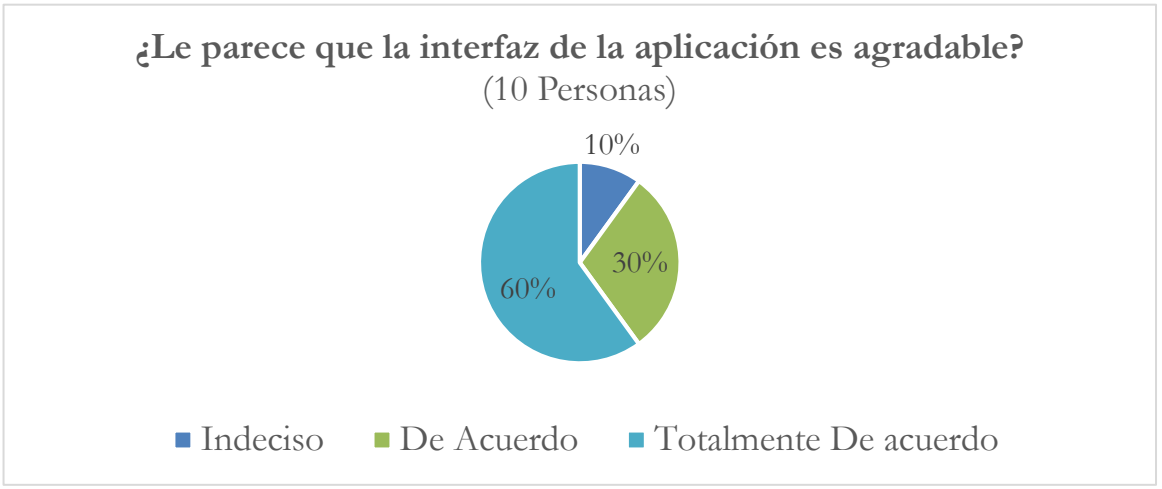
- Totalmente de acuerdo
- De acuerdo
- Indeciso
- En desacuerdo
- Totalmente en desacuerdo

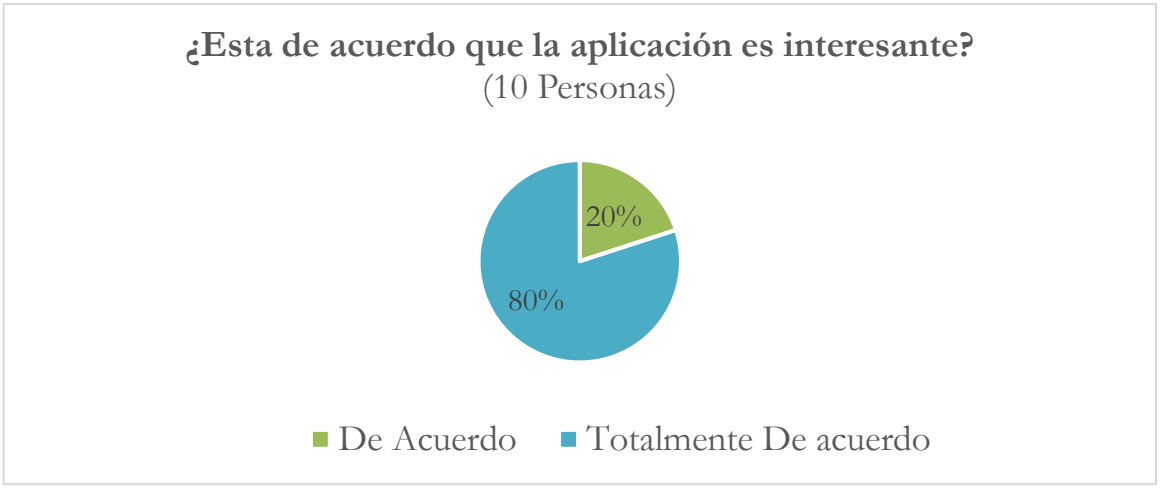
Los resultados obtenidos fueron los siguientes:

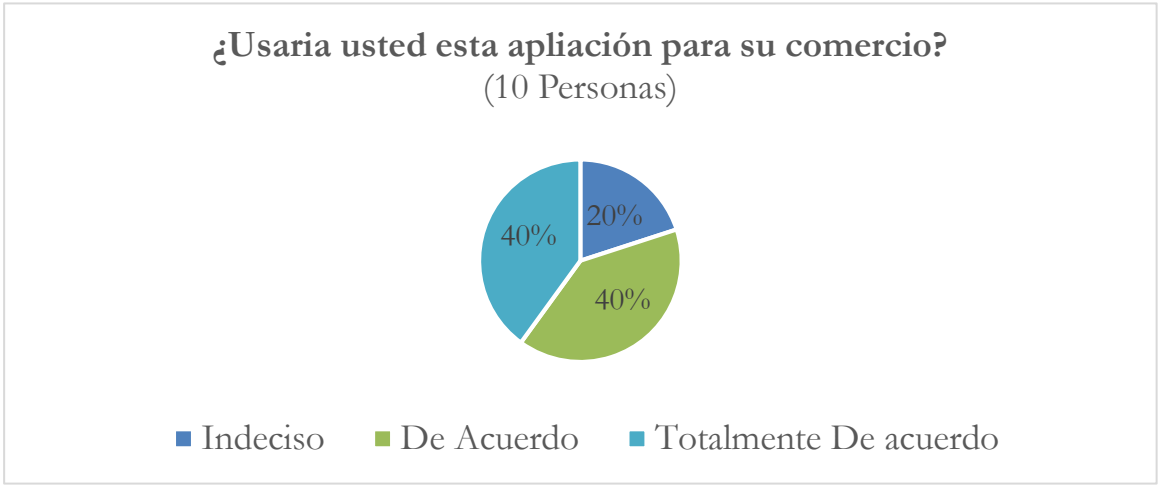


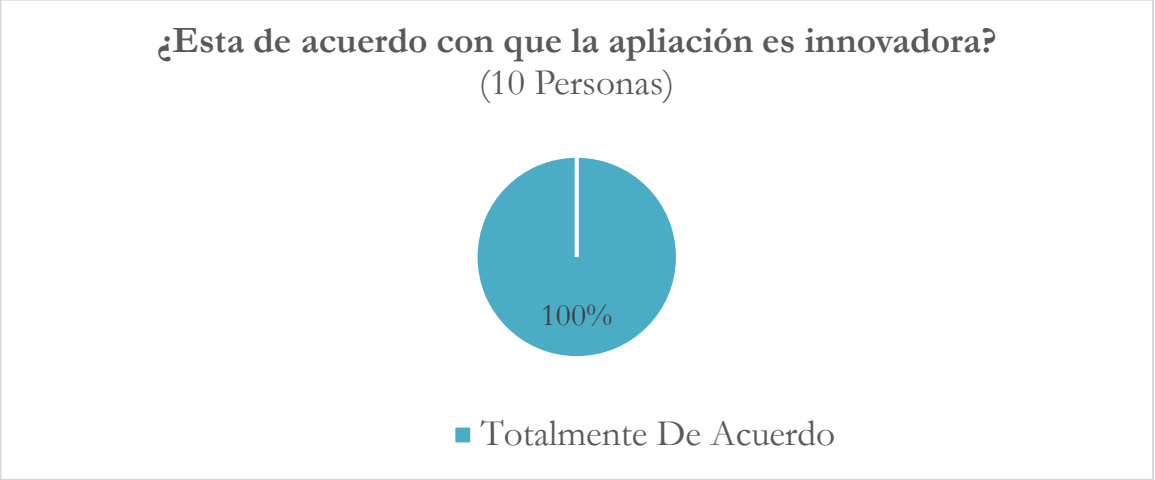


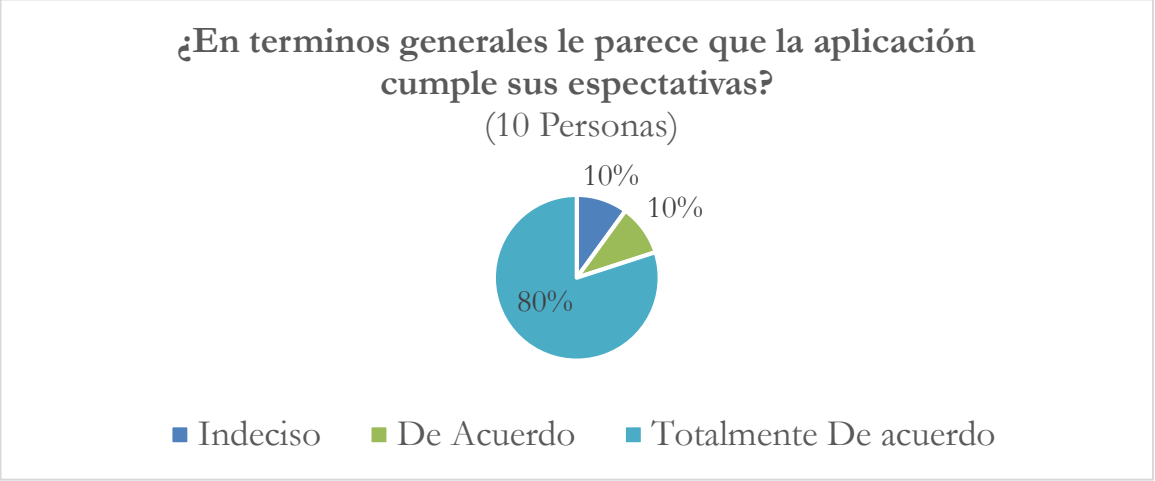



¿Le parece que la interfaz de la aplicación es agradable?
(10 Personas)
10%
30%
60%
Indeciso
De Acuerdo
Totalmente De acuerdo


¿Esta de acuerdo que la aplicación es interesante?
(10 Personas)
20%
80%
De Acuerdo
Totalmente De acuerdo


¿Usaria usted esta apliación para su comercio?
(10 Personas)
20%
40%
40%
Indeciso
De Acuerdo
Totalmente De acuerdo


¿Esta de acuerdo con que la apliación es innovadora?
(10 Personas)
100%
Totalmente De Acuerdo


¿En terminos generales le parece que la aplicación cumple sus espectativas?
(10 Personas)
10%
10%
80%
Indeciso
De Acuerdo
Totalmente De acuerdo

CONCLUSIONES Y RECOMENDACIONES

Una vez completado el objetivo principal del presente Trabajo Especial de Grado, se puede afirmar que se logró exitosamente el desarrollo y construcción de una solución de marketing y gestión publicitaria que permite la visualización de contenido digital en pantallas conectadas a dispositivos Android Tv Box.

El estudio y análisis del mercado publicitario realizado durante el desarrollo e implementación de este proyecto hace entender que el mercadeo digital avanza a un ritmo acelerado el cual es indicado por las necesidades de los consumidores a nivel mundial. Hoy los escenarios de contenidos y distribución audiovisual han estados sometido a una transformación en los últimos años debido a la irrupción de Internet y la llegada de nuevos dispositivos inteligentes como los televisores conectados, las tabletas o los móviles.

Este nuevo contexto audiovisual, todavía en plena evolución y desarrollo, ha generado que la industria publicitaria tenga que evolucionar y adaptar sus procesos, herramientas y metodologías para la toma de decisiones de planificación publicitaria audiovisual.

Teniendo esto en cuenta se entiende la necesidad de que la publicidad deba avanzar tecnológicamente a una medida proporcional a estos puntos mencionados. Nuestro proyecto presenta un emprendimiento para la digitalización de la publicidad haciendo el uso de pantallas conectadas a dispositivos Android TV Box, esta implementación está diseñada para brindar autonomía en el manejo y administración del contenido digital con la intención de presentar una diferencia notable con respecto a las existentes en el mercado con una idea de negocio parecida.

Esta implementación presenta una capa Web que cuenta con dos módulos, el módulo de Front-end desarrollado en jsf usando el framework Primefaces y el módulo de servicios desarrollados en Java usando el Frameworks de Spring. La web está

diseñada para el manejo y administración del contenido digital de una forma sencilla e intuitiva. También se presenta una aplicación móvil desarrollada en Ionic usando el framework angularjs, esta tiene como función transmitir el contenido digital usando la configuración almacenada desde la web. Ambas aplicaciones interactúan con la base de datos no relacional couchDB para el almacenado y consumo del contenido digital, esto mediante un API restful basado en el formato Json que esta misma ofrece.

Para la organización de las asignaciones se implementó la metodología de desarrollo ágil Scrum, se pudo apreciar cómo esta metodología es lo suficientemente flexible como para permitir agregar requerimientos de trabajo al proyecto en general, en distintos momentos del período de desarrollo esta permitió concretar las tareas de forma organizada y con una visión en conjunto de las iteraciones a realizar para completar el desarrollo de las aplicaciones. Las iteraciones fueron planteadas acorde a las tareas finalizadas teniendo en cuenta la prioridad de los requerimientos.

El aplicar el modelo canvas a la solución Cartelera Digital fue una muy buena opción porque permitió definir todas las partes del negocio y así poder aplicar una adecuada implementación. Al terminar de completar el lienzo, se obtuvo una perspectiva general del modelo de negocio que ha permitido reconocer factores tales como, saber en qué se puede diferencia la solución a la competencia y alcanzar el éxito, la flexibilidad, innovación e inmediatez de la gestión e las publicidades, la audiencia adecuada para iniciar el negocio, tener una mejor idea de cómo serán las fuentes de ingreso y la estructura de costes y determinar si el negocio es razonable teniendo en cuenta el segmento de mercado al que se dirige.

Esta aplicación representa una oportunidad para los comerciantes que quieren expandir el alcance de su publicidad, teniendo en cuenta que podrán transmitir o publicar sus productos en cualquier dispositivo que tengan asociado en la plataforma web de la aplicación. Teniendo como idea fundamental que el alcance de inversión se adapta a un rango bastante amplio de consumidores así presentar esta aplicación como accesible para la mayoría de los consumidores.

REFERENCIAS BIBLIOGRÁFICAS Y DÍGITALES

- Acosta, Eleonora (2005) AgilUs: Construcción ágil de la Usabilidad. Recopilado de: http://www.ciens.ucv.ve:8080/genasig/sites/interaccion-humano-comp/archivos/234_CLEI_Acosta_Paper.pdf

- AlFedaghi, Sabah (2011) Developing Web Applications. Recopilado de: http://www.sersc.org/journals/IJSEIA/vol5_no2_2011/6.pdf

- Arnold y Gosling (2001). El Lenguaje de Programación Java TM. Recopilado de: https://docs.google.com/file/d/0Byy7aUl9u4fBRnJwc1U5Vkdnalk/edit

- Beck (1999), *Extreme Programming Explained (Second Explained).*Recopiladode:http://ptgmedia.pearsoncmg.com/images/9780321278654/samplepages/9780321278654.pdf

- Blanco, S. Metodologías de desarrollo (2008). Recuperado de: http://www.marblestation.com/?p=644

- Canós, J., Letelier, P., y Penadés, C. (2003) *Metodologías Ágiles en el Desarrollo de Software.* Recuperado de:http://noqualityinside.com/nqi/nqifiles/XP_Agil.pdf

- Date, C. (2001). Introducción a los Sistemas de Bases de Datos. Séptima edición. ISBN 968-444-419-2. Recopilado de:

https://unefazuliasistemas.files.wordpress.com/2011/04/introducion-a-los-sistemas-de-bases-de-datos-cj-date.pdf

- Díaz (2009). *Rup vs Extreme Programming.* Recopilado de: http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf

- Gutiérrez (2010). *Frameworks* y Componentes, Universidad de los Andes Venezuela. Recopilado de: http://www.codecompiling.net/files/slides/IS_clase_10_frameworks_componentes.pdf

- Martínez, Eduardo (2014). Las 8 grandes ventajas de las metodologías ágiles. Recopilado de:

https://www.iebschool.com/blog/que-es-agile-agile-scrum/

- Márquez A. (2013). Conceptos sobre APIs REST. Recopilado de http://asiermarques.com/2013/conceptos-sobre-apis-rest/

- Oré (2009), en Functional Testing – Pruebas Funcionales. Recopilado de: http://www.calidadysoftware.com/testing/pruebas_funcionales.php

- Osterwalder, Alex (2010). *Business Model Canvas.* Recopilado de: https://www.amazon.com/Business-Model-Generation-Visionaries-Challengers/dp/0470876417

- Porto J., Marino M. (2011), Definición de Dispositivo. Recopilado de: http://definicion.de/dispositivo/

- Quin (2015). *Extensible Markup Language* (XML). Recopilado de: https://www.w3.org/XML/

- Rincón J. (2012). Ciclo de Vida Rup. Recopilado de: http://josemiguelrincon.blogspot.com/2012/04/ciclo-de-vida-rup.html

- Sanz, Roberto (2017). Metodologías Ágiles Vs Metodologías Tradicionales. Recopilado de: http://www.uv-mdap.com/programa-desarrollado/bloque-iv-metodologias-agiles/metodologias-agiles-vs-tradicionales/

- Silberschatz, Korth y Sudarshan. (2002). Fundamentos de bases de datos. Recopilado de: https://unefazuliasistemas.files.wordpress.com/2011/04/fundamentos-de-bases-de-datos-silberschatz-korth-sudarshan.pdf

- Sommerville I. (2002). Ingeniería de Software, Pearson Educación, Recuperado de:http://zeus.inf.ucv.cl/~bcrawford/AULA_ICI441/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

- Techopedia (2018) Mobile Application (Mobile App). Recopilado de: https://www.techopedia.com/definition/2953/mobile-application-mobile-app

- Vasilis, (2011). Metodologías de desarrollo de software. Recuperado de: http://ingenieriasoftwarecufm.blogspot.com/2012/10/metodologias-para-desarrollo-de-software.html

- Vila, (2015). Eventos Scrum: El Scrum diario. Recopilado de: http://managementplaza.es/blog/el-scrum-diario/

- Vladimir Zwass (2018). La Enciclopedia Británica. Recopilado de: https://global.britannica.com/topic/information-system

Anexos

1. Manual de aplicación Cartelera Digital Web App

Cartelera Digital Web App

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Manual de usuario

Versión 1.0

Autores:

Br. Enrique Marín

Br. Richard Franco

Caracas, septiembre de 2018

Cartelera Digital Web App

Table of Contents

1. Descripción de la aplicación 3
2. Inicio de sesión 3
3. Página principal (Home) 4
4. Imágenes y Videos 5
5. Galerías 7
6. Dispositivos 8
7. Gestión de parrillas 9
8. Salir 11

Cartelera Digital Web App

1. Descripción de la aplicación

Cartelera Digital Web App está pensado para aquellos comercios que están siempre pendiente de hacer llegar la información de sus productos y servicios al público cercano a sus locales. Aquellos comerciantes que disfrutan manejar su contenido, les gusta tener sus publicidades actualizadas y que desean que su audiencia esté conectada en todo momento.

En esta aplicación se podrá contar con un perfil de Administrador y gestionar todo el contenido multimedia (imágenes, GIF, videos) que deseen. Podrán crear dispositivos lógicos que representan cada televisor o pantalla física y asociar el contenido previamente subido a cada dispositivo creado y así tener control total de lo que muestra cada una de las pantallas que se posean. Una vez gestionado el contenido y asignado a los dispositivos, será sincronizado a la aplicación móvil inmediatamente y poder estar a la vanguardia tecnológica y comercialmente.

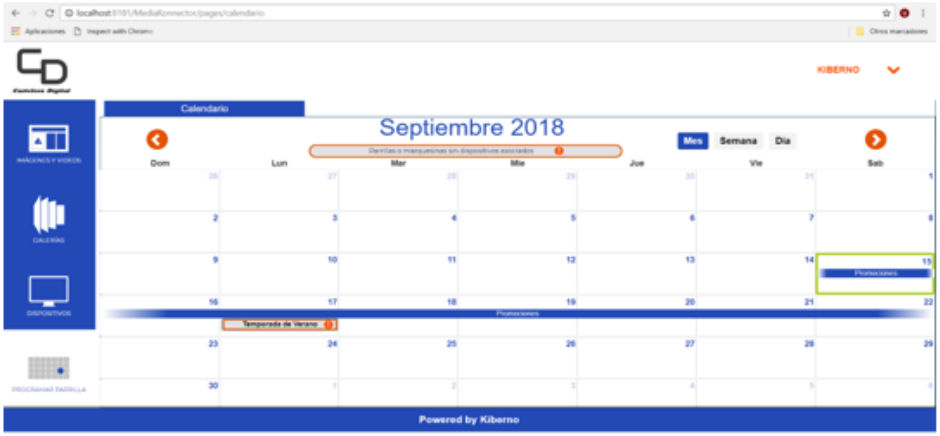
2. Inicio de sesión

A través del inicio de sesión, se podrá ingresar al portal administrativo y ver la página principal de Cartelera Digital Web App.

Cartelera Digital Web App

3. Página principal (Home)

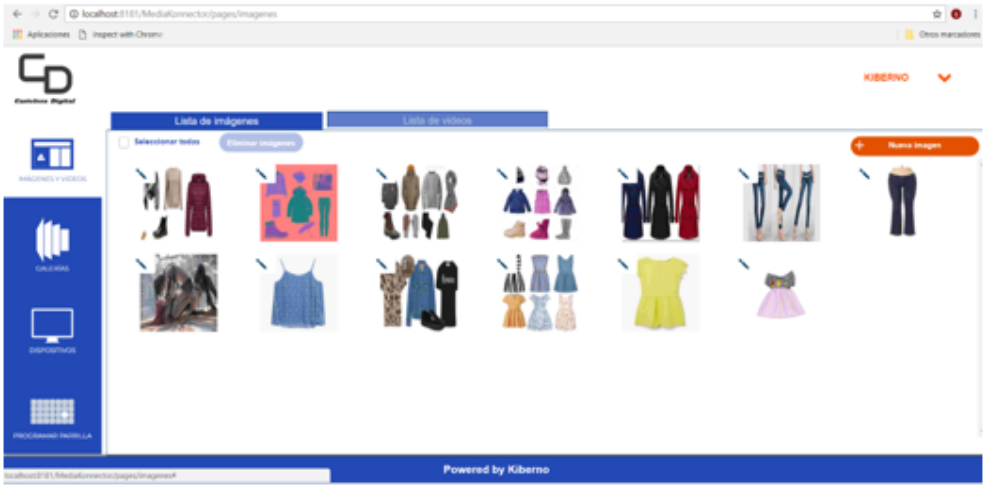
La aplicación cuenta con un Calendario que se despliega al iniciar sesión, el cual muestra las parrillas programadas en el mes en curso indicando aquellas gráficamente si a las parrillas se les ha asignado dispositivos, alerta importante para notificar que posiblemente algún contenido no se esté reproduciendo. A su vez del lado izquierdo está visible en todo momento el menú lateral para poder acceder a cualquiera de los módulos de la aplicación e igualmente del lado superior la posibilidad de cerrar sesión en cualquier momento.



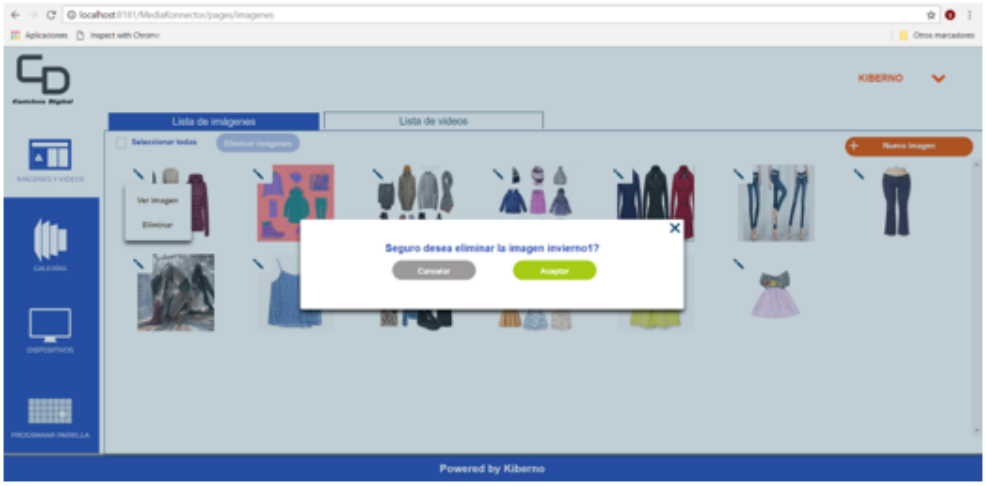
4. Imágenes y Videos

En esta sección de la aplicación, el usuario puede gestionar el contenido multimedia que desea utilizar, es decir, subir imágenes y/o videos desde su computadora o dispositivo al servidor de la aplicación. Una vez subido el contenido multimedia al servidor, se muestra una vista previa con una resolución de baja calidad para disminuir los tiempos de carga. Se presenta la posibilidad de ver una imagen y/o video donde se visualiza con su calidad original.

Cartelera Digital Web App



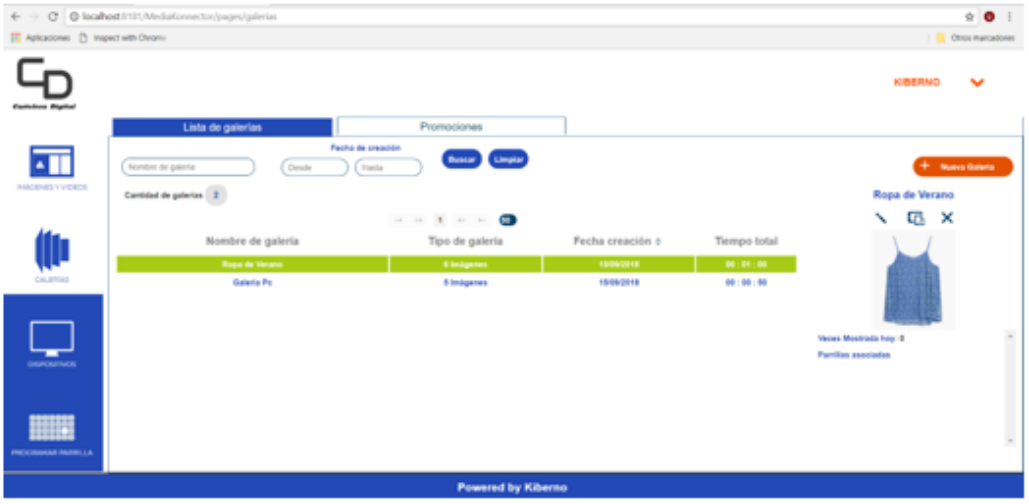
También cuenta con la opción de eliminar una o varias imágenes y/o videos de la aplicación. Si el contenido a eliminar representa el 100% del contenido asociado a una galería, se preguntará confirmación y de ser afirmativo, se eliminará también la galería asociada.



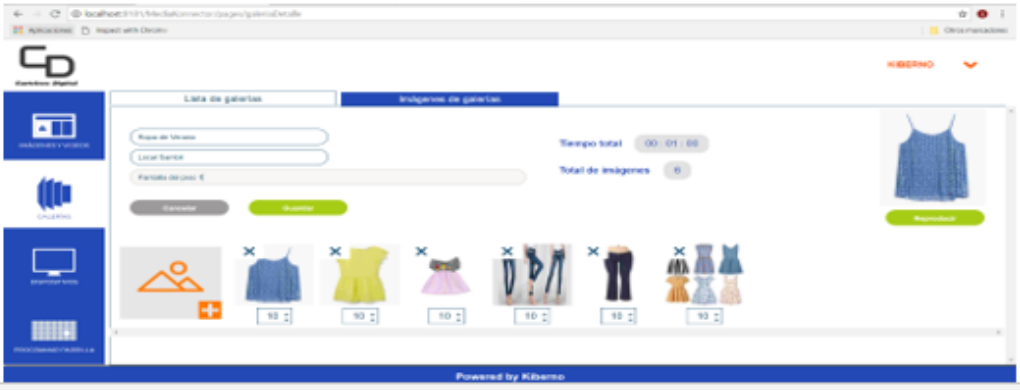
Cartelera Digital Web App

5. Galerías

En esta sección, el usuario visualiza el listado de galerías creadas. Filtros de búsqueda sobre el listado de galerías y la posibilidad de crear una nueva galería de imagen o video. También se visualiza un *preview* de la galería seleccionada y una *metadata* de la misma, como, por ejemplo, cantidad de veces reproducida y si la galería fue asociada a una parrilla digital.



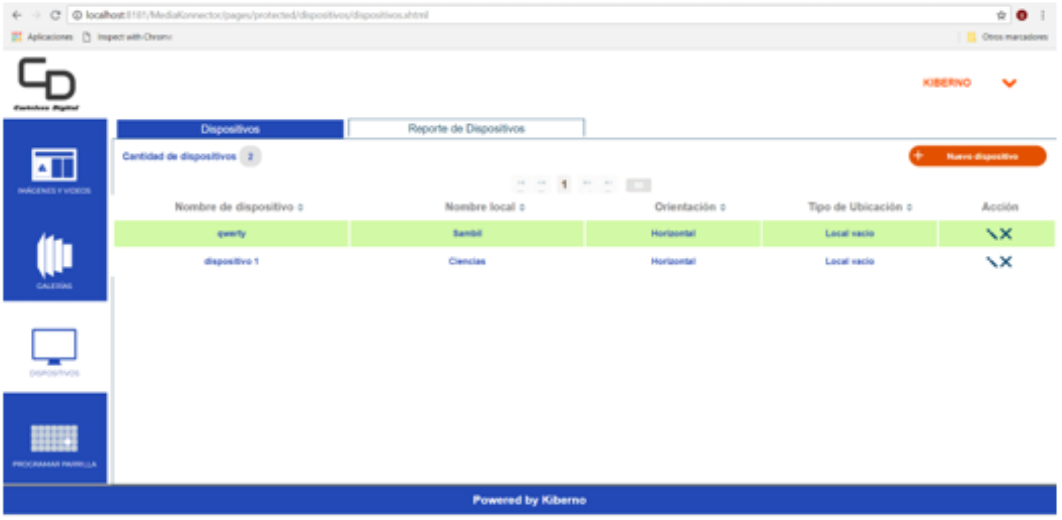
Una vez seleccionada el tipo de galería que se desea crear, llegamos al detalle de la galería donde llenamos un pequeño formulario sobre la galería y agregamos las imágenes o videos que deseamos asociar a dicha galería. Se puede elegir modificar el orden de reproducción y aumentar o disminuir la duración en que cada *item* se mostrará en las pantallas. Se puede visualizar un *preview* de la galería para observar cómo se va a reproducir en las pantallas la galería.



Cartelera Digital Web App

6. Dispositivos

En esta sección el usuario visualiza la lista de dispositivos creados con una información breve, como, por ejemplo, nombre de dispositivo, nombre del local donde estará instalado, orientación del televisor, etc. La posibilidad de crear, editar o eliminar algún dispositivo-

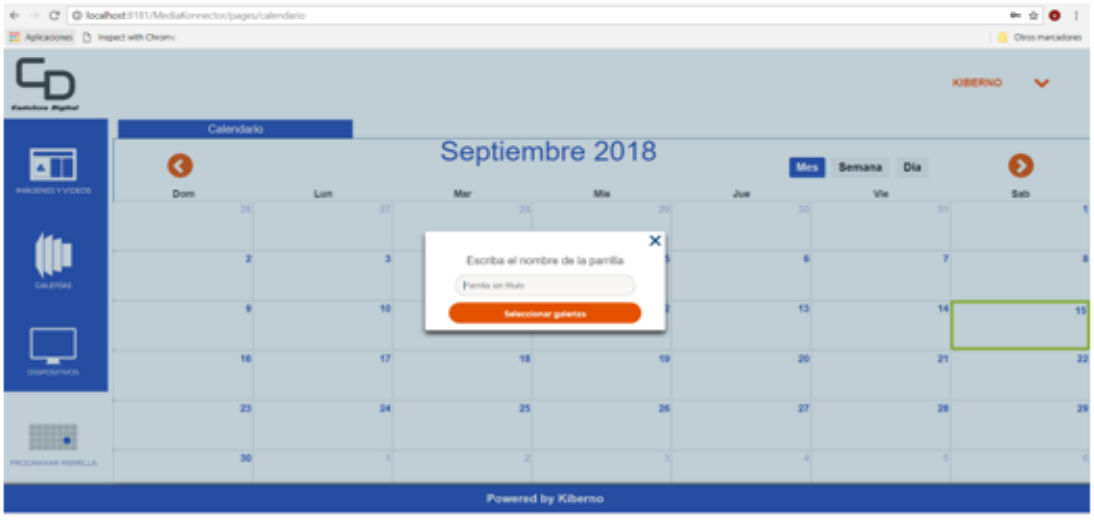


Una vez seleccionada la opción de crear nuevo dispositivo, llegamos al detalle del dispositivo donde llenamos un pequeño formulario sobre los datos del dispositivo que se desea crear para tener un buen control del contenido que se reproduce en cada uno de ellos. Si poseemos varias oficinas o locales podemos asignarle coordenadas al dispositivo para tenerlo geolocalizados y poder llevar de alguna manera la ubicación exacta y la posible audiencia que abarca cada uno de los dispositivos.

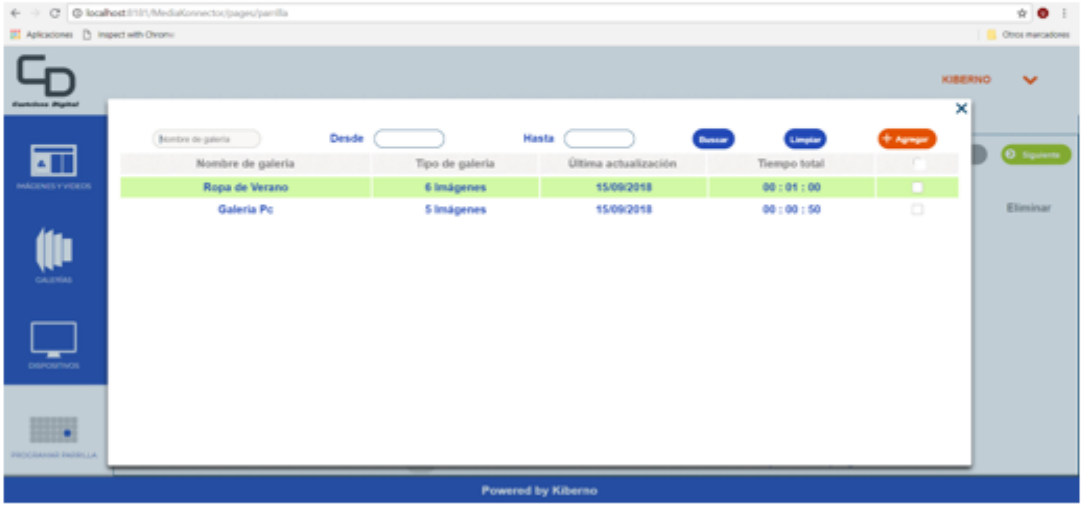
Cartelera Digital Web App

7. Gestión de parrillas

En el home de la ampliación, tenemos el calendario con todas las parrillas digitales que han sido programadas y la posibilidad de crear o editar una de ellas. S toma como consideración que no se pueden crear parrillas digitales en días anteriores al día en curso y el inicio de la parrilla nueva será inicialmente el día seleccionado.

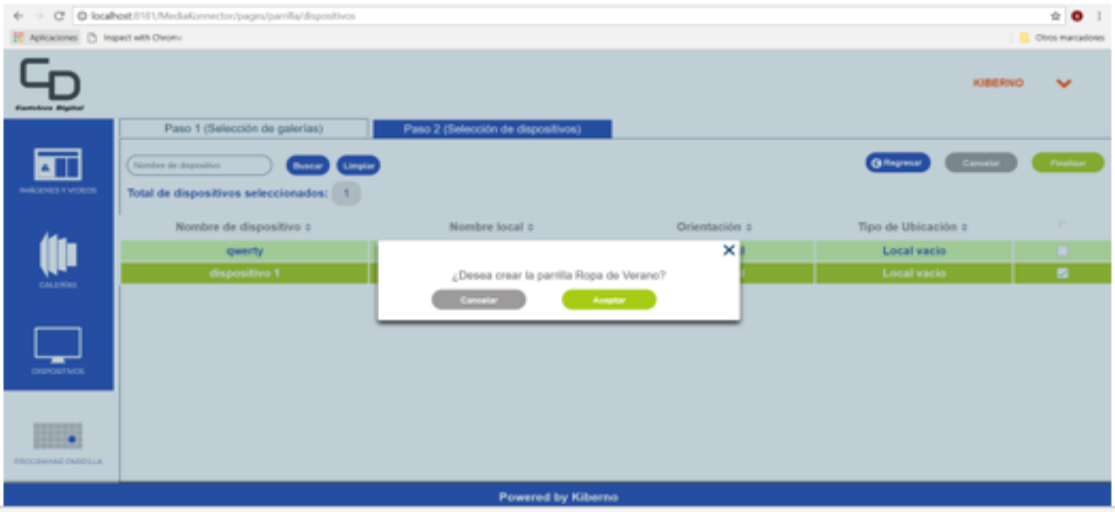


Una vez seleccionada la opción de crear nueva parrilla, se indica el nombre con el que se llamará y se procede al detalle de la parrilla donde se en un primer paso, se elige qué galerías desean agregar a dicha parrilla.



realizadas sobre la parrilla detendrán la reproducción en curso y actualizarla inmediatamente o si se desea que las actualizaciones se ejecuten en *background* haciendo que la reproducción en curso se mantenga mientras es descargado el contenido nuevo y una vez terminada la descarga se procede a la actualización del reproductor multimedia de la aplicación móvil.

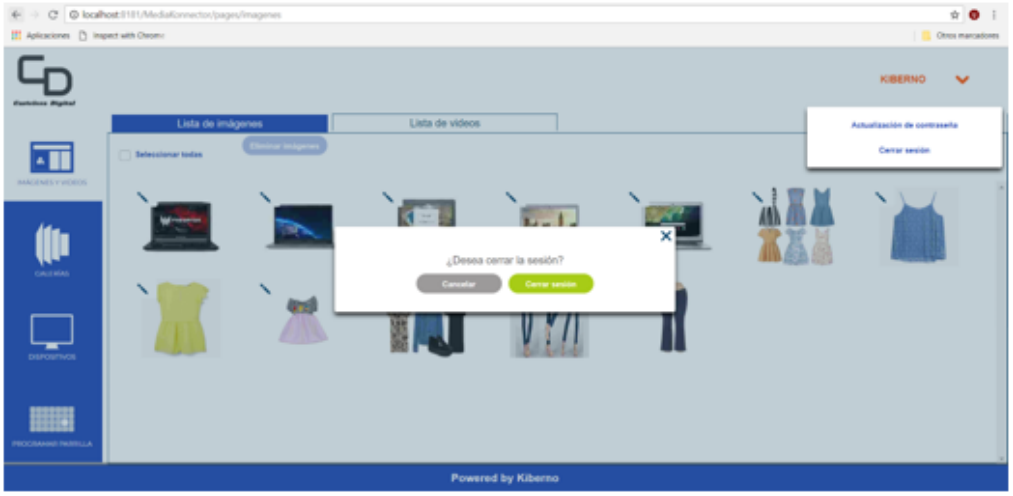
Una vez echa la configuración se procede al paso 2 donde se elige qué dispositivos deseamos asociarle a la parrilla. También esta la opción de crear la parrilla sin asociarle ningún dispositivo para tenerla guardad y en cualquier momento se puede editar y asociarle los dispositivos.



Cartelera Digital Web App

8. Salir

En todo momento que estemos utilizando la aplicación se tiene la opción de cerrar la sesión y salir. Si se detecta un período de inactividad de 20 minutos, la aplicación automáticamente cierra la sesión y redirecciona a la pantalla de *Login*.



2. Manual de aplicación Cartelera Digital Móvil App

Cartelera Digital Movil App

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Manual de usuario

Versión 1.0

Autores:

Br. Enrique Marín

Br. Richard Franco

Caracas, septiembre de 2018

Cartelera Digital Movil App

Tabla de Contenidos

1. Descripción de la aplicación....3
2. Inicio de sesión 3
3. Selección de dispositivos (Home) 4
4. Descarga y sincronización de contenido....5
5. Reproducción de contenido....6

Cartelera Digital Movil App

1. Descripción de la aplicación

Cartelera Digital Móvil App está pensado para aquellos comercios que están siempre pendiente de hacer llegar la información de sus productos y servicios al público cercano a sus locales. Aquellos comerciantes que disfrutan manejar su contenido, les gusta tener sus publicidades actualizadas y que desean que su audiencia esté conectada en todo momento.

Como se ha mencionado, Cartelera Digital móvil app es muy sencilla de utilizar, la idea es que en menos de 5 minutos ya se esté reproduciendo el contenido digital deseado, por lo tanto, se llenarán los datos mínimos pata descargar la configuración y el usuario podrá disfruta de sus publicidades en tiempo mínimo. En esta aplicación se podrá contar con un perfil de usuario donde se podrá decidir el contenido digital que se desea visualizar. La app ofrece la posibilidad de decidir el dispositivo que queremos sincronizar, es decir, todo el contenido digital gestionado desde la app web será sincronizado al dispositivo que se elija aquí e inmediatamente comienza la descarga (de no haber descargado el contenido anteriormente) y una vez finalizado empieza la reproducción del contenido deseado. También ofrece compatibilidad con televisores que estén desplegados tanto horizontal como vertical.

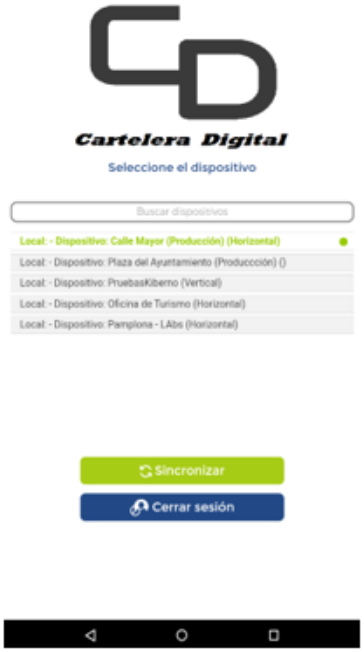
2. Inicio de sesión

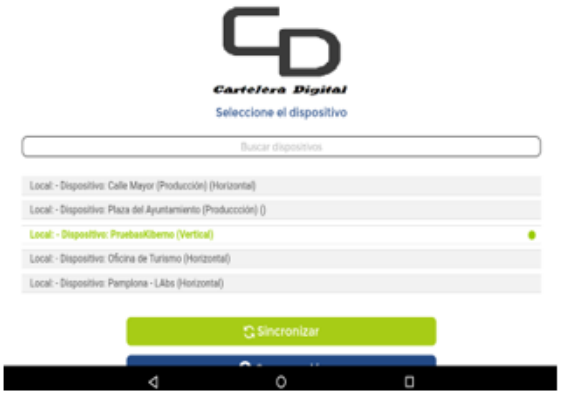
A través del inicio de sesión, se podrá ingresar al home de la aplicación.

Cartelera Digital Movil App

3. Selección de dispositivos (Home)

Una vez iniciado sesión en la aplicación se despliega la pantalla de selección de dispositivo. Aquí se podrá seleccionar el dispositivo al cual se le ha asignado el contenido a mostrar desde la web app. De esta manera si poseemos distintos locales se puede tener un dispositivo conectado a una pantalla en cada local y asignarle contenidos diferentes, por ejemplo, a la tienda de un centro comercial se le envía contenido de promoción y a la tienda de un boulevard se le envía contenido de una temporada en especial. Con este tenemos control independiente de la publicidad a mostrar en cada una de las pantallas.





Cartelera Digital Movil App

4. Descarga y sincronización de contenido

Una vez seleccionada el dispositivo, se inicia la descarga y sincronización del contenido asignado al dispositivo desde la web app. Mientras se ejecuta la descarga se muestra una pantalla de espera con una barra de progreso de la descarga. Es muy importante hacer notar que la descarga del contenido solo se ejecuta una vez, es decir, sin importar que tantas galerías o parrillas digitales se creen en la web app, si las imágenes y videos ya fueron descargados previamente, estos no volverán a descargarse y la reproducción se inicia inmediatamente.

Cartelera Digital Movil App

5. Reproducción de contenido

Por último, una vez hechas las configuraciones necesarias, se procede a la reproducción del contenido digital, exactamente como fue programado desde la web app.