ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Máster

Diseño e implementación de una aplicación móvil multiplataforma gestionable por web para la notificación de eventos locales

AUTOR: Nuria Martínez Ortuño

DIRECTOR: José Fernando Cerdán Cartagena

CODIRECTOR: Andrés Cabrera Lozoya

Julio / 2016

Agradecimientos

En primer lugar, me gustaría dar las gracias a mi madre, por ser una súper madre y apoyarme siempre en cada decisión que he tomado, por ayudarme a mantener la calma y a levantarme en situaciones difíciles, pero sobre todo por no soltarme nunca de la mano.

A mi abuela Tana, por estar siempre preocupándose de mí y por seguirme en cada paso que doy.

A Emilio, por ser mi mayor apoyo y estar siempre a mi lado, por aconsejarme y ayudarme a conseguir mis metas, y por quererme como me quiere.

A mis sobrinos, por ser mi vía de escape y mi mayor alegría.

A mis amigas, Irene, Inma, Rocio, Cristina, Marina, Maria, Raquel, Maria y Fuen, por ayudarme a superar las adversidades, por confiar en mí y hacer que me ría.

A mi director Fernando Cerdán, una vez más, por haberme dado otra gran oportunidad como ha sido la de participar en la cátedra de Hidrogea.

Y por supuesto, a mi codirector Andrés Cabrera, por estar siempre en todo, por hacerme siempre el trabajo más sencillo, por animarme y sobre todo por hablar bien de mí.

Índice

Índice de ilustraciones

Índice de tablas

1. Introducción

A lo largo de la historia, todas las ciudades sufren un proceso continuo de evolución fundamentado en satisfacer las demandas de los habitantes marcado por la necesidad de incorporación y adaptación a los cambios que marca la evolución tecnológica.

En la actualidad aparece un nuevo escenario para las ciudades y sus habitantes, este nuevo modelo de ciudad se denomina “Smart City” o Ciudad Inteligente.

Durante los últimos años, el concepto de Smart City ha ido evolucionando en función del número de actividades que se han visto involucradas en el entorno urbano. Este hecho está enormemente relacionado con el gran avance de las tecnologías, sobre todo en el caso de aquellas disponibles para el ciudadano (smartphones, tablets, etc.). Estas nuevas herramientas, junto con el avance propio de cada uno de los sectores colaboradores en la actividad de la ciudad han elevado las posibilidades de las urbes modernas a límites inimaginables.

Dentro de este contexto, se define Smart City como aquella ciudad que emplea las tecnologías de la información y las comunicaciones (conocidas como TIC) para lograr una interacción máxima entre los ciudadanos y todos los entes que componen la vida urbana. Por lo tanto, las ciudades inteligentes deben de convertirse en una plataforma digital donde se optimicen todos los aspectos de la ciudad: mejora del entorno y el bienestar de los habitantes, fomento del desarrollo sostenible, posibilidad de nuevos modelos de negocio, mejoras económicas, etc.

Una ciudad inteligente debe ser aquella que tenga conocimiento de su estado y favorezca a la población a saber más de ella. Por ejemplo, en una Smart City ya no nos preguntaremos si una calle está cortada, la propia calle informará de su estado. No nos preguntaremos si existen fugas de agua en nuestra vivienda, la red de abastecimiento inteligente nos reportará los problemas.

En definitiva, una ciudad inteligente debe de llegar a ser un ecosistema en el que intervengan diferentes agentes (ciudadanos, empresas y organismos públicos), capaz de escuchar, recoger y transmitir la información de la actividad de la ciudad para mejorar la toma de decisiones y la prestación de servicios.

1.1.Funcionamiento de las Smart Cities

Para lograr estos objetivos, una Smart City debe de ser capaz de llevar a cabo tres acciones principales: recopilación de datos, comunicación y análisis de la información.

- **Recopilación de datos**. Dentro de este apartado toma importancia la red de sensores de la ciudad. La presencia de dispositivos y sensores de medición es una parte fundamental en el funcionamiento de las Smart Cities. Contadores inteligentes de electricidad, agua y gas, sensores de tráfico y aparcamiento, estaciones meteorológicas e incluso los propios smartphones de los ciudadanos son algunos de los engranajes que forman el mecanismo de captación de la información.
- **Comunicación.** Una vez recopilados los datos, las ciudades inteligentes deben de tener la capacidad de proporcionar conectividad para el intercambio de información en todos los lugares y para todas las personas y dispositivos.
- **Análisis de la información**. Tras recopilar los datos se debe de proceder a su análisis para extraer su importancia y utilidad. Estos objetivos se pueden dividir en:
    - Presentación de la información.
    - Optimización de operaciones.
    - Predicción

1.2.Descripción del problema

El proyecto surge debido al auge existente actualmente en el sector Smart Cities. Desde la cátedra Hidrogea-UPCT se propuso en un primer momento realizar un estudio del arte de la situación en la que se encontraban actualmente las Smart cities, tanto en España como fuera de España. Tras este estudio, se llegó a la conclusión de que muchas de las ciudades que ya tenían el concepto de Smart city aplicado, no tenían tanta tecnología como la que tiene la ciudad de Cartagena.

El problema que se encontró fue que actualmente se tenía una gran cantidad de información sobre la ciudad, como cortes de calles, datos de los pluviómetros, obras que se estaban realizando, calidad del agua, etc., pero que no se mostraban al usuario, y que por lo tanto no podían ayudar al ciudadano a tener una vida más cómoda.

Con este problema nació la idea de crear una ***plataforma de notificaciones***, en la que el usuario de la aplicación pudiera estar al tanto de todos los servicios, obras y demás que existían en su ciudad.

1.3.Estado del arte

1.3.1. Smart Cities

Este resumen consta de los cinco aspectos generales que se consideran en las Smart Cities analizadas con sus correspondientes líneas generales de actuación.

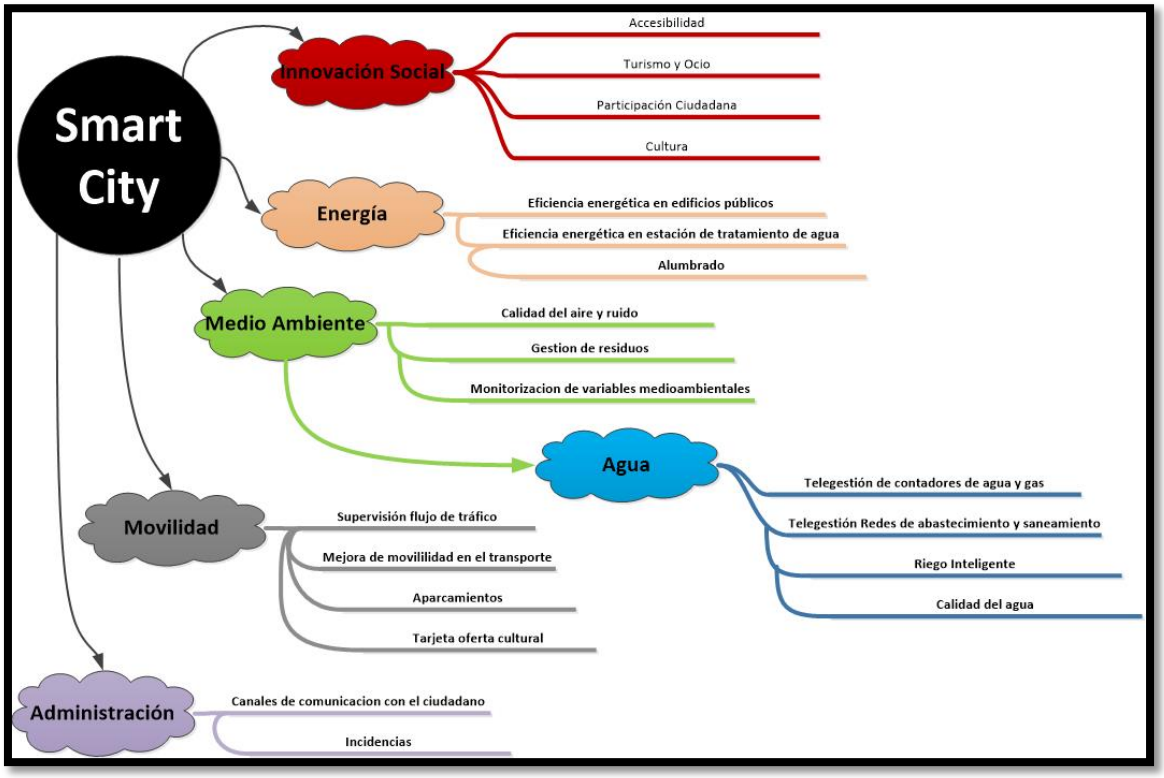


Ilustración 1: Conceptos y Proyectos Generales de Algunas Ciudades Españolas

Dentro del concepto de Smart City, se pueden destacar 5 grupos de actuación principales:

- **Innovación Social:** que incluye accesibilidad, cultura y deporte, participación ciudadana y e-Participación, salud y tele asistencia, seguridad y gestión de Servicios Públicos de emergencias, turismo y ocio, educación y gobierno abierto y open data.
- **Energía**: Información, formación y difusión a los ciudadanos en el ámbito de la eficiencia energética e instalaciones municipales (edificios Smart Space, eficiencia en el alumbrado público, instalaciones de energías renovables, etc.)
- **Medio Ambiente, Infraestructuras y Habitabilidad Urbana:** Calidad ambiental, edificación sostenible, gestión de edificios públicos y domótica, gestión de infraestructuras públicas y equipamiento urbano, gestión de parques y jardines públicos, habitabilidad, medición de parámetros ambientales, recogida y tratamiento de residuos y urbanismo.

- **Movilidad urbana:** Movilidad eléctrica y sistemas inteligentes de transportes.
- **Gobierno, Economía y Negocios:** Administración electrónica en digitalización, modernización, integración e interoperabilidad, nuevos modelos de negocio, empleo, e-Comercio, plataformas de pago NFC (pagos móviles), entornos iCloud, etc.

En la siguiente tabla, se pueden visualizar los ámbitos destacados en el concepto Smart Cities para cada ciudad analizada de España.

	Innovación Social		Agua
	Energía		Movilidad
	Medio Ambiente		Administración

Tabla 1. Leyenda

A CORUÑA						
ALCORCÓN						
ALICANTE						
ALMERÍA						
BARCELONA						
CARTAGENA						
ELCHE						
GRANADA						
GUADALAJARA						
JUN						
MADRID						
MÁLAGA						
MOLINA DE SEGURA						

MURCIA						
PAMPLONA						
RIVAS						
SABADELL						
SANTANDER						
SANTIAGO DE COMPOSTELA						
SEVILLA						
SOTO DEL REAL						
VALENCIA						
VALLADOLID						
VITORIA						

Tabla 2. Análisis de una muestra de Ciudades Españolas

LJUBLJANA (ESLOVENIA)						
VICTORIA (CANADÁ)						
COPENHAGUEN WHEEL (DINAMARCA)						
ÁMSTERDAM (PAÍSES BAJOS)						
LONDRES						
HELSINKI (FINLANDIA)						
ESTOCOLMO (SUECIA)						
BERLIN (ALEMANIA)						
VIENA (AUSTRIA)						
PARIS (FRANCIA)						
NUEVA YORK (ESTADOS UNIDOS)						

SAN FRANCISCO (ESTADOS UNIDOS)						
BOSTON (ESTADOS UNIDOS)						
SEATTLE (ESTADOS UNIDOS)						
SAN JOSÉ (ESTADOS UNIDOS)						
ITALIA						

Tabla 3. Análisis de una muestra de ciudades extranjeras

Los municipios de entre 50.000 a 500.000 cuentan, en términos generales, con el suficiente apoyo de fuerzas habilitadoras para desarrollar apropiadamente una Smart City en todas sus dimensiones. No obstante, aparentemente están menos preparados que las grandes metrópolis en términos de recursos, potencial tecnológico y capacidad de organización, por lo que se enfrentan a retos diferentes y en ocasiones poco explorados.

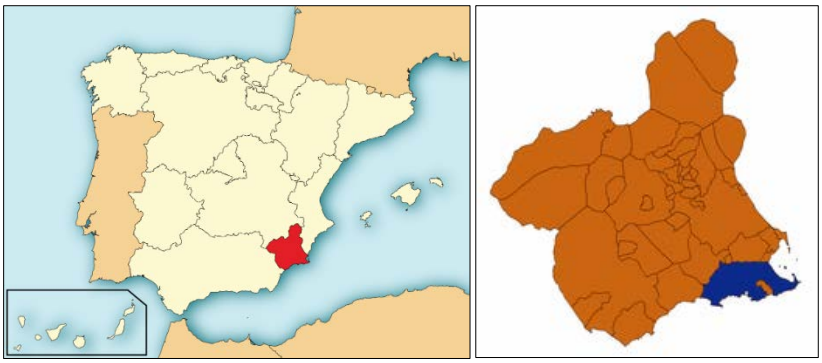


Ilustración 2: Situación geográfica de Cartagena

Cartagena es una ciudad y municipio español localizado en la parte sureste de la comunidad autónoma de la Región de Murcia. En la actualidad cuenta con una población total de 218.070 según datos del ayuntamiento, aunque es aconsejable remarcar los incrementos que estas cifras pueden sufrir dada la fuerza de su turismo y la presencia permanente de los residentes universitarios. Estos datos colocan al municipio de Cartagena en el número 23 dentro de los municipios españoles más poblados. Como capital legislativa de la comunidad autónoma, Cartagena es la sede de la Asamblea Regional.

En cuanto a su extensión, el término municipal de Cartagena abarca una superficie total de 558 km^2, con una densidad media de población de 387 habitantes por km^2 según datos del centro regional de estadística de Murcia.

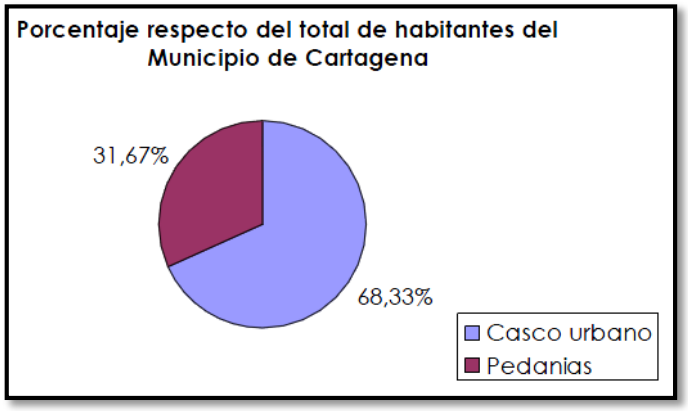


Ilustración 3: Porcentaje de reparto de habitantes (Fuente: Informe mercado inmobiliario Cartagena 2012)

Como se puede observar en el anterior gráfico, cerca del 68% de la población del municipio se sitúa en el caso urbano mientras que el 32% restante se reparte en las pedanías del resto de diputaciones, hecho que fortalece al núcleo urbano como la principal área de desarrollo.

Dejando a un lado la agricultura del Campo de Cartagena y los sectores industriales y de construcción naval, en los últimos años el sector que está alcanzando el desarrollo más remarcable dentro de la economía es el terciario, fundamentalmente asociado al turismo dadas las inmejorables oportunidades que tanto su localización como su pasado ofrecen.

Dado el nuevo panorama en lo que a tecnologías de la información y comunicación se refiere, Cartagena tiene la obligación de explotar todos sus recursos y oportunidades para convertirse en una ciudad *Smart*, con todos los esfuerzos y ventajas que esto supone. El objetivo debe ser impulsar a Cartagena como uno de los principales centros económicos y tecnológicos dentro de España, logrando que la ciudad utilice la tecnología para ser más segura, saludable y eficiente ofreciendo a su vez infraestructuras y servicios generadores de competitividad y riqueza.

Este camino requerirá la dotación de las herramientas necesarias para involucrar al ciudadano en el día a día de la ciudad. El horizonte es convertir a Cartagena en una ciudad competitiva, conectada y sostenible, es decir, una ciudad inteligente.

Una vez recopilada toda esta información, se pasó a buscar toda la información que actualmente hay disponible para el usuario que en la actualidad no se muestra de forma unificada. Durante esta fase se encontró la siguiente información:

- Información disponible de Cortes de tráfico y ocupación de vía pública.

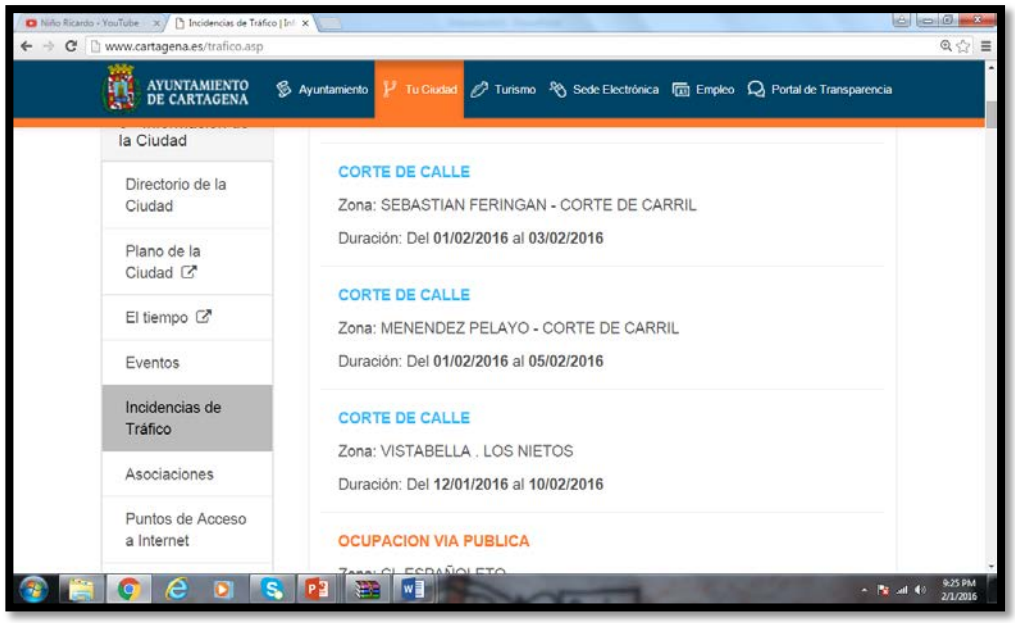


Ilustración 4: Información Cortes de tráfico

- Información disponible de eventos y actuaciones en la ciudad

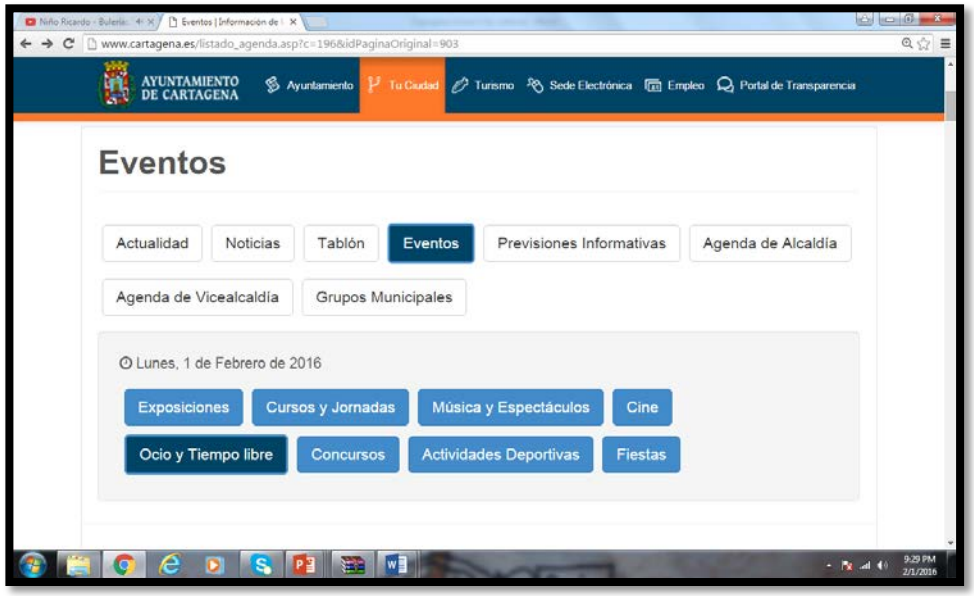


Ilustración 5: Información disponible de eventos y actuaciones en la ciudad

- Información disponible de rutas turísticas.

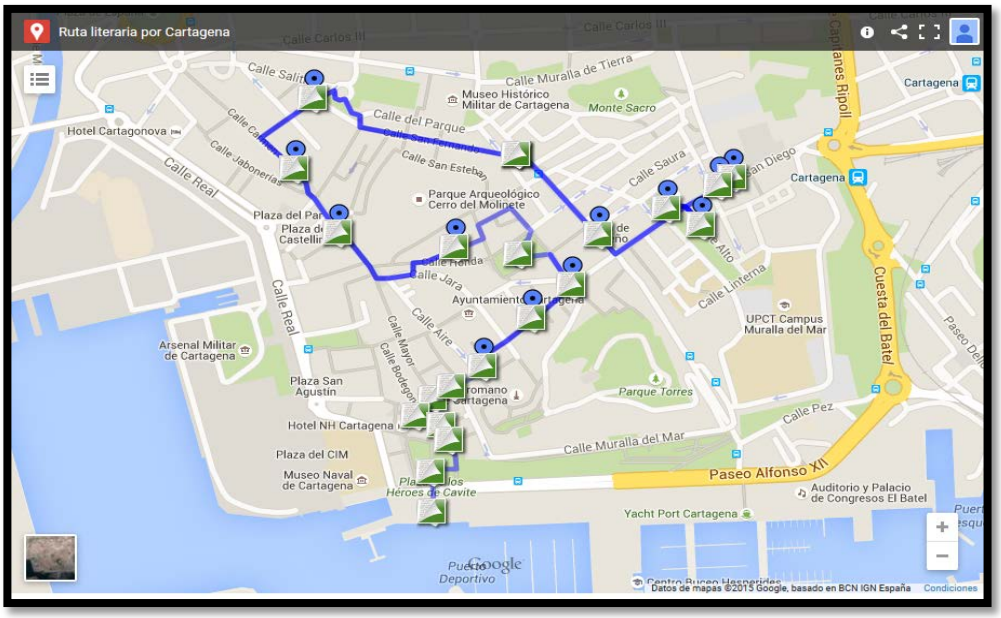


Ilustración 6: Información disponible de rutas turísticas

- Información disponible del estado de las playas

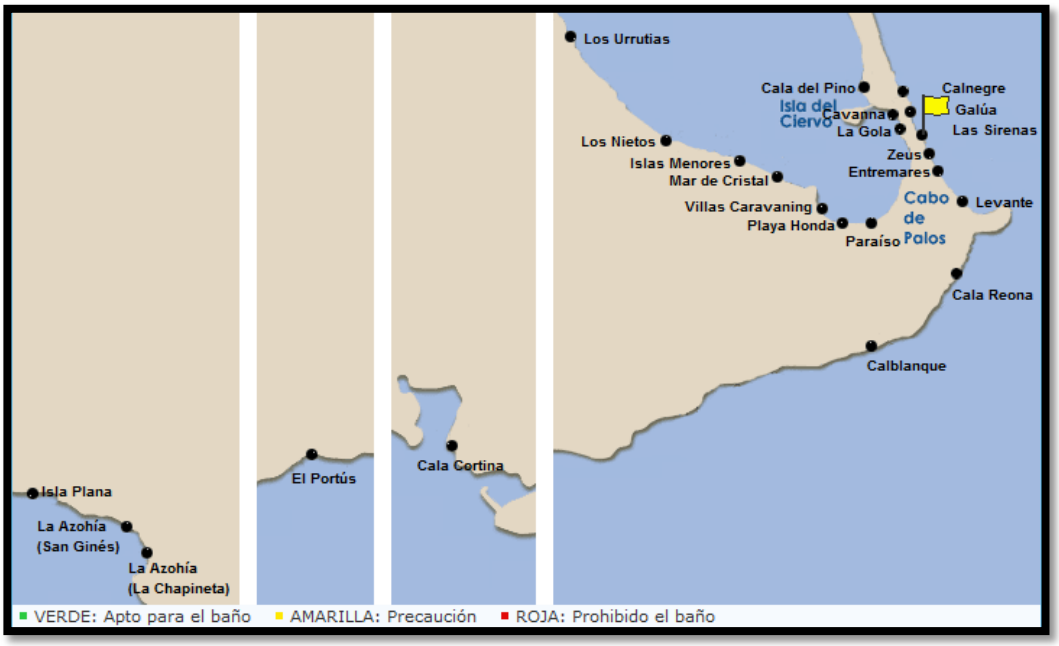


Ilustración 7: Información disponible de estado de las playas

- Redes de abastecimiento, saneamiento y pluviales, totalmente tele gestionada con funcionamiento vinculado a los procesos de pluviometría.

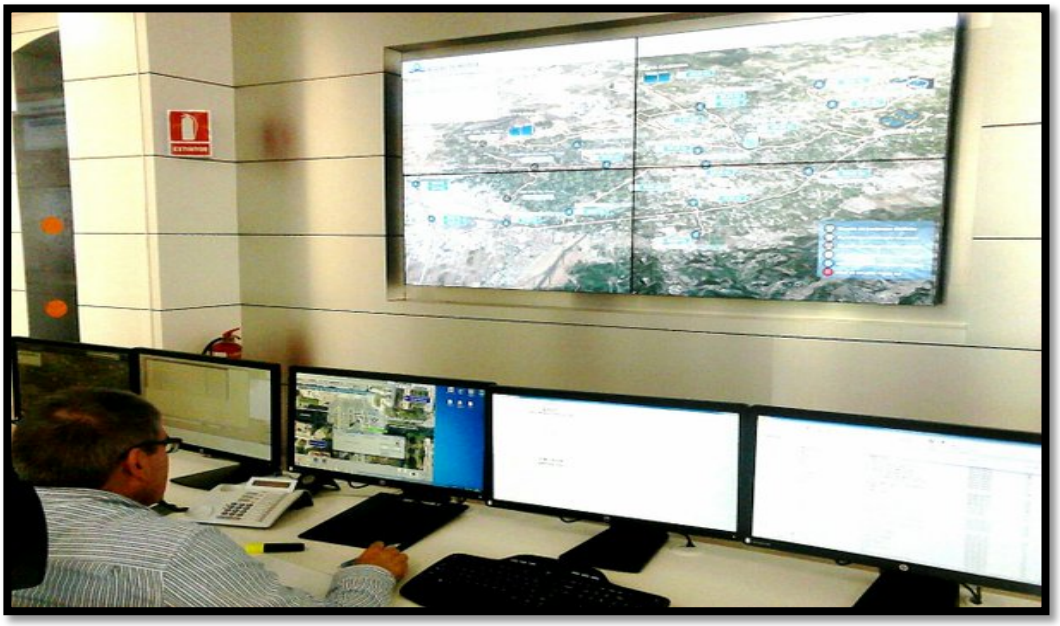


Ilustración 8: Redes de abastecimiento, saneamiento y pluviales

En este último punto, se pensó que sería muy interesante mostrar toda la información al usuario, ya que en Hidrogea se dispone de mucha información pero que todavía no se muestra al usuario, como puede ser información de los pluviómetros, imbornales, estado de las obras en curso, programa de desratización, etc.

1.3.2. Cloud Computing

El término **cloud computing** hace referencia a una concepción tecnológica y a un modelo de negocio que reúne ideas tan diversas como el almacenamiento de información, las comunicaciones entre ordenadores, la provisión de servicios o las metodologías de desarrollo de aplicaciones, todo ello bajo el mismo concepto: todo ocurre en la nube.

El concepto de cloud computing es un término acuñado recientemente. Fue muy usado durante años para referirse a toda aplicación que se despliega en Internet, se usa en un móvil, etc., pero esta afirmación no es completamente cierta: básicamente el cloud computing es el acceso a recursos de computación de otros, auto gestionable, pero sin acceso a detalles relacionados con la infraestructura del proveedor, con un mínimo esfuerzo de gestión, ya que la reserva de recursos es automatizada.

Las grandes empresas desde hace años disponen de mecanismos para gestionar gran cantidad de recursos computacionales (CPU, almacenamiento, comunicaciones, etc.), y que requieren de una gran infraestructura y de personal muy cualificados.

Cloud computing pretende que las empresas pequeñas puedan tener esos mismos servicios sin los costes asociados. Para tener por lo tanto un ahorro para los usuarios y mayores ingresos para los proveedores.

Cloud computing se creó cuando Amazon en 2006 decidió ofrecer sus recursos computacionales sobrantes a particulares.

Tratando de engranar todas estas cuestiones con vistas a la definición de qué es el cloud computing, quizá haya que detenerse un instante y definir el concepto de nube o, para ser más exactos, de internet. Internet, definida de una manera deliberadamente simple, es un conjunto de ordenadores, distribuidos por el mundo y unidos por una tupida malla de comunicaciones, que ofrece espacios de información a todo el que tenga acceso. El acceso a la información que ofrecen los ordenadores que componen Internet es “transparente”, es decir, no es relevante para el usuario el lugar en el que está alojada físicamente la información. De ahí que Internet se represente de una manera universal, como una nube a la que se accede en busca de información y servicios.

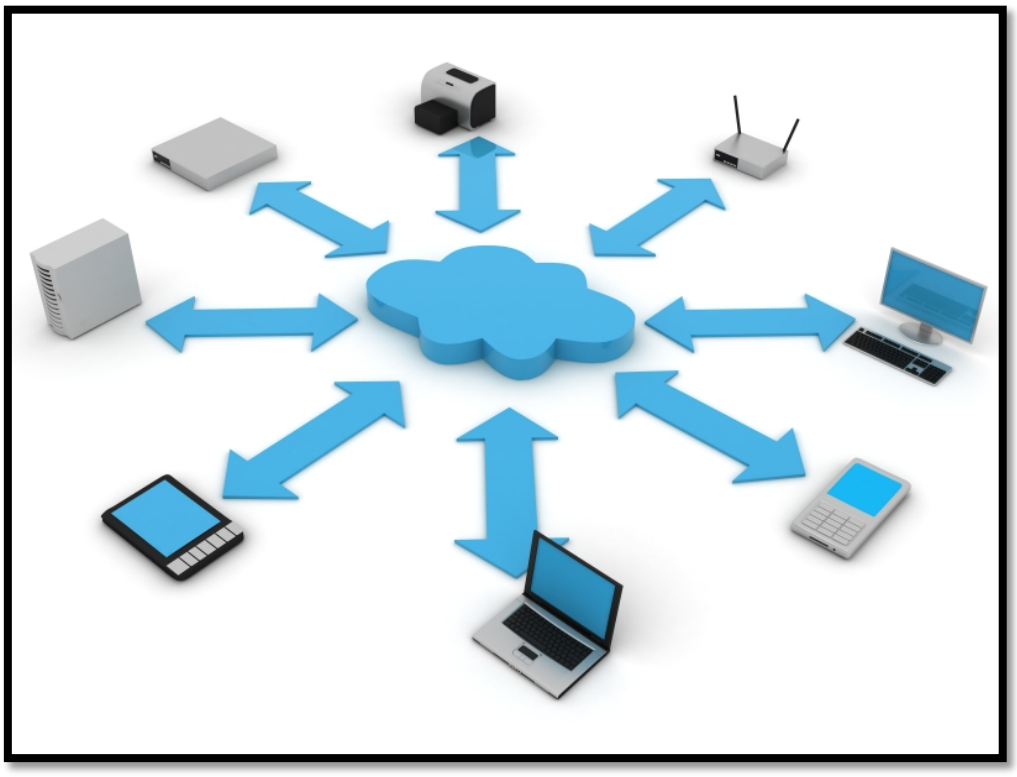


Ilustración 9: Cloud Computing

No todo lo que ocurre en Internet es cloud computing; Internet es un universo que, básicamente, ofrece dos cosas: publicación de información y oferta de servicios. Se puede afirmar que la mera publicación de información no forma parte del modelo de cloud computing, así que, con esto, se obtiene una primera frontera que separa lo que está dentro del ejercicio de definición de aquello que no lo está. Por lo tanto, se hablará de los servicios.

Internet es también un gran mercado de servicios de diversa naturaleza y formato que se podría dividir en dos grandes grupos, en base al uso que se le da en la red: los servicios que utilizan la red como canal y los que se encuentran en la red y le ofrecen recursos propios. Respecto a los primeros, se piensa en un banco que ofrece sus servicios transaccionales, oficinas virtuales de atención al cliente, canales de venta o subasta. En realidad, la utilidad de internet en estos procesos no es sino un mero canal de comunicación. Estos servicios no se consideran cloud computing.

En cuanto a los servicios que se encuentran en la red y le ofrecen recursos propios, destacan los servicios de hosting que permiten guardar información fuera de los ordenadores, es decir, en servidores que están en la nube y a los que se puede acceder a través una red de comunicaciones. Otro ejemplo sería el servicio de correo electrónico, en este caso todo, tanto la aplicación que se utiliza como los datos que se intercambian con los destinatarios, están almacenados en la nube. Estos servicios sí pueden considerarse cloud computing.

Por tanto, y resumiendo todo lo tratado anteriormente, se podría definir cloud computing como una concepción tecnológica y un modelo de negocio en el que se prestan servicios de almacenamiento, acceso y uso de recursos informáticos esencialmente radicados en la red, en los que el concepto de canal es un mero instrumento.

1.3.2.1. Recursos que la red ofrece como servicio

¿Y cuáles son los recursos que la red ofrece como servicio? Se puede considerar tres tipologías básicas de servicios que constituyen el modelo de negocio del *cloud computing* y que, dentro del vocabulario informático han venido a llamarse la generación *"As a service"* (IaaS, PaaS, SaaS).

Para la definición de estos términos se seguirá el camino que siguen los datos de los ordenadores hacia la nube, comenzando por los servicios más vinculados a las máquinas (hardware) hasta llegar a los de naturaleza más lógica (software) pasando por los que hacen posible estos últimos (herramientas de desarrollo).

Infraestructura como servicio (IaaS)

Ofrecer al cliente espacio de almacenamiento o capacidad de procesamiento en sus servidores. Así el usuario tendrá a su disposición “un disco duro de capacidad ilimitada” y un procesador de rendimiento casi infinito, solo restringido a su capacidad económica de contratación del servicio. Este servicio se basa en el acceso al uso de hardware radicado en la nube.

El cliente puede desplegar cualquier tipo de software en las instalaciones del proveedor (sistema operativo + aplicaciones) y no gestiona la infraestructura del proveedor, solo tiene control sobre el software que ha desplegado.

Plataforma como servicio (PaaS)

El servicio de Plataforma pone a disposición de los usuarios herramientas para la realización de desarrollos informáticos, de manera que aquellos pueden construir sus aplicaciones o piezas de software sin necesidad de adquirir e implantar en sus ordenadores locales dichas herramientas. Este servicio tiene dos claras ventajas para el desarrollador de aplicaciones: no tiene que adquirir las costosas licencias para desarrollo de las herramientas de mercado y, por otra parte, el proveedor de servicios se encarga de que dichas herramientas estén en óptima situación de mantenimiento.

Software como servicio (SaaS)

En el punto más alto de las habituales clasificaciones de componentes del mundo informático se encuentran las aplicaciones finales; productos terminados que ofrecen servicios concretos para los que fueron desarrollados. Estas aplicaciones son infinitas en sus distintas naturalezas y usos. El servicio conocido como SaaS consiste directamente en la utilización por parte del usuario final de los servicios ofrecidos por dichas aplicaciones, situadas en los servidores del proveedor cloud, con un mecanismo de facturación (en caso de no ser un servicio gratuito) más o menos simple, de pago por uso.

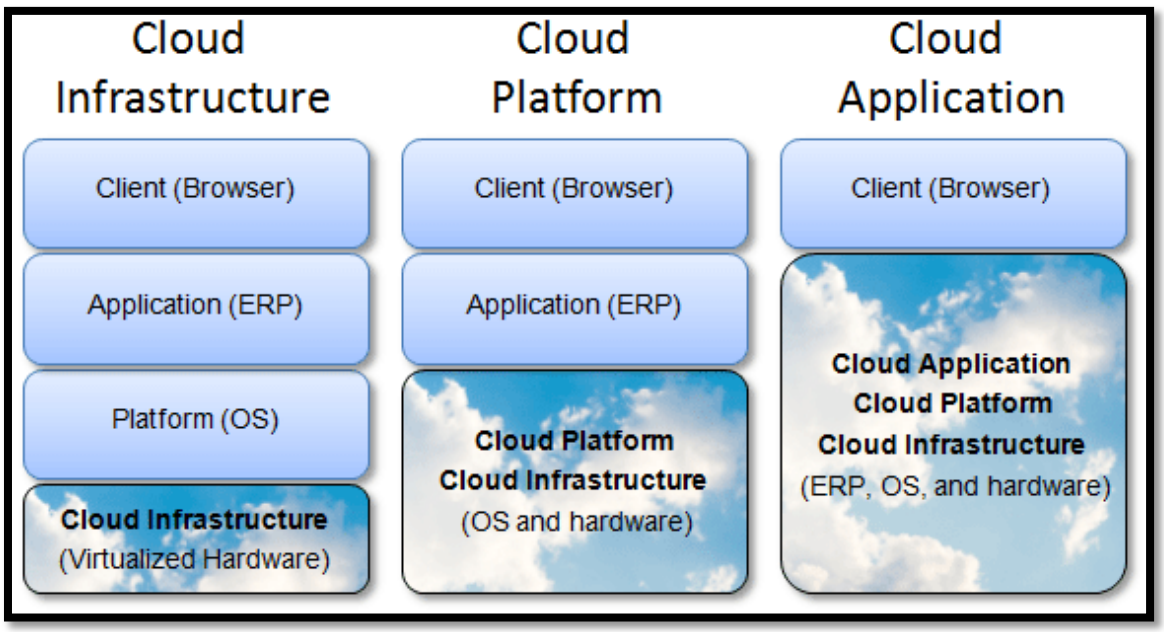


Ilustración 10: Servicios Cloud

1.3.2.2. Modelos de nube según la privacidad

El hecho de que la información manejada resida temporal o definitivamente en servidores en la nube lleva a que dichos servicios ofrezcan distintos formatos de privacidad que pueden elegir los usuarios. De ahí que se planteen varios modelos de nubes como espacios de desarrollo de los servicios ofertados. Estos serían:

- **Nubes públicas**

  Los usuarios acceden a los servicios de manera compartida sin que exista un exhaustivo control sobre la ubicación de la información que reside en los servidores del proveedor. El hecho de que sean públicas no es un sinónimo de que sean inseguras.

- **Nubes privadas**

  Para los clientes que necesiten, por la criticidad de la información que manejen, una infraestructura, plataforma y aplicaciones de su uso exclusivo. Sólo disponible para los miembros de una determinada organización.

- **Nubes comunitarias**

  Disponibles para clientes de organizaciones con intereses similares. No se suele usar.

- **Nubes híbridas**

  Combinan características de las anteriores, de manera que parte del servicio se puede ofrecer de manera privada (por ejemplo, la infraestructura) y otra parte de manera compartida (por ejemplo, las herramientas de desarrollo).

1.3.2.3. Cinco características esenciales que debe cumplir un servicio para considerarse 'cloud'

El NIST (National Institute of Standards and Technology) es un estándar para saber si un servicio es cloud, del departamento de Comercio de los Estados Unidos de América para citar las cinco características esenciales que debe cumplir un servicio para considerarse cloud.

- **Autoservicio bajo demanda**

  Un usuario debe poder, de forma unilateral, proveerse de recursos informáticos tales como tiempo de proceso o capacidad de almacenamiento en la medida de sus necesidades sin que sea necesaria la intervención humana del proveedor del servicio.

- **Acceso amplio a la red**

  Los servicios proporcionados deben poder ser accesibles a través de mecanismos estándares (HTTP, etc.) y desde plataformas heterogéneas (por ejemplo: ordenadores, teléfonos móviles o tabletas).

- **Asignación común de recursos**

  Los recursos son puestos a disposición de los consumidores siguiendo un modelo de multipropiedad, asignándose y reasignándose dispositivos físicos o lógicos atendiendo a la demanda de dichos consumidores. En este sentido el usuario no tiene un estricto control del lugar exacto en el que se encuentra su información, aunque sí debe poder especificar un ámbito mínimo de actuación (por ejemplo: un país, un estado o un centro de proceso de datos concreto).

- **Capacidad de rápido crecimiento (elasticidad)**

  Si un cliente necesita más recursos se le asignan con facilidad, incluso automáticamente. Al cliente le debe de parecer que los recursos del servidor son infinitos.

- **Servicio medible**

  Los sistemas *cloud* deben controlar y optimizar sus recursos dotándose de capacidades para medir su rendimiento en un nivel de abstracción suficiente para la naturaleza del servicio proporcionado. Además, dicho control debe permitir ser reportado de manera transparente tanto al proveedor del servicio como al consumidor del mismo.

1.3.2.4. Ventajas del cloud computing

Las ventajas más destacables del cloud computing son:

- **Simplificación del proceso de desarrollo:**
  - ✓ Proveedores ofrecen servicios como balanceado de carga, almacenamiento de datos, envíos de emails, etc.
- **Costes:**
  - ✓ Coste de desarrollo menores.
  - ✓ Inversión inicial baja.
  - ✓ Pago por uso. Si no se usa no se paga.
- **Escalado y provisión instantánea de recursos:**
  - ✓ Los recursos según las necesidades en cada instante se pueden contratar o liberar, de forma casi instantánea.

1.3.2.5. Inconvenientes del cloud computing

Las inconvenientes más destacables del cloud computing son:

- **Dificultad para portar aplicaciones:**
  - ✓ Aplicaciones ya existentes (no cloud) pueden requerir un rediseño importante para llevarlas a un proveedor de cloud.
  - ✓ Una aplicación para un determinado proveedor puede ser difícil de portar a otro proveedor
- **Uso de bases de datos:**
  - ✓ Las bases de datos relacionales son difíciles de escalar.
    - ▪ Precio elevado en cloud.
  - ✓ En la medida de lo posible se usan bases de datos no relacionales (para BD grandes).
- **Aplicaciones deben ser tolerantes a fallos:**
  - ✓ Las máquinas en un entorno cloud pueden fallar (baratas y sobrecargadas).
  - ✓ Según lo contratado se crean y destruyen instancias (servidores) según demanda.
- **Posibles fallos del proveedor de Cloud (caída del servicio):**
  - ✓ Muy poco habitual.
  - ✓ Quiebra del proveedor.

- Seguridad:
  - ✓ El proveedor de Cloud puede tener vulnerabilidades, aunque por lo general son seguros.

- Muchas aplicaciones realmente no requieren un servicio cloud (no se requiere escalabilidad, se van a ofrecer a pocos clientes, se tienen recursos infrautilizados, personal capacitado, etc.)
  - ✓ Aun así, podrían ser más económicas con proveedores Cloud.

Hasta ahora se ha desarrollado el estudio del arte tanto de Smart Cities como de Cloud Computing. A partir de toda esta información reunida, se llega a la conclusión de que una aplicación alojada en la nube y con toda la información disponible sobre Cartagena como una ciudad inteligente puede ser una herramienta con mucha utilidad para el ciudadano, ya que podrá acceder a toda esta información a través de su terminal, ya sea web o móvil, y en cualquier momento y lugar.

2. Solución propuesta

La solución que se propuso fue la de crear una Plataforma de notificaciones en la que se mantuviera toda la información existente actualmente en la ciudad. Para saber qué información era la que había disponible se pasó a investigar todos los portales de información que disponía tanto el ayuntamiento como en Hidrogea.

Fue en esta etapa en la que se encontró toda la información que actualmente hay disponible para los ciudadanos pero que no se muestra de forma adecuada, ya que para una persona sería mucho más sencillo disponer de una app en la que cada uno de los apartados contuviera información referente a la ciudad.

Por lo que el objetivo de este TFM fue el de desarrollar una plataforma web hibrida de notificaciones, para la plataforma Android, disponible para el usuario, tanto en entorno web como en móvil, que tuviera acceso ubicuo y que actualizara toda la información de forma dinámica. Además, debía de disponer de actualización de los contenidos de forma remota por los administradores vía web, fácil gestión, acceso ubicuo y la posibilidad de sincronización con contenidos de terceros usando infraestructura IoT.

La primera propuesta que se hizo de la App fue la siguiente:

Ilustración 11: Primera propuesta de la App

Como se puede observar en la imagen, se trataba de una App con los siguientes apartados:

- Transporte.
- Ocio y Cultura.
- Medio Ambiente.
- Salud.
- Ayuntamiento.
- Sugerencias.

El pensamiento de añadir todos estos apartados surgió a partir de una Brainstorming (tormenta de ideas) que es una técnica grupal para la generación de ideas. En esta reunión, cada uno de los participantes iban dando sus ideas y opiniones acerca de las necesidades que tenía la ciudad y lo útil que sería para el ciudadano el de disponer de información sobre cada uno de esos apartados en una sola plataforma.

El apartado que más se destacó fue el de sugerencias, en el que el usuario de la aplicación podría añadir cualquier recomendación, proposición, incidencia, etc., sobre la aplicación o sobre la ciudad de Cartagena, adjuntando además una imagen si así lo deseaba.

En las siguientes secciones se verá cómo ha ido avanzando la aplicación a lo largo de todo este proyecto, ya que ha ido pasando por diferentes fases para ajustarla lo máximo posible a las necesidades del usuario.

3. Descripción técnica de la solución

3.1. Arquitectura

Hoy en día basta con mirar al alrededor para darse cuenta del avance y presencia de la tecnología en todas partes; siendo el concepto de "movilidad" el más apreciado, pues permite usar dispositivos con funciones cada vez más útiles y novedosas sin importar el lugar donde se encuentre y sin necesidad de cables.

Vivimos en una época de cambios excitantes en cuanto a los modos en los que se accede a Internet propiciados por el avance de los smartphones y tablets. Teniendo en cuenta el comportamiento de los usuarios, una aplicación móvil podría ser el vehículo perfecto para establecer un canal de comunicación directo con el público. Según un reciente estudio publicado por IAB Spain, el acceso a Internet a través de una app se ha consolidado. Prácticamente no hay diferencia cuantitativa entre acceso mediante navegador y por aplicación.

Las apps se han convertido en el instrumento favorito de una sociedad siempre conectada. Además de proporcionar rapidez y agilidad de uso, esta pieza de software establece un vínculo directo con los usuarios, algo que puede ser muy rentable si se sabe aprovechar.

El desarrollo de aplicaciones puede ser algo muy costoso. Se debe definir previamente qué necesidades se quieren cubrir con la app y qué se va a conseguir con ella desde el punto de vista del marketing móvil.

Aunque es muy importante tener la web adaptada, las apps permiten utilizar todas las funcionalidades del terminal de manera directa. Se habla, por ejemplo, de la localización por GPS o el uso de la cámara como forma de interacción. Las posibilidades con este tipo de software son mucho mayores que con la web móvil.

Una app asegura una presencia constante de la marca en el dispositivo móvil del usuario. Las posibilidades de interacción aumentan. Además, la sincronización entre app y redes sociales mejora la difusión de los contenidos.

Las aplicaciones móviles deben concebirse para estar completamente adaptadas a la audiencia, para ser útiles y aportar un valor añadido. Si la app se limita a repetir la información que se tiene en la página web, no valdrá para nada. El uso de funcionalidades especiales, servicios prácticos, ofertas y promociones hará que el usuario recurra con frecuencia a la aplicación con el objetivo de conseguir algo muy concreto. En una palabra, **fidelización**.

Los datos del usuario están mucho más protegidos con una aplicación. El acceso a Internet desde un navegador siempre entraña más riesgos. Esto no quiere decir que se deba de confiar ciegamente en todas las aplicaciones que se encuentren. Lo primero es asegurarse de que es una app oficial, que ha sido debidamente validada. De encontrarse con algún malware, el daño podría ser muy grande, ya que la app falsa tendría acceso a todos los datos personales. Por ello, la **seguridad** es un punto clave en el desarrollo de aplicaciones.

Con una app, los clientes pueden acceder rápidamente a la información de la empresa. El usuario puede contactar con ellos de una forma más ágil que por otras vías. Y lo mejor, gratis. De la misma forma, la marca puede desarrollar una interacción directa con su público en función de los objetivos que se hayan marcado (comerciales, informativos, etc.).

Antes de continuar se debe pensar que tipo de tecnología es la más adecuada para el desarrollo de esta aplicación móvil. Desde un punto de vista tecnológico, no todas las aplicaciones son iguales. A continuación, se explican los tipos de aplicaciones existentes en la actualidad.

3.1.1. Aplicaciones nativas

Una aplicación "nativa", es una aplicación móvil desarrollada en el lenguaje específico para esa plataforma. En otras palabras, si se desea que la aplicación funcione en iPhone, Android y Windows Phone, se tiene que desarrollar la misma aplicación en tres versiones distintas, una para cada plataforma.

Cada versión de esta aplicación se distribuye/comercializa a través de las tiendas online correspondientes: Apple Store (iOS), Google Play (Android), AppWorld (BlackBerry) y Windows Marketplace (Windows Mobile).

Después de su descarga al móvil o tablet, esta aplicación se instala en el sistema de archivos de cada dispositivo.

Características principales de las aplicaciones móviles "nativas"

- Para cada sistema operativo/plataforma hay que desarrollar una aplicación distinta: Android, iOS, Blackberry, Windows Mobile.

- Se desarrollan con lenguajes distintos (Java para Android, Objective-C para iOS, por ejemplo)
- Se distribuyen a través de las tiendas oficiales:
    - Google Play
    - Apple Store
    - AppWorld
    - Windows Marketplace
- Las aplicaciones se instalan en el disco duro o tarjeta de memoria del dispositivo móvil.
- Permiten al usuario acceder con ellas a todos los recursos del dispositivo (cámara, contactos, GPS, NFC ...) e interactuar con otras aplicaciones nativas.

Ventajas de las aplicaciones móvil nativas

1. Posibilidad de utilizar funcionalidades del teléfono: cámara, GPS, contactos, acelerómetro, etc.
2. Rendimiento: Es más rápida que una aplicación híbrida, puede interactuar con otras aplicaciones nativas, saca mejor partido a los recursos del teléfono, uso de CPU, memoria y batería óptimo.
3. Existe un sitio para compartir y comercializar las aplicaciones: Apple Store, Google Play, Blackberry AppWorld, Windows Marketplace.
4. Funciona en modo online y offline: Se puede utilizar sin la necesidad de una conexión a internet.
5. El proceso de verificación y aceptación por parte de Apple o Google suele ser más rápido que en una aplicación híbrida.
6. Almacenamiento local seguro.

Desventajas de las aplicaciones móvil nativas

1. El coste, aunque si se precisa el desarrollo de la aplicación para una única plataforma entonces suele ser más barato.

2. Tiempo de desarrollo es mayor, en el caso que deba desarrollarse una aplicación para cada plataforma.
3. Mayor Coste de mantenimiento: Mantener varias versiones de la misma aplicación para distintas plataformas es costoso.
4. Tiempo de aprobación: La aprobación de una nueva aplicación en el Apple Store puede llegar a tardar un par de semanas. En Google Play son 24h. Cualquier modificación, actualización o nueva funcionalidad de la aplicación tiene que pasar por el mismo sistema de aprobación.
5. Coste adicional: Apple Store y Google Play cobran por cada aplicación disponible en su tienda 30% del precio de la aplicación. Así como un 30% de los ingresos obtenidos a través de la aplicación mediante in-app purchase (compras desde la propia aplicación).

3.1.2. Aplicaciones Web Móvil

El desarrollo de Aplicaciones Web para dispositivos móviles es el desarrollo de páginas web que son optimizadas para ser visualizadas en las pantallas de dispositivos móviles y para ser utilizadas en pantallas táctiles.

El usuario accede a estas aplicaciones mediante el navegador web de su dispositivo, aunque en función del dispositivo puede crear un enlace directo en su escritorio y acceder a ella como si se tratara de cualquier otra aplicación.

Características principales de las aplicaciones web móviles

1. Se acceden mediante un navegador web del dispositivo.
2. La aplicación se visualizará de forma casi idéntica en todos los dispositivos, dependiendo de la resolución de la pantalla.
3. Las tecnologías utilizadas son las mismas que para un sitio web. Se utiliza HTML, CSS y JavaScript.
4. No se distribuyen mediante sitios oficiales de Apple Store o Google Play lo que permite no tener que pagar las tasas y no tener las limitaciones impuestas por estos.
5. Funcionan principalmente online, pero pueden contar con caché local y almacenamiento de datos local para el funcionamiento offline.

Ventajas de las aplicaciones web móvil

1. Coste: Se desarrolla una única aplicación para todos los sistemas operativos a diferencia de una aplicación nativa.

2. Rapidez de desarrollo: Al desarrollar para todas las plataformas a la vez no se multiplica el tiempo de desarrollo.

3. Compatibilidad: Se reutiliza casi en un 100% el mismo código fuente para todos los sistemas operativos de los dispositivos móviles. Se puede optimizar para las distintas resoluciones de pantallas móviles.

4. Mantenimiento de un único código para todas las plataformas.

5. Distribución: Se puede evitar la subida a los repositorios oficiales online (App Stores). Es posible descargar la aplicación desde una página web o accediendo a una URL concreta.

6. Actualizaciones inmediatas: Las actualizaciones de la aplicación son inmediatas, a diferencia de unas semanas de espera de aprobación por Apple Store.

7. Posibilidad de SEO: Ayuda a mejorar la visibilidad de su aplicación puesto que esta y sus contenidos están disponibles en la web.

8. Visibilidad: La URL de la aplicación web es fácil de compartir, e incluso se puede redirigir el tráfico desde redes sociales, blogs, códigos QR y medios de publicidad.

Desventajas de las aplicaciones web móvil

1. Integración con los componentes nativos del dispositivo: Las aplicaciones web, al ser ejecutadas en el navegador, solo pueden acceder a determinados componentes y funcionalidades nativas del dispositivo, aunque las posibilidades evolucionan con rapidez.

2. Diferencias entre dispositivos. El nivel de incorporación de los estándares en los navegadores de cada dispositivo es bastante heterogéneo. Es posible que se dedique más tiempo en testear y corregir problemas de una aplicación web debido al comportamiento distinto de los navegadores que en el propio desarrollo de la misma.

3. El uso offline: Las capacidades de almacenamiento local de las aplicaciones Web son limitadas, solicitándose al usuario la confirmación en caso de sobrepasar ciertos límites.

4. Fluidez: Una interfaz de usuario desarrollada en HTML5 no es tan fluida como una nativa.

Ilustración 12: Comparación de aplicaciones WEB con NATIVAS

3.1.3. Aplicaciones Híbridas

Una aplicación híbrida o multiplataforma, como su nombre bien indica, es una "mezcla entre una aplicación Nativa y una WebApp". Este tipo de aplicaciones se hicieron populares gracias al framework Phonegap (hoy Apache Cordova) pero existen varios que pueden ser utilizados: Kendo UI Mobile, Sencha Touch, Trigger.io o Titanium Appcelerator.

Al desarrollar aplicaciones híbridas se utiliza la tecnología nativa (conjunto de APIs) cuando es necesario o cuando más conviene (para acceder a cámara, acelerómetro, contactos, etc.), y la tecnología web (como HTML5, CSS3 y JavaScript) para el desarrollo de la estructura e interfaz de la aplicación. De este modo se maximiza la base de código que es común a las distintas plataformas y se limita el desarrollo de la funcionalidad nativa a aquellos aspectos que no puedan ser desarrollados de otro modo.

Ejemplos de aplicaciones híbridas:

- Instagram: Utiliza tecnología nativa para tomar, editar y publicar las fotos (incluso sin conexión a internet) y la tecnología web para desplegar las fotos y el perfil. Esto permite a los desarrolladores mejorar la lista de fotografías sin la necesidad de publicar una nueva versión. Si no hay conexión a internet, la fotografía estará en espera para ser subida una vez haya conexión.
- LinkedIn: Híbrida para iOS y nativa para Android.
- Facebook: Ha cambiado de una aplicación totalmente híbrida a una nativa con funcionalidades híbridas.

La solución híbrida pretende aprovechar las capacidades de las aplicaciones nativas permitiendo la re-utilización de la mayor parte del código para todas las plataformas.

Ventajas de una aplicación Híbrida

1. Se minimiza el código específico: La mayor parte del código puede utilizarse para el resto de plataformas. Solo se utiliza código nativo para aquellos aspectos que lo requieran.
2. Menor coste de desarrollo sobre todo si se requiere la aplicación en varias plataformas.
3. Menor coste de mantenimiento al ser la mayor parte del código común a todas las plataformas.
4. Una aplicación Híbrida puede acceder a los recursos del dispositivo móvil prácticamente como una nativa.
5. Se distribuye mediante los respectivos “stores”.

Desventajas de una aplicación Híbrida

1. Rendimiento: El rendimiento y la experiencia de usuario no pueden alcanzar los niveles de la aplicación nativa. Incluso los dispositivos más actuales tienen problemas gestionando interfaces desarrollados en HTML5.
2. Habitualmente los procesos de aprobación en los correspondientes "stores" para aplicaciones hibridas son más estrictos y pueden llegar a ser rechazadas si no queda clara la funcionalidad proporcionada mediante la carga de código remoto.

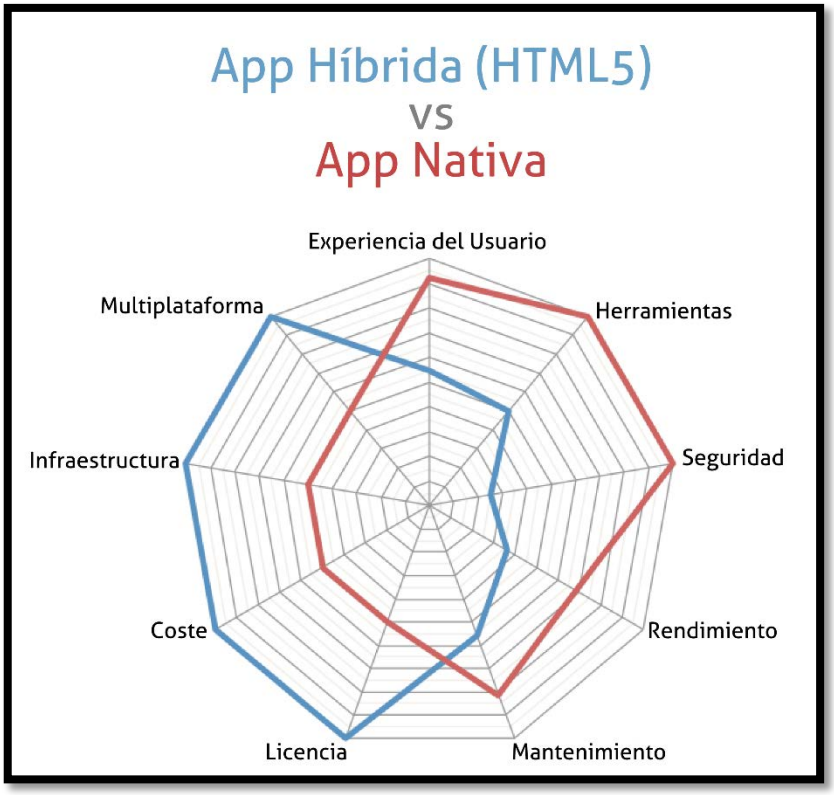


Ilustración 13: Comparación de App Híbrida con App Nativa

	Web	Híbrida	Nativa
Costes de desarrollo	Razonable	Razonable	Caro
Tiempo de desarrollo	Corto	Corto	Largo
Portabilidad	Alto	Alto	Ninguna
Rendimiento	Rápido	Velocidad nativa si se necesita	Muy rápido
Funcionalidad Nativa	No	Todas*	Todas
Distribución en AppStores	No	Si	Si
Extensibilidad	No	Si	Si

Ilustración 14: Comparativa en el desarrollo de aplicaciones.

Una vez evaluadas las ventajas y desventajas de los diferentes tipos de aplicaciones disponibles, se llega a la conclusión de que la mejor opción, es la de desarrollar un tipo de aplicación hibrida ya que de esta manera se puede tener todas las funcionalidades de una aplicación nativa teniendo un menor coste de desarrollo y mantenimiento.

3.2.Lenguajes y herramientas SW utilizados

3.2.1. Entorno de desarrollo

El editor de texto utilizado para el desarrollo de la aplicación es Brackets. Brackets es un editor de código abierto para el diseño y desarrollo web construido sobre tecnologías como HTML, CSS y JavaScript. El proyecto fue creado y es mantenido por Adobe, y se distribuye bajo una licencia MIT.

Ilustración 15: Tecnologías de Brackets.

Brackets tiene una interfaz muy limpia y minimalista. Su uso es muy intuitivo, como el de cualquier otro editor de texto y en vez de usar pestañas para mostrar los diferentes archivos, emplea un explorador de archivos en el lateral muy cómodo.

Brackets te permite trabajar directamente en el navegador editando el código al instante, estableciendo breakpoints y moviéndose con fluidez entre las diferentes vistas de código y del mismo navegador. El editor de texto dentro de Brackets se encuentra basado en CodeMirror.

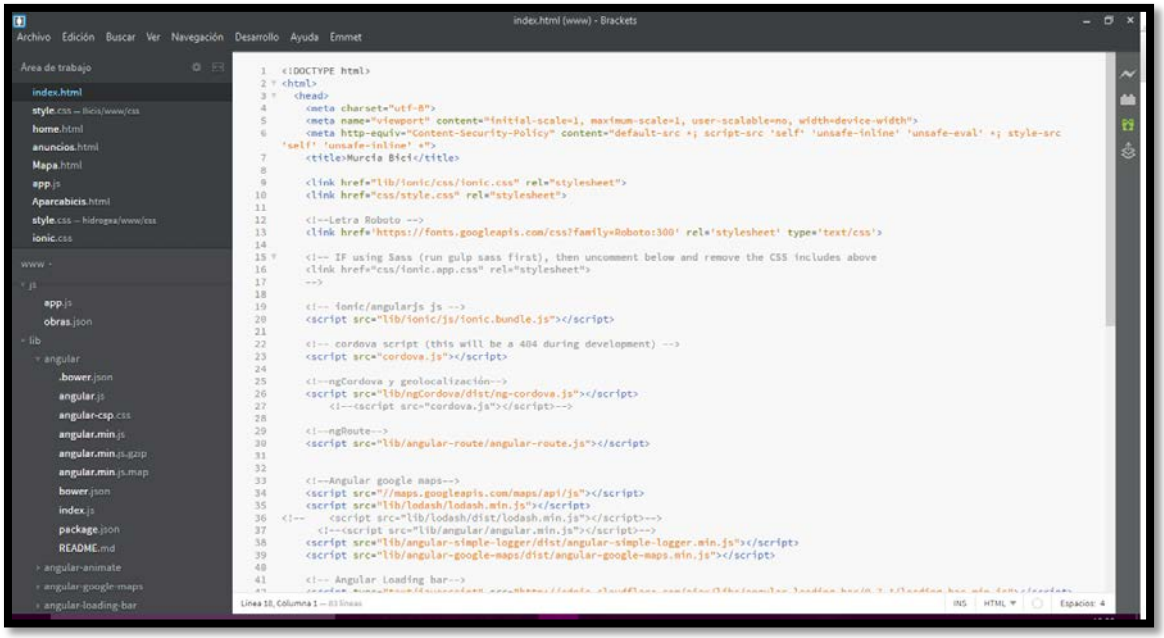


Ilustración 16: Interfaz del editor de texto Brackets.

Características especiales de Brackets para el desarrollo web.

- Vista previa en vivo. Con características como la vista previa y edición rápida, permiten que los cambios que se realicen se vean reflejados instantáneamente en la ventana de previsualización.

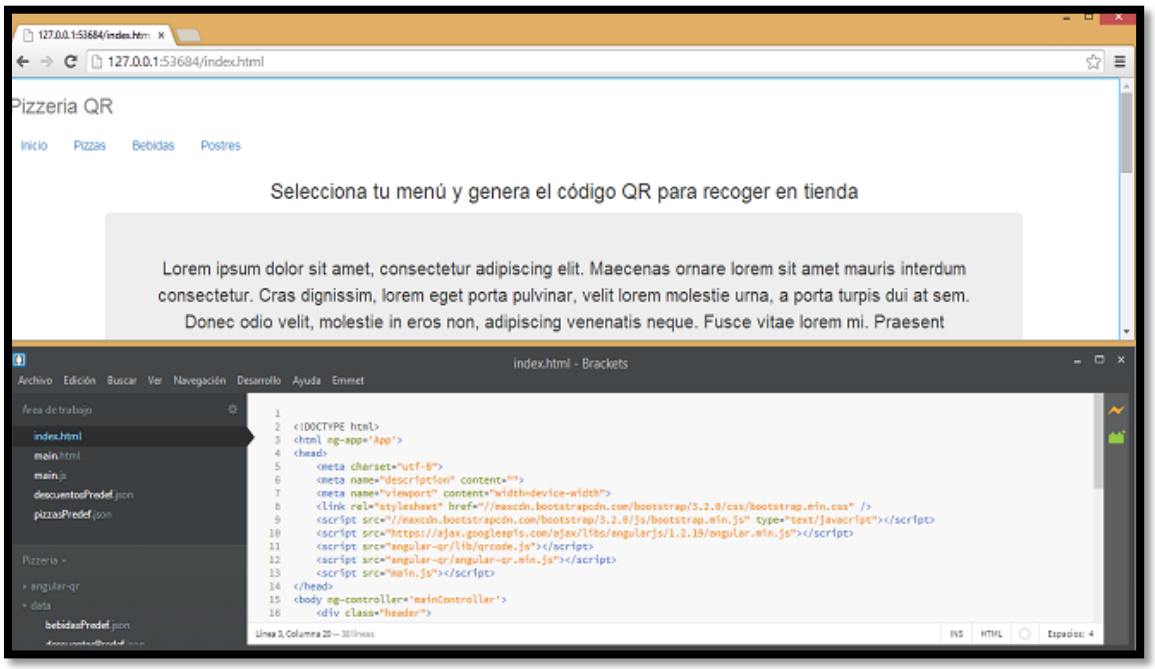


Ilustración 17: Editor Brackets con la vista previa en vivo activada.

- Complementos para Brackets. Brackets dispone de una biblioteca de extensiones que está creciendo rápidamente, ya que el interés en el soporte para este editor de texto aumenta en la comunidad de colaboradores. La mayoría de las extensiones no alteran el programa de ninguna manera drástica; al contrario, debido a los lenguajes de programación en los que se basa, da la posibilidad de brindar mejoras fácilmente por parte de los mismos usuarios.

  En este proyecto se han utilizado los siguientes plugins:

  - Emmet. Es un plugin para muchos editores de texto desarrollado y optimizado para desarrolladores web cuyo flujo de trabajo depende de HTML/ XML y CSS. La mayoría de los editores de texto permiten almacenar y reutilizar trozos de código llamados "fragmentos". Emmet toma la idea de fragmentos a un nivel completamente nuevo: puede escribir expresiones de CSS que se pueden analizar de forma dinámica, y producir una salida en función de lo que se escribe en la abreviatura.
  - HTML Block Selector. Esta es una manera rápida y fácil para seleccionar a la vez un bloque de HTML. Sólo "CTRL + clic" en la etiqueta en cuestión, y será seleccionado, junto con todo su contenido.
  - AutoSave Files. En el caso de este plugin, cada vez que se seleccione una ventana que no sea la de **Brackets**, los archivos se guardan automáticamente. Esto es perfecto para cualquier programador o desarrollador, o al menos para aquellos que estén cambiando entre programas de editor de imagen, navegador, u otra aplicación que manejen.

3.2.2. Lado del cliente

A continuación, se explican los lenguajes y tecnologías web utilizadas para el desarrollo de la plataforma de notificaciones del TFM.

HTML

El servicio World Wide Web (La Telaraña Mundial), también conocido como WWW o simplemente Web, es un Sistema de Información distribuido por Internet basado en la tecnología hipertexto/hipermedia, que proporciona una interfaz común a los distintos formatos de datos (texto, gráficos, vídeo, audio, etc.) y a los servicios de Internet existentes (FTP, news, telnet...). Todo esto hace que el servicio Web sea el servicio más utilizado en Internet.

Un documento hipertexto, es un texto en el que cualquier palabra puede ser especificada como un enlace a otros documentos que contienen más información sobre dicha palabra, por lo que la lectura de un documento hipertexto no es secuencial o lineal, sino que se puede acceder a la información que interese desde otros conceptos relacionados (simplemente haciendo clic con el ratón en la palabra relacionada), y de esta forma avanzar de documento en documento hasta encontrar la información deseada. Estas palabras que poseen enlaces a otros documentos están marcadas de alguna manera para poder diferenciarlas.

Un documento hipermedia es un hipertexto, pero que no incluye sólo información textual sino también información multimedia, es decir, puede incluir gráficos, vídeo, y sonido.

A pesar de las diferencias de estos dos conceptos, a menudo se utiliza el término hipertexto para designar el significado de hipermedia.

Los documentos Web o también llamados páginas Web pueden estar localizados en diferentes sitios de Internet, estos sitios son llamados servidores Web. De manera que un documento WWW puede contener enlaces a otros documentos que se encuentran en el mismo servidor Web o en otros servidores Web, logrando así formar una telaraña mundial de información.

El lenguaje estandarizado para la creación de páginas Web es el lenguaje **HTML** (HyperText Markup Language, Lenguaje de Marcas Hipertexto). HTML es un lenguaje muy sencillo que permite describir documentos hipertexto. La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado).

Para utilizar el servicio Web se necesita una aplicación cliente capaz de entender o interpretar información HTML, a este tipo de aplicaciones se le conoce como browsers/navegadores. Mediante el navegador el usuario puede acceder a los documentos HTML y moverse de un documento a otro a través de sus vínculos o enlaces, este hecho de moverse con el navegador por las páginas WWW a través de sus enlaces se le conoce como Navegar por Internet.

El browser sabe cómo acceder a cada recurso de Internet, sabe cómo acceder a un servidor de FTP anónimo, a un servidor de noticias, etc., y por supuesto cómo conectarse a los servidores Web. El mecanismo que utiliza el browser para acceder a un recurso en cualquier lugar de

Internet es el **URL** (Uniform Resource Locator, Localizador de Recursos Uniforme), comúnmente llamado dirección Internet.

Las URLs combinan el protocolo a utilizar para obtener el recurso: http (es el del Web), ftp, telnet, ...; junto con el nombre del host servidor, y el path completo del recurso (directorios y nombre de archivo). Los URLs constituyen en realidad los enlaces que permiten moverse de una página a otra, es decir Navegar por Internet, y que se puedan identificar dentro de una página WWW porque están incluidos comúnmente como Texto en color subrayado, también puede ir incluidos dentro de una imagen, etc.

HTML es el lenguaje con el que se escriben las páginas web. Las páginas web pueden ser vistas por el usuario mediante el navegador. Se puede decir por lo tanto que el HTML es el lenguaje usado por los navegadores para mostrar las páginas webs al usuario, siendo hoy en día la interfaz más extendida en la red.

El HTML se creó en un principio con objetivos divulgativos. No se pensó que la web llegara a ser un área de ocio con carácter multimedia, de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, sí que se han ido incorporando modificaciones con el tiempo, estos son los estándares del HTML.

Esta evolución tan anárquica del HTML ha supuesto toda una seria de inconvenientes y deficiencias que han debido ser superados con la introducción de otras tecnologías accesorias capaces de organizar, optimizar y automatizar el funcionamiento de las webs como el CSS y el JavaScript.

Otros de los problemas que han acompañado al HTML es la diversidad de navegadores presentes en el mercado los cuales no son capaces de interpretar un mismo código de una manera unificada. Esto obliga al webmaster a, una vez creada su página, comprobar que esta puede ser leída satisfactoriamente por todos los navegadores, o al menos, los más utilizados.

Además del navegador necesario para ver los resultados del trabajo, se necesita evidentemente otra herramienta capaz de crear la página en sí. Un archivo HTML (una página) no es más que un texto. Es por ello que para programar en HTML se necesita un editor de textos, en este caso, se ha utilizado Brackets como se ha comentado anteriormente.

El HTML es un lenguaje de marcas que basa su sintaxis en un elemento de base al que se le llama etiqueta. A través de las etiquetas se van definiendo los elementos del documento, como enlaces, párrafos, imágenes, etc. Así pues, un documento HTML estará constituido por texto y

un conjunto de etiquetas para definir la forma con la que se tendrá que presentar el texto y otros elementos en la página.

La etiqueta presenta frecuentemente dos partes: Una apertura <etiqueta> y un cierre </etiqueta>

Todo lo incluido en el interior de esa etiqueta sufrirá las modificaciones que caracterizan a esta etiqueta. Así, por ejemplo, las etiquetas y definen un texto en negrita. Si en el documento HTML escribimos una frase con el siguiente código:

```
<b>Esto está en negrita</b>
```

El resultado será:

Esto está en negrita

Las etiquetas y definen un párrafo. Si en el documento HTML se escribiera:

```
<p>Hola, estamos en el párrafo 1</p>
<p>Ahora hemos cambiado de párrafo</p>
```

El resultado sería:

Hola, estamos en el párrafo 1

Ahora hemos cambiado de párrafo

Además de todo esto, un documento HTML ha de estar delimitado por la etiqueta <html> y </html>. Dentro de este documento, se pueden distinguir dos partes principales:

El encabezado, delimitado por <head> y </head> donde se colocan las etiquetas de índole informativo como por ejemplo el título de la página.

El cuerpo, flanqueado por las etiquetas <body> y </body>, que será donde se coloca el texto e imágenes delimitados a su vez por otras etiquetas como las que se han visto.

El resultado es un documento con la siguiente estructura:

```
<html>
    <head>
        Etiquetas y contenidos del encabezado
```

```
        Datos que no aparecen en nuestra página pero que son importantes para catalogarla:
Titulo, palabras clave,...
    </head>
    <body>
        Etiquetas y contenidos del cuerpo
        Parte del documento que será mostrada por el navegador: Texto e imágenes
    </body>
</html>
```

Las etiquetas de comienzo y final de un elemento deben estar adecuadamente anidadas, esto significa que las etiquetas de cierre deben escribirse en el orden inverso al de las etiquetas de inicio. La regla del anidamiento de etiquetas tiene que cumplirse de forma escrupulosa para poder escribir código válido.

La etiqueta de comienzo puede contener información adicional, tal y como puede verse en el siguiente ejemplo. Dicha información es lo que se conoce como atributos. Los atributos suelen consistir en dos partes:

- Un atributo nombre (name).
- Un atributo valor (value).

Algunos atributos sólo pueden tener un único valor. Son atributos Booleanos y pueden ser incluidos para especificar el nombre del atributo, o dejar su valor vacío.

```
<option value="2" selected>2 horas</option>
```

Ilustración 18: Ejemplo de uso de un atributo.

Además de las etiquetas y el contenido, un documento de HTML debe contener una declaración doctype en la primera línea. En el HTML actual esto se escribe del siguiente modo:

```
<!DOCTYPE html>
```

Ilustración 19: Uso de doctype.

El doctype le dice al navegador que interprete el código HTML y CSS de acuerdo a los estándares web del W3C (Asociación Global que vela por mantener los estándares HTML) y que no trate de emular que se trata de un Internet Explorer de los 90's.

HTML tiene un mecanismo para poder introducir comentarios al código que no serán mostrados en la página cuando esta sea interpretada o leída por un navegador web. Esto suele emplearse para añadir explicaciones al código, o dejar notas para explicar a otras personas cómo trabaja el código de la página, o simplemente para dejar recordatorios para uno mismo. Los comentarios en HTML están contenidos entre los siguientes símbolos:

Ilustración 20: Comentario en HTML.

Limitaciones de HTML

Falta de interactividad

Uno de los principales problemas de HTML desde su inicio era la falta de mecanismos para interactuar con el usuario: un documento HTML es estático, no cambia ante las acciones del usuario (como mover el ratón).

Esto impide desarrollar contenidos interactivos y aplicaciones complejas para la web.

Solución: se añade código a los documentos que será ejecutado por el cliente (es decir, el navegador).

- Javascript, es el lenguaje "ligero" estándar y se ha convertido en parte fundamental de HTML 5.
- Otra opción es ejecutar código mediante plugins (como applets de Java).

HTML5

La versión actual de la especificación HTML se conoce como **HTML5**.

HTML5 es la nueva versión del lenguaje de marcado que se usa para estructurar páginas web, que actualmente todavía sigue en su evolución, gracias a él con características nuevas y modificaciones que mejorará significativamente este nuevo estándar.

HTML5 es mejor, simplemente, porque es una tecnología que supera al actual HTML, porque es lo nuevo que estandariza la W3C, porque es una nueva tecnología y como toda nueva tecnología siempre viene con cosas que van a impresionar, porque llega de la mano de CSS3,

una evolución notable de las hojas de estilo que se conocían y porque revaloriza el papel de JavaScript en la Web, como el lenguaje que "sabe hablar" con las nuevas APIs que llegan con HTML5.

En resumen, HTML5 conduce a una fusión entre JavaScript como lenguaje de programación, HTML como modelo semántico y CSS3 que es la evolución del CSS como el lenguaje de los estilos, que se dedica a dar un mejor aspecto a los proyectos.

A continuación, alguna de las reglas establecidas para HTML5:

- La nueva característica debe basarse en HTML, CSS, DOM y JavaScript.
- Reducir la necesidad de plugins externos (como Flash).
- Mejor manejo de errores.
- Más marcado para reemplazar secuencias de comandos.
- HTML5 debe ser independiente del dispositivo.
- El proceso de desarrollo debe ser visible para el público.

CSS

CSS [7] son las siglas de "Cascading Style Sheets", que se traduce como "Hojas de Estilo en cascada". CSS es un lenguaje utilizado en la presentación de documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Entonces se puede decir que el lenguaje CSS sirve para organizar la presentación y aspecto de una página web. Este lenguaje es principalmente utilizado por parte de los navegadores web de internet y por los programadores web informáticos para elegir multitud de opciones de presentación como colores, tipos y tamaños de letra, etc.

La filosofía de CSS se basa en intentar separar lo que es la estructura del documento HTML de su presentación. Por decirlo de alguna manera: la página web sería lo que hay debajo (el contenido) y CSS sería un cristal de color que hace que el contenido se vea de una forma u otra. Usando esta filosofía, resulta muy fácil cambiarle el aspecto a una página web: basta con cambiar "el cristal" que tiene delante. Piensa por ejemplo qué ocurre si tienes un libro de papel y lo miras a través de un cristal de color azul: que ves el libro azul. En cambio, si lo miras a través de un cristal amarillo, verás el libro amarillo. El libro (el contenido) es el mismo, pero lo puedes ver de distintas maneras.

Algunas opciones básicas del lenguaje CSS por ejemplo pueden ser el poder cambiar el color de algunas típicas etiquetas HTML como <H1> (h1 es una etiqueta en el lenguaje HTML destinada a mostrar un texto como encabezado, en tamaño grande). Pero también hay

funciones algo más complejas, como introducir espaciado entre elementos <DIV> (div es una etiqueta HTML para identificar una determinada región o división de contenido dentro de una página web) o establecer imágenes de fondo.

CSS es muy intuitivo y sencillo una vez se llega a aprender, ya que para su definición siempre se hace uso de un identificador de etiqueta HTML (como por ejemplo <H1>), y luego se indica con qué aspecto se quiere que se muestren todas las etiquetas <H1> que aparezcan en un documento. Al igual que con <H1> podemos definir cómo se quiere que se muestren las distintas partes del documento HTML, pudiendo en cada caso definir sus propiedades (color, tipo de fuente, tamaño, espacio, imagen) con algún determinado valor deseado.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

```
label{
    font-size: 12px;
}
```

Ilustración 21: Ejemplo de uso de un selector con su declaración.

En este ejemplo, label es el selector y font-size: 12px; es la declaración.

Ilustración 22: Ejemplo de uso de una etiqueta.

El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto. En el ejemplo anterior, el selector label indica que todos los elementos de la clase label se verán afectados por la declaración, donde se establece que la propiedad font-size va a tener el valor 12px para todos los elementos de la clase label del documento o documentos que estén vinculados a esa hoja de estilos.

Selector	Descripción
*	Selector universal, son todos los elementos del CSS
E	E representa cualquier elemento del tipo E (span, p, …)
E F	Todos los elementos F que sean descendentes de E
E > F	Todos los elementos F que sean hijos de E
E:first-child	De esta forma podemos seleccionar el primer elemento de tipo E
E:link , E:visited	Selecciona los elementos E que sean un enlace y no hayan sido visitados (:link) y los sí visitados (:visited)
E:active , E:hover , E:focus	Selecciona los elementos de tipo E, en sus correspondientes acciones.
E:lang(c)	Cogemos los elementos del tipo E que estén en el idioma (humano) especificado en (c).
E + F	Se trata de cualquier elemento F inmediatamente después del elemento del tipo E
E[foo]	Elementos del tipo E con el atributo foo
E[foo=”ejemplo”]	Elementos del tipo E con el atributo foo igual a “ejemplo”
E[foo~=”ejemplo”]	Elementos del tipo E con el atributo foo contenga “ejemplo”. Se pueden añadir varias palabras separadas por espacios. (~ =ALT + 0126)
E[lang|=”es”]	Similar al anterior, pero se referirá a todos los elementos E tal que su atributo lang comience por “es”. Por ejemplo: “es_ES”, “es_CA”,…
E[foo$=”ejemplo”]	Elementos del tipo E en el que el atributo foo termine con “ejemplo”.
DIV.ejemplo	Todos los elementos DIV que sean de la clase ejemplo
E#miID	El elemento E en el que su ID sea igual miID

Tabla 4: Selectores en CSS.

Se denominan hojas en “cascada” ya que los elementos hijos heredan por defecto los valores de estilo de los padres. Las propiedades de un elemento se asignan por valores específicos, heredados o por defecto (de mayor a menor preferencia, respectivamente). Selectores más específicos sobrescriben a los más generales. Las reglas se aplican por especificidad, no por el orden en que aparezcan en el documento, aunque si se repiten, se elige la última aparición.

CSS define cómo se visualizan los elementos

- **Modelo de cajas**: cada elemento tiene asociado una caja (el elemento incluye las marcas y lo que contienen) rectangular.
- Se pueden definir propiedades para cada una de las áreas.

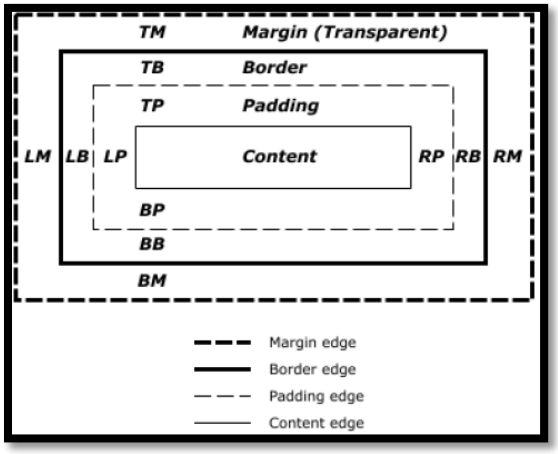


Ilustración 23: Modelo de cajas de CSS.

CSS especifica cómo se posicionan los elementos en pantalla:

Cada elemento del árbol genera cero o más cajas de acuerdo al modelo de cajas. El posicionamiento depende de las dimensiones de la caja, relaciones entre elementos y el esquema de posicionamiento.

El esquema de posicionamiento incluye tres modos:

- **Flujo normal**, es el usado por defecto, posiciona las cajas consecutivamente verticales (block) u horizontalmente (inline).
- **Flotante** (float): una caja se pone en su posición normal y luego se desplaza al máximo a la derecha o izquierda. El resto del contenido fluye alrededor de ella.
- **Absoluto**: puede ser desplazado o fijo. La caja se saca del flujo y se sitúa en una posición, pero el resto del contenido no fluye, es decir, la caja absoluta puede superponerse a otras cajas.

Hay 3 formas básicas de aplicar las hojas de estilo en cascada en un sitio web:

- **Hojas de estilos en cascada externa:** Una hoja de estilos en cascada externa es ideal cuando se aplica el estilo a muchas páginas. Con una hoja de estilos en cascada externa se puede cambiar el aspecto de un sitio web completo con solo modificar un archivo. Cada página debe enlazar a la hoja de estilo con la etiqueta:

```
<link href="css/style.css" rel="stylesheet">
```

Ilustración 24: Link dentro de head que enlaza a una hoja de estilo externa.

Una hoja de estilos en cascada externa puede ser escrita en cualquier editor de texto y el contenido no debe tener etiquetas HTML. La hoja de estilos en cascada debe ser guardada con una extensión .css. Un ejemplo de un archivo de hoja de estilos en cascada se muestra a continuación:

```
.alerta{
    position: relative;
    margin: 5px;

}

a:link {
    text-decoration: none;
}
```

Ilustración 25: Ejemplo de css de una hoja de estilo en cascada.

Muy importante a tener en cuenta es la ruta del archivo que se va a guardar. En este caso si la ruta se llama en la cabecera href=”css/style.css” esto le va a indicar al navegador que la hoja de estilos en cascada se encuentra en un directorio llamado /css en nuestro servidor. También es posible guardarla dentro de la raíz del servidor por lo que la ruta debe de cambiar quedando como href=”style.css” y de esta manera sea posible para el navegador interpretarla y aplicarla en el sitio web.

- **Hojas de estilos en cascada interna:** Una hoja de estilo en cascada interna debe ser usada cuando un único documento tiene un estilo único. Para definir estilos internos en la sección head de una página HTML se escribirá el código mediante las etiquetas <style></style>.

```
<head>
    <style>
        .alerta{
            position: relative;
            margin: 5px;
        }

        a:link {
            text-decoration: none;
        }

    </style>
</head>
```

Ilustración 26: Ejemplo de uso de una hoja de estilo en cascada interna.

- **Hojas de estilo en cascada en línea:** Un estilo en línea pierde muchas de las ventajas de las hojas de estilo por la mezcla de contenido con la presentación. Se recomienda tener mucho cuidado al utilizar este método.

  Para utilizar este método de estilos en línea se debe aplicar directamente en la etiqueta correspondiente. El atributo "style" puede contener cualquier propiedad CSS.

```
<h4 style=" text-align: center;">Puntos de recogida de ropa</h4>
```

Ilustración 27: Ejemplo de uso de hoja de estilo en cascada en línea.

- **Hojas de estilo en cascada múltiples:** Además de estos tres métodos individuales se puede utilizar el CSS combinando los métodos según se necesite. Ejemplo:

  Se tiene una página interna del sitio que se llama ayuda.html. En esta página se quiere personalizar la etiqueta h2 para mostrar subtítulos adaptados al contenido. En la plantilla, la hoja de estilo en cascada está declarada para todas las paginas internas de la siguiente manera:

```
<head>
    <link rel="stylesheet" type="text/css" href="estilos.css">
</head>
```

Ilustración 28: Enlace a hoja de estilo externa.

La etiqueta h2, en esta hoja de estilos en cascada, tiene las siguientes propiedades:

```
h2 {color:#000; font-weight: bold; font-size: 18px;}
```

Ilustración 29: Ejemplo de estilo.

Este formato se aplicará a la etiqueta h2 en todo el sitio web y la etiqueta h2 será de color negro, en negritas y con un tamaño de fuente de 18 píxeles.

Ahora, se quiere personalizar esta etiqueta únicamente para la página interna del ejemplo, ayuda.html. Para esto se puede usar el método de hojas de estilos en cascada interna:

```
<head>
    <link rel="stylesheet" type="text/css" href="estilos.css">
    <style>
        h2 {color:#06F; font-size: 24px;}
    </style>
</head>
```

Ilustración 30: Ejemplo uso de hoja de estilo en cascada interna.

Este formato se aplicará a la etiqueta h2 únicamente en la página interna ayuda.html y la etiqueta h2 ahora será de color azul, seguirá estando en negritas, pero con un tamaño de fuente de 24 píxeles.

Limitaciones

- Los selectores no pueden usarse en orden ascendente (hacia padres u otros ancestros). La razón que se ha usado para justificar esta carencia por parte de la W3C, es para proteger el rendimiento del navegador, que, de otra manera, podría verse comprometido.
- Dificultad para el alineamiento vertical.
- Ausencia de expresiones de cálculo numérico para especificar valores, por ejemplo, margin-left:

```
margin-left: 10% - 3em + 4px;
```

Ilustración 31: Ejemplo de margin-left.

- Las pseudo-clases dinámicas (como :hover) no se pueden controlar o deshabilitar desde el navegador, lo que las hace susceptibles de abuso por parte de los diseñadores en banners, o ventana emergentes.

Ventajas

Algunas ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Separación del contenido de la presentación.
- Optimización del ancho de banda de la conexión, pues pueden definirse los mismos estilos para muchos elementos con un sólo selector; o porque un mismo archivo CSS puede servir para una multitud de documentos.
- Mejora en la accesibilidad del documento.

Javascript

Javascript es un lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario y la realización de automatismos sencillos. En ese contexto se podría decir que nació como un "lenguaje de scripting" del lado del cliente, sin embargo, hoy Javascript es mucho más. Las necesidades de las aplicaciones web modernas y el HTML5 ha provocado que el uso de Javascript que se encuentra hoy haya llegado a unos niveles de complejidad y prestaciones tan grandes como otros lenguajes de primer nivel.

Pero, además, en los últimos años Javascript se está convirtiendo también en el lenguaje "integrador". Se encuentra en muchos ámbitos, ya no solo en Internet y la Web, también es nativo en sistemas operativos para ordenadores y dispositivos, del lado del servidor y del cliente. Aquella visión de Javascript "utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web" se ha quedado muy pequeña.

En el contexto de un sitio web, con Javascript se puede hacer todo tipo de acciones e interacción. Antes se utilizaba para validar formularios, mostrar cajas de diálogo y poco más. Hoy es el motor de las aplicaciones más conocidas en el ámbito de Internet: Google, Facebook, Twitter, Outlook... absolutamente todas las aplicaciones que se disfrutan en el día a día en la Web tienen su núcleo realizado en toneladas de Javascript. La Web 2.0 se basa en el uso de Javascript para implementar aplicaciones enriquecidas que son capaces de realizar todo tipo de efectos, interfaces de usuario y comunicación asíncrona con el servidor por medio de Ajax.

A Javascript se le denomina "del lado del cliente" porque donde se ejecuta es en el navegador (cliente web), en contraposición a lenguajes como PHP que se ejecutan del "lado del servidor". En el lado que se ocupa con Javascript, el cliente, es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con todos los navegadores modernos se ha convertido en un estándar como lenguaje de programación del lado del cliente.

Con Javascript se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, con que cuenta este lenguaje es el propio navegador y todos los elementos que hay dentro de una página (que no es poco). Pero ahora, gracias a las API Javascript del HTML5, que están disponibles en los navegadores actuales de ordenadores y dispositivos, se puede acceder a todo tipo de recursos adicionales, como la cámara, espacio para almacenamiento de datos, creación de gráficos basados en vectores y mapas de bits, flujos de datos con servidores, etc.

Con todo ello se han multiplicado las posibilidades.

Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, Javascript pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

Con Javascript el programador es capaz de alterar cualquier cosa que se muestra en una página, cambiando, insertando o eliminando todo tipo de contenido. Si lo desea, puede controlar cada cosa que ocurre en la página cuando la está visualizando el usuario y comunicar con él con todo tipo de interfaces especiales.

El código a ejecutar por el cliente (script) se incluye junto con el documento, mediante el elemento de HTML <script>. Se puede incluir de dos formas:

- **Interno**: Este elemento incluye el propio código:
  <script> document.write("Hola Mundo"); </script>
- **Externo:** incluye una URL (absoluta o relativa) con el código a ejecutar: <script src="miscript.js“ type= "text/javascript" ></script>

La ejecución del código se produce de tres formas posibles, en función de la presencia de los siguientes atributos:

- **Ejecución diferida**: se ejecuta al terminar de cargar el documento. Ejemplo: <script src="demo_defer.js" defer></script>
- **Ejecución asíncrona** (nuevo en HTML 5): se ejecuta inmediatamente, en paralelo con la carga del resto del documento. Ejemplo:
  <script src="demo_defer.js" async></script>
- **Ejecución normal**: si no aparece ninguno de los atributos anteriores, se ejecuta el código inmediatamente, antes de continuar con la carga y análisis del resto del documento (parse).

En muchos casos el código consiste en funciones que se ejecutarán cuando sean invocadas, normalmente al ocurrir ciertos eventos.

Ejecución basada en eventos: el código javascript se ejecuta al ocurrir un evento.

- Se asocia una función javascript (previamente declarada) a un evento, que es invocada cuando ocurre.

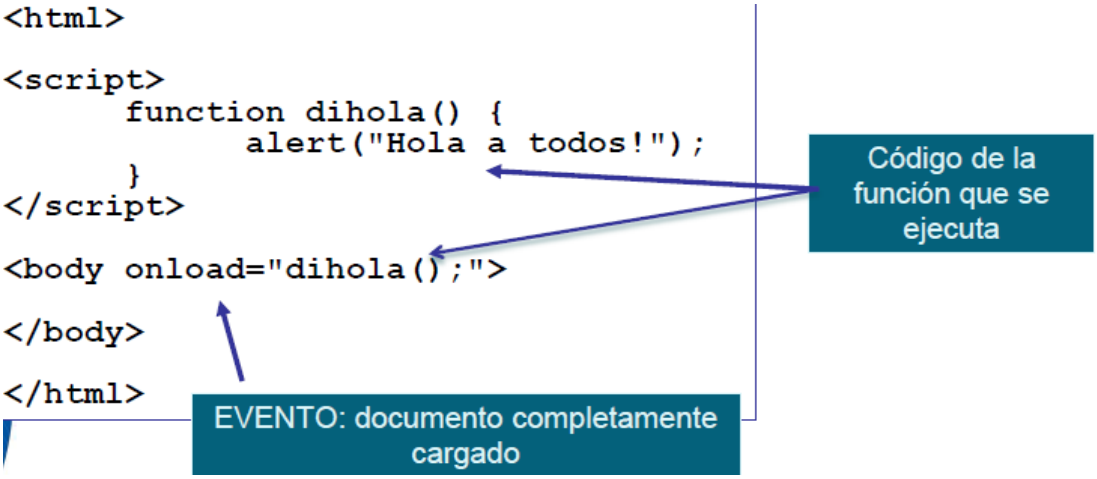


Ilustración 32: Ejemplo de uso del evento onload de Javascript.

Un **evento** es una acción o suceso detectado por el sistema que puede ser procesado.

La asignación de código a eventos se puede hacer de dos formas:

- **Atributos de evento HTML**: son atributos de ciertos elementos HTML que permiten asignar la ejecución de una función cuando ocurren.
  <h1 onclick="changetext(this)">Clic en el texto!</h1>
- **Asignación mediante código:** usando la interfaz DOM mediante javascript.
  <script>

  document.getElementById("miboton").onclick=validarFormulario;

  </script>

  En este caso se tiene que tener en cuenta que el elemento debe existir previamente, o la llamada a getElementById() devolverá null. Es decir, el documento HTML debe haberse procesado (parse) previamente y ese elemento existir.

Ionic

Ionic [1] es una capa que trabaja por encima de Cordova y que nos permite crear apps muy vistosas gracias a Angular.js. El framework está construido con Angular y SASS básicamente, SASS es un preprocesador CSS, que permite trabajar con elementos CSS de una forma muy cómoda. Así, Ionic provee de una serie de componentes con los que crear apps que nada tienen que envidiar a las aplicaciones nativas. Y lo más interesante, es que todo lo que se debe saber es HTML, CSS y Angular.

Las aplicaciones son hibridas, ¿Qué quiere decir eso? Que se puede desarrollar una misma aplicación que corra en Android, iOS, Windows Phone... sin tener que desarrollarla 2 o 3 veces como habría que hacer si se tuviera que hacer con el correspondiente lenguaje nativo de cada plataforma.

Ventajas de Ionic:

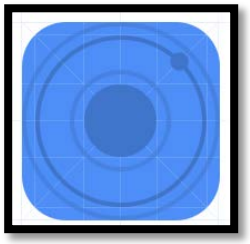
Alto Rendimiento

La velocidad es importante. Tan importante que sólo se nota cuando no está en la app. Ionic está construido para ser rápido gracias a la mínima manipulación del DOM, con cero jQuery y con aceleraciones de transiciones por hardware.

AngularJS & Ionic

Ionic utiliza AngularJS con el fin de crear un marco más adecuado para desarrollar aplicaciones ricas y robustas. Ionic no sólo se ve bien, sino que su arquitectura central es robusta y seria para el desarrollo de aplicaciones. Trabaja perfectamente con AngularJS.

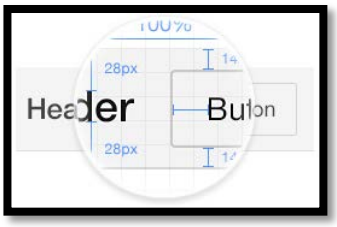
Centro nativo



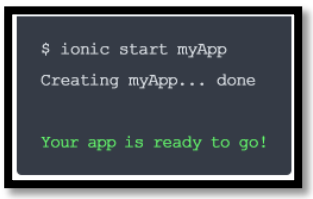
Ionic se inspira en las SDK de desarrollo móviles nativos más populares, por lo que tiene todas las ventajas de una aplicación nativa como la potencia y la flexibilidad. Ionic se ejecuta dentro de Cordova o Phonegap para desplegar de forma nativa, o como una aplicación web progresiva, Lo interesante, es que se desarrolla una vez, y se compila para varios.

Buen diseño

Limpio, simple y funcional. Ionic ha sido diseñado para trabajar y mostrarse adecuadamente en todos los dispositivos móviles. Con muchos componentes usados en móviles, tipografía, elementos interactivos, etc., que se adaptan a cada plataforma.



Un potente CLI



Utiliza un solo comando para crear, construir, probar y desplegar una aplicación creada con Ionic en cualquier plataforma.

Fácil de aprender

Todo lo que se necesita saber son HTML, CSS y JavaScript: los bloques de construcción de la web. Además, a medida que se va trabajando se va aprendiendo AngularJS.

Desarrollado por los nerds de la web

Desarrollado y mantenido por desarrolladores y diseñadores apasionados por las tecnologías web.

Empezar a desarrollar con Ionic

Para empezar a desarrollar con Ionic, se tienen que cumplir unos requerimientos básicos para desarrollar una app. Con Ionic se tiene la libertad de desarrollar bajo cualquier sistema operativo.

En primer lugar, se tiene que instalar node.js desde su página oficial. Node.js es un entorno de programación que permite hacer aplicaciones javascript que se ejecutan desde el servidor.

En segundo lugar, se tiene que instalar la última versión de Apache Cordova [4], que es el responsable de envolver la aplicación (hecha en html, css, javascript) en un contenedor nativo, y las CLI (Command-line tools) de Ionic. Para ello se ejecuta lo siguiente en la línea de comandos:

```
c:\hidrogea>npm install -g cordova ionic
```

Ilustración 33: Comando para instalar Cordova e Ionic.

Este proyecto, en primer lugar, estaba pensado para desarrollar una app en Android y después en iOs, por lo que, además de Cordova y el CLI de Ionic, se necesitan tener instalados los siguientes programas para programar en Android:

- **Apache Ant**: Apache Ant se descarga desde su web oficial, se descarga el archivo .zip y se descomprime en c:/ant. Una vez hecho esto, se tiene que configurar la variable de entorno del sistema para definir la ruta a este programa.
- **Java JDK**: El archivo de instalación de Java JDK se descarga desde su web oficial. Una vez instalado, se tiene que configurar la variable de entorno JAVA_HOME y agregar el directorio bin del JDK al path.
- **Android SDK**: El SDK te provee de una API de librerías, y herramientas de desarrollo para construir, testear y depurar aplicaciones android. Cordova requiere que se establezca la variable de entorno ANDROID_HOME, y, además, se debe incluir en la variable PATH el directorio tools y platform-tools.

El siguiente paso es crear el proyecto, para ello en el directorio que se elija, se tiene que ejecutar el siguiente comando:

```
c:\>ionic start hidrogea blank
```

Ilustración 34: Comando para crear un proyecto con Ionic.

Este comando creará un directorio llamado hidrogea en c:/. Con ese comando, Ionic creará una estructura similar a esta:

```
├── bower.json      // dependencias de bower
├── config.xml      // configuración de cordova
├── gulpfile.js     // tareas de gulp
├── hooks
├── ionic.project   // configuración de ionic
├── package.json    // dependencias de nodejs
├── platforms       // compilaciones de iOS/Android estarán en este dir
├── plugins         // Plugins de cordova y ionic
├── scss            // código scss, que compilara a css y lo guardara en www/css/
└── www             // la aplicación propiamente dicha(codigo javascript, css, html, imagenes, etc)
```

Ilustración 35: Estructura de un directorio de Ionic.

El siguiente paso es decirle a Ionic que se quiere desarrollar una app para Android, para ello se ejecuta el siguiente comando:

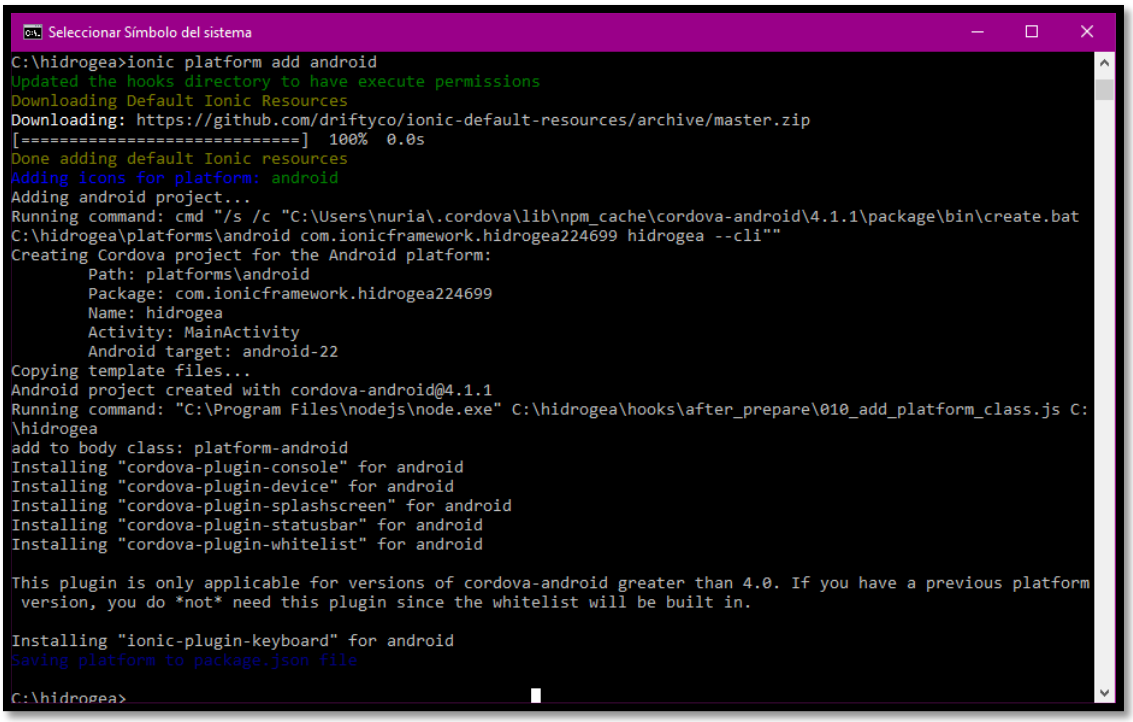


Ilustración 36: Comando para agregar la plataforma.

Una vez que se ha instalado todo, se puede probar la app en un navegador, ejecutando el siguiente comando:

```
c:\hidrogea>ionic serve
```

Ilustración 37: Comando para probar la app en el navegador.

Con este comando, Ionic creará un servidor http y se abrirá una página en el navegador en la dirección http://localhost/8100. La idea es desarrollar utilizando el navegador y después probar en un móvil o un emulador.

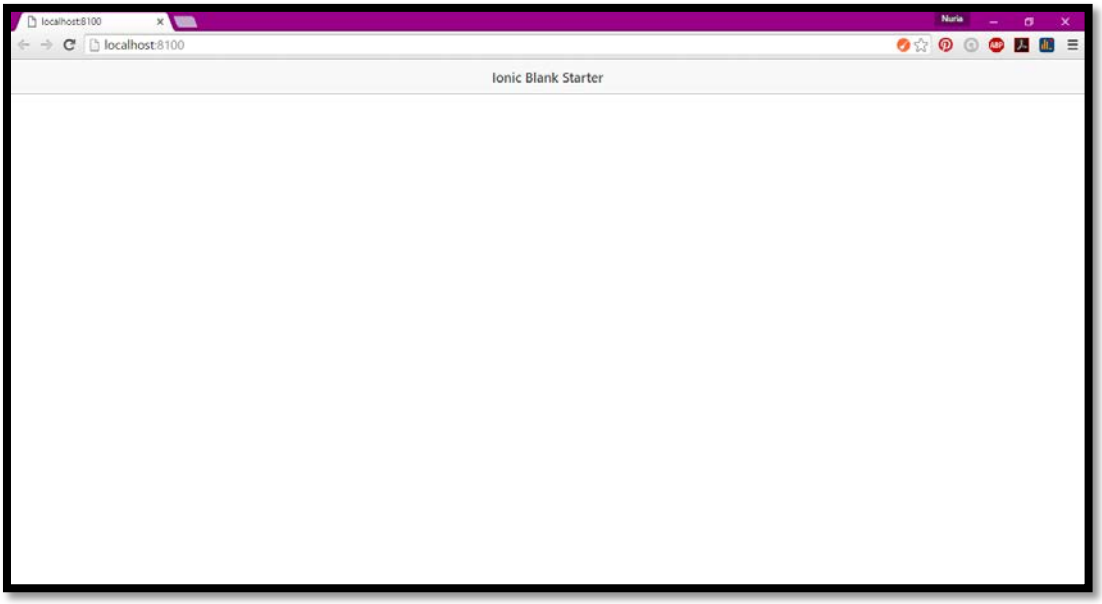


Ilustración 38: Navegador tras ejecutar el comando ionic serve.

Este comando de Ionic tiene la ventaja de que cada vez que cambiemos algo en el código de la aplicación, automáticamente se mostrara en el navegador.

Además de este comando, Ionic dispone de otro comando para compilar la aplicación y crear el archivo apk para poder probarla en un dispositivo móvil. Ejecutando el siguiente comando:

```
c:\hidrogea>ionic build android
```

Ilustración 39: Comando para crear el apk.

Ionic creará el archivo apk de la aplicación que se esté desarrollando.

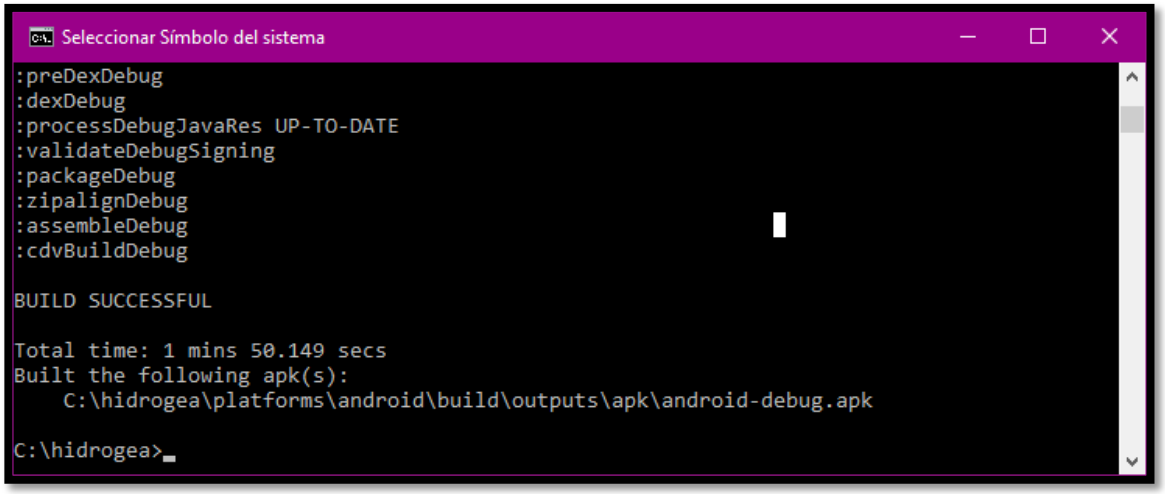


Ilustración 40: Resultado de ejecutar el comando para crear el apk.

Estructura de directorios de Ionic

La estructura que ha creado Ionic es la siguiente:

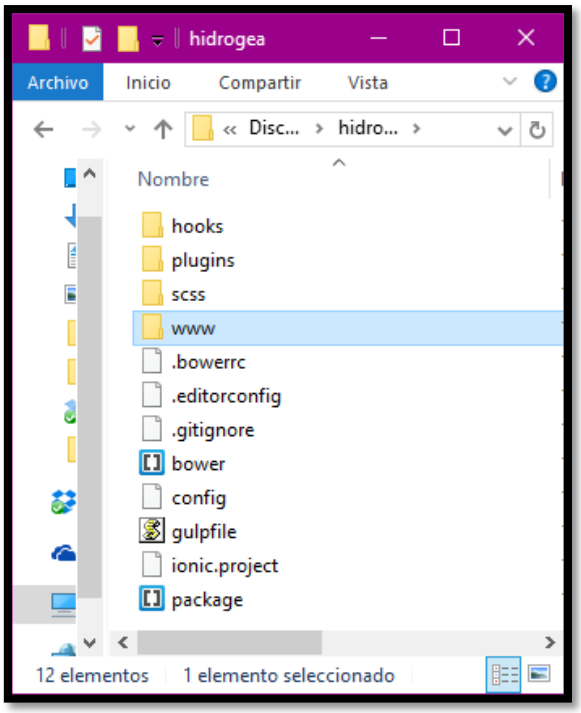


Ilustración 41: Estructura de Ionic.

De aquí lo importante se va a tener el directorio www, que es donde se va a desarrollar el código de la App.

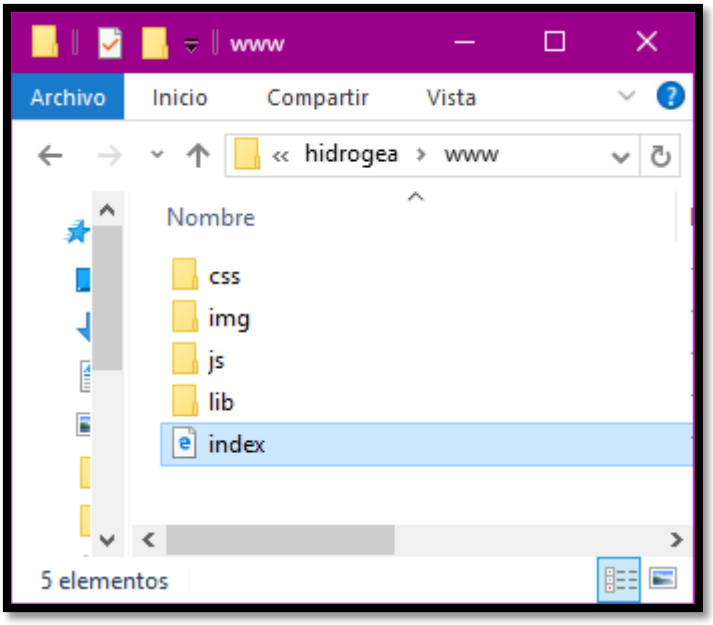


Ilustración 42: Estructura del directorio www.

Como se puede observar este directorio es como el de una aplicación web normal, excepto por tres salvedades:

- Se puede acceder a APIs de dispositivo móvil gracias a Cordova.
- Es ya una aplicación Angular básica con algunos controladores y servicios.
- Y, por último, se pueden usar los componentes de Ionic en forma de directivas de AngularJS.

Por lo demás es lo mismo que se puede encontrar en otras webs, el index.html, carpetas para el CSS y JS, etc.

Ionic Push

Uno de los requisitos de este proyecto es que permitiera el envío de notificaciones nativas, esta fue una de las razones por la que se decidió usar Ionic.

Ionic Push [2] es una parte esencial Ionic, permite enviar notificaciones de dos maneras distintas, a través de un sencillo panel de control, que enviará las notificaciones automáticamente cuando los usuarios se ajusten a unos criterios especificados o a través de una API que permite enviar notificaciones desde un servidor.

El concepto de Push es bastante sencillo, permitir a los usuarios de la aplicación saber algo en un determinado momento. Esto varía mucho de una aplicación a otra, pero el concepto básico sigue siendo el mismo: entregar una notificación al dispositivo del usuario.

Android e iOS disponen de GCM (Google Cloud Messaging) y APNs (Apple Push Notification Service), respectivamente. Cada uno de estos sistemas requieren muchas variables de instalación y configuración.

Ionic Push actúa como un intermediario entre los APN, GCM y la aplicación.

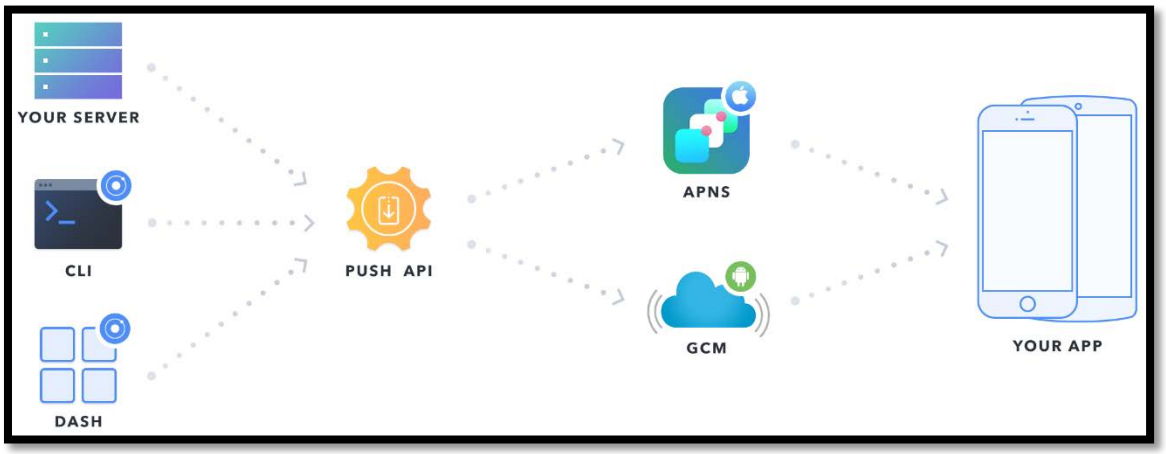


Ilustración 43: Esquema de Ionic Push.

El diagrama representa como se le entrega una notificación a un dispositivo de un usuario. Para ello se necesita una cuenta de Google o Apple o ambos, dependiendo del dispositivo al que se le quiera enviar las notificaciones.

Con una configuración de la cuenta, se tendrá acceso a generar un certificado push, el perfil del móvil o el acceso a un proyecto de mensajería de Google en la nube, donde se puede obtener una clave de la API y el número del proyecto para los dispositivos Android.

Para que se pueda comunicar con el APN o GCM, se va a necesitar almacenar los detalles de la etapa anterior, estos detalles se almacenan en un perfil de seguridad.

Paso 1. Cuenta

Se tendrá que configurar una cuenta de desarrollado de Apple si se va a enviar a los dispositivos iOS o una cuenta de Google si se va a enviar a los dispositivos de Android.

En este caso, se ha desarrollado una aplicación para Android, por lo que a continuación se explicarán los pasos para el caso de Android.

Paso 2. Perfil de Seguridad

Para enviar notificaciones a los dispositivos Android se necesita crear un proyecto en la consola de desarrollo de Google y permitir la API de mensajería en la nube (GCM). Esto dará acceso a una clave de API que permite la comunicación con el GCM, el número del proyecto se

utiliza para asociar el dispositivo a ese proyecto.

Para crear una API en Google, se tiene que acceder a la consola de desarrolladores de Google, a continuación, pulsar sobre “Create Project” e introducir el nombre del proyecto y el ID. Por último, pulsar sobre “Create”.

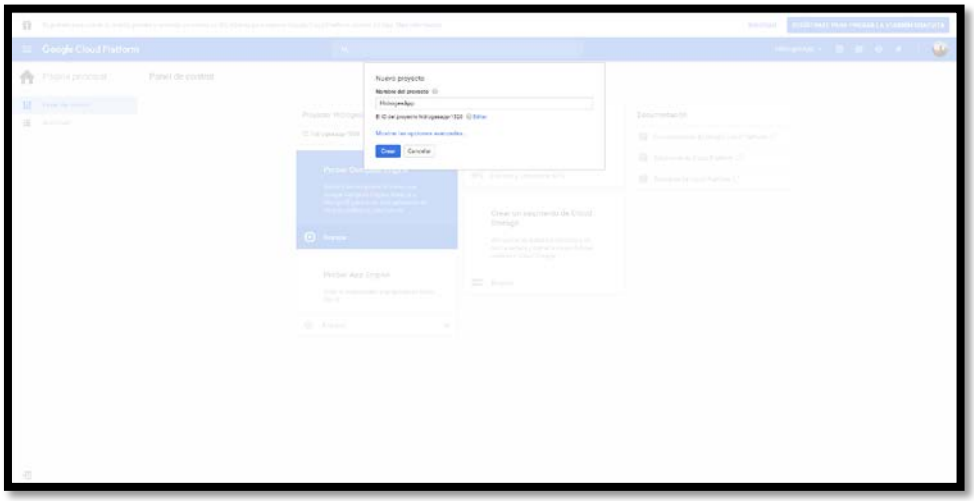


Ilustración 44: Creación de una API de Google.

Una vez creado, solo quedará activar “Google Cloud Messaging for Android” de la lista de las APIs disponibles.

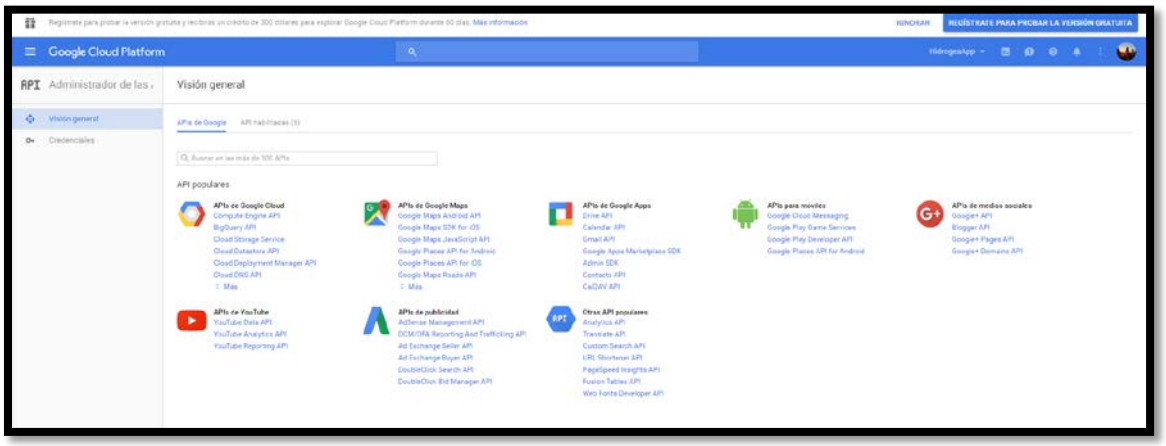


Ilustración 45: Gestor de APIs de Google.

En la pestaña de “Credentials” se encontrarán las claves necesarias para la configuración de las notificaciones Push de la App.

Mediante el siguiente comando se podrá establecer el número del proyecto de la aplicación.

```
c:\hidrogea>ionic config set gcm_key <your-gcm-project-number>
```

Ilustración 46: Comando para establecer el número gcm.

Una vez establecidas las claves, desde la plataforma de Ionic se podrán realizar notificaciones push.

AngularJS

AngularJS [6] es Javascript. Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en este caso sobre el lenguaje Javascript con programación del lado del cliente.

Este Javascript pretende que los programadores mejoren el HTML que hacen. Que puedan producir un HTML que, de manera declarativa, genere aplicaciones que sean fáciles de entender incluso para alguien que no tiene conocimientos profundos de informática. El objetivo es producir un HTML altamente semántico, es decir, que cuando se lea se entienda de manera clara qué es lo que hace o para qué sirve cada cosa. Lógicamente, AngularJS viene cargado con todas las herramientas que los creadores ofrecen para que los desarrolladores sean capaces de crear ese HTML enriquecido. La palabra clave que permite ese HTML declarativo en AngularJS es "directiva", que no es otra cosa que código Javascript que mejora el HTML.

Patrondes de diseño

Angular promueve y usa patrones de diseño de software. En concreto implementa lo que se llama MVC, aunque en una variante muy extendida en el mundo de Javascript que luego se comentará con más detalle. Básicamente estos patrones nos marcan la separación del código de los programas dependiendo de su responsabilidad. Eso permite repartir la lógica de la aplicación por capas, lo que resulta muy adecuado para aplicaciones de negocio y para las aplicaciones SPA (Single Page Application).

Nota: *Las SPA o "Aplicaciones de una sola página", son sitios web donde los usuarios perciben una experiencia similar a la que se tiene con las aplicaciones de escritorio. En este tipo de sitios la página no se recarga, no existe una navegación de una página a otra totalmente diferente, sino que se van intercambiando las "vistas". Técnicamente podríamos decir que, al interactuar con el sitio, el navegador no recarga todo el contenido, sino únicamente vistas dentro de la misma página.*

MVC

Ahora se va a hacer un breve recorrido para nombrar y describir con unos pequeños apuntes aquellos elementos y conceptos que se encuentran dentro de AngularJS.

- **Vistas:** Será el HTML y todo lo que represente datos o información.
- **Controladores**: Se encargarán de la lógica de la aplicación y sobre todo de las llamadas "Factorías" y "Servicios" para mover datos contra servidores o memoria local en HTML5.
- **Modelo de la vista**: En Angular el "Modelo" es algo más de aquello que se entiende habitualmente cuando se habla del MVC tradicional, es decir, las vistas son algo más que el modelo de datos. En modo de ejemplo, en aplicaciones de negocio donde se tiene que manejar la contabilidad de una empresa, el modelo serían los movimientos contables. Pero en una pantalla concreta de la aplicación es posible que se tenga que ver otras cosas, además del movimiento contable, como el nombre de los usuarios, los permisos que tiene, si pueden ver los datos, editarlos, etc. Toda esa información, que es útil para el programador pero que no forma parte del modelo del negocio, es a lo que se llama el "Scope" que es el modelo en Angular.

Además del patrón principal descrito, se tienen los módulos:

Módulos: Es la manera que propone AngularJS para que se desarrolle un código ordenado, para evitar el código espagueti, ficheros gigantescos con miles de líneas de código, etc. Se pueden dividir las cosas, evitar el infierno de las variables globales en Javascript, etc. Con los módulos se puede realizar aplicaciones bien hechas, de las que un programador pueda sentirse orgulloso y, sobre todo, que facilite su desarrollo y mantenimiento.

División de AngularJS

Para poder entender AngularJS se debe dividir en dos partes:

- **Parte del HTML**: Es la parte declarativa, con las vistas, directivas y filtros que nos provee AngularJS.
- **Parte de JavaScript puro**: Serán los controladores, factorías y servicios.

Ilustración 47: Enlace en AngularJS.

En AngularJS el denominado "$scope" representa al modelo, no es más que un objeto Javascript el cual se puede extender creando propiedades que pueden ser datos o funciones. Sirve para comunicar desde la parte del HTML a la parte del Javascript y viceversa. Es donde se produce la "magia" en AngularJS, y aunque esto no sea del todo cierto, para que se entienda mejor, se puede decir que AngularJS se va a suscribir a los cambios que ocurran en el scope para actualizar la vista, y al revés, se suscribirá a los cambios que ocurran en la vista y con eso actualizara el scope.

Primeros pasos con AngularJS

AngularJS se puede descargar desde su página oficial. Una vez descargado se debe incluir en la página (index.html) mediante la etiqueta <script>:

```
<script src="lib/angular/angular.js"></script>
```

Ilustración 48: Etiqueta pa insertar AngularJS.

El siguiente paso para poder usar AngularJS es colocar la directiva ng-app en la etiqueta que englobe la aplicación, más adelante se explicarán las directivas, por ahora se puede pensar que es un contenedor HTML donde AngularJS va a desplegar su contenido.

Ilustración 49: Directiva ng-app.

Binding en AngularJS

El binding no es más que enlazar la información que se tiene en el "scope" con lo que se muestra en el HTML. Esto se produce en dos sentidos:

- **One-way binding:** En este caso la información solamente fluye desde el scope hacia la parte visual, es decir, desde el modelo hacia la representación de la información en el HTML. Se consigue con la sintaxis "Mustache" de las dos llaves. {{ dato }}

  Se estaría trayendo ese dato desde el scope y mostrándolo en la página. La información fluye desde el scope hacia la representación quiere decir que, si por lo que sea se actualiza el dato que hay almacenado en el modelo (scope) se actualizará automáticamente en la presentación (página).

- **Two-way binding:** En este segundo caso la información fluye desde el scope hacia la parte visual (igual que en "one-way binding") y también desde la parte visual hacia el scope. Se implementa por medio de la directiva ngModel.
  <input type="text" ng-model="miDato" />
  En este caso cuando el modelo cambie, el dato que está escrito en el campo de texto (o en el elemento de formulario donde se use) se actualizaría automáticamente con el nuevo valor. Además, gracias al doble binding (two-way) en este caso, cuando el usuario cambie el valor del campo de texto el scope se actualizará automáticamente.

Directivas y expresiones en AngularJS

Al trabajar con AngularJS se sigue desarrollando encima del código HTML, pero ahora se tiene otros componentes útiles que agregan valor semántico a la aplicación. De alguna manera se está enriqueciendo el HTML, por medio de lo que se conoce como "directiva".

Las directivas son nuevos "comandos" que se van a incorporar al HTML y se pueden asignar a cualquiera de las etiquetas por medio de atributos. Son como marcas en elementos

del DOM de la página que le indican a AngularJS que tienen que asignarles un comportamiento determinado o incluso transformar ese elemento del DOM o alguno de sus hijos.

Cuando se ejecuta una aplicación que trabaja con Angular, existe un "HTML Compiler" (Compilador HTML) que se encarga de recorrer el documento y localizar las directivas que se hayan colocado dentro del código HTML, para ejecutar aquellos comportamientos asociados a esas directivas.

AngularJS trae una serie de directivas "de fábrica" que sirven para hacer cosas habituales, así como la posibilidad de poder crear directivas propias para enriquecer el framework. Algunas de estas directivas son:

- **Ng-app**: Esta es la marca que indica el elemento raíz de la aplicación. Se coloca como atributo en la etiqueta que se desea que sea la raíz. Es una directiva que auto arranca la aplicación web AngularJS.
- **Ng-model**: Esta directiva informa al compilador HTML de AngularJS que se está declarando una variable del modelo. Se puede usar dentro de campos INPUT, SELECT, TEXTAREA o controles de formulario personalizados.
- **Ng-repeat:** Esta directiva sirve para implementar una repetición (un bucle). Es usada para repetir un grupo de etiquetas una serie de veces.
- **Ng-click:** Se utiliza para especificar un evento click. En ella se pondrá la expresión que se debe de ejecutar cuando se produzca un click sobre el elemento donde se ha colocado ola directiva.

Módulo en AngularJS

Los módulos son una de las piezas fundamentales en el desarrollo con AngularJS y sirven para organizar el código en esta librería. Se puede entender como un contenedor donde sitúas el código de los controladores, directivas, etc.

La incorporación de módulos en AngularJS es motivada por la realización de aplicaciones con mejores prácticas. Son como contenedores aislados, para evitar que el código interactúe con otros scripts Javascript que haya en la aplicación (entre otras cosas dejarán de producirse colisiones de variables, nombres de funciones repetidos en otras partes del código, etc.). Los módulos también permiten que el código sea más fácilmente reutilizable, entre otras ventajas.

Cuando se arranca una aplicación en AngularJS, usando ng-app se pondrá el nombre del módulo que se quiere que se ejecute en la aplicación, como se puede ver en la ilustración 49.

Desde Javascript se crearán los módulos usando el método angular.module() e indicándoles una serie de parámetros.

```
angular.module('miAplicacion', [ ... ], function(...){ ... })
```

Ilustración 50: Creación de un módulo en AngularJS.

La variable “angular” se tiene como variable global cuando se carga AngularJS, dentro tiene un objeto que estará disponible en cualquier parte del código. A continuación, esta “module” que es un método del objeto “angular” y que sirve para crear el modulo. El primer argumento es el nombre del módulo, que corresponde con el nombre de la aplicación. En el segundo parámetro se puede indicar una serie de módulos adicionales, separados por coma, que serán las dependencias.

Esta llamada a angular.module() devuelve un objeto module, que tiene una serie de métodos como config, run, provider, service, factory, directive, controller, value, etc. que son los que sirven para controlar la lógica de presentación y la lógica de negocio.

Controladores en AngularJS

Los controladores permiten que mediante programación se pueda implementar la lógica de la presentación en AngularJS. En ellos se puede mantener el código necesario para inicializar una aplicación, gestionar los eventos, etc. Se puede decir que gestionan el flujo de la parte del cliente, lo que sería programación para implementar la funcionalidad asociada a la presentación.

En líneas generales se puede entender que los controladores sirven para separar ciertas partes del código de una aplicación y evitar que escribamos Javascript en la vista. Es decir, para que el HTML utilizado para la presentación no se mezcle con el Javascript para darle vida.

Un controlador puede ser agregado al DOM mediante la directiva ngController (con el atributo ng-controller en la etiqueta HTML) y a partir de entonces estará disponible en esa etiqueta (y todas sus hijas) una serie de datos. Esos datos son los que se necesitarán en la vista para hacer la parte de presentación y es lo que se asociaría en el MVC con el "modelo". En la terminología de Angular al modelo se le llama "scope".

A los controladores se les pueden inyectar valores o constantes. Como su propio nombre indica, las constantes son las que no van a cambiar a lo largo del uso de la aplicación y los valores son aquellas variables cuyo dato puede cambiar durante la ejecución de una aplicación. También

se podrá inyectar servicios y factorías, componentes muy parecidos entre sí y que se explican. más adelante.

Para poder avanzar un poco más, es necesario explicar el concepto de “scope”. El "scope" es la pieza más importante del motor de AngularJS y es donde están los datos que se tienen que manejar dentro de la parte de presentación.

El scope es un gran contenedor de datos, que transporta y hace visible la información necesaria para implementar la aplicación, desde el controlador a la vista y desde la vista al controlador. En términos de código el scope no es más que un objeto al que se le puede asignar propiedades nuevas, con los datos que se necesite, o incluso con funciones (métodos).

Esos datos y esas funciones están visibles tanto en el Javascript de los controladores como en el HTML de las vistas, sin que se tenga que realizar ningún código adicional, pues Angular ya se encarga de ello automáticamente. Además, cuando surgen cambios en los datos se propagan entre los controladores y las vistas automáticamente. Esto se realiza por el mecanismo llamado "binding”.

Además, es importante saber qué hacer y qué no hacer desde los controladores:

- Los controladores son adecuados para inicializar el estado del scope para que la aplicación tenga los datos necesarios para comenzar a funcionar y pueda presentar información correcta al usuario en la vista.
- Además, es el lugar adecuado para escribir código que añada funcionalidades o comportamientos (métodos, funciones) al scope.

Con el controlador no se debería en ningún caso manipular el DOM de la página, pues los controladores deben de ser agnósticos a cómo está construido el HTML del documento donde van a estar trabajando.

Tampoco son adecuados para formatear la entrada de datos o filtrar la salida, ni intercambiar estados entre distintos controladores. Para hacer todo eso existen dentro de AngularJS diversos componentes especializados.

Enrutado en AngularJS

En esta aplicación se ha necesitado desarrollar varias vistas para los diferentes menús, por lo que cada vista tendrá asociado un controlador. Para realizar esta asociación se hace uso del módulo ngRoute de AngularJS.

```
$routeProvider
    .when("/", {
        controller:"appCtrl",
        controllerAs: "vm",
        templateUrl: "home.html"
    })
    .when("/ayudasocial", {
        controller:"appCtrl",
        controllerAs: "vm",
        templateUrl: "ayudasocial.html"
    })
```

Ilustración 51: Rutas en AngularJS.

Una cosa que llama la atención es que AngularJS está indicado para hacer aplicaciones de una sola página, concepto que se conoce como “Single Page Application”, SPA, y ahora se habla de tener varias vistas. Pues bien, en AngularJS se pueden tener ambas cosas. Una aplicación de una sola página, pero que es capaz de representar URL distintas, simulando lo que sería una navegación a través de la aplicación, pero sin salirse nunca de la página inicial. Esto sirve para varias cosas, entre otras:

- Memorizar rutas profundas dentro de la aplicación. Se puede contar con enlaces que lleven a partes internas (deeplinks), de modo que no se esté obligado a entrar en la aplicación a través de la pantalla inicial.
- Eso facilita también el uso natural del sistema de favoritos (o marcadores) del navegador, así como el historial. Es decir, gracias a las rutas internas, se puede guardar en favoritos un estado determinado de la aplicación. A través del uso del historial del navegador, para ir hacia delante y atrás en las páginas, se puede navegar entre pantallas de la aplicación con los botones del navegador.
- Mantener vistas en archivos independientes, lo que reduce su complejidad y administrar los controladores que van a facilitar el procesamiento dentro de ellas.

```
<a href="#/hidrogea" class="button icon-left ion-chevron-left button-clear button-dark">Atrás</a>
```

Ilustración 52: Enlace a una ruta.

Como se puede observar en la imagen, las rutas se llaman mediante “#/”, esto significa que toda las URLs corresponden con la misma página, pues se usa el carácter “#” que sirve para

hacer las llamadas como enlaces internos dentro del mismo documento HTML.

Cuando se le pida al navegador que acceda a una ruta creada con "#", este no va a recargar la página yéndose a otro documento, lo que hará es buscar el carácter "#" que corresponda y moverá el scroll de la página a ese lugar.

Por ello, se observa que realmente no son paginas distintas, sino que es la misma página. Lo que se hace es simular la navegación por varias URL cuando realmente es la misma, con enlaces internos.

Para instalar ngRoute, se debe de incluir el script del código Javascript del módulo ngRoute.

```
<script src="js/angular-route.js"></script>
```

Ilustración 53: Incluir ngRoute en AngularJS.

El segundo paso sería inyectar la dependencia con ngRoute en el módulo general de nuestra aplicación. Esto se hace en la llamada al método module() con el que se inicia cualquier programa AngularJS, indicando el nombre de las dependencias a inyectar en un array.

```
angular.module("app", ["ngRoute"])
    .config(function($routeProvider){
        //configuración y definición de las rutas
    });
```

Ilustración 54: Configuración de ngRoute.

El sistema de enrutado de AngularJS permite configurar las rutas que se quieran crear en la aplicación de una manera declarativa. Aunque sea un componente bastante complejo internamente, se puede configurar de una manera ciertamente sencilla.

$routeprovider tiene un par de métodos. El primero es when(), que sirve para indicar qué se debe hacer en cada ruta que se desee configurar, y el método otherwise() que sirve para marcar un comportamiento cuando se intente acceder a cualquier otra ruta no declarada.

Esta configuración se debe realizar dentro del método config(), que pertenece al modelo. De hecho, solo se puede inyectar $routeProvider en el método config() de configuración.

Las rutas se configuran por medio del método when() que recibe dos parámetros, por

un lado, la ruta que se está configurando y por otro lado un objeto que tendrá los valores asociados a esa ruta, estos valores son:

- **Controller**: para indicar el controlador.
- **ControllerAs**: para indicar el nombre con el que se conoce el $scope dentro de esa plantilla.
- **TemplateUrl**: para indicar el nombre del archivo, o ruta, donde se encuentra el HTML de la vista que se debe cargar cuando se acceda a la ruta.

Factorías

Las factorías se crearon porque cada vez que se accede a una vista independiente, los controladores inicializan sus valores a cero. Esto se debe a que se está volviendo a cargar el controlador, ejecutándose la función del controlador.

La solución a este problema fueron las factorías o servicios. Pero ese no el único caso donde las factorías son útiles, por ejemplo, algunas de las cosas que las factorías solucionan son:

- Compartir datos entre varios controladores, lo que permite tener aplicaciones de verdad, capaces de memorizar estados entre varias pantallas.
- Compartir datos entre varias vistas distintas.
- Empaquetar operaciones comunes a varios controladores.

Las factorías son como contenedores de código que se pueden usar en los sitios desarrollados con AngularJS. Son un tipo de servicio, "service" en Angular, con el que se puede implementar librerías de funciones o almacenar datos.

Cuando se usan tienen la particularidad de devolver un dato, de cualquier tipo. Lo común es que devuelvan un objeto de Javascript donde se podrá encontrar datos (propiedades) y operaciones (métodos). Con diferencia de los controladores, las factorías tienen la característica de ser instanciados una única vez dentro de las aplicaciones, por lo que no pierden su estado. Por tanto, son un buen candidato para almacenar datos en la aplicación que se quiere usar a lo largo de varios controladores, sin que se inicialicen de nuevo cuando se cambia de vista.

Angular consigue ese comportamiento usando el patrón "Singleton" que básicamente quiere decir que, cada vez que se necesite un objeto de ese tipo, se enviará la misma instancia de ese objeto en lugar de volver a instanciar un ejemplar.

RESTangular

Para el desarrollo de esta aplicación era necesario disponer de un servicio REST para hacer peticiones a un servidor y obtener así los datos que posteriormente se mostrarán en la aplicación.

REST (REpresentational State Transfer) es un servicio que no tiene estado, lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos. Se usa para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes.

RESTangular es un servicio de AngularJS el cual permite hacer peticiones GET/POST/PUT/DELETE/UPDATE simples y fáciles. La configuración de las peticiones es fácil y puede hacerse de muchas maneras con diferentes iniciaciones de la configuración.

Para instalarlo se tiene que ejecutar el siguiente comando:

```
c:\hidrogea>bower install restangular
```

Ilustración 55: Comando para instalar RESTangular.

E incluirlo como dependencia en la aplicación

```
angular.module('hidrogea', ['ionic','ngRoute', 'restangular'])
```

Ilustración 56: Dependencias de la aplicación.

Una vez instalado se pueden hacer peticiones de la siguiente manera:

```
Restangular.all('Hidrogea/incidencias').getList().then(function (result) {
```

Ilustración 57: Petición GET con RESTangular.

De esta manera se le está indicando a la aplicación que realice una petición GET a la ruta indicada.

La respuesta que se obtendrá será un objeto en formato JSON.

JSON

JSON es un acrónimo de JavaScript Object Notation, un formato ligero originalmente concebido para el intercambio de datos en Internet. JSON permite representar objetos, arrays, cadenas, booleanos y números.

La ventaja de usar JSON para la transferencia de información es que puede ser parseada por varios lenguajes y es un formato estandarizado, es decir que cualquier lenguaje puede intercambiar datos con otro mediante JSON.

Por defecto, JSON se guarda sin espacios entre sus valores lo cual lo puede hacer un poco más difícil de leer. Esto se hace normalmente para ahorrar ancho de banda al transferir los datos, sin los espacios en blanco adicionales, la cadena JSON será mucho más corta y por tanto habrá menos bytes que transferir.

Sin embargo, JSON no se inmuta con los espacios en blanco o saltos de línea entre las claves y valores, así que se puede hacer uso de ellos para hacerlo un poco más legible. JSON es un formato de transferencia de dato y no un lenguaje.

Se debe tener siempre en cuenta que en el formato JSON las cadenas siempre van en comillas dobles, además, los elementos clave y valor debes estar separadas con dos puntos (:), y las parejas clave-valor por una coma (,).

Ilustración 58: Ejemplo de uso de JSON.

Apache Cordova

En la actualidad, existen diversas plataformas móviles en el mercado, como iOS, Android, Windows Phone, Blackberry, etc. Ante tanta variedad, existen diversas maneras de acometer los desarrollos para todas estas plataformas, básicamente, son las siguientes:

- Crear una versión para cada plataforma, usando sus herramientas nativas.
- Utilizar algún framework intermedio, que funcione en todas ellas y que permita reutilizar la lógica y solo desarrollar la interfaz en cada caso.
- Utilizar HTML+CSS+JavaScript y algún host multiplataforma, como PhoneGap o Apache Cordova, que permita compilar la aplicación para todas las plataformas.

Apache Cordova es un framework de licencia libre que cuenta con muchas Apis de diversos dispositivos móviles para desarrollar aplicaciones dentro de un smartphone.

Una de las grandes peculiaridades de este entorno de trabajo es la posibilidad de desarrollar para iOS, Android y demás sistemas operativos sin la necesidad de programar en sus lenguajes nativos (Java, Objetive-C, etc.)

Adicional a la posibilidad de construir aplicaciones móviles basadas en tecnología Web, Cordova es capaz de crear un puente de interacción vía JavaScript que permite interactuar con las funcionalidades del hardware del dispositivo tales como:

- Acelerómetro: Permite monitorear movimiento, orientación y notar agitaciones del dispositivo.
- Camera: Permite a los usuarios tomar fotos para tu aplicación o acceder a las fotos almacenadas en el dispositivo.
- Capturar: Permite capturar video y audio.
- Compass: Devuelve el punto cardinal al que está orientando el dispositivo.
- Conexión: Permite a la aplicación determinar si existe conexión y que tan fuerte es la señal de la misma.
- Dispositivo: Permite acceder a los metadatos del dispositivo.
- Eventos: Soporta varios eventos relacionados con el dispositivo como por ejemplo los cambios en el nivel de la batería.
- Ficheros: Acceso al sistema de ficheros del dispositivo, incluyendo la lectura y escritura de ficheros y directorios.
- Geolocalización: Permite localizar a los usuarios.
- Globalización: Soporta de números y fechas en el formato del idioma nativo del usuario.

- InAppBrowser: Permite el acceso a páginas vía URL sin la necesidad de salir de la aplicación.
- Media: Soporta el grabado y reproducción de audio.
- Notification: Soporta varias formas de llamar la atención del usuario, incluyendo sonidos, y alertas basadas en vibración.
- SplashScreen: Permite que el inicio de tu aplicación posea una experiencia de inicio particular.
- Storage: Provee una base de datos única por usuario para tu aplicación.

Para poder usar Apache Cordova se necesita tener instalado node.js que se explica en el siguiente apartado. Es necesario instalar node.js porque Cordova viene con una serie de comandos de consola (Cordova CLI), los cuales se instalan median el NPM (Node Package Manager) de Node.js.

Una vez instalado node.js, se puede instalar cordova mediante el siguiente comando:

```
c:\hidrogea>npm intall -g cordova
```

Ilustración 59: Comando para instalar Cordova.

Como se ha explicado en un apartado anterior, para el desarrollo de la aplicación, Cordova se instala junto con ionic, por lo que la estructura de directorios de la aplicación contendrá archivos tanto de Cordova como de ionic, ya que trabajan conjuntamente.

Una vez instalado, en el directorio indicado, se observará la siguiente estructura:

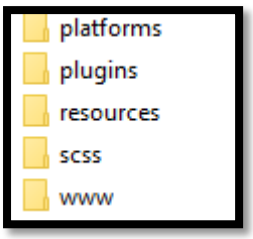


Ilustración 60: Estructura de Cordova.

La carpeta platforms contiene los ficheros necesarios para que Cordova logre la interacción con los diferentes dispositivos con los cuales se quiere poder desplegar la aplicación.

Como se ha comentado en el apartado de Ionic, el siguiente paso sería agregar la plataforma o plataformas sobre las cuales se quiere desplegar la app y, por último, compilarla.

Node.js

NodeJS [5] es una plataforma súper-rápida, especialmente diseñada para realizar operaciones de entrada/salida en redes informáticas por medio de distintos protocolos, apegada a la filosofía UNIX. Es además uno de los actores que ha provocado, junto con HTML5, que Javascript gane gran relevancia en los últimos tiempos, pues ha conseguido llevar al lenguaje a nuevas fronteras como es el trabajo del lado del servidor.

NodeJS se instala desde su web oficial, y una vez instalado ya podremos instalar Cordova e Ionic.

Necesidad de diseño Responsive: Media queries

Una de las necesidades más importantes eran la de tener un diseño responsive, ya que al final lo que se ha desarrollado es una aplicación que se va a ejecutar en distintos dispositivos móviles, los cuales tendrán diferentes tamaños de pantalla.

El diseño responsive es una técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos, desde ordenadores de escritorio a tablets y móviles. Se trata de redimensionar y colocar los elementos de la web de forma que se adapten al ancho de cada dispositivo, permitiendo una correcta visualización y una mejor experiencia del usuario. Se caracteriza porque los layouts (contenidos) e imágenes son fluidos y usan código media-queries de CSS3.

El diseño responsive permite reducir el tiempo de desarrollo, evita los contenidos duplicados, y aumenta la viralidad de los contenidos ya que permite compartirlos de una forma muchos más rápida y natural.

La regla @media permite especificar que cierto conjunto de reglas CSS solo deben aplicarse a cierto tipo de dispositivo. Así las definiciones dentro del bloque de la regla @media screen { ... } solo serían interpretadas por dispositivos conectados a monitores de ordenador y los de la regla @media projection { ... } solo se aplicaría a proyectores.

CSS3 añade importantes y nuevas capacidades que permiten definir conjuntos de estilos dependiendo de propiedades comunes de los dispositivos que acceden a los sitios. Propiedades como el alto y el ancho o la relación de aspecto o el número de colores disponible. Las reglas @media pueden ser utilizadas para adaptar las páginas, no solo para dispositivos comunes, sino para todo tipo de dispositivos.

Hasta ahora, si se necesitaba conocer el tamaño actual de la ventana del navegador, se debía de usar JavaScript para recolectar datos de ese tipo desde el navegador y después darles un uso a esos datos a través de la modificación del DOM a través de métodos programados en JavaScript. Aunque dicho método es válido, no es realmente óptimo ni intuitivo.

CSS3 aporta las media queries que proveen de una forma de conocer bastantes propiedades comunes de los dispositivos.

Aunque media queries dispone de muchas propiedades diferentes, básicamente son:

- Aspect-ratio: Mira las dimensiones relativas del dispositivo expresadas como una relación de aspecto: 16:9 por ejemplo.
- Width y height: Mira las dimensiones del área de visualización. Además, pueden ser expresadas en valores mínimos y máximos.
- Orientation: Mira si el layout es panorámico (el ancho es mayor que el alto) o vertical (el alto es mayor que el ancho). Esto permite ajustar los diseños para dispositivos con propiedades de giro de la pantalla como el iPhone, y otros smartphones y los tablets.
- Resolution: Mira la densidad de los pixeles en el dispositivo de salida. Esto es especialmente útil cuando se quiere aprovechar las ventajas de los dispositivos que tiene una resolución mayor a 72 dpi.
- Color, Color-index y monochrome: Encuentran el número de color o bits por color. Esto permite crear diseños específicos para dispositivos monocromáticos.

Esta aplicación consiste en una pantalla principal con una serie de menús en ella. Se necesitaba que el número de columnas fuese diferente para ordenador, Tablet y móvil, es por esto que se hicieron uso de las media queries.

En la siguiente imagen se puede observar un ejemplo de cómo se pueden usar las media queries, para, en este caso, crear un menú de 3 columnas cuando el dispositivo que se esté utilizando tenga una cierta anchura.

```
/*3 columnas*/
@media (min-width: 684px) and (max-width:900px){

    .cards{
        text-align: center;
        max-width: 30%;
        height: 120px;
        float: left;
        display: flex;
        margin: 0px;
    }
    .h4{
        max-width: 100%;
        width: 500px; position: relative;  top: 30%;
    }
     .p{
        font-size: 16px;
       font-family: 'Roboto', sans-serif, 'Light 300';  color: white;

    }

}
```

Ilustración 61: Ejemplo de uso de media queries.

De esta manera se ha conseguido que la aplicación tenga un diseño responsive y que se adapte a los distintos dispositivos, además, cambiará el número de columnas del menú, según el dispositivo que se está utilizando, si se utiliza un móvil, tendrá dos columnas, si se usa una tablet, serán 3 y si se usa un ordenador, serán 4 columnas.

3.2.3. Lado del servidor

A continuación, se introducen las tecnologías del lado del servidor utilizadas: PHP, el framework PHP Laravel y FTP.

PHP

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de otros lenguajes del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá

el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI (Common Gateway Interface), como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Aunque PHP puede hacer mucho más.

Existen principalmente tres campos principales donde se usan scripts de PHP.

- **Scripts del lado del servidor.** Este es el campo más tradicional y el foco principal. Son necesarias tres cosas para que esto funcione. El analizador de PHP (módulo CGI o servidor), un servidor web y un navegador web. Es necesario ejecutar el servidor con una instalación de PHP conectada. Se puede acceder al resultado del programa de PHP con un navegador, viendo la página de PHP a través del servidor.
- **Scripts desde la línea de comandos.** Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para scripts que se ejecuten con regularidad empleando cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto.
- **Escribir aplicaciones de escritorio.** Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar para escribir dichos programas.

PHP puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix, Microsoft Windows, Mac OS X y probablemente otros más. PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros. PHP funciona tanto como módulo como procesador de CGI.

De modo que, con PHP, se tiene la libertad de elegir el sistema operativo y el servidor

web. Además, se tiene la posibilidad de utilizar programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas.

Con PHP no se está limitado a generar HTML. Entre las capacidades de PHP se incluyen la creación de imágenes, ficheros PDF e incluso películas Flash generadas sobre la marcha. También se puede generar fácilmente cualquier tipo de texto, como XHTML y cualquier otro tipo de fichero XML. PHP puede autogenerar estos ficheros y guardarlos en el sistema de ficheros en vez de imprimirlos en pantalla, creando una caché en el lado del servidor para contenido dinámico.

Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página web con acceso a una base de datos es increíblemente simple utilizando una de las extensiones específicas de bases de datos (p.ej., para mysql).

MySQL

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo, así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQL AB.

El software MySQL tiene una doble licencia. Los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o pueden adquirir una licencia comercial estándar de MySQL AB.

Laravel

Laravel es un framework para aplicaciones web con sintaxis expresiva y elegante. El desarrollo debe ser una experiencia agradable y creativa para que sea verdaderamente enriquecedora. Laravel busca eliminar el sufrimiento del desarrollo facilitando las tareas comunes utilizadas en la mayoría de los proyectos web, como la autenticación, enrutamiento, sesiones y almacenamiento en caché.

Laravel es un framework para el lenguaje de programación PHP. Aunque PHP es conocido por tener una sintaxis poco deseable, es fácil de usar, fácil de desplegar y se le puede encontrar en muchos de los sitios web modernos que usas día a día. Laravel no solo ofrece atajos

útiles, herramientas y componentes para ayudar a conseguir el éxito en los proyectos basados en web, sino que también intenta arreglar alguna de las flaquezas de PHP.

Laravel tiene una sintaxis bonita, semántica y creativa, que le permite destacar entre la gran cantidad de frameworks disponibles para el lenguaje. Hace que PHP sea un placer, sin sacrificar potencia y eficiencia. Es sencillo de entender y permite la modularidad de código lo cual es bueno en la reutilización de código.

Beneficios de Laravel

1. **Incluye un ORM**: A diferencia de CodeIgniter, Laravel incluye un ORM integrado. Por lo cual no se debe instalar absolutamente nada.
2. **Bundles**: existen varios paquetes que extienden a Laravel y dan funcionalidades increíbles.
3. **Se programa de una forma elegante y eficiente**: No más código basura o espagueti que no se entienden, el código estará ordenado de manera de que sea lo más re-utilizable posible.
4. **Se controla la BD desde el código**: Se puede tener un control de versiones de lo que se hace con ella. A esto se llaman migrations, es una excelente herramienta, porque se puede manejar todo desde el IDE, inclusive montar datos en las tablas.
5. **Da soporte a PHP 5.3**.
6. **Rutas elegantes y seguras**: Una misma ruta puede responder de distinto modo a un método GET o POST.
7. **Cuenta con su propio motor de platillas HTML**.
8. **Se actualiza fácilmente desde la línea de comandos**: El framework es actualizable utilizando composer update y listo, nada de descargar un ZIP y estar remplazando.
9. **Cuenta con una comunidad activa que da apoyo rápido.**

Laravel, es uno de los framework en PHP que está teniendo mayor aceptación en los últimos años debido a la simplicidad en la sintaxis, la elegancia en la escritura, la potencia de Composer y Artisan para su manejo y los complementos con los que cuenta, hacen que PHP no se quede en el olvido y vuelva a ser rescatado. Gracias a Laravel se pueden desarrollar proyectos con un lenguaje moderno, rápido, eficiente y profesional.

Artisan es una herramienta de interfaz de línea de comandos, que viene con Laravel. Con Artisan, un desarrollador puede interactuar con su aplicación para desencadenar acciones como la ejecución de las migraciones, la ejecución de pruebas unitarias, y la ejecución de tareas programadas. Artisan también es completamente extensible para que se pueda escribir cualquier tipo de funcionalidad que se desee.

Laravel también proporciona un conjunto de herramientas de última generación para interactuar con las bases de datos. Las **migraciones** son una de las herramientas para interactuar con la base de datos, que permiten diseñar y modificar fácilmente la base de datos de una manera independiente de la plataforma.

Las migraciones pueden ser ejecutadas en cualquiera de los tipos de base de datos que soporta Laravel (MySQL, PostgreSQL, MSSQL, y SQLite) y no se tendrá ningún problema de compatibilidad.

Instalación

Para poder instalar Laravel se necesita tener instalado **Composer**. Laravel utiliza Composer para manejar sus dependencias. Esto quiere decir que Composer va a descargar de sus repositorios todas las librerías y las dependencias y va a manejarlas en un solo lugar de manera ordenada.

Una vez instalado Composer, se puede descargar el instalador de Laravel mediante el siguiente comando:

```
composer global require "laravel/installer=~1.1"
```

Ilustración 62: Comando para descargar el instalador de Laravel.

A continuación, con el comando “laravel new {nombre del proyecto}” se creará una instalación de Laravel 5 en el directorio especificado con todas las dependencias instaladas.

Estructura

La estructura por defecto de Laravel pretende proporcionar un gran punto de inicio para grandes y pequeñas aplicaciones. A demás, permite que se pueda cambiar la estructura como se desee, siempre que se le indiquen las rutas a Composer.

El directorio root de Laravel contendrá los siguientes directorios:

- App: contiene el código del núcleo de la aplicación.
- Bootstrap: contiene unos cuantos archivos del framework Bootstrap.
- Config: contiene todos los archivos de configuración de la aplicación.
- Database: contiene las migraciones y sedes.
- Public: contiene el controlador principal y los assets (imágenes, JavaScript, CSS, etc.).
- Resources: contiene las vistas y los archivos de lenguaje.
- Storage: contiene las plantillas Blade, los archivos de sesiones, caché, y otros archivos generados por el framework.
- Vendor: contiene las dependencias de Composer.

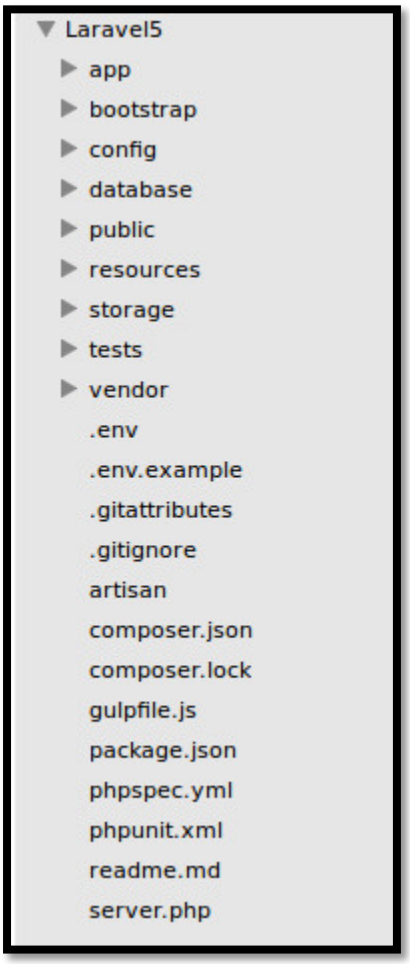


Ilustración 63: Estructura de Laravel 5.

El directorio app viene con una variedad de directorios adicionales, tales como Console, Http, y Providers. Los directorios Console y Http son como proveedores de un API al "core" de la aplicación. El protocolo HTTP y CLI son mecanismos (pasarelas) para interactuar con la aplicación, pero en realidad no contienen la lógica de la aplicación. En otras palabras, son simplemente dos maneras de emitir comandos a la aplicación. El

directorio Console contiene todos los comandos de Artisan, mientras el directorio Http contiene los controladores, filtros y peticiones.

El directorio de commands, por supuesto, alberga los "commands" de la aplicación. Los commands representan tareas que pueden formar parte de una cola de ejecución en la aplicación, también tareas que se pueden ejecutar sincrónicamente con la ejecución de la petición actual.

El directorio Events, como cabe de esperar, almacena clases de eventos. Por supuesto, utilizar clases para representar eventos no es obligatorio. Este es el directorio por defecto donde serán almacenados los que se crean por la línea de comandos Artisan.

El directorio Handlers contiene las clases controladoras para ambos, commands y eventos. Handlers recibe un command o un evento y ejecuta lógica en respuesta a ese command o evento lanzado.

El directorio Services contiene varios servicios "auxiliares" que la aplicación necesita para funcionar. Por ejemplo, el servicio Registrar incluido en Laravel es responsable de validar y crear nuevos usuarios en la aplicación. Otros ejemplos podrían ser servicios que interactúen con APIs externas, sistemas de medición, o incluso servicios que agregan datos desde la propia aplicación.

El directorio Excepcions contiene el controlador de excepciones de la aplicación y también es un buen lugar para meter las excepciones lanzadas por la aplicación.

Enrutado HTTP

Todas las rutas de Laravel están definidas en el archivo app/Http/routes.php, el cual es cargado automáticamente por el framework.

```
Route::get('foo', function () {
    return 'Hello World';
});
```

Ilustración 64: Ejemplo de ruta en Laravel 5.

El archivo routes.php es cargado por el “RouteServiceProvider”, la mayoría de las rutas de la aplicación estarán en este archivo.

Controladores

Los controladores están localizados en la carpeta app/Http/Controllers y se pueden organizar en subcarpetas. Como otras clases de Laravel están dentro del sistema de auto carga de clases, por lo que estarán disponibles siempre que se necesiten en la aplicación.

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

class ArticulosController extends Controller
{
    public function ver($id)
    {
        return view('articulos.ver', ['id' => $id]);
    }
}
```

Ilustración 65: Ejemplo de un controlador en Laravel 5.

En la ilustración anterior se puede observar un ejemplo de un controlador en Laravel. Las dos primeras líneas son concernientes a los espacios de nombres donde se está trabajando. Luego se encuentra la propia clase que define el controlador, en este caso ArticulosController. Como cualquier clase, por convención, se usa la primera letra del nombre en mayúscula y además en Laravel se tiene por costumbre apellidar a las clases con el sufijo "Controller". También es una convención que ayudará a identificar el código de los controladores y a otras personas que puedan trabajar en el proyecto.

Dentro de la clase se colocan las acciones que deseemos. Cada acción se implementa a partir de un método, al que se puede invocar desde el sistema de routing.

A los controladores se les invocan normalmente desde el sistema de rutas, indicando el nombre del controlador y la acción (método) que debe ejecutarse para procesar una solicitud. En la siguiente línea de código se registra una ruta que llamaría a la acción "ver" sobre el controlador "ArticulosController":

```php
Route::get('articulos/{id}','ArticulosController@ver');
```

Ilustración 66: Registro de una ruta.

Crear desde cero un controlador es una tarea repetitiva dentro de Laravel, por lo que existen atajos. El comando "artisan" ofrece una utilidad para crear una nueva clase controlador

de una manera automática.

```
php artisan make:controller CategoriasController
```

Ilustración 67:Comando artisan para crear un controlador.

Modelos

Los modelos son uno de los componentes principales de las aplicaciones desarrolladas bajo el patrón MVC, que tienen la responsabilidad de acceder a los datos, modificarlos, etc. En el patrón además los modelos mantienen lo que se llama la lógica de negocio, que son las reglas que deben cumplirse para trabajar con los datos.

Por tanto, el tipo de acciones que se le va a solicitar a un modelo es, por ejemplo, obtener datos, insertarlos, modificarlos, etc. En las operaciones que modifiquen los datos además se tendrá que realizar cierta validación de esos datos, para asegurarse de que tienen la forma que es necesaria antes de guardarlos.

Los modelos tienen la primera letra en mayúscula, por implementarse mediante clases. Los archivos donde se guarda el código de los modelos también deben tener esa primera letra en mayúscula.

En Laravel los modelos se controlan por un ORM llamado Eloquent, al menos los modelos que están implementados como datos en una base de datos, pero no es un requisito, de modo que se podría trabajar con otros ORM o incluso bajar a un nivel más bajo y trabajar con PDO directamente, o con las extensiones de la base de datos en particular.

En el caso de ser un modelo Eloquent, los modelos están directamente asociados a una entidad y a su vez a una tabla de la base de datos, por lo que un modelo que se llama User está directamente relacionado con una tabla llamada con el mismo nombre en la base de datos, pero en minúscula y acabado en plural, ej. "users".

También existe un comando de artisan para la creación automática de un modelo:

```
php artisan make:model Article
```

Ilustración 68: Comando artisan para la creación de un modelo.

Middleware

Los middlewares en términos generales son partes del software que actúan de mediadores entre distintos actores, permanecen en medio para facilitar la interacción o comunicación entre distintas partes de un sistema. En Laravel se quedan entre medias del sistema de routing y los controladores, permitiendo hacer cosas al pasar de unos a otros, generalmente operaciones de filtrado.

Según la definición de la documentación oficial de Laravel, "HTTP middleware provee un mecanismo adecuado para filtrar solicitudes HTTP entrantes a la aplicación [...] hay diversos middleware incluidos en el framework Laravel, como middleware para mantenimiento, autenticación, protección CSRF mediante token, etc."

Además, se pueden hacer nuevos middlewares para diversas tareas, como añadir una salida en las cabeceras de todas las respuestas, realizar un log de todas las solicitudes al servidor, etc.

Los middleware configurados en el sistema se encuentran en app/Http/Kernel.php. El Kernel es el que le dice a Laravel qué middlewares tiene que cargar.

Para crear un middleware desde la línea de comandos, en la carpeta raíz del proyecto, se puede invocar artisan y solicitarle el comando "make:middleware" indicando a continuación el nombre del middleware que se desea crear.

```
php artisan make:middleware DomingoMiddleware
```

Ilustración 69: Comando artisan para crear un middleware.

Bases de datos en Laravel

En Laravel existen diversas vías para trabajar con bases de datos, a distintos niveles. Principalmente se puede trabajar con:

- Raw SQL: consultas nativas del sistema gestor de bases de datos.
- Fluent Query Builder: un sistema de construcción de las consultas por medio de código, independiente del sistema gestor de bases de datos usado, y sin tener que escribir a mano el código SQL.
- Eloquent ORM: un ORM avanzado, pero sencillo de usar, para trabajar con los datos como si fueran objetos y abstraerse de que realmente estén almacenados en tablas de la base de datos.

El primer paso para trabajar con una base de datos es configurarla en el sistema. Esto se realiza desde un par de lugares.

Archivo .env: Este archivo se encuentra en la carpeta raíz del proyecto y contiene información sobre el entorno. Son básicamente una serie de variables de entorno que tienen entre otras, información sobre la base de datos. Este archivo de entorno podrá en un mismo proyecto tener valores distintos, en distintas máquinas, por ejemplo, cuando está en un servidor en producción y uno en desarrollo.

Archivo config/database.php: Es un archivo donde se definen los valores de conexión de la base de datos. Muchos de ellos los toma directamente de las definiciones del archivo .env (el entorno), pero además se definen asuntos como el driver, codificación, etc. Si se tuvieran varias conexiones con base de datos se deberían definir todas en este archivo.

Los modelos son las clases que tienen el código donde se realizará el acceso a los datos, por tanto, los accesos a las bases de datos son responsabilidad de los modelos. En Laravel se han diseñado para que funcionen con muy poco código y para ello el programador debe respetar una serie de convenciones.

Esas convenciones son las que permiten que se pueda ahorrar código. Estas reglas son:

- El nombre del modelo tiene que ser en singular y mayúscula y la tabla a la que acceden estará en plural y minúscula. Por ejemplo: modelo "User" para tabla "users". Modelo "Product" para tabla "products".
- Por defecto busca siempre la llave primaria con el nombre de campo "id".
- Eloquent usa las columnas created_at y updated_at para actualizar automáticamente esos valores de tiempo

Otra cosa que ahorrará muchas tareas de mantenimiento cuando se trabaja con bases de datos, son las **migraciones**. En Laravel no se definen tablas directamente con SQL o un programa tipo PhpMyAdmin o MySQL Workbench. En lugar de ello se crean unos archivos con código de clases llamados migraciones.

Con las migraciones se pueden definir la creación de tablas, su modificación, etc., llevando un registro de cada uno de los cambios que se han producido en el esquema de la base de datos. Con los seeders se puede incluso crear datos de prueba para poder poblar las tablas. Quizás podría ser más rápido aplicar todos esos cambios con un programa de acceso a las bases de datos, sin embargo, la potencia de las migraciones las hace especialmente útiles en varias situaciones:

- Cualquiera puede actualizar la forma de la base de datos o los datos en sí y compartir los archivos de migraciones con el resto del equipo, de modo que todos se puedan sincronizar y tener la misma versión de la base de datos.
- Cualquier migración se puede volver para atrás, de modo que sirven como una especie de control de versiones de la forma o definición de la base de datos.
- Realiza una abstracción de la base de datos en cuanto a la creación y modificación de su forma. De modo que, si se cambia el motor, las migraciones crearían la misma base de datos en el nuevo sistema gestor.
- Facilita mucho la implantación de un sitio web en un nuevo servidor o en un nuevo componente del equipo de trabajo.

Sobre todo, es clave el tema de la sincronización con todos los componentes de un equipo de trabajo. Si alguien cambia una tabla no necesita pasar el create table nuevo y asegurarse que todos los componentes lo apliquen a mano. Será solo dejar el archivo de las migraciones en el sistema de control de versiones usado para el proyecto (git o el que sea), todos los desarrolladores se sincronizan y ejecutan un comando para procesar las migraciones. Sin embargo, aunque se trabaje solo, las migraciones también ayudarán a ser más productivos.

El asistente de comandos artisan tiene una serie de instrucciones para trabajar con las migraciones.

```
php artisan migrate
```

Ilustración 70: Comando para ejecutar migraciones.

Este comando sirve para ejecutar todo el sistema de migraciones y poner en marcha cada una de las migraciones generadas en el proyecto. Crea el repositorio de migraciones y luego las ejecuta una por una para generar toda la estructura de la base de datos y los datos de prueba que puedan haberse cargado.

FTP

El protocolo FTP (File Transfer Protocol), o Protocolo de transferencia de archivos, es un protocolo de red utilizado para la transferencia de archivos entre los ordenadores conectados a una red TCP. El protocolo FTP se basaba en una arquitectura del tipo Cliente/Servidor.

Ilustración 71: Esquema de FTP.

Básicamente, un **servidor** FTP es un software que se encuentra instalado en un ordenador servidor conectado a Internet, o en el caso de corporaciones, instituciones u otras también puede estar conectada a redes LAN o MAN. El principal propósito de este tipo de software de servidor de FTP es permitir el acceso y el intercambio controlado de archivos contenidos en el PC en que se aloja con otros PC que lo requieren. Algunas de las implementaciones más comunes de servidores FTP en la actualidad son como servidor web para alojar páginas de Internet y como servidor de backup para el respaldo de datos y archivos, entre muchas otras aplicaciones.

El software de **cliente** es el programa que el usuario de un servicio de FTP deberá instalar en su ordenador con el fin de poder acceder al servidor para la carga y descarga de archivos desde y hacia el mismo.

En este caso se ha utilizado Filezilla. Filezilla [3] es un programa de software libre y de código abierto que posee una serie de características y funcionalidades que facilitan la tarea de cargar y eliminar archivos alojados en hostings externos. Además, posee soporte para FTP estándar, SFT y FTP sobre SSL.

3.3.Funcionalidad de la aplicación

La aplicación se podrá descargar desde un enlace. Una vez descargada, cualquier usuario con conexión a internet y con un dispositivo móvil con sistema operativo Android, podrá hacer uso de ella. La aplicación funcionará desde cualquier terminal móvil, por lo que, el diseño de la interfaz se ha diseñado de manera que sea responsive.

Una vez instalada la aplicación, el usuario podrá hacer uso de ella, accediendo al menú principal y navegando por los distintos botones que la aplicación posee.

Como primera versión del menú y botones de la plataforma, se tiene:

- ¿Qué es una Smart city?
- Turismo.
  - *Conoce tu ciudad.
  - *Cruceros
  - *Visita virtual a la ciudad.
  - *Teatro romano.
  - *Ofertas comerciales.
- Portal de transparencia.
- Ayuda social.
  - Cruz Roja.
  - *Cáritas.
  - *Puntos de recogida de ropa.
- *Autobuses.
- *Servicio alumbrado.
- *RSU.
- *Video Streaming.
- Hidrogea.
  - Cuánto llueve ahora.
  - Donde estamos trabajando.
  - *Próximas actuaciones.
  - Calidad del agua.
  - Programa desratización.
  - Datos técnicos.
  - Perfil del contratante.
  - Conoce tu servicio.
  - *Cita previa/contacto.
  - Sugerencias e Incidencias.
  - Hidrogea web oficial.
  - Oficina virtual.

Los datos de la aplicación estarán alojados en la nube, además se podrán realizar notificaciones nativas a los usuarios de la aplicación.

Se enumeran algunas características, que se irán ampliando según necesidades:

- Gestión de datos del servidor.
- Gestión de datos enviados por el usuario a través del formulario.
- Gestión de la BD.

3.3.1. Manual de usuario

En adelante se entenderá “Los usuarios” como cada uno de los clientes que se descarguen e instalen la aplicación en su terminal, y, por tanto, disfruten del servicio.

Cuando un usuario acceda a la aplicación lo primero que verá será una splash screen, en ella habrá dos logos correspondientes a Hidrogea y al ayuntamiento de Cartagena.

Ilustración 72: Splash Screen de la app.

Una vez que el usuario haya accedido a la aplicación, se le mostrará un menú principal, donde se encontrará con un menú clasificado por temáticas.

Ilustración 73: Menú principal de la aplicación.

En este menú el usuario podrá acceder al tema que le interese.

Como se puede observar en la ilustración anterior, algunos de los botones tienen un color gris establecido, esto es debido a que se encuentran sin funcionalidad. Está pensado para que en el futuro se les añada funcionalidad o se modifiquen a otras temáticas según sea considerado oportuno.

Si el usuario pulsa sobre **“¿Qué es una smart city?”**, será redirigido a una página en la que se le mostrará un video explicativo protagonizado por los becarios de la cátedra Hidrogea-UPCT.

Ilustración 74: Página de video ¿Qué es una smart city?

A continuación, si el usuario pulsa sobre **“Turismo”,** podrá ver otra pantalla, con una serie de botones, relacionados con posibles temas que sería interesantes que estuvieran en la aplicación, pero que aún no se han implementado. Como se ha comentado anteriormente, se encuentran en gris porque están inoperativos.

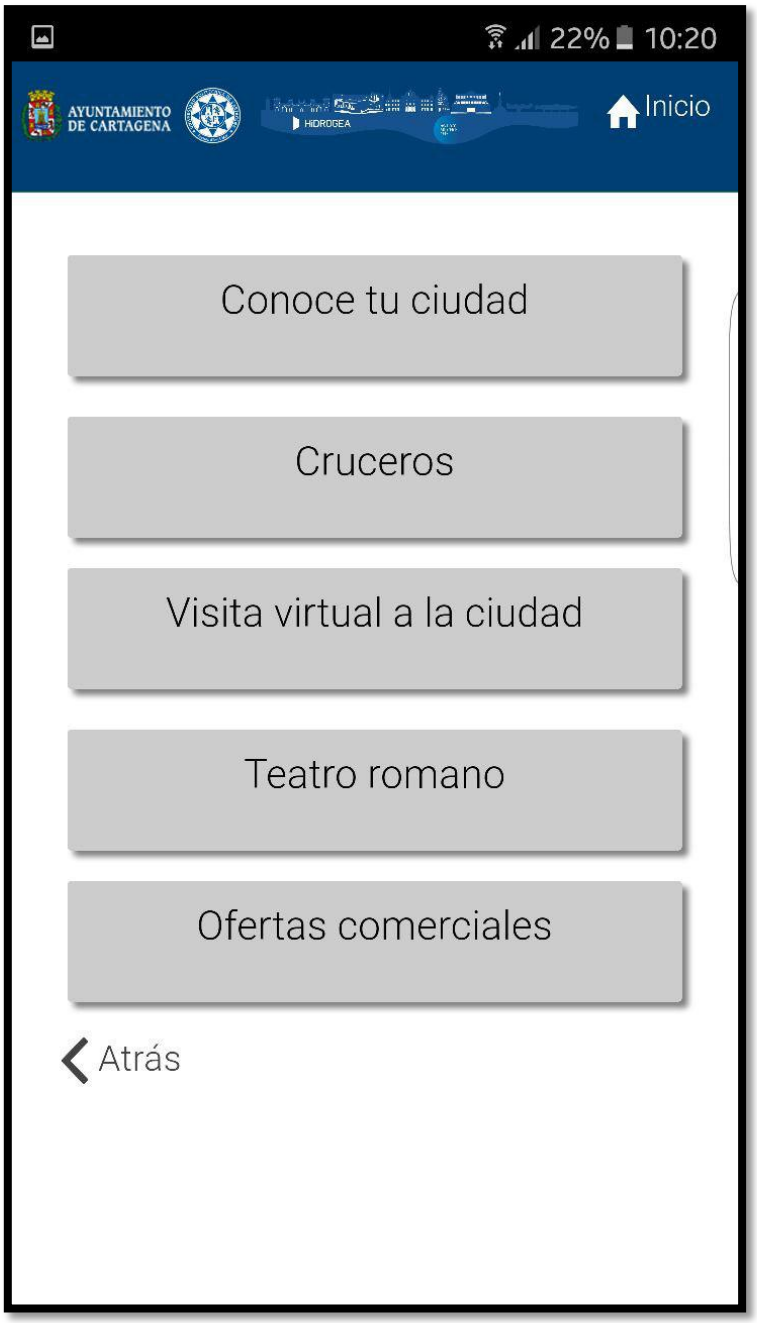


Ilustración 75: Pantalla de Turismo.

Si el usuario pulsa sobre **"Portal de transparencia"**, será redirigido a la página del portal de transparencia del ayuntamiento de Cartagena.

Ilustración 76: Página portal de transparencia.

Por último, en este menú, si el usuario pulsa sobre **"Hidrogea"**, accederá a otra pantalla con un menú con botones relacionados con Hidrogea.

Ilustración 77: Menú de Hidrogea

Este menú consta de los siguientes botones:

- Cuánto llueve ahora
- Donde estamos trabajando
- Próximas actuaciones
- Calidad del agua
- Programa desratización
- Datos técnicos

- Perfil del contratante
- Conoce tu servicio
- Cita previa/contacto
- Sugerencias e Incidencias
- Hidrogea web oficial
- Oficina virtual

Al pulsar sobre **"Cuánto llueve ahora"**, el usuario se encontrará con la siguiente pantalla:

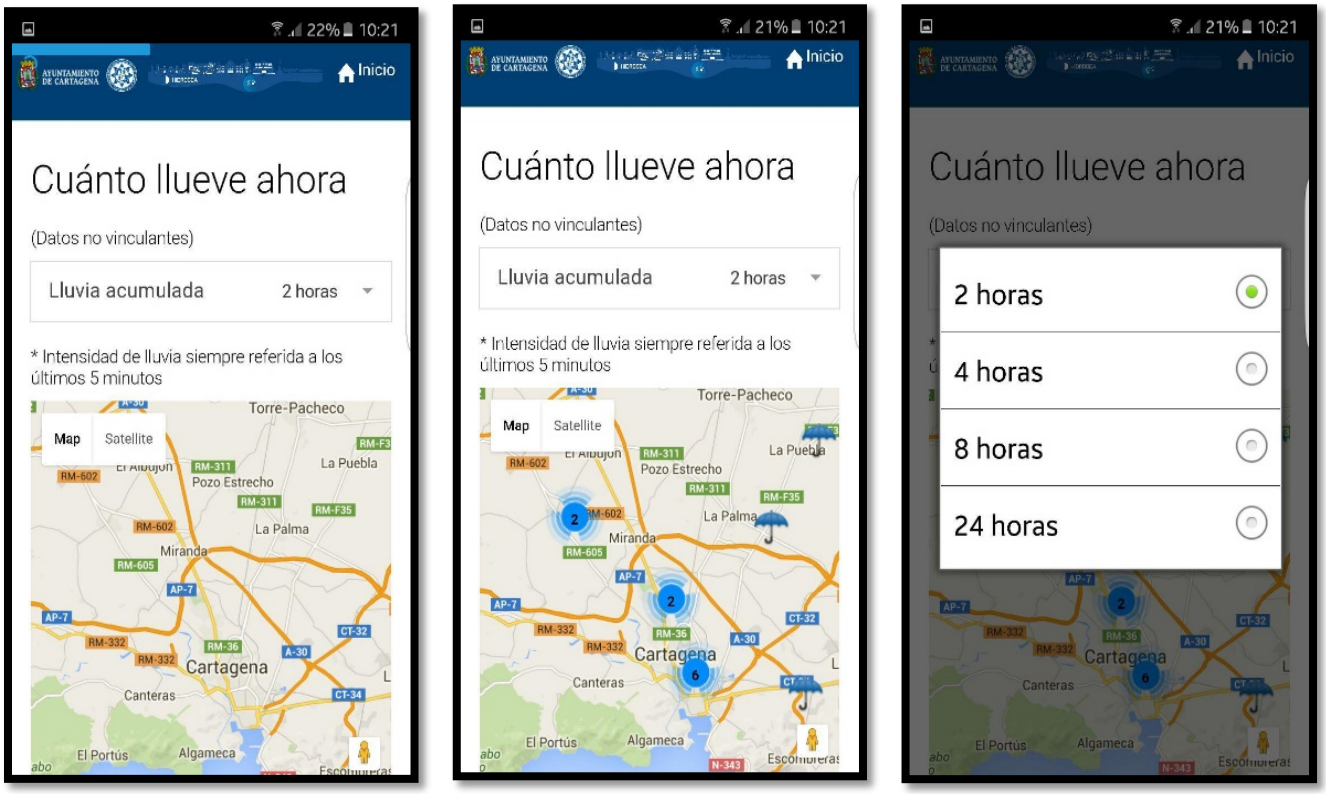


Ilustración 78: Pantalla Cuánto llueve ahora.

En ella, nada más entrar, se mostrará una barra de carga mientras la aplicación obtiene los datos a mostrar, en este caso, muestra un mapa con la ubicación de los diferentes pluviómetros que se encuentran en Cartagena. Además, se dispone de un desplegable, en el que el usuario podrá consultar la lluvia acumulada de las últimas 2, 4, 8 o 24 horas. Es importante destacar que esta información está siempre referida a los últimos 5 minutos.

En este mapa, los usuarios podrán obtener información de los diferentes pluviómetros, pulsando sobre el icono de paraguas del que deseen consultar su información. Esto se puede observar en la siguiente ilustración.

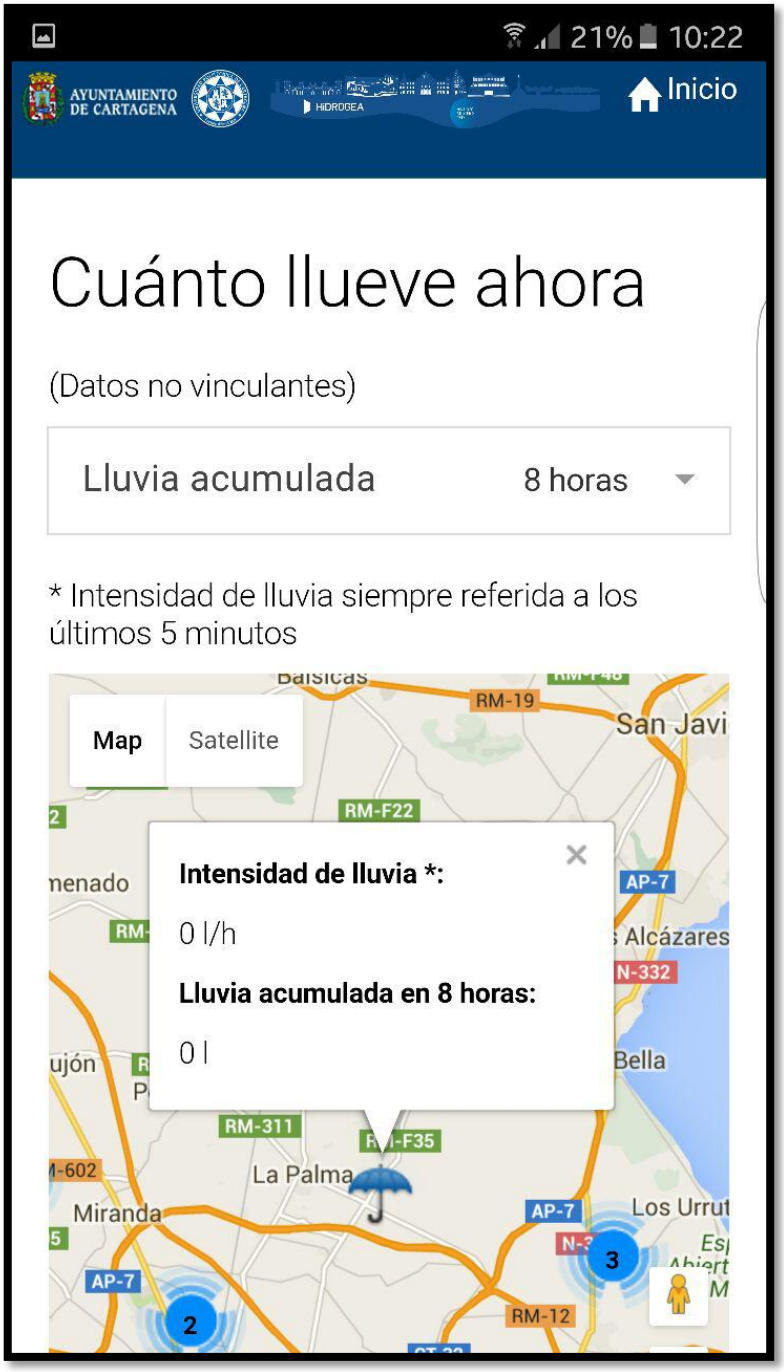


Ilustración 79: Información que se puede obtener de un pluviómetro.

A continuación, si el usuario pulsa sobre **"Donde estamos trabajando"**, podrá observar la siguiente vista:

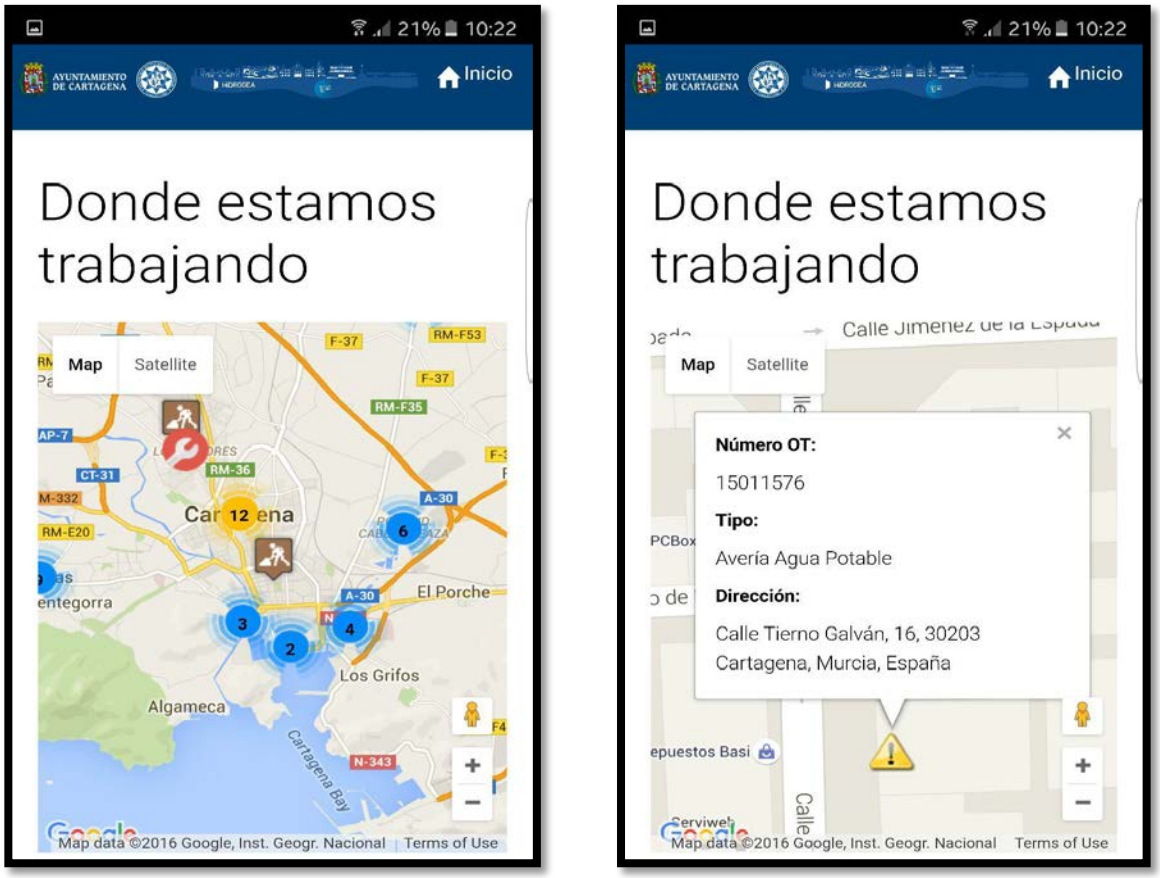


Ilustración 80: Pantalla Donde estamos trabajando.

En esta pantalla, se mostrará un mapa con diferentes iconos de diferentes tipos de obras.

En la imagen anterior, se pueden distinguir tres tipos de iconos:

- Obras (Icono marrón).
- Averías (Icono amarillo).
- Incidencias (Icono rojo).

El icono de obras, es referido a grandes obras, y pulsando sobre él se puede obtener más información sobre el tipo de obra que se está realizando, como se puede ver en la siguiente ilustración.

El icono de averías, muestra información sobre diferentes tipos de averías, en la imagen anterior se puede observar una infowindow de una avería, en la que se muestra el número OT, el tipo de avería y la dirección.

El icono de incidencias muestra la misma información que la de averías, pero esta vez está información proviene de un formulario que dispone un usuario autorizado.

Ilustración 81: Donde estamos trabajando, icono de obras.

En esta ilustración, se puede observar la información que se obtiene al pulsar sobre el icono de una obra. Si el usuario pulsa en “Más info”, puede obtener la información que se muestra en la siguiente ilustración.

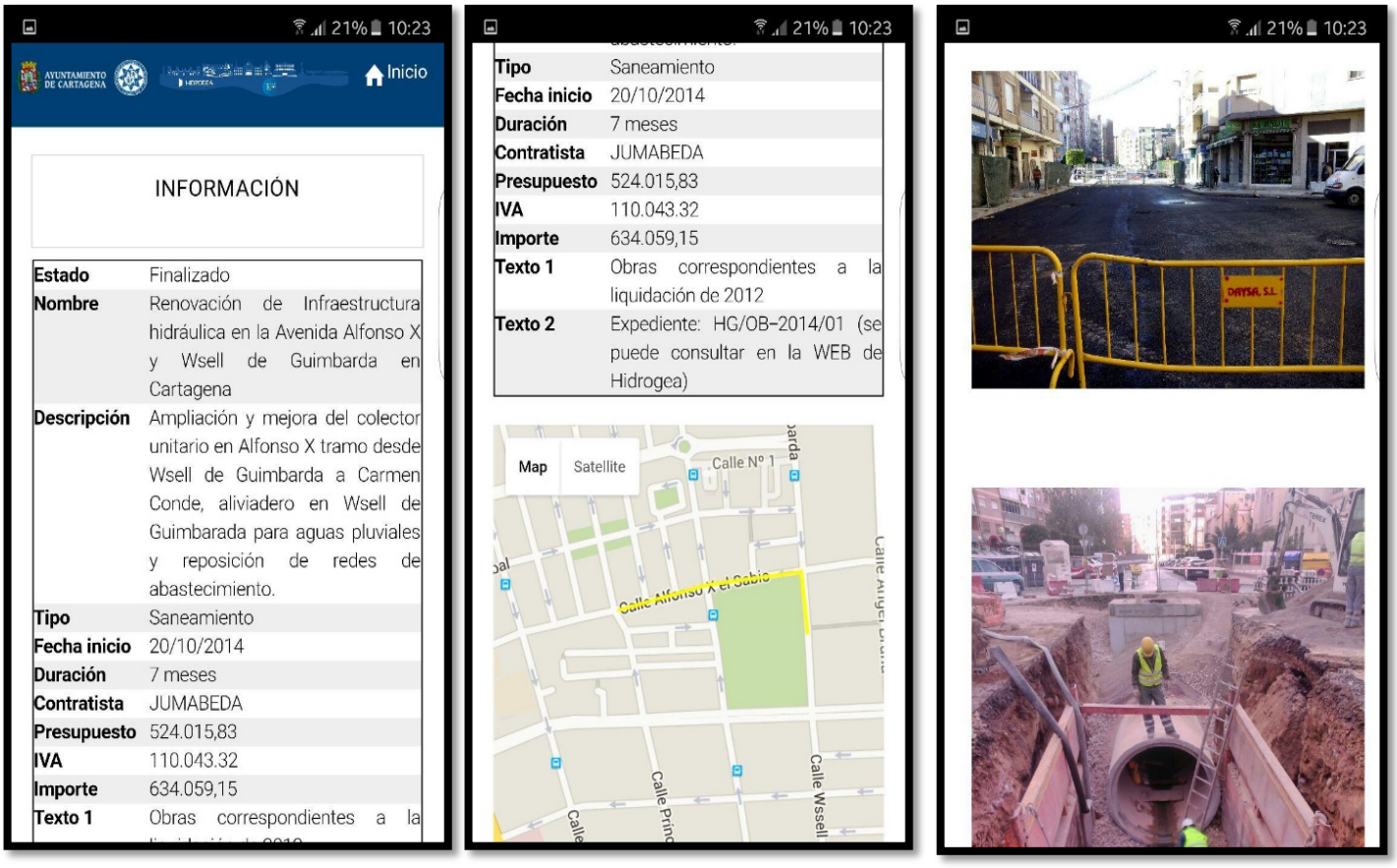


Ilustración 82: Información de una obra.

Como se puede observar, se muestra información del estado de la obra, el nombre, la descripción, el tipo, la fecha de inicio, la duración, el contratista, el presupuesto, IVA, etc.

Además, se incluye un mapa con una ruta marcada en amarillo, que se corresponde con el tramo que se encuentra en obras. También se incluyen unas imágenes de las obras.

Si el usuario ahora pulsa sobre **"Calidad del agua"**, se va a encontrar con lo que se puede observar en la siguiente ilustración.

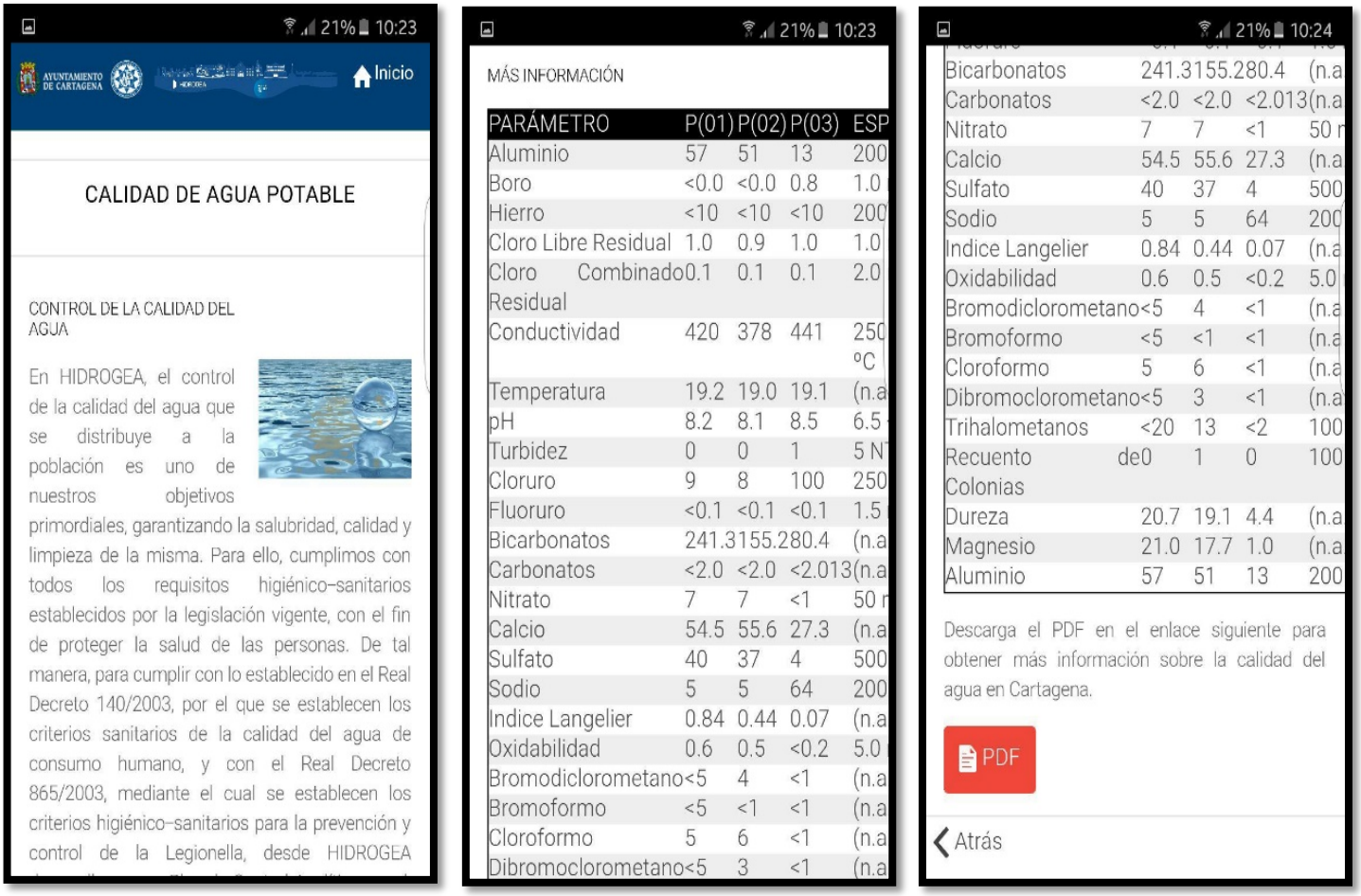


Ilustración 83: Calidad del agua.

En esta pantalla el usuario podrá obtener información sobre la calidad del agua, pudiendo obtener parámetros como Aluminio, Boro, Hierro, Cloro, Conductividad, etc. Además, tiene la posibilidad de descargar esta información en PDF.

La siguiente pantalla a la que el usuario podrá acceder será **"Programa de desratización"**.

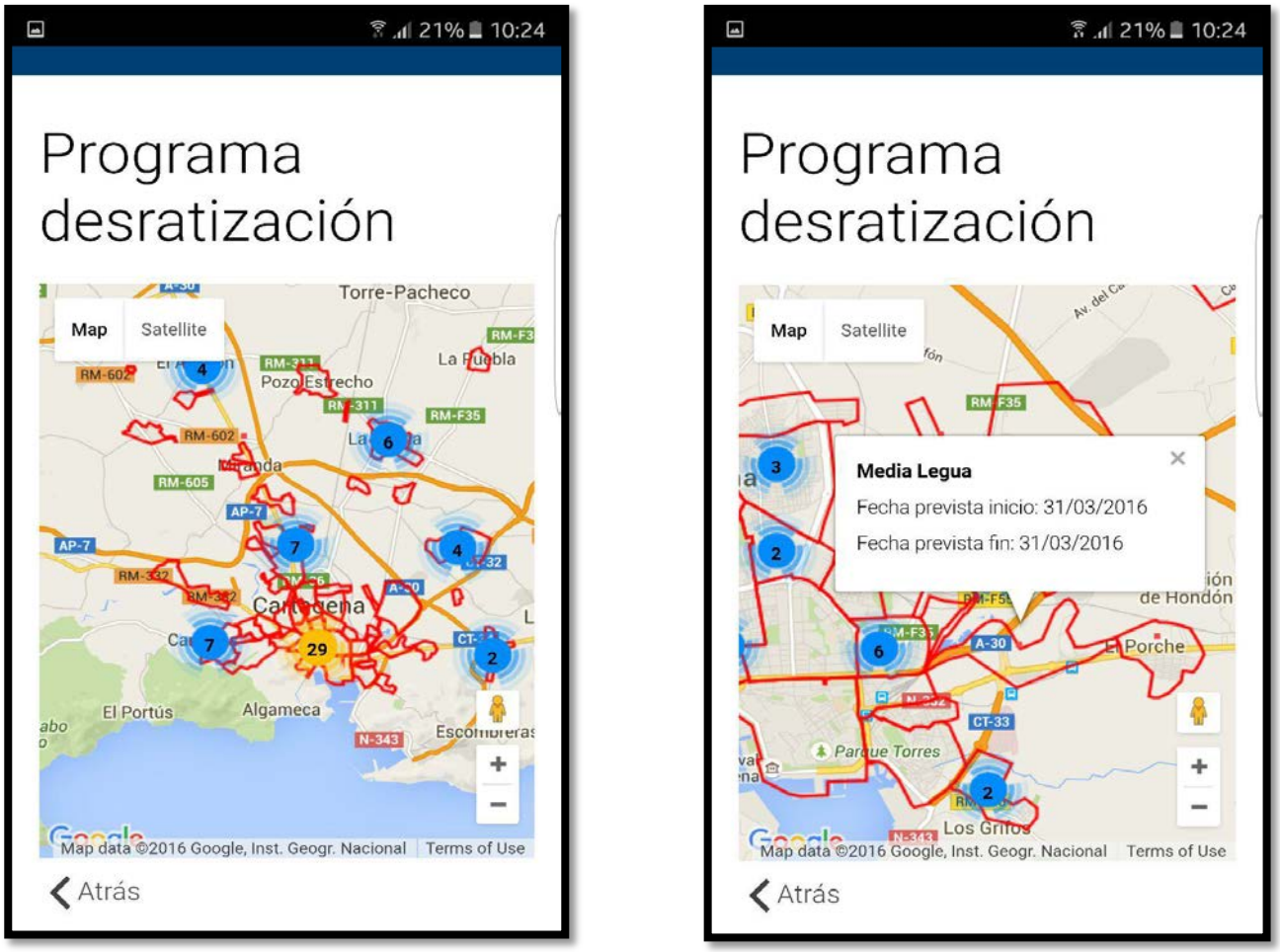


Ilustración 84: Programa desratización.

En esta pantalla el usuario podrá obtener información sobre el programa de desratización. Se mostrará un mapa dividido en áreas, en las que el usuario podrá pulsar y obtener información de las fechas previstas de desratización de esa área.

El siguiente será el botón de **"Perfil del contratante"**, en este, el usuario será redirigido a la página de Hidrogea sección perfil el contratante.

Ilustración 85: Perfil del contratante.

Si el usuario pulsa sobre “Datos técnicos”, accederá a una pantalla en la que se mostrará información sobre datos técnicos de Hidrogea, como puede ser datos básicos del servicio, sistema de abastecimiento de agua potable, etc.

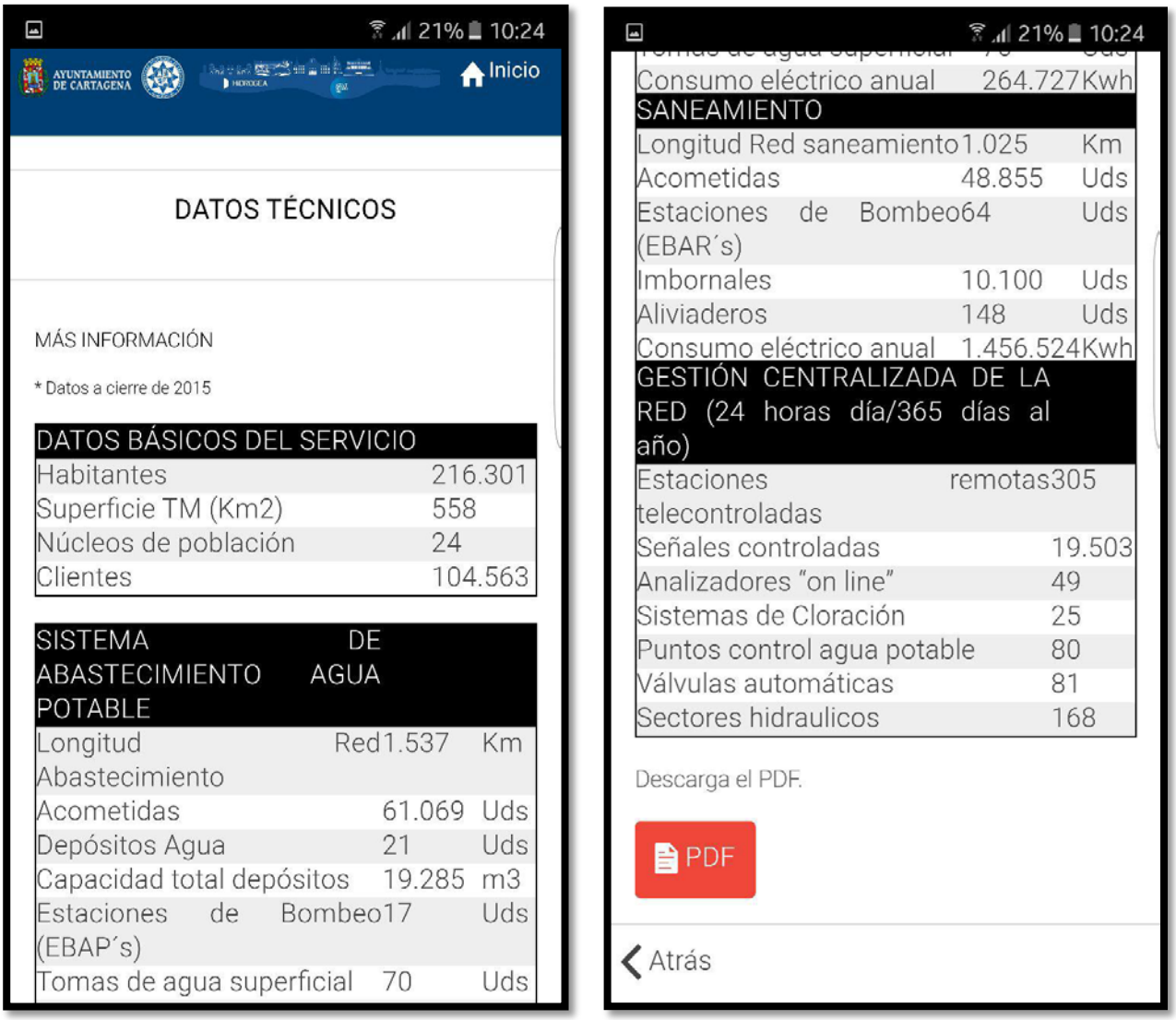
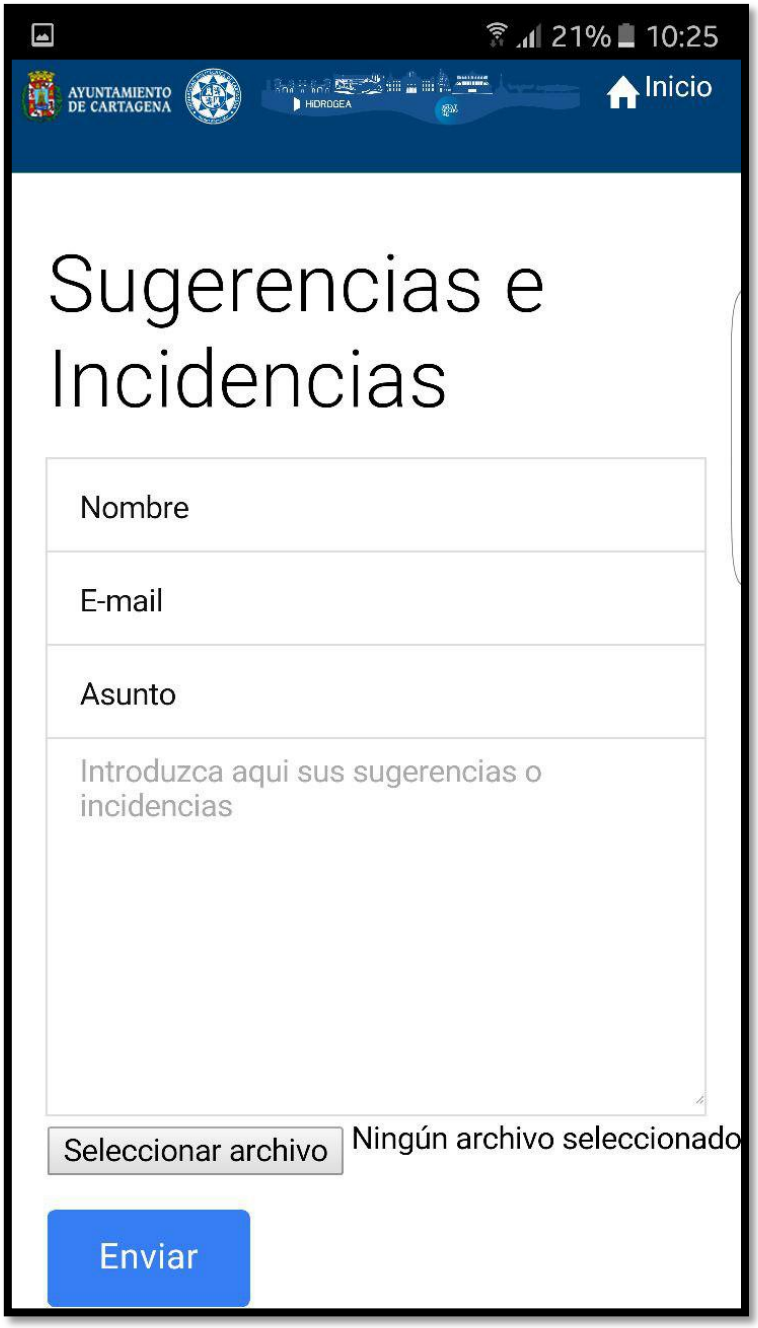


Ilustración 86: Datos técnicos.

La siguiente opción disponible para el usuario es la de **"Descripción del abastecimiento"**. En esta se muestra información sobre la procedencia del agua al termino de Cartagena.

Ilustración 87: Descripción del abastecimiento.

Si el usuario pulsa sobre **"Sugerencias e Incidencias"**, accederá a una pantalla en la que se encontrará con el formulario que se puede ver en la siguiente ilustración.



lustración 88: Sugerencias e Incidencias.

En este formulario, el usuario podrá informar de cualquier avería que se encuentre por la calle, por ejemplo, si el usuario ve una tapa de un alcantarillado levantada, podría sacar una foto y adjuntarla al formulario, e introduciendo sus datos, podría informar de tal avería. Esta solicitud llegaría a un administrador, que gestionará la incidencia, y en el caso en el que se necesite realizar una obra, creará un nuevo icono que se mostrará en el mapa de "Donde estamos trabajando".

El siguiente botón será **“Hidrogea web oficial”**, en este caso, se redirigirá al usuario a la página web oficial de Hidrogea.

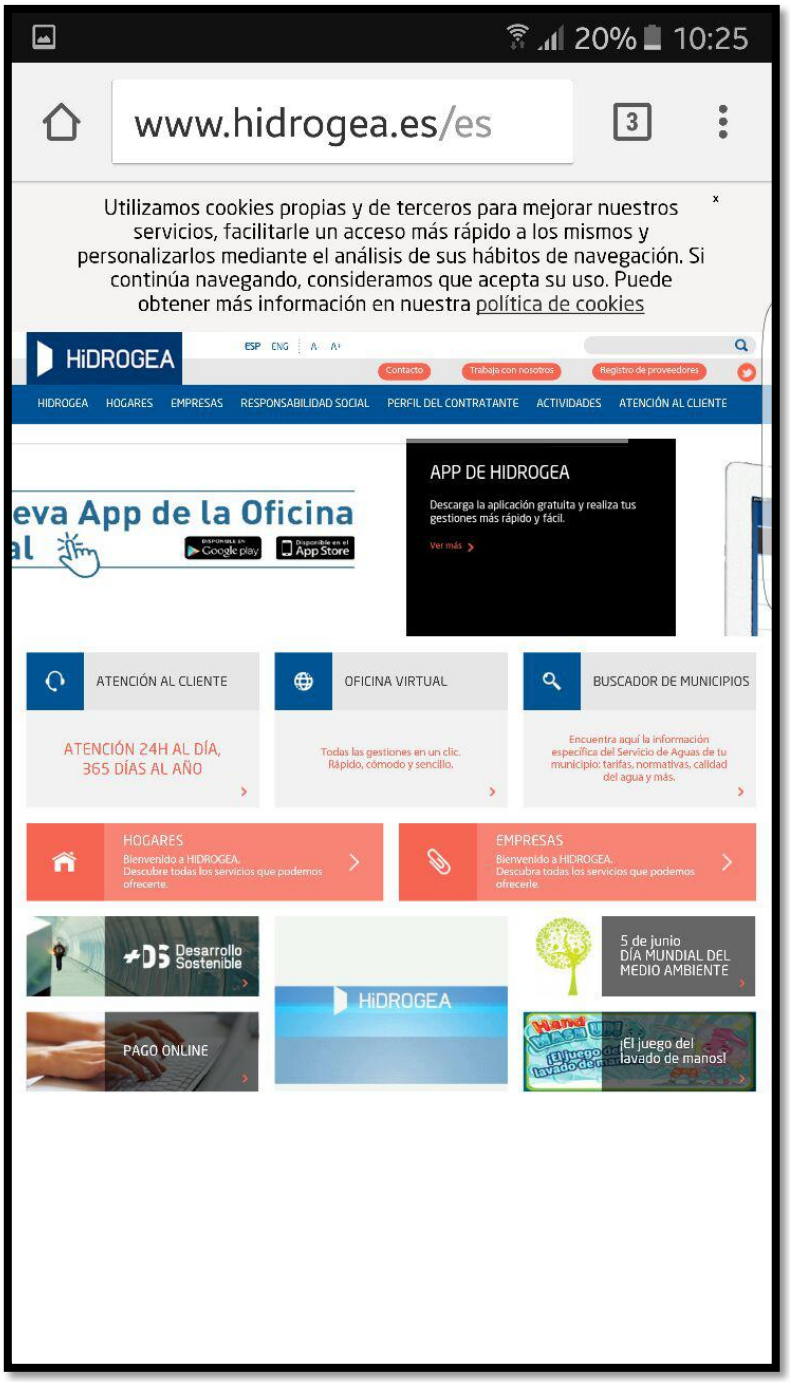


Ilustración 89: Hidrogea web oficial.

Por último, si el usuario pulsa sobre **“Oficina técnica”**, será redirigido a una página donde el usuario podrá descargarse la aplicación de Oficina virtual de Hidrogea.

Ilustración 90: Oficina virtual.

En la siguiente ilustración se muestra la funcionalidad de la aplicación de poder recibir notificaciones. Como se puede observar, en la imagen de la izquierda se recibe una notificación nativa, que llega produciendo un sonido en el dispositivo móvil. En la imagen de la derecha se muestra la notificación abierta dentro de la aplicación.

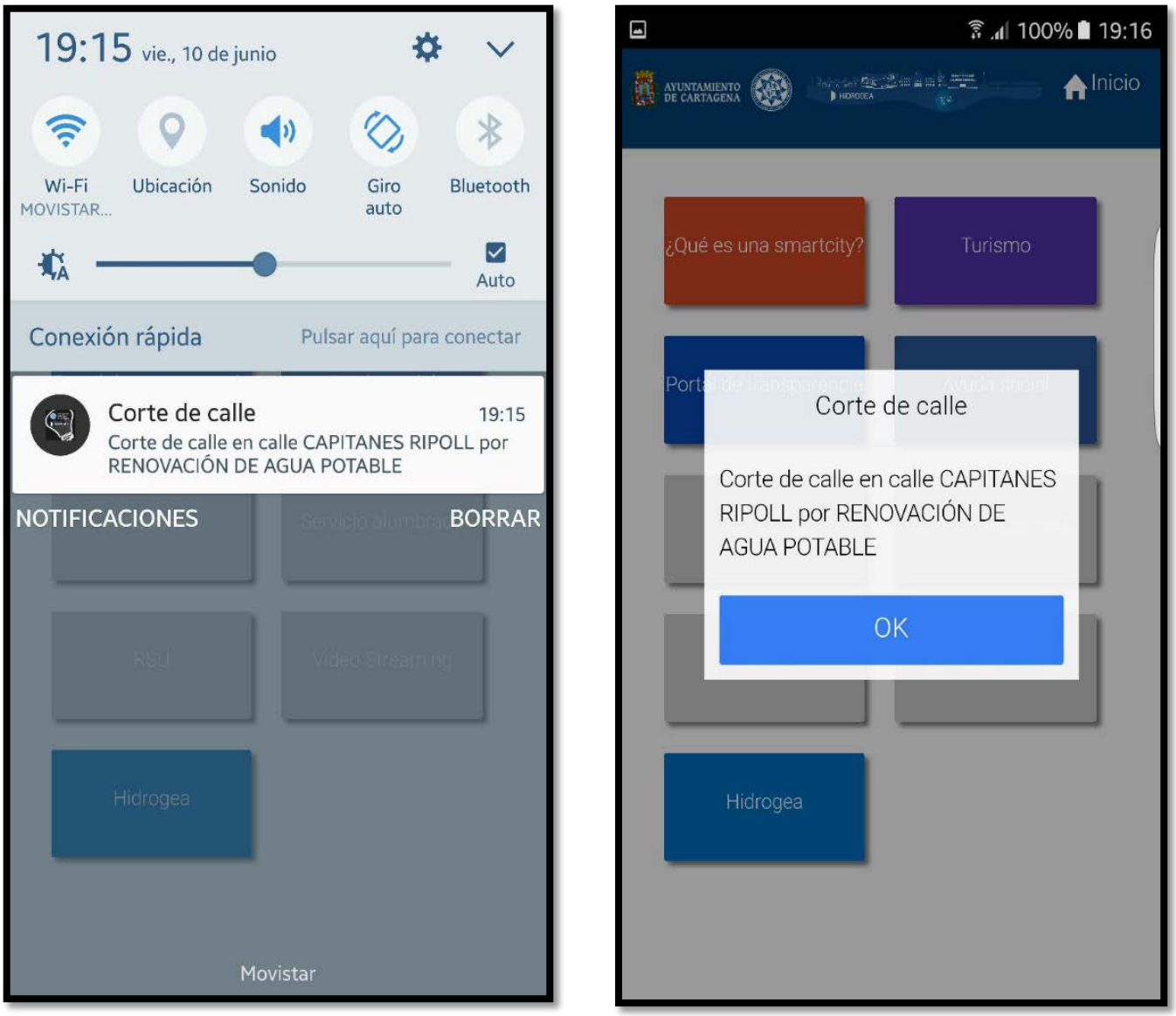


Ilustración 91: Notificación en la aplicación.

4.Conclusiones y líneas futuras

4.1. Conclusión

En este documento se ha descrito la evolución que sufren las Smart cities, se ha definido el concepto de Smart city y se ha explicado cómo funciona una Smart city. Una vez hecho esto, se ha descrito el problema al que se ha hecho frente en este proyecto y se ha realizado un estudio del arte de las Smart cities centrándose en la ciudad de Cartagena.

Una vez hecho esto, se ha mostrado la solución que se propuso al principio del proyecto y se ha visto que la mejor opción para la realización de este proyecto, era realizar una aplicación hibrida.

A lo largo del proyecto, se ha podido ver cómo ha ido cambiando la plataforma de notificaciones, pasando de una interfaz poco amigable para el usuario a una atractiva para el usuario. Esto es debido a que se ha hecho hincapié en que la aplicación fuera agradable para el usuario, permitiendo que se adapte a todo tipo de dispositivos y creando un menú en el que cambia el número de columnas según el dispositivo en el que se esté utilizando, es decir un diseño responsive.

El resultado obtenido ha sido una aplicación, que recibe los datos desde la nube y los muestra de manera automática, y que además permite recibir notificaciones de parte del administrador.

Cabe destacar el framework Ionic, ya que gracias a esta herramienta se ha conseguido implementar la aplicación. Destacar la sencillez de uso y la gran cantidad de documentación que dispone. Además, gracias a este framework ha sido posible desarrollar la parte de las notificaciones.

Finalmente, se ha llegado a la conclusión de que una plataforma de notificaciones desarrollada con Ionic y que recibe datos de la nube es la solución más eficiente para desarrollar una aplicación de este tipo, ya que, de esta manera, se tiene una aplicación siempre disponible y de fácil acceso.

4.2. Líneas futuras

Aunque las características esbozadas hasta el momento dejan abierta la posibilidad de expandir el sistema en infinitas direcciones, a continuación, se listan una serie de posibles extras a nivel operativo para expandir el modelo de negocio aún más:

- Creación de la APP para iOS y Windows Phone.
- Implantación de una fuerte imagen de compañía especializada en el tema.

5. Bibliografía y referencias

[1] Ionic: http://ionicframework.com/

[2] Notificaciones Android usando Ionic y Cordova: http://intown.biz/2014/04/11/android-notifications/

[3] Filezilla: https://filezilla-project.org/

[4] Cordova: https://cordova.apache.org/docs/es/latest/guide/overview/

[5] Nodejs: https://cordova.apache.org/docs/es/latest/guide/overview/

[6] AngularJS: https://angularjs.org/

[7] W3Schools: http://www.w3schools.com/