Escuela Politécnica Superior

GIFY: Sistema de Gestión de Incidencias

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
María Martos Ortega

Tutor/es:
José Vicente Berna Martínez

Universitat d'Alacant
Universidad de Alicante

Septiembre 2016

Justificación y Objetivos

A punto de terminar la carrera y con la experiencia de haber abordado en el curso anterior un proyecto ABP de envergadura, me planteo el trabajo final de grado como un reto personal donde poner en práctica mis habilidades y aprender nuevas tecnologías que ayuden a completar mi formación.

En el proyecto ABP del año pasado desarrollamos una guía televisiva online multiplataforma con algunas funcionalidades interesantes y útiles para el público. Afianzamos nuestros conocimientos en planificación y organización, diseño e implementación de aplicaciones web y móviles, y tuvimos la oportunidad de utilizar herramientas de desarrollo que no dominábamos. El resultado fue muy positivo y es un estímulo para seguir trabajando tales aspectos en este proyecto.

Asimismo, en el periodo de prácticas, estuve durante 6 meses en una empresa dedicada a la creación y distribución de contenidos *e-learning*, y aprendí muchos valores respecto al entorno usuario-dispositivo-Internet, desde el fundamento virtual y la interacción social; las ventajas del *feedback*, la reducción de gastos respecto a un entorno tradicional, el acceso rápido y sencillo desde cualquier dispositivo, la flexibilidad horaria, la desaparición de barreras espaciales y la existencia de una plataforma virtual donde se concentra toda la información.

Mi propósito, de alguna manera, era plasmar estos atributos en mi proyecto final de grado, realizando algo que me permitiese incluir tecnologías con las que no hubiese trabajado con anterioridad. El desafío estaba en encontrar un tipo de proyecto que me motivase, que fuese de utilidad y de interés para la sociedad y pudiese incluir en mi *portfolio* de cara a un posible futuro laboral.

A raíz de todo esto, surge la idea del proyecto: realizar un **sistema de gestión online de incidencias en la vía pública,** que permita a una entidad (ayuntamiento, empresa, superficie comercial, universidad, finca de vecinos, etc.) gestionar las incidencias que les envían los ciudadanos a través de la aplicación móvil de la plataforma. Es un sistema que involucra a cientos o miles de usuarios y que puede ser de interés para pequeñas ciudades y pequeños y medianos organismos.

Esta herramienta se presenta como un **servicio online**, accesible desde cualquier dispositivo conectado a Internet, **multiplataforma**, especialmente orientado hacia móviles, y con un diseño y funcionamiento completamente **intuitivo y sencillo de utilizar**, dirigidos a un público no experto.

Imaginemos que un ayuntamiento se une a la plataforma Gify para gestionar los problemas de mantenimiento que ocurren en su municipio. El primer paso es registrarse. A partir de este

momento, el ayuntamiento tiene, por un lado, un panel de gestión web donde administrar sus incidencias, y por otro, una aplicación móvil con la que los ciudadanos pueden reportar estas incidencias.

Desde el panel de gestión, la persona que ha creado la cuenta puede dar de alta usuarios gestores (encargados de gestionar) y operarios (encargados de resolver), tipos de incidencias (adoquines sueltos, farola rota, pintadas, césped mal cuidado …) y áreas de trabajo (aceras, alumbrado, mobiliario urbano, zonas verdes…), en función de sus necesidades. Además de clasificar, crear, editar y eliminar las incidencias que van llegando, se genera un histórico de cada incidencia en el que pueden incluirse notas. A cada incidencia se le asigna un responsable (operario) para que la resuelva personalmente, pudiendo ellos también registrar incidencias que se encuentren a pie de calle.

La aplicación móvil facilita que los vecinos de la localidad, en el momento que detectan algún problema de estas características, puedan fotografiarlo, ubicarlo, describirlo y enviarlo al sistema. Una vez hecho esto, se informa al ciudadano del proceso que va siguiendo la resolución de la incidencia a través de la aplicación. Los ciudadanos pueden ser baneados desde el panel de gestión si los reportes son inadecuados.

Con este tipo de herramientas el incidente se controla desde el momento que se detecta hasta que se soluciona, beneficiándose el ayuntamiento y los ciudadanos de la retroalimentación en este proceso.

Por tanto, uno de los objetivos del proyecto es crear una vía de comunicación directa entre la comunidad y la entidad cliente, ya que, a veces, puede ser complicado poner un parte de incidencias tradicional, por desconocimiento, motivos de tiempo o distancia.

El otro objetivo, implícito en el primero, es ofrecer un servicio de gestión para que puedan resolverse las necesidades acaecidas a consecuencia de incidencias (fallos, errores, roturas, etc.) con una herramienta cercana al ciudadano, cuyas características conllevan muchos beneficios: facilidad de uso, ahorro de tiempo y de recursos, comodidad, mayor accesibilidad y un incentivo para que la sociedad participe en este tipo de iniciativas.

Esta plataforma se ofrece como servicio a los agentes interesados, de forma que una pequeña población, empresa u entidad interesadas no ha de realizar ninguna inversión en infraestructuras, personal cualificado, mantenimiento ni desarrollo, simplemente ha de registrarse en la herramienta y comenzar a utilizarla, favoreciendo así la digitalización de servicios al ciudadano, la eficiencia en costes y la externalización de servicios no críticos.

Agradecimientos

Por confiar en mí; a mi familia, a mi pareja y a mi tutor.

Índice de contenidos

Índice de figuras

Introducción

Cada día se hace más uso de las aplicaciones distribuidas para la gestión económica, administrativa, documental, bancaria o, incluso, para gestionar las redes sociales (programar la publicación de *posts* o entradas). La gestión online supone una nueva concepción de la relación con la sociedad, empresas e instituciones. Hace referencia a la incorporación de las tecnologías de la información y comunicaciones en las organizaciones públicas y privadas, convirtiendo las oficinas tradicionales y transformando los procedimientos en papel en procedimientos electrónicos con el fin de mejorar los servicios.

Los usuarios pueden interactuar con las aplicaciones de gestión las 24 horas del día de forma online. No hay que ceñirse a un horario de oficina ni a días laborables. Tampoco es necesario acudir a la oficina presencial para realizar las gestiones pues se pueden realizar desde cualquier parte del mundo a través de Internet. El ahorro de tiempo es también considerable en desplazamientos, colas para ser atendidos, etc. Además, no es obligatorio realizar los trámites desde casa o la oficina ya que hoy todo el mundo dispone de un *smartphone,* una *tablet* o un ordenador portátil.

La tendencia actual es ofrecer herramientas en forma de servicios online. Las ventajas son muchas y es cada vez más usual que se tienda a externalizar las necesidades cotidianas utilizando este mecanismo. Servicios de almacenamiento en la nube como Google Drive o Dropbox, servicios de correo electrónico como Gmail o Yahoo!, gestores de incidencias, gestores de proyectos, etc. son usados constantemente por parte de las organizaciones y usuarios.

Las entidades y empresas que incorporan este tipo de servicios en la nube en su trabajo son progresivamente más heterogéneas y sus necesidades también varían, aunque todas coinciden en el propósito de ofrecer las mejores prestaciones posibles a sus clientes u usuarios. Hay muchos tipos de sistemas online de gestión de incidencias técnicas en equipos informáticos, en servidores, en ayuda al usuario, en facturaciones y contratos, etc. que hacen su función y corren un papel imprescindible en el correcto funcionamiento de estas organizaciones.

En ciertos ámbitos, como universidades o ayuntamientos, donde la extensión de infraestructuras es considerablemente amplia, existe la necesidad de gestionar también las incidencias de mantenimiento. Disponer de software que facilite la tarea de registrar incidencias, clasificarlas, asignarles responsables, etc., supondría llegar a una solución más rápidamente que si se pusieran partes de incidencias de forma tradicional.

Normalmente, los organismos que necesitan hacer uso de este tipo herramientas software, suelen desembolsar bastante dinero para poder utilizarlas: compra de producto, licencias, instalaciones, mantenimiento, etc., y los pequeños y medianos organismos no suelen disponer de los recursos necesarios para esto. Es por esto que los servicios en la nube son una buena alternativa al software que se vende convencionalmente, porque muchos de ellos son gratuitos o con cuotas de suscripción mensuales u anuales. Los precios pueden estar basados en algunos parámetros de uso, como el número de usuarios. También puede ofrecerse un modelo gratuito de este servicio con funcionalidades o alcance limitado y cobrar por funcionalidades mejoradas o alcance más amplio, aunque es bastante común que algunos servicios en la nube sean completamente gratuitos y los ingresos vengan de la publicidad.

Dicho esto, mi intención es ofrecer una herramienta software lo más sencilla posible y enfocada directamente a las incidencias que ocurran en instalaciones e infraestructuras de una organización. Será con vocación de servicio, es decir, va a ser un servicio online. Esto implica, como se ha comentado, una reducción de costes a nivel de mantenimiento y de soporte por parte del organismo interesado, además de evitar arriesgar con una gran inversión en un producto, ya que se tendría con este servicio un comportamiento parecido al alquiler. Sin necesidad de descargas ni instalaciones, la empresa u organización interesadas accederían a una página web donde gestionarían sus incidencias cómodamente y desde cualquier dispositivo. Por una pequeña cuota mensual podrían tener constancia y gestionar la resolución de todos esos incidentes que, por muy pequeños que sean, pueden entorpecer el normal funcionamiento del trabajo.

El sistema estará formado por una ***landing–page*** que mostrará la información acerca del sistema de gestión de incidencias y donde los clientes puedan registrarse y acceder a su **panel de gestión de incidencias en la web**.

El panel de gestión será accesible por el creador de la cuenta en la plataforma, que denominaremos *"propietario",* y los usuarios que este dé de alta en el sistema, *"gestores"* y *"operarios"*. El propietario puede encargarse de crear, editar y eliminar las incidencias, los tipos de incidencias, las áreas y los usuarios, entre otras funciones. También es quien tiene la primera toma de contacto con las incidencias enviadas, tanto por los usuarios de la *app* móvil como por los gestores y operarios a través de la *app web*. Después de revisar que la información de las incidencias sea correcta, se asignan a un operario para que la resuelva.

Los usuarios gestores tienen prácticamente los mismos permisos que el propietario, la única diferencia es que solo pueden crear usuarios de tipo operario. Si el propietario de la cuenta no va a ser el encargado de llevar la gestión de las incidencias, puede crear usuarios de este tipo para que lleven a cabo esa función.

El rol de operario se asigna a los usuarios técnicos que reparan y resuelven las incidencias. Sus funciones dentro del panel de gestión giran en torno a las incidencias que les han asignado y a los tipos de incidencias.

Además de las tareas básicas de gestión que se han descrito en los párrafos anteriores, los usuarios pueden reportar a los ciudadanos que molestan a través de la *app* movil, añadir notas a las incidencias y consultar el histórico que se va generando de cada una desde que llegan al sistema. También tienen la tarea de cambiar el estado de las incidencias en función del punto de su resolución en el que se encuentren.

Las incidencias pasan por varios estados desde que se notifican hasta que se solucionan. Estos estados son: pendiente, asignada, en curso, resuelta y cerrada.

Una incidencia está *pendiente* cuando ha llegado al sistema y no ha sido asignada a ningún operario. Una vez *asignada*, cuando el operario la acepte, pasaría a estar *en curso*. En el momento que la incidencia esté solucionada, el operario la marcará como *resuelta* y será devuelta al gestor o propietario que la asignó. Este, una vez dé el visto bueno a esa resolución, la cerrará, y este será el último estado de la incidencia, *cerrada*.
Si en algún momento la incidencia fuera rechazada/devuelta por un operario al superior que se la asignó, pasaría a *pendiente* de nuevo hasta que fuese reasignada a otro usuario.

También, habrá una ***app* móvil** que será la herramienta que los usuarios ciudadanos utilizarán para reportar las incidencias, adjuntando foto, ubicación y texto. Ellos podrán conocer en todo momento el estado de las incidencias que han notificado a través de la propia aplicación.

Estas serían las interfaces de las que harían uso, tanto los usuarios de la entidad registrada que quiere gestionar sus incidencias, como los usuarios ciudadanos que las reportan. Por otro lado, estaría la parte de administración de la plataforma. Un panel de gestión de clientes y usuarios de Gify, cuyo administrador (*yo*) se encargaría de aceptar las altas de nuevos usuarios, eliminar usuarios baneados, o desbloquearlos si fuese necesario, verificar las retribuciones de los clientes, etc.

El esquema del sistema quedaría de la siguiente manera:

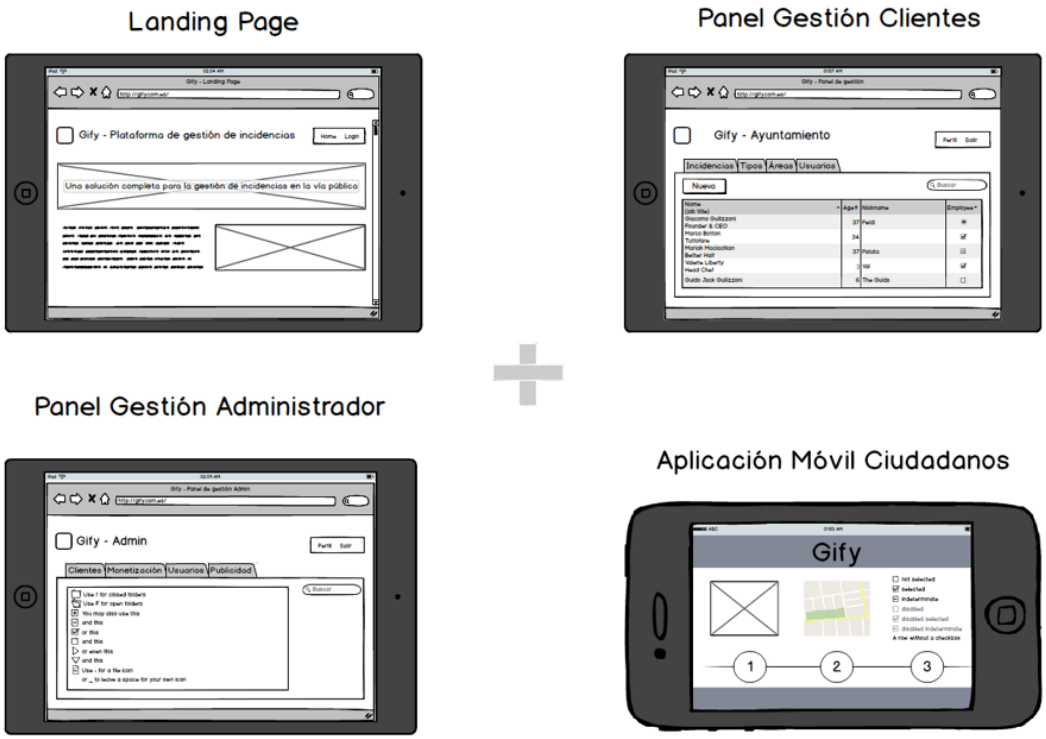


Figura 1: Esquema del sistema. Elaboración propia.

Marco teórico o estado del arte

Introducción: estudio de las necesidades

El número de organizaciones que se preocupan por escuchar las quejas de sus usuarios o clientes crece cada día más. De las quejas se recoge una valiosa información útil para construir un servicio diferencial que vincule afectivamente al usuario. Neutralizar los posibles errores, causados por incidencias de todo tipo, y los descontentos que provocan, implica mejorar la fidelización y confiabilidad en los organismos. Las vías por las que se efectúan las reclamaciones y denuncias ciudadanas también evolucionan con el paso de los años. Se buscan sistemas de comunicación que acerquen las organizaciones a los usuarios y que agilicen los procedimientos de gestión.

En el ámbito de los ayuntamientos de municipios, localidades y ciudades, el mantenimiento urbano sigue siendo una asignatura pendiente. Si observamos las noticias en este último año, no es extraño encontrar denuncias vecinales hacia los gobiernos locales y el descuido de infraestructuras y zonas. Estas quejas vienen, en gran medida, por dejadez por parte de los organismos, pero en ocasiones la causa está en el desconocimiento de los hechos.

Esta noticia, de elnuevodia.com a 3 de diciembre de este año, describe la rotura de la tubería principal de agua de la ciudad de Ponce [1]. La incidencia tardaría en resolverse tres días, igual que otra rotura que ya sucedió en el mes de mayo y que tuvo a la ciudad varios días sin agua. En España, concretamente en Soria, el 9 de junio de 2015 en el 20minutos.es, salta la noticia del descubrimiento de robos de capitales en iglesias sin que nadie se hubiese percatado [2]. Los hurtos sucedieron en varias localidades en 2012 y nadie se dio cuenta del suceso hasta mayo de este año, cuando el párroco pasó por allí y dio la voz de alerta. Cabe destacar que se habla de zonas bastante deshabitadas, aun así, es llamativo que las administraciones en una de las provincias de la región europea con más tesoros artísticos e históricos de toda Europa no vigilen este tipo de cosas.

También ocurrió algo similar en Iguazú, Argentina, cuando vecinos de la ciudad empezaron a ver circulando motocicletas que estaban retiradas en el depósito de vehículos municipal. El pasado día 6 de diciembre, Misionescuatro.com daba la noticia de que faltaban 641 motocicletas de sus instalaciones y nadie se dio cuenta de lo ocurrido [3]. Se cree que se trata de un robo, pero están averiguando como ha sucedido, ya que el subsecretario se percató de lo ocurrido por las quejas vecinales y la prensa. La participación ciudadana es de gran ayuda en el ámbito de la prevención y la seguridad.

En el diariovasco.com, el 12 de Julio de 2015 en la localidad de Ametzagaña, podemos ver como se toman medidas urgentes [4] para reparar los desperfectos de un parque de la ciudad después de años de abandono y numerosas denuncias y quejas ciudadanas. Las protestas contra la dejadez de los ayuntamientos proliferan en toda España. En el distrito de Arganzuela, Madrid, el descontento viene a causa del mal estado de los árboles [5], agravado por una plaga de escarabajos, y el descuido por parte de la administración pública, que atemoriza a los vecinos. Este artículo del día 26 de octubre de 2015 en el diario larazon.es, además, narra la caída de un ejemplar sobre tres vehículos aplastándolos. El suceso tuvo lugar el día 17 del mismo mes teniendo como testigos a todos los vecinos y afectando a una mujer, copiloto de uno de los coches, que fue trasladada al hospital con pequeños cortes.

Hay infinidad de casos de este tipo, donde aun siendo incesantes las quejas de los vecinos al gobierno municipal, no se actúa en consecuencia hasta que el problema no es grave. Como el caso de estos bloques de pisos infectados de ratas y filtraciones fecales [6] en Tavernes, Valencia. Noticia de lasprovincias.es, del 26 de noviembre de este año. O estas obras mal realizadas sobre pavimentos [7] en un pueblo gallego, Vilar de Barrio, en Ourense, donde incluso el informe de un técnico avisaba del pésimo estado en el que se encontrarían al cabo de un tiempo. Este artículo pertenece a lavozdegalicia.es.

Los ayuntamientos no son las únicas organizaciones que gestionan de manera ineficiente el mantenimiento y cuidado de las infraestructuras. En lagacetadesalamanca.es se publica el mal estado de los techos de las bibliotecas de algunas facultades de la Universidad de Salamanca [8]. El 90% de quejas que reciben en el Campus de Miguel de Unamuno y en la biblioteca Abraham Zacut se deben a problemas con las goteras y la climatización de algunos edificios. Otras filtraciones en los techos que desencadenaron polémica fueron las ocurridas en el Hospital de Niños Sor María Ludovica, en la ciudad de la Plata (Argentina). Esta noticia, del día 9 de noviembre de 2015 en eldia.com, comenta las averías que se produjeron a causa de filtraciones de un lavabo en mal estado, que obligaron a aplazar doce operaciones que estaban previstas [9]. Se reclama que en el quirófano que salió perjudicado no había buen mantenimiento desde hacía años.

Otro de los casos de denuncias vecinales ha sido el cierre de un supermercado en San Sebastián de la Gomera, en Las Islas Canarias, por el ruido y las vibraciones constantes generados por la maquinaria industrial del establecimiento [10]. Esta noticia, a fecha de 29 de septiembre de 2011 en el diariodeavisis.com, cuenta que estos ruidos afectaban gravemente a la salud de la comunidad de vecinos del supermercado, quienes se vieron obligados a interponer varias quejas al ayuntamiento. Las denuncias por falta de insonorización del establecimiento y la falta de algunos permisos necesarios para la realización de las actividades que allí se hacían propiciaron el cierre del local. Las manivelas y picaportes defectuosos de otro bloque de apartamentos en la provincia de

Jujuy, al norte de Argentina, son los protagonistas de otra noticia del mal mantenimiento en comunidades de vecinos. Hasta 5 horas estuvo atrapada una joven dentro del baño de su apartamento, [11] hasta que una vecina alcanzó a oír sus llamadas de auxilio y pudo llamar a la policía. Ya había habido quejas y sucesos similares anteriormente, según los vecinos, por el mal funcionamiento de los picaportes del edificio. Esta información se recoge en el portal de noticias villamariavivo.com, el 31 de mayo de 2014. Parecido ha sido el caso de una señora que tuvo también que esperar 5 horas a que las directivas del centro comercial donde forzaron su coche respondieran a su denuncia [12]. Pulzo.com se hace eco de la noticia de 3 casos de robo en el parking del centro comercial de Hayuelos, en Bogotá (Colombia) el día 6 de abril de este mismo año. Ese día era festivo en la región y los servicios mínimos de personal afectaron al procedimiento de denuncia que quería dar la víctima del robo.

La dejadez y mala gestión de los espacios e infraestructuras de los organismos repercuten en todos y cada uno de los que forman parte de él. Hemos visto varios ejemplos de las consecuencias de los problemas de mantenimiento y seguimiento de incidencias; robos, accidentes, insalubridad, daños materiales, etc. aunque hay muchos más sucesos y más graves, incluso con pérdidas humanas. En general, provoca una sensación de temor a lo que pueda ocurrir y de frustración al ver que no se hace lo suficiente para subsanar los errores. Las pérdidas de dinero también son una consecuencia a tener en muy en cuenta. Por ejemplo, En Santa Bárbara, Honduras, se estiman pérdidas millonarias por la mala situación de las vías que dificultan el transporte del producto que se recolectan en las fincas de café de la localidad [13]. El 21 de este mes, laprensa.hn se hacía eco de las denuncias de caficultores que aseguraban que hacía años que no se acondicionaban las vías principales del municipio y esto repercutía directamente en su trabajo. Hecho incomprensible por parte de los dirigentes del lugar, ya que el 80% de la economía de esa región se basa en la producción de estas semillas y esto repercute en toda la comunidad.

Aunque son las organizaciones las que tienen que poner de su parte para mejorar los servicios de mantenimiento, la participación ciudadana es clave en la detección y aviso de estas incidencias. La rápida actuación de la sociedad en casos de este tipo favorece que las consecuencias de algunos sucesos sean mínimas. Aunque es evidente que la prevención debería ser lo principal para anticiparse a los problemas, tener la posibilidad de que la ciudadanía pueda notificar de forma masiva las incidencias de una forma sencilla y directa puede ayudar a ejercer la presión necesaria para que las instituciones pongan remedio a los problemas que afectan a las personas.

En la Universidad Nacional La Molina, Perú, gracias a la colaboración vecinal, se pudo dar aviso de un incendio que se originó [14] la tarde del 26 de noviembre de este mismo año. Elcomercio.pe, un portal de noticias peruano, dispone de un número de teléfono donde la gente informa sobre siniestros que ocurren en toda la región vía WhatsApp. También se hacen eco de las

incidencias y quejas vecinales a través de una cuenta de Twitter (@wasapEC). Es una iniciativa muy acogida por los usuarios que participan activamente en el proyecto.

No solo los portales de noticias, que se nutren de los sucesos, son los interesados en promover la colaboración ciudadana. Vemos otro ejemplo, en esta noticia [15], aunque sea de diferente índole, deja ver como las organizaciones piden ayuda directamente a los usuarios. Elperiodicoextremadura.com, a 29 de septiembre de 2015, se hace eco del aumento de actos vandálicos en distintos parques de Plasencia, donde se tuvo que recurrir a policías de paisano y a una petición de la Concejalía del Interior, demandando ayuda a los vecinos para poder detectar y coger a los vándalos realizando las infracciones.

La asociación Zaragoza Ciudadana presentó en el Ayuntamiento de la ciudad un listado con más de 70 denuncias y propuestas realizadas por la sociedad zaragozana [16]. Elperiodicodearagon.com, el 6 de Julio de 2015, cuenta que esta asociación puso en marcha la iniciativa *Tu Barrio*, que se lanzó para que los vecinos "pasearan, observaran y propusieran medidas para mejorar la ciudad". Los principales focos de quejas se centran en el mal estado u abandono de edificios emblemáticos, la tala de árboles, la suciedad y el deterioro de espacios públicos. Sacan como conclusión que si se atendieran las denuncias con mayor rapidez se evitarían problemas posteriores, puesto que días antes cayó un árbol de 14 metros sobre una terraza, ocasionando, afortunadamente, solo daños materiales.

La ciudad valenciana de Torrent también es un ejemplo de motivación y participación ciudadana en avisos de incidencias. La noticia la proporciona lasprovincias.es el 11 de noviembre de este año y expone que a lo largo del año se han resuelto más de 1500 incidencias gracias a los vecinos de esta localidad [17]. Hay un teléfono de atención al ciudadano que ha sido el método más utilizado (un 44%) para instar las quejas o peticiones a la administración del ayuntamiento. Otros vecinos decidieron hacer uso de la web de incidencias que proporciona el organismo público (22%) y en un menor porcentaje, hubo usuarios que prefirieron la manera tradicional personándose en las oficinas de atención al ciudadano (11%). También, disponen de una aplicación móvil para estos fines, y aunque está teniendo muy buena acogida, solo un 3% de los vecinos ha recurrido a ella desde su lanzamiento, en el mes de febrero. Las áreas de las que más se han recibido quejas son las relacionadas con el alumbrado, la gestión de basuras, la limpieza y la gestión de jardines, y el tiempo medio de respuesta del Ayuntamiento ha sido de 13 días después de las notificaciones. Curiosamente, el proceso de notificación de incidencias online que proporciona la administración pública de esta localidad tiene un funcionamiento muy similar al que va a tener el proyecto que yo estoy realizando. Es un sistema de Gestión de Incidencias en Vía Pública (GIVP) online donde no hace falta registrarse para enviar una incidencia, solamente hay que escribir un texto explicativo de lo que se quiere comunicar y puede geolocalizarse y adjuntar fotografías. Es muy interesante encontrar herramientas

parecidas a las que uno está desarrollando, implementadas en España, y funcionando correctamente y con éxito, cuando estás haciendo un proyecto final de carrera. Por este motivo, más adelante en este apartado, se incluirá un análisis más detallado del funcionamiento del servicio.

Como podemos ver, cada vez más ciudades se unen a este tipo de iniciativas que benefician a todos, incluso a los comercios locales. La Concejalía de Comercio, Mercados y Consumo de la ciudad de Alicante, desde el pasado mes de julio, ha puesto a disposición ciudadana un buzón de incidencias (comercio.incidencias@alicante.es) y ha resuelto ya el 85% de las quejas notificadas [18]. Laverdad.es señala que la idea de este buzón era centralizar todas las incidencias relacionadas con el área de Comercio, puesto que, en ocasiones, los comerciantes y asociaciones no tenían claro a quién dirigir sus quejas. Según la noticia, el área de Atención Urbana ha solucionado todos los problemas en un tiempo record.

Las necesidades de las organizaciones poco a poco se van adaptando al escenario tecnológico actual. La sociedad lleva tiempo haciendo uso de las redes sociales como escaparate de las quejas. De hecho, una noticia del 4 de octubre de 2015, en el diarioinformacion.com, nos muestra como los vecinos de la ciudad alicantina de Elda muestran en Facebook la descuidada imagen de la ciudad [19].

Es importante conocer el rumbo que está tomando la gestión de incidencias con la incorporación de Internet y las tecnologías, y que esto está propiciando el auge de sistemas especializados en el ámbito de las incidencias en vías públicas u organismos.

Toda la presentación de datos que se ha hecho hasta ahora viene a demostrar que hacen falta sistemas que permitan la gestión de incidencias en vías públicas, espacios e infraestructuras de forma ágil, sencilla e instantánea. Que un usuario detecte una incidencia y automáticamente con una foto, texto y ubicación, se la pueda notificar a un gestor, permite a la administración actuar con mayor rapidez para resolver los problemas.

A continuación, se va a realizar una exposición del análisis de las herramientas y servicios online de que dispone la sociedad actualmente para atajar esta problemática.

Análisis de herramientas

Aplicaciones web

Desde hace unos años, han proliferado las aplicaciones web y móviles dedicadas a la notificación de incidencias en la vía pública. El auge de la participación ciudadana en el ámbito de la administración ha hecho que muchos organismos se planteen poner a disposición de sus usuarios herramientas de este tipo que les ofrezcan una vía de comunicación rápida y sencilla.

En concreto, los ayuntamientos son los que más se han unido a esta iniciativa, proporcionando a los ciudadanos de sus localidades una plataforma con la que puedan informar de las incidencias locales.

Estas herramientas suelen ser aplicaciones web desarrolladas por los mismos ayuntamientos o por empresas externas que se dedican a proveer software de este tipo. Las ventajas de que los usuarios dispongan de estos servicios a través de Internet, en lugar de que sean productos instalables en ordenadores, tablets o móviles, mejoran la comodidad de uso de las aplicaciones, puesto que no se requieren unas configuraciones previas antes de utilizarlas. Al disponer todo el mundo de una conexión a Internet, comunicar una incidencia es tan sencillo como dirigirse a la web del sistema de gestión, realizar el reporte y supervisar que se solucione lo antes posible.

Buscando en Internet, podemos encontrar muchos sistemas de gestión online de incidencias en la vía pública. Como hemos dicho, casi todos están relacionados con ayuntamientos que se han sumado a los beneficios de la colaboración ciudadana en el ámbito del mantenimiento de los municipios, pero además de las aplicaciones web, también se suele ofrecer aplicaciones móviles que pueden descargarse y utilizarse de la misma forma a través de Play Store y/o App Store.

Algunas de las plataformas que existen hoy en día para comunicar incidencias o desperfectos en las ciudades se muestran a continuación.

Línea Verde

Este proyecto de participación ciudadana está encuadrado en el marco de la prestación de servicios inteligentes a las ciudades. Entre otros servicios relacionados con temas medioambientales, ofrece este nexo de comunicación directa con la administración local asociada al proyecto, con la que el ciudadano puede comunicar incidencias en su municipio, tanto a través de la web como de la aplicación móvil. Para conocer más sobre este proyecto, se puede acceder a su web [20].

Figura 2: Web de la aplicación "Línea Verde". Qué es.

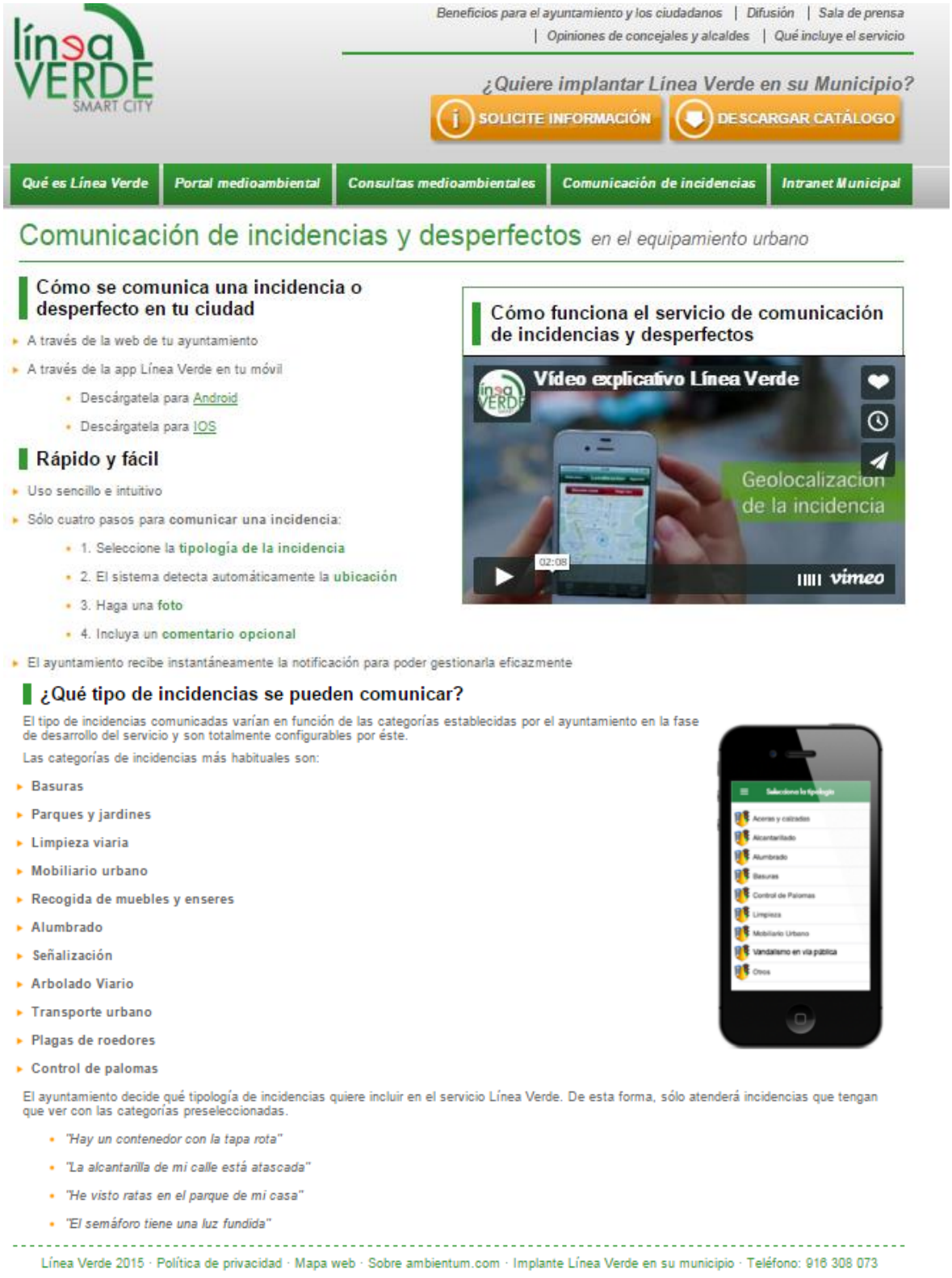


Figura 3: Web de la aplicación "Línea Verde". Cómo funciona.

GIVP – Portal Ciudadano

El ayuntamiento de Torrent, un municipio de Valencia, pone a disposición ciudadana un servicio web donde cualquiera puede dar aviso sobre desperfectos, incidencias o averías en la vía pública. Una vez notificada la incidencia, puede consultarse su estado mediante los datos de referencia que se envían al usuario que ha reportado la incidencia. Se puede acceder a este servicio a través de su página web [21].

Figura 4: Aplicación web GIVP del Ayuntamiento de Torrent. Inicio.

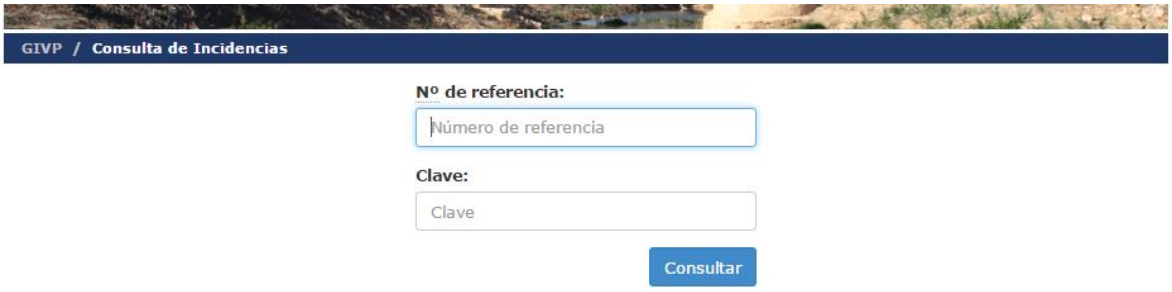


Figura 5: Aplicación web GIVP del Ayuntamiento de Torrent. Consulta de Incidencias.

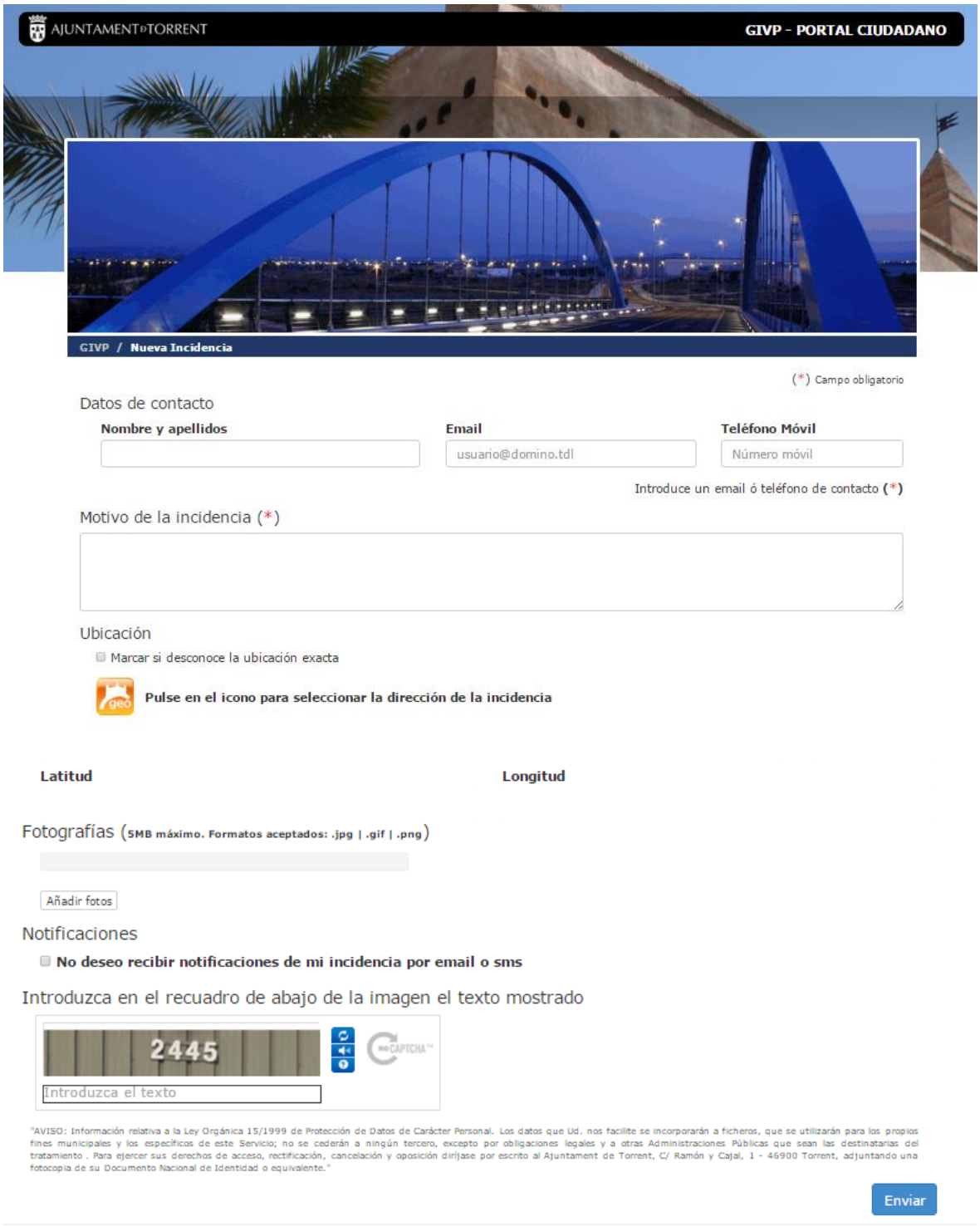


Figura 6: Aplicación web GIVP del Ayuntamiento de Torrent. Nueva incidencia.

Sistema GECOR (Incidencias en la Vía Pública)

GECORTrafic, es un sistema para la gestión del mantenimiento de dispositivos que permite reducir los tiempos de respuesta entre una incidencia y su resolución. El ayuntamiento de Málaga hace uso de esta solución tecnológica, donde el ciudadano puede comunicar las incidencias que se detecten relacionadas con la señalización vertical o elementos semafóricos, permitiendo conocer en todo momento el número de incidencias, su estado y la solución llevada a cabo. Para más información se puede visitar la página web de movilidad de Málaga [22].

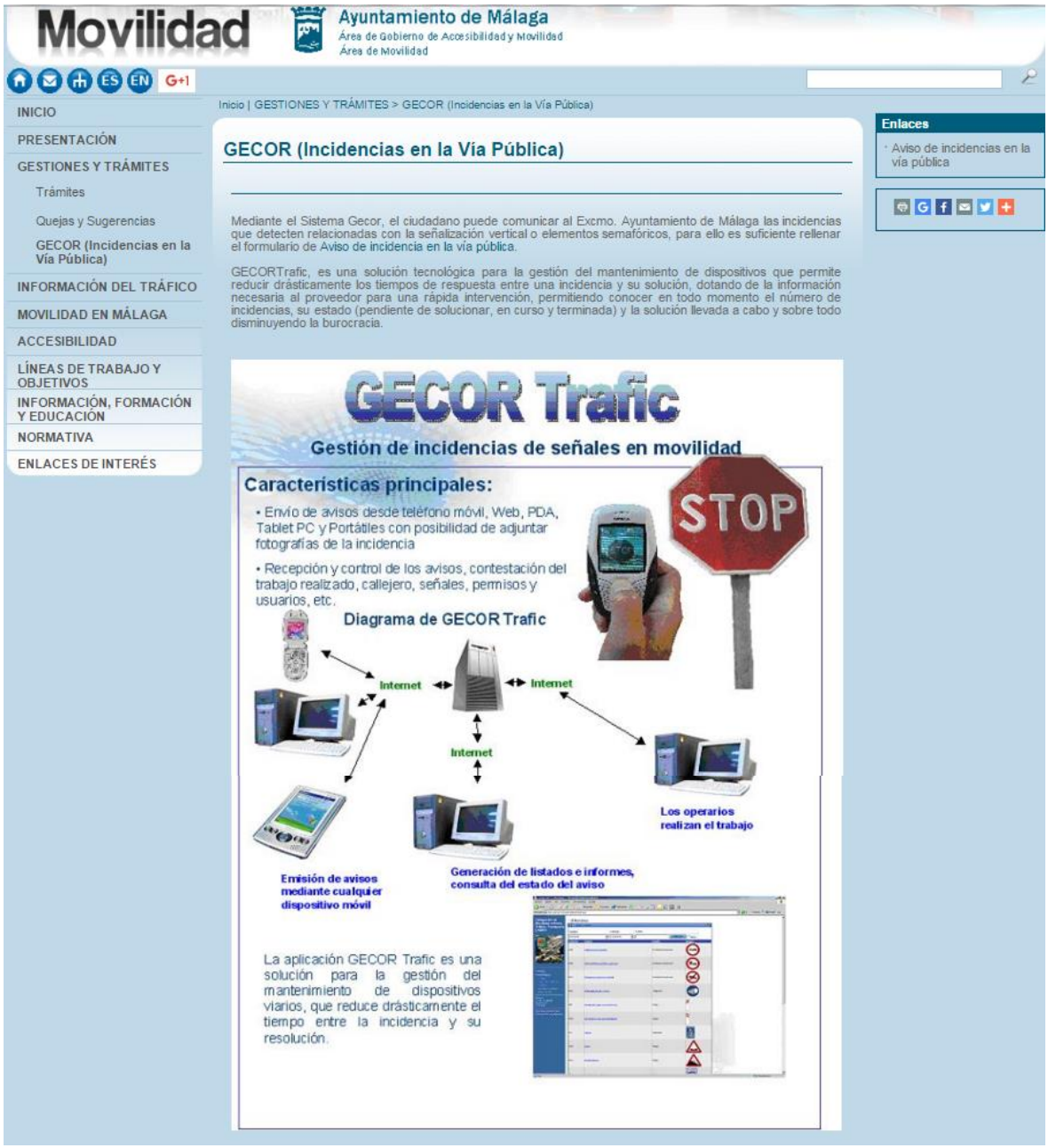


Figura 7: Aplicación GECOR del Ayuntamiento de Málaga. Inicio.

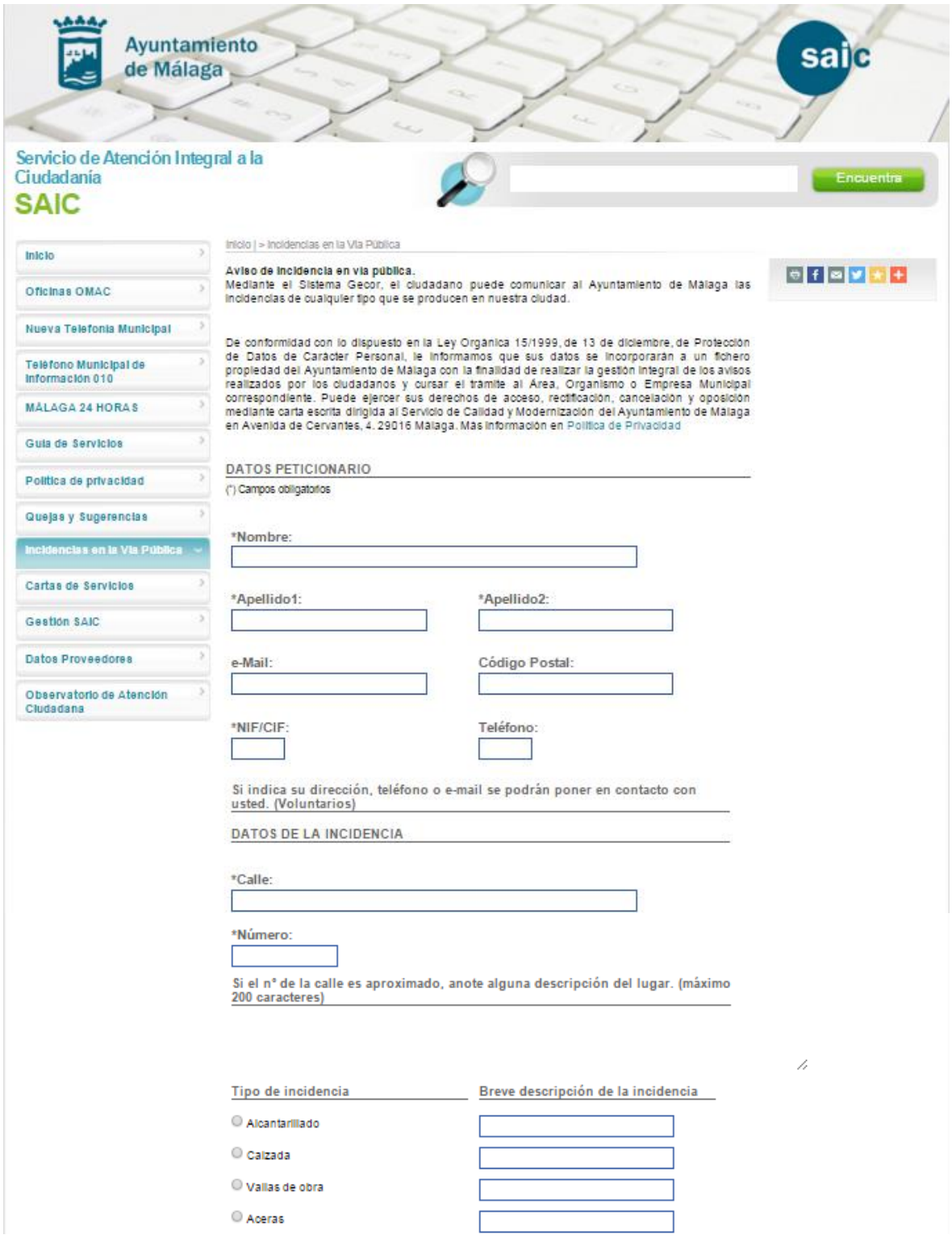


Figura 8: Aplicación GECOR del Ayuntamiento de Málaga. Nueva Incidencia.

SiG-O ciudadanos

El ayuntamiento de Vinaròs proporciona un sistema de comunicación de incidencias formado por aplicación móvil, que puede encontrarse en Google Play Store o App Store, y una aplicación web que puede usarse desde el navegador. Puede encontrarse más información en su dirección web [23].

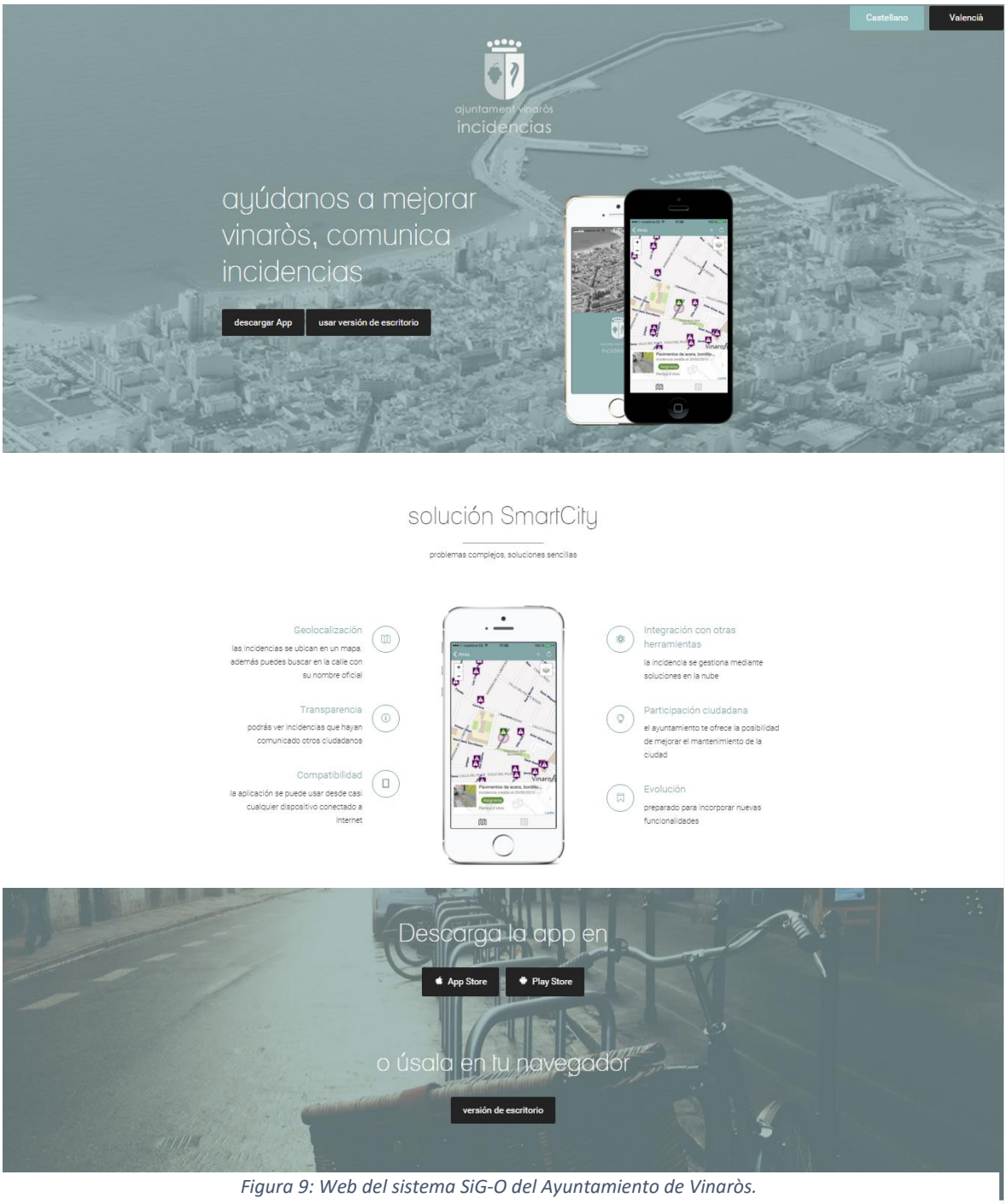


Figura 9: Web del sistema SiG-O del Ayuntamiento de Vinaròs.

Figura 10: Aplicación web del sistema SiG-O del Ayuntamiento de Vinaròs.

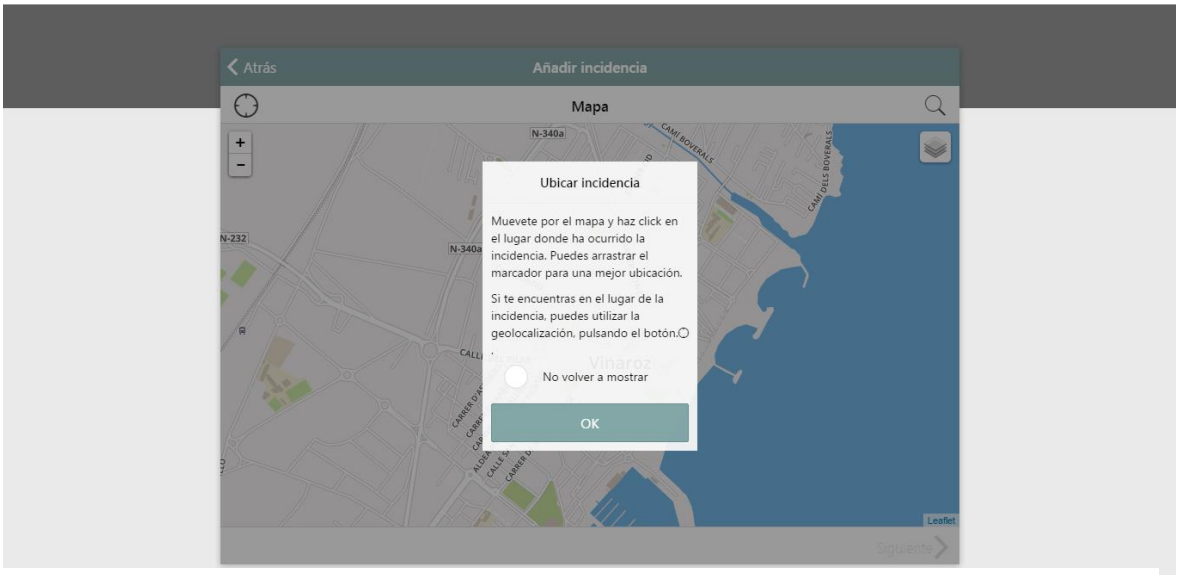


Figura 11: Aplicación web del sistema SiG-O. Añadir incidencia - Seleccionar ubicación.

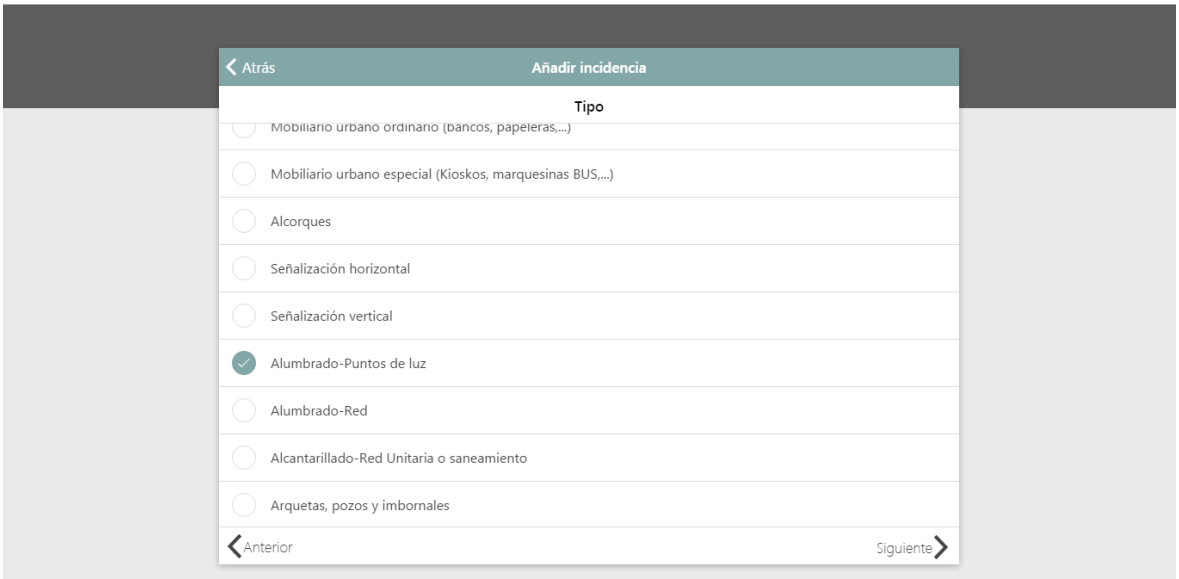


Figura 12: Aplicación web del sistema SiG-O. Añadir incidencia - Seleccionar tipo.

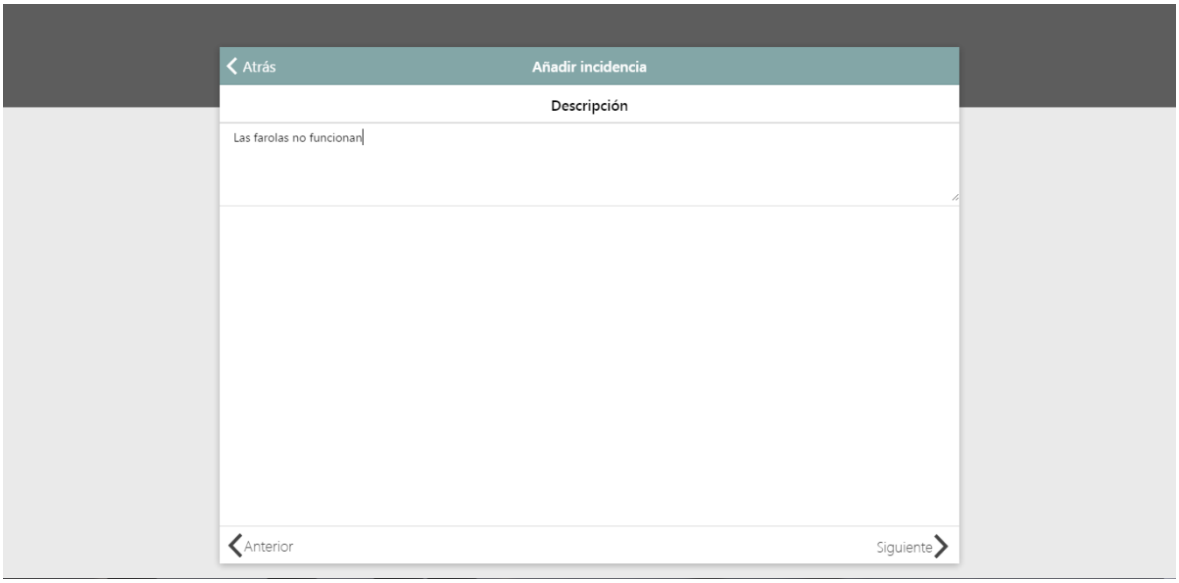


Figura 13: Aplicación web del sistema SiG-O. Añadir incidencia - Escribir descripción.

Aplicaciones móviles

La finalidad de los sistemas de notificación y gestión online de incidencias en la vía pública es poner al alcance de todo el mundo un servicio que permita una comunicación ágil con los organismos públicos, o empresas privadas, que deseen contar con la participación ciudadana para la mejora de su mantenimiento.

Cuando se desarrolla una herramienta de este tipo se enfoca como una aplicación online, principalmente, para hacer más accesibles estos sistemas. Cualquier ciudadano dispone hoy en día de conexión a Internet, ya que forma parte de nuestro estilo de vida. Una aplicación de gestión de incidencias en la vía pública cobra más sentido cuando puedes llevar contigo un dispositivo conectado a Internet y hacer uso de la aplicación en el momento y el lugar en el que se advierte un desperfecto.

Es por esto que la mayoría de sistemas de notificación de incidencias están dirigidos a los dispositivos móviles como *smartphones* o *tablets*, aunque muchos de ellos estén desarrollados como aplicaciones web. Como se ha visto anteriormente, existen muchos *frameworks* que permiten a los desarrolladores crear apps para teléfonos inteligentes de distintas plataformas, embebiendo la vista web en el dispositivo móvil. Esto permite crear una aplicación una vez y que funcione en casi cualquier sistema operativo móvil, en lugar de desarrollar la misma aplicación en código nativo para cada plataforma.

Realizando una búsqueda en Play Store, encontramos bastantes aplicaciones de gestión de incidencias, y todas ellas tienen características comunes a la hora de comunicar una incidencia:

- Seleccionar la tipología de la incidencia.
- Detectar la ubicación del usuario o la incidencia por medio de geolocalización.
- Tomar una fotografía.
- Incluir un comentario opcional.
- Realizar el seguimiento de la resolución de la incidencia.

Casi todas las aplicaciones encontradas en Play Store, pueden descargarse también para el sistema operativo IOS en App Store y/o tienen una versión web.

En las páginas siguientes, se presentan algunas de estas aplicaciones creadas por los ayuntamientos y empresas externas.

Mejora Bilbao

Es una plataforma de participación ciudadana móvil en tiempo real. El ayuntamiento de Bilbao ofrece esta plataforma gratuita, sencilla e intuitiva, que capacita a las personas para reportar incidencias, sugerencias y mejoras en asuntos como redes de agua, saneamiento, alumbrado, limpieza y gestión de residuos, anillo verde, parques y jardines, mobiliario urbano, vialidad, bicicletas y espacio público.

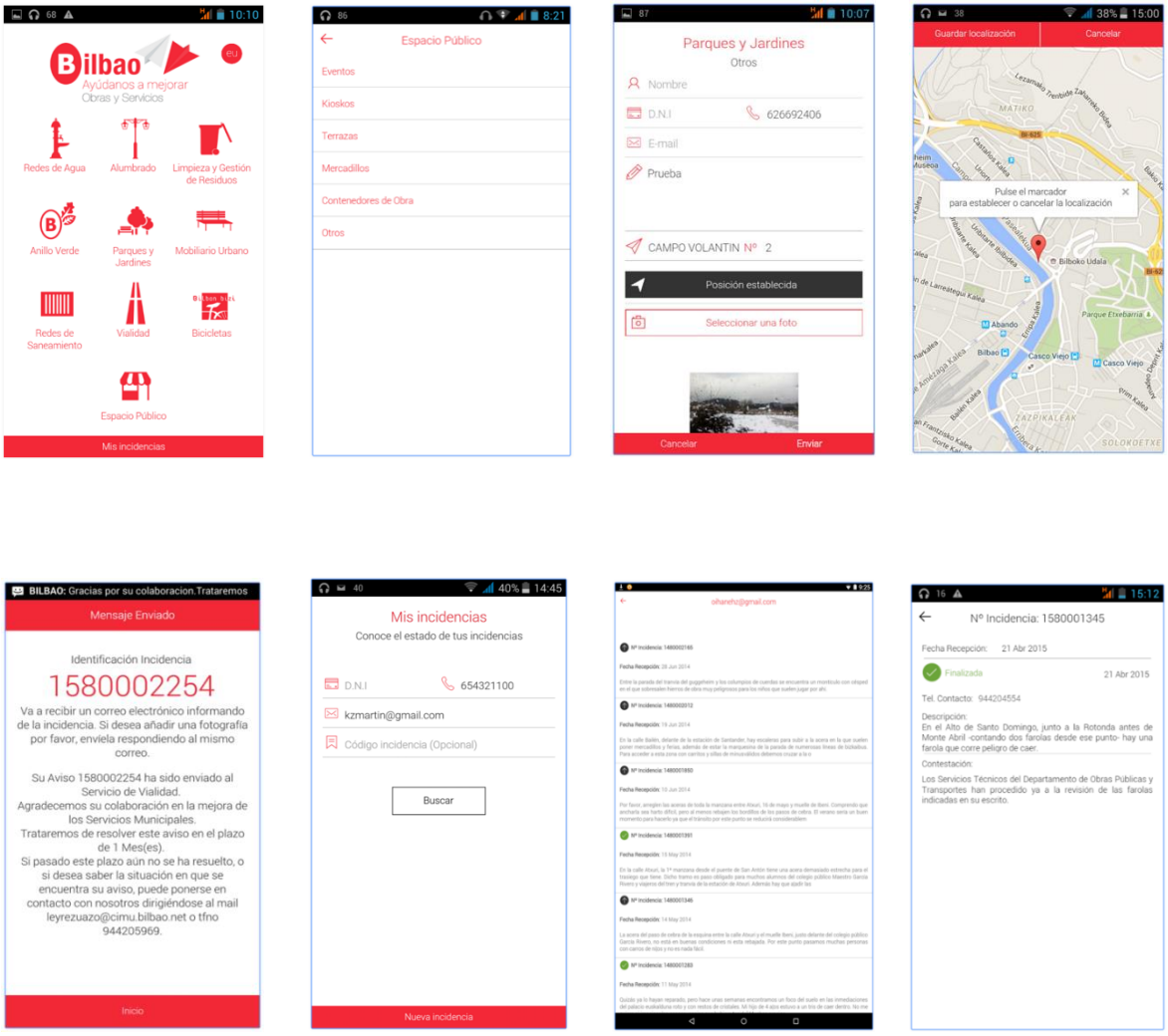
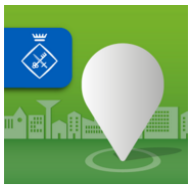


Figura 14: Pantallas de la app móvil "Mejora Bilbao".

Amb tu, un Prat millor



Ha sido creada por el Ayuntamiento de El Prat con el objetivo de facilitar la participación ciudadana para mejorar el mantenimiento de los espacios públicos y del mobiliario urbano de la ciudad. Permite la creación de avisos de incidencias detectadas por los usuarios y categorizar el tipo de incidencia, adjuntar o hacer una fotografía, geolocalizar la imagen en *Google Maps* y enviarla al departamento de Mantenimiento y Servicios del Ayuntamiento de El Prat. Se enviará un correo electrónico para informar de que la incidencia ha sido recibida y su resultado.

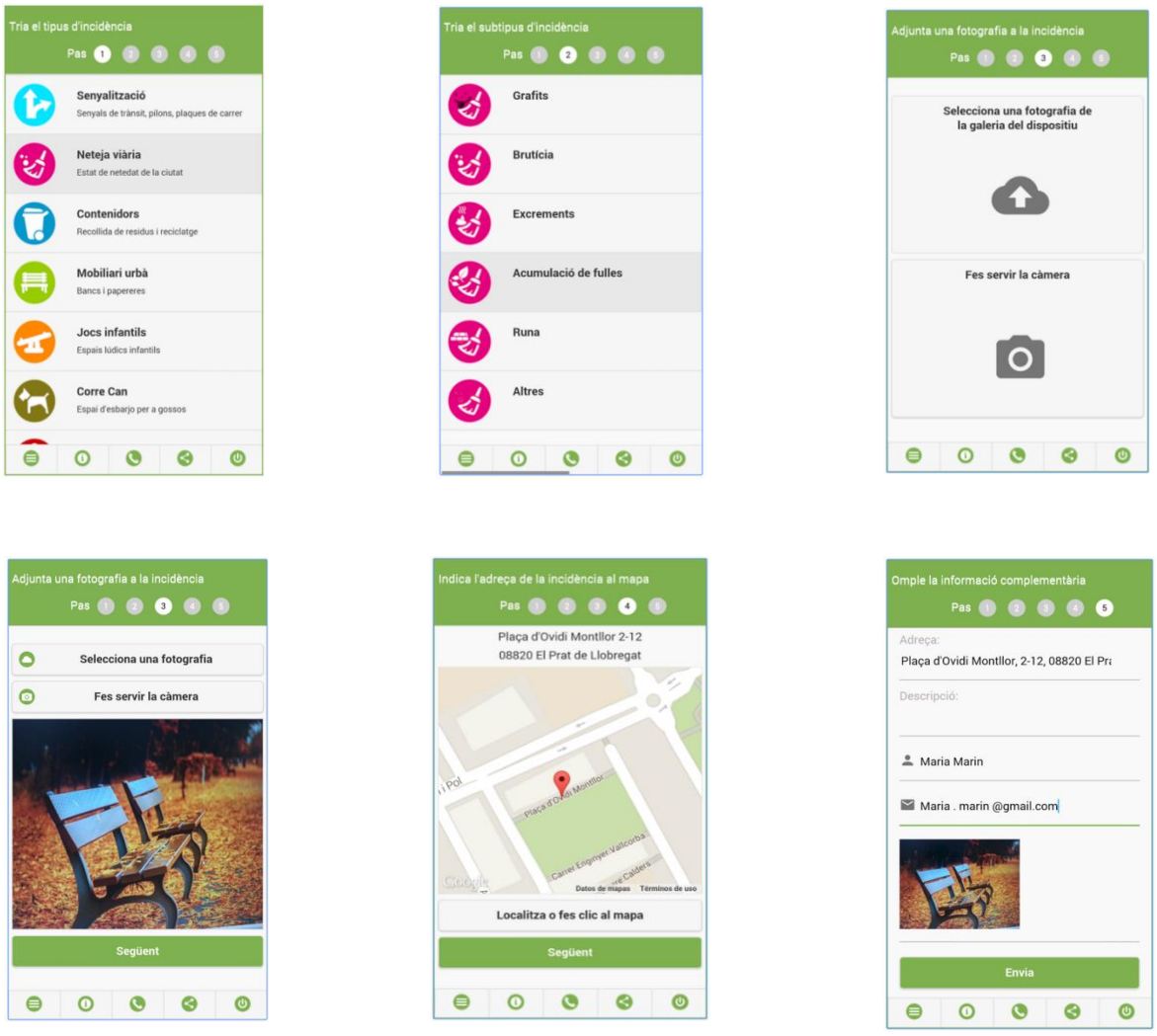


Figura 15: Pantallas de la app móvil "Amb tu, un Prat millor".

Bezana al día

Avisar de la rotura de una acera, el mal estado de una calle o la caída de una señal de tráfico ya es más fácil con El Sistema de Gestion de Incidencias **“Bezana al Dia”.** El Ayuntamiento de Santa Cruz de Bezana ha puesto a disposición de los vecinos una aplicación para móviles que permite comunicar directamente las incidencias en la vía pública para que éstas sean resueltas. El usuario puede seleccionar el tipo de incidencia o sugerencia que quiere comunicar y después, mediante geolocalización por GPS, podrá conocer su posición y seleccionar el elemento de la infraestructura pública en mal estado, y se le ofrecerá la posibilidad de fotografiar la misma. Podrá añadir comentarios sobre ella y enviarla. Automáticamente, los servicios municipales recibirán la notificación.

Todas las notificaciones que se reciban serán clasificadas y se determinará si la intervención debe ser urgente o no. La persona que ha comunicado la incidencia podrá revisar en este programa el estado de la misma, es decir, si está resuelta o en proceso.

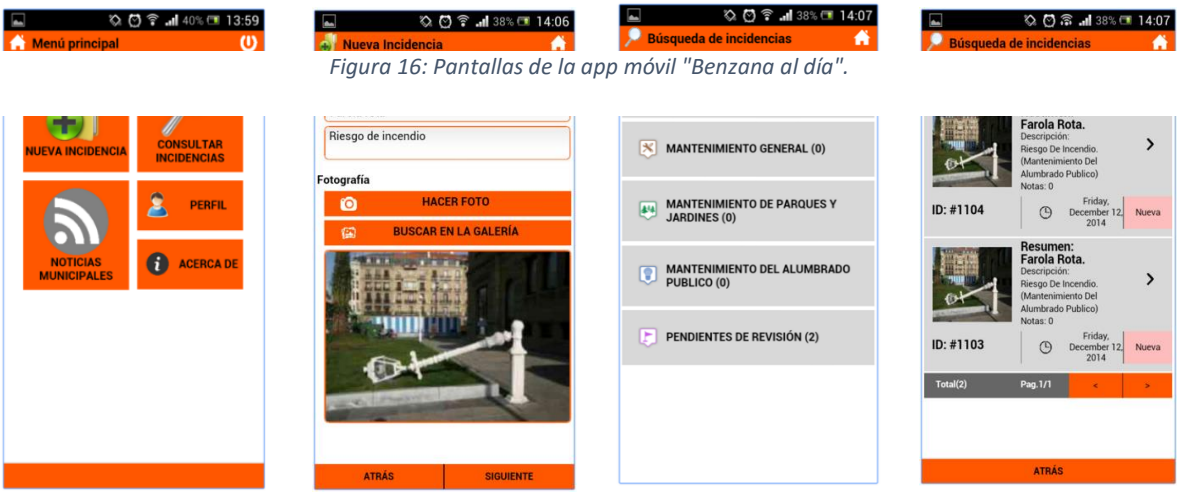


Figura 16: Pantallas de la app móvil "Benzana al día".

“Cuida Los Barrios” es una aplicación móvil que pone a disposición de los ciudadanos de Los Barrios, Cádiz, un sistema de gestión de incidencias en la vía pública. Está desarrollado por el mismo equipo de desarrollo que “Bezana al día”, la anterior aplicación descrita, y tiene las mismas funcionalidades y un diseño muy parecido, aunque se adapta al ayuntamiento de este municipio.

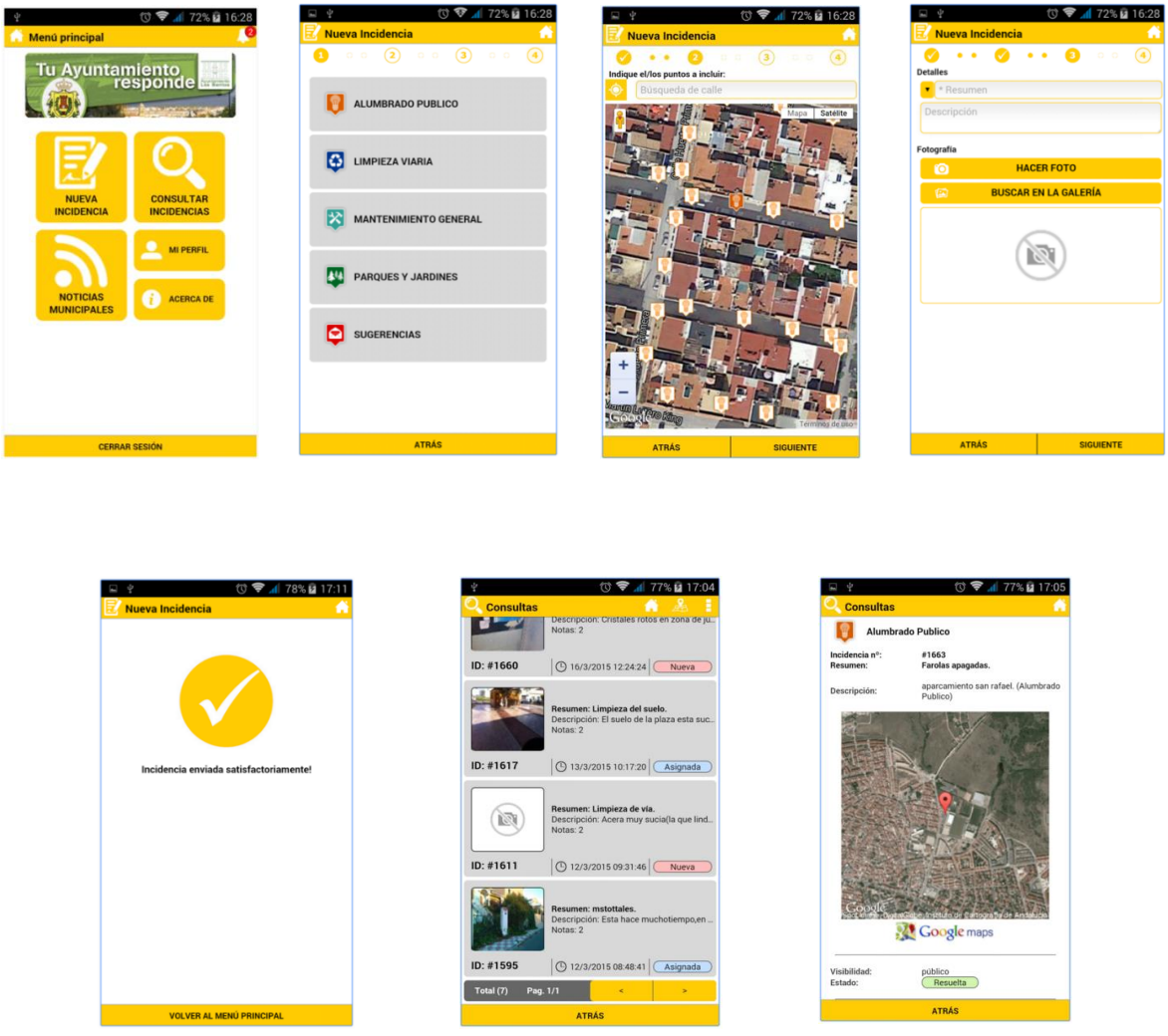


Figura 17: Pantallas de la app móvil "Cuida Los Barrios".

Aplicación Emulsa

Aplicación de la Empresa Municipal de Servicios de Medio Ambiente Urbano del **Ayuntamiento de Gijón (EMULSA)** para el envío de incidencias y avisos desde dispositivos móviles. Gracias a esta *app* los vecinos de Gijón pueden avisar desde su propio teléfono móvil o *tablet* a los técnicos de EMULSA de las incidencias y desperfectos que detecten tanto en la vía pública como en el equipamiento urbano. Los usuarios podrán llevar una gestión en sus teléfonos de los avisos que han enviado, así como de las incidencias que ya se han resuelto. Se podrá adjuntar una fotografía tomada desde el *Smartphone* y las coordenadas GPS.

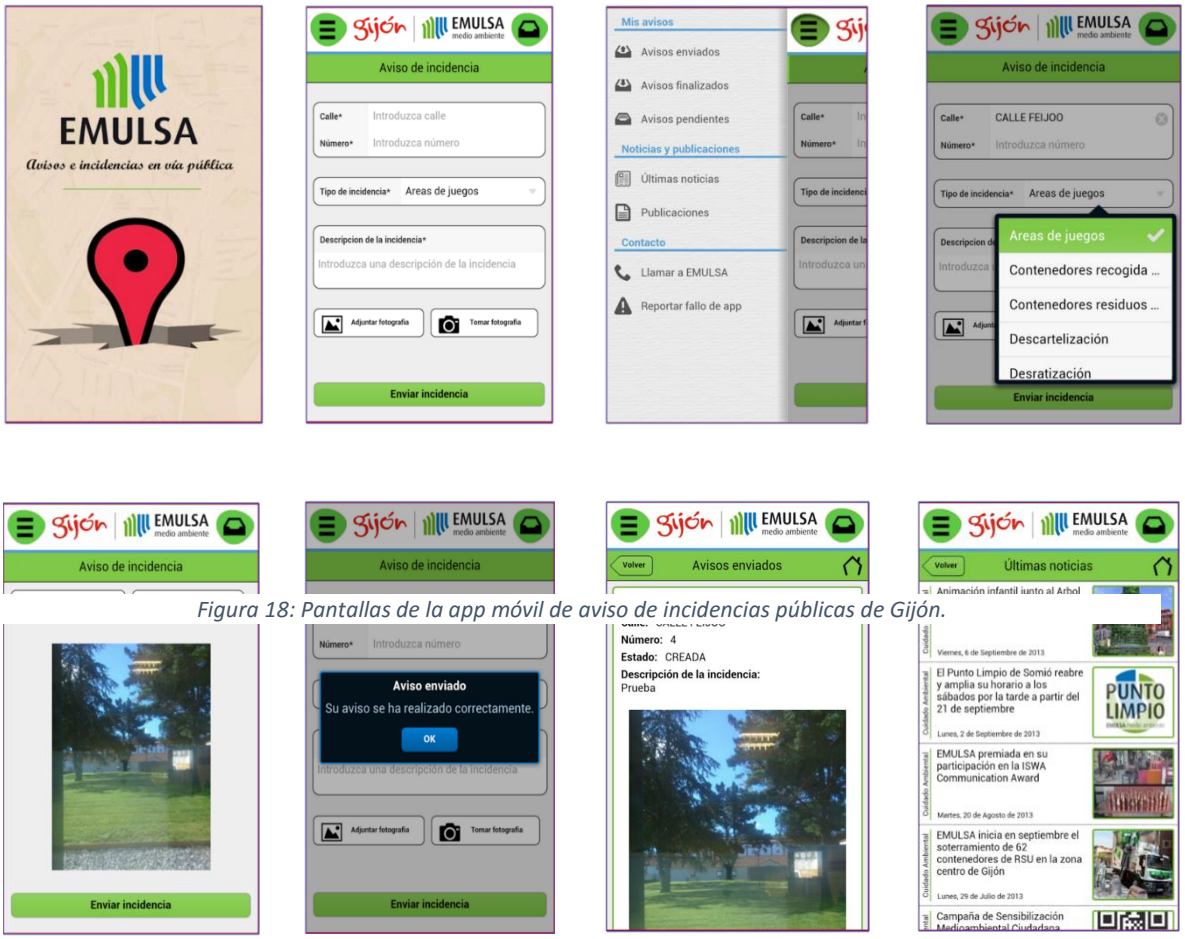


Figura 18: Pantallas de la app móvil de aviso de incidencias públicas de Gijón.

Valladolid en tu mano

Es la aplicación oficial del Ayuntamiento de Valladolid para reportar incidencias, sugerencias o reclamaciones sobre la ciudad a través de dispositivos móviles. Ofrece a la ciudadanía la posibilidad de comunicar sugerencias e incidencias sobre la vía pública, mobiliario urbano, infraestructuras, parque y jardines, etc. indicando para cada una de ellas diversos aspectos como tipología, coordenadas de localización, fotografías y descripción de la incidencia/sugerencia. Todas las comunicaciones que se realicen desde la aplicación se integran directamente con el sistema de gestión de sugerencias y reclamaciones del Ayuntamiento de Valladolid para su correcta atención.

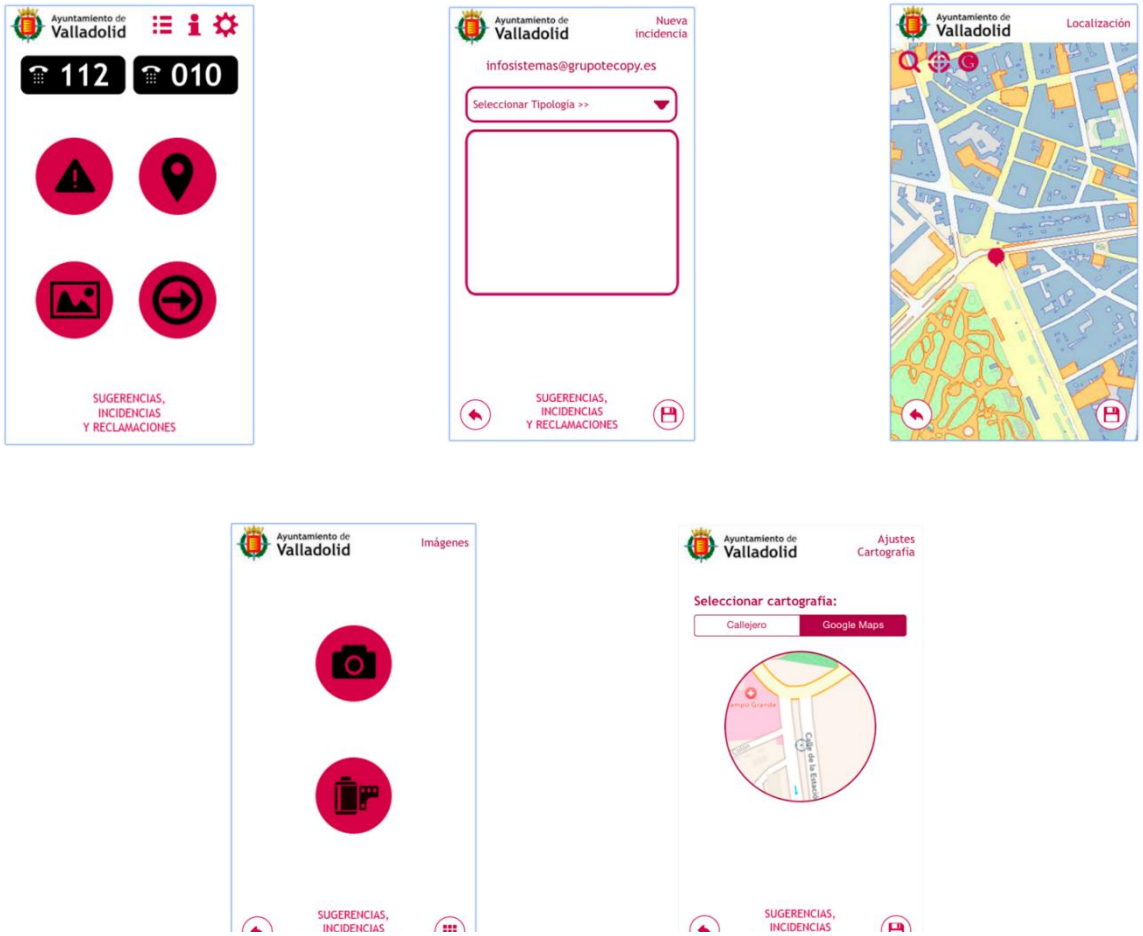


Figura 19: Pantallas de la app móvil "Valladolid en tu mano".

Es una aplicación que permite a cualquier ciudadano comunicar incidencias, sugerencias y quejas relacionadas con el espacio público de su municipio.

Todas las alertas permitirán ubicar en mapa el lugar donde está la incidencia, así como incluir imágenes, vídeos o cualquier documentación que el ciudadano crea oportuna para mejorar o documentar la alerta que se tramita.

El ciudadano podrá hacer un seguimiento de la alerta que ha comunicado, y saber en cada momento si se está tratando y en qué fecha quedará resuelta.

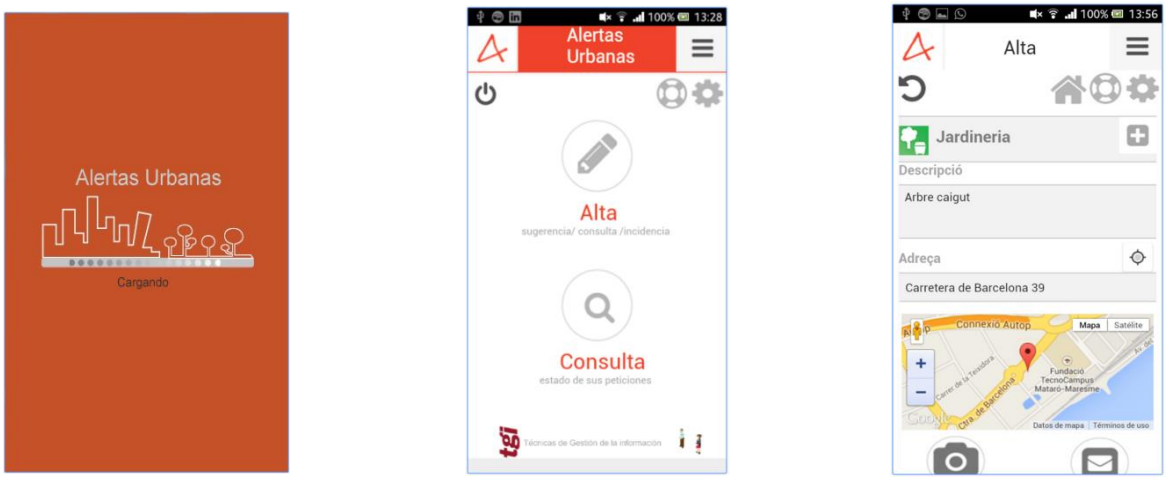


Figura 20: Pantallas de la app móvil "Alertas Urbanas".

Mejora Móstoles

Es una app singular de implicación ciudadana con el objetivo de que, entre todos, se siga mejorando día a día la ciudad. De forma gratuita esta aplicación te permite interactuar con el Ayuntamiento en tiempo real, indicando incidencias en la vía pública y sin tener que desplazarse hasta las dependencias municipales. A través de ella se podrá, de forma sencilla, enviar una fotografía con una breve descripción de la incidencia y su topología para que desde el Ayuntamiento se valore y se adopte la actuación pertinente.

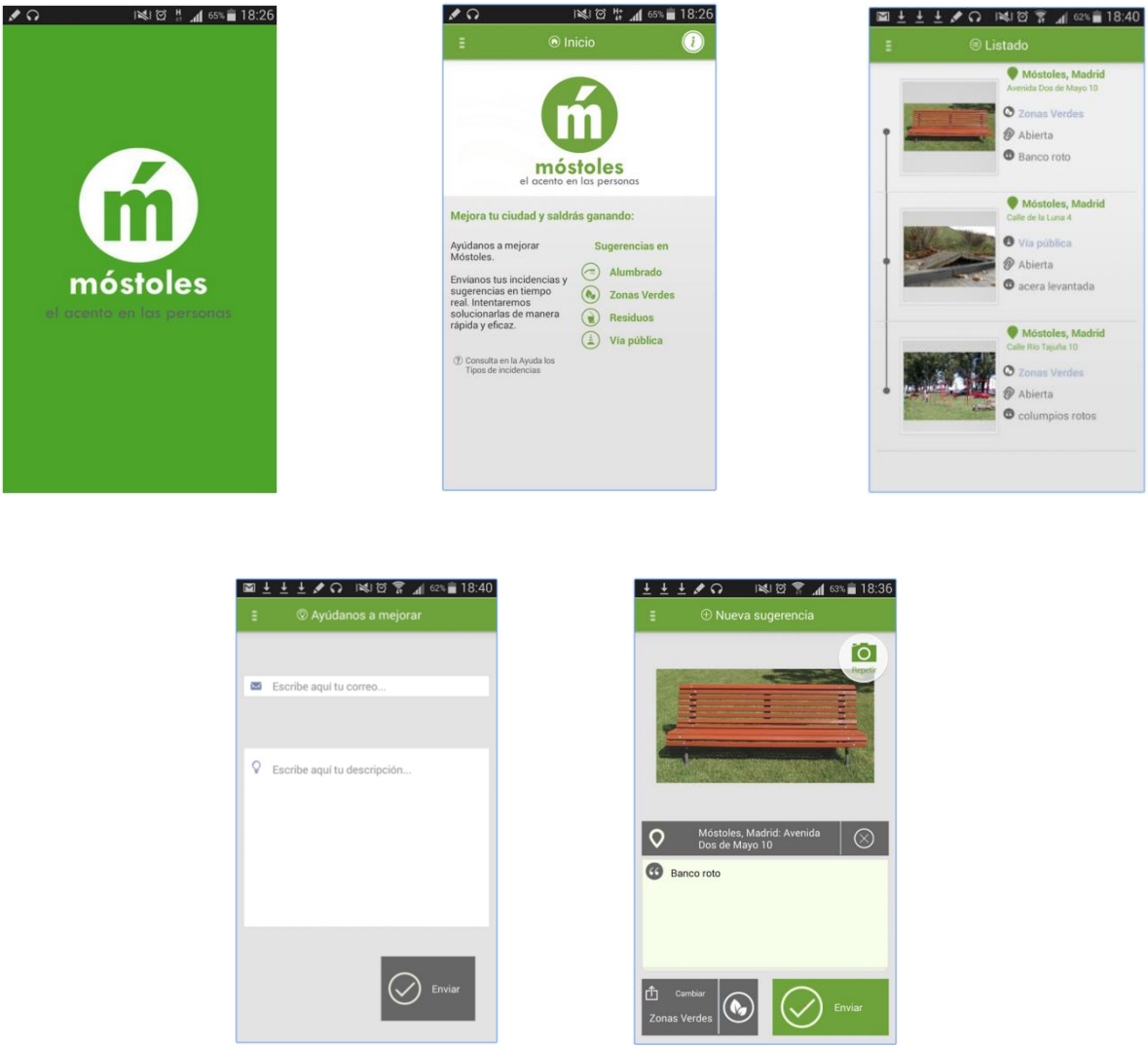


Figura 21: Pantallas de la app móvil "Mejora Móstoles".

Línea Madrid

Es una aplicación móvil del servicio de atención a la ciudadanía del Ayuntamiento de Madrid para el envío de avisos y peticiones sobre incidencias en vías y espacios públicos y zonas verdes del Municipio de Madrid. Una vez descargada e instalada la aplicación, es preciso registrarse introduciendo datos básicos de contacto.

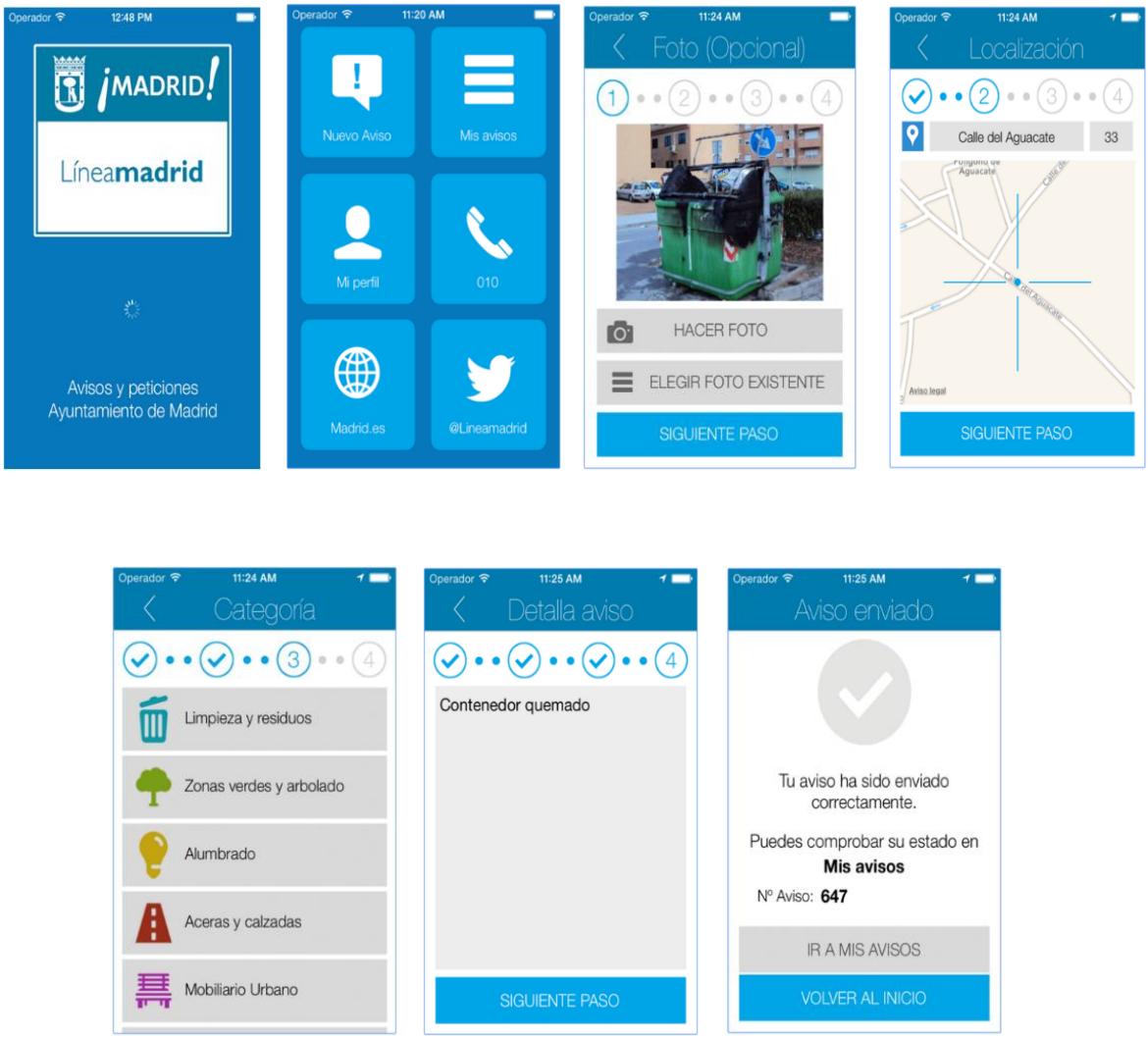


Figura 22: Pantallas de la app móvil "Línea Madrid".

Arroyo Digital

Es la app oficial del Ayuntamiento de Arroyo de la Encomienda que a través de dispositivos iOS y Android, proporciona un canal de comunicación entre el Ciudadano y la Administración disponible desde cualquier lugar, 24 horas al día, 365 días al año, permitiendo una mayor participación y una agilización de la gestión municipal. Permite comunicar al Ayuntamiento incidencias que detecte en la vía urbana, especificando su localización mediante GPS y adjuntar fotografías y vídeos. Se puede recibir información de cómo va la incidencia.

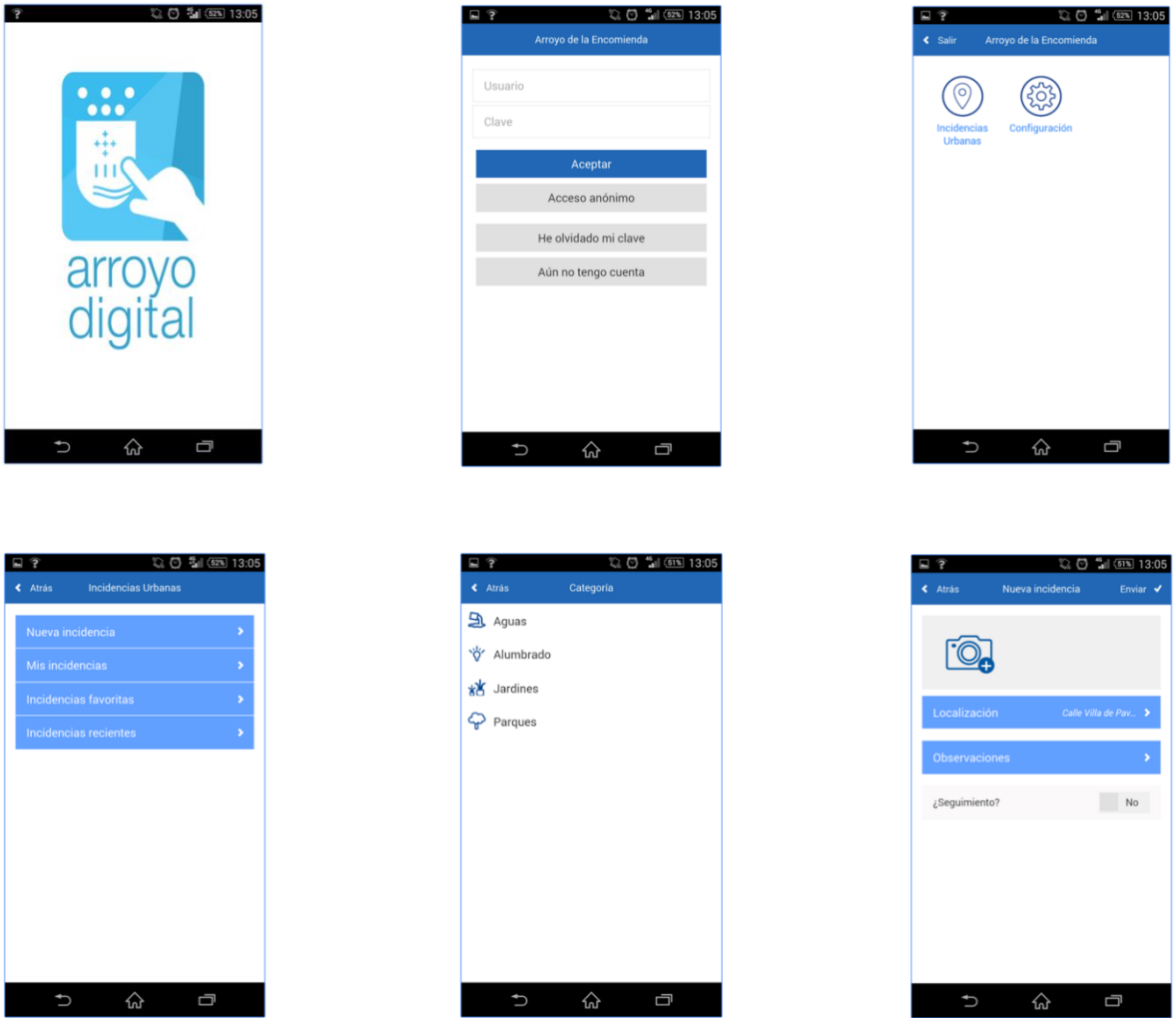


Figura 23: Pantallas de la app móvil "Arroyo digital".

Smartpolis: Ayuntamiento de Camuñas

La aplicación **"Smartpolis"** es la app de gestión de incidencias municipales para el Ayuntamiento de Camuñas, en la provincia de Toledo.

Figura 24: Pantallas de la app móvil "Smartpolis".

Análisis de tecnologías

Aplicación web

Desde hace un tiempo, la integración de la sociedad con las nuevas tecnologías es completa, e Internet parece ser una extensión de nosotros mismos donde buscar soluciones a múltiples problemas del día a día. En gran medida esto es gracias a la web y a la aparición de aplicaciones tipo webmail, chats, blogs, foros, redes sociales, etc. Esto nos ha hecho acostumbrarnos a utilizar este tipo de aplicaciones a través del navegador de Internet.

Una aplicación web es una aplicación *Cliente/Servidor* donde la lógica de la misma se distribuye entre el servidor y el cliente a través de un canal de transmisión de datos. Los datos se almacenan principalmente en el servidor. Es una herramienta completamente funcional, codificada para ser soportada por los navegadores, sin necesidad de nada más que una conexión a Internet. Con el paso del tiempo, se han ido convirtiendo en sistemas complejos con interfaces de usuario tan usables como las de las aplicaciones de escritorio.

En consecuencia, el uso de aplicaciones basadas en web va en aumento y no es extraño empezar a ver que en el ámbito empresarial también se utilizan. Presentan una serie de ventajas y beneficios con respecto al software de escritorio, con los que se logra aprovechar los recursos de una empresa de una forma mucho más práctica que el software tradicional.

- Mayor **facilidad** a la hora de **trabajar a distancia**.
- Para trabajar a través de una aplicación web **solo se necesita un dispositivo conectado a Internet** y un navegador web.
- **Disponibilidad total de horarios.**
- **No es necesario tener conocimientos** previos de informática.
- Las aplicaciones web permiten **centralizar todas las áreas** de trabajo.
- Facilitan el **trabajo colaborativo**.

También ha ido cambiando la forma en que el software se distribuye. Con el uso extendido de las aplicaciones web han ido proliferando los servicios en la nube. Los denominados software como servicio, ScuS, (*Software As a Service*, *SaaS*) son un modelo de distribución de software donde se utiliza una aplicación desde un dispositivo cliente hacia un servidor de la empresa que provee el servicio.

La forma tradicional de distribución de software siempre ha sido la instalación directa en equipos del cliente, a través de comprar el producto físico, un CD, o descargar e instalar un programa. Pero la evolución de las necesidades a través del tiempo ha hecho que modelar software como servicio simplifique estos requerimientos. Las aplicaciones web no requieren canales de distribución, lo que permite que su precio sea inferior que el de los programas instalables. El uso empresarial de los servicios en la nube suele ir dirigido a que los usuarios paguen una cuota mensual o anual para utilizarlos según las funciones o el número de usuarios.

Los servicios online en aplicaciones web son tan populares debido a que es muy práctico que el navegador web se utilice como cliente ligero. Esto implica una serie de ventajas respecto a las aplicaciones de escritorio:

- **Independientes del sistema operativo**: usables desde cualquier plataforma o sistema operativo disponiendo únicamente de un navegador.
- **Ahorro de tiempo:** sin descargas e instalaciones.
- **Ahorro de costes en hardware y software.**
- **Fáciles de usar:** se suele tener experiencia utilizando aplicaciones web. Prácticamente todo el mundo sabe escribir un correo electrónico o utilizar las redes sociales.
- **Escalables y con actualizaciones instantáneas:** se usa siempre la versión más reciente que proporciona el desarrollador.
- **Portables**: independiente del tipo de dispositivo desde el que se acceda.
- **Sin problemas de compatibilidad:** es suficiente con disponer de un navegador actualizado.

Aunque también tiene inconvenientes:

- **Funcionalidades algo más limitadas.**
- **Dependencia de conexión a Internet.**

Las metodologías de desarrollo software se basan o tienen un enfoque más general. Estos enfoques pueden desarrollarse en más de una metodología específica. Cada enfoque tiene unas características y puede ser adecuado para un tipo de proyecto u otro.

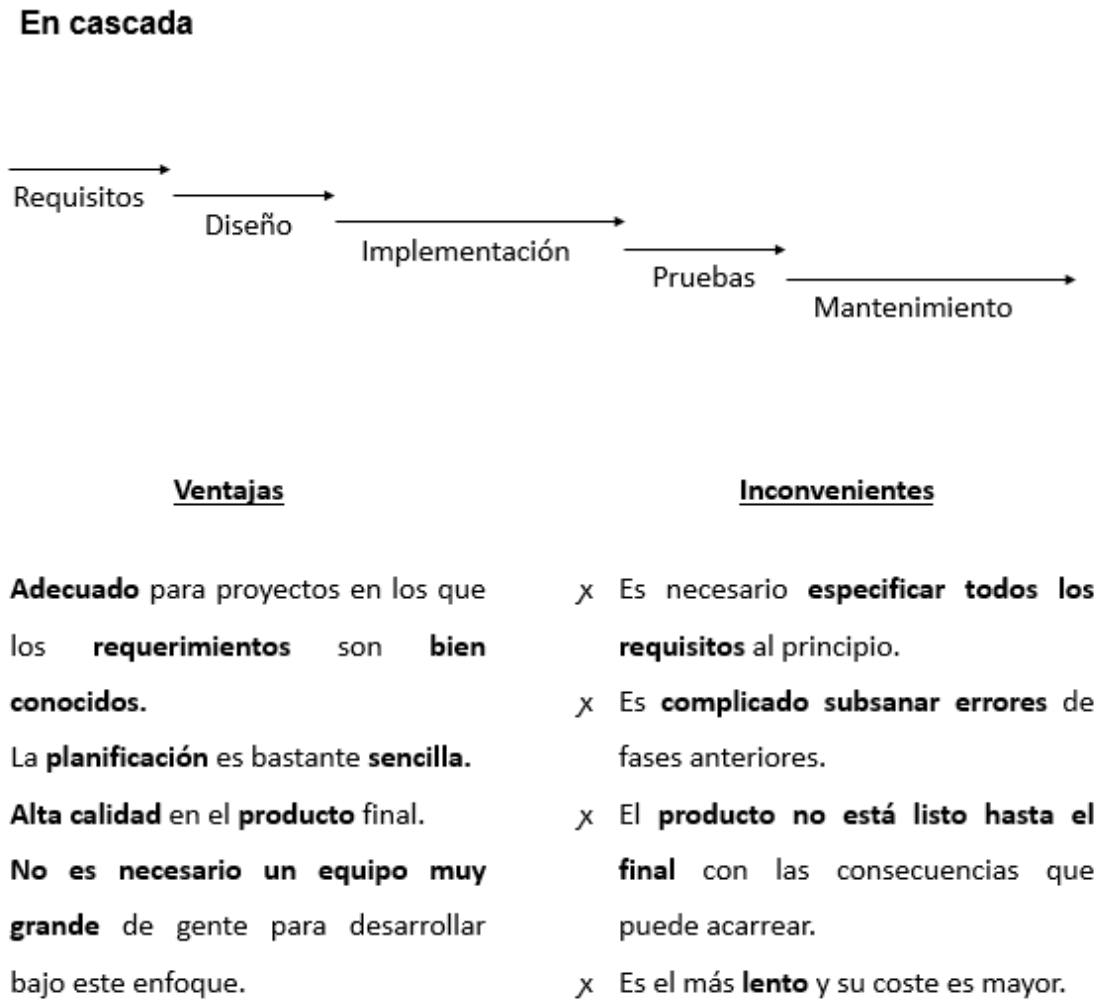
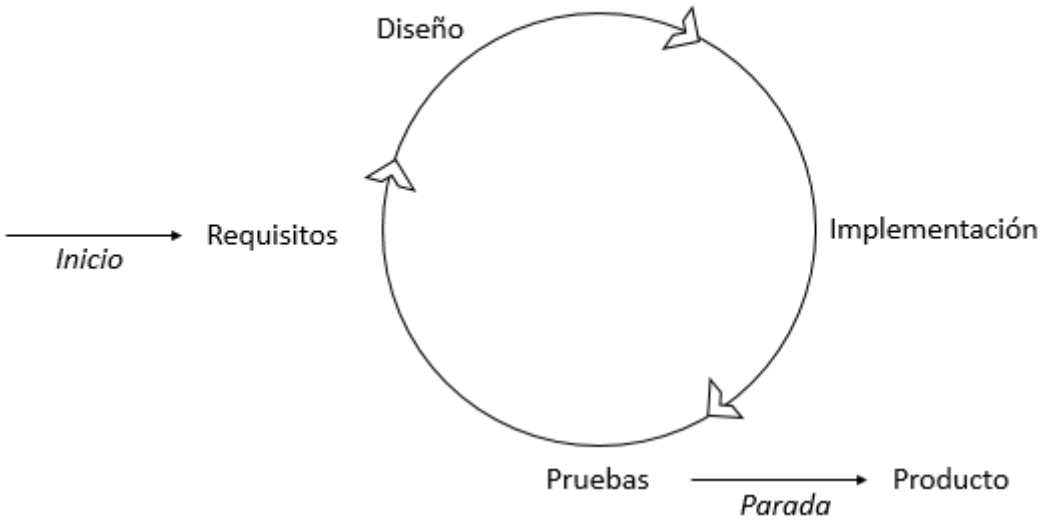


Figura 25: Enfoque en cascada. Ventajas e inconvenientes. Elaboración propia.

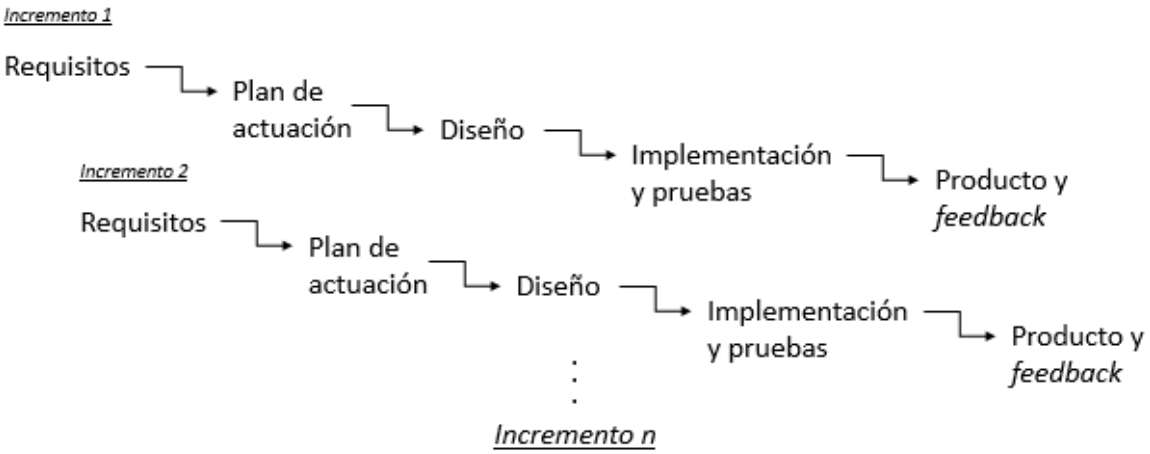
Prototipos



Ventajas	**Inconvenientes**
✓ Este modelo es útil cuando se conocen solo **los objetivos generales del proyecto.**	x Al crear un prototipo de forma rápida **se tienden a desatender algunas cuestiones** que deben abordarse en el siguiente prototipo.
✓ **Reduce el riesgo de construir productos que no satisfagan las necesidades.**	x **Exige disponer de las herramientas adecuadas.**
✓ **Reduce costos.**	x Es necesario un ***feedback*** **constante** con el cliente.

Figura 26: Enfoque basado en prototipos. Ventajas e inconvenientes. Elaboración propia.

Incremental



Ventajas

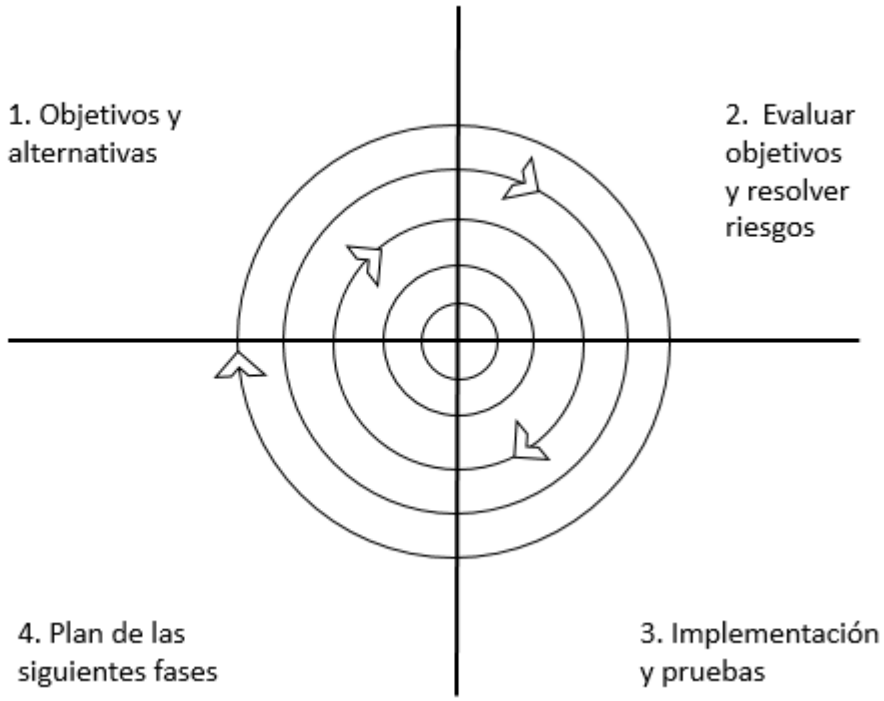
- ✓ El **primer incremento no necesita mucho personal.**
- ✓ Se **reduce el tiempo de desarrollo** inicial.
- ✓ **Entrega temprana** de partes operativas.
- ✓ Proporciona todas **las ventajas del modelo en cascada realimentado**.

Inconvenientes

- x **Con cada incremento se necesita más personal.**
- x **No es recomendable** para sistemas en tiempo real, con un nivel de seguridad alto o de procesamiento distribuido.
- x Requiere **mucha planificación.**
- x Hay que poner **metas claras.**
- x Tiene las **desventajas del modelo en cascada** pero a nivel de incremento.

Figura 27: Enfoque incremental. Ventajas e inconvenientes. Elaboración propia.

Espiral



Ventajas

- ✓ Es un **proceso adaptable**.
- ✓ Puede **aplicarse a lo largo de la vida del software.**
- ✓ **Permite aplicar** cualquiera de los otros enfoques.
- ✓ **Elimina errores y alternativas no adecuadas** al principio.

Inconvenientes

- x Requiere una **gran habilidad para la evaluación de riesgos.**
- x Si algún **riesgo no es detectado** a tiempo puede implicar **muchos problemas.**
- x **El éxito del producto depende de la evaluación de estos riesgos.**

Figura 28: Enfoque en espiral. Ventajas e Inconveniente. Elaboración propia.

Los patrones de arquitectura de software describen los elementos y el tipo de relación que tienen entre sí, atendiendo a unas normas sobre cómo usarse. Suelen utilizarse para favorecer la calidad del producto que se está desarrollando (solucionar problemas de rendimiento, seguridad, disponibilidad…) y es en la fase de diseño cuando se decide que patrón usar para satisfacer los requerimientos deseados para el sistema.

Según la escala o nivel de abstracción en la arquitectura de un sistema, podemos hablar de que existen patrones arquitectónicos, que indican el esquema de organización estructural de un sistema y patrones de diseño, que expresan esquemas para definir el diseño con las que construir sistemas software.

Ejemplos de **patrones arquitectónicos** son:

- ***Peer to Peer,*** donde una serie de objetos interaccionan y no hay distinción entre un proveedor de servicios y el usuario de los mismos.
- ***Pipeline*** es una cadena de procesos, hilos, rutinas y funciones dispuestos de manera que la salida de cada elemento es la entrada del siguiente y la información fluye a través de cada uno de los componentes. En algunas partes de este proceso es necesario aplicar algunos filtros para transformar la información.
- **Programación en capas** en *Cliente-Servidor*, donde se separa la lógica de negocio de la lógica de diseño. Cada capa corresponde a una funcionalidad del sistema. Normalmente se organizan en una estructura jerárquica o de árbol.

El desarrollo de una aplicación se puede llevar a cabo en varios niveles. Teniendo la capa de presentación, la capa de negocio y la capa de datos, podemos distribuirlas físicamente en niveles en función de la complejidad de cada capa.

En las siguientes imágenes se ha abordado el tema de las características deseables que cumplirían sistemas, como el de este proyecto, desarrollados bajo una arquitectura de programación en capas, distribuidas en dos y tres niveles con más o menos peso de trabajo en cada capa.

Cliente ligero / Servidor pesado

Capacidad de respuesta y usabilidad:	★ ☆ ☆ ☆ ☆	Gran cantidad de datos transferidos entre servidor y cliente. No es aplicable al segmento móvil.
Velocidad de desarrollo:	★ ★ ★ ★ ★	Se puede desarrollar en cualquier lenguaje de la parte del servidor y para diferentes necesidades.
Escalabilidad:	★ ★ ★ ★ ☆	Relativamente favorable pero puede requerir un balanceo de carga.
Rendimiento:	★ ★ ★ ☆ ☆	Bajo, debido a la gran cantidad de datos que tienen que ser transferidos. Se generan las páginas desde cero.
Pruebas:	★ ★ ★ ★ ☆	No hay necesidad de herramientas especiales para poner a prueba el *front-end* puesto que *javascript* se interpreta bien.
Paso a aplicación móvil:	☆ ☆ ☆ ☆ ☆	Si el servidor se realiza sobre Node.js y la base de datos no es demasiado grande, podría ser posible.
Uso offline de la aplicación:	★ ★ ☆ ☆ ☆	Solo si las páginas se cachean en el navegador y en caso de desconexión, al usuario se le muestran de esa forma.
Seguridad:	★ ★ ★ ★ ☆	Aunque la lógica se trabaja en el servidor, es recomendable que el canal sea lo más seguro posible.

Figura 29: Esquema y características de una aplicación desarrollada en dos niveles. Elaboración propia.

Servidor más ligero. El cliente asume más funcionalidades.

Capacidad de respuesta y usabilidad:	★ ★ ★ ☆ ☆	Mayor capacidad de respuesta porque la carga de la página es parcial, aunque la primera carga es más lenta. La navegación es más agradable.
Velocidad de desarrollo:	★ ★ ★ ☆ ☆	Necesidad de conocer las tecnologías del lado del cliente, a parte de las del servidor.
Escalabilidad:	★ ★ ★ ★ ☆	Relativamente favorable pero puede requerir un balanceo de carga.
Rendimiento:	★ ★ ★ ★ ☆	El tiempo dedicado a la generación de contenido HTML es menor pero se añade un nivel de abstracción a la aplicación con los JSON.
Pruebas:	★ ☆ ☆ ☆ ☆	Se deben *testear* el servidor, el código cliente y el servidor web.
Paso a aplicación móvil:	☆ ☆ ☆ ☆ ☆	Si el servidor se realiza sobre Node.js y la base de datos no es demasiado grande, podría ser posible.
Uso offline de la aplicación:	★ ☆ ☆ ☆ ☆	Hay dificultades para *cachear* las páginas. Deben indicarse las páginas que van a guardarse desde el servidor.
Seguridad:	★ ★ ★ ★ ☆	Al desplazarse la parte lógica al cliente, puede ser modificada. Necesidad de canal seguro.

Figura 30: Esquema y características de un sistema en tres niveles con un servidor más ligero. Elaboración propia.

Cliente pesado / servidor ligero

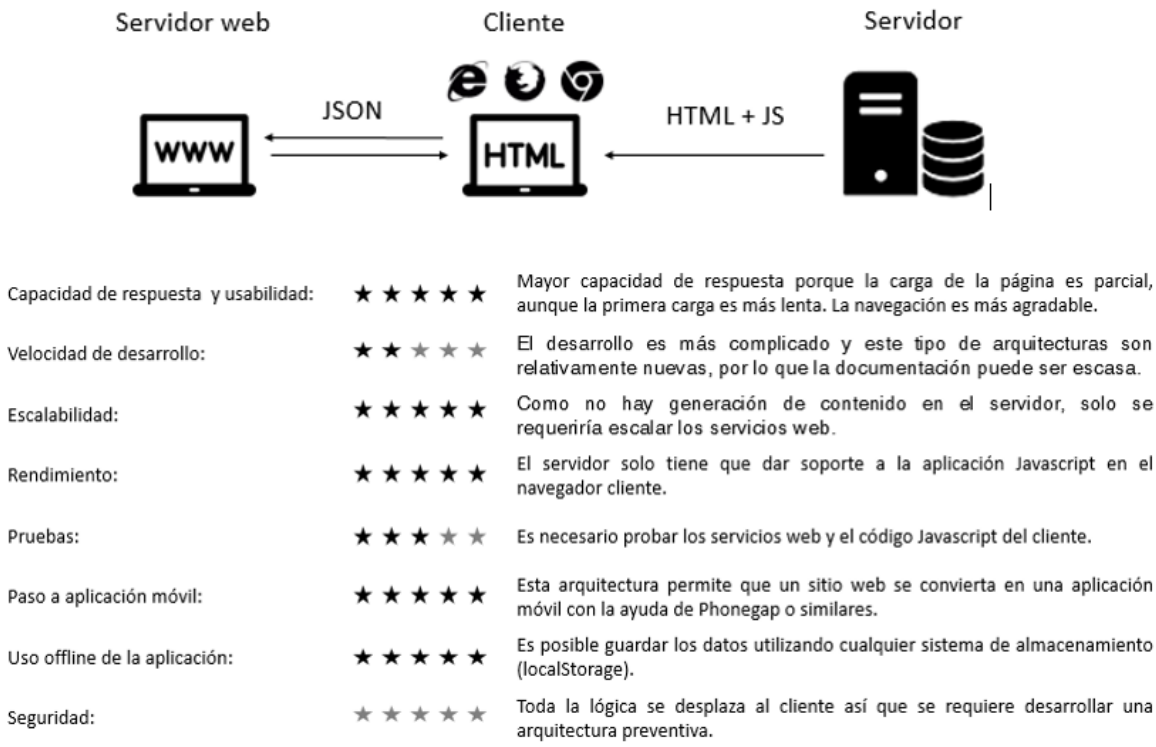


Figura 31: Esquema y características de una aplicación desarrollada en tres niveles con servidor delgado. Elaboración propia.

- **Arquitectura orientada a eventos.** Un sistema construido alrededor de esta estructura podría ser un sistema de correo electrónico, un sensor o un sistema de ventas. Los eventos son cambios que se producen en el estado de los elementos del sistema. El sistema suele estar compuesto de emisores que "disparan" las notificaciones de que algo ha cambiado, y de consumidores de eventos, que actúan de "oyentes" a la espera de identificar un evento y reaccionar. Un evento puede desencadenar que se envíen unos emails, se cambie el estado de un producto que se ha terminado a "sin stock" o se lance un mensaje de "botón activado".

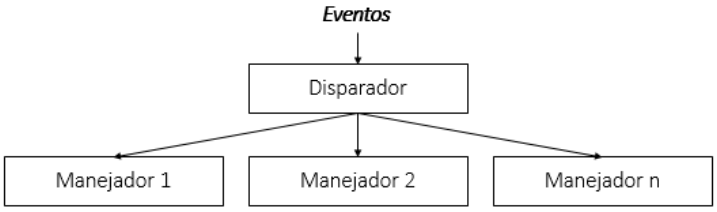


Figura 32: Elementos de una arquitectura orientada a eventos. Elaboración propia.

Ventajas: simplicidad, modularidad y eficiencia.
Desventajas: escalabilidad imprevisible, posibilidad de desborde y posible no respuesta.

- **Arquitectura en pizarra.** Este patrón consta de varios elementos funcionales denominados agentes y un instrumento que controla todo denominado pizarra. Las tareas de cada agente están en la pizarra, y cuando se ha realizado la tarea, se ponen los resultados en la misma. Por tanto, el estado inicial de la pizarra será el problema a resolver y el estado final será la solución a la que han ido llegando los agentes. Cada agente trabaja en sus objetivos.

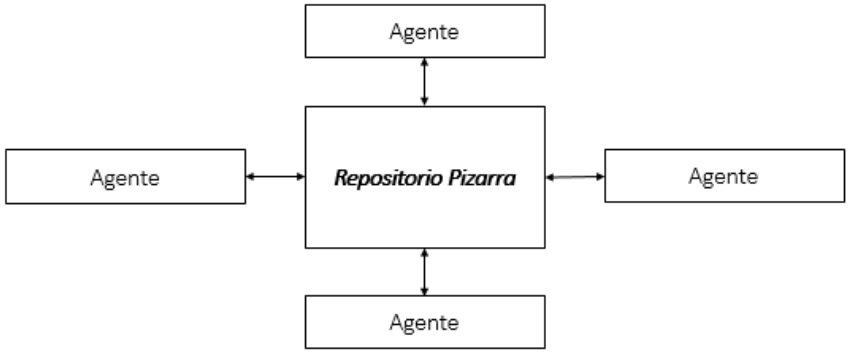


Figura 33: Elementos de una arquitectura en pizarra. Elaboración propia.

Una de las **ventajas** de utilizar este patrón es que puede utilizarse cuando el problema a resolver es muy complicado y no se tiene un conocimiento claro de lo que hay que alcanzar. Sin que sean conscientes, cada agente trabaja en sus objetivos propios, pero existe una cooperación entre ellos que favorece la llegada a la meta.

Las **desventajas** de esto serían que no existe la certeza de que se pueda llegar a una solución y que el tiempo de cómputo puede ser bastante alto.

- **Arquitectura orientada a servicios (SOA).** Es un paradigma para desarrollar y diseñar sistemas distribuidos. Es una filosofía de diseño que sirve para crear sistemas que persiguen satisfacer los objetivos de negocio, proporcionando una metodología y un marco de trabajo para dar soporte las actividades de integración y consolidación de aplicaciones. Una vez se tienen las aplicaciones básicas del sistema, las funcionalidades de la capa aplicativa se exponen en forma de servicios altamente interoperables. HTTP, REST y SOAP son los estándares más utilizados.

Las principales **ventajas** de trabajar con SOA son la agilidad que proporciona, la independencia de plataformas e infraestructuras tecnológicas que favorecen la integración con tecnologías ya existentes.

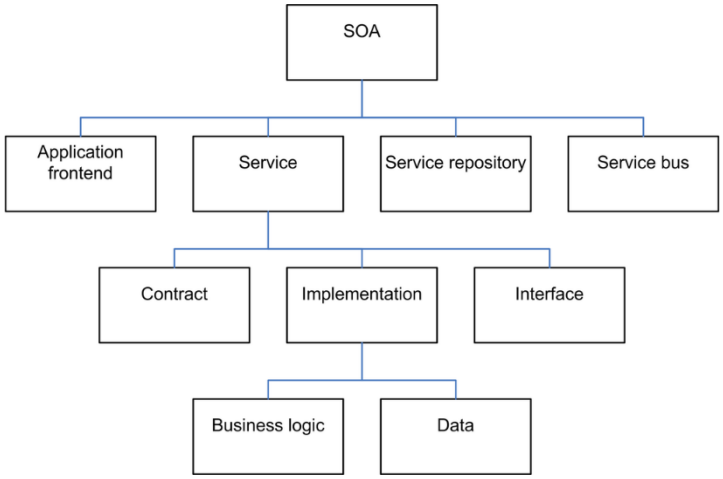


Figura 34: Elementos de una arquitectura SOA. Fuente: https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios.

- **Modelo-Vista-Controlador (MVC)** es un patrón que separa los datos y la lógica de negocio de la interfaz de usuario y el módulo que se encarga de gestionar los eventos y las comunicaciones. Se propone la construcción de tres componentes para la representación de información y la interacción con el usuario: el modelo, la vista y el controlador. El objetivo es la separación de los conceptos y la reutilización del código para facilitar el desarrollo de aplicaciones.

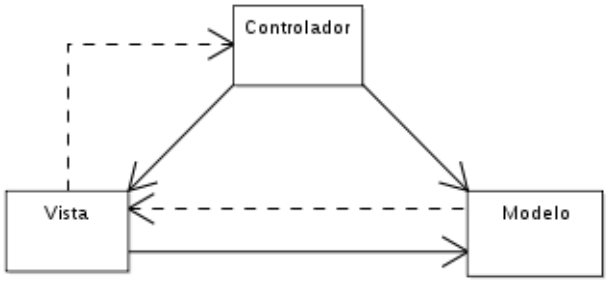


Figura 35: Elementos de una arquitectura MVC. Fuente: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

Base de datos

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y categorizados de distinta manera, que comparten entre sí algún tipo de relación. El objetivo es que los datos estén ordenados y clasificados en conjunto para su posterior uso.

La gestión de estos datos se suele realizar a través de los sistemas de gestión de bases de datos (*Database Management System o DBMS*), que son programas que permiten almacenar los datos y poder acceder a ellos de forma rápida y estructurada. Proporcionan funciones para añadir, borrar, modificar y analizar los datos, además de maneras de administrar la forma en que está organizada la información. Representan la interfaz entre las bases de datos y los distintos usuarios. El acceso a la información se realiza mediante lenguajes de consulta que simplifican la tarea de construir aplicaciones.

En la actualidad, el modelo más utilizado para implementar bases de datos es el modelo relacional. Permiten establecer relaciones entre los datos, guardados en tablas, y a través de dichas conexiones se relacionan los datos de las diferentes tablas. Sin embargo, hace algunos años, empezó a tomar fuerza el desarrollo de una nueva familia de sistemas de bases de datos NoSQL. En vez de guardar los datos en tablas, suelen guardar estructuras de datos en documentos tipo JSON y esto implica que la integridad de datos sea más sencilla y rápida.

Vamos a comparar los sistemas gestores de datos más populares según el ranking de DB-ENGINES [24] de este mismo año. Analizaremos los cinco primeros sistemas: Oracle, MySQL, Microsoft SQL Server, MongoDB y PostgreSQL.

303 systems in ranking, April 2016

Rank			DBMS	Database Model	Score		
Apr 2016	Mar 2016	Apr 2015			Apr 2016	Mar 2016	Apr 2015
1.	1.	1.	Oracle	Relational DBMS	1467.53	-4.48	+21.40
2.	2.	2.	MySQL	Relational DBMS	1370.11	+22.39	+85.53
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1135.05	-1.45	-14.07
4.	4.	4.	MongoDB	Document store	312.44	+7.11	+33.85
5.	5.	5.	PostgreSQL	Relational DBMS	303.73	+4.10	+35.41
6.	6.	6.	DB2	Relational DBMS	184.08	-3.85	-13.56
7.	7.	7.	Microsoft Access	Relational DBMS	131.97	-3.06	-10.22
8.	8.	8.	Cassandra	Wide column store	129.67	-0.66	+24.78
9.	9.	↑ 10.	Redis	Key-value store	111.24	+5.02	+16.69
10.	10.	↓ 9.	SQLite	Relational DBMS	107.96	+2.19	+5.67

Figura 36: Ranking de los DBMS más populares según db-engines.com.

Licencia	Privativa
Versión actual	12.1.0.1.0
Plataformas	Windows, GNU/Linux, Mac OS X, etc.
Características	De tipo objeto-relacional. Uno de los más potentes y completos. Soporta el manejo de grandes volúmenes de datos. Adecuado para grandes empresas. Estable y escalable.
Ventajas	Buen rendimiento y uso de los recursos. Posee un rico diccionario de datos. La herramienta de administración gráfica es intuitiva y cómoda. Brinda soporte a la mayoría de lenguajes de programación. Multiplataforma. Buena relación calidad-precio.
Inconvenientes	Producto con un precio elevado. Coste de mantenimiento alto. Se necesita personal cualificado para su manejo.

Licencia	GPL (*GNU General Public License*). En determinadas soluciones debe adquirirse la licencia comercial.
Versión actual	5.7.9
Plataformas	Windows, GNU/Linux, Mac OS X, Solaris, etc.
Características	Relacional, multihilo y multiusuario. Potente Simple Fácil aprendizaje. Buena integración con otros lenguajes. Uno de los más utilizados. De código abierto.
Ventajas	Gratuito Multiplataforma Velocidad para realizar operaciones. No necesita grandes requerimientos de sistema. Configuración sencilla. Buena compatibilidad con lenguajes y *frameworks.*
Inconvenientes	Poco intuitivo. Poca documentación de ciertas utilidades.

Licencia	Microsoft EULA
Versión actual	SQL Server 2014 (12.0)
Plataformas	Sistemas operativos Microsoft Windows. La próxima versión SQL Server 2016 podrá ejecutarse sobre Linux.
Características	Relacional Potente Posee un entorno gráfico integrado que permite gestionar los datos gráficamente. Permite trabajar en modo cliente/servidor. Posee una extensión al SQL estándar (Transact SQL).
Ventajas	Si se trabaja en una red local, permite administrar información de otros servidores SQL Server. Es un sistema escalable, estable y seguro.
Inconvenientes	No es multiplataforma Consume muchos recursos No tiene una buena relación calidad-precio

Licencia	GNU *(General Public License)* AGPL v3.0 (drivers de licencia Apache).
Versión actual	3.2.4
Plataformas	Windows, GNU/Linux, Mac OS X, Solaris, etc.
Características	NoSQL Guarda estructuras de datos en documentos tipo JSON. De código abierto Soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Tiene un concepto de índices parecido al de los modelos relacionales. Favorece la utilización de Javascript del lado del servidor.
Ventajas	Gratuito Multiplataforma Soporta la mayoría de lenguajes de programación. Adecuado para grandes volúmenes de información. Es rápido en la entrada y salida de datos. Tiene mucho soporte por parte de la comunidad. Escalable.
Inconvenientes	No asegura la integridad y el aislamiento en las transacciones. Puede tener problemas de consistencia.

PostgreSQL

Licencia	PostgreSQL License
Versión actual	9.5.1
Plataformas	Windows, GNU/Linux, Mac OS X, Solaris, etc.
Características	Relacional Orientado a objetos. De código abierto. No tiene versión comercial. Permite el uso de funciones, claves foráneas y *triggers*.
Ventajas	Gratuito Multiplataforma Puede manejar diferentes cargas de trabajo, desde aplicaciones pequeñas a aplicaciones grandes orientadas a Internet con gran concurrencia de usuarios. Buena integración con otros lenguajes Fácil de aprender. Hay mucha documentación.
Inconvenientes	Sintaxis poco intuitiva. Consume bastantes recursos en comparación a MySQL. Es más lento en algunas operaciones. No está disponible en Castellano.

Lenguajes y *Frameworks* del lado del servidor

Los lenguajes de programación del lado del servidor son aquellos que se ejecutan en el servidor web, justo antes de que se envíe la página a través de Internet al usuario. Es el servidor el que maneja toda la información de las bases de datos y cualquier otro recurso que se necesite para luego enviar al cliente una página web con los resultados de las operaciones y consultas. Estos lenguajes se utilizan para generar páginas dinámicas.

En el desarrollo de software, la elección del lenguaje de programación hay que hacerla con sumo cuidado porque puede ser determinante en el éxito del proyecto. Factores como el número de usuarios que soportará, la documentación, el tiempo de desarrollo, los *frameworks*, las funcionalidades a cubrir, etc. deben ser tenidos en cuenta.

Hay muchos lenguajes de programación que se pueden escoger para realizar un proyecto y satisfacer distintas necesidades. Si bien es cierto que muchos de los lenguajes pueden utilizarse en diferentes ámbitos, suele haber alguno que destaque en alguna área.

Es interesante conocer las tendencias y usos extendidos de los lenguajes de programación web del lado del servidor. Según la investigación del Instituto de Ingeniería Eléctrica y Electrónica [25] podemos observar que lenguajes se utilizan para que áreas de desarrollo.

Lenguajes	Entornos		
	Web	Móvil	Empresas
1. Java	✓	✓	✓
2. Python	✓		✓
3. PHP	✓		
4. Javascript	✓	✓	
5. Ruby	✓		
6. Perl	✓		✓

Figura 37: Áreas en las que se utilizan los lenguajes de desarrollo más populares. Elaboración propia.

En función de las características más deseables a la hora de trabajar con un lenguaje se ha elaborado una tabla comparativa.

	ASP.NET	JSP	perl	php	python	ruby	JS
Software libre	✗	✓	✓	✓	✓	✓	✓
Multiplataforma	✗	✓	✓	✓	✓	✓	✓
Aprendizaje sencillo	✓	✗	✓	✓	✓	✓	✓
Suficiente documentación	✓	✗	✗	✓	✓	✗	✗
Buena integración	✗	✓	✓	✓	✓	✗	✓
Orientado a objetos	✓	✓	✓	✓	✓	✓	✓
Organización en capas	✓	✓	✗	✗	✗	✗	✗
Escalable	✓	✗	✗	✓	✓	✓	✓
Relativamente rápido	✗	✗	✗	✓	✓	✓	✓
Framework	Microsoft .NET	spring	catalyst Web Framework	Symfony	dj django	RAILS	node JS

Lenguajes y *Frameworks* del lado del cliente

En la programación de una aplicación web, los lenguajes del lado del cliente se procesan en el cliente web, es decir, se ejecutan en el navegador de Internet. Este tipo de lenguajes se incluyen dentro del código HTML de las páginas web, con las etiquetas necesarias para que navegador lo interprete. Esto tiene como ventaja principal que la ejecución de la aplicación se delega en el cliente con lo que se evita cargar de trabajo al servidor.

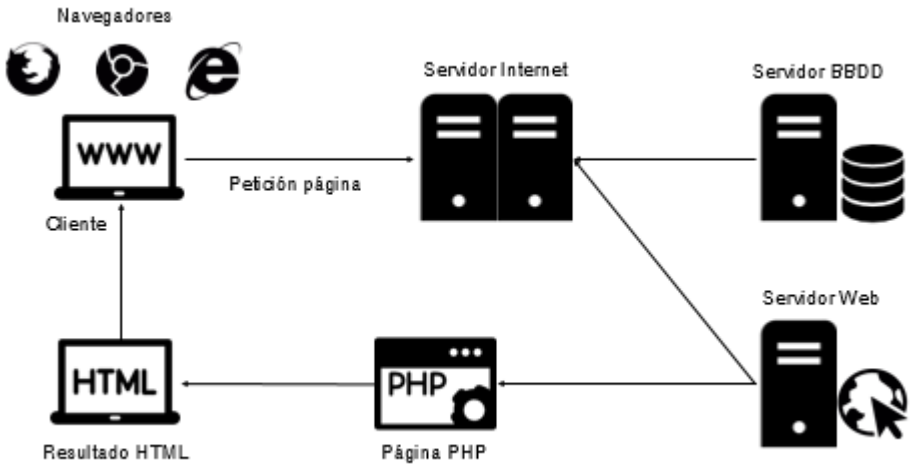


Figura 38: Esquema de funcionamiento de los lenguajes del lado del cliente. Elaboración propia.

Se denomina *front-end* a todas aquellas tecnologías que trabajan del lado del cliente, en el navegador web, generalizándose en tres lenguajes: HTML, CSS y Javascript. Existen más tecnologías que se utilizan del lado del cliente: Applets en Java, Flash, VBScript…También hay *frameworks* y librerías que ayudan al desarrollo de este lado de la aplicación. Por ejemplo, para Javascript están Angular.js y Backbone.js, que son tecnologías más avanzadas. También, nos podemos apoyar en librerías de CSS y de Javascript como anímate.css y JQuery, para poder dar una solución cómoda y amena. Asimismo, existen una serie de lenguajes de transferencia de información como XML, JSON y Ajax, para hacer solicitudes al *back-end* sin necesidad de refrescar la página.

Como sabemos, las habilidades que se necesitan para trabajar en el lado del cliente de una aplicación web han evolucionado muchísimo; ya no son simples documentos con etiquetas HTML y estilos CSS. Hablamos también de la incorporación de patrones como MVC (*Model-View-Controller*) y sus variantes, la evolución y expansión de las APIs, etc.

Por eso, es interesante hacer un pequeño repaso a las herramientas, tecnologías y buenas prácticas para desarrollar en el *front-end* y tenerlas en cuenta a la hora de trabajar en este proyecto.

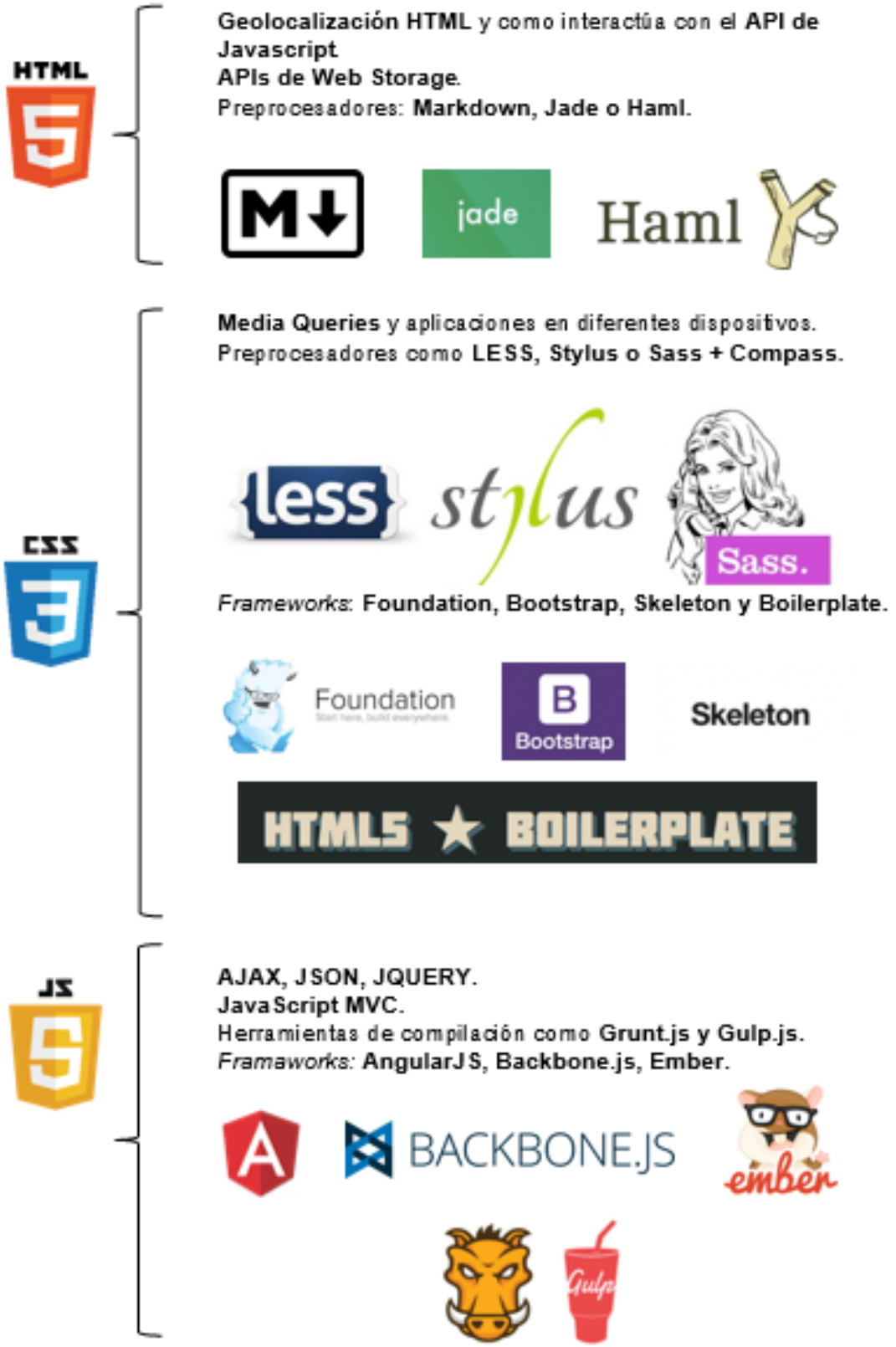


Figura 39: Frameworks front-end. Elaboración propia.

API

Una API (*Application Programming Interface*) es un conjunto de reglas y especificaciones que brinda una librería o biblioteca para que otro software la utilice como capa de abstracción para obtener o actualizar información. La API detalla la forma en que la puede llamarse a cada función para recuperar los datos que se necesiten, y la tarea que esta función debe desempeñar. Esto facilita la relación entre dos aplicaciones que pueden ser independientes entre ellas.

Las APIs pueden diseñarse en cualquier lenguaje de programación y con distintas especificaciones, en función del uso que vaya a hacer de ella la interfaz. Son muy útiles porque permiten reutilizar código que funciona correctamente en vez de tener que volver a pensar y crear nuevas funciones.

Las **APIs de servicios web** permiten el intercambio de información entre un servicio web (software que da acceso a un servicio concreto a través de una URL) y una aplicación. Este intercambio suele producirse a través de peticiones HTTP o HTTPS. La estructura de las peticiones a través de URL y la simplicidad del formato hacen que las APIs sean ampliamente utilizadas para ofrecer datos. La información que se obtiene de las peticiones que hace la aplicación a la API se comparte en formato XML o JSON.

Hay cuatro tipos de API de servicios web habituales entre los desarrolladores: **SOAP** (*Simple Object Access Protocol*), un protocolo estándar de intercambio de información y datos en XML entre dos objetos; **XML-RPC**, un protocolo de llamada a procedimiento remoto que usa XML como formato de datos y llamadas HTTP como sistema de comunicación; **JSON-RPC**, mismo protocolo pero en formato JSON; y **REST** (*Representational State Transfer*), arquitectura de *software* para sistemas hipermedia en la *World Wide Web*; una API REST usa el protocolo HTTP.

Un servicio web *RESTful* es una aplicación que se desarrolla conforme a los principios de *REST*. REST es básicamente una arquitectura de desarrollo web que se apoya en el estándar HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos. Para el intercambio de información se usa *XML* o *JSON*.

Para hacer los accesos a la API más sencillos de entender, REST utiliza formatos de URL con el fin de que sean lo más predecibles y claros posibles y que navegar a través de los recursos sea intuitivo.

En REST cada uno de los verbos que se utilizan en las peticiones equivale a una acción sobre un registro o una colección. Estas acciones suelen ser *craate, read, update y delete (CRUD)* de base de datos, que se equiparan a las operaciones *GET, POST, PUT y DELETE*.

Método	Acción	Seguro	Idempotente
GET	Obtiene un recurso una colección	Sí	Sí
POST	Crea un nuevo recurso	No	No
PUT	Actualiza un recurso específico	No	Sí
DELETE	Eliminar un recurso específico	No	Sí
PATCH	Actualiza parcialmente un recurso específico	No	No

Figura 40: Métodos HTTP. Acción y características. Elaboración propia.

El único que cumple la condición de seguridad es GET, porque solo recupera recursos sin realizar ningún cambio. Por otro lado, la creación, modificación y eliminación cambian los recursos, así que pueden generar inconsistencias en la base de datos.

Idempotente es la capacidad de un método para no repetir una misma acción. DELETE es idempotente porque cuando se elimina un recurso no se puede repetir esta operación. En cambio, POST no lo es, porque cada vez que se utiliza se crea un nuevo recurso.

A la hora de crear una API, es común que no se haga un diseño previo de la misma por problemas de planificación. Esto se traduce, en ocasiones, en inconsistencia entre los objetos y métodos, además de fallos de seguridad.

Cada vez está cobrando más importancia el tema del diseño previo de APIs utilizando herramientas que tengan en consideración las necesidades de las aplicaciones que vayan a utilizar los servicios**. API Blueprint, RAML y Swagger** representan tres excelentes herramientas para diseñar APIs.

API Blueprint

Es un amplio entorno de desarrollo de APIs que cubre el ciclo de vida completo. Utiliza el lenguaje *markdown* (lenguaje de marcado ligero .md) para escribir las definiciones y transformarlas en JSON.

Se puede usar Node.JS, .NET o Ruby para realizar el *binding* con la API. Posibilita crear documentación interactiva, API mocks, validaciones, etc. con la herramienta **Apiary.io** y realizar testing con **Dredd**. También cuenta con plugin para **Sublime Text** y **Atom** para escribir las especificaciones.

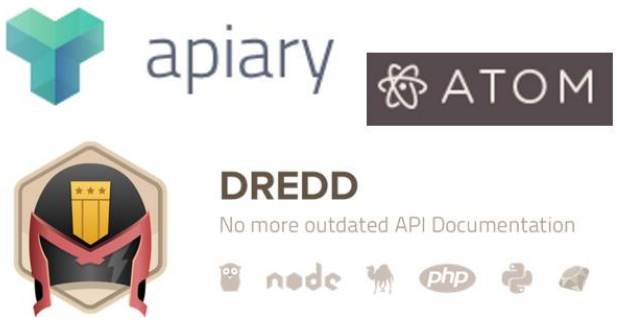


Figura 41: Herramientas compatibles y plugins de API Blueprint. Elaboración propia.

RAML

RESTful API Modeling Language (RAML) permite describir servicios REST de forma completa. Construido a partir de YAML y JSON, destaca su capacidad de reutilización y tiene unos patrones para aplicar las definiciones muy útiles que minimizan las repeticiones.

Swagger

Su potencia reside en el *framework* implementado en Scala, Java, Javascript, Ruby y PHP, que permite ir escribiendo el código al mismo tiempo que implementamos las definiciones del API.

Swagger está rodeado de un gran ecosistema de herramientas que incluyen desde las interfaces de usuarios de aplicaciones, librerías de código de bajo nivel y soluciones de gestión de APIs.

Aplicación móvil

El desarrollo de aplicaciones para teléfonos inteligentes, tabletas y otros dispositivos móviles ha aumentado de forma espectacular en los últimos tiempos. Las personas hacemos uso de ellas para todo tipo de cosas: comunicarnos, organizarnos, conocer gente nueva, jugar, aprender, comprar, estar al día de las noticias, etc.

En el ámbito laboral también han ganado terreno; las *apps* ya no son solo para disfrutarlas y utilizarlas en el tiempo libre, si no que se han vuelto herramientas muy importantes también en el trabajo. Existen muchos negocios que ponen a disposición de la gente aplicaciones móviles para dar a conocer su trabajo y también abrir otra vía de comunicación con ellos para pedir citas, presupuestos, realizar compras, enviar datos, ubicaciones…

Pero desarrollar para dispositivos móviles significa que hay que tener en cuenta, además de las necesidades de los usuarios, las limitaciones de estos dispositivos. Hay infinidad de terminales con diferentes sistemas operativos, requerimientos de software y hardware, configuraciones, baterías y tamaños de pantalla.

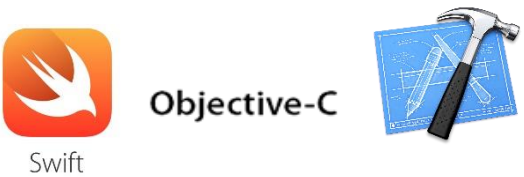
La creación de aplicaciones de este tipo requiere el uso de entornos de desarrollo integrados para facilitar las cosas al programador (*Software Development Kit* o *SDK*). Y según la tecnología de desarrollo podemos clasificar en tres tipos las aplicaciones móviles:

App Nativa

Las *apps* nativas son aquellas que se desarrollan de forma específica para cada sistema operativo. Android, IOS y Windows Phone son las principales plataformas, por lo que habría que desarrollar una aplicación en el lenguaje correspondiente por cada una de ellas.

Objective-C es uno de los lenguajes utilizados para programar aplicaciones nativas de iPhone, iPad e iPod touch. **Xcode** es el entorno de desarrollo de Apple.

Recientemente, Apple ha lanzado **Swift**, un lenguaje potente e intuitivo para crear aplicaciones para IOS, Mac, Apple TV y Apple Watch.



Las aplicaciones se desarrollan habitualmente en **Java**. Se suele utilizar **Eclipse** como IDE de desarrollo, pero desde que Android lanzó **Android Studio** se ha convertido en el IDE de cabecera.

Basic4Android es una de las plataformas enemigas de Android Studio. Programa en Android con **VisualBasic**. Existen más plataformas con las que se puede desarrollar como **AppInventor** o **LiveCode**. Esta última también está disponible para desarrollar en iOS.

La forma más común de desarrollar aplicaciones para Windows Phone es con **Visual Basic .NET.**

Se utiliza **Visual Studio** de Microsoft como entorno de desarrollo, aunque Microsoft ha creado Windows App Studio para crear aplicaciones para móviles, tablets y ordenadores.

Desarrollar aplicaciones nativas en los lenguajes de cada sistema operativo tiene una serie de ventajas e inconvenientes:

Ventajas	Inconvenientes
✓ **No necesitan conexión a Internet.**	✗ El **precio** de desarrollo es **elevado.**
✓ **Puede acceder** a todas las características hardware del dispositivo.	✗ La **velocidad de desarrollo** es **menor.**
✓ La **experiencia de usuario mejora.**	✗ La **curva de aprendizaje** es más **alta.**
✓ Se **actualiza** la *app* automáticamente.	✗ **El código no es reutilizable** entre las plataformas móviles.
✓ **Buena relación calidad-precio.**	✗ Cada plataforma es diferentes y las herramientas de desarrollo también.

Figura 42: Ventajas e inconvenientes de las apps nativas. Elaboración propia.

Web App

Las aplicaciones web se desarrollan con **HTML, Javascript y CSS**. No necesitan instalarse puesto que estas **se ejecutan en el navegador web** del dispositivo móvil. Internet Explorer en la plataforma Windows Phone, Google Chrome o Mozilla Firefox en Android, y Safari en IOS, entre otros.

El contenido de la aplicación se adapta a cada tamaño de pantalla adquiriendo así un **aspecto similar al de las *apps***. Incluso se adapta la navegación para mejorar la experiencia de usuario.

Hasta cierto punto, este tipo de aplicaciones no podrían considerarse *apps* puesto que no necesitan instalación, aunque se podría crear un acceso directo en la pantalla del dispositivo que equivaldría a instalar la aplicación. Incluso la comercialización debe realizarse de forma diferente a las aplicaciones como tal que aparecen en la *app store* de cada plataforma.

Desde luego, las *web apps* son una buena opción si queremos optimizar una página web para dispositivos móviles sin mucha complicación.

Ventajas	Inconvenientes
✓ **Flexibilidad.** Cualquier página puede convertirse en *web app*.	x El **rendimiento** suele ser **menor** que en las *apps* nativas.
✓ **Multiplataforma**	x **Se requiere conexión a Internet.**
✓ **No requiere actualizaciones.**	x El **acceso a las características** hardware del dispositivo es **limitado.**
✓ El **código** es **reutilizable.**	x La **experiencia de usuario** puede ser **mejorable.**
✓ El proceso de **desarrollo** es **más fácil y económico.**	x Pueden producirse problemas de **latencia en web.**
✓ El sitio debe ser ***responsive*** pero existen muchas plantillas y es sencillo de realizar.	
✓ **No se necesita publicar** la aplicación.	

Figura 43: Ventajas e inconvenientes de las webs apps. Elaboración propia.

Web App Nativa

Este tipo de aplicaciones se consideran **híbridas** porque son una combinación de los dos tipos anteriores. **Se desarrollan con** lenguajes típicos de las *web app*s (**HTML, CSS y Javascript**). Esto significa que son aplicaciones **multiplataforma** ya que en la mayoría de casos se ejecutan sobre Android, IOS y Windows Phone sin modificaciones. También tienen la posibilidad de **acceder a casi todas las características hardware del dispositivo**, como lo haría una *app* nativa.

Su distribución es posible en la *app store* por la existencia de *frameworks* que nos acercan a la creación de una experiencia móvil nativa usando solo tecnologías web.

Existen una serie de marcos para trabajar con **Phonegap** y **Cordova** en el desarrollo de aplicaciones híbridas.

Figura 44: Características de los frameworks para desarrollar web apps nativas. Elaboración propia.

Este tipo de aplicaciones tienen sus ventajas e inconvenientes:

Ventajas	Inconvenientes
✓ **Instalación nativa** pero construida con Javascript, HTML y CSS.	x Puede ocurrir que el **diseño de la app no esté relacionado con la vista del sistema** operativo en que se muestre.
✓ **Puede acceder** a parte de las características hardware del dispositivo.	x **La experiencia de usuario** puede ser parecida a la de las aplicaciones web. **Menos fluida** que la de las nativas.
✓ **El mismo código** sirve para todas las plataformas.	
✓ **Se puede distribuir** en las *app store*.	

Figura 45: Ventajas e inconvenientes de las Web App Nativas. Elaboración propia.

Objetivos

- ***Objetivo principal:*** Desarrollar **un sistema de gestión de incidencias en la vía pública online formado por dos aplicaciones integradas**: Una aplicación web donde se clasifiquen, asignen y prioricen las incidencias enviadas por los usuarios y una aplicación móvil que permita reportar esas incidencias. El sistema representaría una solución global cubriendo todos los ciclos del proceso desde que la incidencia es dada de alta hasta que es resuelta y analizada por el área correspondiente.

    - ***Sub-objetivo 1:*** **Elaborar el estado del arte** de forma completa, informando de los precedentes del proyecto a realizar, las herramientas existentes que sean similares a lo que se va a crear y un estudio de las tecnologías con las que se va a desarrollar.
    - ***Sub-objetivo 2:*** **Realizar un análisis funcional** basado en las necesidades para esta aplicación. Identificar los requisitos funcionales y no funcionales y los casos de uso. Crear *mockups* de las diferentes vistas de la aplicación.
    - ***Sub-objetivo 3:*** **Establecer un diseño de arquitectura del sistema** basándonos en un estudio previo de los tipos de arquitecturas web.
    - ***Sub-objetivo 4:*** **Seleccionar las tecnologías a utilizar** en función de las características mencionadas.
    - ***Sub-objetivo 5:*** Desarrollar el sistema propuesto en este proyecto atendiendo a las siguientes directrices:

        - ***Sub-objetivo 5.1:*** **Diseñar una base de datos** que de soporte a la aplicación.
        - ***Sub-objetivo 5.2:*** **Construir** un *back-end* basado en **una API** que pueda ser reutilizada.
        - ***Sub-objetivo 5.3:*** **Fijar una arquitectura** en la aplicación tipo ***single-page***.
        - ***Sub-objetivo 5.4:*** Hacer que la aplicación sea **multiplataforma.**
        - ***Sub-objetivo 5.5:*** Diseñar una **interfaz sencilla, usable e intuitiva**.
        - ***Sub-objetivo 5.6:*** Crear un **panel de administración del sistema** para dar de alta a los clientes, eliminarlos o banear a usuarios molestos. Este panel es una herramienta de gestión para el que presta el servicio (yo).
        - ***Sub-objetivo 5.7:*** Crear la **aplicación web que soporte la gestión de las incidencias**: crear, eliminar, modificar, asignar responsables, crear áreas de trabajo o perfiles de diferentes roles, reportar usuarios, etc.

- ***Sub-objetivo 5.8:*** Crear una ***app* móvil** híbrida que permita el registro de los usuarios y acceder a las características hardware del dispositivo (ubicación, cámara, etc.) para enviar las incidencias.
- ***Sub-objetivo 5.9:*** Permitir **la trazabilidad de las incidencias**. Las incidencias registradas pasarán por varios estados desde su tramitación hasta la resolución de la misma. Esta información también se compartirá con el usuario final que podrá conocer en todo momento a través de la *app* el estado de la incidencia que ha notificado.
- ***Sub-objetivo 5.10:*** **Establecer una jerarquía de responsabilidades en la gestión de las incidencias**: *"propietario", "gestores" y "operadores"*, cada uno de estos tipos de usuario tendrá un rol en el ciclo de vida de la incidencia reportada.

- ***Sub-objetivo 6:*** **Difusión** del sistema a través de una ***landing page*.**
- ***Sub-objetivo general:*** Aprender a **desenvolverme, tomar decisiones y adquirir de experiencia.**

Requisitos Funcionales

Los requisitos funcionales se dividen entre los que expresan el funcionamiento de la aplicación web y los de la aplicación móvil. Dentro de esos dos grupos, se distinguen entre obligatorios y deseables, y a su vez, se muestran en orden de prioridad, de más importante a menos.

Aplicación Web

Obligatorios

RF01: Registro

El sistema requerirá que los usuarios estén registrados para acceder a la aplicación. El primer registro será el del propietario, que dará de alta a su organización, a través de la *landing-page*.

RF02: *Login*

El sistema pedirá que los usuarios registrados inicien sesión con su identificador y su contraseña para acceder al panel de gestión.

RF03: *Logout*

Se podrá cerrar la sesión de usuario en el panel de gestión.

RF04: Mostrar incidencia

En el panel de gestión se mostrarán las incidencias registradas en el sistema y su información adjunta.

RF05: Editar incidencia

El sistema permitirá que los usuarios modifiquen ciertos datos de las incidencias registradas.

RF06: Eliminar incidencia

El sistema permitirá que los usuarios eliminen las incidencias registradas.

RF07: Crear incidencia

La herramienta permitirá dar de alta nuevas incidencias.

RF08: Aceptar incidencia

Para empezar el proceso de resolución de una incidencia, el sistema deberá permitir que el usuario responsable la acepte.

RF09: Cerrar incidencia

El sistema permitirá que una incidencia se cierre cuando se haya resuelto.

RF10: Asignar incidencia

La herramienta permitirá asignar usuarios responsables a cada incidencia.

RF11: Rechazar incidencia

La herramienta posibilitará que los usuarios a los que se les ha asignado una incidencia puedan rechazarla.

RF12: Incidencia resuelta

En el momento en que se haya resuelto una incidencia, el sistema permitirá que el usuario encargado cambie el estado de esta incidencia a *resuelta.*

RF13: Crear usuario

La herramienta permitirá que propietarios y gestores den de alta en el sistema otros usuarios.

RF14: Mostrar usuario

El sistema mostrará a los propietarios y gestores la información referente a los usuarios que pueden gestionar.

RF15: Editar usuario

El sistema permitirá que propietarios y gestores modifiquen ciertos datos de los usuarios que puedan gestionar.

RF16: Eliminar usuario

El sistema permitirá que propietarios y gestores eliminen del sistema a los usuarios que puedan gestionar.

RF17: Formulario de búsqueda

El sistema proporcionará un formulario de búsqueda que permita obtener resultados, a través de palabras clave y filtros, que se muestren en pantalla.

RF18: Generar histórico de incidencia

El sistema mostrará un histórico de cada incidencia, desde el momento en que llega al sistema hasta que se resuelve. Se reflejarán actividades como su registro, modificaciones, asignación, cambios de estado, etc.

RF19: Añadir nota

La herramienta permitirá que se puedan añadir notas a las incidencias.

RF20: Eliminar nota

La herramienta permitirá que se puedan eliminar las notas añadidas.

RF21: Reportar usuario

El sistema dará la posibilidad de banear usuarios ciudadanos que den un mal uso a la *app* móvil notificando incidencias no adecuadas.

RF22: Crear área

El sistema posibilitará la creación de áreas (Jardinería, Alumbrado, etc.) para ayudar a la clasificación de las incidencias. Por defecto, el sistema proporcionará un área base o genérica donde se engloban todas las incidencias.

RF23: Mostrar área

El sistema mostrará las áreas que se han añadido al sistema.

RF24: Editar área

La herramienta permitirá que los usuarios modifiquen el nombre de las áreas.

RF25: Eliminar área

La herramienta permitirá que un área del sistema sea eliminada.

RF26: Crear tipo de incidencia

El sistema permitirá la creación de incidencias tipo dentro del sistema para etiquetar las incidencias y clasificarlas. Por defecto, el sistema proporcionará un tipo genérico que engloba a todas las incidencias.

RF27: Mostrar tipos de incidencia

La herramienta mostrará los tipos de incidencias que se han añadido al sistema.

RF28: Editar tipo de incidencia

La herramienta permitirá que los usuarios modifiquen el nombre los tipos de incidencia del sistema.

RF29: Eliminar tipo de incidencia

La herramienta permitirá que los usuarios puedan eliminar los tipos de incidencias del sistema.

Deseables

RF30: Subir foto

El sistema permitirá que los usuarios adjunten fotos a las incidencias.

RF31: Eliminar foto

El sistema permitirá que los usuarios eliminen las fotos que han adjuntado.

RF32: Recuperar contraseña

El sistema permitirá que un usuario recupere el acceso a su panel de gestión reestableciendo su contraseña.

RF33: Editar datos de perfil

El sistema posibilitará editar ciertos datos en el perfil del panel de gestión de cada usuario.

Aplicación móvil

Obligatorias

RF34: Registro

El sistema requerirá que los usuarios estén registrados para acceder a la aplicación.

RF35: *Login*

El sistema pedirá que los usuarios registrados inicien sesión con su identificador y su contraseña para enviar una incidencia o consultar sus incidencias enviadas.

RF36: *Logout*

El sistema permitirá cerrar la sesión de usuario.

RF37: Seleccionar entidad

El sistema permitirá que los usuarios ciudadanos seleccionen, eligiendo provincia y municipio, la entidad a la que quieren informar de una incidencia.

RF38: Enviar incidencia

La herramienta posibilitará la comunicación entre la *app* móvil y web para que los usuarios ciudadanos puedan reportar incidencias y estas se visualicen en el panel de gestión.

RF39: Enviar descripción

El sistema deberá permitir que los reportes, como mínimo, sean un texto descriptivo de la incidencia y se visualice en el panel de gestión web.

RF40: Consultar incidencias

El sistema mostrará a los usuarios ciudadanos que hayan enviado incidencias el estado en que se encuentran las mismas.

Deseables

RF40: Seleccionar área y tipo

La herramienta permitirá seleccionar el área y el tipo de la incidencia que va a enviarse, en función de las áreas y las tipologías que tenga la entidad cliente definidas.

RF41: Adjuntar ubicación

El sistema permitirá *geolocalizar* la incidencia que el usuario está accediendo a las características hardware del dispositivo móvil.

RF42: Adjuntar fotografía

El sistema permitirá tomar una fotografía de la incidencia o seleccionarla de la galería del móvil para adjuntarla en el reporte de la incidencia.

Panel de administración

Deseable

RF43: Gestionar clientes y usuarios ciudadanos

La herramienta deberá tener un panel de gestión de la propia plataforma donde llevar los asuntos relacionados con los clientes del servicio y los usuarios ciudadanos.

Requisitos No Funcionales

RNF01: Aplicación móvil multiplataforma

La aplicación móvil debe ser capaz de ejecutarse en diferentes sistemas operativos.

RNF02: Integración con terceros y reutilización de servicios

Facilitar la integración y la reutilización de recursos desarrollando el *back-end* en forma de API que pueda ser abierta o prestada a terceros.

RNF03: Accesibilidad y *hosting*

Asegurar que cualquier usuario, desde el máximo de ubicaciones y durante la mayor franja de tiempo, pueda acceder al sistema. Utilizar un *hosting* para alojar la aplicación favorece esta condición.

RNF04: Seguridad

El sistema debe desarrollarse bajo las prácticas básicas de seguridad web como utilizar el *Login* como medida de seguridad, evitar exponerse a ataques que involucran a un usuario modificando la URL, y establecer el uso del protocolo HTTPS para poder cifrar el canal de comunicación por el que enviaremos nuestra información.

RNF05: Agilizar el desarrollo

Se van a definir los requisitos y realizar el diseño del sistema en detalle antes de empezar a implementarlo, para acortar el tiempo desarrollo. Se van a utilizar *frameworks* en la medida en que estos puedan ayudar a agilizar el proceso. Hablamos de Bootstrap, AngulaJS, Slim, etc.

RNF06: Escalabilidad

Se usará un sistema de base de datos relacional que proveerá de un servicio capaz de soportar la carga de tráfico.

RNF07: Diseño Responsive

Las interfaces deben visualizarse correctamente desde cualquier tipo de dispositivo y resolución de pantalla.

Diseño

Metodología seguida

Cuando se trata de elegir una metodología no podemos decir que unas son mejores que otras ya que cada método tiene sus aplicaciones y lo importante es entender cuál de ellas es la más apropiada para nuestro proyecto y sus necesidades.

Si profundizamos un poco en los pros y los contras de las metodologías Ágiles y en Cascada del desarrollo de software, vemos que las primeras son especialmente beneficiosas para proyectos cuyos objetivos finales no están claramente definidos y pueden ir desarrollándose en pequeños módulos. En cambio, la clásica metodología en Cascada se centra en la planificación del proyecto, en tener una visión clara de lo que se quiere conseguir y continuar el plan establecido.

Por las características del desarrollo de un trabajo como es el trabajo final de grado y en concreto, por las particularidades de este proyecto, la metodología que se está siguiendo es claramente en Cascada.

La especificación de requisitos se ha plasmado en este documento desde el principio, consensuando todo lo que el sistema debe hacer y estableciendo los requerimientos del proceso de elaboración que se seguirán en las siguientes etapas.

En el diseño del sistema, se definen tanto la parte de alto nivel o arquitectónico como el diseño de interfaces. Una vez que en la fase de especificación o análisis se ha descrito el problema, el objetivo en esta etapa es concretar la estructura de la solución. También se realiza el diseño del programa donde se realizan los algoritmos y funciones necesarios para el cumplimiento de los requerimientos del sistema que se está desarrollando.

En la codificación, además de implementar el código fuente, se plasman los resultados que se han obtenido en esta memoria. Seguidamente, en la etapa de pruebas, se explican los resultados obtenidos de las comprobaciones que se le han hecho al sistema. Si el sistema se ha proporcionado a una serie de usuarios que lo han probado y han dado su valoración al respecto, esto también se puede incluir en el presente documento.

Por último, en las conclusiones se explica qué objetivos se han cumplido, cuáles no y por qué, y pueden reflejarse las impresiones del propio autor y las consideraciones futuras respecto al sistema.

Casos de uso

Para especificar la comunicación y el comportamiento del sistema, se necesita saber cómo va a ser su interacción con los usuarios y/u otros sistemas. Los casos de uso nos muestran de forma gráfica la relación que mantienen los actores con otros actores y con el sistema.

En el escenario de este proyecto pueden existir los siguientes usuarios:

- **Administrador de la plataforma (yo).** El encargado de gestionar los clientes, los usuarios de la app móvil, de llevar la monetización, etc.
- **Propietario de la cuenta.** Es la persona encargada de dar de alta la cuenta en la aplicación web y de crear usuarios gestores y operarios, además de toda la gestión que permite la plataforma.
- **Gestores.** Una vez dados de alta por el propietario, los gestores pueden crear desde su cuenta usuarios operarios y encargarse de toda la gestión también.
- **Operarios.** Su cometido principal es resolver las incidencias desplazándose al lugar del suceso e informar del proceso a través de la app web.
- **Ciudadanos usuarios de la app móvil.** Tienen el papel de notificar las incidencias que puedan encontrarse en una entidad u organismo registrado a través de la app móvil que tienen a su disposición.

El propietario de la cuenta y los gestores tienen prácticamente los mismos permisos. Esto es así porque puede ocurrir que la persona que se registre en la plataforma no pueda ser la encargada de gestionar las incidencias. Por eso, se permite delegar todas las funciones a los usuarios gestores, los cuales son creados por este usuario propietario de la cuenta, únicamente proporcionando el correo electrónico del futuro gestor y una contraseña. A partir de ese momento, el usuario gestor creado puede *loguearse* en la aplicación web y empezar a recibir las incidencias que van registrando.

Hay muchos procesos compartidos en el sistema, que los realizan todos los actores. Para no reiterar casos de uso ni escenarios, la distribución de los diagramas se ha hecho de manera que aparezcan los procesos comunes de todos y los no comunes, sin repeticiones.

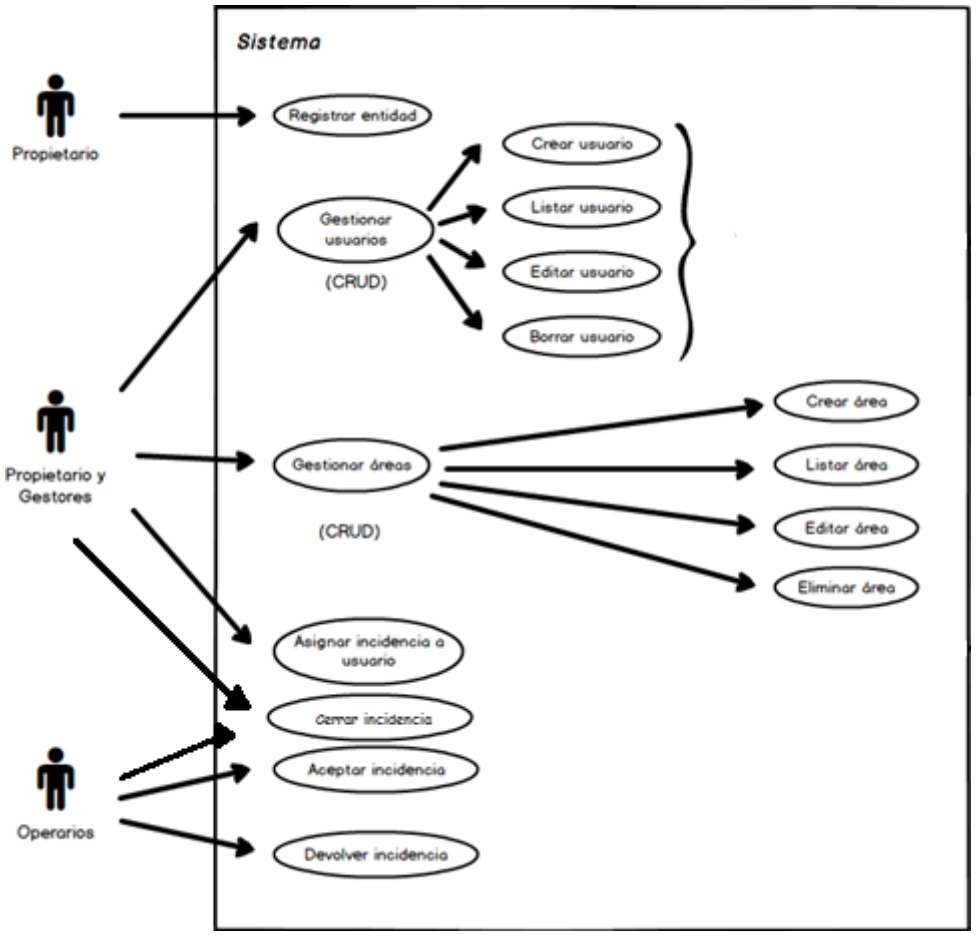


Figura 46: Diagrama de algunos casos de uso del propietario, gestores y operarios. Elaboración propia.

Se considera propietario a la persona que registra la entidad (ayuntamiento, universidad, barrio…) por primera vez en el sistema. Debe proporcionar su email y una contraseña, además de algunos datos sobre dicha entidad. Por tanto, esa actividad puede realizarse una sola vez y por el usuario que denominamos propietario de la cuenta.

El propietario y los gestores que se den de alta, son los únicos que pueden encargarse de la **gestión de usuarios**, de **la gestión de áreas** y la **asignación de incidencias a usuarios.** A continuación, se describen estas acciones o actividades.

- Gestión de usuarios: el propietario de la plataforma puede crear, ver, editar y eliminar gestores y operarios del sistema, pero los gestores solo pueden hacerlo con los operarios. Cuando un usuario operario es dado de alta en el sistema debe especificarse a que área pertenece. Por defecto, en el sistema hay un "área base" pero pueden crearse más en función de las necesidades.

- Gestión de áreas: la plataforma da la posibilidad de definir áreas en las que dividir las incidencias y los operarios. Esto puede ser útil dependiendo del tamaño de la entidad y del origen de las incidencias. Por ejemplo, a un ayuntamiento de un municipio pequeño o mediano puede interesarle dividir la gestión de las incidencias entre varias áreas como pueden ser jardinería, carreteras y vías, alumbrado y aguas. O dividirla en barrios, si hablamos de una ciudad más grande. Igual que si nos ponemos en el caso de una finca de vecinos en una urbanización con varios edificios; quizá al tener una extensión más pequeña o tener un solo encargado de mantenimiento, con una sola área de gestión sería suficiente.
- Asignación de incidencias a usuarios: la incidencia llega al panel de gestión del propietario y los gestores en primer lugar. Se comprueba que la incidencia sea adecuada y que esté clasificada en el área correspondiente. Una vez hecho esto, se asigna a los usuarios operarios de esa área para que la resuelvan.
- Cerrar incidencia: una vez que la incidencia ya ha sido resuelta, el operario debe marcarla como "resuelta" y el gestor que la asignó a ese operario, después de dar el visto bueno, tiene que cambiar su estado a "cerrada" para terminar con el ciclo de vida de la incidencia.

Hay dos funciones que solo pueden realizar los **operarios** y que aparecen en el esquema anterior; aceptar la incidencia que le han asignado y rechazarla. Aceptar una incidencia implica que el operario se encarga de su resolución, pero también, puede ser devuelta. Esto puede ocurrir cuando se ha clasificado en un área equivocada y el operario detecta esta confusión. Puede pasar que se notifique un aviso de una farola caída, por poner un ejemplo, y se clasifique dentro del área de "alumbrado" cuando en realidad sería del área de "aceras" porque se deben arreglar los adoquines que sujetan el poste.

La plataforma en este caso da la posibilidad de, una vez asignadas, puedan rechazarse y devolverse al gestor o propietario que la asignó para que vuelva a reasignarlas.

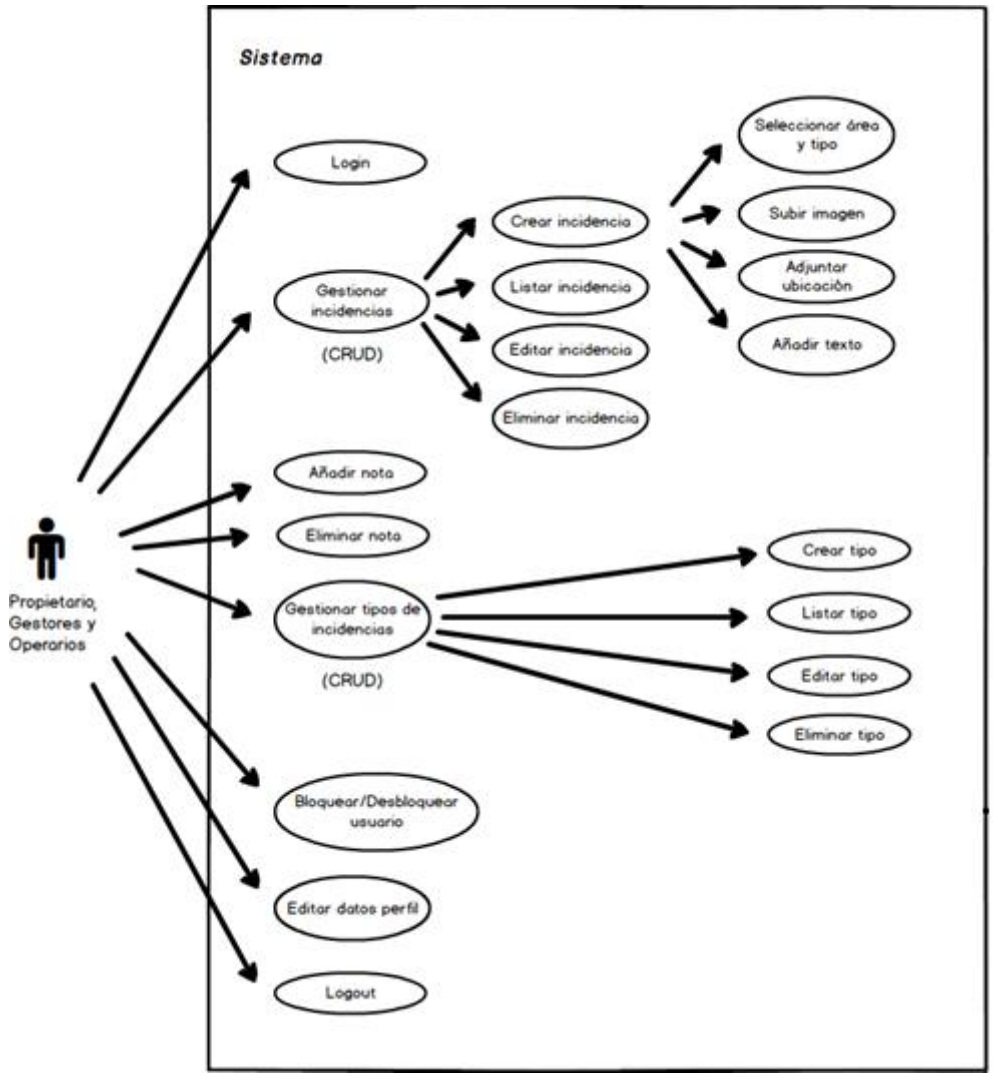


Figura 47: Diagrama de casos de acciones comunes de propietario, gestores y operarios. Elaboración propia.

El inicio de sesión, la edición del perfil y el *logout* son funciones típicas de cualquier sistema y todos los usuarios pueden realizar dichas acciones.

Respecto a las incidencias, pueden registrarse en el sistema por la notificación un usuario ciudadano a través de la app móvil y por los usuarios que forman parte de la plataforma a través de la app web. El propietario y los gestores además de esto, pueden ver todos los avisos que se reciben, editar las incidencias y eliminarlas del sistema. Los operarios solo pueden hacerlo de las incidencias que se les han asignado.

En el sistema puede definirse una serie de incidencias "tipo" que ayuden a la clasificación de las mismas. Por defecto, existe una incidencia tipo genérica que englobaría todos los tipos de incidencias si no se definen más tipos.

Añadir más tipos de incidencias y su gestión puede realizarse tanto por el propietario o gestor como por el operario, que al estar a pie de calle día a día puede conocer con más detalle las clases de averías que suelen producirse.

Cada incidencia cuenta con un histórico que refleja todas las acciones que se han realizado sobre ella desde su registro en la plataforma hasta su resolución. Este registro de actividad permite que todos los usuarios puedan añadir notas a este histórico, o eliminarlas, igual que se haría en un informe o parte de incidencia tradicional. Cuándo y quién ha dado de alta el aviso, estados por los que va pasando la incidencia, usuarios que intervienen en la resolución de la incidencia, anotaciones…

Todos los perfiles dados de alta en la plataforma tienen permisos para gestionar las incidencias y también reportar a usuarios ciudadanos de la *app* móvil que molesten o interfieran en su trabajo. Podría pasar que algún usuario enviara cosas de mal gusto o se reiterara en algún comportamiento. La plataforma da la posibilidad de reportarlos y eliminar automáticamente la incidencia inadecuada. Esto implica que el usuario no puede notificar de más incidencias en ese organismo y que para recuperar ese derecho debe ponerse en contacto con el administrador del servicio para justificarse o que el usuario de la plataforma termine por desbloquearlo.

El papel del administrador del servicio (yo) sería gestionar las altas de los clientes, rentabilizar el uso de las aplicaciones, administrar las cuentas de los usuarios de la *app* móvil, contactar con clientes o pasar presupuestos a través de la *Landing page* del sistema… En principio esto no va realizarse de manera funcional, aunque que puede aparecer reflejado de forma teórica en este documento.

Actualmente, existe la *Landing page* donde se lanza Gify, se describe cómo funciona y proporciona un formulario de contacto para cualquier comunicación, pero funcionalmente solo se implementarán el registro del cliente, el *Login* de los usuarios y la recuperación de contraseña.

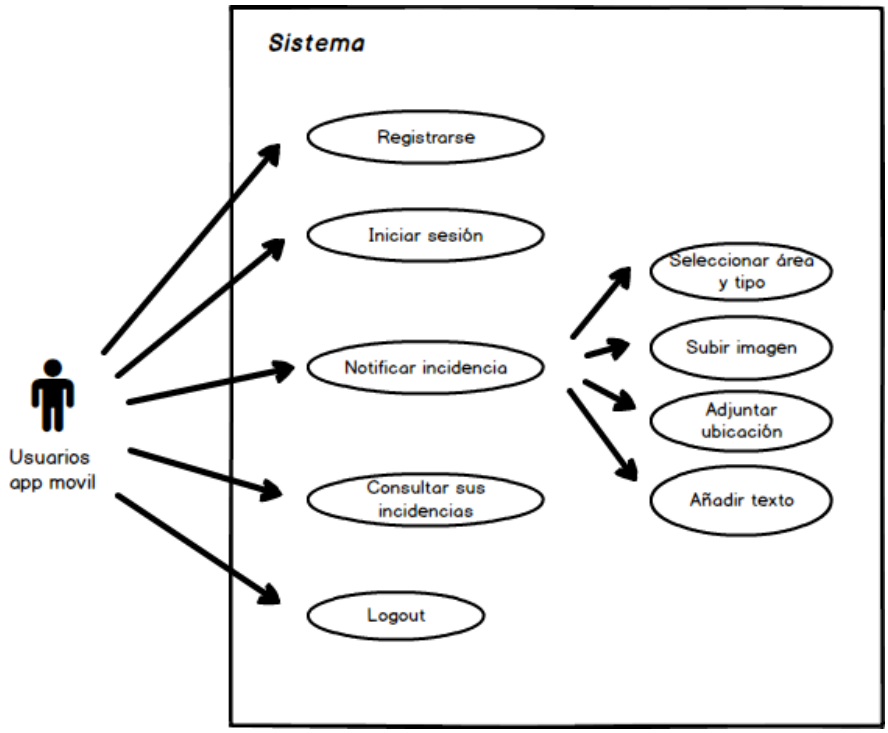


Figura 48: Diagrama de casos de uso de los usuarios de la app móvil. Elaboración propia.

Los usuarios ciudadanos que reportan incidencias a través del móvil o la *tablet*, ayudando así a los organismos registrados en el sistema, realizan las funciones que podemos observar más arriba.

El registro puede realizarse de la manera habitual, creando un usuario para poder *iniciar sesión*, o con el Login de Facebook. Esto permitirá enviar la incidencia al organismo que se haya seleccionado. Después de buscar el ayuntamiento, centro comercial, finca o universidad registrada, empieza el sencillo proceso de notificación de la incidencia:

1. Se selecciona un área, de las que haya definido la entidad, que coincida con el perfil de la avería.
2. Se indica la dirección del lugar donde se ha producido el problema por geolocalización.
3. Se sube una imagen de la avería.
4. Se elige un tipo que concrete la clase de incidencia que se está reportando, y, además, puede añadirse una descripción o texto corto.

Una vez notificada alguna incidencia, se puede consultar su estado durante todo el ciclo de vida del reporte en el apartado “mis incidencias”.

Mockups

Landing Page

En estos dos *mockups* de la *Landing-page* se muestra información sobre el sistema. Son las secciones Home y Servicios.

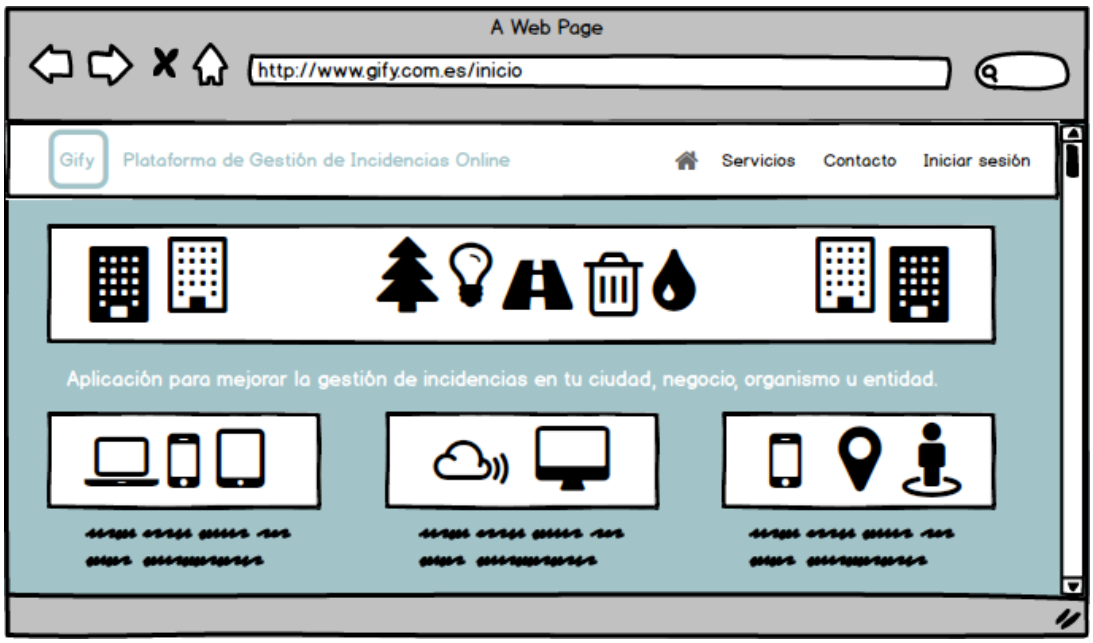


Figura 49: Mockup Landing- Page. Inicio. Elaboración propia.

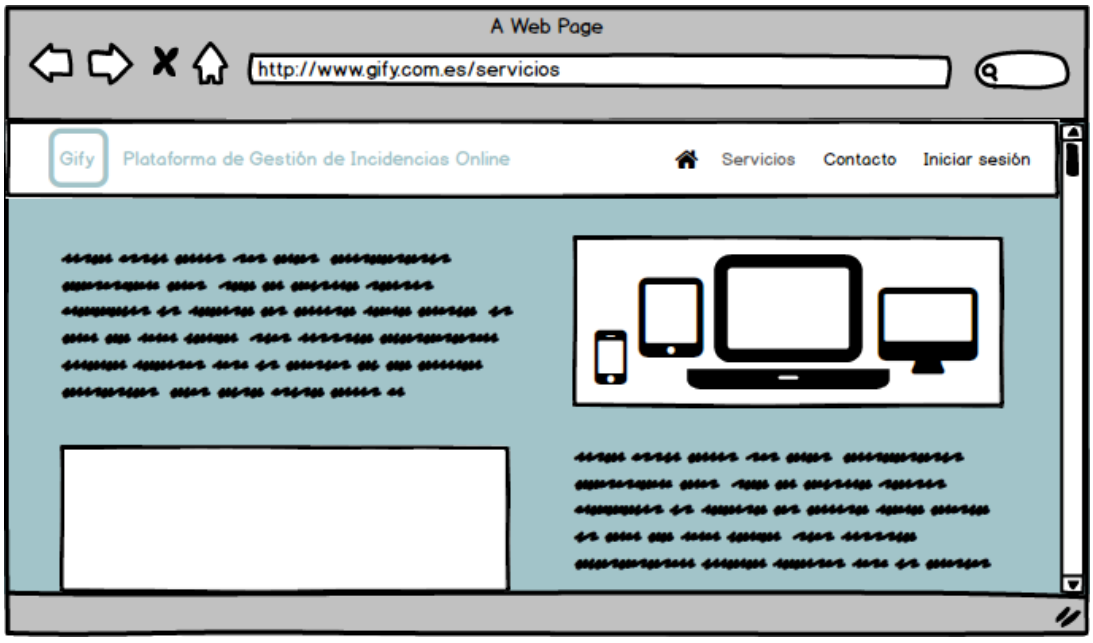


Figura 50: Mockup Landing-Page. Servicios. Elaboración propia.

Aquí se muestra la sección de Contacto, donde aparecen los datos de contacto de Gify y se proporciona un formulario para poder preguntar dudas u obtener información.

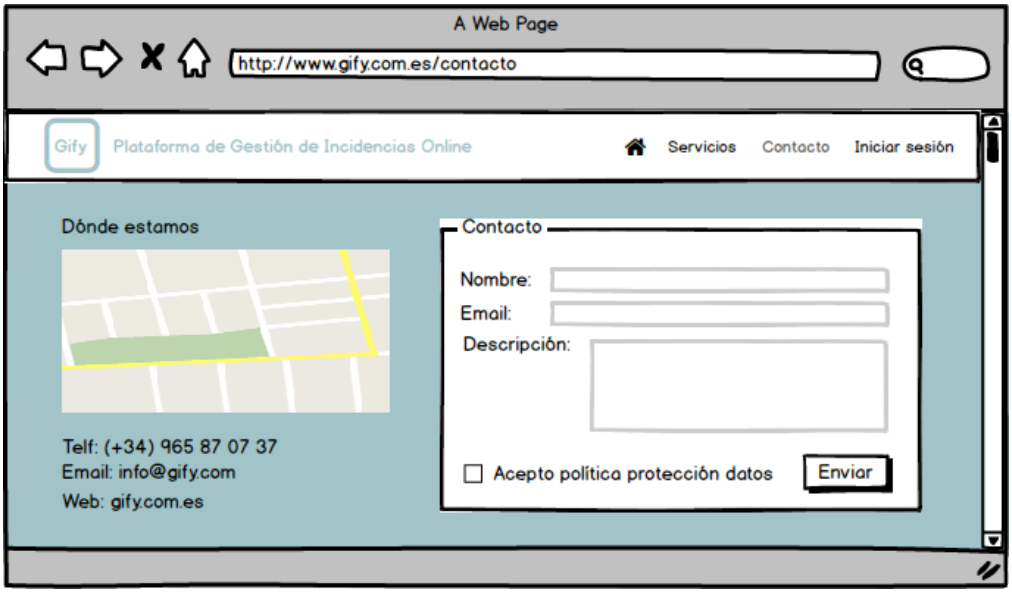


Figura 51: Mockup Landing-Page. Contacto. Elaboración propia.

En el *Mockup* de la sección Iniciar sesión, se representa un aspecto concreto del sistema; el ***Login***¸ que satisface el requisito funcional **RF02**. También se marcan los requisitos **RF01: Registro** y **RF32: Recuperación de contraseña**, que veremos en las siguientes imágenes.

Figura 52: Mockup Landing-Page. Inicio de sesión. Elaboración propia.

La representación del formulario de Registro satisface el requisito funcional **RF01: Registro.**

Figura 53: Mockup Landing-Page. Registro. Elaboración propia.

Este *Mockup* muestra el formulario de recuperación de contraseña, que satisface el **RF32: Recuperar contraseña**. Se ha marcado como opcional este requisito porque si no llegase a implementarse no interferiría en el correcto funcionamiento del sistema.

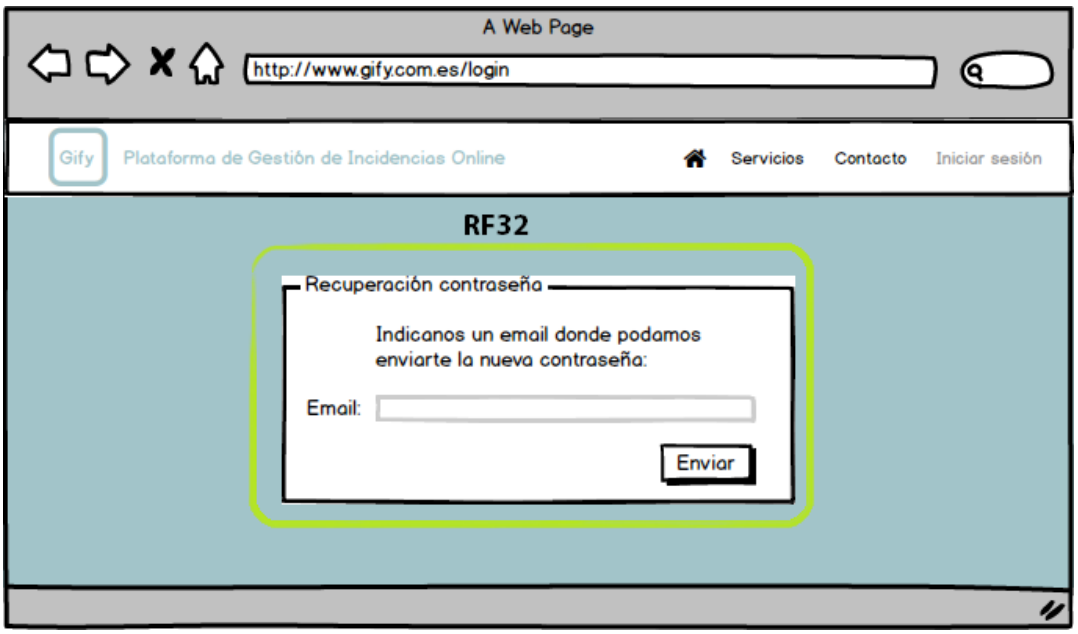


Figura 54: Mockup Landing-Page. Recuperar contraseña. Elaboración propia.

El panel de gestión web va a ser el mismo para los tres tipos de roles del sistema (propietario, gestores y operarios), en función de sus permisos y de las tareas que pueden abordar. Hay muchas vistas del panel que son iguales entre los diferentes usuarios, ya que representan las mismas secciones con los mismos requisitos funcionales.

En los siguientes mockups se van a mostrar el diseño de las maquetas de la parte web, evitando mostrar los mismos requisitos funcionales varias veces.

Usuario propietario

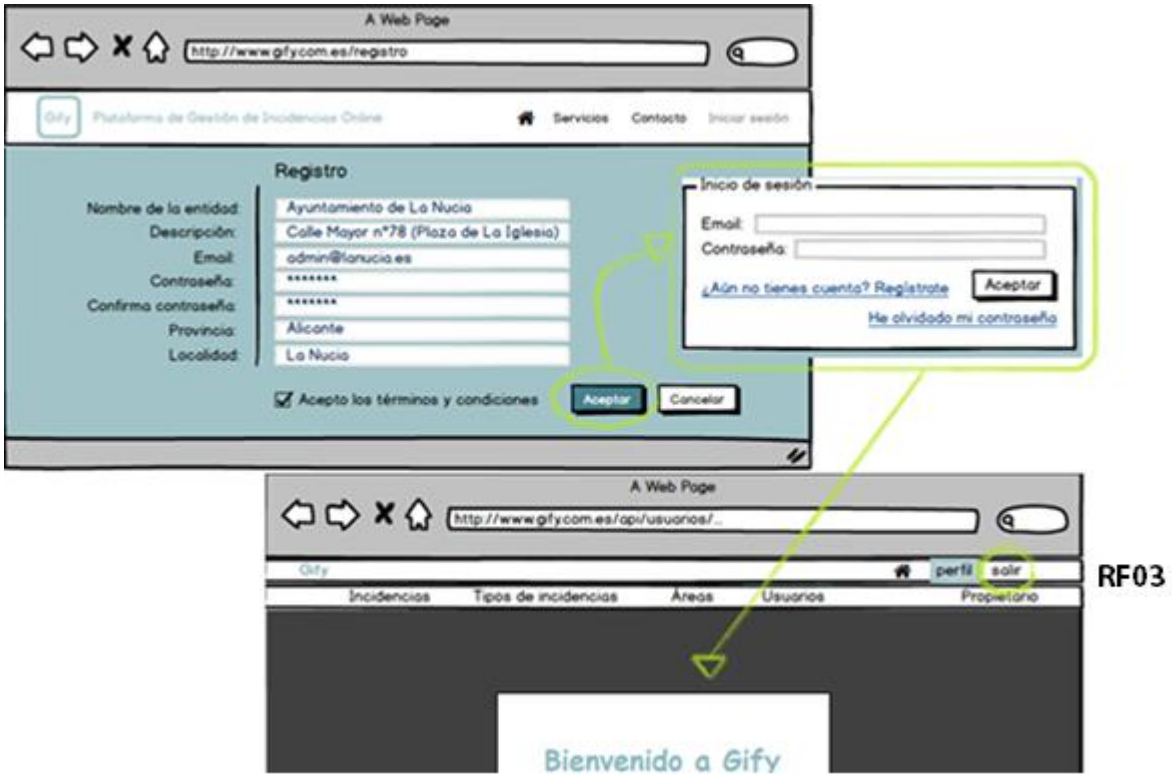


Figura 55: Mockup Landing-Page y Panel Propietario. Elaboración propia.

Una vez que un usuario registra la organización a la que pertenece en la plataforma, se convierte en el propietario de la cuenta. Ya puede iniciar sesión con los datos que ha proporcionado y entrar al panel de gestión de las incidencias.

En esta imagen se representa, además de algunos requisitos funcionales mencionados anteriormente, el **RF03: Salir (Logout)**, para cerrar la sesión de usuario.

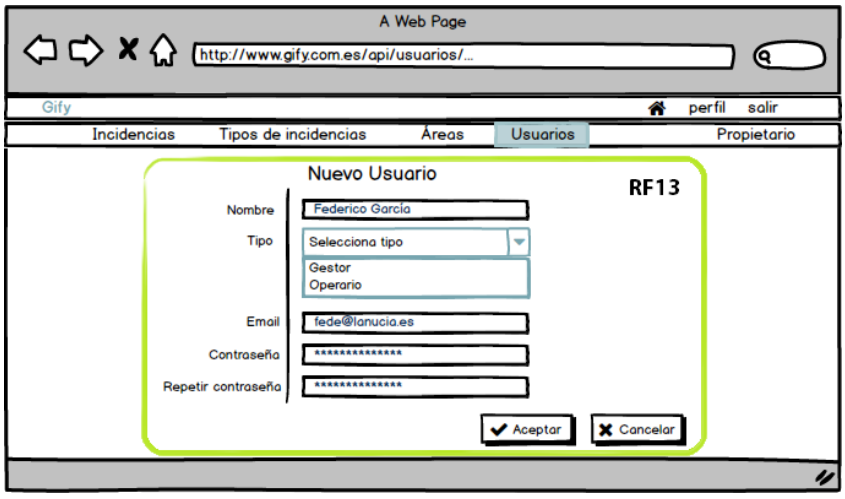


Figura 56: Mockup Panel Propietario - Nuevo usuario. Elaboración propia.

En la imagen anterior se representa el formulario de nuevo usuario donde pueden crearse gestores y operarios para que colaboren con las incidencias. Se satisface con este Mockup el requisito funcional **RF13: Nuevo usuario**. El requisito funcional **RF33: Editar datos de perfil** se plasma en la siguiente maqueta.

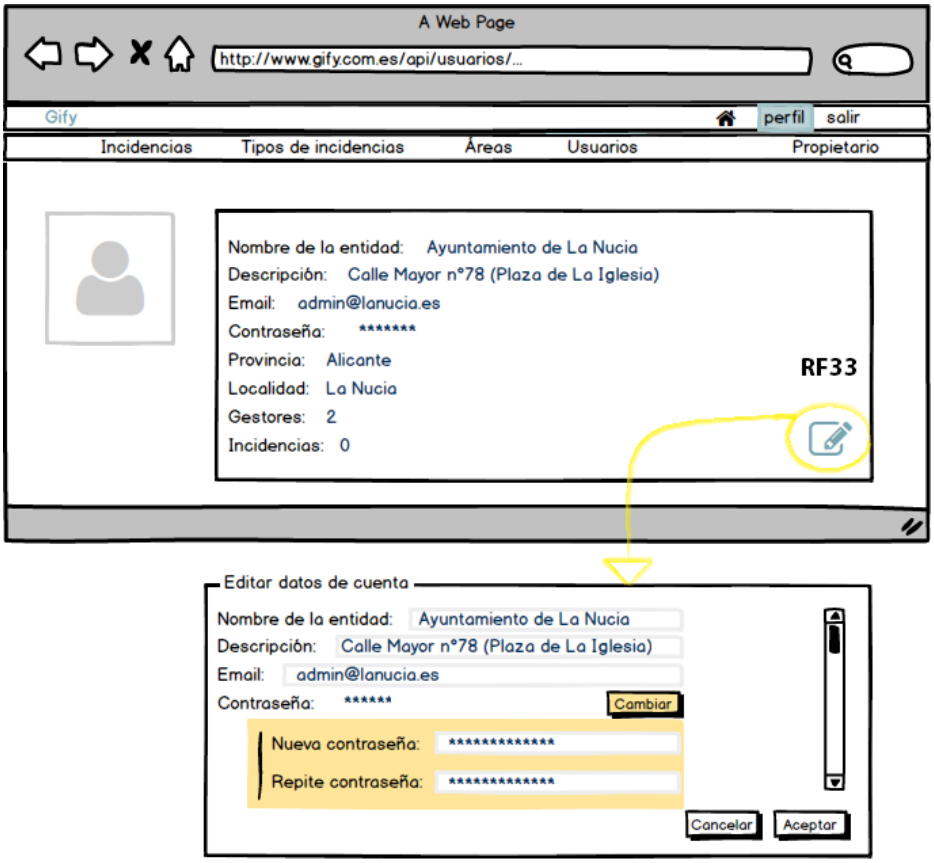


Figura 57: Mockup Panel Propietario - Editar datos de perfil. Elaboración propia.

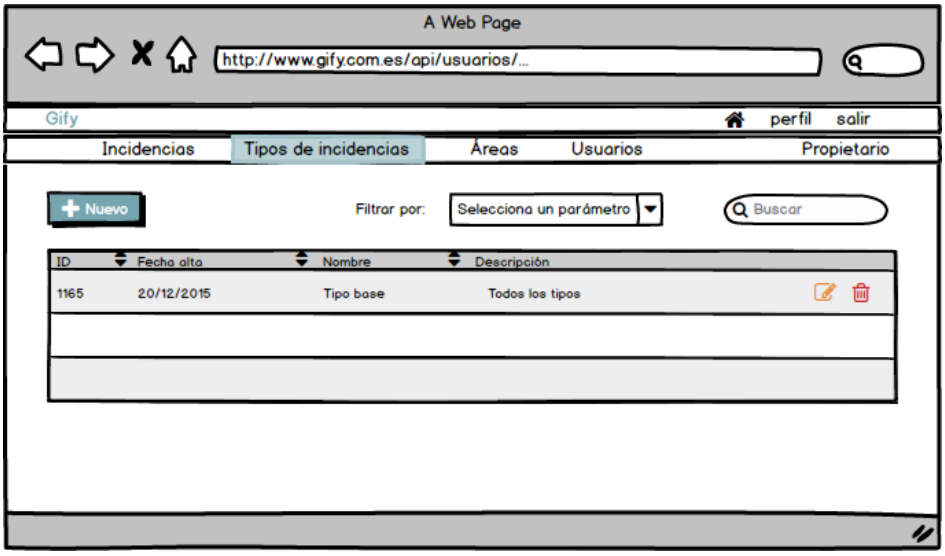


Figura 58: Mockup Panel Propietario - Tipo de incidencias – Tipo base. Elaboración propia.

Por defecto, cuando se crea una cuenta en la plataforma, se tiene un solo tipo de incidencia genérico. A cada incidencia hay que asignarle un tipo acorde con su tipología y si es necesario pueden crearse todos los tipos que hagan falta para clasificar las incidencias. Esta acción pueden realizarla todos los roles de usuario.

Lo mismo ocurre con las áreas. El sistema proporciona un área base en la que se clasificarán todas las incidencias. Si se necesitara añadir áreas para que la clasificación sea más concreta (barrios, facultades, o zona de jardinería, de obras…) puede hacerse, pero solo por parte de los gestores y el propietario de la cuenta.

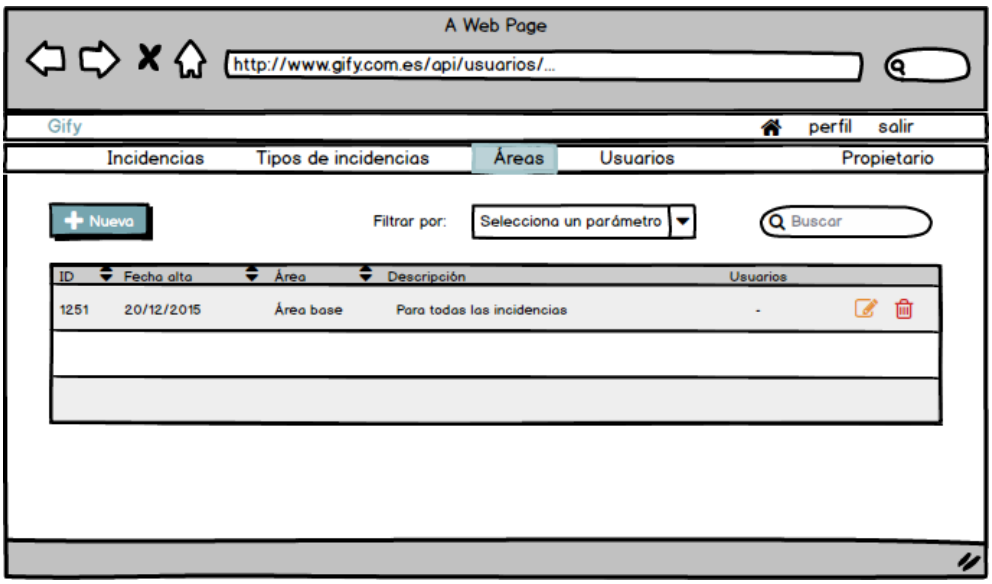


Figura 59: Mockup Panel Propietario - Áreas - Área base. Elaboración propia.

Usuario gestor

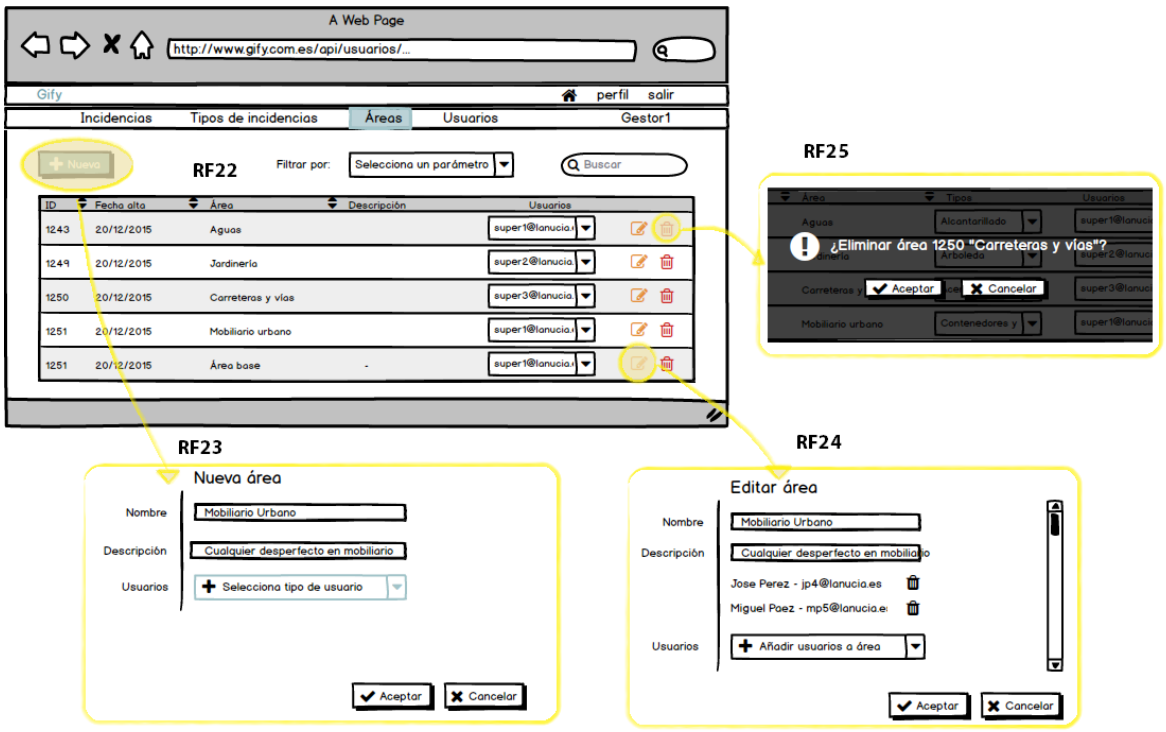


Figura 60: Mockup Panel de gestor – Áreas – Crear, editar, mostrar y eliminar. Elaboración propia.

Los gestores, igual que el propietario, tienen la opción de crear nuevas áreas, editarlas, eliminarlas y, por supuesto, visualizarlas en el panel. En este Mockup se representan los requisitos funcionales **RF22: Crear área, RF23: Mostrar área, RF24: Editar área** y **RF25: Eliminar área.**

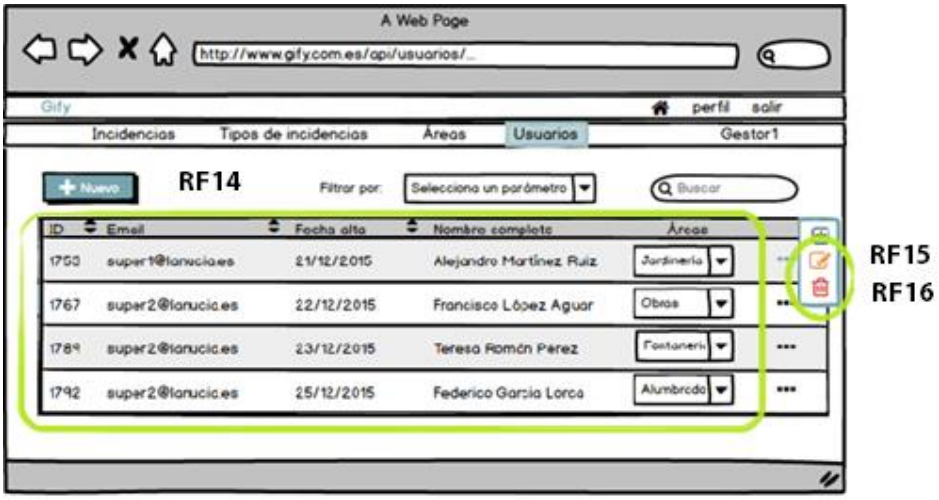


Figura 61: Mockup Panel Gestor - Usuarios – Mostrar, editar y eliminar. Elaboración propia.

En la imagen anterior, se muestra la sección de usuarios del panel de gestión. Los gestores pueden tener toda la información referente a los operarios de la plataforma y el propietario tiene también la información de los gestores.

Las funciones que pueden realizarse y que están definidas en los requisitos funcionales de esta maqueta son, además de crear usuario, **RF14: Mostrar usuario, RF15: Editar usuario y RF16: Eliminar usuario.** A continuación, vemos los mockups que definen estas vistas.

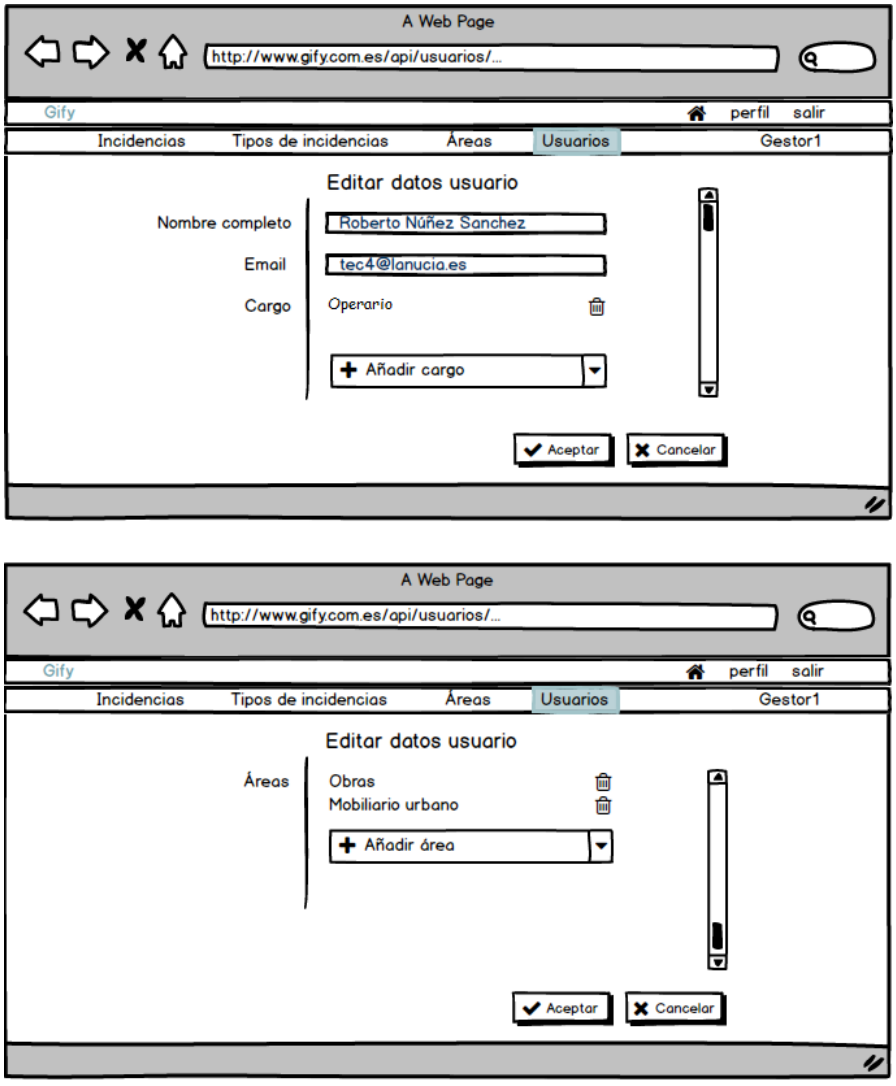


Figura 62: Mockup Panel Gestor - Usuarios - Editar usuario. Elaboración propia.

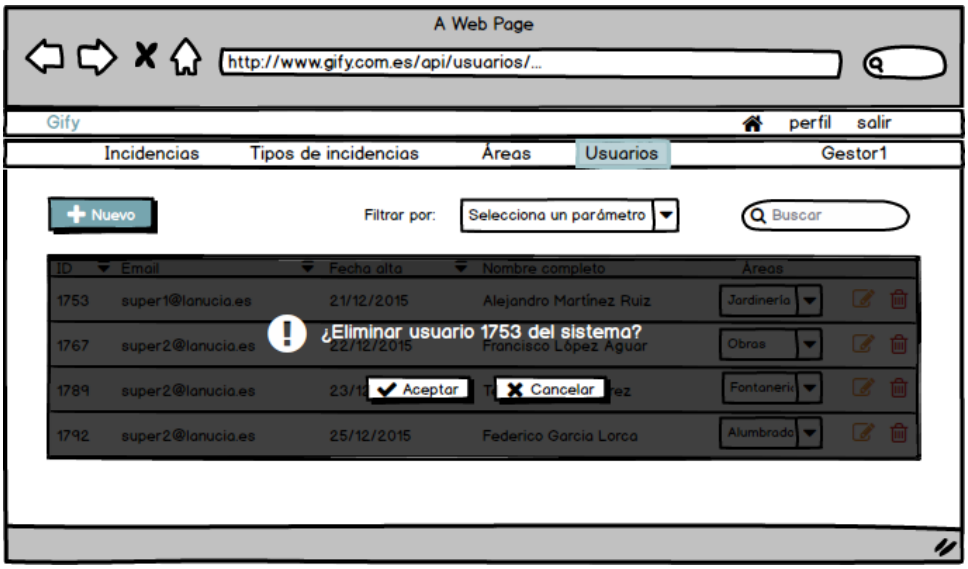


Figura 63: Mockup Panel Gestor - Usuarios - Eliminar usuario. Elaboración propia.

Cuando se crea un usuario, hay que seleccionar el cargo que tiene (gestor u operario) y/o el área o áreas a las que pertenece. Por ejemplo, un usuario puede ser gestor y al mismo tiempo operario de un área concreta. U operario de dos áreas.

Este sistema está indicado para organismos pequeños o medianos, y lo más usual, es que, dentro de la organización del mantenimiento de ese ayuntamiento, universidad o centro comercial, haya definidas varias áreas de trabajo, varios operarios en cada área, y algunos gestores jefes de grupo, que supervisen las reparaciones. En este caso, cada usuario tendría su rol en el sistema, sin compartir funciones. Pero puede ser que la entidad que se registre en la plataforma sea muy pequeña, tenga pocos trabajadores y, por tanto, tengan varias funciones cada uno. Esta es la razón por la que se da un poco de flexibilidad al cliente, a la hora de crear usuarios, para que encuentre la manera de organizarse que más se adecúe a sus necesidades.

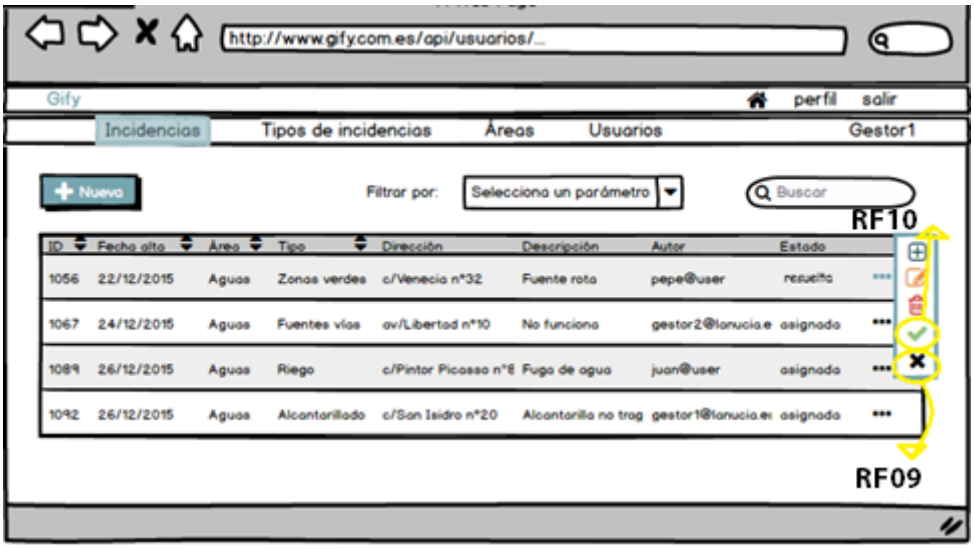


Figura 64: Mockup Panel Gestor - Incidencias - Asignar y Cerrar. Elaboración propia.

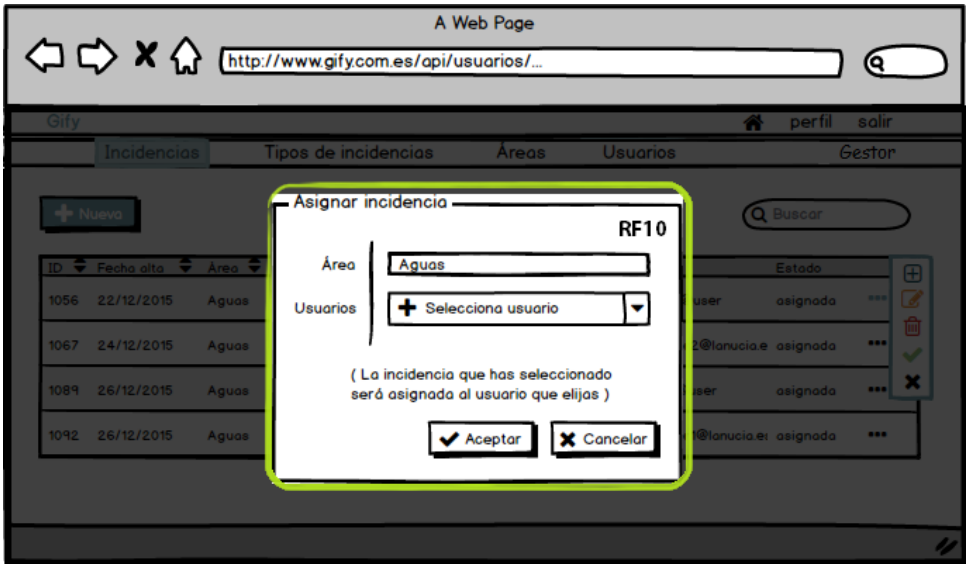


Figura 65: Mockup Panel Gestor - Incidencias - Asignar incidencia. Elaboración propia.

Figura 66: Mockup Panel Gestor - Incidencias - Cerrar incidencia. Elaboración propia.

Las incidencias deben ser asignadas a operarios para que puedan resolverlas. Esta función la realizan los gestores y el propietario. Una vez que la incidencia tiene responsable, el gestor o propietario queda a la espera de que el operario la resuelva y cambie el estado a "resuelta", y así poder cerrarla.

En los Mockup anteriores se representan los requisitos funcionales **RF10: Asignar incidencia** y **RF09: Cerrar incidencia.**

Usuario operario

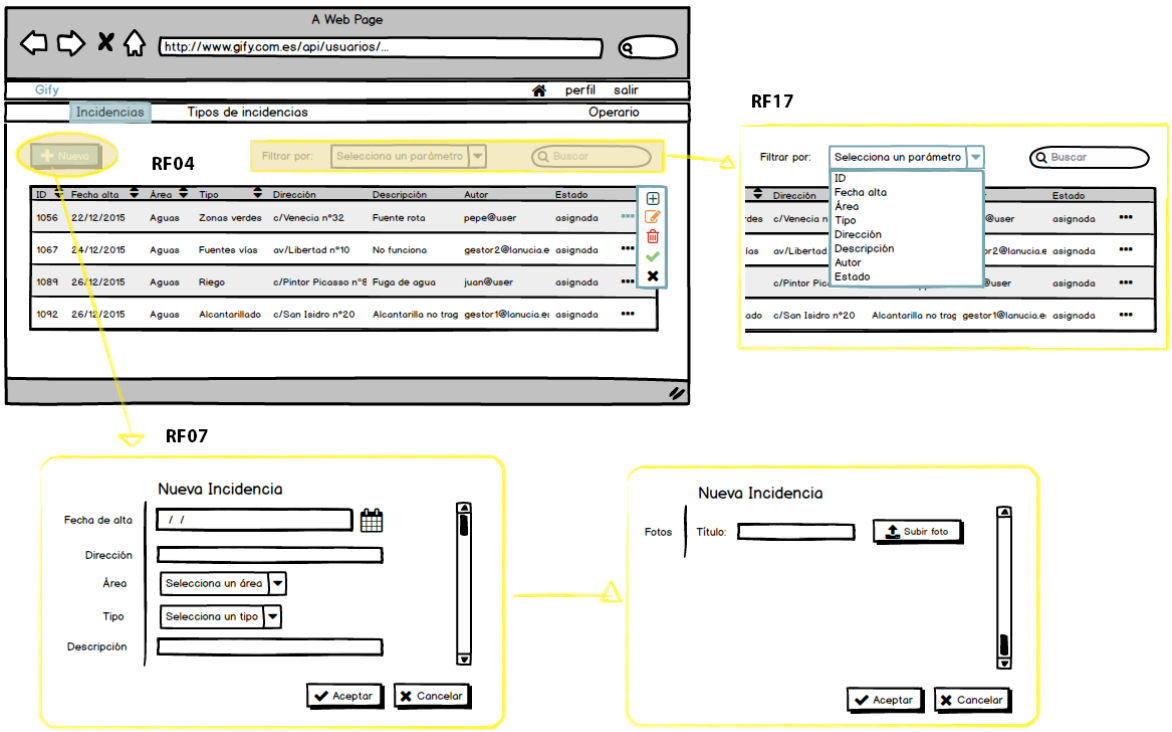


Figura 67: Mockup Panel Operario - Incidencias – Mostrar, Nueva y Buscar por. Elaboración propia.

.

En el panel de gestión de cada usuario, dependiendo de su función, aparecerán unas incidencias u otras. Por ejemplo, en el panel de los operarios, ya sean de un área o de varias, aparecerán las incidencias que les hayan asignado. Si un usuario es gestor, le aparecerán todas las incidencias que vayan llegando al sistema y el tendrá que ir asignándolas.

En la sección de incidencias, como se puede observar en el Mockup se pueden realizar varias funciones: visualizar las incidencias correspondientes, buscar incidencias, añadir, editar, eliminar, rechazar la incidencia, marcar como resuelta y otras funciones que veremos más adelante en este documento.

Los requisitos que satisface este Mockup en concreto son los **RF17: Formulario búsqueda**, **RF07: Crear incidencia** y **RF04: Mostrar incidencias.**

La funcionalidad de búsqueda que se ha representado en esta vista está en todas las secciones del panel de gestión y para todos los usuarios.

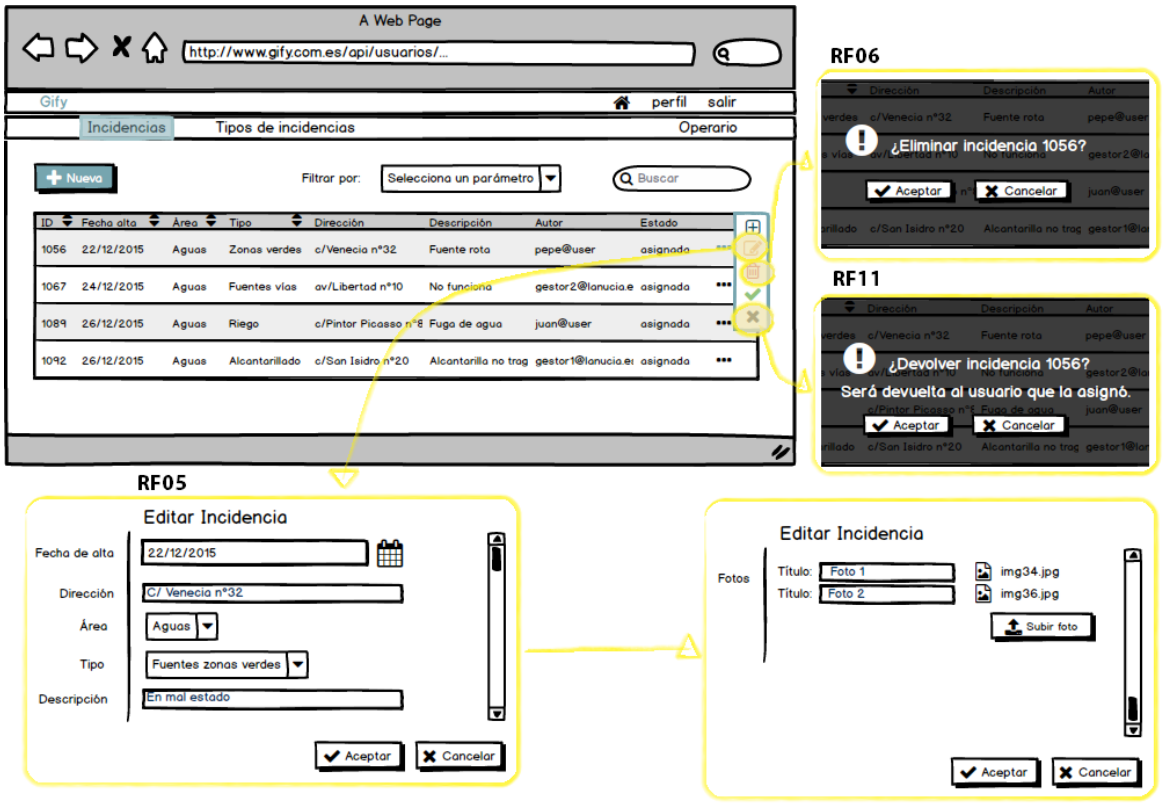


Figura 68: Mockup Panel Operario - Incidencias - Editar, Eliminar, Devolver y cambiar. Elaboración propia.

En esta maqueta se representan otros tres requisitos funcionales que son el **RF05: Editar incidencia**, el **RF06: Eliminar incidencia** y el **RF11: Rechazar incidencia**.

Los operarios pueden devolver las incidencias cuando los gestores o el propietario se han equivocado al asignarlas. Devolver una incidencia quiere decir que el estado de la misma vuelve a un punto anterior.

Una incidencia pasa por varios estados en su ciclo de vida en el sistema. Cuando se registra en la base de datos, el estado inicial de la nueva incidencia es “pendiente”, y sigue así hasta que un gestor o propietario le buscan un responsable. En este momento su estado pasaría a “asignada”.

Figura 69: Mockup Panel Operario - Incidencias - Histórico, Aceptar incidencia e Incidencia Resuelta. Elaboración propia.

Cuando la incidencia tiene el estado de "asignada", el botón de aceptar sirve para cambiarla a "en proceso" y empezar a resolverla. Así se refleja en la maqueta, satisfaciendo el requisito funcional **RF08: Aceptar incidencia**.

Todo por lo que va pasando la incidencia desde que se registra, automáticamente se va mostrando en un histórico que puede consultarse desde el botón "Más" del panel. Esta tarea corresponde al requisito **RF18: Generar histórico de incidencia**.

Una vez el problema está resuelto, hay que marcar la incidencia de nuevo para que su estado cambie a "resuelta" y pueda cerrarse el trámite (**RF12: Incidencia resuelta**).

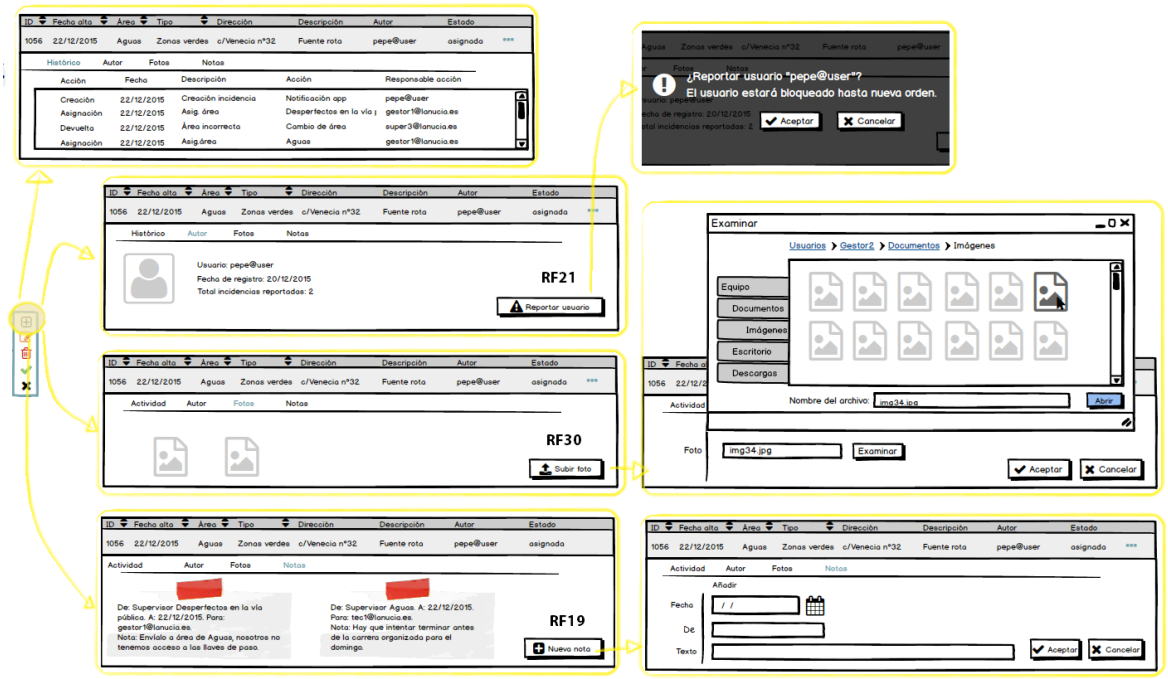


Figura 70: Mockup Panel Operario - Incidencias - Subir foto, Añadir nota y Reportar usuario. Elaboración propia.

Dentro del apartado "Más" de cada incidencia, se encuentran además del histórico, varias funcionalidades más.

Existe la posibilidad de reportar un usuario molesto que entorpece el trabajo de los usuarios de la plataforma. Reportarlo significa que no podrá enviar más incidencias a través de la app móvil a menos que se ponga en contacto con el proveedor del servicio a través del formulario de contacto de la Landing page. Son los propios usuarios de la plataforma quienes deben valorar cuándo un ciudadano debe ser baneado. El requisito funcional que representa esta función es **RF21: Reportar usuario.**

Existe la posibilidad de subir más fotos de la incidencia, que se hayan tomado en la reparación o que sean necesarias por algún motivo. Y también de eliminar las fotos que pueda tener adjuntas la incidencia. Los requisitos **RF30: Subir foto** y **RF31: Eliminar foto** cubrirían estas tareas.

Por último, pueden añadirse notas a una incidencia. Esto es porque un operario o gestor puede tener la necesidad de añadir información relativa a la incidencia o su reparación. También puede eliminarse si se desea. En la maqueta están representados los diseños correspondientes a los **requisitos RF19: Añadir nota** y **RF20: Eliminar nota**.

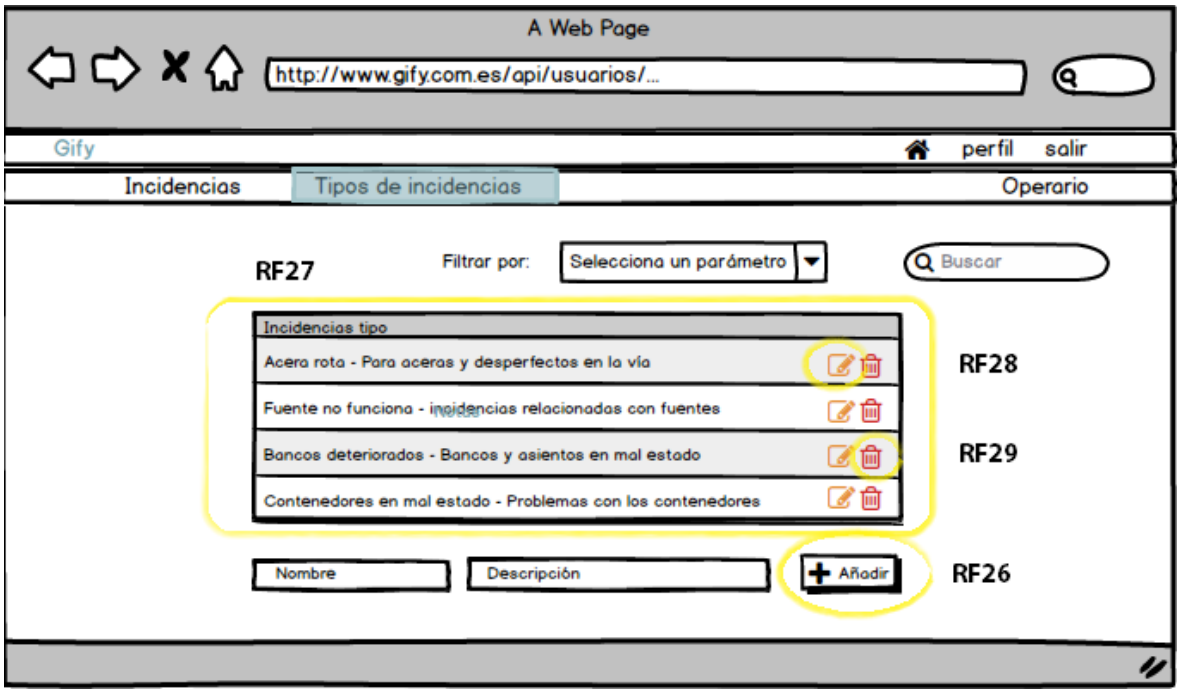


Figura 71: Mockup Panel Operario - Tipo de incidencias - Mostrar, Añadir, Editar y Eliminar. Elaboración propia.

Respecto a los tipos de incidencia, podemos ver en la maqueta que el diseño y las funcionalidades son las mismas que en las demás secciones.

Se muestran en pantalla los tipos que han creado los usuarios de la plataforma, adaptándose a las incidencias que suelen tener. Se añade también una descripción por si fuera necesaria alguna información relativa al tipo.

Representando esta acción se ha plasmado en el Mockup el requisito funcional **RF27: Mostrar tipos de incidencias**. También, relacionamos los demás requisitos con las funciones básicas de gestión: **RF26: Crear tipo de incidencia**, **RF28: Editar tipo de incidencia** y **RF29: Eliminar tipos de incidencia.**

Usuario ciudadano - *app* movil

Figura 72: Mockups App Móvil - Requisitos funcionales. Elaboración propia.

En el panel de inicio de la app encontramos dos apartados: añadir incidencia y consultar incidencias (**RF38: Enviar incidencia** y **RF40: Consultar mis incidencias**).

Si se quiere notificar una incidencia, el primer paso es escoger la entidad de la cual se desea enviar el aviso (**RF37: Seleccionar entidad**), seleccionando primero la provincia y el municipio. Una vez realizado esto, se selecciona el área a la cual corresponde la incidencia y se especifica la dirección de la avería. (**RF41: Adjuntar ubicación).** Luego, se carga una imagen del problema (**RF42: Adjuntar fotografía**), se escoge el tipo de incidencia que más encaje con la tipología de la incidencia (**RF40: Seleccionar área y tipo**) y puede añadirse un texto corto descriptivo de la incidencia (**RF39: Enviar descripción**).

Por último, antes de enviar, se pide al usuario que inicie sesión **(RF35: *Login)*** en la aplicación, y si no tuviera usuario, se le da la posibilidad de registrarse (**RF34: Registro**). Normalmente, el registro en una aplicación se muestra nada más entrar en ella, pero en este caso, se ha creído conveniente poner este paso justo antes de enviar la incidencia para evitar que los ciudadanos dejen la notificación a medias. Muchos usuarios consideran tedioso tener que registrarse

en las *apps* móviles para poder utilizarlas. Sin embargo, si este paso se pone justo antes de terminar, el esfuerzo de haber realizado los anteriores pasos hace que el usuario termine registrándose.

Cuando la incidencia se ha enviado, puede ir consultándose su estado en el apartado “mis incidencias”.

Base de datos

De acuerdo con las características de este sistema y después de identificar los requisitos funcionales y realizar el diseño de *mockups*, podemos hacernos una idea de los datos que vamos a tratar.

Antes de empezar a desarrollar, hay que representar todas las entidades de nuestro modelo. En el caso de este sistema, el diseño de datos se presenta de la siguiente forma:

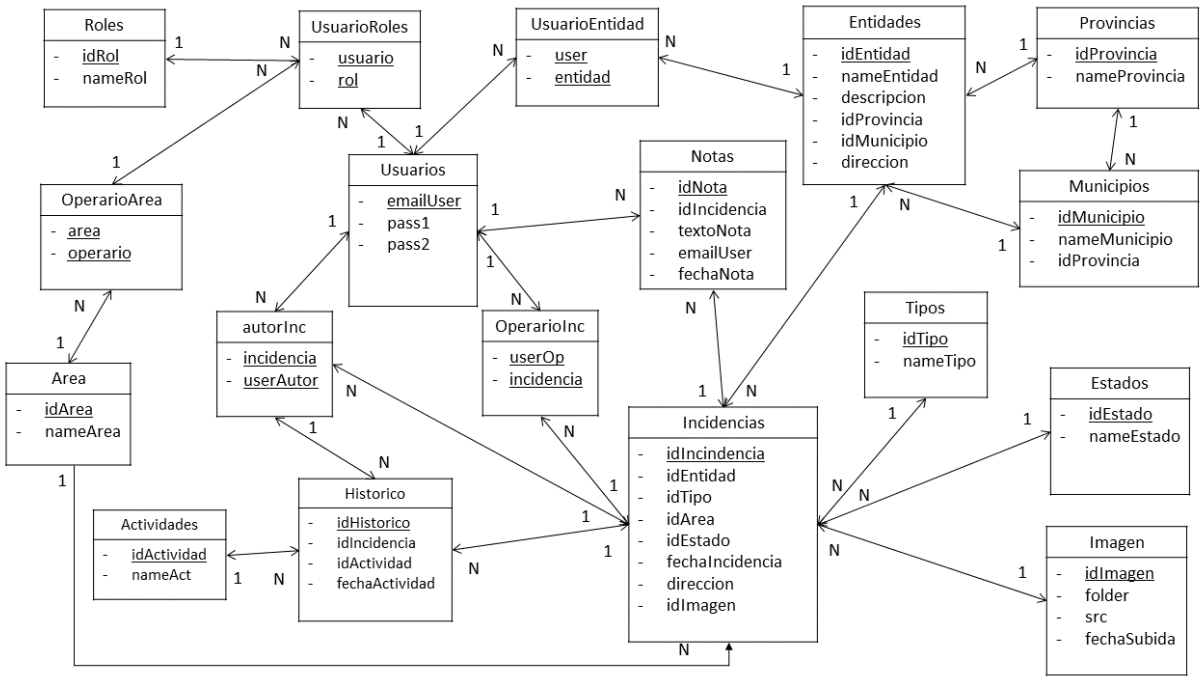


Figura 73: Diagrama del diseño de la base de datos del sistema. Elaboración propia.

Entidades:

- Incidencias
- Usuarios
- Entidades
- Notas
- Relación operario - incidencia
- Tipos
- Estados
- Imagen

- Municipios
- Provincias
- Relación usuario-entidad
- Relación usuario-roles
- Roles
- Relación usuario autor-incidencia
- Área
- Histórico
- Actividades
- Relación operario-area

Se utilizará MySQL para desarrollar la base de datos. Para crear la conexión con las aplicaciones (web y móvil) PHP se usará PDO, La extensión *Objetos de* Datos de PHP que define una interfaz ligera para poder acceder a bases de datos en PHP.

Diseño de Interfaces

El diseño de interfaz de usuario es el diseño de dispositivos de comunicación móvil, aplicaciones de software y sitios web enfocado en la experiencia de usuario y la interacción.

Normalmente, es una actividad multidisciplinar que involucra a varias como el diseño gráfico, industrial, web, de software y la ergonomía; y está implicado en un amplio rango de proyectos, desde sistemas para ordenadores, móviles, relojes inteligentes, vehículos y hasta aviones comerciales.

Su objetivo es que las aplicaciones o los objetos sean más atractivos y, además, hacer que la interacción con el usuario sea lo más intuitiva posible, conocido como el diseño centrado en el usuario.

Nombre y logo

La idea que dio forma al nombre del proyecto/plataforma fue muy sencilla; gracias a la unión de las siglas de **G**estión de **I**ncidencias, junto a la terminación **- *fy***, tan de moda entre las apps móviles, surgió ***Gify***.

El objetivo era encontrar un nombre corto, pegadizo y que sonara *"internacional"*. Lo más *moderno* posible sin caer en un *spanglish* poco profesional.

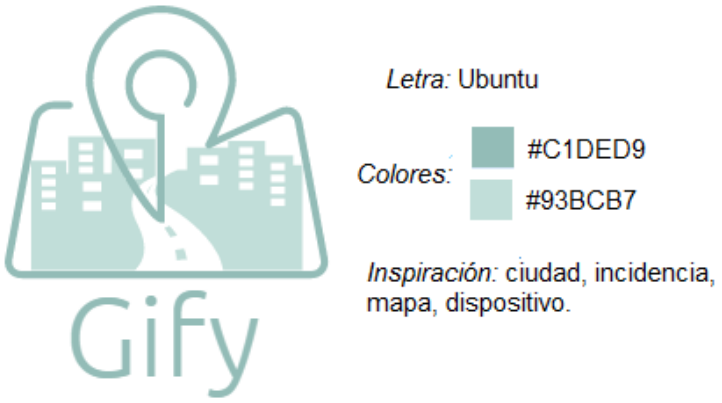


Figura 74: Logo Gify. Elaboración propia.

La sensación que se quiere transmitir con el diseño de las vistas es de tranquilidad, limpieza y orden. Este efecto lo consiguen los fondos blancos en la mayoría de vistas y algunos toques de color.

Los colores escogidos para el logo, y para algunos elementos más, son dos tonos parecidos, que combinan bien y aportan una elegancia natural sobre el blanco.

La tipografía del logo es de un estilo sans-serif humanístico y se utiliza de forma predeterminada en las interfaces de Ubuntu. Esta elección, en concreto, es puramente estética, ya que casa muy bien con las líneas que enmarcan el icono del logo.

Respecto a los tipos de letra utilizados en la aplicación web y móvil han sido de color negro y de estilo Helvetica y Arial, en su mayoría. Como se ha comentado, se busca la simplicidad y fácil lectura (gracias al contraste blanco-negro).

La parte donde más colores y contrastes hay es en la *Landing page*, que por el cometido que tiene en la difusión de un proyecto, es interesante que sea llamativa.

Interfaces

En las siguientes páginas, se muestra el diseño de las interfaces de la *Landing*, de la aplicación web y de la aplicación móvil. Se han incluido todas las vistas necesarias para formar una idea clara del aspecto que tiene el sistema, intentado mostrar todas las funcionalidades que puede ofrecer.

Son prototipos de diseño reales del sistema que sirven de guía visual para representar fielmente el esqueleto de las aplicaciones y la web.

Landing-Page

Una *landing page*, o página de aterrizaje, **e**s una página web diseñada específicamente para convertir visitantes en clientes. El funcionamiento es sencillo: si ofrecemos algo que seduzca al usuario, éste estará más dispuesto a dejar información a través de un formulario, si con ello va a conseguir acceso a éste y otros contenidos de interés.

En este caso, para realizar la web de Gify, se ha utilizado una plantilla específica de Bootstrap. Se han modificado muchos de los elementos para adecuar la página a lo que se quería conseguir.

No es solo una página lanzadera solo con una función informativa, si no, que será el portal por el que pasen todos los usuarios para conectar con el sistema.

Figura 75: Interfaz Landing Page - Home. Elaboración propia.

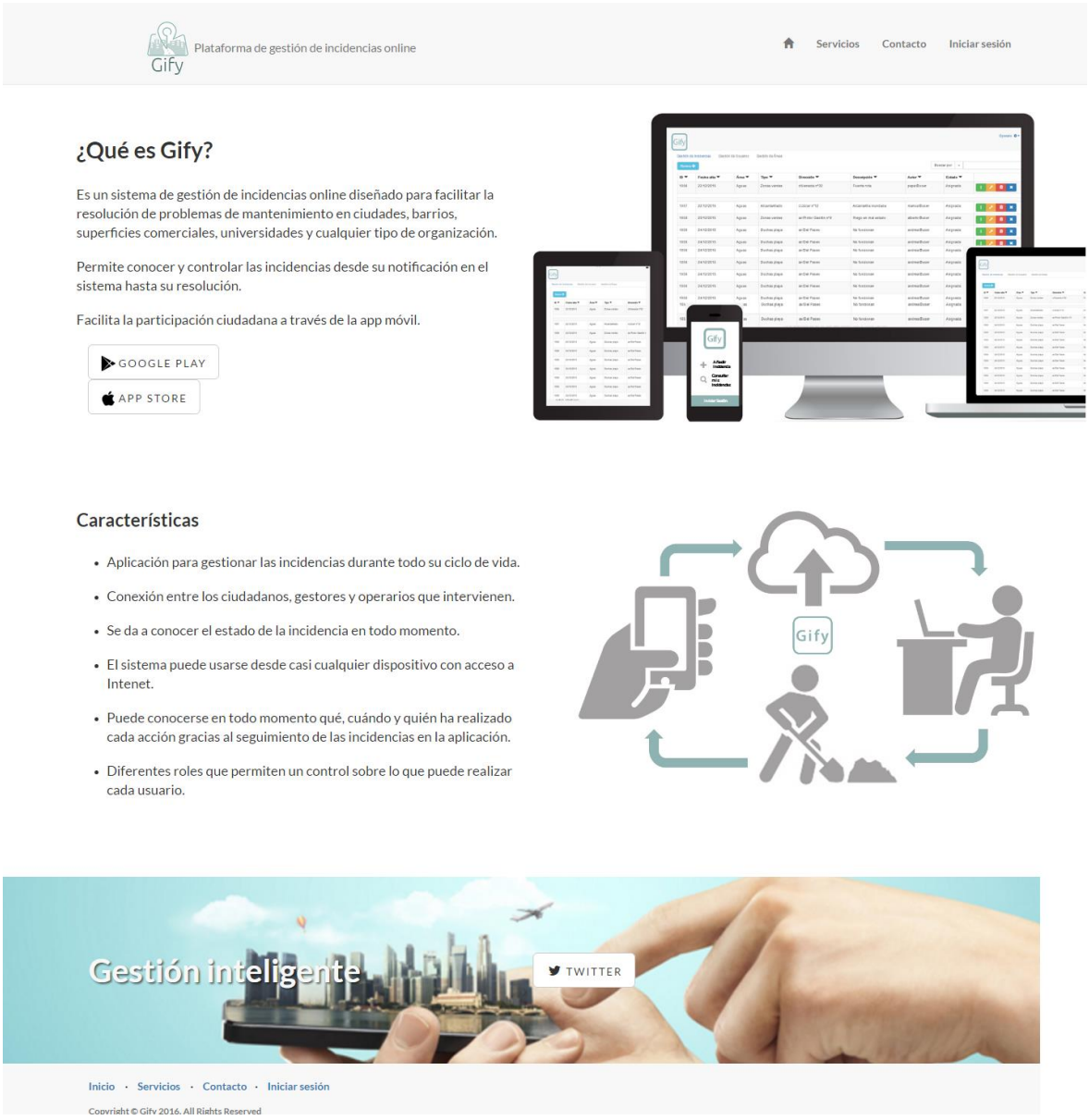


Figura 76: Interfaz Landing Page - Servicios. Elaboración propia.

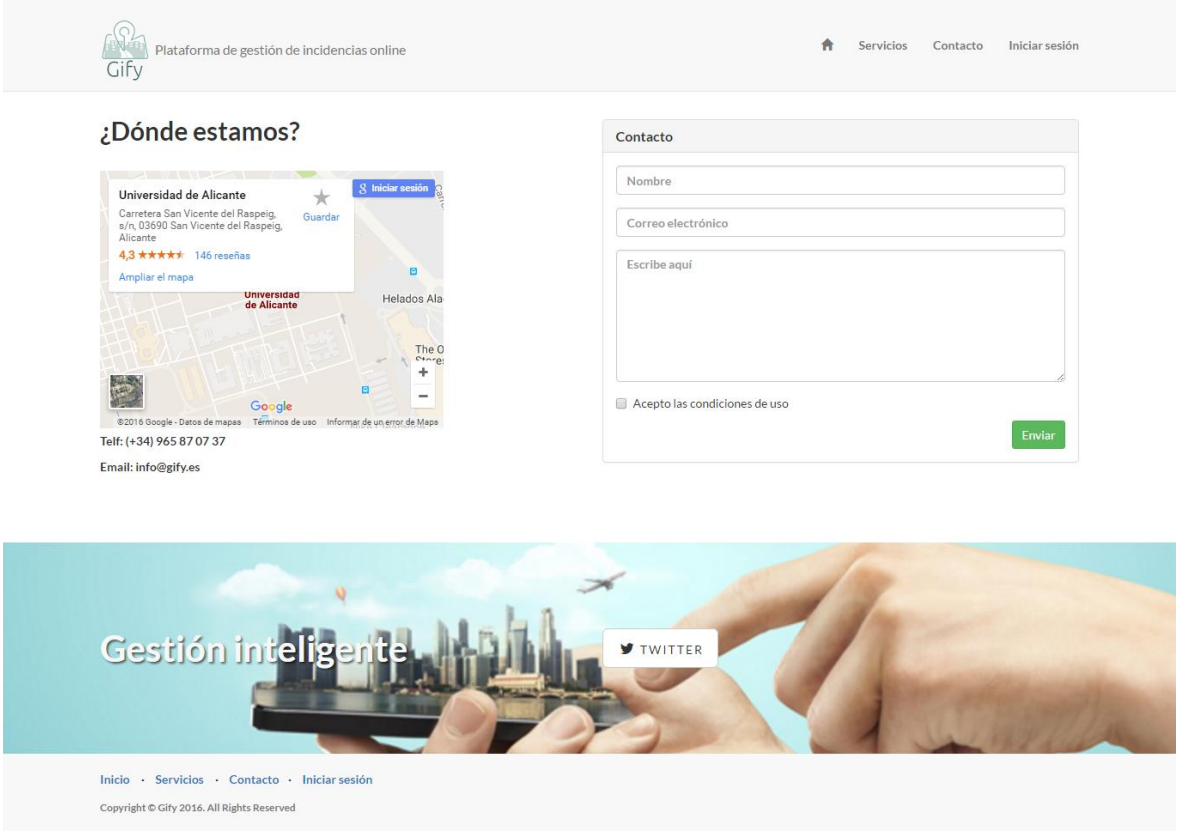


Figura 77: Interfaz Landing Page - Contacto. Elaboración propia.

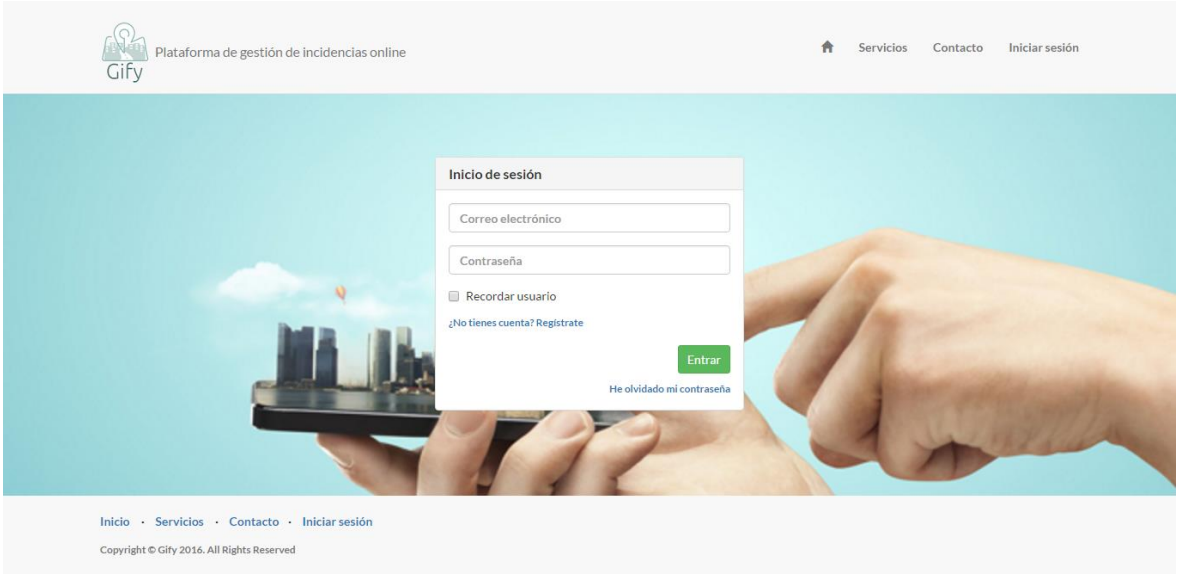


Figura 78: Interfaz Landing Page - Iniciar sesión. Elaboración propia.

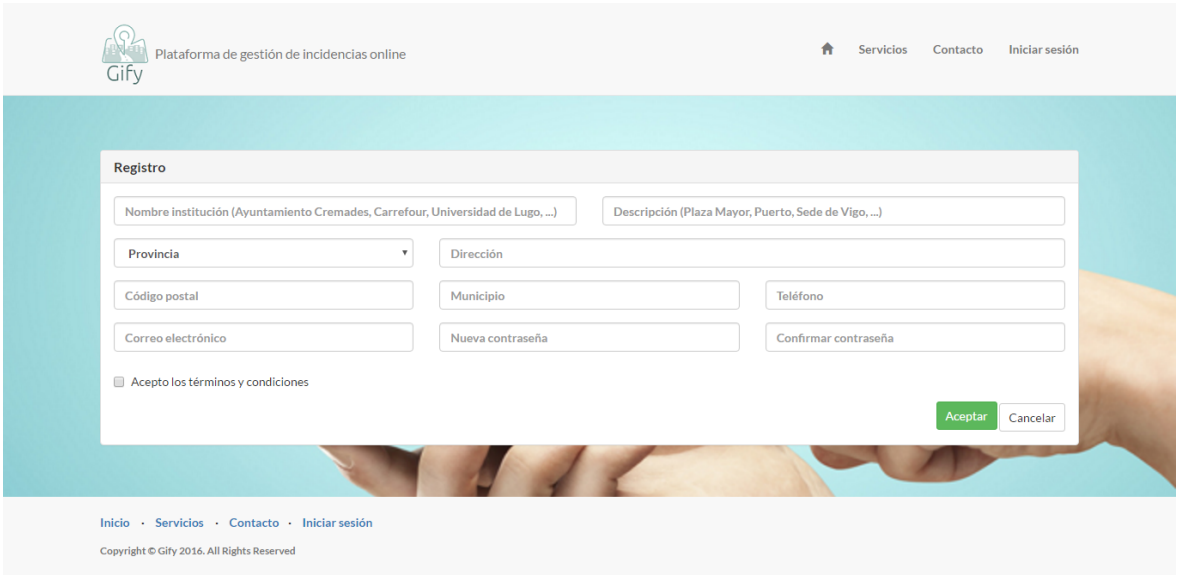


Figura 79: Interfaz Landing Page - Registro. Elaboración propia.

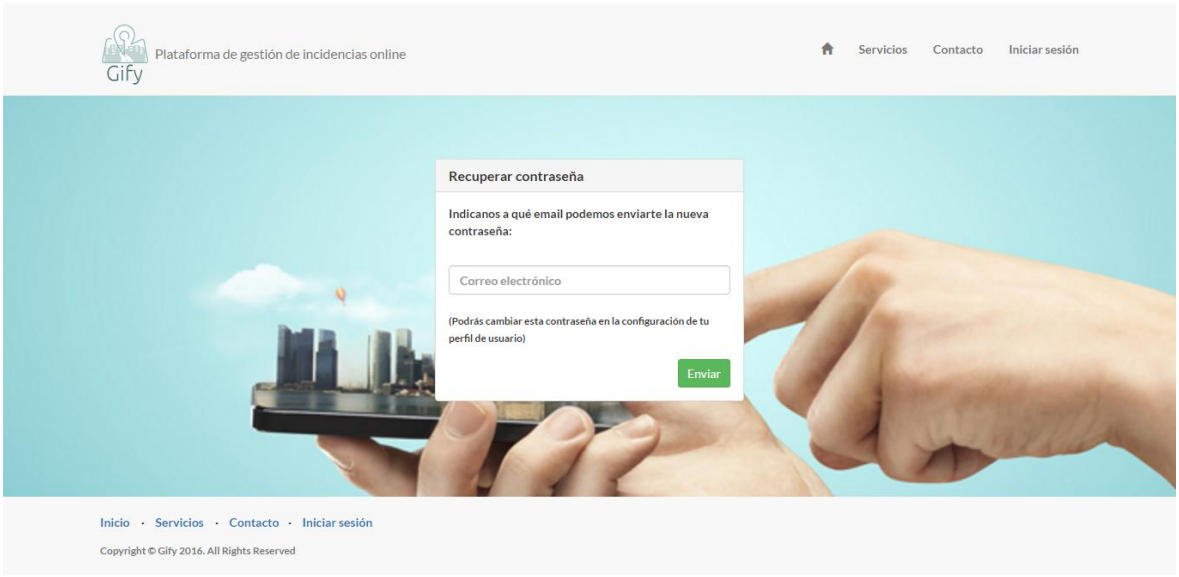


Figura 80: Interfaz Landing Page - Recuperar contraseña. Elaboración propia.

En esta sección se representan las vistas de la aplicación web desde los paneles de gestión de los diferentes usuarios.

Panel de gestión propietario y gestor

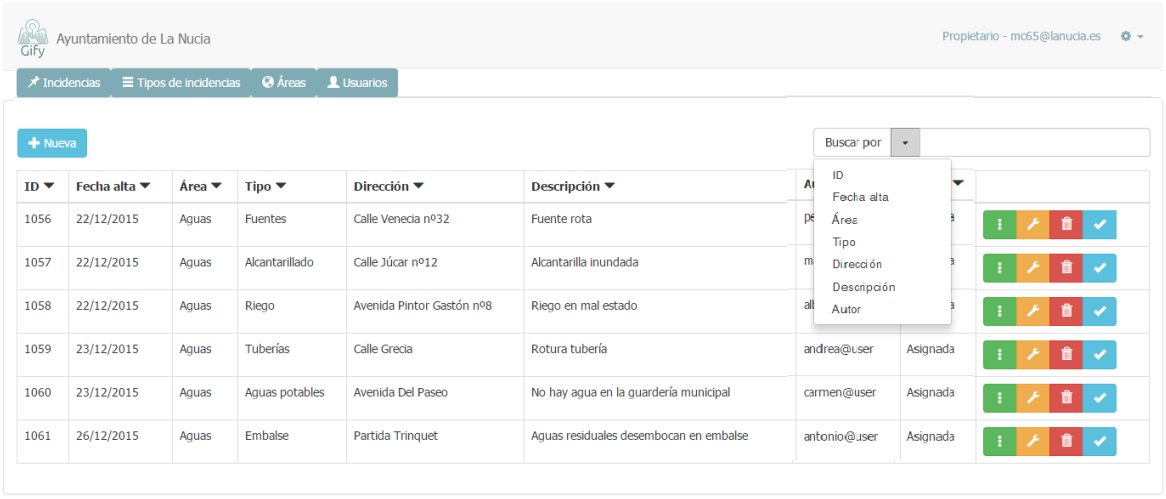


Figura 81: Interfaz Panel de gestión propietario / gestor – Incidencias - Buscar. Elaboración propia.

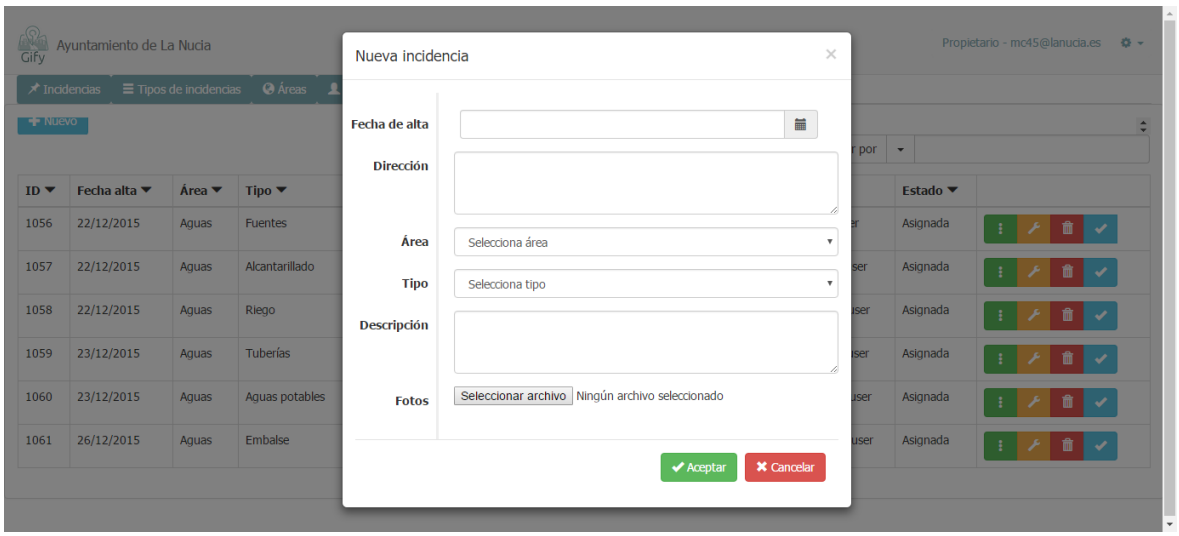


Figura 82: Interfaz Panel de gestión propietario / gestor – Incidencias - Nueva. Elaboración propia.

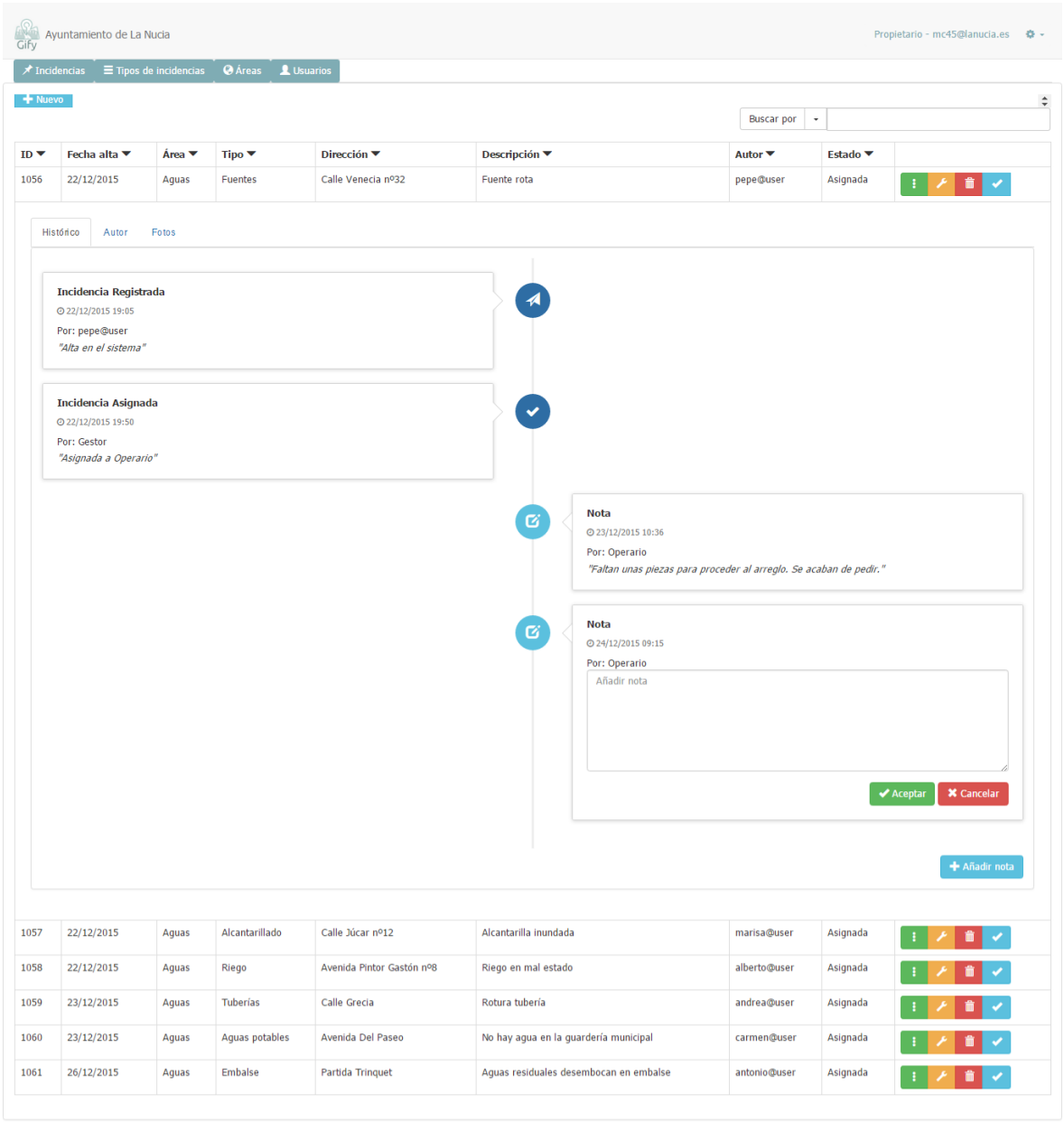


Figura 83: Interfaz Panel de gestión propietario / gestor – Incidencias – Histórico – nueva nota. Elaboración propia.

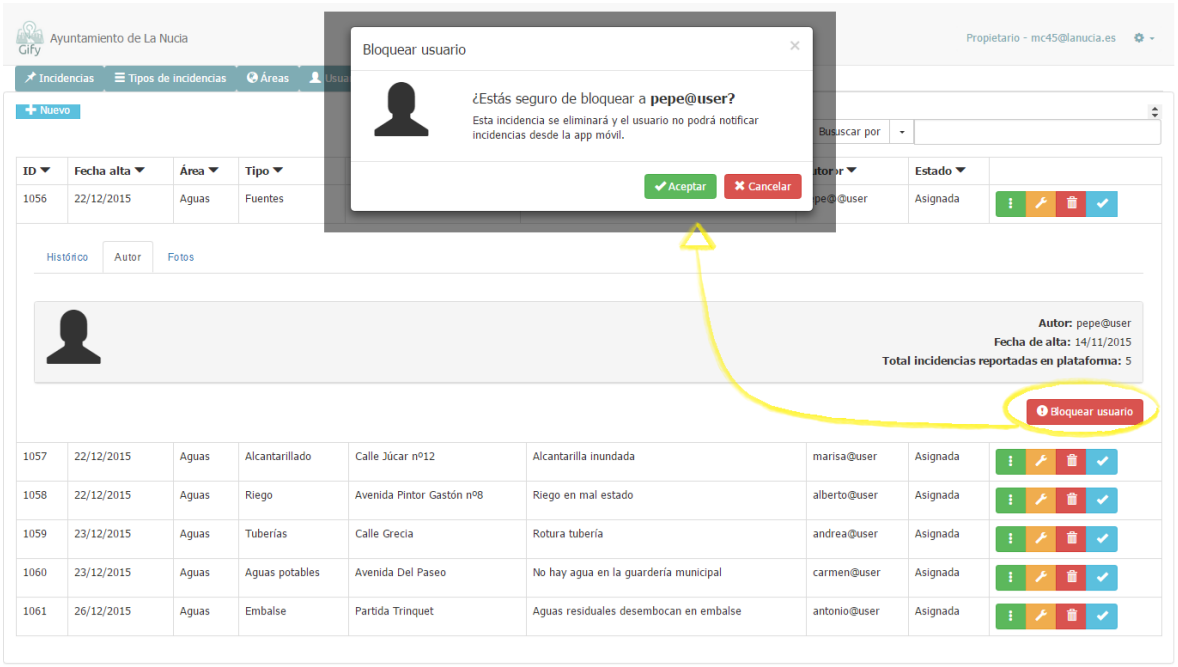


Figura 84: Interfaz Panel de gestión propietario / gestor – Incidencias – Histórico – Autor – Bloquear. Elaboración propia.

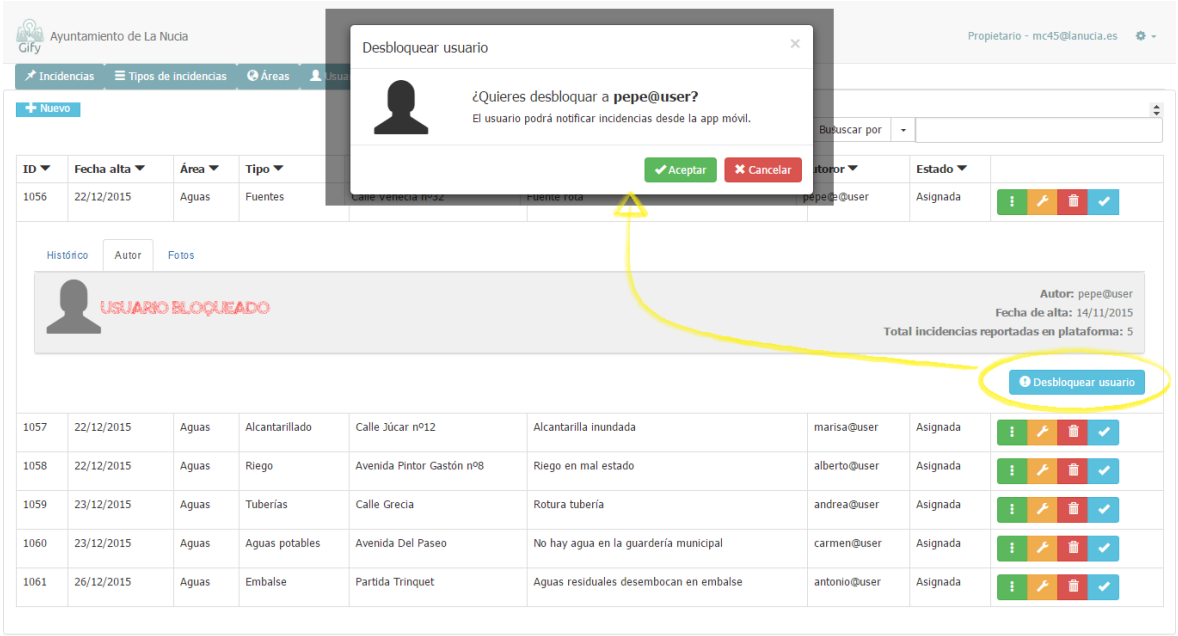


Figura 85: Interfaz Panel de gestión propietario / gestor – Incidencias – Histórico – Autor – Desbloquear. Elaboración propia.

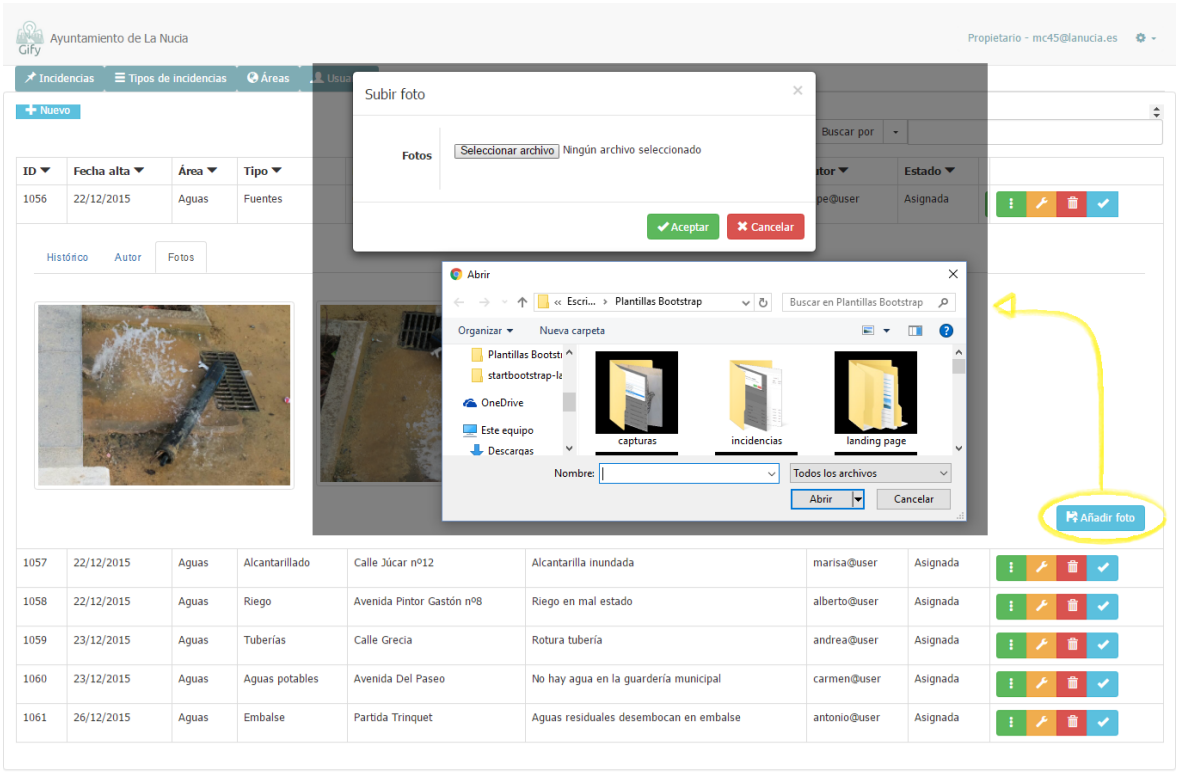


Figura 86: Interfaz Panel de gestión propietario / gestor – Incidencias – Histórico – Fotos – Subir. Elaboración propia.

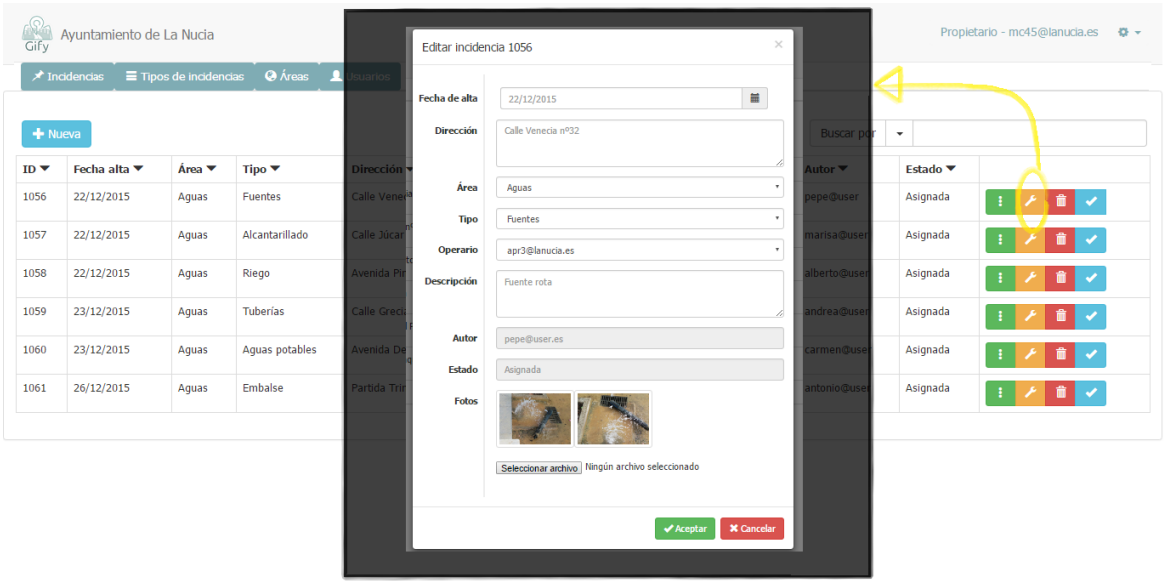


Figura 87: Interfaz Panel de gestión propietario / gestor – Incidencias – Editar. Elaboración propia.

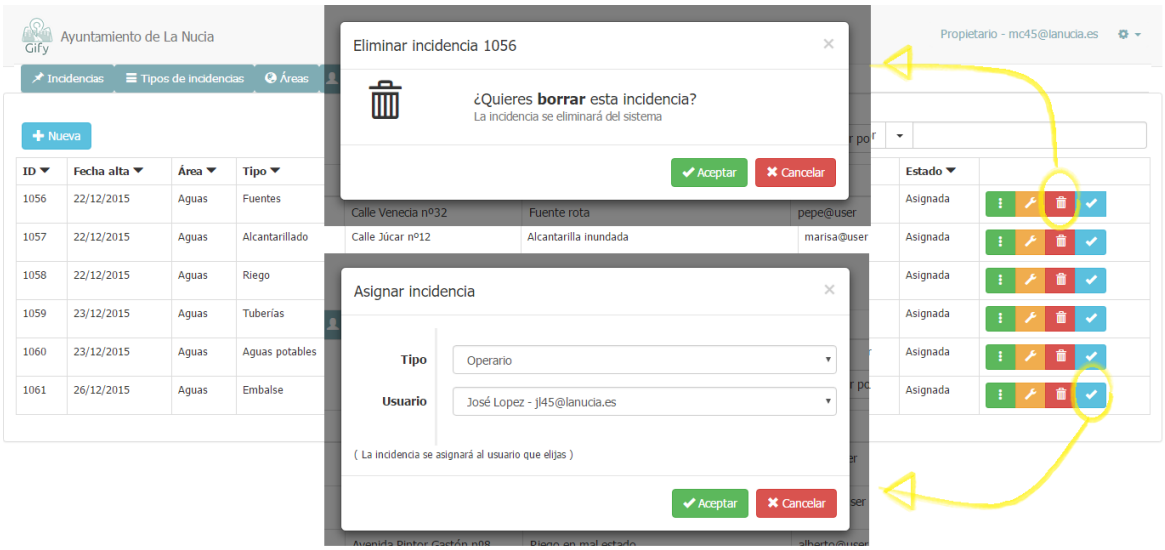


Figura 88: Interfaz Panel de gestión propietario / gestor – Incidencias – Eliminar y asignar. Elaboración propia.

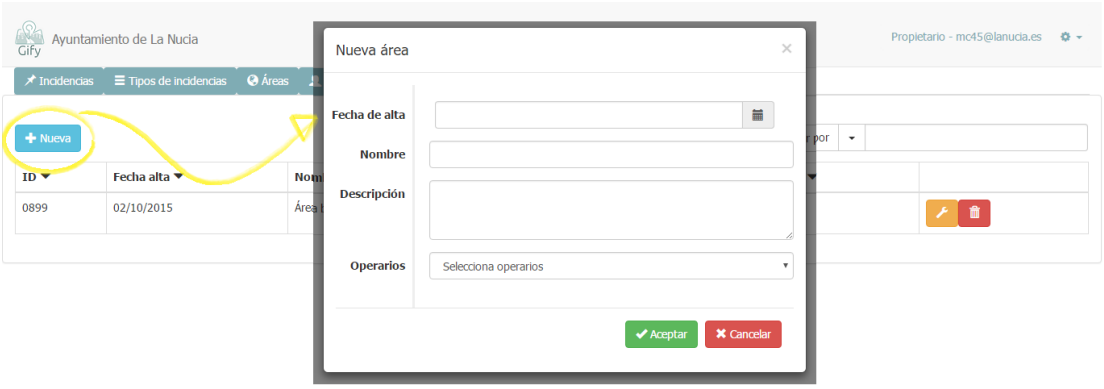


Figura 89: Interfaz Panel de gestión propietario / gestor – Áreas - Nueva. Elaboración propia.

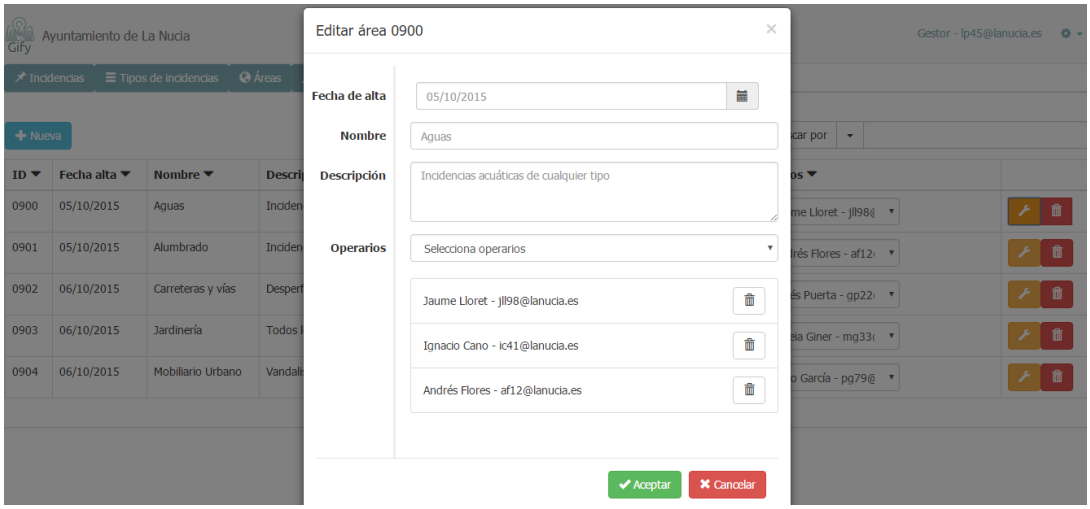


Figura 90: Interfaz Panel de gestión propietario / gestor – Áreas - Editar. Elaboración propia.

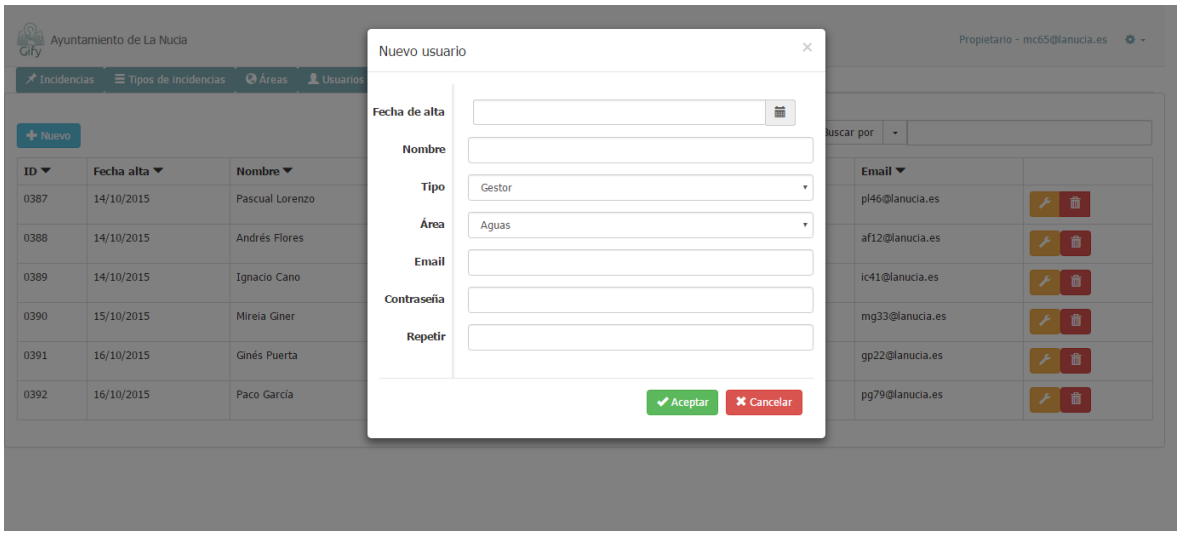


Figura 91: Interfaz Panel de gestión propietario / gestor – Usuarios – Nuevo. Elaboración propia.

Figura 92: Interfaz panel de gestión propietario / gestor - Usuarios - Editar. Elaboración propia.

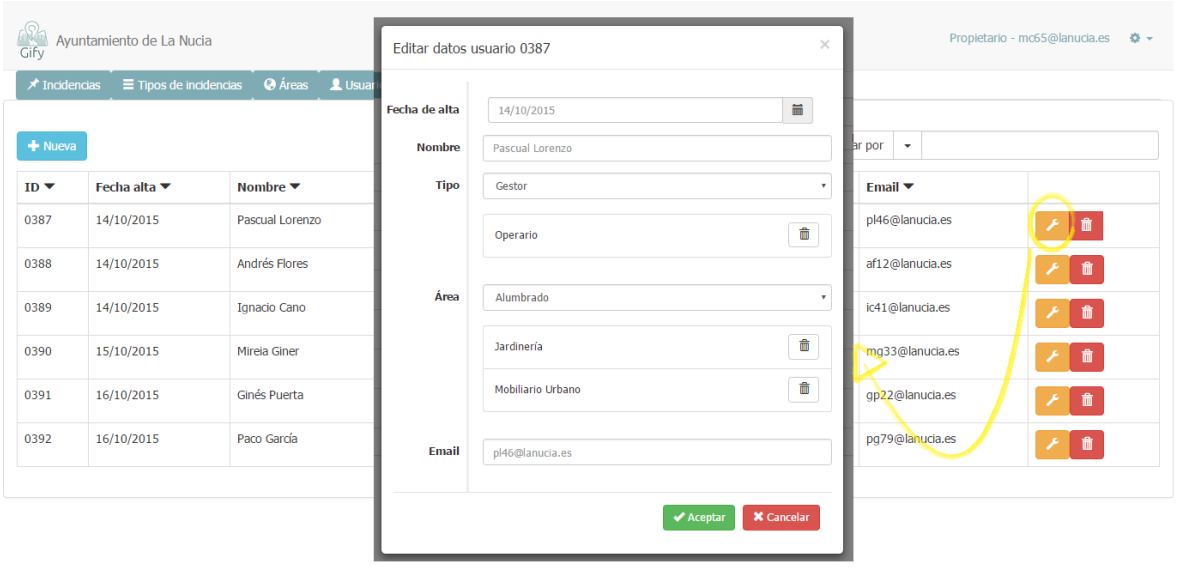


Figura 93: Interfaz Panel de gestión propietario / gestor – Usuarios – Editar. Elaboración propia.

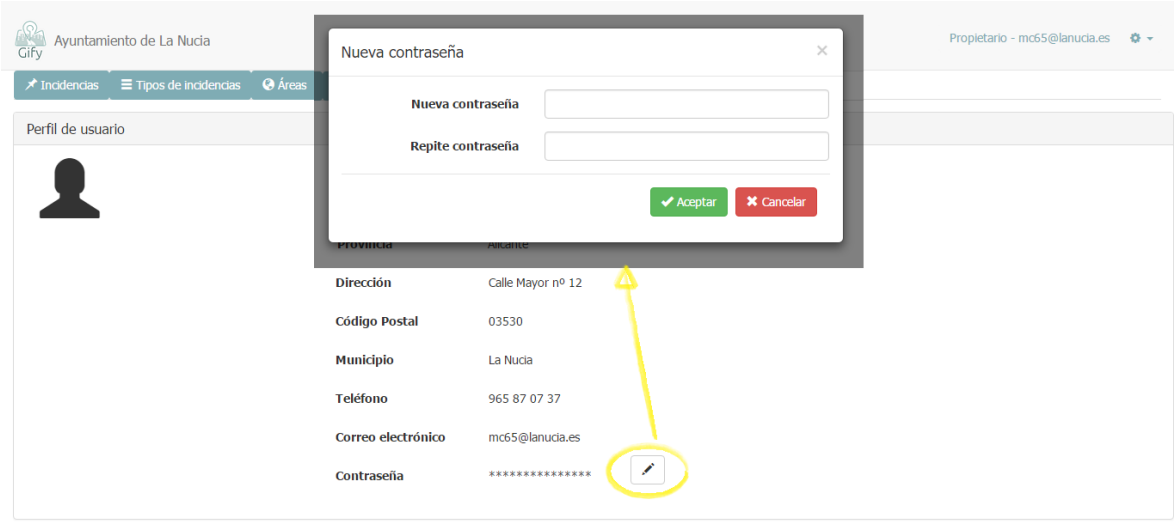


Figura 94: Interfaz Panel de Gestión propietario / gestor - Perfil - Cambio de contraseña. Elaboración propia.

Panel de gestión operario

Figura 95: Interfaz Panel de gestión operario – Incidencias – Rechazar incidencia. Elaboración propia.

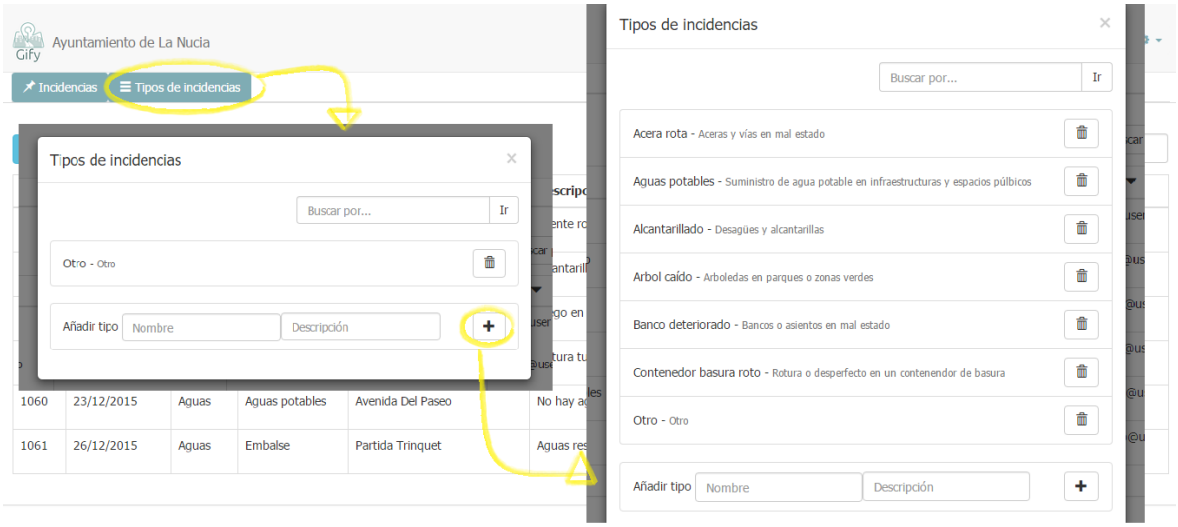


Figura 96: Interfaz Panel de gestión operario – Tipos de incidencias – Agregar. Elaboración propia.

Aplicación móvil

En la etapa de diseño se llevan a un plano tangible los conceptos y definiciones anteriores, primero en forma de *wireframes*, que permiten crear los primeros prototipos para ser probados con usuarios, y posteriormente, en un diseño visual acabado que será provisto al desarrollador, en forma de archivos separados y pantallas modelo, para la programación del código. Estas son las pantallas diseñadas para la aplicación móvil de Gify. Muestran el flujo que sigue el funcionamiento de la *app*.

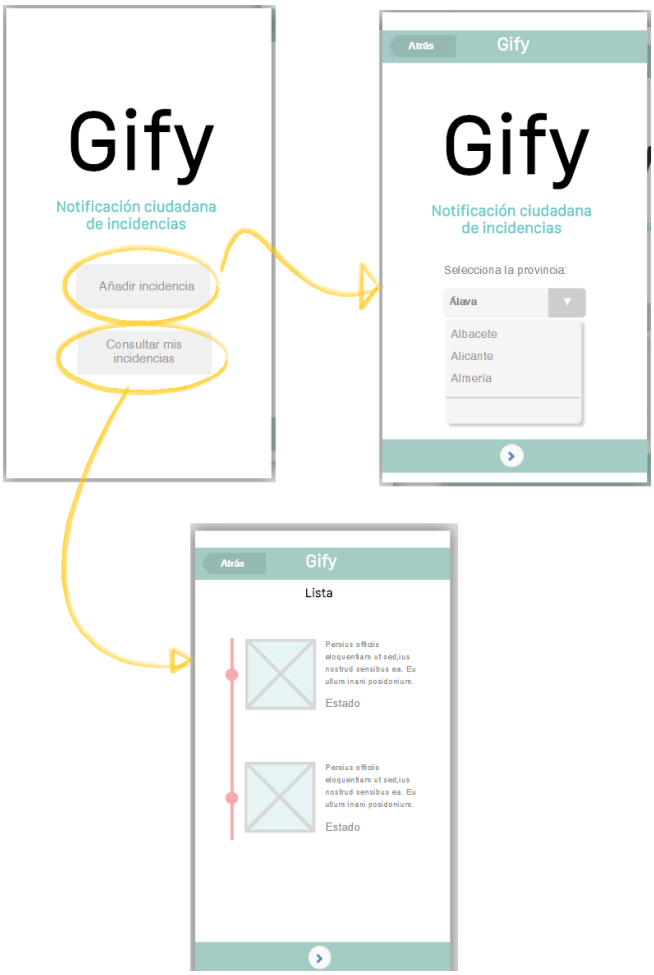


Figura 97: Interfaces app móvil – Inicio. Elaboración propia.

En la pantalla de inicio hay dos opciones: enviar una incidencia o consultar las que ya se han enviado desde esa cuenta.

Figura 98: Interfaces app móvil – Selección de entidad cliente. Elaboración propia.

Para añadir una incidencia, la aplicación te va guiando a través de una serie de pantallas para lograrlo. Los primeros pasos son escoger la provincia y el municipio para poder seleccionar la entidad donde queremos reportar la incidencia.

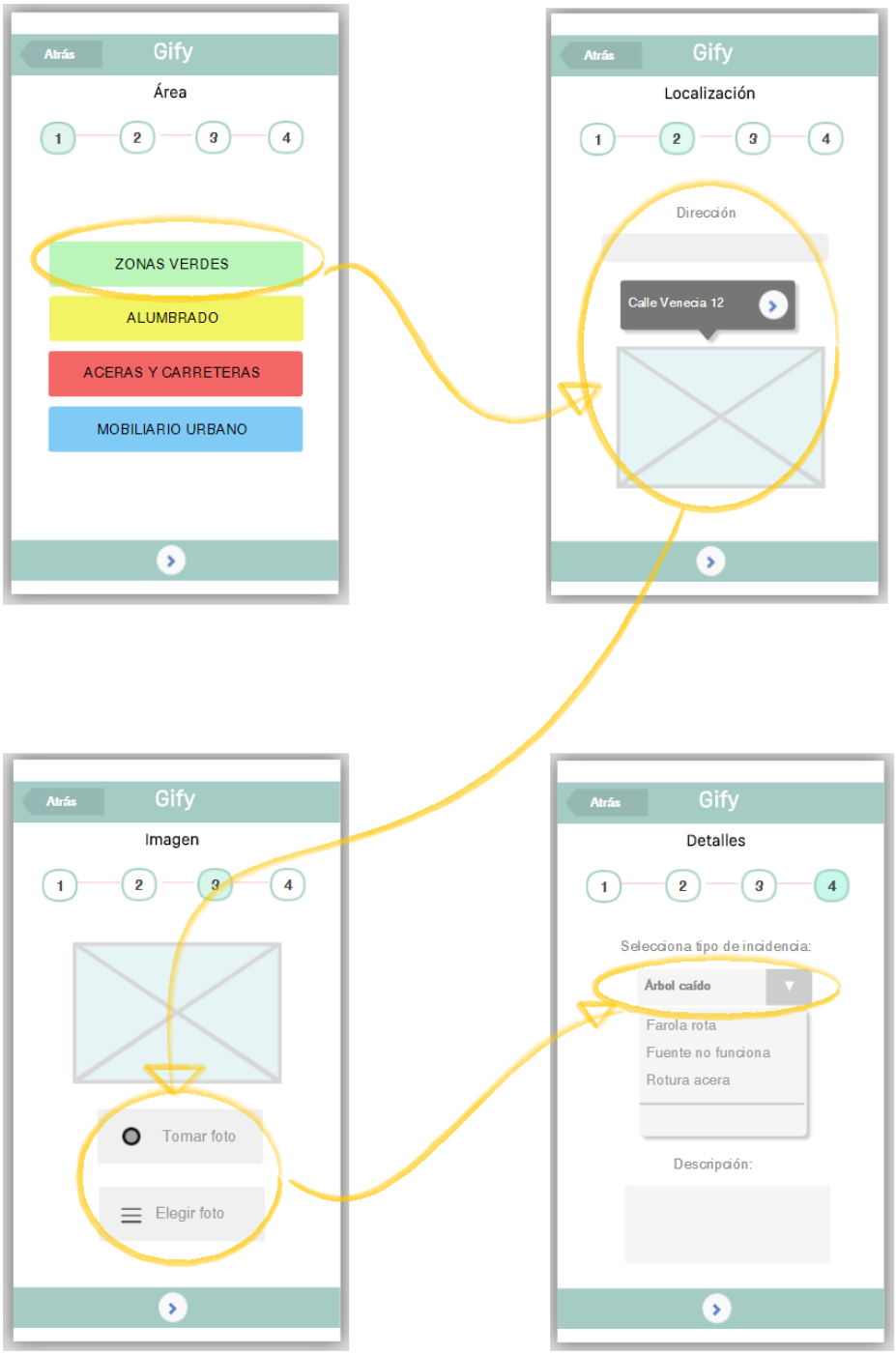


Figura 99: Interfaces app móvil – Completar datos de incidencia. Elaboración propia.

Hay cuatro pasos para definir una incidencia; seleccionar el área a la que se cree que pertenece el problema, indicar una dirección (lo más útil es adjuntar ubicación), tomar una foto de la avería o seleccionar una que ya se tenga en el dispositivo y, por último, escoger el tipo de incidencia que es y escribir una corta descripción si se considera necesario.

Figura 100: Interfaces app móvil – Formas de terminar la notificación. Elaboración propia.

Una vez hecho esto, antes de completar en el envío, hay que iniciar sesión en el sistema. Si el usuario no tuviese una cuenta, se da la posibilidad de crearla, o utilizar algún otro método de autenticación externo como el *Login* de Facebook.

Por último, ya *logueado,* el usuario puede enviar la incidencia y consultar su estado siempre que quiera en el apartado “Consultar mis incidencias” de la pantalla inicial.

Desarrollo

Configuración de herramientas de desarrollo

- Se trabajará con nuestro ordenador como servidor local de pruebas, con el entorno de desarrollo **XAMPP.**
- Se utiliza **PHP** como lenguaje del lado del servidor.
- **PHP Slim**, *framework* para la creación de la API.
- Base de datos **MySQL**.
- **AngularJS y Bootstrap** para el lado del cliente.
- **Underscore.js**, librería para realizar llamadas AJAX que delegan parte de la lógica de negocio en el lado del cliente.
- **Toastr.js** y **Toastr.css** para las notificaciones.
- Para la creación del proyecto se utiliza el editor **Adobe Edge Code CC.**
- Se utilizan **HTML + CSS + Javascript** y **Phonegap y Apache Cordova** para que nos permita compilar la aplicación móvil en todas las plataformas. **Eclipse IDE.**

Estructura de la aplicación

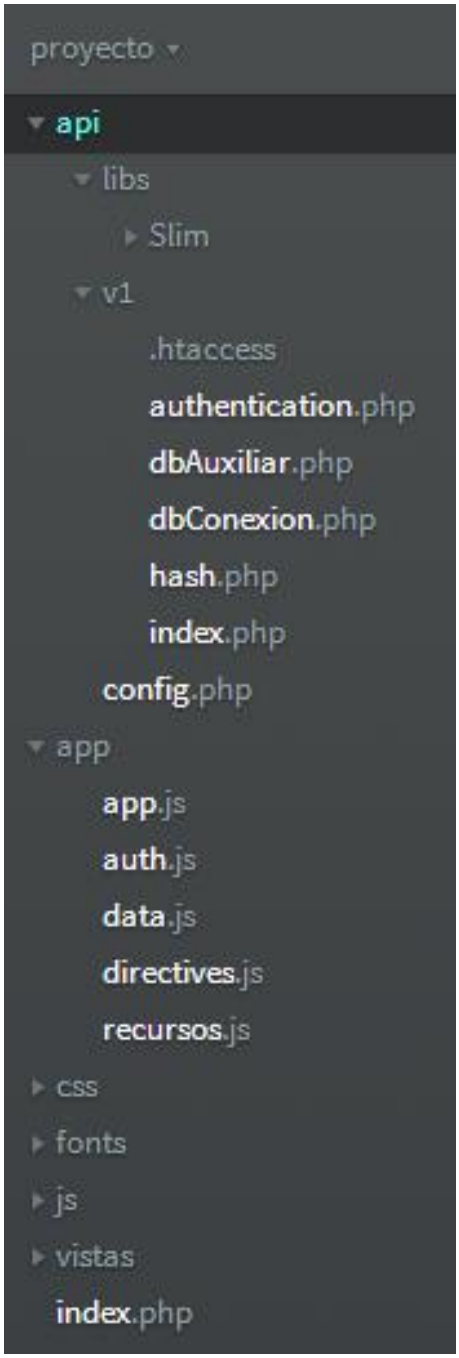


Figura 101: Estructura de la aplicación web. Elaboración propia.

- **Api** – Proveedor del servicio RESTful
  - Libs/Slim – Librería Slim
  - V1
    - ***.htacess*** – Convierte las urls
    - ***Hash.php*** – Genera contraseña cifrada
    - ***DbConexion.php*** – Conectar con la base de datos en MySQL
    - ***DbAuxiliar.php*** – Funciones auxiliares para realizar operaciones sobre la base de datos.
    - ***Index.php*** – Punto donde empieza la API
    - ***Authetication.php*** – Funciones para la autenticación de usuarios, Login, registro, sesión...
  - ***Config.php*** – Información sobre la bbdd
  - App
    - ***App.js*** – controla el enrutamiento y la autenticación
    - ***Auth.js***
    - ***Data.js*** – el middleware para conectar a la API
    - ***Directives.js***
    - ***Controlador.js*** – Los recursos se controlan desde aquí
  - Css
  - fonts
  - js
  - vistas – todas las vistas en .html
  - ***index.html*** – página de inicio

Métodos API REST

El sistema de este proyecto necesita que el servicio web permita utilizar la base de datos y realizar las operaciones sobre estos datos. De la API se nutrirán tanto la aplicación web como la web app móvil.

La estructura del servicio va a seguir un estilo sencillo MVC que maneje las peticiones de la parte del cliente. Para desarrollar este servicio vamos a seguir una serie de pasos:

- Diseñar e implementar de la base de datos
- Configurar las herramientas de desarrollo
- Crear los modelos de datos y las vistas
- Manejar las llamadas al servicio web RESTful (CRUD)
- Realizar pruebas

Las acciones que se desean realizar sobre los recursos están prácticamente definidas en este proyecto. La URL para llamar a las funciones en respuesta a los métodos HTTP específicos son:

Método	**URL**	**Descripción**
GET	/api/usuarios	Devuelve todos los usuarios
POST	/api/usuarios	Añade un nuevo usuario
GET	/api/usuarios/1	Devuelve el usuario con id = 1
PUT	/api/usuarios/1	Actualiza el usuario con id = 1
DELETE	/api/usuarios/1	Elimina el usuario con id = 1
POST	/api/areas	Añade una nueva area
GET	/api/areas	Devuelve la lista de areas
GET	/api/areas/1	Devuelve el area con id = 1
PUT	/api/areas/1	Actualiza el area con id = 1
DELETE	/api/areas/1	Elimina el area con id = 1
POST	/api/tipos	Añade un nuevo tipo de incidencia
GET	/api/tipos	Devuelve la lista de todos los tipos
GET	/api/tipos/1	Devuelve el tipo con id = 1

PUT	/api/tipos/1	Actualiza el tipo con id = 1
DELETE	/api/tipos/1	Borra el tipo con id = 1
POST	/api/incidencias	Añade una nueva incidencia
GET	/api/incidencias	Devuelve la lista de incidencias
GET	/api/incidencias/1	Devuelve la incidencia con id = 1
PUT	/api/incidencias/1	Actualiza la incidencia con id = 1
DELETE	/api/incidencias/1	Elimina la incidencia con id = 1
GET	/api/incidencias?area=jardineria	Devuelve incidencias con area = jardineria
GET	/api/usuarios?email=paquit@gmail.com	Devuelve usuarios con un email específico
GET	/api/areas/search/usuario	Devuelve areas a quien pertenezca un usuario específico
GET	/api/incidencias/1/notas	Mostrar todas las notas de esa incidencia
POST	/api/incidencias/1/notas	Se crea una nueva nota para la incidencia
DELETE	/api/incidencias/1/notas/n1	Se elimina la nota con id=1 de la incidencia con id = n1
POST	api/incidencias/1/responsable	Se asigna un responsable a la incidencia
PUT	api/incidencia/1/estado/3	Actualiza el estado de la incidencia
…	…	…

Como podemos ver, las URLS no deben implicar acciones y deben ser únicas, deben mantener una jerarquía lógica y en el caso del filtrado, la búsqueda o la paginación de recursos, ya que hacer una consulta sobre la propia URL, utilizando parámetros HTTP en lugar de incluirlos en la misma.

Con la ayuda de PHP recogeremos todas las operaciones de bases de datos que se repiten en la mayoría de clases de nuestro sistema. De esta manera, podremos utilizarlas fácilmente cada vez que tengamos que seleccionar, insertar, actualizar o eliminar un registro de la base de datos.

Principalmente, en la base de datos SQL se utilizan las siguientes 4 operaciones para gestionar nuestros datos:

Read (Leer)

```
SELECT * FROM `incidencias`;
```

Los datos seleccionados que nos va a devolver esta operación podemos recibirlos con unas funciones auxiliares que se definen en la clase ***dbAuxiliar.php.***

```
function select($table, $where) {
        try{
            $a = array ();
            $w = "";
            for each ($where as $key => $value) {
                $w .= " and " .$key. " like :".$key;
                $a[":".$key] = $value;
            }
            $stmt = $this->db->prepare("select * from ".$table." where 1=1 ".
$w);
            $stmt->execute($a);
            $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
            if(count($rows)<=0){
                $response["status"] = "warning";
                $response["message"] = "No data found.";
            }else{
                $response["status"] = "success";
                $response["message"] = "Data selected from database";
            }
                $response["data"] = $rows;
        }catch(PDOException $e){
            $response["status"] = "error";
            $response["message"] = 'Select Failed: ' .$e->getMessage();
            $response["data"] = null;
        }
        return $response;
    }
```

Ocurre lo mismo con las demás operaciones CRUD.

Create (Insertar)

```
INSERT INTO `incidencias` (`idIncidencia`, `idEntidad`, `idTipo`,
`idArea`, `idEstado`, `fecha`, `direccion`, `autor`) VALUES (138, 5, 2, 1,
1, CURDATE(), 'calle limones nº4', pepe@user);
```

Update (Actualizar)

```
UPDATE `incidencias` SET `idEstado`= 2 WHERE `idIncidencia`= 0324;
```

Delete (Borrar)

```
DELETE FROM `incidencias` WHERE `idIncidencia`= 0324;
```

Implementación de la API con Slim

Con las librerías de Slim dentro de nuestro proyecto es muy sencillo implementar las llamadas a la API y las funciones.

```
<?php

require 'api/libs/Slim/Slim.php';

$app = new Slim();
$app = \Slim\Slim::getInstance();
$db = new dbAuxiliar();

$app->get('/incidencias', function (){
Global $db;
$rows = $db->select("incidencias","id,idEntidad,idTipo,idArea,idEstado,fecha,direccion,autor",array());
EchoResponse(200,$rows);

…

$app->run();

?>
```

Aplicación web

Para la construcción de la aplicación hay que poner el siguiente div para todas las vistas que se irán sirviendo en función del usuario logueado y las acciones que puede realizar, además de las transiciones propias del funcionamiento de la aplicación.

```
<div ng-view="" id="ng-view"></div>
```

Por ejemplo, si se está visualizando el panel de las incidencias, la tabla tendrá una cabecera parecida a esta. Si en cambio es la tabla de los usuarios la que está en pantalla, se mostrarán los datos de los usuarios, de esta forma:

```
<td>{{c.id}}</td><td>{{c.fecha}}</td><td>{{c.direccion}}</td><td>{{c.area
}}</td><td>{{c.packing}}</td><td>{{c.description}}</td>
```

Las vistas se controlan por un manejador que establece la lógica que debe activarse para añadir, borrar o actualizar los recursos conforme el usuario interactúe con ellos (***controlador.js).***

Autenticación de usuario

En ***app.js*** se configuran las rutas de las vistas:

```
var app = angular.module('myApp', ['ngRoute', 'ngAnimate', 'toaster']);

app.config(['$routeProvider',
  function ($routeProvider) {
        $routeProvider.
                  when('/login', {
                        title: 'Login',
                        templateUrl: 'partials/login.html',
                        controller: 'authCtrl'
                  })

                  .
                  .
                  .
```

Y en ***data.js*** se produce la comunicación con la API RESTful:

```
app.factory("Data", ['$http', 'toaster',
    function ($http, toaster) { // This service connects to our REST API

        var serviceBase = 'api/v1/';

        var obj = {};
        obj.toast = function (data) {
            toaster.pop(data.status,     "",     data.message,     10000,
'trustedHtml');
        }
        obj.get = function (q) {
            return $http.get(serviceBase + q.then(function (results) {
                return results.data;
            });
        };
```

Auth.js es el controlador que se comunica con el *front-end* y maneja las vistas.

```
app.controller('authCtrl', function ($scope, $rootScope, $routeParams,
$location, $http, Data) {
    $scope.login = {};
    $scope.signup = {};
    $scope.doLogin = function (usuario) {
        Data.post('login', {
            customer: customer
        }).then(function (results) {
            Data.toast(results);
            if (results.status == "success") {
                $location.path('dashboard');
            }
        });
    };
    $scope.signup = {email:'',password:'',name:'',phone:'',address:''};
    $scope.signUp = function (usuario) {
        Data.post('signUp', {
            customer: customer
        }).then(function (results) {
            Data.toast(results);
            if (results.status == "success") {
                $location.path('dashboard');
            }
        });
    };
    $scope.logout = function () {
        Data.get('logout').then(function (results) {
            Data.toast(results);
            $location.path('login');
        });
```

Aplicación móvil

Para conectar mi aplicación en Phonegap a la base de datos en *localhost*, se hace mediante la API y nos devuelve los datos en JSON.

Una vez incluida la librería de Javascript de Phonegap, sus APIs están lista para ser usadas y podemos empezar a trabajar.

En nuestro archivo javascript se agrega un escuchador de eventos al elemento *document*, el cual manejara las peticiones a la API.

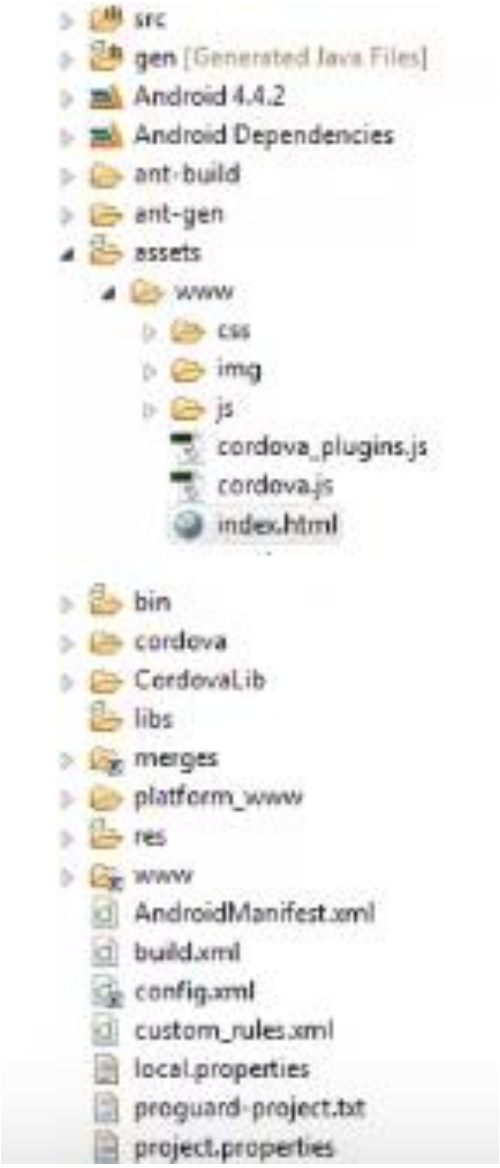


Figura 102: Estructura app móvil. Elaboración propia.

Planificación de tareas

La planificación del tiempo que va a durar cada una de las tareas en las que se divide un proyecto ayuda a establecer metas y prioridades en el desarrollo. Es una de las partes más importantes, ya que una mala planificación del tiempo puede generar muchos problemas.

A continuación, encontramos una imagen de la división de tareas que se hizo para este proyecto y las horas que se ha tardado en realizar cada una:

TAREA	DURACIÓN	COMIENZO	FIN
Plan de seguimiento	**23 horas**	25/09/2015	23/09/2016
Reuniones con profesor	14 horas	25/09/2015	09/09/2016
Transcipción de tutorias (audio - texto)	6 horas	25/09/2015	05/10/2015
Definición de objetivos	3 horas	05/10/2015	10/10/2015
Análisis y diseño	**102 horas**	10/10/2015	20/03/2015
Realización marco teórico	30 horas	10/10/2015	30/10/2015
Análisis de los requisitos funcionales	12 horas	30/10/2015	10/11/2015
Casos de uso	6 horas	10/11/2015	20/11/2015
Diseño modelo relacional	12 horas	20/11/2015	25/11/2015
Diseño de interfaces	12 horas	10/01/2016	20/01/2016
Redacción de esta parte de la memoria	30 horas	20/01/2016	20/03/2016
Implementación	**222 horas**	20/03/2016	20/08/2016
Creación base de datos	6 horas	20/03/2016	22/03/2016
Creación de API	6 horas	22/03/2016	30/03/2016
Integración y pruebas	12 horas	30/03/2016	05/04/2016
Desarrollo *Landing-page*	**24 horas**	05/04/2016	25/04/2016
Implementar interfaces	12 horas	05/04/2016	15/04/2016
Implementación funcionalidades	12 horas	15/04/2016	25/04/2016
Desarrollo aplicación web	**96 horas**	25/04/2016	01/07/2016
Creación de interfaces	36 horas	25/04/2016	11/05/2016
Implementación	26 horas	25/04/2016	03/05/2016
Diseño responsive	10 horas	03/05/2016	11/05/2016
Implementación funcionalidades	60 horas	11/05/2016	01/07/2016
Crear funciones	35 horas	11/05/2016	01/06/2016
Integración con *front-end*	25 horas	01/06/2016	01/07/2016
Desarrollo aplicación movil	**60 horas**	01/07/2016	18/08/2016
Integración phonegap	6 horas	01/07/2016	07/07/2016
Implementación interfaces	10 horas	07/07/2016	20/07/2016
Implementación funcionalidades específicas	44 horas	20/07/2016	18/08/2016
Integración de herramientas y entornos	**12 horas**	20/03/2016	18/08/2016
Pruebas	**10 horas**	18/08/2016	20/08/2016
Cierre	**36 horas**	20/08/2016	23/09/2016
Terminar la memoria	18 horas	01/09/2016	09/09/2016
Preparar presentación	18 horas	09/09/2016	23/09/2016
TOTAL	**393 horas**		

Figura 103: Tabla de planificación de tareas. Elaboración propia.

Modelo de negocio

El *Business Model Canvas* intenta hacer reflexionar sobre el planteamiento estratégico de cada una de las áreas claves de nuestro modelo de negocio.

En el caso de este proyecto, si se quisiera sacar rentabilidad de él y plantearlo como un negocio real, el diseño del modelo del mismo quedaría representado en un *canvas* de la siguiente manera:

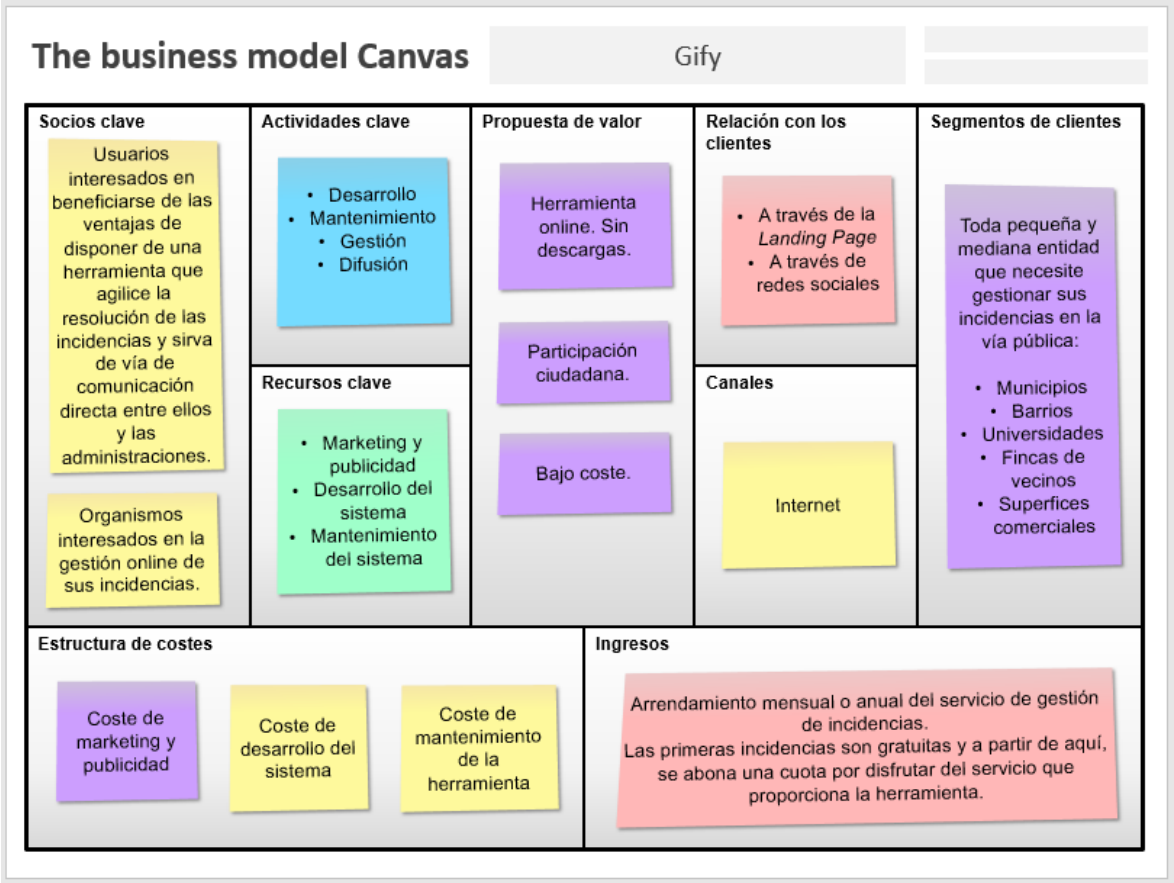


Figura 104: Business Model Canvas de Gify. Elaboración propia.

Respecto a la estrategia de marketing, se centraría básicamente en la promoción a través de internet. No solo creando una *landing-page*, si no, a través de la participación activa en redes sociales, crear video marketing, promocionar la herramienta a través de anuncios y generar estrategias SEO y SEM.

La monetización del sistema se basaría en lanzar al mercado un producto de uso gratuito hasta alcanzar un número de incidencias registradas y resueltas. A partir de ese punto, se ofrecería seguir con el uso de la herramienta a cambio de que las entidades clientes abonen una cuota mensual o anual. Es un comportamiento parecido al alquiler y que utilizan muchos sistemas.

Que un producto sea gratuito favorece que tenga una rápida acogida entre los usuarios y que se pueda hacer rápidamente conocido. Una vez logrados estos objetivos, puede pasar a monetizarse su uso.

Existe el riego de que una vez que se cobre por el servicio un porcentaje de clientes dejen de utilizarlo para no pagar. Hay que intentar que el precio que se establezca sea económico y acorde al perfil económico de los clientes. Si esta herramienta va enfocada a pequeñas y medianas organizaciones, es de suponer que su presupuesto de inversión en herramientas será apretado.

Independientemente de estas conjeturas, hay que confiar en el producto que se ha desarrollado y arriesgar. Si el servicio que se presta, en este caso, es lo suficientemente útil y se proporciona en buenas condiciones, los clientes van a seguir interesados en hacer uso de la herramienta de gestión e invertirá en ella. Además, hay otro aspecto a tener en cuenta y es que cuando un cliente introduce en su día a día una plataforma, además de que se acostumbra a su uso, se genera una cantidad de datos e información lo suficientemente relevante como para abandonar el sistema.

Trabajo futuro

En este proyecto se definieron una serie de objetivos que dieron lugar a una especificación de requisitos, un diseño de *mockups*, un diseño de base de datos y un diseño de *back-end*, con la finalidad de desarrollar un sistema completo de gestión de incidencias, formado por una aplicación web, una aplicación móvil y un panel de administración de la plataforma.

Puede observarse que la importancia del panel de administración va diluyéndose a través de los apartados de diseño. Esto es así porque el curso iba transcurriendo y, en un momento determinado, por cuestiones de tiempo, se tomó la decisión de dejar esta parte como trabajo futuro y centrar la implementación en el sistema de gestión organización – usuario. De hecho, en los requisitos funcionales se definió una sola funcionalidad para en panel de administración de la plataforma, cuyo único requisito era deseable.

- **RF43: Gestionar clientes y usuarios ciudadanos**

  La herramienta deberá tener un panel de gestión de la propia plataforma donde llevar los asuntos relacionados con los clientes del servicio y los usuarios ciudadanos.

En cuanto al sistema desarrollado, se definieron los requisitos funcionales y se clasificaron en base a su prioridad, lo que ha permitido establecer un orden en la implementación de los mismos y llegar a realizar todos los requisitos obligatorios.

Los requisitos funcionales deseables se consideran mejoras que harían más atractiva la herramienta y se tendrían en cuenta como funcionalidades en una fase futura del proyecto. Estos requisitos a desarrollar son:

Aplicación Web

- **RF30: Subir foto**

  El sistema permitirá que los usuarios adjunten fotos a las incidencias.

- **RF31: Eliminar foto**

  El sistema permitirá que los usuarios eliminen las fotos que han adjuntado.

- **RF32: Recuperar contraseña**

  El sistema permitirá que un usuario recupere el acceso a su panel de gestión reestableciendo su contraseña.

- **RF33: Editar datos de perfil**

  El sistema posibilitará editar ciertos datos en el perfil del panel de gestión de cada usuario.

Aplicación móvil

- **RF40: Seleccionar área y tipo**

  La herramienta permitirá seleccionar el área y el tipo de la incidencia que va a enviarse, en función de las áreas y las tipologías que tenga la entidad cliente definidas.

- **RF41: Adjuntar ubicación**

  El sistema permitirá *geolocalizar* la incidencia que el usuario está accediendo a las características hardware del dispositivo móvil.

- **RF42: Adjuntar fotografía**

  El sistema permitirá tomar una fotografía de la incidencia o seleccionarla de la galería del móvil para adjuntarla en el reporte de la incidencia.

Por último, también podría considerarse la posibilidad de incluir como trabajo futuro un plan de marketing, publicidad e ingresos, fruto del lanzamiento de la herramienta.

Conclusiones

Con la realización de este trabajo de final de grado he podido llegar a una serie de conclusiones que han cambiado, de alguna manera, mi perspectiva sobre ciertas cosas, tanto en temas de desarrollo de software como personales.

El objetivo principal de este proyecto era crear un producto de principio a fin, pasando por todas las fases del proceso de desarrollo y documentando los resultados que se van obteniendo en cada una de ellas. El sistema de gestión de incidencias online que escogí desarrollar, al final del camino, me sigue pareciendo atractivo y motivante, igual el día que empecé con él. Son varios los motivos por los que pienso esto:

- Si bien es cierto que ya existen herramientas parecidas a Gify, personalmente no he podido encontrar ninguna que sea gratuita. Todas, por lo que se ve desde sus *Landing pages*, parecen ser herramientas que puedes utilizar después de ponerte en contacto con la empresa, adquirir el producto y demás. Ninguna da la posibilidad de registrar tu organización y tener acceso con esos datos a un panel de gestión de las incidencias. En mi opinión es un valor para esta herramienta.
- Me parece que el objetivo de que los usuarios puedan comunicar directamente incidencias a través de una *app* a los ayuntamientos, universidades o negocios, cubre una necesidad real en el ámbito de los partes de incidencias y, además, es una buena forma de que se produzca la fidelización de los ciudadanos.
- A la construcción del tipo de *back-end* del proyecto también le veo ventajas. He aprendido un poco más sobre desarrollo web y móvil gracias a haber creado una API para un servicio orientado a Internet.

La realización del trabajo final de grado me ha ayudado a entender que a la hora de desarrollar un proyecto lo más importante es identificar muy bien los objetivos que te has propuesto conseguir y dividirlos en tareas a las que se debe asignar un tiempo para realizarlas.

Durante la carrera, los conceptos de planificación del tiempo y las tareas se ven de pasada en algunas asignaturas, pero no se practica lo suficiente como para adquirir una soltura que te permita llevar a cabo sin dificultades todas las fases de desarrollo de un proyecto. Hay una gran cantidad de cosas que hay que pararse a pensar, escoger y diseñar cuando vas a construir un sistema y quieres que salga lo mejor posible, porque de ello depende tu nota o tu puesto de trabajo. Cada estudiante, o desarrollador, debería interiorizar estos conceptos y ponerlos en práctica en cada trabajo que realice.

Otra de las conclusiones que puedo sacar de esta experiencia es que el TFG ha sido el trabajo que más me ha costado llevar a cabo en la universidad. Estar solo frente a algo cuyo resultado repercutirá en tus notas, cuando la mayoría de trabajos realizados durante la carrera han sido en grupo, es como mínimo intimidante.

La índole que adquiere un proyecto final depende de la exigencia de la persona que lo está realizando, sin embargo, en mi opinión, cada una de las personas que llevan a cabo un reto así deben ser constantes en el trabajo. Seguramente sea algo obvio, pero yo lo he aprendido en las veces que he ido apurada de tiempo para cumplir algún plazo con el tutor. Las demás asignaturas, trabajo o circunstancias personales pueden interferir en el desarrollo del proyecto de una manera que no hubiese imaginado.

De cualquier forma, todas estas conclusiones, más internas y personales que otra cosa, se refieren a algunas partes de lo que supone desarrollar un proyecto. En general, es una experiencia muy enriquecedora profesionalmente y que te colma de lecciones y buenas prácticas para el futuro.

Bibliografía y referencias

Referencias a noticias

[1] Caquías, S. (3 de diciembre de 2015). A secas Ponce hasta el sábado. *El Nuevo Día.* http://www.elnuevodia.com/noticias/locales/nota/asecasponcehastaelsabado-2134436/

[2] Fraile, O. (9 de junio de 2015). Buscan a unos ladrones dedicados al robo de capiteles en iglesias románicas de Soria. *20 minutos.* http://www.20minutos.es/noticia/2484601/0/soria/banda-de-ladrones/robo-capiteles-romanicos/

[3] (6 de diciembre de 2015). Faltan 641 motos del corralón municipal y nadie se dio cuenta. *Misiones Cuatro.* http://misionescuatro.com/policiales/iguazu-faltan-641-motos-del-corralon-municipal-nadie-se-dio-cuenta/

[4] Vallejo, E. (12 de julio de 2015). Ametzagaña, de zona silvestre a parque urbano. *El Diario Vasco.* http://www.diariovasco.com/san-sebastian/201507/12/ametzagana-zona-silvestre-parque-20150712001425-v.html

[5] Plaza, C. (26 de octubre de 2015). El árbol más denunciado. *La Razón.* http://www.larazon.es/local/madrid/el-arbol-mas-denunciado-OF11050068#.Ttt1ZWywZmWKLqL

[6] Costa, M. (26 de noviembre de 2015). Técnicos de conselleria inspeccionan los pisos sociales degradados de Tavernes. *Las Provincias.* http://www.lasprovincias.es/horta-morvedre/201511/27/tecnicos-conselleria-inspeccionan-pisos-20151126233129-v.html

[7] Verín, S. M. (17 de noviembre de 2015). Quejas vecinales por el mal estado del pavimento en un pueblo de Vilar de Barrio. *La Voz de Galicia.* http://www.lavozdegalicia.es/noticia/ourense/vilar-de-barrio/2015/11/17/quejas-vecinales-mal-estado-pavimento-pueblo-vilar-barrio/0003_201511O17C9993.htm

[8] (6 de marzo de 2013). La Universidad tapará las goteras de las bibliotecas. *La Gaceta de Salamanca.* http://www.lagacetadesalamanca.es/salamanca/2013/03/05/universidad-tapara-goteras-bibliotecas/87692.html

[9] (9 de noviembre de 2015). Suspenden cirugías por filtraciones en el techo. *El Día.* http://www.eldia.com/la-ciudad/por-filtraciones-cloacales-suspendieron-cirugias-en-el-hospital-de-ninos-95347

[10] EFE (29 de septiembre de 2011). Cierran un supermercado tras varias denuncias vecinales. *Diario de Avisos.* http://www.diariodeavisos.com/2011/09/cierran-un-supermercado-tras-varias-denuncias-vecinales/

[11] (31 de mayo de 2014). 5 horas atrapada en el baño hasta que la rescató la Policía. *Portal de Noticias Villa María Vivo.* http://villamariavivo.com/5-horas-atrapada-en-el-bano-hasta-que-la-rescato-la-policia/

[12] (6 de abril de 2015). Ya son tres los casos de robos a vehículos en parqueadero de centro comercial de Hayuelos. *Pulzo.* http://www.pulzo.com/bogota/ya-son-tres-los-casos-de-robos-vehiculos-en-parqueadero-de-centro-comercial-hayuelos/319491

[13] Bessy, L. (21 de diciembre de 2015). Cafetaleros registran pérdidas millonarias por mal estado de calles. *La Prensa.* http://www.laprensa.hn/economia/913059-410/cafetaleros-registran-p%C3%A9rdidas-millonarias-por-mal-estado-de-calles

[14] (26 de noviembre del 2015). La Molina: incendio se originó en sede de universidad. *El Comercio.* http://elcomercio.pe/whatsapp/sucesos/whatsapp-molina-incendio-se-origino-sede-universidad-video-noticia-1859423

[15] Rodríguez, R. (29 de julio de 2015). Policías de paisano vigilarán los parques de Plasencia para atajar el vandalismo. *El Periódico Extremadura.* http://www.elperiodicoextremadura.com/noticias/plasencia/policias-paisano-vigilaran-parques-plasencia-atajar-vandalismo_883884.html

[16] Carnicero, L. (6 de julio de 2015). Zaragoza Ciudadana presenta más de 70 denuncias vecinales. *El Periódico de Aragón.* http://www.elperiodicodearagon.com/noticias/aragon/zaragoza-ciudadana-presenta-mas-70-denuncias-vecinales_1038236.html

[17] Dasí, A. (11 de noviembre de 2015). Torrent resuelve más de 1.500 incidencias denunciadas por los vecinos en la vía pública. *Las Pronvincias.* http://www.lasprovincias.es/horta-morvedre/201511/12/torrent-resuelve-incidencias-denunciadas-20151111235130-v.html

[18] R. A. (20 de agosto de 2015). Comercio resuelve el 85% de incidencias recogidas en el buzón. *La Verdad. Alicante.* http://www.laverdad.es/alicante/ciudad-alicante/201508/20/comercio-resuelve-incidencias-recogidas-20150820013530-v.html

[19] Pardo, M. (4 de octubre de 2015). Redes sociales, el escaparate de las quejas. *Diario Información.* http://www.diarioinformacion.com/elda/2015/10/04/redes-sociales-escaparate-quejas/1681622.html

Referencias a herramientas

[20] Línea Verde. Smart City. Información en http://www.lineaverdemunicipal.com/comunicacion_incidencias_y_desperfectos.aspx

[21] Portal ciudadano. Sistema de gestión de incidencias en vía pública. Ajuntament de Torrent. Información en http://www.torrent.es/givp/cs

[22] Sistema GECOR de Incidencias en la Vía Pública. Ayuntamiento de Málaga. Información en http://movilidad.malaga.eu/portal/menu/seccion_0002/secciones/subSeccion_0003

[23] Aplicación del Ajuntament de Vinaròs Incidencias. Información en https://vinaros.sigoincidencias.com/app/es/

Referencias de interés

[24] Ranking de popularidad de los gestores de bases de datos. (abril de 2016). Fuente: http://db-engines.com/en/ranking

[25] Los lenguajes de programación más populares en 2015. Fuente: http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages

Referencias bibliográficas

- "¿Qué es SaaS o Software como servicio?" [Blog]
http://aprenderinternet.about.com/od/Glosario/g/Software-como-servicio.htm
- "Programación de aplicaciones web: Historia, principios básicos y clientes web." [Libro]
https://rua.ua.es/dspace/bitstream/10045/16995/1/sergio_lujan-programacion_de_aplicaciones_web.pdf
- "Enciclopedia – Bases de datos" [Blog]
http://es.ccm.net/contents/66-introduccion-bases-de-datos
- "Cómo seleccionar una plataforma de desarrollo para un proyecto web" [Blog]
http://lapastillaroja.net/2013/10/como-seleccionar-plataforma-tecnologica/
- "Javascript del lado del Servidor" [Blog]
http://blog.educacionit.com/2011/05/30/javascript-del-lado-del-servidor/
- "Lenguajes del lado del servidor o cliente" [Blog]
http://www.desarrolloweb.com/articulos/239.php
- "Lenguajes programación del lado del cliente" [Artículo]
http://es.slideshare.net/JeremiasMorales/22-lenguajes-del-lado-cliente
- "¿Qué es una API?" [Blog]
https://hipertextual.com/archivo/2014/05/que-es-api/
- "Conceptos sobre APIs REST" [Blog]
asiermarques.com/2013/conceptos-sobre-apis-rest/
- "El crecimiento imparable de las APIs, fundamentales en desarrollo de webs y aplicaciones móviles actualmente" [Blog]
http://www.genbetadev.com/desarrollo-web/el-crecimiento-imparable-de-las-apis-fundamentales-en-el-desarrollo-de-webs-y-aplicaciones-moviles-actualmente
- "¿Por qué las APIs no deberían usar el modelo de autenticación login/password?" [Blog]
http://www.genbetadev.com/seguridad-informatica/por-que-las-apis-no-deberian-usar-el-modelo-de-autenticacion-login-password
- "Herramientas: El lienzo de modelos de negocio" [Blog]
http://javiermegias.com/blog/2011/11/herramientas-el-lienzo-de-modelos-de-negocio-business-model-canvas/
- "Tipos de estrategias de marketing" [Blog]
http://www.smartupmarketing.com/tips-de-estrategias-de-marketing/