Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Aplicación de información en tiempo real y predicción para mejorar la eficiencia en Smart Cities

Autora: Itziar Rodríguez Hernández

Tutor: Cándido Caballero Gil

5 de julio de 2020

La publicación de este Trabajo Fin de Máster solo implica que el estudiante ha obtenido al menos la nota mínima exigida para superar la asignatura correspondiente, no presupone que su contenido sea correcto, aunque si aplicable. En este sentido, la ULL no posee ningún tipo de responsabilidad hacia terceros por la aplicación total o parcial de los resultados obtenidos en este trabajo. También pone en conocimiento del lector que, según la ley de protección intelectual, los resultados son propiedad intelectual del alumno, siempre y cuando se haya procedido a los registros de propiedad intelectual o solicitud de patentes correspondientes con fecha anterior a su publicación.

Resumen

Si te preguntaran cómo crees que serán las ciudades de aquí a diez años, ¿cuál sería tu respuesta?

El mundo avanza a una velocidad asombrosa debido en gran parte al desarrollo tecnológica de las últimas décadas. Si se echara la mirada unos veinte años atrás se podrían encontrar millones de diferencias a cómo es nuestro planeta en la actualidad. Y en este proceso de evolución las ciudades no han sido meras espectadoras.

Con la implementación del Big Data, la cobertura 5G, el internet de las cosas (IOT), la inteligencia artificial y el resto de las tecnologías emergentes, cada vez, son más las ciudades que se transforman para ser más eficientes y sostenibles.

En el presente proyecto se hablará de esta nueva tipología de ciudades conocidas como Smart Cities y se llevará a cabo la implementación de una aplicación en Android para teléfonos móviles que tiene como objetivo hacer de las zonas urbanas un lugar más innovador, más sostenible y participativo. En definitiva, un lugar dónde las personas y la tecnología se unan para mejorar el mundo y avanzar hacia el futuro.

Abstract

What would be your answer if someone ask you on how you figure cities in 10 years?

The world is progressing at an astonishing speed due mainly to the technological development of recent decades. If you looked back over twenty years, there could be millions of differences to what our planet is like nowadays. And in this process of evolution, cities have not been mere spectators.

The implementation of Big Data, 5G coverage, Internet of Things (IOT), artificial intelligence and the rest of emerging technologies, more and more cities are transforming to be more efficient and sustainable.

In this project it would be discussed this new typology of cities known as Smart Cities and will carry out the implementation of an application on Android for mobile phones that aims to make urban areas a more innovating, sustainable and collaborative place. In short, a place where people and technology come together to improve the world and move into the future.

Índice General

Capítulo 1. Introducción

Índice Introducción

1. Perspectiva general

La contaminación es uno de los grandes problemas del siglo XXI, en este sentido, las ciudades juegan un papel fundamental debido al desarrollo que se vive en las áreas urbanas a nivel poblacional, tecnológico, cultural, económico y social.

La importancia de las ciudades se debe en gran medida al crecimiento que sufren, dado que la mitad de la población mundial habita en zonas urbanas. Para 2050, las Naciones Unidas prevé que aproximadamente el 70% de los seres humanos vivirán en centros urbanos, advirtiendo que dicho aumento podría suponer un problema solo evitable manteniendo un equilibrio entre los aspectos ambientales, sociales y espaciales de los diferentes lugares, así como entre sus habitantes.

Como hemos comentado anteriormente, la contaminación se ve favorecida de manera directa por este crecimiento de las ciudades que, además, repercute significativamente en el calentamiento global, contaminación del aire, agotamiento de los recursos, escasez de agua y energía, etc.

Es por esto por lo que los gobiernos deben afrontar en un corto plazo esta nueva situación, buscar medidas que intenten hacer frente a los nuevos desafíos y mitigar las ya presentes y futuras consecuencias ocasionadas por el crecimiento urbanístico y los nuevos estilos de vida.

En este nuevo escenario demográfico y sociológico, en el que son claros los efectos económicos, medioambientales y políticos, es dónde aparece el concepto de Smart City (ciudad inteligente), entendida como aquellas ciudades que llevan a cabo soluciones innovadoras y eficientes en áreas como la movilidad de las personas, el consumo energético, la política, el medio ambiente, la población y la economía, con el objetivo de mejorar la calidad de vida de los ciudadanos.

En consecuencia, se ha desarrollado el presente trabajo de investigación en el que nos centramos en la implementación de una app para mejorar la eficiencia de las ciudades y poder hacer la transformación hacia una Smart City. En particular, nos centraremos en el área de eficiencia energética y sostenibilidad, siendo nuestro objetivo el de estudiar diferentes mejoras que permitan una mayor eficiencia energética y una menor contaminación en las ciudades del siglo XXI.

2. Antecedentes

El concepto de Smart City surge en diferentes conferencias y cumbres del clima en las que son abordados los principales problemas relacionados con el medioambiente, aunque no exista una definición establecida. La gran mayoría de los expertos coinciden en que la definición esencial recoge la idea de aquellas ciudades que apuestan por mejorar la vida de sus habitantes a la vez que se apuesta por la sostenibilidad.

En el *Libro Blanco Smart Cities* [1],se establece como propósito final de una *Smart City* "*alcanzar una gestión eficiente en todas las áreas de la ciudad (i.e. urbanismo, infraestructuras, transporte, servicios, educación, sanidad, seguridad pública, etcétera), satisfaciendo a la vez las necesidades de la urbe y sus ciudadanos*".

En España contamos con numerosas ciudades que están desarrollando su transformación hacia Smart Cities.

Santander, con su proyecto Smart Santander, iniciado en 2010 cuenta con la intervención de hasta 15 organizaciones, que tienen como objetivo convertir la ciudad en una Smart City, mejorando así la vida de los ciudadanos. Para ello, han instalado alrededor de 20000 dispositivos repartidos por diferentes zonas tales como sensores, cámaras, captadores de temperatura, presión atmosférica, etc., buscando reducir la contaminación de la ciudad, y haciendo eficiente los procesos de recogida de basura, de aparcamiento, y de instalaciones lumínicas.

Otros casos que tenemos en nuestro país son, por ejemplo, las ciudades de Madrid y Barcelona, que adoptan cada año medidas tecnológicas nuevas para reducir los altos niveles de contaminación, mejorar el tráfico, y disponer de un sistema de información global que consigan la mejora de la eficiencia y de la calidad de vida de sus habitantes.

3. Motivación

Dada la importancia actual del cambio climático, contar con ciudades que sean eficientes y busquen reducir sus niveles de contaminación es imprescindible. Por esta razón la motivación principal para la realización de este Trabajo de Fin de Máster, en adelante, TFM, es la de implementar una aplicación móvil que sirva para mejorar la eficiencia de las ciudades y permita transformarlas en Smart Cities contribuyendo a la reducción de los problemas medioambientales a los que nos enfrentamos hoy en día, con la idea de conseguir un mundo

mejor para las futuras generaciones, motivo principal por el que decidí estudiar una ingeniería y por el que he decido llevar a cabo este proyecto.

Por otro lado, dado los avances tecnológicos y la transcendencia que tienen en la actualidad los dispositivos móviles, se considera que es necesario desde el punto de vista de la ingeniería tener conocimientos de programación de aplicaciones Android ya que los dispositivos inteligentes están ganando un peso cada vez mayor.

En conclusión, el permitir aumentar mis conocimientos en tecnología punta y ser capaz de utilizarlo en la búsqueda de metodologías que mejoren el planeta, han sido los principales motivos por los que decidí proponer este Trabajo de Fin de Máster.

4. Objetivos

El objetivo principal de este TFM es la realización de una aplicación, en adelante app, que trabaje con diferentes conjuntos de datos en Android Studio con el fin de mejorar la eficiencia de las ciudades. La app tiene diferentes funciones en primer lugar se dispone de un apartado para que el usuario pueda controlar la energía y reducir sus gastos por medio de la implementación de una calculadora de consumo para los diferentes electrodomésticos, un sistema para la obtención aproximada del coste de la factura eléctrica, y la visualización del precio de la energía en tiempo real. En segundo lugar, se ha programado un apartado de transporte sostenible, que tiene como fin mostrar a los usuarios la localización tanto del transporte público como los puntos de carga de los coches eléctricos para las ciudades de Santa Cruz de Tenerife y Las Palmas de Gran Canaria. Además, se incluye otro apartado donde se muestran diferentes consejos para conseguir ahorrar energía y reducir así la contaminación.

5. Metodología

Para cumplir con los objetivos nombrados en el apartado anterior, se debe acceder a los datos en tiempo real proporcionados por Red Eléctrica de España [2], sobre el precio de la energía para así poder programar tanto la calculadora de consumo de los electrodomésticos, como la opción de generar la factura.

Es necesario también tener acceso a los datos abiertos que proporcionan el Ayuntamiento de Santa Cruz de Tenerife y el de Las Palmas de Gran Canaria para poder programar un mapa que conecta con Google Maps y que muestra la ubicación de los diferentes medios de transporte de los que se tienen datos.

Con el acceso a ambos conjuntos de datos se implementará cada una de las funciones de la app desarrollando una interfaz que sea fácil de entender para todos los usuarios y permita tener un mayor conocimiento de cómo mejorar la eficiencia energética en nuestros hogares y por ende en las ciudades.

6. Estructura del documento

El presente proyecto vendrá estructurado de la siguiente manera:

- En primer lugar, se dispondrá de un capítulo de introducción que permita al lector conocer aquellos aspectos previos y necesarios para poder comprender la elaboración y el funcionamiento de este proyecto.
- En segundo lugar, nos encontraremos con un capítulo de conocimientos previos, que tiene como finalidad explicar una serie de conocimientos básicos sobre los elementos empleados en el proyecto con el fin de facilitar la comprensión y el entendimiento del contenido de este Trabajo de Fin de Máster.
- En tercer lugar, tendremos un capítulo que contendrá el desarrollo del proyecto en sí, explicando todos los pasos y tareas llevadas a cabo para realizar los objetivos previstos.
- Finalmente, podremos encontrar las conclusiones obtenidas y futuros pasos que se podrían implementar.

Capítulo 2. Conceptos Previos.

Índice Conceptos Previos

1. Smart Cities

Las ciudades son de gran interés para la innovación y la tecnología desde hace siglos. Desde la época de los griegos y romanos eran grandes centros de estudios y, con la llegada de la Revolución Industrial, sufrieron un gran cambio a nivel social, económico y cultural.

Actualmente la mitad de la población mundial vive en zonas urbanas, y las Naciones Unidas estima que para 2050 sea el 70 %. Este aumento de habitantes afecta de manera significativa a la huella ecológica, produciendo mayores emisiones de gases a la atmósfera, menor disponibilidad de recursos, aumento de la deforestación, etc. Todas estas cuestiones han puesto de manifiesto la necesidad de mejorar la sostenibilidad y eficiencia de los centros urbanos, surgiendo así el concepto de Smart City o ciudad inteligente.

Las Smart Cities no tienen una definición exacta y su significado ha ido variando a lo largo de los años, aunque hoy en día sigue sin haber un acuerdo a la hora de definir un concepto consolidado. Muchas veces la definición de ciudad inteligente depende del ámbito en el que haga. Por ejemplo, si las analizamos desde el punto de vista de la tecnología e información, son aquellos lugares de donde se dispone de toda la información necesaria para tomar decisiones de manera eficiente, mientras que, si las analizamos desde el punto de vista de la sociedad se trata de áreas limpias con buenos transportes urbanos, responsables con el medio ambiente y tecnológicamente avanzadas. Se tratan, por tanto, de "*ciudades en las que coexisten la tecnología, la información y la población teniendo como objetivo la optimización de todos los recursos, reducción las emisiones de gases peligrosos para el plante, así como el aumento la calidad de vida de sus habitantes*" (Chourabi, et al., 2012).

Dada la inexistencia de una definición universal, se busca identificar y desarrollar las Smart Cities de acuerdo con una serie de características comunes, clasificadas dentro de cinco ámbitos:

- Economía.
- Ciudadanía.
- Gobierno.
- Movilidad.
- Sostenibilidad.

Ilustración 1. Características de las Smart Cities.
Fuente: https://www.etnahitech.com/2014/06/19/smart-cities-ecco-le-migliori-opportunita-di-investimento/

1.1. Características de las Smart Cities

1.1.1. Economía

Las Smart Cities son un modelo de desarrollo urbano que permite mejorar la competitividad de los territorios, siendo la economía de gran importancia en el desarrollo de las ciudades inteligentes que buscan una economía sostenible y competitiva que permita mejorar la calidad de vida de sus habitantes.

Una buena organización económica atrae a inversores y permite crear empleo sostenible como por ejemplo el sector energético o de las TICs, conjunto de servicios públicos, etc., permitiendo un gran crecimiento que posibilita la competitividad entre las diferentes compañías, fomentando el trabajo colaborativo, los proyectos de I+D+i Público-Privada…

Además, se añade un sector de importancia como el turismo, que juega un papel fundamental en el desarrollo de estas ciudades. Cada vez son más los turistas que deciden visitar una ciudad basándose en la eficiencia del sistema de movilidad o la conservación del medio ambiente, por lo que el desarrollo inteligente de una ciudad atrae a un mayor número de visitantes, favoreciendo así a la economía.

Sin embargo, conseguir una economía sostenible no es una tarea sencilla ya que requiere de un gran coste inicial y la necesidad de conseguir un entendimiento por parte de las empresas que representan a los distintos sectores. Para llevar a cabo estas inversiones, es fundamental cuantificar el retorno de inversión, aunque en muchos casos resulte una tarea complicada.

1.1.2. Ciudadanía

El éxito de una Smart City debe tener como eje principal su población. Para ello es necesario que la ciudadanía tenga capacidad de participación en la evaluación de proyectos, en el desarrollo de leyes, en los diferentes planes y programas, etc., manteniendo la implicación de los habitantes en la filosofía de la ciudad.

Los gobiernos deben ser consciente del papel tan importante que juegan sus ciudadanos, siendo imprescindible que colaboren de una manera consciente e informada. Si todos los miembros de la ciudad son conocedores de los diferentes proyectos y se sienten parte de ellos, será mucho más sencillo el éxito de estos y el avance de la ciudad hacia la sostenibilidad y eficiencia. Para ello, existen diferentes maneras de mantener un contacto directo con la población, como pueden ser la creación de sitios web o directamente utilizando las redes sociales dónde cada habitante puede participar de manera rápida y sencilla en encuestas o votaciones que le permitan sentirse parte de un proyecto fomentando además una mayor transparencia del gobierno.

Un aspecto vital en las Smart Cities es el nivel sociocultural de los ciudadanos. Para que la población sea lo más participativa posible es necesario que la gran mayor parte de los habitantes tengan acceso y conocimiento de uso de los diferentes sistemas utilizados por el gobierno. El éxito de la ciudad será mayor cuanto más alto sea su nivel sociocultural.

1.1.3. Gobierno

A la hora de transformar una ciudad en una Smart City es esencial que el Gobierno y la Administración se muestren abiertos y sean accesibles para los habitantes, siendo la transparencia uno de sus grandes retos. En este sentido, movimientos como el Open Data y el Open Government que permiten el intercambio y aportación de datos a la ciudadanía, dejándolos abiertos para que se pueda hacer uso de ellos, son la clave para conseguir el éxito de las Smart Cities.

Para conseguir la transparencia y participación deseada son necesarias las Tecnologías de la Información y las Comunicaciones (TICs) que permiten la buena gestión y el crecimiento de la ciudad. Así pues, como se establece en el Libro Blanco de las Smart Cities [1], para un buen uso de estas se debe llevar a cabo un conjunto de actuaciones:

- La modernización administrativa.
- La integración de servicios digitales.

- La digitalización de la información.
- Implantación de una administración electrónica conocida como “Ciudad Digital” que ofrece servicios online como por ejemplo el acceso a información pago de tasas e impuestos, ejecución de trámites.

Algunas herramientas TICs de uso frecuente en Smart Cities son, por ejemplo, portales multiacceso como páginas web o internet móvil, servicio de atención telefónica o presencial, tarjetas inteligentes para acceder a los servicios, puntos inalámbricos de conexión Wifi, etc.

Por otro lado, el sector servicio es de gran importancia ya que supone un tercio del presupuesto municipal. Las Smart Cities deben ser capaces de ofrecer mejores servicios a un precio menor. Para ello es necesario un control total de la información que permita mejorar la toma de decisiones, aumentar la eficiencia, evaluar el rendimiento y mejorar los servicios.

Para que todos estos pasos se puedan llevar a cabo es de vital importancia el respaldo de un sistema fiscal que funcione de manera correcta. Por ejemplo, mediante incentivos fiscales que sirvan de motivación para las empresas para realizar sus tareas de manera eficiente y sostenible.

1.1.4. Movilidad

La movilidad en las Smart Cities va de la mano con la seguridad, eficiencia y sostenibilidad de sus sistemas de transporte e infraestructuras y se relacionan con la accesibilidad local, nacional e internacional. En este sentido, los Planes de Movilidad Urbana Sostenible (PMUS) cobran un papel fundamental. Estos planes son una agrupación de acciones que tienen como fin la implantación de sistemas de desplazamientos más sostenibles, permitiendo una mejora del medio ambiente y ayudando al crecimiento económico de la ciudad.

Su implantación es necesaria ya que presenta numerosas ventajas como, por ejemplo, la disminución de los atascos y por ende la reducción de los efectos negativos de estos, disminución del consumo de combustibles fósiles, mejora de la calidad del medio ambiente, de los servicios públicos y de la salud de la población, etc.

Para poder cumplir con estos objetivos, el Libro Blanco de las Smart Cities [1] nombra numerosos servicios que pueden implantarse en la Smart City que cumpla al mismo tiempo con la innovación, sostenibilidad, seguridad y eficiencias deseadas:

- Sistemas de información online en tiempo real que permitan a los habitantes tener conocimiento sobre plazas de aparcamiento o puntos de recarga para los vehículos eléctricos.

- Servicios de información online para la ciudadanía mediante sus smartphones o pantallas fijas que les den información sobre el tiempo de llegada del transporte público, servicios para compartir bicicletas o vehículos (*car sharing*), situación del estado del tráfico, etc.
- Detección automática de las infracciones que tengan lugar en la carretera, permitiendo informar con la señalización adecuada y de manera online de los posibles accidentes.
- Análisis de los flujos de tráfico para poder prioridad al transporte de emergencia y transporte público.
- Sistemas inteligentes de control de semáforos que permitan detectar la presencia de vehículos y coordinar el cambio de color de manera eficiente.
- La implementación de las denominadas "supermanzanas" que tienen como objetivo desplazar la circulación del transporte privado a las carreteras exteriores permitiendo únicamente el uso de transporte público en el interior de estas.

1.1.5. Sostenibilidad

La sostenibilidad es una característica importante dentro de las Smart Cities ya que permiten sean ciudades atractivas gracias a sus cualidades medioambientales y naturales. Para lograrlo, es necesario que una ciudad inteligente proteja el medio y tome medidas correctas de gestión en la medida en que se desarrolle una estrategia basada en las características del territorio que potencia los atractivos medioambientales y reduzca las debilidades.

Las medidas que se pueden llevar a cabo dentro del ámbito de la sostenibilidad son numerosas, a continuación, se destacan algunas de ellas:

- El Programa 21 Local, promovido por las Naciones Unidas, basado en la creación de una herramienta que permita controlar la gestión medioambiental de una localidad.
- Promover las iniciativas de e-Administración que fomentan la participación ciudadana de manera telemática disminuyendo el consumo de papel y por ende las emisiones.
- Dotar a las ciudades de programas para la planificación ambiental que incluyan medidas de protección y gestión del territorio y de los recursos.
- Controlar los niveles de contaminación y calidad de agua, utilizando redes de telecontrol.
- Llevar a cabo un estudio para determinar el mejor sistema de gestión de residuos.
- Conocer la situación inicial de la ciudad para poder detectar aquellos sectores que pueden disminuir la contaminación. Es necesario disponer de gran cantidad de datos para identificar dichos puntos, y poder promover soluciones, como, por ejemplo, mayor uso del transporte público, instaurar medidas de fiscalidad ambiental, etc.

- Impulsar la creación y uso de zonas verdes y la buena gestión de sus recursos naturales.
- Establecer criterios de edificación sostenible, por certificaciones de sostenibilidad.
- Facilitar y fomentar el uso de renovables en los servicios y en terrenos de particulares.

1.2. ¿Por qué transformar una ciudad en Smart City?

Como se ha comentado con anterioridad la transformación de una ciudad en una Smart City es un proceso lento y costoso, pero absolutamente necesario. Dado el avance de la tecnología y la situación actual del planeta disponer de ciudades que sean eficientes, respetuosas con el medio ambiente y que mejoren la calidad de vida de sus ciudadanos es un camino hacia el que hay que optar, es el futuro.

Las Smart Cities tienen numerosas ventajas frente a una ciudad convencional y aunque supongan una elevada inversión inicial ese gasto económico se verá retornado en el largo plazo. Las ciudades inteligentes suponen una conexión entre el gobierno, la población y el lugar en dónde vive, permiten que todas las personas se sientan participes de las acciones que se llevan a cabo en su ciudad haciendo que se sientan involucradas y orgullosas de habitar en un lugar que respeta el medio ambiente, que reduce las emisiones de gases a la atmosfera, mejora el control de los residuos y ofrece numerosas oportunidades tanto para su ciudadanía como para las personas que la visitan.

Para poder lograr esta transformación es necesario una correcta planificación y gestión de todos los componentes que intervienen y un compromiso de todas las partes.

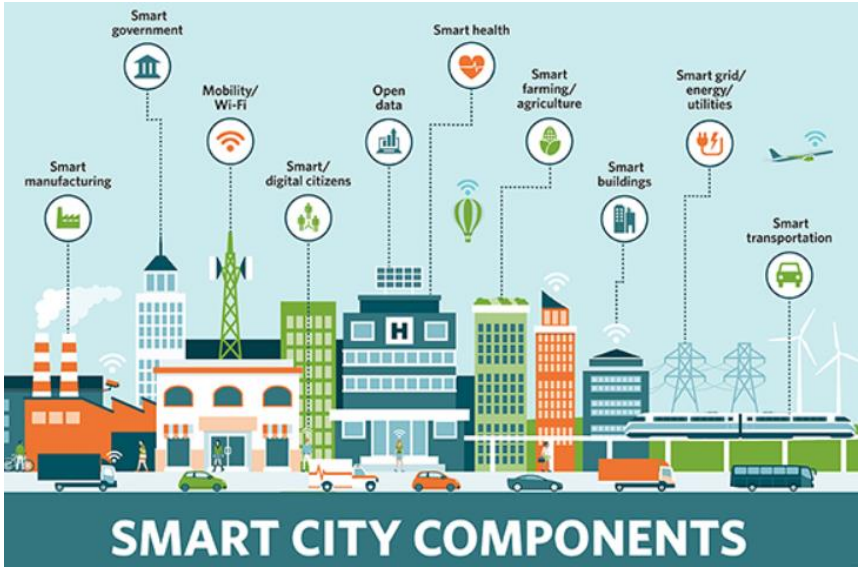


Ilustración 2. Componentes de una Smart City.
Fuente: http://www.xorlogics.com/tag/smart-city/

2. Android

2.1. ¿Qué es?

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para la programación y desarrollo de aplicaciones para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets o phablets; y también para relojes inteligentes, televisores y automóviles.

2.2. Historia

Inicialmente Android fue desarrollado por la empresa Android Inc., respaldada en sus inicios económicamente por Google procediendo en 2005 a su compra.

Android fue presentada en 2007 junto la fundación Open Handset Alliance para avanzar en los estándares abiertos de los smartphones.

Ese mismo año se lanzó la primera versión de Android SDK, siendo un conjunto de herramientas de desarrollo, bibliotecas, simulación, etc., que permiten la programación de aplicaciones Android en lenguaje Java.

El primer móvil con este sistema operativo fue el HTC Dream vendido en octubre de 2008. Evolucionando a lo largo de los años, convirtiéndose en el sistema operativo más utilizado.

2.3. Arquitectura

El sistema operativo Android está desarrollado en varios lenguajes de programación (C, C++, Java y XML). En el artículo "Arquitectura de la plataforma" de Android Developers [3] se indica que tiene una arquitectura basada en capas, como se ve en la *Ilustración 3.*

Analizando desde arriba hacia abajo cada una de las capas nos encontramos con:

1. Capa de Aplicaciones: En este nivel encontramos las aplicaciones que están incluidas por defecto de Android (calendario, email, widgets, contactos, cámara…) y las que el usuario va añadiendo posteriormente.
2. Entorno de Aplicación (*Framework*): Se trata del conjunto de herramientas de desarrollo. Todas las aplicaciones implementadas para Android utilizan estas APIs. Esta capa fue creada para simplificar la reutilización de componentes y permite a cada aplicación publicar sus capacidades para que otras aplicaciones puedan hacer uso de ellas. Algunas de estas APIs son:

- *Activity Manager:* gestiona el ciclo de vida de las aplicaciones de Android.
- *Location Manager:* gestión de información de localización.
- *Package Manager:* da información sobre las apps instaladas en el dispositivo.
- *Notification Manager:* comunica al usuario eventos que ocurran durante la ejecución.
- *Resourse Manager:* controla el acceso a recursos.
- *Telephony Manager:* gestiona las funcionalidades del teléfono.
- *Window Manager:* gestiona ventanas de las aplicaciones.
- *XMPP Service:* permite utilizar el protocolo de intercambio de mensajes de en XML.

3. Librerías Nativas: Escritas utilizando C/C++ proporcionan a Android la mayor parte de sus capacidades. Junto al núcleo basado en Linux, constituyen la parte fundamental de Android. Algunas de estas librerías son:
   - *WebKit:* permite utilizar aplicaciones web.
   - *Media Framework:* posibilita la grabación y reproducción de audios.
   - *OpenGL ES/SGL:* permite gráficos 3D y 2D.
   - *Surface Manager:* compuesto por los sistemas de navegación y ventanas.
   - *SQLite:* permite la creación de bases de datos, y también su gestión.
   - *Free Type:* gestiona los distintos tipos de fuente.
   - *SSL:* posibilita utilizar comunicaciones seguras.
4. Runtime: Formado por las *Core Libraries* (librerías con clases Java y una máquina virtual). Debido a que los dispositivos en los que se ejecuta Android tienen poca memoria y un procesador limitado, no es posible emplear una máquina virtual Java. Para solucionar este problema Google creó una máquina virtual.
5. Capa de abstracción de Hardware (HAL): proporciona las interfaces estándares que permiten la interacción entre el *Framework* de la API Java de alto nivel y las capacidades hardware del dispositivo.
6. Kernel de Linux: Contiene los drivers necesarios para que los componentes hardware puedan ser utilizados. Está basado en *Linux 2.6* y permite, entre otras cosas, la gestión de memoria, de procesos, de batería, instalar los distintos drivers etc.

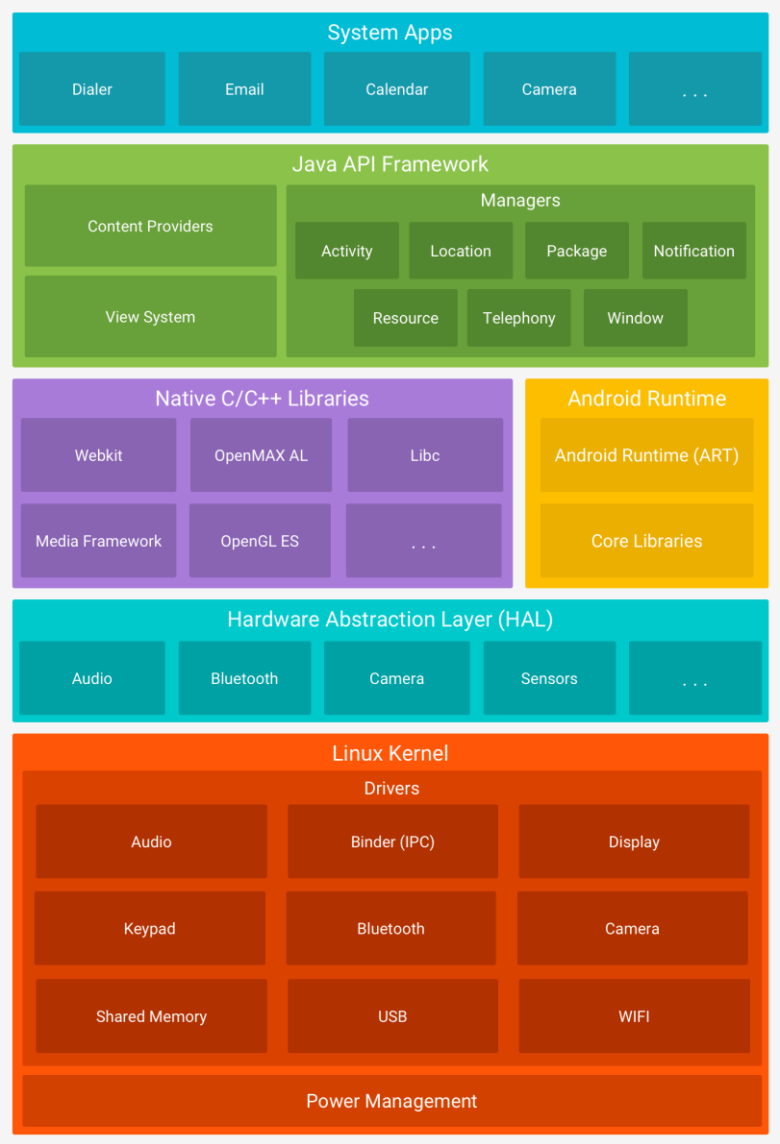


Ilustración 3. Pila de software de Android.
Fuente: https://developer.android.com/guide/platform

2.4. Versiones

Android es un sistema operativo que no ha parado de evolucionar a lo largo de los años. Si se analizan todas las actualizaciones del sistema operativo hasta llegar a la actual, se comprende la gran evolución que ha experimentado. Hoy en día, cada nueva actualización incorpora nuevas o mejores funcionalidades.

Las versiones de Android [4] han sido desarrolladas bajo un nombre clave siguiendo un orden alfabético:

- Android 1.0 – Nivel API 1 (septiembre 2008) – Apple Pie.
- Android 1.1 – Nivel API 2 (febrero 2009) - Banana Bread.
- Android 1.5 – Nivel API 3 (abril 2009) – Cupcake.

- Android 1.6 – Nivel API 4 (septiembre 2009) – Donut.
- Android 2.0-2.1 – Nivel API 5-7 (octubre 2009) – Éclair.
- Android 2.2 – Nivel API 8 (mayo 2010) – Froyo.
- Android 2.3 – Nivel API 9-10 (diciembre 2010) – Gingerbread.
- Android 3.0 – Nivel API 11-13 (febrero 2011) – Honeycomb.
- Android 4.0 – Nivel API 14-15 (octubre 2011) - Ice Cream Sandwinch.
- Android 4.1-4.3 – Nivel API 16-18 (julio 2012) - Jelly Bean.
- Android 4.4 – Nivel API 19-20 (octubre 2013) – KitKat.
- Android 5.0 – Nivel API 21-22 (noviembre 2014) – Lollipop.
- Android 6.0 – Nivel API 23 (octubre 2015) – Marshmallow.
- Android 7.0 – Nivel API 24-25 (junio 2016) – Nougat.
- Android 8.0 – Nivel API 26-27 (agosto 2017) – Oreo.
- Android 9.0 – Nivel API 28 (agosto 2018) – Pie.
- Android 10.0 – Nivel API 29 (septiembre 2019).

La fragmentación de las distintas versiones de Android en los diferentes terminales del mercado se puede contemplar en el siguiente gráfico que muestra para cada versión el porcentaje de dispositivos que la poseen:

Ilustración 4. Gráfico fragmentación versiones Android.
Fuente: Elaboración propia a partir de los datos de
https://upload.wikimedia.org/wikipedia/commons/e/ee/Android_historical_version_distribution_-_vector.svg

3. Conceptos Energéticos

3.1. Tarifas de Red Eléctrica

Uno de los factores que permite que dispongamos de energía eléctrica en nuestros hogares es la red eléctrica formada por las líneas de alta tensión que transportan la electricidad desde la central donde se genera hasta nuestras casas.

En nuestro país hay más de 43.000 kilómetros de líneas de transporte que requieren mantenimiento en las mejores condiciones [5]. Para ello, en España tenemos las denominadas tarifas de acceso que son las que se encargan de recaudar el dinero necesario para el mantenimiento de la red.

Las tarifas de acceso aparecen en la factura de la luz por medio de las siglas ATR (Acceso de Terceros a la Red) y existen diferentes tipos. Solo por estar conectado a la red eléctrica ya se tiene pagar esa tarifa que se verá reflejado en el término fijo de potencia y en el término variable de consumo. En el caso de estar en el mercado regulado (tarifa PVPC) el coste vendrá desglosado, pero si se está en el mercado libre, el coste irá incluido en el precio de kW de potencia y del kWh de consumo.

El coste de las tarifas de acceso es independiente del lugar en el que vivas y de la compañía eléctrica, este precio viene fijado por el Gobierno.

No se puede elegir directamente la tarifa de acceso que se nos va a aplicar, pero ésta dependerá de otros factores sobre los que sí podemos influir:

- Potencia contratada. Todas las potencias inferiores a 10 kW tendrán una tarifa de acceso 2.0. Las que se encuentren entre 10 kW y 15 kW, tendrán una tarifa 2.1. Por encima de 15 kW, y mientras sigamos en baja tensión, la tarifa de acceso será la 3.0.
- Discriminación horaria: todas las tarifas de acceso antes mencionada tienen un "apellido" en función de si tienen o no discriminación horaria. Si no tienen, llevarán una A después del número (2.0A y 2.1A). Si tienen discriminación horaria en dos periodos (punta y valle), llevarán una DHA después del número (2.0DHA y 2.1DHA). Si tienen discriminación en tres periodos (punta, valle y supervalle), llevarán una DHS después del número (2.0DHS y 2.1DHS). La excepción es la 3.0A, que siempre tiene tres periodos (punta, llano y valle).

A mayor potencia contratada mayor será el precio que tenemos que pagar.

Potencia	Sin DH	Con DH 2 periodos	Con DH 3 periodos
< 10 kW	2.0A	2.0DHA	2.0DHS
De 10kW a 15kW	2.1A	2.1DHA	2.1DHS
> 15 kW	3.0A (*Siempre tiene DH 3 periodos)		

Ilustración 5. Tipos de tarifa de acceso.
Fuente: *https://www.endesa.com/es/conoce-la-energia/energia-y-mas/tarifas-acceso-electricidad*

3.2. Consumo de los electrodomésticos

A la hora de calcular el consumo eléctrico de los electrodomésticos hay diferentes factores que debemos tener en cuenta. Por ejemplo, el tipo y marca, ya que no todos cumplen con los mismos barómetros. No es igual un horno que una nevera. Pero tampoco son iguales los consumos de todas las lavadoras, pues existen diferencias por el modelo o si se trata de un electrodoméstico de mayor eficiencia (A++) o de menor.

Es importante conocer cómo y cuánto consumen los diferentes aparatos, para ello disponemos de la etiqueta de eficiencia energética que nos ofrece una información muy relevante a la hora de decidir comprar un nuevo electrodoméstico o si nos interesa saber cuánto consume el que ya tenemos.

Los valores de potencia de consumo pueden oscilar entre varios rangos. Una muestra de alguno de ellos es la siguiente:

Electrodomésticos	Potencia en kW
Nevera	entre 0,250 - 0,350 kW
Vitrocerámica	entre 0,900 - 2,000 kW
Microondas	entre 0,900 - 1,500 kW
Horno	entre 1,200 - 2,200 kW
Lavavajillas	entre 1,500 - 2,200 kW

Ilustración 6. Rango de los valores de consumo de potencia de diferentes electrodomésticos.
Fuente: *https://lucera.es/blog/cuanto-consumen-electrodomesticos*

Una vez que se conoce la potencia que consume se puede calcular el coste del consumo eléctrico de dicho aparato por horas, días, meses o incluso al año. Para ello es necesario saber el precio de la energía, el período de tiempo en el que lo queremos calcular y el impuesto que se le va a aplicar:

$$Coste\ consumo = (potencia * tiempo * precio\ luz) + (potencia * tiempo * impuesto * precio\ luz)$$

Así un horno que consuma 1500 W, se utilice media hora al día, con un precio de la energía de 0.086 €/kWh y al que se le aplique el IGIC:

$$Coste\ consumo = (1.5 * 0.5 * 0.086) + (1.5 * 0.5 * 0.07 * 0.086) = 0.069015\ €/hora$$

3.3. Factura de la luz

La factura de la luz es el reflejo de lo que el cliente debe pagar a su comercializadora como justificación del abastecimiento de electricidad y está estructurada en diferentes apartados explicados a continuación [6]:

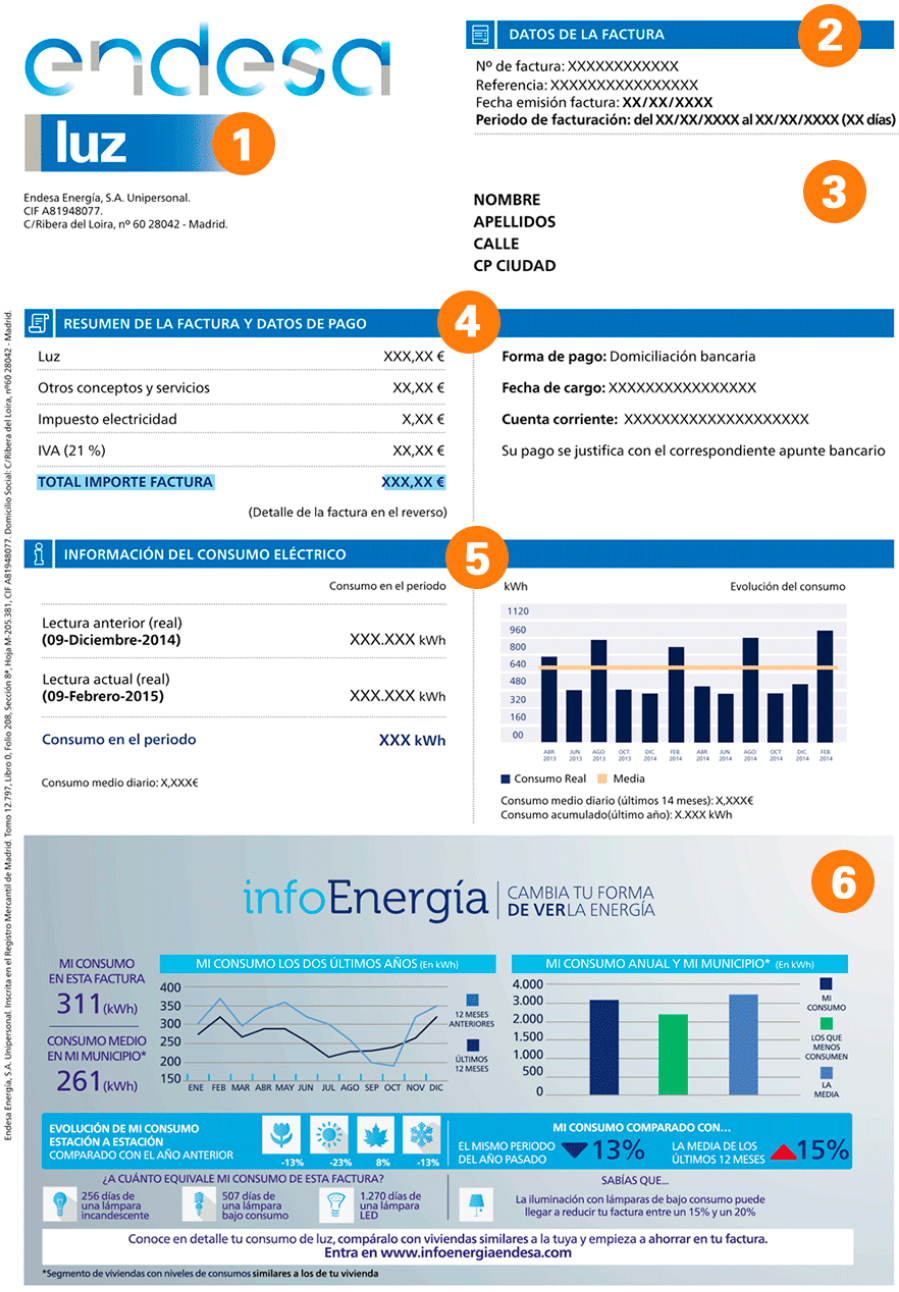


Ilustración 7. Ejemplo factura de luz.
Fuente: https://www.endesa.com/es/te-ayudamos/factura-luz-ml

1. Identificación de la empresa comercializadora.

Este apartado incluye el logotipo, el CIF, la denominación y el domicilio sociales de Endesa Energía.

2. Datos de la factura.

Se muestran todos los datos relativos a la factura:

- Número de la factura: es un número identificativo fiscal para esa factura.
- Referencia: se trata del código de la factura.
- Fecha de emisión factura: es la fecha en la que se emitió la factura.
- Periodo de facturación: consiste en el intervalo de tiempo que se ha facturado. Para el pago domiciliado se muestra la fecha en la que se cobrará el importe, para pago no domiciliado se muestra la fecha límite que se tiene para realizar el pago del importe.

3. Nombre y dirección del cliente.

Contiene los datos personales del titular del contrato.

4. Resumen de la factura y datos.

El resumen de la factura es el apartado en el que nos centraremos en nuestra App y es el más importante puesto que recoge el resumen de los conceptos facturados y el importe total de la factura.

Esta sección está compuesta por los siguientes términos:

- Potencia contratada. Es el importe total que se paga por la potencia que se tiene contratada que dependerá de las características de la instalación eléctrica y del uso que se les dé a los aparatos eléctricos. El precio es siempre el mismo y no depende del consumo.

  Para calcular la potencia contratada se tiene que multiplicar los kW que se han contratado por el precio que tenga el kW por el número de días de facturación:

  $$Potencia\ Contratada = kW\ contratados * precio\ kW * dias\ de\ facturación$$

- Energía consumida. Se trata del importe total que se paga por la energía que realmente se ha consumido durante el período de facturación. El precio es variable ya que depende del consumo que se tenga.

  Para calcular la energía consumida se multiplica el precio del kWh por los kWh consumidos:

  $$Energía\ Consumida = precio\ kWh * kW\ cosumidos$$

- Impuesto de electricidad. Tributo sobre la electricidad que establece el Ministerio de Industria. El impuesto de electricidad supone un 5.113% sobre la suma del término de consumo y el término de potencia:

$$Impuesto\ de\ electricidad = (Término\ de\ potencia + Energía\ cosumida) * 0.05113$$

- Alquiler de equipos de medida y control. Muestra el coste del alquiler por los equipos de medida y control. Se cobra a todos los clientes que no tienen en propiedad su contador de luz.

  En el caso de los contadores inteligentes el Gobierno ha establecido un precio que oscila entre los 0.81€ y 1.36€ al mes.

- Impuesto aplicado. Se trata del IVA (Impuesto sobre el Valor Añadido) o el IGIC (Impuesto General Indirecto de Canarias) para las Islas Canarias.

  En el caso de la península el IVA es del 21% y en el caso de Canarias el IGIC es del 7%, y se calcula sobre toda la factura:

$$Impuesto\ aplicado = (Término\ de\ potencia + Energía\ cosumida + Alquiler\ de\ equipos + Impuesto\ sobre\ la\ electricidad) * 0.07$$

  En el caso de la península se multiplicaría por 0.21

- Total Importe Factura. Importe total que pagar. Se calcula sumando todos los términos:

$$Total\ Importe\ Factura = (Término\ de\ potencia + Energía\ cosumida + Alquiler\ de\ equipos + Impuesto\ sobre\ la\ electricidad + Impuesto\ aplicado)$$

- Otros conceptos. Dependiendo del tipo de factura o de la tarifa contratada pueden aparecer conceptos como la regularización, los derechos de contratación e información sobre el número de cuenta donde se realizará el cargo de la factura si es pago domiciliado o, el código de barras para realizar el pago en caso no domiciliado.

5. <u>Información del consumo eléctrico.</u>

En este apartado se encuentra toda la información detallada sobre el consumo eléctrico:

- Lectura actual y anterior del contador (real o estimada). En el caso de tener Discriminación Horaria se muestra el consumo para cada uno de los periodos.
- Consumo en el periodo en kWh. Muestra el consumo que se ha tenido durante el periodo facturado. Se calcula como la diferencia entre la lectura actual y la anterior.
- Gráfico de evolución del consumo. Se trata de un gráfico hecho durante los últimos 14 meses que incluye información sobre lecturas reales y estimadas.
- Importe en euros del consumo medio diario en el periodo facturado.
- Importe en euros del consumo medio diario de los últimos 14 meses.
- Consumo acumulado del último año en kWh.

6. Ofertas y productos personalizados

La factura puede incluir un conjunto de publicidades personalizadas para el cliente con promociones de la comercializadora u otros datos de interés.

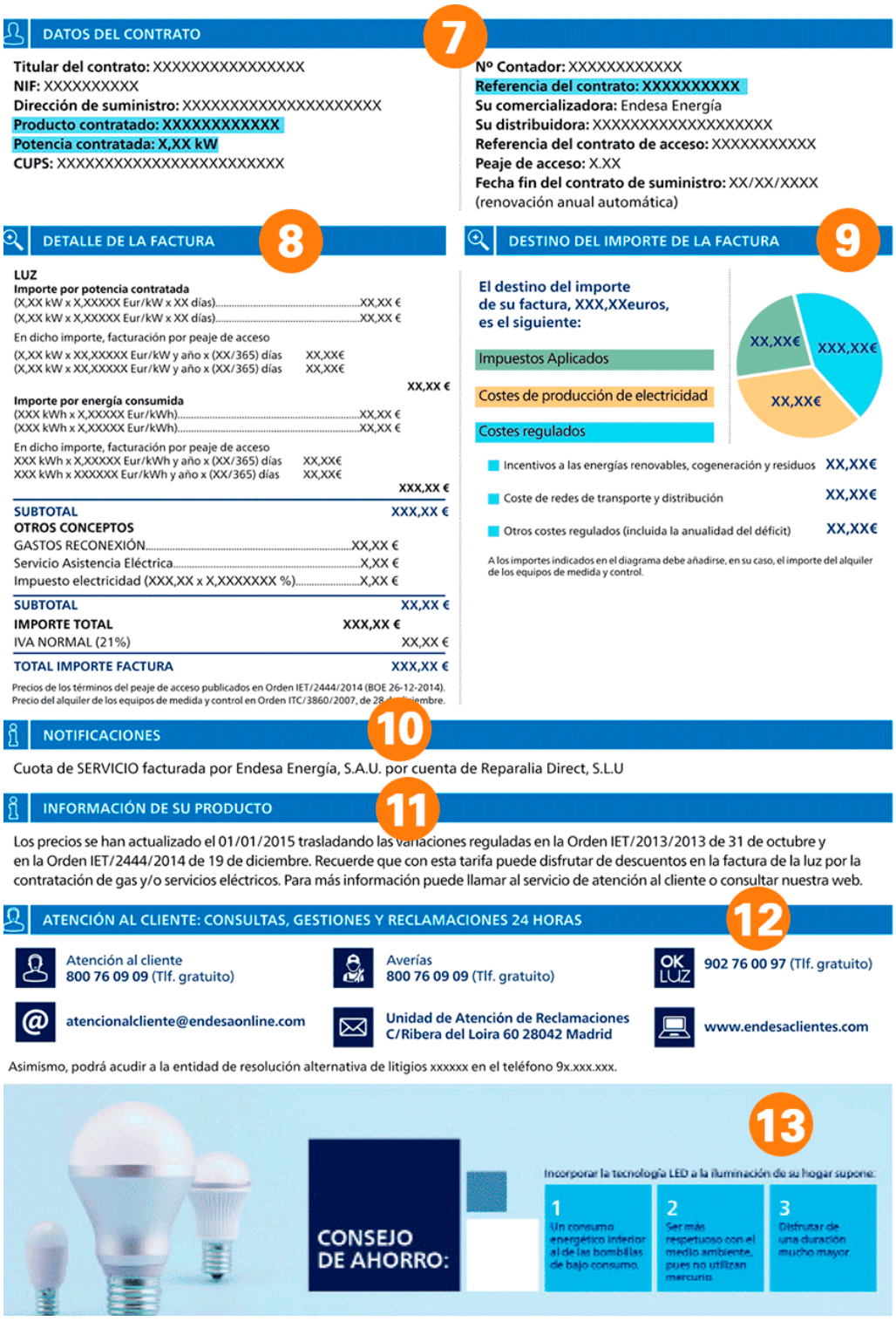


Ilustración 8. Ejemplo factura de luz.
Fuente: https://www.endesa.com/es/te-ayudamos/factura-luz-ml

7. Datos del contrato.

Este apartado engloba los principales datos del titular del contrato y otros datos de interés:

- Datos principales del titular del contrato: Nombre y NIF.
- Dirección del punto de suministro: Dirección postal del lugar.
- Producto contratado: Nombre de la tarifa del contrato.
- Potencia contratada: Potencia que tienes contratada.
- Código unificado de punto de suministro (CUPS): Es un código único que identifica el punto de suministro.
- Número de contador: Es el número identificador del contador.
- Referencia de contrato: Número de contrato con Endesa Energía.
- Comercializadora: Nombre de la compañía con quien tienes tu contrato.
- Distribuidora: Nombre de la compañía que se encarga de llevar la electricidad a tu hogar.
- Referencia de contrato de acceso: Número de contrato con la compañía Distribuidora.
- Peaje de acceso: Es la tarifa que paga la compañía Comercializadora a la compañía Distribuidora por el uso de la red de distribución. Esta tarifa la fija el Gobierno.
- Fin de contrato de suministro: Fecha en la que finaliza este contrato.

8. Detalle de la factura.

En este apartado se detalla cómo se calcula cada uno de los conceptos e importes facturados del apartado 4, que hemos explicado anteriormente.

9. Destino del importe de la factura.

Se muestra en un gráfico desglosado dónde se destina el importe total de la factura:

- Coste de producción de la electricidad. Es la suma de los costes de producir la electricidad que llegar a los hogares.
- Costes regulados:
    - Incentivos a las energías renovables, cogeneración y residuos.
    - Coste de redes de transporte y distribución.
    - Otros costes regulados.
- Impuestos aplicados. Son los tributos que se pagan al Gobierno por el IVA/IGIC de la factura y por el impuesto de electricidad.

10. Notificaciones

Este apartado incluye las aclaraciones de la factura en caso de ser necesarias.

11. Información de su producto

Refleja información relevante del producto contratado.

12. Atención al cliente: consultas, gestiones y reclamaciones 24 horas.

Facilita los canales para contactar con la empresa comercializadora (teléfono, email, página web, etc.).

13. Ofertas personalizas

Contiene un conjunto de publicidades personalizadas para el cliente con promociones de la comercializadora u otros datos de interés.

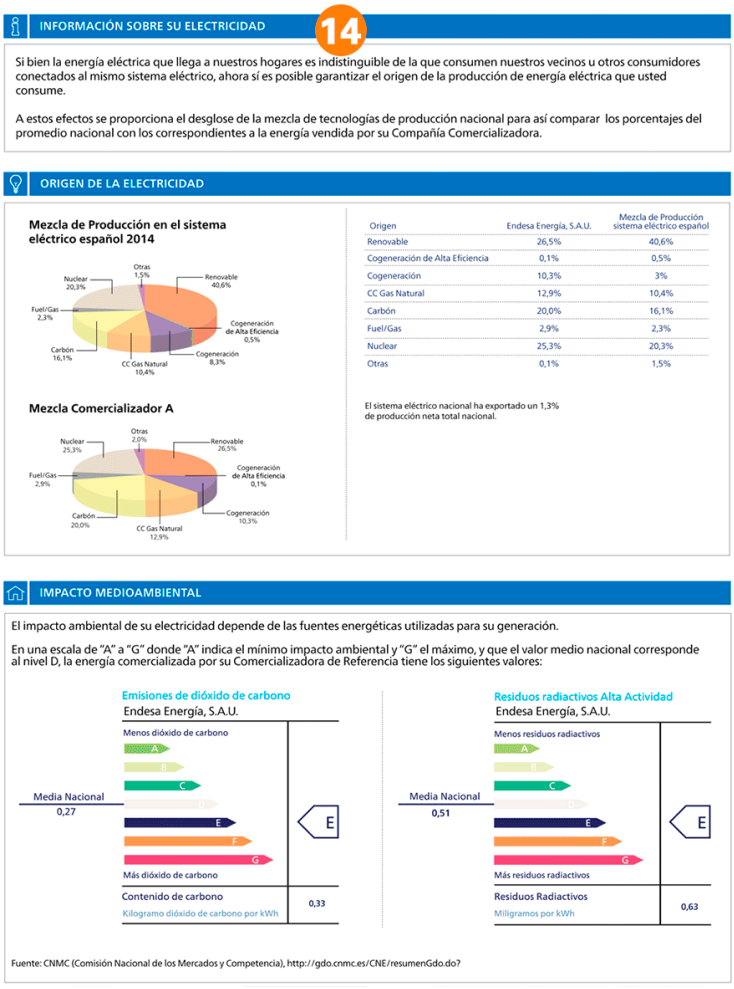


Ilustración 9. Ejemplo factura de luz.
Fuente: https://www.endesa.com/es/te-ayudamos/factura-luz-ml

14. Información sobre su electricidad, origen de la electricidad e impacto medioambiental.

Recoge información sobre el origen de la energía consumida durante el año anterior y su impacto en el medio ambiente.

3.4. Instalación de Autoconsumo

De acuerdo con lo recogido en la Ley 24/2013, del 26 de diciembre del Sector Eléctrico [7], "*el autoconsumo energético consiste en el uso de la energía generada por una instalación para el consumo propio*".

Una instalación de autoconsumo permite a los clientes la incorporación de energías más baratas y respetuosas con el medio ambiente, reduce las necesidades de la red eléctrica y las emisiones de gases de efecto invernadero, generando así mayor independencia energética.

El desarrollo tecnológico que ha tenido lugar en las energías renovables ha conseguido hacer que estas instalaciones sean más eficientes y rentables, logrando producir electricidad en condiciones climáticas menos favorables y permitiendo encontrar en el mercado paneles fotovoltaicos a un 70% más baratos que hace diez años.

Por todos estos motivos, apostar por las energías renovables y optar por instalar paneles fotovoltaicos en nuestra vivienda o empresa son una alternativa de futuro que nos llevan a mejorar el planeta y disponer de ciudades más sostenibles.

Actualmente el Real Decreto 244/2019, de 5 de abril, [8] completa el marco regulatorio sobre autoconsumo, impulsado con el Real Decreto-ley 15/2018 que derogó el denominado impuesto al sol, aportando certidumbre y seguridad a los usuarios. En particular desarrolla [9]:

- Las condiciones administrativas, técnicas y económicas del autoconsumo de energía eléctrica.
- Define las instalaciones próximas a efectos de autoconsumo.
- Desarrolla el autoconsumo individual y colectivo.
- Establece un sistema de compensación simplificada entre los déficits de los consumidores y los excedentes de sus instalaciones de autoconsumo.
- Organiza el registro administrativo de autoconsumo de energía eléctrica, así como su procedimiento de inscripción que no supondrá carga administrativa para los consumidores.

Una instalación de autoconsumo requiere de un estudio exhaustivo de la vivienda o empresa dónde se va a instalar, el mercado de paneles solares, los tipos inversores, el cableado, etc.

Sin embargo, muchas empresas del sector realizan estudios estimados para ayudar al cliente conocer cuánto puede ahorrar si apuesta por esta energía renovable.

En el presente proyecto se optará por la segunda de las opciones, haciendo un estudio estimado de la situación en la que se encuentra el cliente y proporcionándole datos del ahorro mensual que podría conseguir.

El objetivo principal es convencer a los usuarios de optar por este cambio y poder transformar las ciudades en Smart Cities capaces de reducir sus niveles de contaminación y mejorar las condiciones de vida de sus habitantes.

4. Datos Abiertos

Los Datos Abiertos (más conocidos como Open Data) es una filosofía que busca que los datos puedan estar disponibles de forma libre para todas las personas, esto es, sin mecanismos de control, copyright o patentes.

Los datos que se publican en la red como datos abiertos deben estar sin procesar, poseer una buena estructura y tener un tipo de formato conocido que permita que se puedan reutilizar de manera sencilla. Se consideran datos abiertos todos aquellos a los que se puede acceder y ser reutilizados sin la necesidad de emplear permisos.

Para el desarrollo de la aplicación se utilizarán datos abiertos de la administración pública que se pueden encontrar en la web de diferentes ayuntamientos. El sector público produce grandes cantidades de información de interés tanto como para los ciudadanos como para las empresas dadas sus características de fiabilidad, calidad y de información completa.

Los datos abiertos de la administración pública permiten que se implemente la cultura de las Smart Cities ya que favorece la transparencia, participación y colaboración ciudadana, permitiendo que cualquier persona pueda trabajar con esos datos y desarrollar nuevas ideas o mejorar las ya existentes.

Para impulsar la transformación de ciudades convencionales y de nuestro entorno en Smart Cities utilizaremos en este proyecto los datos abiertos de los Ayuntamientos de Santa Cruz de Tenerife [10] y de Las Palmas de Gran Canaria [11]. Trabajaremos con ellos para desarrollar una aplicación que mejore el entorno y permita que ambas capitales caminen hacia un modelo más sostenible.

Se emplearán también los datos abiertos que proporciona Red Eléctrica Española (REE) para hacer un estudio del origen e impacto medioambiental de la energía en las Islas Canarias. REE permite el desarrollo de la cultura Open Data dejando sus datos referentes a la energía a la disposición de aquellas personas que estuviesen interesadas, permitiendo trabajar en muchos

casos con datos en tiempo real. Aprovechando esta ventaja se ha utilizado la web de REE [2] para implementar un método que permita comparar la situación pasada y presente de la energía en Canarias y entender la necesidad de apostar por una energía limpia y renovable que vaya de la mano con la transformación de nuestras ciudades en Smart Cities.

Capítulo 3. Desarrollo del Proyecto.

Índice Desarrollo del Proyecto

Índice de Ilustraciones

1. Entorno de Desarrollo

1.1. Instalación del entorno

El primer paso para poder llevar a cabo este proyecto es instalar el entorno de desarrollo junto con los archivos necesarios para la depuración, simulación y funcionamiento. Para ello se recurre a la página oficial de Android Developers [3] donde debemos descargar el entorno en su versión más actualizada. En nuestro caso, es la versión 3.6.3.

Para poder simular la aplicación creada, Android Studio posibilita la opción de instalar la app en un móvil virtual que funciona prácticamente igual que uno físico, y permite depurar y comprobar el funcionamiento del código. Además del smartphone virtual es posible conectar un dispositivo móvil real para probar la aplicación directamente sobre este.

Trabajar con un teléfono móvil de manera física requiere seguir una serie de pasos además de instalar el entorno y conectarlo al ordenador. En primer lugar, es necesario activar la depuración USB en nuestro dispositivo móvil, esta depuración es una modalidad exclusiva de Android que autoriza el envío de comandos desde el PC al teléfono por medio del cable USB. Para poder activar dicha opción debemos primero activar las opciones de desarrollo y, a continuación, la opción de depuración. Hecho esto, puede que aún Android Studio no reconozca nuestro dispositivo por lo que en algunos casos será necesario instalar los *drivers* del teléfono móvil en el ordenador. Para ello, debemos acudir a la página oficial de la marca del smartphone, buscar nuestro modelo e instalar sus *drivers*.

1.2. Estructura del proyecto

Creado el proyecto el siguiente paso es decidir qué estructura va a tener la aplicación y cómo queremos organizarla. Smart Energy cuenta con veintidós tipos de activitys, es decir, veintidós pantallas diferentes. En el siguiente enlace de github [12] están disponibles los códigos completos tanto en Java como en XML de la aplicación.

A medida que se fue trabajando en el proyecto y condicionados por la disponibilidad de datos abiertos que se encontraban en internet, se fue decidiendo la estructura que iba a tener la aplicación buscando una interfaz sencilla y clara que permitiera cumplir con el objetivo de incorporar métodos para mejorar la eficiencia energética y el medio ambiente de los hogares y de las ciudades y, a su vez, que facilitara la interacción con el usuario permitiendo que toda persona, independientemente de su edad y sus conocimientos, pueda utilizarla con facilidad.

A continuación, se explicarán todas las pantallas programadas por orden de aparición en la app.

1.2.1. Main Activity

Este activity es la pantalla principal, cuando el usuario entra en la app es lo primero que ve, por ello se ha decidido por un diseño sencillo, donde se puede observar el logo de la aplicación, una breve descripción de en qué consiste, un botón de entrar y un botón de información que lleva a una pantalla que explica cómo está estructura la app y que se explica en el apartado 1.2.2. de la presente memoria.

El diseño físico de este del Main Activity es el que se muestra a continuación:

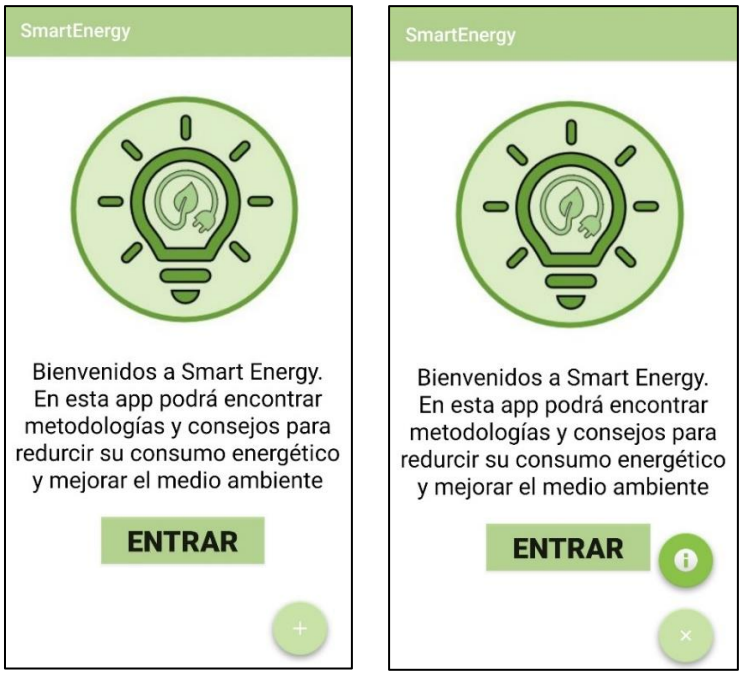


Ilustración 1. MainActivity.

Este activity está compuesto por un ImageView que permite mostrar el icono de la app, por un TextView que enseña la descripción, por un Button que es que da acceso a la aplicación y por último por un menú flotante que es que nos deja acceder a la pantalla de información.

Para poder programar este menú flotante fue necesario implementar en el graddle la siguiente librería:

implementation 'com.getbase:floatingactionbutton:1.10.1'

Sincronizada la librería, mediante el siguiente código en XML se crea el menú flotante y se le indica las características principales, es decir, tamaño, color, logo, ubicación en la pantalla, identificador y el método que tiene asociado en la clase Java que se activa al pulsar el botón:

```
<com.getbase.floatingactionbutton.FloatingActionsMenu
    android:id="@+id/b_informacion"
    android:layout_width="195dp"
    android:layout_height="190dp"
    android:layout_marginStart="270dp"
    android:layout_marginLeft="270dp"
    android:layout_marginBottom="10dp"
    android:clickable="true"
    app:fab_addButtonColorNormal="#C8E2A8"
    app:fab_addButtonColorPressed="@color/colorPrimaryDark"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:srcCompat="@android:drawable/ic_input_add">

    <com.getbase.floatingactionbutton.FloatingActionButton
        android:id="@+id/b_informacion2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="Informacion"
        app:fab_colorNormal="#8BC34A"
        app:fab_icon="@android:drawable/ic_dialog_info"
        app:fab_size="normal"
        app:fab_title="Información" />

</com.getbase.floatingactionbutton.FloatingActionsMenu>
```

Ilustración 2. Código XML botón flotante.

Una vez se tiene el código en XML el siguiente paso es programar el código en Java que es el que hace que los diferentes elementos funcionen. En este activity únicamente tienen una funcionalidad el menú flotante y el botón entrar:

```
//Método floating botón
public void Informacion(View view){
    menuBotones.collapse();
    Intent info = new Intent( packageContext: this, Informacion.class);
    startActivity(info);
}

//Método botón
public void Entrar(View view) {
    Intent ent = new Intent( packageContext: this, Inicio.class);
    startActivity(ent);
}
```

Ilustración 3. Código Java método botones.

En ambos casos por medio de la clase *Intent* se permite que al pulsar el botón de información podamos pasar a la pantalla de Información (apartado 1.2.2.) y que al pulsar el botón entrar vayamos al menú de opciones (apartado 1.2.3.).

1.2.2. Información

El objetivo de este activity es explicar quién diseñó la aplicación, para qué se creó y cómo está estructurada. De esta manera el usuario podrá tener la información necesaria de cada uno de los métodos y poder acceder a aquellos que le interesen según sus necesidades. Se trata de una pantalla meramente informativa que busca aclarar cualquier duda que pueda surgir sobre el diseño de Smart Energy.

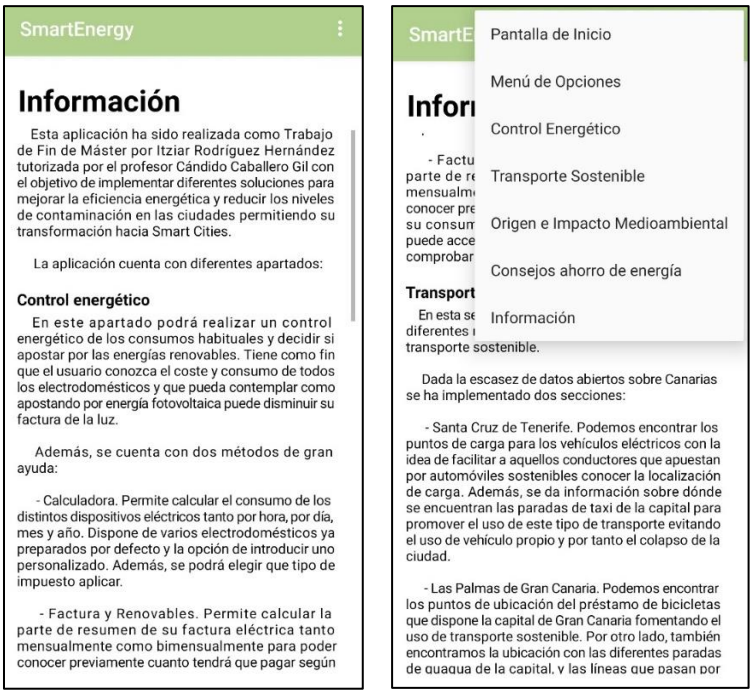


Ilustración 4. Activity Información.

El activity Información está formado por un ScrollView que permite desplazarnos verticalmente por la pantalla, un LinearVertical para introducir los TextView dentro del ScrollView y un ActionBar que es el menú desplegable que se encuentra arriba a la derecha y permite navegar de manera rápida por los diferentes métodos de la app. Además, en esta activity se necesitó la implementación de una librería externa para poder justificar el texto:

implementation 'me.biubiubiu.justifytext:library:1.1'

Ubicado el ScrollView en la pantalla el diseño de este activity consistió en ir colocando dentro de él, usando la librería anterior, los textos que explicaban el funcionamiento de la app.

```
<ScrollView
    android:layout_width="340dp"
    android:layout_height="549dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginBottom="24dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.296"
    app:layout_constraintStart_toStartOf="parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <me.biubiubiu.justifytext.library.JustifyTextView
            android:id="@+id/text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="      Esta aplicación ha sido realizada como Trabajo de Fin de Máster
            por Itziar Rodríguez Hernández tutorizada por el profesor Cándido Caballero Gil con
            el objetivo de implementar diferentes soluciones para mejorar la eficiencia energética
            y reducir los niveles de contaminación en las ciudades permitiendo su transformación
            hacia Smart Cities.\n\n     La aplicación cuenta con diferentes apartados:\n"
            android:textColor="@android:color/black"
            android:textSize="14sp" />
```

Ilustración 5. Código XML ScrollView.

Para poder trabajar con el ActionBar debemos crear en primer lugar un menú overflow. Para ello, nos posicionamos encima de la carpeta res y pulsando botón derecho darle a New y Android Resource File. Hecho esto se nos desplegará la siguiente ventana, donde debemos ponerle nombre al archivo (siempre en minúscula) y elegir en Resource type el tipo Menu:

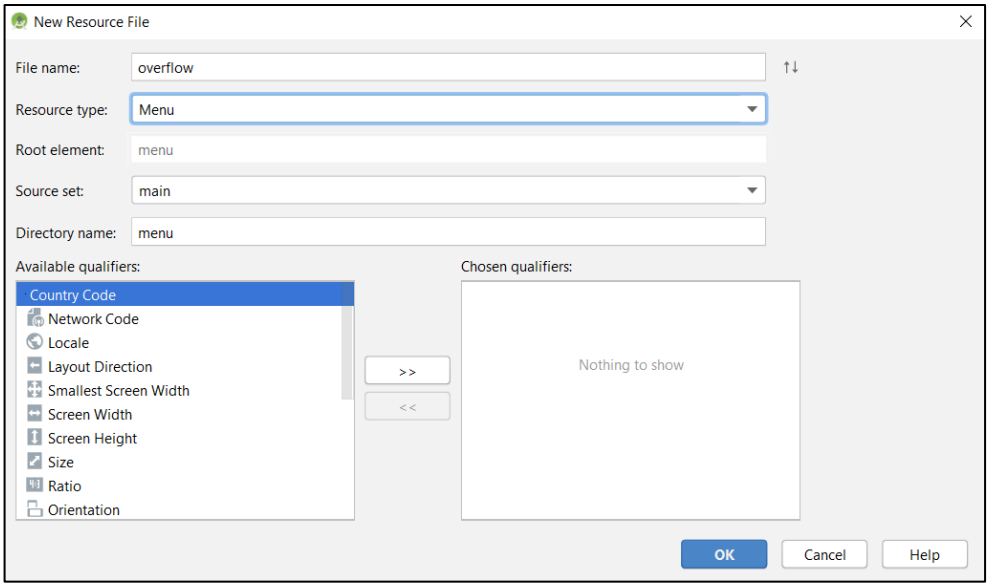


Ilustración 6. Ventana para introducir el menú.

A continuación, se nos creará un nuevo archivo XML donde podremos poner los diferentes ítems a nuestro menú overflow:

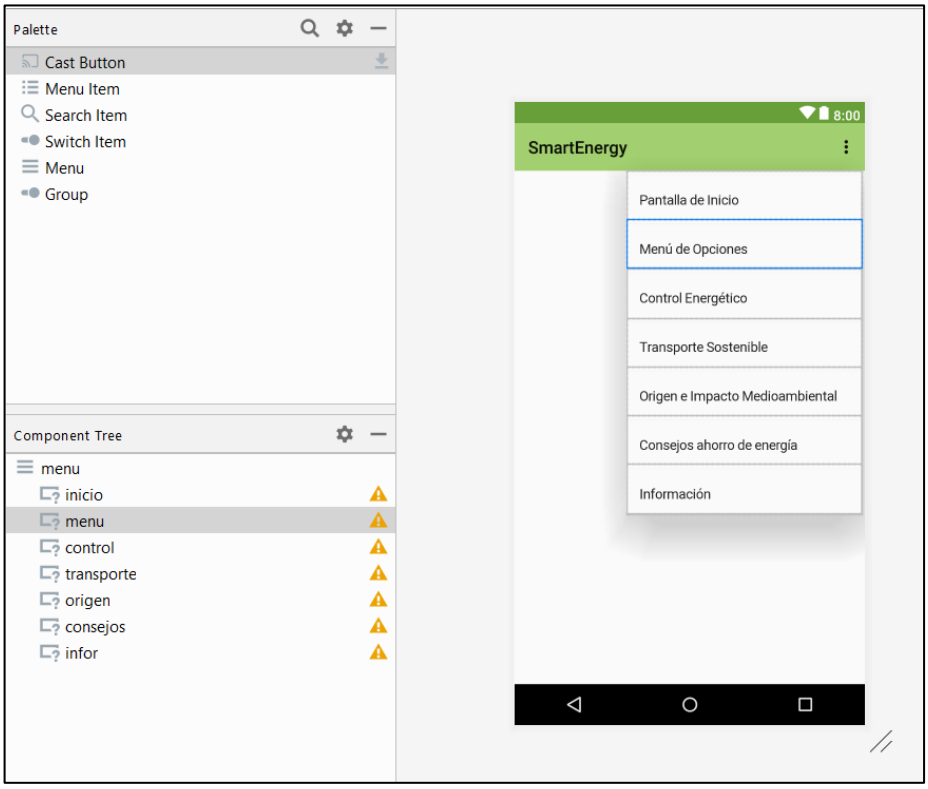


Ilustración 7. Inserción de los items del menú.

Una vez hemos terminado el diseño de nuestro ActionBar pasamos a la parte lógica. Este elemento aparece en prácticamente todas las pantallas de la app, a continuación, se mostrará el del activity Información siendo el del resto igual, únicamente cambiando el Toast que se usa como aviso.

Lo primero que debemos hacer es crear un método que nos permita mostrar el menú overflow, es muy importante que el nombre del método sea onCreateOptionsMenu, en caso contrario no se mostrará. Además, este método debe ser tipo boolean, y debe siempre devolver un valor, en este caso true. Por último, lo que se hace es para la id, del menú previamente creado en el archivo XML para que así se sepa a qué estamos haciendo referencia:

```java
//Metodo mostrar y ocultar menu overflow
public boolean onCreateOptionsMenu(Menu menu){
    getMenuInflater().inflate(R.menu.overflow, menu);
    return true;
}
```

Ilustración 8. Código Java, mostrar y ocultar menú overflow.

El siguiente paso es crear el método para cada uno de los ítems que hemos introducido. En este método se indica que si se pulsa la opción correspondiente al activity en el que se está se enviará un mensaje por medio del Toast indicando que ya se está en esa clase. Por lo contrario, si se pulsa una de las otras opciones, el programa permitirá saltar directamente al activity correspondiente que se ha seleccionado facilitando la navegación del usuario por la app.

```java
//Metodo para asignar las funciones al menu overflow
public boolean onOptionsItemSelected(MenuItem item){
    int id = item.getItemId();

    if (id == R.id.inicio) {
        Intent mprin = new Intent( packageContext: this, MainActivity.class);
        startActivity(mprin);
    }if (id == R.id.menu){
        Intent men = new Intent( packageContext: this, Inicio.class);
        startActivity(men);
    }if (id == R.id.control){
        Intent cont = new Intent( packageContext: this, Control.class);
        startActivity(cont);
    }if (id == R.id.transporte){
        Intent trans = new Intent( packageContext: this, Ciudad.class);
        startActivity(trans);
    }if (id == R.id.origen){
        Intent ori = new Intent( packageContext: this, Origen.class);
        startActivity(ori);
    }if (id == R.id.consejos){
        Intent con = new Intent( packageContext: this, Consejos.class);
        startActivity(con);
    }if (id == R.id.infor) {
        Toast.makeText( context: this,  text: "Ya estás en Información", Toast.LENGTH_SHORT).show();
    }
    return super.onOptionsItemSelected(item);
}
```

Ilustración 9. Código Java navegación por las opciones del menú.

1.2.3. Inicio

El activity Inicio es el primer activity que ve el usuario cuando pasa de la pantalla de presentación. Está compuesto por un dispone de menú de opciones que permite elegir entre las diferentes utilidades que proporciona Smart Energy.

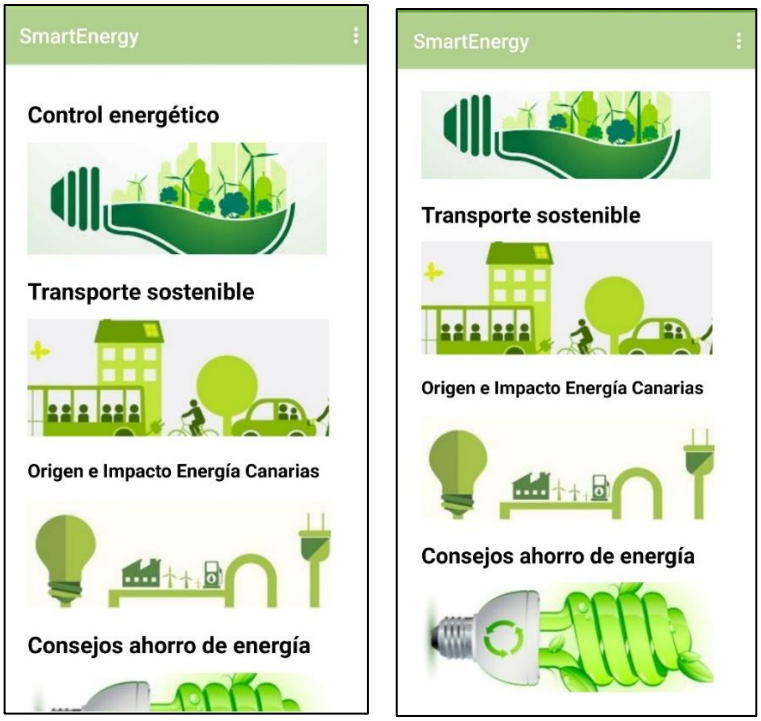


Ilustración 10. Activity Inicio.

Este activity está formado por cuatro TextView que contienen los títulos de las diferentes opciones, y cuatro ImageButton que son botones con imágenes que pulsando sobre ellos permite el acceso al contenido. Además, tiene configurado un ScrollView para navegar verticalmente entre las opciones y el ActionBar comentado anteriormente para saltar entre pantallas.

Mediante el código XML introducimos las imágenes de cada uno de los botones, previamente copiadas en nuestra drawable del directorio res, y le asignamos el método de Java que permite que al pulsar salte a la pantalla correspondiente.

```
<ImageButton
    android:id="@+id/b_ciu"
    android:layout_width="288dp"
    android:layout_height="112dp"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="2dp"
    android:contentDescription="Transporte sostenible"
    android:onClick="Ciudad"
    app:srcCompat="@drawable/transsost" />
```

Ilustración 11. Código XML ImageButton.

```
//Método botón Transporte Sostenible
public void Ciudad(View view) {
    Intent ciu = new Intent( packageContext: this, Ciudad.class);
    startActivity(ciu);
}

//Método botón Consejos Ahorro
public void Consejos(View view) {
    Intent con = new Intent( packageContext: this, Consejos.class);
    startActivity(con);
}
```

Ilustración 12. Código Java métodos ImageButton.

La programación de los ImageButton es la misma que la de los botones normales, es decir, mediante la clase *Intent* se asigna la pantalla a la que se quiere saltar.

1.2.4. Control Energético

Si el cliente decide acceder a la opción de control energético lo primero que se encontrará será una pantalla descriptiva de las diferentes alternativas que se le proporcionan para reducir su consumo energético.

Este activity inicial está formado por tres TextView, uno para el título, otro con preguntas para llamar la atención del cliente y el último explicando las dos opciones que se le ofrecen.

En primer lugar, tiene la alternativa de acceder a la pantalla de Calculadora (apartado 1.2.5.) dónde podrá obtener un valor aproximado de lo que consumen sus electrodomésticos.

La otra opción que se le presenta es la de Factura y Renovables (apartado 1.2.6.) dónde puede elegir entre obtener el resumen de su factura de la luz o bien conocer cuanto ahorraría al mes en su factura si decide apostar por instalar placas fotovoltaicas.

Para la programación del activity Control se han utilizado métodos comentados con anterioridad. Para la parte gráfica los TextView y los Button y para la parte lógica la clase *Intent* para el salto a las otras activitys.

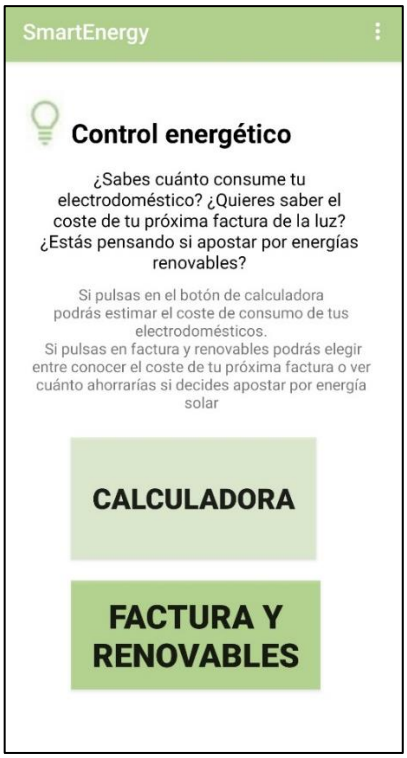


Ilustración 13. Activity Control Energético.

1.2.5. Calculadora

El activity de la Calculadora tiene como objetivo permitir al usuario obtener de manera estimada el consumo de los electrodomésticos que tiene en casa [13]. Esta pantalla cuenta con un diseño sencillo y accesible para todas las personas.

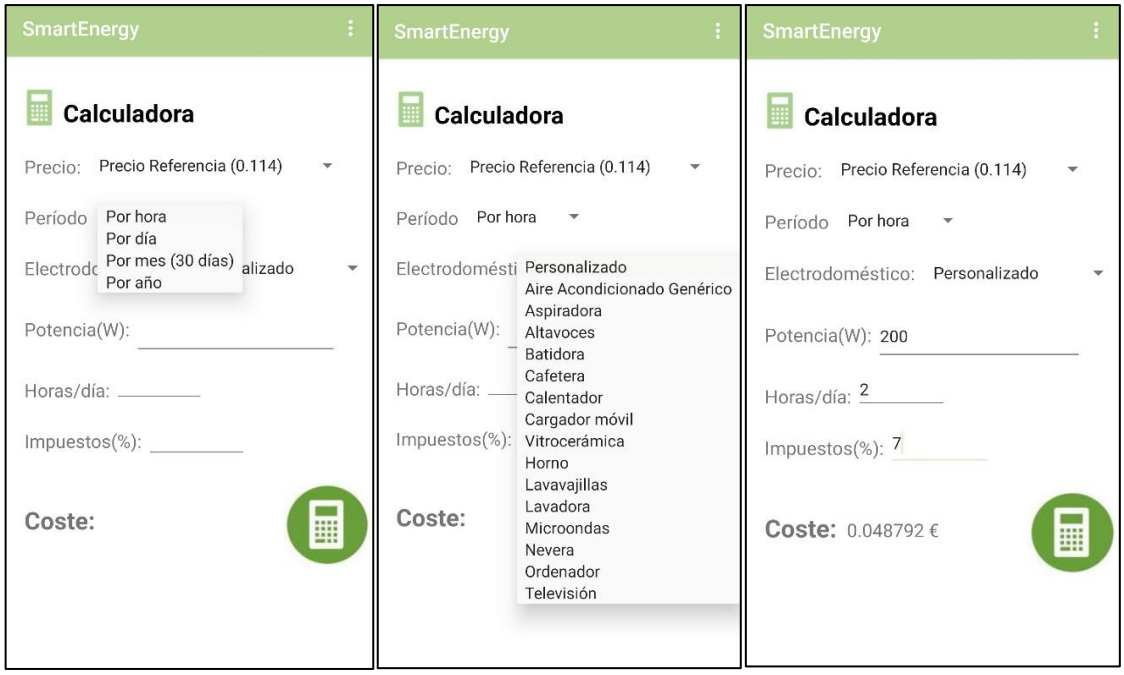


Ilustración 14. Activitys Calculadora.

Lo primero que tenemos es el precio de la energía por kWh que se tomará de referencia (0.114 €/kWh), este precio es una media estimada de las fluctuaciones que se van teniendo a lo largo del año. A continuación, el usuario puede elegir el período de tiempo para el cual quiere conocer el consumo, es decir, si lo quiere calcular por hora, por día, por mes o por año. Por último, se da la opción de elegir entre una serie de electrodomésticos ya prefijados para saber su coste aproximado o optar por la opción personalizado que permite introducir los datos por nosotros mismos. Para esta última opción, se dispone debajo de los datos necesarios que debe introducir el usuario para el cálculo (potencia, horas al día y el impuesto que se aplica).

Para el cálculo del consumo lo que se ha hecho es multiplicar la potencia del electrodoméstico en kW, ya sea la introducida por el usuario o la puesta por defecto, por el precio del kWh, por el número de horas que se esta utilizando. A este valor se le ha sumado el impuesto aplicable.

Para el caso de días, meses o años lo que se ha hecho es pasar es pasar convertir el valor de horas introducidos a la variable de tiempo correspondiente permitiendo así un mayor conocimiento de lo que puede llegar a consumir dicho aparato.

$$Coste\ (€) = \left(potencia\ (kW) * precio\ kWh\ \left(\frac{€}{kWh}\right) * nº\ horas\ (h)\right) +$$

$$\left(potencia\ (kW) * precio\ kWh\ \left(\frac{€}{kWh}\right) * nº\ horas\ (h) * impuestos\right)$$

La programación de la parte gráfica se caracteriza principalmente por la utilización de Spinner que permiten desplegar las distintas opciones que puede elegir el usuario, y por la utilización de EditText que recogen los datos que se introducen.

```
<Spinner
    android:id="@+id/sp_tiempo"
    android:layout_width="129dp"
    android:layout_height="21dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="146dp"
    app:layout_constraintStart_toEndOf="@+id/tv_tiempo"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/et_pot"
    android:layout_width="200dp"
    android:layout_height="52dp"
    android:layout_marginStart="4dp"
    android:layout_marginLeft="4dp"
    android:layout_marginTop="244dp"
    android:ems="10"
    android:inputType="number"
    android:textSize="16sp"
    app:layout_constraintStart_toEndOf="@+id/tv_pot"
    app:layout_constraintTop_toTopOf="parent" />
```

Ilustración 15. Código XML Spinner y EditText.

La parte lógica en este caso ha tenido más trabajo que las hasta ahora vistas, ya que es aquí donde se realiza todos los cálculos necesarios para poder obtener el coste de consumo y dónde realmente se programa el funcionamiento del activity.

En primer lugar, es muy importante la parte de programación de los spinner ya que dependiendo de la elección que haga el usuario se tendrá que calcular una cosa u otra. Lo que se hace primero es declararlo como se muestra y por medio del método *getSelectedItem* obtenemos la opción que se ha elegido.

```
String [] tiempo = {"Por hora", "Por día", "Por mes (30 días)", "Por año"};
ArrayAdapter<String> adapter2 = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_item, tiempo);
sp_tiempo.setAdapter(adapter2);
```

Ilustración 16. Código Java declaración Spinner

Una vez tenemos la opción mediante estructuras condicionales programamos los cálculos que se debe hacer para cada uno de los casos y mostramos el resultado final.

```
if (sel_prec.equals("Precio Referencia (0.114)") && sel_horas.equals("Por hora")) {
    double precio = 0.114;
    if (sel_pot.equals("Personalizado")) {
        if (potencia.equals("") || impuestos.equals("") || tiempo.equals("")) {
            Toast.makeText( context: this, text: "Debes rellenar todos los campos", Toast.LENGTH_SHORT).show();
        } else {
            String valor_pot = et_pot.getText().toString();
            String valor_horas = et_horas.getText().toString();
            String valor_imp = et_imp.getText().toString();

            int pot_int = (Integer.parseInt(valor_pot));
            int hor_int = Integer.parseInt(valor_horas);
            int imp_int = (Integer.parseInt(valor_imp));

            double pot = ((double) pot_int) / 1000;
            double imp = ((double) imp_int) / 100;
            double hor = (double) hor_int;

            double coste = ((pot) * hor * (imp) * precio) + ((pot) * hor * precio);
            float importe = (float) coste;
            String resultado = String.valueOf(importe);
            tv_coste.setText(resultado + " €");
        }
```

Ilustración 17. Código Java, cálculo consumo.

1.2.6. Opciones Consumo

La pantalla opciones de consumo es una introducción a los dos tipos de instalación que disponemos. Es un activity que permite al usuario conocer de manera aproximada el resumen de su factura para una instalación normal y poder ver de manera estimada cuánto dinero ahorraría al mes si decide instalar paneles fotovoltaicos.

Opciones de Consumo está formada por tres TextView que muestran el título, preguntas para captar la atención y una breve descripción de las dos opciones y por dos botones que hacen posible acceder a cada una de las opciones comentadas.

Ilustración 18. Activity Opciones de Consumo.

1.2.7. Factura

Si el usuario opta por calcular el consumo habitual le aparecerán un conjunto de activitys que tendrá que ir rellenando para poder obtener de manera estimada el resumen de su factura de la luz.

El objetivo de este método es que el cliente pueda tener una idea aproximada de cuanto tendrá que pagar en su próxima factura. También le puede servir para ver cuánto puede ahorrar si reduce la potencia consumida motivándole a cambiar su rutina para intentar hacer un uso responsable de la electricidad.

En este caso tiene la opción de obtener el valor mensual o bimensual y se le proporciona un calendario (DatePicker) para que pueda seleccionar la fecha de su factura.

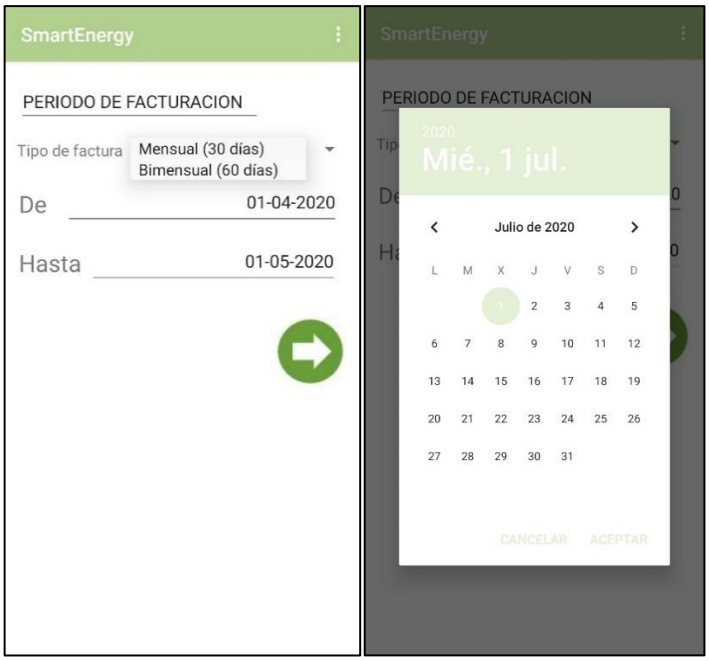


Ilustración 19. Activitys Factura - Períodos de Facturación.

A continuación, pulsando sobre la flecha verde pasará al siguiente activity dónde se le preguntará que potencia tiene contratada en kW y que impuesto quiere que se le aplique, si el Impuesto sobre el Valor Añadido (IVA) o el Impuesto General Indirecto Canario (IGIC).

Ilustración 20. Activitys Factura - Condiciones de suministro.

Introducido estos datos le dará a siguiente y tendrá que indicar cual es la potencia consumida en kWh, y, por último, dándole a siguiente rellenar cuánto paga por el alquiler de equipos.

Ilustración 21. Activitys Factura - Consumo y Otros.

En caso de que no se rellene algún valor, no se permitirá pasar a la siguiente pantalla y se mostrará un mensaje indicando que dato le falta por completar.

Ilustración 22. Activity Factura - Mensaje de seguridad.

Una vez rellenado todos los campos, el cliente le dará al botón de calcular y pasará al último activity dónde se le mostrará el resumen estimado de su factura.

Ilustración 23. Activitys Factura Final.

Muchos de los elementos empleados en todas las pantallas hemos visto ya como funciona anteriormente y se puede encontrar el código en el enlace de github [12], sin embargo, en la parte gráfica hay un elemento que aún no habíamos utilizado que es el DatePicker, su programación en XML se hace mediante un TextView y es en la parte lógica con Java dónde se le asigna el funcionamiento.

```
eText.setInputType(InputType.TYPE_NULL);
eText.setOnClickListener((v) → {
        final Calendar cldr = Calendar.getInstance();
        final int day = cldr.get(Calendar.DAY_OF_MONTH);
        final int month = cldr.get(Calendar.MONTH);
        int year = cldr.get(Calendar.YEAR);
        // date picker dialog
        picker = new DatePickerDialog( context: Factura.this,
                (view, year, monthOfYear, dayOfMonth) → {
                        String fecha="-" + year;
                        if (monthOfYear>8) {
                            fecha = "-" +(monthOfYear + 1)  + fecha;
                        }else{
                            fecha= "-" + "0" +  (monthOfYear + 1) + fecha;
                        }if (dayOfMonth<10) {
                            fecha =  "0" + (dayOfMonth)+ fecha ;
                        }else {
                            fecha = + dayOfMonth+ fecha ;
                        }
                        eText.setText(fecha);
                }, year, month, day);
        picker.show();
});
```

Ilustración 24. Código Java DatePicker.

En cuanto al procedimiento para calcular el resumen de la factura, lo que se ha hecho es ir guardando en diferentes variables el contenido de los datos suministrados por el usuario para poder trabajar con ellos. Las fórmulas para cada uno de los valores calculados son las que se explican en el apartado 3.3. del capítulo 2 de la presente memoria, en concreto en el apartado *4. Resumen de la fatura y datos*, y son las siguientes:

$$Potencia\ Contratada = kW\ contratados * precio\ kW * dias\ de\ facturación$$

$$Energía\ Consumida = precio\ kWh *\ kW\ cosumidos$$

$$Impuesto\ de\ electricidad = (Término\ de\ potencia + Energía\ cosumida) * 0.05113$$

$$Impuesto\ aplicado = (Término\ de\ potencia + Energía\ cosumida + Alquiler\ de\ equipos + Impuesto\ sobre\ la\ electricidad) * 0.07$$

$$Total\ Importe\ Factura = (Término\ de\ potencia + Energía\ cosumida + Alquiler\ de\ equipos + Impuesto\ sobre\ la\ electricidad + Impuesto\ aplicado)$$

En el código Java se utiliza una estructura condicional que calcula el total de la factura en función de que se haya elegido mensual o bimensual, e IVA o IGIC.

```
if(elecR.equals("0") && elecDR.equals("1")){
    double potencia = Double.parseDouble(datPotR);
    double potCont = potencia * precio *30; //* numDias
    float potCont2 = (float) potCont;
    potContrata = String.valueOf(potCont2);
    pot_cont.setText(potContrata + " €");

    double energia = Double.parseDouble(datP1R);
    double consumo = energia * 0.083972;
    float consumo2 = (float) consumo;
    enConsumida = String.valueOf(consumo2);
    energ_cons.setText(enConsumida + " €");

    double impuesto = (potCont + consumo)*0.05113;
    float impuesto2 = (float) impuesto;
    impuestoElec = String.valueOf(impuesto2);
    imp_elec.setText(impuestoElec + " €");
    double igic = (potCont + consumo + equipos + impuesto)*0.07;
    float igic2 = (float) igic;
    double totalfact = potCont + consumo + equipos + impuesto + igic;
    float totalfact2 = (float) totalfact;
    String igicfact = String.valueOf(igic2);
    String totalfactu = String.valueOf(totalfact2);
    total.setText(totalfactu + " €");
    igicR.setText(igicfact + " €");
    imp.setText("IGIC normal (7%)");
    mens.setText("(Mensual)");
}
```

Ilustración 25. Código Java, cálculo de factura.

1.2.8. Paneles

El objetivo de este conjunto de activitys es que el cliente pueda tener una idea aproximada de cuánto puede ahorrar mensualmente en su factura de la luz si decide apostar por el uso de las energías renovables. Se busca intentar concienciar a los usuarios por medio de ese incentivo económico para que apueste por energías limpias que supondrían menores emisiones para el planeta, permitiendo el desarrollo de ciudades más sostenibles y una mayor implicación por parte de los gobiernos para promover este tipo de electricidad.

La primera pantalla que nos encontramos se trata de la solicitud de una serie de datos que el cliente debe introducir para poder conocer su situación actual y poder obtener la estimación de los ahorros. Estos son:

- La potencia que tiene contratada en kW.
- El consumo mensual que suele tener en kWh.
- Las Horas de Sol Pico (HSP) de la zona en dónde se hará la instalación.
- Coste aproximado de su factura mensual habitual.

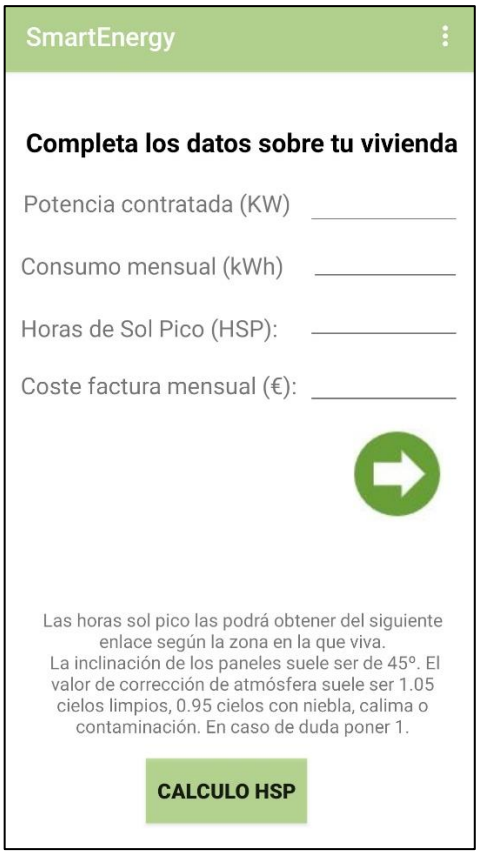


Ilustración 26. Activity Instalación Autoconsumo.

Dado que el dato de HSP es más complejo de obtener se ha introducido un botón que da acceso a una página web [14] dónde se puede obtener este valor de manera sencilla para el lugar dónde se quieren instalar las placas.

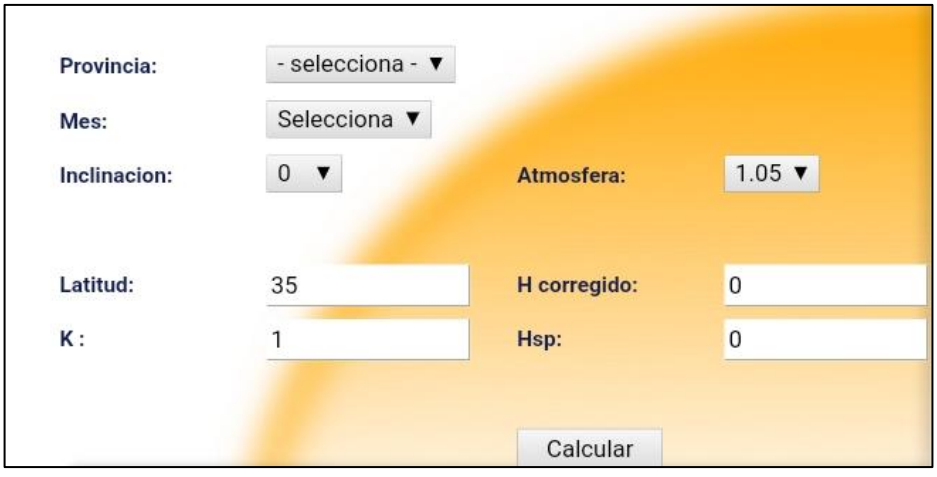


Ilustración 27. Página web obtención HSP.

Una vez se introducen todos los datos solicitados se pulsa en el botón de calcular y se avanza a una nueva pantalla que muestra los valores más relevantes como el número de paneles, la

superficie que ocuparía, el ahorro anual estimado y, finalmente, de manera resaltada, el ahorro mensual que se obtendría.

Ilustración 28. Activity Ahorro Mensual.

Además, esta activity cuenta con un menú flotante que da acceso a una pantalla de información que explica detalladamente como se han hecho los cálculos y qué criterios se han tenido en cuenta dado que el resultado obtenido es una estimación para que el cliente pueda tener una idea aproximada del ahorro que podría alcanzar. Esos criterios son los que se explican a continuación.

Se han considerado tres supuestos según la potencia demandada por el sistema en:

- Menos de 1500 W.
- Entre 1500W y 5000W.
- Más de 5000W.

A partir de los valores que introduce el cliente se hace un cálculo estimado de lo que podría ahorrar al mes mediante los siguientes pasos [15]:

Cálculo de la potencia prevista en kW

El dato de potencia contratada suministrado por el usuario se multiplica por un factor de simultaneidad de 0.7 y obtener así la potencia prevista.

$$Pprevista\ (kW) = Pcontratada * 0.7$$

Cálculo del consumo anual (kWh/año)

El dato de consumo mensual se multiplica por 12 para tener una estimación del consumo anual que tiene el cliente y se multiplica por un factor de seguridad de 1.33 para asegurarnos que no nos quedaremos cortos con el número de paneles.

$$Consumo\ Anual\ (kWh/año) = (Consumo\ Mensual * 12) * 1.33$$

Elección de la tensión de la instalación de corriente continua.

Este valor se ha establecido de manera generalizada según la potencia demandada por el sistema. Si la instalación demanda una potencia menor a 1500W se cogerá una tensión de 12V, si el valor de potencia está entre 1500W y 5000W la tensión será de 48V y para el caso de más de 5000W el valor será de 300V.

Cálculo del número de paneles.

En primer lugar, es necesario calcular la energía diaria que puede producir un solo panel. Para ello es necesario saber la potencia máxima del panel, las horas de sol pico y el rendimiento de trabajo del panel. Las HSP son introducidas por el usuario, el rendimiento será considerado del 90% y los valores de potencia máxima vienen en el datasheet del panel disponible en los anexos de la presente memoria, siendo la potencia máxima 330W para instalaciones menores a 1500W, 385W para instalaciones entre 1500W y 5000W y 405W para las de más de 5000W.

Una vez tenemos la energía diaria que produce un solo panel, podemos obtener el número de placas necesarias, dividiendo la energía que se consume diariamente por la energía que suministra una placa.

$$Eplaca\ (kWh) = PotenciaPlaca * HSP * 0.9$$

$$Nº\ de\ placas = Consumo\ diario\ /\ Eplaca$$

Superficie de la instalación.

Para obtener la superficie que ocupara la instalación de las placas, lo que se ha hecho es considerar que todos se colocan en serie y se multiplica sus dimensiones por el número de paneles.

$$Superficie\ instalación\ (m^2) = nº\ de\ placas * superficie\ de\ las\ placas$$

Estimación producción energía anual

Este valor se ha obtenido considerando únicamente las pérdidas por radiación, a la hora de hacer un estudio más exhaustivo es necesario tener otras pérdidas en cuenta. En nuestro caso, al tratarse de una estimación hemos calculado el valor multiplicando la eficiencia de las placas que viene dada en el datasheet, por la superficie de la instalación por un valor de radiación medio anual.

$$Producción\ anual\ (kWh) = EficienciaPlacas * SuperficieInst * Radiación$$

Ahorro anual estimado [16]

En primer lugar, calculamos la capacidad del sistema, para ello multiplicamos la potencia máxima del panel por la superficie de la instalación por un valor de un 85% que es la capacidad en corriente alterna que se estima que tiene un sistema para viviendas.

$$Capacidad(kW) = PotPlaca * Superficie * 0.85$$

Con este valor de capacidad, considerando un valor medio aproximado del coste de la energía y teniendo en cuenta las HSP podemos calcular el ahorro anual que supondría instalar las placas:

$$Ahorro\ anual\ (€/año) = Capacidad\ HSP * precio\ energía$$

Ahorro mensual

Finalmente, el ahorro mensual en euros que tendría el cliente si decide instalar placas fotovoltaicas se ha calculado dividiendo el ahorro anual entre los doce meses:

$$Ahorro\ mensual\ (€/mes) = Ahorro\ anual\ / 12$$

Teniendo en cuenta todos estos pasos y consideraciones se llevo a cabo la programación en lenguaje Java que permitiera realizar todos estos cálculos y cumplir con el objetivo de concienciación de ahorro de energía.

```
if((1.5 < pPrevista2) && (pPrevista2 <= 5.0)){
    tfotovoltaico.setText("48");
    double horas = Double.parseDouble(horasSol);
    double Ep = 385 * horas * 0.9;
    double numero = (consumodiario*1000) / (Ep);
    float npanel = (float) numero;
    npanel = Math.round(npanel);
    numeroPaneles = String.valueOf(npanel);
    npaneles.setText(numeroPaneles);
    float pPico = (385* npanel)/1000;
    potenciaPico = String.valueOf(pPico);
    ppico.setText(potenciaPico);
    double superf = numero *(0.99314*1.999);
    float sup = (float) superf;
    sup = Math.round(sup);
    superficie = String.valueOf(sup);
    sup_instalacion.setText(superficie);
    double estimac = (0.1941 * 4.8 * sup)*365;
    float est = (float) estimac;
    estimacionProduc = String.valueOf(est);
    est_produc.setText(estimacionProduc);
    double ahorro1 = 0.385*sup*0.85;
    double produccion = horas*365*ahorro1;
    double ahorro2 = produccion * precio;
    float ahorro3 = (float) ahorro2;
    ahorro3 = Math.round(ahorro3);
    ahorroEst = String.valueOf(ahorro2);
    ah_est.setText(ahorroEst);
    double consumoSP = Double.parseDouble(consuSP);
    float consuCP = ahorro3 /12;
    float cSP = (float) consumoSP;
    float consumoCP = cSP - consuCP;
    consumoCP = Math.round(consumoCP);
    consumoCPan = String.valueOf(consumoCP);
    cons_cp.setText(consumoCPan  + " €");
    float ahorroTotal = cSP - consumoCP;
    ahorroTotal = Math.round(ahorroTotal);
    ahTotal = String.valueOf(ahorroTotal);
    ah_mes.setText(ahTotal  + " €");
}
```

Ilustración 29. Código Java, ahorro mensual.

1.2.9. Ciudad

Si el usuario decide pinchar sobre Transporte Sostenible se encontrará con el activity Ciudad que es muy similar al Menú Opciones. En esta pantalla se le muestra al cliente la opción de ver en un mapa dónde se encuentran ubicados diferentes medios de transporte públicos y urbanos, así como los préstamos de bicicletas o puntos de cargas de vehículos eléctricos. Para ello debe pulsar en el ImageButton que le interesa y se pasará al MapsActivity.

El objetivo de este activity es promover dentro de las ciudades el uso de medios de transportes menos contaminantes para reducir así las emisiones de gases a la atmósfera y cumplir con los parámetros de sostenibilidad. Se busca cambiar la conciencia de la gente proporcionando en una misma aplicación la opción de poder elegir que medio de transporte le viene mejor y disponer de toda la información con un click.

Es preciso aclarar que en Canarias solo fue posible encontrar datos abiertos de las ciudades de Santa Cruz de Tenerife y de Las Palmas de Gran Canaria por lo que únicamente se ha trabajado con información de ambas capitales.

Ilustración 30. Activity Transporte Sostenible.

1.2.10. MapsActivity

El MapsActivty es la pantalla que muestra la ubicación del punto de carga o transporte seleccionado y nos permite conectar con Google Maps para saber cómo llegar al lugar.

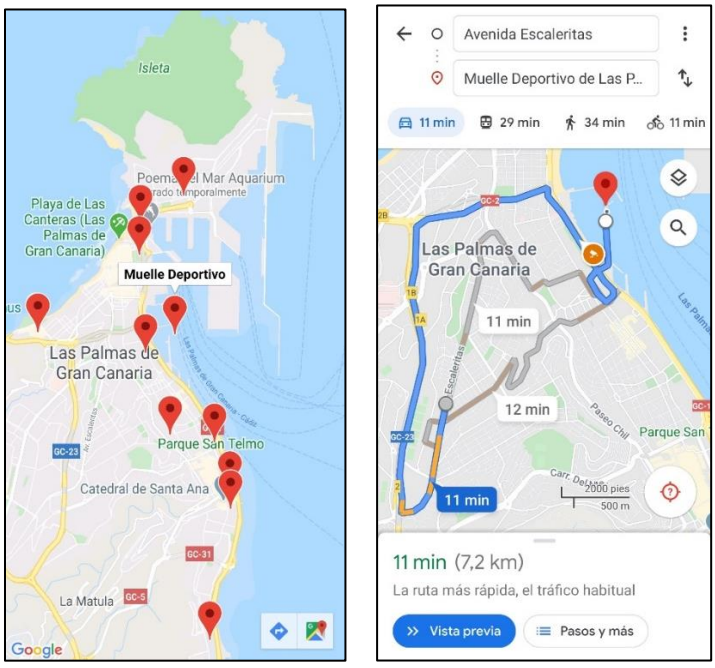


Ilustración 31. Activity MapsActivity.

Los activitys que trabajan con mapas son diferentes al resto por lo que para utilizarnos es necesario llevar a cabo un conjunto de pasos previos.

En primer lugar, tenemos que activar en el SDK Manager, en el apartado SDK Tools, los servicios de Google Play, si no los activamos el mapa no funcionará.

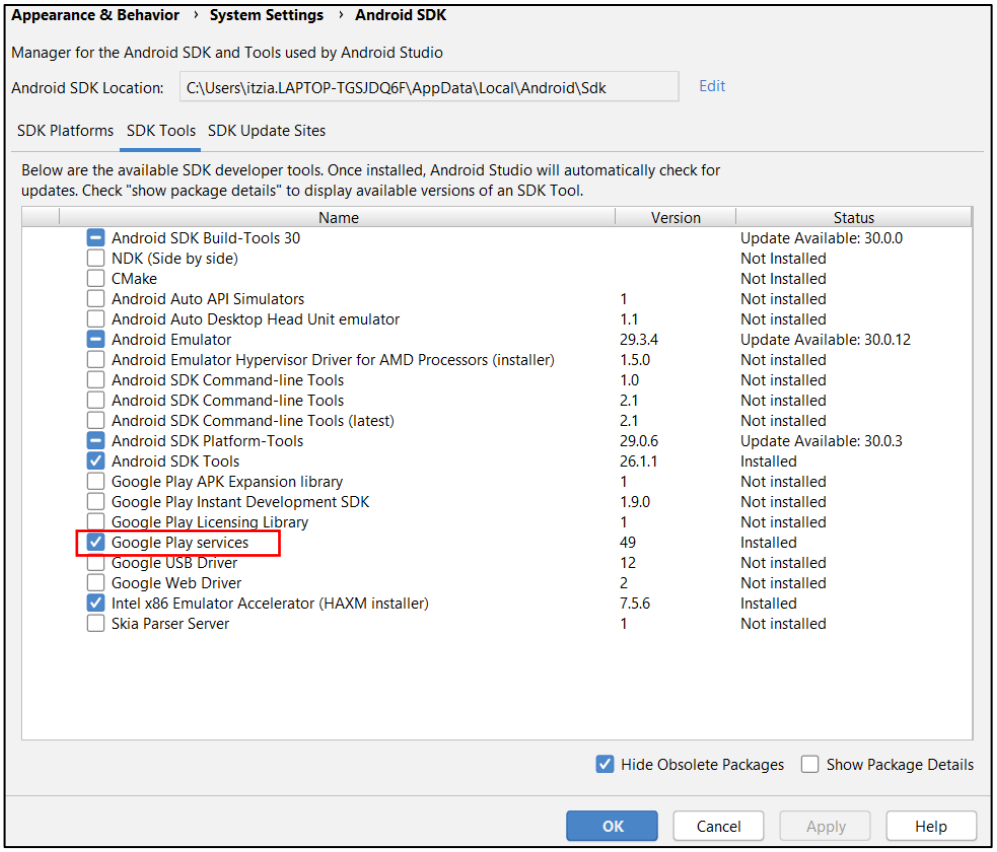


Ilustración 32. Menú para activar distintas SDK Tools.

Cuando hemos comprobado que los servicios de Google Play están activados lo que tenemos que hacer es crear una nueva pantalla con la plantilla ya prediseñada denominada Google Maps Activity. Este activity traerá asignado un link propio para este proyecto en el que es necesario entrar y registrar la app para que se nos cree la clave del API y poder representar los mapas.

Siguiendo los pasos anteriores ya nuestra pantalla nos mostrará el mapa, únicamente nos quedaría programar la ubicación de los puntos de interés en el archivo Java mediante el siguiente código:

```
// Añade un marcador en las etiquetas y mueve la camara
LatLng ayuntSC = new LatLng( v: 28.469811215184006, v1: -16.254823278835648);
mMap.addMarker(new MarkerOptions().position(ayuntSC).title("Ayuntamiento de Santa Cruz de Tenerife"));
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(ayuntSC, v: 15));
LatLng polSC = new LatLng( v: 28.45819734029621, v1: -16.25983839792717);
mMap.addMarker(new MarkerOptions().position(polSC).title("Policia Local de S/C de Tenerife"));
```

Ilustración 33. Código Java, añadir puntos en el mapa.

1.2.11. Origen e Impacto Energía en Canarias

Esta opción tiene como objetivo mostrar a los usuarios el origen que tiene la energía en Canarias y el impacto medioambiental que ocasionan.

Para ello se han creado varios activitys que se conectan directamente por medio de JSON (acrónimo de JavaScript Object Notation, formato de texto ligero para el intercambio de datos) y Volley (planificador de peticiones a la red), con los datos en tiempo real proporcionados por REE [17] analizándolos y dándoles un formato adecuado para representarlos de manera sencilla.

Cuando un usuario decide pinchar en esta opción lo primero que encuentra es una pantalla explicativa con las islas cómo botones dónde tiene la posibilidad de elegir entre ver el origen de la energía o el impacto medioambiental que producen.

Ilustración 34. Activity Origen e Impacto Medioambiental.

Como se puede apreciar esta pantalla está formada por dos TextView para el título y la explicación, un spinner para elegir entre ambas opciones y siete ImageButton para seleccionar cada una de las islas.

Si el usuario marca la opción origen de la energía y pulsa en alguna de las islas se le dirigirá a una pantalla dónde podrá seleccionar una fecha de interés y ver en una gráfica la cantidad de

energía demandada por esa isla. Además, se le ofrece la opción de elegir diferentes tipos de energías para conocer qué cantidad corresponde a cada una de ellas:

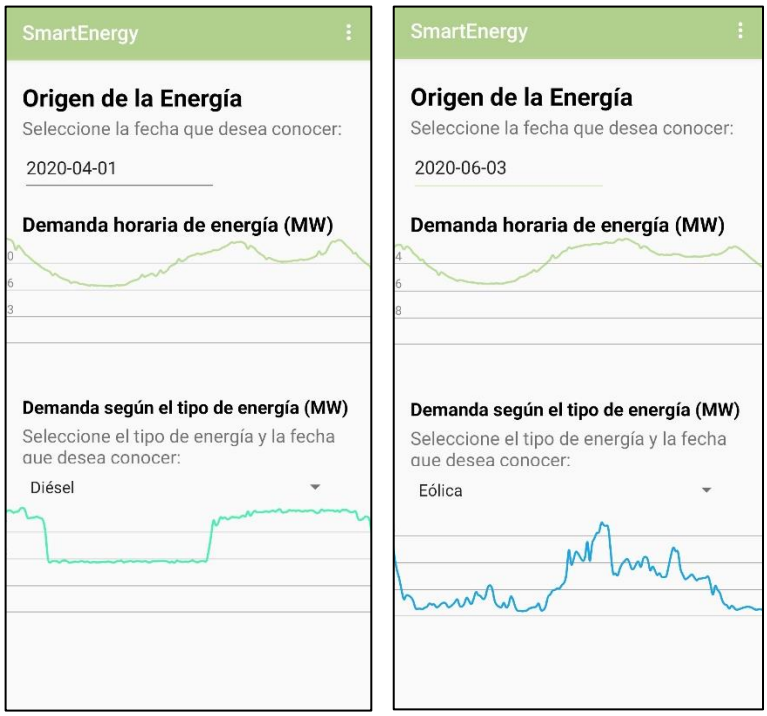


Ilustración 35. Activity Origen Energía.

Esta pantalla está formada por cinco TextView, un DatePicker, un Spinner y dos gráficas lineales. Para poder añadir las gráficas fue necesario implementar la siguiente librería:

implementation 'com.github.blackfizz:eazegraph:1.2.5l@aar'

De esta manera se conseguía usarlas en nuestro código XML.

```
<org.eazegraph.lib.charts.ValueLineChart
    android:id="@+id/mcubiclinechart3"
    android:layout_width="408dp"
    android:layout_height="146dp"
    android:layout_marginTop="174dp"
    app:egCurveSmoothness="0.4"
    app:egLegendHeight="40dp"
    app:egUseCubic="true"
    app:egUseOverlapFill="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.333"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Ilustración 36. Código XML gráfica de líneas.

Para la parte lógica se tuvieron que tomar también una serie de consideraciones previas. En primer lugar, se tuvo que añadir en el Manifest los permisos de internet para poder conectarse con la web de REE y trabajar con los datos, y también implantar una librería que nos permitiera trabajar con Volley para realizar las conexiones:

uses-permission android:name="android.permission.INTERNET"

implementation 'com.android.volley:volley:1.1.1'

Llevados a cabo estos pasos pasamos a la programación en Java del código necesario para acceder a los datos y prepararlos para poder representarlos.

```
public void Demanda (String url1){
    final TextView mTextView = (TextView) findViewById(R.id.editText);
    //Cola para conexiones con Volley
    RequestQueue queue = Volley.newRequestQueue( context: this);
    //Permitimos conexión segura cifrada (htts)
    HttpsTrustManager.allowAllSSL();
    //Solicitamos una respuesta (string) desde la URL indicada
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url1,
            (response) → {
                    String seleccion = spinner.getSelectedItem().toString();
                    ValueLineChart mCubicValueLineChart = (ValueLineChart) findViewById(R.id.mcubiclinechart3);
                    ValueLineChart mCubicValueLineChart2 = (ValueLineChart) findViewById(R.id.mcubiclinechart4);
                    ValueLineSeries series = new ValueLineSeries();
                    ValueLineSeries series2 = new ValueLineSeries();
                    try {
                        //creas un objeto jsonobject a partir de la respuesta json
                        JSONObject jsonobject = new JSONObject(response.substring(21, response.length() - 2));

                        // Almacenamos todos los valores de valoresHorariosGeneracion en un array
                        JSONArray jsonArray = jsonobject.getJSONArray( name: "valoresHorariosGeneracion");
                        for (int i = 0; i < jsonArray.length(); i++) {
                            JSONObject explrObject = jsonArray.getJSONObject(i);
                            String fechas = explrObject.getString( name: "ts");
                            String dem = explrObject.getString( name: "dem");
                            series.setColor(0X8A8BC34A);
                            series.addPoint(new ValueLinePoint(fechas, Float.parseFloat(dem)));
                            mCubicValueLineChart.clearChart();
                            mCubicValueLineChart.addSeries(series);
                            mCubicValueLineChart.startAnimation();

                            String die = explrObject.getString( name: "die");
                            String gas = explrObject.getString( name: "gas");
                            String eol = explrObject.getString( name: "eol");
                            String ele = explrObject.getString( name: "ele");
                            String vap = explrObject.getString( name: "vap");
                            String fot = explrObject.getString( name: "fot");
                            String cc = explrObject.getString( name: "cc");
                            String hid = explrObject.getString( name: "hid");

                           if (seleccion.equals("Diésel")) {
                               series2.setColor(0XFF45F4AC);
                               series2.addPoint(new ValueLinePoint(fechas, Float.parseFloat(die)));
                           }
                           if (seleccion.equals("Turbina de gas")) {
                               series2.setColor(0XFF56B7F1);
                               series2.addPoint(new ValueLinePoint(fechas, Float.parseFloat(gas)));
                           }
                            mCubicValueLineChart2.clearChart();
                            mCubicValueLineChart2.addSeries(series2);
                            mCubicValueLineChart2.startAnimation();
                        }
                    }catch (JSONException e) {
                        e.printStackTrace();
                        mTextView.setText("error: " + e.getMessage());
                    }
            }, (error) → {
            mTextView.setText("That didn't work!: " + error.getMessage());
    });
    queue.add(stringRequest);
}
```

Ilustración 37. Código Java, obtención y preparación de datos.

Si el usuario decide seleccionar la opción de impacto de medioambiental lo que verá es una pantalla similar a la anterior pero que muestra datos distintos. En este caso se encontrará un gráfico de barra que representa las emisiones de CO2 asociadas a los distintos tipos de energía, y un gráfico circular que muestra la relación de energía renovable vs no renovable que se ha utilizado para producir energía el día seleccionado.

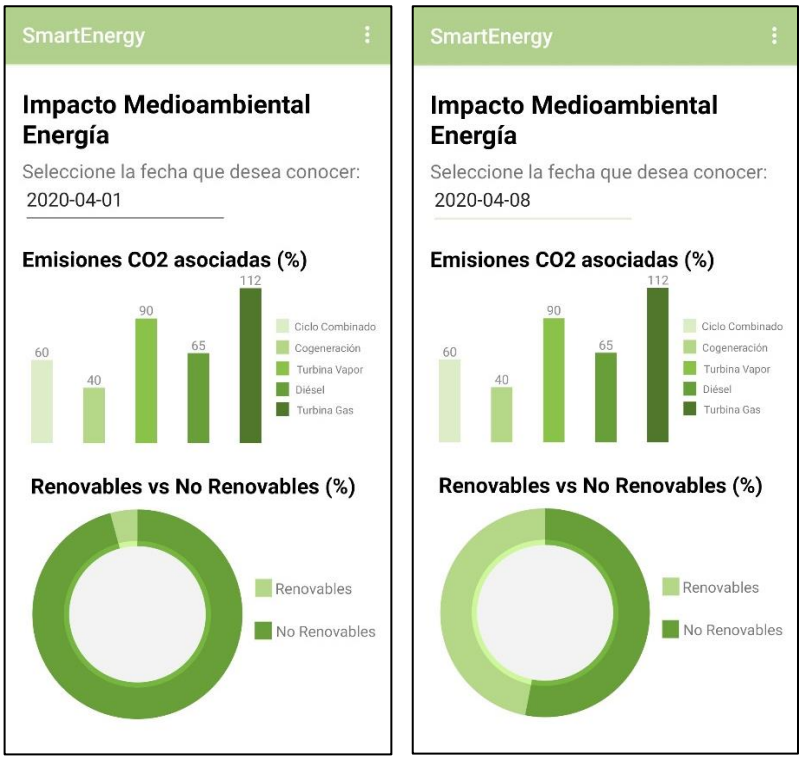


Ilustración 38. Activity Impacto Medioambiental.

La parte programación del XML como del Java es muy similar a la comentada para el origen de la energía, las diferencias se encuentran en el tipo de gráfico introducido y en cómo se han preparado los datos para su representación.

```
<org.eazegraph.lib.charts.BarChart
    android:id="@+id/barchart"
    android:layout_width="266dp"
    android:layout_height="180dp"
    android:layout_marginTop="184dp"
    android:padding="10dp"
    app:egBarWidth="20dp"
    app:egFixedBarWidth="true"
    app:egLegendHeight="0dp"
    app:egShowDecimal="true"
    app:egShowValues="true"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<org.eazegraph.lib.charts.PieChart
    android:id="@+id/piechart"
    android:layout_width="247dp"
    android:layout_height="216dp"
    android:layout_marginTop="406dp"
    android:padding="10dp"
    app:egLegendTextSize="18sp"
    app:egOpenClockwise="false"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="MissingConstraints" />
```

Ilustración 39. Código XML, gráficos de barras y circular.

```
BarChart mBarChart = (BarChart) findViewById(R.id.barchart);
try {
    //creas un objeto jsonobject a partir de la respuesta json
    JSONObject jsonobject2 = new JSONObject(response.substring(21, response.length() - 2));
    JSONObject jsonobject3 = new JSONObject("{\"FACTORCO2\":["+jsonobject2+"]}");
    // Almacenamos todos los valores de valoresHorariosGeneracion en un array
    JSONArray jsonArray2 = jsonobject3.getJSONArray( name: "FACTORCO2");
    JSONObject explrObject2 = jsonArray2.getJSONObject( index: 0);

    float factorEmisionCO2_cc = 0.0f;
    float factorEmisionCO2_cogen = 0.0f;
    float factorEmisionCO2_vap = 0.0f;
    float factorEmisionCO2_die = 0.0f;
    float factorEmisionCO2_gas = 0.0f;

    mBarChart.clearChart();
    factorEmisionCO2_cc = (Float.parseFloat(explrObject2.getString( name: "factorEmisionCO2_cc")))*100;
    factorEmisionCO2_cogen = (Float.parseFloat(explrObject2.getString( name: "factorEmisionCO2_cogen")))*100;
    factorEmisionCO2_vap = (Float.parseFloat(explrObject2.getString( name: "factorEmisionCO2_vap")))*100;
    factorEmisionCO2_die = (Float.parseFloat(explrObject2.getString( name: "factorEmisionCO2_die")))*100;
    factorEmisionCO2_gas = (Float.parseFloat(explrObject2.getString( name: "factorEmisionCO2_gas")))*100;

    mBarChart.addBar(new BarModel( _legendLabel: "cc", factorEmisionCO2_cc,  _color: 0XFFDCEDC8));
    mBarChart.addBar(new BarModel( _legendLabel: "cogen", factorEmisionCO2_cogen,  _color: 0XFFB4D888));
    mBarChart.addBar(new BarModel( _legendLabel: "vap", factorEmisionCO2_vap,  _color: 0XFF8BC34A));
    mBarChart.addBar(new BarModel( _legendLabel: "die", factorEmisionCO2_die,  _color: 0XFF689F38));
    mBarChart.addBar(new BarModel( _legendLabel: "gas", factorEmisionCO2_gas,  _color: 0XFF4F782A));
    mBarChart.startAnimation();
```

Ilustración 40. Código Java gráfico de barras.

```
PieChart mPieChart = (PieChart) findViewById(R.id.piechart);
try {
    //creas un objeto jsonobject a partir de la respuesta json
    JSONObject jsonobject = new JSONObject(response.substring(21, response.length() - 2));

    // Almacenamos todos los valores de valoresHorariosGeneracion en un array
    JSONArray jsonArray = jsonobject.getJSONArray( name: "valoresHorariosGeneracion");
    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject explrObject = jsonArray.getJSONObject(i);
        String fechas = explrObject.getString( name: "ts");
        String die = explrObject.getString( name: "die");
        String gas = explrObject.getString( name: "gas");
        String eol = explrObject.getString( name: "eol");
        String ele = explrObject.getString( name: "ele");
        String vap = explrObject.getString( name: "vap");
        String fot = explrObject.getString( name: "fot");
        String cc = explrObject.getString( name: "cc");
        String hid = explrObject.getString( name: "hid");

        renov = (Float.parseFloat(eol) +Float.parseFloat(fot) + Float.parseFloat(hid))*100;
        norenov = (Float.parseFloat(die) + Float.parseFloat(gas) + Float.parseFloat(ele) +
                Float.parseFloat(vap) + Float.parseFloat(cc))*100;

        mPieChart.clearChart();
        mPieChart.addPieSlice(new PieModel( _legendLabel: "Renovable", renov, Color.parseColor( colorString: "#B4D888")));
        mPieChart.addPieSlice(new PieModel( _legendLabel: "No Renovable", norenov, Color.parseColor( colorString: "#689F38")));
        mPieChart.startAnimation();
```

Ilustración 41. Código Java, gráfico circular.

1.2.12. Consejos

La pantalla de consejos es la última opción que se le presenta al usuario para concienciarlo de métodos de ahorro de energía. Este activity se ha desarrollado de manera similar a las pantallas de Información ya que está formado por un ScrollView que permite el desplazamiento vertical y diferentes TextView que contienen todos los consejos.

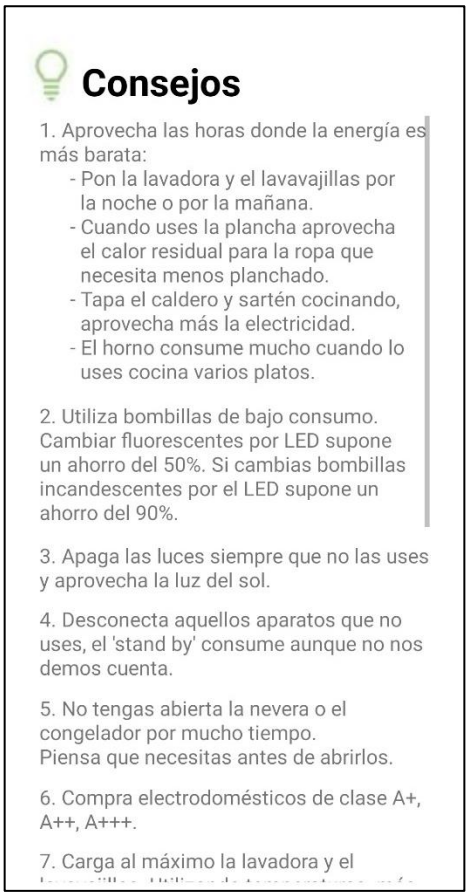


Ilustración 42. Activity Consejos.

Capítulo 4. Anexos.

Índice Anexos

1. Datasheet Placa 330 W

Electrical data (at standard conditions (STC) irradiance 1000 watt/m², spectrum AM 1.5 at a cell temperature of 25°C)

Type	Nominal output Pmpp	Nominal voltage Umpp	Nominal current Impp	Short circuit current Isc	Open circuit voltage Uoc	Module conversion efficiency
AC-330M/156-72S	330 Wp	37.70 V	8.77 A	9.28 A	46.20 V	17.01 %
AC-335M/156-72S	335 Wp	37.90 V	8.85 A	9.29 A	46.40 V	17.26 %
AC-340M/156-72S	340 Wp	38.00 V	8.95 A	9.40 A	46.50 V	17.52 %
AC-345M/156-72S	345 Wp	38.10 V	9.06 A	9.48 A	46.60 V	17.70 %
AC-350M/156-72S	350 Wp	38.20 V	9.16 A	9.56 A	46.70 V	17.90 %

Design

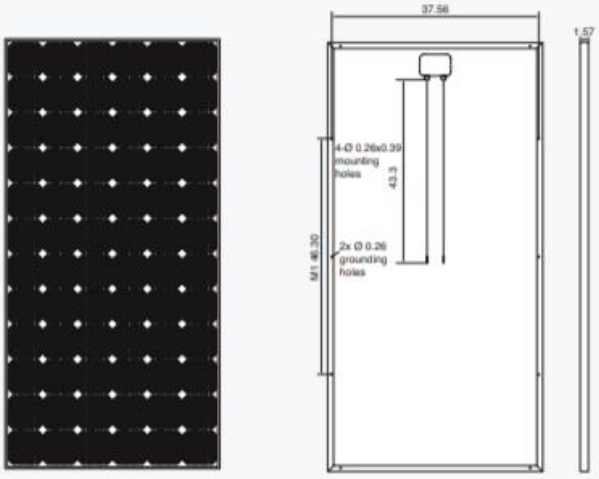
Frontside	0.13 inch (3.2 mm) hardened, low-reflection white glass
Cells	72 monocrystalline high efficiency cells 6 inch (156 x 156 mm)
Backside	Composite film
Frame	1.57 inch (40 mm) silver anodized aluminium frame

Mechanical data

L x W x H	77.01 x 39.06 x 1.57 inch (1956 x 992 x 40 mm)
Weight	50.7 lbs (23 kg) with frame

Power connection

Socket	Protection Class IP67 (3 bypass diodes)
Wire	43.3 inch, AWG 11
Plug-in system	Plug/socket IP67



All dimensions in inch

Limit values

System voltage	1000 VDC (UL) 1000 VDC (IEC)
NOCT (nominal operating cell temperature)*	45°C +/-2K
Max. load-carrying capacity	50 PSF
Reverse current feed IR	15.0 A
Permissible operating temperature	-40°C to 85°C / -40F to 185F

(No external voltages greater than Vo may be applied to the module)

* NOCT, irradiance 800 W/m²; AM 1,5; wind speed 1 m/s; Temperature 20°C

Temperature coefficients

Voltage Uoc	-0.30 %/K
Current Isc	0.04 %/K
Output Pmpp	-0.40 %/K

Low-light performance (Example for AC-300M/156-72S)

I-U characteristic curve	Current Ipp	Voltage Upp
200 W/m²	1.69 A	34.55 V
400 W/m²	3.30 A	35.42 V
600 W/m²	4.93 A	35.70 V
800 W/m²	6.48 A	36.21 V
1000 W/m²	8.18 A	36.73 V

Packaging

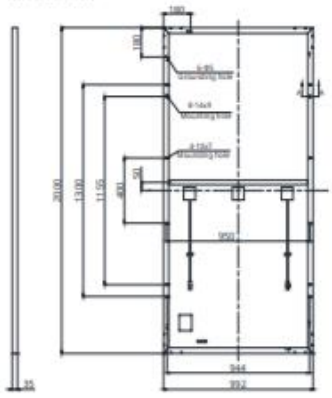
Module pieces per pallet	25
Module pieces per HC-container	550

Datasheet 1. Placa 330 W.
Fuente: https://solarmyplace.com/products/axitec-330w-mono-crystalline-solar-panel-ac-330m-156-72s

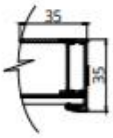
2. Datasheet Placa 385W

ENGINEERING DRAWING (mm)

Rear View

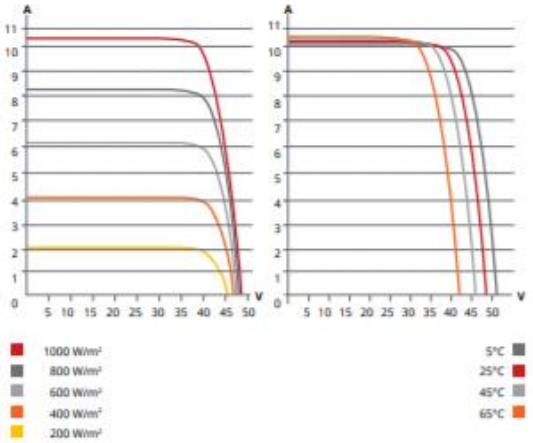


Frame Cross Section A-A



Mounting Hole

CS3U-400MS / I-V CURVES



ELECTRICAL DATA | STC*

CS3U	380MS	385MS	390MS	395MS	400MS
Nominal Max. Power (Pmax)	380 W	385 W	390 W	395 W	400 W
Opt. Operating Voltage (Vmp)	40.0 V	40.2 V	40.4 V	40.6 V	40.8 V
Opt. Operating Current (Imp)	9.50 A	9.58 V	9.66 A	9.73 A	9.81 A
Open Circuit Voltage (Voc)	47.8 V	48.0 V	48.2 V	48.4 V	48.6 V
Short Circuit Current (Isc)	10.01 A	10.09 A	10.17 A	10.25 A	10.33 A
Module Efficiency	19.15%	19.41%	19.66%	19.91%	20.16%
Operating Temperature	-40°C ~ +85°C				
Max. System Voltage	1500V (IEC/UL) or 1000V (IEC/UL)				
Module Fire Performance	TYPE 1 (UL 1703) or Class C (IEC 61730)				
Max. Series Fuse Rating	30 A				
Application Classification	Class A				
Power Tolerance	0 ~ + 5 W				

* Under Standard Test Conditions (STC) of irradiance of 1000 W/m², spectrum AM 1.5 and cell temperature of 25°C.

ELECTRICAL DATA | NMOT*

CS3U	380MS	385MS	390MS	395MS	400MS
Nominal Max. Power (Pmax)	283 W	287 W	290 W	294 W	298 W
Opt. Operating Voltage (Vmp)	37.2 V	37.4 V	37.6 V	37.8 V	38.0 V
Opt. Operating Current (Imp)	7.60 A	7.66 A	7.72 A	7.78 A	7.84 A
Open Circuit Voltage (Voc)	44.8 V	45.0 V	45.2 V	45.4 V	45.6 V
Short Circuit Current (Isc)	8.08 A	8.14 A	8.20 A	8.27 A	8.33 A

* Under Nominal Module Operating Temperature (NMOT), irradiance of 800 W/m², spectrum AM 1.5, ambient temperature 20°C, wind speed 1 m/s.

MECHANICAL DATA

Specification	Data
Cell Type	Mono-crystalline
Cell Arrangement	144 [2 X (12 X 6)]
Dimensions	2000 X992 X35 mm (78.7 X39.1 X1.38 in)
Weight	22.5 kg (49.6 lbs)
Front Cover	3.2 mm tempered glass
Frame	Anodized aluminium alloy, crossbar enhanced
J-Box	IP68, 3 bypass diodes
Cable	4 mm² (IEC), 12 AWG (UL)
Cable Length (Including Connector)	Portrait: 400 mm (15.7 in) (+) / 280 mm (11.0 in) (-); landscape: 1250 mm (49.2 in);leap-frog connection: 1670 mm (65.7 in)*
Connector	T4 series or H4 UTX or MC4-EVO2
Per Pallet	30 pieces
Per Container (40' HQ)	660 pieces

* For detailed information, please contact your local Canadian Solar sales and technical representatives.

TEMPERATURE CHARACTERISTICS

Specification	Data
Temperature Coefficient (Pmax)	-0.36 % / °C
Temperature Coefficient (Voc)	-0.29 % / °C
Temperature Coefficient (Isc)	0.05 % / °C
Nominal Module Operating Temperature	42 ± 3°C

Datasheet 2. Placa 385W.
Fuente: https://solarmyplace.com/products/canadian-solar-kumax-385w-mono-144-cell-slv-wht-solar-panel-cs3u-385ms

3. Datasheet Placa 405W

Mechanical Properties

Cells	6 x 12
Cell Vendor	LG
Cell Type	Monocrystalline / N-type
Cell Dimensions	161.7 x 161.7 mm
# of Busbar	12 (Multi Wire Busbar)
Dimensions (L x W x H)	2024 x 1024 x 40 mm
Front Load	5400 Pa
Rear Load	3000 Pa
Weight	20.3 kg
Connector Type	Genuine MC4, IP68 (Male: PV-KST4) (Female: PV-KBT4)
Junction Box	IP68 with 3 bypass diodes
Length of Cables	2 x 1200 mm
Front cover	High transmission tempered glass
Frame	Anodised aluminum

Certifications and Warranty

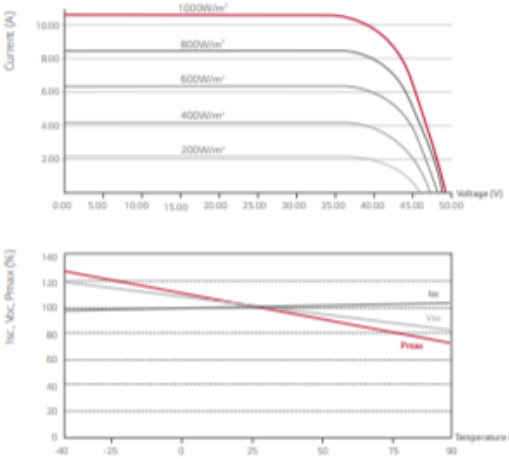
Certifications	ISO 9001, ISO 14001, ISO 50001
	IEC 61215-1/-1-1/2:2016, IEC 61730-1/-2:2016, UL1703
	OHSAS 1001, PV CYCLE
	IEC 61701:2012 Severity 6 (Salt Mist Corrosion Test)
	IEC 62716:2013 (Ammonia Test)
Module Fire Rating	Class C (UL 790, ULC/ORD C 1703)
Product Warranty	25 Years
Output Warranty of Pmax (Measurement Tolerance ± 3%)	Linear Warranty[1]

[1] 1) 1st year 98%, 2) After 1st year 0.35% annual degradation, 3) 89.6% for 25 years

Temperature Characteristics

NMOT	42 ± 3 ℃
Pmax	-0.36 %/℃
Voc	-0.26 %/℃
Isc	0.02 %/℃

Characteristic Curves



Electrical Properties (STC[2])

Module Type	400 W	405 W
Maximum Power Pmax (W)	400	405
MPP Voltage Vmpp (V)	40.6	41.0
MPP Current Impp (A)	9.86	9.89
Open Circuit Voltage Voc (V)	49.3	49.4
Short Circuit Current Isc (A)	10.47	10.51
Module Efficiency (%)	19.3	19.5
Operating Temperature (℃)	-40 ~ +90	
Maximum System Voltage (V)	1000 (IEC) / 1500 (UL)	
Maximum Series Fuse Rating (A)	20	
Power Tolerance (%)	0 ~ +3	

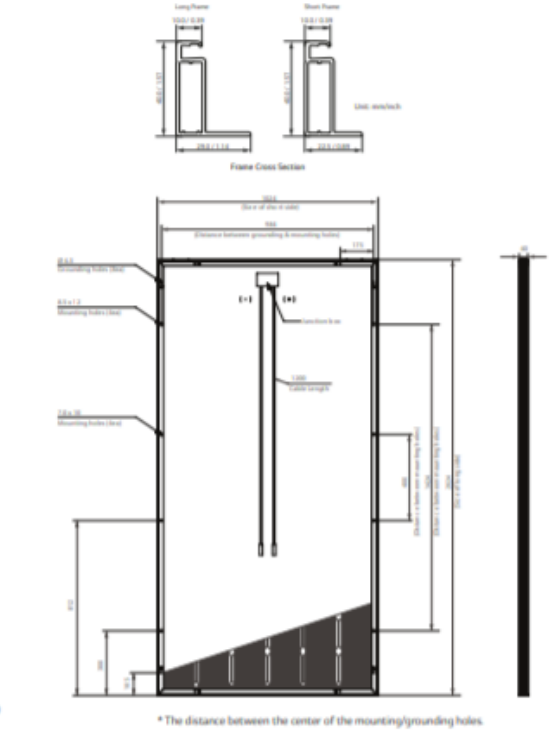
[2] STC (Standard Test Condition): Irradiance 1000 W/m², Module Temperature 25 ℃, AM 1.5.
The nameplate power output is measured and determined by LG Electronics at its sole and absolute discretion.

Electrical Properties (NMOT[3])

Module Type	400 W	405 W
Maximum Power Pmax (W)	300	304
MPP Voltage Vmpp (V)	38.0	38.4
MPP Current Impp (A)	7.88	7.91
Open Circuit Voltage Voc (V)	46.5	46.6
Short Circuit Current Isc (A)	8.40	8.44

[3] NMOT (Nominal Module Operating Temperature): Irradiance 800 W/m², ambient temperature 20 ℃, wind speed 1 m/s, Spectrum AM 1.5.

Dimensions (mm)



Datasheet 3. Placa 405W.
Fuente: https://solarmyplace.com/products/lg-neon-2-405w-mono-72-cell-slv-wht-solar-panel-lg405n2w-v5

Conclusiones

Si bien es cierto que la idea original de este TFM consistía en un desarrollo más práctico, debido a la situación del COVID-19 y, queriendo mantener la idea inicial de transformar nuestras ciudades en un lugar mejor donde vivir, se ha tenido que adaptar este Trabajo de Fin de Máster a los recursos disponibles.

Partiendo de un desconocimiento previo en programación de aplicaciones, el desarrollo de este proyecto ha sido toda una aventura que termina con un resultado satisfactorio para mi desarrollo personal.

Crear Smart Energy ha conllevado un gran estudio del mundo de las Smart Cities y de Android Studio pero ha permitido además la adquisición de nuevos conocimientos y mejora de las habilidades de programación y manejo de datos.

El mundo cambia cada día más rápido y cómo enunció su libro “El origen de las especies” Charles Darwin “no es el más fuerte, ni el más inteligente el que sobrevive si no aquel que se adapta mejor al cambio”.

Debemos transformar nuestras ciudades a los avances tecnológicos que vienen, reducir los niveles de contaminación del planeta y hacer a las personas participes del lugar en el que viven. En este proyecto se ha desarrollado una aplicación que busca concienciar a los usuarios de la necesidad de consumir menor energía, apostar por el transporte sostenible y las energías renovables. Todo ello, acompañado de la innovación tecnológica para el manejo de datos abiertos y en tiempo real, permitiendo tener todas las opciones en un smartphone.

Conclusions

While it is true that the original idea of this TFM included a more practical development, due to the situation of COVID-19 and, wanting to maintain the initial idea of transforming our cities into a better place to live, this Final Master’s Work has had to be adapted to the resources available.

Based on a previous lack of knowledge in application programming, the evolution of this project has been an adventure that ends with a satisfactory result for my personal development.

Creating Smart Energy has led to a major study of the world of Smart Cities and Android Studio but it has also allowed the acquisition of new knowledge and improvement of programming and data management skills.

The world is changing faster every day and how Charles Darwin sets out on his book "The Origin of Species", "is not the strongest, nor the smartest who survives but the one who adapts the best to change."

We must transform our cities to the technological advances that are to come, reduce the pollution levels of the planet and make people to be part of where they live. This project has developed an application that seeks to raise awareness of the need to consume less energy, support sustainable transport and renewable energies. All of this, accompanied by technological innovation for the handling of open data and real time data, allowing to have all the options on a smartphone.

Bibliografía

[1] I. Cebrián, *Libro blanco: smart cities.* Bilbao: Enerlis, 2012.

[2] REE.es. *Inicio Red Eléctrica de España*. [en línea]. [Fecha de consulta: 20 de marzo de 2020]. Disponible en Internet: https://www.ree.es/es.

[3] DEVELOPER.ANDROID.com. *Arquitectura de la plataforma. Desarrolladores de Android. Android Developers*. [en línea]. [Fecha de consulta: 14 de marzo de 2020]. Disponible en Internet: https://developer.android.com/guide/platform?hl=es.

[4] WIKIPEDIA.org. *Anexo: Historial de versiones de Android*. *Wikipedia, la enciclopedia libre*. [en línea]. [Fecha de consulta: 14 de marzo de 2020]. Disponible en Internet: https://es.wikipedia.org/w/index.php?title=Anexo:Historial_de_versiones_de_Android&oldid=127157572.

[5] ENDESA.com. *Las tarifas de acceso de la luz. Endesa*. [en línea]. [Fecha de consulta: 25 de marzo de 2020]. Disponible en Internet: https://www.endesa.com/es/conoce-la-energia/energia-y-mas/tarifas-acceso-electricidad.

[6] ENDESA.com. *Tu factura de luz Endesa. Endesa*. [en línea]. [Fecha de consulta: 25 de marzo de 2020]. Disponible en Internet: https://www.endesa.com/es/te-ayudamos/factura-luz-ml.

[7] España. Ley 24/2013, de 26 de diciembre, del Sector Eléctrico. Boletín Oficial del Estado, 27 de diciembre de 2013, núm. 310.

[8] España. Real Decreto 244/201, de 5 de abril, por el que se regulan las condiciones administrativas, técnicas y económicas del autoconsumo de energía eléctrica. Boletín Oficial del Estado, 6 de abril de 2019, núm. 83.

[9] IDAE.es. *Autoconsumo. IDAE*. [en línea]. [Fecha de consulta: 10 de junio de 2020]. Disponible en Internet: https://www.idae.es/tecnologias/energias-renovables/autoconsumo.

[10] SANTACRUZDETENERIFE.es. *Portal de Datos Abiertos#: Opendata*. [en línea]. [Fecha de consulta: 12 de abril de 2020]. Disponible en Internet: https://www.santacruzdetenerife.es/web/gobierno-abierto/opendata.

[11] DATOSABIERTOS.LASPALMASGC.es. *Portal de Datos Abiertos Las Palmas de Gran Canaria*. [en línea]. [Fecha de consulta: 12 de abril de 2020]. Disponible en Internet: http://datosabiertos.laspalmasgc.es/.

[12] GITHUB.com. *ItziarRH/Smart-Energy, GitHub*. [en línea]. [Fecha de consulta: 1 de marzo de 2020]. Disponible en Internet: https://github.com/ItziarRH/Smart-Energy.

[13] ELECTROCALCULATOS.com. C*uanto gasta un electrodoméstico - tabla completa de aparatos eléctricos*. [en línea]. [Fecha de consulta: 5 de abril de 2020]. Disponible en Internet: https://www.electrocalculator.com/.

[14] HMSISTEMAS.es. *Calculadora de HSP de HMSistemas para un diseño propio de su instalación*. [en línea]. [Fecha de consulta: 12 de junio de 2020]. Disponible en Internet: http://www.hmsistemas.es/shop/catalog/calculadora_hsp.php?osCsid=f438630572d1a06825580180b1de7e7c (accedido jul. 05, 2020).

[15] AREATECNOLOGIA.com. *Calculos Instalación Solar Fotovoltaica*. [en línea]. [Fecha de consulta: 12 de junio de 2020]. Disponible en Internet: https://www.areatecnologia.com/electricidad/calculo-fotovoltaica.html.

[16] THEGRID.REXEL.com. *¿Cómo Estimar el Periodo de Retorno de un Sistema Solar Fotovoltaico? - Wiki - Electricista Wiki Español, A Rexel Customer Community*. [en línea]. [Fecha de consulta: 12 de junio de 2020]. Disponible en Internet: https://thegrid.rexel.com/en-us/knowledge/electricista-wiki-espanol/w/wiki/709/como-estimar-el-periodo-de-retorno-de-un-sistema-solar-fotovoltaico.

[17] REE.es. *Demanda de energía eléctrica en tiempo real, estructura de generación y emisiones de CO2*. [en línea]. [Fecha de consulta: 2 de mayo de 2020]. Disponible en Internet: https://demanda.ree.es/visiona/seleccionar-sistema.