UNIVERSITAT POLITÈCNICA DE VALÈNCIA

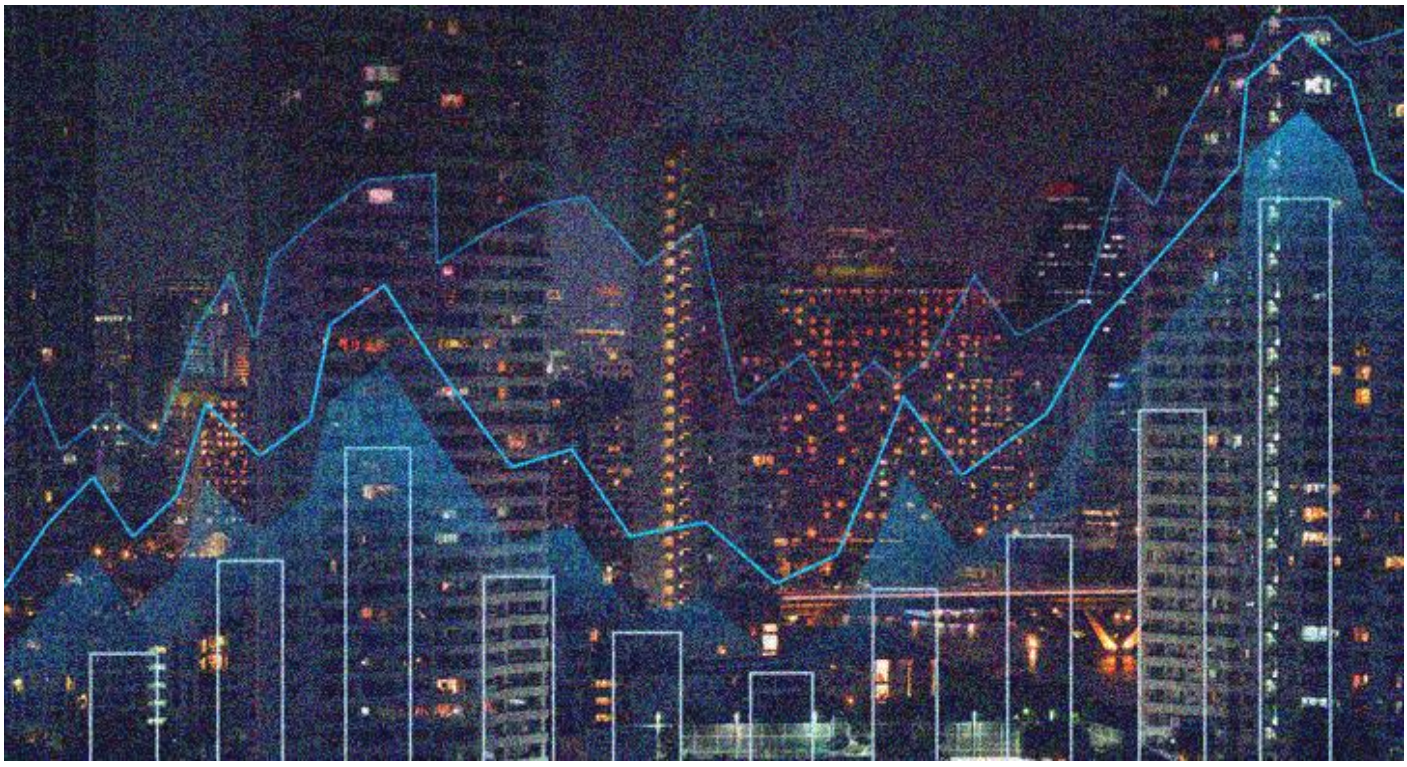
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA CARTOGRÁFICA Y TOPOGRÁFICA

CREACIÓN E IMPLEMENTACIÓN DE APLICACIONES SIG EN EL SECTOR BANCARIO

TRABAJO FIN DE GRADO
INGENIERÍA EN GEOMÁTICA Y TOPOGRAFÍA

Autora: Elena Clemente Ortiz
Tutora: Eloína Peregrina Coll Aliaga

Valencia, diciembre de 2017



Resumen

Las capacidades digitales con los SIG nos permiten enriquecer nuestra información con todo tipo de datos a nivel mundial para tomar mejores decisiones y explotar nuevas capacidades. En el presente proyecto se trata de incorporar esta tecnología en el sector bancario con el fin de obtener aplicaciones con componente espacial que permitan acercarse al cliente y ofrecer servicios más dinámicos. Para ello se adopta la postura de una entidad bancaria ficticia que desarrolla nuevas aplicaciones para aprovechar las ventajas que le ofrece los SIG.

Resum

Les capacitats digitals amb els SIG ens permeten enriquir la nostra informació amb tot tipus de dades a nivell mundial per a prendre millors decisions i explotar noves capacitats. En el present projecte es tracta d'incorporar aquesta tecnologia en el sector bancari amb la finalitat d'obtindre aplicacions amb component espacial que permeten acostar-se al client i oferir serveis més dinàmics. Per a això s'adopta la postura d'una entitat bancària fictícia que desenrotlla noves aplicacions per a aprofitar els avantatges que li oferix els SIG.

Abstract

The digital capabilities with GIS allow us to enrich our information with all types of data worldwide to make better decisions and exploit new capabilities. In the present project, this technology is incorporated in the banking sector in order to obtain applications with a spatial component that allow to approach the client and offer more dynamic services. For this, the position of a fictitious banking entity is adopted, which develops new applications to take advantage of the advantages offered by GIS.

Índice de Contenido

1. Introducción 5
2. Aplicaciones desarrolladas 6
   2.1. Cambio de moneda 7
      2.1.1. Resumen de la aplicación 7
      2.1.2. Tratamiento y Publicación de los datos 8
      2.1.3. Desarrollo y resultados 9
   2.2. Cajero más Cercano 14
      2.2.1. Resumen de la aplicación 14
      2.2.2. Tratamiento y Publicación de los Datos 14
      2.2.3. Desarrollo y Resultados 18
   2.3. Calcula tu hipoteca 21
      2.3.1. Resumen de la aplicación 21
      2.3.2. Tratamiento y Publicación de los Datos 21
      2.3.3. Desarrollo y resultados 25
   2.4. Ubicación de Oficinas 29
      2.4.1. Resumen de la Aplicación 29
      2.4.2. Tratamiento y Publicación de los Datos 30
      2.4.3. Desarrollo y Resultados 41
   2.5. Story Map. Nuestro banco a través de los años 44
      2.5.1. Resumen de la aplicación 44
      2.5.2. Desarrollo y publicación 44
3. Creación de GDB y Versionado 47
4. Diseño de la Organización 51
5. Anexos 54
   5.1. JS 'WorldCurrencyMap' – Cambio de Moneda 54
      5.1.1. CSS Aplicación 57
   5.2. Widget 'ClosestFacility' – Cajero más Cercano 58
   5.3. Widget 'Hipoteca' – Calcula tu Hipoteca 62
      5.3.1. CSS Widget 63
   5.4. Widget 'NuevaOficina' – Ubicación de Oficinas 64
      5.4.1. CSS Widget 67
   5.5. Cambio CSS StoryMap 68
6. Bibliografía 69

Índice de Ilustraciones

Ilustración 1. Conversor de moneda clásico. 7
Ilustración 2. Uso herramienta Simplify Polygon. 8
Ilustración 3. Tipos de cambio del Banco Central Europeo. 9
Ilustración 4. Widget Search proporcionado por Esri. 10
Ilustración 5. Tipos de cambio en formato XML del BCE. 10
Ilustración 6. Esquema de implementación de un Proxy. 11
Ilustración 7. Incorporación del Proxy mediante código. 11
Ilustración 8. Datos que rescatar del archivo XML del BCE. 11
Ilustración 9. Estructura 'árbol' de un archivo XML. 12
Ilustración 10. Acceso a elementos XML mediante código. 12
Ilustración 11. Visualización inicial de la aplicación 'Cambio de Moneda'. 13
Ilustración 12. Detalle de uso de la aplicación 'Cambio de Moneda'. 13
Ilustración 13. Tabla de campos de la capa Cajeros. 15
Ilustración 14. Model Builder creado para el cálculo del cajero más cercano. 16
Ilustración 15. Detalle creación del 'modo a coche'. 17
Ilustración 16. Especificación de la escala para compatibilidad con visores web. 18
Ilustración 17. Servicio 'Closest Facility' alojado en ArcGIS Server. 18
Ilustración 18. Configuración del servicio de ruta en el Widget. 19
Ilustración 19. Visualización del widget 'Indicaciones' en la aplicación. 20
Ilustración 20. Visualización del widget creado 'ClosestFacility' en la aplicación. 20
Ilustración 21. Propiedades en venta con la herramienta 'Crear puntos aleatorios'. 22
Ilustración 22. Tabla campos de la capa 'Propiedades en venta'. 22
Ilustración 23. Incorporación de imágenes adjuntas a la capa 'Propiedades en venta'. 23
Ilustración 24. Datos evolución del suelo por Idealista. 23
Ilustración 25. Herramienta conversión datos Excel a tabla. 24
Ilustración 26. Unión de la tabla 'evolución del suelo' a la capa de 'Barrios de Madrid'. 24
Ilustración 27. Incorporación simbología a la capa 'Evolución del suelo' con ArcGIS Online. 25
Ilustración 28. Cálculo en Excel del método de Amortización Francés. 26
Ilustración 29. Cálculo de una hipoteca en Excel. 26
Ilustración 30. Apariencia del cálculo de la hipoteca en el widget 'Hipoteca'. 27
Ilustración 31. Código implementado para el cálculo de la hipoteca. 27
Ilustración 32. Visualización de la evolución del suelo en la aplicación. 28
Ilustración 33. Visualización de las características de la propiedad en venta. 28
Ilustración 34. Visualización del widget 'Hipoteca' en la aplicación. 29
Ilustración 35. Incorporación de capas en la base de datos SDE. 31
Ilustración 36. Script de Python para el cálculo aleatorio del campo 'ingreso medio'. 31
Ilustración 37. Visualización de las parcelas donde habrá clientes. 32
Ilustración 38. Script de Python empleado para la creación de clientes. 32
Ilustración 39. Resultado de la capa de clientes realizada. 33
Ilustración 40. Capa de barrios de Madrid. 34
Ilustración 41. Población desempleada en los barrios de Madrid. 35
Ilustración 42. Población de 25-29 años en los barrios de Madrid. 36
Ilustración 43. Total población mayor de 60 años en los barrios de Madrid. 36

Ilustración 44. Ingreso medio por año en los barrios de Madrid. 37
Ilustración 45. Gastos totales por año en los barrios de Madrid. 37
Ilustración 46. Fuente de ArcGIS Online para extracción de datos. 38
Ilustración 47. Creación del Service Area. 38
Ilustración 48. Comprobación funcionalidad Service Area. 39
Ilustración 49. Definición de características Service Area. 40
Ilustración 50. Service Area alojado en ArcGIS Server. 41
Ilustración 51. Incorporación de los widget en el config-demo. 42
Ilustración 52. Visualización del mapa de calor de la aplicación. 43
Ilustración 53. Visualización del widget 'Nueva Oficina' en la aplicación. 43
Ilustración 54. Habilitar barra de tiempo en el web map. 45
Ilustración 55. Visualización inicial del Story Map. 46
Ilustración 56. Visualización del Story Map. 47
Ilustración 57. Creación de GDB Corporativa en ArcMap. 47
Ilustración 58. Estructura de las capas en la GDB con usuario SDE. 48
Ilustración 59. Creación de usuarios en la database. 48
Ilustración 60. Tabla de privilegios según capa-usuario. 49
Ilustración 61. Registro de capas editables de la GDB. 50
Ilustración 62. Esquema del Versionado de la GDB. 50
Ilustración 63. Diseño de la cuenta de ArcGIS Online 1. 51
Ilustración 64. Diseño de la cuenta en ArcGIS Online 2. 51
Ilustración 65. Grupos creados en ArcGIS Online y su visualización desde ArcMap. 52
Ilustración 66. Estructura del contenido dentro de la cuenta de ArcGIS Online. 53

1. Introducción

Nos encontramos en una situación que no es fácil, unos hablan de cambio de época otros de época de cambios, pero lo que si es cierto es que hasta ahora la tecnología ha cambiado el mundo y lo seguirá haciendo en el futuro. Esto ha generado que haya mucha oferta para la población, pero a su vez también están sufriendo incertidumbre, ya que hay demasiada información falsa en el mundo digital.

En el presente proyecto nos introducimos en el **sector bancario** adoptando la postura de **Bankgister**, una entidad bancaria que se ha introducido en el mercado recientemente, pero gracias a haber aprovechado las ventajas que el mundo tecnológico ofrece ha sabido posicionarse como uno de los referentes.

La Transformación Digital empodera a los empleados y ayuda a las organizaciones a conocer mejor a los clientes para ofrecerles la mejor oferta en el momento que la necesitan. Queríamos por lo tanto ponernos en el lugar de una gran empresa del sector bancario y analizar las posibilidades que tienen para crecer aprovechando las nuevas tecnologías.

En este caso, centrándonos más en su explotación de los Sistemas de Información Geográfica les ha proporcionado ventajas frente a sus competidores en el sector financiero, este es un entorno complejo en el que se encuentran y por lo tanto necesitan adaptarse, transformarse y evolucionar en esta era de la tecnología.

Se debe buscar una relación con los clientes de confianza, tanto con los actuales como con los posibles futuros y trabajar con claridad y transparencia. Es importante que nuestro banco tenga la capacidad de saber de sus clientes la máxima información posible acerca de: ¿qué?, ¿cuándo?, ¿dónde?, ¿cómo? y ¿por qué? Las respuestas a estas preguntas tienen que ser sencillas, simples y obtenerlas con las herramientas adecuadas. De este modo se puede conseguir transformar dicha información en decisiones acertadas para mejorar los resultados de la empresa y en decisiones útiles en relación con sus clientes.

2. Aplicaciones desarrolladas

Vamos hacia un mundo en el que casi la mitad de la población está conectada a internet y seguirá creciendo al igual que los datos digitales. De estos datos digitales el 80% tiene componente espacial que debidamente aprovechada crea ventajas competitivas. Esta dimensión espacial dota de contenido al proceso de la transformación digital y la visualización de datos nos ofrece entendimiento de lo que está ocurriendo en nuestro negocio.

Los Bancos deben prepararse para poder crear ventajas competitivas en un mundo en el que todo (personas, procesos y cosas) va a estar conectado y generando datos con los cuales se genera información y conocimiento.

Las capacidades digitales con los SIG nos permiten enriquecer nuestra información con todo tipo de datos a nivel mundial para tomar mejores decisiones. Es por ello que Bankgister al incorporar esta tecnología ha conseguido aplicaciones con componente espacial que le permiten acercarse al cliente y ofrecer servicios más dinámicos.

A la hora de idear el enfoque de las aplicaciones a desarrollar nos replanteamos las decisiones que debíamos tomar relacionando nuestros servicios con preguntas relacionadas con la dimensión espacial:

- ¿Dónde viven nuestros clientes?
- ¿Dónde tengo situadas las oficinas?
- ¿Dónde existen oportunidades de mercado?
- ¿Dónde abrir o cerrar la próxima oficina?
- ¿Qué área de influencia abarca cada oficina?
- ¿Qué acceso tienen nuestros clientes a los cajeros automáticos?

La finalidad es responder a estar preguntas dándole al usuario aquellas herramientas que pueden serle útiles para conseguir acercar su relación con el banco. Por ello pensamos en aplicaciones dinámicas, que sabemos que tienen demanda y nos permitan acercar el sector económico de una forma más visual.

Para este proceso nos apoyamos en ArcGIS, la plataforma de información geográfica de Esri, que aúna información de distintas fuentes (datos socioeconómicos, de riesgo y financieros) para obtener una realidad diferente. A través de los mapas conseguimos analizar y entender estos datos como nunca.

2.1. Cambio de moneda

2.1.1. Resumen de la aplicación

"Se llama cambio de moneda a toda operación en la que un determinado activo, instrumento financiero o medio de pago (o sea, billetes, cheques, depósitos, préstamos, etc.) pasa de ser expresado en una determinada divisa (la moneda o unidad de cuenta de un país o zona económica) a estar en una divisa diferente."

Esta definición ofrecida por el Banco de España es más amplia de lo que normalmente se entiende por cambios de moneda que suele ser la compraventa de billetes extranjeros, es decir, si te vas de vacaciones a Estados Unidos y cambias billetes de euros por billetes de dólares. Esta última parte es la situación que ocurre diariamente, cuando por motivos personales, de trabajo, etc. nos encontramos en la tesitura de comprobar cuánto dinero será necesario llevarnos y dónde solicitarlo.

Suele ser a nuestra entidad bancaria a la que le solicitemos dicha operación, pero es común que antes realicemos en internet la consulta del cambio que obtendremos en los clásicos conversores de moneda.

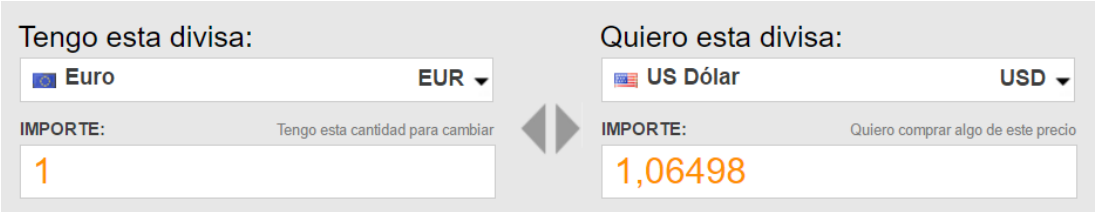


Ilustración 1. Conversor de moneda clásico.

La imagen anterior es un tipo de conversor de moneda que se puede encontrar, y fueron estas calculadoras las que nos dieron la idea, ya que los Bancos al igual que otras páginas también ofrece diariamente a sus clientes la cotización del cambio de divisas. Siendo pues esto un servicio del Banco la idea es que nosotros lo evolucionáramos haciéndolo más dinámico para los clientes.

Imaginemos que un conocido nuestro se va de viaje por primera vez a México por motivos de negocio, sabe el país al que viaja y que la moneda allí no es el euro. A partir de aquí se plantea dos cuestiones, ¿a qué tipo de moneda he de cambiar mi dinero? y ¿cuánto dinero me quiero llevar? Seguramente realizaría las respectivas preguntas al buscador de Google y entraría en diferentes páginas para averiguar la respuesta. Es el momento de poner en marcha nuestra aplicación y evitarle a nuestro cliente tanta búsqueda.

El concepto general de esta aplicación es por lo tanto que el Banco ofrezca un servicio interactivo a sus clientes que junte la geografía con un mapa del mundo (¿dónde viajar?), con el conversor de moneda (¿cuánto me llevo y a qué equivale?).

2.1.2. Tratamiento y Publicación de los datos

Decidimos desarrollar esta aplicación completamente con la API for JavaScript, al tener la idea de una interfaz sencilla consideramos que era factible realizarla sumando los conceptos adquiridos de HTML5, CSS y JavaScript.

Esta aplicación web te permite encontrar el cambio actual de cualquier país del mundo a través de un mapa, necesitábamos de un mapa del mundo dividido por países en el que tuviéramos al menos los nombres de cada uno de ellos. Un mapa interactivo en que el usuario puede visualizar todos los países del mundo, al hacer click, aparece la moneda de ese país, junto con el código de la moneda y el cambio actual a euros. La capa de los países viene de un layer package '**World_Countries**' publicado por ESRI en 2015.

Después de descargar la capa, añadimos dos campos nuevos: la moneda y el código de cada país. Además, realizamos un 'polygon simplification' de tipo remove point con una tolerancia de simplificación de 10km, con esto conseguimos que la capa cargue más rápido para evitar tiempo de espera al cliente. Esta técnica también la puedes ver en algunos Story Maps de ESRI que utilizan una capa del mundo (ej.: http://sdg.esri.com/learn/).

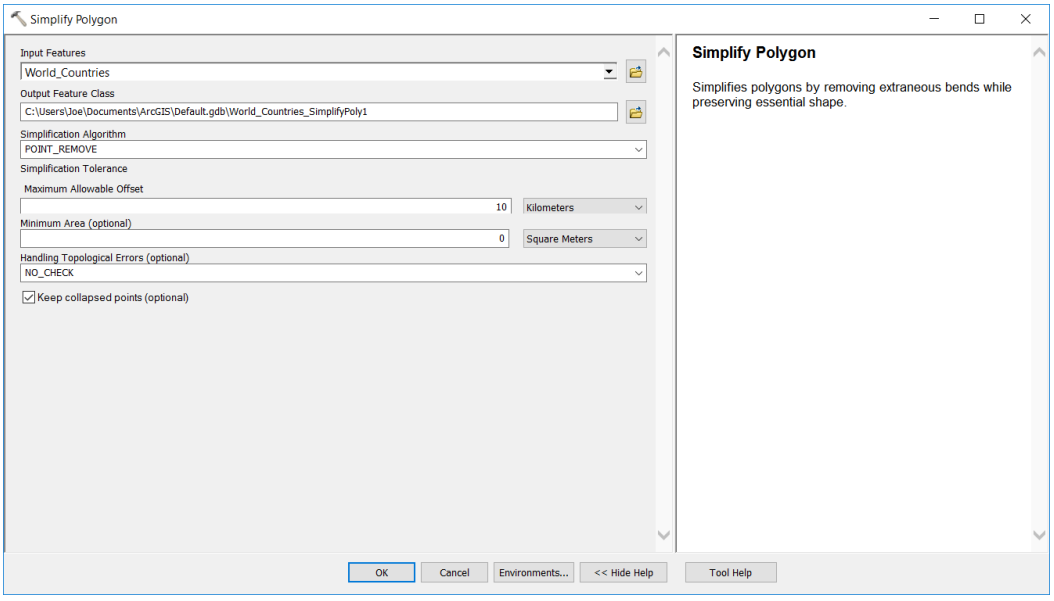


Ilustración 2. Uso herramienta Simplify Polygon.

La imagen anterior muestra la herramienta utilizada para conseguir la simplificación de los polígonos que representan a los países.

La otra parte esencial en nuestra aplicación era disponer de los datos del tipo de cambio aplicados. Los tipos de cambio son libres y son tipos de mercado que pueden cambiar en cualquier momento. En cualquier caso, los tipos de cambio oficiales del euro que publican el Banco Central Europeo (BCE) y otros bancos centrales suelen ser una referencia aproximada.

Bankgister, como banco establecido recientemente, decidió tomar como referencia los cambios publicados diariamente por el BCE hasta que en un futuro publique los suyos propios.

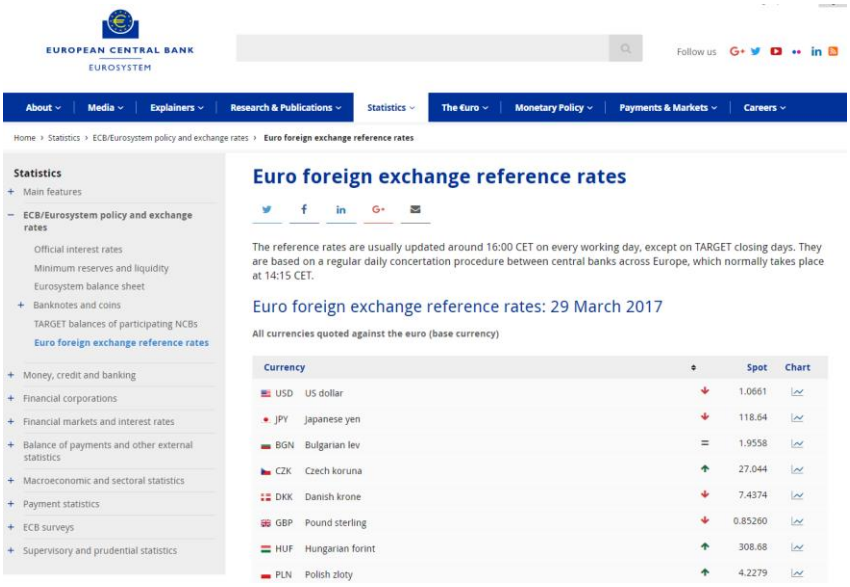


Ilustración 3. Tipos de cambio del Banco Central Europeo.

La imagen anterior muestra el servicio que publica diariamente el BCE, de este modo los datos ya están disponibles y será mediante el desarrollo de la aplicación donde veamos cómo acceder a ellos.

En esta aplicación solo ha sido necesaria la publicación de la capa de países World Map Currency. Una vez simplificada en ArcGIS Desktop, como se ha explicado en el apartado anterior, procedimos a publicarla a nuestro Server. La publicación finalizada con éxito nos permitió desde la cuenta de ArcGIS Online de Bankgister importarla como servicio y disponer de ella también en la organización.

2.1.3. Desarrollo y resultados

Para la capa principal, la idea es aplicarle que al seleccionar un país este se destaque visualmente. Con un evento “Mouse Over”, la ayuda de un ejemplo en el api llamado ‘feature layer hover’ y el CSS de Bootstrap configuramos que cuando el usuario pase el cursor por encima o lo seleccione, el país afectado resalte con un borde blanco.

Hemos incorporado a la aplicación un buscador con el **Widget Search**, desactivando inicialmente los recursos por defecto del buscador de Esri y añadiendo como fuente de búsqueda nuestra capa de países mediante el campo 'Country'. Con esto conseguimos que el cliente pueda realizar la búsqueda del país que desee en caso de no conocer su ubicación o simplemente porque es más cómodo.

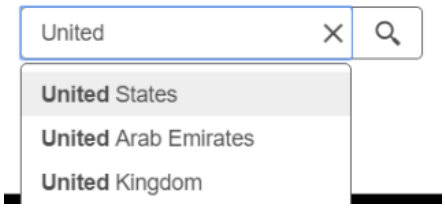


Ilustración 4. Widget Search proporcionado por Esri.

A su vez, el **Widget HomeButton** ha sido también añadido para permitir al usuario volver rápidamente a la extensión inicial del mapa.

Volviendo a la obtención de los datos para tener los cambios de moneda diarios, se realizará esta incorporación de datos dinámicos a través de la página del Banco Central Europeo a partir de su archivo XML:

```
▼<gesmes:Envelope xmlns:gesmes="http://www.gesmes.org/xml/2002-08-01" xmlns="http://www.ecb.int/vocabulary/2002-08-01/eurofxref">
   <gesmes:subject>Reference rates</gesmes:subject>
  ▼<gesmes:Sender>
     <gesmes:name>European Central Bank</gesmes:name>
   </gesmes:Sender>
  ▼<Cube>
   ▼<Cube time="2017-03-29">
      <Cube currency="USD" rate="1.0661"/>
      <Cube currency="JPY" rate="118.64"/>
      <Cube currency="BGN" rate="1.9558"/>
      <Cube currency="CZK" rate="27.044"/>
      <Cube currency="DKK" rate="7.4374"/>
      <Cube currency="GBP" rate="0.85260"/>
      <Cube currency="HUF" rate="308.68"/>
      <Cube currency="PLN" rate="4.2279"/>
      <Cube currency="RON" rate="4.5495"/>
      <Cube currency="SEK" rate="9.5145"/>
      <Cube currency="CHF" rate="1.0682"/>
```

Ilustración 5. Tipos de cambio en formato XML del BCE.

Al tratar de acceder a estos datos resultó que el recurso está en un dominio diferente al de nuestra aplicación y obteníamos el fallo de: 'Cross Origin Resource Sharing (CORS)' no está disponible, CORS es una especificación que permite a un servidor web interactuar con un navegador web, y determinar si una solicitud de origen se debe permitir. Esto quería decir que por parte del BCE nuestro dominio no era admitido y por lo tanto derivamos en que era necesario utilizar un proxy.

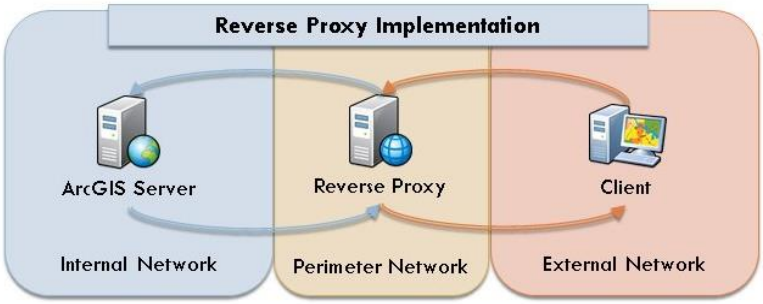


Ilustración 6. Esquema de implementación de un Proxy.

Como vemos en la imagen anterior, es un caso similar a cuando queremos acceder a los servicios del Servidor ArcGIS, en este caso sería a los servicios del BCE. Nuestra aplicación web enviará la petición al proxy y este reenvía la solicitud al servidor web remoto y transmite la respuesta devuelta por el servidor remoto de nuevo a nuestra aplicación, de este modo conseguimos el acceso a los datos dinámicos que queríamos.

```
//Acceso al XML del ECB
esriConfig.defaults.io.proxyUrl = "http://ELENA/DotNet/proxy.ashx";

urlUtils.addProxyRule({
  urlPrefix: "ecb.europa.eu",
  proxyUrl: "http://ELENA/DotNet/proxy.ashx"
});

var url = "http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml";

var requestHandle = esriRequest({
  "url": url,
  "handleAs": "xml"
});
requestHandle.then(requestSucceeded);
```

Ilustración 7. Incorporación del Proxy mediante código.

Las líneas anteriores incorporadas en nuestro código muestran el método para definir el proxy y su permiso a la lista de reglas para el conjunto de recursos. Todas las solicitudes a 'ecb.europa.eu' pasan por la URL del proxy especificado.

Toca recoger la información que nos interesa para aplicarla en nuestro código. Antes se ha mostrado en imagen la estructura tipo árbol del archivo xml, si entramos en detalle vemos que los datos que nos interesa recoger son las siglas que representan cada país (currency) y su cambio aplicado al euro (rate).

```
▼<gesmes:Sender>
   <gesmes:name>European Central Bank</gesmes:name>
 </gesmes:Sender>
▼<Cube>
  ▼<Cube time="2017-03-29">
     <Cube currency="USD" rate="1.0661"/>
     <Cube currency="JPY" rate="118.64"/>
     <Cube currency="BGN" rate="1.9558"/>
     <Cube currency="CZK" rate="27.044"/>
```

Ilustración 8. Datos que rescatar del archivo XML del BCE.

Para recoger dichos valores es necesario entender a qué elementos tenemos que acceder, para ello es interesante desglosar las partes de nuestro 'árbol' como en el siguiente ejemplo:

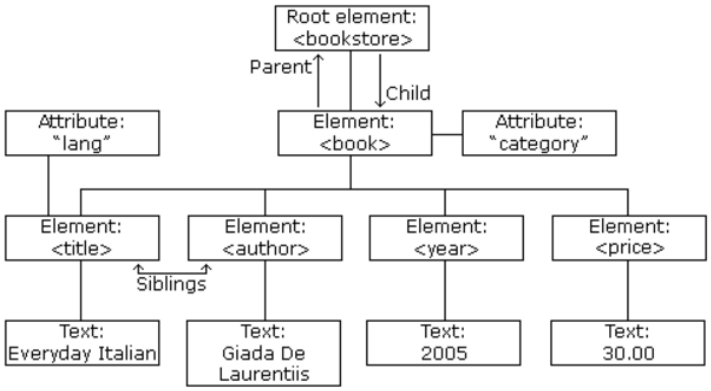


Ilustración 9. Estructura 'árbol' de un archivo XML.

Aplicado al documento xml que nos interesa, comprendemos pues que primero tendremos que acceder a los **elementos 'Cube'** y recoger de estos sus **atributos 'currency' y 'rate'**.

```
function requestSucceeded(response, io){
    var rates, i, xmlDoc, txt, input, output;
    xmlDoc = response;
    txt = "";
    rates = xmlDoc.getElementsByTagName('Cube');
    for (i = 2; i < rates.length; i++) {
      if (rates[i].getAttribute('currency') == codigo){
        txt += rates[i].getAttribute('rate') + "<br>";
```

Ilustración 10. Acceso a elementos XML mediante código.

Una vez disponibles estos datos, mediante código se crea la relación entre el campo de las siglas de la capa de países ('código') con el del atributo del archivo xml ('currency'), nuestra aplicación entonces realiza la búsqueda y cuando obtiene la equivalencia entre ambas nos devuelve el cambio aplicado necesario para realizar la operación de conversión. Estos datos recogidos y la conversión se ofrecen al usuario en un **panel interactivo** dónde pueden cambiar el dinero total del que quieren conocer su cambio a la moneda del país seleccionado.

Esta aplicación ha sido creada completamente mediante el uso de componentes de la API de JavaScript por lo que teníamos además el reto de ofrecer un resultado final estéticamente agradable, sencillo y dinámico para el usuario. Su aspecto final es el siguiente:

Ilustración 11. Visualización inicial de la aplicación 'Cambio de Moneda'.

Se pueden reconocer fácilmente todos los elementos de los cuales hemos comentado anteriormente. El mapa base oscuro superpuesto con nuestra capa de países, a la cual se ha aplicado un color agradable y adaptado a los colores corporativos de Bankgister. El panel dinámico que proporciona la información al usuario y los dos Widgets mencionados (HomeButton y Search).

Con una prueba diaria se comprobó que el acceso a los datos del BCE es el correcto y que estos se actualizan automáticamente, permitiendo una variación diaria ajustada al mercado. La conclusión de esta aplicación es que el resultado ha sido el esperado y la aplicación responde correctamente a las pruebas realizadas.

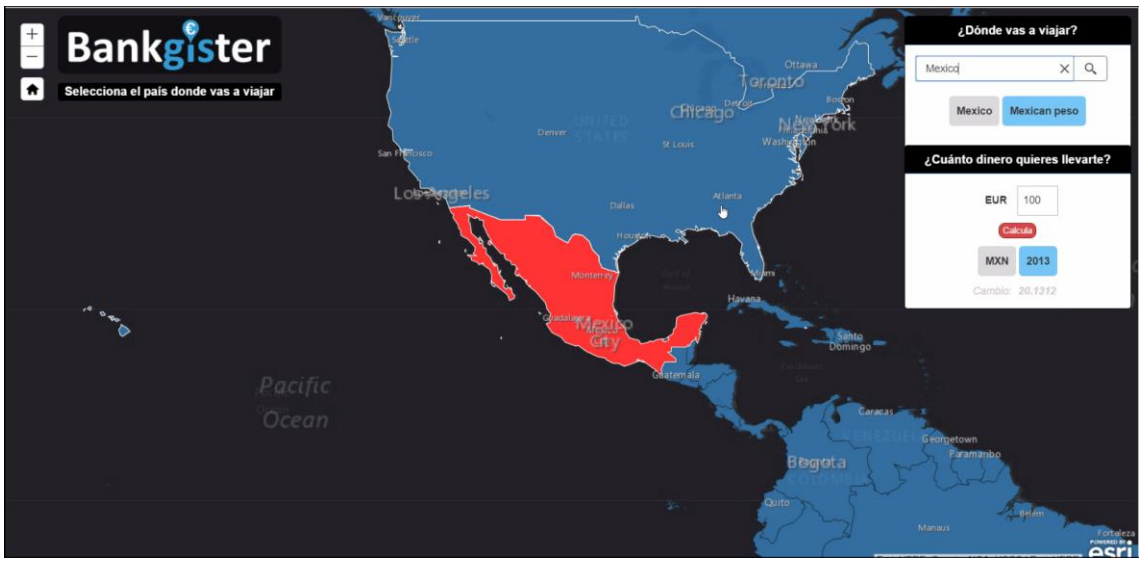


Ilustración 12. Detalle de uso de la aplicación 'Cambio de Moneda'.

2.2. Cajero más Cercano

2.2.1. Resumen de la aplicación

Esta aplicación se diseñó exclusivamente para todos los clientes de nuestra entidad bancaria, los cuales gracias a la misma podrán encontrar el cajero más cercano de Bankinter para realizar las operaciones que ellos consideren.

Gracias a la App los clientes podrán buscar hasta un máximo de cuatro cajeros, que se encuentren más cerca de su posición y podrán indicar el tiempo máximo de trayecto que ellos quieran. Para ello tendrán que dibujar su posición con un punto en el mapa, seleccionar el número de cajeros de destino, el tiempo de viaje y ejecutar la aplicación, que les calculará cuales son los cajeros más cercanos, y les indicara las direcciones que deben tomar para llegar hasta ellos. El cliente podrá dibujar en el mapa barreras, por si hay alguna o varias calles cortadas. Además, el usuario podrá acceder a los atributos de los cajeros, viendo cual es al que más le interesa ir; si se pueden hacer transacciones, si es 24h, o su estado.

El primer objetivo fue realizar el análisis de rutas más cercanas mediante el widget de geoprocesamiento, creando un modelo el ModelBuilder, y accediendo a él a través del propio widget. Pero este sistema se quedaba bastante corto, ya que no se podía acceder al número de 'facilities' que encontrar, ni acceder al tiempo del trayecto. Por lo que se optó a realizarlo mediante JavaScript, para posteriormente agregarlo a un Widget de WebAppBuilder.

El segundo objetivo y definitivo fue crear un Wigdet a través de otro de modelo, ya que se aumentaban las opciones que se le podían dar al cliente. Este nuevo Widget tiene bastante más funcionalidad.

2.2.2. Tratamiento y Publicación de los Datos

Cajeros

Para la creación de los cajeros, se tomó como referencia una capa de oficinas bancarias encontrada en ArcGIS Online. A la que se le cambiaron diversos campos y se le añadieron algunos otros. Los datos que se han mantenido de la capa original son; la dirección, el código postal, el teléfono, el fax etc. Y se han añadido otros campos como:

Horario	Campo de tipo cadena de texto que muestra el horario del cajero. Hay dos horarios diferentes, de 12 horas y de 24 horas. Se ha creado un dominio con los dos formatos de horario.
Cambio de 10	Campo de tipo texto que nos indica si el cajero tiene o no cambio de 10 euros.
Máximo de cambio permitido	Campo numérico que nos indica el dinero máximo que puede sacar un cliente.
Estado del cajero	Campo de tipo cadena de texto que indica el estado y la condición en la que se encuentra el cajero
Transferencias	Campo de tipo cadena de texto que indica si se pueden hacer transferencias en el cajero.

Ilustración 13. Tabla de campos de la capa Cajeros.

Todos estos campos e información serán muy útiles para el cliente, a la hora de consultar los cajeros que se encuentran más cerca de su posición.

Closest Facility (Model Builder)

Para que la aplicación calcule cual es el cajero más cercano además de la Red_Madrid es necesario tirar de un servicio que contenga el solucionador de rutas Closest Facility. La primera opción para crear el servicio fue la creación de un Modelo mediante Model Builder, con los siguientes pasos:

- Agrega el solucionador: desde las herramientas de Network Analyst, dándole los parámetros necesarios, como los acumuladores, las impedancias, y el modo de viaje.
- Agregar las ubicaciones: desde la herramienta de Network Analysist.Tanto los facilities que son en este caso los cajeros, como la incidencia, que va a ser la posición en la que se encuentre el cliente. Para esta última habrá que crear un shapefile de puntos, y localizarlo como ubicación ficticia del cliente, y así poder recoger este campo obligatorio de la herramienta.
- Resolver el solucionador, mediante la herramienta Solve.
- Seleccionar los datos: se seleccionará la ruta que dé como resultado del Solve.
- Copiar Entidades; Para que al final copie la ruta, y se le pueda dar otra simbología a la que viene por defecto.

Se parametrizará el resultado final, los cajeros (facilities) y la localización del cliente (Incidencia), para que a la hora de usar el widget de geoprocesamiento, los dos campos sean editables.

Por último, se le aplicará a la ubicación la propiedad de Feature Set, para que sea editable, y por lo tanto se pueda seleccionar la ubicación donde el cliente escoja.

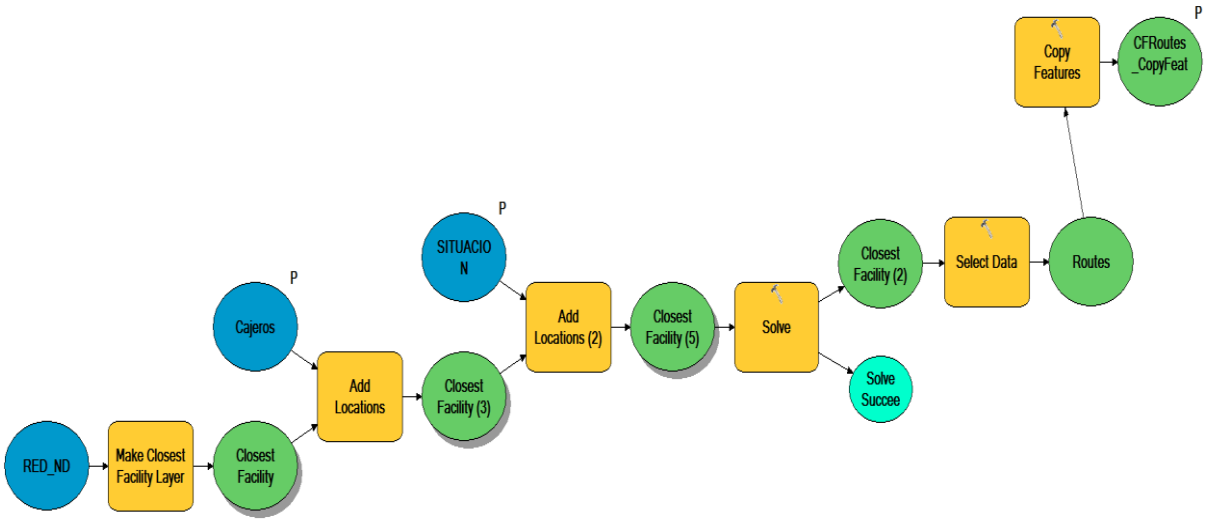


Ilustración 14. Model Builder creado para el cálculo del cajero más cercano.

Pero esta opción como antes se comentó se queda bastante corta, ya que el cliente solo podría ver únicamente un solo cajero como destino, y no elegir el rango de tiempo máximo para encontrarlo.

Closest Facility (NAserver)

Para que la aplicación pueda calcular cual es el cajero más cercano, es necesario crear una Red de la zona en la que se van a realizar los cálculos. Para ello se utilizó el callejero de Madrid, facilitado por Esri.

El primer paso fue definir las impedancias para los dos modos de viaje, a pie y en coche. Para el modo de viaje a coche no se definió ningún tipo de impedancia, ya que se pueden usar las predeterminadas de ESRI. Sin embargo, para el modo de viaje a pie se creó un campo en la capa de callejero, en el que indicaba que tipo de vías llevaban o no aceras. Seguidamente con la capa del callejero se creó un nuevo Dataset de red, al que se le agregaron los dos modos de viaje, con sus diferentes evaluadores de campo:

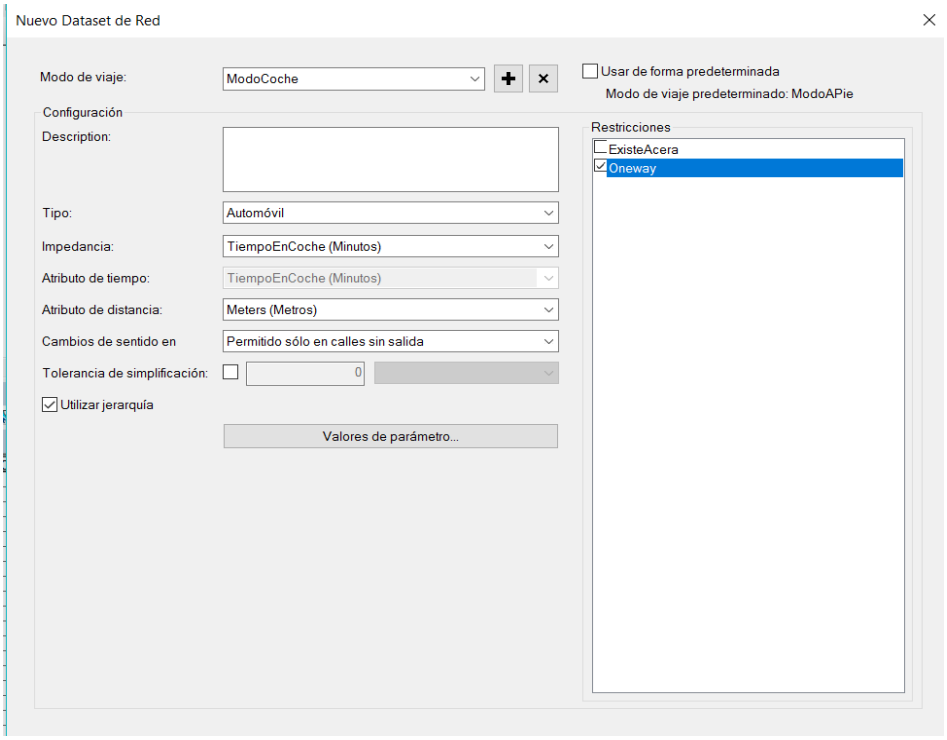


Ilustración 15. Detalle creación del 'modo a coche'.

Desde ArcMap se accede a la venta de NetworkAnalyst, en el que se agrega el solucionador de closest facilities, dando la restricción de Acera, y quitando la de sentido único, ya que lo que nos interesa publicar es el Modo a Pie. Para finalizar se publicará aplicando la capacidad de NA.

Al ser los cajeros datos estáticos y que por lo tanto que no van a sufrir ningún tipo de modificación por parte de la Organización, se ha decidido que los datos sean publicados como servicio de mapa en nuestra Cuenta de Organizaciones Bankgister, y publicada también en ArcGIS Server, haciendo una copia de los datos. Se ha publicado en esta última, para que el widget tire directamente de una referencia de Server, como se verá más adelante.

Para publicar los datos, antes se ha dotado a la capa de una escala acorde a los visores de internet, Google Maps, ArcGIS Online etc. Además de una escala visible.

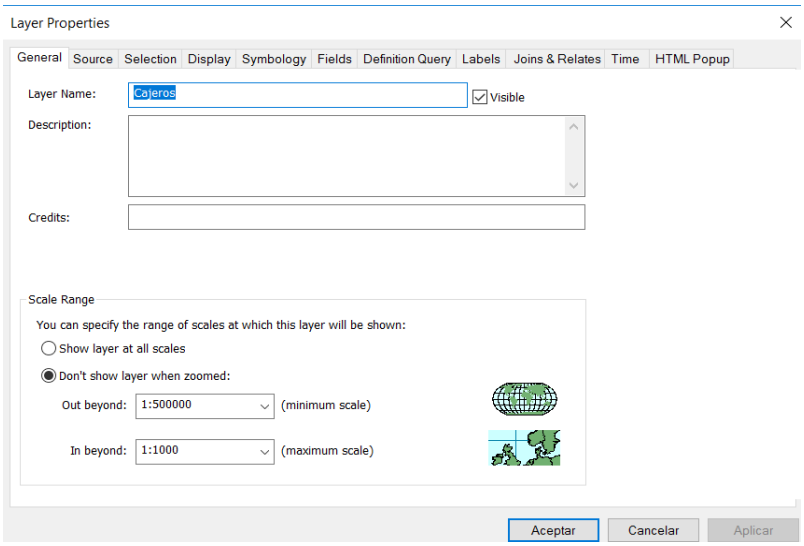


Ilustración 16. Especificación de la escala para compatibilidad con visores web.

Estos dos últimos procesos se repetirán en todos los datos, para tenerlos todos en la misma escala, y en la misma escala visible.

En cuanto a la Red de Madrid, con el solucionador de Closest Facilities habilitado, se publicará a ArcGIS Server, para tirar del servicio desde el widget de WebAppBuilder más tarde. Los datos se registrarán con la Geodatabase Bankgister.sde.

Closest Facility (NAServer)

Layer Type: esriNAServerClosestFacilityLayer

Impedance: TiempoAPie

Restrictions: Acera

Accumulate Attribute Names: Meters, TiempoAPie

Attribute Parameter Values:

- *attributeName:* Oneway
  parameterName: Uso con restricciones
  parameterType: float
  value: Prohibited
- *attributeName:* TiempoAPie
  parameterName: velocidadAPie
  parameterType: float
  value: 5
- *attributeName:* Acera
  parameterName: Uso con restricciones
  parameterType: float
  value: Prohibited

Snap Tolerance: 0

Max Snap Tolerance: 5000

Snap Tolerance Units: esriMeters

Ilustración 17. Servicio 'Closest Facility' alojado en ArcGIS Server.

2.2.3. Desarrollo y Resultados

Debido a la escasez de contenido editable por el widget de geoprocesamiento del que dispone ESRI en WebAppBuilder, fue necesario la búsqueda de una solución o referencia

para el desarrollo del Widget de Closest Facilities, encontrando estas dos referencias que podían servir de ayuda;

https://developers.arcgis.com/javascript/3/jssamples/routetask_closest_facility.html

https://geonet.esri.com/docs/DOC-7573

Con estos dos ejemplos se puede hacer un widget bastante más completo, ya que se pueden elegir el número de facilities que encontrar, con un máximo de hasta cuatro, indicar el tiempo máximo de trayecto e incluso dibujar barreras de tráfico, por las que no pasar.

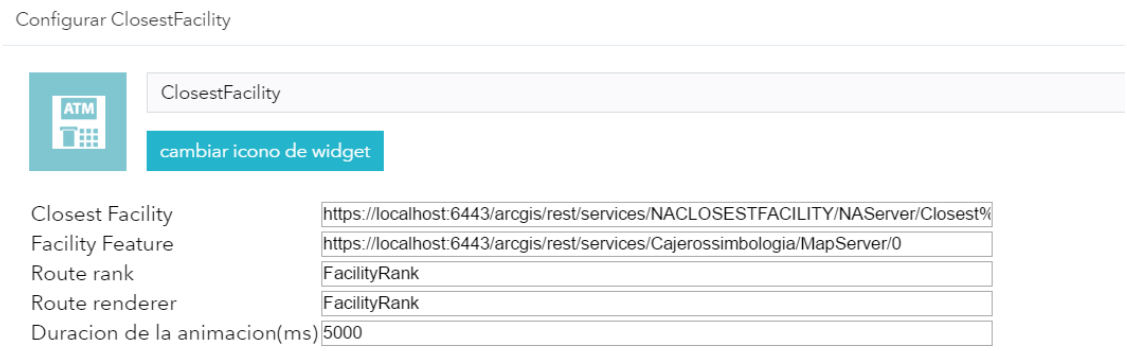


Ilustración 18. Configuración del servicio de ruta en el Widget.

El widget tiene una configuración, en la que la organización puede elegir sobre qué datos realizar el solucionador, y sobre que ruta. Lo que hace que sea un widget reutilizable para otras ciudades y sobre otros datos, aumentando así su utilidad, ya que nuestra Organización se encuentra en otros países como Francia o Portugal.

El resto de datos a introducir son: Rango de Ruta, el nombre del atributo de ruta que contiene el rango de resultado asignado por el Solvery Render de la Ruta; el atributo que el renderizador de valor único debe utilizar al dibujar los resultados.

Los dos primeros parámetros se dejarán tal y como se muestra en la imagen, en la que la capa de facilities, serán los cajeros anteriormente subidos, y la red el NAserver también subido con anterioridad. Para que el cliente tenga así todo preparado para calcular cual es el cajero más cercano de Bankgister.

Tanto la configuración del Widget como la animación de la polyline, se han mantenido, ya que resultan bastantes útiles para los intereses de la Organización.

Por último, se le ha añadido un Widget ya predeterminado de ESRI, como es el de direcciones, en el que el cliente podrá obtener las direcciones hacia el cajero que le

interese, además de poder seleccionar el modo de viaje en el que se desplaza, y así calcular con mayor exactitud el tiempo de recorrido.

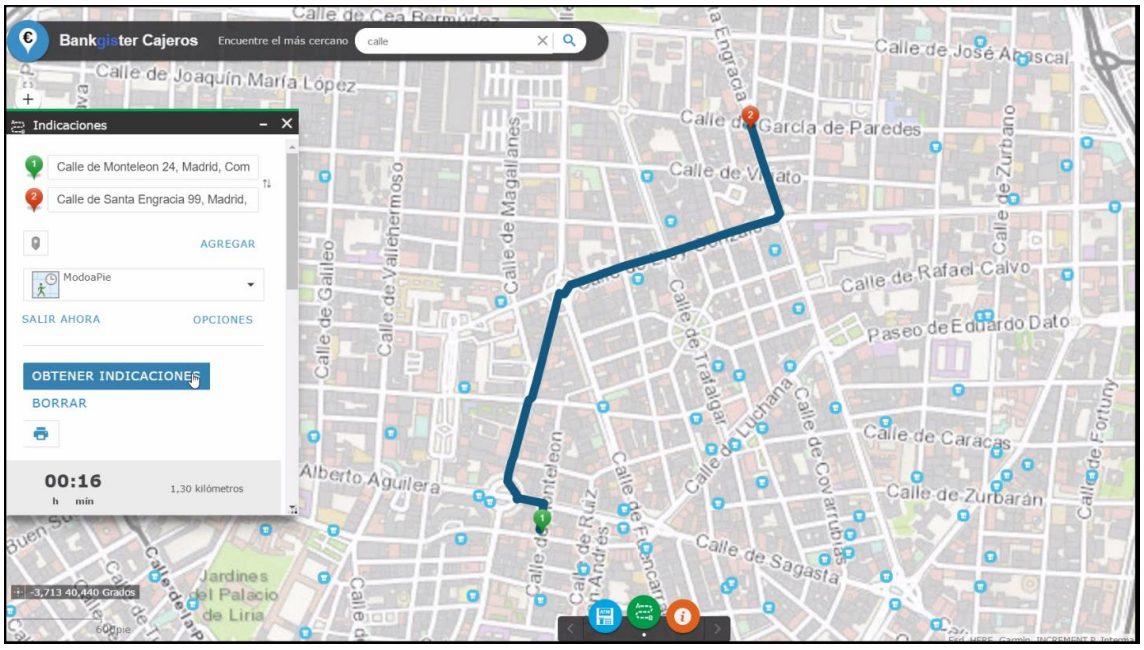


Ilustración 19. Visualización del widget 'Indicaciones' en la aplicación.

Como resultado queda una aplicación bastante intuitiva y fácil de usar por los clientes, que podrán mediante un simple click fijar su posición en el mapa, y calcular cuales son los cajeros más cercanos, para realizar la operación que se desee. Además, viendo las diferentes características de cada cajero, podrá decidir cuál es el que le conviene, y por lo tanto tomar la ruta que él desee. Además, si no se conoce la zona podrá buscar la dirección del cajero, mediante el Widget secundario de direcciones.

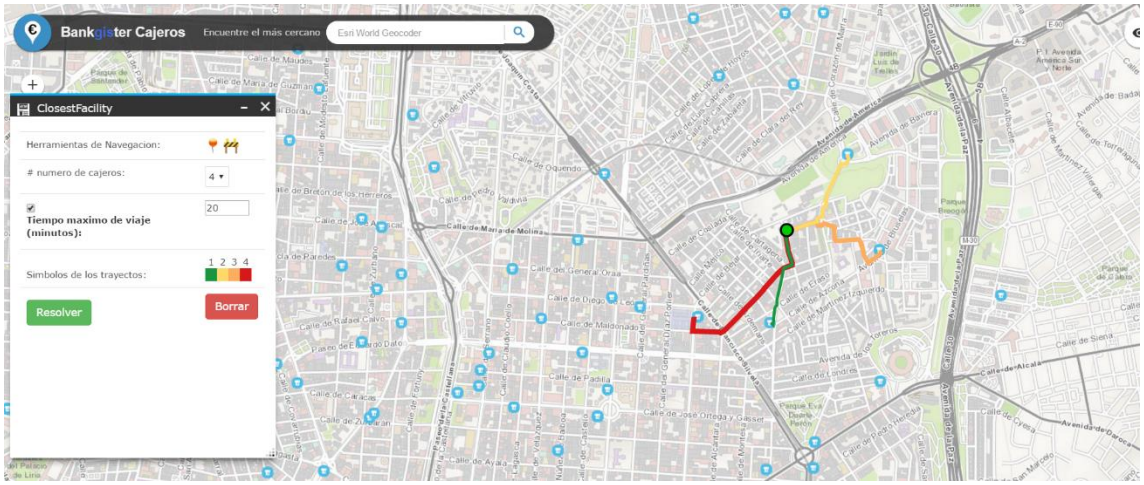


Ilustración 20. Visualización del widget creado 'ClosestFacility' en la aplicación.

2.3. Calcula tu hipoteca

2.3.1. Resumen de la aplicación

Una de las preguntas más populares que suelen hacerse muchas personas, es de dónde obtienen los bancos las casas o los pisos que ponen a disposición de las personas para ser comprados. Básicamente estas casas se obtienen por medio de las subastas, es decir, de los embargos que realizan las entidades financieras a aquellas personas que no pudieron cumplir con sus obligaciones económicas.

Las casas de bancos, aumentan cada día, esto se debe al incumplimiento de los pagos de las personas y las cuentas atrasadas sin haber cumplido incluso con un trato conciliador. Puede que a simple vista parezca una opción muy cruel de hacerse a una casa, pero la realidad es que cada día son más las ofertas y las facilidades que realizan los bancos para que sea más sencillo tener una casa.

Bankgister cuenta con viviendas en su propiedad y por ello ha querido crear su propia aplicación para ofertarlas a sus clientes. Pera esta aplicación está pensada no en solo ofrecer casas y ver sus características (dormitorios, baños, precio, imágenes, etc.), sino también en que el posible comprador pueda simular el posible proceso de su adquisición. Con la creación de un widget que simula el cálculo de la hipoteca, el cliente puede valorar sus opciones de compra ajustando a sus intereses y necesidades.

Por último, la aplicación incorpora una capa con la evolución del precio del suelo a lo largo de los últimos diez años. Datos que son muy interesantes, si el cliente quiere estimar si es el mejor momento para realizar la compra o si a la larga quiere vender su vivienda, para que pueda estimar si el precio del m² de su vivienda bajará o subirá en los próximos años.

Como resultado queda una aplicación, sencilla, fácil de manejar y bastante útil para los clientes que podrán realizar todos los cálculos que vean necesarios para la compra de su futura casa.

2.3.2. Tratamiento y Publicación de los Datos

Para la capa de **propiedades en venta**, al no disponer de ningún dato como referencia, se decidió crear una capa de puntos, que representarían las propiedades en venta, sobre la capa de Barrios_Madrid. Para generar esta capa de puntos se utilizó la herramienta ‘Crear puntos aleatorios’ dando como parámetro de salida el nombre de la capa ‘Propiedades_Venta’, y que se dibujase un punto sobre cada polígono de la capa de barrios.

El resultado de la capa se guarda en el Dataset de Datos_Bankgister aplicando las transformaciones de las coordenadas a WGS 1984 Web Mercator para su correcta relación con el resto de datos.

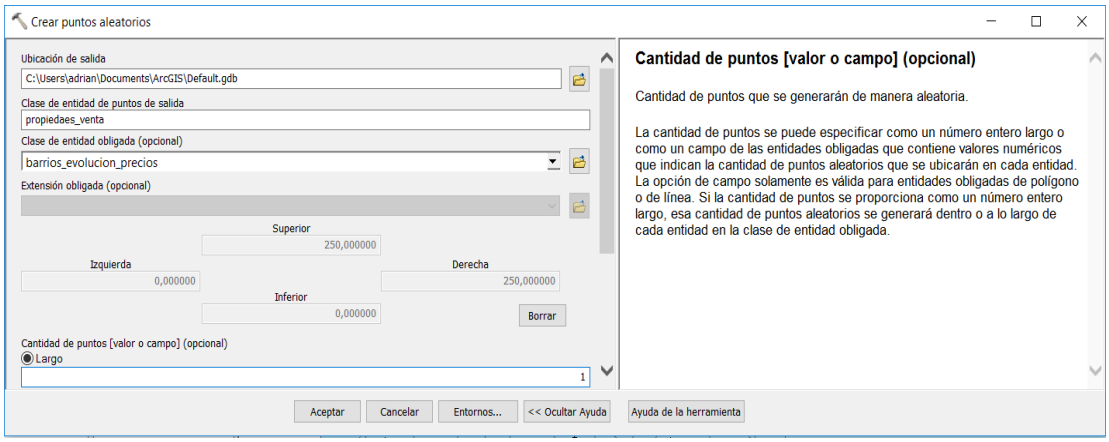


Ilustración 21. Propiedades en venta con la herramienta 'Crear puntos aleatorios'.

Por último, al tratarse de una capa vacía, únicamente con entidades de puntos, se le añadieron una serie de campos informativos sobre las propiedades en venta, para que los clientes puedan consultarlos:

Comisión	Campo con la comisión en porcentaje que tiene la vivienda a la hora de hacer un pago de hipoteca.
Metros cuadrados	Campo con los metros cuadrados totales de los que dispone la vivienda.
Habitaciones	Campo con el número de habitaciones del que dispone la vivienda. Se ha creado de manera aleatoria, y se le han establecido un dominio de rango entre 1-6 habitaciones.
Baños	Campo con el número de baños del que dispone la vivienda. Se ha creado de manera aleatoria, y se le ha establecido un dominio entre 1-6 baños.
Precio metro cuadrado	Precio del metro cuadrado de la vivienda, para que se han tomado los datos del calor del suelo del barrio en el que se encontraba la vivienda.
Precio total de la vivienda	Campo con el precio total de la vivienda, incluida la comisión. Se ha calculado multiplicando el precio del metro cuadrado de la vivienda por los metros de la misma, sumándole el porcentaje de la comisión.

Ilustración 22. Tabla campos de la capa 'Propiedades en venta'.

Por ultimo para complementar la información de las viviendas en venta, de cara a los clientes se han añadido una serie de imágenes por vivienda. Para ello se creó un

campo ráster, al cual se le fueron subiendo fotos una a una, pero estas fotos no eran visibles en ArcGIS Online, por lo que se utilizó el **método de Adjuntos**.

El primer paso fue activar los adjuntos en la capa de viviendas en venta, con lo que se creó una tabla de adjuntos y una clase de relación. El segundo y último paso fue abrir el editor e ir adjuntando las imágenes a las viviendas correspondientes, lo que dio como resultado final:

ATTACHMENTID *	REL_OBJECTID *	CONTENT_TYPE	ATT_NAME	DATA_SIZE	DATA
1	50	image/jpeg	01_salon-piso-venta-.jpg	118028	Blob
2	10	image/jpeg	67410310.jpg	24602	Blob
3	100	image/jpeg	80948507.jpg	16748	Blob
4	102	image/jpeg	82057800.jpg	20760	Blob
5	103	image/jpeg	105732415.jpg	19246	Blob
6	105	image/jpeg	118642703.jpg	17321	Blob
7	11	image/jpeg	123819197.jpg	11389	Blob
8	12	image/jpeg	125695212.jpg	17791	Blob
9	14	image/jpeg	133289197.jpg	20599	Blob
10	15	image/jpeg	133542277.jpg	13397	Blob
11	16	image/jpeg	136556308.jpg	17675	Blob
12	17	image/jpeg	139075230.jpg	15421	Blob
13	2	image/jpeg	140800130.jpg	14378	Blob
14	20	image/jpeg	142951913.jpg	20322	Blob
15	21	image/jpeg	164016518.jpg	11065	Blob
16	22	image/jpeg	164589526.jpg	18056	Blob
17	23	image/jpeg	164808011.jpg	17781	Blob
18	24	image/jpeg	164817741.jpg	15689	Blob
19	25	image/jpeg	174007750.jpg	14935	Blob

Ilustración 23. Incorporación de imágenes adjuntas a la capa 'Propiedades en venta'.

Como se quiere que esta capa pueda ser editada y manipulada por los clientes futuros, se subió como servicio a ArcGIS Server.

La otra capa de datos que necesitamos es la de la **evolución del precio del suelo**. El objetivo es que el cliente pueda ver la evolución que ha sufrido el precio del suelo, en donde se encuentra la vivienda que quiere adquirir, por lo que necesitábamos crear una tabla que reflejara el precio por m² por año de cada barrio de Madrid. En la página de Idealista se recoge anualmente esta variación por ciudades y más específicamente por barrios (info: https://www.idealista.com/informes-precio-vivienda).

	A	B	C	D	E	F	G
1	NOMDIS	NOMBRE	PRECIO_SUELO_2016	PRECIO_SUELO_2015	PRECIO_SUELO_2014	PRECIO_SUELO_2013	PRECIO_SUELO_2012
2	Arganzuela	Acacias	3193	3006	2804	2918	2957
3	Arganzuela	Atocha	n.d.	n.d.	n.d.	n.d.	n.d.
4	Arganzuela	Chopera	2656	2352	2190	2267	2379
5	Arganzuela	Delicias	3025	2624	2564	2710	2825
6	Arganzuela	Imperial	2920	2747	2731	2752	2816
7	Arganzuela	Legazpi	3395	3272	3307	3428	3428
8	Arganzuela	Palos de Moguer	2972	2799	2529	2800	2831
9	Barajas	Aeropuerto	n.d	n.d	n.d	n.d.	n.d.
10	Barajas	Alameda de Osuna	2602	2576	2612	2787	2817
11	Barajas	Casco Histórico de Barajas	n.d	2194	1991	n.d.	n.d.
12	Barajas	Corralejos	3331	3298	3426	3434	3640
13	Barajas	Timón	n.d	2472	2433	n.d.	2703
14	Carabanchel	Abrantes	1466	1546	1359	1647	1718
15	Carabanchel	Buenavista	1482	1356	1438	1741	1956
16	Carabanchel	Comillas	1805	1822	1676	1883	2046
17	Carabanchel	Opañel	1721	1620	1641	1804	1921
18	Carabanchel	Puerta Bonita	1510	1373	1420	1603	1681
19	Carabanchel	San Isidro	1665	1646	1566	1881	1949
20	Carabanchel	Vista Alegre	1593	1496	1508	1719	1779
21	Centro	Cortes	3616	3831	3541	3358	3715
22	Centro	Embajadores	3240	3019	2697	2897	2721
23	Centro	Justicia	4612	4284	3902	3969	4007
24	Centro	Palacio	3731	3776	3369	3397	3519
25	Centro	Sol	3788	3663	3419	3445	3517
26	Centro	Universidad	3845	3468	3199	3297	3230
27	Chamartín	Castilla	3165	3178	3015	3123	3300
28	Chamartín	Ciudad Jardín	3416	3028	2839	3182	3235
29	Chamartín	El Viso	4919	4562	4459	4482	5021
30	Chamartín	Hispanoamérica	4047	3811	3765	3804	4289

Ilustración 24. Datos evolución del suelo por Idealista.

Los datos están comprendidos entre 2007 y 2016, y mediante la información ofrecida por los documentos de Idealista, pasamos los datos a mano a nuestra tabla Excel, ya que no se podían descargar.

Una vez creado el Excel, se guardará en la Geodatabase de Bankgister, para ello, hay que usar la herramienta de conversión 'de Excel a Tabla':

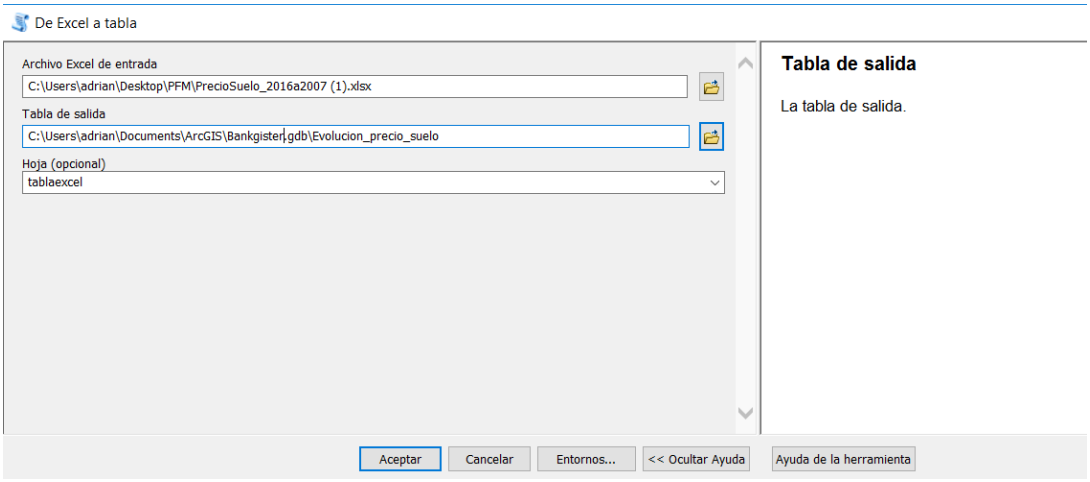


Ilustración 25. Herramienta conversión datos Excel a tabla.

Pero el proceso no está terminado, ya que queda por unir estos datos guardados en una tabla a la capa de Barrios_Madrid. Para ello se va a utilizar el método de unión; unión entre la capa de Barrios_Madrid y la tabla de la Evolución del precio del suelo, mediante el campo en común, nombre.

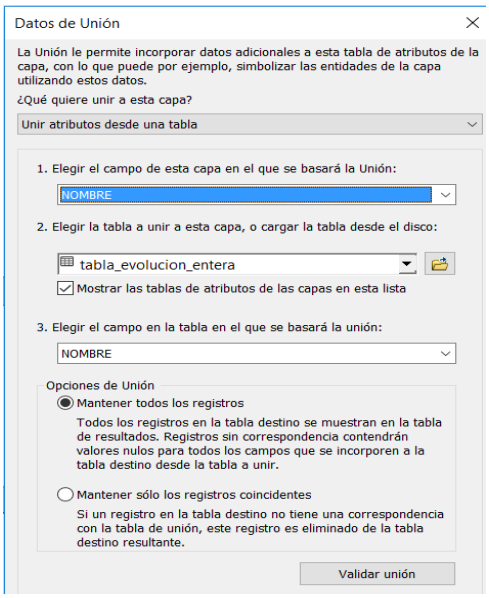
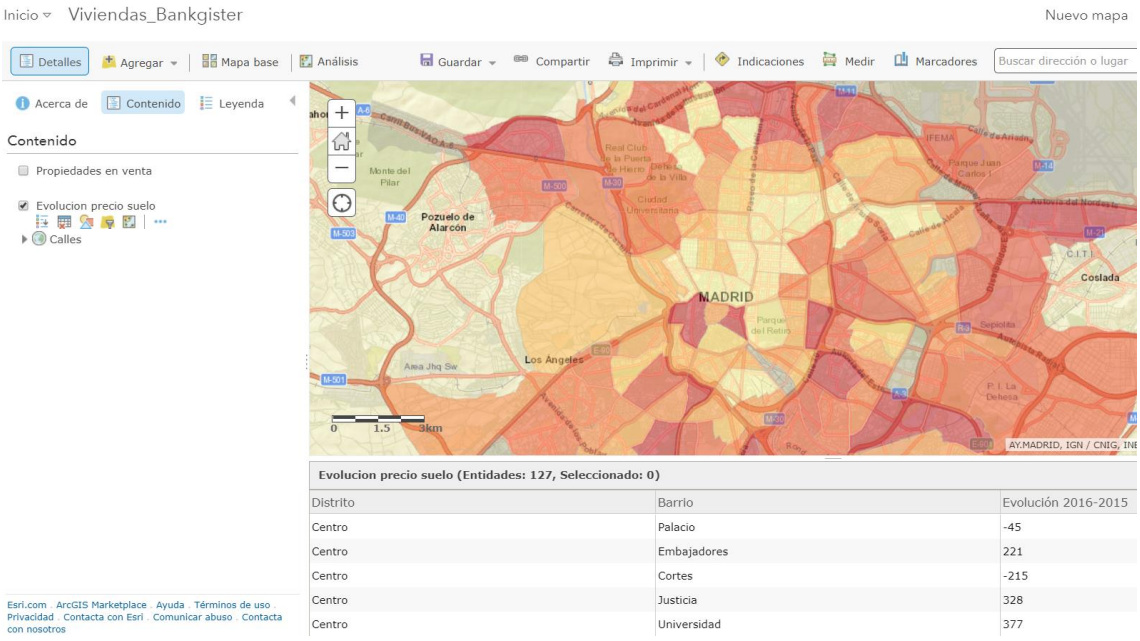


Ilustración 26. Unión de la tabla 'evolución del suelo' a la capa de 'Barrios de Madrid'.

Una vez realizada la unión de datos, para guardarlos, ya que se encuentran en memoria, se exportan los datos de la capa y se crea una nueva capa llamada **Evolucion_Precio_Suelo**.

Al tratarse de un servicio estático, que no requerirá ningún tipo de edición, se publica en ArcGIS Online.

Por último, para que la capa aparte de tener los datos de los últimos diez años tenga una representación en simbología en nuestra aplicación, se le ha añadido un campo en el que se ha calculado la diferencia entre el año actual y el anterior, mostrando así la variación del precio del suelo y permitiendo dar una representación simbólica visualmente atractiva.



Evolucion precio suelo (Entidades: 127, Seleccionado: 0)

Distrito	Barrio	Evolución 2016-2015
Centro	Palacio	-45
Centro	Embajadores	221
Centro	Cortes	-215
Centro	Justicia	328
Centro	Universidad	377

Ilustración 27. Incorporación simbología a la capa 'Evolución del suelo' con ArcGIS Online.

2.3.3. Desarrollo y resultados

Antes de incorporar en el código el cálculo de la simulación de la hipoteca se hizo una simulación en una hoja de cálculo siguiendo el método de amortización francés:

	Mes	Intereses	Amortizado		Total pagado	Pendiente de pago
0	-	-	-	-	-	125.400,00
Año 1	1	365,75	361,52	727,27	727,27	125.038,48
	2	364,70	362,57	727,27	1454,54	124.675,91
	3	363,64	363,63	727,27	2181,81	124.312,28
	4	362,58	364,69	727,27	2909,08	123.947,58
	5	361,51	365,76	727,27	3636,35	123.581,83
	6	360,45	366,82	727,27	4363,62	123.215,00
	7	359,38	367,89	727,27	5090,89	122.847,11
	8	358,30	368,97	727,27	5818,16	122.478,15
	9	357,23	370,04	727,27	6545,43	122.108,11
	10	356,15	371,12	727,27	7272,69	121.736,98
	11	355,07	372,20	727,27	7999,96	121.364,78
	12	353,98	373,29	727,27	8727,23	120.991,49
Año 2	1	352,89	374,38	727,27	9454,50	120.617,11
	2	351,80	375,47	727,27	10181,77	120.241,65
	3	350,70	376,56	727,27	10909,04	119.865,08
	4	349,61	377,66	727,27	11636,31	119.487,42
	5	348,50	378,76	727,27	12363,58	119.108,65
	6	347,40	379,87	727,27	13090,85	118.728,78
	7	346,29	380,98	727,27	13818,12	118.347,81
	8	345,18	382,09	727,27	14545,39	117.965,72
	9	344,07	383,20	727,27	15272,66	117.582,52
	10	342,95	384,32	727,27	15999,93	117.198,20

A continuación podemos observar la **forma de calcular la cuota total y los tipos de interéses en l amortizar:**

$$C = V \cdot \frac{(1+i)^n \cdot i}{(1+i)^n - 1}$$

- **C** = Cuota a pagar
- **V** = Cantidad del préstamo hipotecario
- **i** = tipo de interés del período
- **n** = número de cuotas

$$I_{(p-1,p)} = i \cdot V_{(p-1)}$$

- **I (p - 1, p)** = tipo de interés del período
- **i** = tipo de interés
- **V (p - 1)** = Capital del préstamo hipotecario pendiente de amortizar

Ilustración 28. Cálculo en Excel del método de Amortización Francés.

Con el método aplicado será posible el posterior cálculo de la hipoteca, para ello primero se ha ejecutado también en un fichero Excel para comprobar su cálculo correcto.

Localización	Madrid	
Condición	Nuevo	Segunda mano
(Ahorro mínimo 20%+gastos)	>=34%	>=29%
Precio del inmueble	162000	
Ahorro aportado	48600	30%
Tipo de interés	3,50%	
Plazo en años	20	0-40
Tu cuota mensual	727,27	
Importe hipoteca	125400	
Coste total de la compra	174000	
Precio del inmueble	162000	
Impuestos y gastos	12000	
Gastos de la compra + Gastos hipoteca (Notaria, registro, impuestos..)		
Total con hipoteca	223144,68	
Ahorro aportado	48600	
Importe hipoteca	125400	Total amortizado
Interés hipoteca	49144,68	Total intereses

Ilustración 29. Cálculo de una hipoteca en Excel.

Los siguientes cálculos son aquellos que acoplamos en el código recogiendo los elementos de la capa de propiedades y realizando sus correspondientes operaciones.

Ilustración 30. Apariencia del cálculo de la hipoteca en el widget 'Hipoteca'.

Este resumen es exactamente lo que queremos que realice nuestro widget de Hipoteca, aplicando un diseño atractivo e interactuando con las capas.

Esto en formato código equivale a la siguiente estructura incorporada en el elemento 'on_click' del widget Hipoteca:

```
propiedades.on("click", function(evt){
  var direccion = evt.graphic.attributes.Direccion;
  var precio = evt.graphic.attributes.Precio_total;
  var condicion = evt.graphic.attributes.Condicion;
  var interes = evt.graphic.attributes.Interes;

  var total = parseFloat(precio) + parseFloat(gastos.value);
  var ahorro = (parseFloat(precio) * parseFloat(ahorro2.childNodes[0].data))/100;
  var amortizado = total - ahorro;
  var cuota = parseFloat((amortizado*(interes/100)/12)/(1-(Math.pow((1+((interes/100)/12)),(-parseFloat(plazo2.childNodes[0].data)*12))))).toFixed(2);

  document.getElementById("direccion2").innerHTML = direccion;
  document.getElementById("precio2").innerHTML = precio + "€";
  document.getElementById("condicion2").innerHTML = condicion;
  document.getElementById("comision2").innerHTML = interes + "%";

  // document.getElementById("total2").innerHTML = total + "€";
  document.getElementById("ahorro4").innerHTML = ahorro + "€";
  document.getElementById("amortizado2").innerHTML = amortizado + "€";
  document.getElementById("cuota2").innerHTML = cuota + "€";

  //Calculo amortización para obtener el total de los Intereses
  var pendiente = amortizado;
  var total_intereses = 0;
  var intereses, amortiz;

  while (pendiente > 0){
    intereses = (interes/100)*pendiente/12;
    amortiz = cuota - intereses;
    pendiente = pendiente - amortiz;
    total_intereses += intereses;
  }
  document.getElementById("intereses2").innerHTML = parseFloat(total_intereses).toFixed(2) + "€";
```

Ilustración 31. Código implementado para el cálculo de la hipoteca.

Como podemos comprobar en las siguientes imágenes la aplicación funciona correctamente y visualmente se ha conseguido el estilo deseado, un estilo atractivo y sencillo de comprender para los usuarios.

El primer widget de la aplicación es mera información de la utilidad de la aplicación.

El segundo widget recoge el listado de capas operativas, en este caso en una primera visualización se dispone de la 'Evolución_precio_suelo' la cual proporciona al usuario información de los distintos barrios de Madrid. Más en detalle seleccionando el barrio deseado se puede realizar una consulta de como a cambiado dicho valor en los últimos 10 años mediante una gráfica.

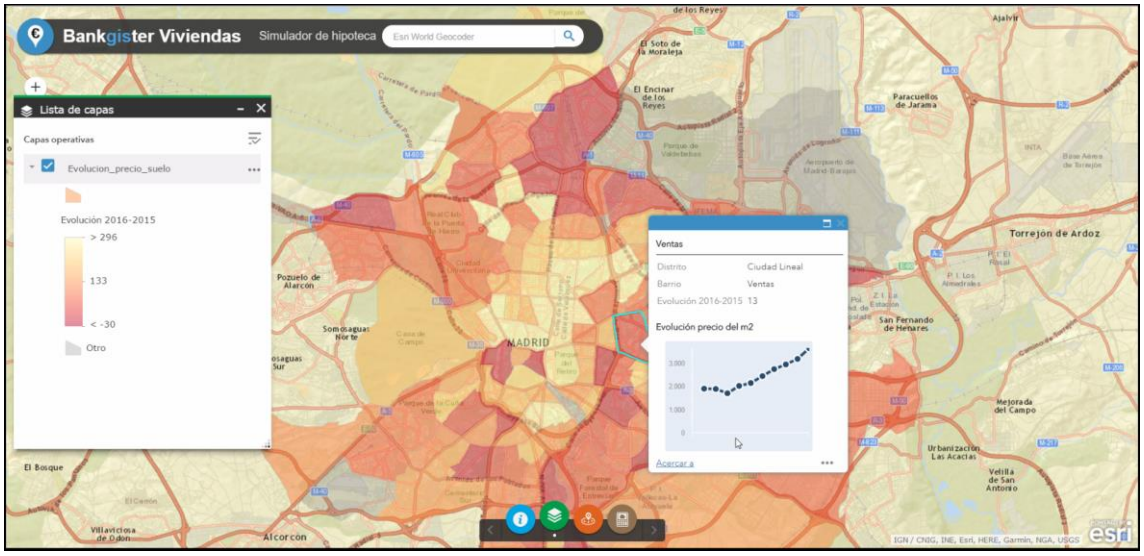


Ilustración 32. Visualización de la evolución del suelo en la aplicación.

Mediante el tercer widget disponible por Esri, es posible realizar una búsqueda según una ubicación y un radio determinado por el usuario, de las viviendas disponibles en esa área solicitada. Además de obtener los campos informativos de la propiedad en venta y una visualización de las posibles fotografías adjuntas.

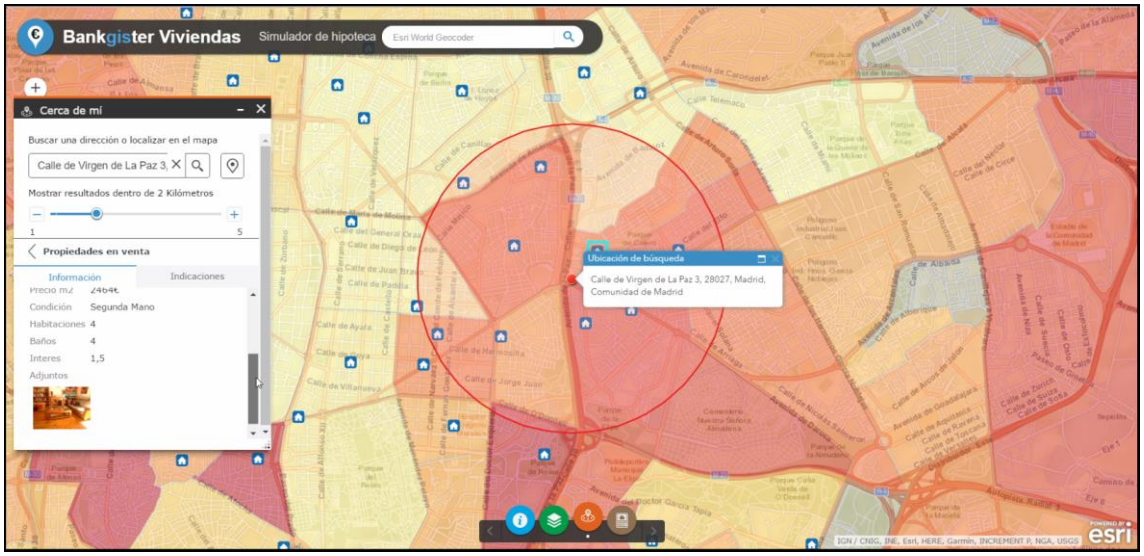


Ilustración 33. Visualización de las características de la propiedad en venta.

Finalmente, el último widget es el de 'Hipoteca', desarrollado enteramente para esta aplicación con el objetivo de proporcionarle al usuario una manera dinámica de estimar el valor de una posible hipoteca según que propiedad sea de su gusto.

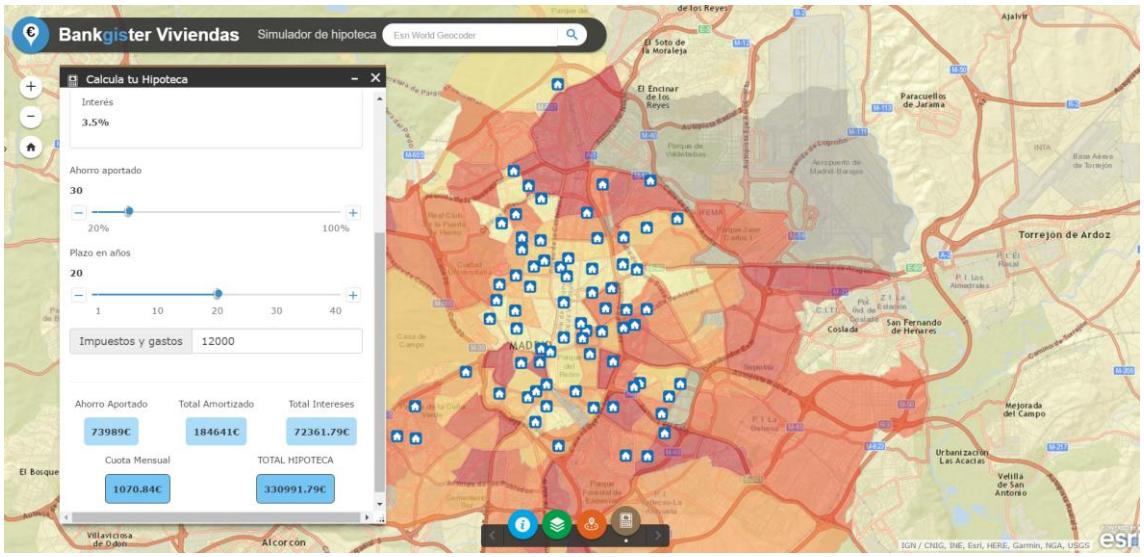


Ilustración 34. Visualización del widget 'Hipoteca' en la aplicación.

Se aprecia también la capa de evolución del suelo mencionada anteriormente y otros tres widgets del WebAppBuilder que como se explican su funcionamiento en el video permite al cliente explotar y consultar más características en la aplicación.

2.4. Ubicación de Oficinas

2.4.1. Resumen de la Aplicación

Esta aplicación está pensada para que sea un producto para que sea utilizado por cualquiera de los miembros de la entidad bancaria Bankgister, en la que puedan ver la localización actual de todas sus oficinas, y toda su información, como puede ser; la localización, el teléfono, horario, el número de empleados, los clientes suscritos, y los ingresos medios a lo largo de los diez últimos años.

Esta Aplicación agilizará y facilitará el contacto entre oficinas, y será de gran utilidad para conocer el estado financiero de las mismas, ya que pensamos que es indispensable en una entidad bancaria estar al tanto de todas sus oficinas, y comprobar cómo funcionan financieramente hablando.

Pero la otra gran utilidad que tiene esta aplicación además de aportar una información muy valiosa, es que la entidad bancaria a través de la App podrá realizar simulacros, para ubicar o quitar oficinas, viendo cómo afectaría a sus clientes. Para realizar este proceso, se ha añadido una capa de puntos simulando los clientes repartidos por la ciudad de Madrid. El usuario podrá realizar un análisis de áreas de servicio sobre la zona que el seleccione, apreciando así entonces como quedaría la repartición de sus clientes, respecto de sus oficinas. El área de servicio creará dos polígonos, uno en el que sitúa los clientes que se encuentran a cinco minutos

andando, y los que se encuentran en diez minutos andando, se dibujarán en otro polígono, lo que facilitará la posibilidad de abrir o cerrar alguna oficina en función de cómo afecte las áreas de servicio a sus clientes. Si hay alguna zona con muchos clientes, en los que no se encuentra una oficina en el rango que marca el área de servicio, interesaría ubicar allí una nueva oficina, pero si, por el contrario, el rango del área de servicio de la oficina, no tiene casi clientes, interesaría cerrar dicha oficina.

El usuario también tiene disponible una capa que muestra un mapa de calor de los clientes para aún más ayudar a la toma de decisiones.

El InfoWindow está deshabilitado para el mapa de calor de clientes (para que cuando hagan click para calcular el service area, no salta el infowindow sobre encima), pero está habilitado para la capa de clientes normal por si quieren saber los detalles de un cliente.

Por último, el usuario de la organización verá además reflejado en pantalla, el tipo de clientes que hay en el área de servicio que seleccione, ya que tirará de los datos de una capa con cuatro campos enriquecidos;

- 2014 Unemployed Population
- 2015 Total Population age 15-29
- 2015 Total Population age 30-44
- 2015 Total Population age 45-59
- 2015 Total Population age 60+

2.4.2. Tratamiento y Publicación de los Datos

Oficinas

Para la creación de las oficinas, se ha utilizado una capa como referencia encontrada en ArcGis Oline en la que vienen representados más de 600 oficinas bancarias y cajeros distribuidos por la ciudad de Madrid. Una vez añadida a Desktop, se realizó una selección por localización. Seleccionando las entidades de puntos que se encuentran dentro de la capa de polígonos de Barrios_Madrid.

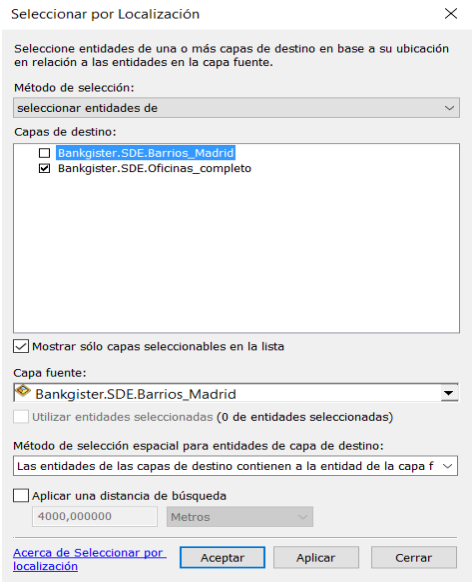


Ilustración 35. Incorporación de capas en la base de datos SDE.

Para complementar la app, se ha añadido una serie de campos informativos a la capa Oficinas, que serán de ayuda para la Entidad Bancaria a la hora de disponer de información útil y actualizada de todas sus oficinas.

Los campos que se han creado son los siguientes:

- Número de empleados; Es el número de empleados que tiene cada oficina, con un mínimo de cinco personas, y un máximo de veinte.
- Clientes suscritos; Número de clientes que están suscritos de forma fija a cada oficina, siendo un dato importante a tener en cuenta a la hora de cerrar oficinas.
- Dinero medio de las transacciones: Dato estadístico, de la media de dinero en transacciones registrada en el último año.
- Horario: El horario que tiene la oficina. Al existir únicamente dos horarios; 8:00-14:00(mañana) o 8:00- 20:00(completo), se ha creado un dominio para este campo con los dos valores anteriores.
- Disponibilidad de cajero: Si la oficina dispone o no de cajero. Al ser únicamente posible “SI” o “NO” se ha creado un dominio con estos dos valores.
- Ingreso medio desde 2016 a 2006: diez campos con los ingresos por año de cada oficina. Se han creado estos campos con Python, desde la ventana de ArcPy:

```
outName = "clientes"


env.workspace = "C:\Master\PFM\Bankgister.gdb"


fieldInt = "CLIENTES_SUSCRITOS
fieldFlt = "DINERO_TRANSACCIONES"
arcpy.AddField_management(outName, fieldInt, "LONG") # add long integer field
arcpy.AddField_management(outName, fieldFlt, "LONG") # add float field

Calculate random values between 1-100 in the new fields
arcpy.CalculateField_management(outName, CLIENTES_SUSCRITOS, "random.randint(1,10000)","PYTHON","import random")
arcpy.CalculateField_management(outName, DINERO_TRANSACCIONES, "random.uniform(1,200)","PYTHON","import random"))
```

Ilustración 36. Script de Python para el cálculo aleatorio del campo 'ingreso medio'.

Clientes

Para la creación de los clientes decidimos apoyarnos en la capa de manzanas de los barrios de Madrid que íbamos a utilizar. Previamente se ha realizado una limpieza de manzanas, esto es, todas aquellas que representan zonas de vegetación, recintos deportivos, espacios públicos, etc. De este modo respetamos la idea de que los clientes se ubican en sus viviendas o en sus puestos de trabajo.

Ilustración 37. Visualización de las parcelas donde habrá clientes.

El método por realizar en esta capa de manzanas de Madrid simplificada, crear un campo 'clientes' y mediante la función random añadirle valores 0 y 1. Haciendo uso de la ventana de Python de ArcMap se realizaron los siguientes pasos:

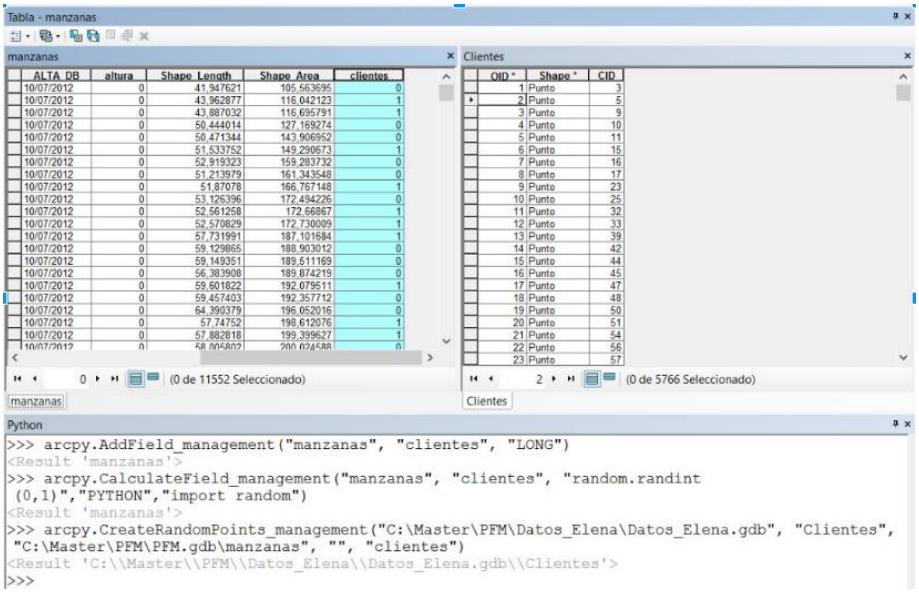


Ilustración 38. Script de Python empleado para la creación de clientes.

La primera línea de código añade un nuevo campo de ‘clientes’ a la capa de manzanas.

Una vez creado este nuevo campo calculamos sus valores con la función ‘random.randit’, le queremos dar valores aleatorios de 0 y 1 para indicar que entidades representarán que tienen un cliente y cuáles no.

Para terminar la última línea de código recoge este nuevo campo ‘clientes’ de la capa de ‘manzanas y crea una nueva capa de puntos de clientes, creando solo en aquellos polígonos donde su valor era 1.

Ilustración 39. Resultado de la capa de clientes realizada.

En el mundo real, el banco tendría creada una capa completa con una variedad de campos sobre sus clientes, pero para la funcionalidad de nuestras aplicaciones no hacen falta, así que de momento la capa de clientes solo viene con un campo de identificador junto con su shape.

Geoenrichment

Para un análisis detallado de la ciudad de Madrid, y conocer mejor el terreno sobre el que se va a colocar una nueva oficina, es necesario disponer de datos estadísticos que interesen para una apertura futura de estas, como pueden ser; la edad de la población, el gasto, los ingresos etc.

Es por ello por lo que se decide crear un web map, en el que se integren diversos datos estadísticos de interés, para posteriormente poderlo integrar en cualquiera de

nuestras aplicaciones. Siendo mayormente aplicable, en la aplicación de apertura de oficinas.

Como base para el mapa web, se añadió desde ArcMap, un feature service de Esri_Madrid, mediante el empleo de la herramienta add data from ArcGis Online.

Esta capa se añadió a Desktop y no a AGOL, ya que se realizarán diversos tratamientos de datos sobre ella.

El primer paso con esta capa fue seleccionar los barrios que nos interesaban, dejando fuera algunos, como es el caso del Pardo, que al tratarse de una zona mayoritariamente verde y no haber edificaciones, no nos interesaba a la hora de colocar las oficinas de la entidad bancaria, dejando como resultado esta capa al exportar los datos seleccionados a la geodatabase del Proyecto;

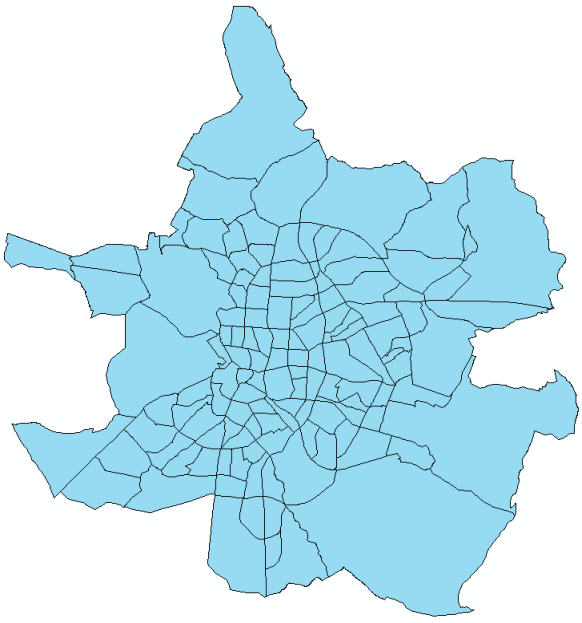


Ilustración 40. Capa de barrios de Madrid.

Seguidamente, se importa la capa al Dataset “Organización”, que tiene como sistema de coordenadas el WGS 1984 Web Mercator Auxiliary Sphere, el mismo en este caso que la propia capa, no siendo necesario ningún tipo de transformación geográfica.

Una vez ubicado el servicio en AGOL, se utilizarán los geoprocesamientos listos para usarse de los que se dispone en la plataforma. Se usará el enriquecimiento de datos, ya que se quiere enriquecer la capa de barrios con datos estadísticos de importancia para la apertura de nuevas oficinas bancarias.

Investigando todos los datos de los que dispone Esri en ArcGis Online para enriquecer las capas, se ha decidido escoger 5 variables, que se consideran muy importantes a la hora de tomar decisiones por parte de Bankgister, en la apertura o cierre de oficinas, con los que se han creado campos en una capa, partiendo todas de un mismo origen; Barrios_Madrid. Los campos son los siguientes:

Desempleados: campo creado añadiéndole la variable "2014 unemployed population" a Barrios_Madrid. Lo que nos aporta importantes datos, para conocer los barrios en los que más población desempleada hay, y por lo tanto no favorables para la apertura de una nueva entidad bancaria.

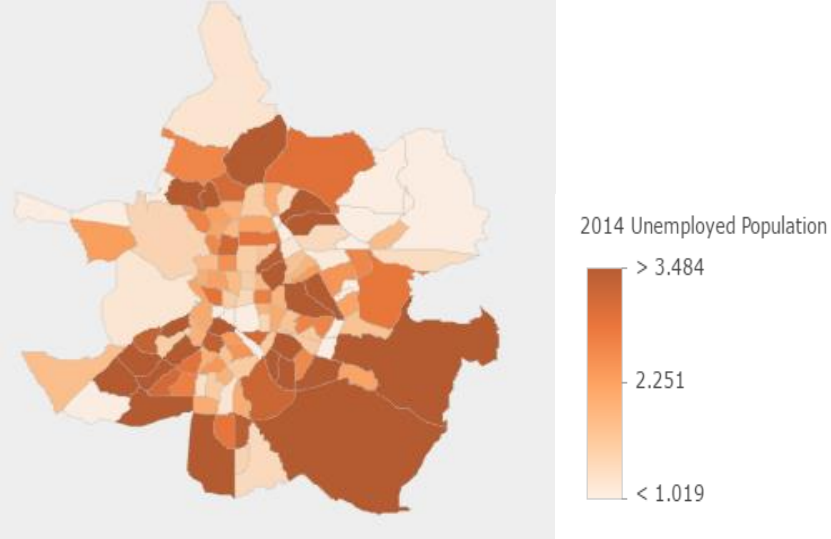


Ilustración 41. Población desempleada en los barrios de Madrid.

Poblacion_Joven: campo creado añadiéndole la variable "2013 total population age 25-29 years" a Barrios_Madrid. Lo que nos aporta importantes datos para conocer los barrios en los que más población pensionista hay. Se ha escogido esta variable, ya que, a partir de 25 años, es cuando se finalizan los estudios y se emprende la búsqueda de un empleo.

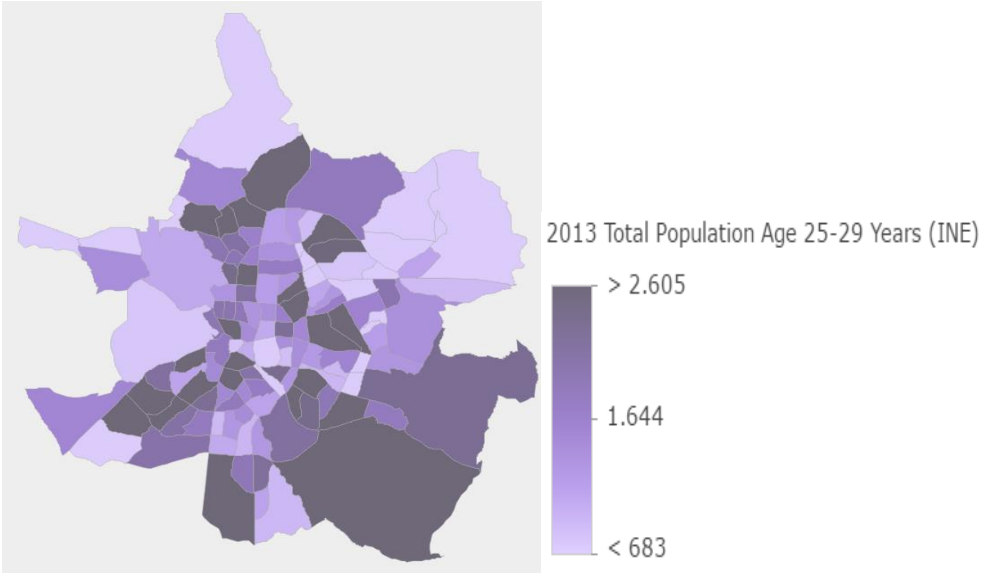


Ilustración 42. Población de 25-29 años en los barrios de Madrid.

Pensionistas: campo creado añadiéndole la variable "2015 total population age 60+ years "a Barrios_Madrid. Lo que nos aporta importantes datos para conocer los barrios en los que más población pensionista hay. Se ha escogido esta variable, ya que la población pensionista puede ser una importante variable a tener en cuenta, a la hora de futuros negocios.

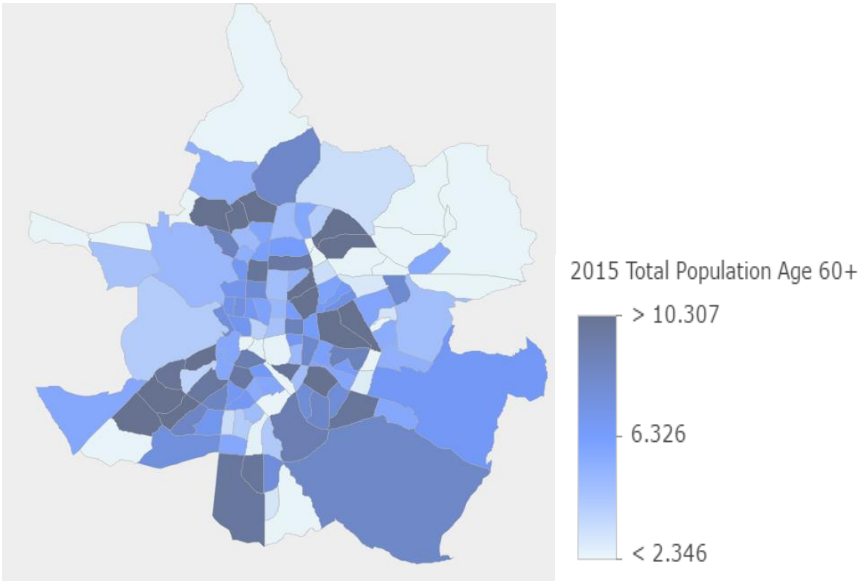


Ilustración 43. Total población mayor de 60 años en los barrios de Madrid.

Ingresos_por_año: campo creado añadiéndole la variable “2012 average household income per year” a Barrios_Madrid. Lo que nos aporta importantes datos, para conocer los barrios en los que la población tiene más ingresos por año. Dato estadístico muy importante a la hora de nuevas aperturas de oficinas.

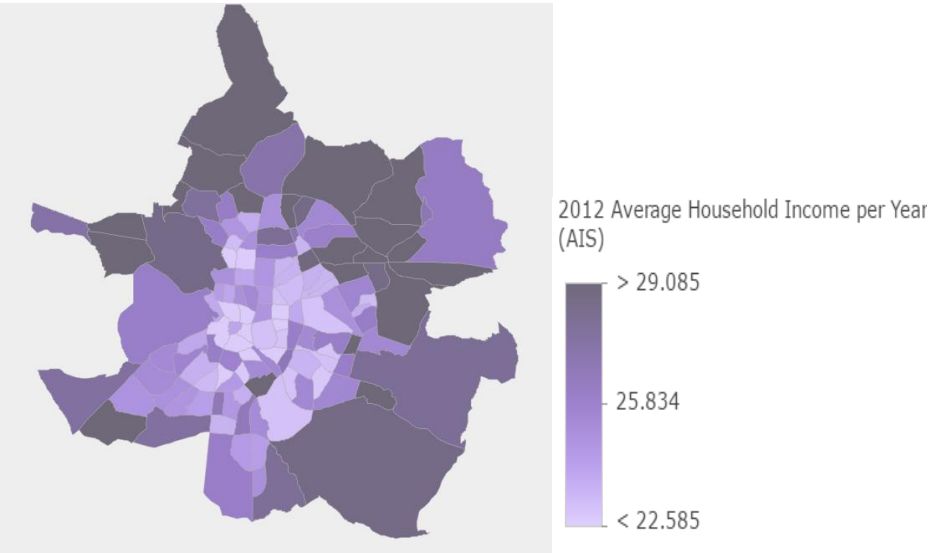


Ilustración 44. Ingreso medio por año en los barrios de Madrid.

Gasto_total: campo creado añadiéndole la variable “Gasto total año 2015” a Barrios_Madrid. Lo que nos aporta importantes datos, para conocer los barrios en los que la población realiza mayores gastos. Dato estadístico muy importante a la hora de nuevas aperturas de oficinas, como el anterior.

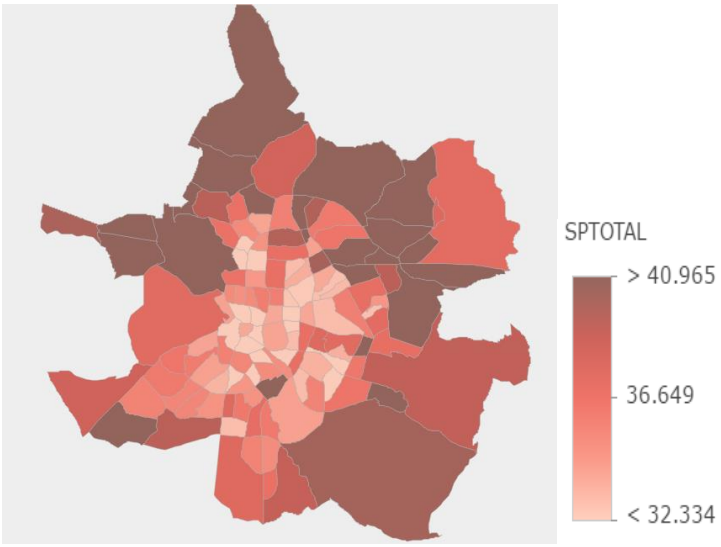


Ilustración 45. Gastos totales por año en los barrios de Madrid.

Ilustración 46. Fuente de ArcGIS Online para extracción de datos.

Service Area

El servicio para crear el service area fue creado a partir del Network Analyst, con el dataset de RED de madrid.

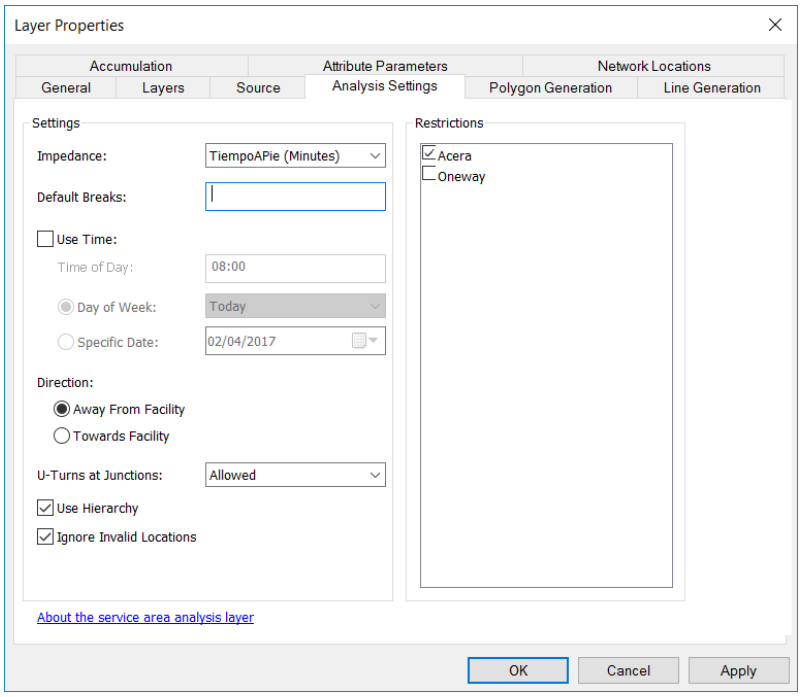


Ilustración 47. Creación del Service Area.

La impedancia elegida es tiempoapie, con un valor de 5km/h. La restricción de Acera fue creada a partir de un campo creado en la tabla de atributos, simplemente asignando un valor con el field calculator de 'Si' o 'No' a cada carretera / calle / vía.

Todos los subtipos tienen el valor ‘Y’ (si) menos ‘Carreteras’ que tienen el valor ‘N’. En este caso, la restricción oneway está deshabilitada porque no aplica a los peatones.

Antes de publicar el servicio, quitamos los defaults breaks, para que podamos luego meterlos como parámetros tanto en el widget.js como en el config del widget.

Probamos la funcionalidad del servicio antes de publicarlo, usando default breaks y ubicaciones distintas.

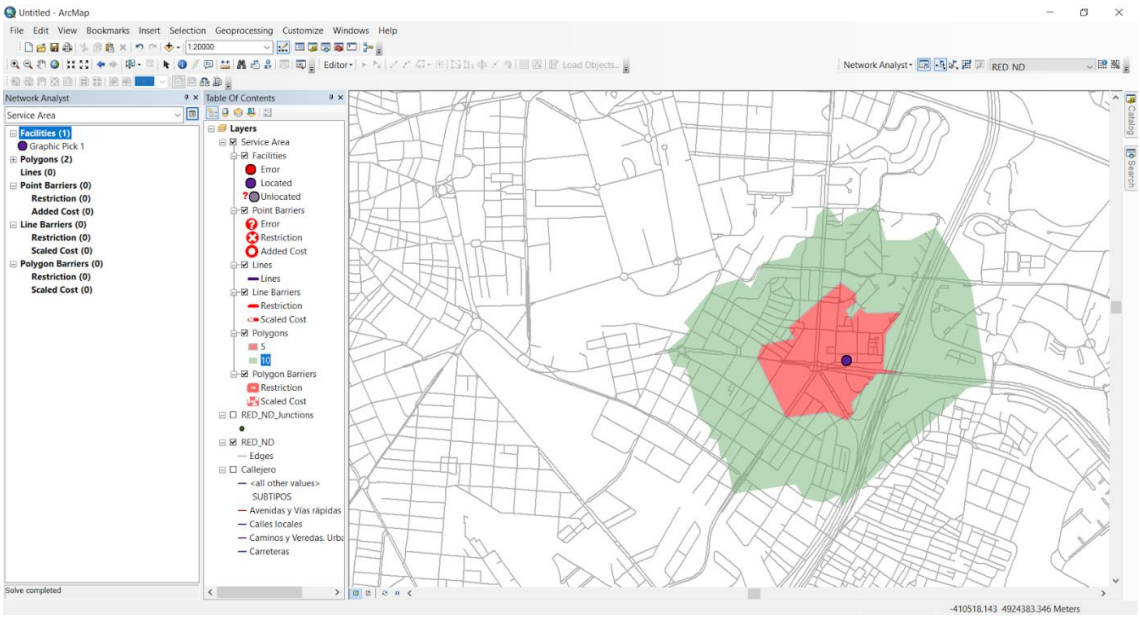


Ilustración 48. Comprobación funcionalidad Service Area.

Los polígonos son generalizados, que permite al usuario visualizar la zona afectada de una forma más general, es decir, aquellos clientes que, aunque por poco tiempo no entran exactamente, pero por distancia les conviene acudir a dicha oficina. Este tipo de polígono es más flexible que el detallado, a nuestro banco le interesa más en este caso el concepto genérico que entrar tanto en detalle.

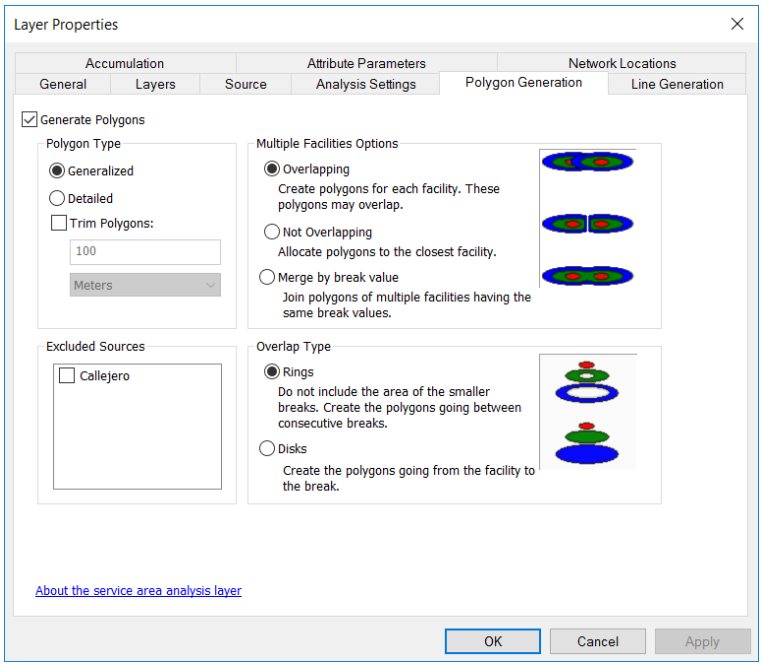


Ilustración 49. Definición de características Service Area.

La capa de barrios se publicará en ArcGis Online. Editando tanto la escala del mapa, como su escala visible. Para la escala del mapa se optó por la escala predefinida ya existente para ArcGis Online y para el rango de escala, que no sea visible por encima de 1:1155581, por debajo de 1:1128. Se publicará en Arcgis Online ya que su uso posterior no requerirá de ningún tipo de edición de sus datos, ni por parte del cliente ni de la organización, siendo por lo tanto un servicio meramente informativo.

Las Oficinas son datos, que requieren una actualización periódica por parte de la Entidad Bancaria, es por ello que esta capa se registrará en Arcgis Server, lo que facilitará la edición de datos en línea. Ha esta capa el administrador y poseedor de los datos, el Técnico Gis la ha habilitado como versionada, dándole poderes de edición al departamento de Gestión de oficinas y cajeros, los cuales han creado una versión de edición a partir de la copia de seguridad de los datos en la que se editarán este tipo de datos. El esquema de referencia se muestra en el apartado 3.0, flujo de versionado. Por último, al registrar la base de datos, se ha subido el servicio dotándolo de la capacidad de feature server, ya que sin esta capacidad las ediciones no son posibles.

La capa de los clientes será editada por el departamento de la gestión de clientes, los que podrán añadir o quitar clientes de la entidad, o simplemente agregar datos a los clientes ya disponibles y por eso está registrada como versionada en una geodatabase Enterprise.

Service area

Publicamos el servicio en el servidor, que tiene las siguientes propiedades (ver imagen abajo).

Cómo los clientes son los que se desplazan para ir a la oficina, el 'travel direction' es 'from facility'.

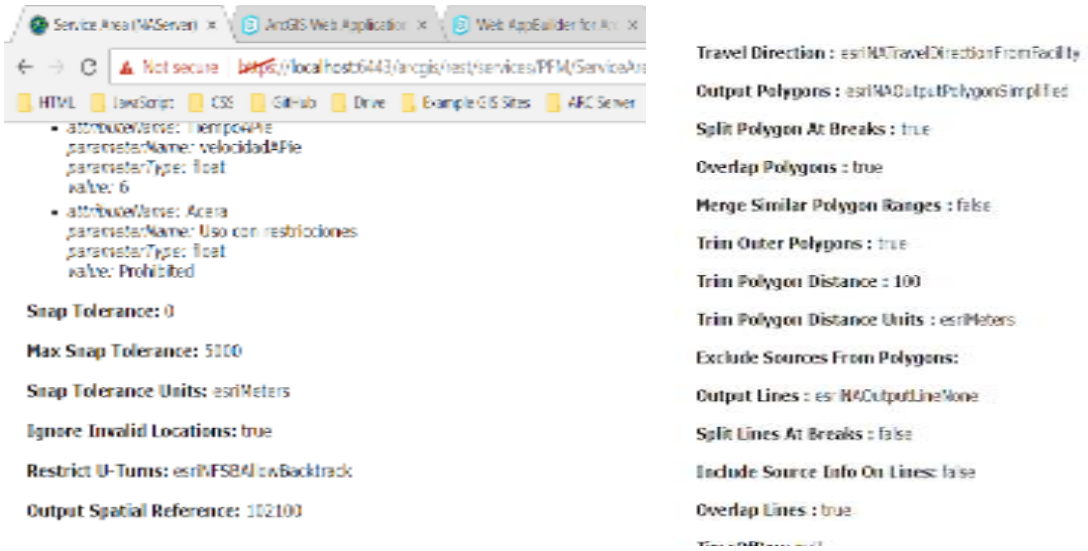


Ilustración 50. Service Area alojado en ArcGIS Server.

2.4.3. Desarrollo y Resultados

Una vez preparada la capa de barrios, la definimos en el código del widget y llamamos a los campos relevantes en un querytask para poblar el HTML del widgetpanel. Los atributos de los campos de Barrio, Población desempleada, y poder adquisitivo van en una tabla, y los datos correspondientes de las edades van aparte en un pie chart (gráfico de sectores), creado con dojo charting (https://dojotoolkit.org/reference-guide/1.10/dojox/charting.html).

Las funciones del widget Nueva Oficina son:

- Crear punto al hacer click
- Crear service area y polígonos
- Querytask spatial_intersect para pintar cada cliente dentro de los polígonos del service area
- Querytask para recoger datos del Barrio
- Poblar tabla y pie chart con los datos recogidos

Las funciones de querytask fueron basados en el ejemplo que trataba de disparar un querytask al hacer click, con un radio determinado (https://joewdavies.github.io/WorldCities/). Solo en este caso, la geometría del query no es un círculo sino los polígonos creados por el ServiceAreaTask. Esta

geometría viene del valor cogido en el 'for' que, por cada polígono creado, da una simbología distinta.

Entonces el querytask para clientes dentro de 5 minutos andando coge el segundo polígono creado (features[1]) y el quertask para clientes dentro de 10 minutos andando coge el primer polígono (features[0]).

Dentro de la función querytask.execute, está definido un contador (countOfFeatures). Si el valor de clientes dentro del polígono es mayor de 0, empieza el for, contando cada feature dentro de la geometría y dándole una simbología, preparado para cuando su graphicsLayer se agrega al mapa.

Para ir probando los cambios en el entorno de desarrollo, la ubicación del widget fue añadido al config-demo.json en la carpeta 'Sample Configs' del stemapp.

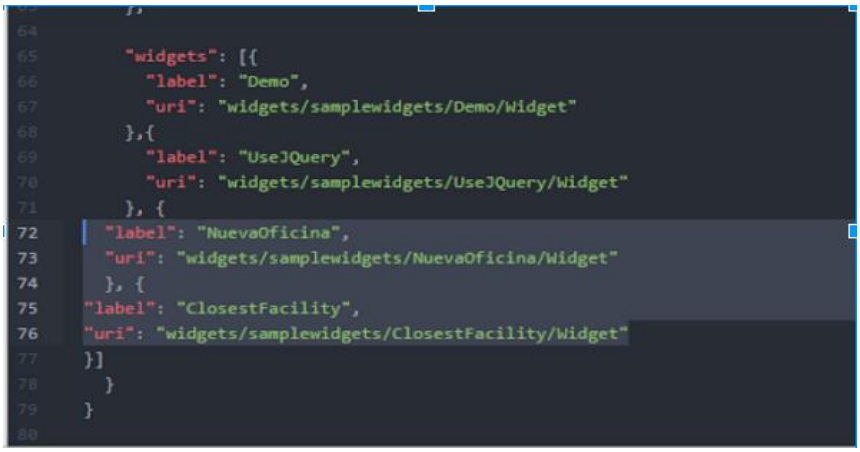


Ilustración 51. Incorporación de los widget en el config-demo.

Como resultado en la aplicación se tienen tres capas disponibles para visualizar: Clientes, Oficinas y Mapa de calor de los clientes.

La primera visualización de la aplicación refleja visualmente con un mapa de calor la distribución de los clientes por la ciudad de Madrid, pudiéndose analizar fácilmente aquellas zonas con mayor o menor concentración y de forma escalable. Del mismo modo con las oficinas, se posiciona gráficamente sus ubicaciones y junto con el mapa de calor proporciona importante información acerca de su situación con respecto a la cantidad de clientes de la zona.

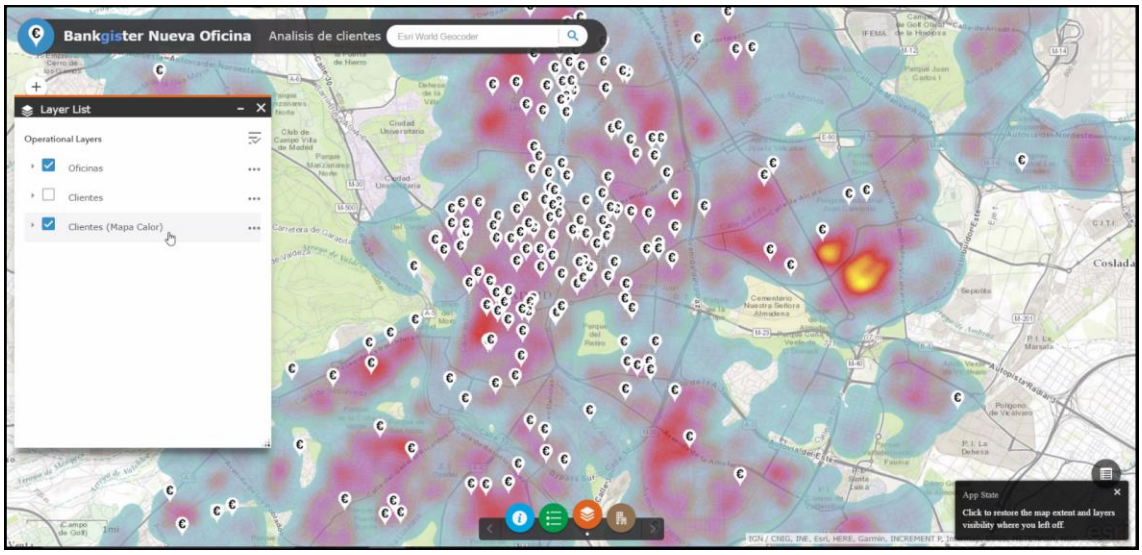


Ilustración 52. Visualización del mapa de calor de la aplicación.

Finalmente el usuario tiene disponible el widget llamado Nueva Oficina, que como se ha definido anteriormente permite realizar un análisis del número de clientes que se encuentran a 5 o 10 minutos andando desde un punto específico.

A la vez, el widget muestra por panel la información demográfica sobre el barrio seleccionado. El usuario puede ver junto al nombre del barrio un detalle del poder adquisitivo per cápita, y la población desempleada, además de un gráfico que muestra los porcentajes de cada rango de edades.

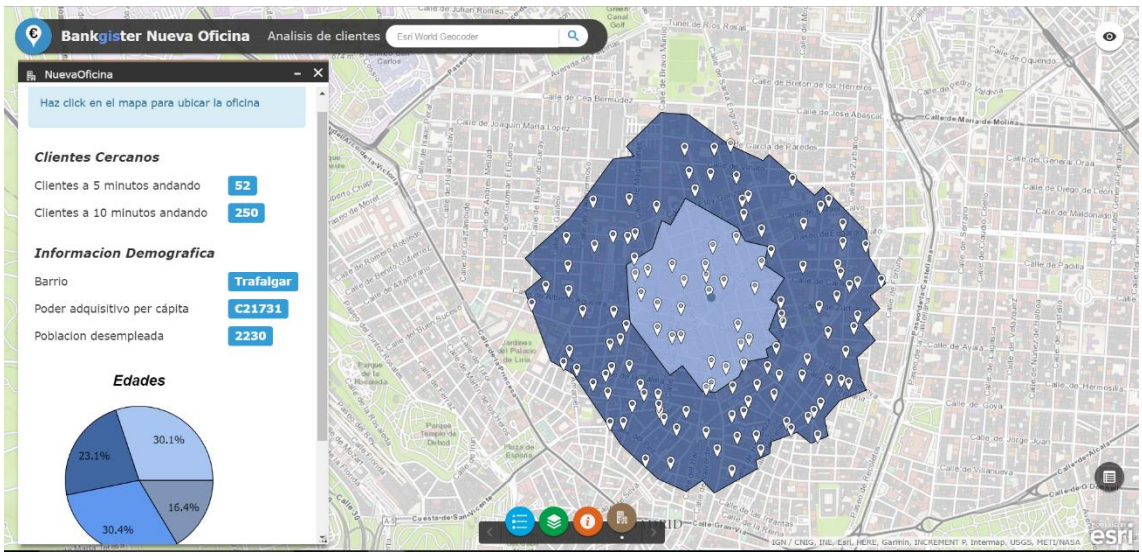


Ilustración 53. Visualización del widget 'Nueva Oficina' en la aplicación.

Estos datos pueden ser muy útiles a la hora de tomar decisiones sobre campañas de marketing que el banco quiere realizar en un barrio específico. Por ejemplo, ofertar planes de pensiones en barrios que tienen una mayoría de la población con más de 60 años o hipotecas en barrios que tienen una mayoría de la población entre 30-44 años.

Además, puede ayudar no solo con la decisión de ubicar una nueva oficina, sino también con el cierre de una ya existente y su repercusión para los clientes.

2.5. Story Map. Nuestro banco a través de los años

2.5.1. Resumen de la aplicación

Los Story Maps de Esri nos permiten combinar mapas acreditados con texto narrativo, imágenes y contenido multimedia. Facilitando así el uso de los mapas para contar una historia como es en nuestro caso.

En el proyecto representamos a Bankgister, una entidad bancaria española muy joven que, gracias a aprovechar las nuevas tecnologías y sobre todo el uso de los Sistemas de Información Geográfica, ha conseguido ubicarse como referente a nivel mundial.

A las grandes entidades bancarias les gusta comunicar sus éxitos, y como nuestro banco está orgulloso de su historia y crecimiento decidimos aprovechar un Story Map que lo plasmará no solo con palabras sino también visualmente.

Eligiendo el tipo de Story Map

Como nuestro objetivo es contar la historia de Bankgister es similar a decir que queremos crear un diario de mapa, por lo que el formato de Story Map Journal es el que mejor se adapta a nuestros intereses: sencillo, pero muy flexible.

Un Map Journal contiene entradas, o secciones, por las que los usuarios pueden desplazarse. Cada sección de un diario de mapa tiene asociado un mapa, una imagen, un vídeo o una página web. Las acciones también se pueden definir en secciones de diario de modo que, por ejemplo, al hacer clic en una palabra o imagen se haga zoom automáticamente el mapa de sección de una ubicación particular.

2.5.2. Desarrollo y publicación

Nos basaremos en dos servicios que tenemos subidos al ArcGIS Online de la empresa:

- La capa de Oficinas Bankgister de Madrid (Esta capa está publicada con la animación del tiempo habilitada)
- La capa de Barrios de Madrid

Como Web Map principal estarán representadas las oficinas con la extensión time-line, para que el usuario pueda ver la evolución de la entidad bancaria a lo largo de los años, acompañado de una sección introductoria de la historia de la Empresa.

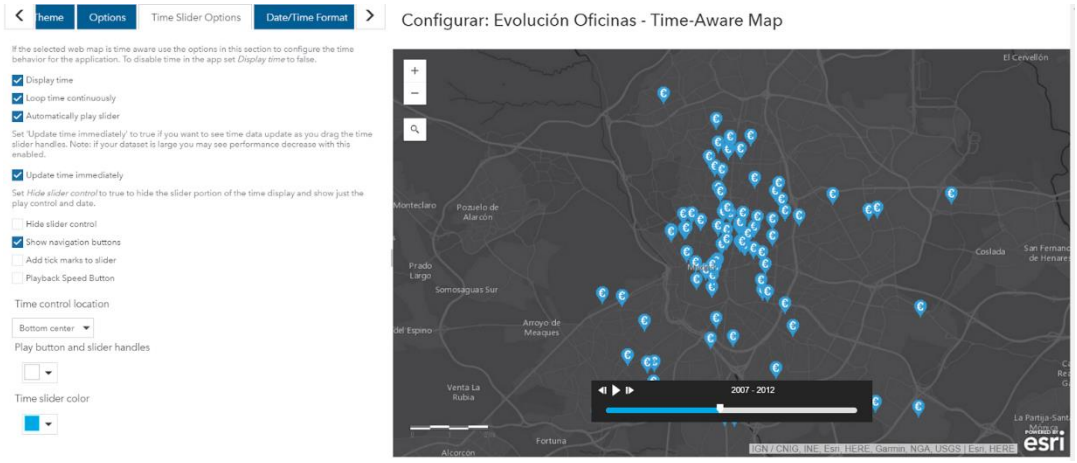


Ilustración 54. Habilitar barra de tiempo en el web map.

Para crear este primer mapa tuvimos que realizarlo con la aplicación Tiempo Habilitado (Time-Aware) e integrarlo en nuestro Story Map para poder incluir la animación temporal en la historia. Desde esta configuramos algunos aspectos del time-line.

A continuación, nuestro propósito era mostrar un Web Map por cada año, desde el 2007 (creación del banco) hasta día de hoy, cómo ha ido siendo la expansión del banco en lo que se refiere a los barrios de Madrid y mostrando donde se iban ubicando las nuevas oficinas respondiendo a las necesidades.

Se han creado 10 secciones, cada una de ellas representando un año desde la creación de la entidad bancaria. Todas las secciones tendrán como mapa principal un Web Map diferente. Cada Web Map representará la creación y evolución de las oficinas de Bankgister en la ciudad de Madrid, según el año que corresponda. Para complementar el Story Map, se le ha añadido además a cada WebMap los barrios de Madrid en los que hay localizadas oficinas. Estos barrios se irán sumando a medida que se avance de año, representando así todos los barrios donde hay oficinas hasta la fecha que el usuario seleccione, lo que aportará una información adicional a los usuarios sobre la expansión de Bankgister.

En el Story Map vamos guiando a los usuarios por varios mapas, imágenes, acciones y vídeo asociados para proporcionarles una información atractiva y detallada.

Añadiendo estilo corporativo

Desde el Map Journal Builder la plantilla permite tocar los diferentes parámetros preestablecidos en la plantilla, pero lo que nos interesa es darle un estilo propio corporativo que nos represente. Desde GitHub nos descargamos la plantilla del Map

Journal con la cual podremos mediante HTML y CSS modificar la plantilla a un nivel de estilo más detallado.

Los colores predominantes de nuestra organización son el negro, azul y blanco por lo que nos basaremos en aplicar estos a los elementos disponibles, además de alguna configuración de posición:

- Crear hoja de estilos (custom.css) para sobrescribir los estilos predeterminados
- Crear enlace en el head (<link rel="stylesheet" href="custom.css">)
- Sustituir el logo de ESRI con el logo de bankgister:
- <img id ="logo" src = "docs/images/bankgister.png" alt ="BankGister">
- Ajustar el header para que quepa el logo
- Centrar logo y panel titles
- Cambiar leyenda a blanco
- Sobreescribir Favicon de ESRI con logo de BankGister:
- <link rel="icon" href="docs/images/eurofavicon.ico" type="image/x-icon" />

A continuación, se muestra una visualización del resultado final del Story Map creado:

Como primera visualización se refleja una introducción histórica en forma de resumen junto a un servicio de mapa con el tiempo habilitado mencionado anteriormente, este refleja de forma dinámica la expansión de las oficinas por el municipio de Madrid.

Ilustración 55. Visualización inicial del Story Map.

En las siguientes diapositivas se ha desarrollado la historia 'ficticia' de la entidad bancario a lo largo de los años desde su creación. Reflejándose sus logros o perjuicios de cada año mediante texto e imágenes, y siempre acompañado de un web map con los barrios abarcados por el banco y la ubicación de las nuevas oficinas de dicho año.

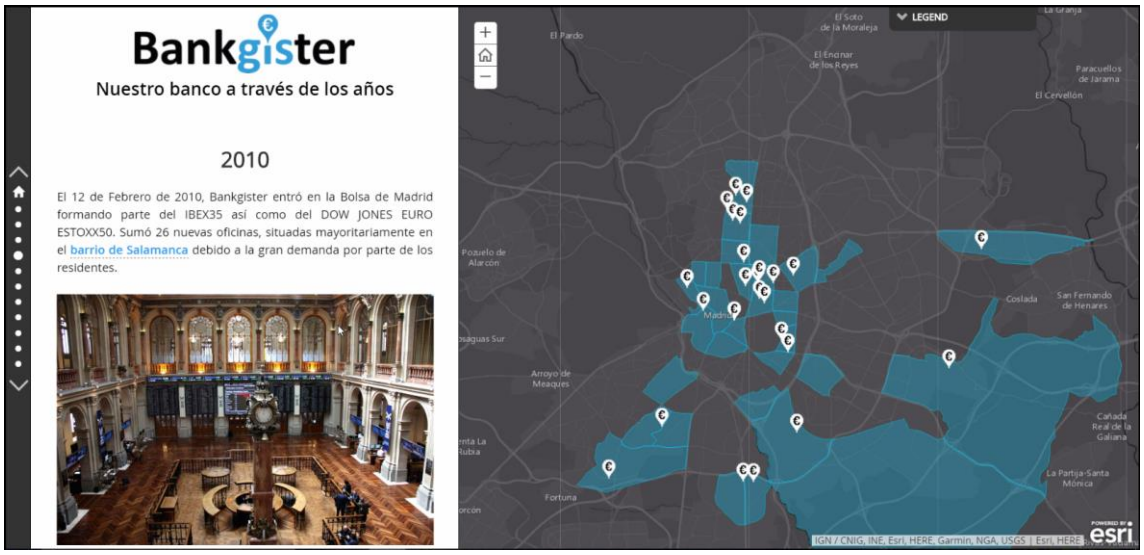


Ilustración 56. Visualización del Story Map.

3. Creación de GDB y Versionado

Para la creación de la Geodatabase Corporativa, se utilizó la herramienta crear Geodatabase Corporativa, que nos crea una base de datos, un administrador de geodatabase y una geodatabase corporativa en Microsoft SQL Server;

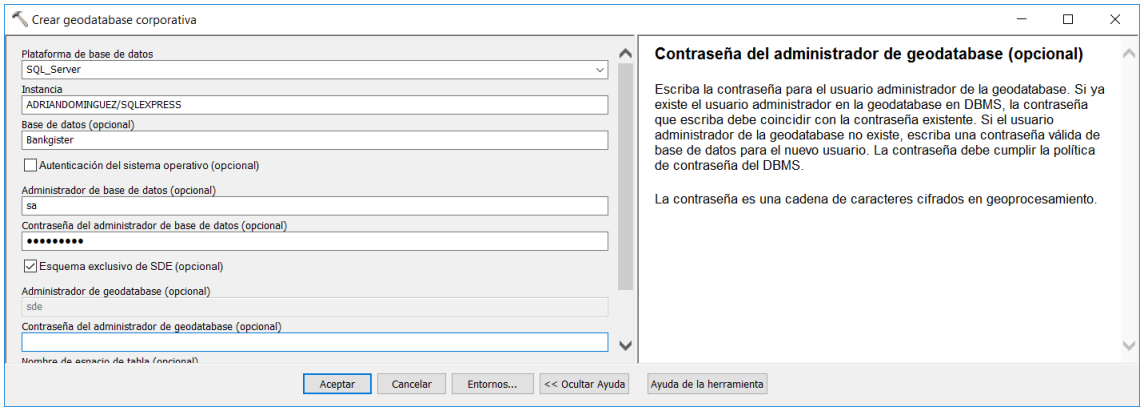


Ilustración 57. Creación de GDB Corporativa en ArcMap.

Una vez creada la Geodatabase, se comprueba que se han establecido todas las conexiones necesarias, y se comprueba en SQL si se han generado las tablas Sde. El siguiente paso fue añadir las capas a la conexión Sde de Bankgister, que va a ser el propietario de todos los datos, siendo a la vez el técnico GIS, dando como resultado el siguiente esquema;

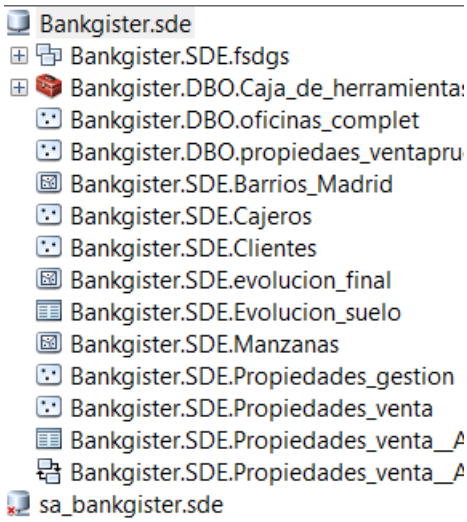


Ilustración 58. Estructura de las capas en la GDB con usuario SDE.

Para pasar las capas a Bankgister.sde, se importan desde la geodatabase, con la herramienta importar clase de entidad múltiple. El siguiente paso fue crear los demás usuarios o Departamentos de nuestra entidad Bancaria. Para ello desde Sa, se crean los diferentes usuarios, y seguidamente se les agrega sus diferentes conexiones con las bases de datos. Creando por lo tanto tres usuarios con tres conexiones distintas;

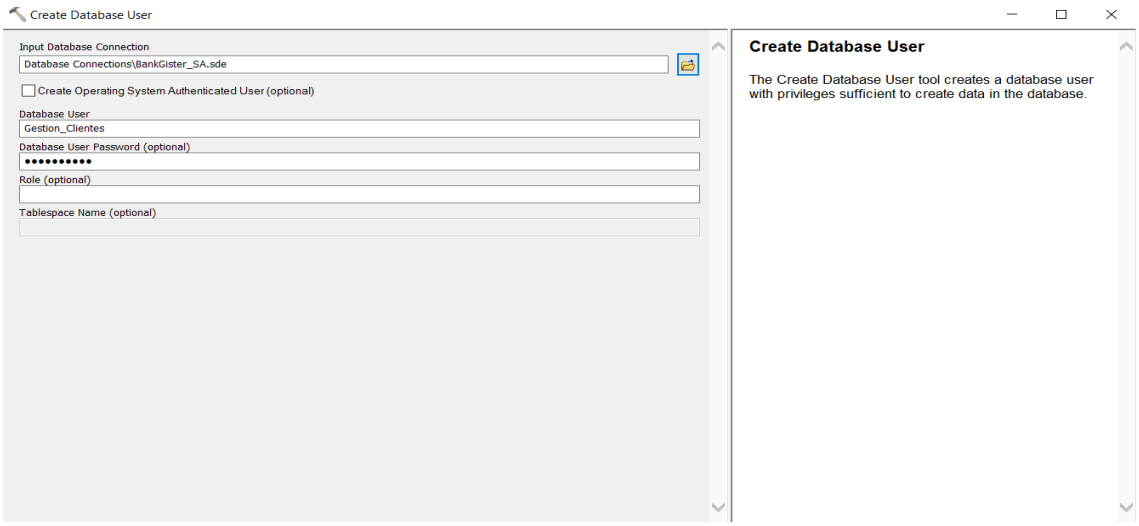


Ilustración 59. Creación de usuarios en la database.

Una vez agregadas las capas, el siguiente paso es darles privilegios a las mismas. Es el sde (Técnico_Gis) el que administra los privilegios de las capas.

Capas	Técnico GIS	Gestión de clientes	Gestión de oficinas y cajeros	Gestión de propiedades
Clientes	Select Insert Update Delete	Select Insert Update Delete	Select Insert Update Delete	
Cajeros	Select Insert Update Delete		Select Insert Update Delete	
Propiedades_Venta	Select Insert Update Delete			Select Insert Update Delete
Evolucion_Suelo	Select Insert Update Delete			Select Insert Update Delete
Manzanas	Select Insert Update Delete			Select Insert Update Delete
Propiedades_gestion	Select Insert Update Delete			Select Insert Update Delete
Propiedades_venta	Select Insert Update Delete			Select Insert Update Delete
Oficinas	Select Insert Update Delete		Select Insert Update Delete	
Evolucion_final	Select Insert Update Delete			Select Insert Update Delete
Propiedades_Adjuntos	Select Insert Update Delete			

Ilustración 60. Tabla de privilegios según capa-usuario.

Para finalizar se deben administrar las versiones:

La primera Versión que hay es la Default, que no puede ser eliminada, ya que fue creada cuando se creó la GDB Multiusuario.

Se crea una versión Security que se usará como control de calidad, para ello se le da seguridad protegida, para que los usuarios puedan verla, pero no puedan realizar ediciones sobre ella.

Se añaden tres versiones más desde la versión Security, Cajeros, Clientes y Propiedades en Venta. Se les da el nivel de permiso de Pública, ya que consideramos que todos los miembros de la organización en algún momento pueden hacer ediciones sobre ellas.

Por último, se registra las capas sobre las que se van a realizar ediciones, dotando a la geodatabase de la capacidad de aislar las ediciones realizadas dentro de una versión específica.

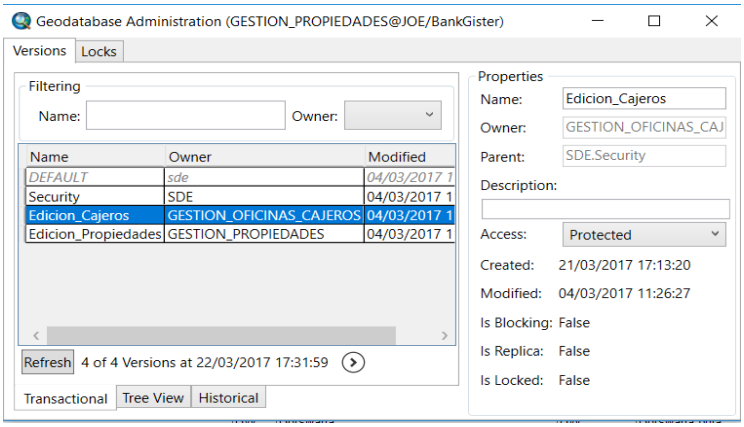


Ilustración 61. Registro de capas editables de la GDB.

El resultado versionado realizado puede verse en el siguiente esquema de elaboración propia:

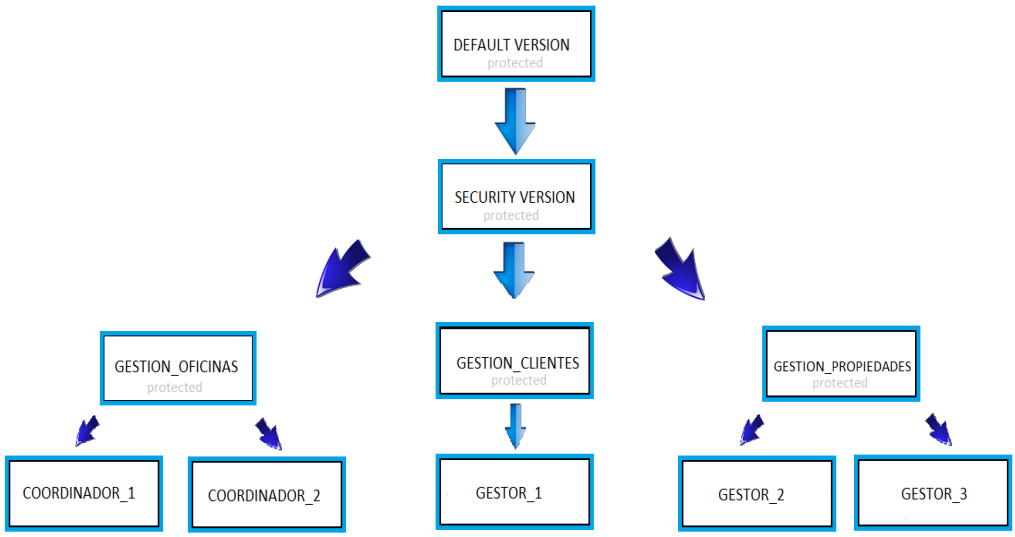


Ilustración 62. Esquema del Versionado de la GDB.

4. Diseño de la Organización

ArcGIS Online es el lugar para explorar datos, crear mapas y compartir historias. La empresa mediante esta herramienta puede usar y crear mapas y escenas, acceder a mapas, capas y análisis listos para usar de Living Atlas of the World, publicar datos como capas web, colaborar y compartir, acceder a mapas desde cualquier dispositivo, crear mapas con sus propios datos de negocio, personalizar el sitio web de ArcGIS Online y ver informes de estado. Es decir, que las aplicaciones expuestas anteriormente y todos los datos publicados desde ArcGIS Desktop, son también accesibles y utilizables desde aquí.

Ilustración 63. Diseño de la cuenta de ArcGIS Online 1.

Ilustración 64. Diseño de la cuenta en ArcGIS Online 2.

Las imágenes anteriores muestran el aspecto configurado para la empresa en la cuenta en ArcGIS Online, dónde se ha configurado un inicio que embebe un acceso directo a las cinco aplicaciones desarrolladas por la entidad bancaria, así también un estilo editable y una breve descripción.

Los distintos equipos desarrollan y mantienen todo el contenido con relación a los Sistemas de Información Geográfica. Estos equipos de la empresa mencionados en el apartado anterior pueden crearse del mismo modo dentro de la cuenta y establecer también permisos y privacidad de los datos publicados para cada uno de los grupos:

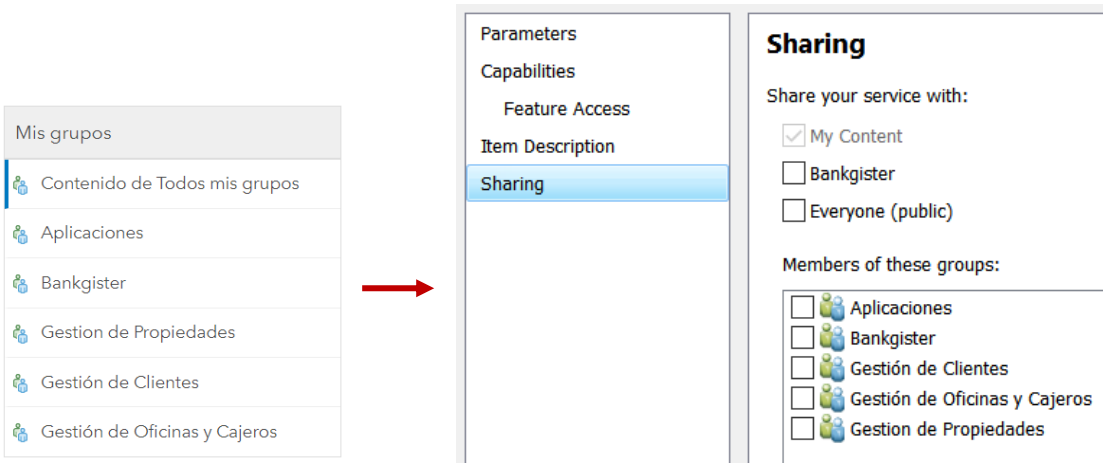


Ilustración 65. Grupos creados en ArcGIS Online y su visualización desde ArcMap.

Y como se refleja en la imagen de la derecha, de cara a la publicación de nuevos contenidos desde ArcMap o a los publicados anteriormente, se podrá seleccionar al grupo al que pertenecen los datos.

Un único punto de acceso a la información en forma de portal para la gestión de contenidos geográficos. Se organiza la información de modo que cuando un usuario accede a este portal puede buscar en su contenido o en el contenido de la organización, esto le garantiza que el contenido al que está accediendo y el dato que está utilizando es el bueno, el fidedigno. A partir de ahí cuando realiza una búsqueda de alguna capa tiene la confianza de que la capa devuelta por el sistema es la oficial de su empresa. Esto permite la colaboración, garantizando que cada uno de los departamentos debe mantener aquella información que es de su responsabilidad. Es importante garantizar que se accede a la fuente de datos correcta ya que si la información de partida no es buena se obtendrán resultados aún peores.

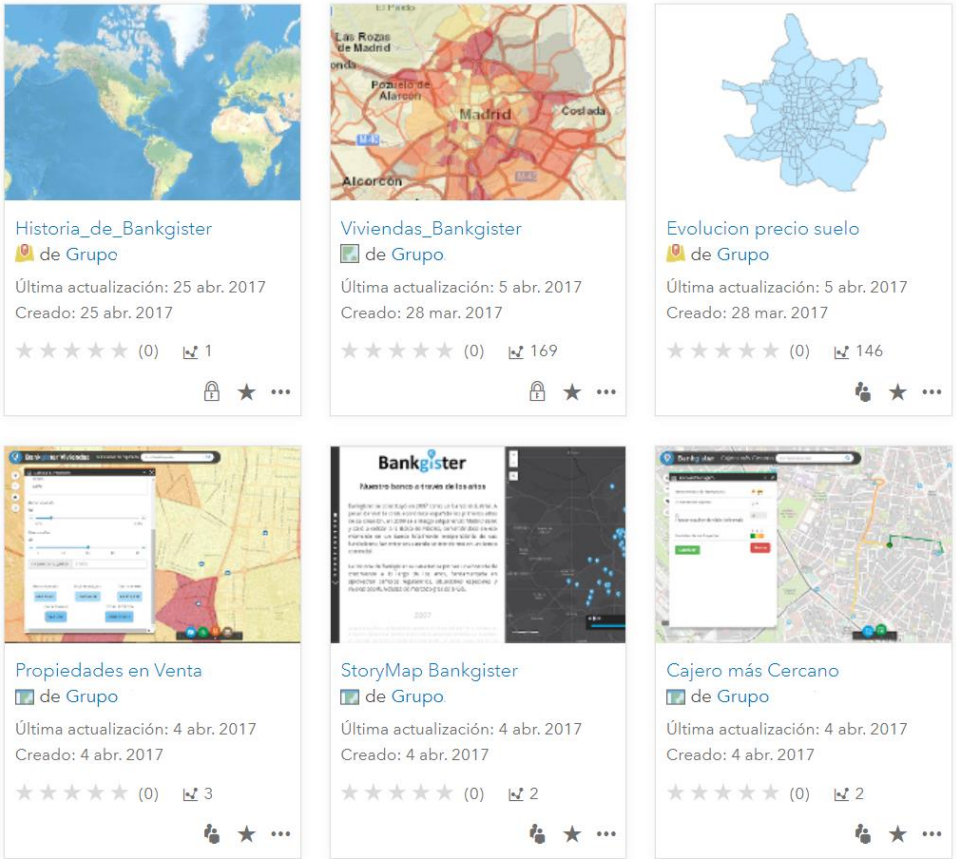


Ilustración 66. Estructura del contenido dentro de la cuenta de ArcGIS Online.

La imagen anterior nos muestra esa interoperabilidad de información, donde encontramos datos publicados desde ArcGIS Desktop como es el caso de la capa 'Evolución precio suelo', la capa 'Historia_de_Bankgiser' que recogía los datos necesarios para la creación del Time-Aware Map y su posterior uso en la aplicación del StoryMap, el web map 'Viviendas_Bankgister' creado desde el propio ArcGIS Online que recoge en forma de rampa de color el dato de evolución precio suelo de la capa alojada anteriormente, y por último el propio acceso a las aplicaciones creadas y almacenadas desde ArcGIS WebAppBuilder.

Las herramientas de ETL permiten añadir la capacidad espacial dentro de nuestras bases de datos corporativas y dentro de nuestros sistemas. Con el enriquecimiento de estos datos se consigue añadir información útil a la hora de realizar los análisis. Herramientas que permiten analizar los datos de manera visual y realizar un análisis exploratorio de los datos con distintos criterios.

5. Anexos

5.1. JS 'WorldCurrencyMap' – Cambio de Moneda

```
var map;
var dialog;

require(["esri/map",
"dojo/dom",
"esri/urlUtils",
"esri/request", "esri/config", "dojox/xml/parser", "dojo/Deferred",
"esri/arcgis/utils",
"esri/layers/FeatureLayer","esri/symbols/SimpleFillSymbol", "esri/symbols/SimpleLineSymbol",
"esri/renderers/SimpleRenderer", "esri/graphic", "esri/lang",
"esri/Color", "dojo/number", "dojo/dom-style",
"dijit/TooltipDialog", "dijit/popup",
"esri/dijit/BasemapToggle", "esri/dijit/Search", "esri/dijit/HomeButton",
"esri/InfoTemplate",
"dojo/domReady!"],

function(Map, dom, urlUtils,
  esriRequest, esriConfig, xmlParser, Deferred,
  arcgisUtils, FeatureLayer,
  SimpleFillSymbol, SimpleLineSymbol,
  SimpleRenderer, Graphic, esriLang,
  Color, number, domStyle,
  TooltipDialog, dijitPopup,
  BasemapToggle, Search, HomeButton,
  InfoTemplate) {

    map = new Map("map", {
      basemap: "dark-gray",
      zoom: 3,
      center: [-3.62,30.42],
    });

    var WorldCountries = new FeatureLayer(
    "http://services7.arcgis.com/ck1aFB1wRWjpAB2g/arcgis/rest/services/WorldCountries2/FeatureServer/0", {
      mode: FeatureLayer.MODE_SNAPSHOT,
      outFields: ["Country", "Currency","Currency_Code"]
    });

    //SET LAYER COLOUR//
    var symbol = new SimpleFillSymbol(
      SimpleFillSymbol.STYLE_SOLID,
      new SimpleLineSymbol(
        SimpleLineSymbol.STYLE_SOLID,
        new Color([1,0,0,0.35]),
        1
      ),
      new Color([51,122,183,0.80])
    );
    WorldCountries.setRenderer(new SimpleRenderer(symbol));


    map.addLayer(WorldCountries);

    //MOUSE_OVER SYMBOL//
    var highlightSymbol =
    new SimpleLineSymbol(
      SimpleLineSymbol.STYLE_SOLID,
      new Color([255,255,255]), 1
    );

    //close the dialog when the mouse leaves the highlight graphic
    map.on("load", function(){
      map.graphics.enableMouseEvents();
      map.graphics.on("mouse-out", clearGraphics);
      var home = new HomeButton({
        map: map
        }, "HomeButton");
      home.startup();
    });

    function clearGraphics() {
      map.graphics.clear();
    }
```

```
74      //Buscador de paises
75      var search = new Search({
76          map: map,
77          enableInfoWindow: false,
78          showInfoWindowOnSelect: false,
79          enableSearchingAll: false,
80          autoNavigate: false,
81          sources: []
82         //  activeSourceIndex: 0,
83        }, "search");
84      //Obtenemos el array con las fuentes que tiene cargadas nuestro widget para hacer las búsquedas
85      var sources = search.get("sources");
86      //Añadimos un elemento a este array de fuentes de búsqueda con mi capa
87      sources.push({
88        featureLayer: new FeatureLayer(
89        "http://services7.arcgis.com/ck1aFB1wRWjpAB2g/arcgis/rest/services/WorldCountries2/FeatureServer/0"),
90        searchFields: ["Country"],
91        suggestionTemplate: "${Country}",
92        displayField: "Country",
93        exactMatch: false,
94        outFields: ["*"],
95        name: "Paises",
96        placeholder: "Nombre del País",
97        maxResults: 5,
98        maxSuggestions: 5,
99        enableSuggestions: true,
100       minCharacters: 0,
101       defaultSource: true
102     });
103
104     var symbol2 = new SimpleFillSymbol(
105       SimpleFillSymbol.STYLE_SOLID,
106       new SimpleLineSymbol(
107         SimpleLineSymbol.STYLE_SOLID,
108         new Color([255, 255, 255]),
109         1
110       ),
111       new Color([255, 50, 50])
112     );
113
114     search.sources[0].highlightSymbol = symbol2;
115
116     search.set("sources", sources);
117     //Inicializamos el widget
118     search.startup();
119
```

```
120     //Por búsqueda en el search
121     search.on("search-results",function(result){
122       var center = result.results[0][0].feature.geometry.getCentroid();
123       map.centerAndZoom(center, 4);
124
125       var pais = result.results[0][0].feature.attributes.Country;
126       var moneda = result.results[0][0].feature.attributes.Currency;
127       var codigo = result.results[0][0].feature.attributes.Currency_Code;
128
129
130       document.getElementById("country").innerHTML = pais;
131       document.getElementById("currency").innerHTML = moneda;
132       document.getElementById("code").innerHTML = codigo;
133
134       var input_euro = document.getElementById("exchangerate1");
135
136       //Acceso al XML del ECB
137       esriConfig.defaults.io.proxyUrl = "http://ELENA/DotNet/proxy.ashx";
138
139       urlUtils.addProxyRule({
140         urlPrefix: "ecb.europa.eu",
141         proxyUrl: "http://ELENA/DotNet/proxy.ashx"
142       });
143
144       var url = "http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml";
145
146       var requestHandle = esriRequest({
147         "url": url,
148         "handleAs": "xml"
149       });
150       requestHandle.then(requestSucceeded);
151
```

```
    function requestSucceeded(response, io){
        var rates, i, xmlDoc, txt, input, output;
        xmlDoc = response;
        txt = "";
        rates = xmlDoc.getElementsByTagName('Cube');
        for (i = 2; i < rates.length; i++) {
          if (rates[i].getAttribute('currency') == codigo){
            txt += rates[i].getAttribute('rate') + "<br>";
            input = parseFloat(rates[i].getAttribute('rate')).toFixed(2);
            output = input_euro.value * input;
            document.getElementById("rate").innerHTML = output;
          }
        }
        document.getElementById("cambio").innerHTML = txt;
      }

});


//Por selección directa en el mapa
WorldCountries.on("click", function(evt){
  var pais = evt.graphic.attributes.Country;
  var moneda = evt.graphic.attributes.Currency;
  var codigo = evt.graphic.attributes.Currency_Code;


  document.getElementById("country").innerHTML = pais;
  document.getElementById("currency").innerHTML = moneda;
  document.getElementById("code").innerHTML = codigo;

  var input_euro = document.getElementById("exchangerate1");

  //Acceso al XML del ECB
  esriConfig.defaults.io.proxyUrl = "http://ELENA/DotNet/proxy.ashx";

  urlUtils.addProxyRule({
    urlPrefix: "ecb.europa.eu",
    proxyUrl: "http://ELENA/DotNet/proxy.ashx"
  });

  var url = "http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml";

  var requestHandle = esriRequest({
    "url": url,
    "handleAs": "xml"
  });
  requestHandle.then(requestSucceeded);
```

```
  function requestSucceeded(response, io){
      var rates, i, xmlDoc, txt, input, output;
      xmlDoc = response;
      txt = "";
      rates = xmlDoc.getElementsByTagName('Cube');
      for (i = 2; i < rates.length; i++) {
        if (rates[i].getAttribute('currency') == codigo){
          txt += rates[i].getAttribute('rate') + "<br>";
          input = parseFloat(rates[i].getAttribute('rate')).toFixed(2);
          output = input_euro.value * input;
          document.getElementById("rate").innerHTML = output;
        }
      }
      document.getElementById("cambio").innerHTML = txt;
    }
});

var button = document.getElementById("calcula");
button.onclick = function(){
  var input_euro = document.getElementById("exchangerate1").value;
  var input = document.getElementById("cambio").textContent;
  output = input_euro * parseFloat(input);
  document.getElementById("rate").innerHTML = parseFloat(output).toFixed(0);
};

//Hover function//
WorldCountries.on("mouse-over", function(evt){
  var highlightGraphic = new Graphic(evt.graphic.geometry,highlightSymbol);
  map.graphics.add(highlightGraphic);
  });

  function Borrardiv(){}
  function Poblardiv(){}

});
```

5.1.1. CSS Aplicación

```css
html body {
    height: 100%;
    margin: 0;
    padding: 0;
}

#map {
    height: 750px;
}

#HomeButton {
    position: absolute;
    top: 92px;
    left: 20px;
    z-index: 100000;
}

.HomeButton .home {
    background-image:url(../images/home.png);
    background-color: #ffffff;
    width: 32px;
    height: 32px;
    border-style: solid;
    border-width: thin;
}

.left-half {
    float: left;
    width: 100%;
    height: 100%
}

.right-half {
    float: left;
    width: 20%;
    right:1rem;
    top:1rem;
    /*border: 3px solid;
    border-color: black;*/
    position: absolute;
}

.panel-primary {
    border-color: black;
    margin-bottom: 0;
    text-align: center;
}
.panel-title {
    text-align: center;
    font-weight: bold;
}
.panel-primary>.panel-heading{
    background-color: black;
}

#title{
    position: absolute;
    left:4rem;
    top: 9rem;
}

#logo{
position: absolute;
left:7rem;
top: 1rem;
width: 30rem;
background-color: rgba(0, 0, 0, 0.5);
padding: 5px 15px;
border-radius: 10px;
}

.arcgisSearch .searchGroup .searchInput {
    width: auto;
}

#search {
text-align: left;
}

.BasemapToggle .basemapTitle {
width: 100%;
}

/*Popup*/
#tooltipDialog {
position: absolute;
width: auto;
font: normal normal normal 10pt Helvetica;
z-index:10000000 }
/*
.dijitTooltipContainer {}*/

/*change cursor to pointer on hover*/
g#graphicsLayer2_layer{
    cursor: pointer;
}

#country, #currency {
    background-color: #dadada;
    padding: 10px;
    border-radius: 5px;
    display: inline-block;
}

#currency, #rate {
    background-color: #75c6f5;
}

#code {
    background-color: #dadada;
}

#euro, #code, #rate {
    padding: 10px;
    border-radius: 5px;
    display: inline-block;
}

.panel-body#conversion {
    height: 180px;
}

#calcula {
    background-color: #d64747;
    border-color: #901717;
    margin-bottom: 10px;
    border-radius: 8px;
}

#cambiotxt, #cambio {
    display: inline-block;
    color: #d4cdcd;
}
```

5.2. Widget 'ClosestFacility' – Cajero más Cercano

```
define([
    'dojo/_base/declare',
    'dojo/_base/lang',
    'dojo/on',
    'dojo/dom',
    'dijit/registry',
    'dojo/dom-construct',
    'dojo/query',
    'dojox/widget/Standby',
    'jimu/BaseWidget',
    'esri/config',
    'esri/Color',
    'esri/layers/GraphicsLayer',
    'esri/layers/FeatureLayer',
    'esri/renderers/UniqueValueRenderer',
    'esri/graphicsUtils',
    'esri/toolbars/draw',
    'esri/graphic',
    'esri/symbols/SimpleMarkerSymbol',
    'esri/symbols/SimpleLineSymbol',
    'esri/tasks/FeatureSet',
    'esri/tasks/ClosestFacilityTask',
    'esri/tasks/ClosestFacilityParameters',
    'esri/tasks/ClosestFacilitySolveResult',
    'esri/tasks/NATypes',
    './widgets/ClosestFacility/PolylineAnimation.js'
],
function(declare, lang, on, dom, registry, domConstruct, query,
  Standby,
  BaseWidget,
  esriConfig,
  Color,
  GraphicsLayer, FeatureLayer,
  UniqueValueRenderer,
  graphicsUtils, Draw,
  Graphic,
  SimpleMarkerSymbol, SimpleLineSymbol,
  FeatureSet,
  ClosestFacilityTask, ClosestFacilityParameters, ClosestFacilitySolveResult, NATypes,
  PolylineAnimation) {

    /*     require(["/WAB_CF/widgets/ClosestFacility/PolylineAnimation.js"], function() {
    console.log("Load animation class");
  }); */
  var m_lyrResultRoutes, m_lyrAllFacilities, m_lyrEvents, m_lyrBarriers;
  var m_drawToolbar;

  var m_aryResultSymbolInfos; // Symbols for ranked results

  // Event handlers needing later removal
  var m_zoomToFacilities, m_clickDrawEvent, m_clickDrawBarrier,
  m_clickSolve, m_clickClear, m_changeFacilitiesCount,
  m_chkLimitTravelTime, m_numMaxTravelTime;
  // Closest Facility solver objects
  var m_closestFacilityTask;
  // Busy indicator handle
  var m_busyIndicator;

  //To create a widget, you need to derive from BaseWidget.
  return declare([BaseWidget], {
    // Custom widget code goes here

    baseClass: 'jimu-widget-customwidget'

    //this property is set by the framework when widget is loaded.
    ,name: 'ClosestFacilityWidget'

    //methods to facilitate communication with app container:

    ,postCreate: function() {
      this.inherited(arguments);
      console.log('postCreate');

      m_lyrAllFacilities = new FeatureLayer(
        this.config.facilities.url, {"mode":FeatureLayer.MODE_SNAPSHOT, "outFields":["*"]});
        m_zoomToFacilities = m_lyrAllFacilities.on('update-end', this.zoomToFacilities);
```

```
77
78          m_lyrResultRoutes = new GraphicsLayer();
79          m_lyrEvents = new GraphicsLayer();
80          m_lyrBarriers = new GraphicsLayer();
81
82          var slsDefault = new SimpleLineSymbol(SimpleLineSymbol.STYLE_SOLID, new Color([32, 32, 32]), 2);
83          var resultRenderer = new UniqueValueRenderer(slsDefault, this.config.symbology.routeRenderer.field1);
84          for (var i = 0; i < this.config.symbology.routeRenderer.uniqueValueInfos.length; i++) {
85            var info = this.config.symbology.routeRenderer.uniqueValueInfos[i];
86            var sls = new SimpleLineSymbol(info.style, info.sym.color, info.sym.width);
87            resultRenderer.addValue(info.value, sls);
88          }
89          m_lyrResultRoutes.setRenderer(resultRenderer);
90
91          m_drawToolbar = new Draw(this.map);
92          var sms = new SimpleMarkerSymbol(this.config.symbology.eventSymbol);
93          m_drawToolbar.setMarkerSymbol(sms);
94          var sls = new SimpleLineSymbol(this.config.symbology.barrierSymbol);
95          m_drawToolbar.setLineSymbol(sls);
96          m_drawToolbar.on("draw-complete", lang.hitch(this, this.onDrawEvent));
97
98          m_closestFacilityTask = new ClosestFacilityTask(this.config.closestFacilitySvc.url);
99        }
100
101       ,startup: function() {
102         this.inherited(arguments);
103         console.log('startup');
104
105         // Add ranks and colors from config
106         var rankNumbers = dom.byId("trRankNumbers");
107         var rankColors = dom.byId("trRankColors");
108         for (var i = 0; i < this.config.symbology.routeRenderer.uniqueValueInfos.length; i++) {
109           var info = this.config.symbology.routeRenderer.uniqueValueInfos[i];
110           var className = this.getRankSymbolDomClassName(info.value);
111           var aryColor = info.sym.color;
112
113           var tdRankNumber = '<td class="' + className + '" '
114           + 'style="text-align:center;">' + info.value + '</td>';
115           domConstruct.place(tdRankNumber, rankNumbers);
116
117           var tdRankColor = '<td class="' + className + '" ' +
118           'style="background-color:rgba(' +
119           aryColor[0] + ',' + aryColor[1] + ',' + aryColor[2] + ',' + aryColor[3] + ')' +
120           ';width:15px;height:15px;"> </td>'
121           domConstruct.place(tdRankColor, rankColors);
122         }
123
124         // Create busy indicator
125         m_busyIndicator = new Standby({target: "busyIndicator"});
126         document.body.appendChild(m_busyIndicator.domNode);
127         m_busyIndicator.startup();
128       }
129
130       ,onOpen: function(){
131         console.log('onOpen');
132
133         this.map.addLayer(m_lyrAllFacilities);
134         this.map.addLayer(m_lyrResultRoutes);
135         this.map.addLayer(m_lyrBarriers);
136         this.map.addLayer(m_lyrEvents);
137
138         m_clickDrawEvent = on(dom.byId("btnDrawEvent"), "click", this.onClickDrawEvent);
139         m_clickDrawBarrier = on(dom.byId("btnDrawBarrier"), "click", this.onClickDrawBarrier);
140         m_clickSolve = on(dom.byId("btnSolve"), "click", lang.hitch(this, this.onClickSolve));
141         m_clickClear = on(dom.byId("btnClear"), "click", lang.hitch(this, this.onClickClear));
142         m_chkLimitTravelTime = on(dom.byId("chkLimitTravelTime"),
143         "change", lang.hitch(this, this.onCheckLimitTravelTime));
144         m_numMaxTravelTime = on(dom.byId("numMaxTravelTime"),
145         "change", lang.hitch(this, this.onChangeMaxTravelTime));
146
147         var numFacilities = dom.byId("numFacilities");
148         m_changeFacilitiesCount = on(numFacilities, "change", lang.hitch(this, this.onChangeFacilitiesCount));
149         on.emit(numFacilities, "change", { bubbles: true, cancelable: true });
150       }
151
152       ,onClose: function(){
153         console.log('onClose');
154
155         this.map.removeLayer(m_lyrBarriers);
156         this.map.removeLayer(m_lyrEvents);
157         this.map.removeLayer(m_lyrAllFacilities);
158         this.map.removeLayer(m_lyrResultRoutes);
159
160         m_clickDrawEvent.remove();
161         m_clickDrawBarrier.remove();
162         m_clickSolve.remove();
163         m_clickClear.remove();
164         m_changeFacilitiesCount.remove();
165         m_chkLimitTravelTime.remove();
166         m_numMaxTravelTime.remove();
167       }
168
```

```
169         //methods to communication between widgets:
170
171         // Other methods
172         ,zoomToFacilities: function(event) {
173           var extent = graphicsUtils.graphicsExtent(event.target.graphics);
174           event.target.getMap().setExtent(extent);
175           m_zoomToFacilities.remove();
176         }
177         /*          ,zoomToResults: function(lyrResults) {
178         var extent = graphicsUtils.graphicsExtent(lyrResults.graphics);
179         this.map.setExtent(extent);
180       } */
181
182       ,onClickDrawEvent: function() {
183         console.log("Draw Event Click");
184         m_drawToolbar.activate(Draw.POINT);
185       }
186       ,onClickDrawBarrier: function() {
187         console.log("Draw Barrier");
188         m_drawToolbar.activate(Draw.POLYLINE);
189       }
190
191       ,onDrawEvent: function(event) {
192         console.log("Draw Event Complete");
193         m_drawToolbar.deactivate();
194
195         var geom = event.geometry;
196         if (event.geometry.type === "point") {
197           var symbol = event.target.markerSymbol;
198           var graphic = new Graphic(geom, symbol);
199           m_lyrEvents.add(graphic);
200           this.checkSolveEnabledState();
201         } else if (event.geometry.type === "polyline") {
202           var symbol = event.target.lineSymbol;
203           var graphic = new Graphic(geom, symbol);
204           m_lyrBarriers.add(graphic);
205         }
206
207       }
208
209       ,onClickSolve: function() {
210         console.log("Solve");
211         var params = new ClosestFacilityParameters();
212         params.defaultCutoff = 3.0;
213         params.returnIncidents = false;
214         params.returnRoutes = true;
215         params.returnDirections = true;
216
217         var facilities = this.fs4gl(m_lyrAllFacilities);
218         params.facilities = facilities;
219
220         var events = this.fs4gl(m_lyrEvents);
221         params.incidents = events;
222
223         var barriers = this.fs4gl(m_lyrBarriers);
224         params.polylineBarriers = barriers;
225
226         params.defaultCutoff = (dom.byId("chkLimitTravelTime").checked
227         ? dom.byId("numMaxTravelTime").value
228         : Number.MAX_VALUE );
229         params.defaultTargetFacilityCount = numFacilities.value;
230         params.outSpatialReference = this.map.spatialReference;
231         params.outputLines = NATypes.OutputLine.TRUE_SHAPE;
232         params.returnFacilities = false;
233
234         m_busyIndicator.show();
235         dom.byId("btnSolve").disabled = "disabled";
236
237         m_closestFacilityTask.solve(params,
238           lang.hitch(this, this.onSolveSucceed),
239           function(err) {
240             console.log("Solve Error");
241             m_busyIndicator.hide();
242             dom.byId("btnSolve").disabled = "";
243             alert(err.message + ": " + err.details[0]);
244         });
245       }
246
247       ,onSolveSucceed: function(result) {
248         console.log("Solve Callback");
249         m_busyIndicator.hide();
250         dom.byId("btnSolve").disabled = "";
251
```

```
      var routes = result.routes;
      routes.sort(lang.hitch(this, function(g1, g2) {
        var rank1 = g1.attributes[this.config.symbology.routeZOrderAttrName];
        var rank2 = g2.attributes[this.config.symbology.routeZOrderAttrName];
        // Reverse sort
        return rank2 - rank1;
      }));
      m_lyrResultRoutes.clear();
      for (var i = 0; i < routes.length; i++) {
        var g = routes[i];
        // Set animation here?
        var pla = new PolylineAnimation({
          graphic           : g,
          graphicsLayer : m_lyrResultRoutes,
          duration      : this.config.symbology.animateRoutesDuration
        });
        pla.animatePolyline();
      }
      // Zoom to results?
      // graphicsUtil.graphicsExtent() not working properly
      /*              if (dom.byId("chkZoomToResults").checked)
      this.zoomToResults(m_lyrResultRoutes); */
    }

    ,onClickClear: function() {
      console.log("Clear");
      m_lyrEvents.clear();
      m_lyrBarriers.clear();
      m_lyrResultRoutes.clear();
      this.checkSolveEnabledState();
    }

    ,onChangeFacilitiesCount: function(event) {
      console.log("Change Facilities Count: " + event.currentTarget.value);
      var count = event.currentTarget.value;
      for (var i = 0; i < this.config.symbology.routeRenderer.uniqueValueInfos.length; i++) {
        var className = this.getRankSymbolDomClassName(i+1);
        if ( i+1 <= count)
        query("." + className).style("visibility", "visible");
        else
        query("." + className).style("visibility", "hidden");
      }
    }

    ,getRankSymbolDomClassName: function(rank) {
      return "rank" + rank;
    }

    ,checkSolveEnabledState: function() {
      dom.byId("btnSolve").disabled = (m_lyrEvents.graphics.length > 0 ? "" : "disabled");
    }
    ,onCheckLimitTravelTime: function(event) {
      dom.byId("numMaxTravelTime").disabled = (event.target.checked ? "" : "disabled");
    }
    ,onChangeMaxTravelTime: function(event) {
      console.log("Change Max Travel Time");
      // Check for valid number
      if (!this.isPositiveInt(event.target.value))
      event.target.value = event.target.oldValue;
      // Update old value
      else
      event.target.oldValue = event.target.value;
    }
    ,isPositiveInt: function(str) {
      var n = ~~Number(str);
      return String(n) === str && n >= 0;
    }

    ,fs4gl: function( lyrGraphics ) {
      var fs = new FeatureSet();
      var ga = [];
      for (var i = 0; i < lyrGraphics.graphics.length; i++) {
        var g = new Graphic(lyrGraphics.graphics[i].geometry);
        // Add ObjectID if the original layer has an ID field
        if (lyrGraphics.objectIdField) {
          g.setAttributes({"ObjectID":lyrGraphics.graphics[i].attributes[lyrGraphics.objectIdField]});
        }
        ga.push(g);
      }
      fs.features = ga;
      // fs.features = lyrGraphics.graphics;
      return fs;
    }
  });
});
```

5.3. Widget 'Hipoteca' – Calcula tu Hipoteca

```
define(['dojo/_base/declare',
        'jimu/BaseWidget',
        "esri/map",
        "esri/layers/FeatureLayer", "esri/InfoTemplate",
        'esri/geometry/Point',
        "esri/symbols/SimpleMarkerSymbol",
        "esri/Color",
        "esri/graphic",
        "esri/dijit/HorizontalSlider",
        "dojo/_base/lang",
        'dojo/dom'],
function(declare,
  BaseWidget,
  Map,
  FeatureLayer, InfoTemplate,
  Point,
  SimpleMarkerSymbol,
  Color,
  Graphic,
  HorizontalSlider,
  lang,
  dom) {
  //To create a widget, you need to derive from BaseWidget.
  return declare([BaseWidget], {
    //these two properties are defined in the BaseWidget
    baseClass: 'jimu-widget-Hipoteca',
    //propiedad para almacenar el símbolo
    symbol: null,

    postCreate: function() {

      console.log('postCreate');
    },

    startup: function() {
      // var infoTemplate = new InfoTemplate("${Direccion}", "${*}");
      var propiedades = new FeatureLayer("http://services.arcgis.com/9kmFnSwUcKxuGQU9/ArcGIS/rest/services
      // mode: FeatureLayer.MODE_SNAPSHOT,
        outFields: ["*"],
        // infoTemplate: infoTemplate
      });

      this.map.addLayer(propiedades);
      this.map.setInfoWindowOnClick(false);

      var SliderAhorro = new HorizontalSlider({
        labels: ["20%", "100%"],
        value: 30,
        minimum: 20,
        maximum: 100,
        discreteValues: 81,
        showButtons: true,
        onChange: function(value){
            document.getElementById("ahorro2").innerHTML = value;
        }
      }, "sliderAhorro");
      var SliderPlazo = new HorizontalSlider({
        labels: ["1", "10", "20", "30", "40"],
        value: 20,
        minimum: 0,
        maximum: 40,
        discreteValues: 41,
        showButtons: true,
        onChange: function(value){
            document.getElementById("plazo2").innerHTML = value;
        }
      }, "sliderPlazo");

      propiedades.on("click", function(evt){
        var direccion = evt.graphic.attributes.Direccion;
        var precio = evt.graphic.attributes.Precio_total;
        var condicion = evt.graphic.attributes.Condicion;
        var interes = evt.graphic.attributes.Interes;

        var total = parseFloat(precio) + parseFloat(gastos.value);
        var ahorro = (parseFloat(precio) * parseFloat(ahorro2.childNodes[0].data))/100;
        var amortizado = total - ahorro;
        var cuota = parseFloat((amortizado*(interes/100)/12)/(1-(Math.pow((1+((interes/100)/12)),
        (-parseFloat(plazo2.childNodes[0].data)*12))))).toFixed(2);

        document.getElementById("direccion2").innerHTML = direccion;
        document.getElementById("precio2").innerHTML = precio + "€";
        document.getElementById("condicion2").innerHTML = condicion;
        document.getElementById("comision2").innerHTML = interes + "%";

```

```
        // document.getElementById("total2").innerHTML = total + "€";
        document.getElementById("ahorro4").innerHTML = ahorro + "€";
        document.getElementById("amortizado2").innerHTML = amortizado + "€";
        document.getElementById("cuota2").innerHTML = cuota + "€";

        //Calculo amortización para obtener el total de los Intereses
        var pendiente = amortizado;
        var total_intereses = 0;
        var intereses, amortiz;

        while (pendiente > 0){
          intereses = (interes/100)*pendiente/12;
          amortiz = cuota - intereses;
          pendiente = pendiente - amortiz;
          total_intereses += intereses;
        }
        document.getElementById("intereses2").innerHTML = parseFloat(total_intereses).toFixed(2) + "€";

        var total_hipoteca = ahorro + amortizado + total_intereses;
        document.getElementById("total_hipoteca1").innerHTML =  parseFloat(total_hipoteca).toFixed(2) + "€";

      });

      console.log('startup');
    },
```

```
    onOpen: function(){
      this.map.setInfoWindowOnClick(false);
      console.log('onOpen');
    },

    onClose: function(){
      this.map.setInfoWindowOnClick(true);
    },

    onMinimize: function(){
      console.log('onMinimize');
    },

    onMaximize: function(){
      console.log('onMaximize');
    },

    onSignIn: function(credential){
      /* jshint unused:false*/
      console.log('onSignIn');
    },

    onSignOut: function(){
      console.log('onSignOut');
    } //RECORDAR: el último no tiene coma
  });
});
```

5.3.1. CSS Widget

```
h3 {
  margin-top: 0.4em;
  text-align: center;
  font-size: 16px;
}

.esriHorizontalSlider.dijitSlider .dijitSliderProgressBar {
    background: #30cefd;
}

#total2, #ahorro4, #amortizado2, #intereses2 {
  background-color: #acdcf8;
  padding: 10px;
  border-radius: 5px;
  display: inline-block;
}

#cuota2, #total_hipoteca1{
  background-color: #75c6f5;
  padding: 10px;
  border-radius: 5px;
  border-style: solid;
  border-width: thin;
  display: inline-block;
}

.row {
  text-align: center;
}
```

5.4. Widget 'NuevaOficina' – Ubicación de Oficinas

```
define([
  'dojo/_base/declare','jimu/BaseWidget',"dojo/dom-style","dojo/dom",
  "esri/map","esri/geometry/Point","esri/Color","esri/graphic","esri/graphicsUtils",
  "esri/tasks/Geoprocessor","esri/tasks/FeatureSet","esri/symbols/SimpleMarkerSymbol",
  "esri/symbols/SimpleLineSymbol","esri/symbols/SimpleFillSymbol",
  "esri/layers/GraphicsLayer",
  "esri/tasks/ServiceAreaParameters","esri/tasks/ServiceAreaTask",
  "esri/tasks/QueryTask","esri/tasks/query",
  "esri/renderers/SimpleRenderer","dojox/charting/Chart",
  "dojox/charting/axis2d/Default","dojox/charting/plot2d/Pie",
  "dojox/charting/themes/PlotKit/green","dojox/charting/themes/PlotKit/blue",
  "dojox/charting/action2d/MoveSlice","dojox/charting/action2d/Tooltip",
  "dojox/charting/action2d/Highlight",
  "dijit/registry","dojo/_base/array",
  "dijit/layout/BorderContainer", "dijit/layout/ContentPane",
  "dijit/form/HorizontalRule","dijit/form/HorizontalRuleLabels","dijit/form/HorizontalSlider",
  "dojo/ready"
], function(
  declare,BaseWidget,domStyle,dom,
  Map,Point,Color,Graphic,graphicsUtils,
  Geoprocessor,FeatureSet,SimpleMarkerSymbol,
  SimpleLineSymbol,SimpleFillSymbol,
  GraphicsLayer,
  ServiceAreaParameters,ServiceAreaTask,
  QueryTask,Query,
  SimpleRenderer,Chart,
  Default,Pie,
  green,blue,
  MoveSlice,Tooltip,
  Highlight,
  registry,arrayUtils,
  ready
)
{
  var barrios = "http://services.arcgis.com/9kmFnSwUcKxuGQU9/arcgis/rest/services/barrios_enriquecida/FeatureServer/0";
  var clientes = "http://services.arcgis.com/9kmFnSwUcKxuGQU9/arcgis/rest/services/Clientes/FeatureServer/0";
  var polygongraphics = new GraphicsLayer();
  var clientesgraphics = new GraphicsLayer();
  var clickpointgraphic = new GraphicsLayer();


  var clazz = declare([BaseWidget], {
    //these two properties are defined in the BaseWiget
    baseClass: 'jimu-widget-NuevaOficina',
    name: 'NuevaOficina',
    //propiedad para almacenar el símbolo
    symbol: null,
    graphic: null,

    postCreate: function(){
      this.inherited(arguments);
      console.log('postCreate');
    },//POSTCREATE

    startup: function(){
      this.inherited(arguments);
      console.log('startup');
      this.map.addLayer(polygongraphics);

      //DRAW PIE CHART
      var chart = new Chart("pieChart1",{
        title: "Edades",
        titlePos: "top",
        titleGap: -100,
        titleFont: "bold italic 15pt Arial",
        titleFontColor: "black"
      });
      chart.setTheme(blue)
      .addPlot("default", {
        type: Pie,
        font: "normal normal 11pt Tahoma",
        fontColor: "black",
        labelOffset: 30,
        radius: 100
      });
      //ADD PIECHART DATA
      chart.addSeries("Series A", [
        {y: 1, stroke: "black", tooltip: " 15 - 29    "},
        {y: 2, stroke: "black", tooltip: " 30 - 44 "},
        {y: 3, stroke: "black", tooltip: " 45 - 59     "},
        {y: 4, stroke: "black", tooltip: " 60 + "}
      ]);
      var anim_a = new MoveSlice(chart, "default");
      var anim_b = new Highlight(chart, "default");
      var anim_c = new Tooltip(chart, "default");
```

```
86
87      //RENDERER FOR CLIENT POINTS
88      var marker = new SimpleMarkerSymbol();
89      var line = new SimpleLineSymbol();
90      line.setWidth(1);
91      marker.setColor(new Color([255, 255, 255, 1]));
92      marker.setSize(19);
93      marker.setOutline(line);
94      marker.setPath("M16,3.5c-4.142,0-7.5,3.358-7.5,7.5c0,4.143,7.5,18.121,7.5,18.121S23.5,15.143,23.5,11C23.5,6.858,20.143,3.5,
95      marker.setStyle(SimpleMarkerSymbol.STYLE_PATH);
96      var renderer = new SimpleRenderer(marker);
97
98      var map = this.map;
99      var serviceAreaTask, params, clickpoint;
100
101     //////////////////with NASERVER///////////////
102     map.on("click", mapClickHandler);
103
104     params = new ServiceAreaParameters();
105     params.defaultBreaks= [5,10];
106     params.outSpatialReference = map.spatialReference;
107     params.returnFacilities = false;
108
109     serviceAreaTask = new ServiceAreaTask("https://localhost:6443/arcgis/rest/services/PFM/ServiceArea/NAServer/Service Area");
110
111     if (clickpoint) {
112       mapClickHandler(clickpoint);
113     }
114
115     function mapClickHandler(evt) {
116       clickpoint = evt;
117       polygongraphics.clear();
118       clickpointgraphic.clear();
119       //clear existing graphics
120
121       //SYMBOL FOR CLICKPOINT
122       var pointSymbol = new SimpleMarkerSymbol("circle", 10,
123       new SimpleLineSymbol("solid",
124       new Color([88,116,152]), 2),
125       new Color([88,116,152, 1]));
126       var inPoint = new Point(evt.mapPoint.x, evt.mapPoint.y, map.spatialReference);
127       var location = new Graphic(inPoint, pointSymbol);
128
129       var features = [];
130       features.push(location);
131       //add clickpoint to facilities featureset//
132       var facilities = new FeatureSet();
133       facilities.features = features;
134       params.facilities = facilities;
135
136       //SOLVE//
137       serviceAreaTask.solve(params, solveResult);
138       polygongraphics.clear();
139       //COLORES DE POLIGONOS//
140       function solveResult(results){
141         var features = results.serviceAreaPolygons;
142
143         for (var f = 0, fl = features.length; f < fl; f++) {
144           var feature = features[f];
145           if (f === 0) {
146             var polySymbolRed = new SimpleFillSymbol();
147             polySymbolRed.setOutline(new SimpleLineSymbol(SimpleLineSymbol.STYLE_SOLID, new Color([0, 0, 0, 1]), 1));
148             polySymbolRed.setColor(new Color([0, 38, 115, 0.6]));
149             feature.setSymbol(polySymbolRed);
150           }
151           else if (f == 1) {
152             var polySymbolGreen = new SimpleFillSymbol();
153             polySymbolGreen.setOutline(new SimpleLineSymbol(SimpleLineSymbol.STYLE_SOLID,
154               new Color([0, 0, 0, 1]), 1));
155               polySymbolGreen.setColor(new Color([190, 210, 255, 0.6]));
156               feature.setSymbol(polySymbolGreen);
157           }
158           else if (f == 2) {
159             var polySymbolBlue = new SimpleFillSymbol();
160             polySymbolBlue.setOutline(new SimpleLineSymbol(SimpleLineSymbol.STYLE_SOLID, new Color([0, 0, 0, 0.5]), 1));
161             polySymbolBlue.setColor(new Color([1, 1, 1, 0.7]));
162             feature.setSymbol(polySymbolBlue);
163           }
164
165           //CLIENTES 5MINUTOS
166           var queryTask = new QueryTask(clientes),
167           query = new Query(),
168           countOfFeatures = 0;
169           //geometry = serviceAreaPolygons
170           query.geometry = features[1].geometry;
171           query.returnGeometry = true;
172           query.spatialRelationship = Query.SPATIAL_REL_INTERSECTS;
173           query.outSpatialReference = {"wkid": 102100};
174
175           queryTask.execute(query, function (results) {
176             if (results.features && results.features.length > 0) {
177               dojo.forEach(results.features, function (feature) {
178                 countOfFeatures++;
179               });
180             }
181             document.getElementById("clientes5").innerHTML = countOfFeatures;
182           });
183
184           //CLIENTES 10MINUTOS
185           var queryTask10 = new QueryTask(clientes),
186           query10 = new Query(),
187           countOfFeatures10 = 0;
```

```
188
189             query10.geometry = features[0].geometry;
190             query10.returnGeometry = true;
191             query10.spatialRelationship = Query.SPATIAL_REL_INTERSECTS;
192             query10.outSpatialReference = {"wkid": 102100};
193
194             queryTask10.execute(query10, function (results) {
195               if (results.features && results.features.length > 0) {
196                 dojo.forEach(results.features, function (feature) {
197                   countOfFeatures10++;
198                 });
199                 var result = results.features;
200
201                 //DRAW POINTS FOR CLIENTS WITHIN 10M POLYGON
202                 clientesgraphics.clear();
203                 if(result.length!=0){
204                   for (var i = 0; i<result.length; i++){
205                     clientesgraphics.setRenderer(renderer);
206                     clientesgraphics.add(result[i]);
207                   }//for each client
208                 }//if client within
209               }//if results is more than 0
210               document.getElementById("clientes10").innerHTML = countOfFeatures10;
211             });
212
213             polygongraphics.add(feature);
214
215             map.setExtent(graphicsUtils.graphicsExtent(polygongraphics.graphics), true);
216           }//for each polygon
217         }//solve
218
219         //QUERY BARRIOS
220         var queryTaskBarrios = new QueryTask(barrios),
221         queryBarrios = new Query();
222
223         queryBarrios.geometry = evt.mapPoint;
224         queryBarrios.returnGeometry = true;
225         queryBarrios.outSpatialReference = {"wkid": 102100};
226         queryBarrios.outFields = ["NOMBRE", "PPPC_CY", "UNEMP_CY","PAGE02_CY","PAGE03_CY","PAGE04_CY","PAGE05_CY"];
227
228         queryTaskBarrios.execute(queryBarrios, function (results) {
229           var result = results.features[0];
230           var barrio = result.attributes.NOMBRE;
231           var income = Math.round(result.attributes.PPPC_CY);
232           var unemployment = result.attributes.UNEMP_CY;
233           var agegroup1 = result.attributes.PAGE02_CY;
234           var agegroup2 = result.attributes.PAGE03_CY;

235           var agegroup3 = result.attributes.PAGE04_CY;
236           var agegroup4 = result.attributes.PAGE05_CY;
237
238           document.getElementById("barrio").innerHTML = barrio;
239           document.getElementById("income").innerHTML = "€" + income;
240           document.getElementById("unemployment").innerHTML = unemployment;
241
242           //DRAW PIE CHART
243           chart.updateSeries("Series A", [
244             {y: agegroup1, stroke: "black", tooltip: " 15 - 29   "},
245             {y: agegroup2, stroke: "black", tooltip: " 30 - 44   "},
246             {y: agegroup3, stroke: "black", tooltip: " 45 - 59  "},
247             {y: agegroup4, stroke: "black", tooltip: " 60 +  "}
248           ]);
249           chart.render(); //DRAW CHART
250
251         }); //querytask BARRIOS
252
253         clickpointgraphic.add(location);
254
255       } //onclick
256       this.map.addLayer(clientesgraphics);
257       this.map.addLayer(clickpointgraphic);
258
259     },//STARTUP
260
261     onOpen: function(){
262       //centerandzoom used when developing in demo-widgets
263       // this.map.centerAndZoom([-3.643683,40.416374],12);
264       this.map.setInfoWindowOnClick(false);
265       console.log('onOpen');
266     },
267     //
268     onClose: function(){
269       clientesgraphics.clear();
270       clickpointgraphic.clear();
271       polygongraphics.clear();
272       this.map.setInfoWindowOnClick(true);
273
274       console.log('onClose');
275     }
276   });
277   return clazz;
278 });
```

5.4.1. CSS Widget

```css
.jimu-widget-nuevaoficina div:first-child{
  color: red;
}

span.label.label-success{
  font-size: 1.5rem !important;
  background-color: #349CD7 !important;
}

tr{
  font-size: 1.5rem;
}

th span.label.label-info{
  font-size: 1.5rem;
}

.tablehead{
 font-size: 1.7rem;
 font-weight: bold;
 font-style: italic;
}

.space{
  border: none;
}

 .table>tbody>tr>td, .table>tbody>tr>th,
 .table>tfoot>tr>td, .table>tfoot>tr>th,
.table>thead>tr>td, .table>thead>tr>th {
  border:none !important;
}

rect{
  fill:none !important;
}
```

5.5. Cambio CSS StoryMap

```css
/*LOGO BANKGISTER*/
#logo{
display: block;
text-align: center;
width: 55%;
margin-left: auto;
margin-right: auto;
padding: 1rem;
}

/*COLOR SCALEBAR*/
.calcite .esriScalebarLabel{
color:white !important;
}

/*AJUSTAR EL HEADER*/
.sectionPanel .header {
  height:auto !important;
  }
.header{
  background-color: white !important;
  }


/*Cambiar Leyenda*/
.esriLegendServiceLabel{
  Color:white;
}

/*FONT DEL TITULO*/
.title span{
  font-size: 29px !important;
  font-weight: bold !important;
}

/*CENTRAR TITULO*/
.title p{
  text-align: center;
}

/*CENTRAR TITULO FIXED*/
.sectionPanel .appTitle p {
    font-size: 24px!important;
    font-weight: 400;
    text-align: center;
}

/*QUITAR LINEA BORDER DEL HEADER*/
.sectionPanel .separator {
  border-bottom: none !important;
}

#playControls {
  position:fixed !important;
}
```

6. Bibliografía

Para la consulta acerca del desarrollo de las aplicaciones:

- https://developers.arcgis.com/javascript/3/
- https://developers.arcgis.com/web-appbuilder/
- https://dojotoolkit.org/
- https://developer.mozilla.org/en-US/
- https://github.com/
- http://getbootstrap.com/
- https://www.w3schools.com/
- http://stackoverflow.com/
- https://geonet.esri.com/
- http://desarrolladores.esri.es/category/tutorial/javascript-tutorial/
- https://libro.cursohtml5desdecero.com/
- https://www.gitbook.com/book/mundogister/seminario_dojo/details
- http://doc.arcgis.com/es/web-appbuilder/
- https://developers.arcgis.com/javascript/3/samples/playground/index.html
- https://www.esri.com/training/catalog/search/

Para la consulta acerca del sector económico:

- http://www.bde.es/bde/es/
- http://www.ecb.europa.eu
- https://www.idealista.com
- http://www.mviv.es/por-que-los-bancos-tienen-casas-y-de-donde-se-las-consiguen/
- https://www.aulaingles.es/blog/diferencias-entre-moneda-y-divisa/
- https://www.bbva.es/particulares/ahorro-inversion/bolsa-mercados/cambio-divisas-moneda/index.jsp
- https://www4.caixabank.es/apl/divisas/verTodos_es.html?JSESSIONID=JnTO_7NelDSaMAXB9h7-byp