**Big Data Architecture – Práctica**

**Alumno: Guillermo Barrio Colongues**

**Brainstorm**

El objetivo es montar una plataforma DAaaS de estudio de resultados electorales en España con, al menos, los siguientes objetivos:

* Storage de los resultados electorales a nivel de sección electoral de los comicios al Congreso en los últimos, pongamos, 17 años, es decir, desde las generales de 2004.
* Storage de datos socioeconómicos presentes y pasados, que puedan ser razonablemente (co)relacionados geográfica y temporalmente con los resultados electorales
* Visualización geográfica de las secciones electorales, y de una comparativa temporal de los resultados, a petición del cliente.
* Visualización de los resultados electorales en función de los datos socioeconómicos, a nivel geográfico, a elección del cliente; por ejemplo, cuál ha sido la evolución del resultado electoral en las secciones con una renta per cápita entre 24 y 30 mil euros.
* Modelización de un determinado territorio (sección, distrito, municipio…), a elección del cliente, en función del resto de territorios, se supone que más pequeños que el dado.
* Interacción posterior con el cliente. Ser capaz de, al menos:
  + Recabar opiniones y recomendaciones
  + Contactos vía email
* Diseño de una arquitectura compatible con una futura monetización del sistema.

**Diseño del DAaaS**

### **Definición la estrategia del DAaaS**

*Definir el catálogo de servicios que proporcionará la plataforma DAaaS, que incluye incorporación de datos, limpieza de datos, transformación de datos, datapedias, bibliotecas de herramientas analíticas y otros.*

Los servicios que proporcionará la plataforma se han mencionado brevemente en el brainstormig anterior.

He pensado que el formato, o arquitectura a grandes rasgos deberá ser una página web que proporcione al cliente el resultado de sus querys (eso es realmente lo que el cliente va a introducir en la plataforma) de una forma rápida, pongamos en segundos si es posible.

Los datos provendrán de los siguientes organismos, todos de carácter público, y los datos lo son también por definición:

* **Ministerio del Interior:** Datos de los resultados electorales, que son los oficiales y, afortunadamente, los definitivos.
* **INE:** Definiciones geográficas de las secciones electorales, que vienen en varios formatos. En este caso los estos datos NO son constantes en el tiempo, sino que se ajustan cada 6 meses. El INE proporciona datos socioeconómicos de todo tipo, tales como económicos, poblacionales, etc…
* **Agencia Tributaria:** Datos de carácter económico, tales como rentas por declarante distribuidas por códigos postales.
* **Eurostat:** Es posible que se encuentren datos socioeconómicos interesantes nivel agregado para distintos territorios dentro de España.
* **Ministerio de Asuntos Económicos, Ministerio de Inclusión, Seguridad Social y Migraciones:** Más datos económicos, en especial la SS y datos de desempleo a nivel municipal

Y pienso que estos son solo una muestra. Lo importante será, por supuesto, conectarlo unos con otros, a ser posible a nivel de municipio o distrito, y deducir de alguna manera su valor a nivel de sección.

Una cosa importante es que, en principio, no haría falta hacer un “full fledged” scrapping, pues los datos en la mayor parte de casos se pueden descargar a partir de las propias páginas webs/API, en especial el INE. Por otro lado, en algún caso como la Agencia Tributaria o el Ministerio del Interior, quizás merezca la pena ser prudente. Lo que sí ocurre es que muchos de esos datos se actualizan de forma mensual y con fechas conocidas de ante mano, algunos incluso cambias retrospectivamente. Se podrá pues automatizar de alguna manera la descarga de estos datos mediante Scrappy o Selenium, por ejemplo.

Una ventaja de esto último es que la limpieza de los datos puede ser aceptable, al menos en principio. Otra cosa será, como hemos dicho, la conexión entre ellos.

Como herramientas analíticas, sin duda aprenderé (más bien apreHEnderé) técnicas de análisis, etc. De momento estoy haciendo mis pinitos con bibliotecas como Pandas, pero imagino que debe ser solo el principio. Qué duda cabe que para, por ejemplo, modelizar un territorio harán falta herramientas más potentes como tensor Flow, o Scikit Learn como mínimo, y esas son solamente las que actualmente conozco.

### **Arquitectura DAaaS**

*Definir la selección de componentes, la definición de procesos de ingeniería y el diseño de interfaces de usuario. Diseño y ejecución de Proofs-of-Concept (PoC) para demostrar la viabilidad del enfoque DAaaS.*

Intentaré que la arquitectura esté alojada lo más posible dentro de Google Cloud Platform (GCP), algo que pienso que debería ser posible, a fin de cuentas, desde fuera, no parece que lo que pienso hacer sea tan, tan difícil.

Los componentes que, al menos debería haber serían los siguientes:

* **Página Web.** La página web que quiera tener habrá que crearla. Si es aceptable, al menos al principio, me inclinaría por hacerla en Wix o algún sitio parecido, ya que personalmente no sé cómo diseñarlas (de momento).
* **Web Hosting.** GCP ofrece varias posibilidades. Quizás la que de momento encajaría sería una sencilla, por ejemplo, WordPress en el Computer Engine, que dice ser la solución ideal para sitios web de tráfico bajo y medio, como sería el caso sin duda.
* **Cloud Function de inicio del scrapper.** Si es posible se podría utilizar este elemento para descargar datos socioeconómicos de carácter periódico, con fechas de emisión conocidas. Al ser los datos electorales definitivos, y que aparecen cada varios años, los puedo introducir yo personalmente.
* **Bucket Datos.** Los datos electorales, y el histórico de los datos socioeconómicos se alojarán en un bucket a tal efecto.
* **Cluster Dataproc.** Se creará un cluster dataproc, con una estructura adecuada (master + n workers, CPU, etc…), que se encargue de la gestión inicial de los datos, tales como una tarea que levante un proceso de mapreduce inicial para casar los datos electorales con los socioeconómicos produciendo un dataset digamos que enriquecido. El dataset resultado del mapreduce se exportarán a una tabla o varias tablas en HIVE. Se procurará que la estructura de los datos sea compatible con una basa de datos relacional, ya que la mayor parte de datos deberían tener esa estructura.
* **Representación geográfica.** Se utilizarán las herramientas de representación geográfica presentes en GCP, quizás Fillium u otras. Desde luego, deberán ser compatibles con los datos proporcionados con el INE. Habrá que asociar las secciones electorales presentes ahora en HIVE con estos datos geográficos, que pueden ser no relacionales al tener una estructura key: dato (json). Es una posibilidad.

Estos serían los elementos que debería tener la plataforma activados cuando se encienda. Ahora supongamos que un cliente hace una query. Si se trata de una sencilla como el histórico electoral de un municipio no habría problema y HIVE a través de beeline se encargaría de ello. Sí que lo habría si pidiese modelar un territorio. En ese caso pienso que haría falta:

* **Centro de cálculo**. Llamémosle así. Se trataría de hacer la labor que supongo que dataproc en conjunto no sería capaz, y de los que se encargaría una red neuronal a través de Tensor Flow, por ejemplo. Incluso labores menos complicadas, tales como el estimar datos socioeconómicos a nivel de sección en función de los datos de municipios, utilizando KNN me imagino, serían efectuados por el centro de cálculo. Imagino que GCP tiene esta funcionalidad, que deberé programar, por supuesto. El centro de cálculo haría una infinidad de queries a HIVE, y enviaría la solución al web host, y éste al cliente.

Otros elementos que harían falta al interaccionar con clientes incluirían:

* **Bucket Input/Output.** Convendría almacenar las queries y su resultado que se fueran produciendo en un bucket concreto, para su posterior estudio estadístico. Estos datos se podrían trasladar a HIVE, la duda es si dentro del cluster ya creado u, quizás menos probable, otro que se encargue en exclusiva de ello.
* **Bucket Comentarios.** Se podría crear un bucket que almacenase los comentarios. Si hubiese muchos, algo no muy probable, se podría aplicar un Elastic Search aquí que los estudie según palabras clave. Es una posibilidad que podemos implementar aunque solo fuese para aprender a desplegar esta herramienta.
* **Cloud Function de Resumen de Actividad.** Se encargará de forma regular, tal como una vez al día, del envió a mi email de la evolución de los bucket Input/Output y Comentarios.
* **Bucket Contactos.** Quizás se pueda llevar a cabo esto dentro del propio web host, pero en algún sitio se deben almacenar los datos de los clientes que estén interesados que se les contacte con novedades, etc…
* **Cloud Function de envío de emails.** Una cloud function podría automatizar el envío de emails.
* **Seguridad.** Debemos evitar que algún listo se ponga a hacer queries de forma automática y acabe por tirarnos la web o costarnos un ojo de la cara por el uso del CPU, un coste que supongo tiene un componente variable.

### **DAaaS Operating Model Design and Rollout**

*Personalizar los modelos operativos DAaaS para cumplir con los procesos, la estructura organizacional, las reglas y el gobierno de los clientes individuales. Realizar seguimiento de consumo y mecanismos de informe.*

El operation model se podría resumir de la forma siguiente, aunque algo de él se puede inferir del apartado anterior:

1. **Web Hosting.** En principio, la página web deberá estar activa continuamente. Sí sería conveniente que ésta enviase emails automáticos cada 12 horas para asegurarse que sigue activa. Supongo que GCP tiene prevista esta funcionalidad.
2. **Actualización de datos.** El scrapper debería activarse si le hemos programado las fechas concretas en que estarían disponibles los nuevos datos. Con todo, quizás sería más sencillo decirle que haga una carga general cada semana, pongamos todos los sábados. Se podría implementar un trigger (Google Storage) que me avisase via email si ha habido nuevos datos cargados. Los datos electorales los puedo cargar yo personalmente al bucket datos.
3. **Cluster Dataproc.** Pienso que el cluster deberá estar activo mientras lo esté la web, pues alguien se tiene que encargar de las queries. Va a ser algo caro, pero no veo opción de momento.
4. **Queries “normales”.** Las queries que sean solo de consulta de datos históricos las puede resolver HIVE solito. Con todo, se activará el módulo de representación geográfica para plotear el territorio pedido por el cliente. Por cierto, este debería ser capaz de introducir su query a través de la representación geográfica directamente. HIVE enviaría la respuesta al web host, y este al cliente. La query y su resultado se almacena en el bucket Input/Output.
5. **Queries “Complicadas”.** Estas son las que requieren que se active el Centro de cálculo, que deberá estar también activo siempre. El Centro resolvería la query tras hacer, se supone, miles de queries a HIVE-beeline. Enviaría la solución al web host y a los buckets.
6. **Resumen de actividad.** El cloud function de resumen de actividad nos enviaría su reporte de forma regular, una vez al día, por ejemplo.

### **Desarrollo de la plataforma DAaaS.**

*Construcción iterativa de todas las capacidades de la plataforma, incluido el diseño, desarrollo e integración,* ***pruebas****, carga de datos, metadatos y población de catálogos, y despliegue.*

La página web podría ser sencilla al principio, y como hemos dicho, se podría implementar con Wix. En un futuro aspiro a que sea un diseño personal mío.

El scrapper lo diseñaría con Scrappy en principio, dentro de su cloud function.

Espero utilizar Python para el Centro de cálculo, integrando Tensor Flow. Con todo, es muy probable que a medida que avance el bootcamp puedan conocer de herramientas que, para este proyecto en concreto, pudiesen ser resultar más útiles.

Link a Diagrama: