Tesis:

Infraestructura para Ciudades Inteligentes en la Nube: Solución para procesamiento de datos de las Ciudades, aplicado al estudio de muchedumbres.

Ing. Marisabel Guadalupe Rodríguez Bilardo

Tesis presentada para cumplir con los requisitos finales para la obtención del título de Magister en Telecomunicaciones

Director de Tesis: Magister Claudio Muñoz

Lectores:
PhD Leticia Gómez
PhD Silvia Gómez
PhD Daniela Lopez De Luise

2017

Abstract

Este trabajo estudia las alternativas de recolección y análisis de grandes cantidades de datos, de tal manera que sea más accesible para los gobiernos de las ciudades realizar inferencias útiles sobre la vida de los habitantes, que redunden en un mejoramiento de la calidad de vida.

Se presenta un método a fin de realizar un prototipo para recolectar información proveniente de crowdsensing, que permite consolidar datos y realizar inferencias y visualización dinámica en tiempo real tomando como base de análisis la información pública de la ciudad de Buenos Aires.

Tabla de Contenidos

Capítulo 1

Introducción

La población mundial que vive en las ciudades ha experimentado un crecimiento sin precedentes durante el siglo pasado. Mientras que sólo el 10% de la población vivía en ciudades durante 1900, hoy este porcentaje corresponde al 50% y se proyecta que aumente más allá de esa cifra. Por lo tanto, el desarrollo sostenible desempeña un papel crucial en el desarrollo de la ciudad.

Mientras que sólo el 2% de la superficie del Mundo está ocupada por entornos urbanos, las ciudades contribuyen al 80% de las emisiones mundiales de gas, al 75% del consumo mundial de energía.

Los recientes avances en Internet de las Cosas (Internet of Things en inglés) y el avance en el procesamiento de datos, proporcionan una visión hacia el futuro de nuestro planeta y revelan interesantes puntos de vista sobre muchas cosas inteligentes: ciudades inteligentes, casas inteligentes, coches inteligentes, y otros espacios inteligentes, tales como centros comerciales, lugares de trabajo, escuelas, hoteles, y mucho más.

Impulsado por una oferta de revolución tecnológica "de muchas cosas de baja potencia y casi todo inalámbrico" se ha podido, en tan sólo una década, avanzar en la construcción de sistemas inteligentes y prototipos impresionantes que mejoran la calidad de vida, aumentan el conocimiento de los recursos y el medio ambiente, y enriquecen la experiencia de usuario.

Sin embargo, la creación de prototipos es una cosa y las implementaciones reales a gran escala son otra. La escala masiva de sensores y dispositivos que serán desplegados en las ciudades inteligentes del futuro será un reto. Sin un ecosistema y una arquitectura escalable y comprensible, será extremadamente difícil de manejar un programa de expansión masiva de Internet de las Cosas.

Hoy en día las necesidades de las ciudades para lograr esta mejora en la calidad de vida pueden obtenerse con mucha certeza. Pero hay un problema que radica en que son diferentes organismos y fuentes los que obtienen los datos, todavía no hay sistemas implementados que puedan cruzar la gran cantidad de información y puedan proveer claves más exactas sobre el comportamiento de los distintos factores económicos y políticos, que tengan impacto en la sociedad, ya sea desde el punto de vista económico como ecológico, político o de participación ciudadana.

El presente trabajo propone investigar sobre una posible mejor y más eficiente infraestructura de recolección y análisis de datos, que permita recibir información de distintas fuentes y procesarla de tal forma que se crucen los diversos factores económicos, políticos y sociales de una ciudad, produciendo una mejora tangible en su desarrollo y funcionamiento.

Motivación

Hoy en día hay una gran cantidad de fuentes de información, pareciera ser que los datos necesarios para solucionar los problemas de las ciudades y que hagan posible mejorar la calidad de vida de los ciudadanos se pueden llegar a recolectar, pero se está muy lejos de poder

analizarla y procesarla de manera útil para la mayoría de las variables urbanas.

Podremos evaluar si una ciudad es inteligente o no, teniendo en cuenta los siguientes aspectos:

- Tecnología
- Transporte
- Energía
- Información abierta
- Economía

Entonces las preguntas clave para analizar si una ciudad es inteligente podrían formularse como:

- ¿Cuáles son las principales fortalezas y debilidades de las principales ciudades inteligentes?
- ¿Qué estrategias deben buscar las partes interesadas para aplicar al desarrollar proyectos de ciudades inteligentes?
- ¿Cuáles son las tendencias clave que configuran los mercados de políticas y servicios de las ciudades inteligentes?
- ¿Cuáles son las perspectivas para la energía inteligente de la ciudad, los servicios de tránsito y la iluminación?
- ¿Cuál es la oportunidad económica para proyectos urbanos inteligentes?

Todas estas preguntas pueden ser contestadas, con suficiente información, y todas las ciudades del Mundo pueden avanzar si trabajan en analizar datos.

En la Ciudad Autónoma de Buenos Aires, “Buenos Aires Data” (http://data.buenosaires.gob.ar/) es una iniciativa de datos públicos y transparencia que tiene las siguientes características:

1. Acceso programático vía RESTful API a los conjuntos de datos.
2. Visualización de los conjuntos de datos en formato de lista, gráficos y mapas interactivos.
3. Diferentes formatos.
4. Filtrado de datasets.

Hay también aplicaciones mobile oficiales, que si bien ayudan a mejorar la calidad de vida de la gente, no hacen uso de crowdsensing para obtener insights relevantes o para lograr ahorros significativos a partir de datos objetivos.

Algunas podrían relevar datos de esa manera, pero al probarlas no parecen recopilar datos masivos.

La aplicación que más parece aprovechar información de usuarios es “AllGreenUp”. Es internacional y funciona como una red social para medir el impacto ambiental de cada persona, como se observa en las figuras.

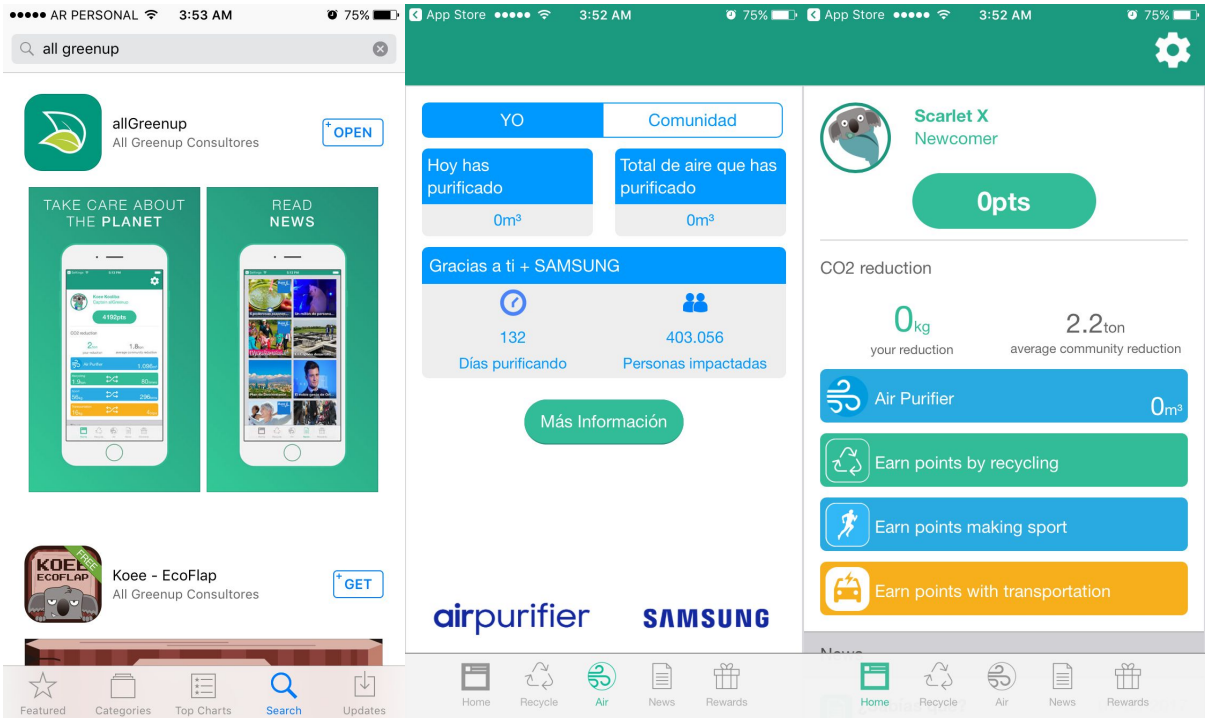

Figura 1: Aplicación AllGreenUp, para medir parámetros ecológicos.

Por otro lado, aplicación "Denuncia Vial", toma los datos pero no reúne información por crowdsensing en forma por lo menos abierta.

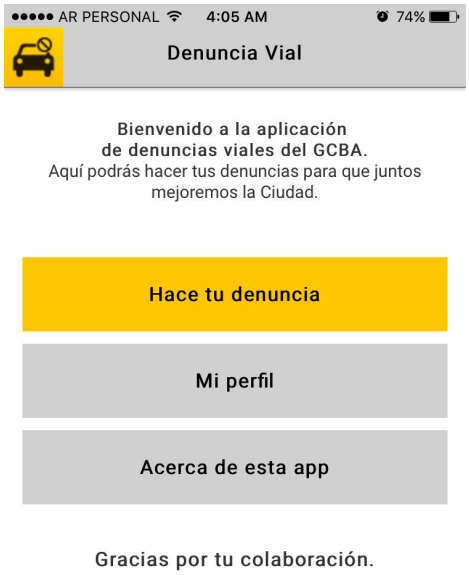

Figura 2: Aplicación "Denuncia Vial".

La aplicación "Localizar emergencia" solo ayuda a marcar rápidamente un número de emergencias y a enviar la ubicación de un evento, sin crowdsensing abiertamente utilizado.

Figura 3: Aplicación "Localizar Emergencia".

A continuación se muestra un detalle de dichas aplicaciones del Gobierno de la Ciudad tomadas del sitio http://data.buenosaires.gob.ar/:

Aplicaciones del Gobierno de la Ciudad de Buenos Aires	Descripción
BA Taxi	Exclusiva para taxistas. Controla la seguridad y calidad del servicio, y optimiza el contacto con los pasajeros para brindar un servicio más transparente.
BA Cómo llego	Permite consultar cómo llegar de un punto a otro en la Ciudad de Buenos Aires usando colectivo, tren, subte, bici, caminando o en auto.
BA Ecobici	Toda la información de ciclovías y estaciones del sistema público de la Ciudad.
BA Subte	Muestra el estado del subte en tiempo real y permite buscar cómo viajar usando el subte.
BA Wifi	Muestra la conexión WiFi gratis del Gobierno de la Ciudad más cercana.

BA 147	Solicitudes para el Gobierno de la Ciudad de Buenos Aires.
BA Medios	Para escuchar las radios de Buenos Aires y mirar el Canal de la Ciudad. Streaming de Radio Ciudad AM 1110 y FM 92.7 La 2x4.
BA Denuncia Vial	Denuncias de vehículos mal estacionados en Buenos Aires. *Podría usar y no usa crowdsensing para agregar información.
BA Ferias	Muestra los productos en las ferias y mercados de la Ciudad de Buenos Aires
BA Vacunación	La aplicación permite llevar el control de las vacunas aplicadas a cada uno de los hijos del grupo familiar, las vacunas que faltan aplicar, visualizar los centros de vacunación más cercanos, e información general de todas las vacunas del calendario oficial (datos de las edades de aplicación, que enfermedades se previenen, consejos de la vacunación).
BA Turismo	Para buscar todas las actividades Culturales, Deportes, Restaurantes, Hoteles que hay en la Ciudad de Buenos Aires.
BA CoPS	La aplicación TURNOS COPS permite visualizar la cartilla online, programar turnos médicos, recibir recordatorios y administrar los turnos de una familia.
BA Accesible	Permite señalar los lugares que cuentan con apoyos de accesibilidad, para que quienes los necesitan sepan dónde contar con ellos.
Noches BA	Agenda dinámica de las actividades llevadas a cabo en "Las Noches de Buenos Aires".
Biciplus	Biciplus ayuda a recorrer Buenos Aires en bicicleta de manera más simple y divertida.
Dónde	Dónde? La aplicación que ayuda a disfrutar mucho más de la Ciudad Autónoma de Buenos Aires brindando la información necesaria para encontrar desde puntos de accesos libres de WI-FI hasta Centros Médicos.
Localizar Emergencia	La aplicación contiene un acceso directo a los números de emergencia más importantes de Capital Federal: - Policia, Same, Bomberos, Defensa Civil, Trata de personas, Subte alerta. Al iniciar una llamada de emergencia, mostrará en pantalla nuestra localización para ser provista al operador telefónico. Útil para aquellos casos en donde un ciudadano sufre un accidente y quien lo asiste no tiene al alcance el nombre de las calles o la misma carece de señalización. *Podría usar crowdsensing, pero no parece usarlo, solo disponible para Android.

Me voy en bici	Sirve para consultar la disponibilidad de bicis y el mapa de ciclovías desde tu celular. Esta aplicación muestra la disponibilidad bicicletas del servicio "Mejor en Bici" de la Ciudad Autónoma de Buenos Aires en tiempo real ordenándolas por distancia usando GPS. También cuenta con el mapa de ciclovías protegidas, con sus respectivas estaciones, y los mejores locales para alquilar bicicletas para los turistas. Los datos son provistos por el Mapa de ciclovías de "Mejor en Bici", Gobierno de la Ciudad de Buenos Aires.
Cruz Roja Argentina	Aplicación desarrollada por la Cruz Roja Argentina para saber qué hacer en caso de un accidente, enfermedad o fenómeno natural adverso.
AllGreenUP	Ayuda a medir tu impacto sobre el medio ambiente y te premia por cuidar el planeta. *Es una aplicación internacional, que usa crowdsensing porque recolecta información de todos los usuarios que se registran.

Tabla 1: Aplicaciones del Gobierno de la Ciudad de Buenos Aires.

El siguiente gráfico muestra la lista de las ciudades más inteligentes, y las razones por las cuales han sido elegidas. Si bien las ciudades más avanzadas han invertido en el mejoramiento de la calidad de vida de los ciudadanos, la información más objetiva proveniente de crowdsourcing innegablemente produciría todavía mayor impacto positivo.

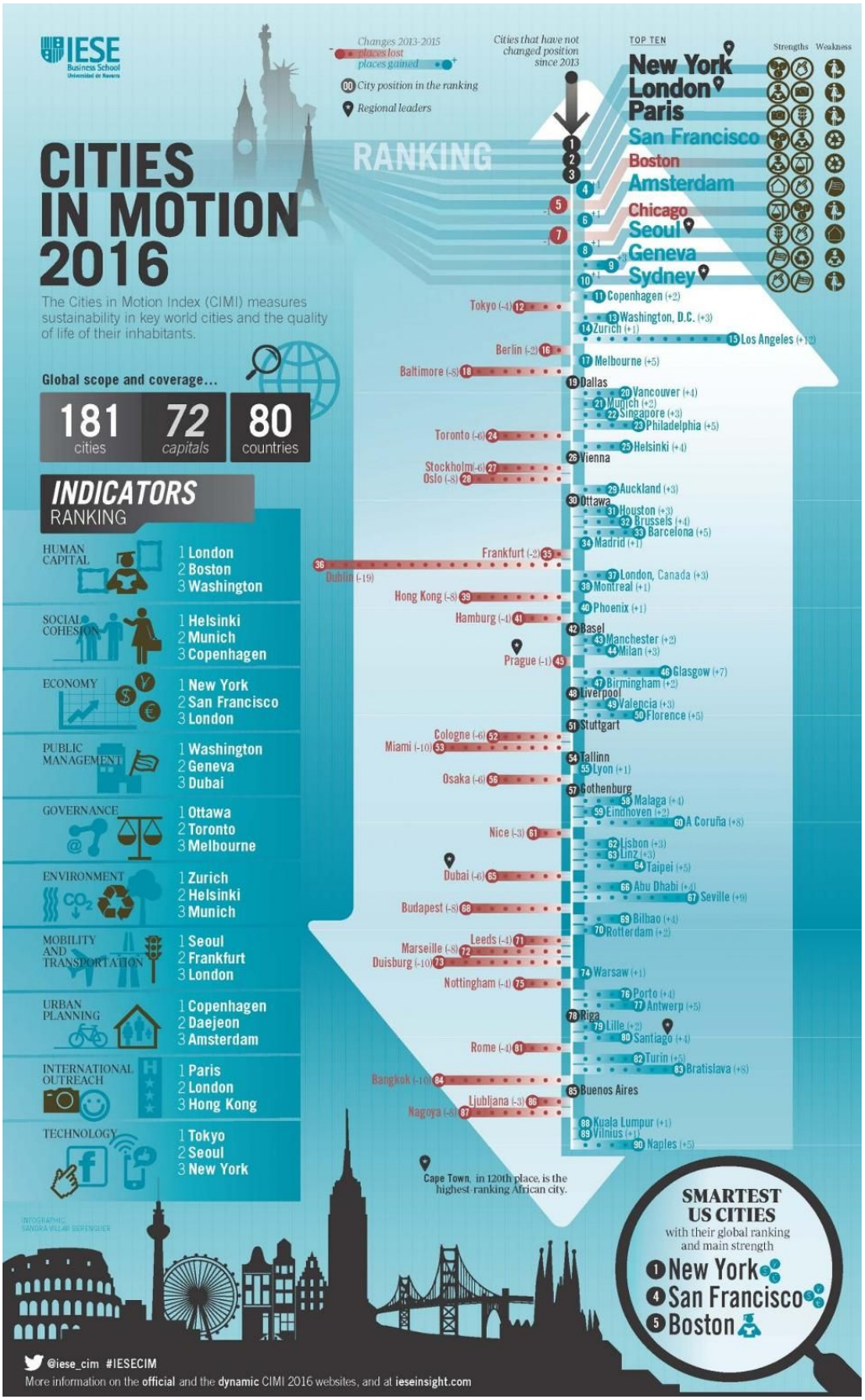

Figura 4: Ranking de ciudades inteligentes 2016 en el Mundo.

Objetivo

El objetivo de este trabajo es hacer un estudio de las herramientas disponibles, para facilitar el acceso de los organismos de gobierno, o público en general, para llegar a obtener información útil, relacionando los datos recopilados en una ciudad, y así mejorar la calidad de vida de la gente que la habita, teniendo en cuenta las inferencias para tomar decisiones inteligentes en los ámbitos económicos, ecológicos y sociales de la comunidad.

En particular se considera crowdsensing por ser una disciplina de recolección masiva de datos que se vale muchas veces de los sensores presentes en los smartphones, porque su disponibilidad es prácticamente absoluta entre los habitantes de las grandes ciudades.

Hipótesis del trabajo

Se han elegido las siguientes hipótesis para desarrollar el presente trabajo:

I) Las decisiones políticas y también políticas ambientales que se toman en las ciudades parecen ser más bien arbitrarias o provenientes de estudios sobre datos estadísticos, pudiéndose usar herramientas al alcance de todos los ciudadanos para obtener insights en forma colaborativa y más precisa.

II) La barrera de acceso al monitoreo de eventos en las ciudades donde cerca del 80% de los ciudadanos poseen smartphones es cada vez más baja porque se necesita solamente integrar los elementos ya presentes en el mercado.

Delineamiento de la tesis

El trabajo se centra en el capítulo 2, en evaluar el estado del arte en la disciplina de la recolección de datos de multitudes, con respecto a diferentes factores, y en el capítulo 3, generar un modelo básico con los conceptos de recolección de datos on-line en tiempo real, para su posterior procesamiento.

Esta infraestructura de procesamiento que podría llegar a cientos de Terabytes, no necesariamente conlleva un gasto inaccesible, ya que las nuevas tecnologías en la nube hacen posible que sea cercano para el público en general, con las habilidades técnicas y de ingeniería suficientes, pero ya no con una gran inversión en hardware o insumos costosos. La siguiente figura ilustra la secuencia de conceptos en los capítulos 2 y 3.

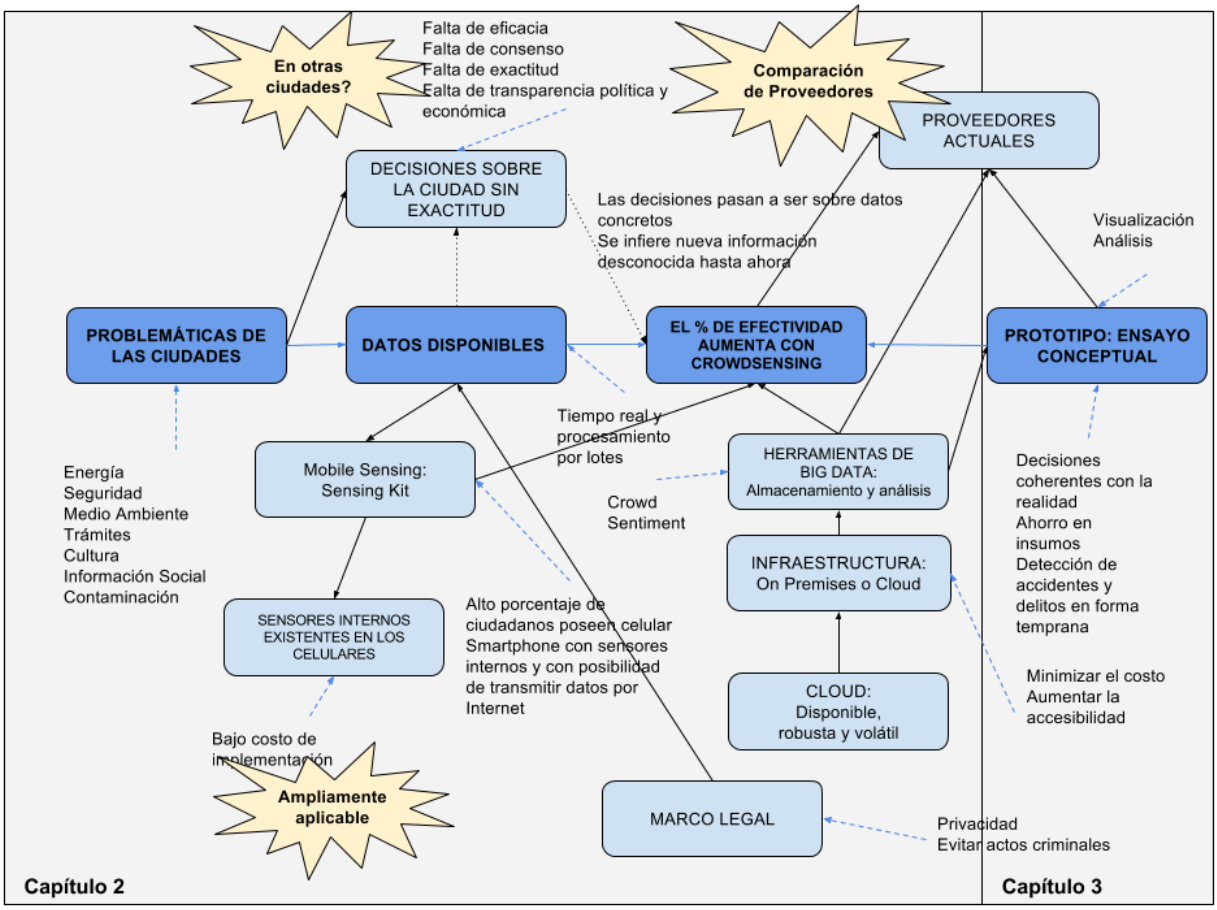

Figura 5 - Diagrama conceptual de la tesis.

Capítulo 2

Este capítulo describe los componentes del modelo de recolección de información a través de los celulares, y la adquisición de los datos para su análisis, comparando las diferentes herramientas disponibles en el mercado.

Sensores presentes en los celulares

El objetivo de la investigación es recopilar información sobre los posibles sensores ya presentes en los celulares smartphones, de tal manera que sea económico recolectar evidencias de mucha más gente, ya que cada persona habitante de la ciudad, posee prácticamente un aparato o más, cada vez de mejor versión de hardware, a medida que pasan los años.

Existe un modelo recientemente desarrollado, denominado Sensing as a Service (S2 aaS), que pone a disposición de los desarrolladores y usuarios finales los datos públicos recogidos. Con S2 las empresas aaS ya no necesitan invertir y adquirir infraestructura para realizar una campaña de detección. IoT y mobile crowdsensing son factores clave en el modelo S2 aaS.

La eficiencia de los modelos de S2 aaS se define en términos de los ingresos obtenidos de la venta de datos versus los costos de la campaña de detección, que incluyen los costos de reclutamiento e indemnización de los participantes por su participación. De todas maneras si bien este modelo existe, el presente trabajo se trata de un análisis de la tecnología interna de los sistemas.

El teléfono inteligente (smartphone en inglés) es un tipo de teléfono móvil construido sobre una plataforma informática móvil, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional. El término inteligente, que se utiliza con fines comerciales, hace referencia a la capacidad de usarse como un computador de bolsillo, y llega incluso a reemplazar a una computadora personal en algunos casos.

Generalmente, los teléfonos con pantallas táctiles son los llamados teléfonos inteligentes, pero el soporte completo al correo electrónico parece ser una característica indispensable encontrada en todos los modelos existentes y anunciados desde 2007. Casi todos los teléfonos inteligentes también permiten al usuario instalar programas adicionales, habitualmente incluso desde terceros, hecho que dota a estos teléfonos de muchísimas aplicaciones en diferentes terrenos; sin embargo, algunos teléfonos son calificados como inteligentes aun cuando no tienen esa característica. Además cuentan con al menos 14 sensores para tomar datos del ambiente y del usuario.

Figura 6 - Sensores presentes en Smartphones - Fuente: http://www.transportationtechnologyventures.com.

Entre otros rasgos comunes está la función multitarea, el acceso a Internet vía Wifi o redes 4G, 3G o 2G, función multimedia (cámara y reproductor de videos/mp3), a los programas de agenda, administración de contactos, acelerómetros, GPS y algunos programas de navegación, así como ocasionalmente la habilidad de y leer documentos de negocios en variedad de formatos como PDF y Microsoft Office.

En la figura se muestra la evolución de la penetración de los smartphones en diferentes mercados mundiales.

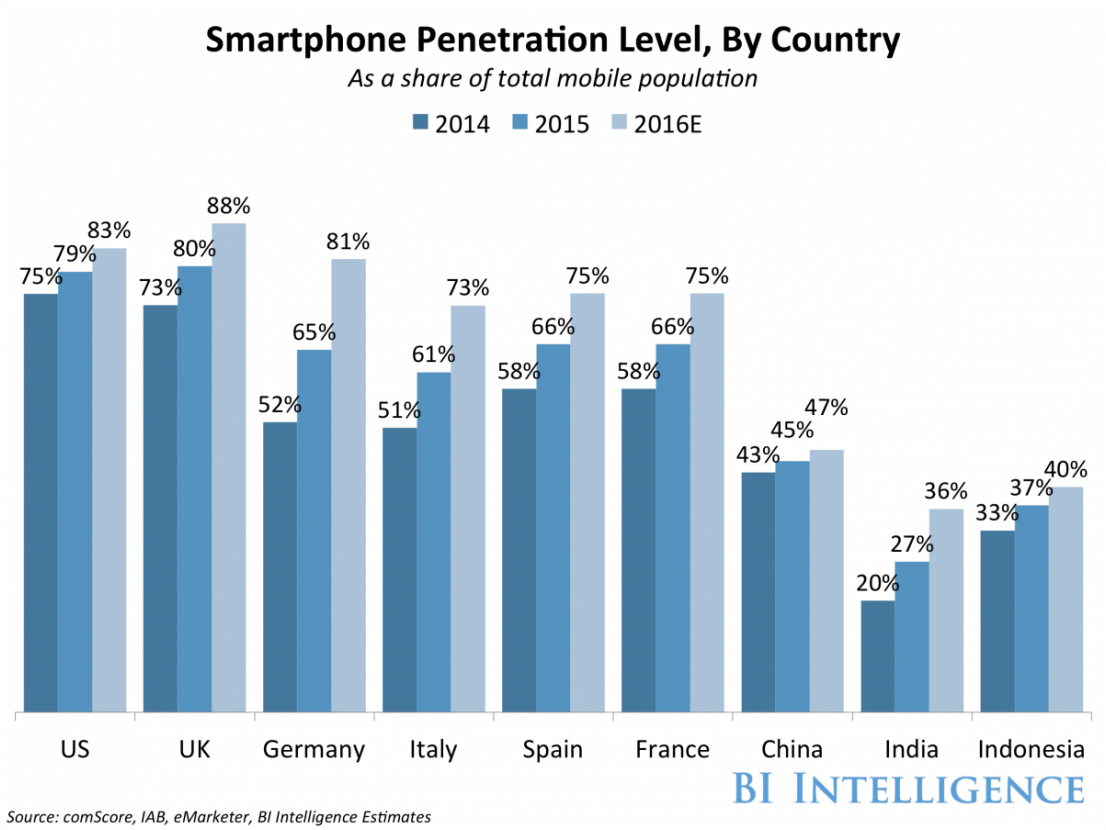

Figura 7: Smartphones en el Mundo.

Mobile and Social Sensing para problemas en tiempo real

Por lo general para realizar crowdsensing se trata de usar los celulares para recolectar información. Los celulares tienen una gran variedad de sensores, y cada año tienen más y más micro sensores que van evolucionando. El GPS, Bluetooth, acelerómetros, barómetros, se usan para aplicaciones especiales comerciales, y también se pueden reutilizar para muchas otras aplicaciones.[1]

Hoy los teléfonos son capaces sensar variables ambientales, lo cual facilita la implementación de proyectos de sensores.

En los países desarrollados y subdesarrollados, gran parte de la población lleva consigo teléfonos con distintos grados de avance tecnológico, por lo tanto se puede relevar información que de otro modo sería muy costoso y dificil ya que habría que entregar dispositivos a cada participante uno a uno con el costo y logística que eso implica.

En crowdsensing, la adquisición o recolección de datos, puede ser clasificada como participativa u oportunista. En sistemas de detección oportunista, la participación del usuario es mínima: las decisiones de detección son aplicadas o dirigidas por dispositivos.

1: Mobile and Social Sensing for Real Time Problems, Laurissa Tokarchuk, Queen Mary University, 2015

En los sistemas de detección participativa, los usuarios participan activamente en el proceso de detección.
Los usuarios, también llamados participantes, son reclutados por una plataforma central, que envía tareas de detección. A continuación, los usuarios pueden decidir qué solicitud aceptar y, una vez aceptados, deben realizar tareas específicas de detección y notificación de datos. Por un lado, la detección oportunista reduce la carga de la participación de los usuarios, ya que los dispositivos o aplicaciones son responsables de tomar decisiones de detección. Por el contrario, los sistemas de detección participativa se adaptan a las arquitecturas de crowdsensing con una "plataforma central", que facilita las operaciones de control del sistema como asignación de tareas, incentivos al usuario y compensación para compensar a los participantes por su contribución.[2]

También se pueden tomar otras consideraciones como duración de las campañas de sensado, para sacar conclusiones. Por ejemplo, podrían considerarse 10.000 usuarios que producen datos con una duración de sólo 30 minutos por día, y con este volumen de datos sacar insights, o también tomar datos en tiempo real y utilizar sistemas acordes para lograr objetivos similares pero on-line.
Muchas veces se confunde Mobile Sensors con Wireless Sensors, nosotros nos referimos únicamente a los sensores que se encuentran en los celulares de gama media.
Los teléfonos móviles usualmente llevan varios tipos de sensores y microsensores:

- Posición
- Movimiento
- Audio
- Video
- Entorno

Ejemplos de aplicación de estos sensores
Para ejemplificar los resultados que se pueden obtener con esta tecnología, se consideran dos ejemplos.

El primero sobre el tránsito en las ciudades:

- Se ha estimado que la congestión del tráfico cuesta a la economía mundial cientos de miles de millones de dólares cada año, aumenta la contaminación y tiene un impacto negativo en la calidad de vida general en las áreas metropolitanas. Una parte significativa de la congestión en las áreas urbanas se debe a los vehículos que buscan el estacionamiento en la ciudad. Los mapas detallados y precisos de estacionamiento en la calle pueden ayudar a los conductores a localizar fácilmente áreas con un gran número de plazas de aparcamiento legales y así aliviar la congestión. Se puede identificar la locación de espacios de estacionamiento legales a partir de datos recolectados con crowdsensing.

  Se ha demostrado que los datos de crowdsensing de los sensores de estacionamiento del vehículo se pueden utilizar para clasificar las áreas de la calle en los espacios de estacionamiento legales / ilegales. Basado en más de 2 millones de puntos de datos recolectados en Highland Park, New Jersey y en el centro de Brooklyn, New York, se mostró que los mapas de estacionamiento en la calle se pueden estimar con una precisión

de ~ 90% usando el algoritmo de umbral de ocupación ponderado.[2]

El segundo es sobre ahorro de energía de iluminación:

- Un ejemplo de aplicación se extrajo del paper "CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments", se refiere a la iluminación pública que es un servicio tradicional de las ciudades provisto por farolas ampliamente distribuidas en calles y caminos. La iluminación provoca casi el 19% del uso mundial de energía eléctrica y supone un 6% de las emisiones mundiales de gases de efecto invernadero. Una disminución del 40% de la energía gastada para fines de iluminación equivale a eliminar la mitad de las emisiones de la producción de electricidad y generación de calor de los Estados Unidos. Específicamente, la iluminación de la calle, que es un servicio comunitario esencial, repercute alrededor del 40% en el presupuesto energético de las ciudades. Por lo tanto, la optimización del servicio de iluminación es un objetivo primordial de los municipios.

  Las soluciones de alumbrado público actualmente implementadas en las ciudades no son eficientes energéticamente. Por lo general, cada lámpara funciona a plena intensidad 12 horas al día en promedio: 8 horas durante el verano y 14 horas durante el invierno. Como resultado, los costos que sufren los municipios son altos.

  Muchos tipos diferentes de lámparas se pueden usar para la iluminación pública de la calle, incluyendo alta presión del sodio (HPS), lámparas del Metal-haluro (MH), lámparas fluorescentes compactas (CFL) y diodo Lightemit (diodo emisor de luz).

  Los LEDs tienen una vida media 4 veces más larga que las lámparas HPS y 10 veces más si se comparan con las lámparas MH. La instalación de LEDs es eficaz para reducir los costos de hardware, instalación y mantenimiento.

  El bajo voltaje proporciona ahorros de energía significativos y permite aumentar la eficiencia de la lámpara. Las lámparas HPS no soportan el oscurecimiento y sólo pueden emplearse LEDs para realizar el oscurecimiento adecuadamente.

  El uso de LEDs está ganando gradualmente popularidad debido a sus características fotométricas, tales como un bajo consumo de energía ponderada (kW / 1000hrs), alta eficacia luminosa (lm / W), alta resistencia mecánica, larga vida útil y reducción de la contaminación lumínica.

  Las lámparas LED pueden atenuar la intensidad de la luz en más del 50% modificando por lo tanto el nivel de salida de la luz de acuerdo a las circunstancias. Por ejemplo, cuando el tráfico es bajo o en áreas raramente visitadas de la ciudad, como los parques por la noche. Se puede medir ese tráfico con crowdsensing, a partir de sensar la cantidad de personas o vehículos que circulan.

2: Crowdsensing Maps of On-street Parking Spaces, Vladimir Coric y Marco Gruteser, 2013

De acuerdo con el paper que describe el simulador CrowdSenSim, la solución de iluminación inteligente con tecnología LED y el oscurecimiento ligero ahorran en promedio casi el 68% del consumo de energía con respecto a la corriente adoptada.[3]

Sensores actualmente utilizados

Debajo detallamos algunos de los sensores que se pueden utilizar en las distintas recolecciones de datos:

- **Bluetooth**: Es el más poderoso, cuando los celulares aparecieron, estaba habilitado por default, y muy rápidamente la gente hizo fuertes reclamos porque consumía mucha batería, aunque es muy útil para realizar Mobile Sensing. No se necesita permiso del usuario para tener la información de Bluetooth de su teléfono, si una aplicación escanea se puede descubrir, no se necesita una aplicación para enviar datos, por default se obtiene sin necesidad de activar otra aplicación adicional. Esto es también una preocupación de seguridad ya que brinda muy buena información de posición y tiempo, además de interacción social, porque tiene un rango de 10 metros. Las personas que un usuario con teléfono móvil se cruza durante el día son fácilmente detectables por Bluetooth, y se guarda el orden en el cual se las encuentra. Los primeros estudios se focalizaron en mapear las actividades de la vida cotidiana, mapeando la gente con la que los usuarios se cruzaban, utilizando Bluetooth. También Wifi puede llegar a utilizarse con el mismo fin.

- **iBeacon**: Es muy interesante para Mobile Sensing y para coleccionar datos de comportamiento, se llama también Bluetooth Smart y es usado como un sensor muy exacto de posicionamiento puertas adentro, se pueden sensar elementos en varias formas, con mucha exactitud, o sino también clasificando los dispositivos que rodean a una persona como cerca, lejos o desconocido. Desde IPhone 5 en adelante ya viene instalado, también está disponible en Android. Es uno de los sensores más modernos.

- **WiFi**: Sirve para detección de ubicación puertas adentro, se usa para determinar densidad de multitudes, por lo general se combina GPS y triangulación para tener más exactitud en la determinación de una ubicación. La razón por la que se usan este tipo de tecnologías puertas adentro, además de Bluetooth e iBeacon, es porque el GPS no tiene suficiente alcance y no es confiable.

- **GPS**: Da la posición, latitud y longitud del dispositivo, la exactitud es en promedio 8 metros, no sirve para puertas adentro.

- **Brújula**: Mide la dirección y dice en qué sentido camina una persona, por lo tanto se usa en los mapas, combinándolo con el acelerómetro.

- **Magnetómetro**: Tiene uso limitado, porque hay que calibrar para sensar objetos metálicos y campos magnéticos.

3: CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments, Claudio Fiandrino; Andrea Capponi; Giuseppe Cacciatore; Dzmitry Kliazovich; Ulrich Sorger; Pascal Bouvry; Burak Kantarci; Fabrizio Granelli; Stefano Giordano, 2017

- **Acelerómetro**: Usa los términos pitch, roll, way, heave, sway, surge. Mide la fuerza de aceleración de un dispositivo, es el más popular para crowdsensing, mide en 3 ejes, se calculan las coordenadas a través de trigonometría. Determina la orientación del dispositivo. Por ejemplo cuando pasamos de visualización landscape a portrait.

- **Giróscopo**: Dice cuánto se giró el dispositivo en el tiempo, es muy intenso, no ruidoso pero poco exacto. En la figura se ve la forma de tomar la referencia de la posición sensada.

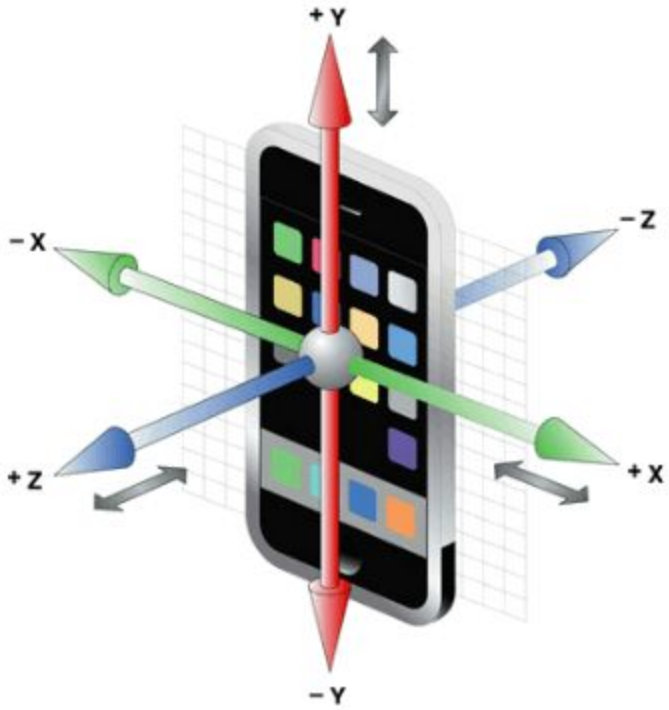

Figura 8: Giróscopo - Fuente: http://www.pandaancha.mx/

Como ejemplo de una aplicación que hace uso de las características de los smartphones más allá del uso habitual, Science Journal de Google permite recopilar datos de los sensores del teléfono, tablet o Chromebook compatible con aplicaciones de Android.

Utiliza sensores para medir su entorno, como la luz y el sonido, para graficar datos, registrar experimentos y organizar preguntas e ideas. En la figura 5 se ve un ejemplo del sensor de luz.

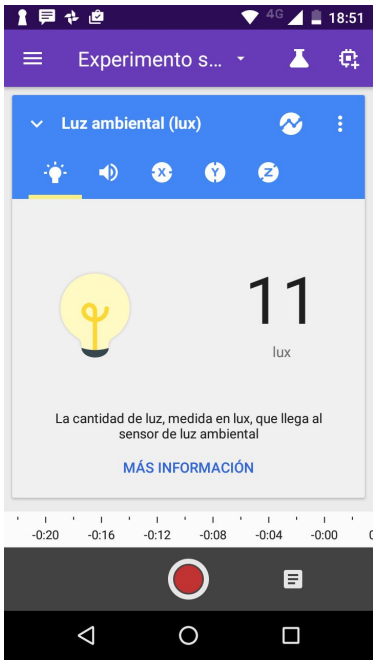

Figura 9: Science Journal - Fuente: https://makingscience.withgoogle.com/science-journal?lang=en.

Sensingkit

Se han desarrollado varios software para tomar información de los celulares. El que se ha elegido para analizar es Sensinkit, que es un proyecto de software libre de la Universidad de Queen Mary de Londres, por lo tanto no hay que desarrollar la colección de bibliotecas para sensores, porque ellos desarrollaron un kit especial para desarrolladores y estudiantes.

Uno de los desafíos de hacer Mobile Sensing es acceder a todas las plataformas. Entonces este kit cobra mucha relevancia ya que este desarrollo consta de bibliotecas para iPhone, Android así como también para Windows Mobile.

La siguiente figura muestra los módulos presentes en una implementación promedio.

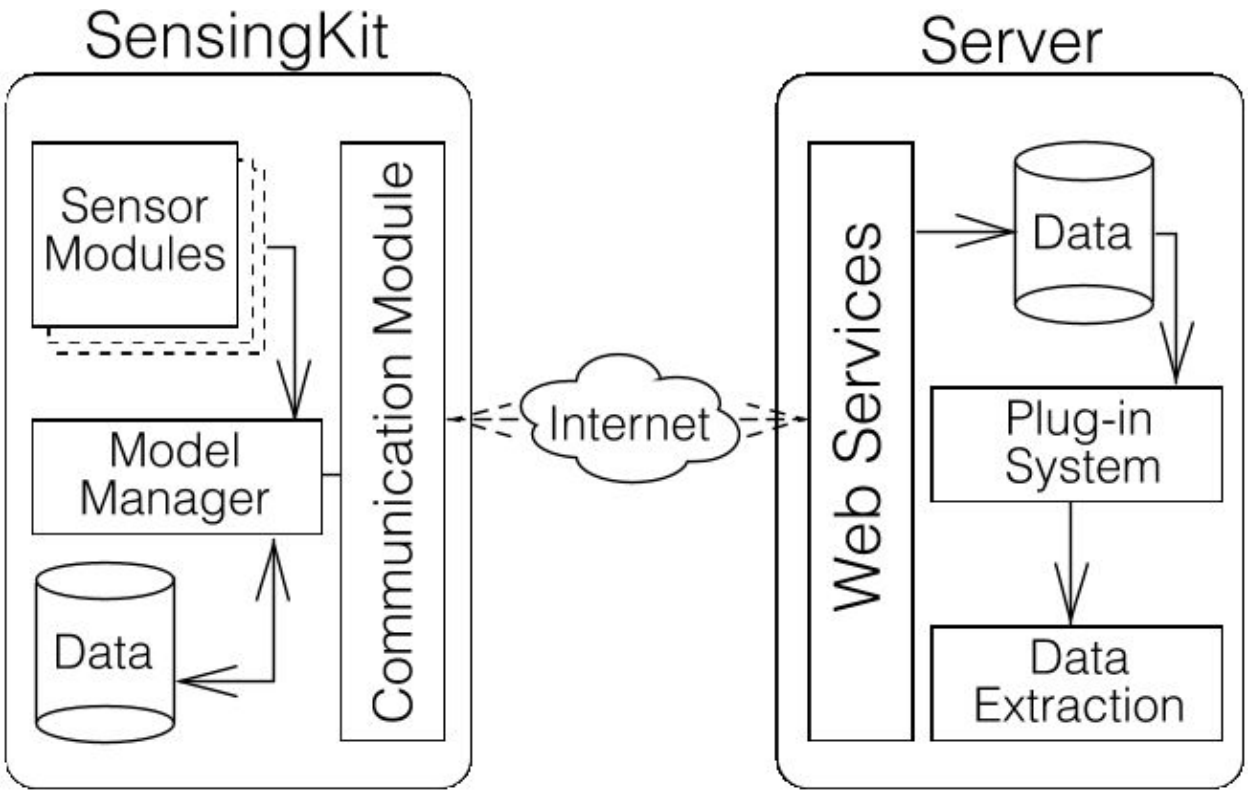

Figura 10: Sensing Kit - Fuente: www.sensingkit.org.

Para Android hay una versión, pero no tiene la misma cantidad de funciones, solamente recolecta información capturada y hay que utilizar otro programa para sacar los datos y enviarlos al repositorio general, en forma de CSV.

Para instalar la aplicación en iOS, hay que acceder a AppleStore y luego buscar "Sensingkit":

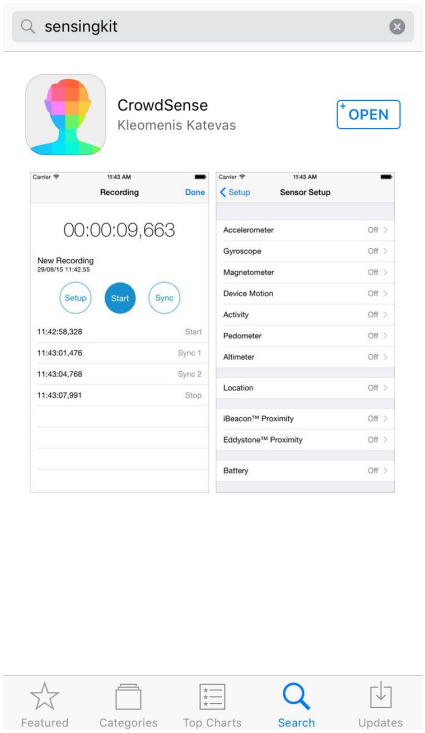

Figura 11: Instalación de Sensing Kit en iPhone.

Una vez que se baja la aplicación, se inicia y se configuran los sensores que se van a habilitar:

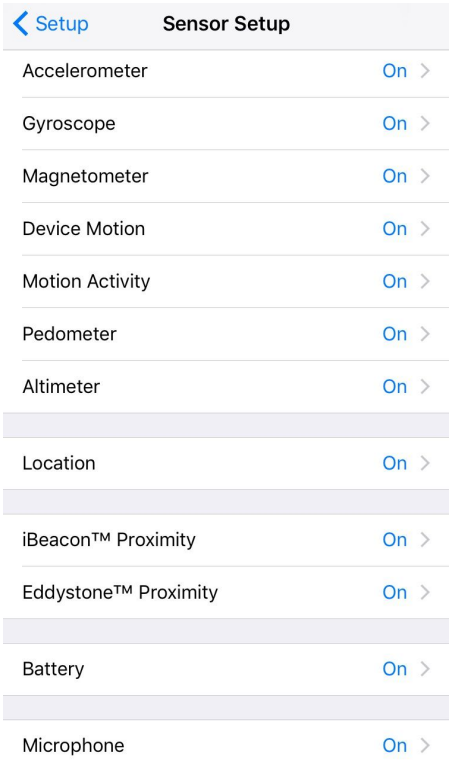

Figura 12: Habilitación de Sensores.

Luego se configura para comenzar a recolectar información, presionando la tecla "Start":

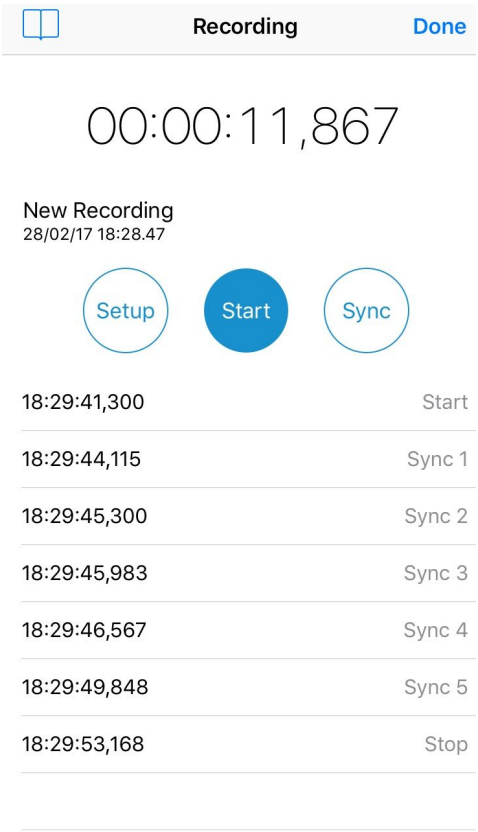

Figura 13: Comienzo de sensado.

Al terminar de recolectar, se presiona "Stop" para luego guardar bajo un nombre identificable, la información resultante:

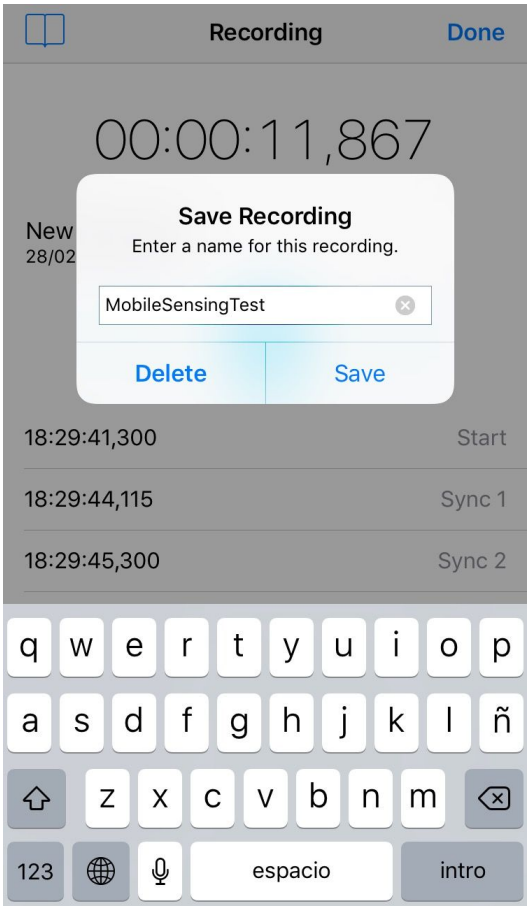

Figura 14: Guardar los datos sensados.

Una vez que se guardó, podemos pasarlo a otro medio más accesible para procesar luego, por ejemplo en:

- Dropbox
- Mail
- Google Drive
- Nota en el teléfono
- iCloud
- Webex
- Entre otras

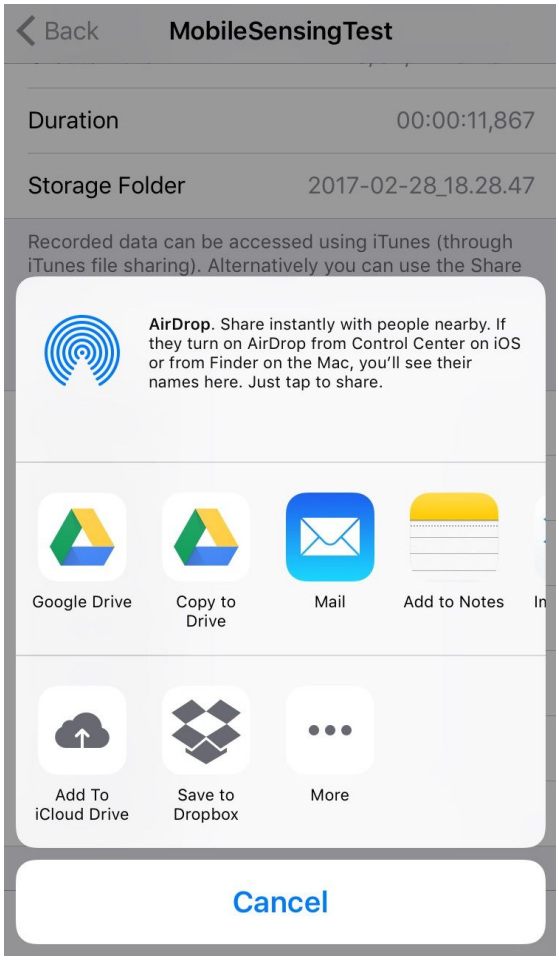

Figura 15: Envío de información.

Los parámetros sensados están descriptos en la siguiente tabla:

Variable	Parámetros sensados
Acelerómetro	timestamp, timeIntervalSince1970, x, y, z
Altímetro	timestamp, timeIntervalSince1970, relativeAltitude, pressure
Batería	timestamp, timeIntervalSince1970, state, level
Movimiento	timestamp, timeIntervalSince1970, attitude.roll, attitude.pitch, attitude.yaw, gravity.x, gravity.y, gravity.z, magneticField.x, magneticField.y, magneticField.z, magneticField.accuracy, rotationRate.x, rotationRate.y, rotationRate.z, userAcceleration.x, userAcceleration.y, userAcceleration.z
Eddystone Proximity	timestamp, timeIntervalSince1970, namespaceID, instanceID, rssi, txPower
Giroscopio	timestamp, timeIntervalSince1970, x, y, z
iBeacon	timestamp, timeIntervalSince1970, major, minor, accuracy, proximity, rssi

Proximity	
Location	timestamp, timeIntervalSince1970, latitude, longitude, altitude, horizontalAccuracy, verticalAccuracy, speed, course
Magnetómetro	timestamp, timeIntervalSince1970, x, y, z
Micrófono	timestamp, timeIntervalSince1970, state
Movimiento Actividad	timestamp, timeIntervalSince1970, createDate, createDateSince1970, stationary, walking, running, automotive, cycling, unknown, confidence
Pedómetro	startTimestamp, startTimeIntervalSince1970, endTimestamp, endTimeIntervalSince1970, numberOfSteps, distance, currentPace, currentCadence, floorsAscended, floorsDescended
Log de Grabación	timestamp, timeIntervalSince1970, label

Tabla 2: Parámetros sensados para Mobile Sensing con Sensingkit.

Las distintas plataformas necesitan diferentes bibliotecas, Sensingkit es un conjunto que se puede instalar en cualquier sistema operativo de dispositivo. Es modular y extensible, por eso invitan a re-utilizarlo. Está bajo licencia de Open Source.

Aunque no es el alcance de este trabajo, vale decir que se pueden crear aplicaciones que envíen automáticamente los datos en forma de tabla a los repositorios para luego ser procesados.

Aplicaciones de Mobile Sensing

Las aplicaciones de Mobile Sensing son infinitas, generalmente se describen según el tipo de información que se obtiene de un dispositivo. Personal, de grupo o comunidad.

Recolección de datos personales: Pueden ser Wearables, fitness devices, o dispositivos más completos como Apple Watch, pero también el teléfono puede tener la misma función. Se recolecta información propia que uno mismo configura, como por ejemplo las aplicaciones que registran ejercicios del estilo estilo “Map my run”.

Otra de las aplicaciones es calcular los pasos, estimar las calorías, casi siempre por otro lado incluye distintos niveles de recolección de datos. Las aplicaciones también procesan lo que uno agregue en forma manual, como seguimiento de las calorías consumidas y las entradas personales de los usuarios.

La aplicación para monitoreo del sueño se puede poner debajo de la almohada, usando el acelerómetro mide cuánto se descansa, tiene también sensores de audio, entonces se monitorea la calidad de sueño de la persona.

Además se puede calcular la huella de carbono del usuario. Las aplicaciones personales pueden incluir también registros de los ciudadanos con respecto a la seguridad, higiene, trámites, tránsito, entre muchos otros.

Recolección de datos de grupos: Se pueden coleccionar datos de grupos de personas interesados en el mismo tema, por ejemplo gente que saca fotos de basura motivando el reciclado, la gente que suma los kilos perdidos entre todos a lo largo de un mes, que realiza colas para un trámite o que asiste a un evento.

Recolección de datos de comunidad: Se adquiere información en forma colectiva para mapear fenómenos. Se puede sensar el patrón de tránsito en una ciudad. En vez de producir información a través de un único sensor, los sensores son colectivos y se sacan conclusiones de una comunidad de los mismos.

Muchas compañías usan estas aplicaciones para promediar las respuestas de muchos usuarios, ya que en definitiva se llega a un resultado más cercano a la realidad, que sería el promedio de todos los resultados colectados.

Cómo hacerlo usando teléfonos

Hay muchos modelos de cómputo de información de multitudes que pueden requerir distinto grado de involucramiento del usuario. Se puede monitorear por ejemplo polución, en ese caso se necesita que el celular esté conectado a otro dispositivo. Un grupo de usuarios puede monitorear la polución de un lugar con instrucciones adecuadas. De la misma manera, también se puede obtener información de los usuarios en el tránsito.

Para realizar crowdsensing móvil, como paradigma de detección emergente, se depende significativamente de las redes de comunicación inalámbrica para proporcionar la transmisión de datos garantizada por QoS (Quality of Service o Calidad de Servicio) entre usuarios de teléfonos inteligentes sobre fenómenos de interés común. El número cada vez mayor y las variedades de teléfonos inteligentes imponen pesadas cargas a las infraestructuras de comunicación del crowdsensing móvil.

El rendimiento de la comunicación de crowdsensing puede deteriorarse en algunas áreas de alta densidad debido a las abrumadoras solicitudes de comunicación, mientras que el ancho de banda inalámbrico en otras áreas puede no ser plenamente utilizado con las solicitudes de comunicación esporádicas e infrecuentes.

Por lo tanto, es de gran necesidad considerar la desequilibrada utilización de los recursos inalámbricos en el diseño de un paradigma de comunicación en el crowdsensing móvil para cumplir con los estrictos requisitos de QoS. Es necesario entonces un paradigma de comunicación con congestión mínima para tener una densidad móvil dinámica y sostenible para lograr un equilibrio de carga eficiente y una comunicación fiable en crowdsensing móvil.[4]

Crowdsensing puede desarrollarse en forma participativa o aprovechando información recolectada en forma involuntaria, en donde se registra información sin que el usuario se de cuenta. El uso de la batería es un tema importante, hay que diseñar aplicaciones para sensar lo menos posible y así gastar menos energía. Entonces la recolección de datos puede ser explícita o implícita, según sepa o no el usuario.

Dependiendo del estado del acelerómetro, se buscan picos de movimiento, aunque es difícil darse cuenta si alguien está en colectivo o en bicicleta. La aplicación involuntaria más usual es Google Maps, que está corriendo en background recolectando información de los movimientos, a menos que específicamente lo anulemos. Ya somos participantes de la recolección de datos de Google cuando lo utilizamos en nuestros teléfonos.

Las posibles aplicaciones de esta técnica se clasificarían según la variable a analizar:

- Entorno: de polución, niveles de agua, hábitos de vida animal
- Infraestructura: congestión de tránsito, disponibilidad de estacionamiento
- Social: ejercicio, rutas de bicicleta, hábitos de comida

Pasos para coleccionar datos de multitudes

Podemos describir en forma genérica, los pasos para obtener la información de un grupo grande de gente, siempre y cuando todos o la gran mayoría cuenten con el smartphone correspondiente, y una aplicación activa:

1. Captura de los datos, almacenamiento
2. Análisis
3. Clasificación usando programas o machine learning

Cuando se recolecta la información de los sensores, se guarda en algún dispositivo de almacenamiento masivo.

La cantidad puede ser exorbitante, y también se incrementa cuando hay más usuarios, por eso es necesario contar con almacenamiento acorde y buenos sistemas de backup para conservar los datos en forma confiable y segura. Típicamente se usa algún tipo de Machine Learning, clusters que clasifican con algoritmos que obtienen alguna información.

Ejemplo de Aplicación: Google Maps

Todos estamos familiarizados con Google Maps, y somos participantes del crowdsensing. Si instalamos Google Maps, y no apagamos explícitamente la recolección de datos, está en el background del teléfono smartphone recolectando información de los movimientos que realizamos.

La aplicación utiliza la información para actualizar las líneas rojas, amarillas y verdes en los mapas, para mostrar congestión de tránsito.

Cuando introdujeron por primera vez el sistema no era muy confiable, pero a medida que la gente lo fue usando más, evolucionó el aprendizaje de patrones y se llega a distinguir alguien que para por un café de cuando hay una congestión de tránsito y la persona está varada en su auto. Ahora es posible diferenciar la situación de un auto en una autopista o en una avenida donde hay cafeterías. Además usa mucha más información que recoge constantemente.

4: Congestion-Aware Communication Paradigm for Sustainable Dense Mobile Crowdsensing, Wen Sun, Jiajia Liu, 2017

Otro ejemplo puede ser cuando se arma un plan de viajes utilizando los datos de GPSs de taxis, Four Square, etc. El objetivo puede ser por ejemplo, ayudar a turistas a recorrer los lugares más atractivos de una ciudad en el menor tiempo posible.

Seguridad con Crowd Monitoring y Crowdsensing
Lo que interesa registrar en estas disciplinas es la topología de una multitud, necesitamos saber en términos de evacuación y seguridad en qué consiste la multitud, qué tan densa es, cómo se agrupa la gente en el área. También se busca saber cuánta gente hay.

Otra cosa es el tipo de multitud, de acuerdo a diferentes factores que pueden tener o no riesgo de conducir a comportamiento problemático. También se pueden detectar grupos dentro de grupos. Por ejemplo en una protesta o un recital, se podría ver qué grupos se comportan normalmente y cuáles no.

Si cada persona lleva consigo un celular, se puede medir la densidad y así evitar aglutinamientos de gente que produzca accidentes graves. Cuando se trata de realizar Crowd Monitoring de esta manera, lo más importante es el tiempo real. No sirve detectar que hubo problemas una hora después del evento.

Una gran herramienta para realizar este tipo de estudios es Bluetooth, que sirve para estudiar interacciones entre personas. Se puede realizar un monitoreo de una multitud en un festival, en donde la gente usa sensores como scanners. De la misma manera podemos sensar la movilidad y las interacciones sociales de esa multitud.

Crowd Topology
Las herramientas más poderosas utilizando celulares son Bluetooth, trazas de GPS y WIFI fingerprinting.

Este tipo de estudio puede ser utilizado por la policía para conseguir información valiosa sobre multitudes, como su velocidad, presión y riesgos.

También se puede cuantificar la cantidad de gente en una multitud, por las acciones en Twitter y actividad en Internet, llamadas y SMS.

Utilizando herramientas para geolocalizar los tweets individuales, se puede saber la ubicación de la persona. Muy poca gente localiza los tweets entonces se precisan otros significantes para saber de dónde vienen. Además, otra información útil que se puede estudiar, es que sabiendo la cantidad de tweets y SMSs en un área dada, se puede estimar la cantidad de gente en un lugar.

Por ejemplo, en un estadio cuando hay un partido hay picos de accesos, y la cantidad suele ser proporcional a la cantidad total. Por lo tanto hay un grado alto de similitud con la audiencia. Este mapeo puede dar una estimación bastante precisa del tamaño de la multitud.

Sensado de Eventos de Multitudes en las Redes Sociales
Las aplicaciones móviles se pueden utilizar para estudiar Crowd Sentiments, o sea para saber

sobre el estado de ánimo de una multitud en un evento. El objetivo de esta práctica es analizar los posteos y frases de los espectadores en las redes sociales. Se puede ver si hay una relación entre los resultados del análisis de sentimientos (Sentiment Analysis) analizando las palabras en cada tweet buscando una relación entre esa información y la situación de la multitud.

También se pueden encontrar sub eventos, por ejemplo en ocasión de un festival de música porque pueden haber picos cuando hay bandas específicas. Otra cosa interesante que se puede detectar es si hay escenarios de emergencias.

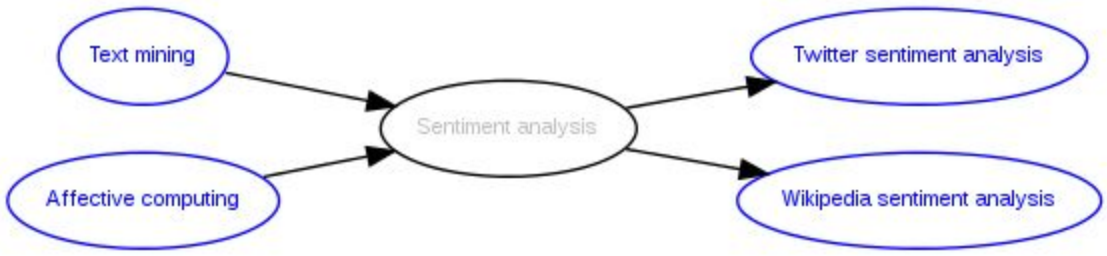

Figura 16: Sentiment Analysis - Fuente: Wikipedia.

Por lo general se intenta buscar información de todo el evento con ciertas palabras clave.

No solamente usando las palabras obvias porque se pierde mucha información que aparece con otras palabras del contexto. Se puede verificar que recolectando las palabras clave estándar se recogen muchas menos impresiones. Por ejemplo para obtener información de Crowd Sentiments en los juegos Olímpicos se puede usar:
#football
#olympic

Y también hay impresiones para el mismo contexto con los siguientes hashtags:
#FIFA
#gol

Entonces se usan técnicas adaptativas para coleccionar un conjunto extendido de palabras clave identificando términos adicionales para filtrar de forma automática.

Se puede detectar de esta forma si hay riesgos de desbordes sociales utilizando varias técnicas para adaptar las palabras clave.

Un posible procedimiento sería:

1. Se colocan palabras iniciales y luego se colecta información sobre otras palabras y sus ocurrencias, y las que tienen más, se convierten en candidatas.
2. Se considera la similitud entre las palabras.
3. Se genera un diccionario con el vocabulario libre de Twitter y se ven relaciones que no existen en otro ámbito.
4. Una lista de candidatas, se compara con los tweets generados con esas palabras y se ve si están de acuerdo con el evento a estudiar.

5. Se comparan 100 tweets de la palabra original y los 100 de la palabra candidata y se hace un análisis de similitud para evitar el ruido de otras palabras con otros temas.
6. Entonces luego se crea una lista de textos emergentes para seguir.

Este procedimiento consigue mucha más información que cuando se usan palabras estándar. Con técnicas similares, se pueden detectar en tiempo real de sub eventos en un evento general.

Detectar grupos de peatones usando sensores de multitud
La idea de este procedimiento es determinar varios tipos de peatones, por ejemplo ciertos grupos que se mueven juntos como una unidad, o sino lado a lado si el espacio no está lleno.
Se pueden identificar individuos en el mismo grupo cuando todos se mueven a la misma velocidad y siguen las mismas trayectorias. Si se tienen que abrir porque viene otra persona, se vuelven a unir luego. También se puede considerar un grupo si su paso está sincronizado.

Los primeros trabajos para detectar grupos de peatones se realizaron con pocos individuos, utilizando WIFI, brújula y acelerómetro. Se definió que en un grupo de peatones es un cluster en movimiento con una duración, que consiste en una cantidad de gente que se mueve a la misma velocidad durante un período de tiempo.
Se pueden determinar clusters con sensores de 3 tipos:

- WIFI
- Acelerómetro
- Brújula

Entonces con esta información se crean clusters para cada medición en particular y se ve la correlación.
Habría entonces 3 tipos diferentes de cluster, y se mezclan para crear el “conjunto candidato”.

La idea es detectar interacción sin video, utilizando solamente sensores móviles: proximidad, acelerómetro, etc. Porque en muchos lugares no se puede usar video, no se puede tener cámaras en todos lados. Además requeriría mucha infraestructura instalada, y el propósito es solamente detectar la composición del grupo con número, tamaño y trayectoria usando solo sensores de teléfonos móviles.

Una de las aproximaciones es estudiar la sincronicidad del paso de los peatones. Este parámetro se puede detectar. Se realiza una correlación entre lo detectado con cada participante. Y hay ventanas discernibles donde los participantes pueden estar sincronizados en los gráficos de posición. Cuanto más gente hay, es más difícil encontrar la sincronización. El acelerómetro da mucha información sobre la situación relativa de los peatones. Ayuda a detectar si 2 personas están interactuando. Pero no sirve estudiar aisladamente, hay que correlacionar con otros sensores, por ejemplo iBeacon, Bluetooth Smart, Giroscopio, magnetómetro.

También se pueden usar sensores de presencia para que los peatones no tengan sensores puedan aportar información. Se pueden realizar también grafos de proximidad para estudiar cómo se mueve la gente y cómo interactúan, para identificar clusters de gente. Los mismos son muy dependientes del contexto. Por ejemplo una estación de tren no es lo mismo que una galería de

arte.

La imagen muestra un ejemplo de grafo de proximidad.

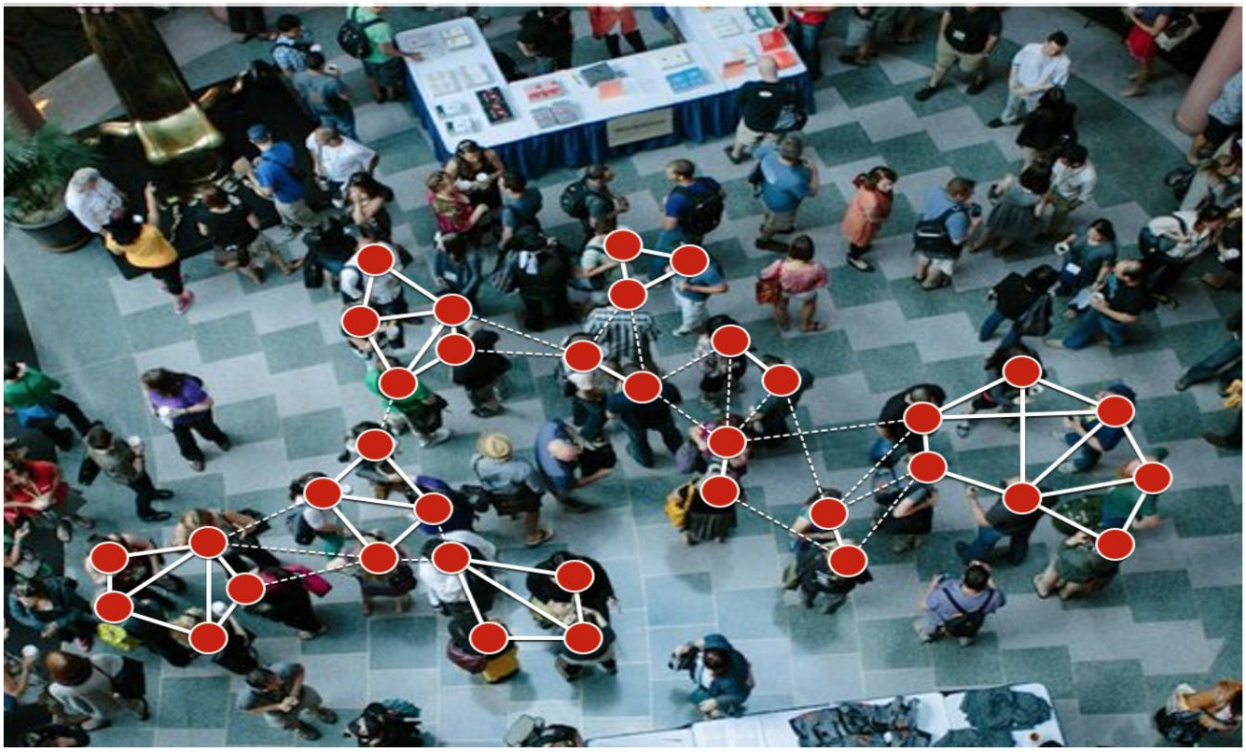

Figura 17: Gráficos de detección de clusters - Fuente: Mobile and Social Sensing, Dr. Larissa Turchak.

Los desafíos

- Privacidad: No todos quieren dar su información proveniente de sus sensores , hay que ver cómo hacer anónima la recolección de datos. Aunque esto no es suficiente, es posible que contando con información de lugar y hora se pueda deducir quién es la persona, por lo tanto es muy difícil que en un conjunto de datos, con el tiempo y procesamiento suficientes, no se pueda inferir de quien o quienes se trata la información.
- El consumo de batería.
- Temas de infraestructura: procesamiento centralizado o distribuido.
- Fusión de datos de diversos sensores.
- Aceptación de los usuarios para instalar aplicaciones.

Las fuentes de información para recolectar datos son cada vez más abundantes. Lo que se busca es explorar las mejores herramientas para procesar esta información en forma eficiente. Es por esto que en la próxima sección se abordará el tema del estado del arte del manejo de grandes cantidades de datos, y su posible implementación para obtener insights útiles y visualización de los mismos.
No se tratarán los temas de seguridad, automatización, dimensionamiento, aunque son fundamentales para el correcto desarrollo de la recolección de datos.

Comparación de Herramientas para el Manejo de grandes cantidades de datos

En el mundo del cómputo de grandes cantidades de datos, existe una gran variedad de herramientas. Podemos dividirlas en dos grupos desde el punto de vista de la infraestructura, dado el sentido que le queremos dar a este trabajo.

Por un lado están las herramientas que llamaremos “legacy”, porque son las que tradicionalmente se vienen usando para procesar grandes cantidades de información. Estas herramientas deben ser instaladas por los administradores de sistemas en servidores dentro de un datacenter propio. En contraposición a eso, las herramientas en la “nube”, son las administradas por un proveedor, y tienen escalabilidad ampliada y grandes capacidades que resultaría muy caras de construir con la primera opción. Por lo general se cobran por el uso, por consulta realizada, por almacenamiento y transferencia de datos.

Las herramientas de “nube” se dividen a su vez en 2 tipos, las “unmanaged” que son aplicaciones directamente instaladas en servidores en la nube, y las “managed” que se consumen como servicios, y se pagan por uso.

Para el usuario está abstraída la capa de operación de infraestructura y aplicación, solamente debe configurar los parámetros e interfaces de sus sistemas de procesamiento, el resto queda del lado del proveedor. De esta forma pueden haber servicios de almacenamiento, mensajería, bases de datos, big data, machine learning, entre muchos otros.

Big Data Ecosystem

El procesamiento de las grandes cantidades de datos recolectadas, es una tarea colosal, es por esto que han proliferado una gran cantidad de herramientas y aplicaciones específicas para cada tipo de dato o implementación. Las más importantes están listadas en el siguiente gráfico.

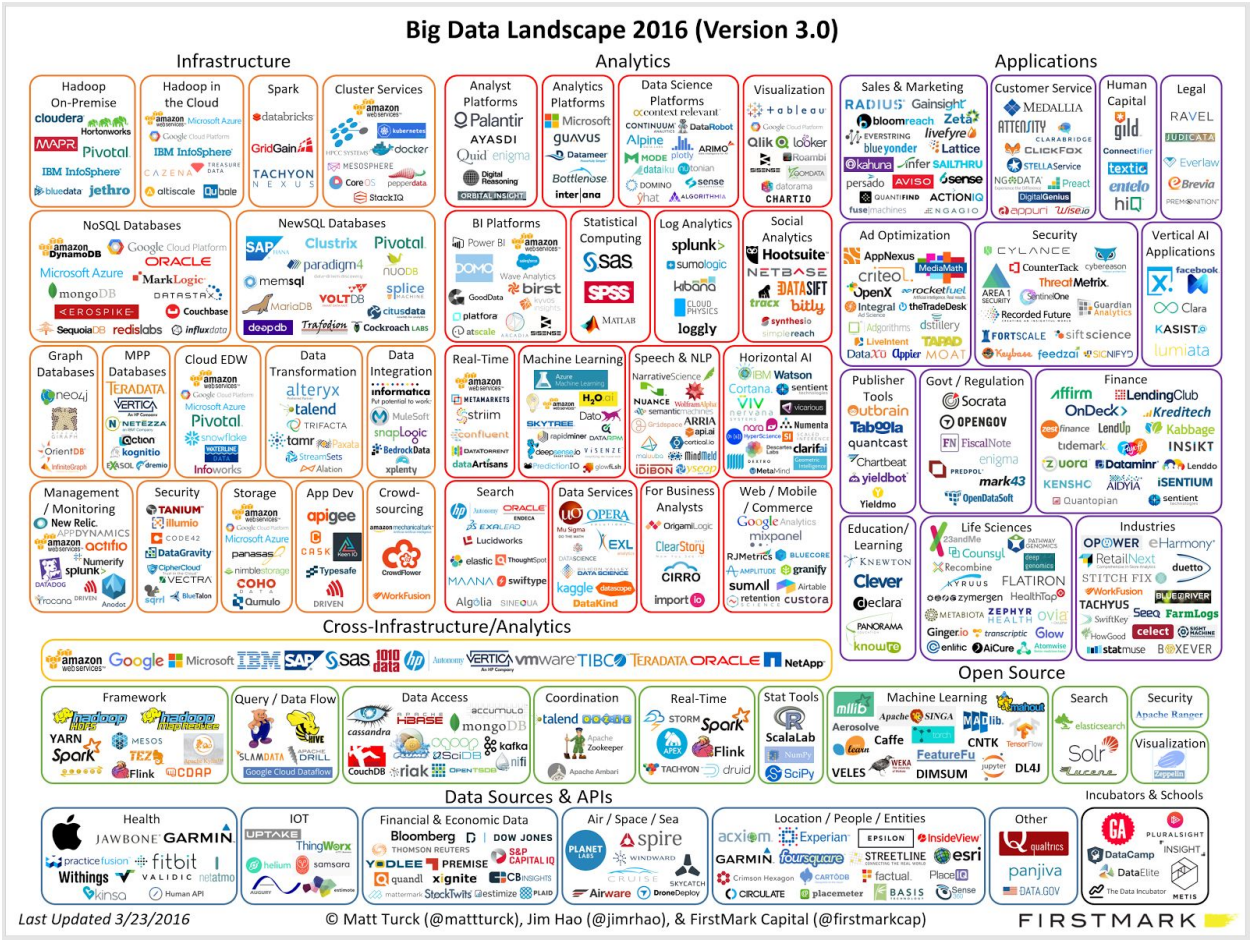

Figura 18: Big Data Landscape - Fuente: http://mattturck.com/2016/02/01/big-data-landscape/.

Herramientas "Legacy"

A continuación se describen varias de las aplicaciones más importantes del ecosistema de procesamiento de grandes cantidades de datos.

Apache Hadoop

El proyecto Apache Hadoop se ocupa de desarrollar software open source para cómputo confiable, escalable y distribuido.

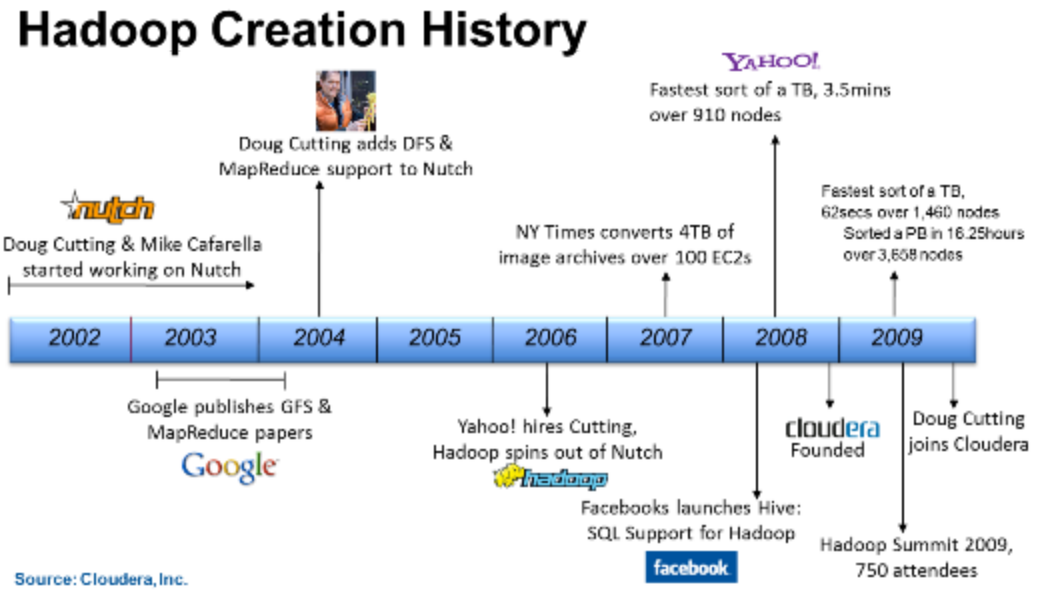

Figura 19: Proyecto Hadoop, cronología - Fuente: http://hadoopgeek.com/category/hadoop/page/3/.

La génesis de Hadoop vino a partir del paper del sistema de archivos de Google que fue publicado en octubre de 2003. Este paper generó otro paper de investigación de Google - "MapReduce: El procesamiento de datos simplificado en clusters grandes".

El desarrollo comenzó en el proyecto de Apache Nutch, pero fue movido al nuevo subproyecto de Hadoop en enero de 2006. Doug Cutting, quien estaba trabajando en Yahoo! en ese momento, lo nombró Hadoop por el elefante de juguete de su hijo. El código inicial que se tomó en cuenta para Nutch consistía en 5k líneas de código para HDFS y 6k líneas de código para MapReduce.

El primer committer agregado al proyecto de Hadoop fue Owen O'Malley en marzo de 2006. Hadoop 0.1.0 fue lanzado en abril de 2006 y continúa evolucionando por los muchos contribuyentes al proyecto de Apache Hadoop.

El framework de Hadoop permite procesamiento distribuido de conjuntos grandes de datos a través de clusters de computadoras usando modelos simples de programación. Está diseñado para escalar desde servidores aislados a miles de máquinas, cada una ofreciendo cómputo local y storage.

En vez de confiar en el hardware para entregar alta disponibilidad, la biblioteca está diseñada para detectar y manejar las fallas en la capa de aplicación, para entregar un servicio de alta disponibilidad sobre un cluster de computadoras, cada una de las cuales puede tener sus propias fallas y tener resiliencia con otra.

El proyecto consta de los siguientes módulos:

- Hadoop Common: Utilidades comunes que soportan los otros módulos.
- Hadoop Distributed File System (HDFS™): Un filesystem distribuído que provee alta

tasa de entrada/salida para acceder a los datos.
- Hadoop YARN: Un framework para hacer un schedule de tareas y administración de recursos.
- Hadoop MapReduce: Un sistema basado en YARN para procesamiento paralelo de conjuntos grandes de datos.

NIFI
Trabaja con Apache Kafka, Apache Storm, Apache HBase y Spark para procesamiento en tiempo real de mensajes distribuidos en flujos de datos.

NiFi es una excelente plataforma para ingestar flujos de datos en tiempo real y persistir grandes cantidades de los mismos, como por ejemplo, Internet of Things, sensores y sistemas transaccionales. Si los datos vienen como basura, NiFi ofrece herramientas para filtrar los datos no deseados. Adicionalmente, NiFi puede también actuar como mensajero y enviar datos a los lugares que se necesiten.

El sistema de diseño de NiFi puede pensarse como un Cajero Automático, en donde los datos entrantes se procesan en forma segura y se escriben en forma secuencial a disco. Hay 4 componentes principales involucrados en el movimiento de datos hacia adentro o hacia afuera de NiFi:
- FlowFile
- Processor
- Connections
- Flow Controller

En NiFi, un FlowFile es un conjunto de datos traídos al flujo desde cualquier fuente de datos, y se mueve a través del mismo. Connections son las relaciones entre componentes que permiten a los Flowfiles moverse a través del flujo de datos. Un Flow Controller regula el intercambio de Flowfiles entre 2 Processors.

Processors son acciones que se realizan sobre los Flowfiles para procesar su contenido y atributos para asegurarse de que los datos deseados se mueven a través de los procesos del flujo de datos para eventualmente ser guardados en un lugar seguro. Entonces NiFi actúa como un productor para publicar mensajes de uno o más temas o tópicos (en inglés "topic"). Entonces, desde un alto nivel, los productores envían mensajes en la red hacia el cluster Kafka.

KAFKA
En una arquitectura moderna de datos construida sobre YARN-enabled Apache Hadoop. Kafka trabaja en combinación con Apache Storm, Apache HBase y Apache Spark para realizar flujos en tiempo real de mensajes distribuidos. Kafka es una plataforma excelente para enviar mensajes con baja latencia para fuentes en tiempo real de flujo de datos, como Internet de las Cosas, sensores, y sistemas transaccionales. Cualquiera sea el uso de la industria, Kafka maneja y analiza mensajes masivos con baja latencia en Enterprise Apache Hadoop.

STORM

Apache Storm es un sistema distribuído en tiempo real para cómputo de grandes volúmenes de alta velocidad de datos en paralelo y en escala. Storm es al procesamiento en tiempo real de datos, como Apache Hadoop y MapReduce es al procesamiento en Batch de datos. Con su interfaz simple para programar, Storm permite a los desarrolladores de aplicaciones escribir software para analizar flujos de tuplas de datos, y también objetos.

Storm se integra con las tecnologías más comunes en la industria. Usa "topologías" para consumir flujos de datos y procesar esos flujos en formas arbitrarias y complejas, reparticionando los flujos entre cada etapa de cómputo según sea necesario.

Herramientas en la nube

Con el advenimiento de los servicios en la nube, se aceleraron los tiempos de implementación de proyectos, ya que esta tecnología permite virtualizar, clonar y automatizar servidores de manera simple y pautada, además de documentada. Los ciclos de desarrollo e implementación contínua se hicieron más eficientes y efectivos, y la puesta en marcha de productos es mucho más rápida.

También se mejoró cuantitativamente la eficacia de los tests y la solución de bugs de software. La seguridad, además, es más granular y también más amplia para posibilitar la autenticación de muchos más dispositivos en forma centralizada. La implementación de estos servicios permite focalizarse en los datos y las aplicaciones, no tanto en la operación, además el bajo costo de propiedad hace que no haya que pagar la capacidad ociosa.

Otra ventaja que realmente empuja los desarrollos de nuevos negocios en la nube, es la escalabilidad según la demanda, ya que los servicios son elásticos, pueden crecer o achicarse para satisfacer el consumo de recursos, protegiendo las inversiones.

Los proveedores de servicios de nube han desarrollado una oferta de plataforma específica para manejo de recolección de datos en forma masiva. Microsoft Azure IoT Suite, Amazon AWS IoT y Google Cloud Platform serían las marcas líderes en este momento, como se puede ver en la figura 20.

Las tres compañías han recorrido un largo camino en el año 2015 y 2016 para llevar la funcionalidad básica de la plataforma de IoT a sus plataformas en la nube. Aunque muchas de las capacidades fundamentales ya existen, todavía falta que maduren para dar un servicio más integral.

Por ejemplo, específicamente en las áreas de administración de dispositivos, administración de seguridad, clase de asignación de servicios, administración de políticas y monitoreo, no hay buenas interfaces para el desarrollo pesado que se necesita hacer para que todo se integre. Por ahora, la base de comparación es realmente la integridad de los componentes de la solución para permitir la creación de un producto o servicio basado en IoT.

En el mercado existen otros proveedores de servicios de cloud, como ser Rackspace, Softlayer de IBM, VMWare, Cisco, Digital Ocean, OVH, GoDaddy y muchos otros proveedores que dan

servicios similares pero con menos prestaciones. En la figura 20 se puede ver la ubicación en el cuadrante de Gartner de cada uno, en el año 2015, hoy por hoy la distribución es parecida, aunque Google está avanzando rápidamente por su gran apuesta estratégica en este negocio.

Figura 20: Proveedores de Servicios - Fuente: www.gartner.com Mayo 2015.

Resultado de comparación

El siguiente cuadro comparativo, muestra las principales características de los vendors, y sus equivalencias.

Capacidad	Microsoft Azure	Amazon AWS	Google Cloud Platform
Ingeniería de hardware - Sistema operativo	Ofrece una versión específica de Windows para dispositivos IoT. Windows 10 IoT Core es una versión de Windows 10 que está optimizada para dispositivos más pequeños con o sin pantalla y que se ejecuta en el Raspberry Pi 2 y 3, Arrow DragonBoard 410c y MinnowBoard MAX	No proporciona un sistema operativo personalizado para dispositivos	No proporciona un sistema operativo personalizado para dispositivos
SDK de dispositivos	Proporciona acceso de código abierto a SDKs de dispositivos de Microsoft Azure IoT para conectarse y ser administrado por el servicio Cloud de Hub de IoT de Microsft Azure. Los dispositivos SDK de IoT pueden utilizarse en una amplia variedad de dispositivos y soportar múltiples idiomas como C #, Java y JavaScript	El SDK del dispositivo AWS IoT permite a los dispositivos conectarse, autenticarse e intercambiar mensajes con el servicio cloud de AWS IoT. Open Source y disponible para personalización. Embedded C, JavaScript y Arduino Yun disponible en el sitio de AWS	Google ha trabajado con Seeed Studio y BeagleBoard.org para ofrecerte un kit de prototipos para desarrolladores basado en la nueva placa BeagleBone Green Wireless de BeedleBone Green.
Conectividad del dispositivo: autenticación y autorización	La autorización y autenticación a nivel de sistema se basan en identidades por dispositivo. Se usan credenciales y permisos de acceso casi instantáneamente revocables. Las operaciones de conectividad del dispositivo se supervisan y registran.	AWS IoT Registry asigna una identidad única a cada dispositivo. AWS IoT proporciona autenticación mutua y cifrado en todos los puntos de conexión, de modo que los datos nunca se intercambian entre dispositivos y AWS IoT sin identidad comprobada. AWS IoT admite el método AWS de autenticación (denominado 'SigV4'), así como la autenticación basada en certificados X.509. Las conexiones que utilizan HTTP pueden utilizar cualquiera de estos métodos, mientras que las conexiones que utilizan MQTT utilizan autenticación basada en certificados y las conexiones que utilizan WebSockets pueden utilizar SigV4.	OAuth 2.0 es un estándar de la industria para la autenticación y autorización con servicios de red. Google Cloud Platform utiliza OAuth 2.0 para autenticación y autorización de API. Los escenarios más utilizados son el flujo centrado en el usuario y el flujo centrado en el servidor.
Conectividad del dispositivo: protocolos compatibles	MQTT v3.1.1, HTTP 1.1 o AMQP 1.0	MQTT, WebSockets, HTTP 1.1	MQTT, WebSockets, HTTP 1.1
Conectividad del dispositivo: puerta de enlace del dispositivo	Protocol Gateway para convertir los protocolos AND Field Gateway para permitir que los dispositivos locales de baja potencia se comuniquen con Azure IoT	No	No
Interfaz de administración de dispositivos	Hay API de administración de dispositivos masivos, pero no hay un portal para el monitoreo de la integridad del dispositivo o la administración de dispositivos	Múltiples API disponibles para realizar funciones de administración de dispositivos. No hay una aplicación fácil de usar lo que hace difícil administrar un gran número de dispositivos. AWS forzaría al usuario a usar múltiples conjuntos de servicios de Amazon como AWS IAM, AWS IOT Device Registry para administrar dispositivos manualmente	No
Kits de inicio compatibles	Raspberry Pi, Intel Edison	Intel Edison, Instrumentos de Texas Launchpad, Arduino Yun compatibility	Arduino, Development kits
Ingesta de datos	Como parte de Azure IoT Hub, aunque el apoyo adicional en forma de Azure Event Hubs	Amazon IoT y Amazon Kinesis	Pub/Sub
Procesamiento de eventos de flujo y reglas	La suite Aure IOT se compone de Azure Stream Analytics Service que permite que permite utilizar la sintaxis SQL para analizar los datos ingeridos por IoT Hub. Stream Analytics permite aprovechar los datos históricos y utilizarlos como parte del análisis de secuencias en vivo desde IoT Hub	Como parte de AWS IoT, Amazon proporciona un servicio de Motor de reglas que soporta SQL como sintaxis pero no admite el acceso a datos históricos como parte del análisis de transmisión en tiempo real	No
Aprendizaje automático	Azure ML con capacidad para generar API incorporable	ML Amazon	CLOUD MACHINE LEARNING ENGINE
Almacenamiento de datos	Azure soporta el almacenamiento de datos en: 1) Servicio de base de datos relacional - SQL Server 2) Servicio de almacenamiento de datos MPP - SQL Server 3) Azure Data Lake Store - de alto rendimiento 4) Azure Blob almacenamiento 5) Azure HDInsight (Hadoop distribución) 6) Spark en HDInsight 7) Base de datos de documentos NoSQL	AWS soporta el almacenamiento de datos en: 1) S3 2) MPP Data Warehouse - Amazon RedShift 3) Servicio de base de datos relacional - Amazon RDS 4) NoSQL - Amazon DynamoDB	1) Cloud Storage 2) Cloud SQL 3) Cloud Big Table 4) Cloud Spanner 5) Cloud Datastore 6) Persistent disk
Soporte Hadoop	Sí con HDInsight	Sí con Amazon EMR	Sí con Dataproc
Visualización de datos	Servicio PowerBI	Todavía en la vista previa - Amazon QuickSight	Cloud Data Lab
Notificaciones y alertas	Azure Centros de Notificación	Amazon SNS	GCloud Pub/Sub
Almacenamiento en caché	Redis Cache Service	Amazon Elasticache	No

Tabla 3: Comparación de proveedores de cloud.

Después de la comparación, Microsoft Azure parece ser la plataforma completa por sobre Amazon AWS y Google Cloud principalmente debido a los siguientes factores:

- Azure tiene una plataforma de análisis de flujo más madura que soporta el análisis basado en SQL de datos de flujo, aunque Amazon y Google Cloud no están muy lejos.
- Herramientas más maduras en Azure para machine learning.
- Un servicio maduro de la visualización de datos en PowerBI (QuickSight de Amazon está en la inspección previo actual y Google Cloud no está maduro en este punto).
- Un robusto marco ETL en Azure Data Factory.
- La existencia de Azure Data Lake Store, que es un repositorio de escala híbrida compatible con HDFS nuevo y puede ser utilizado de forma nativa por una amplia variedad de servicios de datos en la nube sin tener que mover datos alrededor (como habría que hacer si decidiera almacenar datos En S3 y usar Redshift).
- Hay herramientas para IoT que no están tan maduras en Amazon Web Services y Google Cloud.
- Sin embargo en términos de búsquedas en grandes cantidades de datos, Redshift de Amazon parece superar a ambos proveedores, según un informe de Panoply.io.

Se agrega también una lista de aplicaciones de Amazon Web Services, sabiendo que las características una a una pueden ser diferentes:

Amazon Web Services	**Google Cloud Platform**
Compute	
Amazon EC2	Google Compute Engine
Amazon EC2 Container Service	Google Container Engine
AWS Elastic Beanstalk	Google App Engine
AWS Lambda	Google Cloud Functions
Storage	
Amazon Glacier and Amazon S3 Standard - Infrequent Access	Google Cloud Storage Nearline
Amazon S3	Google Cloud Storage Standard
Amazon EC2 Container Registry	Google Container Registry
Database	
Amazon DynamoDB	Google Cloud Datastore or Google Cloud Bigtable
Amazon RDS	Google Cloud SQL
Big data	
Amazon EMR and AWS Data Pipeline	Google Cloud Dataflow and Google Cloud Dataproc

Amazon Kinesis and Amazon Simple Queue Service (SQS)	Google Cloud Pub/Sub
Amazon Redshift	Google BigQuery
Monitoring	
Amazon CloudWatch	Google Cloud Monitoring and Google Cloud Logging
Networking	
Amazon Elastic Load Balancing	Google Cloud Load Balancing (HTTP/HTTPS Load Balancing and Network Load Balancing)
AWS Direct Connect (3)	Google Cloud Interconnect
Deployment	
Amazon Route 53	Google Cloud DNS and Google Domains
AWS CloudFormation	Google Cloud Deployment Manager

Tabla 4: Amazon Versus Google Cloud - Fuente Documentación de Google Cloud.

Solución utilizada en el prototipo: Google Cloud

Si bien los servicios de los 3 principales proveedores son similares y tienen distintas prestaciones según la aplicación, elegimos Google para el prototipo por la versatilidad de una herramienta como BigQuery, y la interfaz intuitiva para importar los datos y luego procesarlos.

Este vendor ofrece soluciones de integración para cómputo y Big Data, adecuadas para una gran escalabilidad y procesamiento de datos en tiempo real, de características similares a los demás.

Modelo de prototipos de IoT utilizados para crowdsensing

En los siguientes diagramas se puede ver la secuencia de ingreso de datos, a través de diferentes sensores, para luego ser procesados en tiempo real.

Una vez que los sensores distribuidos en una locación reciben la información, la misma es procesada por dispositivos estándar para su transmisión hacia la nube.

Para ello se utilizan protocolos de comunicación usuales como HTTP.

Luego una aplicación de cómputo recibe estos datos y los encola para ser procesados por lotes, o sea, por grupos (en inglés "batch").

A partir de ahí se envía a una aplicación como BigQuery que mantiene grandes cantidades de datos y permite realizar consultas masivas en muy poco tiempo.

Los resultados de dichas consulta, se pueden obtener en tablas o archivos más pequeños, para luego ser nuevamente procesados por aplicaciones de cómputo en App Engine, o %R, que es una herramienta que permite realizar correlación de datos en tiempo real y entregar resultados útiles.

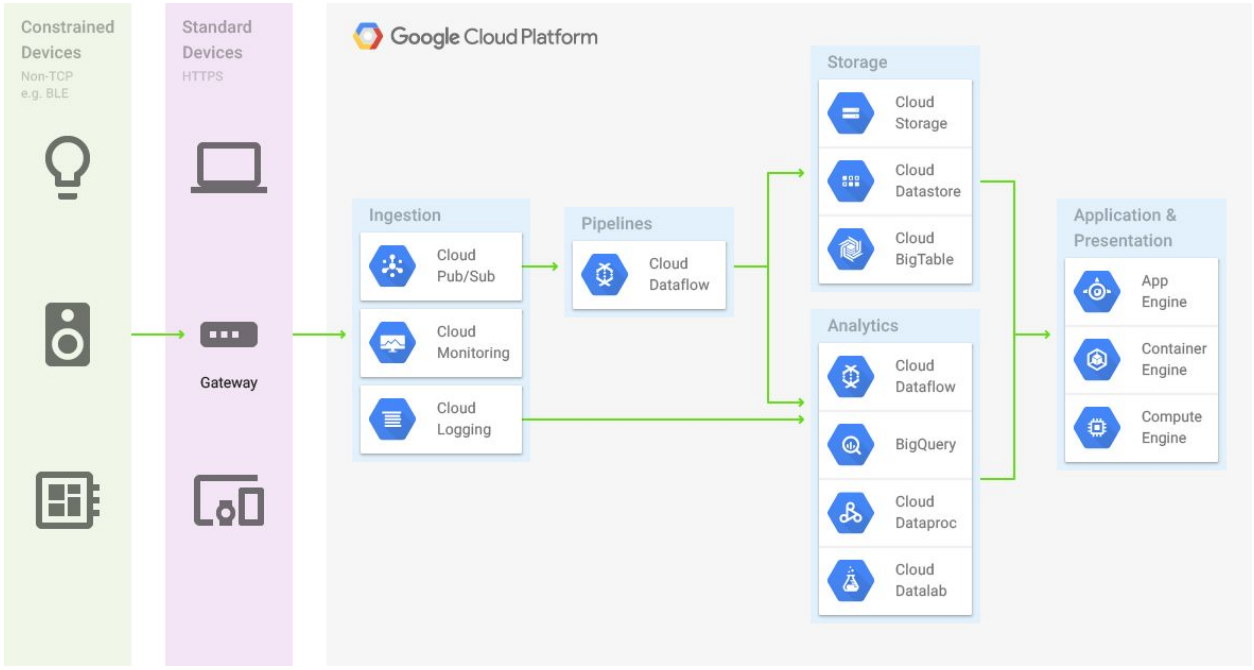

Figura 21: Modelo de implementación de IoT sobre Cloud en el proveedor seleccionado.

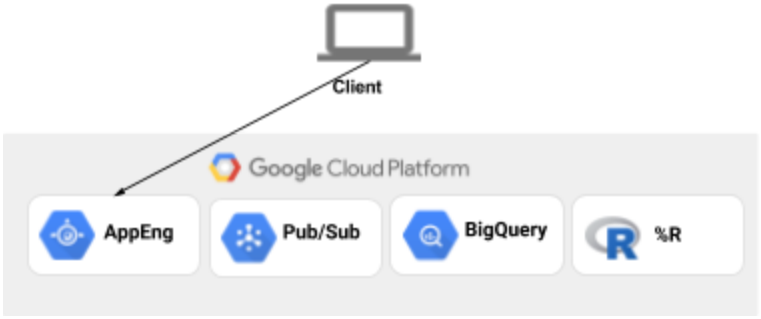

Figura 22: Modelo de prototipo para análisis de datos.

Capítulo 3

Modelo de aplicación de recolección de datos de crowdsensing

El ejemplo que presentamos, es un desarrollo de la infraestructura de backend de una aplicación para reportar eventos como baches en el asfalto, delitos, acumulaciones de basura, emergencias médicas, gente en situación de calle, infracciones de locales al público, congestión de trámites en oficinas públicas, entre muchos otros.

El ingreso de datos se realiza desde el smartphone del usuario, y se enviaría a través de Aplicaciones de Frontend a los repositorios de datos en forma de archivos CSV.

Las aplicaciones de Backend, constan de diferentes etapas, que se visualizan en la figura:

1. Ingreso de datos desde CSV (archivo de texto que contiene una tabla cuyas columnas están separadas por comas), hacia las colas de procesamiento de mensajes en Pub/Sub.
2. Inserción de los mensajes con el formato correspondiente desde Pub/Sub a BigQuery, que es un repositorio de datos para su posterior análisis.
3. El cambio de formato de los campos provenientes de los archivos CSV, es realizado en este caso con scripts de Python, aunque en la plataforma de Google se pueden procesar con otra herramienta llamada Dataflow, que se puede parametrizar para lograr la inserción de los datos en la base. El siguiente gráfico ilustra el proceso.

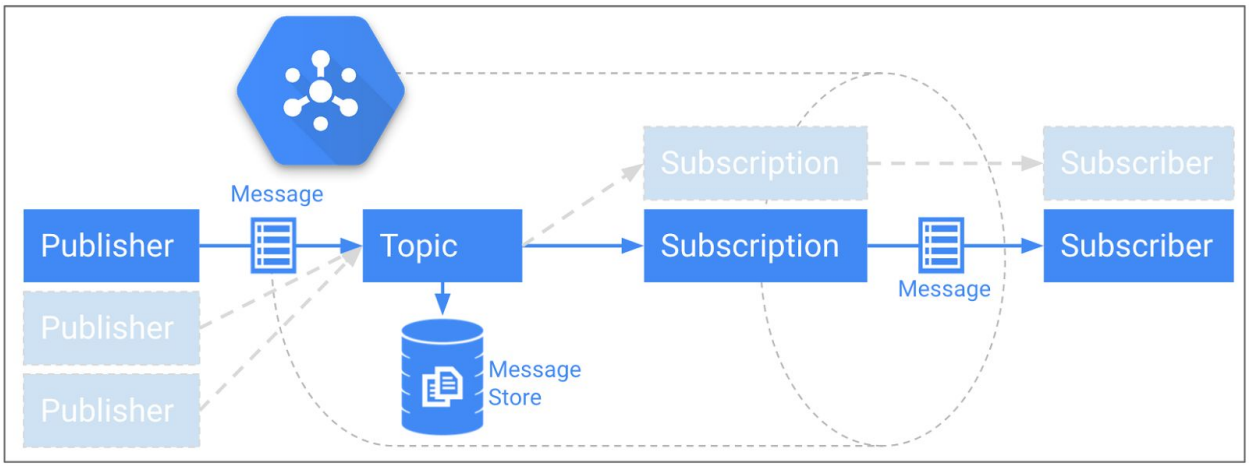

Figura 23: Diagrama de Prototipo - Fuente: Documentación de Google Cloud.

4. Compute Engine es un servidor en la nube encargado de realizar las consultas a BigQuery y procesar las respuestas para su visualización en gráficos o mapas. El gráfico ilustra una posible implementación.

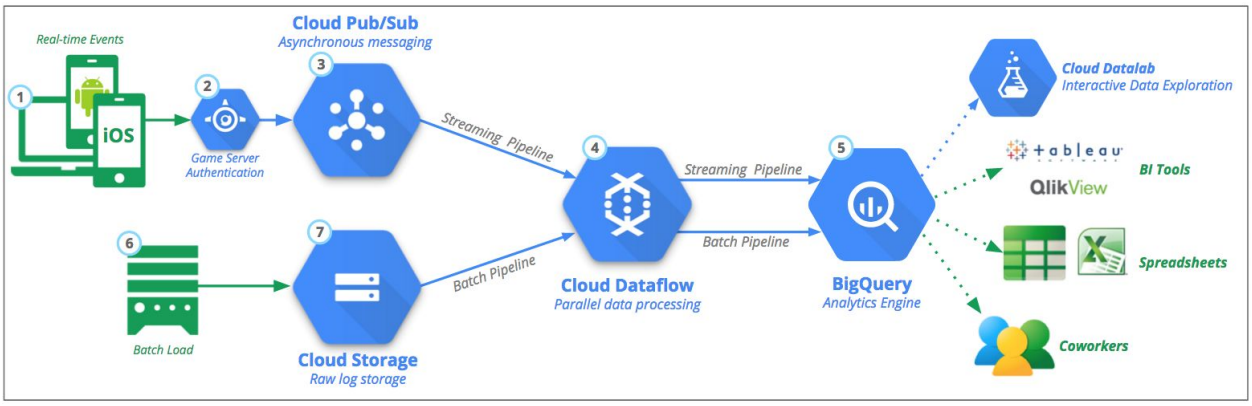

Figura 24: Esquema de aplicación, visualización de datos - Fuente: Documentación de Google Cloud.

Acopio de información pública para ser analizada

Muchas ciudades del Mundo están publicando información sobre el clima, el consumo de servicios, datos poblacionales y económicos, para que puedan ser analizados y estudiados por entes o empresas que colaboren con el desarrollo de la comunidad.

Dicha información es una fuente de datos que pueden ser correlacionados con eventos particulares y reportes para llegar a inferir información útil para la ciudad.

Otra fuente importante de datos, como dijimos en párrafos anteriores, es la recopilación de datos de Twitter y Facebook, y es una de las maneras de acceder a información de la actividad de grupos grandes de personas. Por ejemplo BirdIQ permite bajar información de redes sociales, como Twitter, y de esta manera correlacionar datos, que deben ser formateados antes de ser interpretados.

Como se describió al principio del texto, la ciudad de Buenos Aires, el Gobierno publica información pública real para ser consultada por los ciudadanos en la siguiente URL: http://data.buenosaires.gob.ar/

Muchas ciudades también publican sus datos, y BigQuery cuenta con un repositorio para hacer tests y probar las herramientas: https://cloud.google.com/bigquery/public-data/

Desarrollo

Construcción de un sistema en tiempo real con BigQuery y Python

Los conceptos fundamentales del procesamiento de flujos de datos a grandes escalas, precisan de componentes de mucha potencia.
Afortunadamente, los servicios de nube alcanzan los requerimientos, de tal forma que se pueden implementar soluciones complejas y poderosas sin necesidad de recurrir a la instalación de un Data Center. La infraestructura detrás del DataWarehouse es invisible, el mantenimiento también.

Las herramientas elegidas en este caso son Cloud Pub/Sub y BigQuery, para implementar streaming en tiempo real de datos.

Estas herramientas, se basan en implementaciones anteriores de sistemas open source de streaming de datos como Spark, Apache Storm y Apache Flink.
Se puede tratar el procesamiento de datos como 2 conceptos diferentes, procesamiento de datos y análisis de datos.
El procesamiento de datos comprende el movimiento, filtrado y complemento por elemento, típicamente independiente del tiempo. El análisis de datos comprende entonces el cómputo, el agrupamiento, reducción, agregación, y es por lo general dependiente del tiempo.
En el contexto de análisis de datos en streams, es común unir streams de datos para analizarlos y procesarlos.

Entonces los acercamientos al estudio de los 2 tipos de proceso son diferentes.
Por lo general el diseño de sistemas de procesamiento y análisis, es realizado dependiendo del propósito y de la velocidad de respuesta necesaria. Desde el punto de vista de los negocios, por ejemplo, es cada vez más necesario tener respuestas rápidas para acomodar las estrategias de ventas a la realidad de los consumidores.
El desafío es que cuando la actividad crece, hay más preguntas y se necesita aún más procesamiento y análisis de datos.

Por lo general, se utilizaba procesamiento por lotes (batch), para realizar determinadas tareas. Este tipo de procesamiento, en ambientes administrados estáticos, conlleva riesgos de control de costos, variación de tiempos de respuesta, e inexactitud en las respuestas.

Se puede comparar la performance de las implementaciones, considerando el tiempo y los recursos utilizados. Estos sistemas escalan agregando procesadores y trabajo en paralelo. Por lo general, de esta forma, en caso de precisar más procesamiento, se controla el tiempo para cada tarea, y así se aprovechan los recursos. Si no hay trabajos para realizar, los recursos quedan ociosos. En estos casos cuando se agregan recursos, también hay que controlar el ciclo de vida de los mismos, lo cual conlleva más trabajo operativo, que nada tiene que ver con el análisis de los datos y la velocidad de procesamiento.

BigQuery, como ejemplo de aplicación para procesamiento de TB de datos en segundos
Google BigQuery, es una parte esencial de la solución de análisis y datos de Google Cloud Platform. La arquitectura sin servidor de BigQuery es un ejemplo muy claro de lo que significa una aplicación totalmente administrada. Escala a la perfección, analiza terabytes de datos en segundos y soporta consultas de datos simultáneas para grandes organizaciones, sin necesidad de aprovisionamiento de capacidad o ajuste de sistemas. Como siempre, BigQuery está altamente disponible y encripta todos los datos en reposo.

BigQuery es la externalización de la implementación de una de las principales tecnologías de la compañía cuyo nombre de código es Dremel. BigQuery proporciona el núcleo de funciones disponibles en Dremel. Lo hace a través de una REST API, una interfaz de línea de comandos, una interfaz de usuario web, control de acceso y mantiene el desempeño sin precedentes en las consultas de Dremel.

Específicamente podemos agregar algunas características de las nuevas versiones del software a

la descripción:

- La compatibilidad de BigQuery para Standard SQL, que implementa el estándar SQL 2011, ahora está disponible.
- Los nuevos controladores ODBC permiten utilizar BigQuery con una serie de herramientas que van desde Microsoft Excel hasta los sistemas tradicionales de inteligencia empresarial como Microstrategy y Qlik.
- Más control y visibilidad.
- La capacidad de actualizar, eliminar e insertar filas y columnas en conjuntos de datos de BigQuery mediante SQL Estándar.
- Integración con Cloud Identity y Access Management para administrar políticas de seguridad de para usuarios y recursos de BigQuery.
- Supervisión a través de Google StackDriver para realizar un seguimiento del rendimiento y el uso de la carga de trabajo.
- Intercambio de consultas a través de enlaces, para fomentar el intercambio de conocimientos y la colaboración dentro de las organizaciones.
- Permite procesar 1 petabyte de información en 245 segundos
- Comandos como “JOIN EACH” y “GROUP EACH BY” que permiten unir tablas inmensas y seleccionar resultados en segundos.
- Posee Funciones de Pearson, que sirven para correlacionar eventos, de tal forma de disparar alertas.

El Modelo

Esta descripción del modelo muestra cómo usar herramientas de nube, en este caso Google Cloud Platform, para crear una aplicación que reciba datos telemétricos sobre geolocalización, los procese y a continuación, almacene los datos procesados y transformados para su posterior análisis.

Se propone entonces tomar por un lado información real de una ciudad, en este caso la Ciudad de Buenos Aires, y correlacionar con datos en tiempo real, para luego obtener insights y visualizarlos en forma de gráficos o en Google Maps utilizando la API de Heatmap.

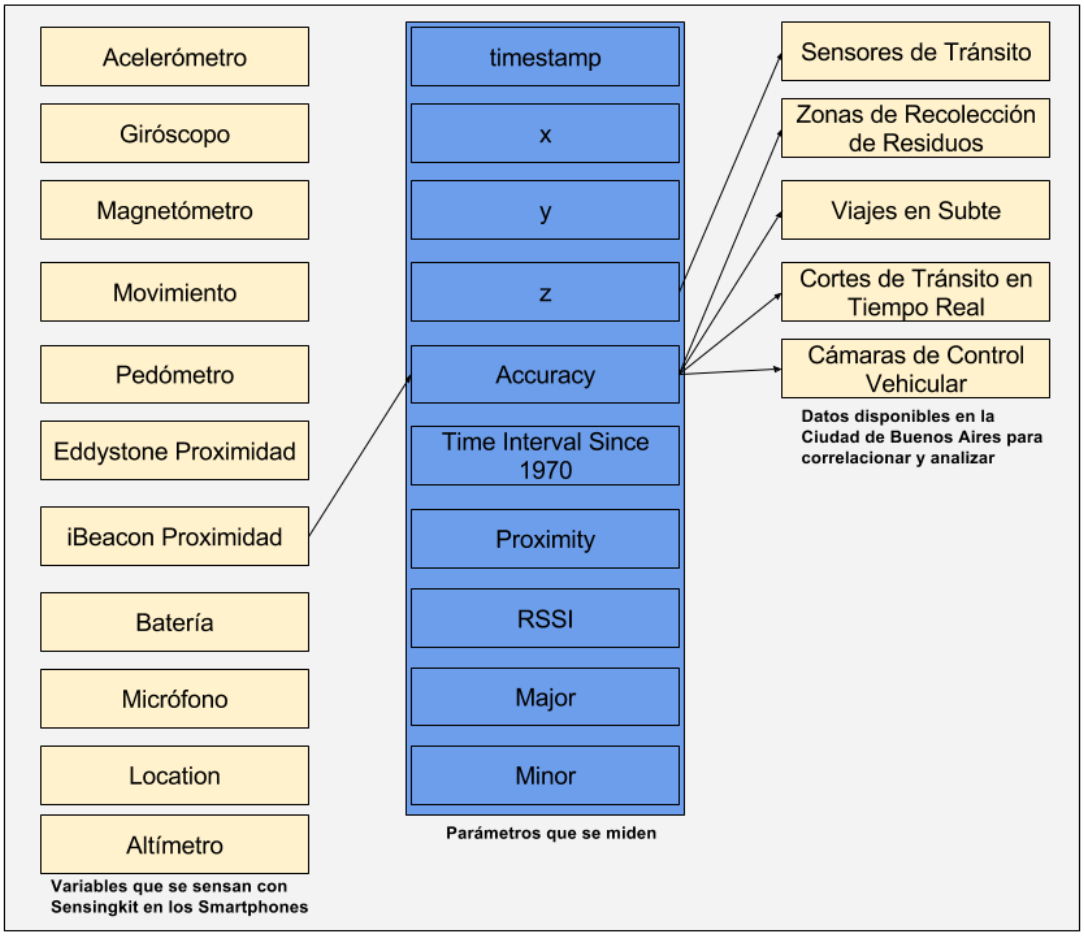

Figura 25: Modelo para correlacionar información de los smartphones y los datos públicos de la ciudad.

En la siguiente sección se describe una guía de los elementos necesarios para la creación de un sistema de telemetría de geolocalización escalable en la nube con API de Google Maps. Los datos de entrada son tomados de un repositorio en Google Cloud en donde periódicamente llegan tablas de datos provenientes de smartphones. Los datos se toman de los sensores de los celulares del público en general, o también puede ser a través de una aplicación especial para ese fin. En este prototipo se considera que los datos llegan directamente al repositorio.

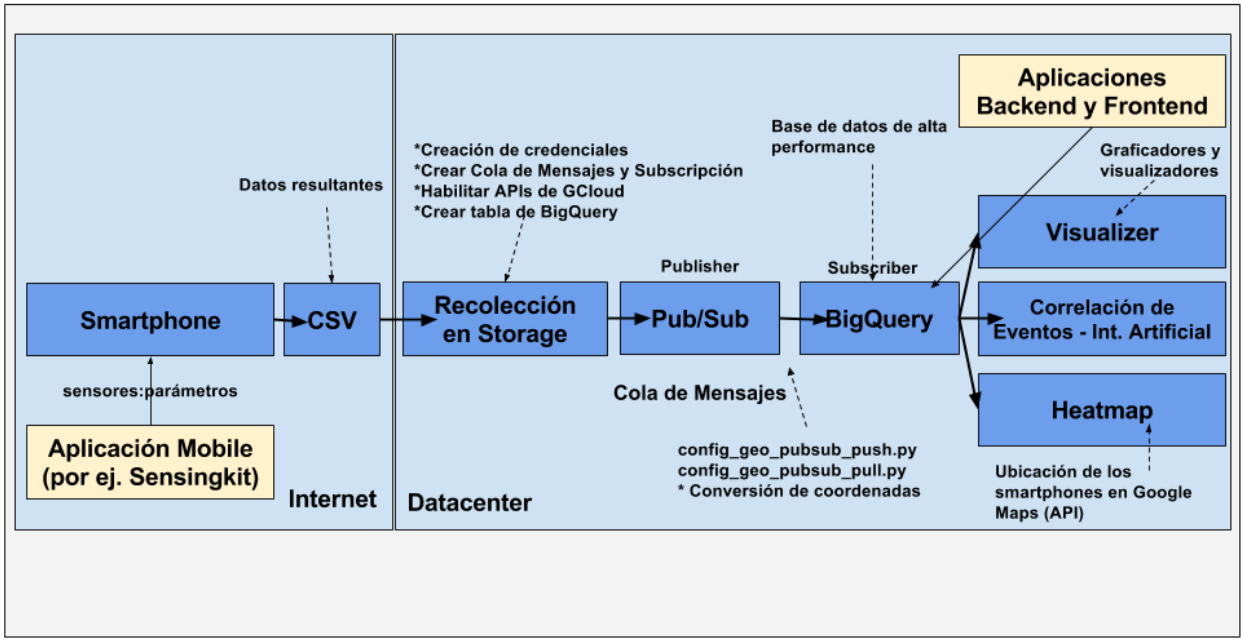

Figura 26: Diagrama del modelo.

La idea es recibir mensajes e insertarlos en una cola, para que luego sean insertados en la herramienta de almacenamiento de datos. Una vez ahí se procede a las consultas que darán como resultado tablas que se podrán graficar y poner en mapas de Google Maps, por ejemplo. La siguiente figura muestra un diseño típico de publishers y subscribers de mensajes. Estos mensajes son los que tomaremos para cambiar el formato e insertarlos en otra entidad de almacenamiento hasta el destino final que será la visualización.

Las entidades que toman los mensajes para enviarlos a otro punto de la cadena, se llaman publishers, y las que los toman, se llaman subscribers.

En la arquitectura de software, publish-subscribe es un patrón de mensajería donde los remitentes de mensajes, llamados editores, no programan los mensajes que se envían directamente a receptores específicos, llamados abonados, sino que categorizan los mensajes publicados en clases sin saber cuáles son los suscriptores.

Del mismo modo, los suscriptores expresan interés en una o más clases y sólo reciben mensajes que son de interés, sin conocimiento de qué editores, si los hay, existen. Este modelo es similar al de la sintonización de las radios. La radio emisora transmite su contenido, y los receptores sintonizan determinado canal para escucharlo.

Publish-subscribe es un hermano del paradigma de la cola de mensajes, y es típicamente una parte de un sistema de middleware orientado a mensajes más grande. La mayoría de los sistemas de mensajería son compatibles con los modelos Pub/Sub y de cola de mensajes en su API, Servicio de mensajes Java (JMS).

Este patrón proporciona una mayor escalabilidad de red y una topología más dinámica, con una

consiguiente disminución de la flexibilidad para modificar el editor y la estructura de los datos publicados.

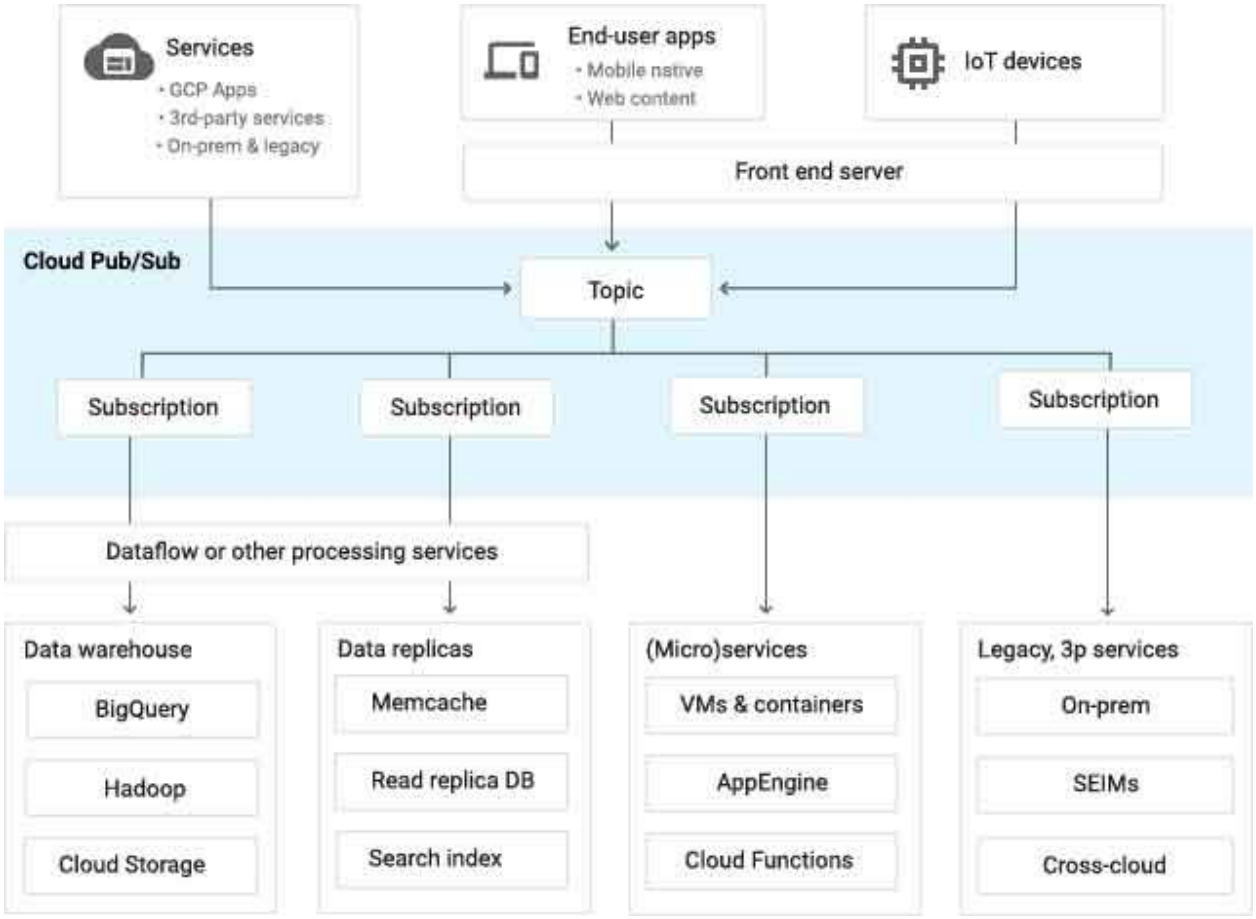

Figura 27: Ejemplo de aplicación de colas de mensajes - Fuente: Documentación Google Cloud.

Creación del entorno

Lo principal antes de realizar aplicaciones en la nube, es registrarse y crear las cuentas necesarias para tener acceso al conjunto de datos completo.
Se muestran los elementos necesarios para ejecutar el código manualmente en el Backend. Una aplicación comercial utilizará otras herramientas de interfaz de usuario y de automatización, o sea que contará con un Frontend acorde a las necesidades comerciales o de distribución.

Pasos del proceso:

1. Se inicia con datos de tráfico almacenados en archivos CSV provenientes de los equipos celulares de los usuarios. Cada uno posee un ID identificatorio.
2. Procesamiento de mensajes en una cola Pub/Sub de Google Cloud.
3. Invertir latitud y longitud de los geocódigos para convertir las coordenadas en una dirección.
4. Calcular la elevación sobre el nivel del mar.

5. Convertir de Tiempo Universal Coordinado (UTC) a la hora local consultando la zona horaria en la que se encuentra cada ubicación.
6. El programa escribe los datos, con la información contextual geográfica agregada, en un dataset de BigQuery para su análisis.
7. Visualiza los datos como mapas de calor superpuestos sobre un mapa geográfico.

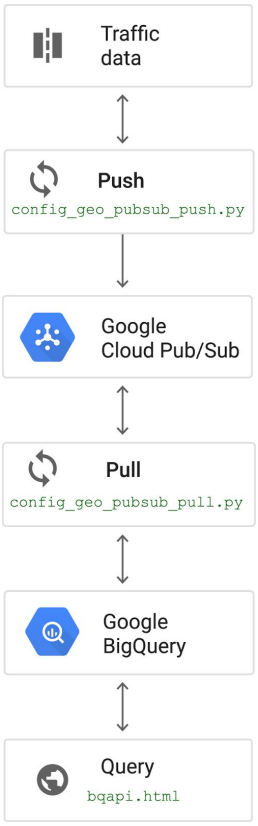

Figura 28: Diagrama de ejecución de scripts.

El diagrama ilustra los principales componentes y cómo se mueven los datos.
Se inicia tomando los datos de archivos CSV en una carpeta. Se considera que llegan a través de otra aplicación, provenientes de celulares de los usuarios.

A partir de ahí, se convierten en mensajes en la cola de la aplicación Pub/Sub, para luego ser insertados en BigQuery. A partir de ahí se genera la información necesaria para armar las coordenadas que necesita la API de Google Maps para mostrar un heat map (mapa de calor) de los datos.

Objetivos
Los elementos necesarios para obtener un entorno de mensajería, importación y visualización son los siguientes:

1. Creación de credenciales.

2. Configuración de Cloud Pub/Sub y BigQuery.
3. Carga y análisis de los datos en BigQuery.
4. Visualización de los datos en una página web.
5. Entender el código.
6. Costos.

Se utilizan entonces componentes facturables del entorno de nube, en este caso de Google Cloud Platform, que incluye:

1. Google BigQuery (5 GB de almacenamiento, 5 GB de inserciones de streaming)
2. Google Cloud Pub/Sub (<200k operaciones)
3. API de Google Maps

El costo de ejecutar este prototipo variará dependiendo del tiempo de ejecución.
El plan estándar de Google Maps API, como varios proveedores de nube, ofrece una cuota libre y una facturación de pago a medida que se supera la cuota.
Debe tener una licencia de Google Maps para cualquier aplicación que restrinja el acceso, como detrás de un firewall o en una intranet corporativa.

Configurar Google Cloud Platform

1. Se crea o selecciona un proyecto existente en la Google Cloud Platform Console.
2. Se precisa habilitar la facturación para el proyecto.
3. Se habilitan las API de Cloud Platform necesarias.

Las API necesarias incluyen las siguientes funcionalidades:

- API de BigQueryGoogle Cloud Pub/Sub API
- Google Cloud Storage
- API de geocodificación de Google Maps
- Google Maps Elevation API
- API de zona horaria de Google Maps
- API de JavaScript de Google Maps
- Configure su entorno de desarrollo

Luego es necesario instalar el SDK de la nube.
Inicializar y autenticar la herramienta de línea de comandos gcloud:

```
$ gcloud init
```

Configurar el entorno de desarrollo de Python
Se instala Python en Linux , OSX ó Windows.

En Debian / Ubuntu, se agrega build-essential:

```
$ sudo apt-get install build-essential libssl-dev libffi-dev python-dev
```

Nota: Se toma el directorio raíz para el código fuente se denomina “bigquery-reverse-geolocation”.
Para sistemas basados en Debian, instale requisitos previos adicionales.
Cambie al directorio "resources" en donde se localizarán todos los scripts, en este caso puede ser "resources":

```
$ cd resources
```

Hay que instalar “pip”, se aconseja no utilizar Mac OSX porque no se pueden cumplir los requisitos mínimos de versiones de software:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

Es necesario instalar los requisitos previos:

```
$ sudo pip install -r requirements.txt
```

Creación de credenciales
Para realizar este procedimiento, se necesitan las siguientes credenciales, de tal forma que se pueda acceder a los recursos de nube desde un browser o desde una aplicación:

1. Un ID de cliente OAuth 2.0.
2. Una clave de servidor de API de Google.
3. Una clave del navegador de la API de Google.

Para obtener las claves, hay que crear primero un ID de cliente que puede utilizar para autenticar las solicitudes de usuario final a BigQuery. Hay que seguir estos pasos:

1- Tomar la dirección IPv4 de la computadora. Por ejemplo, en el navegador, entrar en una página como http://ip-lookup.net. Si la computadora está en una intranet corporativa, necesita obtener la dirección de su sistema operativo. Por ejemplo, ejecute para Linux:

```
$ ifconfig
```

Y para Windows:

```
c:\> ipconfig -all
```

Desde la página Credenciales en la consola de plataforma de Cloud se crean una a una. Si tiene más de un proyecto, hay que seleccionar uno. Luego:

1. Hay que hacer clic en Crear credenciales > ID de cliente de OAuth.
2. Seleccionar Aplicación web.
3. Hay que ingresar "ID de cliente de API de Google Maps" en el campo Nombre.

4. Se agregan las dos URL siguientes de origen en la sección Restricciones, en el cuadro en donde se agregan los posibles orígenes de JavaScript Autorizados.
5. Se reemplaza [YOUR_IP_ADDRESS] con la dirección IPv4 de su computadora.

```
Http: // [YOUR_IP_ADDRESS]: 8000
Https: // [YOUR_IP_ADDRESS]: 8000
```

La adición de estas URL permite a un usuario final acceder a datos de BigQuery a través del JavaScript que se ejecuta en un navegador. Necesita esta autorización para otros pasos posteriores en el procedimiento, por ejemplo para mostrar los datos en un mapa en el navegador web.

Creación de una clave de servidor

1. En la consola de plataforma de Cloud, se precisa hacer clic en Crear credenciales > clave de API.
2. Se hace clic en la tecla Restringir.
3. Se agrega el nombre de la clave "Clave de servidor de procedimiento de mapas".
4. En la sección Restricción de teclas, se selecciona Direcciones IP.
5. En el campo Aceptar peticiones de estas direcciones IP del servidor, introducir la dirección IPv4 de su equipo, que se anotó en la sección anterior.
6. Se hace luego clic en Guardar.

Creación de una clave del navegador
La clave del navegador es un requisito para usar la API JavaScript de Google Maps. Hay que seguir entonces estos pasos:

1. Se hace clic en Crear credenciales > clave de API.
2. Se hace clic en la tecla Restringir.
3. Nombre la clave "Clave de navegador de procedimiento de mapas".
4. Se hace clic en Guardar.

Configuración de Cloud Pub/Sub
Cloud Pub/Sub es la cola de mensajes que maneja el movimiento de los datos de archivos CSV a BigQuery. Se debe crear un "topic", que publica los mensajes y una "subscription" o suscripción, que recibe los mensajes publicados.

Crear un tema Pub/Sub de Cloud
El "topic" o tema publica los mensajes. Hay que crearlo de la siguiente manera:

1. Hay que posicionarse en la página de la lista de temas de Cloud Pub/Sub en la consola de plataforma de Cloud.
2. Se hace clic en Crear un "topic". Se abrirá un cuadro de diálogo.

3. Se agrega el nombre que se quiere poner al topic o tema, en este caso "report" al final de la ruta de acceso que se proporciona en el campo Nombre. El camino es determinado por el sistema.
4. Los temas Pub/Sub de Cloud requieren un nombre.
5. Se hace clic en Crear.

Crear una suscripción Cloud Pub/Sub
La suscripción recibe los mensajes publicados. Se siguen estos pasos para crear la "subscription":

1. En la lista de "topics", en la fila que contiene el tema de "report" , se hace clic en la flecha hacia abajo en el extremo derecho de la fila.
2. Se hace clic en "Nueva suscripción" para abrir la página Crear una nueva suscripción.
3. Se escribe "mysubscription" al final de la ruta que se proporciona en el campo Nombre.
4. Los temas Pub/Sub de Cloud requieren un nombre.
5. Se selecciona Pull para el tipo de entrega.
6. Se hace clic en Crear.

Configuración de BigQuery.
Para preparar BigQuery para recibir los datos que desea analizar, se debe crear un conjunto de datos, que es el contenedor lógico para las tablas de datos y, a continuación, agregar una tabla al conjunto de datos que especifica el esquema y almacena los datos en el formato especificado.
Hay que acceder al directorio de recursos en donde se encuentran los scripts:

```
$ cd resources
```

Cree el conjunto de datos vacío, reportes_buenosaires :

```
$ bq mk reportes_buenosaires
```

Al terminal se muestra un mensaje de confirmación.
Agregue la tabla, geocoded_report , al conjunto de datos:

```
$ bq mk --schema geocoded_report.json reportes_buenosaires.geocoded_report
```

El terminal imprime un mensaje de confirmación.
Este comando crea una tabla que se ajusta al esquema definido en geocoded_report.json.

Visualización del esquema
Para ver el esquema en la consola de BigQuery, se siguen estos pasos:

1. Se abre la Consola de BigQuery.
2. Cerca de la parte superior del panel izquierdo, se marca el nombre del proyecto.
3. Se ve el esquema debajo del nombre del proyecto.
4. Se expande el nodo.
5. Se hace clic en la tabla.

6. Se ve el esquema que se muestra en el panel derecho de la consola.
7. El esquema de BigQuery se puede ver en una tabla para este caso en particular en donde el esquema se corresponde los parámetros del sensor de localización de Sensingkit. Se mapea con el CSV que se recibe en el storage de Google Cloud.

timestamp	TIMESTAMP
timeIntervalSince1970	FLOAT
latitude	FLOAT
longitude	FLOAT
altitude	FLOAT
horizontalAccuracy	INTEGER
verticalAccuracy	INTEGER
speed	INTEGER
course	INTEGER

Tabla 5: Ejemplo de esquema de la tabla de BigQuery.

Cargando los datos en BigQuery

Ahora es necesario importar los datos de los archivos CSV, transcodificar los datos y cargarlos en una tabla de BigQuery.

Ingresar los datos al Sistema

Para ingresar los datos al tema, hay que configurar los parámetros de los scripts y, a continuación, se debe ejecutar la secuencia de comandos Python.

Modificación del archivo de instalación

Se utiliza un editor de texto preferido para editar setup.yaml:

```
Env:
 # Cambiar a su ID de proyecto
     PROJECT_ID: 'smart_ba'
 # Cambiar a datasetid
     DATASET_ID: 'reportes_buenosaires'
 # Cambiar a tabla
     TABLE_ID: 'geocoded_report'
 # Cambiar esto a su tema pubsub
     PUBSUB_TOPIC: 'proyectos / su-proyecto-id / temas / tráfico'
 # Cambie lo siguiente a su rootdir
     ROOTDIR: '/ tmp / creds / data'
 # Cambie lo siguiente a su suscripción de extracción
     SUBSCRIPCIÓN: 'projects / your-project-id / suscripciones /
mysubscription'
 # Cambie a la clave de la API de Google Maps, consulte
https://developers.google.com/maps/web-services/.
     MAPS_API_KEY: 'Su clave de servidor'
```

Se debe reemplazar:

Para PROJECT_ID , PROJECT_ID your-project-id con su ID de proyecto. Se mantienen las comillas simples en este y todos los demás valores que reemplaza.
Para DATASET_ID , reportes_buenosaires.
Para TABLE_ID , no cambiar geocoded_report .
Para PUBSUB_TOPIC , PUBSUB_TOPIC your-project-id con su ID de proyecto.
Para ROOTDIR , resources/data el camino resources/data por resources/data.
Para SUBSCRIPTION , SUBSCRIPTION your-project-id con su ID de proyecto.
Para MAPS_API_KEY , MAPS_API_KEY Your-server-key por la Your-server-key del servidor que creó y MAPS_API_KEY Your-server-key del servidor de MAPS_API_KEY Maps".

Luego se guarda y cierra el archivo.

Ejecución de la secuencia de comandos de inserción
Desde el directorio de los scripts "bigquery-reverse-geolocation", se debe ejecutar la secuencia de comandos Python que rellena el tema Pub/Sub de Cloud.
Se debe entonces cambiar el directorio a la raíz del proyecto:

```
$ cd ..
```

Ahora sí, se debe ejecutar el script:

```
$ python config_geo_pubsub_push.py
```

Entonces luego se ve una lista líneas de salida. Esta salida confirma que se ha interpretado cada línea de datos de los archivos CSV. Si no se ven muchas líneas de tal salida, se debe comprobar que esté correcto el archivo setup.yaml y que la información sea correcta para la ruta y el nombre. Puede tomar algún tiempo ejecutar este script.

Descripción del script de inserción
El código en config_geo_pubsub_push.py realiza varias tareas:

1. En primer lugar, el código crea un cliente Cloud Pub/Sub.
2. Luego encuentra un archivo de datos CSV y lo abre.
3. Para cada línea del archivo CSV, el script realiza una conversión básica en los valores de latitud y longitud para darles formato en unidades de grados a timestamp.
4. Formatea un timestamp de tiempo basada en la información de tiempo en el archivo CSV y guarda la marca de tiempo en la variable msg_attributes.
5. Después de registrar los valores en la ventana de terminal, el código formatea los datos en una línea y publica los datos en el tema Pub/Sub de Cloud.

```
#!/usr/bin/env python
# Copyright 2016 Google Inc. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
```

```
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""
This script reads traffic sensor data from a set of CSV files,
adds vehicle IDs, geoencodes the files  and publishes that data
to Cloud Pub/Sub. If you run it on a GCE instance, this instance must be
created with the "Cloud Platform" Project Access enabled. Click on
"Show advanced options" when creating the image to find this setting.

Usage:

Run the script passing in the location of the folder that contains the CSV
files.
% python geo_pubsub.py --fileloc 'your_folder_location'
Run 'python traffic_pubsub_generator.py -h' for more information.
"""
import argparse
import base64
import csv
import datetime
import random
import sys
import time
import os
import datetime
import yaml
import googlemaps

from apiclient import discovery
from dateutil.parser import parse
import httplib2
from oauth2client import client as oauth2client

with open("resources/setup.yaml", 'r') as  varfile:
    cfg = yaml.load(varfile)

# Defaults to an environment variable.
# Change to your traffic topic name. Can override on command line.
TRAFFIC_TOPIC = cfg["env"]["PUBSUB_TOPIC"]
PUBSUB_SCOPES = ['https://www.googleapis.com/auth/pubsub']
NUM_RETRIES = 3
ROOTDIR = cfg["env"]["ROOTDIR"]

# [START createclient]
def create_pubsub_client(http=None):
```

```
    credentials = oauth2client.GoogleCredentials.get_application_default()
    if credentials.create_scoped_required():
        credentials = credentials.create_scoped(PUBSUB_SCOPES)
    if not http:
        http = httplib2.Http()
    credentials.authorize(http)
    return discovery.build('pubsub', 'v1', http=http)
# [END createclient]

# [START publish]
def publish(client, pubsub_topic, data_line, msg_attributes=None):
    """Publish to the given pubsub topic."""
    data = base64.b64encode(data_line)
    msg_payload = {'data': data}
    if msg_attributes:
        msg_payload['attributes'] = msg_attributes
    body = {'messages': [msg_payload]}
    resp = client.projects().topics().publish(
        topic=pubsub_topic, body=body).execute(num_retries=NUM_RETRIES)
    return resp
# [END publish]

def create_timestamp(hms,dmy):
    """Format two time/date columns as a datetime object"""
    h = int(hms[0:2])
    m = int(hms[2:4])
    s = int(hms[4:6])

    d= int(dmy[0:2])
    m = int(dmy[2:4])
    y = int(dmy[4:6]) + 2000

    return (str(datetime.datetime(y,m,d,h,m,s)))


def main(argv):
    parser = argparse.ArgumentParser()

    parser.add_argument("--fileloc", default=ROOTDIR, help="input folder
with csv files")
    parser.add_argument("--topic", default=TRAFFIC_TOPIC,
                        help="The pubsub topic to publish to. " +
                        "Should already exist.")

    args = parser.parse_args()

    pubsub_topic = args.topic
    print "Publishing to pubsub topic: %s" % pubsub_topic

    rootdir = args.fileloc
    print "Folder to process: %s" % rootdir
```

```
    client = create_pubsub_client()

    for subdir, dirs, files in os.walk(rootdir):
        for file in files:
                        name_ext = file.split(".")
            vehicleID = name_ext[0][15:]

            myfile = os.path.join(subdir,file)
            print myfile
            line_count = 0
            # [START processcsv]
            with open(myfile) as data_file:
                reader = csv.reader(data_file)
                for line in reader:
                    line_count += 1

                    if line_count > 1:
                        # Convert NMEA GPS format to decimal degrees.
                        # See
http://www.gpsinformation.org/dale/nmea.htm#position for NMEA GPS format
details.
                        lat = float(line[3][0:2])
                        lng = float(line[5][0:3])
                        lng_minutes = float(line[5][3:])/60
                        lat_minutes = float(line[3][2:])/60
                        latitude = lat + lat_minutes
                        longitude =  0 - (lng + lng_minutes)
                        ts = create_timestamp(line[1],line[9])
                        msg_attributes = {'timestamp': ts}
                        print "Vehicle ID: {0}, location: {1}, {2}; speed:
{3} mph, bearing: {4} degrees".format(vehicleID, latitude,longitude,
line[7], line[8])
                        proc_line =  "{0}, {1}, {2}, {3} ,{4}
".format(vehicleID, latitude,longitude, line[7], line[8])
                        publish(client, pubsub_topic, proc_line,
msg_attributes)
            # [END processcsv]

if __name__ == '__main__':
        main(sys.argv)
```

Obtención de datos del "topic"

Para extraer los datos del "topic" y cargarlos en la tabla de BigQuery, se precisa ejecutar otro script de Python. Este script utiliza el mismo archivo setup.yaml anterior.

Ejecución de la secuencia de comandos de extracción

Se ejecuta luego un script Python que extrae datos de Cloud Pub/Sub y lo carga en BigQuery:

```
$ python config_geo_pubsub_pull.py
```

Con este script se debería ver un mensaje por cada nueva línea agregada. Esta salida confirma que cada línea de datos se ha cargado en su tabla de BigQuery.

Puede tomar un tiempo extraer todos los datos del "topic". Cuando se termina de ejecutar la carga de datos, la ventana de terminal dejará de mostrar líneas de salida mientras espera datos adicionales. Se puede salir del proceso en cualquier momento pulsando Control + C.

Descripción de la secuencia de comandos de extracción
Cuando ejecuta config_geo_pubsub_pull.py, el proceso es bastante complejo.
En primer lugar, el código crea un objeto cliente Cloud Pub/Sub, exactamente igual que el script push. El código también establece algunos valores de configuración, como el tamaño de un lote de mensajes y algunos límites para que las operaciones de geocodificación permanezcan dentro de las cuotas diarias.

El script se llama Config_geo_pubsub_pull.py:

```
#!/usr/bin/env python
# Copyright 2016 Google Inc. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import sys
import base64
from apiclient import discovery
from dateutil.parser import parse
import httplib2
import yaml
import googlemaps
import time
import datetime
import uuid
import json
import signal
import sys
# from oauth2client.client import GoogleCredentials
from oauth2client import client as oauth2client

with open("resources/setup.yaml", 'r') as  varfile:
    cfg = yaml.load(varfile)

# Uses an environment variable by default. Change this value to the name
```

```
of your traffic topic.
# Can override on the command line.
TRAFFIC_TOPIC = cfg["env"]["PUBSUB_TOPIC"]
PUBSUB_SCOPES = ['https://www.googleapis.com/auth/pubsub']
running_proc = True

def signal_term_handler(signal, frame):
    global running_proc
    print "Exiting application"
    running_proc = False
    sys.exit(0)



def create_pubsub_client(http=None):
    credentials = oauth2client.GoogleCredentials.get_application_default()
    if credentials.create_scoped_required():
        credentials = credentials.create_scoped(PUBSUB_SCOPES)
    if not http:
        http = httplib2.Http()
    credentials.authorize(http)
    return discovery.build('pubsub', 'v1', http=http)

def create_bigquery_client():
    credentials = oauth2client.GoogleCredentials.get_application_default()
    # Construct the service object for interacting with the BigQuery API.
    return discovery.build('bigquery', 'v2', credentials=credentials)

def stream_row_to_bigquery(bigquery, row,
                           num_retries=5):
    # Generate a unique row ID so retries
    # don't accidentally insert duplicates.
    insert_all_data = {
        'insertId': str(uuid.uuid4()),
        'rows': [{'json': row}]
    }
    return bigquery.tabledata().insertAll(
        projectId=cfg["env"]["PROJECT_ID"],
        datasetId=cfg["env"]["DATASET_ID"],
        tableId=cfg["env"]["TABLE_ID"],
        body=insert_all_data).execute(num_retries=num_retries)

# Use Maps API Geocoding service to convert lat,lng into a human readable
address.
def reverse_geocode(gmaps, latitude, longitude):
    return gmaps.reverse_geocode((latitude, longitude))

# Extract a named property, e.g. formatted_address, from the Geocoding API
response.
def extract_address(list, property):
    address = ""
    if(list[0] is not None):
        address = list[0][property]
```

```
    return address

# Extract a structured address component, e.g. postal_code, from a
Geocoding API response.
def extract_component(list, property):
    val = ""
    for address in list:
        for component in address["address_components"]:
            if component["types"][0] == property:
                val = component["long_name"]
                break
    return val

# Calculate elevation using Google Maps Elevation API.
def get_elevation(gmaps, latitude, longitude):
    elevation = gmaps.elevation((latitude, longitude))
    elevation_metres = None
    if(len(elevation)>0):
        elevation_metres = elevation[0]["elevation"]
    return elevation_metres

# Get the timezone including any DST offset for the time the GPS position
was recorded.
def get_timezone(gmaps, latitude, longitude, posix_time):
    return gmaps.timezone((latitude, longitude), timestamp=posix_time)

def get_local_time(timezone_response):
    # get offset from UTC
    rawOffset = float(timezone_response["rawOffset"])
    # get any daylight savings offset
    dstOffset = float(timezone_response["dstOffset"])

    # combine for total offset from UTC
    return rawOffset + dstOffset

# [START maininit]
def main(argv):

    client = create_pubsub_client()

    # You can fetch multiple messages with a single API call.
    batch_size = 100

    # Options to limit number of geocodes e.g to stay under daily quota.
    geocode_counter = 0
    geocode_limit = 10

    # Option to wait for some time until daily quotas are reset.
    wait_timeout = 2
# [END maininit]
# [START createmaps]
    # Create a Google Maps API client.
    gmaps = googlemaps.Client(key=cfg["env"]["MAPS_API_KEY"])
```

```
    subscription = cfg["env"]["SUBSCRIPTION"]

    # Create a POST body for the Cloud Pub/Sub request.
    body = {
        # Setting ReturnImmediately to False instructs the API to wait
        # to collect the message up to the size of MaxEvents, or until
        # the timeout.
        'returnImmediately': False,
        'maxMessages': batch_size,
    }
# [END createmaps]
    signal.signal(signal.SIGINT, signal_term_handler)
    #[START pullmsgs]
    while running_proc:
        # Pull messages from Cloud Pub/Sub
        resp = client.projects().subscriptions().pull(
            subscription=subscription, body=body).execute()

        received_messages = resp.get('receivedMessages')
    # [END pullmsgs]


        if received_messages is not None:
            ack_ids = []
            bq = create_bigquery_client()
            for received_message in received_messages:
                pubsub_message = received_message.get('message')
                if pubsub_message:
                    # process messages
                    msg =
base64.b64decode(str(pubsub_message.get('data')))

                    # We stored time as a message attribute.
                    ts = pubsub_message["attributes"]["timestamp"]

                    # Create a datetime object so we can get a POSIX
timestamp for TimeZone API.
                    utc_time = datetime.datetime.strptime(ts, "%Y-%m-%d
%H:%M:%S")
                    posix_time = time.mktime(utc_time.timetuple())

                    # Our messages are in a comma-separated string.
                    #Split into a list
                    data_list = msg.split(",")
                    #[START extract]
                    # Extract latitude,longitude for input into Google
Maps API calls.
                    latitude = float(data_list[1])
                    longitude = float(data_list[2])

                    # Construct a row object that matches the BigQuery
table schema.
                    row = { '[VARIABLE_1]': data_list[0], 'VARIABLE_2':
```

```
None, 'VARIABLE_3': 0, 'VARIABLE_4':"", 'VARIABLE_5':"",
'VARIABLE_6':data_list[3], 'VARIABLE_7':data_list[4], 'Elevation':None,
'Latitude':latitude, 'Longitude': longitude }

                    # Maps API Geocoding has a daily limit - this lets us
limit API calls during development.
                    if geocode_counter <= geocode_limit:

                        # Reverse geocode the latitude, longitude to get
street address, city, region, etc.
                        address_list = reverse_geocode(gmaps, latitude,
longitude)
                    # [END extract]
                        #Save the formatted address for insert into
BigQuery.
                        if(len(address_list) > 0):
                            row["Address"] = extract_address(address_list,
"formatted_address")
                            #extract the zip or postal code if one is
returned
                            row["Zipcode"] =
extract_component(address_list, "postal_code")

                        # Increment counter - in case you want to limit
daily geocodes.
                        geocode_counter += 1

                        # get elevation
                        row["Elevation"] = get_elevation(gmaps, latitude,
longitude)

                        # Get the timezone, pass in original timestamp in
case DST applied at that time.
                        timezone = get_timezone(gmaps, latitude,
longitude, posix_time)

                        # Store DST offset so can display/query UTC time
as local time.
                        if(timezone["rawOffset"] is not None):
                            row["Offset"] = get_local_time(timezone)

                        row["UTCTime"] = ts
                        # [START saverow]
                        # save a row to BigQuery
                        result = stream_row_to_bigquery(bq, row)
                        # [END saverow]

                        # Addresses can contain non-ascii characters, for
simplicity we'll replace non ascii characters.
                        # This is just for command line output.
                        addr = row['Address'].encode('ascii', 'replace')
                        msg = "Appended one row to BigQuery."
                        print msg
```

```
                        msg = "Address: {0}".format(addr)
                        print msg
                        msg = "Elevation: {0}
metres".format(row["Elevation"])
                        print msg
                        msg = "Timezone:
{0}".format(timezone["timeZoneId"])
                        print msg
                        print " "
                    else:
                        time.sleep(wait_timeout)
                        geocode_counter = 0
                        print "counter reset"

                    # Get the message's ack ID.
                    ack_ids.append(received_message.get('ackId'))

            # Create a POST body for the acknowledge request.
            ack_body = {'ackIds': ack_ids}

            # Acknowledge the message.
            client.projects().subscriptions().acknowledge(
                subscription=subscription, body=ack_body).execute()



if __name__ == '__main__':
            main(sys.argv)
```

El código crea una instancia del cliente de API de Google Maps y crea un cuerpo de HTTP POST para las solicitudes de Cloud Pub/Sub que se publicarán. El código también recupera el nombre de la suscripción del archivo setup.yaml.

A continuación, el código entra en un bucle que se ejecuta hasta que finaliza el proceso, o hasta que se presiona Control + C. Este bucle extrae mensajes utilizando el nombre de suscripción Cloud Pub/Sub en caché anterior:
Mientras ejecuta_proc:

```
# Extraer mensajes de Cloud Pub/Sub
     Resp = client.projects (). Suscripciones (). Pull (
         Subscription = suscripción, body = body) .execute ()
Receive_messages = resp.get ('receivedMessages')
```

El código procesa cada mensaje. El punto clave aquí es que el código utiliza la API de Google Maps para invertir el geocódigo de latitud y longitud a la dirección de la calle:

```
 # Extraer latitud y longitud para ingresar a las llamadas a la API de
Google Maps.
 Latitude = float (data_list [1])
 Longitude = float (data_list [2])
```

```
 # Construya un objeto de fila que coincida con el esquema de tabla de
BigQuery.
 Row = {'VehicleID': data_list [0], 'UTCTime': Ninguno, 'Offset': 0,
'Dirección' : Data_list [4], 'Elevation': Ninguno, 'Latitude': latitud,
'Longitud': longitude}

 # Maps API Geocoding tiene un límite diario - esto nos permite limitar
las llamadas API durante el desarrollo.
 Si geocode_counter <= geocode_limit:

     # Geocode inverso la latitud, longitud para obtener dirección,
ciudad, región, etc.
     Address_list = reverse_geocode (gmaps, latitud, longitud)
```

Con este código se guarda la fila de datos en BigQuery:

```
# Guardar una fila en BigQuery
 Resultado = stream_row_to_bigquery (bq, fila)
```

Finalmente, el código envía un acuse de recibo para el mensaje original y luego repite el bucle.

Análisis de los datos

Ahora que han transcodificado y cargado los datos en BigQuery, se puede utilizar el mismo BigQuery para obtener información. En este momento se puede utilizar la consola de BigQuery para consultar los datos y procesarlos, y sacar los insights de las diferentes tablas que se hayan cargado.

Entonces hay que abrir la Consola de BigQuery. En el cuadro de texto se puede escribir la siguiente consulta que genera un resultado útil:

```
SELECT
GROUP BY ORDER BY
```

Visualización de los datos

Luego de cargar los datos de los archivos CSV, cargarlos en las colas de Pub/Sub, y luego en BigQuery, se puede utilizar Google Maps para visualizar los datos almacenados y filtrados por diferentes consultas. Un forma de visualizar los datos es superponer una visualización de mapa de calor en un mapa de la región. El mapa de calor muestra el volumen de actividad de reportes de usuarios capturado en los datos en BigQuery.

El ejemplo proporcionado utiliza OAuth 2.0 para autenticar al usuario para el servicio BigQuery. Puede elegirse otro enfoque que podría ser más adecuado para otros escenarios. Por ejemplo, puede exportar los resultados de consulta de BigQuery y crear una capa de mapa estático que no requiere que el usuario final se autentifique en BigQuery, o puede configurar la autenticación mediante una cuenta de servicio y un servidor proxy.

Para mostrar la visualización de datos, hay que levantar un servidor web y crear un archivo que contenga la API para ver el mapa.

Hemos creado un archivo de ejemplo llamado "bqapi.html" como el fuente de la página web. Se deben utilizar las claves y credenciales creadas anteriormente. Los valores se encuentran en la Consola de plataforma de Cloud en la página:

```
#!/usr/bin/env python
# Copyright 2016 Google Inc. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import argparse
import base64
import csv
import datetime
import random
import sys
import time
import os
import datetime
import yaml
import googlemaps

from apiclient import discovery
from dateutil.parser import parse
import httplib2
from oauth2client import client as oauth2client

with open("resources/setup.yaml", 'r') as  varfile:
    cfg = yaml.load(varfile)

# Defaults to an environment variable.
# Change to your traffic topic name. Can override on command line.
TRAFFIC_TOPIC = cfg["env"]["PUBSUB_TOPIC"]
PUBSUB_SCOPES = ['https://www.googleapis.com/auth/pubsub']
NUM_RETRIES = 3
ROOTDIR = cfg["env"]["ROOTDIR"]

# [START createclient]
def create_pubsub_client(http=None):
    credentials = oauth2client.GoogleCredentials.get_application_default()
    if credentials.create_scoped_required():
        credentials = credentials.create_scoped(PUBSUB_SCOPES)
    if not http:
        http = httplib2.Http()
```

```
    credentials.authorize(http)
    return discovery.build('pubsub', 'v1', http=http)
# [END createclient]

# [START publish]
def publish(client, pubsub_topic, data_line, msg_attributes=None):
    """Publish to the given pubsub topic."""
    data = base64.b64encode(data_line)
    msg_payload = {'data': data}
    if msg_attributes:
        msg_payload['attributes'] = msg_attributes
    body = {'messages': [msg_payload]}
    resp = client.projects().topics().publish(
        topic=pubsub_topic, body=body).execute(num_retries=NUM_RETRIES)
    return resp
# [END publish]

def create_timestamp(hms,dmy):
    """Format two time/date columns as a datetime object"""
    h = int(hms[0:2])
    m = int(hms[2:4])
    s = int(hms[4:6])

    d= int(dmy[0:2])
    m = int(dmy[2:4])
    y = int(dmy[4:6]) + 2000

    return (str(datetime.datetime(y,m,d,h,m,s)))


def main(argv):
    parser = argparse.ArgumentParser()

    parser.add_argument("--fileloc", default=ROOTDIR, help="input folder
with csv files")
    parser.add_argument("--topic", default=TRAFFIC_TOPIC,
                        help="The pubsub 'traffic' topic to publish to. "
+
                        "Should already exist.")

    args = parser.parse_args()

    pubsub_topic = args.topic
    print "Publishing to pubsub 'traffic' topic: %s" % pubsub_topic

    rootdir = args.fileloc
    print "Folder to process: %s" % rootdir


    client = create_pubsub_client()

    for subdir, dirs, files in os.walk(rootdir):
        for file in files:
```

```
            # San Diego data file names include trip ID, so use this to
identify each journey.
            name_ext = file.split(".")
            vehicleID = name_ext[0][15:]

            myfile = os.path.join(subdir,file)
            print myfile
            line_count = 0
            # [START processcsv]
            with open(myfile) as data_file:
                reader = csv.reader(data_file)
                for line in reader:
                    line_count += 1

                    if line_count > 1:
                        # Convert NMEA GPS format to decimal degrees.
                        # See
http://www.gpsinformation.org/dale/nmea.htm#position for NMEA GPS format
details.
                        lat = float(line[3][0:2])
                        lng = float(line[5][0:3])
                        lng_minutes = float(line[5][3:])/60
                        lat_minutes = float(line[3][2:])/60
                        latitude = lat + lat_minutes
                        longitude =  0 - (lng + lng_minutes)
                        ts = create_timestamp(line[1],line[9])
                        msg_attributes = {'timestamp': ts}
                        print "Vehicle ID: {0}, location: {1}, {2}; speed:
{3} mph, bearing: {4} degrees".format(vehicleID, latitude,longitude,
line[7], line[8])
                        proc_line =  "{0}, {1}, {2}, {3} ,{4}
".format(vehicleID, latitude,longitude, line[7], line[8])
                        publish(client, pubsub_topic, proc_line,
msg_attributes)
            # [END processcsv]

if __name__ == '__main__':
        main(sys.argv)
```

Crear las credenciales

Luego hay que realizar una copia del archivo bqapi.html en el directorio donde residen los scripts:

```
bigquery-reverse-geolocation/web/
```

En el siguiente elemento de script, en el atributo src, se debe colocar la clave "Your-Maps-API-Key" de la API de Google Maps Your-Maps-API-Key de su navegador de Google Maps API.

Como ejemplo se puede servir la página web desde el servidor HTTP simple de Python.

En la ventana del terminal, en el directorio bigquery-reverse-geolocation/web de trabajo, hay que ejecutar el servidor web:

```
$ python -m SimpleHTTPServer
```

Desde un navegador web, se debe ir a la siguiente URL reemplazando [YOUR_IP_ADDRESS] con la dirección de su computadora. Esta dirección IP es la que se cargó en un paso anterior y se utilizó para establecer la URL de origen de su ID de cliente OAuth 2.0.

```
http://[YOUR_IP_ADDRESS]:8000/bqapi.html
```

Hay que hacer clic en Permitir en el cuadro de diálogo emergente de autenticación OAuth 2.0.

Después de cargar el mapa, hay que seleccionar la herramienta de rectángulo en la esquina superior izquierda del mapa. Entonces hay que utilizar la herramienta para dibujar un rectángulo alrededor del mapa.
La página muestra un mapa de calor. Las regiones del mapa de calor aparecen en el mapa dependen de los datos que se cargaron en BigQuery.

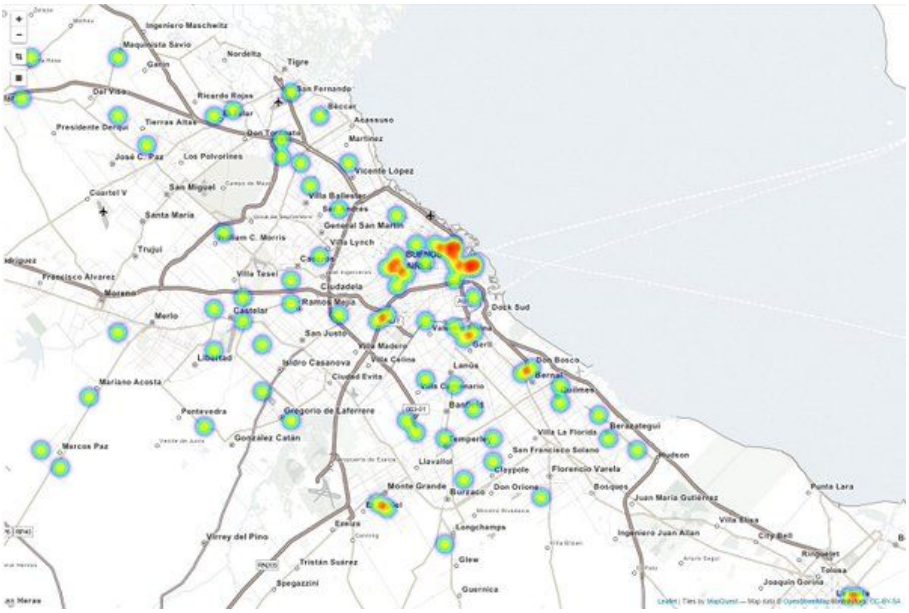

Figura 29: Ejemplo de Heat Map (Mapa de Calor) que se puede lograr con la API de Google.

Entendiendo el script de la página web

La página web utiliza la API JavaScript de Google Maps para realizar su trabajo. En esta sección, se examina de manera más profunda cómo la página autoriza al usuario, recupera datos y se arman las regiones de mapa de calor. El sitio de la API es: https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap.

Autorización del usuario

Las siguientes funciones gestionan la autenticación y la autorización a través de OAuth 2.0. La función llamada "Authorize" hace la solicitud de autorización. La función handleAuthResult

recibe una llamada de la API OAuth 2.0. Si el resultado es exitoso, la función llamada loadAPI carga la API de BigQuery:

```
function authorise(event) {
  gapi.auth.authorize({client_id: clientId, scope: scopes, immediate:
false}, handleAuthResult);
  return false;
}

// If authorized, load BigQuery API.
function handleAuthResult(authResult) {
  if (authResult && !authResult.error) {
    loadApi();
  } else {
    console.log("Sorry, you are not authorised to access BigQuery.")
  }
}

// Load BigQuery client API.
function loadApi(){
  gapi.client.load('bigquery', 'v2').then(
    function() {
      console.log('BigQuery API loaded.');
      createMap();
    }
  );
}
```

Obteniendo los datos

El programa dibuja un rectángulo alrededor de la región del mapa donde desea ver los mapas de calor. Lo que dibuja el usuario con el mouse define un conjunto de límites de coordenadas que restringen el subconjunto de datos que se recuperarán de BigQuery. La función llamada setUpDrawingTools añade un detector de eventos que notifica su código cuando se dibuja el rectángulo.

El callback es manejado por la función rectangleQuery es el siguiente:

```
function rectangleQuery(latLngBounds){
  var queryString = rectangleSQL(latLngBounds.getNorthEast(),
latLngBounds.getSouthWest());
  sendQuery(queryString);
}
```

La función rectangleQuery llama a rectangleSQL, que construye una cadena SQL basada en los límites del rectángulo.

```
// Construct the SQL for a rectangle query.
function rectangleSQL(ne, sw){
  var queryString = "SELECT Latitude, Longitude "
  queryString +=  "FROM [" + projectId + ":" + datasetId + "." +
table_name + "]"
  queryString += " WHERE Latitude > " + sw.lat();
```

```
  queryString += " AND Latitude < " + ne.lat();
  queryString += " AND Longitude > " + sw.lng();
  queryString += " AND Longitude < " + ne.lng();
  queryString += " LIMIT " + recordLimit;
  return queryString;
}
```

Se puede ver que esta función usa las esquinas suroeste y noreste del rectángulo para definir los límites de latitudes y longitudes en el conjunto de datos. Por ejemplo, la longitud representada por sw.lng coincide con el borde vertical izquierdo del rectángulo.

Cualquier longitud en el conjunto de datos que sea mayor a este valor sería a la derecha de ese borde, y por lo tanto dentro de los límites del rectángulo. Una lógica similar se aplica a los otros tres lados del rectángulo.

La función sendQuery ejecuta la consulta a través de la API de BigQuery mediante la Biblioteca de cliente de API de Google para JavaScript :

```
function sendQuery(queryString){
  var request = gapi.client.bigquery.jobs.query({
      "query": queryString,
      "timeoutMs": 30000,
      "datasetId": datasetId,
      "projectId": projectId
  });
  request.execute(function(response) {
      console.log(response);
      checkJobStatus(response.jobReference.jobId);
  });
}
```

La biblioteca cliente produce una URL como la que sigue para consultar los datos a BigQuery:

```
{
"query":"SELECT Latitude, Longitude FROM
[YOUR_PROJECT_ID]:smart_buenosaires.geocoded_journeys WHERE Latitude >
32.685041939169665 AND Latitude < 32.85536439443039 AND Longitude >
-117.31063842773438 AND Longitude < -117.05451965332031 LIMIT 10000"
}
```

BigQuery responde a través de la API con un jobID para saber el estado del trabajo hasta que los resultados estén listos para ser recuperados.
La función checkJobStatus muestra el estado del trabajo periódicamente, llamando al método get con el jobId devuelto por la consulta original. La función de la muestra tiene un tiempo de espera de 500 milisegundos:

```
function checkJobStatus(jobId){
  var request = gapi.client.bigquery.jobs.get({
    "projectId": projectId,
```

```
    "jobId": jobId
  });
  request.execute(function(response){
    if(response.status.errorResult){
      console.log(response.status.error);
    } else {
      if(response.status.state == 'DONE'){
        //get the results
        clearTimeout(jobCheckTimer);
        getQueryResults(jobId);
      } else {
        // No error, not finished, check again in a moment.
        console.log("Job running, waiting 0.5 seconds...");
        jobCheckTimer = setTimeout(checkJobStatus, 500, [jobId]);
      }
    }
  });
}
```

La función getQueryResults llama a getQueryResults. Esta función utiliza el método jobs.getQueryResults para obtener los resultados y luego los pasa a la función doHeatMap:

```
function getQueryResults(jobId){
  var request = gapi.client.bigquery.jobs.getQueryResults({
    "projectId": projectId,
    "jobId": jobId
  });
  request.execute(function(response){
    // Draw a heatmap from the list of rows returned.
    doHeatMap(response.result.rows);
  })
}
```

Mostrando el mapa de calor

Es importante entender que la cantidad de datos que pueden devolverse de las consultas de BigQuery puede ser enorme, a veces equivale a petabytes de datos. Se debe tener cuidado de agregar estos datos de una manera que haga utilizable el modelo para que pueda procesarse y mostrarse en un tiempo razonable.

Por ejemplo, intentar trazar la ubicación de cada fila de datos de reportes sería insostenible en este escenario. Afortunadamente, la API de Maps visualization.HeatmapLayer objeto visualization.HeatmapLayer, administra estas limitaciones. La función doHeatMap crea el mapa de calor y luego superpone la visualización en el mapa que se muestra en el navegador:

```
function doHeatMap(rows){
  // Remove the user drawing.
  if(currentShape){
    currentShape.setMap(null);
  }
  var heatmapData = [];
```

```
  if(heatmap!=null){
    heatmap.setMap(null);
  }
  if(rows){
    for (var i = 0; i < rows.length; i++) {
        var f = rows[i].f;
        var coords = { lat: parseFloat(f[0].v), lng: parseFloat(f[1].v) };
        var latLng = new google.maps.LatLng(coords);
        heatmapData.push(latLng);
    }
    heatmap = new google.maps.visualization.HeatmapLayer({
        data: heatmapData
    });
    heatmap.setMap(map);
  }
}
```

Consejos adicionales
Si se está trabajando con tablas muy grandes, las consultas pueden devolver demasiadas filas para mostrar de forma eficiente en un mapa. Se pueden limitar los resultados agregando una cláusula WHERE o una instrucción LIMIT en la consulta SQL.

BigQuery analiza toda la tabla con cada consulta. Para optimizar el uso de su cuota de BigQuery, seleccione sólo las columnas que necesita. De esta forma se gasta menos dinero.
Las consultas se ejecutan más rápido si almacena la latitud y la longitud como float en lugar de string.

Hay otras maneras de utilizar SQL para ejecutar consultas espaciales con datos en BigQuery, incluidas las consultas que se aproximan a un círculo delimitador y las funciones definidas por el usuario que se pueden utilizar para construir operaciones de geometría más avanzadas.

Correlación de Pearson
La función CORR() en BigQuery es una poderosa herramienta para procesar los datos. Se puede saber qué variables son similares o tienen comportamientos fuera de lo normal. También cuáles son las variables que permiten predecir el futuro.

Se basa en la función matemática de Pearson que permite decir qué tan relacionadas están 2 series de números. Se expresa como un número. Sus parámetros son 2 listas de variables.

Si cuando una variable aumenta, la otra también, y cuando una disminuye, la otra también, entonces se habla de correlación positiva. Cuanto más cerca de 1 esté, más correlacionado está el conjunto de variables. Entonces están alineadas.

Si la relación es inversa, se dice que tiene correlación negativa. Cuando la función correlación da un número cercano a 1, la correlación es alta.

En estadística, el coeficiente de correlación de Pearson es una medida de la relación lineal entre dos variables aleatorias cuantitativas. A diferencia de la covarianza, la correlación de Pearson es independiente de la escala de medida de las variables.

De manera menos formal, podemos definir el coeficiente de correlación de Pearson como un índice que puede utilizarse para medir el grado de relación de dos variables siempre y cuando ambas sean cuantitativas.

Se puede tomar el mismo acercamiento para analizar comportamientos de muchos eventos, como por ejemplo hábitos de consumo, eventos climáticos, distancia geográfica.

Si se deben procesar Terabytes, esta herramienta de análisis es extremadamente potente para trabajar. La API que permite correr queries en segundos, sobre volúmenes grandes de datos. Se pueden correr queries de Gigabytes y obtener los resultados en segundos. Es una herramienta muy poderosa para desarrolladores.
Hay otros programas estadísticos como "R", que no soportan cantidades grandes de registros, esto hace que la función de Correlación de Pearson que provee BigQuery sea realmente la herramienta adecuada.

Utilizando el comando DATE_ADD, se puede llegar a correlacionar las fechas anteriores, de tal forma de poder ver la evolución en el tiempo de cada variable, y de alguna manera predecir el comportamiento.

Si algo sucede en una habitación con sensores, hace falta, por ejemplo, correlacionar las mediciones con eventos anteriores. Por lo tanto de esta forma se pueden tener alertas rápidas sobre comportamientos con significados interesantes para la aplicación. También se puede llegar a predecir la evolución probable de un evento, comparando fechas anteriores.

Procesamiento estadístico de datos con R
Una vez que tenemos los datos en BigQuery, sobre los reportes de los usuarios, y la visualización en un mapa, podemos analizar los datos estadísticamente con mucha más profundidad con R.
Los paquetes bigrquery proporcionan una interfaz R a Google BigQuery. R facilita la recuperación de metadatos sobre proyectos, conjuntos de datos, tablas y trabajos, y proporciona un conveniente contenedor para trabajar con Bigquery.

Reconocimiento de patrones con Machine Learning
El Aprendizaje de Máquina o Machine Learning es el subcampo de la informática que da a las computadoras la capacidad de aprender sin ser explícitamente programadas. Explora el estudio y la construcción de Algoritmos que pueden aprender y hacer predicciones en los datos.

El aprendizaje automático se emplea en una serie de tareas informáticas en las que el diseño y la programación de algoritmos explícitos con un buen rendimiento es difícil o imposible. Por ejemplo el filtrado de spam, la detección de intrusos de la red, reconocimiento óptico de caracteres (OCR), motores de búsqueda y visión por computadora.

Dentro del campo del análisis de datos, el aprendizaje automático es un método utilizado para diseñar modelos y algoritmos complejos que se prestan a la predicción.

Por lo tanto esta herramienta puede llegar a utilizarse con los datos recopilados a través de sensores, para conseguir inferencias sobre los reportes realizados.

Visualización gráfica de resultados
Una vez que se tienen los datos procesados, existe gran cantidad de herramientas que permiten visualizar inteligentemente y en tiempo real las conclusiones. Por ejemplo para visualización son soporte nativo de Bigquery se puede usar:

- Redash.io
- OWOX+Google Sheets
- Tableau
- Qlik
- BIME
- Looker

Otros visualizadores muy poderosos son Vega-Lite (https://vega.github.io/vega-lite/), CCRI (http://www.ccri.com/case-studies/stealth/), Geomesa (http://www.geomesa.org/).

Ahorro de insumos en la nube
Una vez que se haya terminado de utilizar el prototipo, se pueden limpiar los recursos que se crearon en Google Cloud Platform para que no se facture en el futuro. En las siguientes secciones se describe cómo eliminar o desactivar estos recursos.

Eliminación del proyecto
La forma más sencilla de limpiar la mayoría de los recursos de Cloud Platform es eliminar el proyecto desde la consola.
Advertencia: la eliminación de un proyecto tiene las siguientes consecuencias:

- Si se utilizó un proyecto existente, también eliminará cualquier otro trabajo que haya realizado en el proyecto.
- No puede reutilizar el ID de proyecto de un proyecto eliminado. Si creó un ID de proyecto personalizado que planea utilizar en el futuro, debería eliminar los recursos dentro del proyecto.
- Si está explorando varios tutoriales y quickstarts, la reutilización de proyectos en lugar de eliminarlos evita que se excedan los límites de cuota del proyecto.

En la lista de proyectos, se selecciona el proyecto que se desea eliminar y hay que hacer clic en Eliminar proyecto. Después de seleccionar la casilla de verificación junto al nombre del proyecto, hay que hacer clic en Eliminar proyecto.
En el cuadro de diálogo, se escribe el ID del proyecto y, a continuación hay que hacer clic en Apagar para eliminarlo del todo.

Eliminación de datos almacenados en BigQuery
Para eliminar datos almacenados y dejar de consumir espacio de almacenamiento se siguen los siguientes pasos:

1. Hay que acceder a la consola de BigQuery.
2. En el panel de la izquierda, se selecciona el nombre del conjunto de datos y, a continuación, se hace clic en la flecha hacia abajo.
3. Se hace clic en Eliminar conjunto de datos.
4. Luego se siguen las instrucciones para confirmar la eliminación.

Para eliminar los componentes Cloud Pub/Sub:

1. Se abre la página de la lista de temas de Cloud Pub/Sub en la consola de Google Cloud Platform.
2. En la lista de temas, se selecciona la casilla de verificación del tema.
3. Se hace clic en Eliminar y se confirma la operación.

Capítulo 4

Discusiones, ventajas y desventajas

Luego de todo el análisis realizado, pueden surgir diferentes discusiones sobre la aplicabilidad de la solución. Siendo que hay cada vez más tecnología disponible para recolectar y analizar información, es crucial que los organismos de Gobierno hagan uso de los mismos en función de la resolución de problemáticas sociales y urbanas.

Si bien cada partido político puede tener distintas propuestas, la información debería ser objetiva y neutral, de tal forma que se solucionen realmente las cuestiones que tienen que ver con el desarrollo de la vida en las ciudades, la salud, el urbanismo y la capacidad de crecimiento ordenado.

Por lo tanto la tecnología de crowdsensing permite que los propios ciudadanos aporten sus conocimientos del entorno a los organismos correspondientes, y de esta manera no dejan en manos del estado la responsabilidad de tomar las decisiones en forma arbitraria, aunque en lo posible bien intencionada pero inexacta.

Con respecto a las consecuencias negativas de recolectar información masiva, pudimos comentar en el capítulo 2, que en algunas circunstancias es difícil enmascarar la información de las personas, de tal forma de anonimizar, y puede haber entes que utilicen dichos datos en forma delictiva, o nociva para los usuarios. Tal como se gestiona la seguridad en otros ámbitos las leyes deben proteger al ciudadano de actos que vayan contra ellos.

Privacidad y legalidad
Los sistemas para crowdsensing, son sistemas centrados en el ser humano. Es importante preservar la seguridad y la privacidad de las personas. Aplicaciones crowdsensing potencialmente recoger datos de sensores sensibles.

Por ejemplo, las lecturas del sensor GPS se pueden utilizar para estimar los niveles de congestión del tráfico en una comunidad dada. Pero al mismo tiempo estas lecturas del sensor GPS se pueden utilizar para inferir información privada sobre el individuo, como las rutas que toman durante sus desplazamientos diarios, su hogar y lugares de trabajo. Se deben gestionar políticas de seguridad y privacidad inevitablemente ya que la búsqueda de esta información.[5]

Una cuestión importante en crowdsensing es la de la privacidad; Sin mecanismos adecuados de preservación de la privacidad, muchos usuarios no estarán dispuestos a participar en el proceso de recolección de datos. Este documento presenta el estado de la técnica en la preservación de la privacidad de los mecanismos de los sistemas de crowdsensing. Entonces es importante abordar los temas más importantes a considerar en el diseño, implementación y evaluación de los mecanismos de preservación de la privacidad.[6]

5: Internet of Things and Big Data Analytics for Smart and Connected Communities, YUNCHUAN SUN, HOUBING SONG, ANTONIO J. JARA3, AND RONGFANG, 2016
6: Privacy-Preserving Mechanisms for Crowdsensing: Survey and Research Challenges, Luis G. Jaimes and Miguel A. Labrador, 2016

Conclusiones

Tomando como ejemplo la Ciudad de Buenos Aires, vemos que efectivamente hay un largo camino por recorrer para poder aprovechar la información proveniente de crowdsensing.
El proceso para realizar el modelo, demostró ser sencillo y estar al alcance de la mano para ser efectivamente realizable.

Resta claramente implementar la gestión y la elección de los factores de riesgo y posibilidades de ahorro prioritarias para la ciudad, de tal forma que se pueda lograr un impacto real sobre la calidad de vida de los ciudadanos.

Si bien el estudio de las tecnologías para desarrollo de aplicaciones para SmartCities está muy extendida, hay de alguna manera una distancia muy grande entre la teoría y la práctica para que

se adopten los sistemas amplia e institucionalmente.
Siendo que cada vez hay más aplicaciones y sensores en los smartphones de los ciudadanos, se vuelven herramientas fundamentales en la participación ciudadana y la eficiencia de los recursos del gobierno.

Incluir a los ciudadanos en la recolección de datos con crowdsensing aumenta las capacidades de las infraestructuras existentes sin introducir costes adicionales y ha demostrado ser una estrategia en donde todos ganan para las aplicaciones de ciudades inteligentes.

Las herramientas SaaS (Software as a Service), y el crowdsensing, prueban ser hoy en día los catalizadores de la infraestructura de las ciudades para convertirlas en Smart y así lograr los objetivos del mejoramiento de la calidad de vida de la gente.

Trabajos futuros

Aunque fuera del alcance de este trabajo, se puede pensar en ampliar el prototipo para utilizar aplicaciones más sofisticadas en los smartphones, de tal forma que puedan hacer un preprocesamiento de los datos y que las consultas sean más ágiles.

Por otro lado, queda un largo camino por recorrer con respecto al análisis de los datos, y la búsqueda de posibles insights que tengan un impacto económico, político y social real para las ciudades.

Por último, también habría que trabajar en la publicación de aplicaciones para los ciudadanos, para los distintos sistemas operativos de smartphones, y las campañas de concientización para que la gente aporte información útil, que luego redundará en el bienestar público.

Recomendaciones

En el comienzo de la redacción de esta tesis, comencé por buscar software instalable en servidores, porque en el análisis parecía ser la solución que permite más versatilidad.
Hoy en día las herramientas en la nube permiten focalizarse en el estudio de las conclusiones, sin tener que lidiar con el trabajo de la implementación de aplicaciones difíciles de configurar, y que no se integran de manera sencilla.

Los proveedores de soluciones de nube, ya cuentan con la experiencia para acercar a los usuarios, mejores interfaces, más seguras, y sobre todo con APIs tendientes a incrementar la adaptabilidad a diferentes lenguajes de programación y sistemas de autenticación.

Las nuevas aplicaciones basadas en estos sistemas resultan ser más seguras y complejas que las instaladas en Data Centers propios, ya que el nivel de estandarización es tal, que es mucho más costoso, sobre todo operativamente, tener un equipo de soporte local, y es más conveniente especializarse en herramientas de nube.

Agradecimientos

Agradezco a mi familia y amigos que apoyaron mis ideas y me permitieron estudiar sin

interrupciones por todo el tiempo de trabajo.

Agradezco también al ITBA y su staff que me posibilitó avanzar en esta Maestría.

De la misma manera, a los compañeros que me apoyaron y asesoraron en esta búsqueda que espero crezca y aumente más adelante.

Lista de Figuras

Lista de Tablas

Apéndices

Listados de información pública de la Ciudad de Buenos Aires

- Catálogo de datos públicos publicados en data.buenosaires.gob.ar:
http://data.buenosaires.gob.ar/dataset/datos-publicos/resource/e686ca8f-7e46-40ea-a340-987b69d28b5d

- Catálogo de cada uno de los recursos que contiene el portal data.buenosaires.gob.ar. En el mismo se incluye nombre de datasets, recursos y enlace para visualización:
http://data.buenosaires.gob.ar/dataset/datos-publicos/resource/844cf3dd-ede6-4289-a536-de232da8ccb1

- Listado de bases de datos contiene el portal data.buenosaires.gob.ar:

ESPACIOS VERDES
EDIFICIOS PUBLICOS DEL GCBA
METROBUS
ESTABLECIMIENTOS EDUCATIVOS
RECOLECCION DE RESIDUOS SOLIDOS SECOS
PUNTOS DE WI-FI PUBLICOS
CENTROS MEDICOS BARRIALES
SUBTE: ESTACIONES (EN EL FILEZILLA ESTA COMO SUBTERRANEOS)
SUBTE: GTFS
COMISARIAS METROPOLITANA
COMISARIAS PFA
SUBTE: TRENES DESPACHADOS
SUBTE: CRONOGRAMA DE SERVICIO
PUNTOS VERDES
EDIFICIOS PUBLICOS EN 3D
UNIVERSIDADES
VEREDAS
INFORMACION CENSAL POR RADIO
CIRCUITOS ELECTORALES
COMUNAS
DISTRITOS ECONOMICOS
BARRIOS
REGISTRO CIVIL
CAPILLAS
EMBAJADAS
ESTACIONES DE FERROCARRIL
ESTADIOS
BASILICAS
CONSULADOS

ENCUESTA JOVEN
ELEVACION TERRENO
IGLESIAS
PARROQUIAS
ELECCIONES 2015
NOMBRES PERMITIDOS
ELECCIONES 2013
DISTRITOS ESCOLARES
SECCIONES POLICIALES
MURALES
PLANCHETAS CODIGO DE PLANEAMIENTO URBANO
RELEVAMIENTO USOS DEL SUELO
PRESUPUESTO SANCIONADO
DEPENDENCIAS CULTURALES
CONTROLADORES DE SEMAFOROS
RELOJES SOLARES
TRANSFORMADORES PCB
PREMETRO
TEATRO COLON: PROGRAMACION ACTUAL
TEATRO COLON: PROGRAMACION HISTORICA
AREAS HOSPITALARIAS
COLECTIVOS
OFICINAS DE EMPLEO
DECLARACIONES JURADAS
SECCIONES
SUMIDEROS
CODIGO DE PLANEAMIENTO URBANO
FERIAS Y MERCADOS
HOSPITALES
MANZANAS
ACCESO A LA INFORMACION PUBLICA
FUENTES
INSPECTORES DE TRABAJO, INDUSTRIA Y COMERCIO
ZONAS DE RECOLECCION DE RESIDUOS
PORTALES INCLUSIVOS
UNIDADES TERRITORIALES DE INCLUSION URBANA
REGISTRO DE ADMINISTRADORES DE CONSORCIO
EMPRESAS DE FUMIGACION Y DESRATIZACION
CONSULTORES Y PROFESIONALES EN AUDITORIAS Y ESTUDIOS AMBIENTALES
INSCRIPCION ESCOLAR
SENDEROS SEGUROS
CAMPANAS VERDES
CENTROS DE ATENCION PRIMARIA (CESAC)
EMPRESAS AUDIOVISUALES
EMPRESAS DISTRITO DE LAS ARTES

EMPRESAS DISTRITO DEL DISEÑO
EMPRESAS DISTRITO TECNOLOGICO
CENTROS DE SALUD PRIVADOS
SITIOS PASIBLES DE ANEGAMIENTO
INTERPRETES DE LENGUAJE DE SEÑAS
EMPRESAS DE LIMPIEZA DE TANQUES DE AGUA
CICLOVIAS
CATALOGO CENTRALIZADO DE LA RED DE BIBLIOTECAS PUBLICAS
REDES SOCIALES
ORGANIZACIONES SOCIALES
GERIATRICOS
MEDIOS VECINALES
OPERATIVOS DE DEFENSA CIVIL
BIBLIOTECAS
FARMACIAS
TERRENOS: VALOR DE OFERTA
BOLETO ESTUDIANTIL
INSPECTORES Y VERIFICADORES DE LA AGC
DEPARTAMENTOS EN VENTA
EDITORIALES INDEPENDIENTES
SET FILMACIONES
ORGANIZADORES DE EXPOSICIONES
ESTACIONES SALUDABLES
INSPECTORES APRA
ESTACIONAMIENTOS CONCESIONADOS DE MOVILIDAD SUSTENTABLE
POLIDEPORTIVOS
FEDERACIONES
DEPENDENCIAS DEL MINISTERIO PUBLICO FISCAL
RAMPAS ACCESIBILIDAD
CENTROS DE CLASIFICACION DE RESIDUOS
CLUBES
CALLES
CUARTELES Y DESTACAMENTOS DE BOMBEROS
ARBOLADO EN ESPACIOS VERDES
ARROYOS
BANCOS CON ACCESIBILIDAD
CAJEROS AUTOMATICOS
"ARBOLADO PUBLICO LINEAL
"
AGENCIAS DE VIAJES
ALOJAMIENTOS TURISTICOS
INTERVENCIONES PEATONALES
EMPRENDEDORES
TRANSPORTES AUTORIZADOS DE PASAJEROS (SOLO LAS FILAS FILTRADAS SE DEBEN TOMAR DE TODOS LOS ARCHIVOS)

BANDAS POR BARRIO
BICICLETAS PUBLICAS
CALIDAD DE AIRE
OBRAS REGISTRADAS
SENSORES DE TRANSITO
FABRICANTES, REPARADORES, INSTALADORES Y MANTENEDORES DE INSTALACIONES FIJAS ...
REGISTRO DE FABRICANTES, REPARADORES Y RECARGADORES DE EXTINTORES
AREAS DE PROTECCION HISTORICA
VISITAS A LA WEB DEL GCBA
CAPACITADORES DE MANIPULADORES DE ALIMENTOS
EMPRESAS DE SEGURIDAD PRIVADA HABILITADAS
INSPECCIONES A LAS OBRAS (OBRAS INSPECCIONES EN FILEZILLA)
PERSONAS BUSCADAS
VIGILADORES LOCALES BAILABLES
PRESUPUESTO EJECUTADO
SUELDO FUNCIONARIOS
BICICLETEROS EN VIA PUBLICA
ESTACIONES BICICLETAS PUBLICAS
ORGANIGRAMA
PARCELAS
LOCALES BAILABLES
OBSERVATORIO DE OBRAS URBANAS
BICICLETERIAS
COMERCIOS CON BENEFICIOS A CICLISTAS
CAMARAS FIJAS DE CONTROL VEHICULAR
SISTEMA DE ADMINISTRACION DE DOCUMENTOS ELECTRONICOS: PASE DE EXPEDIENTES
SISTEMA UNICO DE ATENCION CIUDADANA
GIMNASIO (NO SE ACTUALIZO, PORQUE SIGUE SIENDO LA MISMA BASE QUE ANTES)
REGISTRO DE GUIAS DE TURISMO
DATOS PUBLICOS
DESCARGAS DE APLICACIONES
SUBTE: VIAJES MOLINETES (EL HISTORICO NO SE SUBE AL PORTAL POR LA CANTIDAD DE CASOS QUE TIENE, SOLO SE REALIZA UNA PREVISUALIZACION PERO SI SE CARGA EN EL FILEZILLA)
TEATRO COLON: VISITAS GUIADAS
CENTRO DE EXPERIMENTACION DEL TEATRO COLON: ARCHIVO GERARDO GANDINI
MEDICION DE RADIACIONES NO IONIZANTES (RNI)
RESIDUOS PATOGENICOS
CONCESIONES
FLUJO VEHICULAR POR UNIDADES DE PEAJE AUSA (AUTOPISTAS EN FILEZILLA)
SEMAFOROS

VALUACION DE AUTOMOTORES
INSPECTORES Y VERIFICADORES DE AMBIENTE Y ESPACIO PUBLICO
BAFICI
PADRON DE ANUNCIOS EMPADRONADOS
ALTO RIESGO FISCAL
ACTUACIONES DE FISCALIZACION
MOBILIARIO URBANO
RECAUDACION IMPOSITIVA
INSPECTORES DE LA AGIP
BAFICI ANIMADO
GRANDES CONTRIBUYENTES
EJECUCION FISCAL MOROSOS Y EMBARGOS TRABADOS
FESTIVAL Y MUNDIAL DE TANGO
CIUDAD EMERGENTE
SHAKESPEARE BUENOS AIRES
VERANO EN LA CIUDAD
SEGURIDAD VIAL AUTOPISTAS AUSA
BUENOS AIRES DANZA CONTEMPORANEA
SEGUIMIENTO DE OBRAS ADJUDICADAS (SOA)
PAUTA PUBLICITARIA
AGENDA CULTURAL
BOLETIN OFICIAL
BUENOS AIRES COMPRAS (BAC)
CONTAMINACION SONORA
CORTES DE TRANSITO EN TIEMPO REAL
DISPONIBILIDAD DE BICICLETAS PUBLICAS
ESTADO DE LA AUTOPISTA
INFORMACION METEOROLOGICA
MAPA INTERACTIVO

Nomenclaturas, términos, símbolos y acrónimos

La siguiente lista es una recopilación de conceptos utilizados en el presente trabajo, necesarios para su total entendimiento:

Palabra	Definición
Amazon Web Services	Infraestructura de Amazon en la nube. Proveen servicios de cómputo, bases de datos, almacenamiento, automatización, microservicios, todo es facturado según el uso a los usuarios.
API	Es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.
Aplicación de Backend	Aplicación de procesamiento de datos y repositorio de las bases.
Aplicación de Frontend	Aplicación de acceso a los usuarios, no incluye el procesamiento de los datos o las bases.
Archivos CSV	Archivos de columnas separadas por comas.
Biblioteca de funciones	Una biblioteca de software es un conjunto de datos y código de programación que se utiliza para desarrollar programas y aplicaciones de software. Está diseñado para ayudar tanto al programador como al compilador de lenguaje de programación en la construcción y ejecución de software.
Big Data	Big data, macrodatos, datos masivos o datos a gran escala es un concepto que hace referencia a conjuntos de datos tan grandes que aplicaciones informática tradicionales del procesamiento de datos no son suficientes para tratar con ellos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos. En los textos científicos en español con frecuencia se usa directamente el término en inglés big data, tal como aparece en el ensayo de Viktor Schönberger big data: La revolución de los datos masivos. La disciplina dedicada a los datos masivos se enmarca en el sector de las tecnologías de la información y la comunicación. Esta disciplina se ocupa de todas las actividades relacionadas con los sistemas que manipulan grandes conjuntos de datos. Las dificultades más habituales vinculadas a la gestión de estas cantidades de datos se centran en la recolección y el almacenamiento, búsqueda, distribución, análisis y visualización. La tendencia a manipular enormes cantidades de datos se debe a la necesidad en muchos casos de incluir dicha información para la creación de informes estadísticos y modelos predictivos utilizados en diversas materias, como los análisis de negocio, publicitarios, los datos de enfermedades infecciosas, el espionaje y seguimiento a la población o la lucha contra el crimen organizado. En el caso de este trabajo se refiere a los

	datos provenientes de las Ciudades para poder tomar decisiones de mejora de calidad de vida de la población.
BigQuery	BigQuery es un servicio web RESTful que permite el análisis interactivo de grandes conjuntos de datos que funcionan conjuntamente con Google Storage. Es una Infraestructura como Servicio (IaaS) que se puede utilizar de forma complementaria con MapReduce.
bigrquery	Los paquetes bigrquery proporcionan una interfaz R a Google BigQuery. Facilita la recuperación de metadatos sobre sus proyectos, conjuntos de datos, tablas y trabajos, y proporciona un conveniente contenedor para trabajar con bigquery de R.
Cloud	Cloud es un término que viene de los años 60 cuando los ingenieros que diseñaban redes informáticas utilizaban diagramas en forma de nubes para graficar que la información viajaba de un lugar a otro por la red. Se puede decir en forma sencilla que el Cloud Computing consiste en que los recursos de información, bases de datos, aplicaciones, están almacenados en storages de terceros, que brindan sus servicios a través de internet a grandes empresas y pequeños usuarios, ofreciéndoles capacidad de computo y soporte de operación según demanda en tiempo real. Como concepto más tecnológico tenemos el que enuncia el NIST (National Institute of Standards and Technology, del departamento de comercio de EEUU): “El Cloud Computing es un modelo para proporcionar, bajo demanda, acceso de red a una reserva configurable y compartida de recursos informáticos (redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser provisionados rápidamente y liberados con un esfuerzo mínimo por parte del proveedor de servicios.
Colas de procesamiento de mensajes	En ciencias de la computación, las colas de mensajes y los buzones son componentes de ingeniería de software utilizados para la comunicación entre procesos (IPC) o para la comunicación entre hilos dentro del mismo proceso. Utilizan una cola para la mensajería - el paso del control o del contenido. Los sistemas de comunicación de grupo proporcionan tipos similares de funcionalidad.
Crowd Sentiments	Análisis sistemático de las opiniones y actitudes de una multitud, a través de información recolectada.
Crowdsensing	Es una técnica en la que un gran grupo de individuos que tienen dispositivos móviles capaces de detectar y computar (tales como teléfonos inteligentes, tabletas, portátiles) comparten datos colectivamente y extraen información para medir, mapear, analizar, estimar o inferir cualquier proceso de interesar. En resumen, esto significa crowdsourcing de datos de sensores de dispositivos móviles.
Google Cloud Platform	Google Cloud (Nube de Google), es una plataforma que ha reunido

	todas las aplicaciones de desarrollo web que Google estaba ofreciendo por separado. Es utilizada para crear ciertos tipos de soluciones a través de la tecnología almacenada en la nube y permite por ejemplo destacar la rapidez y la escalabilidad de su infraestructura en las aplicaciones del buscador.
Insight	Deducción o idea útil revelada luego de un estudio exhaustivo de un tema.
Microsoft Azure	Es un servicio en la nube ofrecida como servicio y alojado en los Data Centers de Microsoft. Anunciada en el Professional Developers Conference de Microsoft (PDC) del 2008 en su versión beta, pasó a ser un producto comercial el 1 de enero de 2010. Windows Azure es una plataforma general que tiene diferentes servicios para aplicaciones, desde servicios que alojan aplicaciones en alguno de los centros de procesamiento de datos de Microsoft para que se ejecute sobre su infraestructura (Cloud Computing) hasta servicios de comunicación segura y federación entre aplicaciones.
Mobile Sensing	Las aplicaciones tradicionales de detección móvil utilizan equipos adicionales que no son realistas para la mayoría de los usuarios. Los teléfonos inteligentes se desarrollan a una velocidad rápida en los últimos años, y se están convirtiendo en indispensable llevar de la vida cotidiana. Los sensores incorporados en ellos proporcionan varias posibilidades para aplicaciones móviles, y estas aplicaciones están ayudando y cambiando la forma de nuestra vida.
OAuth 2.0	OAuth (Open Authorization) es un protocolo que permite flujos simples de autorización para sitios web o aplicaciones informáticas. Se trata de un protocolo propuesto por Blaine Cook y Chris Messina, que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y web. OAuth permite a un usuario del sitio A compartir su información en el sitio A (proveedor de servicio) con el sitio B (llamado consumidor) sin compartir toda su identidad. Para desarrolladores de consumidores, OAuth es un método de interactuar con datos protegidos y publicarlos. Para desarrolladores de proveedores de servicio, OAuth proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta. En este trabajo se utilizó para simplificar el uso del prototipo.
Pip	Es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.
Procesamiento en tiempo real	
Procesamiento por lotes o batch	

Pub/Sub	Google Cloud Pub/Sub aporta la escalabilidad, flexibilidad y fiabilidad del middleware orientado a mensajes empresariales a la nube. Al proporcionar mensajería asíncrona de muchos a muchos, que desacopla a los remitentes y receptores, permite una comunicación segura y altamente disponible entre aplicaciones escritas independientemente. Google Cloud Pub/Sub ofrece mensajes de baja latencia y duraderos que ayudan a los desarrolladores a integrar rápidamente sistemas alojados en Google Cloud Platform y externamente.
Python	Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
R	R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.
SaaS	Software as a Service, representa la oferta de software que se cobra por uso de la infraestructura.
Sensingkit	Biblioteca de funciones elegida para instalar en teléfonos móviles que toma información de los distintos sensores que estos poseen. Es un Marco de Detección Móvil Multi-Plataforma para Experimentos a Gran Escala.
Sentiment Analysis	El proceso de identificar y categorizar las opiniones expresadas en un texto, especialmente para determinar si la actitud del escritor respecto a un tema, producto, etc. en particular es positiva, negativa o neutral.
Softlayer	SoftLayer Technologies, Inc. es un servidor dedicado, hosting administrado y proveedor de cloud computing. Fundada en 2005, fue adquirida por IBM en 2013.
Subscripción/Subscription	El suscriptor recibe mensajes pendientes de su suscripción y reconoce cada uno al servicio Pub/Sub.
Tema/Topic	Una aplicación de editor crea un tema en el servicio Google Cloud Pub/Sub y envía mensajes al tema. Un mensaje contiene una carga útil y atributos opcionales que describen el contenido de la carga útil.

Bibliografía

Tipo	Título	Autor	Año, Editorial
Artículo	The Top Five Smart Cities In The World	Peter High	Forbes, 2017
Figura	http://www.transportationtechnologyventures.com/blog/2012/09/new-sensors-new-application-domains/		2017
Informe	Redshift vs BigQuery - Closing the Gaps in Data Warehousing	Panoply.io	2017
Libro	Amazon Virtual Private Cloud	User Guide	2016
Libro	Amazon Web Services For Dummies	Bernard Golden	Agosto, 2013
Libro	Smart Cities: Big Data, Civic Hackers, and the Quest for a New Utopia	Anthony Townsend	2013
Paper	A Cognitive Oriented Framework for IoT Big-data Management Prospective	Nilamadhab Mishra	Diciembre, 2014
Paper	http://journals.sagepub.com/doi/full/10.1155/2013/272916	Ming Liu	2013
Paper	Evaluation of Smartphone Inertial Sensor Performance for Cross-Platform Mobile Applications	Anton Kos, Sašo Tomažič, Anton Umek	2016
Paper	Optimizing Cloud-Based Video Crowdsensing	Hua-Jun Hong	2016
Paper	Perpetual video camera for Internet-of-things	Yen-Kuang Chen	2012
Paper	Scalable Cloud–Sensor Architecture for the Internet of Things	Yi Xu	2016
Paper	Video Streaming Considerations for Internet of Things	Rubem Pereira	Agosto, 2014
Paper	Crowdsensing Maps of On-street Parking Spaces	Vladimir Coric Dept. of Comput. & Inf. Sci., Temple Univ., Philadelphia, PA, USA Marco Gruteser WINLAB, Rutgers Univ., North	2013

		Brunswick, NJ, USA	
Paper	Congestion-Aware Communication Paradigm for Sustainable Dense Mobile Crowdsensing	Wen Sun, Jiajia Liu	2017
Paper	CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments	Claudio Fiandrino; Andrea Capponi; Giuseppe Cacciatore; Dzmitry Kliazovich; Ulrich Sorger; Pascal Bouvry; Burak Kantarci; Fabrizio Granelli; Stefano Giordano	2017
Paper	Internet of Things (IoT): A vision, architectural elements, and future directions	Jayavardhana Gubbi a, Rajkumar Buyya b, *, Slaven Marusic a, Marimuthu Palaniswami a	2013
Paper	Mobile and Social Sensing for real-time problems	Dr. Laurissa Tokarchuk	2015
Paper	Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture	Evangelos A. Kosmatos, Nikolaos D. Tselikas, Anthony C. Boucouvalas	2011
Sitio Web	http://machinelearningmastery.com/java-machine-learning/	Sitio	Julio, 2014
Sitio Web	http://mcelectronics.com.ar/site/	Sitio	2016
Sitio Web	http://raheelretiwalla.com/home/2016/3/2/comparing-a	Comparación IoT	Febrero 2017
Sitio Web	http://web.mit.edu/course/21/21.guide/th-form.htm	Contenido de una tesis	2017

Sitio Web	http://www.businessinsider.com/smartphone-adoption-platform-and-vendor-trends-in-major-mobile-markets-around-world-2015-3	Smparphone adoption	2017
Sitio Web	http://www.germanna.edu/documents/formatting-academic-papers-in-google-docs.pdf	Formato de paper académico en Google Docs	2016
Sitio Web	https://bigquery.cloud.google.com/results/smartcities-149023:bquijob_58eb2c68_159e72567c6	Google SmartCities - NY stats	2017
Sitio Web	https://cloud.google.com/bigquery/docs/reference/legacy-sql	BigQuery Reference	2017
Sitio Web	https://cloud.google.com/solutions/iot-overview	IoT Google	Febrero 2017
Sitio Web	https://cran.r-project.org/web/packages/bigrquery/README.html	Bibliotecas Bigrquery	2016
Sitio Web	https://dspace.mit.edu/handle/1721.1/45277#files-area	Tesis del MIT	2017
Sitio Web	https://labs.spotify.com/2016/02/25/spotifys-event-delivery-the-road-to-the-cloud-part-i/	Spotify Event Delivery	Febrero, 2016
Sitio Web	https://northconcepts.com/birdiq/twitter-search/?q=%23messi	Bajar datos de Twitter por hashtag	Febrero 2017
Sitio Web	https://phone-lab.org/	Crowdsensing	2017
Sitio Web	https://plus.google.com/u/1/+Datasensinglab	Design Lab Group	Febrero, 2017
Sitio Web	https://www.r-project.org/	Statistics Software	2016
Sitio Web	https://www.youtube.com/watch?v=6Nv18xmJirs	Pushing Limits of Big Query	2016
Sitio Web	Qualcomm Smart Cities	Sitio	2016
Sitio Web	www.zigbee.org	Sitio	2016
Sitio Web	https://plus.google.com/u/0/collection/MX98P?cfem=1	Data Display	2017
Software	https://makingscience.withgoogle.com/science-journal?lang=en	Google Science Journal	2017
Software	https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap	Google Heatmap Tool	2017
Tesis	Repensando el rol del “Cloud Computing” en el	Edgard Luis	2014

	Negocio de la TELCO: estrategias para un nuevo escenario.	Dapas	
Video	https://www.youtube.com/watch?v=JuaBy3e6fd4	Google I/O	2016
Video	https://www.youtube.com/watch?v=kdmAiQeYGgE	Streaming Data	Septiembre, 2016, Google Developers
Video	https://www.youtube.com/watch?v=Rnm83GqgqPE	Google - IoT Implementation	2013
Video	https://www.youtube.com/watch?v=tqS4vZ2Rxlo	BigQuery Correlation	Septiembre, 2013, Google Developers
White Paper	An Inside Look at Google BigQuery	Google	2012