Diseño del DAaaS

Definición la estrategia del DAaaS

* La estrategia consiste en ofrecer un módulo a los sistemas de conducción autónoma para que sean capaces, a la hora de seleccionar su ruta de viaje, de predecir y reaccionar a diferentes eventos del tráfico en tiempo real mediante el aprendizaje autónomo y escoger su ruta dependiendo de los factores introducidos.

Arquitectura DAaaS

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

***Entrada de datos al sistema***

* APIRest DGT (Información de sensores de tráfico)
* Servicio MQTT DGT para disponibilidad de aparcamiento y otros eventos (se podría utilizar algún tipo de gestor como CloudMQTT)

***Servicios de Google Cloud***

* Cloud *functions* para refrescar los datos de los sensores cada minuto o cuando se dispongan nuevos datos como respuesta MQTT (activar cuando llegan nuevos datos que procesar a la cola de Pub/Sub) También se utilizarán para enviar los datos de Pub/Sub a Google Cloud Storage

***Máquinas virtuales***

* Tienen instalado MongoDB, Apache Spark y PySpark

***Plataformas de Google Cloud***

* Google Cloud Pub/Sub será la encargada de recibir la información de los sensores. Se decide utilizar Pub/Sub ya que está preparado para trabajar con datos en tiempo real, o grandes volúmenes de datos en continuo. Estos datos los publicará en un *topic* específico accesible desde los servicios que empleen esta información.
* Apache Spark en Dataproc procesa los datos de manera distribuida en la nube recibiendo datos de las *cloud fucntions* definidas, en su salida, genera información útil como estadísticas o visualizaciones, para ello, utilizará varios scripts en PySpark con una serie de normas que quedarán integradas en el flujo automatizado de trabajo.
* MongoDB recibe los datos de ApacheSpark y se utiliza como base de datos, se decide emplear MongoDB por la flexibilidad en cuanto a recibir datos semiestructurados como lo podrían ser los de algunos Json.

***Frameworks y herramientas***

* Debería de conectarse un *framework* como Tensorflow a la base de datos de MongoDB para entrenar los modelos de predicción en base a los datos recopilados.
* Flask, servirá para la visualización final en el navegador interactuando como capa intermedia entre la base de datos y MongoDB

***Interfaz de usuario***

* Los datos extraídos de las bases de datos y los sistemas de aprendizaje se integrarán en la interfaz del sistema de navegación del automóvil que utilizará un sistema de API REST propia.

DAaaS Operating Model Design and Rollout

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

* Se realiza una petición a la API de la DGT aproximadamente cada minuto utilizando *cloud functions* como “cronjob”
* En Google *cloud* Pub/Sub los datos que llegan a través de MQTT se envían a una cola que disparará un *trigger* de actualización cuando tengamos nuevos elementos en cola como actualización de aparcamientos ocupados o accidentes de tráfico, controles etc.
* Del mismo modo cuando todo esto suceda, *cloud functions* carga estos datos a la base de datos de MongoDB.
* MongoDB utiliza un conector para comunicarse con Apache Spark realizar las transformaciones o visualizaciones necesarias mediante los algoritmos o *string* de códigos necesarios e implementados con PySpark. Una vez procesados, los datos van de nuevo a la base de datos de MongoDB. Para generar este conector en la máquina virtual se utilizará un archivo “jar” en Dataprocs,
* Las actualizaciones de MongoDB pueden enviar un mensaje al sistema de Pub/Sub con otro topic. Esta cola puede actuar como intermediaría con el pipeline de los datos, estos mensajes serían procesados por *Dataflow* con un código de *PySpark* especifico que utilice la librería de *Tensorflow*.
* Se recurriría a otra *cloud function* como *webhook* que se activará cada vez que se entrene el modelo predictivo y adquiera o actualice nuevos datos, estos datos actualizarán la base de datos del sistema.
* Cuando se introduzca la ruta en el automóvil, este utilizará una *APIRest* integrada para solicitar toda esta información del estado actual de las carreteras y los aparcamientos junto a los modelos predictivos que podrían enriquecerse con otros eventos como cambios en el clima o la calidad del aire (mediante otras APIS distintas que se llamarían en el proceso de extracción de datos) Estas solicitudes de la API Rest del navegador comunicarán con flask que actuará de intermediario con la base de datos para recibir y entregar los paquetes de datos necesarios a la interfaz del usuario.

Desarrollo de la plataforma DAaaS. (ligera descripción del desarrollo)

*DGT 3.0 -> https://www.dgt.es/muevete-con-seguridad/tecnologia-e-innovacion-en-carretera/forma-parte-de-la-dgt-3.0/*