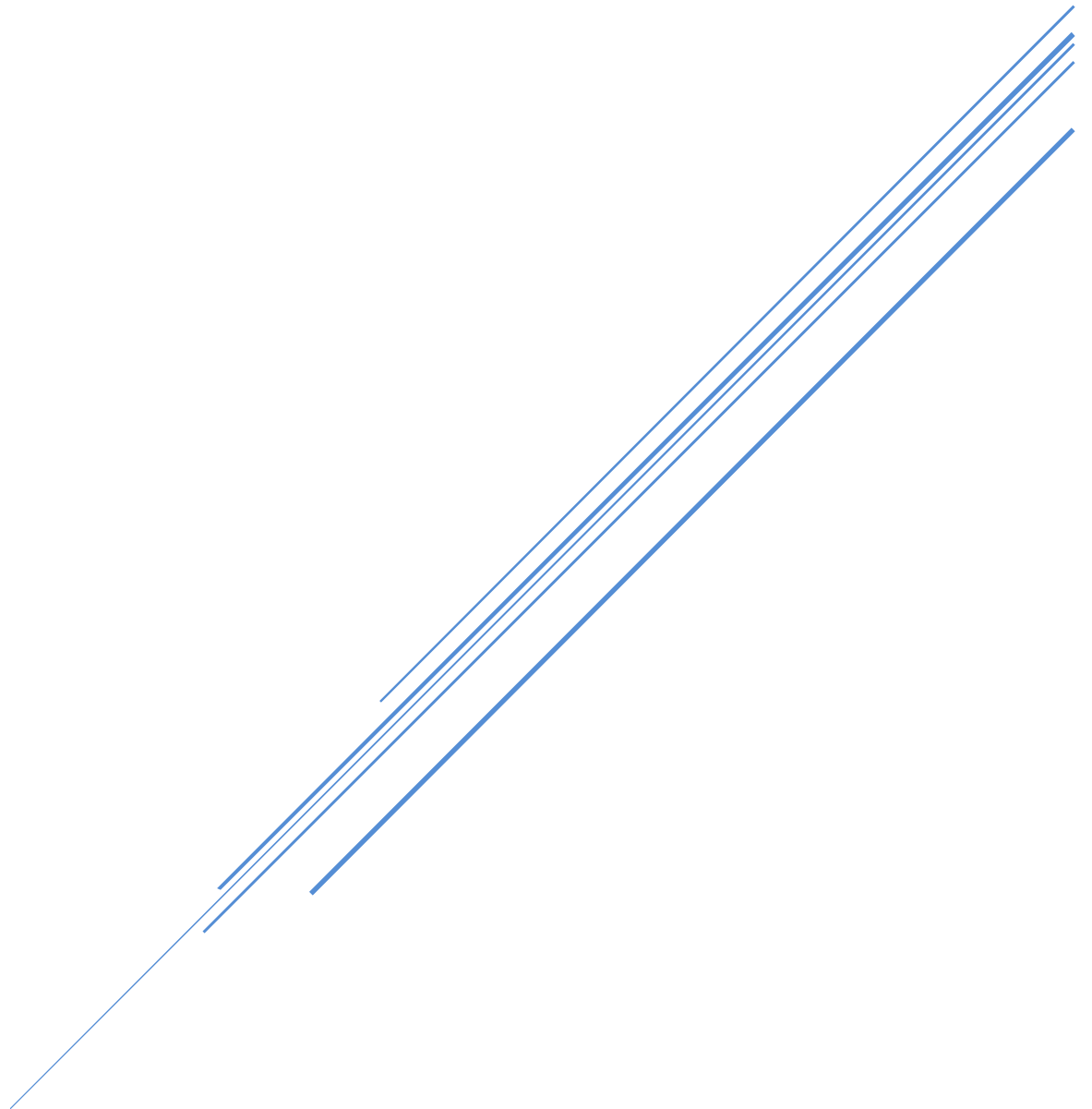


# PLATAFORMA COVID-19

Datacenters



La Salle URL – 2020/21  
Clàudia Aymerich – Víctor Juez – Arturo Menchaca

## Problemática

Nos encontramos en la segunda ola del Covid-19, la gran pandemia mundial de este 2020. Durante estos meses, igual que muchos hospitales, hemos estado estudiando a la gente que se infecta para poder encontrar un patrón.

Obviamente, infectarse o no depende de cómo esté el paciente físicamente, pero también hemos podido observar que depende mucho del estilo de vida que tengan los pacientes en su día a día.

La gente cuando sale a la calle va con mascarilla, y con cuidado, pero ¿qué es lo que pasa una vez estamos con un grupo de gente conocido? La gente se relaja, se saca la mascarilla, comparte comida, entre otras acciones que sabemos que no son correctas, pero por el simple hecho de estar relajados se olvidan de las transmisiones.

Es por eso por lo que hemos empezado este proyecto, en el que queremos concienciar al usuario de lo expuesto que está a contraer el virus en su propio entorno, y así, disminuir los casos de Covid lo máximo posible.



## Solución

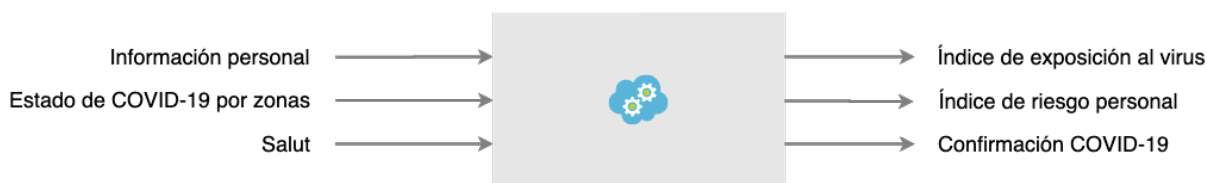
Como grupo de investigación del departamento de Salud de Catalunya, hemos planteado la siguiente propuesta.

Consiste en tener una aplicación móvil, y un conjunto de datos externos que conjuntamente conseguirán observar cual es el riesgo que tiene cada usuario según sus contactos y su día a día. Además, tendremos notificaciones instantáneas sobre los resultados o pasos que debe tener cada usuario según la etapa del PCR en la que se encuentre.

## ¿Qué necesitamos?

Para poder brindar la información detallada al siguiente apartado, es necesario tener ciertos datos acerca del usuario, las pruebas realizadas y la situación Covid por región. Estos se obtendrán de distintas fuentes como puede ser la propia aplicación, departamento de salud o APIs públicas. Cada entrada se detalla a continuación:

- **Información Personal:** A través de la aplicación el usuario proporciona su información personal, entre los que se encuentra lugar de residencia, su estilo de vida, edad, hábitos y patologías. Además, puede seleccionar los usuarios interesados para tener en cuenta por la aplicación.
- **Covid Por Zonas:** Información acerca del estado actual del Covid por zonas. Esta información se puede obtener de distintas APIs públicas como puede ser <https://covid19tracking.narrativa.com>.
- **Salud:** El departamento de Salud proporciona información del usuario como son los resultados de test, estado de cuarentenas o síntomas que presente.



## ¿Qué datos nos aporta la aplicación?

La aplicación aporta un conjunto de datos para tener informado al usuario en todo momento de su situación actual con respecto al Covid. Esos datos se ven reflejados en 3 medidas:

- **Confirmación COVID:** Ya que se tiene contacto directo con los servicios de salud, se puede informar al usuario instantáneamente de los resultados de test que se haya realizado. Sin tener que esperar una llamada o un SMS, recibe una notificación a la aplicación con el resultado del test.
- **Índice de Riesgo Personal:** Valor que indica en qué medida nos puede afectar físicamente el virus teniendo en cuenta la información que ha proporcionado el usuario al registrarse en la aplicación.
- **Índice de Exposición al Virus:** Esta medida indica que tan expuestos estamos al virus dependiendo del estilo de vida y de cómo influye en el usuario el comportamiento de sus contactos y su relación con ellos.

# Algoritmos

La aplicación realiza una serie de algoritmos para calcular la información que brinda a partir de los datos que se tienen como entrada. Estos algoritmos se centran en dos tareas: el índice de riesgo personal y el índice de exposición al virus.

- **Índice de Riesgo Personal:** Este se calcula a partir de la información personal que ha proporcionado el usuario, su condición física, patologías, edad, etc. Con estos datos se compara con los estudios realizados acerca de la enfermedad junto con el conocimiento que se ha adquirido mediante el mismo uso de la aplicación, y se le asigna un nivel de riesgo al usuario.

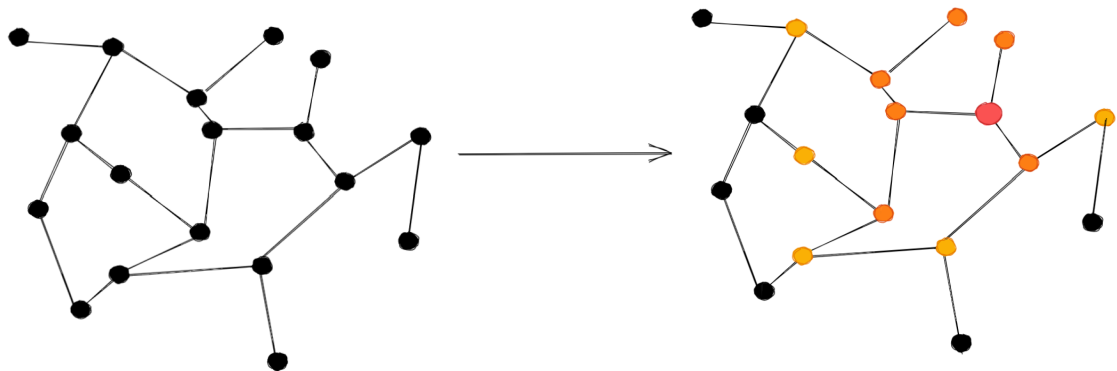


¿Cuáles son los grupos vulnerables ante el COVID19?

◆ mayores de 60 años

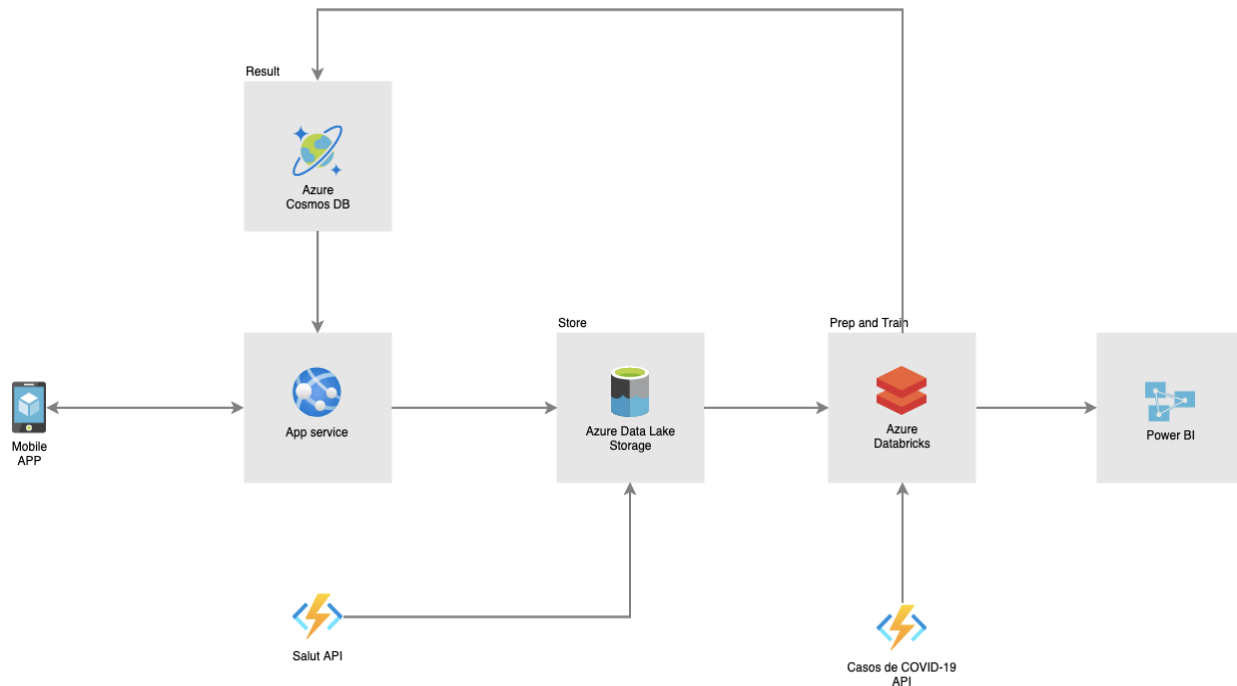
◆ personas diagnosticadas de hipertensión arterial, diabetes, enfermedades cardiovasculares o pulmonares crónicas, cáncer, inmunodeficiencias y embarazadas, por precaución

- **Índice de Exposición al Virus:** Se calcula teniendo en cuenta el estilo de vida de la persona y su interacción con sus contactos. A través de los contactos de los usuarios de la aplicación, se crea un grafo de usuarios donde se reflejan las conexiones personales. Durante el transcurso del tiempo, se mide cómo se comporta el virus y su propagación entre los distintos nodos del grafo, a partir de esta información se tiene un modelo para poder predecir qué tan probable un usuario puede infectarse dependiendo de qué tan cerca está el virus a través de sus contactos.



# Arquitectura

A continuación, vamos a hablar de la arquitectura de nuestro proyecto. Hemos apostado en construir una arquitectura Cloud, donde con un conjunto de servicios podemos conseguir una mejor experiencia.



- **Azure Functions:** Tiene como objetivo la ingesta y transformación de los datos de Salud con la información y los casos actuales del Covid.
- **Azure Data Lake Storage:** En este punto guardamos algunos de los datos que nos entran con la finalidad de retener los hasta que se ejecute el algoritmo.
- **Azure Databricks:** En este punto es donde se crea la magia de la plataforma. Cada ejecución de spark va a buscar la nueva información a Data Lake Storage para ejecutar el algoritmo que se ha comentado anteriormente.
- **Azure Cosmos DB:** El último paso de la transformación de datos es guardarlos. Para eso hemos creado una base de datos NoSQL donde se guardarán para cada usuario su información que se querrá mostrar a la aplicación.
- **App Service:** Nuestra aplicación estará desplegada en la App Service de Azure. Desde ahí podremos recibir la información del usuario y todos sus contactos para el algoritmo, y recoger la información del Cosmos DB para poder mostrar los resultados a los usuarios.
- **Power BI:** Finalmente, tenemos el servicio Power BI que con todos los datos históricos guardados en el Data Lake puede mostrar un conjunto de gráficos para tener una idea general de cómo está la situación del Covid y para tener algunos apuntes de mejora de la aplicación.

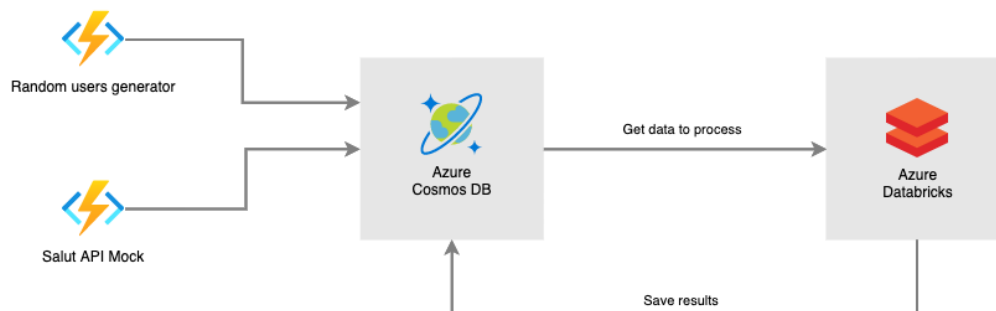
## Prueba de concepto (PoC)

Hemos realizado una prueba de concepto en Azure para ver de primera mano cómo sería implementar el proyecto EndToEnd.

- Consulta el resultado de la última ejecución en el siguiente enlace: <https://covidexposure.z6.web.core.windows.net/>
- También puedes consultar el código fuente en este repositorio de GitHub: <https://github.com/VictorJuez/PoC-covid-exposure>.

A continuación, la arquitectura del PoC

Visual Paradigm Online Express Edition



Visual Paradigm Online Express Edition

- **Function - Random users generator:** Ejecutado una única vez al deployar el proyecto, genera un conjunto de usuarios aleatorios con la información que podemos observar abajo y los guarda en la instancia de Azure Cosmos DB

```
{
  "id": "id del usuario",
  "contacts": "lista de ids de los contactos del usuario",
  "location": "Provincia de España donde reside el usuario"
}
```

- **Function - Salut API Mock:** Programada su ejecución de forma periódica cada media hora. Escoge algunos de los usuarios presentes en la base de datos y computa de forma aleatoria si han dado positivo o no a un test de COVID. Abajo la información generada y guardada en Azure Cosmos DB.

```
{
  "id": "id del usuario",
  "covidTest": "Resultado positivo/negativo de covid"
}
```

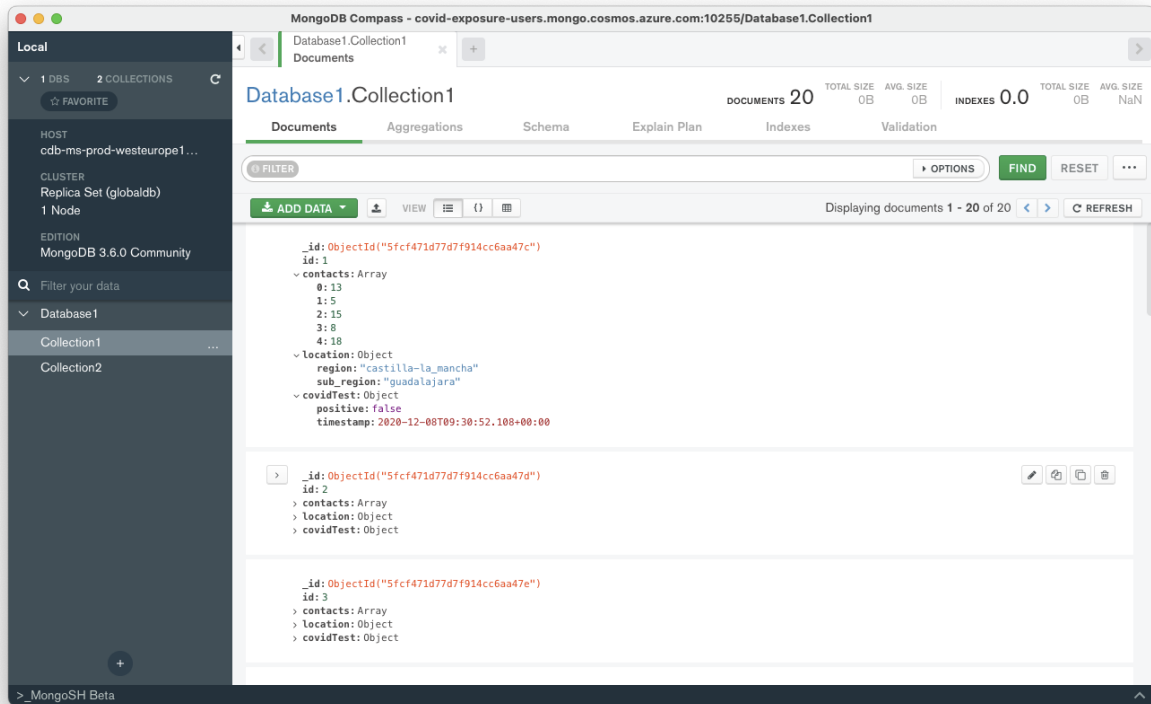
- **Azure Databricks:** Coge la información de los usuarios y por cada uno computa cuántos de sus contactos están contagiados. El resultado es guardado de vuelta a la

misma instancia de Azure Cosmos DB.

- **Azure Cosmos DB:** Base de datos no relacional (MongoDB). Consta de dos Collections:
  - **users:** contiene la información de los usuarios y de los tests covid
  - **covid-exposure:** contiene los resultados del algoritmo, es decir, cuántos de los contactos de un usuario están contagiados

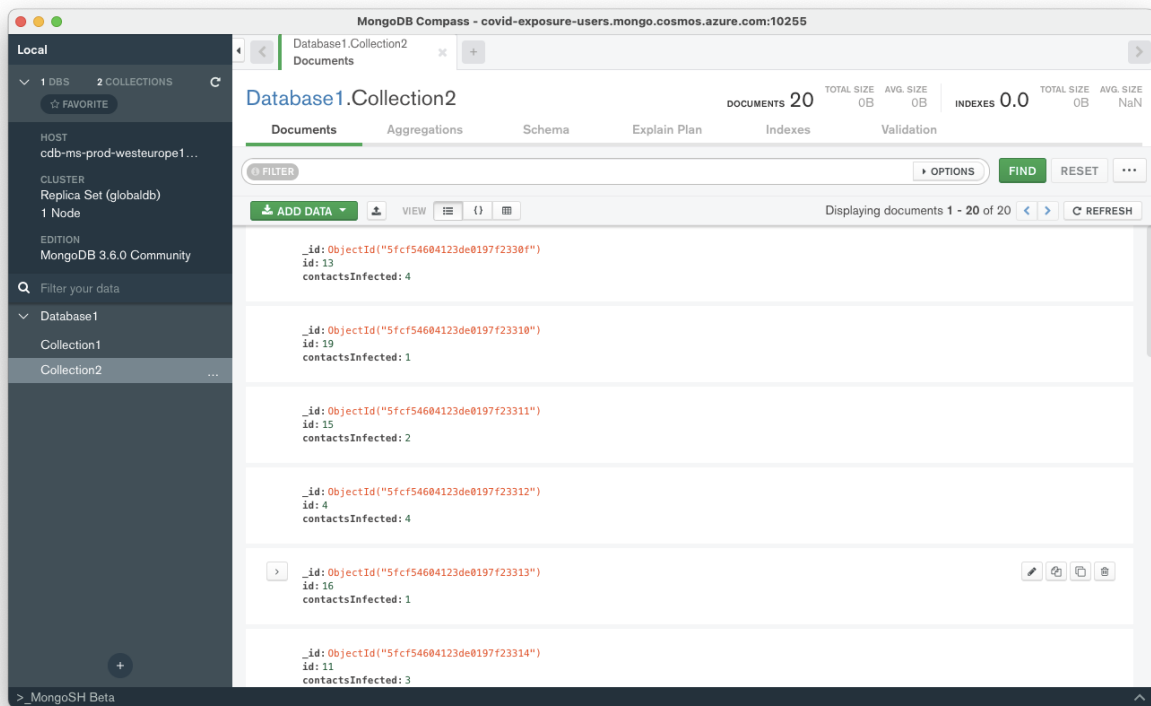
# Anexo

## Azure Cosmos DB



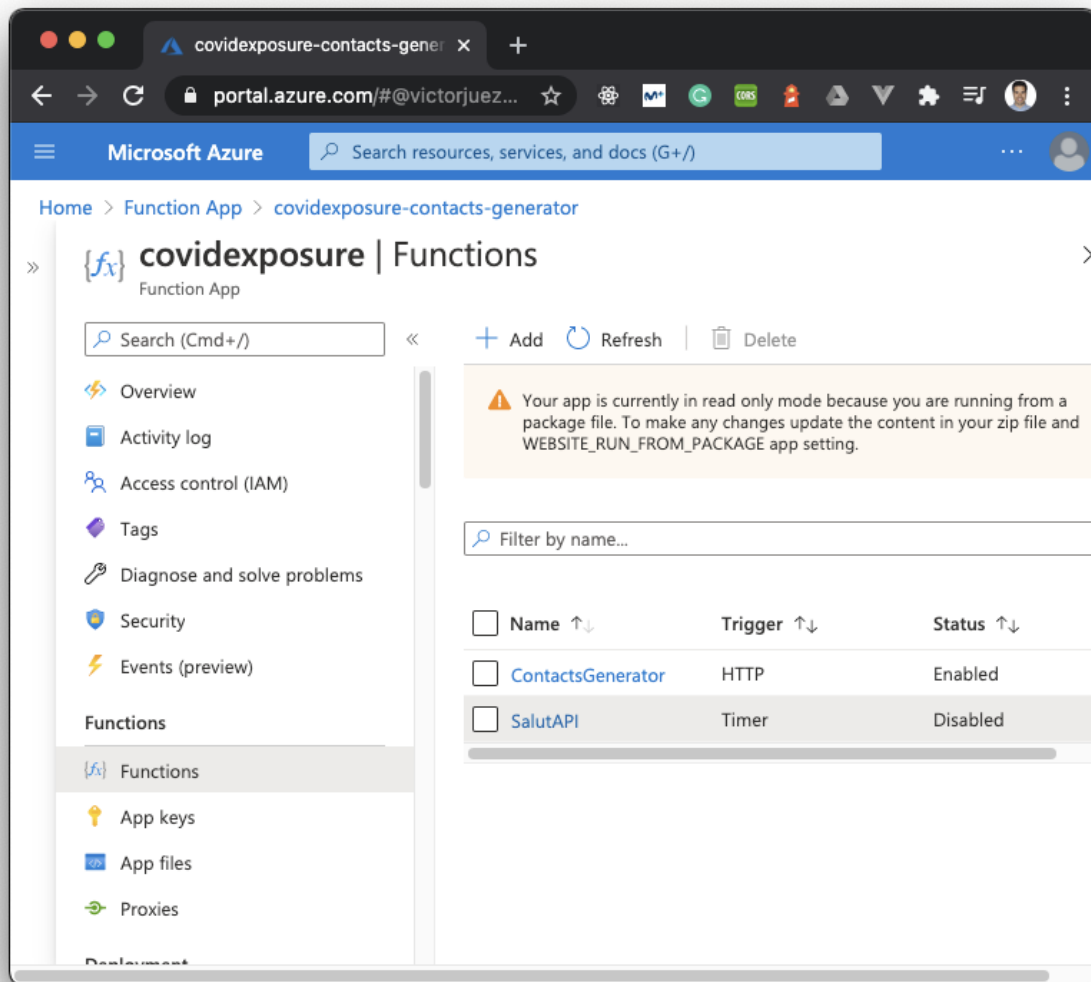
## Users collection





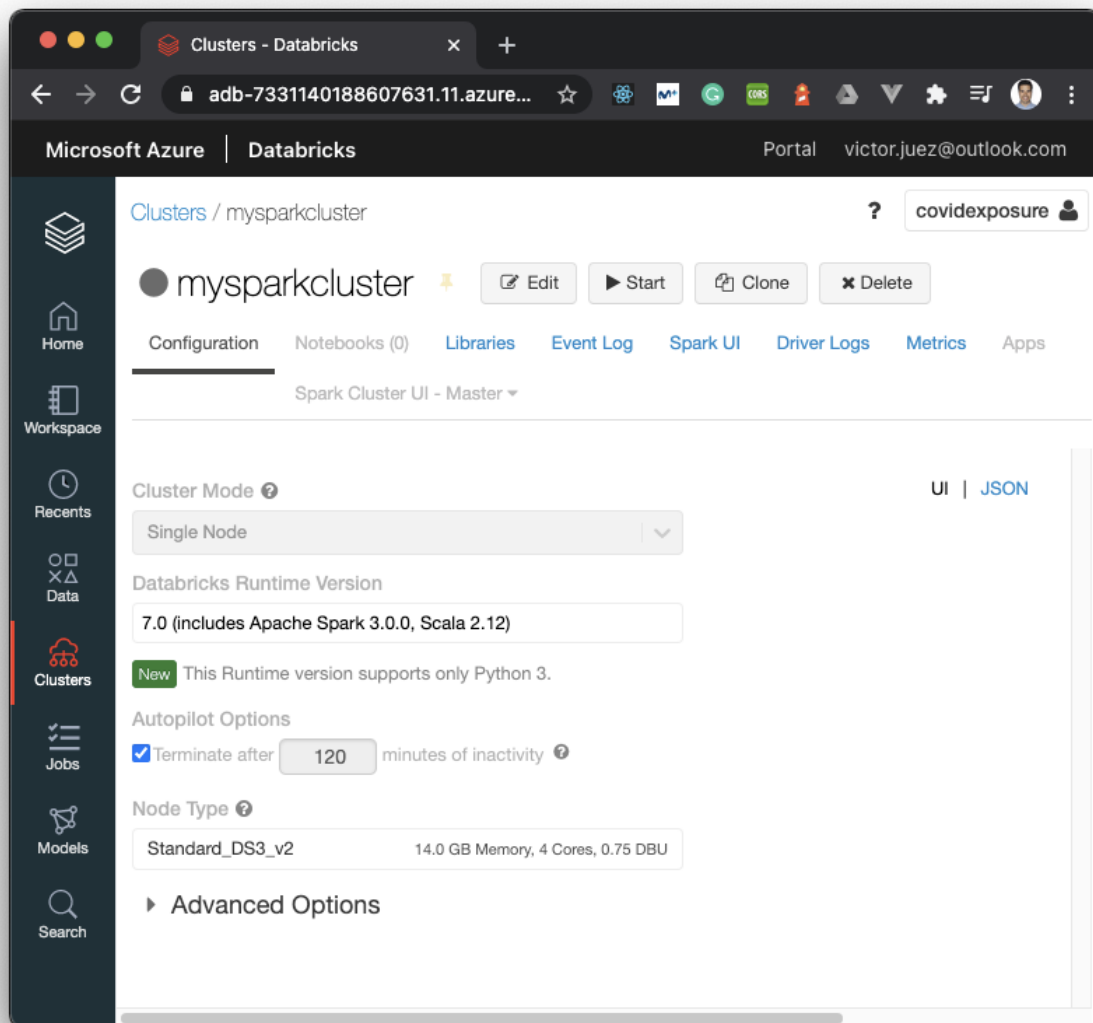
Covid Exposure Collection

# Functions

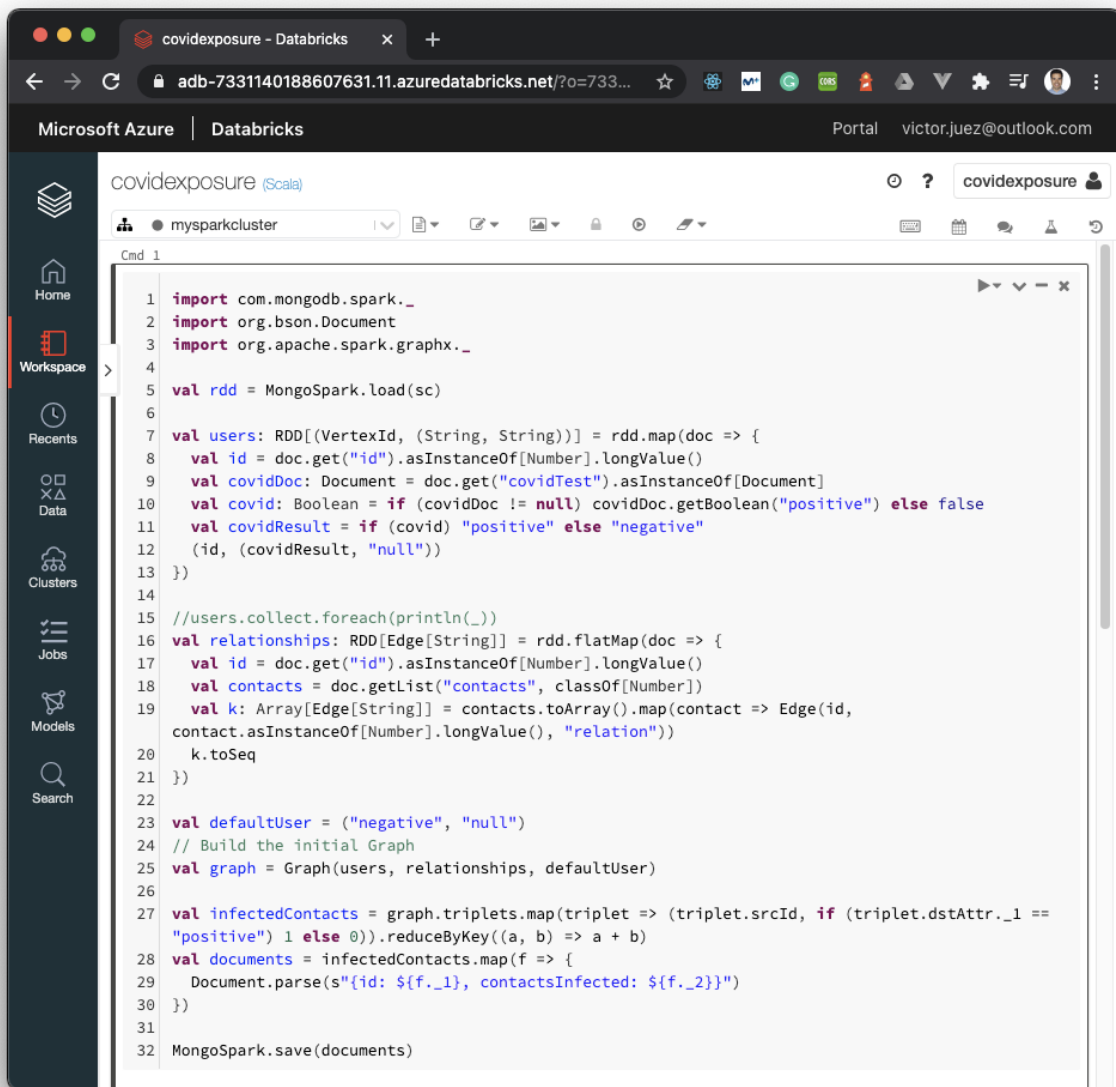


Random users generator & Salut API Mock

# Databricks



## Databricks Cluster



The screenshot shows a Databricks notebook interface with the following components:

- Header:** Microsoft Azure | Databricks. Portal victor.juez@outlook.com
- Left Sidebar:** Home, Workspace, Recents, Data, Clusters, Jobs, Models, Search.
- Notebook Title:** covidexposure (Scala)
- Cluster:** mysparkcluster
- Code Editor:** Contains Scala code for processing COVID exposure data.

```
1 import com.mongodb.spark._
2 import org.bson.Document
3 import org.apache.spark.graphx._
4
5 val rdd = MongoSpark.load(sc)
6
7 val users: RDD[(VertexId, (String, String))] = rdd.map(doc => {
8   val id = doc.get("id").asInstanceOf[Number].longValue()
9   val covidDoc: Document = doc.get("covidTest").asInstanceOf[Document]
10  val covid: Boolean = if (covidDoc != null) covidDoc.getBoolean("positive") else false
11  val covidResult = if (covid) "positive" else "negative"
12  (id, (covidResult, "null"))
13 })
14
15 //users.collect.foreach(println(_))
16 val relationships: RDD[Edge[String]] = rdd.flatMap(doc => {
17   val id = doc.get("id").asInstanceOf[Number].longValue()
18   val contacts = doc.getList("contacts", classOf[Number])
19   val k: Array[Edge[String]] = contacts.toArray().map(contact => Edge(id,
20     contact.asInstanceOf[Number].longValue(), "relation"))
21   k.toSeq
22 })
23
24 val defaultUser = ("negative", "null")
25 // Build the initial Graph
26 val graph = Graph(users, relationships, defaultUser)
27
28 val infectedContacts = graph.triplets.map(triplet => (triplet.srcId, if (triplet.dstAttr._1 ==
29   "positive") 1 else 0)).reduceByKey((a, b) => a + b)
30 val documents = infectedContacts.map(f => {
31   Document.parse(s"{id: ${f._1}, contactsInfected: ${f._2}}")
32 })
33
34 MongoSpark.save(documents)
```

Databricks notebook