



Asistiendo al monitoreo de datos médicos con almacenamiento no relacional

Ingeniería Civil Informática

Maximiliano Tomás Navarrete Ulloa

11/12/2023

Tabla de Contenidos

1.1 Resumen Ejecutivo	3
1.2. Abstract.....	4
2. Introducción.....	5
3. Objetivos.....	7
4. Estado del Arte	8
5. Propuestas de Solución	9
6. Evaluación Económica	11
7. Medidas de Desempeño	13
8. Metodología y Desarrollo	13
9. Matriz de Riesgos	15
10. Resultados Finales	15
11. Conclusiones	17
12. Bibliografía.....	18
13. Anexo	21

1.1 Resumen Ejecutivo

El presente informe detalla, de forma concisa, la experiencia, problemática y aplicación de una solución a un problema ingenieril realizadas durante el proceso de pasantía full-time.

La empresa Blueng Ingenieros Spa. fue contratada por YoMeControlo y Avante para crear dos plataformas web de medicina preventiva. Se encontró que Avante en particular deseaba manejar sus datos médicos históricos para visualizar datos específicos de sus miembros y generar algoritmos predictivos desde estos. Sin embargo, se tenía una accesibilidad muy limitada a su base de datos relacional actual, además de una estructura no optimizada y poco eficiente para este propósito, que complejizaba sus consultas y hubiese ralentizado el monitoreo de los miembros en general. Por tanto, se propuso instaurar un modo de almacenamiento nuevo que se adecuase a los datos médicos conectados en cuestión, probar su funcionamiento y velocidad de consultas y, además, realizar los algoritmos predictivos deseados, evaluados por exactitud, conectándolos en la plataforma de forma dinámica.

Apoyado en el estado del arte, se propuso una base de datos de grafos donde se guardasen solo los datos relevantes, realizar los algoritmos predictivos con la base de datos que ya tenían y cargar todos los datos necesarios y obtenidos de estos algoritmos en la plataforma web desde la base de datos de grafos de forma eficiente. Esto se logró, en primer lugar, replicando los datos relevantes de la base de datos actual de forma local, recorriéndolos en un entorno de Python, limpiándolos bajo criterios médicos adecuados, formateándolos como consultas y subiéndolos a una base de datos de grafos en CosmosDB, en Azure, la cual luego se conectó a la plataforma web de la Armada por una API utilizando NodeJS y Express. Con los datos locales ya mencionados, se generaron distintos modelos de machine learning para predecir el riesgo de síndrome metabólico y riesgo cardiovascular de los individuos. Los mejores resultados, en ambos casos, fueron de árboles de decisión con F1-Score alrededor de 0.84, los cuales fueron guardados en archivos de extensión “pkl” y utilizados para evaluarlos en usuarios registrados en la base de datos de grafos por medio de una API de Python, entregando una respuesta a la plataforma web en 4.05 segundos, cumpliendo con los criterios de eficiencia esperados. Se espera que los dos algoritmos creados queden implementados en producción para el lanzamiento de la plataforma web en Marzo a menos que se aplase por temas de investigación. El proyecto presenta un beneficio social positivo según los supuestos definidos y se adhiere a reglas de seguridad digital suficientes tanto en estructura como en mitigación de riesgos posibles a futuro.

1.2. Abstract

This paper is meant to detail, concisely, the experience, problematic and application of a solution to an engineering problem executed during the full-time internship process.

The company Blueng Ingenieros Spa. was hired by YoMeControlo and Avante to create two websites focused on preventive medicine. It was found that Avante, particularly, wished to use its medical records history to visualize specific data from their members and generate predictive algorithms with them. However, there was very limited accessibility to their current relational database, coupled with a non-optimized and inefficient structure for this purpose, that raised the complexity of queries and would have slowed down the monitoring of the members in general. Therefore, it was proposed to set up a new storage method that adapted better to the interconnected medical data in question, to test its performance and query speed and, additionally, to build up the predictive algorithms, evaluated on their accuracy, dynamically connecting them to the web page.

Supported on real life cases, it was proposed to use a graph database where only relevant data is stored, carry out the predictive algorithms with their preexisting database and load the data necessary and obtained from the algorithms onto the web page efficiently from the graph database. This was achieved, firstly, replicating the relevant data from the preexisting database locally, traversing it on a Python environment, filtering it under adequate medical criterium, formatting them as queries and uploading them to the graph database in CosmosDB, in Azure, which was then connected to the navy's web page through an API utilizing NodeJS and Express. With the aforementioned local data, different machine learning models were created to predict metabolic syndrome risk and cardiovascular risk on the individuals. The best results, in both cases, came from decision trees with an F1-Score of around 0.84, which were saved on files with the "pkl" extension and used to evaluate the data on registered users on the graph database through a Python API, giving a response to the web page in under 4.05 seconds, fulfilling the expected efficiency criteria defined for them. It is expected that both algorithms get implemented in the official deployment of the web page around March, unless it gets delayed for researching purposes. The project shows a positive social benefit and adheres itself to sufficient cybersecurity rules in its structure as well as in possible future risk mitigation.

2. Introducción

La presente pasantía fue realizada en Blueng Ingenieros Spa., una empresa chilena dedicada principalmente a realizar consultorías sobre temas referentes al área de Tecnologías de la Información. Esta empresa fue contratada por YoMeControlo, un grupo chileno dedicado a la innovación en el rubro de medicina preventiva, para trabajar en la creación de una plataforma web de medicina preventiva, capaz de generar órdenes médicas digitales con validez a nivel nacional, con el propósito de incentivar la realización de exámenes médicos preventivos, eliminando la necesidad de realizar una consulta inicial con un médico para obtener dicha orden. Al proyecto se suma la participación de la Armada de Chile, Avante, que desea tener su propia plataforma personal de Sanidad Naval con la misma función de crear órdenes médicas digitales, pero de forma obligatoria para sus miembros para poder realizar seguimiento médico de estos. El proyecto se enfoca en esta última organización. La pasantía yace bajo la supervisión de Dagoberto Álvarez, gerente y fundador de Blueng, y el equipo de desarrollo del proyecto consiste en: un coordinador, el mismo supervisor, dos ingenieros, una gerencia, un ingeniero de datos y un grupo de dev-ops para programar del que fui parte junto a dos integrantes más, los cuales asistieron en la creación de la plataforma web de base, pero no tuvieron parte en mi proyecto particular de pasantía.

Para la funcionalidad teórica de la plataforma web de Sanidad Naval, desean tener un seguimiento de ciertas mediciones médicas y condiciones de riesgo de los pacientes, tales como el riesgo de tener síndrome metabólico o problemas cardíacos. Para este propósito, Avante nos da acceso a una base de datos (BDD) relacional¹ con registros médicos históricos de sus miembros, que incluyen sus antecedentes médicos personales y familiares, sus exámenes realizados y sus respectivos resultados. De estos datos se desea derivar las variables antes mencionadas y generar algoritmos de Machine Learning (ML)² capaces de predecirlas desde la data existente. Esta información ayudaría a los médicos de la Armada a hacer seguimiento de los pacientes e imponerles exámenes preventivos adicionales de ser necesario. Las fórmulas para calcularlas, sin embargo, son confidenciales, ya que son fórmulas conocidas con modificaciones hechas por los médicos a cargo del proyecto, para ofrecer cálculos de mayor valor a nivel competitivo. Al principio de esta pasantía, el proyecto trataba sobre crear un algoritmo predictivo

¹ “Una base de datos relacional es una colección de información que organiza datos en relaciones predefinidas, en las que los datos se almacenan en una o más tablas (o “relaciones”) de columnas y filas, lo que facilita su visualización y la comprensión de cómo se relacionan las diferentes estructuras de datos entre sí.” (Google Cloud, 2023).

² “Un modelo de machine learning es una expresión de un algoritmo que analiza montañas de datos para encontrar patrones o realizar predicciones.” (Parsons C., 2021).

que sirviera a YoMeControlo con estos datos y los suyos propios para recomendar exámenes a clientes, pero se descubrieron discrepancias importantes entre la información de Avante y la de YoMeControlo que imposibilitaban esta unión, por lo que el enfoque cambió a mediados de Septiembre a realizar algoritmos más simples solo para Avante.

El problema sobre esto recae en el almacenamiento actual de sus datos. La BDD utilizada, aunque muy segura (el acceso requiere una VPN³, una clave y varios comandos específicos), yace muy pobremente estructurada y no optimizada para el trabajo deseado. En primer lugar, todas las tablas carecen de una índice único o llave principal (primary key) salvo la que tiene datos de usuarios, todas las demás utilizan los mismos id de usuarios sin índices de registros propios, lo cual puede llevar a data duplicada y afectar la consistencia física y referencial de los datos (Bisso I. L., 2021). Esto resulta en un problema para encontrar registros específicos (complejiza las consultas necesarias para lograrlo y por tanto tardan más en correr), tarea que es necesaria para llenar los perfiles individuales en la plataforma web propuesta. En segundo lugar, la información no se encuentra bien distribuida. Una tabla, posiblemente la más importante, contiene los datos de distintas visitas médicas de cada usuario, y contiene más de ciento-cincuenta variables, que definen información médica básica (ej: peso, presión sanguínea, etc.), exámenes, evaluaciones, antecedentes médicos y comentarios. Tener todas estas variables en una misma tabla logra que la mayor parte de los datos en una fila sean nulos, pero reserven espacio de todos modos. Además, los antecedentes médicos son constantes, cada fila con un usuario repetido también repite estos datos, lo cual equivale a un cuarto de la data en una fila siendo duplicada cada vez que un usuario se registra más de una vez. Otro problema importante yacía en la poca flexibilidad que les entrega el esquema relacional (Lutkevich B., 2021), siendo que cada examen es una columna extra en la tabla antes mencionada y Avante planeaba agregar más exámenes a sus registros, implica que se planeaba agregar más columnas a la misma tabla, dejando registros vacíos en todas las mediciones anteriores. Esto sería más de 200,000 espacios vacíos, habiendo uno por fila, lo cual no influye especialmente en el almacenamiento (1 byte por espacio vacío no llegaría a más de 5 Mb en la BDD entera), pero complejiza las consultas y baja por ello el rendimiento de la BDD, ya que cargar una sola fila obliga a obtener todos los campos del registro y filtrar fuera los que no sirvan, que sería la mayoría de estos (suponiendo que sirvan treinta variables, ya se debe filtrar más del 80%). No les sería

³ “Una VPN o red privada virtual crea una conexión de red privada entre dispositivos a través de Internet. Las VPN se utilizan para transmitir datos de forma segura y anónima a través de redes públicas.” (Amazon Web Service, 2023).

posible agregar variables sin este problema si no se crean nuevas tablas, lo cual entraría en conflicto con el esquema actual y requeriría unir más tablas para sacar datos estadísticos de los usuarios. Entonces el cambio que se requiere debe ser de otro tipo.

Mantener la data médica actualizada de los usuarios a la vez que se tiene un registro histórico ordenado es de suma importancia para hacerle seguimiento a las condiciones de los pacientes, que es el objetivo de esta plataforma web. Las variables por predecir, en particular, tienen gran importancia en este contexto. Según el informe de una encuesta de salud nacional en Chile publicado el año 2018, entre 2016 y 2017 un 41.2% de la población chilena padecía síndrome metabólico y un 23.2% registró riesgo cardiovascular alto, cifras que subieron 5.9% y 1.9% desde sus mediciones anteriores los años 2009 y 2010 (Departamento de Epidemiología, 2018). También fue calculado que aproximadamente un 70% de las muertes por riesgo cardiovascular el año 2017 pudieron, en potencia, ser prevenidas aplicando correcciones a sus factores de riesgo respectivos (Troncoso-Pantoja C., Martínez-Sanguinetti M. A., Ulloa N., Celis-Morales C., 2020). Teniendo las mediciones actualizadas de estas variables, además de una visualización de sus factores de riesgo, la Armada sería capaz de hacer seguimiento activo de la salud de sus miembros y asistirles en bajar su riesgo de salud. Y al monitorear a cientos de personas a la vez, es importante que la velocidad de carga sea lo más rápida posible, que en promedio para sitios web de medicina serían 5.6 segundos (Das S., 2023).

3. Objetivos

El objetivo general del proyecto consiste en proponer e implementar un nuevo sistema de almacenamiento de datos, crear algoritmos predictivos de riesgo y conectarlos de forma eficiente a la plataforma de Sanidad Naval antes del 15 de Diciembre de 2023 para el lanzamiento preliminar de la plataforma web, a modo de visualizar la información médica de los pacientes de forma eficiente y mantener un seguimiento de sus condiciones.

Entre los objetivos específicos tenemos, en primer lugar, diseñar e instaurar un nuevo esquema de almacenaje virtual de datos, o una mejora al existente, que pueda cumplir con los criterios de consistencia y respuesta rápida, que almacene solo la información relevante y adecuada de forma ordenada y se adhiera a las reglas CIS Versión 8 sobre protección y manejo de datos en un contexto de seguridad virtual, en específico las aplicadas al nivel IG2 que es el segundo de tres niveles de seguridad y

el más pertinente según el tamaño de la empresa y la sensibilidad de los datos en cuestión (Center for Internet Security, CIS Controls, 2021).

En segundo lugar, construir, en base a los registros históricos médicos de Avante, un algoritmo predictivo que sea capaz de procesar e interpretar los datos de usuarios y sus exámenes realizados, además de entregar una respuesta final que se vea reflejada en la página de Sanidad Naval para cada usuario de forma rápida y precisa.

4. Estado del Arte

En el contexto de querer lograr un mejor almacenamiento de los datos, es importante entender casos similares y sus resoluciones respectivas. Si bien una estructura relacional puede ser robusta y consistente, los problemas que presenta la BDD mantenida por Avante afectan la consistencia de sus datos y la velocidad de sus consultas. Un caso similar es Medicare Plus, el sistema de salud principal de los Estados Unidos, posee una BDD relacional en la nube, en lenguaje NoSQL, que comúnmente se utiliza para modelos no relacionales, pero en su caso permite realizarle consultas más adecuadas para sus datos relacionados a la vez que se mantiene segura y escalable en la nube (Patil P., 2023). Esto nos demuestra que un cambio en la sintaxis a NoSQL de las consultas puede ser beneficioso para la búsqueda de registros específicos.

Por otro lado, un caso similar de manejo de datos interrelacionados se ve en las redes sociales, donde las relaciones entre personas definen qué contenido mostrarles y qué perfiles o productos sugerirles. Redes sociales como Snapchat utilizan BDD no relacionales⁴ para estos fines (Buuck B., 2023). Sin embargo, Facebook sigue utilizando una relacional como principal, en conjunto con varias BDDs no relacionales para distintas funcionalidades, donde se destaca el uso de Apache Cassandra en la barra de búsqueda, para un filtrado de opciones eficaz (Sarawagi S., 2023). Esto prueba que en la industria se puede utilizar más de una BDD aplicando cada una en tareas distintas.

Enfocando más el análisis en la capacidad de una BDD de asistir en proyectos de ML, dos ejemplos llaman la atención en especial. En primer lugar, el uso de una base de datos NoSQL en Apache Cassandra por parte de Netflix, en conjunto con Elasticsearch y Apache Iceberg para crear Marken, un

⁴ “Una base de datos no relacional es aquella que no usa el esquema tabular de filas y columnas que se encuentra en la mayoría de los sistemas de bases de datos más tradicionales.” (Microsoft Learn, 2023).

servicio interno de anotaciones en un formato de esquema muy simple y conveniente para la carga a algoritmos de ML (Stiller E., 2023). El uso de herramientas de analítica especializados, conectados a la BDD, era posible para el proyecto, aunque complejo de instaurar y pensado para proyectos de mayor tamaño, así que se tuvo en cuenta, pero no se terminó por realizar.

El segundo ejemplo es Amazon DynamoDB, otra base de datos NoSQL, que es utilizada por Capcom, Zoom, Dropbox, etc. y es preferida por su velocidad de respuesta y capacidad de escalamiento horizontal. Esta última refiere a la división de la data en varios servidores para escalar el espacio de almacenamiento, en vez de requerir más hardware para ello, lo cual es de suma importancia para estas compañías, para entregar una respuesta rápida en sus algoritmos para miles de usuarios a la vez. (Amazon Web Services, 2023).

De lo dicho anteriormente, la literatura nos indica que es más común y útil tener una BDD no relacional con NoSQL por diversas razones: provee un sistema flexible y dinámico de almacenamiento, alta velocidad transaccional para sets de datos de gran tamaño y alta escalabilidad gracias a su escalamiento horizontal, lo cual no es posible en BDDs relacionales con SQL. Es también común tenerlas funcionando junto a otras BDDs con propósitos distintos, enfocándolas en tareas que les sean más simples según su sintaxis y estructura. Las BDD más utilizadas para este propósito son MongoDB, Apache Cassandra y MLDB (Kumar P., 2022).

5. Propuestas de Solución

Habiendo visto la relevancia en la industria de las estructuras no relacionales, y sabiendo que Avante guarda su BDD relacional con alta seguridad y privacidad, no está al alcance del proyecto ofrecerle una BDD relacional nueva y mejorada, sino más bien introducirle a una BDD no relacional que la complemente en temas de respuesta rápida y algoritmos predictivos.

La elección de la BDD es la más importante en esta instancia, ya que esta elección definiría las opciones que se tienen para estructurar los datos, iterar a través de ellos y conectarlos a la plataforma web. Las opciones principales encontradas fueron una BDD documental, una de grafos y una orientada a objetos. Las tres son buenas en velocidad y en procesamiento para datos estadísticos. Sin embargo, la BDD orientada a objetos tiene una complejidad que escapa a las necesidades de Avante, ya que cada registro se toma como una instancia de un objeto, con soporte para métodos, herencia de componentes,

etc. Las dos restantes guardan los datos en formato JSON, aunque la documental se utiliza para datos complejos e individuales en documentos, mientras que las de grafos se estructuran en nodos, relaciones y propiedades para datos estrictamente relacionados y flexibles, lo cual sirve de forma especial para el presente proyecto y permite sacar provecho de sus interrelaciones (Kumar P., 2022).

Al inicio del proyecto se planeó realizar un grafo en la plataforma de Neo4J, pero más adelante se decidió realizarlo en la nube de Azure, dado que Avante ya tiene una cuenta activa como empresa y esta plataforma entrega buenas herramientas para agregar seguridad a los datos. Esto limita el trabajo a CosmosDB, la herramienta de Azure para crear BDDs no relacionales, entre ellas las de grafos. El análisis y limpieza de datos, junto con la construcción de los algoritmos de ML, se realizaron en VSCode en un entorno de Python. Sin embargo, se descubrió que, a pesar de ser muy conveniente para encontrar registros individuales, la BDD de grafos no es capaz de entregar grandes cantidades de registros a la vez (a menos que se pague para que resista tal carga de consultas, lo cual fue impedido por el presupuesto). De este modo, se decidió que los algoritmos predictivos serían realizados con la data retirada directamente de la BDD relacional, la cual fue replicada de forma local en el computador de trabajo corriendo archivos de lenguaje SQL para recrear las tablas y registros, a modo de recorrerla con libertad sin necesidad de pasar por el extenso proceso de conexión a la BDD real. Entonces, si un algoritmo de ML probase ser más eficiente que aplicar las fórmulas directamente, estos quedarían guardados en archivos de extensión “pkl” en el entorno de Python, listos para su uso en casos reales. De otro modo, no habrá necesidad de utilizar la BDD relacional para este propósito, esta opción se tiene en cuenta porque podría resultar más veloz. La BDD de grafos tendría entonces todos los datos necesarios para cargarlos a la página, los datos de un usuario con los exámenes necesarios realizados pasarían por los algoritmos predictivos al entrar en la página web, y sus resultados se subirían a la BDD de grafos conectada a través de una API Gremlin (integrada en Azure por defecto), listos para ser cargados a la plataforma web con tiempos de espera mínimos en lo posible.

6. Evaluación Económica

El costo de oportunidad de este proyecto en particular existe en todo lo relacionado a la BDD: su tamaño, atributos y capacidad de respuesta. Se consideran costos hundidos los incurridos en la creación de las plataformas web a las que se conectarían la BDD y los algoritmos de ML.

Para una BDD en CosmosDB, el procesamiento es a través de RU/s (unidades de consultas por segundo), donde cada 100 RU/s cuestan \$5.84 USD al mes. La licencia de Azure permite que los primeros 1000 RU/s sean gratis, pero se considera tener escalamiento automático, que permite utilizar el doble de RU/s por hora de ser necesario, con un precio de 1.5 veces el de las RU/s actuales. En el cálculo del costo, se considera que, en un mes de 720 horas, 220 puedan ser con escalamiento aplicado. Por otro lado, se suman a los costos de operación el costo mensual de almacenamiento (menor a 1GB), la firewall de Azure y API de Python más básicas para producción y la aplicación de registros analíticos de Azure Monitor para control de acceso (Azure, 2023). Azure ofrece dos copias de seguridad mensuales y redundancia local gratis, la cual se prefirió por temas de presupuesto, sabiendo que la redundancia regional puede casi duplicar el costo de las RU/s por cada región extra. El capital de trabajo incluye el salario anual del desarrollador de la BDD el año cero, seguido del salario promedio anual de \$85,333,398 CLP de un administrador de BDD NoSQL para el resto del proyecto (GLASSDOOR, 2023).

Por otro lado, como el proyecto es para la Armada chilena, no tiene retornos monetarios, y por tanto se mide, en cambio, el beneficio social de su realización. Para esto consideramos dos variables: La primera es el dinero que los miembros de la Armada se ahorrarían en ir a consultas de diagnóstico, la segunda es los sueldos que la Armada, teóricamente, se ahorraría al no tener que realizar dichas consultas. La primera variable toma en cuenta que, según su data histórica en la BDD relacional, de 755,696 usuarios registrados, solo el 3.61% de los usuarios se han hecho exámenes. Además, un registro del año 2021 que indica que Avante solo tenía 20,000 miembros ese año, y que iban restando aproximadamente 1,000 miembros anuales desde el año 2019 (Datosmacro, 2021). El precio de una consulta médica se promedió a \$57,500 CLP (Maremaa T., 2022). La segunda variable considera un registro del año 2018 que indica que 262 enfermeros trabajaban en la Armada en ese entonces, con un sueldo promedio mensual de \$1,985,750 CLP (Armada de Chile, 2018). Se utilizaron estos datos en ausencia de cifras más actualizadas y suponiendo, de forma arbitraria, que un 2% de los médicos trabajan en realizar exámenes diagnósticos, por lo que sus sueldos pudiesen ser ahorrados con la propuesta de solución. Los datos mencionados y consideraciones se pueden encontrar en el Anexo (Tabla 1).

El flujo de caja resultante (Tabla 2), fue medido en CLP, considerando el dólar a \$887.06 CLP (17/11/2023), un horizonte de evaluación de 3 años, una inversión inicial de \$10,000,000 CLP equivalente a un cuarto de la inversión del proyecto grande sobre crear las dos plataformas web, un descuento por IVA del 19% y una tasa de descuento del 6%, fijada para proyectos sociales en Chile (Departamento de Metodologías, División de Evaluación Social de Inversiones., 2023). La VAN resultante, al igual que la TIR, son positivas, lo cual nos indica que el proyecto sería beneficioso, aunque no entrega retornos reales. Se realizó también un análisis de sensibilidad al porcentaje de sueldos ahorrados de enfermeros (Tabla 3) y al horizonte de evaluación (Tabla 4). En el primer caso se demostró que, si realmente se ahorraran menos de un 1.2% de los sueldos, la TIR se vuelve menor a la tasa de descuento, indicando que ya no sería beneficioso realizar el proyecto. Para el segundo caso, se demostró que extender el proyecto el doble de años seguiría siendo beneficioso, pero se recomienda que no se extienda a más de 5 años, para poder reentrenar los algoritmos de ML con la data nueva obtenida, para así robustecerlos.

	Año 0	Año 1	Año 2	Año 3
Beneficio Social (+)	-	\$41,547,527	\$39,470,151	\$37,392,774
Costos de Operación (-)	-	-\$4,146,704	-\$4,146,704	-\$4,146,704
Resultado Operacional	-	\$37,400,823	\$35,323,447	\$33,246,070
Sueldos Ahorrados (+)	-	\$124,863,960	\$124,863,960	\$124,863,960
Utilidad antes de Impuestos	-	\$162,264,783	\$160,187,407	\$158,110,030
IVA (-)	-	-\$30,830,309	-\$30,435,607	-\$30,040,906
Utilidad después de Impuestos	-	\$131,434,474	\$129,751,800	\$128,069,125
Inversión (+)	-\$10,000,000	-	-	-
Capital de Trabajo (-)	-\$2,500,000	-\$85,333,398	-\$85,333,398	-\$85,333,398
Flujo efectivo de activos	-\$12,500,000	\$46,101,076	\$44,418,402	\$42,735,727

Tasa social de Descuento	6%
--------------------------	----

VAN Social	\$133,002,719
------------	---------------

TIR	362%
-----	------

Tabla 2: Flujo de Caja del proyecto. Cabe recalcar que la VAN y la TIR no representan retornos reales, solo demuestran que el proyecto es beneficioso con las consideraciones definidas.

7. Medidas de Desempeño

En el caso de la BDD de grafos, se decidió medir la eficiencia de su respuesta comparando la carga de datos ingresados con la capacidad de respuesta para consultas simples y complejas, además de una comparación teórica de su velocidad de respuesta y complejidad de consultas con cómo sería si se utilizara la BDD relacional para entregar datos. Se puso como objetivo que la data se pueda cargar en la plataforma web en menos de 5.6 segundos, dado que es el tiempo promedio de una página web de medicina en la actualidad (Das S., 2023).

Para medir la eficiencia de los algoritmos predictivos, se utilizará el tiempo entre que entra un usuario nuevo y se entrega la respuesta del algoritmo desde su data (de ser suficiente), comparando también con la velocidad del algoritmo sin modelo aplicado (midiendo las variables con fórmulas teóricas). La exactitud de los algoritmos también es de suma importancia, por lo que se medirá utilizando los puntajes de Silhouette⁵ o Accuracy⁶, F1-Score⁷ y curvas ROC⁸ según sean pertinentes en los modelos aplicados.

8. Metodología y Desarrollo

Para la realización del proyecto, se siguió una metodología de desarrollo de software tipo espiral, debido a que los pasos a seguir eran claros, pero existía una necesidad de iterar por distintos caminos dentro de la solución para ver cual se adecuaba más al proyecto (distintos algoritmos predictivos con mejoras o cambios en los filtros de datos y distintas conexiones posibles de la BDD a la plataforma web). De este modo, se siguieron los siguientes pasos:

1. Diseñar la BDD de grafos en la plataforma draw.io y validar su diseño con la empresa, refinar y modificar según deseasen. En el transcurso del proyecto, nuevos exámenes, nodos y funcionalidades

⁵ "El Silhouette Score es una métrica ampliamente utilizada para evaluar la calidad de los resultados de agrupación. Mide qué tan similar es un punto de datos a su propio grupo en comparación con otros grupos." (Linux-Console.net, 2023).

⁶ "La métrica accuracy representa el porcentaje total de valores correctamente clasificados, tanto positivos como negativos." (Díaz R., 2020).

⁷ "F1-score es una métrica de evaluación en machine learning que mide la exactitud de un modelo. Combina los puntajes de precisión y recall de un modelo." (Kundu R., 2022).

⁸ "El análisis ROC (Receiver operating characteristic o característica operativa del receptor) es una forma útil de evaluar la precisión de las predicciones de modelo al trazar la sensibilidad frente a la especificidad de una prueba de clasificación (ya que el umbral varía en todo un rango de resultados de pruebas de diagnóstico)". (IBM, 2021).

fueron agregadas al diseño de forma dinámica, apoyados por la flexibilidad del modelo de grafos, lo cual permitió añadir nueva data a la plataforma web sin perjudicar la estructura o velocidad del sistema.

2. Estructurar la BDD en Azure CosmosDB con API Gremlin y un servidor de Azure, luego conectarla a un entorno de NodeJS por una API web de Azure, donde quede habilitada para hacerle consultas desde la plataforma web. El acceso de la API web se limitó por IP y por el dominio de la página web entregado por Avante.

3. Replicar las tablas necesarias de la BDD relacional de forma local y limpiar los datos en un entorno de Python (Imagen 1), incluyendo eliminación de datos erróneos según límites de mediciones médicas reales. Los criterios utilizados para la limpieza de los datos y sus justificaciones fueron documentados en la página de Confluence del proyecto.

4. Encriptar la data sensible (resultados de exámenes), formatearla en consultas de Gremlin e ingresarla a la BDD de grafos de forma procedural. La encriptación de los datos sensibles se logró con la biblioteca de Python Fernet, utilizando encriptación simétrica con una secuencia de 128 caracteres no visible en la plataforma web (Imagen 2, 3, 4).

5. Tabular los datos y calcular las variables objetivo con fórmulas revisadas por los doctores a cargo. Por querer los algoritmos basados en fórmulas personalmente modificadas, el avance de estos estuvo a merced de los doctores entregando tales fórmulas, lo cual retrasó el proyecto de forma significativa, pero se logró finalmente concretar la propuesta de solución.

6. Generar diversos algoritmos de ML cuidando el número de variables, su relevancia, su colinealidad y su uso correcto en los algoritmos, midiendo eficiencia y eficacia y escogiendo el mejor modelo para cada variable objetivo. De tener un modelo suficientemente adecuado, se guardó en un archivo “pkl” para traerlo al entorno de Python.

6. Generar una función POST en la API de Python que corra el algoritmo para un usuario con data suficiente, a través de la fórmula o del algoritmo en archivo “pkl” guardado como corresponda, y suba su resultado a la BDD de grafos, cargándolo en la página web a la vez (Imagen 5, 6).

7. Aplicar el firewall, editar control de acceso para evitar conexiones ajenas en IP ya no utilizadas por el quipo y aplicar Azure Monitor para monitorear quién entra, sale y qué se hace dentro de la BDD de grafos.

8. Subir la plataforma web a producción y evaluar su funcionamiento.

9. Matriz de Riesgos

La matriz de riesgos se puede encontrar en el Anexo (Imagen 7), al igual que sus mitigaciones respectivas resumidas (Imagen 8). Concretando sus definiciones, en lo que respecta a seguridad virtual, es de suma importancia tener un registro actualizado de accesos y acciones en la BDD con Azure Logs, conectado a Azure Monitor, con accesos jerarquizados a la base de datos por credencial, abriéndolos solo cuando sea necesario. Todo borrado o modificado de información se puede revertir con copias de seguridad, utilizando los registros para reparar daños que se le escapen. De caerse el servidor de la BDD, como está en redundancia local, no queda opción más que notificar en la página y esperar que vuelva. Redundancia en otras regiones mitigaría esto, pero duplicaría el precio de los RU/s, así que queda a discreción futura de la empresa. Toda vulnerabilidad en la conexión a la plataforma web se responde cambiando los códigos de conexión y realizando las acciones necesarias, migrando o asegurando la conexión nuevamente. Ante un sobre entrenamiento de los datos, se reevaluarán los procedimientos, pero es probable que la data sea relativamente carente para los algoritmos en los casos actuales (menos de 10,000 registros por algoritmo, ya que pocos registros de los existentes cumplen con tener la data suficiente para calcular las variables). La empresa ha decidido correr este riesgo, en oposición de escalar la data para mejorarlo, por ser data médica sensible.

10. Resultados Finales

Con respecto a la BDD de grafos, el diseño final se puede encontrar en el Anexo (Imagen 9, 10). Se le agregaron a la fecha un total de 27,476 vértices y 817,186 relaciones. La respuesta entregada para encontrar un registro específico toma alrededor de 0.7 segundos en aparecer, cercano a 1 segundo en consultas complejas. En comparación, la BDD relacional tarda aproximadamente 1.3 segundos en encontrar un registro específico, 1.5 con filtros complejos.

Sobre los algoritmos de ML, al cálculo del riesgo de síndrome metabólico se le aplicaron modelos de K-means, GMM y DBScan para detección de grupos y un modelo de Árbol de Decisión para la clasificación. Los primeros cuatro fueron a petición de Avante para encontrar grupos en la data en ausencia de una clasificación formal del índice obtenido por fórmula y todos entregaron puntajes de Silhouette bajos (alrededor de 0.2, un buen puntaje está entre 0.8 y 1). Solo al concluirse que estos no servirían para la predicción fue cuando se propuso formar un modelo de clasificación. De las dos variables de riesgo evaluadas, se logró obtener dos modelos de Árbol de Decisión capaces de

clasificarlas, con puntajes alrededor de 0.84 (de un valor máximo 1) tanto para sus F1-Score como para sus Accuracy, probando ser suficientes para su aplicación en la plataforma web (Tabla 5, Imagen 11, 12).

Variable	Modelo ML	Puntaje (S=Silhouette, A=Accuracy, F=F1-Score)
Riesgo Síndrome Metabólico	K-means	S = 0.23
	GMM	S = 0.26
	DBSCAN	S = 0.3
	Árbol de Decisión	F = 0.842 A = 0.842
Riesgo Cardiovascular	Árbol de Decisión	F = 0.841 A = 0.844

Tabla 5: Puntajes obtenidos en modelos de machine learning por variable. Los algoritmos de agrupación tienen puntaje Silhouette y los de clasificación tienen puntajes Accuracy y F1-Score.

Las curvas de ROC por categoría de riesgo cardiovascular, sin embargo, son carentes (Imagen 13), pero Avante decidió aceptar los modelos de no encontrarse una mejor opción a futuro. Por temas de tiempo, la variable de síndrome metabólico aún no tiene una clasificación oficial para conectar a la plataforma, pero la variable de riesgo cardiovascular logró ser conectada y se midió su funcionamiento de forma local, logrando una respuesta del algoritmo en 4.05 segundos. En comparación se encontró que correr su fórmula por sí sola tardaba hasta 6 segundos, por lo que se justifica tener los modelos de ML a pesar de tener poca data para entrenarlos. Al tardar menos de 5.6 segundos, se toma por cumplido el objetivo de eficiencia estipulado. La vista final de los datos se evidencia al final del Anexo (Imagen 14).

11. Conclusiones

En la realización de este proyecto, se logró, en primer lugar, entender el funcionamiento de una BDD de grafos y su justificación de uso en temas de respuesta eficiente para navegar a través de data estrictamente relacionada. Además, queda justificado su uso en conjunto con la base de datos relacional de la Armada que, a pesar de tener una estructura ineficiente, es esta misma estructura la que permitió tabular los datos y limpiarlos de forma ordenada para crear los algoritmos de ML y conectarlos entonces a la BDD de grafos. Prueba esto que, para proyectos de ML, es beneficioso tener más de una BDD, siempre y cuando se adecuen a los datos y se utilicen con propósitos distintos.

Por otro lado, el trabajo con las herramientas de Azure mostró ciertas ventajas y desventajas claras. Tener la mayor parte de los elementos en una sola plataforma fue de suma importancia para manejar la seguridad y las conexiones. La BDD de grafos, el servidor y las API se crearon todas en Azure (las API se mantienen arriba utilizando el código guardado en una cuenta de Github), sin mencionar que las plataformas web del proyecto son subidas al dominio web por parte de otro servidor en Azure. Sin embargo, los costos de estas herramientas, especialmente el de las RU/s y el firewall, presentaron limitaciones en el uso concreto de la inversión del proyecto. Una principal razón para reevaluar el proyecto una vez finalice el horizonte de evaluación es la posibilidad de necesitar utilizar más RU/s a medida que se obtienen más registros para cada usuario.

Otra limitación importante fue el uso de CosmosDB. A pesar de estar en la misma plataforma, facilitando su conexión a través del proyecto, se encontró que su lenguaje de consultas, Gremlin, no está completo. Gremlin posee una biblioteca amplia de comandos y ciertos comandos de utilidad no se pueden interpretar en CosmosDB. Uno de los más importantes ausentes fue el comando de crear un vértice o relación solo si no existe ya, lo cual se tuvo que reemplazar por un comando que ve si el registro existe y condicionar en NodeJS que se cree si no. Además, la interfaz de CosmosDB no tiene herramientas para filtrar, borrar, agregar o modificar data sin utilizar consultas de Gremlin, lo cual significó que, para toda acción relacionada a la data ingresada en la BDD de grafos, se tenía que formar una consulta especial, y esto supone un trabajo más complejo para quien quede administrando la BDD una vez se lance el proyecto.

Por último, a pesar de haber cumplido los objetivos en términos de eficiencia y de exactitud predictiva, la mayor limitante en su uso teórico sigue siendo la cantidad de registros que cumplían con poder entrar a los algoritmos predictivos. Los doctores a cargo del proyecto están al tanto de esto y

esperan a futuro conseguir información general de los miembros de la Armada que se pueda internalizar en los algoritmos para que pueda aplicarse a un número mayor de usuarios y robustecer sus predicciones con ello, lo cual podría atrasar la implementación del proyecto un par de meses.

12. Bibliografía

- Bisso I. L. (2021). *"Why Do You Need a Primary Key in a Database Table?"*. LearnSQL. Recuperado de: <https://learnsql.com/blog/what-is-a-primary-key/>
- Lutkevich B. (2021). *"relational database"*. TechTarget Data Management. Recuperado de: <https://www.techtarget.com/searchdatamanagement/definition/relational-database>
- Departamento de Epidemiología. (2018). *"Informe Encuesta Nacional de Salud 2016-2017: Riesgo cardiovascular"*. Ministerio de Salud: Santiago de Chile. Recuperado de: http://epi.minsal.cl/wp-content/uploads/2021/06/Informe_RCV_ENS_2016_17.pdf
- Troncoso-Pantoja C., Martínez-Sanguinetti M. A., Ulloa N., Celis-Morales C. (2020). *"La mayoría de las enfermedades cardiovasculares se atribuyen a factores de riesgo que podrían ser modificados con cambios de los estilos de vida"*. Revista médica de Chile, 148(1), 126-128. <https://dx.doi.org/10.4067/S0034-98872020000100126>
- Das S. (2023). *"How fast should a website load in 2023?"* BrowserStack. Recuperado de: <https://www.browserstack.com/guide/how-fast-should-a-website-load>
- Johnson J. (2023). *"6 Different Types of Databases with Companies That Build Each One"*. History-Computer. Recuperado de: <https://history-computer.com/different-types-of-databases-with-companies-that-build-each-one/>
- Kumar P. (2022). *"Why and Which Database in Machine Learning, MySQL or MongoDB"*. Medium. Recuperado de: <https://medium.com/geekculture/why-and-which-database-in-machine-learning-mysql-or-mongodb-d8e43b29aeb2#:~:text=Some%20of%20the%20reasons%20Why,Processing%20for%20large-scale%20projects%3F&text=SQL%20databases%20can%20store%20a,biggest%20flaw%20in%20SQL%20databases.>
- Buuck B. (2023). *"Seven Real-Life Database Examples"*. StreamSets. Recuperado de: <https://streamsets.com/blog/seven-real-life-database-examples/>

- Patil P. (2023). "10 Real-Life Examples of Databases In Action". RedSwitches. Recuperado de: <https://www.redswitches.com/blog/examples-of-databases/#The-10-Real-Life-Database-Examples>
- Amazon Web Service. (2023). "What's the Difference Between a Graph Database and a Relational Database". Recuperado de: <https://aws.amazon.com/compare/the-difference-between-graph-and-relational-database/#:~:text=Graph%20databases%20are%20useful%20for,%2C%20and%20DELETE—on%20tables.>
- Sarawagi S. (2023). "Facebook database [Updated] – A thorough insight into the databases used @Facebook" Scaleyourapp. Recuperado de: <https://scaleyourapp.com/what-database-does-facebook-use-a-1000-feet-deep-dive/#:~:text=1.-,What%20database%20does%20Facebook%20use%3F,as%20the%20MySQL%20Database%20engine.>
- GLASSDOOR. (2023). "How much does a Database Administrator NoSQL (Database Administrator) make?". Recuperado de: https://www.glassdoor.com/Salaries/us-nosql-salary-SRCH_IL0,2_IN1_KO3,8.htm
- Azure. (2023). "Azure CosmosDB Pricing". Recuperado de: <https://azure.microsoft.com/en-us/pricing/details/cosmos-db/autoscale-provisioned/>
- Azure. (2023). "Azure Firewall Pricing". Recuperado de: <https://azure.microsoft.com/en-us/pricing/details/azure-firewall/>
- Azure. (2023). "Azure Monitor Pricing". Recuperado de: <https://azure.microsoft.com/en-us/pricing/details/monitor/>
- Azure. (2023). "API Management Pricing". Recuperado de: <https://azure.microsoft.com/en-us/pricing/details/api-management/>
- Maremaa T. (2022). "¿Cuál es el costo promedio de una consulta médica en Chile?". Encuadrado. Recuperado de: <https://www.encuadrado.com/blog/valor-consulta-medica-particular-en-chile#:~:text=Una%20consulta%20médica%20en%20Chile%20generalmente%20puede%20llegar%20a%20costar,a%2057%20dólares%20estadounidenses%2C%20aproximadamente.>
- Armada de Chile. (2018). "Escala de Remuneraciones del Personal de Médicos Ley 15.076 de la Armada a enero 2018". Gobierno Transparente. Recuperado de: https://transparencia.armada.cl/transparencia_activa/remuneraciones/remuneracionmed.html

- Armada de Chile. (2018). *"Otras contrataciones sujetas al Código del Trabajo"*. Gobierno Transparente. Recuperado de: https://transparencia.armada.cl/transparencia_activa/codtrabajo/sanidad/per_otros.html
- Departamento de Metodologías, División de Evaluación Social de Inversiones. (2023). *"Informe Precios Sociales 2023"*. Ministerio de Desarrollo Social y Familia. Recuperado de: https://sni.gob.cl/storage/docs/230401_Informe_Precios_Sociales_2023_SNI.pdf
- Expansión. (2021). *"Chile – Personal de defensa"*. Datosmacro. Recuperado de: <https://datosmacro.expansion.com/estado/defensa-ejercitos/chile>
- Center for Internet Security, CIS Controls. (2021). *"CIS Critical Security Controls Version 8"*. Recuperado de: <https://learn.cisecurity.org/CIS-Controls-v8-guide-pdf>
- Choudhury A. (2020). *"Top Databases Used In Machine Learning Projects"*. Analytics India Magazine. Recuperado de: <https://analyticsindiamag.com/top-databases-used-in-machine-learning-projects/>
- Stiller E. (2023). *"Netflix Built a Scalable Annotation Service Using Cassandra, Elasticsearch and Iceberg"*. InfoQ. Recuperado de: <https://www.infoq.com/news/2023/02/netflix-annotations-cassandra/#:~:text=The%20team%20at%20Netflix%20decided,database%20that%20provides%20horizontal%20scalability>
- Amazon Web Services. (2023). *"Amazon DynamoDB Customers"*. Recuperado de: <https://aws.amazon.com/dynamodb/customers/>
- Amazon Web Services. (2023). *"¿Qué es una red privada virtual (VPN)?"*. Recuperado de: <https://aws.amazon.com/es/what-is/vpn/#:~:text=Una%20VPN%20o%20red%20privada,a%20través%20de%20redes%20públicas>
- Google Cloud. (2023). *"¿Qué es una base de datos relacional?"*. Recuperado de: <https://cloud.google.com/learn/what-is-a-relational-database?hl=es-419>
- Parsons C. (2021). *"¿Qué Es un Modelo de Machine Learning?"*. NVIDIA. Recuperado de: <https://la.blogs.nvidia.com/2021/09/28/que-es-un-modelo-de-machine-learning/#:~:text=Un%20modelo%20de%20machine%20learning%20es%20una%20expresión%20de%20un,motores%20matemáticos%20de%20la%20IA.>
- Linux-Console.net (2023). *"Evaluación del desempeño de agrupaciones en Scikit Learn"*. Recuperado de: <https://es.linux-console.net/?p=26174>
- Díaz R. (2020). *"Métricas de Clasificación"*. The Machine Learners. Recuperado de: <https://www.themachinelearners.com/metricas-de-clasificacion/>

- Kundu R. (2022). “F1 Score in Machine Learning: Intro & Calculation”. V7. Recuperado de: <https://www.v7labs.com/blog/f1-score-guide>
- IBM Corporation. (2021). “Análisis ROC”. Recuperado de: <https://www.ibm.com/docs/es/spss-statistics/beta?topic=features-roc-analysis>
- Microsoft Learn. “Datos no relacionales y NoSQL”. Recuperado de: <https://learn.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data>

13. Anexo

Variable	Costo (CLP)	Tiempo	Tipo
Consulta médica promedio	\$57,500	-	Variable (unidad)
Consultas a la BDD	\$281,021	Anual	Variable
Firewall	\$3,027,358	Anual	Fijo
Redundancia Regional	-	Anual	Fijo (local)
Sueldo Admin BDD NoSQL	\$85,333,398	Anual	Fijo
Almacenamiento	\$2,661	Anual	Variable
Copias de Seguridad Mensuales	-	Anual	Fijo
Salario Enfermeros	\$1,985,750	Mensual	Variable
Azure Monitor	\$30,923	Anual	Variable
API Python	\$67,062	Mensual	Variable

Tabla 1: Tabla de costos referentes al proyecto. Las variables “Consulta médica promedio” y “Salario enfermeros” son las únicas que incurren en el beneficio social del proyecto.

Enfermeros diagnósticos	VAN	TIR
2.5%	\$200,801,848	572%
2%	\$133,002,719	367%
1.5%	\$65,627,891	156%
1.2%	\$25,075,704	6%

Tabla 3: Análisis de sensibilidad de porcentaje de enfermeros cuyos salarios se podrían ahorrar con el proyecto. Se descubrió de este análisis que si el porcentaje fuese menor a 1.2% la TIR sería menor a la tasa de descuento y, por tanto, ya no sería beneficioso el proyecto.

Horizonte de Evaluación	VAN	TIR
3 años	\$133,002,719	367%
4 años	\$166,182,099	370%
5 años	\$196,089,029	370%
6 años	\$223,116,896	370%

Tabla 4: Análisis de sensibilidad de horizonte de evaluación. Se descubrió de este análisis que el proyecto permanece rentable hasta el doble del tiempo pensado inicialmente, pero se recomienda reevaluarlo a los 5 años de todos modos para robustecer los algoritmos de ML con nueva data.

```

# Antecedentes personales, familiares y hábitos
# Datos médicos básicos y evaluaciones
try:
    connection = mysql.connector.connect(host='localhost',
                                         database='AvanteStuff',
                                         user='root',
                                         password= )

    sql_select_Query = "select id,fecha_examen,ap_embarazo,ap_puerpera,ap_cardiovascular,ap_respiratorio,ap_digestivo,ap_endoc
    cursor = connection.cursor()
    cursor.execute(sql_select_Query)
    # get all records
    medrecords = cursor.fetchall()
    print("Total number of rows in table: ", cursor.rowcount)

except mysql.connector.Error as error:
    print("Failed in MySQL: {}".format(error))
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")

```

✓ 0.4s

Python

Total number of rows in table: 169832
MySQL connection is closed

Imagen 1: Ejemplo de recolección de registros de la base de datos relacional replicada de forma local del hardware de trabajo a un entorno de Python en un archivo de Jupyter Notebook en Visual Studio Code. En esta consulta se obtienen los datos de antecedentes de usuarios. Una vez obtenidas, se formatearon a lista, luego a tabla para análisis y filtrado.

```

##### Función para relación entre antecedentes y usuarios #####

antecedentes = []
for j in range(len(records_recent)):
    #for j in range(20):
        for i in filling:
            if(records_recent[i].iloc[j] == "SI"):
                if(i == "ap_bebe" or i == "ap_fuma") and records_recent[frequent[i]].iloc[j] != None:
                    enc = f.encrypt(records_recent[frequent[i]].iloc[j].encode()).decode()
                    A = ".property('frequency','{}').format(enc)"
                    antecedentes.append(("g.V('{}').addE('HAS_HISTORY').format(records_recent['id'].iloc[j])+A+.to(g.V('{}')).format(res[i])"))
                    continue
                if(i == 'ap_tratamiento_med' and records_recent['ap_trat_med_esp'].iloc[j] != None):
                    enc = f.encrypt(records_recent['ap_trat_med_esp'].iloc[j].encode()).decode()
                    A = ".property('treatment','{}').format(enc)"
                    antecedentes.append(("g.V('{}').addE('HAS_HISTORY').format(records_recent['id'].iloc[j])+A+.to(g.V('{}')).format(res[i])"))
                    continue
                if(i == 'e_hipert' and records_recent['e_hipert_trat'].iloc[j] != None):
                    enc = f.encrypt(records_recent['e_hipert_trat'].iloc[j].encode()).decode()
                    A = ".property('treatment','{}').format(enc)"
                    antecedentes.append(("g.V('{}').addE('HAS_HISTORY').format(records_recent['id'].iloc[j])+A+.to(g.V('{}')).format(res[i])"))
                    continue
            antecedentes.append(("g.V('{}').addE('HAS_HISTORY').format(records_recent['id'].iloc[j])+.to(g.V('{}')).format(res[i])"))

antecedentes

[77]
Python
...
"g.V('62783').addE('HAS_HISTORY').to(g.V('HIS31'))",
"g.V('235597').addE('HAS_HISTORY').property('frequency','gAAAAAB1SUs-EYdc91gVe9-jZliPKXGtbD5Bfn1M2AIAEpkxxx_V8oKVoFZte4wuBtUTZ34MeGsxwBkoSuJ1ZScXc9G4r3C2i",
"g.V('235597').addE('HAS_HISTORY').property('frequency','gAAAAAB1SUs-chs1ZuHpl183XfWPYLCrd67KcanVDnw_ALWEfwnWn6jtLQmuc1Sot1GB1ZXTQKYYnKhFWgNX1obcJsHo-ZQ0i",
"g.V('235597').addE('HAS_HISTORY').property('treatment','gAAAAAB1SUs-eQmJZ9Jp4YMaDXq2gb19IgcX8mD6x2u6YFeEHidc_L0x3L0TCF6U9d9ryzTfV4gyGQxfYt9gLvhNTSUrRMsUXI",
"g.V('235597').addE('HAS_HISTORY').to(g.V('HIS31'))",
"g.V('64247').addE('HAS_HISTORY').to(g.V('HIS9'))",
"g.V('58911').addE('HAS_HISTORY').to(g.V('HIS20'))",

```

Imagen 2: Ejemplo de cómo se formatearon y encriptaron los datos ya limpios de la BDD relacional replicada al lenguaje de consultas Gremlin. Aquí se muestra cómo se trataron los antecedentes de usuarios únicos, donde si un usuario tenía un antecedente, se creaba una relación entre ellos, y se agregaba una propiedad si había información adicional (ejemplo: frecuencia en que fuma).

```

def insert_vertices(gremlin_client,V):
    for vertex in V:
        callback = gremlin_client.submitAsync(vertex)
        if callback.result() is not None:
            #print("Inserted this vertex:\n{}".format(callback.result().one()))
            continue
        else:
            print("Something went wrong with this query: {}".format(vertex))

def insert_edges(gremlin_client,E):
    for edge in E:
        callback = gremlin_client.submitAsync(edge)
        if callback.result() is not None:
            #print("Inserted this edge:\n{}".format(callback.result().one()))
            continue
        else:
            print("Something went wrong with this query: {}".format(edge))

```

Imagen 3: Ejemplo de funciones utilizadas para cargar los datos guardados en las listas creadas con consultas (Imagen 2) a la BDD de grafos.


```

ENDPOINT = 'msdocs-gremlin-XXXXXXXXXX.gremlin.cosmos.azure.com'
DATABASE = 'AvanteDB'
COLLECTION = 'AvanteGraph'
PRIMARY_KEY = XXXXXXXXXX

[151] ✓ 0.0s Python

# Initialise client
print('Initialising client...')
gremlin_client = client.Client(
    'wss://' + ENDPOINT + ':443/', 'g',
    username="/dbs/" + DATABASE + "/colls/" + COLLECTION,
    password=PRIMARY_KEY
)
print('Client initialised!')

# Purge graph
#cleanup_graph(gremlin_client)
#cleanup_users(gremlin_client)

print('Finished!')

[158] ✓ 2.9s Python

... Initialising client...
Client initialised!
Finished!

```

Imagen 4: Conexión de BDD de grafos al entorno de Python para llenarla con los datos de la BDD relacional replicada de forma local.

```

//Calling the python function
await appendFile(
    join('args.json'),
    JSON.stringify(finalvars),
    {
        encoding: 'utf-8',
        flag: 'w',
    },
);
const pythonProcess = spawnSync('python3', [
    'parserPython.py',
    'first_function',
    'args.json',
    'results.json'
]);
const result = pythonProcess.stdout?.toString()?.trim();
const error = pythonProcess.stderr?.toString()?.trim();

const status = result === 'OK';
if (status) {
    const buffer = await readFile('results.json');
    const resultParsed = JSON.parse(buffer?.toString());
    let cat = ''
    if(resultParsed[0] == 0) cat = 'Límite'
    else if(resultParsed[0] == 1) cat = 'Alto'
    else if(resultParsed[0] == 2) cat = 'Bajo'
    else cat = 'Medio'
    //console.log(cat)
    var adding = await client.submit("g.V('59559').property('risk_c',category)",{
        category:cat,
    });
    res.send(adding)
} else {
    console.log(error)
    res.send(JSON.stringify({ status: 500, message: 'Server error' })))
}
client.close();

```

```

import sys
import json
import pickle

def first_function():
    model_pkl_file = "cardiovascular_risk_model.pkl"
    with open(model_pkl_file, 'rb') as file:
        model = pickle.load(file)

    json_obj = open(sys.argv[2])
    data = json.load(json_obj)

    cols = ['talla', 'cm_cintura', 'porcentaje_grasa', 'presion_sistolica',
            'presion_diastolica', 'glicemia_venosa', 'colesterol_ldl',
            'colesterol_hdl', 'colesterol_total_venoso', 'trigliceridos_venoso',
            'ind_sexo', 'fuma', 'diabetes_per', 'IMC', 'edades']
    data_array = []
    for i in cols:
        data_array.append(data[i])
    data_array.append(1)

    y_predict = model.predict([data_array])

    json_object_result = json.dumps(y_predict.tolist(), indent=4)
    #json_object_result = json.dumps(data_array)

    with open(sys.argv[3], "w") as outfile:
        outfile.write(json_object_result)
    print("OK")

if sys.argv[1] == 'first_function':
    first_function()

sys.stdout.flush()

```

Imagen 5 y 6: A la izquierda se muestra el código en NodeJS de la API que llama a la función de Python en la imagen a la derecha para evaluar el algoritmo ML de riesgo cardiovascular, en este caso para el usuario de ID 59559, utilizado para probar su funcionamiento.

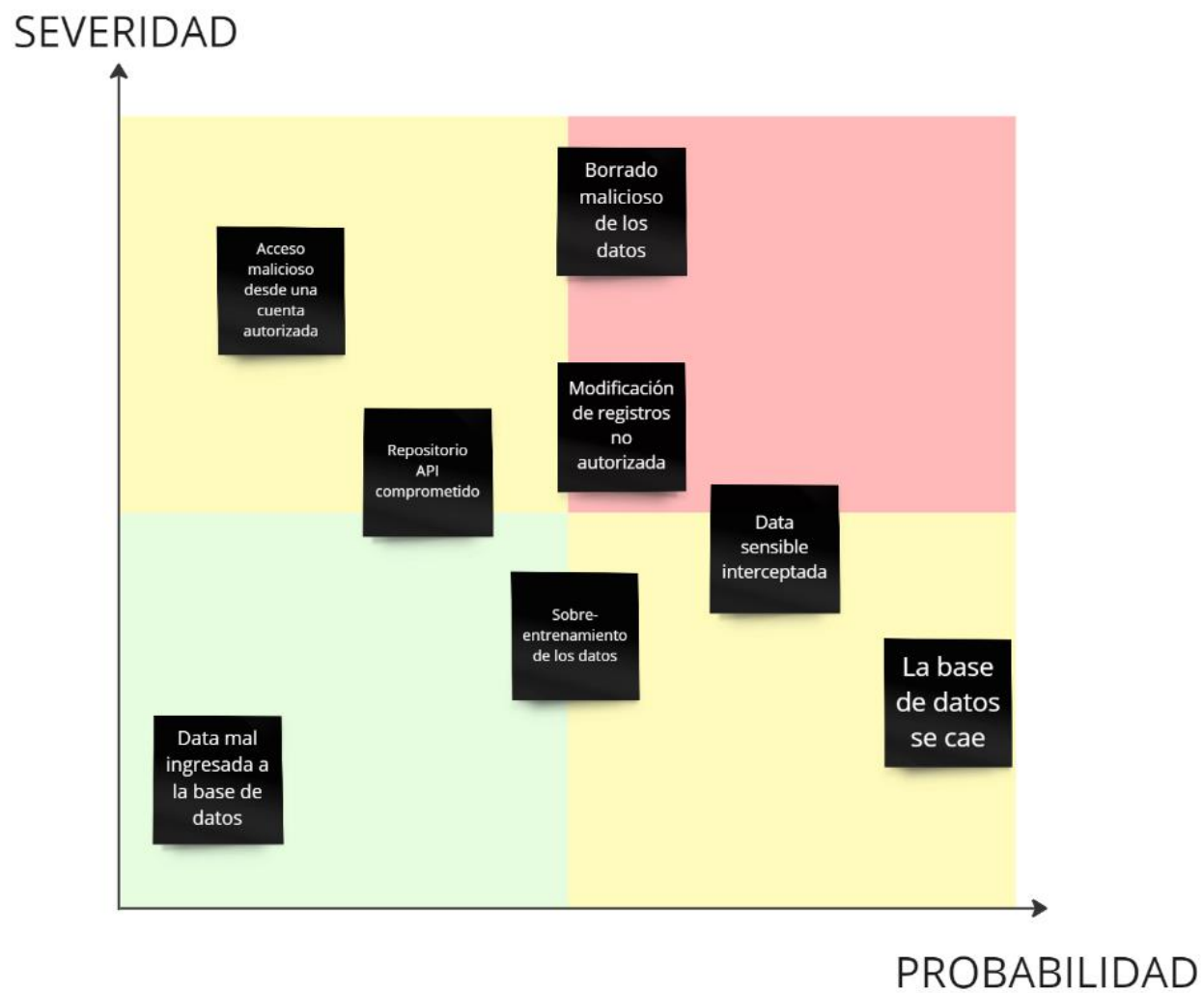


Imagen 7: Matriz de Riesgos, clasificados por severidad y probabilidad. No se considera la caída de la plataforma web en sí, pues no afecta a la BDD y es por tanto parte del proyecto macro.

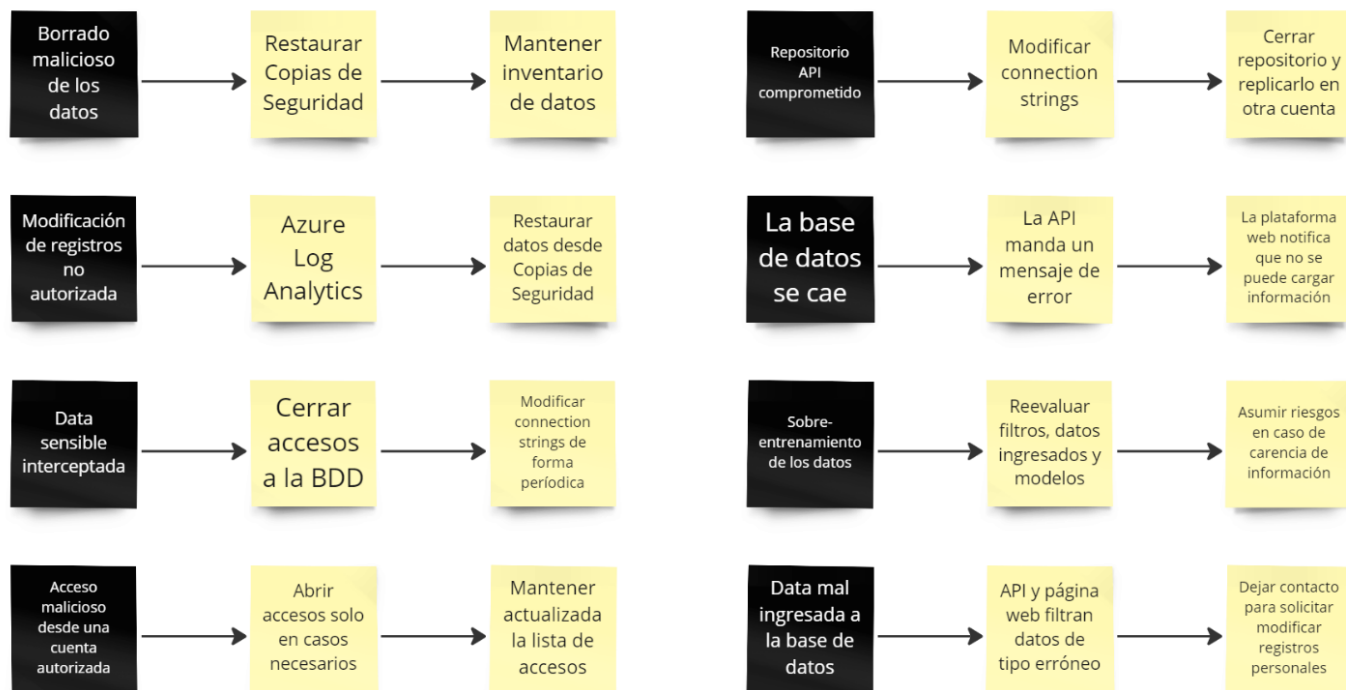


Imagen 8: Mitigación de riesgos. Muestra los riesgos (cuadros negros) seguidos de pasos resumidos para mitigarlos (cuadros amarillos). Los “connection strings” mencionados refieren a los códigos utilizados para conectarse a la BDD de grafos desde la API.

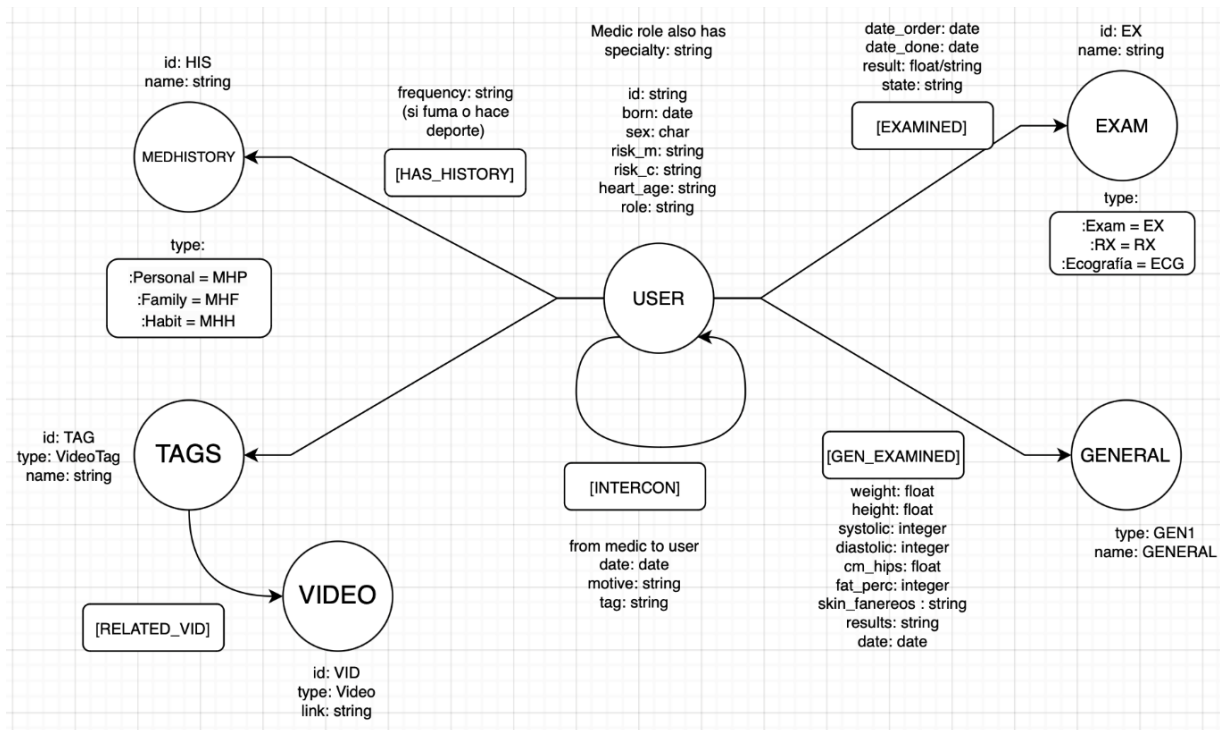


Imagen 9: Diseño final de BDD de Grafos en CosmosDB, fabricado en draw.io. Como CosmosDB no permite poner subcategorías a las etiquetas, se definieron en una propiedad "type", la cual se utiliza internamente para separar los datos en servidores distintos. Las propiedades "risk_m" y "risk_c" en el vértice USER corresponden a las variables a calcular en el proyecto.

APACHE GREMLIN API

Home Graph x

DATA

AvanteDB

Scale

AvanteGraph

Graph

Settings

Stored Procedures

User Defined Functions

Triggers

NOTEBOOKS

Notebooks is currently not available. We are working on it.

g.V()

Execute Gremlin Query x

JSON Graph Query Stats

Results

112887

TAG1

TAG2

TAG3

TAG4

TAG5

TAG7

TAG8

TAG6

TAG9

Graph

Properties

id 112887

label USER

type M

birth 1967-10-27

paternal_surname 112887-AP_PATERNO

maternal_surname 112887-AP_MATERNO

first_name 112887-PRIMER_NOMBRE

second_name 112887-SEGUNDO_NOMBRE

Sources

Imagen 10: Evidencia de la existencia de la BDD de grafos en Azure CosmosDB.

✓ Árbol de Decisión para Riesgo de Síndrome Metabólico

```
[109] data_array=V.values
      considered = V.drop(['sind_met', 'talla', 'TRI/HDL'], axis=1, inplace=False).astype(float)
      y = V['sind_met']
      X_train, X_test, y_train, y_test = train_test_split(considered, y, test_size=0.2, random_state=10)

[110] clf = tree.DecisionTreeClassifier()
      clf = clf.fit(X_train, y_train)
      y_pred = clf.predict(X_test)

[111] print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
      print(metrics.confusion_matrix(y_test, y_pred))
      print("F1-Score:", metrics.f1_score(y_test, y_pred, average='weighted'))

... Accuracy: 0.8449525934188511
[[784  71   4   1   0   0]
 [ 76 469  39   3   0   0]
 [   7  38 219  14   0   0]
 [   0   4  16  35   1   1]
 [   0   0   1   0   8   2]
 [   0   0   0   0   0   0]]
F1-Score: 0.8452357906573055
```

Imagen 11: Evidencia del Árbol de Decisión logrado para Síndrome Metabólico. Utiliza 8,967 registros y 34 variables para clasificar. Sin embargo, la clasificación oficial de los datos aún no se concreta, por lo que se espera aplicarle cambios antes de instaurarlo en la plataforma web.

```

Árbol de Decisión para Riesgo Cardiovascular a 10 años con fórmula de Tablas

y = mltabled['risk_cat']
X_train, X_test, y_train, y_test = train_test_split(considered, y, test_size=0.15, random_state=2)

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

[195] ✓ 0.0s Python

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print(metrics.confusion_matrix(y_test, y_pred))
print("F1-Score:", metrics.f1_score(y_test, y_pred, average='weighted'))

[213] ✓ 0.0s Python

... Accuracy: 0.8440860215053764
[[ 7  2  8  1]
 [ 2  1  1  0]
 [ 6  0 143  3]
 [ 3  1  2  6]]
F1-Score: 0.8419936237773049

model_pkl_file = "cardiovascular_risk_model.pkl"
with open(model_pkl_file, 'wb') as file:
    pickle.dump(clf, file)

[178] ✓ 0.0s Python

```

Imagen 12: Evidencia del Árbol de Decisión logrado para Riesgo Cardiovascular según tablas de clasificación modificadas por los doctores a cargo del proyecto. Utiliza 1,238 registros y 16 variables para hacer el cálculo. Al ya estar concretado, en la última celda se guarda el modelo como archivo "pkl".

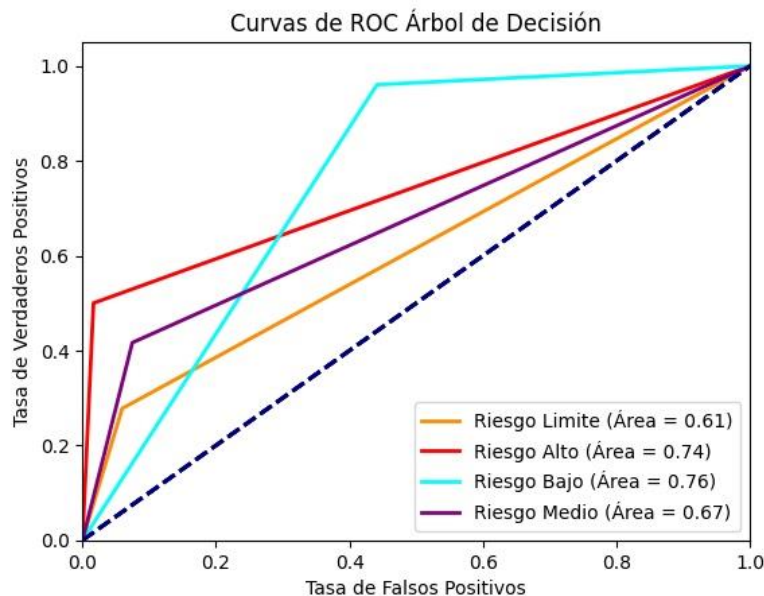



Imagen 13: Curvas de ROC por categoría del Árbol de Decisión formado para predicción de riesgo cardiovascular. El área indicada junto a sus etiquetas indica el porcentaje de cada categoría que es predicho correctamente. El resultado ideal (que se espera tener a futuro) es que el área sea mayor a 0.8 en cada categoría.

Sanidad Naval

Cerrar sesión



Maximiliano Pérez, 34

Rut: 59559

Sex: Male

Antecedentes Personales

Indicar

Especialidad

Indicar

Salud ocupacional

Indicar

Repartición

Indicar

Última evaluación

Plan de control

Próximo control

Último control

APTO

Semestral

04 / 09 / 24

04 / 06 / 24 (2 meses)

Síndrome Metabólico

C/ ALTERACIÓN

Última actualización 2 meses

Riesgo Cardiovascular

5.8 % BAJO

Última actualización 2 meses

Indicadores del paciente

Antropométricos

De Riesgo

IMC: SALUDABLE

Peso: 60.3kg

Última medida 63.5kg / -3.20kg

Talla: 1.57m

Última medida 1.58m / -0.01m

% de grasa: 27.4%

Última medida 31.2% / -3.80%

Cintura: 77.0cm

Última medida 82.0cm / -5.00cm

Presión Arterial 140.0/90.0

Última medida 137.0/71.0 / +13.67

Antecedentes del paciente

Estadísticas

Ord. Exámenes

Interconsultas

Última evaluación (15)

Fecha 2023/12/04

Hemograma

Perfil Hepático

Orina Completa

Creatinina

Microalbuminuria aislada

T4

TSH

Glicemia

Perfil Lipídico

Uremia

Urocultivo

Microalbuminuria

T3

T4 Libre

Hemorragia Oculta en Deposiciones

Última evaluación (18)

Fecha 2023/12/05

Hemograma

Perfil Hepático

Orina Completa

Creatinina

Microalbuminuria aislada

T4

TSH

Hepatitis B y C

VIH

Perfil Lipídico

Uremia

Urocultivo

Microalbuminuria

T3

T4 Libre

Hemorragia Oculta en Deposiciones

VDLR

Glicemia

Imagen 14: Perfil de usuario Avante. Importante es aclarar que el nombre mostrado es de prueba y el RUT presenta el ID del usuario, dado que esta data real no se tiene por temas de seguridad de Avante. Los “Indicadores del paciente” sacan datos de la BDD de grafos y comparan la última medición con la anterior del usuario. Las “Órdenes de Exámenes” son creadas en la página y guardadas en la BDD de grafos como relaciones a exámenes con propiedad “pendiente”. La visualización de síndrome metabólico y riesgo cardiovascular no se encuentran del todo instauradas y esta última podría finalmente no mostrar el porcentaje.

31