Informe de Avance RF1

Datacript Equipo 1 26/09/2023

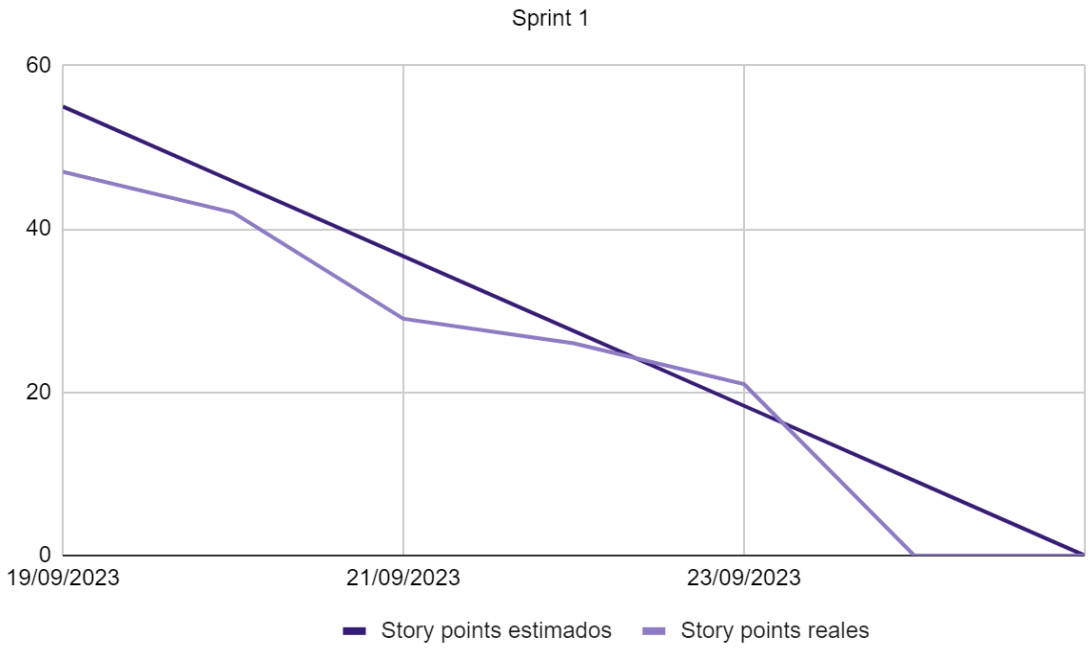
Integrantes

Apellido y Nombre	Rol
Dikenstein, Leandro	Product Owner
Perez, Giannina	Scrum Master
Prieto, Lucas	Líder Técnico
Benitez, Micaela	Desarrolladora
Benitez, Yamila	Desarrolladora
Clauser, Nahuel	Desarrollador
Torrico, Franco	Desarrollador
Gómez, Federico	Tester

Funcionalidad Comprometida

Requerimientos prometidos	Completo	Responsable
Arquitectura del Sistema	Si	Prieto, Lucas (LT)
API Flask para pruebas	Si	Prieto, Lucas (LT)
Datos Maestros(MDM)	No	Perez Giannina (SM)
Base de datos	Si	Clauser, Nahuel (Dev)
Encriptación	Si	Torrico, Franco (Dev)

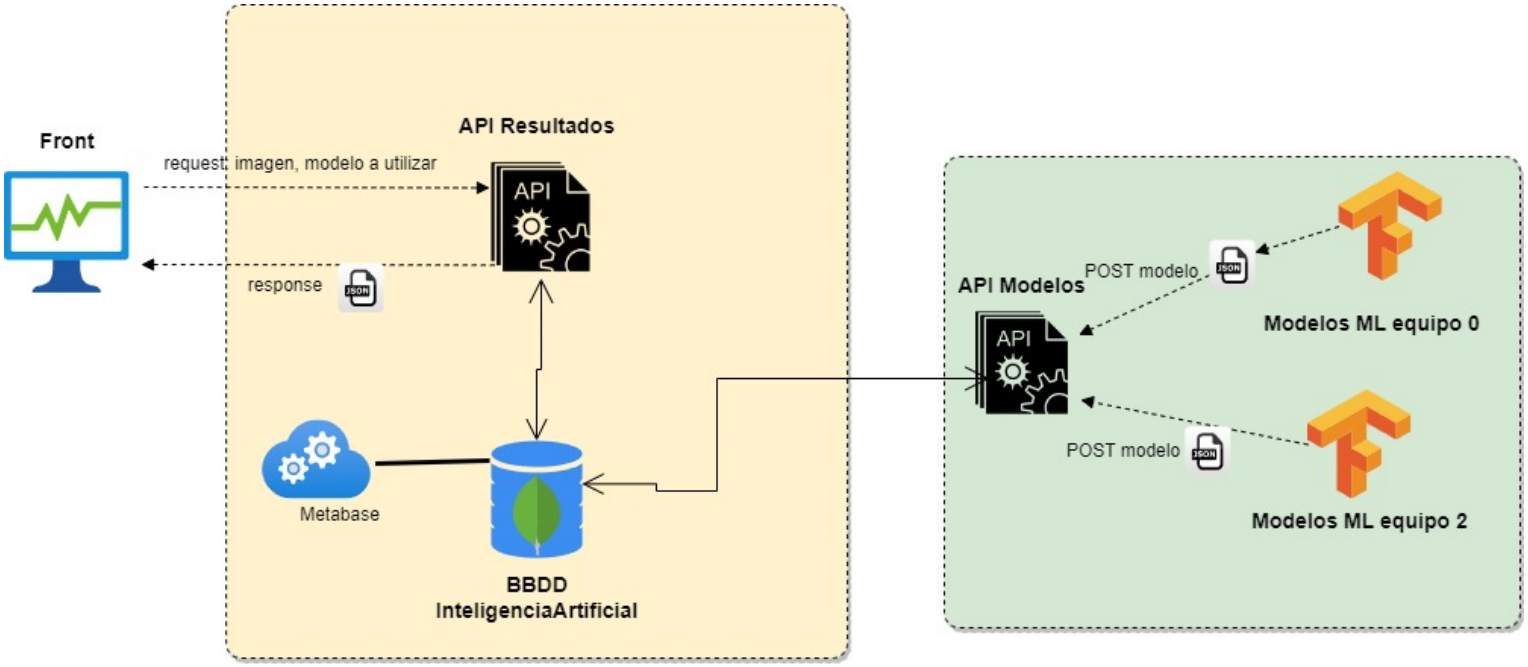
Burndown Chart



Actividades realizadas

1. Arquitectura del Sistema

Pensamos una primera versión de la arquitectura que después verificamos con los Líderes Técnicos de los demás equipos. Esta primera versión la pensamos enfocada en la funcionalidad principal más básica, que sería ingresar una imagen y recibir la respuesta del modelo. Es por esto que, como se verá más adelante en el documento, esta arquitectura se modificó y se mejoró teniendo en cuenta más funcionalidades y detalles.



Lo que planteamos con esta primera versión es una interacción entre todos los equipos. Los equipos 0 y 2 desarrollan por su lado los modelos y una vez que los tienen, los almacenan en una base de datos a través de una API (Modelos). Con esto lo que logramos es tener en la base de datos siempre los modelos en su última versión. Una vez que los modelos están disponibles en la base de datos, al llegar una petición desde el front, la API de Resultados es la que se encarga de levantar de la base el modelo a utilizar, lo ejecuta y devuelve la respuesta.
Los problemas que tenemos con esta primera versión, en primer lugar, es que falta definición de los requisitos del sistema. Encontramos que falta adaptación a otros requerimientos, como por ejemplo, la gestión de entidades. Además, asumir la

responsabilidad de la API de resultados podría generar una sobrecarga de trabajo y una mayor responsabilidad para nuestro equipo, ya que estaríamos encargados de ejecutar los modelos desarrollados por otros equipos.

Por estos motivos, hemos decidido abordar un enfoque más detallado para la arquitectura y distribuir las responsabilidades de manera más equitativa.

2. API Flask para pruebas

A través de conversaciones con el grupo 3, acordamos en hacer una API de pruebas que nos serviría tanto para nosotros que estamos empezando a conocer y practicar la tecnología de Flask, como para ellos, que necesitaban capacitar a sus desarrolladores en la conexión desde Angular a un backend.
Esta API no nos llevó mucho tiempo de desarrollo y nos permitió capacitarnos y aprender sobre Flask que es una tecnología que seguramente utilicemos en el proyecto.
Investigamos también sobre sitios para alojar servicios como esta API pero la mayoría eran pagos o con créditos gratis por poco tiempo, como por ejemplo el caso de Heroku o Google Cloud. Una alternativa que encontramos interesante, y fue la que elegimos para este caso, es **Render**. Render es una nube gratuita para alojar servicios, bases de datos, y más. Permite subir el código directamente desde GitHub lo que hace muy fácil el despliegue y también la actualización, ya que al actualizar GitHub se realiza automáticamente el despliegue con los nuevos cambios.
Para generar la documentación de la API utilizamos una extensión de Flask que se llama **Flask-Restx**, la cual automáticamente al levantar el proyecto arma la documentación en **Swagger**.
La api se puede ver en el siguiente enlace: https://api-resultados.onrender.com/

3. Datos Maestros(MDM)

Durante estos días, nos dedicamos a investigar sobre Datos Maestros y cómo llevar una gestión de ellos. Gracias a esto pudimos entender que los Datos Maestros son aquellos datos centrales que cumplen una misión fundamental para el correcto

funcionamiento de los sistemas. Estos datos actúan como la fuente principal de información utilizada por múltiples sistemas y aplicaciones.

Si bien aún nos quedan por definir más, pudimos identificar algunos datos maestros dentro de nuestro proyecto.

- **Profesionales Médicos:** El uso del sistema está enfocado en profesionales de la salud, es por esto que una de las funcionalidades será registrar a los médicos que lo utilicen. La información que se solicite de los médicos serán datos cruciales para el sistema, ya que ellos serán los principales usuarios del mismo.
- **Establecimientos:** La información sobre los establecimientos donde trabajan los profesionales que utilicen el sistema.
- **Modelos:** Los modelos son una parte primordial del sistema, son los que realizan las clasificaciones y ayudan a los médicos en los diagnósticos con sus resultados y predicciones. Consideramos que la información sobre estos modelos son datos importantes y centrales a tener en cuenta.

A pesar de que aún faltan definir datos maestros, comenzamos a pensar en un proceso de gestión, en el que los siguientes pasos son:

- **Identificación de Datos Maestros faltantes:** Una vez que terminemos de definir las funcionalidades del sistema con el resto de equipos, identificaremos el resto de entidades, atributos y relaciones de datos críticos para el proyecto.
- **Creación de un Repositorio Centralizado:** Establecer un lugar centralizado para almacenar los datos maestros, como una base de datos.
- **Proceso de Ingreso de Datos Inicial:** Definir un proceso básico para ingresar datos maestros iniciales en el sistema, con un enfoque en la captura de nuevos registros.
- **Validación de Datos:** Implementar reglas de validación de datos para garantizar la calidad y consistencia de los datos ingresados.

4. Base de Datos

En un principio teníamos pensado utilizar una base de datos no relacional, tal como lo es MongoDB. Sin embargo, nos encontramos con la dificultad de que ningún integrante del equipo lo utilizó anteriormente, además de las siguientes cuestiones:

- **Grandes gastos generales de almacenamiento:** El almacenamiento de datos de imágenes binarias en MongoDB puede generar una sobrecarga de almacenamiento significativa debido a la serialización BSON (Binary JSON).
- **Capacidades limitadas de consulta e indexación:** las capacidades de consulta e indexación de MongoDB están diseñadas principalmente para datos estructurados. Si bien puede almacenar imágenes como datos binarios, no

podrá realizar consultas complejas basadas en imágenes ni indexar funciones de imágenes de manera eficiente.

Debido a esto, decidimos tener en cuenta Postgres, pues fue la recomendación de nuestro PO. Podría ser la opción adecuada para almacenar datos de imágenes para el Machine learning, especialmente cuando se necesita una gestión de metadatos estructurada y capacidades de queries avanzadas.

Sin embargo, aún nos queda el inconveniente sobre la eficiencia de Postgres en cuanto a la búsqueda de imágenes, pues no se destaca en el almacenamiento de imágenes. Con respecto a esto, hemos optado por tener en cuenta las soluciones de almacenamiento de imágenes dedicadas (por ejemplo, almacenamiento de objetos). De esta manera, en la base de datos de Postgres almacenaremos metadatos asociados a las imágenes para que sean referenciadas. Este enfoque puede ayudar a mitigar algunas de las limitaciones del uso de una base de datos relacional para el almacenamiento de imágenes.

5. Encriptación

La encriptación en la gestión de datos es un tema fundamental para garantizar la seguridad y privacidad de la información. Comprendiendo la seguridad de la información como acercamiento en el marco del ONTI (Oficina Nacional de Tecnologías de Información). En primera instancia, se presentan conceptos clave en el ámbito de la seguridad de la información “triada CID” como confidencialidad, integración y disponibilidad.

- **Confidencialidad:** Se refiere a la protección de datos y la restricción de acceso solo a las personas o entidades autorizadas (la encriptación orientada a fortalecer la confidencialidad).
- **Integridad:** La integridad garantiza que la información sea precisa, confiable y consistente. Los datos deben permanecer inalterados durante la transferencia.
- **Disponibilidad:** La disponibilidad garantiza que la información esté disponible a las personas (planes de recuperación o contingencia orientados a proteger la disponibilidad, como copias de respaldo y software actualizados).

La encriptación será parte fundamental para proteger datos sensibles a definir en nuestro producto, convierte un texto normal en un lenguaje codificado que sólo alguien con la clave adecuada puede descifrar. Se utiliza para asegurar la comunicación, proteger la información sensible y evitar la filtración de datos.
En nuestro sistema de administración de claves criptográficas estaremos usando técnicas de clave secreta (criptografía simétrica), cuando dos o más actores comparten

la misma clave y ésta se utiliza tanto para cifrar información como para descifrar la misma.

La encriptación la llevaremos a cabo con el algoritmo simétrico AES (Advanced Encryption Standard), es un estándar de cifrado simétrico ampliamente utilizado para proteger datos confidenciales. AES se utiliza para cifrar datos, esto garantiza que solo las personas autorizadas puedan acceder y comprender la información. Las claves AES son claves simétricas que pueden tener tres longitudes de clave diferentes (128, 192 o 256 bits). Es el estándar de cifrado reconocido y recomendado actualmente por el gobierno de EE.UU en cuanto a cumplimientos normativos. Contribuye a la seguridad en línea, los servicios de almacenamiento en la nube suelen utilizar AES para cifrar los datos que se almacenan en sus servidores.

Cambios

El primer cambio que acordamos con el resto de equipos es que los modelos los ejecuten los equipos encargados de su desarrollo. Nuestro equipo recibirá la petición y se la enviará al modelo para que realice la clasificación y retorne una respuesta.
También se empezó a planificar la funcionalidad de Log in, donde se definió que los profesionales se identificarán con su DNI y un email, para luego enviar su contraseña para poder utilizar el sistema.

Propuesta próximo ciclo

Durante nuestro siguiente Sprint, vamos a estar definiendo con los demás equipos los datos maestros faltantes, pues es esencial que primeramente se lleven a cabo las entrevistas con los profesionales médicos para conocer mejor las necesidades de los mismos.
Una vez definidas estas cuestiones, comenzaremos a estructurar nuestra base de datos, en primera instancia definiendo las tablas y registros que serán esenciales para cumplir el objetivo de nuestro proyecto.

A la par, en nuestro equipo vamos a comenzar a capacitarnos para poder realizar la estructura de los datos maestros. Primeramente diseñaremos una arquitectura de estos datos, la cual utilizaremos como base para comenzar la construcción de los datos maestros. Previo a dicha construcción, también investigaremos sobre qué tecnología es la más adecuada a nuestras necesidades para utilizar en este paso.

Por último, también nos vamos a encargar de definir nuestra estrategia de encriptación, tomando como base las investigaciones que realizamos durante este primer Sprint.