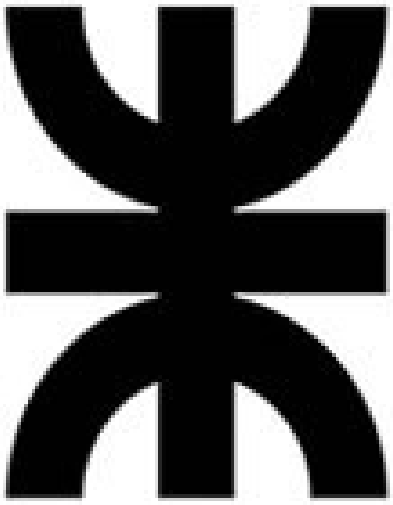


# UTN - FRC - Ing. en Sistemas de Información

# PROYECTO FINAL - ESTUDIO INICIAL

## Curso 5K1 – AÑO 2022

**DOCENTES**

- Ing. Cecilia Ortiz
- Ing. Ma. Irene Mac William
- Ing. Lorena Barale

**INTEGRANTES:**



| LEGAJO | APELLIDO, NOMBRE | CONTACTO |
|---|---|---|
| 65144 | Diaz, Nahuel | nahuel.diazz.nd@gmail.com \| 3516521633 |
| 65503 | Santamarina, Manuel | manu.santamarina95@gmail.com \| 2302467734 |
| 75950 | Piemonte, Juan Pablo | jppiemonte@gmail.com \| 3512190772 |
| 79557 | Madrid Doña, Roberto | roberto.madridd@gmail.com \| 3513168343 |
| 79558 | Quinteros, Manuel | manuquinteros06@gmail.com \| 3517494724 |

# ÍNDICE

# HISTORIAL DE REVISIÓN

| Fecha | Versión | Descripción | Autor/es |
|---|---|---|---|
| 02/05/2022 | 1.0 | Creación inicial del documento con la estructura y detalle de cada uno de los puntos principales. Se incluye descripción del sprint 0 del proyecto (definiciones y actividades iniciales) | Diaz, Nahuel<br>Santamarina, Manuel<br>Piemonte, Juan Pablo<br>Madrid Doña, Roberto<br>Quinteros, Manuel |

# INTRODUCCIÓN

El objetivo principal de este documento es generar un informe de los avances realizados en cada sprint de trabajo del proyecto AGRAR.IO, desarrollado como Proyecto Final de la carrera de Ing. en Sistemas de Información en la UTN FRC durante el ciclo lectivo 2022 por nuestro grupo de trabajo. Como primer apartado tendremos las definiciones propuestas y adoptadas por el equipo para todo este proyecto en lo que denominamos SPRINT 0 del proyecto, el cual tiene como fin realizar todos los acuerdos y definiciones internas con respecto al proceso de desarrollo, permitiendo una mejor organización del grupo y de las tareas a realizar, como así también mantener un funcionamiento ágil, aunando el esfuerzo realizado por cada miembro del equipo.

Posteriormente, se agregan a este documento los resúmenes de las actividades realizadas separadas en cada uno de los Sprints pertenecientes al proyecto. En cada apartado perteneciente a los sprints del proyecto se especifican trabajos realizados, métricas del sprint, resumen de las actividades de desarrollo y documentación, redefiniciones y cualquier otro aspecto importante de destacar con respecto a lo perteneciente al sprint.

# DEFINICIÓN DE PRODUCT OWNER

La metodología de trabajo elegida dispone que se cuente con la participación de un Product Owner (PO), quien es el responsable de definir qué productos formarán parte del Product Backlog, comprendiendo las necesidades de los usuarios dentro del negocio.

Por motivos de conocimiento del dominio y facilidad de comunicación con nuestro proveedor de herramientas electrónicas (ya que es nuestro contacto directo con el Ing. Electrónico que fabrica los sensores que utilizaremos), optamos por definir como Product Owner a Manuel Santamarina, miembro del equipo de desarrollo de este proyecto. Él es el encargado de realizar el planteo de los requerimientos a satisfacer con el sistema fabricado, entendiendo las necesidades tanto del modelo de negocio como de los usuarios del sistema entre otras tareas propias de un PO dentro de un equipo de desarrollo ágil.

# DEFINICIÓN DEL EQUIPO Y SCRUM MASTER

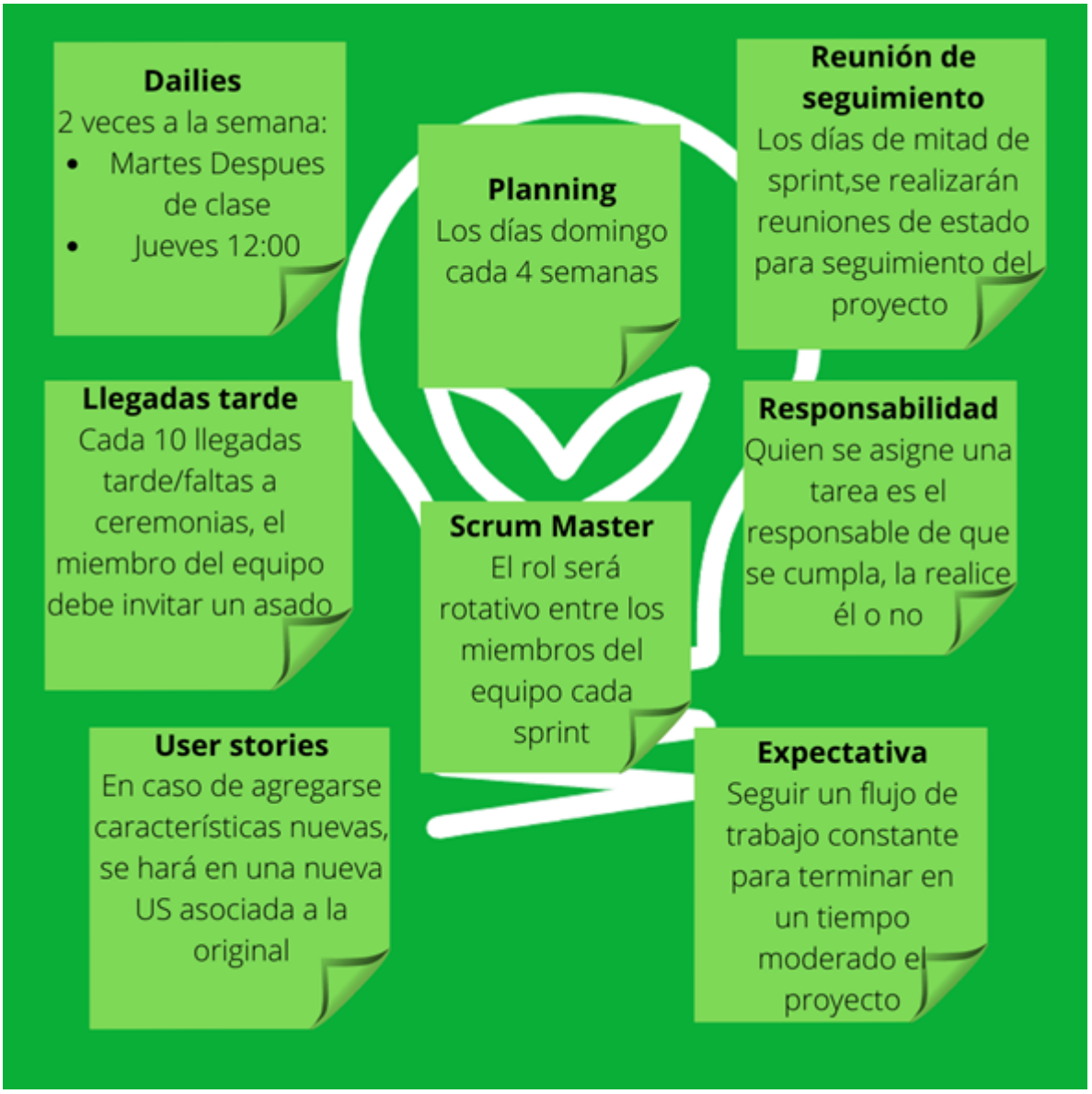
El equipo de trabajo ágil de AGRAR.IO está conformado por 5 integrantes:

- Manuel Santamarina
- Manuel Quinteros
- Nahuel Díaz
- Juan Pablo Piemonte
- Roberto Madrid Doña

Por definición del mismo equipo, se ha acordado que el rol de Scrum Master será rotativo cada sprint entre los miembros del equipo. Quien asume este rol cada sprint es el encargado de liderar al equipo de trabajo en las metodologías ágiles, Scrum específicamente, y garantizar que los miembros se enfoquen en los principios de este marco de trabajo. Es el encargado de convocar a las ceremonias, de hacer que se cumplan y de tener cierta responsabilidad sobre el funcionamiento interno del equipo, optando siempre por las decisiones que permitan el desarrollo de procesos lo más ágiles y dinámicos posibles.

# ACUERDOS

Teniendo siempre el objetivo común de garantizar la eficiencia y calidad de trabajo dentro del equipo, entre los miembros del mismo se determinan ciertos acuerdos a ser tenidos en cuenta durante el desarrollo del proyecto:



# DEFINICIÓN DEL PRODUCT BACKLOG INICIAL

A continuación detallamos el listado de Épicas e Historias de Usuarios que utilizaremos para construir la versión base del producto o MVP. Las mismas se incluirán desde un inicio en nuestro Product Backlog, el cual es un listado de todas las tareas que se pretenden hacer durante el desarrollo del proyecto, con el fin de que cada miembro del equipo de trabajo pueda tener un panorama general de la cantidad de actividades que restan por hacer para finalizar el proyecto.
Por otro lado, haciendo referencia a las Historias de Usuario o User Stories, podemos decir que son las definiciones generales e informales de lo que el usuario final, o cliente del sistema, espera obtener del proceso de desarrollo. Su propósito es articular cómo un elemento de trabajo puede aportar un valor particular al usuario final. Asimismo, las Épicas son agrupaciones de Historias de Usuario que tienen como objetivo nuclear una funcionalidad más grande y más compleja, que puede subdividirse en unidades más pequeñas con el fin de simplificar su análisis y desarrollo.

## Framework

El objetivo de esta épica es construir una primera versión simplificada de la estructura que dará origen a la plataforma integral. Está compuesta por las siguientes Historias de Usuario:

- Menú lateral
- Log in a la plataforma
- Log out a la plataforma
- Enviar email
- Emitir notificación push
- Crear Usuario
- Editar Usuario
- Barra de Estado de la Plataforma

## Gestionar campos

El objetivo de esta épica es construir el conjunto de funcionalidades que permitirán al usuario configurar la plataforma integral. Está compuesta por las siguientes Historias de Usuario:

- Crear Coordenadas y Forma del Terreno
- Editar Coordenadas y Forma del Terreno
- Registrar Desperfecto en Dispositivo o Plataforma Web
- Actualizar ubicación de recursos agropecuarios
- Registrar campos

## Gestionar colaboradores

El objetivo de esta épica es agrupar el conjunto de funcionalidades que permitan al productor gestionar los colaboradores de los distintos campos. Está compuesta por las siguientes Historias de Usuario:

- Visualizar colaboradores

- Registrar colaboradores
- Dar de baja colaboradores
- Visualizar campos y estadísticas

## Contratación de Servicios

El objetivo de esta épica es agrupar el conjunto de funcionalidades que permitan al productor gestionar los colaboradores de los distintos campos. Está compuesta por las siguientes Historias de Usuario:

- Registrar contratación de servicio de instalación
- Pausar suscripción
- Dar de baja servicio contratado
- Reactivar suscripción
- Visualizar lista de instancias de servicios contratados
- Visualizar servicios de instalación contratados

## Servicio de Telemetría para Silobolsa

El objetivo de esta épica es construir una primera versión funcional del servicio que permitirá visualizar los datos de sensado de una silobolsa. Está compuesta por las siguientes Historias de Usuario:

- Contratar servicio de telemetría de silosbolsa
- Asociar sensor de silobolsa con un campo
- Visualizar datos sensados para un silobolsa
- Registrar límites de sensado en los silosbolsa para emitir alertas
- Visualizar estado del silobolsa

## Servicio de Telemetría para Niveles de Agua

El objetivo de esta épica es construir una primera versión funcional del servicio que permitirá visualizar los datos de sensado de los niveles de agua de cualquier reservorio (tanque, bebedero, pileta, etc). Está compuesta por las siguientes Historias de Usuario:

- Contratar servicio de telemetría de niveles de agua
- Asociar sensor de nivel de agua con un campo
- Visualizar datos sensados para un tanque de agua
- Registrar límites de sensado en los tanques de agua para emitir alertas
- Visualizar estado del tanque de agua

## Alertas y Notificaciones

El objetivo de esta épica es definir todas las alertas, notificaciones y emails que el sistema debe mandar. Está compuesta por las siguientes Historias de Usuario:

- Configurar Notificaciones y Alertas
- Emitir Notificación por Servicio Contratado
- Emitir Notificación por Servicio Instalado y Funcionando

- Emitir Notificación por Alerta en Valores Límite de Sensado
- Emitir Notificación por Cierre de Facturación
- Emitir Notificación por Vencimiento de Facturación

## Facturación y Pagos

El objetivo de esta épica es construir una versión simple del facturador y registro de pagos. Está compuesta por las siguientes Historias de Usuario:

- Calcular Total y  Detalle a Facturar
- Registrar Pago de instalación
- Registrar Pago de suscripción
- Definir Precios
- Definir Promociones y Políticas de Precios
- Agregar Medio de Pago
- Registrar baja de pago de suscripción

## Dashboard y Analíticas

El objetivo de esta épica es construir una versión simple del facturador y registro de pagos. Está compuesta por las siguientes Historias de Usuario:

- Visualizar Históricos de Datos Sensados
- Visualizar Todos los Servicios de Sensado en una Pantalla
- Visualizar estadísticas e información para la gestión inteligente del campo

## Infraestructura

Esta épica está compuesta por Spikes que permitirán reducir la incertidumbre de cuestiones relacionadas a la infraestructura y su implementación. Está compuesta por los siguientes spikes:

- Crear Repositorio
- Crear Instancia de Cloud SQL
- Crear Instancia de Compute Engine
- Crear Pipeline de Frontend
- Crear Pipeline de Microservicio
- Documentar Proceso Para Levantar Un Nuevo Entorno Completo
- Integrar Cloud Logging
- Integrar Secret Manager

## Diseño de Arquitectura

El objetivo de esta épica es construir la arquitectura y entregar valor al PO mediante diagramas y vistas del sistema. Está compuesta por las siguientes Historias de Usuario:

- Crear Vista de Componentes de Hardware
- Crear Vista de Componentes de Lógicos

- Crear Vista de Componentes del Frontend

Además contiene los siguientes Spikes:

- Investigar Servicios de GCloud Para la Ingesta de Datos de Sensores.
- Crear api-gateway
- Crear Template de Microservicio
- Integrar TypeORM
- Integrar BigTable
- Integrar IoT Core
- Integrar Microservicios con BigTable
- Integrar Microservicios con Iot Core

## STORY MAP

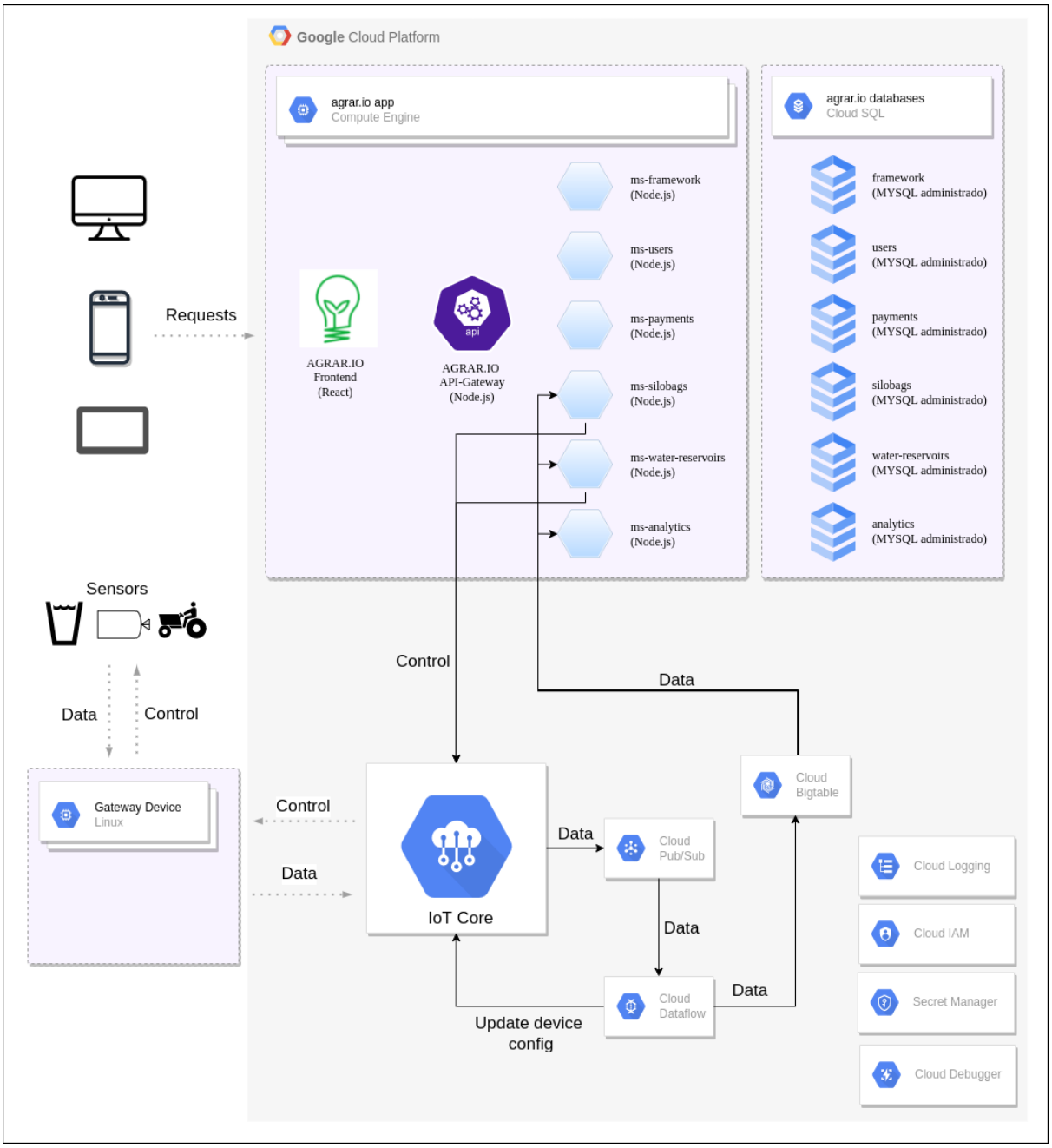
# DEFINICIÓN DE LA HERRAMIENTA DE SOFTWARE PARA LA GESTIÓN DE PROYECTO A UTILIZAR

# TEMPLATE DE LA USERS STORIES

# TÉCNICA DE ESTIMACIÓN A UTILIZAR

# DEFINICIÓN DE LA TECNOLOGÍA A UTILIZAR EN EL DESARROLLO DEL PROYECTO

El proyecto utiliza las tecnologías presentes en el siguiente diagrama:




Como primer punto es clave resaltar que la solución utiliza **Google Cloud Platform** (GCP) debido a su **flexibilidad, disponibilidad, escalabilidad** y varios servicios administrados que brinda.

Para hostear la aplicación de **Frontend**, el **API Gateway** y los **Microservicios** que constituyen el backend se utiliza **Compute Engine** el cual es un servicio de GCP que brinda máquinas virtuales ajustables a las necesidades particulares de cada cliente. Además se utiliza **React** para construir el Frontend y **Node.js** tanto para el API Gateway como los **Microservicios**.

Para las **Bases de Datos Transaccionales** se utiliza otro servicio de GCP llamado **Cloud SQL**. Este servicio totalmente administrado permite levantar bases de datos MySQL, PostgreSQL y SQL Server de una forma muy rápida y sencilla. En nuestro caso elegimos **MySQL**.

Cada uno de los dispositivos de control y/o sensado estarán conectados a un **Gateway** que estará corriendo sobre un SO **Linux**. Luego estos Gateway estarán conectados a GCP mediante otro servicio también de la nube denominado **IoT Core**. Este servicio 100% administrado nos permitirá tanto la ingesta de datos de sensado y el control sobre los dispositivos para configurarlos, ajustarlos, etc.

Además se utilizan otros servicios de GCP como **Cloud Pub/Sub** para distribuir los datos de sensado hacia los demás servicios que los requieran, **Cloud Dataflow** para hacer transformaciones optimizadas sobre datos y retroalimentar a los dispositivos y **Cloud Bigtable** para almacenar de forma persistente todos los datos de sensado.

Finalmente se utilizan servicios como **Cloud IAM** para gestionar permisos de acceso a recursos de GCP tanto para entornos de desarrollo como productivos. **Secret Manager** para almacenar credenciales de entorno. **Cloud Logging** para administrar de forma centralizada los logs que cada servicio emite y **Cloud Debugger** para monitorear y depurar el estado de cada servicio.
Pautas de codificación y testing

# PAUTAS DE CODIFICACIÓN Y TESTING

Cuando se desarrolla una aplicación o un proyecto de software y se tiene una participación importante en el mismo, ya sea con el aporte de centenares de líneas de código o simplemente con la declaración de una variable, es importante mantener cierta uniformidad con el trabajo realizado por el resto del equipo. El objetivo de esto no es solamente obtener una similaridad entre los productos de cada miembro que participa en el desarrollo, sino que en caso de necesitar sustituir a uno de los participantes del proyecto o que sus tareas sean tomadas por un compañero, éste ultimo no tenga que lidiar con la deducción de las intenciones del autor original al momento de escribir ese bloque de código, sino que la lectura sea lo más clara y comprensible a primera vista posible. Para lograr esto, el equipo optó por adoptar las reglas, convenciones y buenas prácticas especificadas en un documento utilizado por uno de nuestros miembros que trabaja como programador full-stack. Algunas propuestas que se encuentran allí fueron obtenidas de los libros Código Limpio, de Uncle Bob, y "Javascript: The Good Parts", de Douglas Crockford. Dentro de este documento, se encuentran las definiciones de las convenciones adoptadas por el equipo con respecto a identación, simbología, declaración y nomenclatura de variables, comentarios. Cabe aclarar que el archivo originalmente estuvo orientado al lenguaje JavaScript, pero como en el desarrollo de este proyecto el código escrito pertenece al lenguaje TypeScript, realizamos ciertas modificaciones al documento para mantener coherencia con lo que se puede encontrar en el código fuente de nuestro sistema. A continuación incluimos un enlace a nuestro repositorio donde pueden encontrar el archivo al que hacemos alusión, y además el mismo será incluido al final de este informe, a modo de apéndice.
https://drive.google.com/file/d/1-bTFt8JJ__Z-z7yIObgqS1g8lpqXKgNQ/view?usp=sharing

Por otro lado, en lo que a Testing respecta, el equipo optó por plantear una modalidad de testing funcional manual, aplicando metodologías de pruebas de caja blanca y caja negra, elaborando los casos de prueba (Definición Caso de Prueba: conjunto de condiciones o variables bajo las cuales se determinará si una aplicación, un sistema de software o una característica o comportamiento de estos resulta o no aceptable - https://es.wikipedia.org/wiki/Caso_de_prueba) y utilizándolos para evaluar si las funcionalidades del sistema tienen un desempeño aceptable y que la aplicación hace lo que se diseñó que haga. El testing de caja negra (metodología de testing en la que se ignora la estructura interna del proceso a probar) será utilizado como predeterminado para el testing funcional, y se incluirá en los Casos de Prueba del proyecto como metodología predilecta. Sin embargo, utilizaremos testing de caja blanca (metodología de testing en la que se conoce la estructura interna del sistema para el proceso bajo análisis) para determinadas funcionalidades específicas que requieran un refinamiento más pulido del funcionamiento del proceso a probar, y en los que necesitemos saber exactamente cómo funciona por dentro dicho proceso.

Además del testing funcional, se utilizarán metodologías de testing no funcional para evaluar el desempeño de las API's utilizadas en el proyecto, la performance de la capa de acceso a datos, y la prueba de microservicios. Su aplicación usualmente será "En Caliente", es decir, durante el proceso de

desarrollo y con fines de determinar si el producto obtenido cumple con las condiciones mínimas de calidad como para considerarlo listo para someterse a pruebas funcionales.

# GESTIÓN DE CONFIGURACIÓN

La Gestión de Configuración del Software (GCS) es un proceso ingenieril que tiene como objetivo integrar los procedimientos técnicos para controlar y mejorar la calidad del software. Busca integrar las actividades vinculadas al desarrollo desde las fases más tempranas posibles, asignando roles claros y responsabilidades al personal de trabajo, con la finalidad de obtener un producto al que sea más fácil realizar mantenimiento, más escalable y más eficiente al momento de modificar. En este proceso se utilizan varias herramientas que facilitan la administración de los recursos informáticos del proyecto, ayudando con su organización y manipulación, como repositorios GIT u otros gestores de configuración de software.

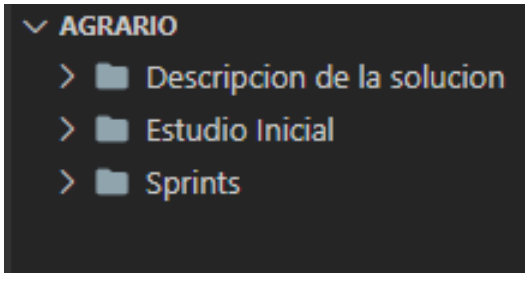
Para la gestión de configuración de nuestro proyecto optamos por las siguientes herramientas:

- Git: software esencial para el control de versiones, contando con la ventaja de que es la herramienta más utilizada y conocida por los integrantes del grupo, permitiendo así el manejo fluido de la herramienta.
- GitHub: herramienta de colaboración de codificación, con alojamiento en git, el cual nos permitirá llevar el control de versiones de cada una de las documentaciones necesarias para la descripción del proyecto

El proyecto va a ser creado como un conjunto de repositorios de la herramienta Bitbucket en donde cada repositorio corresponderá con la capa definida: FrontEnd, Microservicios, Apigateway.

| Nombre de Ítem de configuración | Regla de nombrado | Ubicación Física |
|---|---|---|
| Estudio inicial | AG_Estudio_Inicial_v<n.n> | https://github.com/nahueldiazz/AGRARIO/blob/master/Estudio%20Inicial/AG_Estudio_Inicial_v1.3.pdf |
| Sprint | AG_SPRINT<N> | |

| Sigla | Significado |
|---|---|
| AG | AGRARIO |
| <n.n> | Versión del archivo iniciando por 1.0 |
| <N> | Número de sprint iniciando por el 0 (cero) |



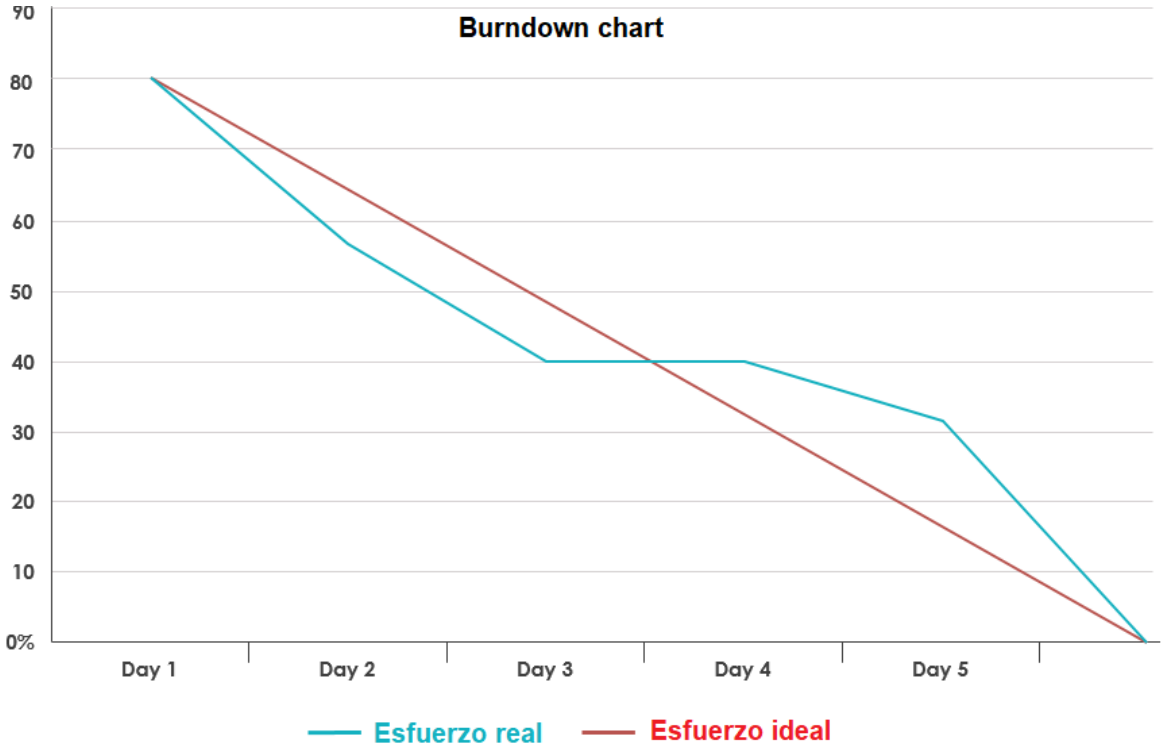
# MÉTRICAS

Dentro del marco de la Ingeniería del Software, podemos decir que una métrica es un estándar de la medida del grado en la que un aspecto específico varía de un producto, sistema, proceso o equipo de trabajo a otro. El objetivo de obtener estas medidas lo más objetivas, reproducibles y cuantificables posible es aplicar

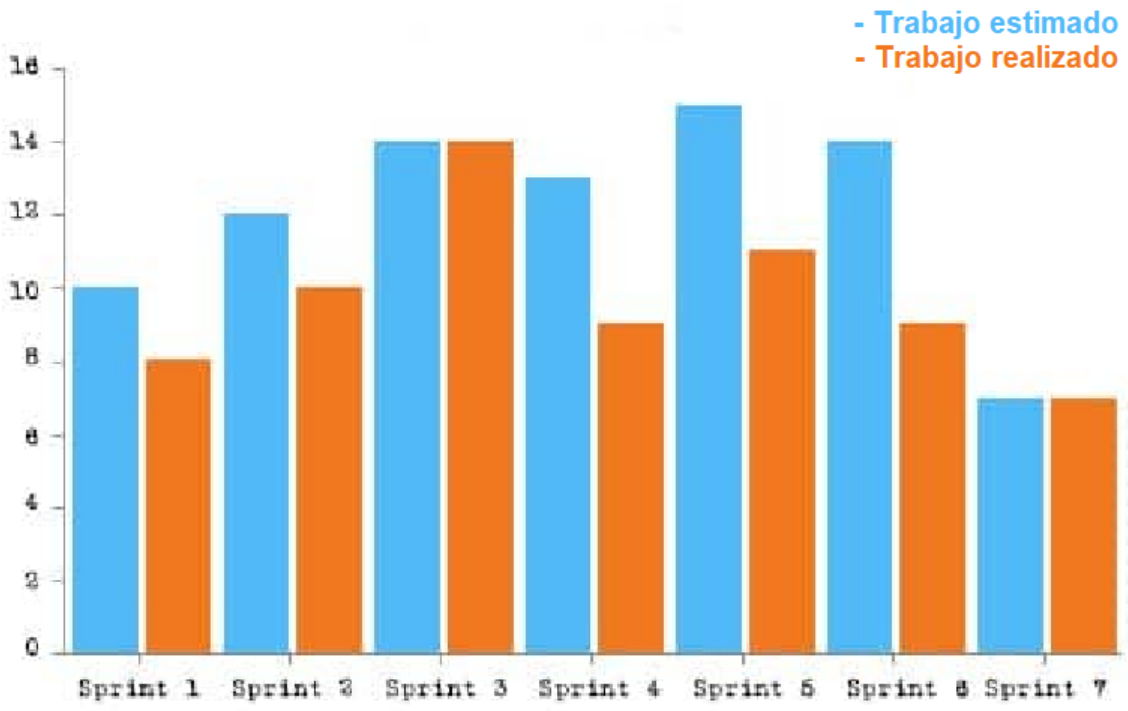
La meta, es obtener mediciones objetivas, reproducibles y cuantificables, que posibilitan tener valiosas y numerosas aplicaciones, en planificación de calendarios y presupuestos, planificación presupuestaria, aseguramiento de calidad, pruebas, depuración de software, optimización del rendimiento del software y asignaciones óptimas de tareas del personal.

Para obtener un feedback del trabajo realizado sprint a sprint hemos decidido el uso de dos métricas: burndown chart y velocity chart.

- Burndown chart es una representación gráfica del trabajo por hacer en un proyecto en el tiempo (en este caso un mes, la duración del sprint). El trabajo remanente se muestra en el eje vertical y el tiempo en el horizontal. Muestra una comparativa entre el flujo de trabajo “ideal” o constante, en contraste con cómo fue realmente en la iteración.




- Velocity chart es también una representación gráfica, donde se muestra para cada sprint la comparativa entre el trabajo estimado a realizar y el que realmente se llegó a realizar en cada iteración.




Con la confección de estas dos métricas mencionadas en cada sprint, buscamos representar de manera visual si el trabajo realizado coincide con lo que se estima con el objetivo de ajustar la forma o calidad de trabajo de ser necesario.