DISEÑO ARQUITECTONICO DE SOFTWARE PARA LA EMPRESA AYT ARENAS Y TRITURADOS.

Autor:
JAVIER LEONARDO GARZON PRADA

Director:
EDGAR ALEXIS ALBORNOZ ESPINEL
Ingeniero de Sistemas
M.Sc en computación

UNIVERSIDAD DE PAMPLONA
Facultad de Ingenierías y Arquitectura
Departamento de Ingenierías Electrónica, Eléctrica, Sistemas y Telecomunicaciones
Programa de Ingeniería de Sistemas
Pamplona, Norte de Santander – Colombia, 2021

TABLA DE CONTENIDO

LISTA DE IMÁGENES

1 **INTRODUCCION**

Los sistemas de información al interior de las entidades han sido fundamentales para el desarrollo y crecimiento de las empresas, haciendo que los procesos de flujo de información se puedan manejar, procesar y acceder de una manera rápida y eficiente, reduciendo los costos de producción, distribución y manejo del personal.

En la actualidad contar con un sistema de información dentro de una empresa da soporte para los diferentes procesos que se desarrollan dentro de esta, facilitando la comunicación y el manejo de información entre el personal de trabajo, permite tener la información en tiempos mínimos y poder distribuirla fácilmente, también permite guardar dicha información y tener registro de esta para ser accedida en cualquier momento.

Este proyecto nace de una necesidad que tiene la empresa AYT arenas y triturados la cual no cuenta con ningún sistema de información, todos sus procesos son llevados a cabo manualmente y el flujo de información se realiza de una manera rudimentaria, a raíz de que la empresa se encuentra en proceso de expansión y cada vez hay mayor flujo de información se vio necesario optar por algún método que optimizara los procesos en la empresa y pudiese tratar grandes volúmenes de información.

Se realizo un diseño arquitectónico de software como primer paso para empezar a cubrir las necesidades que tiene la empresa AYT arenas y triturados en el tratamiento de los procesos de flujo de información, en el cual se evaluaron diferentes métodos para la obtención de requerimientos y de esta manera empezar a suplir las necesidades de los trabajadores en la empresa, estos

requerimientos fueron plasmados en el modelo de diagrama de casos de uso mediante el cual se organizó la información y se estableció los usuario finales del sistema y las funciones que cada uno de ellos va a tener habilitada en el mismo, luego se procede a realizar los modelos de estos casos de uso mediante los diagramas UML de alto nivel 4+1, estos modelos de vistas ayudan a representar desde la vista lógica, de despliegue, de procesos, física, y de escenarios, la manera como el sistema se va a comportar y facilita en gran medida el proceso de implementación.

1.1 **PLANTEAMIENTO DEL PROBLEMA**

En los últimos años se ha reflejado el aumento por el uso de los sistemas de información para manejar, controlar y mejorar procesos en todo tipo de entidades, esto brinda una mejor administración de los procesos que suceden al interior de la empresa, independientemente si es grande, mediana o pequeña, gracias a la capacidad de los sistemas de información para tratar grandes volúmenes de información permite receptar y procesar información en tiempos mínimos, maximizando las áreas de trabajo y llevando un control más exacto de lo que sucede en la empresa (tiempos de trabajo, de producción, control de inventario, ventas y producción).

La empresa AYT arenas y triturados carece de la sistematización de sus procesos y flujos de información los cuales generan sobrecostos en sus procesos operacionales y administrativos, hace uso de métodos rudimentarios para llevar a cabo los procesos dentro de la empresa, llevando información valiosa como facturas, contratos, etc.. en libros, acarreando problemas de acceso, control y manipulación de estos documentos generando un caos empresarial.

Esta situación es un obstáculo para el proceso de expansión que la empresa AYT arenas y triturados tiene como visión en los próximos años, la cual tiene como objetivo llegar a ser uno de los principales proveedores de material en la región del oriente antioqueño. Al ser una de las empresas que aún no cuenta con una sistematización de sus procesos corre el riesgo de quedarse atrás frente a sus inmediatos competidores los cuales han adoptado tecnologías de la información para la optimización, rendimiento y control sobre los procesos ejecutados en la misma.

1.2 JUSTIFICACION

La empresa AYT arenas y triturados tiene como objetivo la implantación de un sistema de información que le permita controlar de una manera óptima los diferentes procesos y flujos de información que participan en estos, la empresa AYT se encuentra en proceso de expansión lo cual ha ocasiona el incremento desbordado de los procesos en la empresa y por tanto el flujo de datos que se procesan en esta misma, al no poder contar con la sistematización de la información ha ocasionado perdida de la misma y una demora en las tareas que deben realizarse para llevar a cabo un proceso, debido a esto es fundamental buscar la manera en la cual se puedan procesar los datos de manera eficiente y rápida, y se pueda llevar un registro de estos mismos, los cuales puedan ser almacenados y accedidos en cualquier momento.

Para dar solución a este problema se buscó la forma de suplir las necesidades inmediatas de la empresa, dando como resultado la implementación de un sistema de información el cual ayudará a

manejar el flujo de datos de una manera rápida y eficiente, agilizando los procesos tanto administrativos como operacionales.

Consciente de la envergadura de este proyecto se planteó un conjunto de fases para llevar a cabo el diseño, desarrollo y la implantación del sistema, la práctica empresarial del presente proyecto abarca la primera fase la cual corresponde al diseño arquitectónico de software para la empresa AYT arenas y triturados, en el cual se realizó un diseño para dar solución a los problemas planteados por la empresa, los cuales incluyen el control de los procesos de la sistematización de datos referentes a la parte operacional de la empresa, los sistemas de control referentes a la mano de obra en la empresa y sus respectivos sistema de control de distribución el cual permitirá tener un registro exacto de cuanto y cómo se distribuye el material ofrecido por la empresa.

1.3 OBJETIVOS

1.3.1 Objetivo general

- Diseñar una arquitectura de software para la empresa AYT arenas y triturados

1.3.2 Objetivos específicos

- Recopilar y documentar información de la arquitectura del software.
- Desarrollar gestión de requerimientos en la empresa AYT arenas y triturados.
- Elaborar el modelo de vistas 4+1 UML para la empresa AYT arenas y triturados.

1.4 METODOLOGIA

Se dispone de una serie de herramientas de investigación las cuales permiten recopilar información para hacer el levantamiento de requerimientos e iniciar de esta manera la construcción de nuestra arquitectura de software, en el presente proyecto se realiza una investigación con enfoque cualitativo en el cual se dispuso de métodos de observación para recopilar datos no numéricos como: entrevistas, encuestas, técnicas de observación etc.

El proyecto va dirigió a la empresa AYT arenas y triturados el cual busca dar solución inmediata a los procesos de control y gestión al interior de la empresa, de este modo se busca habilitar procesos que mejoren la calidad laboral y reduzcan los tiempos de ejecución sobre procesos realizados en la empresa. Los instrumentos utilizados para la captura de requerimientos fueron la observación y el método de entrevistas, los cuales arrojaron la información necesaria para establecer los parámetros de trabajo.

Por último, se hace uso de la información recolectada y se proyecta en una serie de diagramas los cuales ayudan a representar el funcionamiento de los procesos que se quieren sistematizar y mejoran considerablemente la perspectiva de la empresa a la hora de tomar decisiones sobre la misma.

2 MARCO TEORICO

2.1 SISTEMA DE INFORMACION

Un sistema de información puede concebirse como un conjunto estructurado de personas, máquinas y procesos, destinado a producir un flujo de informaciones recogidas, a su vez, dentro y fuera de la organización. Estas informaciones servirán de base a la toma de decisiones a cada nivel de responsabilidad. Los sistemas de Información dan soporte a las operaciones empresariales, la gestión y la toma de decisiones, proporcionando a las personas la información que necesitan mediante el uso de las tecnologías de la información. Las empresas y, en general, cualquier organización, los utilizan como un elemento estratégico con el que innovar, competir y alcanzar sus objetivos en un entorno globalizado. Los sistemas de información integran personas, procesos, datos y tecnología, y van más allá de los umbrales de la organización, para colaborar de formas más eficientes con proveedores, distribuidores y clientes. (Batalle, 2015)

Objetivos de un sistema de información

Orden	Motivos de implantación de un sistema de respuestas
1.	Información más rápida sobre la marcha de la empresa.
2.	Economía del gasto de personal.
3.	Informaciones más detalladas sobre la marcha de la empresa.
4.	Necesidad de utilización del ordenador como consecuencia de la calidad.
5.	Mejora del servicio postventa.
6.	Falta de mano de obra.
7.	Economía del gasto de material.
8.	Liberación de capitales (reducción de stocks, por ejemplo).
9.	Mejor utilización del capital invertido.
10.	Efectos publicitarios.
11.	Diversos.

2.2 ARQUITECTURA DEL SOFTWARE

De acuerdo el Software Engineering Institute (SEI)(Cervantes), la Arquitectura de Software se refiere a “las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.

El término “elementos” dentro de la definición del SEI es vago a propósito, pues puede referirse a distintas entidades relacionadas con el sistema. Los elementos pueden ser entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios).

La arquitectura de software, en un sentido estricto, se define como el conjunto de estructuras que componen el sistema, lo que incluye elementos de software, las relaciones entre los mismos, y las propiedades tanto de los elementos como de sus relaciones (SWEBOK, 2014). En otras palabras, la arquitectura de software define el conjunto de componentes de un sistema, las interfaces de comunicación de estos, y la manera como estos componentes se comunican entre ellos usando estas interfaces. Las organizaciones dependen de los sistemas de software que soportan su operación, y el buen funcionamiento de estos depende de sus arquitecturas. La arquitectura de un sistema de software se diseña para satisfacer los requerimientos funcionales y no funcionales establecidos por los interesados en el sistema (ej.: usuarios, clientes, proveedores). Los funcionales se refieren a las tareas que se espera el sistema pueda realizar, mientras que los no-funcionales se refieren a la calidad con que se espera el sistema realice estas tareas. (Villegas, 2020)

Importancia de la arquitectura de software

La necesidad del manejo de la arquitectura de un sistema de software nace con los sistemas de mediana o gran envergadura, que se proponen como solución para un problema determinado. En la medida que los sistemas de software crecen en complejidad, bien sea por número de requerimientos o por el impacto de estos en general, la técnica es descomponer el sistema en componentes más pequeños que interactúan entre ellos mismos y que al organizarse de cierta manera constituyen la solución a un problema.(Barajas & Carrillo, 2014)

Una arquitectura de software define la estructura del sistema. Esta estructura se constituye de componentes, cumpliendo funciones específicas, la arquitectura de software puede ser vista como la estructura del sistema en función de la definición de los componentes y sus interacciones. La arquitectura de software puede considerarse un intermediario entre los requerimientos del sistema y la implementación (Hofmeister et al., 2000). que servirá de medio de comunicación entre los miembros del equipo de desarrollo, los clientes y usuarios finales, dado que contempla los aspectos que interesan a cada uno. Además, pasa a ser la base del diseño del sistema a desarrollar.

De igual manera, serán de particular importancia las propiedades no funcionales del sistema de software, pues influyen notoriamente en la calidad de este. Estas propiedades tienen un gran impacto en el desarrollo y mantenimiento del sistema, su operabilidad y el uso que éste haga de los recursos. Entre las propiedades no funcionales más importantes se encuentran: modificabilidad, eficiencia, mantenibilidad, interoperabilidad, confiabilidad, reusabilidad y facilidad de ejecución de pruebas.(Camacho et al., 2004)

2.2.3 PATRÓN DE ARQUITECTURA

La arquitectura del software es una representación gráfica del sistema que comunica lo siguiente:

- La organización del Sistema.
- Diferentes componentes con respecto a sus funciones.

Una arquitectura de software típica requiere decisiones relacionadas con la seguridad, el rendimiento, la capacidad de gestión, etc (Dhaduk, 2020).

A continuación, se hace una descripción de los patrones de arquitectura más comunes.

2.2.3.1 Arquitectura Cliente-Servidor

Cliente-Servidor es uno de los estilos arquitectónicos distribuidos más conocidos, el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumido por el Cliente.(Blancarte, 2020)

En una arquitectura Cliente-Servidor existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado.(Blancarte, 2020)

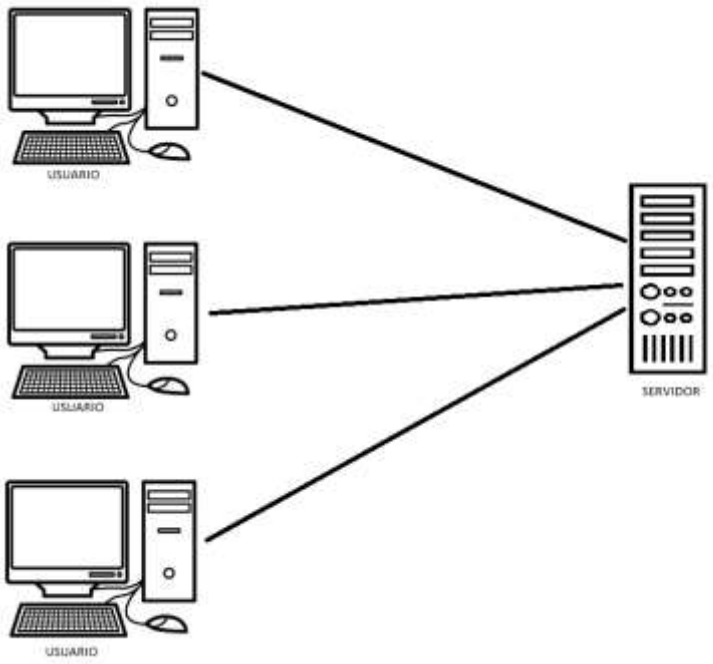

Ilustración 1: Estructura Cliente-Servidor

En esta arquitectura, el servidor deberá exponer un mecanismo que permite a los clientes conectarse, que por lo general es TCP/IP, esta comunicación permitirá una comunicación continua y bidireccional, de tal forma que el cliente puede enviar y recibir datos del servidor y viceversa.(Blancarte, 2020)

Cliente-Servidor es considerada una arquitectura distribuida debido a que el servidor y el cliente se encuentran distribuidos en diferentes equipos (aunque podrían estar en la misma máquina) y se comunican únicamente por medio de la RED o Internet.(Blancarte, 2020)

El cliente y el servidor son aplicaciones diferentes, por lo que pueden tener un ciclo de desarrollo diferente, así como usar tecnologías y un equipo diferente entre sí. Sin embargo, en la mayoría de

los casos, el equipo que desarrolla el servidor también desarrolla el cliente, por lo que es normal ver que el cliente y el servidor están construidos con las mismas tecnologías.(Blancarte, 2020)

Ventajas

1. **Centralización**: El servidor fungirá como única fuente de la verdad, lo que impide que los clientes conserven información desactualizada.
2. **Seguridad**: El servidor por lo general está protegido por firewall o subredes que impiden que los atacantes pueden acceder a la base de datos o los recursos sin pasar por el servidor.
3. **Fácil de instalar (cliente)**: El cliente es por lo general una aplicación simple que no tiene dependencias, por lo que es muy fácil de instalar.
4. **Separación de responsabilidades**: La arquitectura cliente-servidor permite implementar la lógica de negocio de forma separada del cliente.
5. **Portabilidad**: Una de las ventajas de tener dos aplicaciones es que se puede desarrollar cada parte para correr en diferentes plataformas, por ejemplo, el servidor solo en Linux, mientras que el cliente podría ser multiplataforma.(Blancarte, 2020)

Desventajas

1. **Concurrencia**: Una cantidad no esperada de usuarios concurrentes puede ser un problema para el servidor, quien tendrá que atender todas las peticiones de forma simultánea.
2. **Todo o nada**: Si el servidor se cae, todos los clientes quedarán totalmente inoperables.

3. **Depuración**: Es difícil analizar un error, pues los clientes están distribuidos en diferentes máquinas, incluso, equipos a los cuales no se tiene acceso, lo que hace complicado recopilar la traza del error.(Blancarte, 2020)

2.2.3.2 Arquitectura Peer To Peer (P2P)

El estilo arquitectónico Red entre iguales (Peer-to-peer, P2P) es una red de computadoras donde todos los dispositivos conectados a la red actúan como cliente y servidor al mismo tiempo. En esta arquitectura no es necesario un servidor central que administre la red (aunque puede existir), si no que todos los nodos de la red pueden comunicarse entre sí.(Blancarte, 2020)

La arquitectura P2P es para muchos solo una variante de la arquitectura Cliente-Servidor, sin embargo, tiene una diferencia importante que hace que se pueda clasificar como un estilo arquitectónico independiente, y es que la arquitectura Cliente-Servidor tiene como punto medular la centralización, mientras la arquitectura P2P busca la descentralización.(Blancarte, 2020)

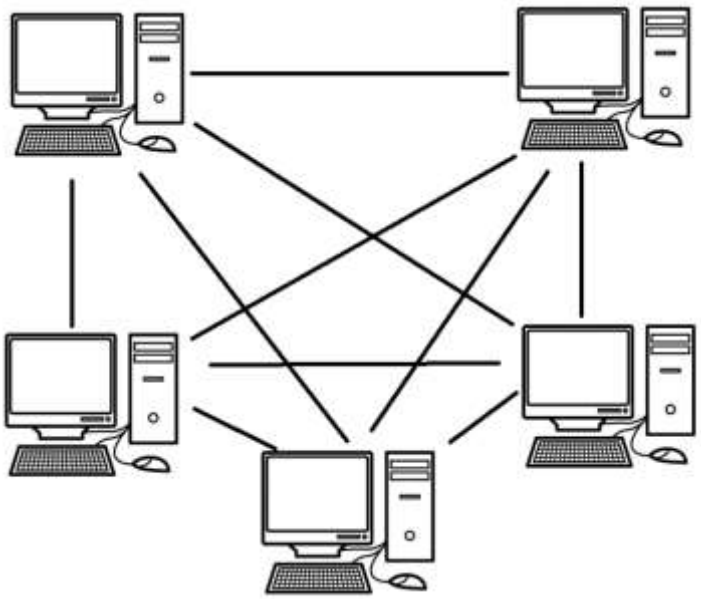

Ilustración 2: Estructura arquitectura P2P

La arquitectura P2P trata de unir el Cliente y el Servidor en una sola aplicación, lo que permite conectarse a otras computadoras de la red para consumir los recursos expuestos por los otros nodos de la red, pero al mismo tiempo, funciona como un Servidor, lo que permita que otros nodos se conecten a nuestro software para leer los recursos que se exponen.(Blancarte, 2020)

Algo interesante de esta arquitectura es que el software puede funcionar como Cliente y Servidor al mismo tiempo, por lo que se puede compartir recursos al mismo tiempo que consumir recursos de otro nodo de la red.(Blancarte, 2020)

Ventajas

1. **Alta escalabilidad**: Las redes P2P permiten escalar fácilmente, pues a mayor número de nodos, más recursos hay disponibles y la carga de trabajo se divide entre todos los participantes.
2. **Tolerancia a fallas**: Tener los recursos distribuidos por varios nodos permite que los recursos estén disponibles sin importar si algunos de los nodos se caen.
3. **Descentralización**: Los sistemas P2P puros tiene una autonomía completa, por lo que pueden funcionar sin un servidor central que organice toda la red.
4. **Equilibrio de carga**: En una arquitectura P2P, todos los nodos de la red tienen el mismo rol y responsabilidades, por lo que toda la carga de trabajo se equilibra entre todos los nodos de la red.(Blancarte, 2020)

Desventajas

1. **Alta complejidad**: Las arquitecturas P2P tiene una complejidad muy alta para desarrollarse, ya que se debe crear aplicaciones que funcionen como cliente y servidor, al mismo tiempo que debe garantizar que las búsquedas de los recursos sean eficientes.
2. **Control**: A menos que se tenga un servidor central para administrar los recursos que se buscan y los usuarios que se conectan a la red, es muy fácil perder el control sobre el contenido que se puede compartir en la red.
3. **Seguridad**: Esta es una de las características menos implementadas en este tipo de arquitecturas, para evitar la intercepción de las comunicaciones, nodos maliciosos, contenido falso o adulterado o la propagación de virus o programas maliciosos.(Blancarte, 2020)

2.2.3.3 Arquitectura en Capas

La arquitectura en capas es una de las más utilizadas, no solo por su simplicidad, sino porque también es utilizada por defecto cuando no se tiene seguridad que arquitectura se debe utilizar para la aplicación.(Blancarte, 2020)

La arquitectura en capas consta en dividir la aplicación en capas, con la intención de que cada capa tenga un rol muy definido.

En la práctica, la mayoría de las veces este estilo arquitectónico es implementado en 4 capas, presentación, negocio, persistencia y base de datos, sin embargo, es habitual ver que la capa de

negocio y persistencia se combinan en una solo capa, sobre todo cuando la lógica de persistencia está incrustada dentro de la capa de negocio.(Blancarte, 2020)

En una arquitectura en capas, todas las capas se colocan de forma horizontal, de tal forma que cada capa solo puede comunicarse con la capa que está inmediatamente por debajo, por lo que, si una capa quiere comunicarse con otras que están mucho más abajo, tendrán que hacerlo mediante la capa que está inmediatamente por debajo. Por ejemplo, si la capa de presentación requiere consultar la base de datos, tendrá que solicitar la información a la capa de negocio, la cual, a su vez, la solicitará a la capa de persistencia, la que, a su vez, la consultará a la base de datos, finalmente, la respuesta retornará en el sentido inverso hasta llegar la capa de presentación.(Blancarte, 2020)

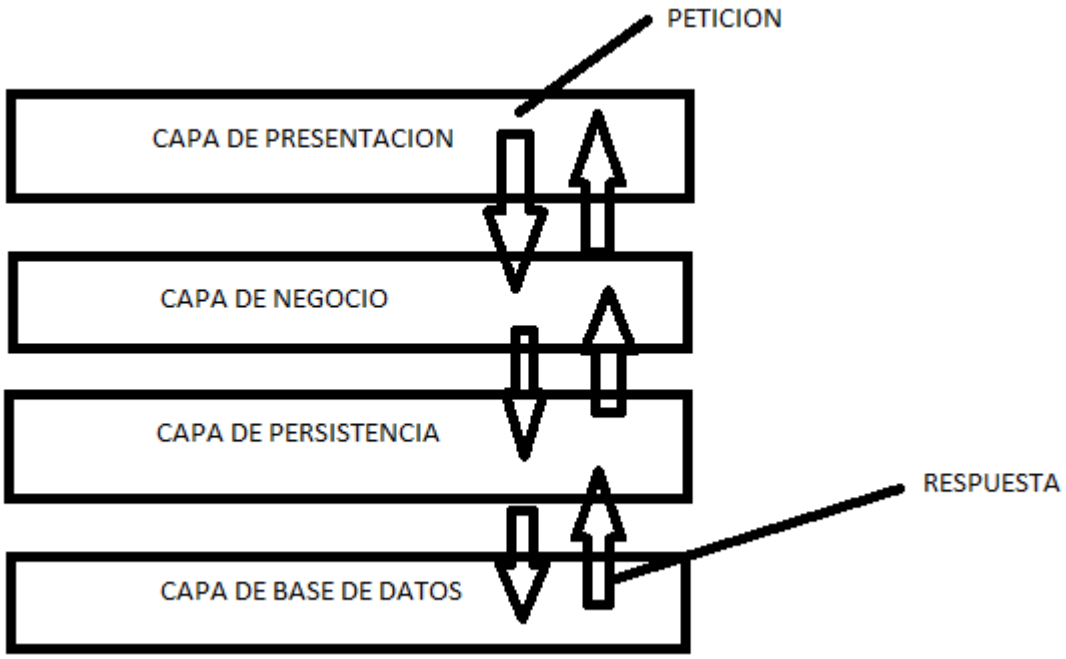

Ilustración 3: Estructura arquitectura en capas

Un detalle a tener en cuenta en esta arquitectura es que cada capa debe de ser un componente independiente, de tal forma que se puedan desplegar por separado, incluso, es habitual que estos componentes residan en servidores separados pero que se comunican entre sí.(Blancarte, 2020)

La separación de la aplicación en capas busca cumplir con el principio de separación de preocupaciones, de tal forma que cada capa se encargue una tarea muy definida, por ejemplo, la capa de presentación solo se preocupa por presentar la información de forma agradable al usuario, pero no le interesa de donde viene la información ni la lógica de negocio que hay detrás, en su lugar, solo sabe que existe una capa de negocio que le proporcionará la información. Por otra parte, la capa de negocio solo se encarga de aplicar todas las reglas de negocio y validaciones, pero no le interesa como recuperar los datos, guardarlos o borrarlos, ya que para eso tiene una capa de persistencia que se encarga de eso. Por otro lado, la capa de persistencia es la encargada de comunicarse a la base de datos, crear las instrucciones SQL para consultar, insertar, actualizar o borrar registros y retornarlos en un formato independiente a la base de datos. De esta forma, cada capa se preocupa por una cosa y no le interesa como le haga la capa de abajo para servirle los datos que requiere.(Blancarte, 2020)

Ventajas

1. **Separación de responsabilidades**: Permite la separación de preocupaciones (SoC), ya que cada capa tiene una sola responsabilidad.
2. **Fácil de desarrollar**: Este estilo arquitectónico es especialmente fácil de implementar, además de que es muy conocido y una gran mayoría de las aplicaciones la utilizan.

3. **Fácil de probar**: Debido a que la aplicación es construida por capas, es posible ir probando de forma individual cada capa, lo que permite probar por separada cada capa.
4. **Fácil de mantener**: Debido a que cada capa hace una tarea muy específica, es fácil detectar el origen de un bug para corregirlo, o simplemente se puede identificar donde se debe aplicar un cambio.
5. **Seguridad**: La separación de capas permite el aislamiento de los servidores en subredes diferentes, lo que hace más difícil realizar ataques.(Blancarte, 2020)

Desventajas

1. **Performance**: La comunicación por la red o internet es una de las tareas más tardadas de un sistema, incluso, en muchas ocasiones más tardado que el mismo procesamiento de los datos, por lo que el hecho de tener que comunicarse de capa en capa genera un degrado significativo en el performance.
2. **Escalabilidad**: Las aplicaciones que implementan este patrón por lo general tienda a ser monolíticas, lo que hace que san difíciles de escalar.
3. **Complejidad de despliegue**: En este tipo de arquitecturas es necesarios desplegar los componentes de abajo arriba, lo que crea una dependencia en el despliegue.
4. **Tolerancia a los fallos**: Si una capa falla, todas las capas superiores comienzan a fallar en cascada.(Blancarte, 2020)

2.2.3.4 Arquitectura de Microservicios

El estilo arquitectónico de Microservicios se ha convertido en el más popular de los últimos años. El estilo de Microservicios consiste en crear pequeños componentes de software que solo hacen una tarea, la hace bien y son totalmente autosuficientes, lo que les permite evolucionar de forma totalmente independiente del resto de componentes.(Blancarte, 2020)

El nombre de “Microservicios” se puede mal interpretar pensando que se trata de un servicio reducido o pequeño, sin embargo, ese es un concepto equivocado, los Microservicios son en realidad pequeñas aplicaciones que brindan un servicio.(Blancarte, 2020)

Un Microservicios es un pequeño programa que se especializa en realizar una pequeña tarea y se enfoca únicamente en eso, por ello, se puede decir que los Microservicios son Altamente Cohesivos, pues toda las operaciones o funcionalidad que tiene dentro está extremadamente relacionadas para resolver un único problema.(Blancarte, 2020)

Una de las grandes ventajas de los Microservicios es que son componentes totalmente encapsulados, lo que quiere decir que la implementación interna no es de interés para los demás componentes, lo que permite que estos evolucionen a la velocidad necesaria, además, cada Microservicios puede ser desarrollado con tecnologías totalmente diferentes, incluso, es normal que utilicen diferentes bases de datos.(Blancarte, 2020)

La arquitectura de Microservicios divide toda la funcionalidad en pequeños componentes autosuficientes e independientes del resto de componentes.

Debido a que los Microservicios solo realizan una tarea, es imposible que por sí solos puedan realizar una tarea de negocio completa, por lo que es común que los Microservicios se comuniquen con otros Microservicios para delegar ciertas tareas, de esta forma, se puede ver que todos los

Microservicios crean una red de comunicación entre ellos mismos, incluso, se puede apreciar que diferentes Microservicios funcionan con diferentes bases de datos.(Blancarte, 2020)

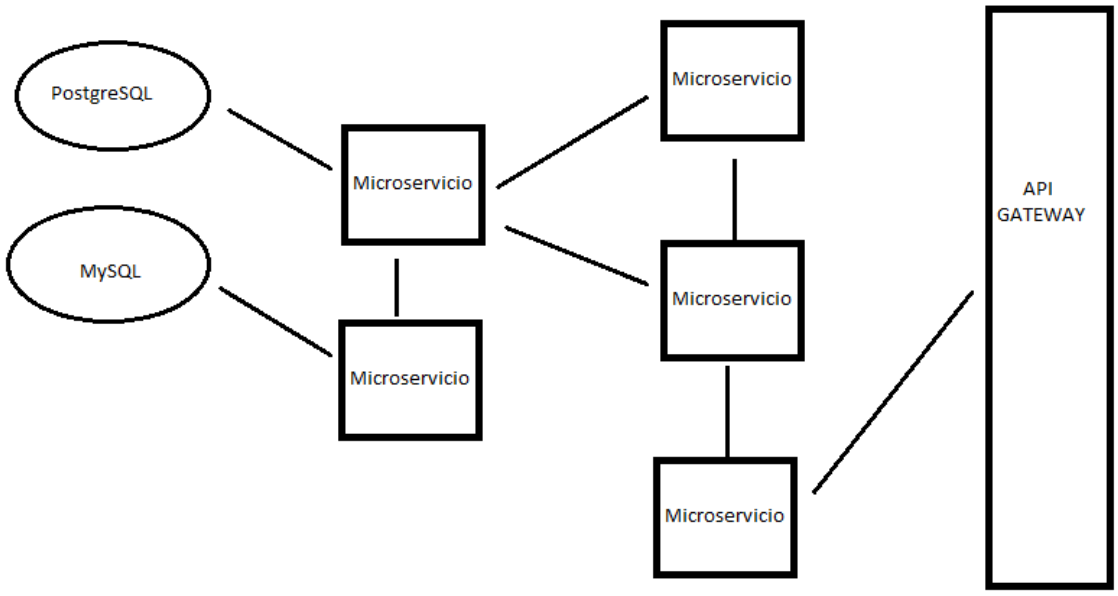

Ilustración 4: Estructura arquitectura de microservicios

Ventajas

1. **Alta escalabilidad**: Los Microservicios es un estilo arquitectónico diseñado para ser escalable, pues permite montar numerosas instancias del mismo componente y balancear la carga entre todas las instancias.
2. **Agilidad**: Debido a que cada Microservicios es un proyecto independiente, permite que el componente tenga ciclo de desarrollo diferente del resto, lo que permite que se puedan hacer despliegues rápidos a producción sin afectar al resto de componentes.

3. **Facilidad de despliegue**: Las aplicaciones desarrolladas como Microservicios encapsulan todo su entorno de ejecución, lo que les permite ser desplegadas sin necesidad de dependencias externas o requerimientos específicos de Hardware.
4. **Testeabilidad**: Los Microservicios son especialmente fáciles de probar, pues su funcionalidad es tan reducida que no requiere mucho esfuerzo, además, su naturaleza de exponer o brindar servicios hace que sea más fácil de crear casos específicos para probar esos servicios.
5. **Fácil de desarrollar**: Debido a que los Microservicios tiene un alcance muy corto, es fácil para un programador comprender el alcance del componente, además, cada Microservicios puede ser desarrollado por una sola persona o un equipo de trabajo muy reducido.
6. **Reusabilidad**: La reusabilidad es la médula espinal de la arquitectura de Microservicios, pues se basa en la creación de pequeños componentes que realice una única tarea, lo que hace que sea muy fácil de reutilizar por otras aplicaciones o Microservicios.
7. **Interoperabilidad**: Debido a que los Microservicios utilizan estándares abiertos y ligeros para comunicarse, hace que cualquier aplicación o componente pueda comunicarse con ellos, sin importar en que tecnología está desarrollado.(Blancarte, 2020)

Desventajas

1. **Performance**: La naturaleza distribuida de los Microservicios agrega una latencia significativa que puede ser un impedimento para aplicaciones donde el performance es lo más importante, por otra parte, la comunicación por la red puede llegar a ser incluso más tardado que el proceso en sí.
2. **Múltiples puntos de falla**: La arquitectura distribuida de los Microservicios hace que los puntos de falla de una aplicación se multipliquen, pues cada comunicación entre Microservicios tiene una posibilidad de fallar.
3. **Trazabilidad**: La naturaleza distribuida de los Microservicios complica recuperar y realizar una traza completa de la ejecución de un proceso, pues cada Microservicio arroja de forma separa su traza o logs que luego deben de ser recopilados y unificados para tener una traza completa.
4. **Madurez del equipo de desarrollo**: Una arquitectura de Microservicios debe ser implementada por un equipo maduro de desarrollo y con un tamaño adecuado, pues los Microservicios agregan muchos componentes que deben ser administrados, lo que puede ser muy complicado para equipo poco maduros.(Blancarte, 2020)

2.3 MODELO DE VISTAS UML 4+1

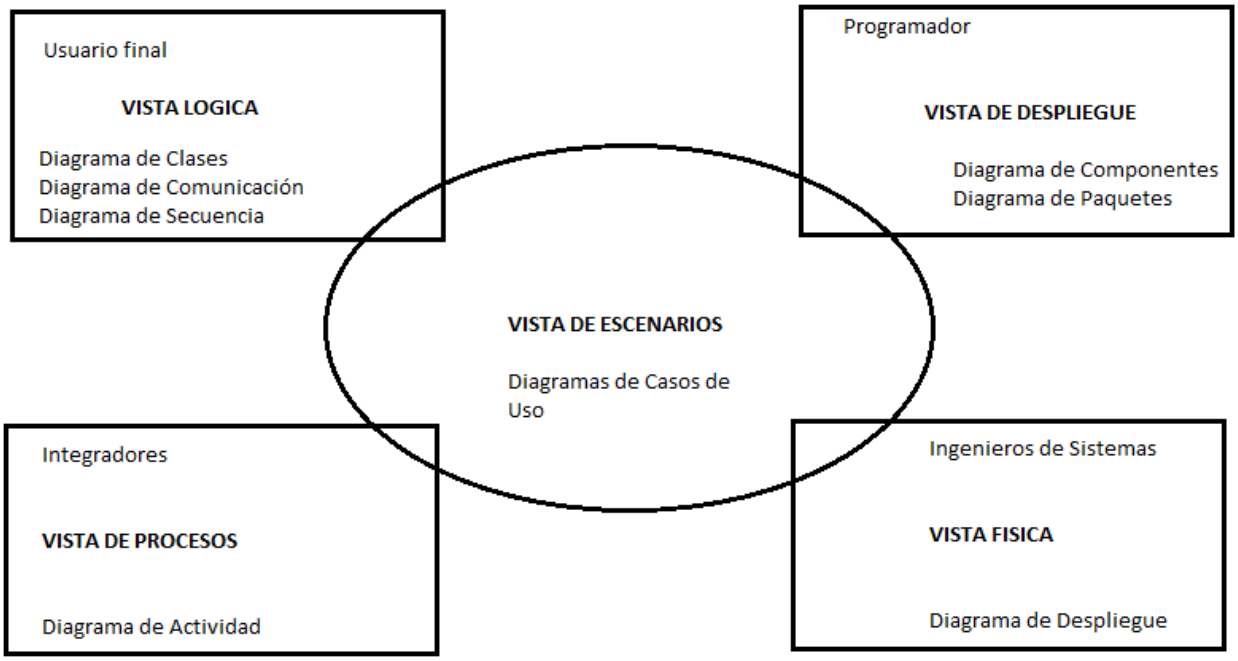

Ilustración 5: Estructura modelo de vistas 4+1

El modelo de vistas 4 + 1, fue creada en el año 1995 por Philippe Kruchten, debido a que la captura de la arquitectura de un sistema de software no se podía implementar adecuadamente en un solo modelo o diagrama, por lo que, ante la necesidad de capturar la arquitectura de un sistema, se procedió a representar los diferentes aspectos y características de la arquitectura en múltiples vistas (Kruchten, 1995)

El modelo de vistas 4 + 1, se define en 4 principales vistas para definir la arquitectura, pero también existe una vista adicional, la cual se define por las necesidades funcionales que envuelven el sistema, las cuales se identifican como la vista de casos de uso (Philippe Kruchten, 1995). Los

modelos se definen por las 4 vistas principales y la vista “+ 1”, las cuales son según Philippe Kruchten (1995):

Vista lógica: En esta vista se representa la funcionalidad que el sistema proporcionara a los usuarios finales**,** se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece. Esta vista puede ser representada por medio de los diagramas de clases, comunicación o secuencia.

La vista de despliegue: En esta vista se muestra el sistema desde la perspectiva de un programador, se ocupa de la gestión del software; o en otras palabras, se va a mostrar cómo está dividido el sistema software en componentes y las dependencias que hay entre esos componentes. Esta vista puede ser representada por medio de los diagramas de paquetes y diagrama de componentes.

Vista de procesos: En esta vista se muestran los procesos que hay en el sistema y la forma en la que se comunican estos procesos. Esta vista puede ser representada por medio de los diagramas de actividades.

Vista física: Muestra el despliegue de la aplicación en la red de computadoras. Describe el mapeo del software en el hardware y refleja los aspectos de distribución.

La vista de escenarios: Presenta a los actores y sus respectivos casos de uso, de modo que contiene los requisitos desarrollados en las vistas restantes.

3 ARQUITECTURA PROPUESTA

Se dispuso de diferentes metodologías para la creación del diseño arquitectónico, en primer instancia se requiere capturar los requerimientos funcionales para la empresa AYT arenas y triturados para solventar los problemas del manejo de flujo de información al interior de la empresa, y poder expresarlos de una manera lógica y comprensible, en este sentido se dispuso de metodologías para la captura de estos requerimientos, en este proyecto se hizo uso de la técnica por medio de entrevistas, estas se realizaron tanto a los trabajadores como a los directivos de la empresa, para escuchar las necesidades que tienen en cuanto al manejo de información, así mismo se hizo uso de la técnica de observación, por medio de la cual fue posible entender la manera como opera la empresa y como se trata la información al interior de esta, el sistema de almacenamiento de datos y los procesos para transmitir esta información entre los mismos trabajadores.

El siguiente paso fue reconocer los actores que interactúan en los procesos de la empresa y conocer que funciones realizan en la misma, de esta manera es posible realizar la sistematización de estas funciones de acuerdo con el rol asignado en la empresa, y mejorar considerablemente el procesamiento, distribución y almacenamiento de la información.

En la captura de requerimiento, se realizó la identificación de 4 actores, los cuales serán los que interactúen con el sistema haciendo uso de sus respectivas funciones.

- **Gerente**: será la persona encargada de tener el control en el manejo de usuarios, así como las funciones de control sobre los procesos llevados a cabo en la empresa.
- **Despachador**: será el encargado de la parte del proceso de datos y el proceso de gestión de pedidos en la empresa.

- **Operario**: es el encargado de la parte operacional de la empresa, y el uso de herramientas en la misma.
- **Seguridad** Trabajo: es el encargado de autorizar el uso de herramientas y reportar cualquier novedad que tenga que ver con la seguridad interna en la empresa.

3.1 CAPTURA DE REQUERIMIENTOS

3.1.1 Funcionales

1. Permitir el ingreso a los usuarios registrados y estos estarán clasificados según su rol al interior de la empresa que pueden ser: gerente, despachador, operario, seguridad trabajo.
2. Permitir la administración de los usuarios con acciones de crear, actualizar, consultar y eliminar.
3. Se debe llevar el control del registro de ingreso y salida en el horario laboral de los diferentes trabajadores, Los trabajadores registraran el horario de entrada a las 7 am, y el horario de salida a las 5 pm, de lunes a sábado.
4. Controlar los tiempos de funcionamiento de la planta de procesos, permitiendo tener registro de los intervalos de uso.
5. Los vehículos de transporte del material despachado deben poder ser registrados, consultados y eliminados.
6. Se debe permitir gestionar el proceso de despachos al interior de la empresa, permitiendo registrar, eliminar, y finalizar un pedido, asignándole un vehículo de transporte a cada despacho registrado.

7. Permitir la apertura de solicitudes por parte de los operarios al querer hacer uso de algún recurso (herramientas: mecánicas, eléctricas, hidráulicas etc..) al interior de la empresa y dejando registro para su aprobación.
8. Permitir la gestión de los recursos de solicitudes para su autorización o no y su posterior consulta por parte del operario para determinar si puede hacer uso del recurso solicitado.
9. Los usuarios podrán registrar eventos que sucedan en el transcurso de la jornada laboral, registrando acontecimientos inusuales que puedan suceder en el interior de la empresa en los horarios laborales establecidos.
10. Se podrá registrar bitácoras de funcionamiento de la maquinaria perteneciente a la planta de procesamiento en la empresa y sus posteriores mantenimientos, dejando registros de cómo funcionan y el proceso de mantenimiento realizado en cada una de ellas.
11. Se requiere la generación de reportes como: reporte de eventos, reporte registro entrada-salida, reporte registro encendido-apagado, reporte actualización bitácora de funcionamiento.

3.1.2 No Funcionales

1. El sistema debe ser capaz de operar con una concurrencia de 100 personas accediendo al sistema simultáneamente.
2. El sistema debe proteger los datos ante los accesos no autorizados.
3. El sistema debe mantenerse en funcionamiento las 24 horas del día, en los periodos de actividad laboral.
4. El sistema debe mantener los datos actualizados en un tiempo no mayor a 5 segundos ya que continuamente serán accedido por los usuarios.

La captura de requerimientos es un proceso muy importante que se lleva a cabo para entender e interpretar las necesidades que tiene la empresa y la manera como opera, estos requerimientos brindan un enfoque más claro de lo que se quiere lograr al sistematizar los procesos, si no se realiza un buen levantamiento de la información es posible que el sistema tome un rumbo diferente a lo que espera el cliente, identifica los actores que harán parte del sistema y que procesos se van a desarrollar al interior de la empresa, ayudan a entender lo que espera el cliente del sistema y la manera en la que va a operar, nos ayudan a reducir costos a futuro pues una mala implementación por no capturar información importante para el desarrollo puede generar que se tenga que reparar o cambiar parte de lo que ya se ha implementado.

3.2 PROCESO PARA EL DISEÑO ARQUITECTÓNICO

Una vez analizados algunos de los patrones y/o estilos de arquitectura de software más usados para este tipo de desarrollos se seleccionó el patrón de arquitectura cliente-servidor, se llegó a la conclusión después de identificar las necesidades de la empresa.

La empresa AYT arenas y triturados espera alojar un límite máximo de 100 usuarios, por esta razón se opta por hacer uso de un solo servidor el cual será suficiente para gestionar las peticiones de todos los usuarios simultáneamente, así como albergar el servidor de aplicaciones y la base de datos, sin necesidad de hacer uso de hardware adicional, o componentes que ayuden a la gestión de peticiones.

El cliente y el servidor son entidades que operan de manera independiente, esto hace posible que la interfaz del usuario sea desarrollada de una manera más amigable, además pueden consumir la aplicación desde diferentes dispositivos (móvil, pc, etc.), que pueden estar corriendo en diferentes sistemas operativos, el lado del servidor se encargara de ejecutar toda la lógica del negocio y será el que realice las tareas pesadas que demandan alto consumo computacional, al ser un hardware que cumple con alta demanda de especificaciones se puede realizar actualizaciones de sistema hardware, afectando en lo mínimo al cliente.

Partiendo de los requerimientos capturados y del producto final esperado por la empresa AYT arenas y triturados, se requiere un sistema a la medida, que se acople a las necesidades levantadas en este momento y la arquitectura propuesta es suficiente para cubrir los requerimientos identificados.

3.3 MODELO DE VISTAS 4+1

El modelo de vistas 4+1 es un conjunto de diagramas que representan el comportamiento del sistema y su arquitectura software en diferentes vistas, cada una de estas vistas van enfocadas a los interesados en el proyecto.

La vista lógica permite visualizar los datos contenidos en el sistema, la relación que tienen los procesos y se hizo uso de un modelo de datos para representarla, esta vista va dirigida a los usuarios finales.

La vista de procesos permite visualizar la secuencia de pasos al ejecutar un proceso en el sistema, se hizo uso del diagrama de actividades para representarlo, esta vista va dirigida a los integradores y permite distinguir los actores identificados en el sistema y cada uno de los procesos en los que interactúan.

La vista de despliegue permite representar la manera como va dividido el código, por un lado, el cliente realiza las peticiones por medio de una interfaz grafica al servidor para obtener una respuesta, esta petición la recibe el servidor y describe el proceso de cómo es gestionada y procesada para retornar una respuesta al cliente. Esta vista va dirigida a los programadores.

La vista física es la encargada de representar los componentes físicos (hardware) del sistema, como está dividido estos componentes y la conexión que tiene cada componente, esta vista va dirigida a los ingenieros de sistemas.

La vista de escenarios da un panorama general de los actores identificados y las funciones que tendrán disponibles según su rol en la empresa AYT arenas y triturados.

3.3.1 Vista lógica

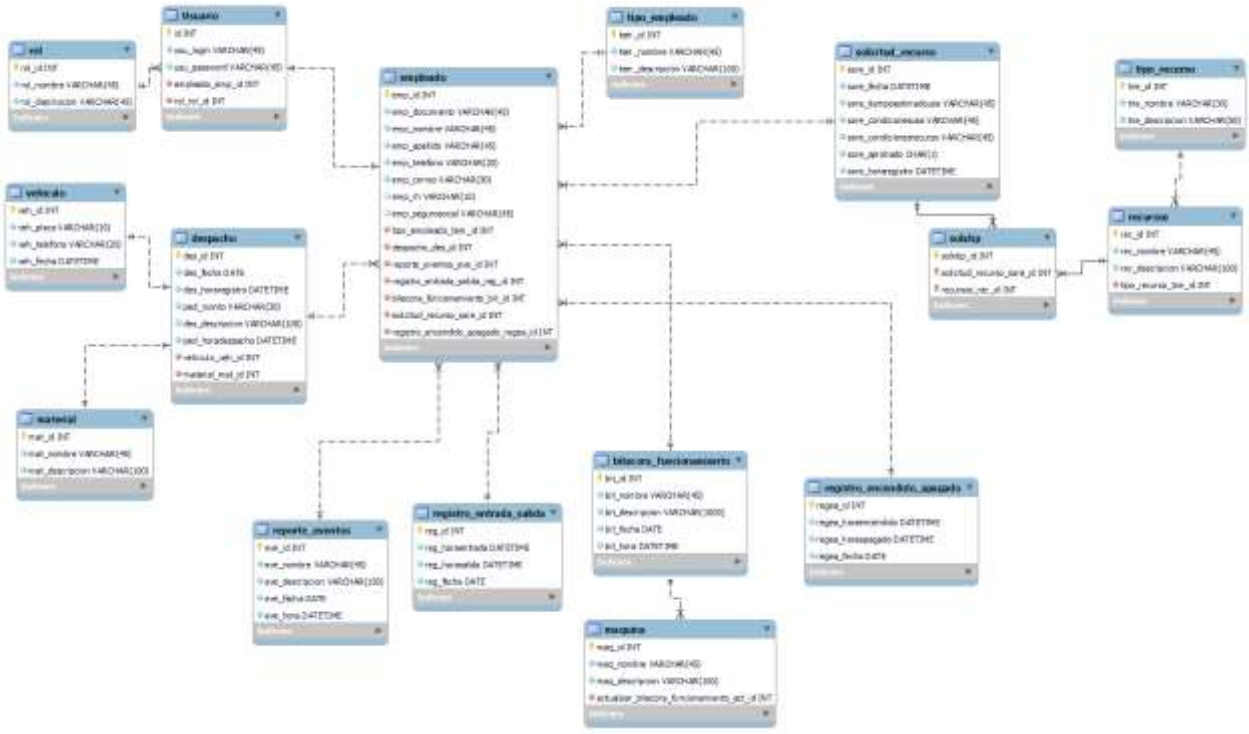

Ilustración 6: Modelo De Datos Representado por la Vista Lógica.

La ilustración 6 representa el modelo de datos del sistema de información en la empresa AYT arenas y triturados, sus respectivas relaciones e interacciones, este modelo representa las tablas contenidas en la base de datos y sus parámetros de información, ayuda a visualizar las relaciones que se presentan en cada proceso, y da un enfoque global de la estructura de datos del sistema. El diagrama tiene una tabla central la cual tiene relación con todos los procesos, la cual es la tabla

empleada, esta tabla va a contener los datos relacionados al empleado y las llaves foráneas de las tablas con las que se relaciona. Un empleado podrá hacer login en el sistema con un usuario y una contraseña, estos datos estarán almacenados en la tabla usuario que a su vez tendrá relación con la tabla rol que establece el tipo de rol que tendrá el usuario en el sistema como puede ser: administrador o usuario. Un empleado puede tener diferentes tipos de roles en la empresa, los cuales pueden ser: Gerente, Despachador, Operario, Seguridad Trabajo, y están registrados en la tabla tipo_empleado. Un empleado puede ser despachador y será el encargado de gestionar los despachos en la empresa y estos se van a registrar en la tabla despacho, que tendrá los datos necesarios para el registro, cada despacho se le asignará un vehículo y el material, el vehículo será el encargado de transportar el material y estará registrado en la tabla vehículo, el material será escogido en la tabla material que tiene las diferentes opciones para ofrecer. Una solicitud de recurso que va a estar almacenada en la tabla Solicitud_recurso será la interacción entre el operario y el empleado a cargo de la seguridad en el trabajo, en el cual el operario podrá abrir solicitudes para el uso de recursos de la empresa que van a estar registrados en la tabla recursos y que pueden pertenecer a un tipo de recurso como: mecánico, manual, eléctrico, hidráulico, y están contenidos en la tabla tipo_recurso.

Existen tablas independientes que van a estar relacionadas con la tabla empleado, la tabla reporte_eventos, esta tabla registrara diferentes eventualidades que puedan suceder en la jornada laboral y a cualquier momento. La tabla Registro_entrada_salida es la tabla que albergara los datos de registro de los empleados cuando inician la jornada laboral y cuando la terminan. La tabla bitácora_funcionamiento registrara la documentación del funcionamiento de la maquinaria en la empresa AYT y podrá documentar los diferentes mantenimientos realizados a estas máquinas, de esto modo poder tener acceso a esta información cuando sea requerido a futuro teniendo una base

de conocimiento para realizar la solución del problema, adicionalmente se podrá elegir el tipo de maquina que se quiere documentar y la cual podrá ser elegida de la tabla máquina. Con la intención de conocer los intervalos de funcionamiento de la planta de procesos en la empresa AYT se hará registro del horario de encendido y apagado de la planta de procesos que permite establecer un control mas exacto de los tiempos de funcionamiento que permite realizar una mejor estimación de costos.

3.3.2 Vista de despliegue

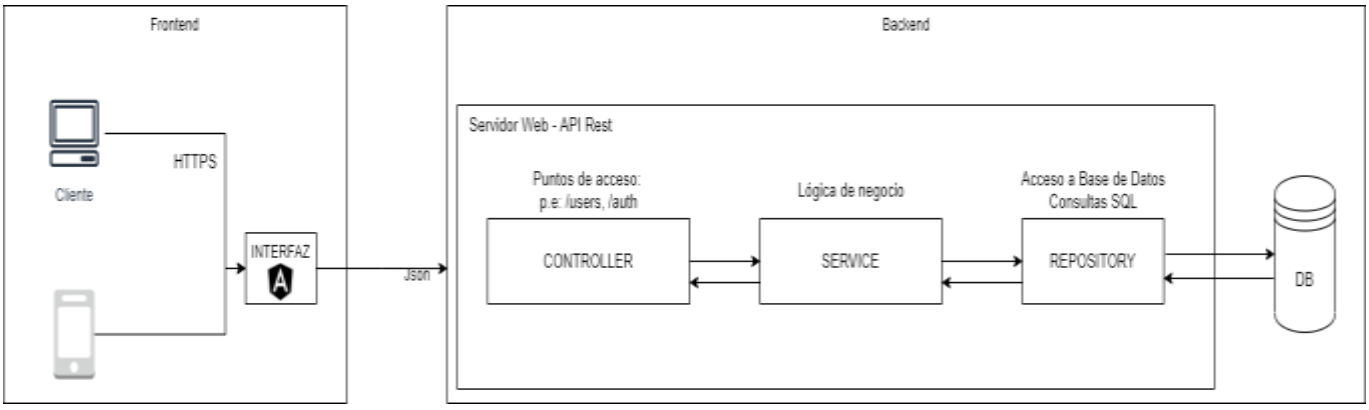

Ilustración 7: Distribución de código representado por la vista de despliegue.

La ilustración 7 representa la distribución del código que se va a ejecutar ante cualquier petición por parte de un usuario en el sistema.

Este sistema consta de 2 partes principales las cuales son: el frontend que es el encargado de la interfaz grafica que es lo que el usuario puede visualizar directamente, y por el otro lado el backend que será el encargado del manejo de las peticiones, procesando los datos, modificándolos si es necesario y almacenar esos cambios finales en la base de datos.

En la parte del cliente el usuario podrá acceder a través de una aplicación web que se desarrollara en Angular haciendo uso del lenguaje de programación JavaScript, el cliente realiza el consumo de los servicios web de la API REST a través del protocolo HTTPS en el cual se enviaran objetos tipo JSON para las peticiones que lo requieran, de lado del servidor estará compuesto por 3 capas, las cuales son: controlador, servicio, repositorio. El controlador es el encargado de recibir las peticiones por parte del cliente y gestiona los puntos o rutas de acceso de estas peticiones, direccionando esta petición hacia la capa de servicio en la cual se encuentra la lógica de negocio, donde se encuentran las funcionalidades que se necesitan para que opere la aplicación, esta capa se conecta con la capa de repositorio que es la encargada de realizar las consultas a la base de datos para entregársela a la capa de servicios y que pueda operar con la información consultada, una vez obtenida y procesada la información el controlador será el encargado de gestionar la ruta de vuelta a la interfaz gráfica y así ser mostrada al usuario, en este proceso los datos pueden ser cambiados al formato adecuado para que sea fácil la visualización con el framework angular.

La intención de los repositorios es manejar las operaciones CRUD entre otras mas complejas que son hacer consulta compuestas a la base de datos, además de separar la lógica que accede a los datos con la capa de servicios, gracias a esto es posible probar la lógica de negocio en la capa de servicios sin tener que acceder a los datos.

3.3.3 Vista de procesos

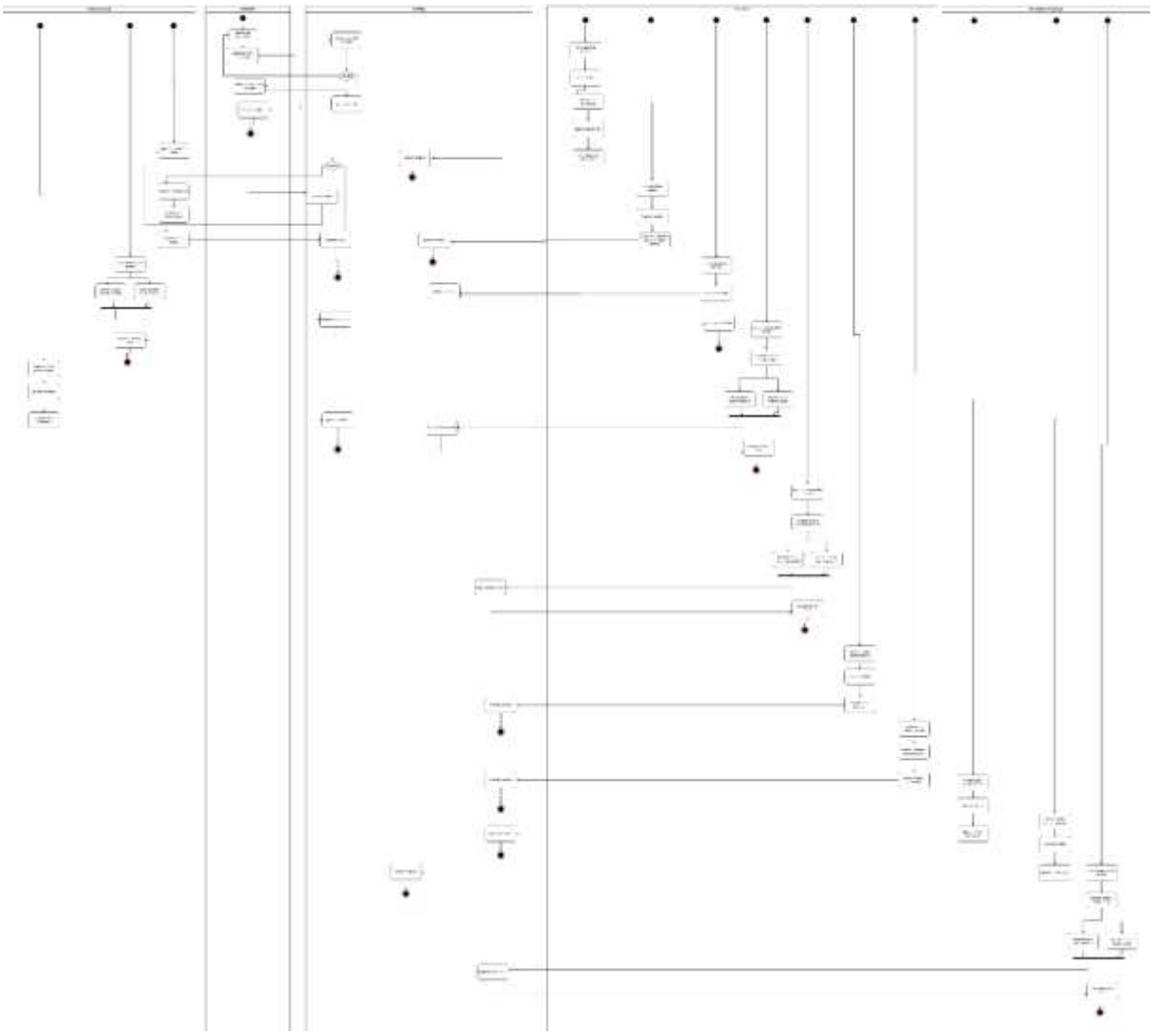

Ilustración 8: Procesos realizados de acuerdo con el rol.

La ilustración 8 representa los procesos realizados al interior de la empresa AYT arenas y triturados de acuerdo con cada rol de usuario y su interacción con el sistema. Cada carril contiene un actor identificado en el levantamiento de requerimientos y los procesos que va a llevar a cabo dentro de la empresa, estos procesos tienen su respectiva interacción con el sistema. Cada proceso inicia con las peticiones por parte de los actores, en una arquitectura cliente-servidor siempre el cliente inicia los procesos.

3.3.4 Vista Física

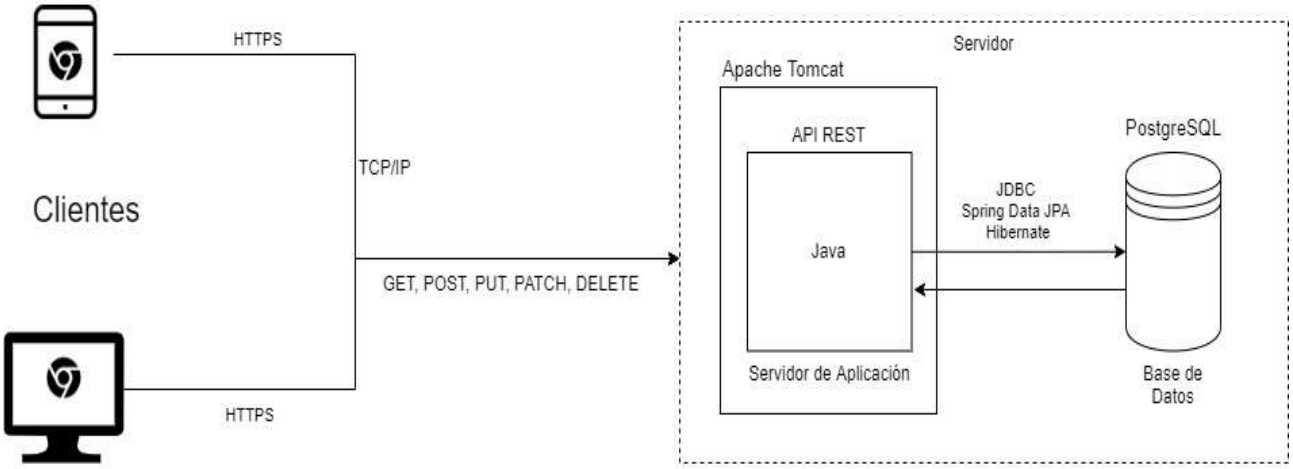

Ilustración 9: Vista física de los componentes del sistema y su comunicación.

La ilustración 9 representa la vista de procesos la cual ayuda a visualizar la distribución de componentes físicos y las respectivas conexiones que se realizan al realizar y contestar una petición.

Por un lado se tiene los componentes físicos del cliente los cuales pueden ser dispositivos móviles o de escritorio, que a través de un navegador podrán visualizar la aplicación web, estos dispositivos harán conexión con el servidor por medio del protocolo HTTPS y hará uso de algunos métodos HTTP según la petición requerida, como son: GET, POST, PUT, PATCH, DELETE entre otros, en la parte del servidor se tiene la instancia de base de datos y la instancia del servidor de la aplicación, la comunicación entre el servidor de aplicación y la base de datos se podrá hacer por medio de un driver de conexión para bases de datos.

Nuestra aplicación web estará albergada en un contenedor de aplicación como lo es el apache tomcat, este permite el despliegue y la disponibilidad del servicio para todos los usuarios y se puede

manejar todas las peticiones y respuestas a través de nuestra API REST. También se hará uso del motor de base de datos PostgreSQL que es de tipo relacional y que va a almacenar toda la información contenida de la aplicación web y la cual será accedida a través de consultas SQL de acuerdo a las peticiones de la API REST.

3.3.5 Vista de escenarios

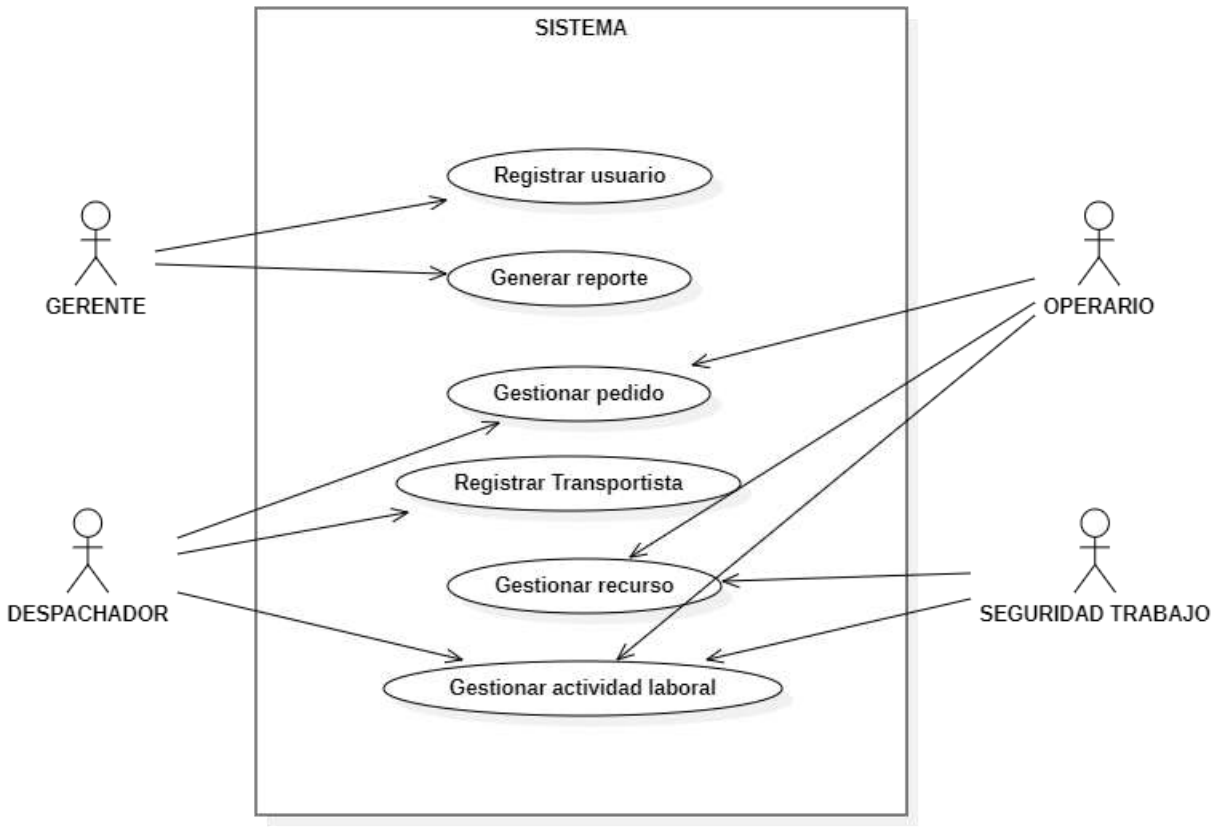

Ilustración 10: Diagrama casos de uso general.

La ilustración 10 representado con un diagrama de casos de uso identifica los roles dentro del sistema y las acciones que van a tener a disposición según su rol.

El gerente tendrá disponible los casos de uso registrar usuario que permite administrar los usuarios en el sistema, ya sea para registrarlos, actualizarlos o eliminarlos, y el caso de uso generar reporte, que permite visualizar los reportes registrados por los demás empleados en la empresa.

El despachador tendrá a disposición los casos de uso gestionar pedido, registrar transportista por medio de los cuales puede registrar un despacho asignándolo a un transportista que se encuentre registrado, el caso de uso gestionar actividad laboral que le permite registrar eventos inusuales que sucedan en la empresa dentro de la jornada laboral y le permite registrar a que hora ingresa a la empresa, y su hora de salida.

El operario tendrá a disposición los casos de uso gestionar pedido que le permite finalizar un pedido que ya ha sido despachado, el caso de uso gestionar actividad laboral que le permite registrar eventos inusuales que sucedan en la empresa dentro de la jornada laboral y le permite registrar a que hora ingresa a la empresa, y su hora de salida, también tendrá disponible el registro de encendido y apagado de la planta de procesos así como documentar los mantenimientos que se realicen a las maquinas de la empresa y su respectivo funcionamiento, el caso de uso gestionar recurso que le permite abrir solicitudes para hacer uso de cualquier recurso de la empresa y posteriormente consultar estos registros para observar si se aprobó o no el uso del recurso.

Seguridad en el trabajo tendrá a disposición los casos de uso gestionar actividad laboral que le permite registrar eventos inusuales que sucedan en la empresa dentro de la jornada laboral y le permite registrar a que hora ingresa a la empresa, y su hora de salida, también tendrá disponible el caso de uso gestionar recurso, que le permite consultar solicitudes abiertas por parte de los operarios para hacer uso de algún recurso en la empresa, y le permite registrar si autoriza o no el uso del recurso solicitado.

La representación de los diagramas 4+1 le permiten a la empresa AYT arenas y triturados tener el enfoque global de los procesos de control que suceden al interior de la empresa, permitiendo tener ideas más claras de las actividades realizadas por los trabajadores, gestionando los procesos y permitiendo que se realice de una manera más rápida y eficiente el manejo de la información requerida por parte de los directivos, así mismo permite identificar la manera como operan los procesos y la interacción que tendrá el sistema con cada uno de los usuarios, el inmediato acceso a los datos guardados en el sistema, y la generación de los reportes respectivos a la identificación de actividades en el levantamiento de requerimientos en la empresa.

4 CONCLUSIONES Y TRABAJO FUTURO

La arquitectura de software diseñada en este proyecto permite mejorar la calidad de los procesos en la empresa AYT arenas y triturados, identifica nuevos procesos que no se han tenido en cuenta que ayudan con la ejecución de tareas y mejora el desempeño laboral.

La identificación de actores y los respectivos roles desempeñados en la empresa ayudan en gran medida en la toma de decisiones para el control de personal en la empresa, da un enfoque general de que procesos se desarrollan en la empresa y quienes pueden realizar estos procesos, logrando asignar mejor los recursos en la empresa AYT arenas y triturados y asignando de una manera más organizada las tareas que se deben realizar por parte de los trabajadores.

Los sistema de información son de vital importancia en la actualidad para optimizar los procesos de información y el flujo de datos al interior de una entidad, existen diferentes metodologías las cuales ayudan a la hora de implementar un sistema de información, estas metodologías dan parámetros y una serie de pasos para realizar un diseño funcional y en la medida de lo posible ausente de fallos, es importante saber elegir los tipos de metodologías que se quieren usar, así como su implementación en el desarrollo del sistema de información.

La gestión de requerimientos es una parte primordial en todo proceso de implementación de un sistema de información, en la cual se conocen las necesidades que tiene una empresa, así como el funcionamiento de esta, de esta manera se puede tener comunicación directa con el cliente el cual hará parte fundamental del proceso de diseño y ayudara a identificar las necesidades que deben ser suplidas a la brevedad posible, estos diseños en lo posible deben ser los más explícitos posibles, desarrollándose de una manera clara, concisa y que sea entendible para los diferentes participes de

esta implementación, en este caso el equipo de desarrollo, el arquitecto del sistema y por supuesto para el cliente.

Existen diferentes métodos de plasmar una captura de requerimientos, en este caso se hizo uso de la captura por medio de diagramas de casos de uso, la cual es sencilla de entender y da un enfoque de que requiere cada usuario en la interacción con el sistema.

Así mismo las vistas 4+1 permiten mostrar el funcionamiento del sistema desde diferentes vistas las cuales son representadas por diagramas UML, estos diagramas no son una camisa de fuerza, simplemente ayudan entender lo que se quiere y espera del sistema, hay muchas maneras de representar estos diagramas, pero es el arquitecto el que debe identificar cual es la manera más clara en la cual puede plasmar el diseño del sistema.

La introducción de la bitácora de funcionamiento fue una adición significativa para conservar los registros de mantenimiento en la empresa. A causa de no contar con un sistema de registros los procesos para la solución de problemas reincidentes eran lentos y costosos debido a que siempre debían ser atendidos por un especialista ajeno a la empresa.

La implementación de los diagramas 4+1 permiten tener una visión global de todos los procesos realizados en la empresa, de esta manera la persona encargada de tomar decisiones podrá ser más asertiva a la hora de realizar un cambio en los procesos y de administrar los recursos disponibles para un uso en particular.

Trabajo futuro

El desarrollo del sistema de información en su fase uno fue completado, la siguiente fase a realizar será: definir los entornos de desarrollo, el equipo de trabajo y la implementación del sistema.

5 BIBLIOGRAFIA

Barajas, S., & Carrillo, E. (2014). *modelo de arquitectura de software para un prototipo de aplicación móvil integrado con realidad aumentada basado en servicios de localización*. https://repository.unab.edu.co/bitstream/handle/20.500.12749/3351/2014_Articulo_Sergio_Suarez_Barajas.pdf?sequence=2&isAllowed=y

Batalle, P. (2015). *los sistemas de información en la organización y gestión de la universidad.* https://www.raco.cat/index.php/Convivium/article/download/76435/98653/

Blancarte, O. J. (2020). *Introducción a la arquitectura de software – Un enfoque práctico* (Vol. 1).

Camacho, E., Cardeso, F., & Nuñez, G. (2004, April). *Arquitecturas de software*. http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf

Cervantes, H. (n.d.). *Arquitectura de Software | SG Buzz*. Retrieved October 13, 2021, from https://sg.com.mx/revista/27/arquitectura-software

Dhaduk, H. (2020, July 4). *Patron de Arquitectura*. https://www.simform.com/blog/software-architecture-patterns/

Kruchten, P. (1995). *Planos arquitectónicos - la vista "4 + 1" Modelo de arquitectura de software*. https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

Mendoza, R., & Iriarte, J. (2006). *Diseño y prototipo de una aplicación cliente-servidor que permita ejecutar comandos básicos en un servidor remoto desde un dispositivo móvil*. https://biblioteca.utb.edu.co/notas/tesis/0032773.pdf

Moya, E., & Callejas, L. (2015). *Análisis diseño e implementación de un sistema degestión de Talento Humano en la comunidad Salesiana San Juan Bosco.* https://dspace.ups.edu.ec/handle/123456789/10342

Villegas, N. (2020). *Importancia de la arquitectura de software en las organizaciones*. https://www.icesi.edu.co/unicesi/todas-las-noticias/1949-importancia-de-la-arquitectura-de-software-en-las-organizaciones

6 ANEXOS

A & T
ARENAS Y TRITURADOS S.A.S

Medellín, 22 de noviembre de 2021.

Doctor:
LUIS ARMANDO PORTILLA GRANADOS
Director Del Programa Ingeniería De Sistemas
Universidad de Pamplona

Asunto: Finalización de la práctica profesional.

Cordial saludo,

A&T ARENAS Y TRITURADOS S.A.S, identificada con NIT. **901.023.744-9**, le agradece la colaboración que ha recibido por parte del estudiante **JAVIER LEONARDO GARZÓN PRADA**, identificado con cédula de ciudadanía **Nº 1.053.606.303**, quien desarrolló su práctica profesional, cumpliendo sus objetivos de manera satisfactoria en nuestra compañía, apoyando el área de gestión de información y llevando a cabo el proyecto: "DISEÑO ARQUITECTONICO DE SOFTWARE PARA LA EMPRESA AYT ARENAS Y TRITURADOS.", el cual representa un avance importante para nuestra organización debido a las ventajas que ofrece en términos de manejo y disponibilidad de la información, durante los meses que estuvo con nosotros se destacó por su, responsabilidad, compromiso y por su interés en mejorar nuestros procesos informáticos con sus ideas innovadoras.

Estaremos atentos a cualquier solicitud.

Cordialmente,

JUAN ESTEBAN JIMENEZ BUILES
GERENTE
A&T ARENAS Y TRITURADOS S.A.S
E- mail: gerencia@arenasytriturados.com.co