Instituto Tecnológico de Costa Rica

Escuela Ingeniería en Computación

“Signage System”

Primer Informe de práctica profesional

De Ingeniería en Computación

“Brian Salazar Sánchez”

Sede San Carlos, I semestre 2017

**Tabla de Contenidos**

1) Introducción 3

2) Contexto del Problema 3

3) Descripcion del Problema 5

3.1) Enunciado del Problema 5

3.2) Enunciado de la Solución 6

3.3) Descripcion de los Stakeholders 7

3.4) Necesidades y Espectativas 7

3.5) Suposiciones y Dependencias 8

3.6) Requerimientos no Funcionales 8

4) Estado del Arte 9

5) Analisis de Riesgos 15

6) Objetivos y Alcances del Sistema 17

6.1) Objetivo General 17

6.2) Objetivos Especificos 18

6.3) Requerimientos Funcionales 19

7) Productos de la Fase de Conceptualización 20

7.1) Diagramas de Casos de Uso…………………………………….…………………20

7.2) Especificación de casos de uso 21…………………………….………………….21

7.3) Modelo conceptual (del dominio del problema) …………………………….…….25

7.4) Arquitectura…………………………….…………………………….………….…...26

7.5) Glosario .....…….………………………...………………………………….……….27

8.0) Plan de Trabajo…………………………………………………………………………28

**Tabla de Figuras**

Figura 2.1 (Organigrama Avantica San Carlos) 4

Figura 3.3.1 (Descripción Stakeholder) 7

Figura 3.4.1 (Necesidades de Visibilidad) 7

Figura 3.4.1 (Necesidades de Información) 8

Figura 5.1 (Riesgo Cambio en el API 15

Figura 5.2 (Riesgo Tiempo) 16

Figura 5.3 (Riesgo Requerimientos Confusos) 17

Figura 6.3.1 (Requerimientos Funcionales) 19

Figura 7.1.1 (Diagrama Casos de Uso) 20

Figura 7.2.1 (CU-01) 21

Figura 7.2.2 (CU-02) 22

Figura 7.2.3 (CU-03) 24

Figura 7.4.1 (Arquitectura) 26

Figura 8.1 (Plan de Trabajo) 28

**1) Introducción.**

Es muy importante conocer todo el entorno que envuelve el proyecto, esto tanto para facilitar el desarrollo del mismo como para entender mejor lo que el problema conlleva y de esta forma encontrar una mejor solución, además de poder avanzar con objetivos claros, en este documento se pretende aclarar y definir el proceso que se va a seguir para el desarrollo de la solución.

**2) Contexto del proyecto.**

Avantica se especializa en colaborar con compañías establecidas o emergentes para crear productos innovadores. El software se desarrolla en conjunto con los clientes y las relaciones que se mantienen son usualmente de largo plazo y basadas en metodologías rigurosas e ingeniería de calidad.

Es una empresa de software a la medida, pero también brinda otros servicios como aseguramiento de la calidad, préstamo de colaboradores para trabajos externos, entre otros.

La empresa se subdivide organizacionalmente de la siguiente forma:

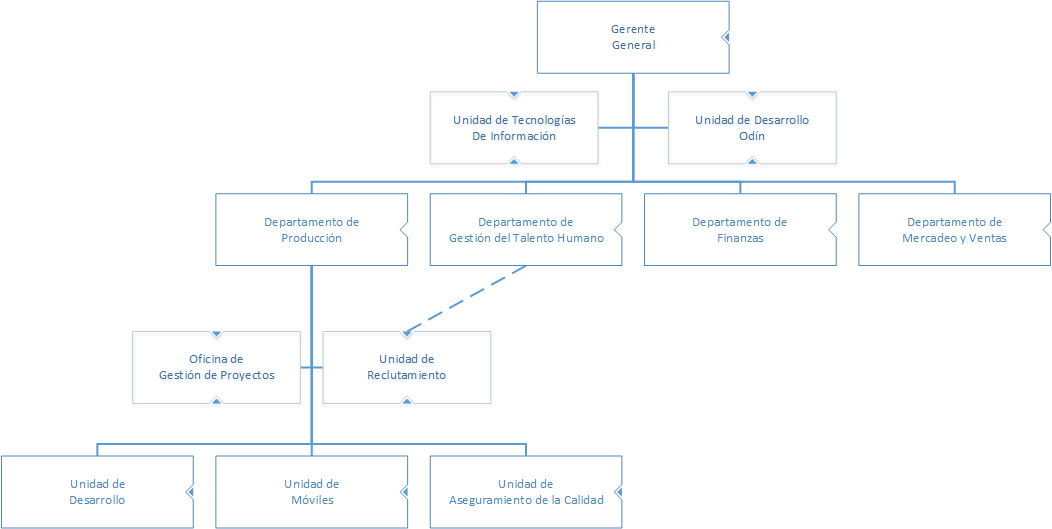
****

Figura 2.1 (Organigrama Avantica San Carlos)

Donde la mayoría de divisiones son de administración y ventas, luego están los 3 departamentos que se ven más envueltos con los proyectos como lo son las unidades de desarrollo, de móviles y de aseguramiento de la calidad.

**Unidad de Aseguramiento de la Calidad:** Son los encargados poner a prueba las aplicaciones para verificar si lo que hace la aplicación es concordante con lo que se pide en los requerimientos y además de que se realice de forma correcta, sin problemas de seguridad, rendimiento ni algún otro inconveniente.

**Unidad de Móviles:** Son los encargados de desarrollar los proyectos enfocados únicamente en dispositivos móviles, desde celulares hasta tabletas.

**Unidad de Desarrollo:** Esta unidad se encarga tanto de proyectos de escritorio como de proyectos web, en esta unidad los conocimientos deben de ser muy variados porque se trabaja con muchos tipos de tecnologías.

El proyecto a realizar por mi persona se realizará en la unidad de desarrollo ya que es un proyecto de enfoque web basado en dos pantallas que actualmente se tienen dónde se muestran los proyectos en ejecución y el estado del build de los mismos, se espera mejorar la información a mostrar para aumentar la visibilidad empresarial.

**3) Descripción del Problema.**

**3.1) Enunciado del problema.**

Como toda organización, Avantica no está exenta de los desafíos que representa el tema de comunicación y exposición de la información a sus colaboradores. Aunque existen canales en uso como el email, chats internos e incluso signos visuales como carteles o anuncios impresos, se vuelve cada vez más importante tener visibilidad inmediata.

Esto afecta a todos los colaboradores de la empresa ya que la visibilidad y la comunicación inmediata es casi nula para prácticamente todos, lo cual provoca un impacto negativo a la unión y el conocimiento empresarial lo que impide que se avance en conjunto, conociendo lo que se está realizando en la empresa y lo que se está buscando mejorar.

**3.2) Enunciado de la Solución.**

En busca de una solución se decide implementar el Signage System un proyecto que se enfocará en aumentar esa visibilidad entre los colaboradores de la empresa de una forma más apegada a la tecnología actual, para esto se pretende colocar pantallas en lugares estratégicos de la empresa donde con estas se podrá comunicar eventos importantes o simplemente información de interés. Para empezar con el desarrollo de la idea se pretende en primera instancia mostrar información en tiempo real sobre los proyectos que se realizan en la empresa ya que muchas veces los colaboradores solo tienen conocimiento del proyecto en el que ellos están trabajando, por esto se pretende mostrar en las pantallas los proyectos con diferentes estados de ejecución, comercial, de calidad y avance del mismo, además de mostrar el cliente con una descripción de este y del producto, además de mostrar los colaboradores involucrados en el proyecto con su respectivo puesto, esto aumentaría en gran instancia la visibilidad empresarial en de temas esenciales al negocio, desde cuál es la calidad que se está entregando a los clientes, así como información importante para la organización.

**3.3) Descripción de los Stakeholders.**

|  |  |
| --- | --- |
| Nombre | Erick Vargas |
| Departamento |  |
| Labores |  |
| Responsabilidades | Project Manager del Proyecto.  Verificación del adecuado seguimiento del cronograma.  Validación del trabajo realizado en los avances semanales. |
| Criterios de Éxito |  |

Figura 3.3.1 (Descripción Stakeholder)

**3.4) Necesidades y Expectativas**

|  |  |
| --- | --- |
| Necesidad | Visibilidad |
| Prioridad | Alta |
| Problema | Falta de conocimiento en temas de interés |
| Solución Actual | Enviar correos electrónicos |
| Solución Propuesta | Mostrar la información en pantallas informativas ubicadas en puntos estratégicos de la empresa. |

Figura 3.4.1 (Necesidades de Visibilidad)

|  |  |
| --- | --- |
| Necesidad | Conocer los involucrados en los proyectos. |
| Prioridad | Alta |
| Problema | Desconocimiento en lo que realizan los demás colaboradores en la empresa |
| Solución Actual | No hay |
| Solución Propuesta | Mostrar en las pantallas los proyectos en ejecución con los nombres, foto y puesto de cada uno de los involucrados en el proyecto. |

Figura 3.4.1 (Necesidades de Información)

**3.5) Suposiciones y dependencias**

Las Apis de Jenkins y Sonarqube no cambiaran en el proceso de desarrollo.

**3.6) Requerimientos no funcionales**

Las vistas deben ser web responsive, adaptarse a cualquier resolución (dentro de lo razonable).

**4) Estado del Arte**

La tecnología avanza a pasos agigantados y en la rama del software no es la excepción, cada día se ponen al mercado nuevos frameworks con mejores características o diferentes a los ya existentes, además de un mejoramiento y actualización de los frameworks más usados, todo esto para mejorar y simplificar la forma en que se desarrollan las aplicaciones, además de que se vuelve una obligación el cambio constante ya que la demanda de páginas web más robustas e interactivas es cada vez mayor.

Debido a todo esto y enfocándose en desarrollo web, existen diversos estilos de programación, en primera instancia se abusa en exceso del servidor ya que los navegadores no eran suficiente para poder interpretar de manera ágil y eficiente el código, por esto cada cambio que se realizaba en el DOM lo ejecutaba el servidor por lo que la pagina necesitaba actualizarse constantemente, actualmente los navegadores son mucho más potentes por lo que las paginas se reenderizan desde el cliente en el mismo navegador lo que le permite al usuario una mejor experiencia.

Debido a estos constantes cambios y a la gran cantidad de opciones que existen para lograr un desarrollo web optimo, se vuelve muy difícil lograr escoger los mejores frameworks actuales o los más indicados para la aplicación a desarrollar, para desarrollar el Signage System se escogieron las siguientes tecnologías:

**Nodejs:** Es uno de los frameworks más utilizados actualmente para trabajar del lado del servidor, además brinda excelentes beneficios al ser un entorno de ejecución para javascript lenguaje ampliamente flexible ya que nodejs utiliza ECMAScript, lo que nos permite utilizar las características de javascript del lado del servidor.

Hernán Alejandro Quintana Cruz se refiere a como no se reutilizaba el código y como ahora si se apunta a esto:

Por estas consideraciones, se comenzó a pensar en una

arquitectura web que permita ejecutar un mismo código,

tanto en el cliente como en el servidor. De esta manera,

es perfectamente factible reutilizar un código que genere,

modifique y renderice una página web en el servidor

del cliente, dependiendo del contexto. Es así como nace

el concepto de una aplicación isomórfica. (Cruz, 2015)

Por esto se ha vuelto tendencia la utilización de node con algún framework de front-end, por lo general angular2 o react ya que se adaptan perfectamente para lograr en muchos casos reducción de código ya que tanto el servidor como el cliente lo pueden interpretar.

Nodejs trabaja con un npm que es el manejador de paquetes o librerías más grande del mundo lo que permite utilizar diversas ventajas y facilidades que nos brindan estas, la mayoría se encuentran ampliamente documentadas lo que se hace más fácil su implementación, por esto la escogencia de node, al brindar distintos tipos de librerías ofrece un soporte amplio y un mejor manejo de las funciones a realizar y constantemente se agregan nuevas librerías o se mejoran funciones.

Un caso importante de node es el que menciona Massimo Siboldi sobre la utilización de funciones asíncronas:

The other method of dealing with this bottleneck is through

what is called asynchronous execution. In one thread, a

piece of code initializes an operation and waits for an event

to complete it, allowing execution of other pieces of code.

Javascript follows this pattern, which allows low memory

usage and simpler code. (Siboldi, 2014)

Nodejs no trabaja con los hilos del sistema operativo, cada vez que se establece una conexión se ejecuta el callback, lo cual es diferente a la mayoría de servidores que trabajan con los hilos, pero esto puede ser una ventaja ya que trabajar con hilos es muy tedioso y en algunas ocasiones son relativamente ineficientes.

Además, nodejs es una muy buena opción porque es muy liviano y eficiente ya que es orientado a eventos y utiliza operaciones de entrada y salida sin bloqueo en casi todas sus funciones, ya que la mayoría son asíncronas.

**Reactjs:** Se escogió utilizar reactjs para realizar las vistas del front-end ya que es uno de los frameworks más utilizados actualmente, ya que brinda una increíble flexibilidad a la hora de crear componentes para las vistas, además de que realiza un flujo de datos muy claro. Su flexibilidad nos permite crear cualquier tipo de componente y a nuestro gusto y luego poder reutilizar ese componente en cualquier otro lugar de nuestra página, lo que nos reduce la posibilidad de reutilización de código y nos da un mejor entendimiento de la aplicación.

Cabe a señalar que es de los frameworks junto a Angular2 que mejor se adaptan a nodejs, facilitando ampliamente la utilización del mismo código tanto desde el servidor como en el cliente.

Cabe a mencionar Reactjs es un framework basado específicamente en las vistas y presenta mejor rendimiento comparado con otros frameworks ya que realiza un mejor manejo del DOM, utilizando una característica llamada virtual DOM que es una copia en memoria donde se realizan los cambios en primera instancia y se comparan con el DOM principal para cambiar solo los componentes diferentes, diferente a lo realizado por otros frameworks que reenderizan el DOM completo con cada cambio.

Como mejor lo explica Ramiro Blázquez González:

Su principal insignia es el Virtual-DOM. Este Virtual-DOM,

reenderizan partes del DOM únicamente cuando es

necesario, ahorrando costes de computación y hacienda

que la pagina repinte estrictamente aquello que ha cambiado.

Por tanto, ahora tenemos dos árboles DOM, el principal que

es el que se pinta en el navegador y el Virtual-DOM que es

el de React, que es un objeto JavaScript, y que es mucho

más ligero. Cuando se actualiza el estado, se ejecuta un

algoritmo muy eficiente que compara por referencia partes

del DOM para comprobar si ha habido diferencias y solo se

reemplazan o cambian aquellas partes nuevas o que han

mutado. (Gonzáles, 2016)

Otra de las características más importantes de react es el hecho de buscar la unión de los archivos html con los archivos javascript por eso podemos escribir código javascript y utilizarlo en nuestros componentes, esto para manejar un mejor reenderizado de la página.

Además, bootstrap proporciona una rescritura especial para react lo que brinda mayor facilidad a la hora de proporcionar estilos.

**React-Bootstrap:** Para lograr que el diseño de la página fuese lo mayormente responsive que se podía, se decidió utilizar react-bootstrap que es una reescritura especialmente hecha para react, de uno de los frameworks más populares para lograr componentes responsive y estéticos en las vistas como lo es bootstrap.  
  
Al ser una reescritura especialmente para react se logra una clara reducción del código que se utiliza para crear un componente.

**Aplicación Nativa:**

button(size=SMALL, color=GREEN, text="Something", onClick=someCallback)

**Bootstrap:**

<button id="something-btn" type="button" class="btn btn-success btn-sm">  
 Something  
</button>

**React-Bootstrap:**

<Button bsStyle="success" bsSize="small" onClick={someCallback}>  
 Something  
</Button>

Como software para evaluar la calidad de las aplicaciones desarrolladas se utilizó Sonarqube el cual es muy importante ya que:

En ellos se toma la posición de asegurar el software durante

la fase de diseño y se critica como las compañías hacen

frente a la seguridad de sus programas por medio de parches,

culpando directamente la implementación de un sistema de

desarrollo que se enfoca principalmente en añadir características

y no en la calidad y la seguridad del código. (Delgado, 2015)

**Sonarqube:** Es un analizador de código externo de Avantica que se encarga de detectar distintos tipos de vulnerabilidades en el código, así como las líneas con clean code y muchas características más, en este caso de sonarqube se tomará el porcentaje de cobertura del código que contenga unit test, el porcentaje debe de ser de más del 70% para que el estado de cobertura sea el óptimo.

En otras palabras, Sonarqube es:

Sonarqube es una plataforma de código abierto usada por los equipos de desarrollo para controlar la calidad del código. Sonar fue desarrollado con el principal objetivo de hacer accesible la administración de la calidad del código con un mínimo esfuerzo. Como tal, Sonar contiene en su núcleo de funcionalidades un analizador de código, una herramienta de reportes, un módulo que detecta defectos y una función para regresar los cambios realizados en el código. (Delgado, 2015)

**Jenkins:** Es un sistema externo a Avantica que funciona como servidor de automatización generando diversos beneficios como el build automático y testing de las aplicaciones. A través de jenkins se tomará solamente el estado del build automático, si el estado del build está bien o si alguien al juntar código nuevo daño el build del proyecto.

**5) Análisis de los Riesgos**

|  |  |
| --- | --- |
| **Descripción** | Cambios en el api |
| **Categoría** | Tecnologico |
| **Causa** | Actualizacion o mejoramiento del API |
| **Impacto** | 80 |
| **La probabilidad de ocurrencia** | 0.1 |
| **Exposición que se tendrá ante el riesgo** | 8 |
| **Estrategia de Evasión** | - |
| **Estrategia de Mitigación** | Utilizar buenas practicas de programacion para que los cambios a realizar sean minimos y facil de implementar. |

Figura 5.1 (Riesgo Cambio en el API)

|  |  |
| --- | --- |
| **Descripción** | Tiempo Reducido |
| **Categoría** | Tiempo |
| **Causa** | Un error o problema en el codigo. |
| **Impacto** | 50 |
| **La probabilidad de ocurrencia** | 0.1 |
| **Exposición que se tendrá ante el riesgo** | 5 |
| **Estrategia de Evasión** | Seguir el plan de trabajo |
| **Estrategia de Mitigación** | Invertir más tiempo, hasta recuperar el atraso. |

Figura 5.2 (Riesgo Tiempo)

|  |  |
| --- | --- |
| **Descripción** | Requerimientos Confusos |
| **Categoría** | Personas |
| **Causa** | Falta de Comunicación |
| **Impacto** | 90 |
| **La probabilidad de ocurrencia** | 0.6 |
| **Exposición que se tendrá ante el riesgo** | 54 |
| **Estrategia de Evasión** | Manterse en comunicacion con el cliente |
| **Estrategia de Mitigación** | Preguntar al cliente, hasta aclarar el requerimiento e invertir tiempo extra en modificarlos. |

Figura 5.3 (Riesgo Requerimientos Confusos)

**6) Objetivos y Alcances del Sistema.**

**6.1) Objetivo general**

Mejorar la comunicación a lo interno de la empresa Avantica, implementando un sistema de comunicación interna para aumentar el acercamiento de los colaboradores con los valores corporativos, principalmente el de calidad.

**6.2) Objetivos específicos**

1. Crear un componente para la integración del consumo de datos de sistemas internos y despliegue de información en los canales de distribución.
2. Crear un componente tipo cliente que será implantado en Raspberry para mostrar la información definida en el sistema central.
3. Implantar un sistema para poder configurar lo que se quiere mostrar.

**6.3) Requerimientos Funcionales**

|  |  |
| --- | --- |
| Código | Descripción |
| RF-01 | El sistema deberá desplegar una vista de todos los proyectos donde se muestre el estado del build, el estado de cobertura, estado de ejecución y estado comercial de cada proyecto. |
| RF-02 | El sistema deberá mostrar una vista con información detallada de un proyecto como:   1. Foto y descripción del cliente 2. Descripción del proyecto 3. Estado del build 4. Estado de la cobertura 5. Estado Comercial 6. Estado de Ejecución 7. Foto, nombre y puesto de los involucrados en el proyecto   Como información adicional, se puede mostrar si el proyecto está en fase de garantía o si ya se cerró. |
| RF-03 | El sistema deberá mostrar la razón por la que un estado se encuentra en rojo o en amarillo. |
| RF-04 | El sistema deberá de mostrar un panel administrativo donde se podrán configurar las siguientes características:   1. Integración completa con base de datos, Jenkins y Sonar 2. Usuarios administrativos 3. Tiempo de rotación entre vistas 4. Proyectos a mostrar 5. Editar Proyectos (logo del cliente, descripción del cliente, descripción del proyecto) |
| RF-05 | El sistema deberá de rotar las vistas automáticamente cada cierto tiempo. |

Figura 6.3.1 (Requerimientos Funcionales)

**7) Productos de la Fase de Conceptualización.**

**7.1) Diagramas de casos de uso**

Caso de uso Siganage

Figura 7.1.1 (Diagrama Casos de Uso)

**7.2) Especificación de casos de uso**

|  |  |
| --- | --- |
| **Código** | CU-01 |
| **Nombre del Caso de uso** | Ver Vista General |
| **Autores** | Brian Salazar Sánchez |
| **Actores** | El sistema |
| **Precondiciones** | Se debe de estar ejecutando el servidor. |
| **Flujo principal** | El sistema recibe la URL /main  El sistema muestra la vista general con una lista de todos los proyectos con sus respectivos estatus.  Fin de caso de uso. |
| **Flujos alternativos** |  |
| **Postcondiciones** |  |

Figura 7.2.1 (CU-01)

**Requisitos Implementados:**

El sistema deberá mostrar una vista de todos los proyectos donde se muestre el estado del build, el estado de cobertura, estado de ejecución y estado comercial de cada proyecto.

|  |  |
| --- | --- |
| **Código** | CU-02 |
| **Nombre del Caso de uso** | Ver Vista Detallada |
| **Autores** | Brian Salazar Sánchez |
| **Actores** | El sistema |
| **Precondiciones** | Se debe de estar ejecutando el servidor. |
| **Flujo principal** | El sistema recibe la URL /detailed  El sistema muestra la vista detallada con la información de los proyectos, el resumen de estatus y los colaboradores involucrados.  Fin de caso de uso. |
| **Flujos alternativos** |  |
| **Postcondiciones** |  |

Figura 7.2.2 (CU-02)

**Requisitos Implementados:**

El sistema deberá mostrar una vista con información detallada de un proyecto como:

1. Foto y descripción del cliente
2. Descripción del proyecto
3. Estado del build
4. Estado de la cobertura
5. Estado Comercial
6. Estado de Ejecución
7. Foto, nombre y puesto de los involucrados en el proyecto

Como información adicional, se puede mostrar si el proyecto está en fase de garantía o si ya se cerró.

|  |  |
| --- | --- |
| **Código** | CU-03 |
| **Nombre del Caso de uso** | Ver Vista de Configuración |
| **Autores** | Brian Salazar Sánchez |
| **Actores** | El sistema |
| **Precondiciones** | Se debe de estar ejecutando el servidor. |
| **Flujo principal** | El sistema recibe la URL /admin  El sistema muestra la vista de configuración.  Fin de caso de uso. |
| **Flujos alternativos** |  |
| **Postcondiciones** |  |

Figura 7.2.3 (CU-03)

**Requisitos Implementados:**

El sistema deberá de mostrar un panel administrativo donde se podrán configurar las siguientes características:

1. Integración completa con base de datos, Jenkins y Sonar
2. Usuarios administrativos
3. Tiempo de rotación entre vistas
4. Proyectos a mostrar
5. Editar Proyectos (logo del cliente, descripción del cliente, descripción del proyecto)

**7.3) Modelo conceptual (del dominio del problema)**

Introducción: El dominio es la comunicación global en un sentido, mostrar información importante a un gran número de personas puede ser de gran beneficio en este caso para la empresa.

Conocimientos generales sobre el dominio: Al comunicar un mensaje a un gran número de personas, es necesario tener en cuenta los diferentes métodos que existen para comunicar lo que se quiere, motivar a los receptores para que estén siempre atentos al canal donde recibirán el mensaje, ya que este es uno de los mayores problemas.

Clientes y usuarios: Los usuarios en este caso serán principalmente los receptores, que serán los compañeros de la empresa que se encuentran principalmente en sus cubículos en las habitaciones, por lo que es necesario que la información se muestre en lugares estratégicos, en un canal visible, cada vez que se entre o salga de la habitación y que sea fácilmente accesible y visible.

Tareas y procedimientos actualmente realizados: La información del proyecto se muestra actualmente a través de dos pantallas ubicadas en cada habitación y la información importante se envía a través de correos electrónicos.

**7.4) Arquitectura.**

Untitled Diagram (2)

Figura 7.4.1 (Arquitectura)

El sistema se divide en tres secciones, la sección de fuente de datos es responsable de proporcionar los datos que se mostrarán en las vistas a través de llamadas a las API, la sección de vistas es responsable de mostrar esos datos al usuario, mientras que la sección de base de datos es encargada de almacenar algunos datos necesarios como datos para la integración de los diferentes sistemas. El panel de configuración es el único responsable de guardar datos en la base de datos, pero los datos almacenados se utilizan en las otras vistas.

**7.5) Glosario de términos.**

**API:** Conjunto de funciones que cumplen con muchas funciones con el fin de ser utilizadas por otro software. Application Programming Interface.

**Back-end:** Lado del servidor. Es la labor de ingeniería que compone el acceso a bases de datos y generación de plantillas del lado del servidor.

**Build:** Construcción de todas las partes de la aplicación.

**Callback:** Una devolución de llamada es una función "A" que se usa como argumento de otra función "B". "B" se ejecuta hasta que termine la ejecución de "A".

**Clean Code:** Código escrito de acuerdo a ciertos estándares de programación.

**Cliente:** Es una aplicación que pide datos, servicios, funciones a otra(servidor).

**DOM:**  El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

**Frameworks:** Implementación de un lenguaje para utilizarlo de forma mas sencilla.

**Front-end:** En diseño de software el front-end es la parte del software que interactúa con el o los usuarios.

**Hilos:** Un programa en ejecución, entre mas hilos, se puede ejecutar mas el mismo programa al mismo tiempo.

**Librerías:** Son paquetes de códigos que se pueden utilizar para nuestro beneficio.

**Servidor:** Aplicación que envía datos y demás servicios a uno o varios clientes.

**Software:** Programas o Aplicaciones para computadoras y otros dispositivos electrónicos.

**Stakeholders:** Involucrados en el proyecto.

**8) Plan de trabajo**

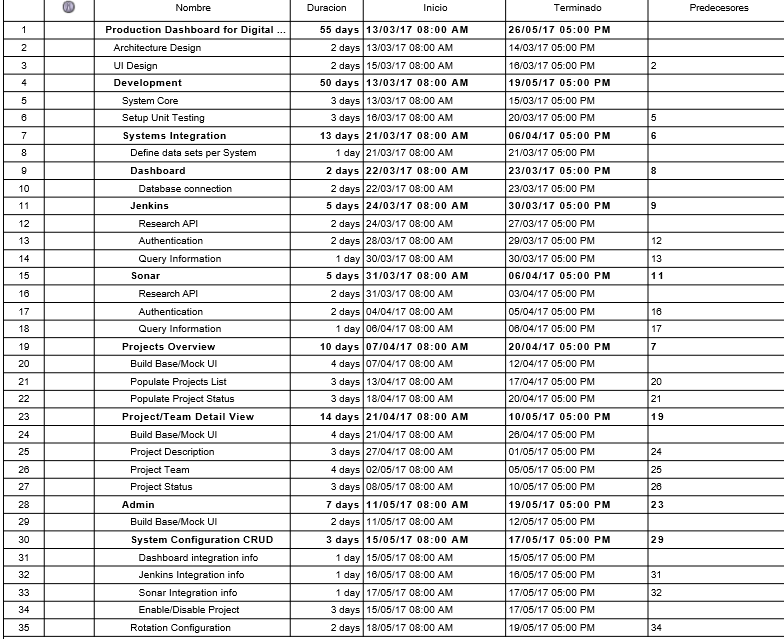


Figura 8.1 (Plan de Trabajo)

Se adjunta completo en la carpeta.