**Documento de arquitectura de software**

Documento de arquitectura de software aplicativo Escaleta-Teclado GC

2024

Documento oficial arquitectónico del proyecto Escaleta-teclado GC, que tiene como fin brindar una vista macro de la estructura del proyecto que sirva como base para el entendimiento del comportamiento del sistema y de la estructura interna para el usuario final.

#### Control de Cambios

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Creación del documento** | | |  | |
| **Autor** | Wílmer E. León | | | **Fecha** | 12 de septiembre del  2024 |
| **Revisado por** | **Nombre** |  | | **Fecha** |  |
| **Cargo** |  | |
| **Aprobado por** | **Nombre** |  | | **Fecha** |  |
| **Cargo** |  | |
| **Versión** | **Descripción** | **Autor** | **Aprobado Por** | **Fecha Aprobación** | |
| 1.0 | Creación del documento | Wílmer E. León |  |  | |
|  |  |  |  |  | |

Tabla de contenido

[1. Introducción 4](#_Toc177074257)

[1.1. Propósito 4](#_Toc177074258)

[1.2. Alcance 4](#_Toc177074259)

[1.3. Definiciones, siglas y abreviaturas 4](#_Toc177074260)

[1.4. Referencias 4](#_Toc177074261)

[1.5. Vista global 5](#_Toc177074262)

[2. Macroarquitectura 5](#_Toc177074263)

[2.1. Modelo multicapa 5](#_Toc177074264)

[2.2. Metas y restricciones arquitectónicas 6](#_Toc177074265)

[3. Vista física 8](#_Toc177074266)

[Atributos de calidad 8](#_Toc177074267)

[Notación del diagrama informal físico: 8](#_Toc177074268)

[4. Vista funcional o lógica 9](#_Toc177074269)

[4.1. Diagrama de componentes 9](#_Toc177074270)

[Atributos de calidad 10](#_Toc177074271)

[5. Vista de despliegue 12](#_Toc177074272)

[5.1. Diagrama de despliegue 12](#_Toc177074273)

[Atributos de calidad 12](#_Toc177074274)

[6. Vista de procesos 13](#_Toc177074275)

[6.1. Diagrama de procesos 13](#_Toc177074276)

[Atributos de calidad 13](#_Toc177074277)

[Diagrama de actividades: Proceso de selección de plantilla 13](#_Toc177074278)

[Diagrama de actividades: Proceso de gestión de zócalos 14](#_Toc177074279)

[Diagrama de actividades: Proceso de gestión de zócalos 14](#_Toc177074280)

# 1. Introducción

## 1.1. Propósito

El propósito de este documento es describir la arquitectura del sistema *Escaleta-Teclado GC*. Este sistema está diseñado para gestionar la creación y edición de escaletas de producción, permitiendo a los usuarios con roles específicos (productor y asistente de producción) interactuar con plantillas, asignar tiempos y gestionar contenido de manera eficiente.

## 1.2. Alcance

El sistema abarca la gestión de escaletas de producción, incluyendo la creación, edición y almacenamiento de plantillas. También incluye la integración con módulos de históricos y la capacidad de importar datos desde estos módulos. El sistema está diseñado para ser utilizado por productores y asistentes de producción en un entorno de emisión de contenido.

## 1.3. Definiciones, siglas y abreviaturas

ETgc: Aplicativo Escaleta-Teclado GC.

DCSHA: De ciclismo se habla así (escaleta y teclado para prototipo)

ID: Identificador único.

VTR: Video Tape Recorder.

TX: Transmisión.

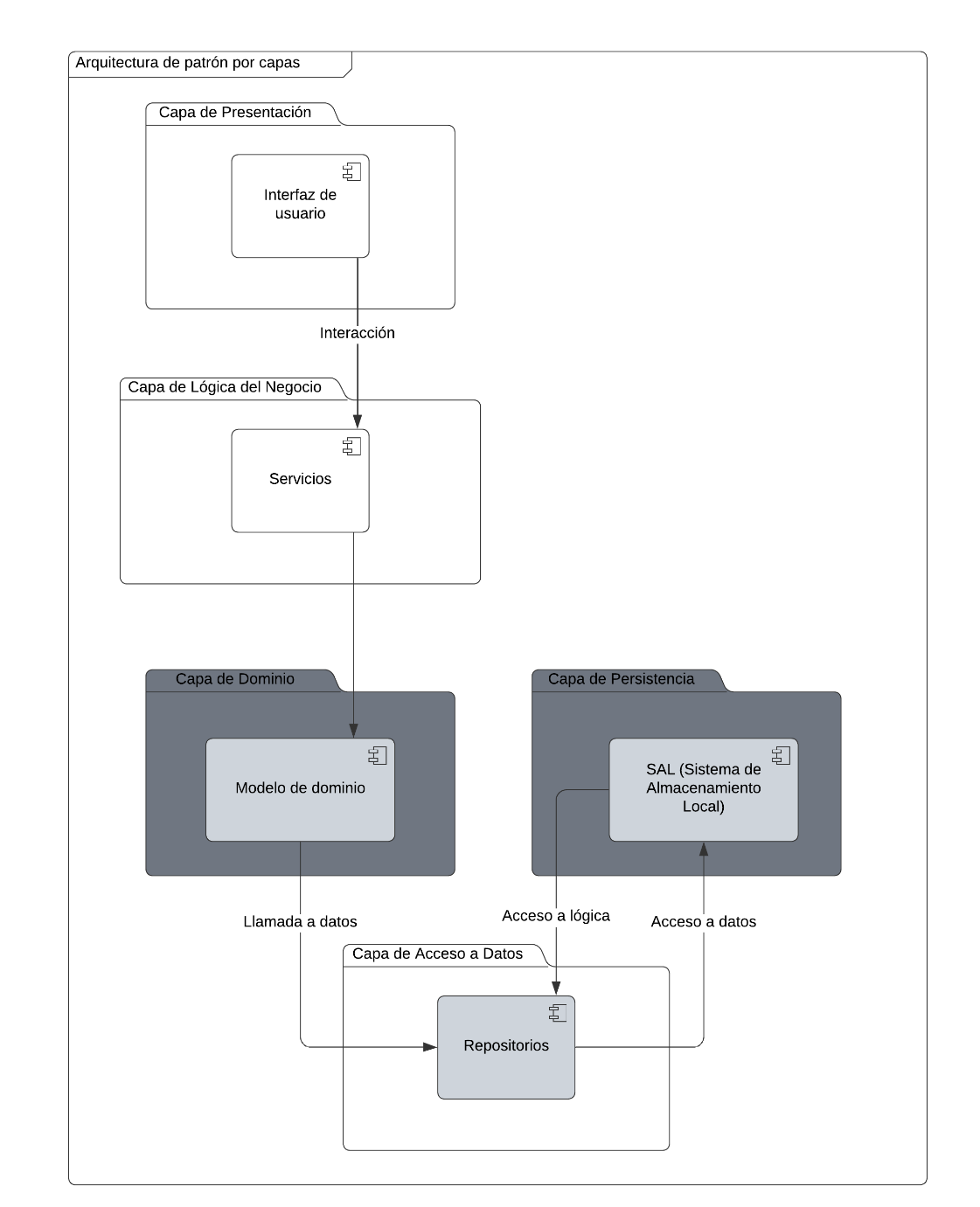
GC: Generador de Caracteres.

## 1.4. Referencias

|  |  |  |
| --- | --- | --- |
| **Documento** | **Versión** | **Fecha de la versión** |
| Carta de proyecto Estimación de tiempo y recursos | 1.0 | 10/09/2024 |
| Especificación de requerimientos | 1.0 | 10/09/2024 |
| Formatos Plan de Mejoramiento 1- 2 - 3 | 1.0 | 10/09/2024 |
| Enunciado de creación del diagrama de casos de uso | 1.0 | 12/09/2024 |

## 1.5. Vista global

El sistema *Escaleta-Teclado GC* está compuesto por varias capas que interactúan entre sí para proporcionar una funcionalidad completa de gestión de escaletas. Estas capas incluyen la capa de presentación, lógica de negocio, acceso a datos, persistencia, integración con Excel y gestión de imágenes.



# 2. Macroarquitectura

La macroarquitectura del sistema describe la estructura general y los componentes principales que lo conforman, así como sus interacciones.

## 2.1. Modelo multicapa

El modelo multicapa se compone de tres capas principales:

* **Capa de Presentación:** Interfaz de usuario que permite la interacción con el sistema.
* **Capa de Lógica de Negocio:** Procesos y reglas que gestionan la lógica del sistema.
* **Capa de Acceso a Datos:** Módulos que gestionan la persistencia y recuperación de datos.

## 2.2. Metas y restricciones arquitectónicas

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Atributos de Calidad ‘Observables’** | |  |
| **Atributo de calidad** | **Descripción** | **Tácticas / Patrón de arquitectura** | **Donde se aplica** |
| **Confidencialidad**  **Formato de archivo**  **(subcategoría seguridad)** | Capacidad del *software* de asegurar que cualquiera de sus características y/o contenidos estén enmascarados en los ejecutables que generan las terminales de Redacción para que sean leídas en la terminal de Máster. | **Mount Points (puntos de montaje):** Permite que diferentes sistemas de archivos sean accesibles desde un único sistema. | Se aplica en el manejo de los ejecutables entre las terminales con el aplicativo ETGC. Los archivos generados entre terminales solo son reconocibles por medio del aplicativo. |
| **Funcionalidad-subcategoría**  **Interoperabilidad** | Permitir estandarización de componentes que fueron desarrollados separadamente para luego integrarlos. | **Programación por Capas:** Separar el Diagrama de Clases usando el patrón de desarrollo multicapa. | Se aplica al separar el Diagrama de Clases usando el patrón de desarrollo multicapa. |
| **Performance** | Tiempo que requiere el sistema para responder a un evento o estímulo (tiempo de respuesta), o bien el número de eventos procesados en un intervalo de tiempo (*throughput*). | **Singleton:**  Garantiza que el control total de la aplicación la tiene el programador y no el cliente, y que el mismo puede recurrir a usar un *Template Method* si es necesario, garantizando la reutilización del código y la adaptación a nuevas necesidades. | Se aplica en la creación de los objetos y en los constructores del código del programa en los microservicios. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Atributos de calidad ‘No observables’** | | |
| **Atributo** | **Descripción** | **Tácticas / Patrón de arquitectura** | **Donde se aplica** |
| **Modificabilidad** | Es la habilidad de realizar cambios futuros al sistema. | **Patrón:**  GRAS  Planteamiento modular del sistema, división por capas lógicas de *software* con responsabilidades definidas de cada capa y protocolos de comunicación entre ellas utilizado en la construcción de los microservicios. | En el componente que se encarga de la carga, descarga y ejecución de archivos ejecutables entre terminales en la interfaz de gestión de archivos.  En la parte del sistema que interactúa con los usuarios para gestionar los archivos, directamente en la interfaz de usuario en la capa de presentación.  En la configuración del sistema que permite a los usuarios definir parámetros de operación.  En el sistema de pruebas que valida la funcionalidad del sistema. |
| **Mantenibilidad** | Es la capacidad de someter a un sistema a reparaciones y evolución Capacidad de modificar el sistema de manera rápida y a bajo costo. | **Patrón:**  Capas de *software*  Mediante este patrón permite que las modificaciones a los servicios web de integración sean más sencillas, dado que van a puntos específicos del código.  Estandarización en el nombramiento de tablas, módulos, clases, métodos de clase y variables a nivel  de diseño y programación. | En la interfaz de usuario, ya sea una aplicación de escritorio, una interfaz de línea de comandos o una aplicación web (cuando se haga la integración al servidor virtual).  Entre la capa de presentación y la lógica de negocio.  Dentro de la lógica, en la capa de Lógica del Negocio, que determina cómo se deben manejar los ejecutables.  En la interacción con el sistema de archivos, en la capa de Acceso a Datos.  Todos llegan a funcionar un flujo completo:   1. **Interacción del Usuario**: El usuario selecciona "Cargar Ejecutable" en la interfaz. 2. **Capa de Presentación**: Llama al método *load\_executable()* de *FileService.* 3. **Capa de Servicio**: FileService llama a *BusinessLogic* para validar el archivo. 4. **Capa de Lógica de Negocio**: Si el archivo es válido, *FileService* llama a *DataAccess* para ejecutar el archivo. 5. **Capa de Acceso a Datos**: *DataAccess* utiliza *os.system()* para ejecutar el archivo en el sistema. |
| **Portabilidad** | Capacidad de transferencia del producto de *software* de un | **Responsive design:**  Garantiza adaptar tu diseño al tamaño de pantalla del dispositivo y escalamiento manual de las ventanas. | Brindarle al usuario la Una forma de poder ingresar a la página desde un PC con diferentes resoluciones de pantalla |

# 3. Vista física

La vista física del sistema incluye los componentes físicos y su interconexión. El sistema se despliega en un servidor virtual que gestiona la base de datos y los módulos de históricos. Los usuarios acceden al sistema a través de terminales PC conectadas a la red.

**Diagrama informal de la vista física:** El Diagrama Informal de la Vista Física describe las características físicas de los componentes que integran la arquitectura del sistema Escaleta-teclado GC (ETgc).

Este diagrama contempla el uso del aplicativo por parte del usuario, quien accede a través de un navegador desde diversos dispositivos, que pueden incluir computadoras de escritorio, laptops y dispositivos móviles. El usuario ingresará a la página web del *software* *Escaleta-teclado GC*, que está alojado en un servidor de aplicaciones utilizando Internet (aún por definir y en caso de que se desee ir más allá de la entregable de prototipado). Además, el servidor cuenta con Oracle APEX (Application Express) como gestor de bases de datos.

### Atributos de calidad

**La escalabilidad:** El sistema *Escaleta-teclado GC* tiene la capacidad de crecer y adaptarse a un aumento en la carga de trabajo (número de solicitudes). Cada máquina tiene recursos limitados, lo que significa que puede manejar solo un número determinado de solicitudes simultáneamente. El sistema incorpora un patrón de diseño multicapa, separando los datos y la lógica de negocio de la interfaz de usuario. Esto permite que, al implementar nuevos módulos o funcionalidades, se realicen modificaciones de manera eficiente dentro de sus respectivas capas.

**Portabilidad**: El aplicativo web *Escaleta-teclado GC* es portátil, ya que implementa el patrón de diseño *Responsive Design*, con instalación local en cada terminal con Electron, de npm, permitiendo que se adapte a diferentes dispositivos con múltiples tamaños de pantalla, desde computadoras.

## Notación del diagrama informal físico:

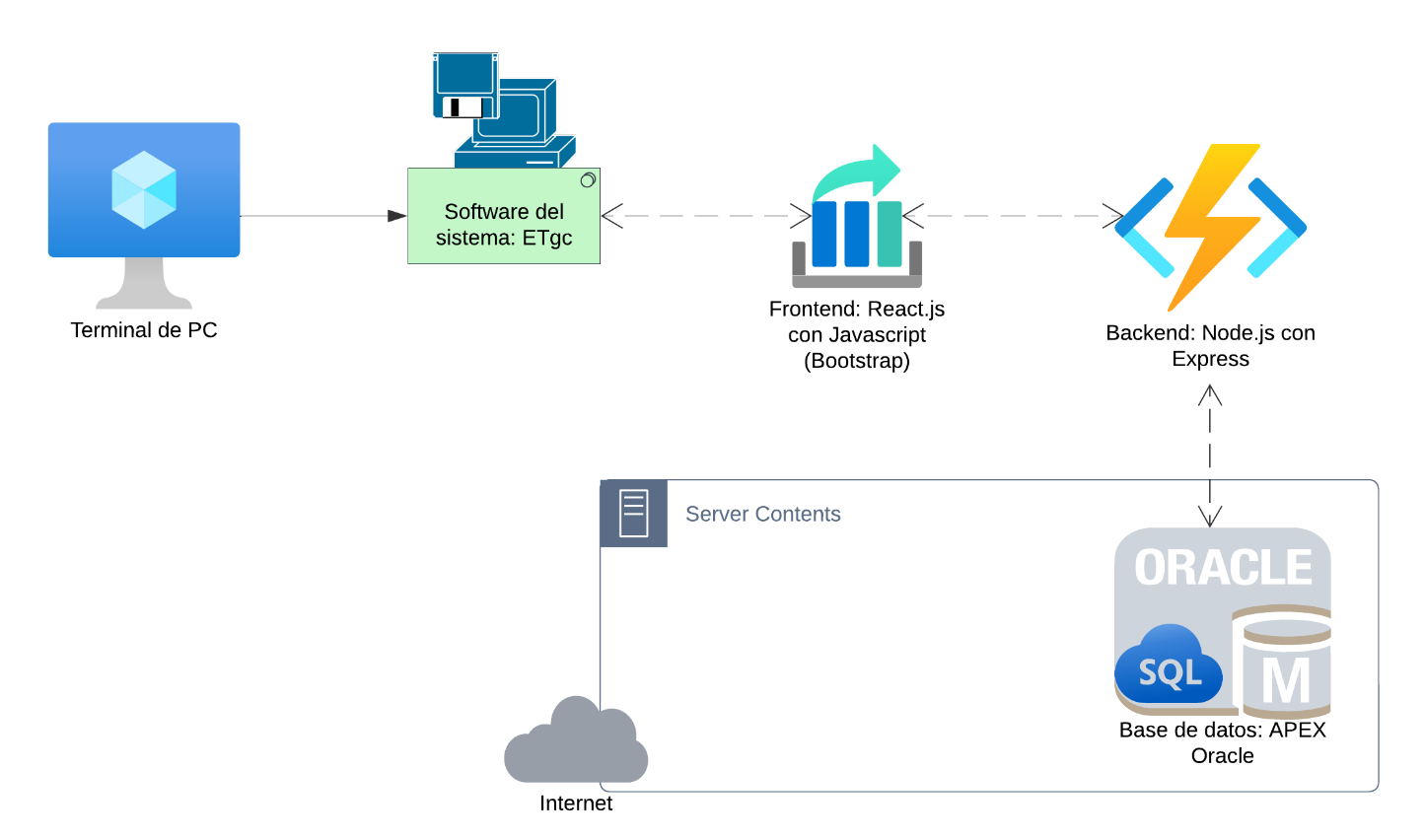
**Usuarios:** Personas que utilizan regularmente el servicio del sistema Escaleta-teclado GC.

**Computador (PC o portátil):** Dispositivo electrónico capaz de procesar datos que recibe del exterior (entradas) y proporcionar resultados (salidas). Este dispositivo puede tener conexión a una red de datos (terminal de Maestro | TX 2020.2 y láptop de asistencia de producción), ya sea de forma permanente o intermitente.

**Internet:** Conexión global que une todas las redes y computadoras distribuidas en el mundo. Es una red que integra todas las redes que utilizan el protocolo TCP/IP y que son compatibles entre sí (al menos en ciclos posteriores de la creación de software en donde se emplee el servidor virtual).

**Servidor:** Computador encargado de proporcionar información a una serie de clientes, que pueden ser tanto personas como otros dispositivos conectados (por restricciones de seguridad y, por el momento, se restringe y se opta, a futuro, por un servidor virtual). La información transferida puede incluir archivos de texto, imágenes, programas informáticos, bases de datos, entre otros.

**Aplicativo web:** Herramientas accesibles para los usuarios a través de un servidor web mediante un navegador de internet.



# 4. Vista funcional o lógica

## 4.1. Diagrama de componentes

El siguiente diagrama representa el marco de trabajo de la relación entre las diferentes dependencias del sistema Escaleta Teclado GC y los componentes que se encuentran dentro de esta arquitectura para la funcionalidad del aplicativo web. Este sistema está diseñado como una arquitectura multicapa, que incluye la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos. Esta estructura multicapa permite una mantenibilidad óptima y eficiente, ya que se puede brindar soporte a un punto específico sin necesidad de modificar el resto del sistema. Además, facilita la modificabilidad, permitiendo la inclusión de nuevos componentes en una línea de tiempo sin afectar lo que ya existe, lo que a su vez potencia la funcionalidad general del sistema.

En la capa de presentación, se utiliza React.js, con JavaScript (Bootstrap) para el desarrollo del *frontend*, lo que permite crear una interfaz de usuario dinámica y responsiva. Esta elección asegura que el diseño sea portable y accesible en diferentes plataformas y navegadores web.

En la capa de lógica de negocio, se implementa un *backend* utilizando Node.js, con Express, que gestiona las rutas y controla la lógica de procesos del sistema. Este backend se encarga de redirigir las solicitudes a los controladores adecuados, donde se ejecutan funciones específicas según la ruta invocada. La lógica de negocio se organiza de manera que se optimicen los procesos y se mantenga la claridad en la estructura del código.

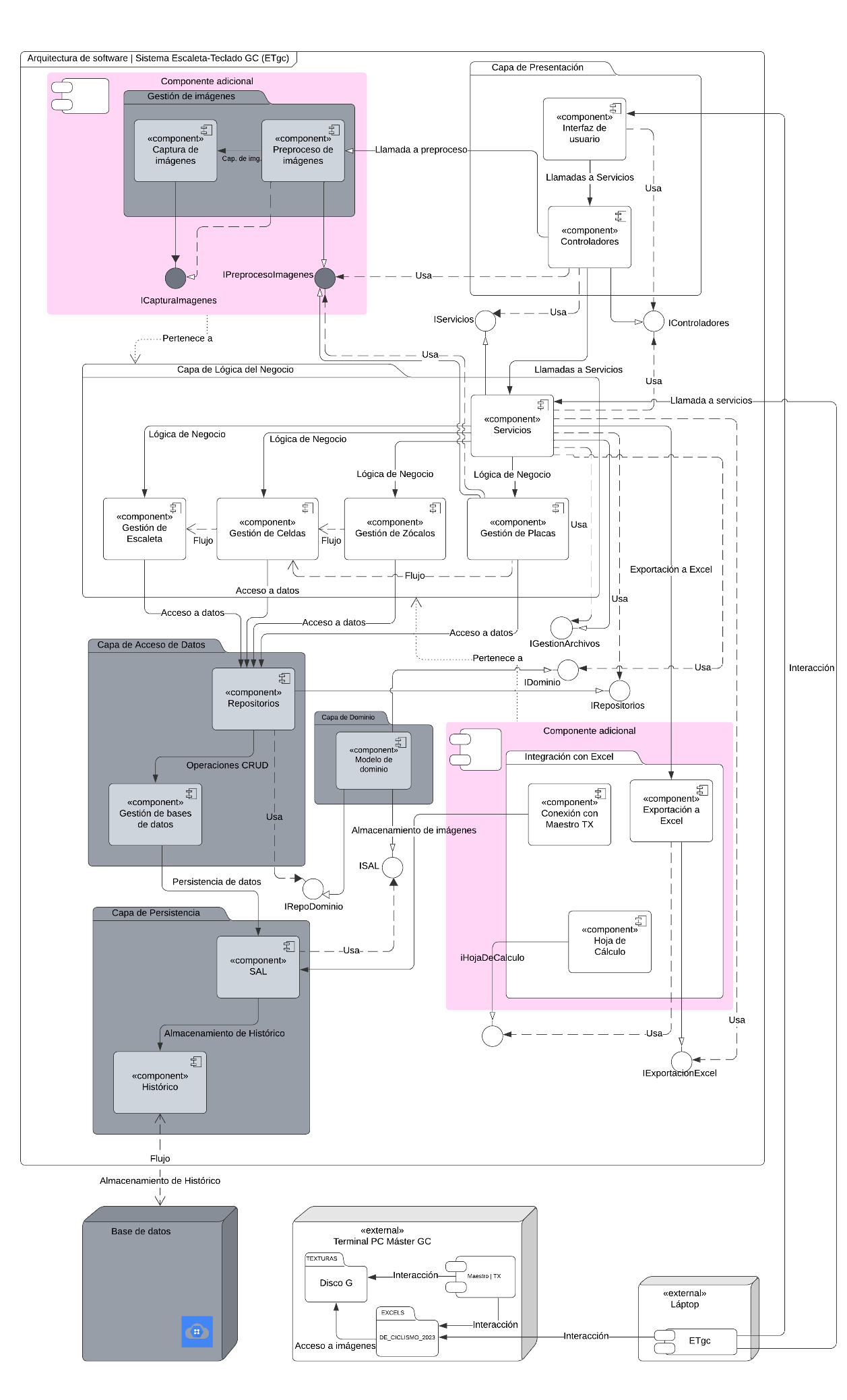
La capa de acceso a datos se basa en un enfoque de microservicios, permitiendo un acceso directo y eficiente a la base de datos. En esta capa, se implementa el patrón de diseño Singleton en las entidades, lo que mejora el rendimiento y la concurrencia en las peticiones a los servicios de la aplicación, priorizando así las solicitudes más críticas.

### Atributos de calidad

**Mantenibilidad:** Se busca disminuir la alta cohesión de los componentes para garantizar que todo el sistema sea más dinámico y adaptable a las necesidades de expansión. Se dedican los recursos necesarios a cada una de las capas, permitiendo un procesamiento paralelo de las responsabilidades, lo que facilita la gestión en caso de una posible expansión del sistema mediante el patrón multicapa.

**Modificabilidad:** Se aplica a través de la separación de tareas, lo que permite que la aplicación sea ligera y comprensible. Los cambios pueden realizarse en una parte de la aplicación sin afectar a los demás componentes, gracias a la estructura multicapa.

**Funcionalidad:** Se manifiesta en la estandarización de componentes desarrollados de forma separada por capas, lo que permite su integración con otros y fomenta la interoperabilidad del aplicativo. Esto asegura que el sistema pueda adaptarse a nuevas subcategorías y funcionalidades según sea necesario.



### Capa de Presentación

* Interfaz de Usuario: Interactúa con el usuario final y recibe entradas desde la Láptop (Terminal de Máster de Asistente de Producción).
* Controladores: Procesan las solicitudes del usuario y llaman a los servicios necesarios en la capa de Lógica de Negocio.

### Capa de Lógica de Negocio

* Servicios: Ejecutan la lógica de negocio y gestionan las operaciones relacionadas con la escaleta, celdas, zócalos y placas. También interactúan con la \*\*PC Externa\* para recibir solicitudes y enviar respuestas.
* Gestión de Escaleta, Gestión de Celdas y Zócalos, Gestión de Placas: Componentes específicos que manejan diferentes aspectos de la escaleta.

### Capa de Acceso a Datos

* Repositorios: Proveen métodos para acceder y manipular los datos almacenados en la base de datos.
* Gestión de Base de Datos: Maneja las operaciones CRUD en la base de datos.

### Capa de Dominio

* Modelo de dominio: Representa las entidades y conceptos del negocio; encapsulan el comportamiento y asegura que las opresiones se realicen de forma adecuada.
* **UI --> Servicios : Interacción**:
* Esta interacción representa cómo la **Interfaz de Usuario** (UI) se comunica con los **Servicios**. La UI envía solicitudes a los servicios para realizar acciones, como crear, leer, actualizar o eliminar datos. Es la entrada del sistema.
* **Servicios --> Modelo de Dominio : Lógica de negocio**:
* Aquí, los **Servicios** invocan métodos en los **Modelos de Dominio** para aplicar la lógica de negocio. Por ejemplo, un servicio podría llamar a un método en el modelo de dominio para validar un pedido antes de guardarlo. Esta interacción es crucial porque asegura que las reglas del negocio se apliquen correctamente.
* **Modelo de Dominio --> Repositorios : Llamada a datos**:
* Esta interacción indica que los **Modelos de Dominio** pueden solicitar datos a los **Repositorios**. Los repositorios son responsables de la persistencia y recuperación de datos. Los modelos de dominio no acceden directamente a la SAL, sino que utilizan los repositorios para obtener la información necesaria.
* **Repositorios --> SAL : Acceso a datos**:
* Los **Repositorios** interactúan con la **Capa de Persistencia** (SAL) para realizar operaciones de acceso a datos.
* **SAL --> DomainModels : Acceso a documento**:
* Esta interacción representa cómo la **Capa de Persistencia** devuelve datos a los **Modelos de Dominio**. Después de realizar una consulta, la SAL proporciona los datos necesarios para que los modelos de dominio puedan ser actualizados o utilizados en la lógica de negocio.

### Capa de Persistencia

* Base de Datos: Almacena de manera persistente los datos del sistema.
* Histórico: Almacena los datos históricos para su consulta y recuperación.

### Integración con Excel

* Exportación a Excel: Permite exportar los datos de la escaleta a un archivo Excel.
* Conexión con Maestro TX: Facilita la conexión con el sistema Maestro TX para la transmisión de datos.
* Hoja de Cálculo: Representa la hoja de cálculo de Excel donde se exportan los datos.

### Gestión de Imágenes

* Preproceso de Imágenes: Realiza el preprocesamiento de las imágenes antes de su captura.
* Captura de Imágenes: Captura las imágenes y las prepara para su almacenamiento.

Nodos Externos

* Láptop: Representa la terminal del asistente de producción, que interactúa con la Interfaz de Usuario y los Servicios.
* Terminal PC Máster GC: Incluye la terminal principal y la carpeta de imágenes.

# 5. Vista de despliegue

## 5.1. Diagrama de despliegue

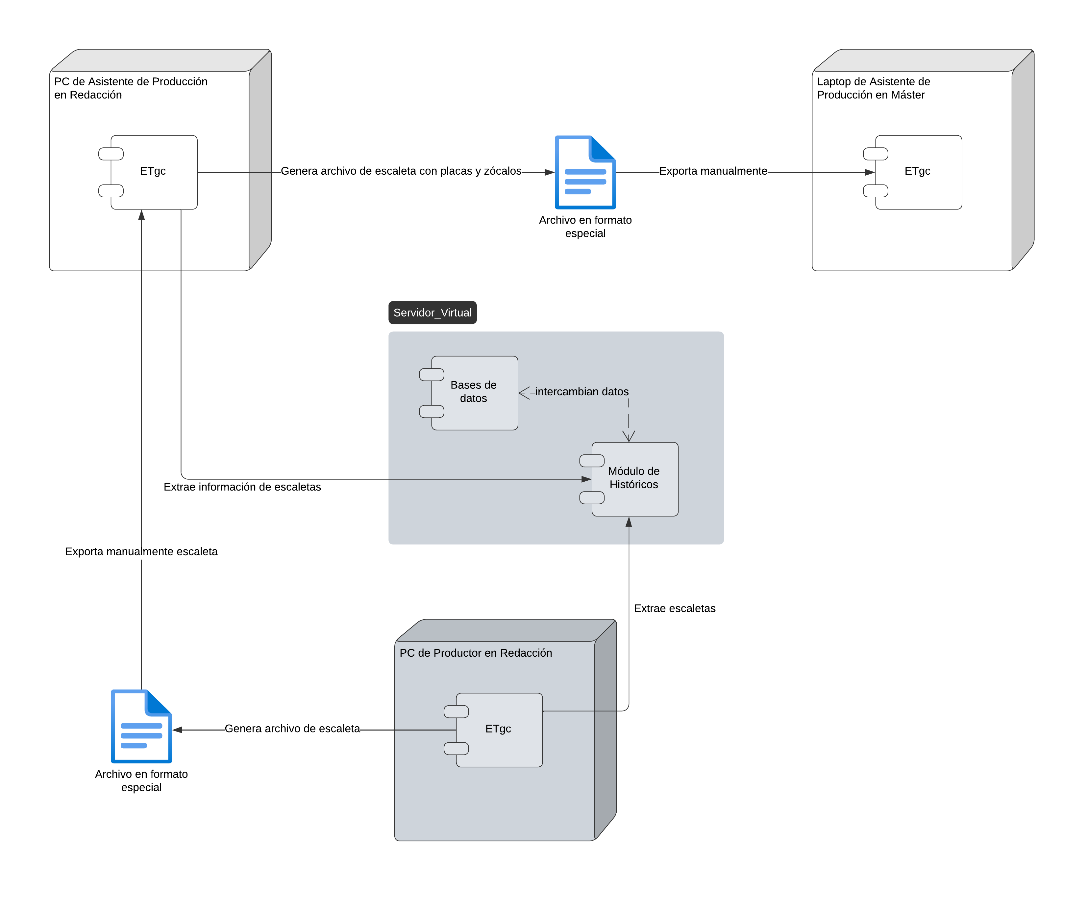
Este diagrama describe las configuraciones de redes físicas sobre las cuales se desarrollará el *software*. Se indican los nodos físicos que interactúan con la aplicación web, incluyendo un nodo para el dispositivo desde el cual se accede a la aplicación en un PC con acceso a Internet y un navegador. Este dispositivo envía las peticiones a través de HTTP, garantizando una mayor seguridad en la transmisión de información, y se conecta al nodo del servidor web que aloja el entorno de Node.js con Express (esta funcionalidad no se estipula en el prototipo que se entrega como resultado; se espera aprobación de despliegue en servidor virtual).

En este entorno, se gestiona la lógica de negocio, que se organiza en microservicios, permitiendo una arquitectura flexible y escalable. La base de datos se gestiona mediante APEX Oracle, lo que proporciona un acceso eficiente y robusto a los datos. La aplicación está estructurada en capas, incluyendo la capa de presentación, que utiliza React.js para ofrecer una interfaz de usuario dinámica y responsiva, la lógica de negocio, y la capa de acceso a datos, que facilita la interacción con la base de datos.

La gestión de archivos se realiza de forma manual entre terminales, y los archivos solo pueden ser ejecutados en el aplicativo ETgc de cada una de las terminales, lo que asegura un control adecuado sobre el acceso y la ejecución de estos.

### Atributos de calidad

**Modificabilidad:** Se aplica a través de la separación de tareas, lo que permite que la aplicación sea ligera y comprensible. Los cambios pueden realizarse en una parte de la aplicación sin afectar a los demás componentes, gracias a la estructura multicapa que se ha implementado. Esto facilita la adaptación y evolución del sistema según las necesidades del negocio.



# 6. Vista de procesos

## 6.1. Diagrama de procesos

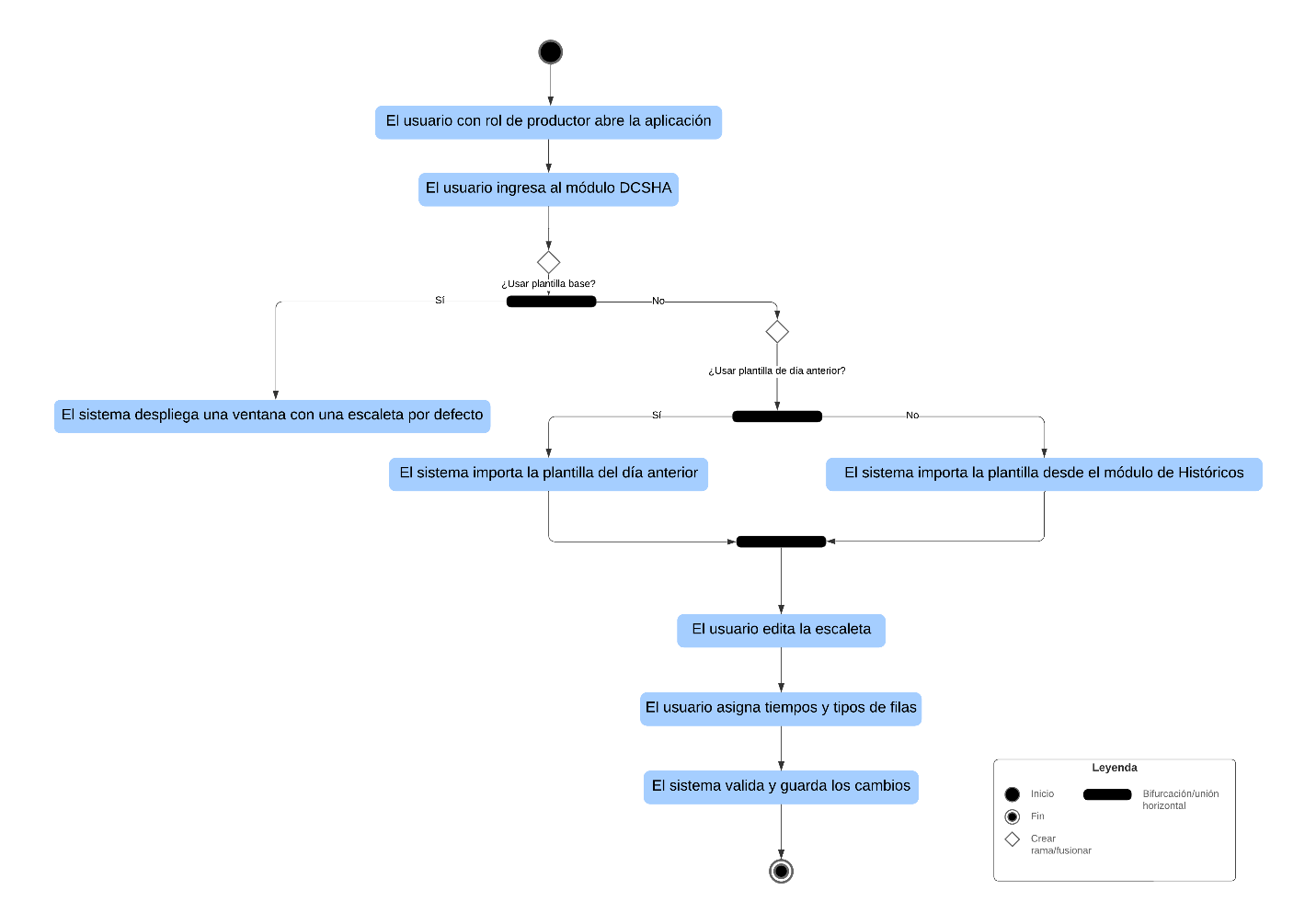
En esta sección se presenta el Diagrama de Procesos, que ilustra cómo los diferentes componentes del sistema interactúan y se comunican entre sí. Este diagrama es esencial para entender el flujo de trabajo y los procesos operacionales que se llevan a cabo en el sistema.

### Atributos de calidad

**Confidencialidad:** Este atributo se aplica en el manejo de los ejecutables entre las terminales utilizando el aplicativo ETGC. Los archivos generados entre las terminales son accesibles únicamente a través del aplicativo, lo que garantiza que solo los usuarios autorizados puedan interactuar con ellos. A través del patrón de Mount Points, se controla el acceso a estos ejecutables, asegurando que la información sensible esté protegida y que cada usuario solo pueda acceder a los datos necesarios para su función. Este enfoque se complementa con la autenticación JWT, que, valida los privilegios de cada usuario, reforzando así la seguridad del sistema.

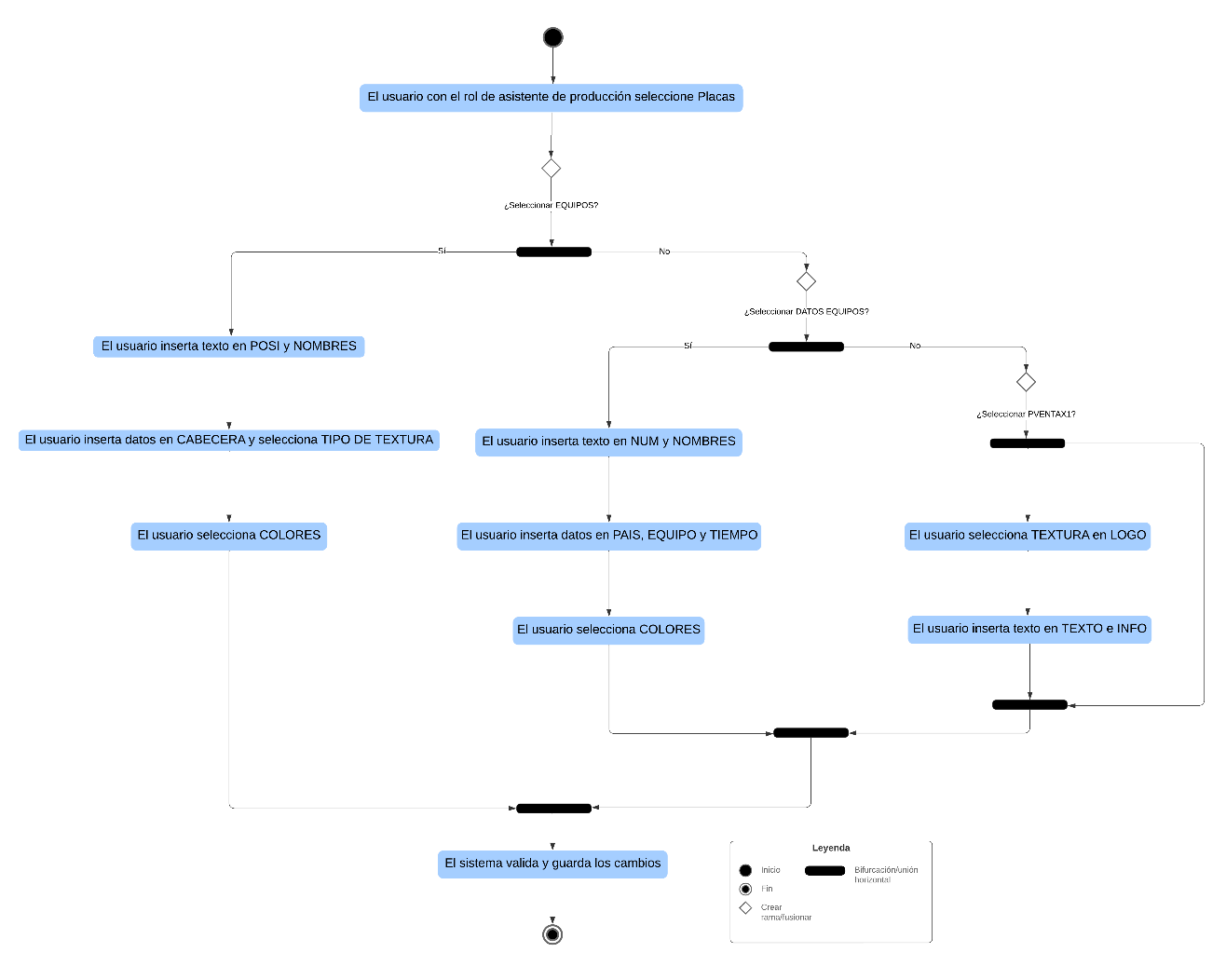
El diagrama de actividades (Proceso de selección de plantilla) describe cómo el usuario ingresa al módulo de selección de plantilla, crea los campos de filas y los tiempos:

### Diagrama de actividades: Proceso de selección de plantilla



El diagrama de actividades (Proceso de gestión de zócalos) describe cómo el usuario con el rol de asistente de producción a insertar contenido de texto de cada uno de los tipos de zócalos disponibles para la producción DCSHA.

### Diagrama de actividades: Proceso de gestión de zócalos



El diagrama de actividades (Proceso de numeración de ID y generación de índice) describe cómo el sistema realiza el proceso de numeración e índice que se impronta en el documento de Excel.

### Diagrama de actividades: Proceso de gestión de zócalos

