**Especificación de Requisitos Funcionales:**

1. **Comprar Entrada:** Permite al usuario comprar una entrada para una o varias butacas, reservándola cos un usuario.
   * 1. El usuario selecciona una butaca disponible.
     2. El usuario selecciona los complementos deseados.
     3. Se genera la entrada (un archivo HTML) que contiene los detalles (butacas, socio, complementos y fecha de compra).
2. **Inicio de Sesión:** permite al usuario usar una cuenta propia con sus datos
   * 1. El usuario puede registrarse
     2. El usuario puede iniciar sesión con una cuenta previamente registrada
     3. El usuario puede ser de tipo admin y de tipo cliente
3. **Devolver Entrada:** Permite al usuario devolver una entrada previamente comprada, liberando los recursos asociados.
   * 1. El usuario ingresa el número de la entrada a devolver.
     2. Se libera la butaca y los complementos asociados.
4. **Estado del Cine:** Muestra el estado actual del cine, incluyendo la disponibilidad de las butacas y su estado (libre, reservada u ocupada).
   * 1. Se muestra una representación visual del cine con el estado de cada butaca.
5. **Obtener Recaudación:** Calcula la recaudación total del cine en una fecha específica.
   * 1. El usuario ingresa una fecha válida en formato AAAA/MM/DD.

Se calcula la suma total de las ventas realizadas en esa fecha.

1. **Importar Complementos:** Importa complementos desde un archivo CSV dado y los agrega a la base de datos.
   * 1. Se lee el archivo CSV.
     2. Se importan los complementos a la base de datos.
2. **Exportar Estado del Cine:** Exporta el estado actual del cine en un archivo JSON, para una fecha específica.
   * 1. El usuario ingresa una fecha válida en formato AAAA/MM/DD.
     2. Se genera un archivo JSON con el estado del cine en esa fecha.
3. **Configurar Butacas:** Configura las butacas del cine utilizando un archivo CSV dado.
   * 1. Se lee el archivo CSV.
     2. Se configuran las butacas según la información proporcionada.
4. **Actualizar Butaca:** Permite cambiar la información de una butaca dada su identificador.
   * 1. El usuario ingresa el identificador de la butaca a actualizar.
     2. Se actualiza la información de la butaca según los datos proporcionados.
5. **Gestión de Usuarios:** se pueden añadir borrar y editar usuarios
   * 1. un administrador debe poder:
        1. leer usuario
        2. borrar usuario
        3. crear usuario
        4. editar/actualizar usuario
     2. un usuario debe poder ver su cuenta y editar sus datos
6. **Gestión de Productos:** se pueden añadir borrar y editar productos
   * 1. un administrador debe poder:
        1. leer Producto
        2. borrar Producto
        3. crear Producto
        4. editar/actualizar Producto

**Especificación de Requisitos No Funcionales:**

1. **Base de Datos:**
   * La aplicación usa SQLite para gestionar los datos.
   * Uso de un manejador de base de datos y mapeo manual
2. **Uso de JavaFx e Interfaces**
   * Cada vista o pantalla consta de un archivo .fxml
   * En caso de excepción o error sale una ventana de aviso.
3. **Importación y exportación de datos CSV**
   * Al inicio se importar a un estado por defecto del cine, con el estado de las butacas, así como los productos y usuarios activos
4. **Git y GitHub:**
   * Para este proyecto colaborativo usamos un repositorio en el que vamos subiendo en distintas ramas la implementación de la app.
5. **Mecanismos de Inyección de Dependencias:**
   * Usamos mecanismos automatizados de inyección de dependencias para mejorar la modularidad y la testabilidad del código.
6. **Manejo de Excepciones Adaptadas al Dominio:**
   * Implementación de excepciones adaptadas al dominio para manejar errores de manera eficiente y clara.

**Especificación de Requisitos de Información:**

1. **Tablas de la Base de Datos:**
   * **Butacas**: Para almacenar información sobre las butacas.
     + ID: FC (fila, columna)
     + Tipo: normal o VIP
     + Estado: Activa, Ocupada o en Mantenimiento
     + Ocupamiento: Libre u Ocupada
     + IsDeleted: Booleano
     + CreatedAt: fecha de creación
     + UpdatedAt: fecha de actualización
   * **Complementos**: Para almacenar información sobre los complementos/productos disponibles.
     + ID: Identificador único del complemento
     + Nombre: Nombre del complemento o producto
     + Precio: Precio del complemento
     + Stock: Cantidad disponible en stock
     + Tipo: Tipo de complemento
     + IsDeleted: Booleano
     + CreatedAt: fecha de creación
     + UpdatedAt: fecha de actualización
   * **Ventas**: Para almacenar información sobre las entradas vendidas, incluyendo los detalles de la butaca, el socio y la fecha de compra.
     + ID: Identificador único de la venta
     + ID\_Butaca: ID de la butaca asociada a la venta
     + ID\_Socio: ID del socio asociado a la venta
     + IsDeleted: Booleano
     + CreatedAt: fecha de creación
     + UpdatedAt: fecha de actualización
   * **Cuentas:** Con la información de usuarios que entran para poder comprar la entrada
     + **Nombre:** String
     + **Correo:** String
     + **Contraseña:** String
     + **Tipo:** (administrador, cliente)
     + **IsDeleted**: Booleano
     + **CreatedAt**: fecha de creación
     + **UpdatedAt**: fecha de actualización
   * **LineaVenta**: para relacionar el Usuario con su venta tanto de butaca como de complementos
     + ID: Identificador único de la línea de venta
     + ID\_Venta: ID de la venta asociada a la línea
     + ID\_Complemento: ID del complemento asociado a la línea
     + Cantidad: Cantidad del complemento vendido
     + Precio: Precio unitario del complemento
     + IsDeleted: Booleano
     + CreatedAt: fecha de creación
     + UpdatedAt: fecha de actualización
2. **Integridad de Claves Primarias y Referenciales:**
   * Se debe garantizar la integridad de las claves primarias y referencias entre las tablas para mantener la consistencia de los datos.
3. **Formato de Archivos Generados:**
   * Los archivos generados (HTML, JSON) deben seguir un formato específico que contenga la información relevante de manera clara y estructurada.