

Manual Técnico Proyecto Gestión de Eventos de Comedia

Miguel Alejandro Bejarano Perdomo

1. Presentación general

Este documento tiene como objetivo proporcionar una guía general para el uso del software de gestión de eventos y boletos desarrollado para el grupo de Toxicómicós, ofreciendo una perspectiva técnica que justifica las decisiones de diseño y resalta los detalles más relevantes del proceso de desarrollo.

El software se desarrolló siguiendo los criterios establecidos por el cliente, que requerían principalmente la capacidad de gestionar tres tipos de eventos, un sistema completo de gestión de boletos, un mecanismo para regular el acceso a los eventos, la generación de informes comerciales y la implementación de un panel de análisis de datos (Dashboard).

Se diseñó el programa con una interfaz gráfica para facilitar la interacción con el administrador del sistema. Para esto, se utilizó el framework Streamlit, un framework de código abierto que permite la implementación del front-end y simplifica el despliegue web, como se muestra en las siguientes ilustraciones (Figura 1, Figura 2 y Figura 3).

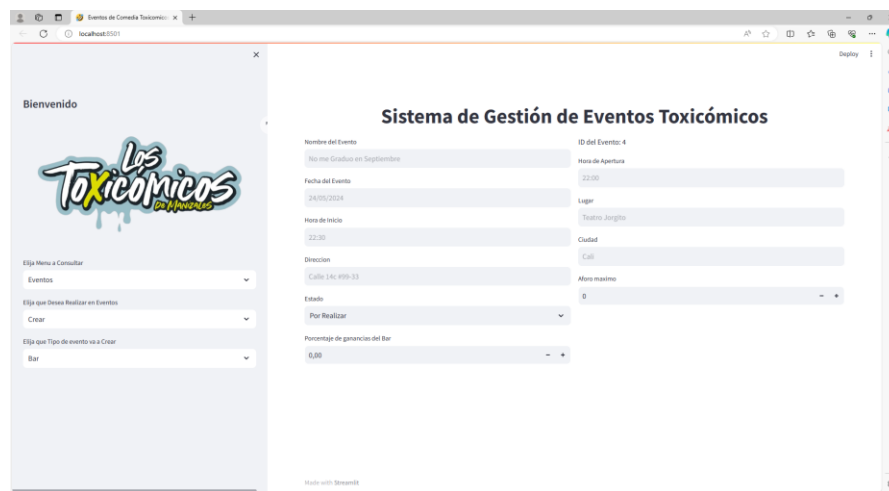


Figura 1. Menú de Creacion de Eventos



Figura 2. Menú de Creacion de Boletería

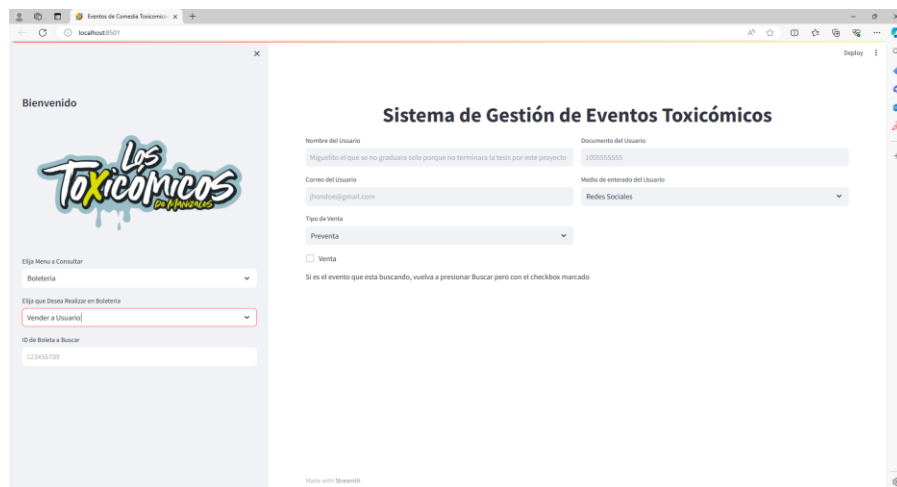


Figura 3. Menú de Venta a Usuario

El flujo de navegación en la página se realiza a través de los menús desplegables ubicados en la barra lateral, debajo del logo de Toxicómicos. En el primer menú desplegable, se presenta una selección de opciones para consultar, que incluyen Eventos, Boletería, Reportes/Dashboard, Artistas, Patrocinadores y, finalmente, Ingreso a Evento.

Dentro de cada uno de estos menús, se encuentran uno o varios submenús que ofrecen diversas opciones correspondientes a cada parte del programa.

a. Creación de Eventos

Para crear eventos, se accede al menú de Eventos y se elige la opción de crear en el segundo menú desplegable. Luego, se selecciona el tipo de evento entre las opciones disponibles: Bar, Teatro o Filantrópico. Una vez seleccionado, se completan los campos necesarios. Una vez todos los campos están llenos, aparece un botón de "crear". Si la creación se realiza con éxito, se muestra una animación indicando el proceso exitoso.

b. Mostrar Eventos

Dentro del menú de Eventos, en la función de mostrar eventos por ID, se presenta un campo para ingresar la ID del evento que se desea buscar. Una vez completado este campo, se activa un botón para realizar la búsqueda. En caso de que el evento sea encontrado, se muestra su información general.

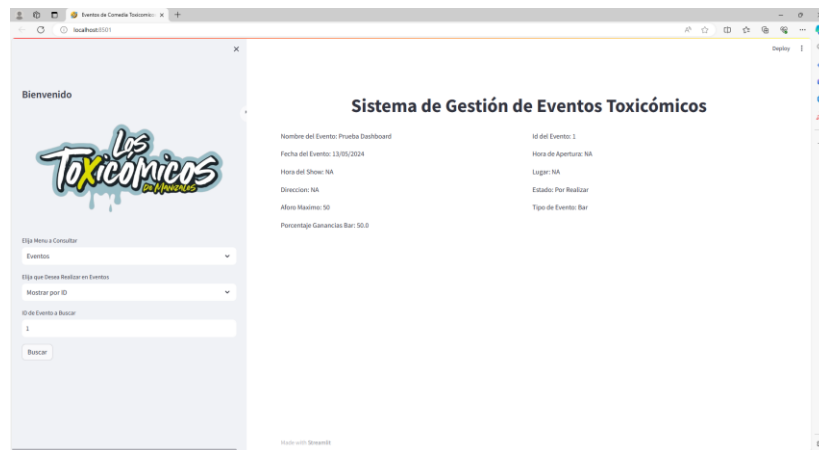


Figura 4. Mostrar Evento por ID

La parte más fundamental del programa reside en el método de almacenamiento de datos. En la versión anterior del software, desarrollada en C++, se encontraba un problema recurrente: los datos se borraban tras cada ejecución del programa. Para evitar esta situación, se optó por utilizar archivos JSON para guardar las entradas de datos. En lo que respecta a eventos, boletería, artistas y patrocinadores, esta información se registra en varios archivos de notación de objeto de JavaScript dentro del directorio denominado "data".

c. Creacion de Boletas

Dentro del menú de Boletería, en el submenú de crear, se presenta un tercer menú desplegable que determina el tipo de boleta que se va a crear. Este menú desplegable ofrece opciones específicas según el tipo de evento: Bar, Filantrópico o Teatro. Una vez seleccionado el tipo de boleta, se muestran los campos correspondientes a dicha

opción. Una vez completados todos los campos requeridos, aparece el botón de crear para proceder con la creación de la boleta.

d. Agregar Boleta a Evento

Una vez creada la boleta, es necesario adjuntarla a un evento existente. Para esto, en el segundo menú de Boletería, se selecciona la opción de "Agregar a Evento". Aquí, se proporciona la ID del evento y la ID de la boleta en los campos correspondientes. Una vez completados estos campos, al presionar el botón "Buscar", la boleta se añade al evento especificado.

e. Mostrar por ID

El funcionamiento de esta función es similar al mostrado para eventos, pero en este caso se utiliza la ID de la boleta en lugar de la del evento. Se proporciona la ID de la boleta en el campo correspondiente y al presionar el botón "Buscar", se mostrará la información detallada de la boleta correspondiente a esa ID.

f. PDF por ID

Con el objetivo de optimizar los procesos de ingreso a los eventos, se ha implementado un mecanismo para la generación de boletas en formato PDF. Para generar una boleta en PDF, se requiere únicamente el ID de la boleta y que está ya haya sido vendida a una persona. El PDF generado se almacenará en el directorio "boletas_generadas".

g. Vender a Usuario

Una vez creada la boleta, se puede proceder con la venta a un usuario. Para ello, se ingresa la ID de la boleta a vender. Una vez que este campo queda registrado, aparece el botón de búsqueda. Al presionarlo, se muestra la información de la boleta que se va a vender. Si la información es correcta, se completan los campos relativos al usuario y se marca la casilla "Venta". Luego, se presiona nuevamente el botón de búsqueda y la boleta será vendida.

h. Ver Dashboard

Para visualizar el Dashboard, dentro del menú de Reportes se selecciona el submenú "Ver Dashboard". Se solicita al usuario que seleccione una fecha de inicio y una fecha de finalización. Una vez definidas estas fechas, se habilita el botón de "Generar Dashboard". Al presionar este botón, se muestran en pantalla los gráficos que representan los tipos de eventos vs su frecuencia en el período de tiempo establecido, así como los ingresos por evento único.

i. Crear Artistas

Dentro del menú Artistas, en el submenú de crear, se presentan los campos necesarios para ingresar la información del artista. Una vez que todos los campos están completos, se activa el botón de crear. Al presionarlo, el artista queda creado en el sistema.

j. Agregar Artista a Evento

Este mecanismo funciona de manera similar a agregar una boleta a un evento, pero en lugar de utilizar boletas, se emplean artistas. Se proporciona la ID del artista y la ID del evento al que se desea agregar al artista. Una vez completados estos campos, al presionar el botón "Buscar", el artista se añade al evento especificado.

k. Patrocinadores

En cuanto a los patrocinadores, su funcionamiento es exactamente igual a todo lo relacionado con el menú de artistas. El proceso de creación y agregación de patrocinadores sigue el mismo flujo que se describió para los artistas.

l. Ingreso a Evento

Como se mencionó anteriormente en relación a la generación de PDFs para la boletería, la idea es que el código QR redirija al usuario a la sección correspondiente del sistema de ingreso al evento. En esta sección, el usuario ingresa el ID del evento y su propia ID. El sistema se encarga de buscar entre las boletas vendidas si existe alguna asociada a la identidad del usuario proporcionada. Si se encuentra una boleta válida, se permite el acceso al evento. En caso contrario, se indica al personal encargado de regular la entrada que tome las medidas adecuadas contra el fraude.

2. Diagrama de Clases

El código mermaid con el diagrama de clases es el siguiente:

classDiagram

```
GuiController -- SistemaGestionEventos: utiliza
GuiController -- mainView: utiliza
GuiController : +__init__()
GuiController : -sistema_gestion
GuiController : -run_page
GuiController : +main()
```

```
mainView : +dibujar_pagina_inicial(gui_controller)
mainView : +dibujar_menu_eventos(gui_controller)
mainView : +dibujar_menu_boleteria(gui_controller)
mainView : +dibujar_menu_artistas(gui_controller)
mainView : +dibujar_menu_patrocinadores(gui_controller)
mainView : +dibujar_menu_reportes(gui_controller)
mainView : +dibujar_menu_ingreso_evento(gui_controller)
mainView : +mostrar_informacion_evento(objeto_evento)
mainView : +mostrar_informacion_boleta(objeto_boleta)
mainView : +mostrar_informacion_artista(objeto_artista)
mainView : +mostrar_informacion_patrocinador(objeto_patrociador)
```

```
SistemaGestionEventos o-- Evento: utiliza
SistemaGestionEventos o-- Artista: utiliza
SistemaGestionEventos o-- Patrocinador: utiliza
SistemaGestionEventos o-- Boleta: utiliza
SistemaGestionEventos : +__init__()
SistemaGestionEventos : +guardar_contadores()
SistemaGestionEventos : +crear_evento_bar()
SistemaGestionEventos : +crear_evento_teatro()
SistemaGestionEventos : +crear_evento_filantropico()
SistemaGestionEventos : +agregar_evento_objeto()
SistemaGestionEventos : +buscar_evento_id()
SistemaGestionEventos : +crear_boleta_bar()
SistemaGestionEventos : +crear_boleta_filantropico()
SistemaGestionEventos : +crear_boleta_teatro()
SistemaGestionEventos : +agregar_boleta_a_evento()
SistemaGestionEventos : +agregar_boleta_objeto()
SistemaGestionEventos : +buscar_boleta_id()
SistemaGestionEventos : +agregar_usuario_a_boleta()
SistemaGestionEventos : +generar_boleta_pdf()
SistemaGestionEventos : +crear_artista_objeto()
SistemaGestionEventos : +agregar_artista_objeto()
SistemaGestionEventos : +buscar_artista_id()
SistemaGestionEventos : +agregar_artista_a_evento()
SistemaGestionEventos : +crear_patrocinador_objeto()
SistemaGestionEventos : +agregar_patrocinador_objeto()
SistemaGestionEventos : +buscar_patrocinador_id()
SistemaGestionEventos : +agregar_patrocinador_a_evento()
SistemaGestionEventos : +buscar_usuario_ingreso()
SistemaGestionEventos : +obtener_datos_dashboard_frecuencia()
SistemaGestionEventos : +obtener_datos_dashboard_ingresos()
SistemaGestionEventos : +generar_dashboard()
```

```

SistemaGestionEventos : -eventos
SistemaGestionEventos : -boletas
SistemaGestionEventos : -artistas
SistemaGestionEventos : -patrocinadores
SistemaGestionEventos : -contador_id_eventos
SistemaGestionEventos : -contador_id_artistas
SistemaGestionEventos : -contador_id_boletas
SistemaGestionEventos : -contador_id_patrocinadores

Evento : <<abstract>>
Evento o-- Artista: utiliza
Evento o-- Boleta: utiliza
Evento : +__init__(id_evento, nombre, fecha, hora_apertura, hora_show, lugar,
direccion, ciudad, estado, aforo_maximo)
Evento : +aumentar_vendidos()
Evento : -id_evento
Evento : -nombre
Evento : -fecha
Evento : -hora_apertura
Evento : -hora_show
Evento : -lugar
Evento : -direccion
Evento : -ciudad
Evento : -estado
Evento : -artistas
Evento : -aforo_max
Evento : -aforo_vendido
Evento : -boletas

Bar --|> Evento: hereda de
Bar : +__init__(id_evento, nombre, fecha, hora_apertura, hora_show, lugar,
direccion, estado, ciudad, aforo_maximo, porcentaje_ganancias)
Bar : +aumentar_vendidos()
Bar : -tipo
Bar : -porcentaje_ganancias_bar
Bar : -porcentaje_ganancias_boleteria

Filantropico --|> Evento: hereda de
Filantropico o-- Patrocinador: utiliza
Filantropico : +__init__(id_evento, nombre, fecha, hora_apertura, hora_show,
lugar, direccion, ciudad, estado, aforo_maximo)
Filantropico : +aumentar_vendidos()
Filantropico : -tipo

```

```
Filantropico : -patrocinadores
```

```
Teatro --|> Evento: hereda de
```

```
Teatro : +__init__(id_evento, nombre, fecha, hora_apertura, hora_show, lugar,  
direccion, ciudad, estado, aforo_maximo, alquiler_teatro)
```

```
Teatro : +aumentar_vendidos()
```

```
Teatro : -tipo
```

```
Teatro : -alquiler
```

```
Boleta : <<abstract>>
```

```
Boleta : +__init__(id_boleta, medio_pago, precio_estandar, precio_preventa,  
porcentaje_descuento, es_cortesia)
```

```
Boleta : -id_boleta
```

```
Boleta : -medio_pago
```

```
Boleta : -medio_enterado
```

```
Boleta : -precio_estandar
```

```
Boleta : -precio_preventa
```

```
Boleta : -porcentaje_descuento
```

```
Boleta : -es_cortesia
```

```
Boleta : -nombre_usuario
```

```
Boleta : -documento_usuario
```

```
Boleta : -correo_usuario
```

```
Boleta : -vendida
```

```
Boleta : -tipo_venta
```

```
BoletaBar --|> Boleta: hereda de
```

```
BoletaBar : +__init__(id_boleta, medio_pago, precio_estandar,  
precio_preventa, porcentaje_descuento, es_cortesia)
```

```
BoletaBar : -porcentaje_ganancias_boleteria
```

```
BoletaBar : -tipo_boleta
```

```
BoletaFilantropico --|> Boleta: hereda de
```

```
BoletaFilantropico : +__init__(id_boleta, medio_pago, precio_estandar,  
precio_preventa, porcentaje_descuento, es_cortesia)
```

```
BoletaFilantropico : -tipo_boleta
```

```
BoletaTeatro --|> Boleta: hereda de
```

```
BoletaTeatro : +__init__(id_boleta, medio_pago, precio_estandar,  
precio_preventa, porcentaje_descuento, es_cortesia)
```

```
BoletaTeatro : -porecentaje_retencion
```

```
BoletaTeatro : -tipo_boleta
```

```
Artista : +__init__(id_artista, nombre, cobro)
```


Artista : -id_artista

Artista : -nombre

Artista : -cobro

Patrocinador : +__init__(id_patrocinador, nombre, apoyo)

Patrocinador : -id_patrocinador

Patrocinador : -nombre

Patrocinador : -apoyo

De igual manera se recomienda ver el diagrama de clases en el directorio “docs” del proyecto

3. Pruebas de Funcionamiento

Dada la extensión del programa, se presentarán solo algunas funcionalidades en este manual. No obstante, es importante destacar que todo lo mencionado en la primera sección del manual está plenamente operativo y funcional.

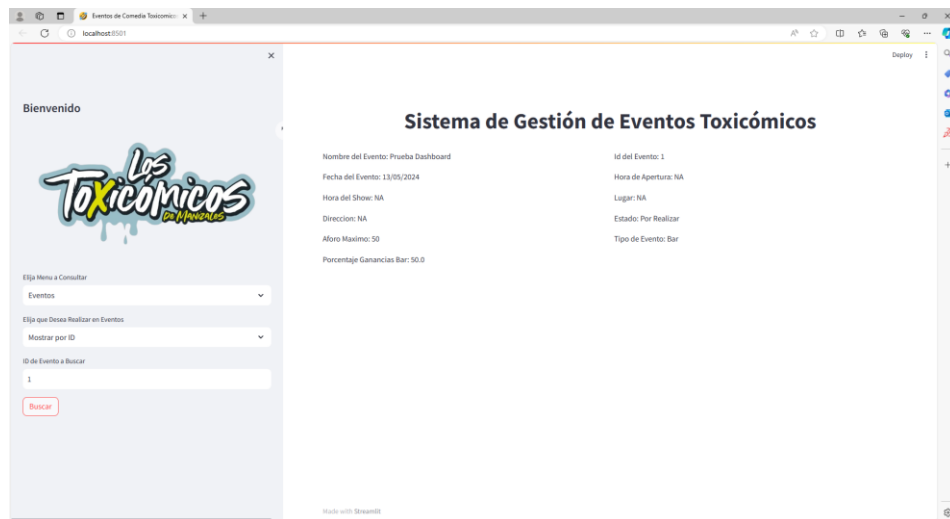


Figura 5. Muestra de Funcionamiento Búsqueda Eventos

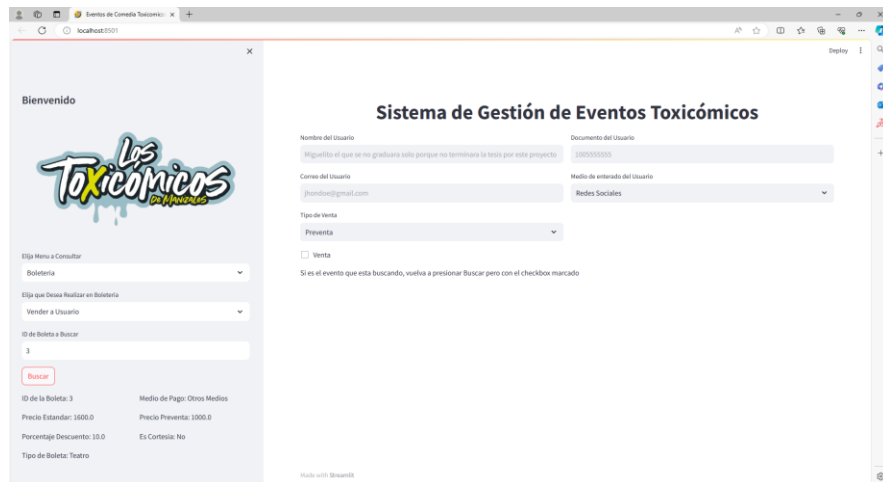


Figura 6. Muestra de Funcionamiento Venta a Usuarios

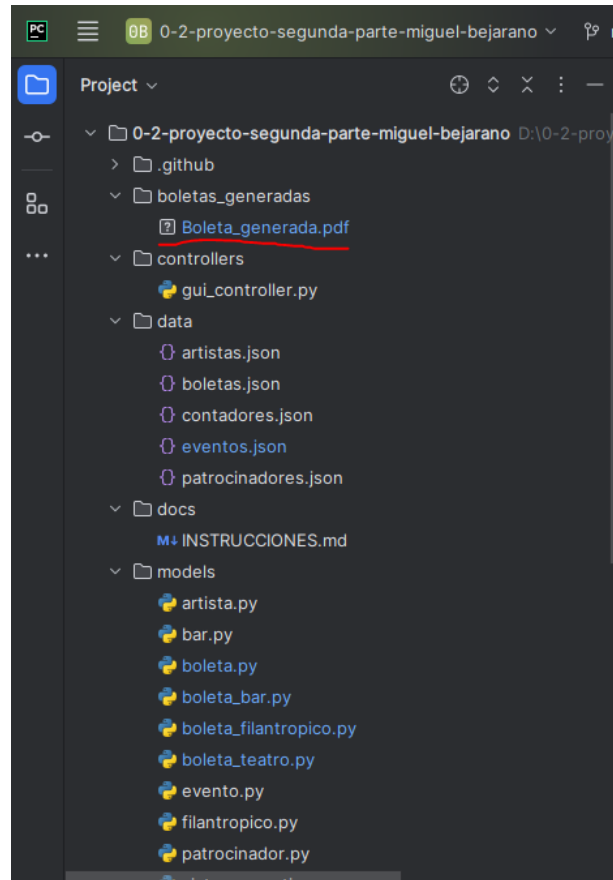
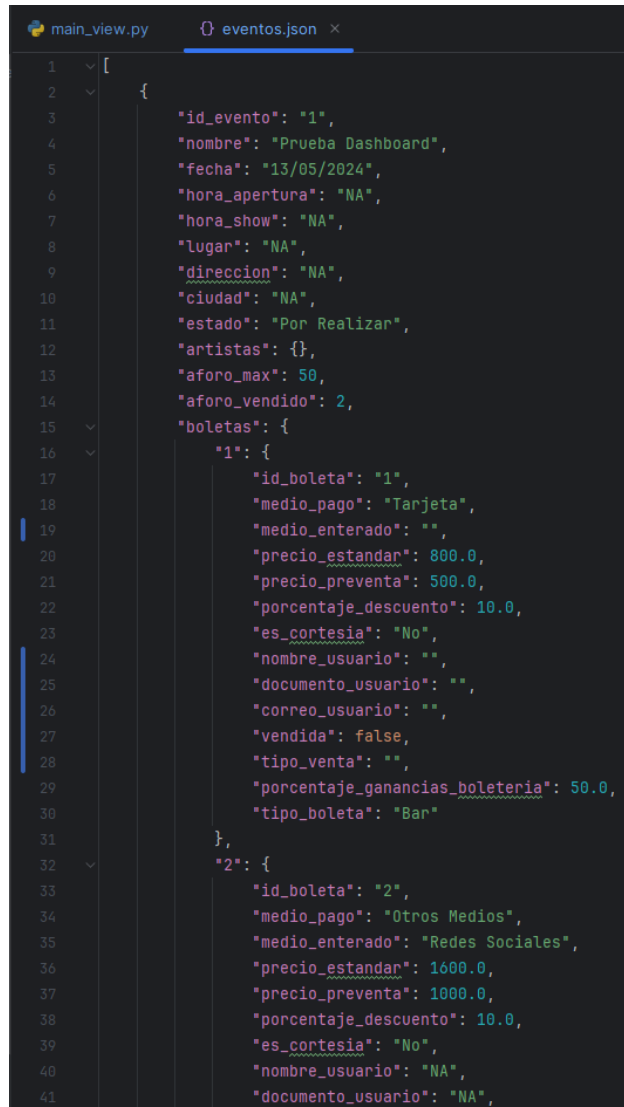


Figura 7. Muestra de Funcionamiento Generación de Boletas



```
main_view.py  eventos.json x
1  [
2    {
3      "id_evento": "1",
4      "nombre": "Prueba Dashboard",
5      "fecha": "13/05/2024",
6      "hora_apertura": "NA",
7      "hora_show": "NA",
8      "lugar": "NA",
9      "direccion": "NA",
10     "ciudad": "NA",
11     "estado": "Por Realizar",
12     "artistas": {},
13     "aforo_max": 50,
14     "aforo_vendido": 2,
15     "boletas": {
16       "1": {
17         "id_boleta": "1",
18         "medio_pago": "Tarjeta",
19         "medio_enterado": "",
20         "precio_estandar": 800.0,
21         "precio_preventa": 500.0,
22         "porcentaje_descuento": 10.0,
23         "es_cortesia": "No",
24         "nombre_usuario": "",
25         "documento_usuario": "",
26         "correo_usuario": "",
27         "vendida": false,
28         "tipo_venta": "",
29         "porcentaje_ganancias_boleteria": 50.0,
30         "tipo_boleta": "Bar"
31       },
32       "2": {
33         "id_boleta": "2",
34         "medio_pago": "Otros Medios",
35         "medio_enterado": "Redes Sociales",
36         "precio_estandar": 1600.0,
37         "precio_preventa": 1000.0,
38         "porcentaje_descuento": 10.0,
39         "es_cortesia": "No",
40         "nombre_usuario": "NA",
41         "documento_usuario": "NA",
```

Figura 8. Muestra de Funcionamiento de Guardado de Archivos



Figura 9. Muestra Funcionamiento Dashboard

Sistema de Gestión de Eventos Toxicómicos

EJECUTENLO!!
EJECUTENLO!!

Elige Menu a Consultar
Ingreso Evento

ID del Evento a Buscar
1

Documento del Usuario
1

Buscar

Figura 10. Muestra de Funcionamiento Ingreso a Evento (Se pone dos veces para mayor énfasis)

4. Pruebas Unitarias

Para garantizar la calidad del programa y mitigar posibles fallos, se han implementado pruebas unitarias para funciones clave. Dichas pruebas unitarias se encuentran en el directorio raíz del proyecto, específicamente en un subdirectorio llamado "test". Dentro de este directorio, se han creado dos subdirectorios adicionales: uno para las pruebas de las clases del directorio "controller" y otro para las del directorio "models".

Las pruebas unitarias se han realizado para los métodos de los siguientes archivos:

- Gui_controller.py
- artista.py
- bar.py
- boleta_bar.py
- boleta_filantropico.py
- boleta_teatro.py
- filantropico.py
- patrocinador.py
- sistema_gestion.py
- teatro.py

Para facilitar la revisión de estas pruebas unitarias, se ha incluido un archivo llamado "run_tests" dentro de la carpeta "test", el cual permite ejecutar todas las pruebas unitarias simultáneamente.

5. Estándar PEP8

Para promover la estandarización en todos los archivos del programa, se ha seguido el estándar de estilo PEP8, lo que facilita el mantenimiento del código. Además de esto, se ha adoptado el paradigma orientado a objetos para abordar el problema, aplicando arquitectura de software para estructurar las carpetas del programa.

Adicionalmente, todos los archivos están debidamente comentados, y se ha seguido el estándar snake_case para nombrar todas las variables y métodos declarados.

6. Despliegue WEB

Para el despliegue WEB se usaron las bondades de Streamlit para poder utilizar el programa como una especie de "IaaS" sin la parte de cobrar.

El enlace resulta ser: