🚀 Flujo de Arranque Detallado - Aplicación de Imputaciones

📋 Resumen Ejecutivo

El arranque sigue un patrón de **inicialización progresiva** con splash screen que informa al usuario del progreso, seguido de una **verificación de sesión** que determina si mostrar login o cargar directamente la interfaz de usuario.

🔄 Secuencia de Ejecución Completa

1. INICIO DEL PROGRAMA (`main.py`)

```python
def main():
    root = tk.Tk()
    root.withdraw()  # ⬅️ Oculta ventana principal
    splash = SplashScreen(root)  # ⬅️ Muestra splash
    app = ImputacionesApp(root, splash)  # ⬅️ INICIA TODO
    root.deiconify()  # ⬅️ Muestra ventana principal
    root.mainloop()  # ⬅️ Control al usuario
```

---

2. CONFIGURACIÓN INICIAL (`ImputacionesApp.__init__()`)

2.1 Preparación del Entorno

```python
# Variables de estado iniciales
self._running = False
self._paused = False
self.elapsed_time = 0
self.register_dic = None
self.empresas_dic = {}
self.empresas_suasor = {}

# Configuración del sistema
verificar_o_crear_carpeta_archivos()  # Crea ~/.imputaciones/
self.configure_root()  # Ventana principal
configure_styles()  # Estilos ttk
```

---

3. INICIALIZACIÓN DE COMPONENTES CORE (Con splash progress)

3.1 SQL Server Manager (0% → 25%)

python

```python
splash.update_message("Iniciando Conexión...")
self.sql_server_manager = SQLServerManager(self.root)
```

Dentro de `SQLServerManager.__init__()`:

python

```python
self.connection = None
self.cursor = None
self.servidor_activo = None
self.conectado = self.comprobar_conexion()  # 🔙 CONEXIÓN CRÍTICA
```

Proceso de conexión (`comprobar_conexion()`):

1. 🔍 Intenta conectar a `srv-suasor:1433`
2. 🔄 Si falla → fallback a `192.168.55.7:1433`
3. ✅ Si conecta → establece `self.conectado = True`
4. ❌ Si falla todo → muestra error y `self.conectado = False`

3.2 Database Manager (25% → 50%)

python

```python
splash.update_message("Cargando datos desde SUASOR...")
self.db_manager = DatabaseManager(self, DB_PATH)
```

Dentro de `DatabaseManager.__init__()`:

python

```python
self.conexion = sqlite3.connect(db_path)  # SQLite local
self.cursor = self.conexion.cursor()
self._crear_tabla()  # Tabla 'registros'
```

3.3 Session Manager (50% → 75%) - SINCRONIZACIÓN CRÍTICA

python

```python
splash.update_message("Inicializando sesión...")
self.session = SessionManager(self)
```

Dentro de `SessionManager.__init__()` - SECUENCIA CRÍTICA:

3.3.1 Sincronización de Datos

python

```
# ⚠️ OPERACIONES DE RED - PUEDEN FALLAR
self.app.db_manager.sincronizar_empleados()
self.app.db_manager.sincronizar_empresas()
```

Proceso `sincronizar_empleados()`:

1. 🔗 Verifica conexión SQL Server
2. 📊 `sql_server_manager.obtener_empleados_dataframe()`
   - Query: `SELECT id,nombre,apellido_1,apellido_2,department_name,activo FROM DimEmpleados WHERE activo=1`
   - Filtra departamento != 'ADMINISTRACION'
3. 💾 Guarda en SQLite tabla `empleados` (REPLACE)

Proceso `sincronizar_empresas()`:

1. 🔗 Verifica conexión SQL Server
2. 📊 `sql_server_manager.obtener_empresas_dataframe()`
   - Query: `SELECT vat,name,origen,baja FROM DimClientes WHERE baja=0 AND vat IS NOT NULL`
3. 🔍 Aplica filtros: 'otros', 'suasor', 'lexon'
4. 💾 Guarda en SQLite tabla `empresas` (REPLACE)

3.3.2 Carga de Configuración

python

```
self.empleados_dict, self.empleados_df = self.app.db_manager.load_empleados_sqlite()
self.config = self.load_config()  # ~/.imputaciones/config.json
self.user_id = self.config["session"]["id"]
```

3.3.3 Verificación de Sesión

python

```python
# Busca usuario en empleados_df por ID
mask = self.empleados_df["id"] == self.user_id
if mask.any():
    fila = self.empleados_df.loc[mask, ["name","department_name"]].iloc[0]
    self.user = fila['name']
    self.department = fila["department_name"]
else:
    self.user = ""  # ⬅️ NO LOGUEADO
    self.department = None
```

3.4 Systray Manager (75% → 100%)

python

```python
splash.update_message("Terminando configuración...")
self.systray = SystrayManager(self, ICON, "SIN USUARIO")
```

Dentro de `SystrayManager.__init__()`:

python

```python
self.initialized = threading.Event()
systray_thread = threading.Thread(target=self.create_systray, daemon=True)
systray_thread.start()  # ⬅️ Hilo separado para systray
```

---

4. CREACIÓN DE INTERFAZ

python

```python
# Componentes de UI
self.create_user_section()  # Frame usuario + logout
self.search_frame = BusquedaFrame(...)  # Combobox empresas
self.create_toggle_section()  # Controles timer + campos
self.tasks_admin = TasksAdmin(self)  # Treeview registros

# Espera a que systray esté listo
self.systray.initialized.wait()
```

---

5. FINALIZACIÓN DE SPLASH

python

```python
splash.update_progress(100)
splash.update_message("¡Listo!")
time.sleep(0.5)
splash.destroy()
```

---

6. DECISIÓN DE ESTADO INICIAL - PUNTO CRÍTICO

python

```python
if self.session.logged_in:  # self.user != ""
    self.logged_in()  # ⬅️ USUARIO YA LOGUEADO
else:
    self.set_logout_state()  # ⬅️ MOSTRAR LOGIN
```

🎯 Estados Finales Posibles

ESCENARIO A: Usuario Ya Logueado (`self.logged_in()`)

python

```python
def logged_in(self):
    # 1. Cargar datos de empresas
    self.empresas_suasor, self.empresas_dic = self.session.return_empresas_combo_values()

    # 2. Cargar registros del usuario en treeview
    self.tasks_admin.cargar_datos_desde_sqlite()

    # 3. Subir registros pendientes a SQL Server
    self.sql_server_manager.subir_registros(self.db_manager, self.session.user)

    # 4. Configurar interfaz para usuario logueado
    self.set_login_state()
```

Resultado: ✅ **Aplicación lista para usar**

ESCENARIO B: Usuario No Logueado (`self.set_logout_state()`)

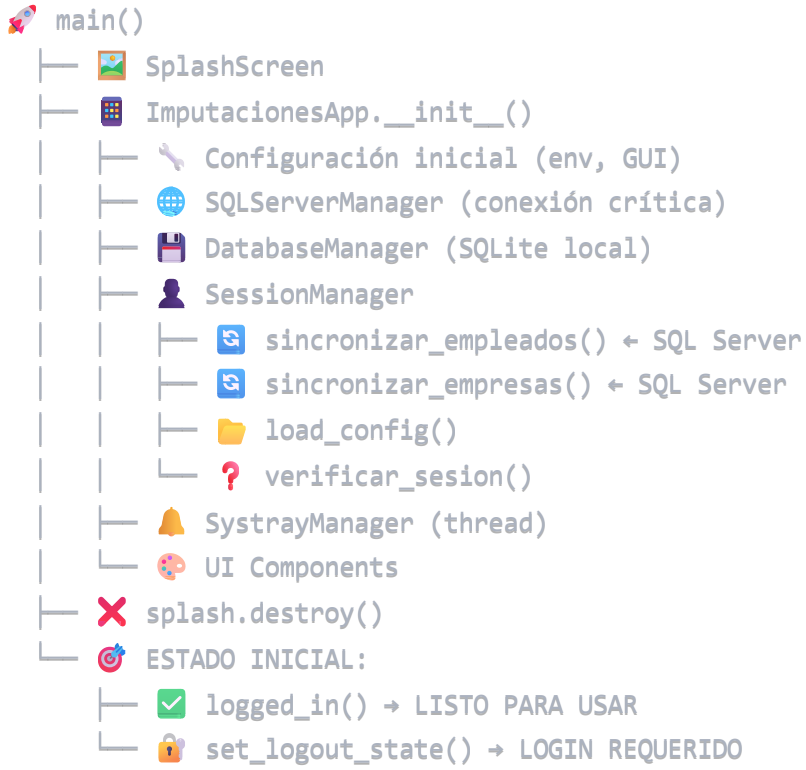
python

```python
def set_logout_state(self):
    # 1. Ocultar toda la interfaz funcional
    self.search_frame.pack_forget()
    self.toggle_frame.pack_forget()
    self.tasks_admin.pack_forget()

    # 2. Mostrar popup de login
    self.create_login_popup()  # 🔙 MODAL BLOCKING
```

Resultado: 🔐 **Esperando selección de usuario**

📊 Diagrama de Flujo Visual



⚠️ Puntos Críticos de Fallo

1. Conexión SQL Server

- **Fallo:** No hay red/servidor caído
- **Resultado:** `sincronizar_empleados/empresas` fallan
- **Impacto:** App funciona solo con datos locales cached

2. Sincronización de Datos

- **Fallo:** Tablas SQL Server cambió estructura

- **Resultado:** Error en `DataFrame.to_sql()`
- **Impacto:** Login no funciona (no hay empleados)

3. Configuración Corrupta

- **Fallo:** `config.json` corrupto
- **Resultado:** Se crea configuración por defecto
- **Impacto:** Usuario debe loguearse de nuevo

4. Systray

- **Fallo:** No se puede crear icono
- **Resultado:** App funciona sin systray
- **Impacto:** Funcionalidad reducida

🏁 Control al Usuario

La aplicación queda bajo control del usuario cuando:

1. ✅ **Splash desaparece**
2. ✅ **Ventana principal visible** (`root.deiconify()`)
3. ✅ **Mainloop iniciado** (`root.mainloop()`)
4. ✅ **Estado determinado:**
   - **Logueado:** Timer disponible, treeview cargado
   - **No logueado:** Popup login esperando selección

Tiempo total estimado: 2-5 segundos (depende de red SQL Server)