# 1. UEditorTabelaController.pas

## Importar novos services/DTO:

`System.SysUtils, System.Classes, Data.DB, Datasnap.DBClient, VCL.Dialogs;` (padrão Delphi)

`UTabelaDTO, UEditarTabelaService, UXMLService;` (existentes)

`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.CloseViewEditorTabela;`)

## Novos objetos/métodos no private do `TEditorTabelaController`:

`FOnSolicitarSalvar: TProc<TTabelaDTO>;` // Vai fazer: Disparar o salvamento da tabela via `FService`.
`FOnSolicitarFechar: TNotifyEvent;` // Vai fazer: Disparar o fechamento da view via `TShowViewService`.
`FService: TEditarTabelaService;` // Vai fazer: Realizar a lógica de edição/salvamento.
`FXMLService: TXMLService;` // Vai fazer: Apoiar o `FService` no salvamento/carregamento XML.
`procedure ManipuladorSolicitarSalvar(ATabela: TTabelaDTO);` // Vai fazer: Chama `FService.ExecutarSalvarTabela`.
`procedure ManipuladorSolicitarFechar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewEditorTabela;`.

## Novos objetos/métodos no public:

`property OnSolicitarSalvar: TProc<TTabelaDTO> read FOnSolicitarSalvar write FOnSolicitarSalvar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarFechar: TNotifyEvent read FOnSolicitarFechar write FOnSolicitarFechar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarSalvar(ATabela: TTabelaDTO);` // Vai fazer: Dispara o evento `FOnSolicitarSalvar`.
`procedure SolicitarFechar;` // Vai fazer: Dispara o evento `FOnSolicitarFechar`.

## Alterar existentes:

`constructor Create(AService: TEditarTabelaService; AXMLService: TXMLService);` // Vai fazer: Injeção de dependência e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller (os serviços injetados não são liberados aqui).

---

# 2. UEditorRelatorioController.pas

## Importar novos services/DTO:

`System.SysUtils, System.Classes;` (padrão Delphi)
`URelatorioDTO;` (existente)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.CloseViewEditorRelatorio;`)

## Novos objetos/métodos no private do `TEditorRelatorioController`:

`FOnSolicitarSalvar: TProc<TRelatorioDTO>;` // Vai fazer: Disparar o salvamento do relatório via serviço.
`FOnSolicitarFechar: TNotifyEvent;` // Vai fazer: Disparar o fechamento da view via `TShowViewService`.
`procedure ManipuladorSolicitarSalvar(ARelatorio: TRelatorioDTO);` // Vai fazer: Chama serviço de salvamento.
`procedure ManipuladorSolicitarFechar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewEditorRelatorio;`.

## Novos objetos/métodos no public:

`property OnSolicitarSalvar: TProc<TRelatorioDTO> read FOnSolicitarSalvar write FOnSolicitarSalvar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarFechar: TNotifyEvent read FOnSolicitarFechar write FOnSolicitarFechar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarSalvar(ARelatorio: TRelatorioDTO);` // Vai fazer: Dispara o evento `FOnSolicitarSalvar`.
`procedure SolicitarFechar;` // Vai fazer: Dispara o evento `FOnSolicitarFechar`.

## Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.

---

# 3. TSalvarAssociacaoController (Novo)

## Importar novos services/DTO:

`System.SysUtils, System.Classes;` (padrão Delphi)
`UTabelasRelatoriosDTO;` (existente)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.CloseViewSalvarAssociacao;`)

## Novos objetos/métodos no private do `TSalvarAssociacaoController`:

`FOnSolicitarSalvar: TProc<TTabelasRelatoriosDTO>;` // Vai fazer: Disparar o salvamento da associação via serviço.
`FOnSolicitarFechar: TNotifyEvent;` // Vai fazer: Disparar o fechamento da view via `TShowViewService`.
`procedure ManipuladorSolicitarSalvar(AAssociacao: TTabelasRelatoriosDTO);` // Vai fazer: Chama serviço de salvamento.
`procedure ManipuladorSolicitarFechar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewSalvarAssociacao;`.

## Novos objetos/métodos no public:

`property OnSolicitarSalvar: TProc<TTabelasRelatoriosDTO> read FOnSolicitarSalvar write FOnSolicitarSalvar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarFechar: TNotifyEvent read FOnSolicitarFechar write FOnSolicitarFechar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarSalvar(AAssociacao: TTabelasRelatoriosDTO);` // Vai fazer: Dispara o evento `FOnSolicitarSalvar`.
`procedure SolicitarFechar;` // Vai fazer: Dispara o evento `FOnSolicitarFechar`.

## Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.

---

# 4. TCriadorTabelaDadosController (Novo)

## Importar novos services/DTO:

`System.SysUtils, System.Classes, Data.DB, Datasnap.DBClient;` (padrão Delphi)
`UTabelaDTO, UTabelaConfiguracaoDTO;` (existentes)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.ShowViewConfigurarTabela;` ou `TShowViewService.Instance.ShowViewSelecionarPlanilhaParaTabela;`)

## Novos objetos/métodos no private do `TCriadorTabelaDadosController`:

`FOnSolicitarAvancar: TNotifyEvent;` // Vai fazer: Disparar a navegação para a próxima etapa do fluxo (Configurar).
`FOnSolicitarCancelar: TNotifyEvent;` // Vai fazer: Disparar o fechamento do fluxo via `TShowViewService`.
`procedure ManipuladorSolicitarAvancar;` // Vai fazer: Chama `TShowViewService.Instance.ShowViewConfigurarTabela(...);`.
`procedure ManipuladorSolicitarCancelar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewCriadorTabelaDados;`.

## Novos objetos/métodos no public:

`property OnSolicitarAvancar: TNotifyEvent read FOnSolicitarAvancar write FOnSolicitarAvancar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarCancelar: TNotifyEvent read FOnSolicitarCancelar write FOnSolicitarCancelar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarAvancar;` // Vai fazer: Dispara o evento `FOnSolicitarAvancar`.
`procedure SolicitarCancelar;` // Vai fazer: Dispara o evento `FOnSolicitarCancelar`.

## Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.

---

# 5. TConfigurarTabelaController (Novo)

## Importar novos services/DTO:

`System.SysUtils, System.Classes;` (padrão Delphi)
`UTabelaConfiguracaoDTO;` (existente)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.ShowViewSelecionarPlanilhaParaTabela;` ou `TShowViewService.Instance.ShowViewCriadorTabelaDados;`)

## Novos objetos/métodos no private do `TConfigurarTabelaController`:

`FOnSolicitarAvancar: TProc<TConfiguracaoTabelaDTO>;` // Vai fazer: Disparar a navegação para a próxima etapa do fluxo (Selecionar Planilha), passando a configuração.
`FOnSolicitarVoltar: TNotifyEvent;` // Vai fazer: Disparar a navegação para a etapa anterior (Criador Dados).
`FOnSolicitarCancelar: TNotifyEvent;` // Vai fazer: Disparar o fechamento do fluxo via `TShowViewService`.
`procedure ManipuladorSolicitarAvancar(AConfig: TConfiguracaoTabelaDTO);` // Vai fazer: Chama `TShowViewService.Instance.ShowViewSelecionarPlanilhaParaTabela(AConfig);`.
`procedure ManipuladorSolicitarVoltar;` // Vai fazer: Chama `TShowViewService.Instance.ShowViewCriadorTabelaDados(...);`.
`procedure ManipuladorSolicitarCancelar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewConfigurarTabela;`.

## Novos objetos/métodos no public:

`property OnSolicitarAvancar: TProc<TConfiguracaoTabelaDTO> read FOnSolicitarAvancar write FOnSolicitarAvancar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarVoltar: TNotifyEvent read FOnSolicitarVoltar write FOnSolicitarVoltar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarCancelar: TNotifyEvent read FOnSolicitarCancelar write FOnSolicitarCancelar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarAvancar(AConfig: TConfiguracaoTabelaDTO);` // Vai fazer: Dispara o evento `FOnSolicitarAvancar`.
`procedure SolicitarVoltar;` // Vai fazer: Dispara o evento `FOnSolicitarVoltar`.
`procedure SolicitarCancelar;` // Vai fazer: Dispara o evento `FOnSolicitarCancelar`.

### Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.

---

## 6. TSelecionarPlanilhaController (Novo)

### Importar novos services/DTO:

`System.SysUtils, System.Classes;` (padrão Delphi)
`UTabelaConfiguracaoDTO, UPlanilhaDTO;` (existentes)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.ShowViewCriadorTabelaDados;` ou `TShowViewService.Instance.ShowViewConfigurarTabela;`)

### Novos objetos/métodos no private do `TSelecionarPlanilhaController`:

`FOnSolicitarAvancar: TProc<TConfiguracaoTabelaDTO, TPlanilhaDTO>;` // Vai fazer: Disparar a navegação para a criação final, passando configuração e planilha.
`FOnSolicitarVoltar: TNotifyEvent;` // Vai fazer: Disparar a navegação para a etapa anterior (Configurar).
`FOnSolicitarCancelar: TNotifyEvent;` // Vai fazer: Disparar o fechamento do fluxo via `TShowViewService`.
`procedure ManipuladorSolicitarAvancar(AConfig: TConfiguracaoTabelaDTO; APlanilha: TPlanilhaDTO);` // Vai fazer: Chama `TShowViewService.Instance.ShowViewCriadorTabelaDados(AConfig, APlanilha);`.
`procedure ManipuladorSolicitarVoltar;` // Vai fazer: Chama `TShowViewService.Instance.ShowViewConfigurarTabela(...);`.
`procedure ManipuladorSolicitarCancelar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewSelecionarPlanilhaParaTabela;`.

### Novos objetos/métodos no public:

`property OnSolicitarAvancar: TProc<TConfiguracaoTabelaDTO, TPlanilhaDTO> read FOnSolicitarAvancar write FOnSolicitarAvancar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnSolicitarVoltar: TNotifyEvent read FOnSolicitarVoltar write FOnSolicitarVoltar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.

`property OnSolicitarCancelar: TNotifyEvent read FOnSolicitarCancelar write FOnSolicitarCancelar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarAvancar(AConfig: TConfiguracaoTabelaDTO; APlanilha: TPlanilhaDTO);` // Vai fazer: Dispara o evento `FOnSolicitarAvancar`.
`procedure SolicitarVoltar;` // Vai fazer: Dispara o evento `FOnSolicitarVoltar`.
`procedure SolicitarCancelar;` // Vai fazer: Dispara o evento `FOnSolicitarCancelar`.

### Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.

---

## 7. TImprimirRelatorioController (Novo)

### Importar novos services/DTO:

`System.SysUtils, System.Classes;` (padrão Delphi)
`URelatorioDTO;` (existente)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.CloseViewImprimirRelatorioPronto;`)

### Novos objetos/métodos no private do `TImprimirRelatorioController`:

`FOnSolicitarImprimir: TProc<TRelatorioDTO>;` // Vai fazer: Disparar a impressão do relatório via serviço.
`FOnSolicitarFechar: TNotifyEvent;` // Vai fazer: Disparar o fechamento da view via `TShowViewService`.
`procedure ManipuladorSolicitarImprimir(ARelatorio: TRelatorioDTO);` // Vai fazer: Chama serviço de impressão.
`procedure ManipuladorSolicitarFechar;` // Vai fazer: Chama `TShowViewService.Instance.CloseViewImprimirRelatorioPronto;`.

### Novos objetos/métodos no public:

`property OnSolicitarImprimir: TProc<TRelatorioDTO> read FOnSolicitarImprimir write FOnSolicitarImprimir;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.

`property OnSolicitarFechar: TNotifyEvent read FOnSolicitarFechar write FOnSolicitarFechar;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`procedure SolicitarImprimir(ARelatorio: TRelatorioDTO);` // Vai fazer: Dispara o evento `FOnSolicitarImprimir`.
`procedure SolicitarFechar;` // Vai fazer: Dispara o evento `FOnSolicitarFechar`.

### Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.

---

## 8. TLoginController (Novo)

### Importar novos services/DTO:

`System.SysUtils, System.Classes;` (padrão Delphi)
`UUsuarioDTO, ULoginDTO;` (existentes, ou criar novos DTOs para login se necessário)
`UShowViewService;` (**Novo:** Para chamar `TShowViewService.Instance.ShowViewPrincipalModal;` ou `TShowViewService.Instance.ShowViewModalTermos;`)

### Novos objetos/métodos no private do `TLoginController`:

`FOnLogin: TProc<TLoginDTO, Boolean>;` // Vai fazer: Disparar o processo de login via serviço, passando DTO e modo público.
`FOnCancelarLogin: TNotifyEvent;` // Vai fazer: Disparar o encerramento da aplicação via `TShowViewService`.
`procedure ManipuladorLogin(ALoginDTO: TLoginDTO; AModoPublico: Boolean);` // Vai fazer: Chama serviço de autenticação e, se bem-sucedido, chama `TShowViewService.Instance.ShowViewModalTermos` ou `TShowViewService.Instance.ShowViewPrincipalModal`.
`procedure ManipuladorCancelarLogin;` // Vai fazer: Chama `TShowViewService.Instance.EncerrarAplicacao;` (ou `Application.Terminate` se for direto).

### Novos objetos/métodos no public:

`property OnLogin: TProc<TLoginDTO, Boolean> read FOnLogin write FOnLogin;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.
`property OnCancelarLogin: TNotifyEvent read FOnCancelarLogin write FOnCancelarLogin;` // Vai fazer: Permitir que a View conecte o handler estático do `TShowViewService`.

`procedure SolicitarLogin(ALoginDTO: TLoginDTO; AModoPublico: Boolean);` // Vai fazer: Dispara o evento `FOnLogin`.
`procedure SolicitarCancelarLogin;` // Vai fazer: Dispara o evento `FOnCancelarLogin`.

## Alterar existentes:

`constructor Create;` // Vai fazer: Inicialização e conexão dos manipuladores aos eventos.
`destructor Destroy; override;` // Vai fazer: Liberar o controller.