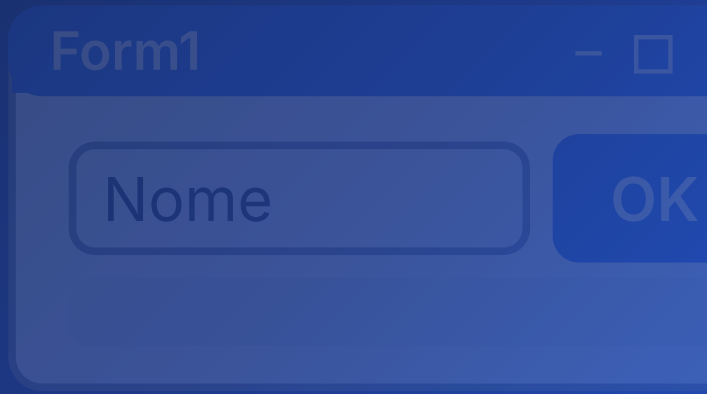
# Do Delphi para as linguagens de hoje




Glossário ilustrado de equivalência de componentes: o nome que você conhece da paleta do Delphi (VCL) e como o mesmo controle se chama em HTML5/Bootstrap, PHP (Laravel/Filament), React, Flutter, C#, Python, Android e iOS.

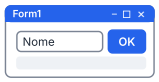
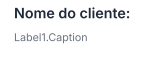
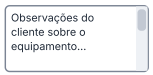
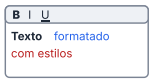
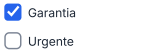
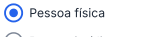
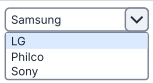
Três partes: **componentes visuais** (com ilustração), **componentes não visuais** e **propriedades e eventos** — onde mora a maior parte da "tradução mental".

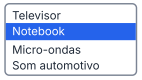
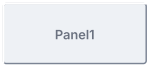
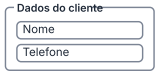
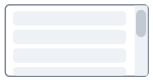
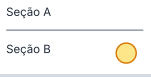
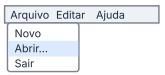
HTML5 · Bootstrap 5 | PHP · Laravel · Filament | React · MUI | Flutter | C# WinForms · WPF | Python · PySide6 · Tkinter | Jetpack Compose | SwiftUI

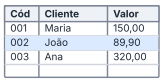
Preparado para Carlos · Bits e Bytes Eletrônica e Informática · Belém/PA · novembro de 2026

# 1. Componentes visuais

A ilustração mostra a aparência típica do controle. Quando houver dois nomes separados por "·", o primeiro é o mais comum; em Python, o primeiro é PySide6/PyQt e o segundo é Tkinter; em C#, "(WPF)" indica o equivalente no WPF.

| Ilustração | Delphi | HTML5 / Bootstrap 5 | PHP (Laravel · Filament) | React (MUI) | Flutter (Dart) | C# WinForms / WPF | Python (PySide6 · Tkinter) | Android (Jetpack Compose) | iOS/macOS (SwiftUI) |
|---|---|---|---|---|---|---|---|---|---|
|  | **TForm**<br>Janela / tela da aplicação | <body> (a página) · <dialog> · .modal | Page / Resource (Filament) · view Blade | Componente de página · <Dialog> | Scaffold (dentro de MaterialApp) | Form · Window (WPF) | QMainWindow / QWidget · Tk() / Toplevel | Activity + Scaffold | View · WindowGroup / Window |
|  | **TLabel**<br>Texto fixo / rótulo | <label> ·  · <p> | TextEntry · Placeholder · {{ $var }} | <Typography> | Text | Label · TextBlock (WPF) | QLabel · Label | Text | Text |
|  | **TEdit**<br>Campo de texto de uma linha | <input type="text"> · .form-control | TextInput::make('nome') | <TextField> | TextField / TextFormField | TextBox | QLineEdit · Entry | TextField / OutlinedTextField | TextField · SecureField (senha) |
|  | **TMaskEdit**<br>Campo com máscara (CPF, CEP, fone) | <input pattern> + lib IMask.js | TextInput ->mask('999.999.999-99') | <TextField> + react-imask | TextField + inputFormatters (mask_text_input_formatter) | MaskedTextBox | QLineEdit.setInputMask() · Entry + validação | TextField + VisualTransformation | TextField + .onChange formatando |
|  | **TMemo**<br>Texto de várias linhas | <textarea> | Textarea | <TextField multiline> | TextField(maxLines: null) | TextBox (Multiline = true) | QPlainTextEdit · Text | TextField(singleLine = false) | TextEditor |
|  | **TRichEdit**<br>Texto formatado (negrito, cores) | contenteditable · TinyMCE / Quill | RichEditor · MarkdownEditor | Tiptap · Lexical | flutter_quill (pacote) | RichTextBox | QTextEdit · Text com tags | BasicTextField + AnnotatedString | TextEditor + AttributedString |
|  | **TButton**<br>Botão de comando | <button> · .btn .btn-primary | Action · <x-filament::button> | <Button variant="contained"> | ElevatedButton / FilledButton / TextButton | Button | QPushButton · Button | Button / OutlinedButton | Button |
|  | **TBitBtn / TSpeedButton**<br>Botão com ícone / de ferramenta | <button> + Bootstrap Icons | Action ->icon('heroicon-o-...') · <x-filament::icon-button> | <IconButton> · Button startIcon | IconButton · ElevatedButton.icon | Button.Image · ToolStripButton | QToolButton · Button(image=) | IconButton | Button { Label("", systemImage:) } |
|  | **TCheckBox**<br>Caixa de marcação (sim/não) | <input type="checkbox"> · .form-check | Checkbox | <Checkbox> + FormControlLabel | Checkbox · CheckboxListTile | CheckBox | QCheckBox · Checkbutton | Checkbox | Toggle (.toggleStyle(.checkbox) no macOS) |
|  | **TToggleSwitch**<br>Chave liga/desliga | checkbox role="switch" · .form-switch | Toggle | <Switch> | Switch · SwitchListTile | CheckBox (Appearance=Button) · ToggleButton (WPF) | QCheckBox estilizado · ttk.Checkbutton | Switch | Toggle |
|  | **TRadioButton**<br>Opção exclusiva | <input type="radio"> | Radio | <Radio> | Radio · RadioListTile | RadioButton | QRadioButton · Radiobutton | RadioButton | Picker (.radioGroup no macOS) |
|  | **TRadioGroup**<br>Grupo de opções exclusivas | <fieldset> + radios com mesmo name | Radio ->options([...]) · ToggleButtons | <RadioGroup> | RadioGroup · Column de RadioListTile | GroupBox + RadioButtons | QButtonGroup + QGroupBox | Column + Modifier.selectableGroup() | Picker |
|  | **TComboBox**<br>Lista suspensa | <select> · .form-select · <datalist> | Select ->options() ->searchable() | <Select> · <Autocomplete> | DropdownButton · DropdownMenu | ComboBox | QComboBox · ttk.Combobox | ExposedDropdownMenuBox | Picker (.menu) |

| Ilustração | Delphi | HTML5 / Bootstrap 5 | PHP (Laravel · Filament) | React (MUI) | Flutter (Dart) | C# WinForms / WPF | Python (PySide6 · Tkinter) | Android (Jetpack Compose) | iOS/macOS (SwiftUI) |
|---|---|---|---|---|---|---|---|---|---|
|  | **TListBox**<br>Lista de itens selecionáveis | <select multiple> · <ul> · .list-group | Select →multiple() | <List> + <ListItemButton> | ListView + ListTile | ListBox | QListWidget · Listbox | LazyColumn | List |
|  | **TCheckListBox**<br>Lista com caixas de marcação | <ul> com checkboxes · .list-group | CheckboxList | <List> + <Checkbox> | ListView + CheckboxListTile | CheckedListBox | QListWidget (itens checáveis) | LazyColumn + Checkbox | List + Toggle |
|  | **TPanel**<br>Contêiner / painel |  · .card · .container | Grid · Group ·  no Blade | <Box> · <Paper> | Container · Row / Column | Panel · Grid / StackPanel (WPF) | QFrame / QWidget · Frame | Box · Surface · Row / Column | VStack / HStack / ZStack |
|  | **TGroupBox**<br>Moldura com título | <fieldset> + <legend> | Section · Fieldset | <Card> + <CardHeader> | Card · InputDecorator | GroupBox | QGroupBox · LabelFrame | Card / OutlinedCard | GroupBox · Section (em Form) |
|  | **TScrollBox**<br>Área com rolagem |  | CSS overflow via ->extraAttributes() | <Box sx={{overflow:'auto'}}> | SingleChildScrollView · ListView | Panel (AutoScroll=true) · ScrollViewer (WPF) | QScrollArea · Canvas + Scrollbar | Modifier.verticalScroll() | ScrollView |
|  | **TScrollBar**<br>Barra de rolagem avulsa | CSS ::-webkit-scrollbar / scrollbar-* | (nativa do navegador) | (nativa do navegador) | Scrollbar / RawScrollbar | VScrollBar / HScrollBar | QScrollBar · Scrollbar | VerticalScrollbar (Desktop) | .scrollIndicators() |
|  | **TImage**<br>Imagem / foto | <img> · <picture> · .img-fluid | ImageColumn · ImageEntry · FileUpload →image() | <img> · <Avatar> · <CardMedia> | Image.asset / Image.network | PictureBox · Image (WPF) | QLabel + QPixmap · Label + PhotoImage | Image · AsyncImage (Coil) | Image · AsyncImage |
|  | **TBevel / TShape**<br>Linha divisória / forma | <hr> · CSS border / border-radius | <hr> / CSS no Blade | <Divider> | Divider · DecoratedBox / CustomPaint | Label (BorderStyle) · Separator / Shapes (WPF) | QFrame (HLine) · ttk.Separator | HorizontalDivider · Canvas | Divider · Circle() / Rectangle() |
|  | **TPageControl / TTabSheet**<br>Abas (guias) | .nav-tabs + .tab-content | Tabs + Tab · Wizard (passo a passo) | <Tabs> + <Tab> + TabPanel | TabBar + TabBarView (DefaultTabController) | TabControl + TabPage | QTabWidget · ttk.Notebook | PrimaryTabRow + HorizontalPager | TabView |
|  | **TMainMenu**<br>Menu principal da janela | <nav> · .navbar · .dropdown | Menu lateral (navigationGroup) · UserMenu | <AppBar> + <Menu> | MenuBar · AppBar | MenuStrip · Menu (WPF) | QMenuBar · Menu | TopAppBar · MenuBar (Desktop) | .commands { CommandMenu } |
|  | **TPopupMenu**<br>Menu de contexto (botão direito) | evento contextmenu · .dropdown-menu | ActionGroup (dropdown) | <Menu> (anchorEl) | PopupMenuButton · showMenu() | ContextMenuStrip · ContextMenu (WPF) | QMenu · Menu.post() | DropdownMenu | .contextMenu { } |
|  | **TToolBar**<br>Barra de ferramentas |  flex · .btn-toolbar · .btn-group | getHeaderActions() · headerActions | <Toolbar> | AppBar(actions:) · Row de IconButton | ToolStrip · ToolBar (WPF) | QToolBar · Frame com botões | TopAppBar(actions) · BottomAppBar | .toolbar { ToolbarItem } |
|  | **TStatusBar**<br>Barra de status (rodapé) | <footer> · .fixed-bottom | renderHook (rodapé) · <footer> Blade | <Box position="fixed"> · <BottomNavigation> | BottomAppBar · bottomNavigationBar | StatusStrip · StatusBar (WPF) | QStatusBar · Label(relief=SUNKEN) | BottomAppBar | .safeAreaInset(edge: .bottom) |
|  | **TProgressBar**<br>Barra de progresso | <progress> · .progress | (sem nativo) · ViewColumn com <progress> | <LinearProgress> | LinearProgressIndicator | ProgressBar | QProgressBar · ttk.Progressbar | LinearProgressIndicator | ProgressView(value:) |

| Ilustração | Delphi | HTML5 / Bootstrap 5 | PHP (Laravel · Filament) | React (MUI) | Flutter (Dart) | C# WinForms / WPF | Python (PySide6 · Tkinter) | Android (Jetpack Compose) | iOS/macOS (SwiftUI) |
|---|---|---|---|---|---|---|---|---|---|
|  | **TActivityIndicator**<br>Indicador de "aguarde" | .spinner-border | wire:loading · <x-filament::loading-indicator> | <CircularProgress> | CircularProgressIndicator | ProgressBar (Style=Marquee) | QProgressBar (range 0,0) · ttk.Progressbar indeterminate | CircularProgressIndicator | ProgressView() |
|  | **TTrackBar**<br>Controle deslizante | <input type="range"> · .form-range | Slider (Filament 4) · <input type="range"> | <Slider> | Slider · RangeSlider | TrackBar · Slider (WPF) | QSlider · Scale | Slider · RangeSlider | Slider |
|  | **TSpinEdit / TUpDown**<br>Número com setas +/– | <input type="number"> | TextInput →numeric() ->step() | <TextField type="number"> | TextField + IconButtons (sem nativo) | NumericUpDown | QSpinBox / QDoubleSpinBox · Spinbox | TextField + IconButtons (sem nativo) | Stepper |
|  | **TDateTimePicker**<br>Seletor de data/hora | <input type="date" \| "time" \| "datetime-local"> | DatePicker · DateTimePicker · TimePicker | <DatePicker> (MUI X) | showDatePicker() · showTimePicker() | DateTimePicker · DatePicker (WPF) | QDateEdit / QDateTimeEdit · tkcalendar.DateEntry | DatePickerDialog · TimePicker | DatePicker |
|  | **TMonthCalendar**<br>Calendário embutido | (sem nativo) · libs: FullCalendar, flatpickr | (sem nativo) · plugin FullCalendar | <DateCalendar> (MUI X) | CalendarDatePicker | MonthCalendar · Calendar (WPF) | QCalendarWidget · tkcalendar.Calendar | DatePicker (embutido) | DatePicker(.graphical) |
|  | **TStringGrid / TDBGrid**<br>Grade de dados (linhas × colunas) | <table> · .table · DataTables / AG Grid | Table + TextColumn (Resource) | <DataGrid> (MUI X) | DataTable · PaginatedDataTable · pluto_grid | DataGridView · DataGrid (WPF) | QTableView / QTableWidget · ttk.Treeview | LazyColumn + Row (sem nativo) | Table (macOS/iPadOS) |
|  | **TTreeView**<br>Árvore hierárquica | <details>/<summary> · <ul> aninhado | (sem nativo) · plugins de árvore | <SimpleTreeView> / <RichTreeView> (MUI X) | TreeSliver · flutter_fancy_tree_view | TreeView | QTreeView / QTreeWidget · ttk.Treeview | LazyColumn com recuo (sem nativo) | OutlineGroup · List(children:) |
|  | **TListView**<br>Lista com ícones / colunas | .list-group · CSS Grid de cards | Table →contentGrid() (cards) | <List> · <ImageList> | ListView · GridView | ListView · ListView+GridView (WPF) | QListView (IconMode) · ttk.Treeview | LazyColumn · LazyVerticalGrid | List · LazyVGrid |
|  | **TSplitter**<br>Divisor redimensionável | CSS resize · Split.js | Split (layout fixo, sem arrastar) | react-resizable-panels | multi_split_view (pacote) | SplitContainer · GridSplitter (WPF) | QSplitter · ttk.PanedWindow | ListDetailPaneScaffold (adaptive) | HSplitView · NavigationSplitView |
|  | **TFrame**<br>Bloco de tela reutilizável | Web Component · <template> | Componente Blade <x-...> · componente Livewire | Componente próprio (function) | StatelessWidget / StatefulWidget próprio | UserControl | QWidget customizado · subclasse de Frame | Função @Composable | struct ... : View |
|  | **ShowMessage / MessageDlg**<br>Caixa de mensagem / confirmação | alert() · confirm() · <dialog> · .modal | Notification::make() · Action ->requiresConfirmation() | <Dialog> · <Snackbar> | showDialog + AlertDialog · SnackBar | MessageBox.Show() | QMessageBox · messagebox | AlertDialog · Snackbar | .alert() · .confirmationDialog() |
|  | **Hint / ShowHint**<br>Dica ao passar o mouse | atributo title="" · Bootstrap Tooltip | →tooltip() · ->helperText() | <Tooltip> | Tooltip | ToolTip · ToolTip (WPF) | setToolTip() · (Tk: sem nativo) | TooltipBox | .help("...") |
| bitsebytes.net | **TLinkLabel**<br>Texto com link | <a href> | TextColumn →url() · <x-filament::link> | <Link> | TextButton + url_launcher | LinkLabel · Hyperlink (WPF) | QLabel com HTML (setOpenExternalLinks) | Text + LinkAnnotation | Link |

# 2. Componentes não visuais (aba “System”, “Dialogs”, “Data Access”...)

No mundo moderno, muitos deixaram de ser “componentes soltos no form” e viraram classes, funções ou pacotes que você instancia no código.

| Delphi | HTML5 / Bootstrap 5 | PHP (Laravel · Filament) | React (MUI) | Flutter (Dart) | C# WinForms / WPF | Python (PySide6 · Tkinter) | Android (Jetpack Compose) | iOS/macOS (SwiftUI) |
|---|---|---|---|---|---|---|---|---|
| **TTimer**<br>Executa código a cada intervalo | setInterval() · setTimeout() | wire:poll (Livewire) · Task Scheduling / cron | useEffect + setInterval | Timer.periodic (dart:async) | Timer (System.Windows.Forms) · DispatcherTimer | QTimer · root.after() | LaunchedEffect + delay() | Timer.publish · .task { } |
| **TOpenDialog / TSaveDialog**<br>Abrir / salvar arquivo | <input type="file"> · showOpenFilePicker() | FileUpload · $_FILES + move_uploaded_file() | <input type="file"> · react-dropzone | file_picker (pacote) | OpenFileDialog / SaveFileDialog | QFileDialog · filedialog | rememberLauncherForActivityResult | .fileImporter() / .fileExporter() |
| **TColorDialog**<br>Seletor de cor | <input type="color"> | ColorPicker | react-colorful | flutter_colorpicker (pacote) | ColorDialog | QColorDialog · colorchooser | (sem nativo) · libs | ColorPicker |
| **TImageList**<br>Coleção de ícones | SVG sprites · Bootstrap Icons · Font Awesome | Heroicons ('heroicon-o-...') | @mui/icons-material | Icons.* · Cupertino Icons | ImageList | QIcon + recursos .qrc | Icons.Default.* (Material Icons) | SF Symbols · Image(systemName:) |
| **TActionList / TAction**<br>Ação centralizada (menu + botão) | funções JS reutilizadas · atalhos via keydown | Classe Action reutilizável | funções/hooks reutilizados | Actions + Shortcuts + Intents | (WinForms: sem nativo) · ICommand (WPF) | QAction · bind() | lambdas / ViewModel | .commands · .keyboardShortcut() |
| **TFDConnection / TFDQuery**<br>Conexão e consulta a banco | fetch() → API PHP (PDO) | PDO (pdo_firebird, pdo_mysql) · Eloquent / DB:: | fetch / axios · React Query | http / dio · sqflite · drift | ADO.NET (DbConnection) · Dapper · EF Core | QSqlDatabase/QSqlQuery · sqlite3 · firebird-driver · SQLAlchemy | Room · Retrofit | SwiftData · Core Data · URLSession |
| **TDataSource + TDBEdit**<br>Ligação de dados (data-aware) | JS preenchendo o DOM · Alpine.js / Vue | form() do Resource ligado ao Model Eloquent | useState / estado + props | Provider / Riverpod · FutureBuilder | BindingSource · Binding (WPF/MVVM) | QDataWidgetMapper · QSqlTableModel | ViewModel + StateFlow | @State · @Observable · @Query |
| **TClientDataSet / TFDMemTable**<br>Tabela em memória | Array de objetos / JSON · IndexedDB | array · Collection (Laravel) | Array em estado (useState) | List<Map> · classes modelo | DataTable · List<T> | list[dict] · pandas.DataFrame | List<data class> | [Struct] · Array |
| **TIniFile / TRegistry**<br>Gravar configurações | localStorage · cookies | .env · config/*.php · $_SESSION | localStorage · Context | shared_preferences | appsettings.json · Settings | QSettings · configparser | DataStore | @AppStorage · UserDefaults |
| **Clipboard**<br>Área de transferência | navigator.clipboard | TextColumn →copyable() | navigator.clipboard | Clipboard (services) | Clipboard | QClipboard · clipboard_get() | LocalClipboard | UIPasteboard / NSPasteboard |

# 3. Propriedades e eventos: o Object Inspector traduzido

O nome do componente muda pouco; o que mais muda é o jeito de mexer nele. Esta tabela traduz as propriedades e os eventos mais usados.

| Delphi | HTML5 / Bootstrap 5 | PHP (Laravel · Filament) | React (MUI) | Flutter (Dart) | C# WinForms / WPF | Python (PySide6 · Tkinter) | Android (Jetpack Compose) | iOS/macOS (SwiftUI) |
|---|---|---|---|---|---|---|---|---|
| **`Caption / Text`**<br>Texto exibido no controle | innerText · value | →label('...') | children · value / label | Text('...') · controller.text | Text · Content (WPF) | setText() / text() | text = "..." | Text("...") |
| **`Edit1.Text (ler valor)`**<br>Obter o que o usuário digitou | document.getElementById('x').value | $_POST['nome'] · $get('nome') (Filament) | estado: const [v,setV] = useState() | TextEditingController.text | textBox1.Text | lineEdit.text() · entry.get() | var v by remember { mutableStateOf("") } | @State var v = "" |
| **`Name`**<br>Identificador do componente | atributo id | TextInput::make('nome') (nome do campo) | useRef() · props | GlobalKey · controller | Name · x:Name (WPF) | setObjectName() · variável | variável / state | propriedade da View |
| **`Align (alTop, alClient…)`**<br>Encaixe automático no pai | Flexbox · CSS Grid | Grid →columns(3) · ->columnSpan() | <Box display="flex"> · <Grid> | Column · Row · Expanded | Dock · DockPanel (WPF) | QVBoxLayout / QHBoxLayout · pack() | Column · Row · fillMaxSize() | VStack · HStack · .frame(maxWidth: .infinity) |
| **`Anchors`**<br>Fixar bordas ao redimensionar | position · flex · % / vw | →columnSpanFull() · classes Tailwind | sx / flex | Align · Positioned (Stack) | Anchor · HorizontalAlignment (WPF) | layouts + stretch · grid(sticky=) | Modifier.align() | .frame(alignment:) · Spacer() |
| **`Visible`**<br>Mostrar / ocultar | hidden · display:none · .d-none | →visible() / →hidden() | renderização condicional {cond && ...} | Visibility · if (cond) ... | Visible · Visibility (WPF) | setVisible() · pack_forget() | if (cond) { ... } · AnimatedVisibility | if cond { ... } · .hidden() |
| **`Enabled`**<br>Habilitar / desabilitar | atributo disabled | →disabled() | prop disabled | onPressed: null · enabled: false | Enabled · IsEnabled (WPF) | setEnabled() · state='disabled' | enabled = false | .disabled(true) |
| **`OnClick`**<br>Evento de clique | onclick · addEventListener('click') | Action →action(fn () ⇒ ...) | onClick={...} | onPressed / onTap | Click | clicked.connect() · command= | onClick = { ... } | Button(action: { ... }) |
| **`OnChange`**<br>Valor alterado | input / change | →live() + ->afterStateUpdated() | onChange | onChanged | TextChanged · SelectedIndexChanged | textChanged · trace_add() | onValueChange | .onChange(of:) |
| **`OnCreate / OnShow`**<br>Ao abrir a tela | DOMContentLoaded | mount() · mutateFormDataBeforeFill() | useEffect(() ⇒ {...}, []) | initState() | Load · Loaded (WPF) | __init__ · showEvent() | LaunchedEffect(Unit) | .onAppear · .task |
| **`OnClose / OnDestroy`**<br>Ao fechar a tela | beforeunload | afterSave() · redirect (HTTP não tem estado) | retorno (cleanup) do useEffect | dispose() | FormClosing · Closing (WPF) | closeEvent() · protocol('WM_DELETE_WINDOW') | DisposableEffect { onDispose } | .onDisappear |
| **`Font / Color`**<br>Aparência | CSS (font, color, background) | →color('danger') · classes Tailwind | sx={{...}} · theme | TextStyle · Colors · ThemeData | Font · ForeColor · BackColor | setStyleSheet() (QSS) · font= / bg= | TextStyle · Modifier.background() | .font() · .foregroundStyle() |
| **`Hint`**<br>Dica | title | →tooltip() · →hint() | <Tooltip> | Tooltip | ToolTip | setToolTip() | TooltipBox | .help() |
| **`TabOrder`**<br>Ordem do Tab | tabindex | tabindex · ordem dos campos no schema | tabIndex | FocusTraversalOrder | TabIndex | setTabOrder() | Modifier.focusProperties | .focusable() · @FocusState |
| **`Tag`**<br>Valor extra livre | atributos data-* | →extraAttributes(['data-x' ⇒ ...]) | props extras | campos no Widget / Key | Tag | setProperty() · setData() | parâmetros extras | .tag() |

| Delphi | HTML5 / Bootstrap 5 | PHP (Laravel · Filament) | React (MUI) | Flutter (Dart) | C# WinForms / WPF | Python (PySide6 · Tkinter) | Android (Jetpack Compose) | iOS/macOS (SwiftUI) |
|---|---|---|---|---|---|---|---|---|
| **`Owner / Parent`** Quem contém o controle | elemento pai no DOM | schema aninhado ->schema([...]) | árvore de componentes (children) | árvore de widgets (child/children) | Parent · Controls.Add() | setParent() · master | hierarquia de composables | hierarquia de Views |

**1. Do "Edit1.Text := ..." para o estado.** Em React, Flutter, Compose e SwiftUI você normalmente não altera o controle diretamente: altera uma *variável de estado* e a tela se redesenha sozinha. É a maior mudança de mentalidade para quem vem do Delphi.

**2. Do Align/Anchors para os layouts.** Em vez de posicionar com Left/Top, você empilha: `Column` / `Row` (Flutter), `VStack` / `HStack` (SwiftUI), Flexbox/Grid (CSS). Pense num TPanel com `alTop` dentro de outro TPanel com `alClient`.

**3. Onde o Delphi ainda se sente em casa.** C# WinForms (criado por Anders Hejlsberg, o mesmo arquiteto do Delphi) e Qt/PySide6 têm o modelo "arrastar componente + evento" mais parecido com a VCL. O Lazarus/Free Pascal mantém os mesmos nomes da VCL.

**4. FireMonkey (FMX).** O próprio Delphi moderno usa nomes quase iguais no FMX (multiplataforma): TEdit, TLabel e TButton continuam, enquanto TListBox, TListView e TTabControl ganham versões próprias. Útil como ponte para Android/iOS sem sair do Pascal.

**5. E o PHP? A turma da antiga não foi esquecida.** O PHP puro não tem componentes visuais: ele gera o HTML (veja a coluna HTML5). Quem dá ao PHP uma "paleta de componentes" é o **Filament** (sobre Laravel + Livewire). Ali, `TextInput::make('nome')->label('Nome')->required()` é praticamente um TEdit configurado no Object Inspector, e um *Resource* funciona como o par TDataSource + TDBGrid + form de edição. Para quem vem do Delphi, é o lugar mais confortável do mundo web. Nomes de classes conforme o Filament 3/4.

**Legenda:** "MUI X" = componentes avançados do Material UI (alguns recursos pagos); "(pacote)" = biblioteca de terceiros via pub.dev/npm; "(sem nativo)" = não existe controle pronto e monta-se combinando outros. Nomes conferidos com as versões estáveis de 2025–2026; frameworks evoluem, então vale consultar a documentação oficial ao implementar.