6.1.3 Projekt

Nach den Ausführungen in den letzen beiden Kapiteln wird Ihnen das Projekt *PathProject* vorgestellt, dass die ausführbare Datei *PathProject.gambas* als Gambas-Archiv erzeugt. Das Programm wurde für die folgenden drei Szenarien unter dem Aspekt 'Arbeit mit Pfaden in Gambas' entworfen und getestet:

```
1. Start in der IDE (F5 oder über den Button in der Symbolleiste mit dem Symbol ▶)
2. Start im Projektverzeichnis mit
   2.1  $ gbx3 $HOME/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/PathProject
   oder
   2.2  $ gbr3 $HOME/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/PathProject/PathProject.gambas
3. Start auf dem Desktop mit hans@mint-183 ~/Schreibtisch $ ./PathProject.gambas, wenn zum Beispiel nur das
   Gambas-Archiv auf den Desktop kopiert wurde.
```

Das Projekt realisiert einen einfachen Datenbank-Client. Eine SQLite3-Datenbank mit einer Tabelle (Datei contacts.s3db) und weitere Dateien (Hilfe-Datei, Skript-Datei, Sound-Dateien und Bild-Dateien) werden in speziellen Verzeichnissen im Projektverzeichnis bereitgestellt oder dort erzeugt. Die Datenbank-Tabelle kann gelesen und geändert werden.

```
~/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/PathProject
├── data
│   ├── databases
│   │   ├── contacts_original.s3db
│   │   └── contacts.s3db
│   ├── texts
│   └── xml
├── help
│   └── help.txt
├── leds
│   ├── gray16.png
│   ├── green16.png
│   └── red16.png
├── scripts
│   └── dump.sh
├── sounds
│   ├── sound_e.wav
│   ├── sound_s.wav
│   └── start.ogg
└── symbols
    ├── db32.png
    ├── form_icon.png
    ├── logo3.png
    └── project_icon.png
```

Erste Überlegungen in Bezug auf die Arbeit mit Dateien gelten auch in diesem Projekt generell den Antworten auf Fragen wie diesen:

- Werden Konfigurationsdateien benötigt?
- Auf welche Dateien wird nur lesend zugegriffen (r)? Pfade?
- Welche Dateien müssen gelesen und beschrieben werden (rw)?
- Werden Skripte, NamedPipes oder (lokale) Unix-Sockets verwendet?
- Erfordern Dateien das Recht zum Ausführen?
- Werden Dateien benötigt, die nur zur Laufzeit temporär erzeugt werden (Zwischenergebnisse speichern, Skripte, NamedPipes, (lokale) Unix-Sockets)?
- Müssen relevante Daten – gespeichert in Dateien – auch nach einem Neustart des Systems zur Verfügung stehen?

Für den Fall, dass Sie Dateien in ein geeignetes Verzeichnis im System temporär oder permanent kopieren müssen, sollten Sie den folgenden allgemeinen Vorschlägen und Empfehlungen folgen.

Konfigurationsdatei:
Setzen Sie bei Konfigurationsdateien konsequent die Komponente gb.settings ein. Verwenden Sie als Pfad für die Konfigurationsdatei nicht den Standard-Pfad. Setzen Sie in diesem Fall als Basis-Verzeichnis ~/Desktop.ConfigDir/... ein. Bei einer CSS-Datei zum Beispiel müssen Sie entscheiden, ob diese als Konfigurationsdatei oder als Daten-Datei aufgefasst werden soll.

Daten-Datei:
Müssen Daten auch nach einem Neustart des Systems (permanent) zur Verfügung stehen, dann gehören diese Daten-Dateien in das Basis-Verzeichnis ~/Desktop.DataDir/... . Daten, die zur Laufzeit als Zwischenergebnisse erzeugt und gespeichert werden, sollten Sie in temporären Dateien entweder in

/tmp/... oder ~/Desktop.RuntimeDir/... speichern. Nach /tmp/... oder ~/Desktop.RuntimeDir/... gehören auch NamedPipes und lokale UnixSockets, weil sie an die Laufzeit des Gambas-Prozesses gebunden sind.

Datenbank (DB):
Eine Datenbank respektive deren DB-Tabellen müssen nur gelesen werden, wenn Sie deren Datensätze lediglich anzeigen wollen. Sofern Änderungen an der DB-Tabellen zugelassen, müssen die Datenbank und deren Tabellen lesbar sein und vom Benutzer beschrieben werden können. Beachten Sie: Eine SQLite3-Datenbank hat nach dem Erzeugen automatisch nur die Rechte: rw-r--r-- .

Skript-Datei:
Skripte "*.{sh,sql,pl,py,gbs,...}" müssen gelesen und ausgeführt werden können. Da die Shell- und Exec-Instruktionen absolute Pfade erfordern, müssen Sie die Skripte in ein geeignetes Verzeichnis im System kopieren und mit *CHMOD path2script TO permission_string* ausführbar machen. Da Shell-Skripte lediglich temporär verwendet werden, gehören sie nach /tmp/... oder ~/Desktop.RuntimeDir/... .

Es wurde festgelegt, dass das Shell-Skript dump.sh nur temporär verwendet wird und daher in das Verzeichnis /run/user/user-id/gambasbook/pathproject/scripts kopiert wird.

Sound-Datei:
Eine Sound-Datei muss nur gelesen werden. Räumen Sie aber dem Nutzer die Chance ein, die Sound-Dateien – unter den alten Dateinamen – zu ersetzen, dann müssen Sie die Sound-Dateien in ein geeignetes Basis-Verzeichnis wie ~/Desktop.DataDir/... permanent kopieren.

Bild-Datei:
Benötigen Sie für bestimmte Steuerelemente jeweils ein spezielles Icon, dann müssen diese nur gelesen werden.

Hilfe-Datei:
Eine Hilfe-Datei muss im Normalfall einzig gelesen werden. Es könnte aber ein Interesse des Benutzers bestehen, die Hilfe-Datei außerhalb des Programmes zu verändern, um zum Beispiel eine Übersetzung in einer weiteren Sprache einzufügen.
Ist ein derartiges Interesse nicht gegeben, dann bleibt sie versteckt in den Projekt-Dateien – sonst ist sie in ein geeignetes Basis-Verzeichnis wie ~/Desktop.DataDir/... permanent zu kopieren.

In der folgenden Tabelle sehen Sie eine Übersicht zu den vorgesehenen Pfaden spezieller Verzeichnisse:

Dateien	Art	rwx	Verzeichnis-Pfad
Bild-Dateien1	lokal	r	Projektverzeichnis/leds
Bild-Dateien2	lokal	r	Projektverzeichnis/symbols
Hilfe-Dateien	lokal	r	Projektverzeichnis/help
Sound-Dateien	lokal	r	Projektverzeichnis/sounds
Skript-Dateien	lokal	-	Projektverzeichnis/scripts
Daten-Dateien	lokal	-	Projektverzeichnis/data
Datenbanken	lokal	-	Projektverzeichnis/data/database
Daten-Dateien	global	rw	~/.local/share/gambasbook/pathproject/data
Datenbank	global	rw	~/.local/share/gambasbook/pathproject/data/database
Sound-Dateien	global	r	~/.local/share/gambasbook/pathproject/sounds
Skript-Dateien	global	rx	/run/user/user-id/gambasbook/pathproject/scripts (temporär)
Konfigurationsdateien	global	rw	~/.config/gambasbook/pathproject

Tabelle 6.1.3.1 : Übersicht zu den genutzten (Basis)Verzeichnissen

Die Kombination aus Anbieter-Name/ProjektName als *gambasbook/pathproject* dient konsequent als NameSpace. Beachten Sie, dass sowohl der Anbieter-Name (Vendor) – oft ist es der Name des Ent-

wicklers – als auch der Name der Anwendung in den beiden Variablen sVendor und sAppName fest codiert wurden. Der Grund liegt beim Projektnamen darin, dass Sie die Eigenschaft *Application.Name* nicht sicher verwenden können, worauf im vorangehenden Kapitel eingegangen wurde. Den Namen des Anbieters könnten Sie aus der (versteckten) Datei .project auslesen – was aber voraussetzt, dass dieser Namen in den Projekteigenschaften eingetragen wurde. Ist das nicht der Fall, dann wird er festgelegt:

```
If Exist(".../.project") Then
   For Each sRow In Split(File.Load(".../.project"), gb.NewLine)
     If sRow Begins "Vendor" Then
        sVendor = Scan(sRow, "*=*")[1]
     Endif
   Next
Endif
If Not sVendor Then sVendor = "gambasbook" ' The vendor for all projects of the author is `gambasbook`.
```

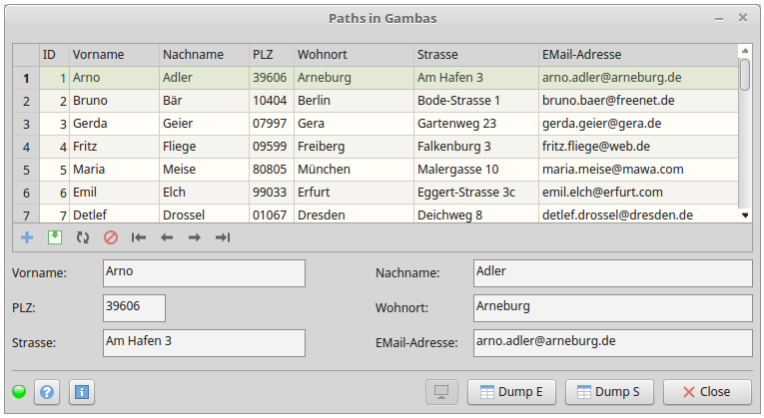

Abbildung 6.1.3.1: Demonstrationsprogramm ´Pfade in Gambas´

Der Quelltext für die Klasse FMain.class wird vollständig angegeben. Alle weiteren Klassen und Module finden Sie im Download-Verzeichnis im Projekt-Archiv.

```
' Gambas class file

' SQLite has no concept of users. Access to a database is controlled by the actual file permissions
' of the database file. This means that the Login is always the user id executing the Gambas program.

Public cC As Component

Public sTempDir As String
'-----------------------------------
Public sGlobalConfigDir As String
Public sGlobalRuntimeDir As String
Public sGlobalDataDir As String
Public sGlobalDBHostDir As String
Public sGlobalScriptsDir As String
Public sGlobalSoundsDir As String
'-----------------------------------
Public sLocalDataDir As String
Public sLocalDBHostDir As String
Public sLocalHelpDir As String
Public sLocalLEDsDir As String
Public sLocalScriptsDir As String
Public sLocalSoundsDir As String
Public sLocalSymbolsDir As String
'-----------------------------------
Public sVendor As String
Public sAppName As String
Public sDBName As String
Public sFile As String
Public hSettings As Settings
Public sMessage As String
Public sDBTableName As String

Public Sub _new()

  Dim sRow As String

  If Exist(".../.project") Then
     For Each sRow In Split(File.Load(".../.project"), gb.NewLine)
       If sRow Begins "Vendor" Then
          sVendor = Scan(sRow, "*=*")[1]
```

```
      Endif
    Next
  Endif
  If Not sVendor Then sVendor = "gambasbook"

  sAppName = "PathProject"
  sDBName = "contacts.s3db"
  sDBTableName = "contacts"
'-----------------------------
  sLocalDataDir = "data"
  sLocalDBHostDir = "databases"
  sLocalHelpDir = "help"
  sLocalLEDsDir = "leds"
  sLocalScriptsDir = "scripts"
  sLocalSoundsDir = "sounds"
  sLocalSymbolsDir = "symbols"
'-------------------------------------------------------------------------------------------
  sGlobalConfigDir = Desktop.ConfigDir &/ sVendor &/ Lower(sAppName)
  If Not Exist(sGlobalConfigDir) Then Shell.MkDir(sGlobalConfigDir)
'-------------------------------------------------------------------------------------------
  sGlobalRuntimeDir = Desktop.RuntimeDir &/ sVendor &/ Lower(sAppName)
  If Not Exist(sGlobalRuntimeDir) Then Shell.MkDir(sGlobalRuntimeDir)
'-------------------------------------------------------------------------------------------
  sGlobalDataDir = Desktop.DataDir &/ sVendor &/ Lower(sAppName) &/ GetDirPath(sLocalDataDir)
  If Not Exist(sGlobalDataDir) Then Shell.MkDir(sGlobalDataDir)
'-------------------------------------------------------------------------------------------
  sGlobalDBHostDir = sGlobalDataDir &/ sLocalDBHostDir
  If Not Exist(sGlobalDBHostDir) Then Shell.MkDir(sGlobalDBHostDir)
  For Each sFile In Dir(sLocalDataDir &/ sLocalDBHostDir, "*.s3db")
    If Not Exist(sGlobalDBHostDir &/ sFile) Then
       Copy sLocalDataDir &/ sLocalDBHostDir &/ sFile To sGlobalDBHostDir &/ sFile
    Endif
  Next
'-------------------------------------------------------------------------------------------
  sGlobalScriptsDir = sGlobalRuntimeDir &/ GetDirPath(sLocalScriptsDir)
  If Not Exist(sGlobalScriptsDir) Then
     Shell.MkDir(sGlobalScriptsDir)
  Endif
  For Each sFile In Dir(sLocalScriptsDir, "*.{sh,sql,pl,py,gbs}")
    If Not Exist(sGlobalScriptsDir &/ sFile) Then
       Copy sLocalScriptsDir &/ sFile To sGlobalScriptsDir &/ sFile
       Chmod sGlobalScriptsDir &/ sFile To "..x......"
    Endif
  Next
'-------------------------------------------------------------------------------------------
  sGlobalSoundsDir = Desktop.DataDir &/ sVendor &/ Lower(sAppName) &/ GetDirPath(sLocalSoundsDir)
  If Not Exist(sGlobalSoundsDir) Then
     Shell.MkDir(sGlobalSoundsDir)
  Endif
  For Each sFile In Dir(sLocalSoundsDir, "*.{ogg,wav}")
    If Not Exist(sGlobalSoundsDir &/ sFile) Then
       Copy sLocalSoundsDir &/ sFile To sGlobalSoundsDir &/ sFile
    Endif
  Next
'-------------------------------------------------------------------------------------------

  hSettings = New Settings(sGlobalConfigDir &/ File.SetExt(Lower(sAppName), "conf"))
  SetNotification("dialog-information", "Attention!")

End

Public Sub Form_Open()

  Dim i As Integer

  If Not System.Exist("sqlite3") Then
     sMessage = "<b><font size='+1', color='DarkRed'>"
     sMessage &= "The application requires the program 'sqlite3'."
     sMessage &= "</b></font><hr>"
     sMessage &= "Installation console (MINT): $ sudo apt-get install sqlite3"
     sMessage &= "\nThe application is therefore terminated!"
     MAddOns.PlaySound(sGlobalSoundsDir &/ "sound1.ogg")
     Message.Info(sMessage)
     Quit
  Else
    ' Message.Info("The programm path from 'sqlite3' is: " & System.Find("sqlite3"))
  Endif

' The 'About ...' window is displayed exactly twice.
  If hSettings["First/Value", 0] < 2 Then
     i = hSettings["First/Value", 0]
     Inc i
     hSettings["First/Value"] = i
     hSettings.Save()
```

```
    FAbout.ShowModal()
  Endif

  DataSource1.Connection = DBCS.DBConnection
  sGlobalDBHostDir = DBCS.DBConnection.Host
  DataSource1.Table = sDBTableName

  SetDBBrowserProperties()
  SetDataControlProperties()

  btnShowDumpData.Enabled = False

  pboxOnOff.Picture = Picture.Load(".../leds/green16.png")
' pboxOnOff.Picture = Picture["leds/green16.png"] ' Alternative

  FMain.Icon = Picture[sLocalSymbolsDir &/ "project_icon.png"]

End

Public Sub btnDBSicherungDumpE_Click()
  btnShowDumpData.Enabled = True
  MAddOns.PlaySound(sGlobalSoundsDir &/ "sound_e.wav")
  pboxOnOff.Picture = Picture["leds/red16.png"]
  GetDBDumpExec()
  Wait 0.2
  pboxOnOff.Picture = Picture["leds/green16.png"]
End

Public Sub btnDBSicherungDumpS_Click()
  btnShowDumpData.Enabled = True
  MAddOns.PlaySound(sGlobalSoundsDir &/ "sound_s.wav")
  pboxOnOff.Picture = Picture.Load(".../leds/red16.png")
  GetDBDumpShell()
  Wait 0.2
  pboxOnOff.Picture = Picture.Load(".../leds/green16.png")
End

Public Sub btnHelp_Click()
  FHelp.ShowModal()
End

Public Sub btnShowDumpData_Click()
  FShowDump.ShowModal()
  btnShowDumpData.Enabled = False
End

Public Sub btnHelpLibrary_Click()

  Dim cl As Class

  cC = Component.Load(":gambasbook/LPathProject:1.2")
  cl = Class.Load("Module1")
  Object.Call(cl, cl.Symbols[0], Null)
  Catch
  Message.Error(Error.Text)

End

Private Sub SetDBBrowserProperties()
  DataBrowser1.Labels = ["ID", "Vorname", "Nachname", "PLZ", "Wohnort", "Strasse", "EMail-Adresse"]
  DataBrowser1.CanCreate = True
  DataBrowser1.CanDelete = True
  DataBrowser1.Editable = False
  DataBrowser1.View.Clear()
  DataBrowser1.Columns = ["id", "vorname", "nachname", "plz", "wohnort", "strasse", "email"]
  DataBrowser1.View.Columns[0].Width = 30
  DataBrowser1.View.Columns[1].Width = 90
  DataBrowser1.View.Columns[2].Width = 90
  DataBrowser1.View.Columns[3].Width = 45
  DataBrowser1.View.Columns[4].Width = 120
  DataBrowser1.View.Columns[5].Width = 125
  DataBrowser1.View.Columns[6].Width = 1
  DataBrowser1.View.MoveTo(0, 0) ' … necessary!
End

Private Sub SetDataControlProperties()
  dcVorname.Field = "vorname"  ' Data control - DB fields are data-sensitive!
  dcNachname.Field = "nachname"
  dcPLZ.Field = "plz"
  cdWohnort.Field = "wohnort"
  dcStrasse.Field = "strasse"
  dcEMailAdresse.Field = "email"
End
```

```
Private Sub GetDBDumpShell()

  Dim sShellCommand, sParameter1, sParameter2 As String

' The file extension .sql is automatically supplemented with the Dump command!
' The quotation marks prevent the interpretation of the blanks as a tax mark.
  sParameter1 = Shell$(sGlobalDBHostDir &/ sDBName)
  sParameter2 = Shell$(sGlobalDBHostDir &/ "dump." & sDBTableName)
  sShellCommand = Subst$("sqlite3 &1 .dump .quit > &2.sql", sParameter1, sParameter2)
  Shell sShellCommand Wait
End

Private Sub GetDBDumpExec()

  Dim aExecCommand As String[]
  Dim sTempScriptPath, sLocalScriptPath As String

  sLocalScriptPath = sLocalScriptsDir &/ "dump.sh"
  sTempScriptPath = Temp(File.BaseName(sLocalScriptPath)) & ".sh" ' Absolute path!

' The query is necessary because an existing file is not overwritten.
  If Not Exist(sTempScriptPath) Then Copy sLocalScriptPath To sTempScriptPath
' Chmod sTempScriptPath To "rwxr--r--"  ' All rights are *explicitly* set anew!
' Only the "Execute" permission is set for the owner - all others are retained.
  Chmod sTempScriptPath To "..x......"

' The file extension .sql is automatically supplemented with the Dump command!
  aExecCommand = [sTempScriptPath, sGlobalDBHostDir &/ sDBName, sGlobalDBHostDir &/ "dump." & sDBTableName]
  Exec aExecCommand Wait
End

Private Function GetDirPath(sDir As String) As String
  If sDir Not Begins "." Then
     Return sDir
  Else
     Return Scan(sDir, "*/*")[1]
  Endif
End

Private Sub SetNotification(Icon As String, BaseText As String)

  Dim sNotificationIcon, sNotificationBaseText, sNotificationText As String

  sNotificationIcon = Icon ' Options: "dialog-information" or "dialog-warning"
  sNotificationBaseText = BaseText ' For example: "Attention!"
  sNotificationText = ("The program demonstrates the use of file paths in Gambas.")

  MAddOns.SendNotification(sNotificationIcon, sNotificationBaseText, sNotificationText, -1)
  MAddOns.PlaySound(sGlobalSoundsDir &/ "start.ogg")

End

Public Sub btnClose_Click()
  If DBCS.DBConnection.Opened Then DBCS.DBConnection.Close()
  FMain.Close()
End

Public Sub Form_Close()
  DBCS.DBConnection.Close()
End
```