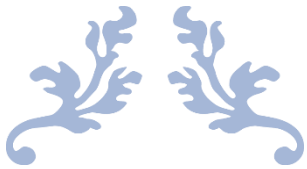

GUIA DE REFERENCIA

MANUAL DE REFERENCIA BÁSICO DE LA APLICACIÓN

Autor del documento

Centro de Apoyo Tecnológico a Emprendedores, Fundación Parque Científico y Tecnológico de Albacete

Datos de contacto

E-Mail: bilib@bilib.es

Página Web: www.bilib.es

Teléfono: 967 555 311

Versión del documento

1.0

Fecha: 07-06-2013

Contenido

GUIA DE REFERENCIA

```
# Creamos un modelo relacional de SQL
self.modelo = QSqlRelationalTableModel(db=db)
# Establecemos Album como tabla del modelo
self.modelo.setTable("Alumno")
# Establecemos la relación entre el ID de los artistas y su nombre, para que se muestre este último
# Hacemos el select del modelo
self.modelo.select()
# Renombramos las cabeceras de la tabla
self.modelo.setHeaderData(0, Qt.Horizontal, "alumnoId")
self.modelo.setHeaderData(1, Qt.Horizontal, "nombre")
self.modelo.setHeaderData(2, Qt.Horizontal, "Apellidos")
self.modelo.setHeaderData(3, Qt.Horizontal, "cursoId")
self.modelo.setHeaderData(4, Qt.Horizontal, "Media1ªEV")
self.modelo.setHeaderData(5, Qt.Horizontal, "Media2ªEV")
self.modelo.setHeaderData(6, Qt.Horizontal, "Media3ªEV")


# Establecemos el modelo en el tableView
self.tabla.setModel(self.modelo)
# Ajustamos el tamaño de las columnas al contenido
self.tabla.resizeColumnsToContents()
# Deshabilitamos la edición directa de la tabla
self.tabla.setEditTriggers(QAbstractItemView.NoEditTriggers)
# Establecemos que se seleccionen filas completas en lugar de celdas individuales
self.tabla.setSelectionMode(QAbstractItemView.SingleSelection)
self.tabla.setSelectionBehavior(QAbstractItemView.SelectRows)
# Creamos la señal: Cuando cambie la seleccion, ejecuta self.seleccion
self.tabla.selectionModel().selectionChanged.connect(self.seleccion)
# Creamos la señal: Cuando se ejecute la acción Modificar, ejecuta self.modificar
self.actionModificar.triggered.connect(self.modificar)
# Creamos la señal: Cuando se ejecute la acción Insertar, ejecuta self.nueva
self.actionInsertar.triggered.connect(self.nueva)
# Creamos la señal: Cuando se ejecute la acción Eliminar, ejecuta self.borrar
self.actionEliminar.triggered.connect(self.borrar)
# Ponemos la fila inicial a un valor que indica que no está seleccionada ninguna fila
self.fila = -1
self.actionGenerarInforme.triggered.connect(self.informe)
```

Inserción de la Base de Datos

Seleccionamos la tabla a mostrar, creamos los encabezados que contendrán

Cada tupla en el table widget y ponemos todas las conexiones necesarias

SELECCIONAR

Método que nos permite seleccionar una fila en el table view y se lleva los datos

A los LineEdit de la zona superior de las tablas

```
def seleccion(self, seleccion):
    # Recuerda que indexes almacena los índices de la selección
    if seleccion.indexes():
        # Nos quedamos con la fila del primer índice (solo se puede seleccionar una fila)
        self.fila = seleccion.indexes()[0].row()
        # Obtenemos los valores id, titulo y artista del modelo en esa fila
        id = self.modelo.index(self.fila, 0).data()
        nombre = self.modelo.index(self.fila, 1).data()
        Apellidos = self.modelo.index(self.fila, 2).data()
        cursoId = self.modelo.index(self.fila, 3).data()
        Media1ªEV = self.modelo.index(self.fila, 4).data()
        Media2ªEV = self.modelo.index(self.fila, 5).data()
        Media3ªEV = self.modelo.index(self.fila, 6).data()
        # Modificamos los campos del formulario para establecer esos valores
        self.lineEdit_ID.setText(str(id))
        self.lineEdit_Title.setText(nombre)
        self.lineEdit.setText(str(cursoId))
        self.lineEdit_2.setText(Apellidos)
        self.lineEdit_3.setText(str(Media1ªEV))
        self.lineEdit_4.setText(str(Media2ªEV))
        self.lineEdit_5.setText(str(Media3ªEV))

    else:
        # Si no hay selección,  ponemos la fila inicial a un valor que indica que no está seleccionada ninguna fila
        self.fila = -1
```

MODIFICAR

Obtenemos los datos de los LineEdit, para luego cambiar los del modelo

Y finalmente actualizarmos el modelo

```python
def modificar(self):
    # Si es una fila válida la seleccionada
    if self.fila >= 0:
        # Obtenemos los valores de los campos del formulario

        nombre = self.lineEdit_Title.text()
        Apellidos = self.lineEdit_2.text()
        cursoId = self.lineEdit.text()
        Media1ªEV = self.lineEdit_3.text()
        Media2ªEV = self.lineEdit_4.text()
        Media3ªEV = self.lineEdit_5.text()
        # Actualizamos los campos en el model

        self.modelo.setData(self.modelo.index(self.fila, 1), nombre)
        self.modelo.setData(self.modelo.index(self.fila, 2), Apellidos)
        self.modelo.setData(self.modelo.index(self.fila, 3), cursoId)
        self.modelo.setData(self.modelo.index(self.fila, 4), Media1ªEV)
        self.modelo.setData(self.modelo.index(self.fila, 5), Media2ªEV)
        self.modelo.setData(self.modelo.index(self.fila, 6), Media3ªEV)
        # Ejecutamos los cambios en el modelo
        self.modelo.submit()
```

INSERTAR

Aquí contamos las filas totales para insertar una nueva vacía en la última posición

```python
def nueva(self):
    # Guardamos en la variable nuevaFila el número de filas del modelo
    nuevaFila = self.modelo.rowCount()
    # Insertamos una nueva fila en el modelo en la posición de ese valor
    self.modelo.insertRow(nuevaFila)
    # Seleccionamos la fila nueva
    self.tabla.selectRow(nuevaFila)
    # Ponemos en blanco el texto del título en el formulario
    self.lineEdit_Title.setText("")
    # Ponemos el comboBox de artistas al primero de la lista
    self.lineEdit.setText("")
    self.lineEdit_2.setText("")
    self.lineEdit_ID.setText("")
    self.lineEdit_3.setText("")
    self.lineEdit_4.setText("")
    self.lineEdit_5.setText("")
    # Establecemos en blanco los valores (título y artista) de esa nueva fila
    self.modelo.setData(self.modelo.index(nuevaFila, 1), "")
    self.modelo.setData(self.modelo.index(nuevaFila, 2), "")
    self.modelo.setData(self.modelo.index(nuevaFila, 3), "")
    self.modelo.setData(self.modelo.index(nuevaFila, 4), "")
    self.modelo.setData(self.modelo.index(nuevaFila, 5), "")
    self.modelo.setData(self.modelo.index(nuevaFila, 6), "")
    # Ejecutamos los cambios en el modelo
    self.modelo.submit()
```

BORRADO

Eliminamos la fila y dejamos vacíos los campos LineEdit

```python
def borrar(self):
    # Si es una fila válida la seleccionada
    if self.fila >= 0:
        # Borramos la fila en el modelo
        self.modelo.removeRow(self.fila)
        # Actualizamos la tabla
        self.modelo.select()
        # Y ponemos la fila actual a -1
        self.fila = -1
        # Reseteamos los valores en los campos del formulario
        self.lineEdit_ID.setText("")
        self.lineEdit_Title.setText("")
        self.lineEdit.setText("")
        self.lineEdit_2.setText("")
        self.lineEdit_3.setText("")
        self.lineEdit_4.setText("")
        self.lineEdit_5.setText("")
```

GENERACION DE INFORME MEDIANTE WIZARD

WIZARD

Creamos el QWizard le damos un estilo, le añadimos las pestañas, recogemos los datos que necesitaremos posteriormente en el informe y llamamos al método que genera el informe

```
def informe(self):
    try:
        self.wizard = QWizard()
        self.wizard.setWizardStyle(QWizard.ModernStyle)
        self.wizard.setPixmap(QWizard.WatermarkPixmap,QPixmap('Watermark.png'))
        self.wizard.setPixmap(QWizard.LogoPixmap,QPixmap('Logo.png'))
        self.wizard.setPixmap(QWizard.BannerPixmap,QPixmap('Banner.png'))
        self.page1 = QWizardPage()
        self.page1.setTitle('Informe del alumno')
        self.textarea = QTextEdit()
        etiqueta = QLabel()
        VLayout1 = QVBoxLayout(self.page1)
        etiqueta.setText("Observaciones del Alumno")
        VLayout1.addWidget(etiqueta)
        VLayout1.addWidget(self.textarea)
        self.wizard.addPage(self.page1)


        self.page2 = QWizardPage()
        formlayout=QFormLayout(self.page2)
        self.page2.setTitle('Media del alumno y Graficas')
        etiqueta2 = QLabel()
        etiqueta2.setText("¿Quiere Mostrar la media del alumno?")
        self.checkbox=QCheckBox()


        etiquetapage2=QLabel()
        etiquetapage2.setText("¿Quiere Mostrar una gráfica de notas?")
        self.checkbox2=QCheckBox()
        formlayout.addWidget(etiqueta2)
        formlayout.addWidget(self.checkbox)
        formlayout.addWidget(etiquetapage2)
        formlayout.addWidget(self.checkbox2)
        self.wizard.addPage(self.page2)

        self.finish = self.wizard.button(QWizard.FinishButton)
        self.page2.registerField('miCampo', self.checkbox)
        self.page2.registerField('miCampo2', self.checkbox2)

        id = self.modelo.index(self.fila, 0).data()
        self.nombre = self.modelo.index(self.fila, 1).data()
        self.Apellidos = self.modelo.index(self.fila, 2).data()
        self.cursoId = self.modelo.index(self.fila, 3).data()
        self.media1 = self.modelo.index(self.fila, 4).data()
        self.media2 = self.modelo.index(self.fila, 5).data()
        self.media3 = self.modelo.index(self.fila, 6).data()

        self.page1.setSubTitle(str(id)+","+self.nombre+","+self.Apellidos+","+str(self.cursoId))
        self.wizard.show()
        self.observaciones=self.textarea.toPlainText()

        self.finish.clicked.connect(self.generate)
    except:
        QMessageBox.warning(self,"WARNING","Debes seleccionar un alumno")
```

INFORME

Dibujamos los datos en el pdf, en el primer if marcado incrustaremos la grafica en el pdf y

En el segundo if todos los comentarios del alumno

```
def generate(self):
    outfile = "result.pdf"

    template = PdfReader("template.pdf", decompress=False).pages[0]
    template_obj = pagexobj(template)

    canvas = Canvas(outfile)

    xobj_name = makerl(canvas, template_obj)
    canvas.doForm(xobj_name)

    today = datetime.today()
    canvas.drawString(410, 40, today.strftime('%F'))

    canvas.drawString(150, 535, self.nombre)

    canvas.drawString(350, 535, self.Apellidos)

    canvas.drawString(150, 490, str(self.cursoId))

    if self.page2.field('miCampo')==True:
        canvas.drawString(300, 490,"Nota Media:")
        notamedia=(self.media1+self.media2+self.media3)/3
        canvas.drawString(375, 490, str(notamedia))
    if self.page2.field('miCampo2')==True:
        plt = pg.plot([self.media1,self.media2,self.media3])
        plt.setBackground('w')
        exporter = pg.exporters.ImageExporter(plt.plotItem)

        exporter.parameters()['width'] = 100   # (afecta a la altura de forma proporcional)

        exporter.export('graphic.png')
        canvas.drawImage("graphic.png", 50, 50, width=None,height=None,mask=None)

    self.observaciones=self.textarea.toPlainText()
    comments = self.observaciones.replace('\n', ' ')
    if comments:
        lines = textwrap.wrap(comments, width=40)
        first_line = lines[0]
        remainder = ' '.join(lines[1:])

        lines = textwrap.wrap(remainder, 75)
        lines = lines[:4]

        canvas.drawString(160, 443, first_line)
        for n, l in enumerate(lines, 1):
            canvas.drawString(110, 443 - (n*47), l)

    canvas.save()
    QMessageBox.information(self, "Finalizado", "Se ha generado el PDF")
```

QWEB

Creamos el componente QWeb y le asociamos el resultado del pdf

```
self.web = QWebEngineView(self.tab_2)

self.web.settings().setAttribute(QWebEngineSettings.PluginsEnabled, True)

rutaConPDF = Path("result.pdf")

self.web.load(QUrl(rutaConPDF.absolute().as_uri()))
self.web.setGeometry(0, 0, 1300, 500)
self.web.setSizePolicy(QtWidgets.QSizePolicy.Expanding,QtWidgets.QSizePolicy.Maximum)
self.tabWidget.update()
```