T2-INTERFACES.pdf

bloodyraintatii

Desarrollo de interfaces

1º Desarrollo de Aplicaciones Multiplataforma

Estudios España

Accede al documento original

Desarrollo de Interfaces

Tema 9: Confección de informes II

CONEXIÓN CON FUENTES DE DATOS Y EJECUCIÓN DE CONSULTAS (no?)

→ Para crear informes, es necesario conectarse a una fuente de datos, como una base de datos MySQL.

→ Configuración básica:

- Seleccionar el driver JDBC (com.mysql.jdbc.Driver).
- Especificar la URL de la base de datos (ej: localhost).
- Añadir el nombre de la base de datos y configurar el classpath del driver.

→ Se pueden ejecutar consultas SQL para filtrar y seleccionar datos específicos usando la cláusula WHERE.

INCLUSIÓN DE DATOS DESDE BASE DE DATOS (no?)

→ Los informes pueden incluir datos dinámicos desde una base de datos.

→ Diferencias entre textos estáticos y campos de texto:

- Textos estáticos → etiquetas o títulos fijos.
- Campos de texto (Text Field) → datos dinámicos extraídos de la base de datos.

→ Los campos de texto se colocan en la sección *Details* para mostrar todos los registros.

SUBINFORMES (no?)

→ Los subinformes son informes dentro de otros informes, útiles para organizar datos relacionales.

→ Diferencias entre informes y subinformes:

- Informes: Pueden contener subinformes, tienen encabezado y pie de página, y existen como objetos independientes.
- Subinformes: No pueden contener más subinformes, no tienen encabezado ni pie, y dependen del informe principal.

→ Se pueden crear nuevos subinformes o incluir informes ya existentes.

EDITOR DE EXPRESIONES (no?)

→ Permite personalizar la presentación de los datos en el informe.

→ Se pueden concatenar variables y textos usando el operador +.

```
$V(NOMBRE_VARIABLE1) + " " + $V(NOMBRE_VARIABLE2)
```

INFORMES PARAMETRIZABLES (no?)

→ Los parámetros permiten crear informes dinámicos, donde el usuario puede seleccionar qué datos mostrar.

→ Tipos de parámetros:

- Integrados → disponibles por defecto en el entorno de desarrollo.
- De usuario → definidos por el usuario para operar con campos específicos.

→ Ejemplo de uso: $P{nombre_campo}.

INCLUSIÓN DE IMÁGENES EN UN INFORME (no?)
→ Las imágenes añaden valor visual a los informes.
→ Para incluir una imagen:

1. Seleccionar el elemento *Image*.
2. Especificar el origen de la imagen (ruta local, URL, recurso del proyecto, etc.).

→ Es importante evitar rutas absolutas locales para garantizar la portabilidad del informe.

COLOCAR GRÁFICOS EN UN INFORME
→ Los gráficos son esenciales en los informes para representar visualmente datos que también aparecen en texto o cifras.
→ Para incluir un gráfico en iReport, seleccionamos Chart y el tipo de gráfico:

- Gráficos de barras → muestran datos agrupados con barras verticales u horizontales.
- Gráficos lineales → representan valores en ejes X e Y unidos por líneas, ideales para series temporales o comparativas.
- Gráficos circulares → útiles para mostrar distribuciones, dividiendo el total en segmentos proporcionales.

→ Tras seleccionarlo, podemos personalizarlo usando la opción Chart Wizard (clic derecho sobre el gráfico).

CREACIÓN DEL PRIMER INFORME CON DATOS (CASO PRÁCTICO) (no?)
→ Objetivo → crear un informe básico con datos de productos (nombre y precio) desde una base de datos.
→ Pasos:

1. Conectar con la base de datos *Sample DB*.
2. Ejecutar una consulta SQL para seleccionar los campos deseados.
3. Arrastrar los campos a la zona de diseño y añadir un título con Static Text.
4. Visualizar el informe en la pestaña Preview.

REALIZAR OPERACIONES SOBRE LOS DATOS MOSTRADOS EN UN INFORME (CASO PRÁCTICO) (no?)
→ Objetivo → extender el primer caso, añadiendo una variable que suma los costes de los productos.
→ Pasos:

1. Colocar el campo PRODUCT_COST en la sección *Column Footer*.
2. Seleccionar la operación Suma para calcular el total de los precios.
3. Visualizar el informe con el total de la suma en la parte inferior.

Tema 10: Documentación de aplicaciones: Ayuda

FICHEROS DE AYUDA Y FORMATOS

→ Los ficheros de ayuda contienen información para usuarios acerca de la herramienta desarrollada y pueden estar en múltiples tipos de formatos.
→ Suelen tener dos partes → mapa de fichero (mapa de navegación) y vista de información (índice, glosario, tabla de contenidos o buscador de temas).
→ Formatos comunes:

- HLP → antiguo formato de ayuda de Windows, sustituido por CHM.
- CHM (Ayuda HTML Compilado) → formato HTML compilado, más moderno.
- HPJ → usa ficheros .cnt y .shg para contenido y gráficos, respectivamente.
- IPF (Information Presentation Facility) → usan IPF, un lenguaje muy similar a HTML.
- JavaHelp → formato para aplicaciones Java.

HERRAMIENTAS DE GENERACIÓN DE AYUDA

A. Help Workshop → crea ficheros de ayuda para Windows. Tiene editor de imágenes, administrador de proyectos y compilador.
B. Help Maker → herramienta gratuita para crear ayuda en un solo fichero. Incluye personalización de formato de texto.
C. Shalom Help Maker → herramienta gratuita que permite crear documentos de ayuda con índices y enlaces.

TABLAS DE CONTENIDO

→ Reflejan la estructura de un documento con títulos y subtítulos.
→ Características habituales:

- Pueden mostrar el número de página o no.
- Incluye un enlace directo a cada título y subtítulo.
- Están al inicio.
- Requieren el análisis completo de la documentación a exponer.
- No se debe duplicar información.
- Los títulos deben ser claros.

→ Pautas de diseño:

1. Selección de temas y subtemas.
2. Numeración de temas y subtemas.
3. Creación de la tabla con los enlaces oportunos.
4. Constante revisión.

TIPOS DE DOCUMENTACIÓN

→ Dependiendo del tipo de usuario o fase de la app, puede haber dos tipos de documentación:

- Documentación de pruebas → incluye escenarios de prueba (documentación de entrada) y los resultados (documentación de salida).
- Documentación técnica → incluye los comentarios en el código (documentación interna) y los manuales o guías (documentación externa).

TIPOS DE MANUALES

- Manual de usuario → guía clara y amigable para usuarios.
- Guía rápida → información concreta sobre los procedimientos de una aplicación. Pueden desarrollarse varias en función de la complejidad.
- Guía de referencia → detalles técnicos para usuarios avanzados, con cierto conocimiento y experiencia en el uso de la aplicación.

GENERACIÓN DE UN SISTEMA DE AYUDA CON JAVAHELP (no sé? 6e?)

→ JavaHelp permite crear sistemas de ayuda para aplicaciones Java.
→ Incluye tablas de contenido, motor de búsqueda y glosario.
→ Los pasos son:

1. Diseñar la ayuda.
2. Descargar e instalar JavaHelp.
3. Crear ficheros necesarios de JavaHelp (*map_file.jhm*, *toc.xml*, *indice.xml*, *help_set.hs*).
4. Construir un fichero JAR con todos los ficheros.
5. Integrar la ayuda en la aplicación.

DISEÑO DE UN SISTEMA DE AYUDA CON JAVAHELP (CASO PRÁCTICO 1) (no? 6e)

→ Se implementa un sistema de ayuda con JavaHelp, creando ficheros *map_file.jhm*, *toc.xml* e *index.xml* para organizar la ayuda.

DISEÑO DE UN SISTEMA DE AYUDA CON JAVAHELP (CASO PRÁCTICO 2) (no? 6e)

→ Se crea el fichero *help_set.hs* para vincular los ficheros del caso práctico 1, incluyendo el índice y la tabla de contenidos.
→ Tiene dos partes:

- <maps> → referencia el sistema de mapeo.
- <views> → donde se desarrollan las vistas. Recoge la tabla de contenidos y su índice y el fichero *índice.xml*.

Tema 11: Distribución de aplicaciones

COMPONENTES DE UNA APLICACIÓN

→ Los paquetes incluyen todos los componentes de diseño de una aplicación software.
→ Contienen el código que modela el programa o aplicación pero también lo necesario para desplegar de nuevo la aplicación y que esta funcione en cualquier entorno.
→ Principales componentes:

- Ficheros ejecutables.
- Carpetas de elementos multimedia.
- Bibliotecas y librerías necesarias.

PAQUETES EN LINUX

→ En Linux algunos paquetes deben crearse por comandos.
→ Formatos principales:

- DEB → distribuciones basadas en Debian, como Ubuntu o Kubunt.
- RPM → formato generado por la herramienta *Redhat Package Manager*.
- TGZ → específico de UNIX, son paquetes TAR comprimidos.
- TAR → paquetes sin compresión.

→ Se crean mediante herramientas como *checkinstall*.

PAQUETES EN WINDOWS

→ Mecanismos y formatos de empaquetado en Windows:

- MSI (Microsoft Silent Installer) → formato estándar para paquetes de software que también permite la instalación de su contenido. Incluye ficheros .exe.
- AppX → usado para aplicaciones universales de Windows. Menos frecuente.

EMPAQUETADO DE APLICACIONES JAVA CON ECLIPSE (no? 7d)

→ Permite crear ficheros .jar para aplicaciones Java.
→ Proceso mediante Export > JAR File.
→ Requiere especificar la clase principal (main).

INSTALADORES Y PAQUETES AUTOINSTALABLES

→ Instaladores → herramientas que automatizan el despliegue de una aplicación software.
→ Los más conocidos → InstallBuilder, Windows Installer, MSI Studio.
→ Siguen un algoritmo común que consiste en:

- Comprobación de especificaciones de software y hardware del equipo.
- Verificación de la autenticidad del software.
- Creación de directorios necesarios.
- Extracción de ficheros del paquete.
- Compilación de librerías.
- Definición de variables de entorno.

→ En **Windows** existen dos tipos principales de instaladores:

- EXE → archivo binario ejecutable que ofrece mayor flexibilidad al usuario, permitiendo seleccionar rutas de instalación o elegir qué componentes instalar.
- MSI (Microsoft Silent Installer) → instalación predefinida con menor flexibilidad pero automatización. Ideal para despliegues empresariales.

POV: quedas con alguien

y no se parece a sus fotos.

Si te pasa con un pedido, el seguro de compra online* de tu **Cuenta NoCuenta** de ING te cubre. **¡Y es gratis!**

Saber más

*Compras superiores a 30 €. Más info en ing.es

→ En **Linux**, para paquetes .deb, se utilizan comandos específicos para instalar, desinstalar, limpiar archivos de descarga y eliminar archivos de configuración.
→ Para instalaciones **desde un servidor** web:

- Se pueden alojar paquetes en servidores web, accesibles mediante hipervínculos.
- La herramienta *AptUrl* facilita la descarga e instalación.
- El proceso varía según el tipo de paquete (EXE, ISO, DEB).

CREACIÓN DE UN INSTALADOR EXE
→ Launch4j es una herramienta popular para crear ejecutables EXE a partir de archivos JAR.
→ El proceso detallado incluye:

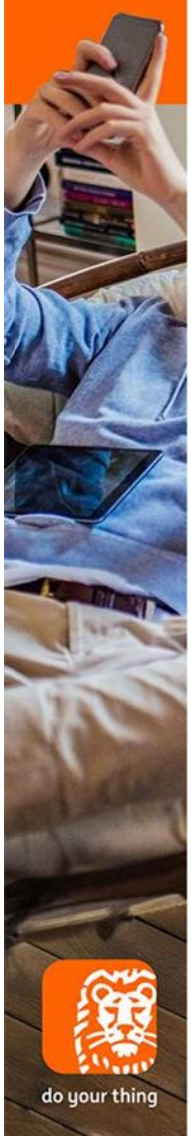
1. Preparación → crear una carpeta de salida (output) y recopilar todos los archivos necesarios: (paquete jar, librerías e imágenes de icono y de splash screen/carga).
2. Configuración en Launch4j → especificar ruta de salida y nombre del ejecutable, seleccionar el jar y el icono, seleccionar modo GUI para que se ejecute desde la interfaz gráfica, establecer versión de JRE requerida y configurar pantalla de splash (opcional).

INTERACCIÓN CON EL USUARIO (no sé?)
→ Se deben tener en cuenta los menús y configuraciones:

- Selección de idioma.
- Aceptación de licencia.
- Selección de componentes de la herramienta.
- Elección de ruta de instalación.
- Indicador de progreso.
- Opciones post-instalación: iniciar la app, reiniciar el sistema, etc.

FICHEROS FIRMADOS DIGITALMENTE (no?)
→ Garantizan la autenticidad del software.
→ Utilizan sistema de clave pública y privada y se envía adicionalmente un certificado en el que el usuario afirma ser el dueño de la clave pública.
→ Se usa la herramienta JarSigner para firmar archivos JAR.

CÓMO CREAR UN ASISTENTE DE INSTALACIÓN. CREACIÓN DE UN INSTALADOR .EXE Y PAQUETE JAR (CASO PRÁCTICO 1) (no? 7e)
ASISTENTE DE INSTALACIÓN. CREACIÓN DE UN ASISTENTE DE INSTALACIÓN (CASO PRÁCTICO 2) (no? 7e)



Tema 15: Desarrollo de interfaces para iOS

DESCARGA E INSTALACIÓN DEL ENTORNO XCODE
→ Xcode → IDE de Apple, necesario para desarrollar en iOS: macOS, watchOS y tvOS.
→ También tendremos que conocer Swift, el lenguaje de programación usado en iOS.
→ Es imprescindible tener un equipo con macOS X para poder desarrollar en iOS.

XCODE
→ Se pueden crear nuevos proyectos, abrir proyectos existentes, crear *playgrounds* (para implementar bloques de código que no son aplicaciones como tal) o descargar proyectos desde repositorios.
→ Creación → "Create a new Xcode project", se elige el sistema operativo (iOS) y una plantilla (por ejemplo, una plantilla en blanco).
→ Configuración → se define el nombre, el equipo de desarrollo (necesario para publicar en la App Store) y el lenguaje de programación (Swift). Se elige la interfaz de usuario: Storyboard (tradicional) o SwiftUI (moderna y recomendada para nuevos desarrollos).

ANÁLISIS DEL ENTORNO (puede ser?)
- Menú de configuración (zona superior de Xcode):
  - La barra superior permite ejecutar y detener la aplicación, seleccionar el simulador o dispositivo físico para pruebas, y acceder a la librería de componentes (botón "+").
  - La librería incluye componentes gráficos, bloques de código, imágenes y colores preconfigurados.
- Zona central de la aplicación:
  - Zona izquierda → muestra el navegador de archivos (*Navigator*), advertencias, análisis de rendimiento y puntos de ruptura.
  - Zona central → muestra el código y una previsualización en tiempo real de la interfaz (*Preview*) al pulsar el botón *Resume*.
  - Zona derecha → permite modificar atributos de los componentes seleccionados (texto, fuente, tamaño, etc.).

CREACIÓN DE LA PRIMERA APLICACIÓN
→ Se accede al archivo *ContentView.swift* para comenzar a programar.
→ Se importa la librería *SwiftUI* y se define la función *ContentView*, que contiene el código de la interfaz.
→ La función *ContentView* define el contenido de la pantalla.

SWIFT. PRIMEROS PASOS
Creación de variables
→ Las variables se declaran con *var* y siguen la convención CamelCase.
→ Las estructuras (*struct*) son equivalentes a las clases en JavaScript y se usan para crear objetos.

struct nuevoDato { * var primeraVariable: String* *}*	*var nuevoDato1 =* *nuevoDato(primeraVariable : "Hola")*

Comentarios
→ Comentarios de una línea → // blablabla
→ Comentarios de varias líneas → /* blablabla */
Botones
→ Los botones se pueden agregar desde la librería o directamente en el código.

```
Button(action: signIn) {
    Text("Sign In")
}
```

PAUTAS PARA LA CREACIÓN DE UNA INTERFAZ EN IOS

→ El desarrollo de las interfaces para aplicaciones en iOS se caracteriza por presentar diseños limpios y minimalistas.
→ Características de diseño:

- Claridad → texto legible, iconos precisos y distribución equilibrada de elementos. Las fuentes y los gráficos resaltan el contenido importante y aportan interactividad.
- Adaptación al contenido → navegación fluida y uso eficiente del espacio.
- Profundidad → uso de capas visuales para crear sensación de movimiento.

→ Criterios de diseño → integridad estética, consistencia, manipulación directa, retroalimentación, metáforas y control del usuario.

COLORES DEL SISTEMA

→ iOS ofrece una paleta de colores que se adapta a los modos claro y oscuro y ofrece accesibilidad.
→ Pilares clave para la elección del color:

- Usar colores con prudencia para resaltar elementos importantes.
- Usar colores complementarios.
- Seleccionar una paleta concreta y limitada.
- Escoger un color característico para toda la aplicación.
- Proporcionar colores que funcionen bien en ambos modos (claro y oscuro).

CREACIÓN DE UN PROYECTO DESDE CERO CON XCODE (CASO PRÁCTICO)

→ Se importa *SwiftUI*.
→ Se define la función ContentView con un componente de texto.
→ Se implementa el código. Ejemplo:

```
import SwiftUI
struct ContentView: View {
    var body: some View {
        Text("Soy tu primera aplicación")
            .padding()
    }
}
```

DISEÑA UNA INTERFAZ PARA UNA APLICACIÓN iOS (CASO PRÁCTICO)

→ Usar herramientas de prototipado para diseñar la interfaz.
→ Seguir las pautas de diseño de Apple para garantizar claridad, consistencia y usabilidad.

CRITERIOS

Tema 9

5e) Se han incluido gráficos generados a partir de los datos.

Tema 10

6a) Se han identificado sistemas de generación de ayudas.

6b) Se han generado ayudas en los formatos habituales.

Tema 11

7a) Se han empaquetado los componentes que requiere la aplicación.

7b) Se ha personalizado el asistente de instalación

7c) Se ha empaquetado la aplicación para ser instalada de forma típica, completa o personalizada.

Tema 15

4a) Se han creado menús que se ajustan a los estándares.

4b) Se han creado menús contextuales cuya estructura y contenido siguen los estándares establecidos.

4c) Se han distribuido las acciones en menús, barras de herramientas, botones de comando, entre otros, siguiendo un criterio coherente.

4d) Se han distribuido adecuadamente los controles en la interfaz de usuario.

4e) Se ha utilizado el tipo de control más apropiado en cada caso.

4f) Se ha diseñado el aspecto de la interfaz de usuario (colores y fuentes entre otros) atendiendo a su legibilidad.

4g) Se ha verificado que los mensajes generados por la aplicación son adecuados en extensión y claridad.