**Guía de Instalación Rápida de IBM SPSS Statistics**

**27**

Procedimiento Simple

15 de octubre de 2025

# Resumen

Este documento proporciona una guía rápida y simplificada para la instalación de IBM SPSS Statistics 27 en Windows. Se enfoca en el proceso de descompresión con la contraseña proporcionada (soft98) y los pasos principales del asistente de instalación hasta el primer inicio del programa.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.  
  
El layout es el que se encarga de como se vea el entorno de la aplicación, o sea el menú de al lado, a como se puede apreciar este después de inicar sesión sale el nombre del usuario y pues hace diferente función dependiendo el rol del usuario  
  
La pagina Home es la que se encarga de redirigir el botón de login o registrar usuario

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

En registrar usuario se hace uso de una api que hace conexión entre la base de datos y el Backend, este hace POST que inserta el usuario que ingresa el usuario que se esta registrando desde el frontend, este tiene la función de seleccionar el rol que tiene el usuario (Profesor o estudiante), este se guarda en la base de datos para asi guardar el usuario y después poder iniciar sesión

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

En login, este se encarga de iniciar sesión, a como se aprecia se utiliza una api en el cual hace GET y POST que hace que inserta el usuario y busca en la base de datos, si existe inicia sesión y me redirige a la pagina HOME (dependiendo el rol de usuario), pero si no existe este no inicia sesión.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

La paginas admin, profesor o estudiante, esta muestra las distintas páginas que hace función de la aplicación de encuestas, solo hacer click a los botones y te dirige a la pagina que seleccionas, esto es dependiendo del rol del usuario pues muestra diferentes elementos, por ejemplo en la imagen se muestra la pagina admin.

Pagina Admin:  
  
Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Pagina Profesor:

Texto

El contenido generado por IA puede ser incorrecto.

Pagina estudiante:  
  
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

<h3>Respuestas</h3>

@if (encuestas is null)

{

<p>Cargando...</p>

}

else if (encuestas.Count == 0)

{

<p>No hay respuestas.</p>

}

else

{

<table class="table table-striped">

<thead>

<tr>

<th>#</th>

<th>Nombre</th>

<th>Apellido</th>

<th>Sexo</th>

<th>Departamento</th>

<th>Ciudad</th>

<th>Facultad</th>

<th>Carrera</th>

<th>Matricula</th>

<th>Becado</th>

<th>XII</th>

<th>XIII</th>

<th>XIV</th>

<th>XV</th>

<th>XVI</th>

<th>XVII</th>

</tr>

</thead>

<tbody>

@foreach (var r in encuestas)

{

<tr>

<td>@r.Numero</td>

<td>@r.Nombre</td>

<td>@r.Apellido</td>

<td>@r.Sexo</td>

<td>@r.Departamento</td>

<td>@r.Ciudad</td>

<td>@r.Facultad</td>

<td>@r.Carrera</td>

<td>@r.Matricula</td>

<td>@r.Becado</td>

<td>@r.Xii</td>

<td>@r.Xiii</td>

<td>@r.Xiv</td>

<td>@r.Xv</td>

<td>@r.Xvi</td>

<td>@r.Xvii</td>

</tr>

}

</tbody>

</table>

}

@code {

private List<RespuestaDto>? encuestas;

protected override async Task OnInitializedAsync()

{

// Llamando al controller correcto

encuestas = await Http.GetFromJsonAsync<List<RespuestaDto>>("api/Respuestas");

}

// DTO para mostrar solo los valores string de las relaciones

public class RespuestaDto

{

public int Numero { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public string Sexo { get; set; }

public string Departamento { get; set; }

public string Ciudad { get; set; }

public string Facultad { get; set; }

public string Carrera { get; set; }

public string Matricula { get; set; }

public string Becado { get; set; }

public string Xii { get; set; }

public string Xiii { get; set; }

public string Xiv { get; set; }

public string Xv { get; set; }

public string Xvi { get; set; }

public string Xvii { get; set; }

}

}

La pagina respuestas se encarga de ver las respuestas de la encuestas que llenaron, utiliza una api en el backend que utiliza GET y POST en la cual se encarga de buscar las respuestas de la tabla respuesta de la base de datos y asi mostrarlo en el frontend

// GET: api/Respuestas

[HttpGet]

public async Task<ActionResult<IEnumerable<object>>> GetRespuestas()

{

var respuestas = await \_context.respuestas

.Include(r => r.Sexo)

.Include(r => r.Departamento)

.Include(r => r.Ciudad)

.Include(r => r.Facultad)

.Include(r => r.Carrera)

.Include(r => r.Matricula)

.Include(r => r.Becado)

.Include(r => r.Xii)

.Include(r => r.Xiii)

.Include(r => r.Xiv)

.Include(r => r.Xv)

.Include(r => r.Xvi)

.Include(r => r.Xvii)

.Select(r => new

{

r.Numero,

r.Nombre,

r.Apellido,

Sexo = r.Sexo != null ? r.Sexo.Valor : null,

r.Identidad,

Departamento = r.Departamento != null ? r.Departamento.Valor : null,

Ciudad = r.Ciudad != null ? r.Ciudad.Valor : null,

Facultad = r.Facultad != null ? r.Facultad.Valor : null,

Carrera = r.Carrera != null ? r.Carrera.Valor : null,

r.PreguntaIX,

Matricula = r.Matricula != null ? r.Matricula.Valor : null,

Becado = r.Becado != null ? r.Becado.Valor : null,

Xii = r.Xii != null ? r.Xii.Valor : null,

Xiii = r.Xiii != null ? r.Xiii.Valor : null,

Xiv = r.Xiv != null ? r.Xiv.Valor : null,

Xv = r.Xv != null ? r.Xv.Valor : null,

Xvi = r.Xvi != null ? r.Xvi.Valor : null,

Xvii = r.Xvii != null ? r.Xvii.Valor : null

})

.ToListAsync();

return Ok(respuestas);

}  
  
// POST: api/Respuestas

[HttpPost]

public async Task<ActionResult<Respuesta>> PostRespuesta(Respuesta respuesta)

{

if (respuesta == null)

return BadRequest("No se recibió ninguna respuesta.");

try

{

\_context.respuestas.Add(respuesta);

await \_context.SaveChangesAsync();

// Devuelve la respuesta creada con su ID generado

return CreatedAtAction(nameof(GetRespuestas), new { id = respuesta.Numero }, respuesta);

}

catch (Exception ex)

{

return StatusCode(500, "Error guardando la encuesta: " + ex.Message);

}

}

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.  
<h3>Llenar Encuesta</h3>

<**EditForm** **Model**="@respuesta" **OnValidSubmit**="GuardarEncuesta">

<**DataAnnotationsValidator** />

<**ValidationSummary** />

<!-- Nombre -->

<div>

<label>Nombre:</label>

<**InputText** @bind-Value="respuesta.Nombre" class="form-control" />

</div>

<!-- Apellido -->

<div>

<label>Apellido:</label>

<**InputText** @bind-Value="respuesta.Apellido" class="form-control" />

</div>

<!-- Sexo -->

<div>

<label>Sexo:</label>

<**InputSelect** @bind-Value="respuesta.SexoClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in sexos)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Identidad -->

<div>

<label>Identidad:</label>

<**InputText** @bind-Value="respuesta.Identidad" class="form-control" />

</div>

<!-- Departamento -->

<div>

<label>Departamento:</label>

<**InputSelect** @bind-Value="respuesta.DepartamentoClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in departamentos)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Ciudad -->

<div>

<label>Ciudad:</label>

<**InputSelect** @bind-Value="respuesta.CiudadClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in ciudades)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Facultad -->

<div>

<label>Facultad:</label>

<**InputSelect** @bind-Value="respuesta.FacultadClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in facultades)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Carrera -->

<div>

<label>Carrera:</label>

<**InputSelect** @bind-Value="respuesta.CarreraClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in carreras)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Año de estudios -->

<div>

<label>Año de estudios:</label>

<**InputNumber** @bind-Value="respuesta.PreguntaIX" class="form-control" />

</div>

<!-- Matrícula -->

<div>

<label>Matrícula:</label>

<**InputSelect** @bind-Value="respuesta.MatriculaClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in matriculas)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Becado -->

<div>

<label>Becado:</label>

<**InputSelect** @bind-Value="respuesta.BecadoClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in becados)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XII -->

<div>

<label>I. Usted ha usado el Sistema de Matrícula en línea?</label>

<**InputSelect** @bind-Value="respuesta.XiiClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xii)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XIII -->

<div>

<label>II. Qué le parece el nuevo sistema de matrícula en línea?</label>

<**InputSelect** @bind-Value="respuesta.XiiiClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xiii)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XIV -->

<div>

<label>III. Qué tan accesible está el nuevo sistema?</label>

<**InputSelect** @bind-Value="respuesta.XivClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xiv)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XV -->

<div>

<label>IV. De qué manera le ha ayudado el nuevo sistema?</label>

<**InputSelect** @bind-Value="respuesta.XvClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xv)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XVI -->

<div>

<label>V. Cuánto dinero ha ahorrado usar el nuevo sistema?</label>

<**InputSelect** @bind-Value="respuesta.XviClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xvi)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XVII -->

<div>

<label>VI. Qué recomienda para mejorar el sistema?</label>

<**InputSelect** @bind-Value="respuesta.XviiClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xvii)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<button type="submit" class="btn btn-primary mt-3">Guardar</button>

</**EditForm**>

@code {

private Respuesta respuesta = new Respuesta

{

SexoClave = 0,

DepartamentoClave = 0,

CiudadClave = 0,

FacultadClave = 0,

CarreraClave = 0,

MatriculaClave = 0,

BecadoClave = 0,

XiiClave = 0,

XiiiClave = 0,

XivClave = 0,

XvClave = 0,

XviClave = 0,

XviiClave = 0

};

private List<iii\_sexo> sexos = new();

private List<v\_departamento> departamentos = new();

private List<vi\_ciudad> ciudades = new();

private List<vii\_facultad> facultades = new();

private List<viii\_carrera> carreras = new();

private List<x\_matricula> matriculas = new();

private List<xi\_becado> becados = new();

private List<xii> xii = new();

private List<xiii> xiii = new();

private List<xiv> xiv = new();

private List<xv> xv = new();

private List<xvi> xvi = new();

private List<xvii> xvii = new();

protected override async Task OnInitializedAsync()

{

sexos = await \_context.sexos.ToListAsync();

departamentos = await \_context.departamentos.ToListAsync();

ciudades = await \_context.ciudades.ToListAsync();

facultades = await \_context.facultades.ToListAsync();

carreras = await \_context.carreras.ToListAsync();

matriculas = await \_context.matriculas.ToListAsync();

becados = await \_context.becados.ToListAsync();

xii = await \_context.xiis.ToListAsync();

xiii = await \_context.xiiis.ToListAsync();

xiv = await \_context.xivs.ToListAsync();

xv = await \_context.xvs.ToListAsync();

xvi = await \_context.xvis.ToListAsync();

xvii = await \_context.xviis.ToListAsync();

}

private async Task GuardarEncuesta()

{

try

{

\_context.respuestas.Add(respuesta);

await \_context.SaveChangesAsync();

await JS.InvokeVoidAsync("alert", "Encuesta guardada correctamente!");

Navigation.NavigateTo("/llenarencuesta");

}

catch (DbUpdateException dbEx)

{

var innerMessage = dbEx.InnerException?.Message ?? dbEx.Message;

await JS.InvokeVoidAsync("alert", "Error guardando la encuesta: " + innerMessage);

}

}

}

La pagina llenar encuesta se encarga de insertar datos de una encuesta y asi guardarlo en la base de datos mediante una api que hace conexión desde la base de datos y el backend y asi despues mostrarlo en el frontend con la pagina respuestas.

[HttpGet]

public async Task<ActionResult<List<Respuesta>>> GetEncuestas()

{

var encuestas = await \_context.respuestas

.Include(r => r.Sexo)

.Include(r => r.Departamento)

.Include(r => r.Ciudad)

.Include(r => r.Facultad)

.Include(r => r.Carrera)

.Include(r => r.Matricula)

.Include(r => r.Becado)

.Include(r => r.Xii)

.Include(r => r.Xiii)

.Include(r => r.Xiv)

.Include(r => r.Xv)

.Include(r => r.Xvi)

.Include(r => r.Xvii)

.ToListAsync();

return Ok(encuestas);

}

// POST: api/Encuesta

[HttpPost]

public async Task<ActionResult<Respuesta>> PostEncuesta(Respuesta respuesta)

{

if (respuesta == null)

return BadRequest("No se recibió ninguna respuesta.");

try

{

\_context.respuestas.Add(respuesta);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetEncuestas), new { id = respuesta.Numero }, respuesta);

}

catch (Exception ex)

{

return StatusCode(500, "Error guardando la encuesta: " + ex.Message);

}

}

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Eliminar encuesta o eliminar respuestas, este busca las encuesta como la página respuestas con GET y tiene la función de Eliminar cualquier encuesta hecha mediante una api que hace la conexión de la base de datos y el backend haciendo DELETE.

<h3>Respuestas</h3>

@if (encuestas is null)

{

<p>Cargando...</p>

}

else if (encuestas.Count == 0)

{

<p>No hay respuestas.</p>

}

else

{

<table class="table table-striped">

<thead>

<tr>

<th>#</th>

<th>Nombre</th>

<th>Apellido</th>

<th>Sexo</th>

<th>Departamento</th>

<th>Ciudad</th>

<th>Facultad</th>

<th>Carrera</th>

<th>Matricula</th>

<th>Becado</th>

<th>XII</th>

<th>XIII</th>

<th>XIV</th>

<th>XV</th>

<th>XVI</th>

<th>XVII</th>

<th>Acciones</th> @\* 👈 Nueva columna \*@

</tr>

</thead>

<tbody>

@foreach (var r in encuestas)

{

<tr>

<td>@r.Numero</td>

<td>@r.Nombre</td>

<td>@r.Apellido</td>

<td>@r.Sexo</td>

<td>@r.Departamento</td>

<td>@r.Ciudad</td>

<td>@r.Facultad</td>

<td>@r.Carrera</td>

<td>@r.Matricula</td>

<td>@r.Becado</td>

<td>@r.Xii</td>

<td>@r.Xiii</td>

<td>@r.Xiv</td>

<td>@r.Xv</td>

<td>@r.Xvi</td>

<td>@r.Xvii</td>

<td>

<button class="btn btn-danger btn-sm" @onclick="() => DeleteRespuesta(r.Numero)">

Eliminar

</button>

</td>

</tr>

}

</tbody>

</table>

}

@code {

async Task DeleteRespuesta(int id)

{

bool confirmar = await JS.InvokeAsync<bool>("confirm", $"¿Seguro que quieres eliminar la respuesta {id}?");

if (confirmar)

{

var response = await Http.DeleteAsync($"api/Respuestas/{id}");

if (response.IsSuccessStatusCode)

{

// Solo quitar la fila de la lista local

var item = encuestas.FirstOrDefault(e => e.Numero == id);

if (item != null)

{

encuestas.Remove(item);

StateHasChanged();

}

}

else if (response.StatusCode == System.Net.HttpStatusCode.NotFound)

{

await JS.InvokeVoidAsync("alert", $"No se encontró la respuesta con ID {id}.");

}

else

{

await JS.InvokeVoidAsync("alert", "Error al eliminar la respuesta.");

}

}

}

private List<Respuesta>? encuestas;

protected override async Task OnInitializedAsync()

{

// Llamando al controller correcto

encuestas = await Http.GetFromJsonAsync<List<Respuesta>>("api/Respuestas");

}

// DTO para mostrar solo los valores string de las relaciones

public class Respuesta

{

public int Numero { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public string Sexo { get; set; }

public string Departamento { get; set; }

public string Ciudad { get; set; }

public string Facultad { get; set; }

public string Carrera { get; set; }

public string Matricula { get; set; }

public string Becado { get; set; }

public string Xii { get; set; }

public string Xiii { get; set; }

public string Xiv { get; set; }

public string Xv { get; set; }

public string Xvi { get; set; }

public string Xvii { get; set; }

}

}

API

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteRespuesta(int id)

{

var respuesta = await \_context.respuestas.FindAsync(id);

if (respuesta == null)

{

return NotFound();

}

\_context.respuestas.Remove(respuesta);

await \_context.SaveChangesAsync();

return NoContent(); // 204

}

Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Editar encuesta o editar respuestas, este se encarga de editar las respuestas ya hechas en llenar encuesta, este buscar u obtiene las respuestas de lo base de datos con la api del backend con GET para mostrarlo en el frontend y actualizarlo mediante PUT.

<h3>Editar Encuesta</h3>

@if (respuesta is null)

{

<p>Cargando...</p>

}

else

{

<**EditForm** **Model**="@respuesta" **OnValidSubmit**="ActualizarEncuesta">

<**DataAnnotationsValidator** />

<**ValidationSummary** />

<!-- Nombre -->

<div>

<label>Nombre:</label>

<**InputText** @bind-Value="respuesta.Nombre" class="form-control" />

</div>

<!-- Apellido -->

<div>

<label>Apellido:</label>

<**InputText** @bind-Value="respuesta.Apellido" class="form-control" />

</div>

<!-- Sexo -->

<div>

<label>Sexo:</label>

<**InputSelect** @bind-Value="respuesta.SexoClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in sexos)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Identidad -->

<div>

<label>Identidad:</label>

<**InputText** @bind-Value="respuesta.Identidad" class="form-control" />

</div>

<!-- Departamento -->

<div>

<label>Departamento:</label>

<**InputSelect** @bind-Value="respuesta.DepartamentoClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in departamentos)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Ciudad -->

<div>

<label>Ciudad:</label>

<**InputSelect** @bind-Value="respuesta.CiudadClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in ciudades)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Facultad -->

<div>

<label>Facultad:</label>

<**InputSelect** @bind-Value="respuesta.FacultadClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in facultades)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Carrera -->

<div>

<label>Carrera:</label>

<**InputSelect** @bind-Value="respuesta.CarreraClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in carreras)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Año de estudios -->

<div>

<label>Año de estudios:</label>

<**InputNumber** @bind-Value="respuesta.PreguntaIX" class="form-control" />

</div>

<!-- Matrícula -->

<div>

<label>Matrícula:</label>

<**InputSelect** @bind-Value="respuesta.MatriculaClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in matriculas)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- Becado -->

<div>

<label>Becado:</label>

<**InputSelect** @bind-Value="respuesta.BecadoClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in becados)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XII -->

<div>

<label>I. Usted ha usado el Sistema de Matrícula en línea?</label>

<**InputSelect** @bind-Value="respuesta.XiiClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xii)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XIII -->

<div>

<label>II. Qué le parece el nuevo sistema de matrícula en línea?</label>

<**InputSelect** @bind-Value="respuesta.XiiiClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xiii)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XIV -->

<div>

<label>III. Qué tan accesible está el nuevo sistema?</label>

<**InputSelect** @bind-Value="respuesta.XivClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xiv)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XV -->

<div>

<label>IV. De qué manera le ha ayudado el nuevo sistema?</label>

<**InputSelect** @bind-Value="respuesta.XvClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xv)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XVI -->

<div>

<label>V. Cuánto dinero ha ahorrado usar el nuevo sistema?</label>

<**InputSelect** @bind-Value="respuesta.XviClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xvi)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<!-- XVII -->

<div>

<label>VI. Qué recomienda para mejorar el sistema?</label>

<**InputSelect** @bind-Value="respuesta.XviiClave" class="form-control">

<option value="0">Seleccione...</option>

@foreach (var item in xvii)

{

<option value="@item.Clave">@item.Valor</option>

}

</**InputSelect**>

</div>

<button type="submit" class="btn btn-success mt-3">Actualizar</button>

<**Component** />

</**EditForm**>

}

@code {

[Parameter] public int id { get; set; }

private Respuesta respuesta;

private List<iii\_sexo> sexos = new();

private List<v\_departamento> departamentos = new();

private List<vi\_ciudad> ciudades = new();

private List<vii\_facultad> facultades = new();

private List<viii\_carrera> carreras = new();

private List<x\_matricula> matriculas = new();

private List<xi\_becado> becados = new();

private List<xii> xii = new();

private List<xiii> xiii = new();

private List<xiv> xiv = new();

private List<xv> xv = new();

private List<xvi> xvi = new();

private List<xvii> xvii = new();

protected override async Task OnInitializedAsync()

{

// Cargar listas para los Selects

sexos = await \_context.sexos.ToListAsync();

departamentos = await \_context.departamentos.ToListAsync();

ciudades = await \_context.ciudades.ToListAsync();

facultades = await \_context.facultades.ToListAsync();

carreras = await \_context.carreras.ToListAsync();

matriculas = await \_context.matriculas.ToListAsync();

becados = await \_context.becados.ToListAsync();

xii = await \_context.xiis.ToListAsync();

xiii = await \_context.xiiis.ToListAsync();

xiv = await \_context.xivs.ToListAsync();

xv = await \_context.xvs.ToListAsync();

xvi = await \_context.xvis.ToListAsync();

xvii = await \_context.xviis.ToListAsync();

// Cargar datos existentes de la encuesta

respuesta = await \_context.respuestas.FindAsync(id);

}

private async Task ActualizarEncuesta()

{

try

{

\_context.Entry(respuesta).State = EntityState.Modified;

await \_context.SaveChangesAsync();

await JS.InvokeVoidAsync("alert", "Encuesta actualizada correctamente!");

Navigation.NavigateTo("/respuestas");

}

catch (DbUpdateException dbEx)

{

var innerMessage = dbEx.InnerException?.Message ?? dbEx.Message;

await JS.InvokeVoidAsync("alert", "Error actualizando la encuesta: " + innerMessage);

}

}

}

API:

// PUT: api/Respuestas/5

[HttpPut("{id}")]

public async Task<IActionResult> PutRespuesta(int id, Respuesta respuesta)

{

if (id != respuesta.Numero)

{

return BadRequest("El id no coincide con el registro.");

}

\_context.Entry(respuesta).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.respuestas.Any(e => e.Numero == id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent(); // 204

}

Interfaz de usuario gráfica, Texto, Correo electrónico

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

La pagina exportar, esta exporta las encuestas hechas con las opciones de exportar solo una encuesta que el usuario selecciona y si quiere en PDF o EXCEl o si prefiere exportarla todas en PDF o EXCEL, esta función se hace mediante la api para hacer la conexión entre la base de datos y el backend y asi obtener las encuestas (GET), y mostrarlo en el frontend para asi seleccionar cual quiere exportar

<h3>Respuestas</h3>

@if (encuestas is null)

{

<p>Cargando...</p>

}

else if (encuestas.Count == 0)

{

<p>No hay respuestas.</p>

}

else

{

<table class="table table-striped">

<thead>

<tr>

<th>#</th>

<th>Nombre</th>

<th>Apellido</th>

<th>Sexo</th>

<th>Departamento</th>

<th>Ciudad</th>

<th>Facultad</th>

<th>Carrera</th>

<th>Matricula</th>

<th>Becado</th>

<th>XII</th>

<th>XIII</th>

<th>XIV</th>

<th>XV</th>

<th>XVI</th>

<th>XVII</th>

<th>Acciones</th> @\* 👈 Nueva columna \*@

</tr>

</thead>

<tbody>

@foreach (var r in encuestas)

{

<tr>

<td>@r.Numero</td>

<td>@r.Nombre</td>

<td>@r.Apellido</td>

<td>@r.Sexo</td>

<td>@r.Departamento</td>

<td>@r.Ciudad</td>

<td>@r.Facultad</td>

<td>@r.Carrera</td>

<td>@r.Matricula</td>

<td>@r.Becado</td>

<td>@r.Xii</td>

<td>@r.Xiii</td>

<td>@r.Xiv</td>

<td>@r.Xv</td>

<td>@r.Xvi</td>

<td>@r.Xvii</td>

<td>

<button class="btn btn-danger btn-sm" @onclick="() => ExportarPDF(r.Numero)">PDF</button>

<button class="btn btn-success" @onclick="() => ExportarExcel(r.Numero)">EXCEL</button>

</td>

</tr>

}

</tbody>

</table>

<hr />

<h5>Exportar todas las encuestas:</h5>

<button class="btn btn-danger me-2" @onclick="ExportarTodasPDF">Exportar todas a PDF</button>

<button class="btn btn-success" @onclick="ExportarTodasExcel">Exportar todas a Excel</button>

}

@code {

private List<Respuesta>? encuestas;

protected override async Task OnInitializedAsync()

{

// Llamando al controller correcto

encuestas = await Http.GetFromJsonAsync<List<Respuesta>>("api/Respuestas");

}

// DTO para mostrar solo los valores string de las relaciones

public class Respuesta

{

public int Numero { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public string Sexo { get; set; }

public string Departamento { get; set; }

public string Ciudad { get; set; }

public string Facultad { get; set; }

public string Carrera { get; set; }

public string Matricula { get; set; }

public string Becado { get; set; }

public string Xii { get; set; }

public string Xiii { get; set; }

public string Xiv { get; set; }

public string Xv { get; set; }

public string Xvi { get; set; }

public string Xvii { get; set; }

}

// Exportar individual

private async Task ExportarPDF(int id)

{

await JS.InvokeVoidAsync("open", $"/api/Respuestas/exportarpdf/{id}", "\_blank");

}

private async Task ExportarExcel(int id)

{

await JS.InvokeVoidAsync("open", $"/api/Respuestas/exportarexcel/{id}", "\_blank");

}

// Exportar todas

private async Task ExportarTodasPDF()

{

await JS.InvokeVoidAsync("open", "/api/Respuestas/exportartodospdf", "\_blank");

}

private async Task ExportarTodasExcel()

{

await JS.InvokeVoidAsync("open", "/api/Respuestas/exportartodosexcel", "\_blank");

}

}

API:

[HttpGet("exportarpdf/{id}")]

public async Task<IActionResult> ExportarEncuestaPDF(int id)

{

var r = await \_context.respuestas

.Include(x => x.Sexo)

.Include(x => x.Departamento)

.Include(x => x.Ciudad)

.Include(x => x.Facultad)

.Include(x => x.Carrera)

.Include(x => x.Matricula)

.Include(x => x.Becado)

.Include(x => x.Xii)

.Include(x => x.Xiii)

.Include(x => x.Xiv)

.Include(x => x.Xv)

.Include(x => x.Xvi)

.Include(x => x.Xvii)

.FirstOrDefaultAsync(x => x.Numero == id);

if (r == null) return NotFound();

using var stream = new MemoryStream();

Document.Create(container =>

{

container.Page(page =>

{

page.Margin(20);

page.Header().Text("Encuesta de Análisis de Sistemas de Información").Bold().FontSize(16);

page.Content().Column(col =>

{

col.Item().Text("Datos Generales:").Bold();

col.Item().Text($"Nombres: {r.Nombre}");

col.Item().Text($"Apellidos: {r.Apellido}");

col.Item().Text($"Sexo: {r.Sexo?.Valor}");

col.Item().Text($"Cédula de Identificación: {r.Identidad}");

col.Item().Text($"Departamento: {r.Departamento?.Valor}");

col.Item().Text($"Ciudad: {r.Ciudad?.Valor}");

col.Item().Text("Datos Académicos:").Bold();

col.Item().Text($"Facultad: {r.Facultad?.Valor}");

col.Item().Text($"Carrera: {r.Carrera?.Valor}");

col.Item().Text($"Año de estudios: {r.PreguntaIX}");

col.Item().Text($"Tipo de Matrícula: {r.Matricula?.Valor}");

col.Item().Text($"Becado: {r.Becado?.Valor}");

col.Item().Text("Desarrollo:").Bold();

col.Item().Text($"I. Uso del Sistema de Matrícula: {r.Xii?.Valor}");

col.Item().Text($"II. Opinión del nuevo sistema: {r.Xiii?.Valor}");

col.Item().Text($"III. Accesibilidad del sistema: {r.Xiv?.Valor}");

col.Item().Text($"IV. Cómo le ha ayudado: {r.Xv?.Valor}");

col.Item().Text($"V. Ahorro de dinero: {r.Xvi?.Valor}");

col.Item().Text($"VI. Recomendaciones de mejora: {r.Xvii?.Valor}");

});

});

}).GeneratePdf(stream);

return File(stream.ToArray(), "application/pdf", $"Encuesta\_{id}.pdf");

}

[HttpGet("exportartodospdf")]

public async Task<IActionResult> ExportarTodasPDFFormatoIndividual()

{

var todas = await \_context.respuestas

.Include(x => x.Sexo)

.Include(x => x.Departamento)

.Include(x => x.Ciudad)

.Include(x => x.Facultad)

.Include(x => x.Carrera)

.Include(x => x.Matricula)

.Include(x => x.Becado)

.Include(x => x.Xii)

.Include(x => x.Xiii)

.Include(x => x.Xiv)

.Include(x => x.Xv)

.Include(x => x.Xvi)

.Include(x => x.Xvii)

.ToListAsync();

using var stream = new MemoryStream();

Document.Create(container =>

{

container.Page(page =>

{

page.Margin(20);

page.Size(PageSizes.A4);

page.Content().Column(col =>

{

foreach (var r in todas)

{

col.Item().Text("Datos Generales:").Bold();

col.Item().Text($"Nombres: {r.Nombre}");

col.Item().Text($"Apellidos: {r.Apellido}");

col.Item().Text($"Sexo: {r.Sexo?.Valor}");

col.Item().Text($"Cédula de Identificación: {r.Identidad}");

col.Item().Text($"Departamento: {r.Departamento?.Valor}");

col.Item().Text($"Ciudad: {r.Ciudad?.Valor}");

col.Item().Text("Datos Académicos:").Bold();

col.Item().Text($"Facultad: {r.Facultad?.Valor}");

col.Item().Text($"Carrera: {r.Carrera?.Valor}");

col.Item().Text($"Año de estudios: {r.PreguntaIX}");

col.Item().Text($"Tipo de Matrícula: {r.Matricula?.Valor}");

col.Item().Text($"Becado: {r.Becado?.Valor}");

col.Item().Text("Desarrollo:").Bold();

col.Item().Text($"I. Uso del Sistema de Matrícula: {r.Xii?.Valor}");

col.Item().Text($"II. Opinión del nuevo sistema: {r.Xiii?.Valor}");

col.Item().Text($"III. Accesibilidad del sistema: {r.Xiv?.Valor}");

col.Item().Text($"IV. Cómo le ha ayudado: {r.Xv?.Valor}");

col.Item().Text($"V. Ahorro de dinero: {r.Xvi?.Valor}");

col.Item().Text($"VI. Recomendaciones de mejora: {r.Xvii?.Valor}");

// Salto de página para la siguiente encuesta

col.Item().PageBreak();

}

});

});

}).GeneratePdf(stream);

return File(stream.ToArray(), "application/pdf", "Todas\_Encuestas\_Formato\_Individual.pdf");

}

[HttpGet("exportarexcel/{id}")]

public async Task<IActionResult> ExportarExcel(int id)

{

var r = await \_context.respuestas

.Include(x => x.Sexo)

.Include(x => x.Departamento)

.Include(x => x.Ciudad)

.Include(x => x.Facultad)

.Include(x => x.Carrera)

.Include(x => x.Matricula)

.Include(x => x.Becado)

.Include(x => x.Xii)

.Include(x => x.Xiii)

.Include(x => x.Xiv)

.Include(x => x.Xv)

.Include(x => x.Xvi)

.Include(x => x.Xvii)

.FirstOrDefaultAsync(x => x.Numero == id);

if (r == null) return NotFound();

using var workbook = new XLWorkbook();

var ws = workbook.Worksheets.Add("Encuesta");

int row = 1;

ws.Cell(row, 1).Value = "Encuesta de Análisis de Sistemas de Información";

ws.Cell(row, 1).Style.Font.Bold = true;

ws.Range(row, 1, row, 2).Merge();

row += 2;

ws.Cell(row, 1).Value = "Datos Generales";

ws.Cell(row, 1).Style.Font.Bold = true;

row++;

ws.Cell(row, 1).Value = "Nombres";

ws.Cell(row, 2).Value = r.Nombre;

row++;

ws.Cell(row, 1).Value = "Apellidos";

ws.Cell(row, 2).Value = r.Apellido;

row++;

ws.Cell(row, 1).Value = "Sexo";

ws.Cell(row, 2).Value = r.Sexo?.Valor;

row++;

ws.Cell(row, 1).Value = "Cédula de Identificación";

ws.Cell(row, 2).Value = r.Identidad;

row++;

ws.Cell(row, 1).Value = "Departamento";

ws.Cell(row, 2).Value = r.Departamento?.Valor;

row++;

ws.Cell(row, 1).Value = "Ciudad";

ws.Cell(row, 2).Value = r.Ciudad?.Valor;

row += 2;

ws.Cell(row, 1).Value = "Datos Académicos";

ws.Cell(row, 1).Style.Font.Bold = true;

row++;

ws.Cell(row, 1).Value = "Facultad";

ws.Cell(row, 2).Value = r.Facultad?.Valor;

row++;

ws.Cell(row, 1).Value = "Carrera";

ws.Cell(row, 2).Value = r.Carrera?.Valor;

row++;

ws.Cell(row, 1).Value = "Año de estudios";

ws.Cell(row, 2).Value = r.PreguntaIX;

row++;

ws.Cell(row, 1).Value = "Tipo de Matrícula";

ws.Cell(row, 2).Value = r.Matricula?.Valor;

row++;

ws.Cell(row, 1).Value = "Becado";

ws.Cell(row, 2).Value = r.Becado?.Valor;

row += 2;

ws.Cell(row, 1).Value = "Desarrollo";

ws.Cell(row, 1).Style.Font.Bold = true;

row++;

ws.Cell(row, 1).Value = "I. Uso del Sistema de Matrícula";

ws.Cell(row, 2).Value = r.Xii?.Valor;

row++;

ws.Cell(row, 1).Value = "II. Opinión del nuevo sistema";

ws.Cell(row, 2).Value = r.Xiii?.Valor;

row++;

ws.Cell(row, 1).Value = "III. Accesibilidad del sistema";

ws.Cell(row, 2).Value = r.Xiv?.Valor;

row++;

ws.Cell(row, 1).Value = "IV. Cómo le ha ayudado";

ws.Cell(row, 2).Value = r.Xv?.Valor;

row++;

ws.Cell(row, 1).Value = "V. Ahorro de dinero";

ws.Cell(row, 2).Value = r.Xvi?.Valor;

row++;

ws.Cell(row, 1).Value = "VI. Recomendaciones de mejora";

ws.Cell(row, 2).Value = r.Xvii?.Valor;

// Ajuste automático de columnas

ws.Columns().AdjustToContents();

using var stream = new MemoryStream();

workbook.SaveAs(stream);

return File(stream.ToArray(),

"application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",

$"Encuesta\_{id}.xlsx");

}

// Exportar todas las encuestas a Excel con formato de ficha individual

[HttpGet("exportartodosexcel")]

public async Task<IActionResult> ExportarTodasExcel()

{

var todas = await \_context.respuestas

.Include(x => x.Sexo)

.Include(x => x.Departamento)

.Include(x => x.Ciudad)

.Include(x => x.Facultad)

.Include(x => x.Carrera)

.Include(x => x.Matricula)

.Include(x => x.Becado)

.Include(x => x.Xii)

.Include(x => x.Xiii)

.Include(x => x.Xiv)

.Include(x => x.Xv)

.Include(x => x.Xvi)

.Include(x => x.Xvii)

.ToListAsync();

using var workbook = new XLWorkbook();

var ws = workbook.Worksheets.Add("Encuestas");

int currentRow = 1;

int row = 1;

foreach (var r in todas)

{

ws.Cell(row, 1).Value = "Encuesta de Análisis de Sistemas de Información";

ws.Cell(row, 1).Style.Font.Bold = true;

ws.Range(row, 1, row, 2).Merge();

row += 2;

ws.Cell(row, 1).Value = "Datos Generales";

ws.Cell(row, 1).Style.Font.Bold = true;

row++;

ws.Cell(row, 1).Value = "Nombres";

ws.Cell(row, 2).Value = r.Nombre;

row++;

ws.Cell(row, 1).Value = "Apellidos";

ws.Cell(row, 2).Value = r.Apellido;

row++;

ws.Cell(row, 1).Value = "Sexo";

ws.Cell(row, 2).Value = r.Sexo?.Valor;

row++;

ws.Cell(row, 1).Value = "Cédula de Identificación";

ws.Cell(row, 2).Value = r.Identidad;

row++;

ws.Cell(row, 1).Value = "Departamento";

ws.Cell(row, 2).Value = r.Departamento?.Valor;

row++;

ws.Cell(row, 1).Value = "Ciudad";

ws.Cell(row, 2).Value = r.Ciudad?.Valor;

row += 2;

ws.Cell(row, 1).Value = "Datos Académicos";

ws.Cell(row, 1).Style.Font.Bold = true;

row++;

ws.Cell(row, 1).Value = "Facultad";

ws.Cell(row, 2).Value = r.Facultad?.Valor;

row++;

ws.Cell(row, 1).Value = "Carrera";

ws.Cell(row, 2).Value = r.Carrera?.Valor;

row++;

ws.Cell(row, 1).Value = "Año de estudios";

ws.Cell(row, 2).Value = r.PreguntaIX;

row++;

ws.Cell(row, 1).Value = "Tipo de Matrícula";

ws.Cell(row, 2).Value = r.Matricula?.Valor;

row++;

ws.Cell(row, 1).Value = "Becado";

ws.Cell(row, 2).Value = r.Becado?.Valor;

row += 2;

ws.Cell(row, 1).Value = "Desarrollo";

ws.Cell(row, 1).Style.Font.Bold = true;

row++;

ws.Cell(row, 1).Value = "I. Uso del Sistema de Matrícula";

ws.Cell(row, 2).Value = r.Xii?.Valor;

row++;

ws.Cell(row, 1).Value = "II. Opinión del nuevo sistema";

ws.Cell(row, 2).Value = r.Xiii?.Valor;

row++;

ws.Cell(row, 1).Value = "III. Accesibilidad del sistema";

ws.Cell(row, 2).Value = r.Xiv?.Valor;

row++;

ws.Cell(row, 1).Value = "IV. Cómo le ha ayudado";

ws.Cell(row, 2).Value = r.Xv?.Valor;

row++;

ws.Cell(row, 1).Value = "V. Ahorro de dinero";

ws.Cell(row, 2).Value = r.Xvi?.Valor;

row++;

ws.Cell(row, 1).Value = "VI. Recomendaciones de mejora";

ws.Cell(row, 2).Value = r.Xvii?.Valor;

// Ajuste automático de columnas

ws.Columns().AdjustToContents();

// Espacio entre encuestas

currentRow += 2;

}

using var ms = new MemoryStream();

workbook.SaveAs(ms);

return File(ms.ToArray(),

"application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",

"Todas\_Encuestas\_Formato\_Individual.xlsx");

}

## Pasos de Instalación Rápida

### Fase I: Extracción del Paquete

#### Paso 1. Extracción con Contraseña (WinRAR)

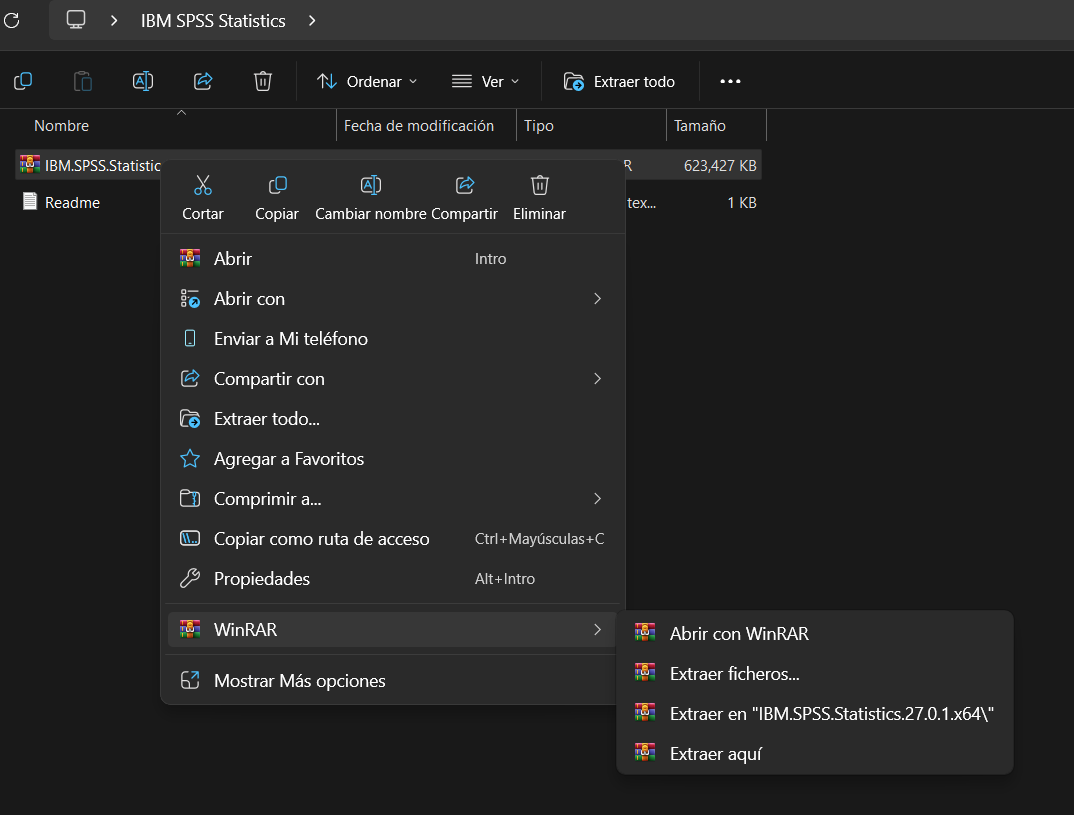
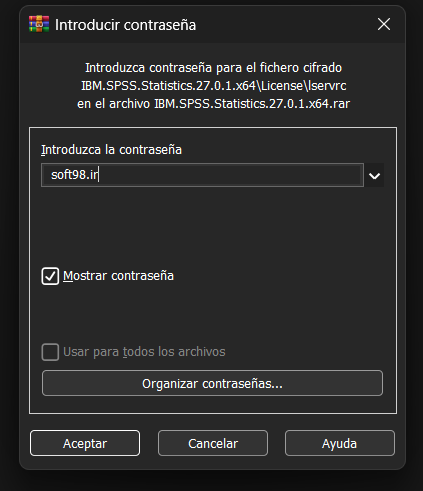
**Acción:** Haz doble clic en el archivo comprimido (.rar o similar). Al solicitarse, ingresa la contraseña **soft98** para iniciar la descompresión de los archivos.*.*

Figura 1. **Figura 1.** Ventana de WinRAR solicitando la contraseña ’soft98’ para la extracción.

**Nota.** Captura de la ventana del extractor de archivos.

#### Paso 2. Ubicar la Carpeta Descomprimida

**Acción:** Una vez completada la extracción, navega hasta la carpeta generada (por ejemplo, SPSS\_Statistics\_27) para verificar su contenido.

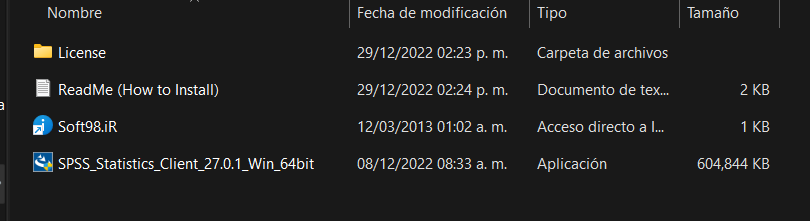
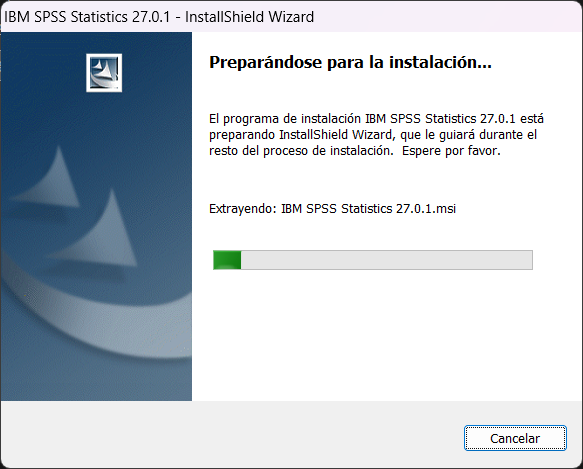


Figura 2. **Figura 2.** Carpeta de instalación con los archivos extraídos listos para usar.

**Nota.** Captura del Explorador de archivos mostrando la carpeta.

### Fase II: Asistente de Instalación

#### Paso 3. Ejecutar el Asistente de Instalación

**Acción:** Dentro de la carpeta descomprimida, busca y ejecuta el archivo de instalación principal (usualmente setup.exe o install.exe). Es recomendable hacer clic derecho y seleccionar **Ejecutar como administrador**.

#### Paso 4. Iniciar y Aceptar Licencia

**Acción:** Sigue las instrucciones iniciales del asistente (**Siguiente** / *Next*) y marca la casilla **Acepto los términos del acuerdo de licencia** para proceder.

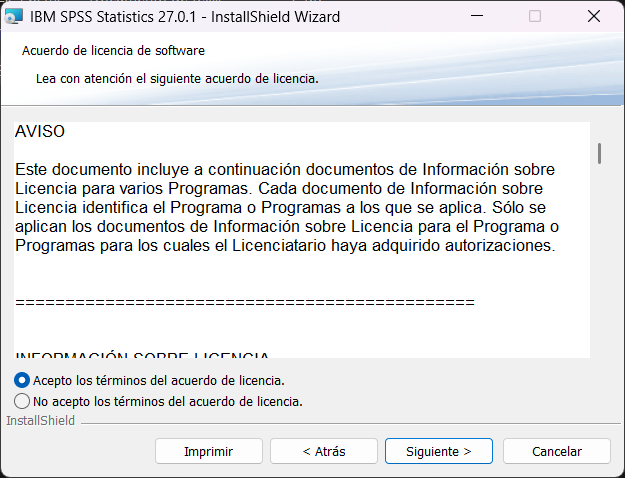
****

Figura 4. **Figura 4.** Pantalla de aceptación del acuerdo de licencia.

**Nota.** Captura mostrando la casilla de aceptación marcada.

#### Paso 5. Seleccionar la Ubicación de Instalación

**Acción:** Confirma la ubicación de instalación. Por defecto, será algo similar a C:

Program Files

IBM

SPSS

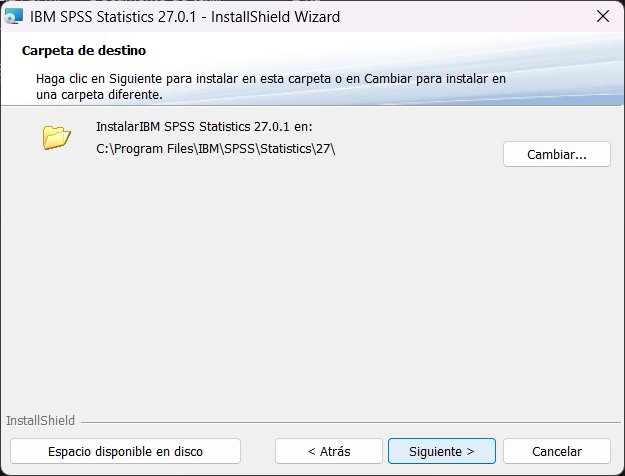
Statistic 27. Pulsa **Siguiente**.  
  
  
  


Figura 5. **Figura 5.** Revisar y aceptar la ubicación predeterminada del archivo.

**Nota.** Captura mostrando la ruta de instalación (p. ej., C: Program Files...).

#### Paso 6. Iniciar la Instalación y Finalizar

**Acción:** Haz clic en **Instalar** y espera a que el proceso de copia de archivos finalice.

Una vez que la instalación se haya completado, pulsa el botón **Finalizar** (*Finish*).

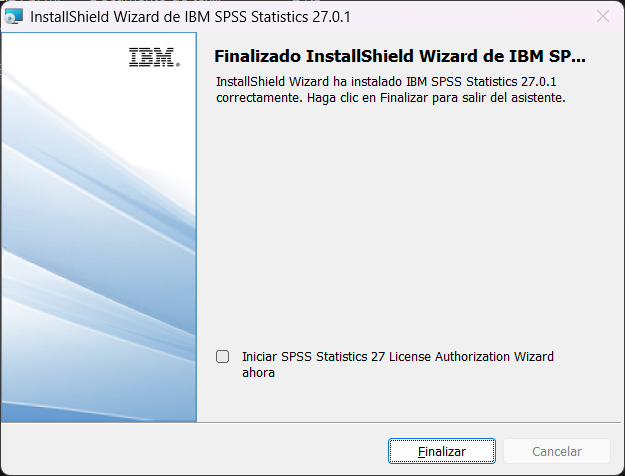


Figura 6. **Figura 6.** Instalación en progreso y confirmación finalizada.

**Nota.** Captura del asistente con la barra de progreso y el botón de ”Finalizar”.

### Fase III: Primer Inicio

#### Paso 7. Abrir el Programa y Aceptar Términos

**Acción:** Abre IBM SPSS Statistics 27 desde el icono del escritorio o el menú de inicio. En el primer inicio, puede aparecer una ventana de Bienvenida o de Licenciamiento.

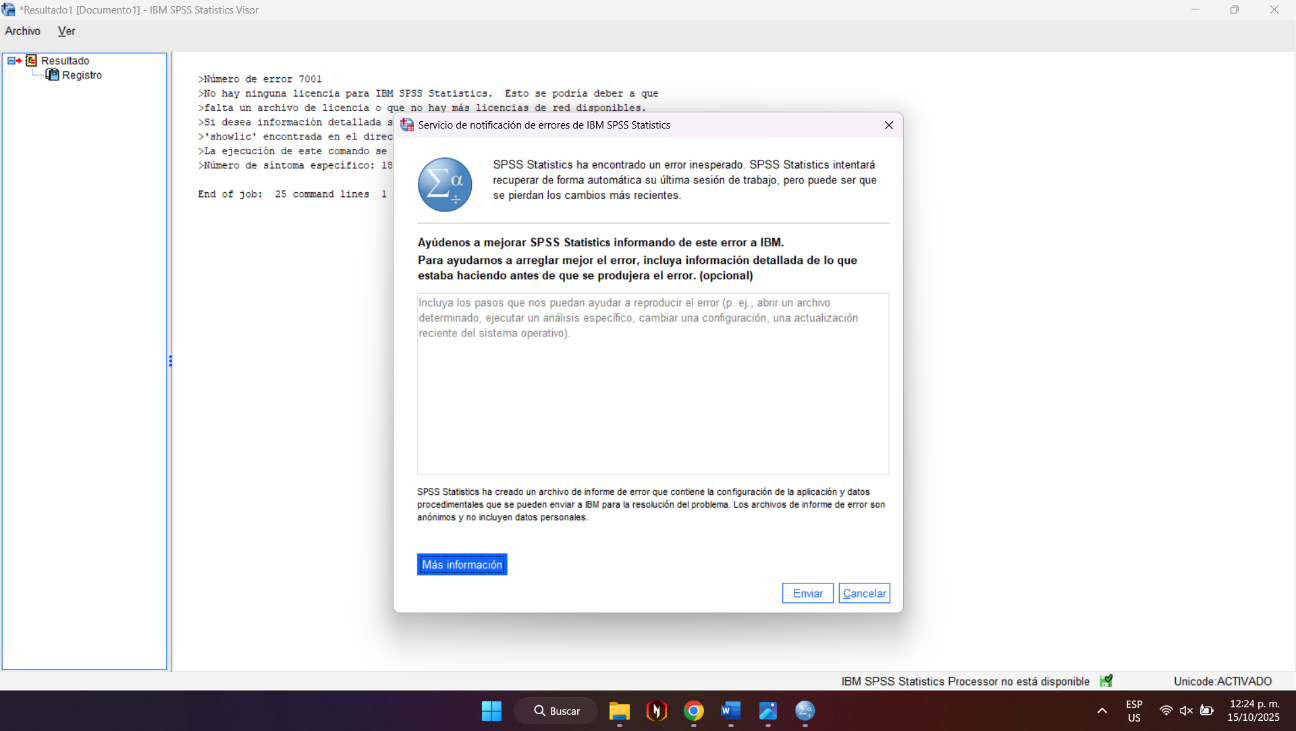
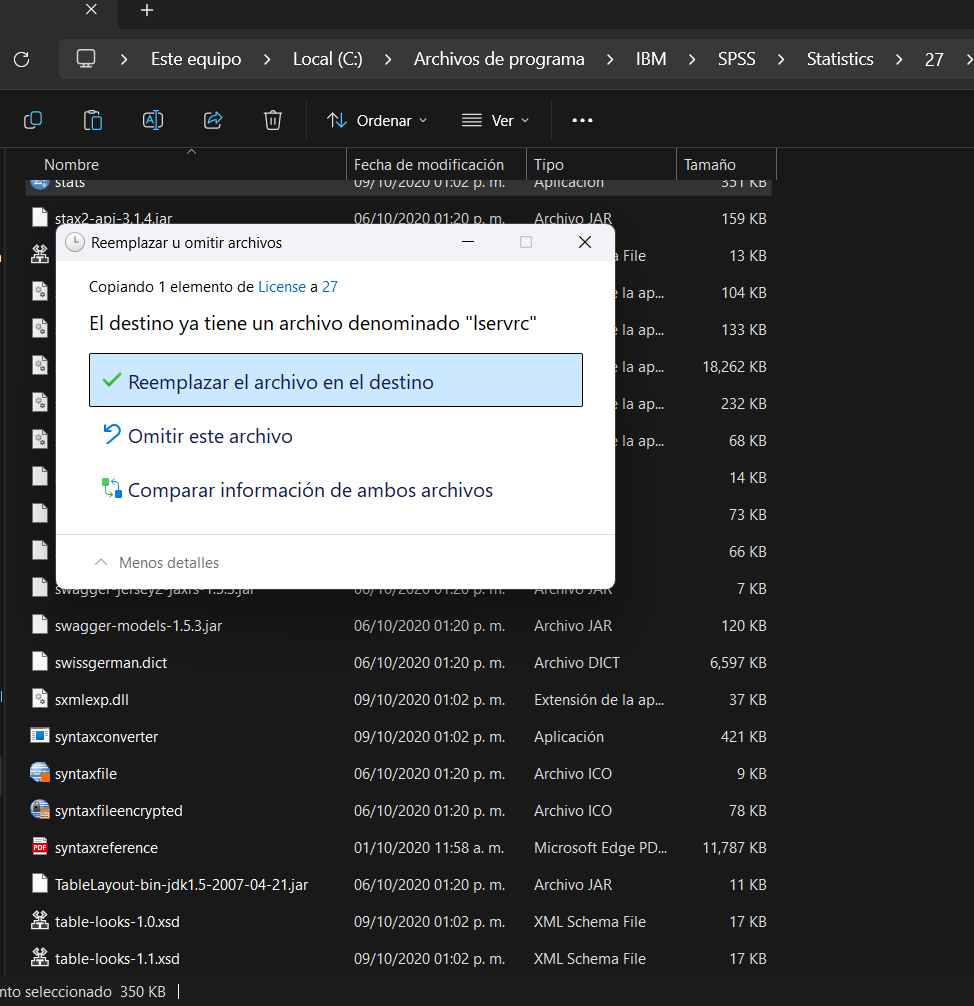
Acepta los términos o selecciona la opción para gestionar la licencia.

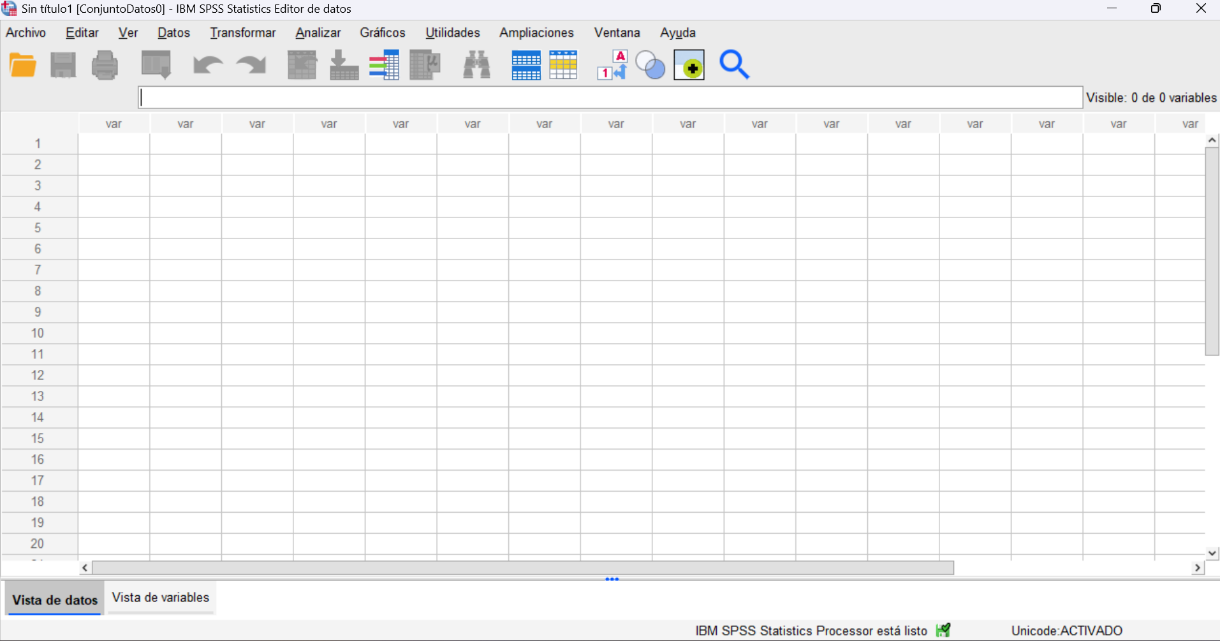
Figura 7. **Figura 7.** Primer inicio del programa y ventana de aceptación de términos. **Nota.** Captura mostrando la interfaz inicial de SPSS y la ventana de inicio de sesión/licencia.

### Fase IV: Activar SPSS. Pasos 1. Descomprimir 2. Instalar 3. Copiar el archivo "lservrc" dentro de la carpeta License al directorio de instalacion (C:\Program Files\IBM\SPSS\Statistics\27\ 4. Ejecutar



**Pasos para exportar tablas de bases de datos a IBM SPSS**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.Como primer paso a realizar, vamos a abrir la aplicación IBM SPSS Statistics.  
Luego para poder importar datos de una base de datos, buscaremos la pestaña de **“archivos”**, luego al submenú **“importar datos”**, según de donde vayamos a importar los datos usted va a elegir la opción que mas le convenga, en este caso, eligira la opción de **“Base de datos”** y le dará click a nueva consulta

Luego vamos a seleccionar el origen de los datos mediante **“ODBC”** y seleccionaremos el correspondiente.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Luego de elegir la conexión ODBC, usted va a seleccionar todas las tablas de las que se van a tomar los datos y seleccionar el botón finalizar.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.De esa manera, usted ya debería de estar viendo en el formulario una consulta básica en la que se vera todos los datos de las tablas importadas.

* Interfaz de usuario gráfica, Aplicación

  El contenido generado por IA puede ser incorrecto.Esta es la vista de los datos en bruto.
* Interfaz de usuario gráfica, Aplicación, Tabla, Excel

  El contenido generado por IA puede ser incorrecto.Esta es la importación de las relaciones y la estructura de la tabla con sus atributos.

**Pasos en SPSS para generar resultados con imágenes.**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Teniendo ya importados los datos de la base de datos, usted va a ir a menú de **“análisis”**, luego va al menú de **“estadísticas descriptivas”**, después elegirá el análisis que usted desee, en este caso se hará con el **“análisis descriptivo”.**

Después vamos a elegir los atributos de la tabla que se van a analizar, los cuales son los que están en el cuadro de la derecha.

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

Luego haga click en **“opciones”** y eliges las estadísticas que se van hacer con los datos y le da aceptar.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Luego usted le da click al botón aceptar y ya le mostrara la información estadística manera visual.

