

**Tema**

CONFIGURAR NETBEANS PARA DESARROLLAR APLICACIONES JAVA ENTERPRISE EDITION

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU. CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

31/05/2025

Tabla de contenido

[JAVA ENTERPRISE EDITION 6](#_Toc134613096)

[INTRODUCCIÓN 6](#_Toc134613097)

[OBJETIVO 9](#_Toc134613098)

[INSTALACIÓN DE LAS HERRAMIENTAS 9](#_Toc134613099)

[INSTALACIÓN DE POWER DESIGNER 9](#_Toc134613100)

[INSTALACIÓN DE NETBEANS 16](#_Toc134613101)

[INSTALACIÓN DE PAYARA SERVER 24](#_Toc134613102)

[INSTALACIÓN DE MYSQL 32](#_Toc134613103)

[POOL DE CONEXIÓN, BASE DE DATOS MYSQL – PAYARA 39](#_Toc134613104)

[Comandos útiles dentro de Java 42](#_Toc134613105)

[CONCLUSIONES 43](#_Toc134613106)

[RECOMENDACIONES 43](#_Toc134613107)

[BIBLIOGRAFÍA 44](#_Toc134613108)

[CONCLUSIONES 46](#_Toc134613109)

[RECOMENDACIONES 46](#_Toc134613110)

[BIBLIOGRAFÍA 46](#_Toc134613111)

**ÍNDICE DE IMÁGENES**

[Figura 1. Logotipo identificativo de Java Enterprise Edition 6](#_Toc134613134)

[Figura 2. Especificaciones de APIs de Java Enterprise Edition. 6](#_Toc134613135)

[Figura 3. Integración de JEE con Base de Datos mediante JPA. 7](#_Toc134613136)

[Figura 4. Integración de JEE Session Beans y capas con el API CDI. 7](#_Toc134613137)

[Figura 5. Mensajería en Java Enterpise Edition. 8](#_Toc134613138)

[Figura 6. Arquitectura SOA con JEE. 8](#_Toc134613139)

[Figura 7. Página principal de instalación de Power Designer 9](#_Toc134613140)

[Figura 8. Página de registro Power Designer 9](#_Toc134613141)

[Figura 9. Página descarga Power Designer 10](#_Toc134613142)

[Figura 10. Ventana Guardar archivo 10](#_Toc134613143)

[Figura 11. Selección del idioma 10](#_Toc134613144)

[Figura 12. Ventana instalación SAP 11](#_Toc134613145)

[Figura 13. Selección tipo de licencia SAP 11](#_Toc134613146)

[Figura 14 Acuerdo licencia SAP 12](#_Toc134613147)

[Figura 15 Selección carpeta destino SAP 12](#_Toc134613148)

[Figura 16. Selección características a instalar 13](#_Toc134613149)

[Figura 17. Ventana selección características a importar 13](#_Toc134613150)

[Figura 18. Selección nombre de la carpeta SAP 14](#_Toc134613151)

[Figura 19. Ventana revisión de las opciones seleccionadas 14](#_Toc134613152)

[Figura 20. Instalación de Power Designer 15](#_Toc134613153)

[Figura 21. Página inicio Power Designer 15](#_Toc134613154)

[Figura 22. Página descarga NetBeans 16](#_Toc134613155)

[Figura 23. Página descarga imagen Java 16](#_Toc134613156)

[Figura 22 Guardar archivo en computador 17](#_Toc134613157)

[Figura 23 Ventana instalador java 17](#_Toc134613158)

[Figura 24. Selecciones características JDK, JRE 17](#_Toc134613159)

[Figura 25 Selección carpeta destino Java 18](#_Toc134613160)

[Figura 26 Finalizar la instalación Java 18](#_Toc134613161)

[Figura 27 Instalación NetBeans 19](#_Toc134613162)

[Figura 28 Guardar el archivo 19](#_Toc134613163)

[Figura 29 Configuración del instalador NetBeans 20](#_Toc134613164)

[Figura 30 Características a instalar NetBeans 20](#_Toc134613165)

[Figura 31 Selección de paquetes y servidores a instalar 21](#_Toc134613166)

[Figura 32 Aceptación de términos 21](#_Toc134613167)

[Figura 33 Selección de carpeta de instalación 22](#_Toc134613168)

[Figura 34 Instalación de paquetes seleccionados 22](#_Toc134613169)

[Figura 35 Finalización de la instalación 23](#_Toc134613170)

[Figura 36. Servidor de Aplicaciones de NetBeans. 23](#_Toc134613171)

[Figura 37. Servidor de Aplicaciones de NetBeans y BDD que pueden trabajar con NetBeans 24](#_Toc134613172)

[Figura 38. Logo Identificativo de Payara 24](#_Toc134613173)

[Figura 39. Instalación Drivers Payara Server 25](#_Toc134613174)

[Figura 40. Instalación Drivers Payara Server 25](#_Toc134613175)

[Figura 41. Descarga de Payara. 26](#_Toc134613176)

[Figura 42. Pegar los archivos del Servidor Aplicaciones Payara en el Disco C 26](#_Toc134613177)

[Figura 43. Dirigirse al Directorio Program Files 27](#_Toc134613178)

[Figura 44. Add Server 27](#_Toc134613179)

[Figura 45. Insertar Payara 28](#_Toc134613180)

[Figura 46. Ubicación de Payara en Netbeans 28](#_Toc134613181)

[Figura 47. Dominio Payara 29](#_Toc134613182)

[Figura 48. Verificación instalación 29](#_Toc134613183)

[Figura 49. Ejecución Payara Server 29](#_Toc134613184)

[Figura 50. Sitio web de Payara 30](#_Toc134613185)

[Figura 51. Admin Console de Payara 30](#_Toc134613186)

[Figura 52. Ejecución Payara Server 31](#_Toc134613187)

[Figura 53. Ejecución Payara Server 31](#_Toc134613188)

[Figura 54. Ejecución Payara Server 31](#_Toc134613189)

[Figura 55. Integrar un driver para un RDBMS en NetBeans. 32](#_Toc134613190)

[Figura 56. Integrar un driver para un RDBMS en NetBeans. 32](#_Toc134613191)

[Figura 57. Selección del sistema operativo 33](#_Toc134613192)

[Figura 58. Acuerdo de licencia MySQL. 33](#_Toc134613193)

[Figura 59. Selección del tipo de instalación MySQL. 34](#_Toc134613194)

[Figura 60. Paquetes MySQL a instalarse. 34](#_Toc134613195)

[Figura 61. Configuración del producto. 35](#_Toc134613196)

[Figura 62. Configurar la disponibilidad MySQL. 35](#_Toc134613197)

[Figura 63. Configuraciones de conectividad. 36](#_Toc134613198)

[Figura 64. Selección del método de autenticación. 36](#_Toc134613199)

[Figura 65. Agregar usuarios y contraseñas. 37](#_Toc134613200)

[Figura 66. Ventana cambiar configuraciones del servicio. 37](#_Toc134613201)

[Figura 67. Ventana ejecución de las configuraciones realizadas. 38](#_Toc134613202)

[Figura 68. Ventana de conexión con el servidor. 38](#_Toc134613203)

[Figura 69. Ventana de instalación completa. 39](#_Toc134613204)

[Figura 70. Configurar base de datos 39](#_Toc134613205)

[Figura 71. Configurar base de datos 40](#_Toc134613206)

[Figura 72. Configurar base de datos 40](#_Toc134613207)

[Figura 73. Configurar base de datos 40](#_Toc134613208)

[Figura 74. Creando Pool de conexiones 41](#_Toc134613209)

[Figura 75. Connection Pool Properties 41](#_Toc134613210)

[Figura 76. Propiedades del Pool de Conexiones. 42](#_Toc134613211)

[Figura 77. Comandos útiles dentro de Java 43](#_Toc134613212)

**ÍNDICE DE TABLAS**

[Tabla 1. Resumen comandos útiles dentro de Java 42](#_Toc134613431)

# Proyecto en .NET con Restful

## Servicio en RestFull

Se crea primero el proyecto, se usa la plantilla de **API ASP.NET Core** (Plantilla: "ASP.NET Core WebAPI"), Asigna un nombre: ViajecitosAPI.

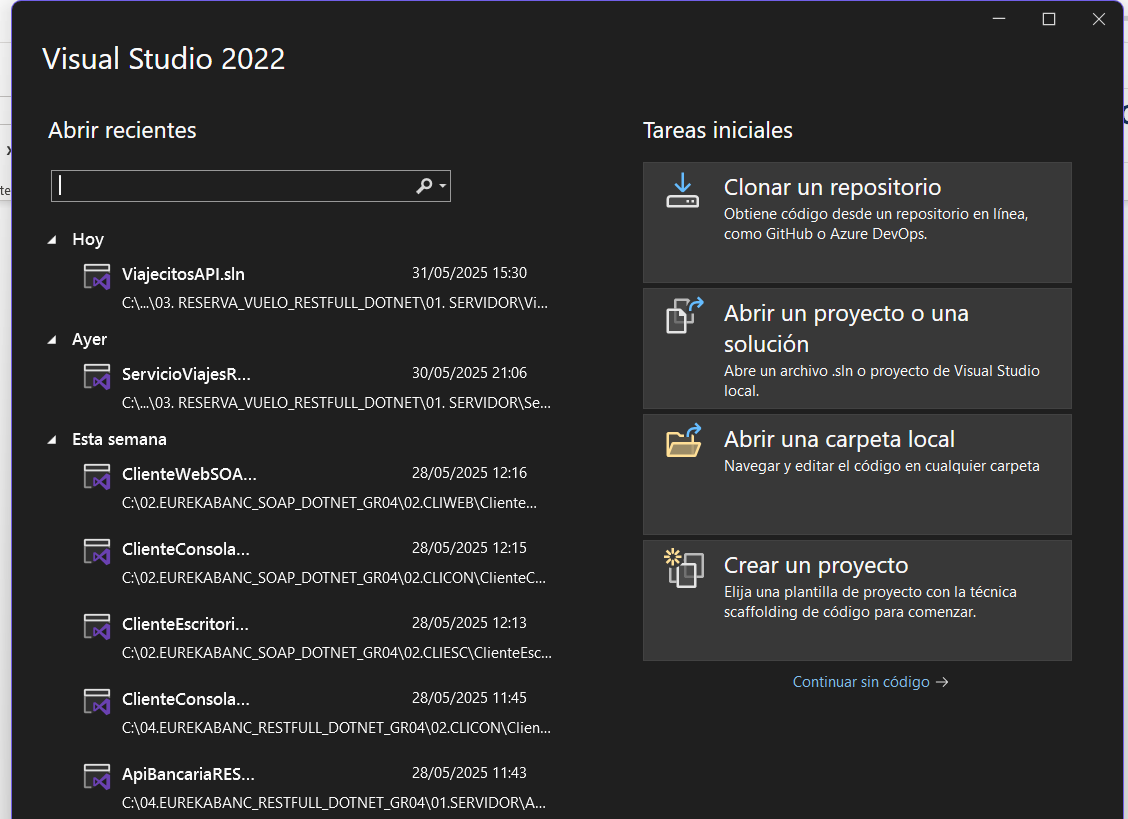


Figura 1. Creación del proyecto.

### Configuración del proyecto

1. Instala los paquetes NuGet necesarios:
   * Microsoft.EntityFrameworkCore
   * Microsoft.EntityFrameworkCore. SqlServer
   * Microsoft.EntityFrameworkCore. Tools
2. En appsettings. json, agrega la cadena de conexión:

Seguido de estos pasos, tenemos la estructura del proyecto de la siguiente manera, donde se divide en modelo, controlador y Data.

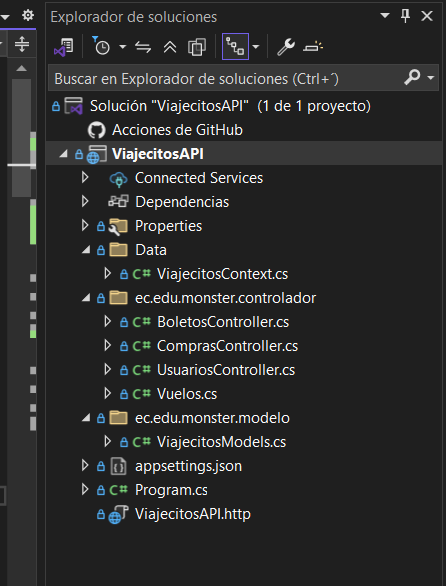


Figura 2. Estructura del proyecto.

### Crear los modelos

En la carpeta de ec.edu.monster.modelo se creara dentro un archivo ViajecitosModels.cs.

Que contendrá el siguiente código. Esta clase sirve como modelo de datos para la aplicación ViajecitosAPI, definiendo las entidades principales del sistema: Vuelo representa los vuelos disponibles con información de origen, destino, precio, hora de salida y asientos; Usuario representa a los usuarios con datos personales y credenciales; Boleto almacena los detalles de cada boleto comprado, como el usuario, el vuelo y el número de asiento; y Compra registra las transacciones realizadas por los usuarios, incluyendo el monto total y la fecha de compra. Estas clases permiten estructurar la información en la base de datos, realizar operaciones CRUD y comunicar datos en la API RESTful de manera coherente.

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace ViajecitosAPI.ec.edu.monster.modelo

{

public class Vuelo

{

[Key]

public int id\_vuelo { get; set; }

public string ciudad\_origen { get; set; }

public string ciudad\_destino { get; set; }

public decimal valor { get; set; }

public DateTime hora\_salida { get; set; }

public int asientos\_disponibles { get; set; }

}

public class Usuario

{

[Key]

public int id\_usuario { get; set; }

public string nombre\_usuario { get; set; }

public string apellido\_usuario { get; set; }

public string cedula { get; set; }

public string celular { get; set; }

public string email { get; set; }

public string contrasena { get; set; }

}

public class Boleto

{

[Key]

public int id\_boleto { get; set; }

[ForeignKey("Usuario")]

public int id\_usuario { get; set; }

[ForeignKey("Vuelo")]

public int id\_vuelo { get; set; }

public DateTime fecha\_compra { get; set; }

public string numero\_asiento { get; set; }

}

public class Compra

{

[Key]

public int id\_compra { get; set; }

[ForeignKey("Usuario")]

public int id\_usuario { get; set; }

[ForeignKey("Boleto")]

public int id\_boleto { get; set; }

public DateTime fecha\_compra { get; set; }

public decimal monto\_total { get; set; }

}

}

### Crear el Contexto de Base de Datos

La clase ViajecitosContext es el contexto de base de datos para la aplicación ViajecitosAPI, y sirve como el puente entre la base de datos y el código en C#. Se crea dentro de la carpeta Data y hereda de DbContext de Entity Framework Core, lo que permite mapear las clases del modelo (Vuelo, Usuario, Boleto y Compra) a tablas en la base de datos. Esta clase define cuatro propiedades DbSet (una por cada entidad del modelo), que representan las tablas y permiten realizar operaciones como consultas, inserciones, actualizaciones y eliminaciones en la base de datos. Además, en el constructor recibe opciones de configuración (como la cadena de conexión) que permiten establecer cómo conectarse a la base de datos. En resumen, ViajecitosContext es el motor que conecta el código con la base de datos y permite trabajar con las entidades del proyecto.

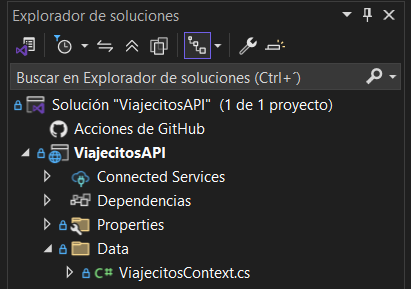


Figura 3. Carpeta de la Data.

using Microsoft.EntityFrameworkCore;

using ViajecitosAPI.ec.edu.monster.modelo;

namespace ViajecitosAPI.Data

{

public class ViajecitosContext : DbContext

{

public ViajecitosContext(DbContextOptions<ViajecitosContext> options) : base(options) { }

public DbSet<Vuelo> Vuelos { get; set; }

public DbSet<Usuario> Usuarios { get; set; }

public DbSet<Boleto> Boletos { get; set; }

public DbSet<Compra> Compras { get; set; }

}

}

### Configurar el Contexto en Program.cs

En esta parte del código se configura el contexto de la base de datos en Program.cs, que es el archivo de inicio del proyecto ViajecitosAPI. Se utiliza inyección de dependencias para registrar el servicio ViajecitosContext dentro del contenedor de servicios de la aplicación, permitiendo que se pueda utilizar el contexto en cualquier parte del proyecto. Esto se hace con el método AddDbContext, donde se indica que se usará SQL Server y se obtiene la cadena de conexión llamada ViajecitosConnection desde el archivo de configuración (appsettings.json). Además, se configuran los servicios de controladores, la documentación con Swagger, el HTTPS y el mapeo de rutas de la API. En resumen, esta configuración permite que la aplicación esté conectada a la base de datos y tenga disponibles los servicios para exponer los endpoints REST.

using Microsoft.EntityFrameworkCore;

using ViajecitosAPI.ec.edu.monster.controlador;

using ViajecitosAPI;

using ViajecitosAPI.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddDbContext<ViajecitosContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("ViajecitosConnection")));

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

### Crear los Controladores REST

Este código crea un controlador REST llamado VuelosController, que permite exponer los endpoints de la API para manejar las operaciones CRUD (crear, leer, actualizar y eliminar) sobre la entidad Vuelo. Se utiliza el atributo [ApiController] para marcarlo como controlador de API y [Route("api/[controller]")] para definir la ruta base como /api/vuelos. El controlador tiene un constructor que recibe el contexto de base de datos ViajecitosContext mediante inyección de dependencias, permitiendo acceder a la base de datos. Se definen cinco métodos: GetVuelos devuelve la lista de todos los vuelos, GetVuelo busca un vuelo por su ID, PostVuelo crea un nuevo vuelo, PutVuelo actualiza un vuelo existente y DeleteVuelo elimina un vuelo por su ID. Cada método es asincrónico (async) y utiliza las herramientas de Entity Framework Core para interactuar con la base de datos. En resumen, este controlador permite a la API recibir y procesar solicitudes HTTP para la gestión de los vuelos.

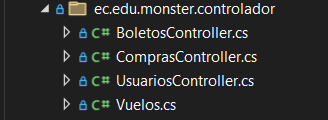


Figura 4. Carpeta de los controladores.

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ViajecitosAPI.Data;

using ViajecitosAPI.ec.edu.monster.modelo;

namespace ViajecitosAPI.ec.edu.monster.controlador

{

[ApiController]

[Route("api/[controller]")]

public class VuelosController : ControllerBase

{

private readonly ViajecitosContext \_context;

public VuelosController(ViajecitosContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<Vuelo>>> GetVuelos()

{

return await \_context.Vuelos.ToListAsync();

}

[HttpGet("{id}")]

public async Task<ActionResult<Vuelo>> GetVuelo(int id)

{

var vuelo = await \_context.Vuelos.FindAsync(id);

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

return vuelo;

}

[HttpPost]

public async Task<ActionResult<Vuelo>> PostVuelo(Vuelo vuelo)

{

\_context.Vuelos.Add(vuelo);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetVuelo), new { id = vuelo.id\_vuelo }, vuelo);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutVuelo(int id, Vuelo vuelo)

{

if (id != vuelo.id\_vuelo) return BadRequest();

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

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteVuelo(int id)

{

var vuelo = await \_context.Vuelos.FindAsync(id);

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

\_context.Vuelos.Remove(vuelo);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

El controlador UsuariosController es una clase que gestiona las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para los usuarios de la aplicación. Se marca como un controlador de API ([ApiController]) y se expone bajo la ruta api/usuarios. En su constructor, recibe el contexto de base de datos ViajecitosContext por inyección de dependencias para acceder a la base de datos. Los métodos definidos son: GetUsuarios para obtener la lista de todos los usuarios, GetUsuario para obtener un usuario específico por su ID, PostUsuario para crear un nuevo usuario, PutUsuario para actualizar un usuario existente y DeleteUsuario para eliminar un usuario. Cada método es asíncrono (async) y utiliza Entity Framework Core para interactuar con la base de datos, facilitando así la gestión de datos de los usuarios mediante peticiones HTTP.

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ViajecitosAPI.Data;

using ViajecitosAPI.ec.edu.monster.modelo;

namespace ViajecitosAPI.ec.edu.monster.controlador

{

[ApiController]

[Route("api/[controller]")]

public class UsuariosController : ControllerBase

{

private readonly ViajecitosContext \_context;

public UsuariosController(ViajecitosContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<Usuario>>> GetUsuarios()

{

return await \_context.Usuarios.ToListAsync();

}

[HttpGet("{id}")]

public async Task<ActionResult<Usuario>> GetUsuario(int id)

{

var usuario = await \_context.Usuarios.FindAsync(id);

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

return usuario;

}

[HttpPost]

public async Task<ActionResult<Usuario>> PostUsuario(Usuario usuario)

{

\_context.Usuarios.Add(usuario);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetUsuario), new { id = usuario.id\_usuario }, usuario);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutUsuario(int id, Usuario usuario)

{

if (id != usuario.id\_usuario) return BadRequest();

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

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteUsuario(int id)

{

var usuario = await \_context.Usuarios.FindAsync(id);

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

\_context.Usuarios.Remove(usuario);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

El controlador BoletosController permite gestionar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para los boletos de la aplicación. Se configura como un controlador API ([ApiController]) y expone sus rutas bajo api/boletos. Usa el contexto de base de datos ViajecitosContext para acceder a la información de los boletos en la base de datos. Define métodos para obtener todos los boletos (GetBoletos), obtener un boleto específico por su ID (GetBoleto), crear un nuevo boleto (PostBoleto), actualizar un boleto existente (PutBoleto) y eliminar un boleto (DeleteBoleto). Cada método es asíncrono (async) y usa Entity Framework Core para manipular los datos, facilitando así la gestión de los boletos a través de solicitudes HTTP en la API.

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ViajecitosAPI.Data;

using ViajecitosAPI.ec.edu.monster.modelo;

namespace ViajecitosAPI.ec.edu.monster.controlador

{

[ApiController]

[Route("api/[controller]")]

public class BoletosController : ControllerBase

{

private readonly ViajecitosContext \_context;

public BoletosController(ViajecitosContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<Boleto>>> GetBoletos()

{

return await \_context.Boletos.ToListAsync();

}

[HttpGet("{id}")]

public async Task<ActionResult<Boleto>> GetBoleto(int id)

{

var boleto = await \_context.Boletos.FindAsync(id);

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

return boleto;

}

[HttpPost]

public async Task<ActionResult<Boleto>> PostBoleto(Boleto boleto)

{

\_context.Boletos.Add(boleto);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetBoleto), new { id = boleto.id\_boleto }, boleto);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutBoleto(int id, Boleto boleto)

{

if (id != boleto.id\_boleto) return BadRequest();

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

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteBoleto(int id)

{

var boleto = await \_context.Boletos.FindAsync(id);

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

\_context.Boletos.Remove(boleto);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

El ComprasController es un controlador REST que permite manejar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para las compras realizadas en la aplicación. Este controlador está configurado como un API ([ApiController]) y expone sus rutas bajo api/compras. Usa el contexto de base de datos ViajecitosContext para acceder y manipular la información de las compras. Tiene métodos para obtener la lista completa de compras (GetCompras), buscar una compra por su ID (GetCompra), registrar una nueva compra (PostCompra), actualizar una compra existente (PutCompra) y eliminar una compra (DeleteCompra). Los métodos son asíncronos (async) y utilizan Entity Framework Core para interactuar con la base de datos, asegurando la persistencia de datos y la funcionalidad de la API.

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ViajecitosAPI.Data;

using ViajecitosAPI.ec.edu.monster.modelo;

namespace ViajecitosAPI.ec.edu.monster.controlador

{

[ApiController]

[Route("api/[controller]")]

public class ComprasController : ControllerBase

{

private readonly ViajecitosContext \_context;

public ComprasController(ViajecitosContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<Compra>>> GetCompras()

{

return await \_context.Compras.ToListAsync();

}

[HttpGet("{id}")]

public async Task<ActionResult<Compra>> GetCompra(int id)

{

var compra = await \_context.Compras.FindAsync(id);

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

return compra;

}

[HttpPost]

public async Task<ActionResult<Compra>> PostCompra(Compra compra)

{

\_context.Compras.Add(compra);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetCompra), new { id = compra.id\_compra }, compra);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutCompra(int id, Compra compra)

{

if (id != compra.id\_compra) return BadRequest();

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

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteCompra(int id)

{

var compra = await \_context.Compras.FindAsync(id);

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

\_context.Compras.Remove(compra);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

En el archivo appsettings.json se configura la cadena de conexión a la base de datos SQL Server utilizada por la aplicación ViajecitosAPI. La propiedad "ViajecitosConnection" contiene la información necesaria para conectarse: el servidor (localhost), el nombre de la base de datos (viajecitos\_sa), el usuario (sa) y la contraseña (Adali020). También incluye TrustServerCertificate=True para aceptar certificados no seguros en el entorno de desarrollo. Además, el archivo configura el nivel de logging (registro de eventos) de la aplicación, estableciendo el nivel predeterminado en Information y limitando los mensajes de Microsoft a Warning. Finalmente, "AllowedHosts": "\*" indica que la API acepta solicitudes de cualquier host. En resumen, este archivo centraliza las configuraciones clave del proyecto, como la conexión a la base de datos y los ajustes de log.

{

"ConnectionStrings": {

"ViajecitosConnection": "Server=DESKTOP-GM48FCK;Database=viajecitos\_sa;User Id=sa;Password=Adali020;TrustServerCertificate=True;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

## INSTALACIÓN DE LAS HERRAMIENTAS

Se procede a instalar todas las herramientas con las que se trabajará para el desarrollo de aplicaciones en Java Enterprise Edition.

### INSTALACIÓN DE POWER DESIGNER

1. Dirigirse a la página oficial de SAP para su descarga.

* <https://www.sap.com/products/powerdesigner-data-modeling-tools.html>

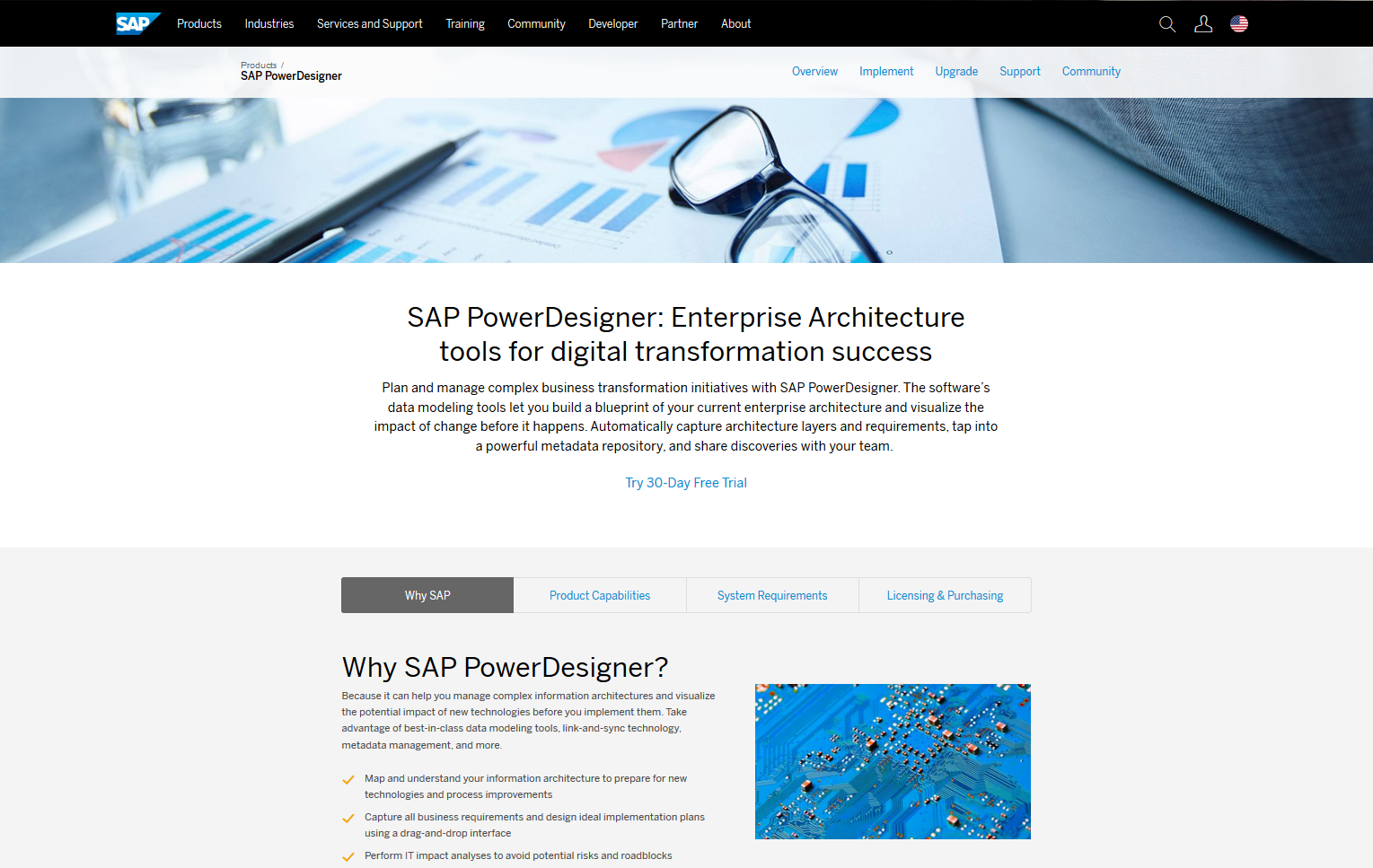


Figura 7. Página principal de instalación de Power Designer

1. Se procede dando clic en Try 30-Day Free Trial que reenvía a la siguiente página:



Figura 8. Página de registro Power Designer

1. ¡En esta página deberán ingresarse los datos y dar clic en “try now!”, lo que redirige a la siguiente página:

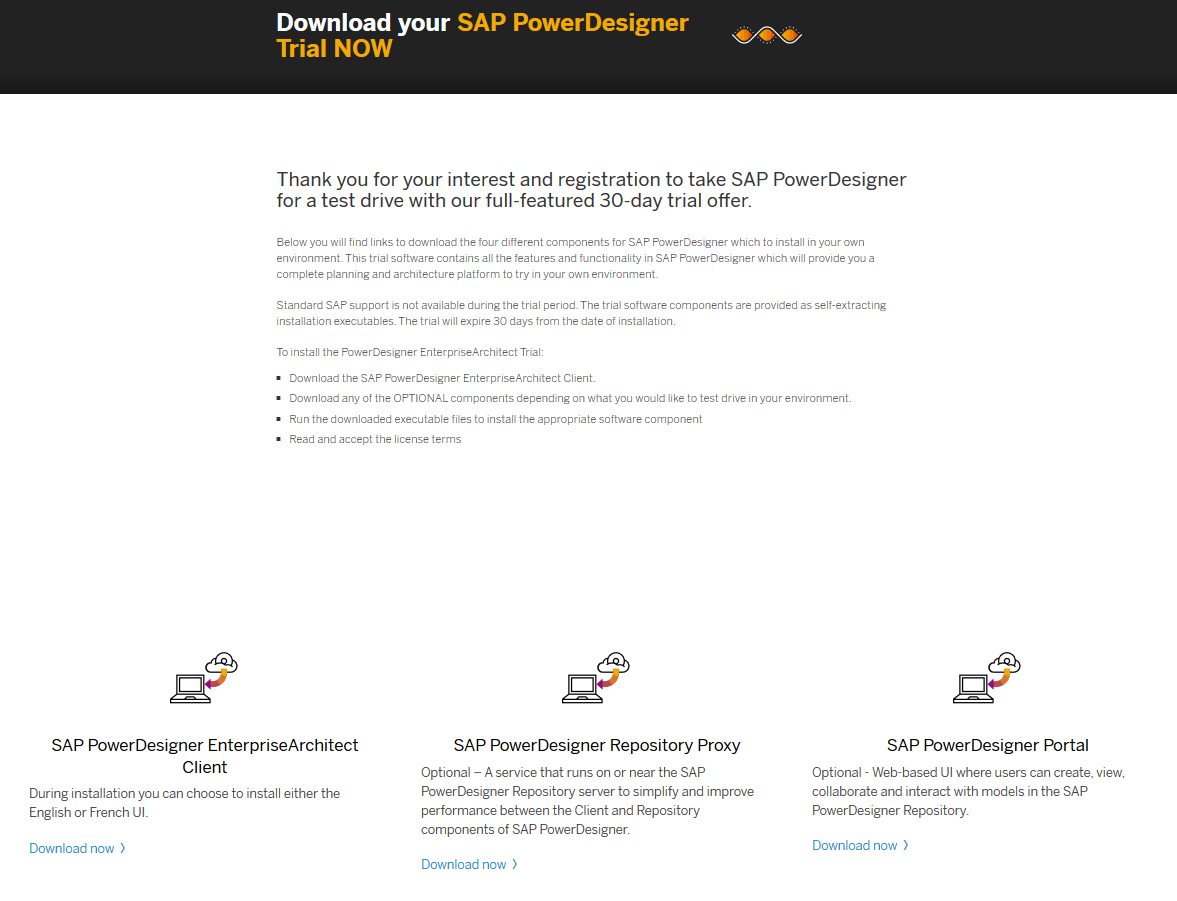


Figura 9. Página descarga Power Designer

1. Con esta página se obtiene el registro necesario para descargar el trial, lo siguiente es seleccionar SAP PowerDesigner EntrerpriseArchitect Client y en “Download now”

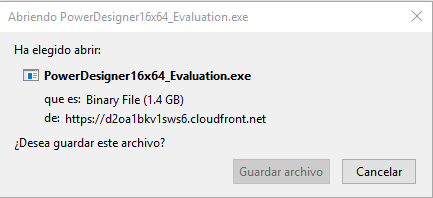


Figura 10. Ventana Guardar archivo

1. Se ejecuta el archivo para su instalación, en caso de requerirse se conceden permisos de administrador.

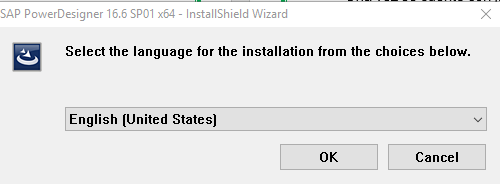


Figura 11. Selección del idioma

1. Al abrirse el instalador, se muestra la selección de idioma, se selecciona inglés por defecto y se da clic en ok.

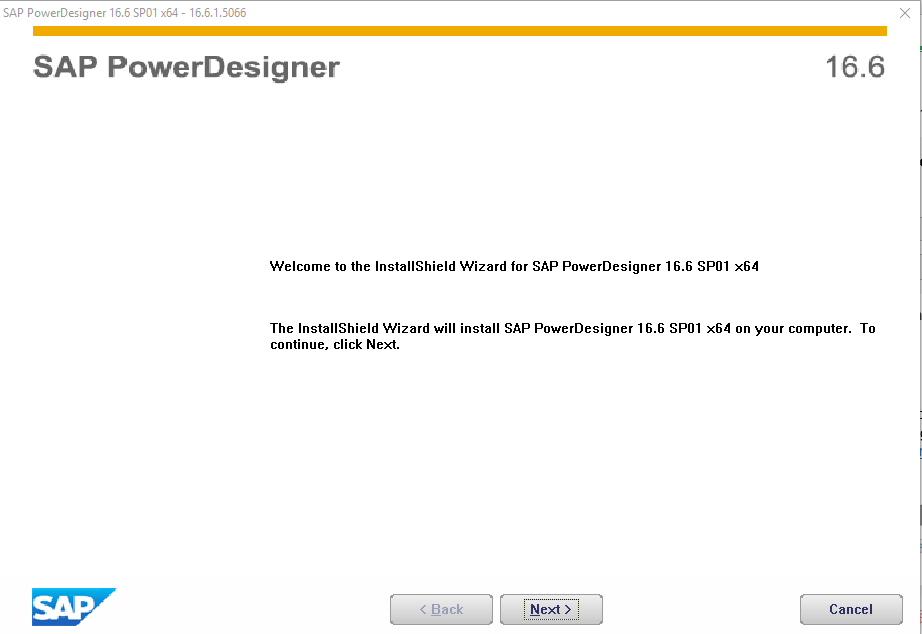


Figura 52. Ventana instalación SAP

1. Al seleccionar el idioma se muestra la siguiente ventana, que le informa que SAP instalará Power Designer en su computadora, clic en next.

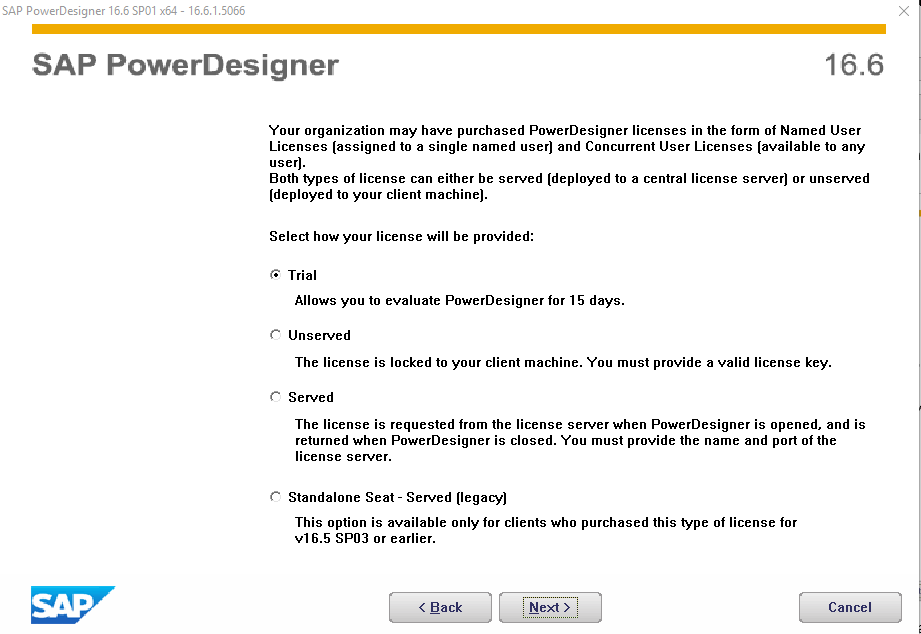


Figura 13. Selección tipo de licencia SAP

1. En la siguiente ventana se selecciona que se cuenta con una licencia trial.



Figura 14. Acuerdo licencia SAP

1. En la siguiente ventana se selecciona la localización donde va a instalar el software, y se aceptan los términos de licencia.

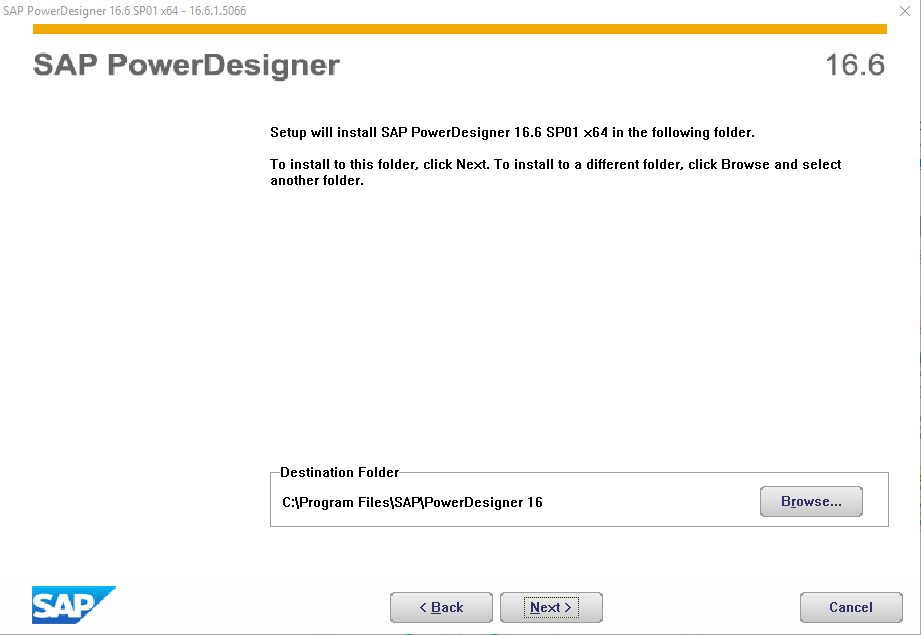


Figura 15. Selección carpeta destino SAP

1. En la siguiente ventana se selecciona la carpeta de destino para power designer y se da clic en siguiente.

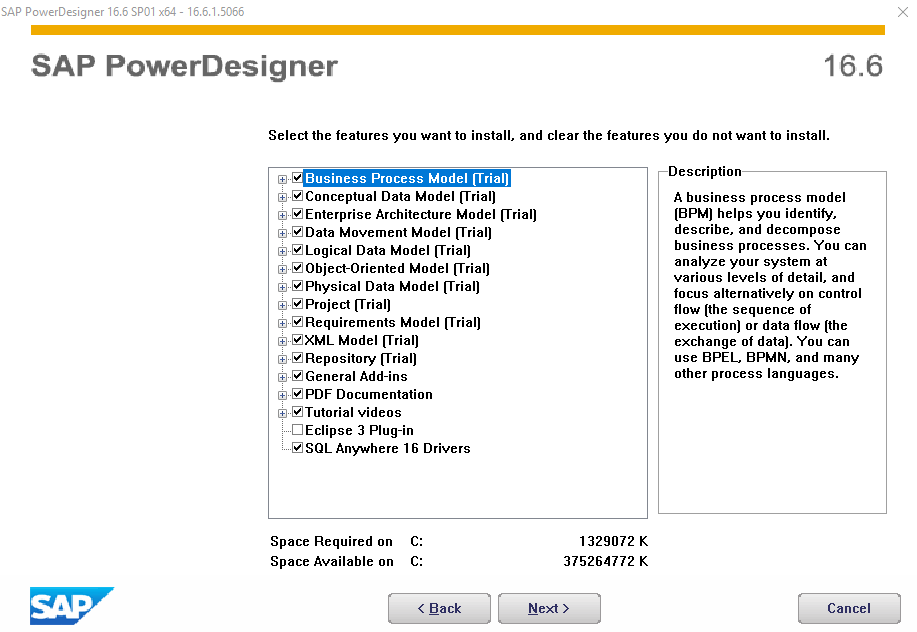


Figura 16. Selección características a instalar

1. En la siguiente carpeta se seleccionan las características a instalar, aquí se seleccionan todas a excepción de eclipse, o en caso de contar con eclipse se puede instalar el plugin.

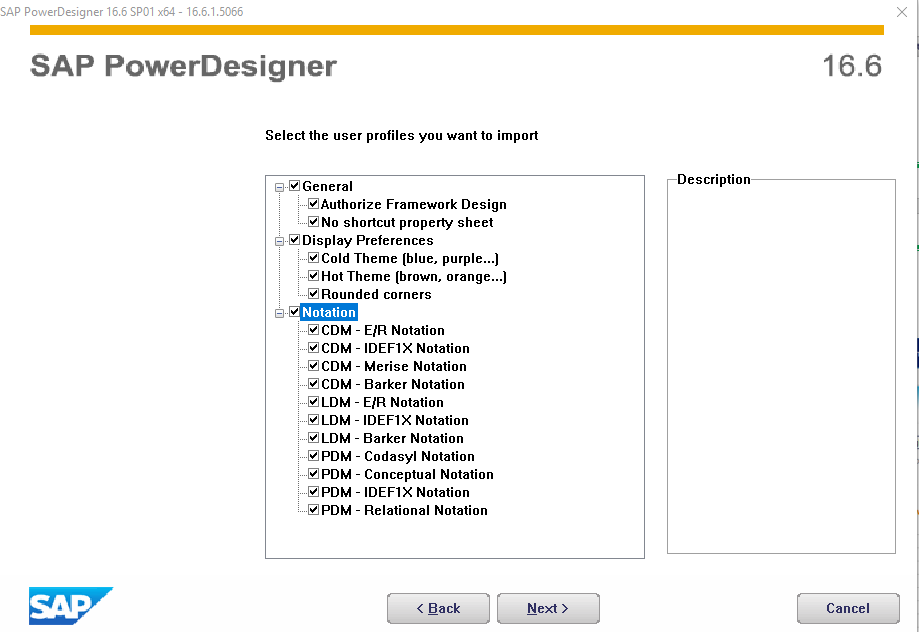


Figura 17. Ventana selección características a importar

1. En la siguiente ventana se seleccionan los perfiles de usuario a importar, estos dependerán de las necesidades de cada usuario, en este caso se seleccionaron todos los perfiles disponibles

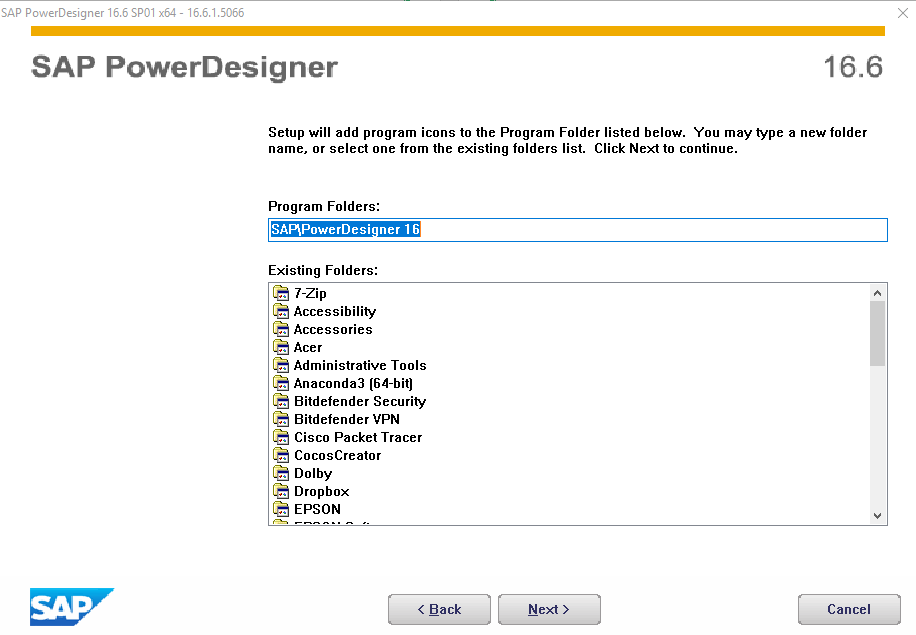


Figura 18. Selección nombre de la carpeta SAP

1. Es la siguiente ventana se puede cambiar el nombre de la carpeta del programa, se lo deja por defecto y clic en siguiente.

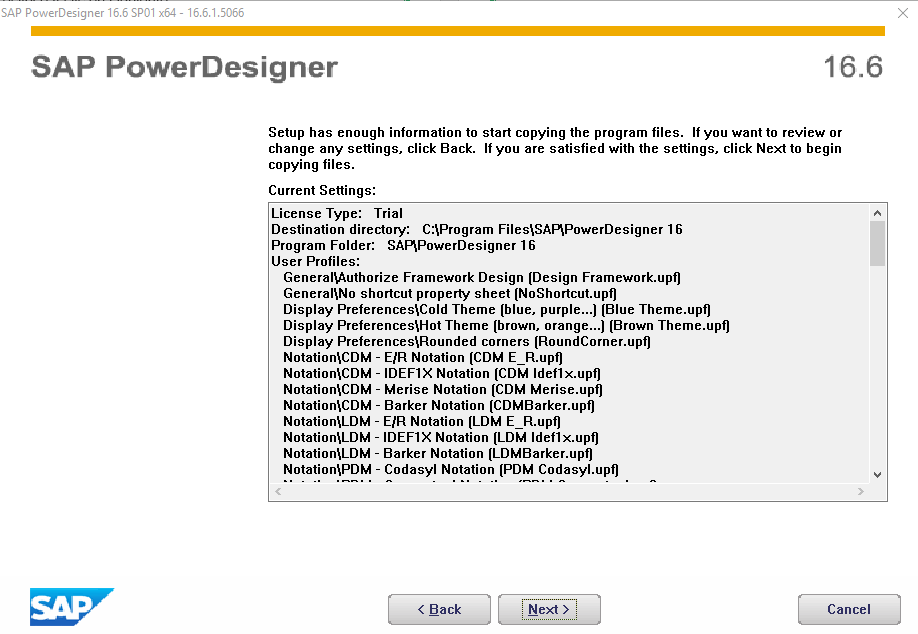


Figura 19. Ventana revisión de las opciones seleccionadas

1. En la siguiente ventana se muestra una revisión de las opciones seleccionadas anteriormente y al estar de acuerdo con las configuraciones, clic en siguiente.

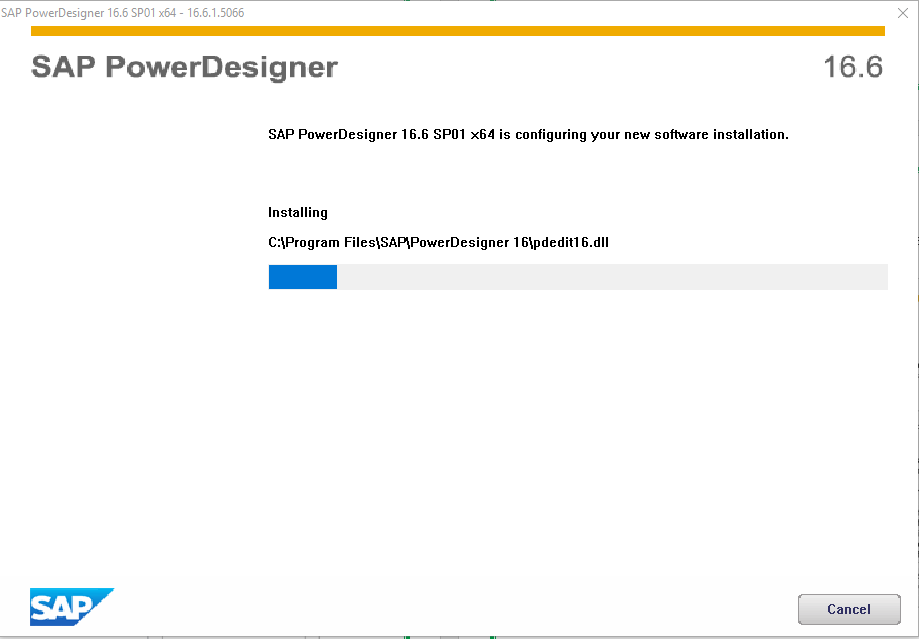


Figura 20. Instalación de Power Designer

A continuación, se muestra la pantalla de inicio de Power Designer

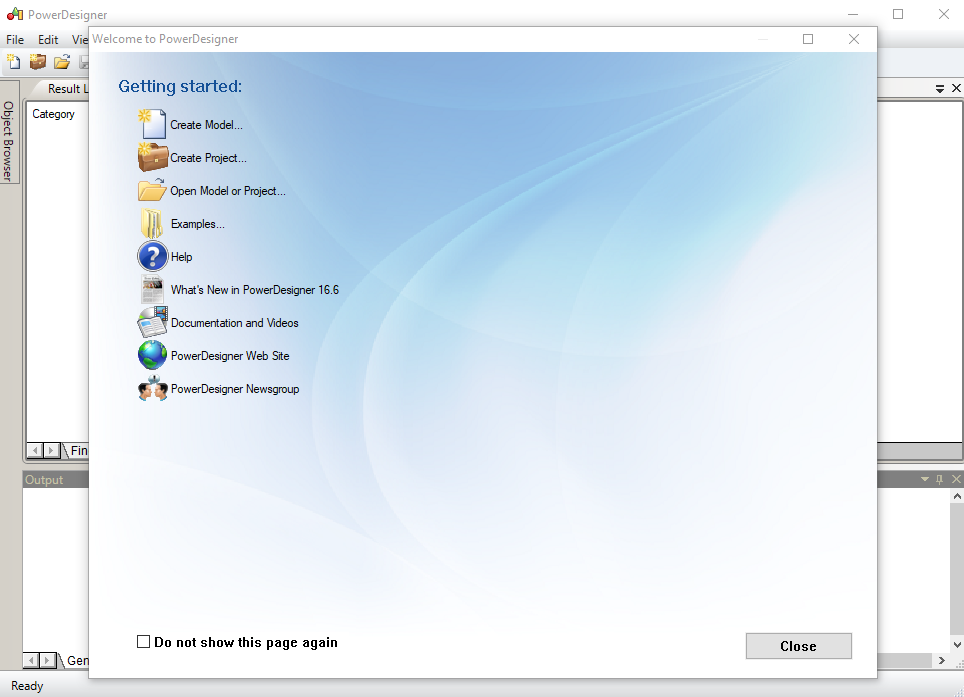


Figura 21. Página inicio Power Designer

### INSTALACIÓN DE NETBEANS

1. Para instalar NetBeans, es necesario instalar java JDK y JRE, para esto se procede mediante la instalación de java SE que proporciona estos dos, la instalación será la versión 8u211 o superior que se conseguirá en la siguiente página:

* <https://www.oracle.com/technetwork/es/java/javase/downloads/index.html>

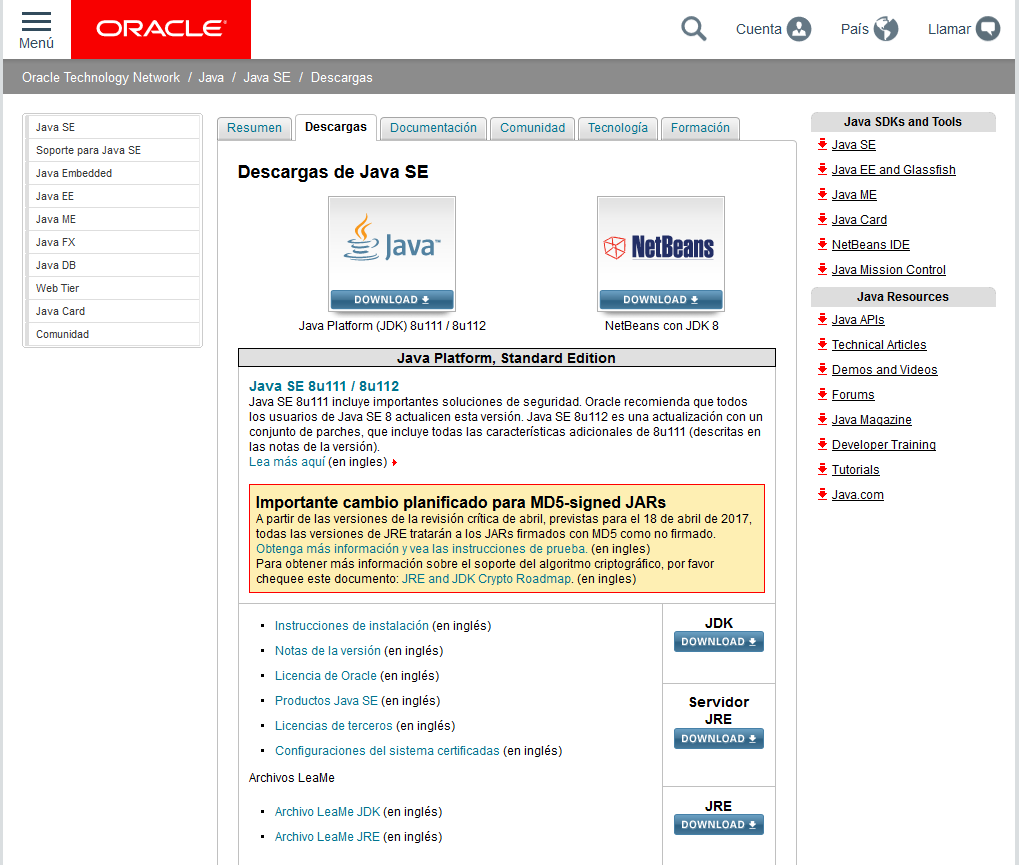


Figura 22. Página descarga NetBeans

1. En esta página se da clic en Down load con la imagen de java.

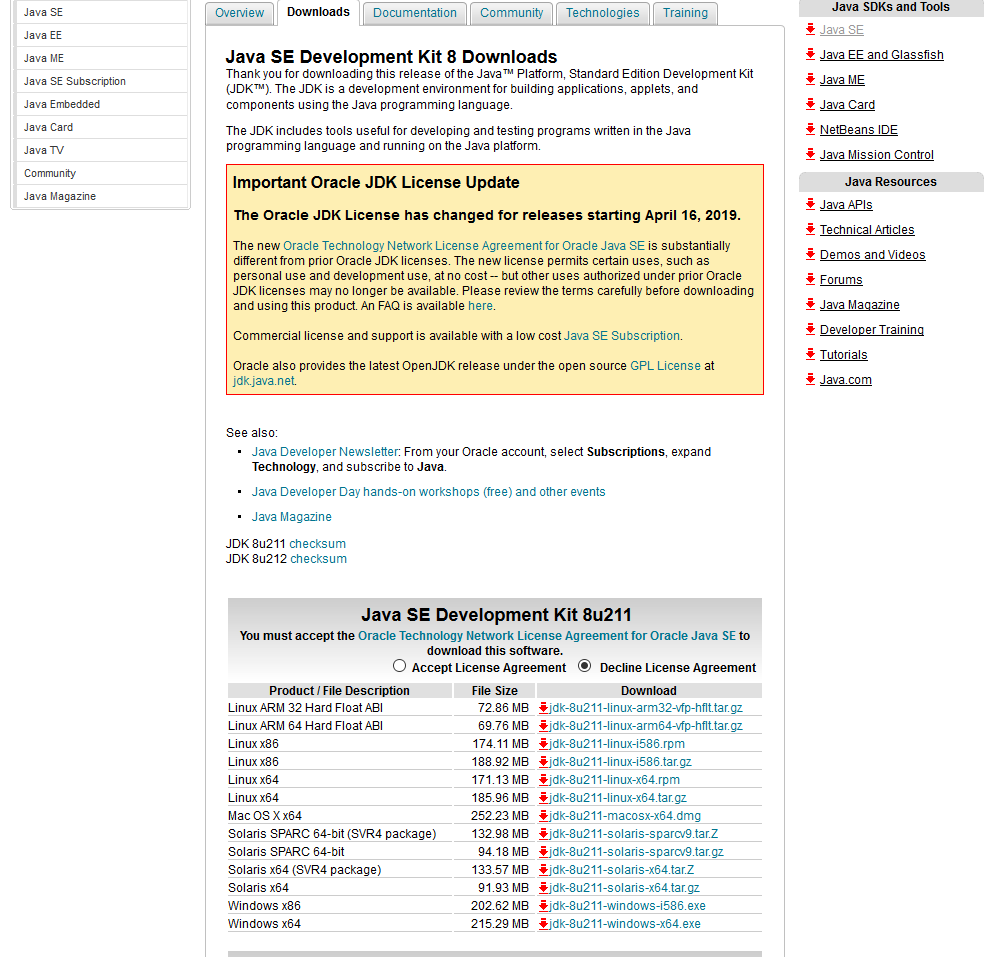


Figura 23. Página descarga imagen Java

1. Posteriormente se aceptan los acuerdos de licencia y se da clic en la versión del sistema operativo con el que se cuente, en este caso se utiliza Windows de 64 bits.

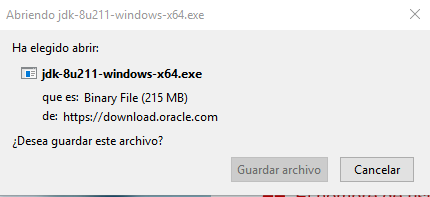


Figura 24. Guardar archivo en computador

1. Se guarda el archivo en el computador. Se ejecuta y en caso de requerirse se conceden permisos de administrador.

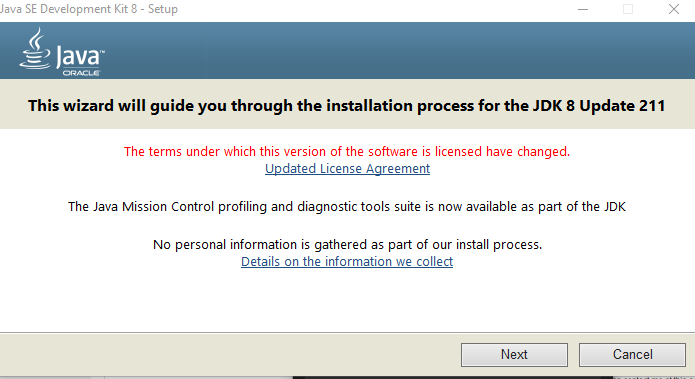


Figura 25. Ventana instalador java

1. Se presenta el instalador de java y se procede dando clic en siguiente.

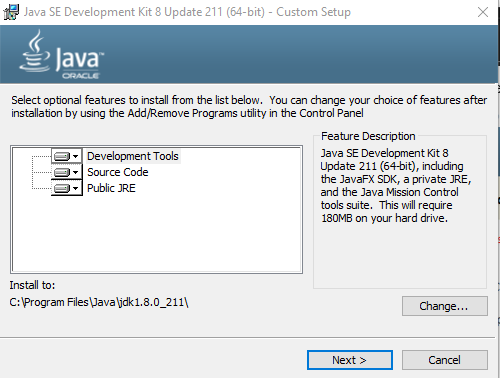


Figura 26. Selecciones características JDK, JRE

1. Seleccione las características que irán con el JDK, al ser JSE también se instalará el JRE como se evidencia en la imagen anterior, se procede dando clic en siguiente.

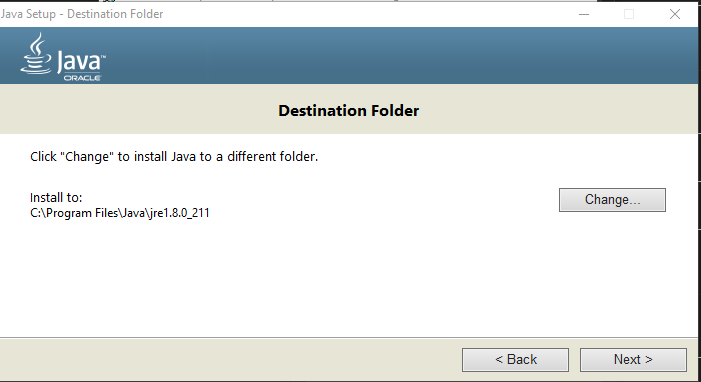


Figura 6 Selección carpeta destino Java

1. A la mitad de la instalación se instalará el JRE correspondiente, se procede dando clic en siguiente.

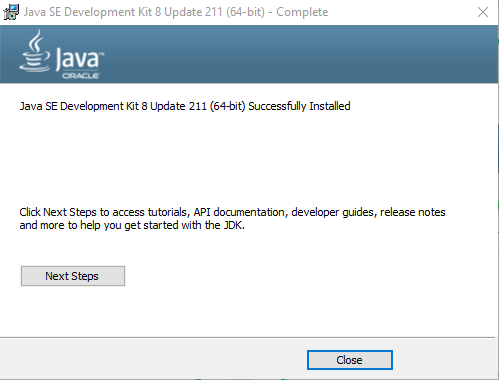


Figura 28. Finalizar la instalación Java

1. Al finalizar la instalación se concluye la misma dando clic en cerrar. Una vez se cuente con java instalado se procede con la instalación de NetBeans, su descarga se encuentra en la siguiente url:

* <https://netbeans.org/downloads/8.2/>

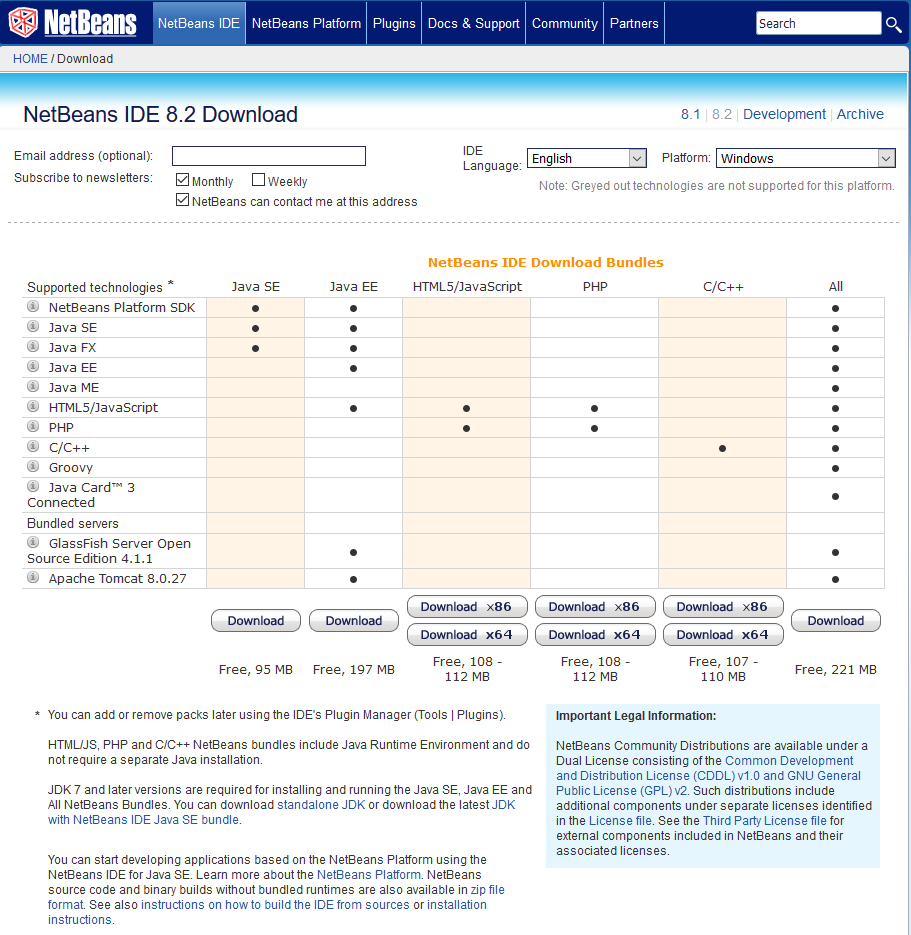


Figura 29. Instalación NetBeans

1. Dentro de esta página se selecciona el idioma del IDE, la plataforma, en la que se va a instalar por defecto Windows, y debe seleccionarse la versión completa del IDE, se encuentra en el campo All y se procede dando clic en Down load.

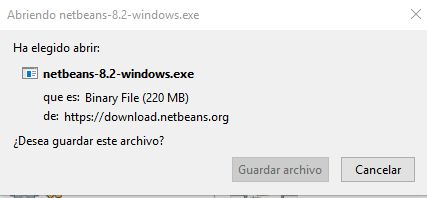


Figura 30. Guardar el archivo

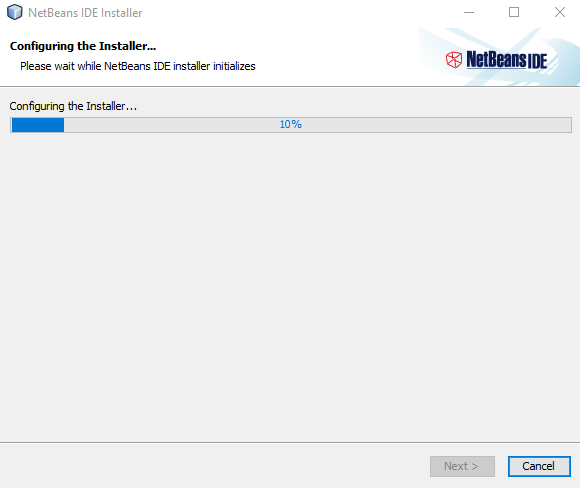


Figura 7. Configuración del instalador NetBeans

1. Al abrirse, NetBeans realiza una configuración del instalador.



Figura 8. Características a instalar NetBeans

1. Posteriormente mostrará las características a instalar, en lo que se selecciona personalizar.

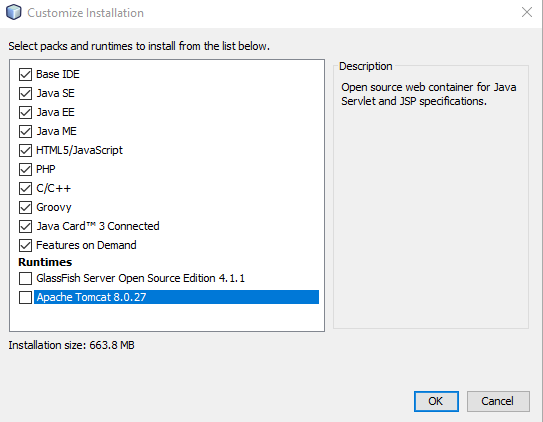


Figura 33. Selección de paquetes y servidores a instalar

1. En esta ventana se seleccionan los paquetes y servidores que se utilizaron en cuanto a paquetes, se seleccionan todos los disponibles de java y no se selecciona ningún servidor, puesto a que estos se instalarán posteriormente a la instalación de NetBeans, se da clic en ok y posteriormente en siguiente.

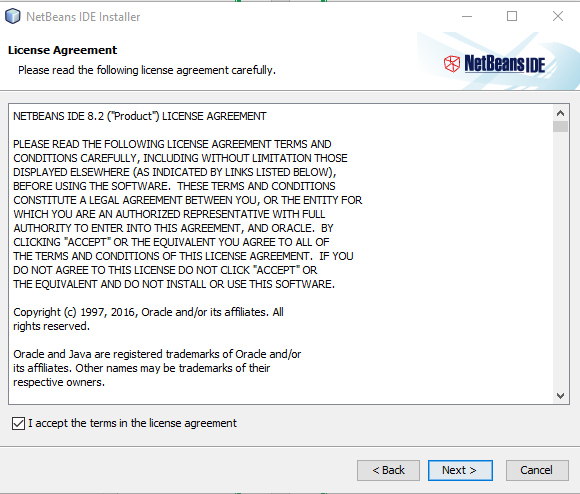


Figura 34. Aceptación de términos

1. En esta ventana se aceptan los términos y acuerdos de licencia y se da clic en siguiente.

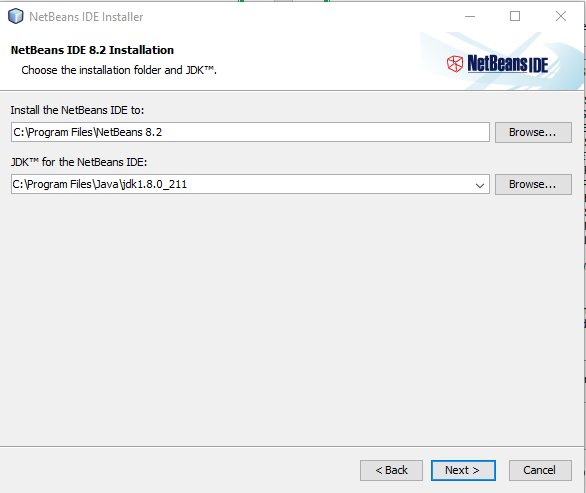


Figura 35. Selección de carpeta de instalación

1. Se seleccionan las carpetas de instalación de NetBeans y carpeta del JDK, esta última se cargará por defecto, pero en caso de que no deberá hacérselo, una vez puestas las carpetas se procede dando clic en siguiente.

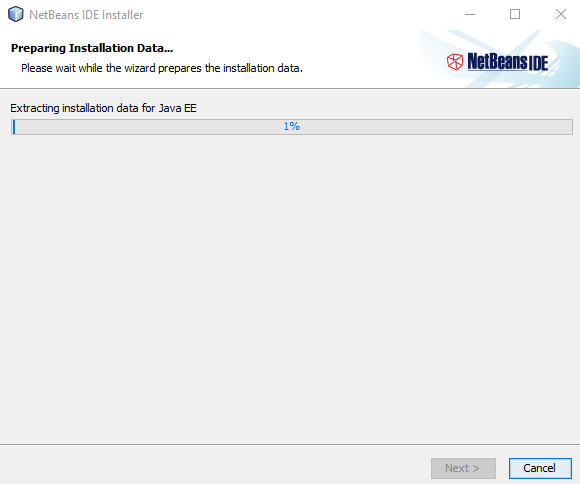


Figura 36. Instalación de paquetes seleccionados

1. Se instalarán todos los paquetes seleccionados previamente.

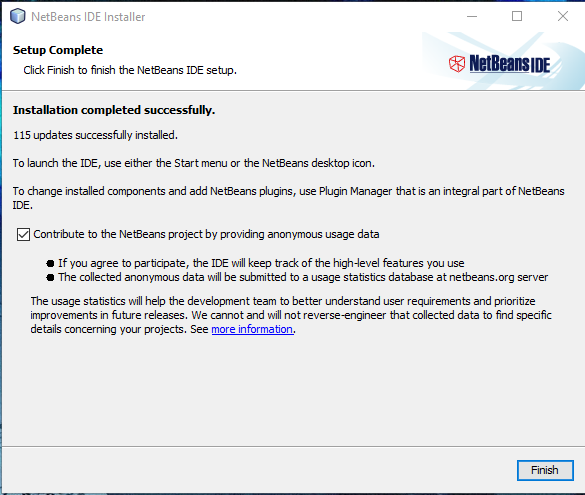


Figura 9. Finalización de la instalación

Al acabar la instalación solo se da clic en finalizar y listo, se ha concluido con la instalación de NetBeans.

NetBeans viene preconfigurado para el Servidor de Aplicaciones GlassFish 3.1 y Java DB para manejo de Sistema de Gestión de Base de Datos Relacionales.



Figura 10.

Pero se puede integrar NetBeans con otros servidores de Aplicaciones como JBoss, WebLogicBea o WebSphere y con otras bases de datos relacionales como MySql, Postgress o cualquier otra.

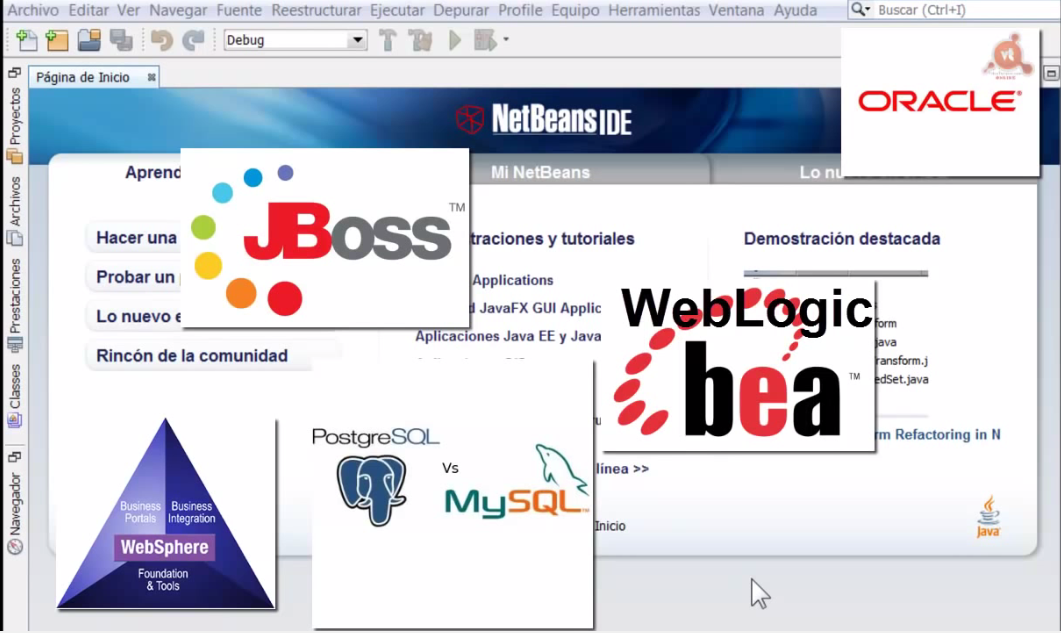


Figura 39. BDD que pueden trabajar con NetBeans

### INSTALACIÓN DE PAYARA SERVER

Payara Server es un servidor de aplicaciones de código abierto derivado de GlassFish Server Open Source Edition. Fue creado en 2014 por C2B2 Consulting como una caída en el reemplazo de GlassFish después de que Oracle anunció que suspendería el soporte comercial para GlassFish. En comparación con GlassFish, Payara Server se lanza con mayor frecuencia trimestralmente con correcciones de errores, parches y mejoras adicionales. Desde abril de 2016, Payara Server está patrocinado por Payara Services Ltd., que es responsable del desarrollo continuo y la coordinación de las contribuciones de la comunidad. Oficialmente compatible con Jakarta EE y MicroProfile.



Figura 40. Logo Identificativo de Payara

En la parte superior del menú, seleccionar Tools-> Plugins, en el buscador de Plugins disponibles escribir “Payara” para obtener los Plugins disponibles y seleccionar los siguientes Plugins:

1. Payara Server
2. Payara Common
3. Payara EE Common
4. Payara Tooling

Clic en instalar, una vez los plugins mencionados anteriormente hayan sido seleccionados.

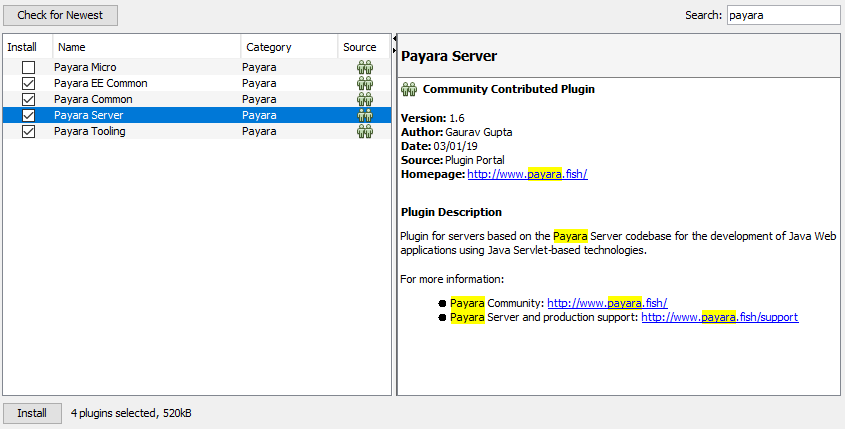


Figura 41. Instalación Drivers Payara Server

Después de la instalación del plugin, clic en finish, para reiniciar NetBeans y así los cambios se apliquen.

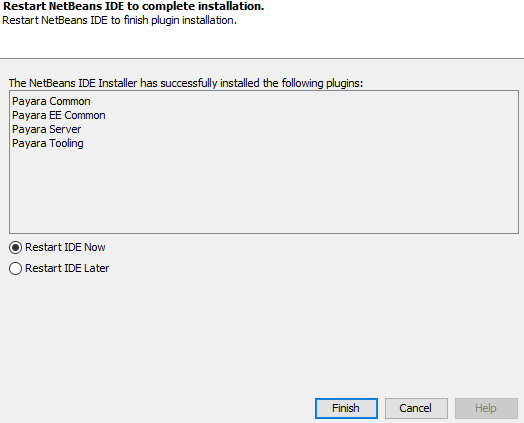


Figura 42. Instalación Drivers Payara Server

Una vez instalados los plugins de payara, se procede a instalar el servidor Payara en NetBeans previamente descargado y descomprimido dentro del equipo.

Clic en Tools-Servers y dar clic en Add Server

1. Dirigirse a la página oficial de Payara

<https://www.payara.fish/downloads/payara-platform-community-edition/>

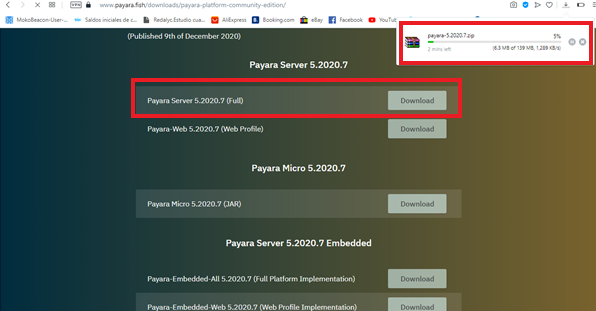


Figura 43. Descarga de Payara.

1. Pegarlo en el disco C

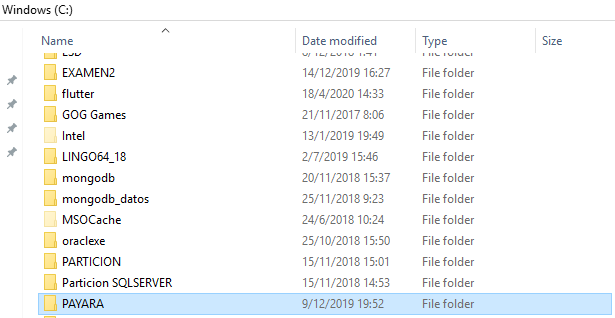


Figura 44. Pegar los archivos del Servidor Aplicaciones Payara en el Disco C

**EJECUCIÓN DE PAYARA**

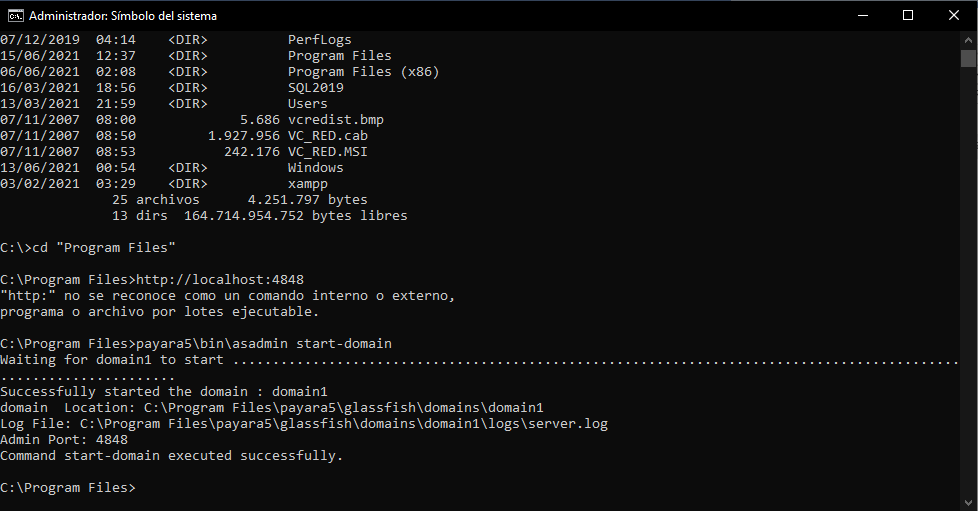


Figura 45. Dirigirse al Directorio Program Files

1. Dirigirse a NetBeans
2. Dirigirse a Servers y clic derecho Add Server

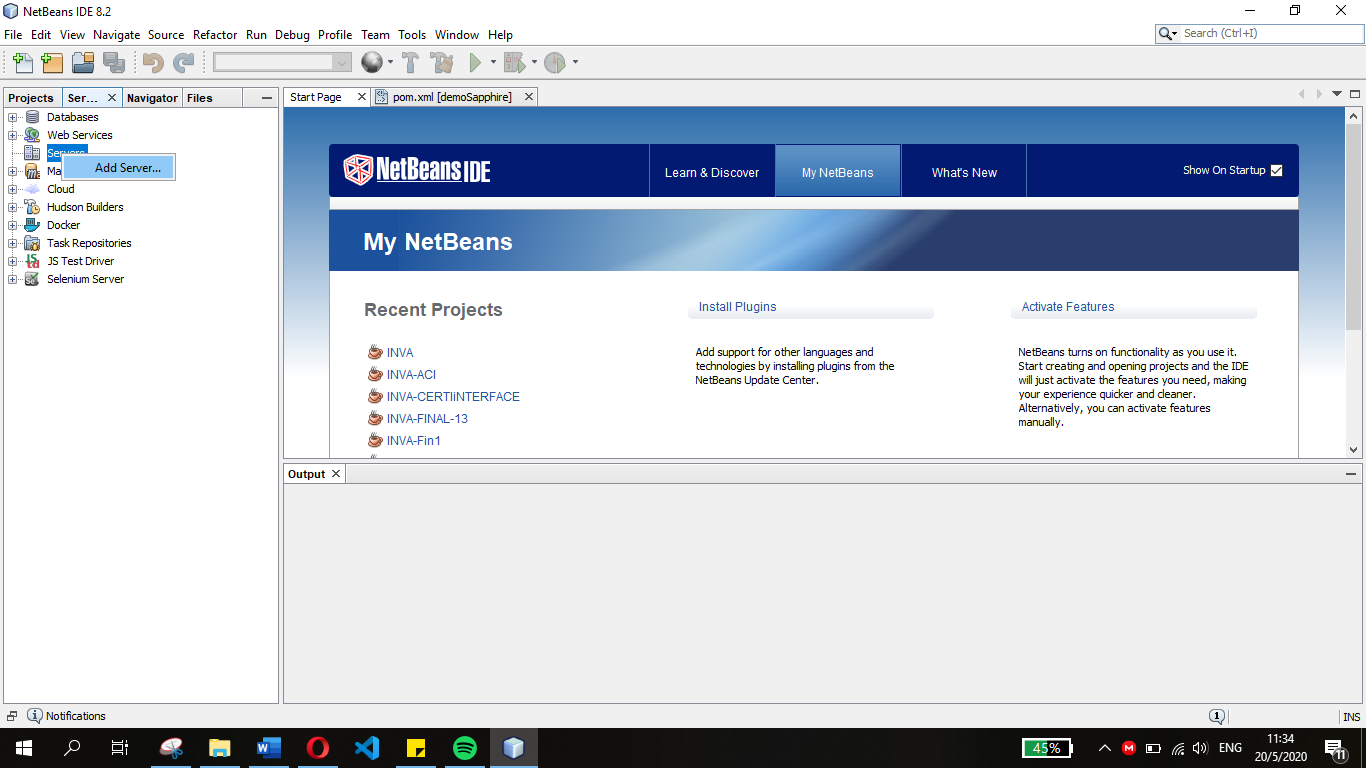


Figura 46. Add Server

1. Elegir Payara Server

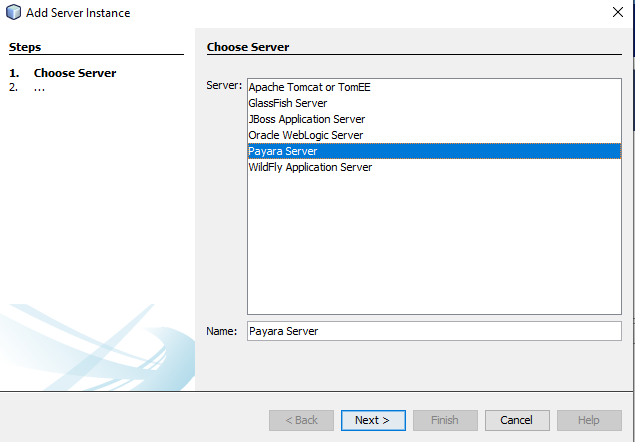


Figura 47. Insertar Payara

1. Seleccionar la ubicación del archivo descargado

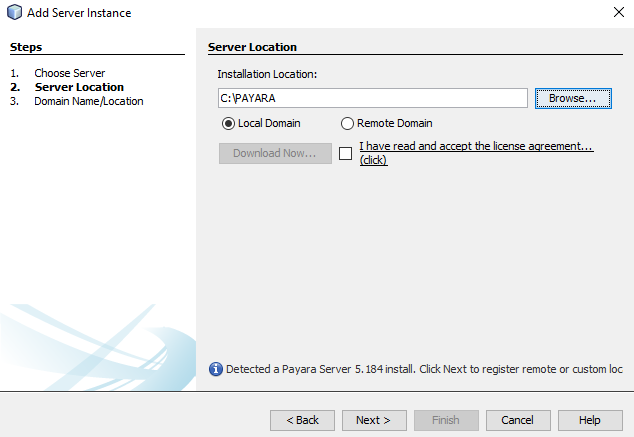


Figura 48. Ubicación de Payara en Netbeans

1. Dejar los valores por defecto y clic en Finish

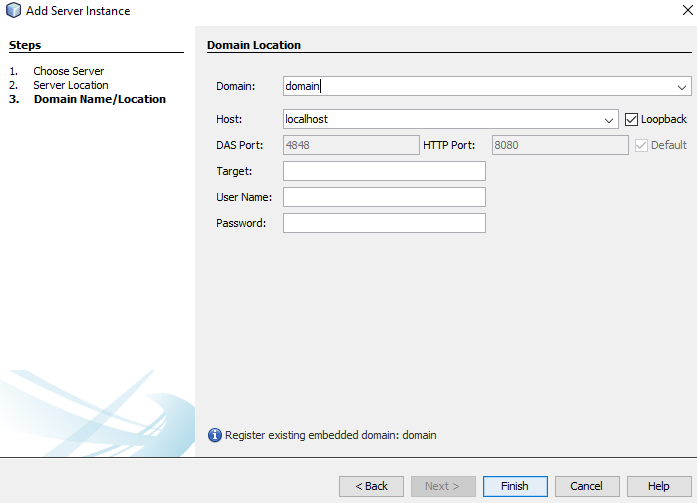


Figura 49. Dominio Payara

Ahora se probará el correcto funcionamiento del Servidor Payara. Primero en la pestaña Services->Servers, dar clic derecho y Start en la pestaña Payara Server

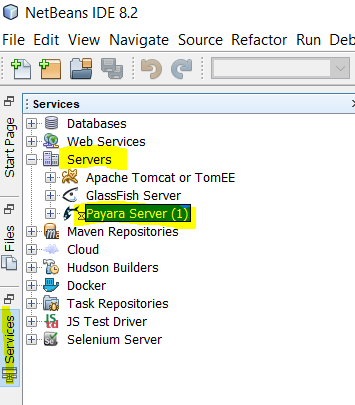


Figura 50. Verificación instalación

Se puede observar que el servidor esta funcionando correctamente.



Figura 51. Ejecución Payara Server

1. Ingresar a: <http://localhost:8080> y clic Administration Console

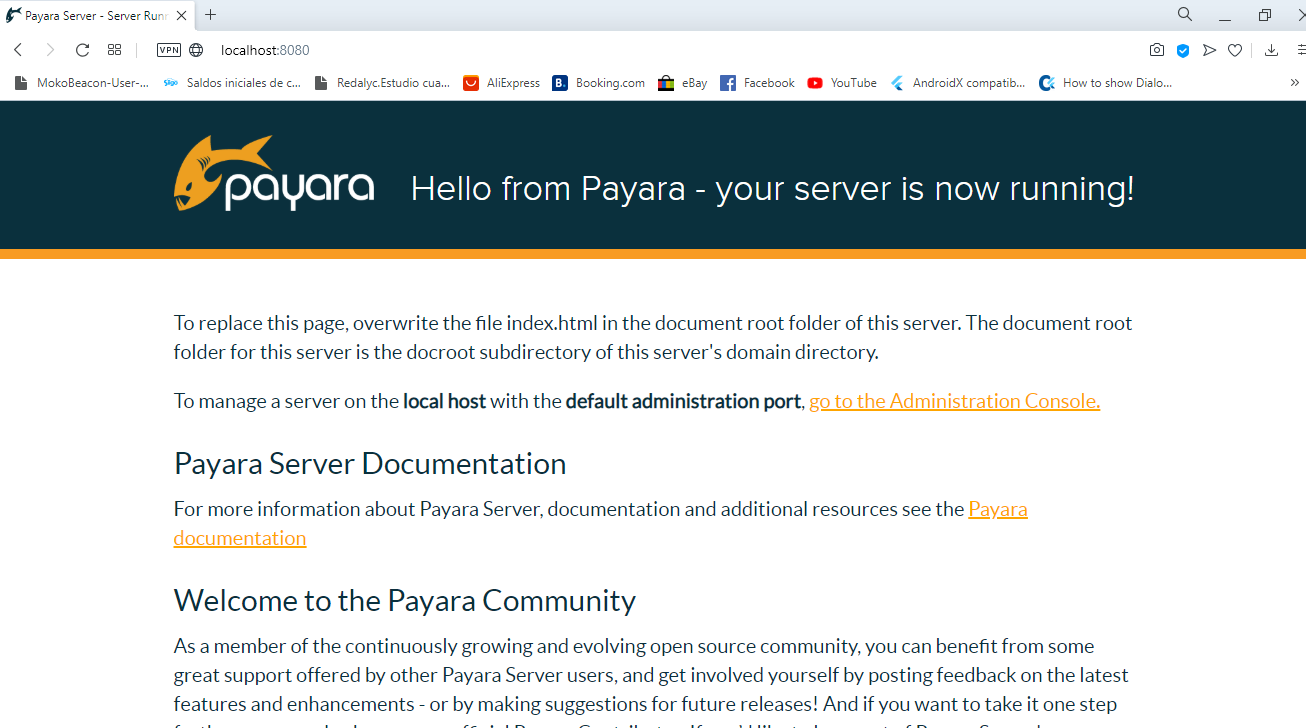


Figura 52. Sitio web de Payara

1. La consola se mostrará

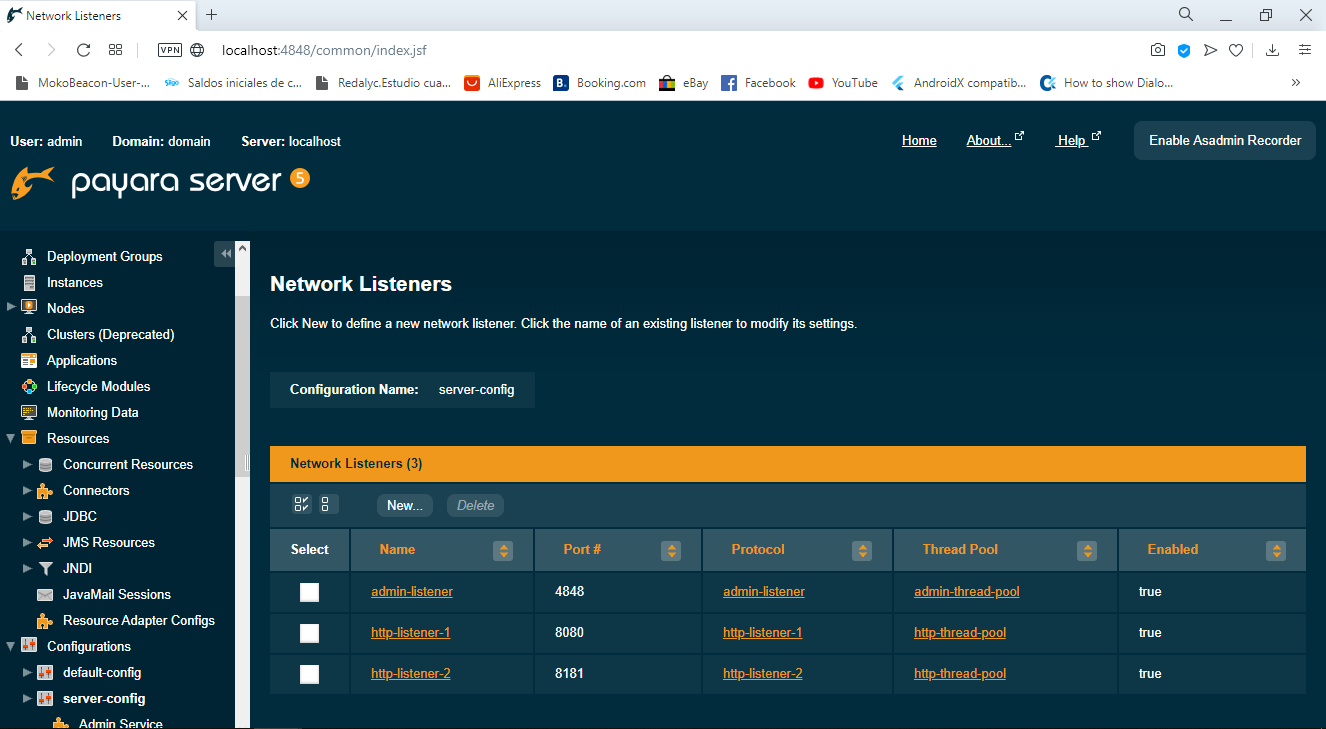


Figura 53. Admin Console de Payara

Se probará nuevamente el correcto funcionamiento del servidor, mediante la ejecución de una aplicación de ejemplo que viene por default en netbeans.

Para ello dar clic en nuevo proyecto->Samples->Web Services->Calculator

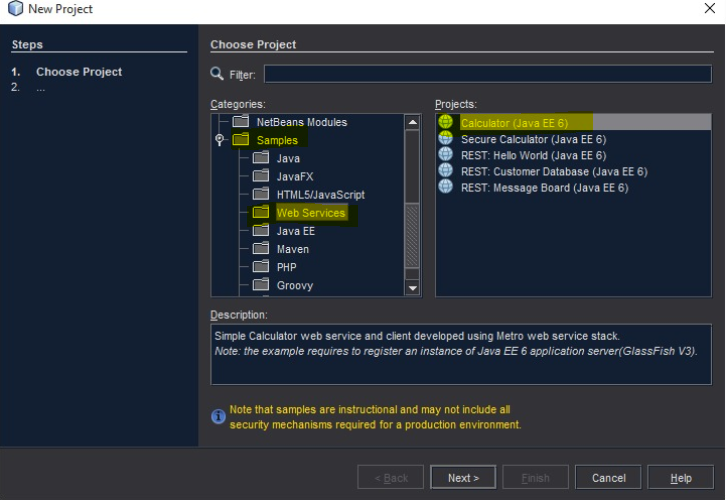


Figura 54. Ejecución Payara Server

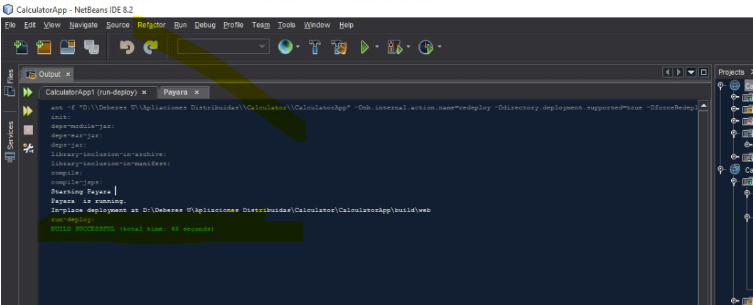


Figura 55. Ejecución Payara Server

Se puede comprobar nuevamente que el servidor funciona correctamente.

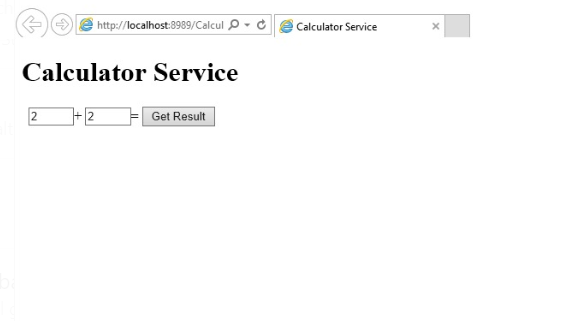


Figura 56. Ejecución Payara Server

### INSTALACIÓN DE MYSQL

A continuación, se va a integrar un RDBMS, por default viene JDBC, pero se puede integrar cualquier otro.

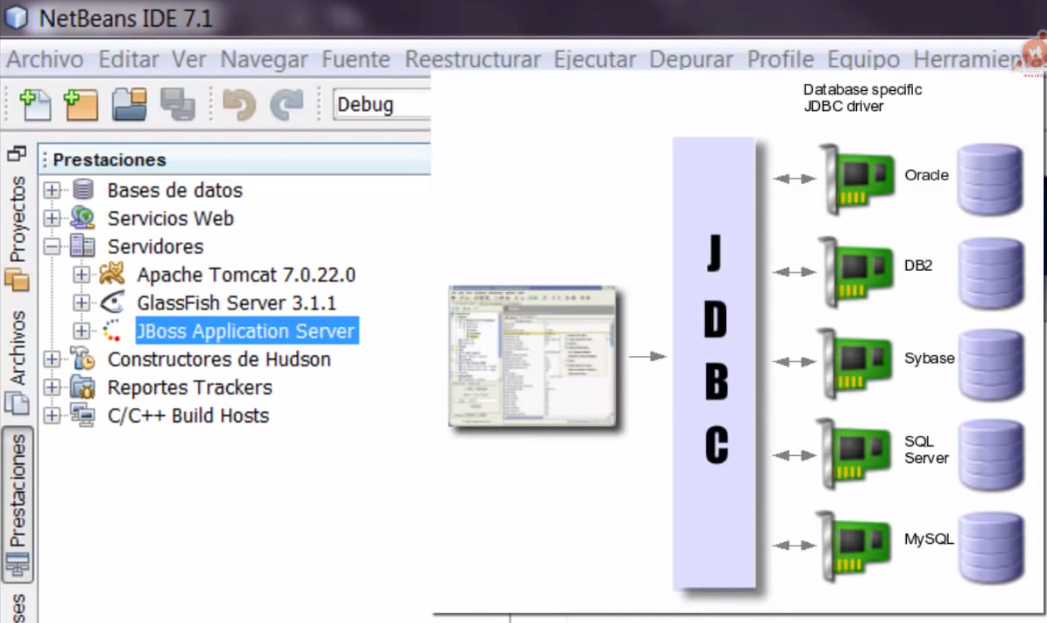


Figura 57. Integrar un driver para un RDBMS en NetBeans.

NetBeans también incluye driver para los principales Gestores de Bases de Datos Relacionales como se muestra en la figura

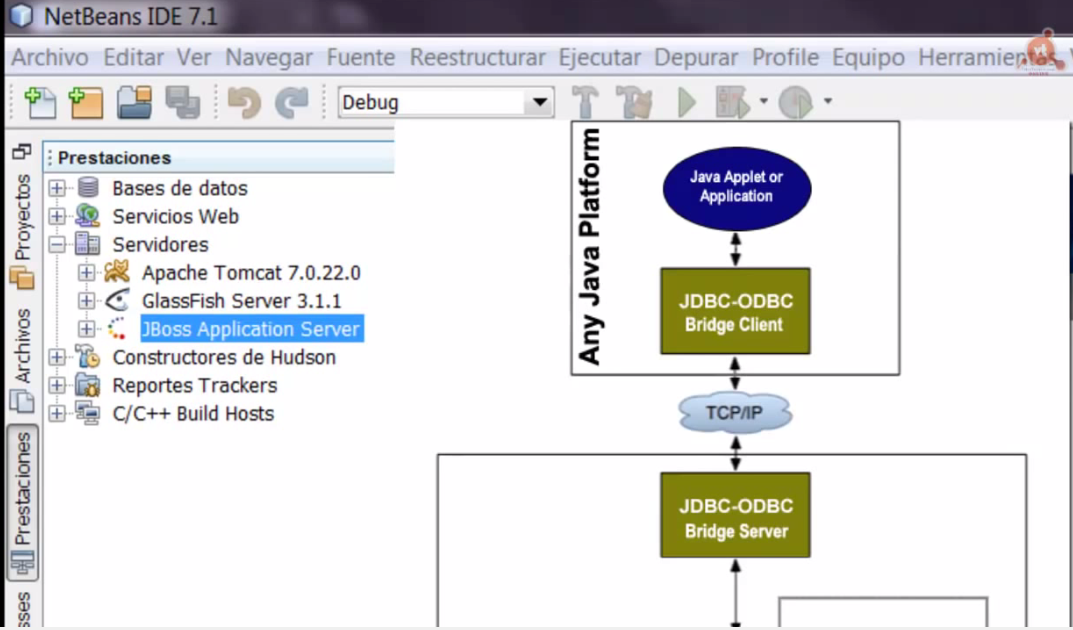


Figura 58. Integrar un driver para un RDBMS en NetBeans.

Se va a conectar al RDBMS de MySQL o María DB

1. Teniendo todo lo necesario para la programación de la aplicación, se requiere la instalación de un sistema gestor de base de datos, en este caso MySQL, se comienza dirigiéndose a la siguiente dirección:

* <https://dev.mysql.com/downloads/windows/installer/8.0.html>

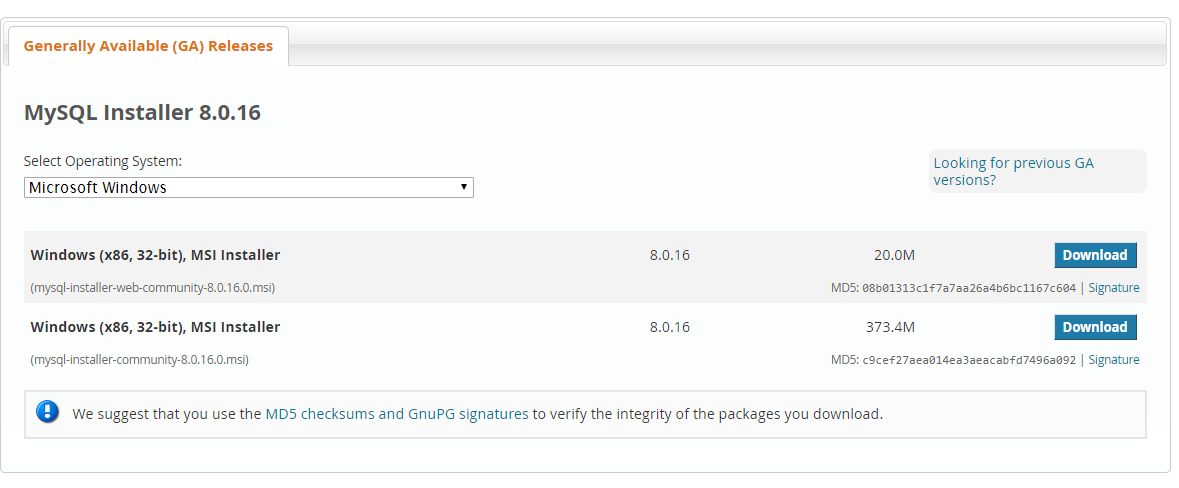


Figura 59. Selección del sistema operativo

1. Se selecciona el sistema operativo y clic en descarga la versión completa con 373.4M. Con la descarga finalizada, se procede a ejecutar el instalador, en caso de requerirse se conceden permisos de usuario.

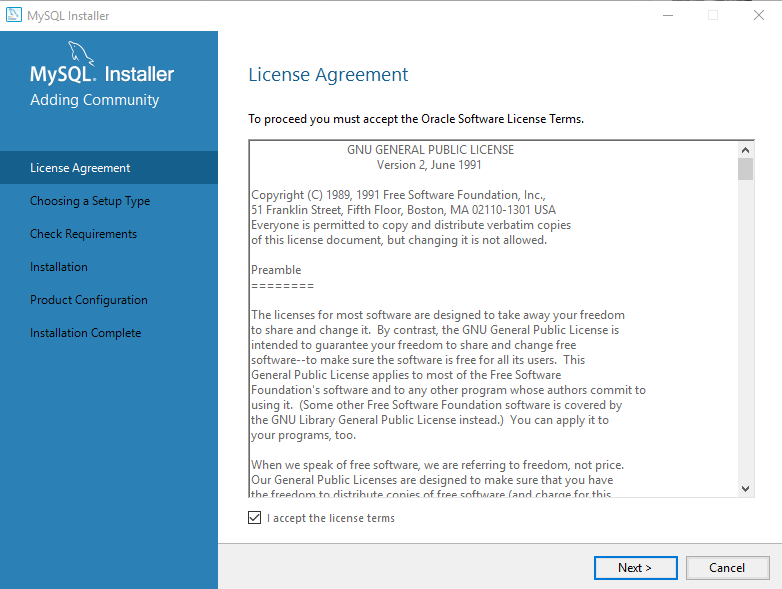


Figura 60 Acuerdo de licencia MySQL.

1. En la primera ventana se aceptan los términos y condiciones y clic en siguiente.

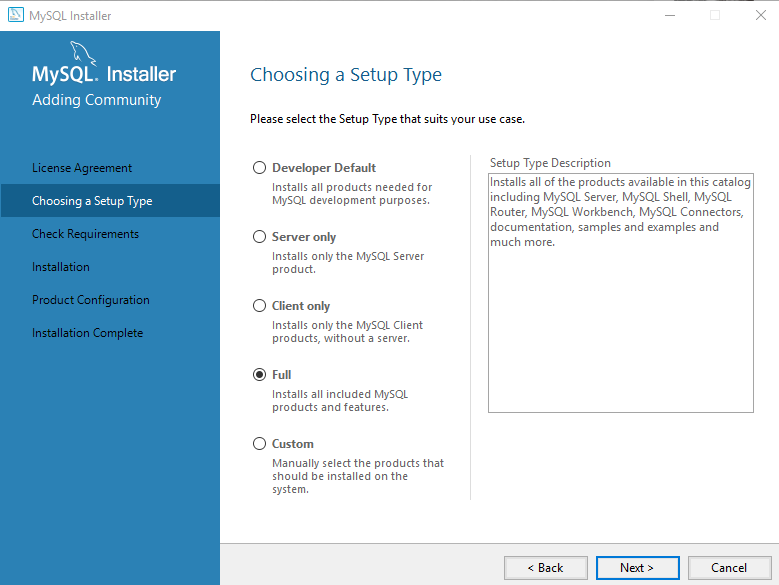


Figura 61. Selección del tipo de instalación MySQL.

1. En la siguiente se selecciona el tipo de instalación Full, para obtener las características completas de MySQL, de requerirse, el siguiente paso mostrará una serie de características adicionales que se requieren y como instalarlas, se debe hacerlo.

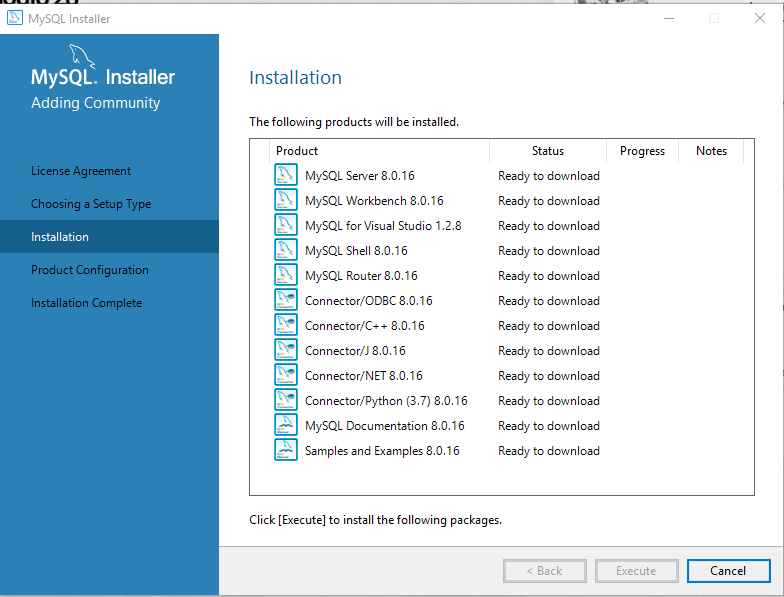


Figura 62. Paquetes MySQL a instalarse.

1. En la siguiente ventana se da clic en el botón ejecutar y la instalación comenzará.

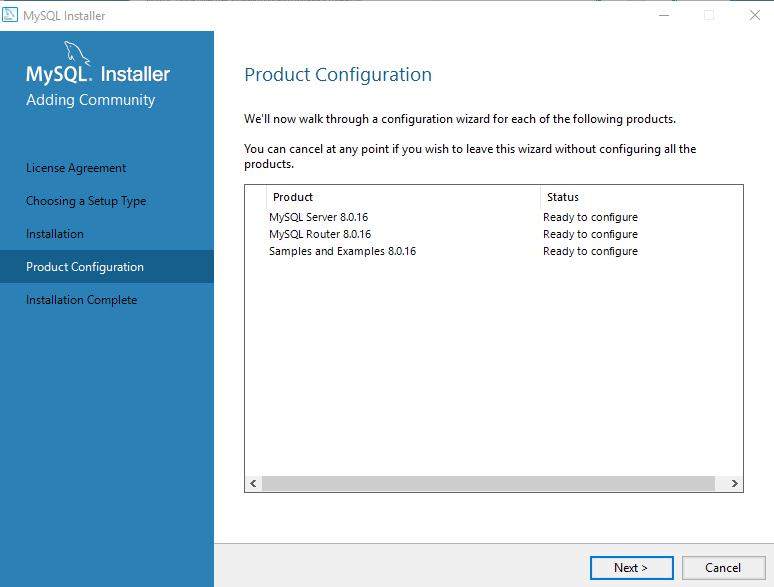


Figura 63 Configuración del producto.

1. Clic en siguiente.

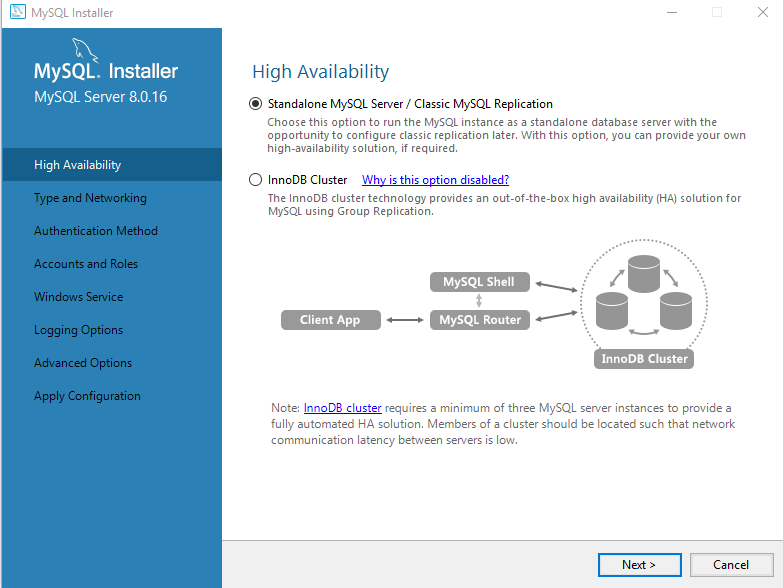


Figura 64 Configurar la disponibilidad MySQL.

1. En esta ventana es posible configurar la disponibilidad, sea con la clásica o InnoDB, se selecciona la adecuada y clic en siguiente.

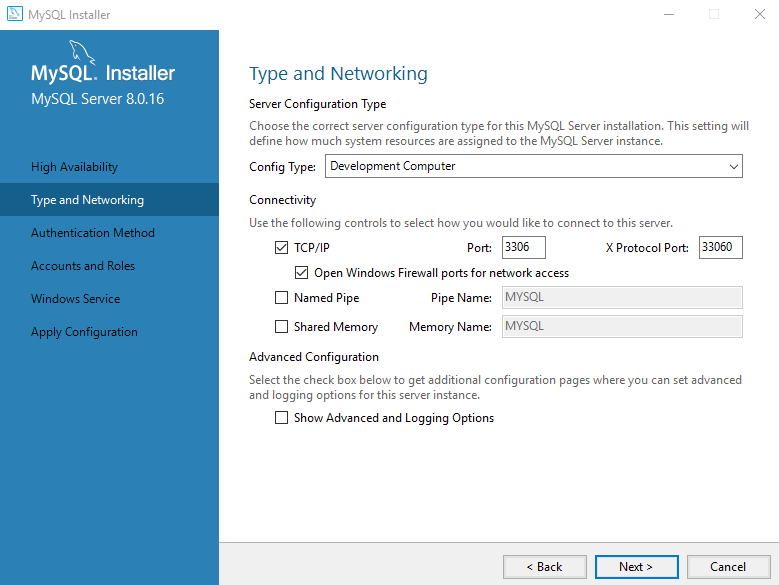


Figura 11 Configuraciones de conectividad.

1. En la siguiente ventana es posible cambiar algunas configuraciones de conectividad, normalmente se quedan por defecto, pero cada usuario puede cambiarlas de acuerdo a lo que necesite, se procede con clic en siguiente.

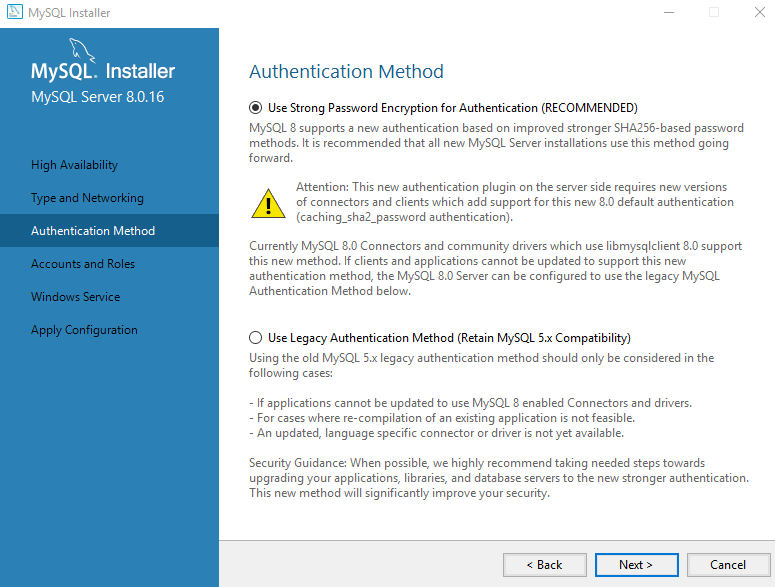


Figura 66 Selección del método de autenticación.

1. Seleccionar el método de autenticación, se procede dejando el recomendado, pero es posible utilizar el otro, pero compatible con versiones 5X.

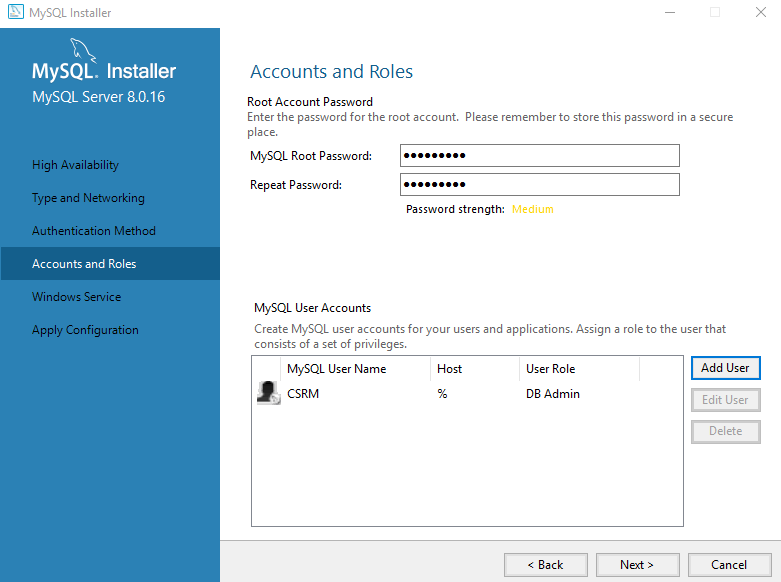


Figura 12. Agregar usuarios y contraseñas.

1. En la siguiente ventana se colocará la contraseña de la aplicación, de igual forma es posible añadir uno o más usuarios con su nombre, host, rol y contraseña, esto de acuerdo a lo que se necesite. Posteriormente clic en siguiente.



Figura 68. Ventana cambiar configuraciones del servicio.

1. Posteriormente se pueden cambiar unas configuraciones del servicio, esto se dejará por defecto y se procede con la instalación, clic en siguiente.

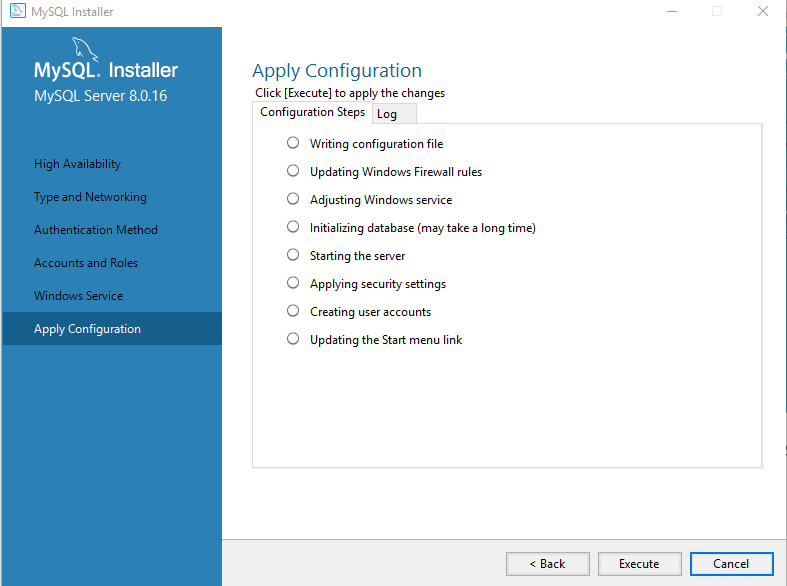


Figura 69. Ventana ejecución de las configuraciones realizadas.

1. En esta ventana se procede ejecutando las configuraciones realizadas anteriormente.

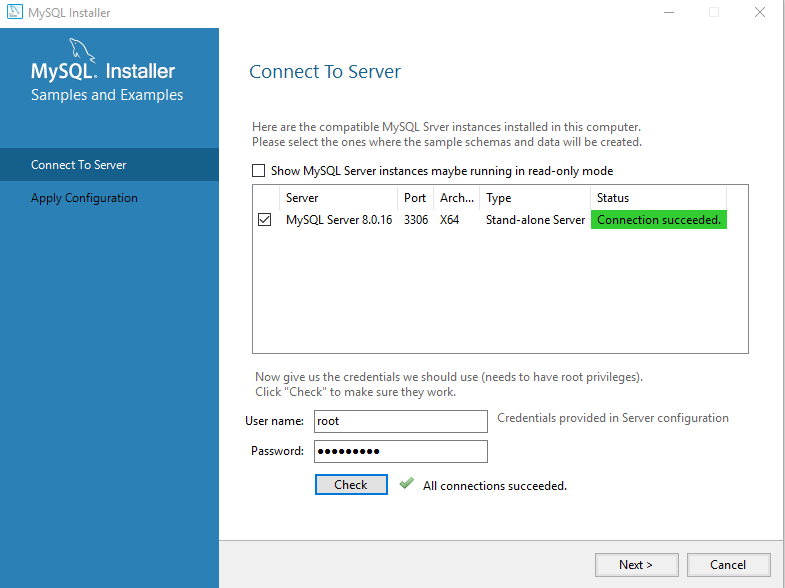


Figura 70. Ventana de conexión con el servidor.

1. Para el siguiente paso se debe verificar la conectividad, para esto se coloca el nombre de usuario + la contraseña y se chequea si la conexión ha sido exitosa, clic en siguiente.

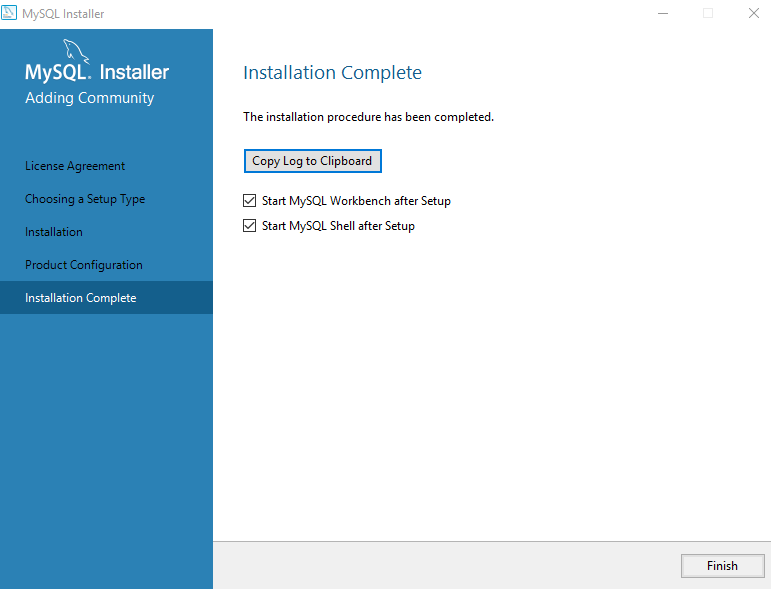


Figura 71.13 Ventana de instalación completa.

Se pueden seleccionar las casillas de iniciación de la línea de comandos de MySQL o Workbench, clic en finalizar, y con eso se concluirá la instalación de la herramienta.

### 

### POOL DE CONEXIÓN, BASE DE DATOS MYSQL – PAYARA

#### CREACIÓN DE LA CONEXIÓN

1. En Netbeans, click derecho sobre las **DataBases**



Figura 72. Configurar base de datos

1. Crear nueva conexión de base de datos, escoger MySQL y el driver correspondiente, preferible el conector de java 5.1.47 que mejor se adapta a la versión de base de datos 5.7 y la de ***Payara 5.182***

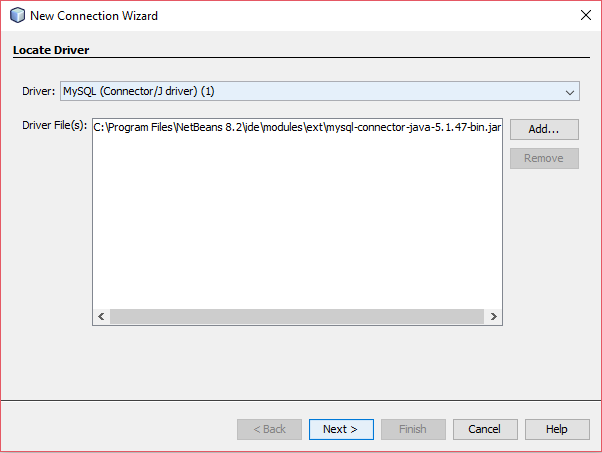


Figura 14Configurar base de datos

1. Configurar la nueva conexión, copiar la ***JDBC URL,*** necesaria para el pool de conexión.

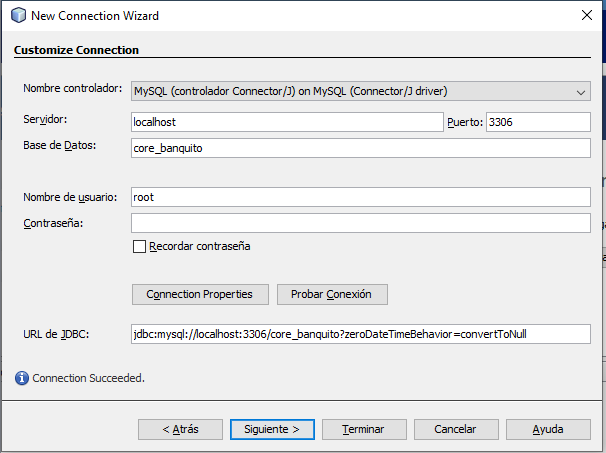


Figura 74. Configurar base de datos

1. Enlace con la Base de Datos que usted haya seleccionado



Figura 75. Configurar base de datos

#### CREACIÓN DEL POOL DE CONEXIONES

1. Se crea el Pool de conexión y se pone el nombre de preferencia
2. Clic en next

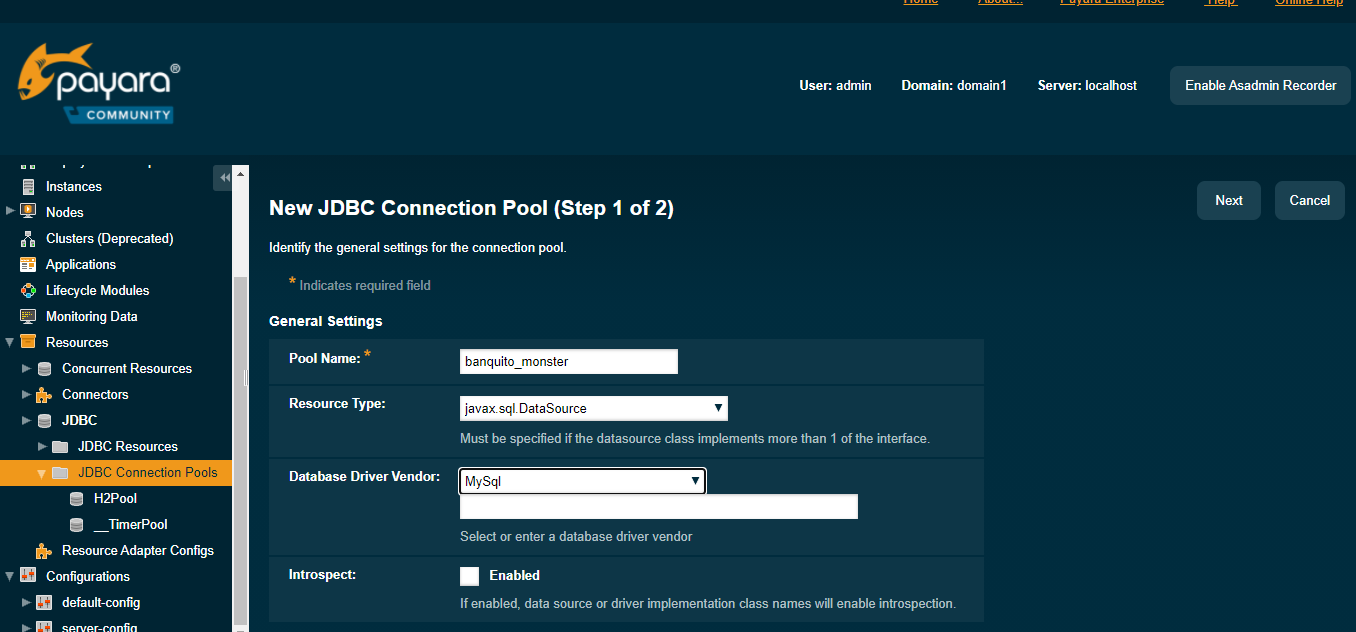


Figura 76. Creando Pool de conexiones

1. Añadir las propiedades adicionales y clic en save

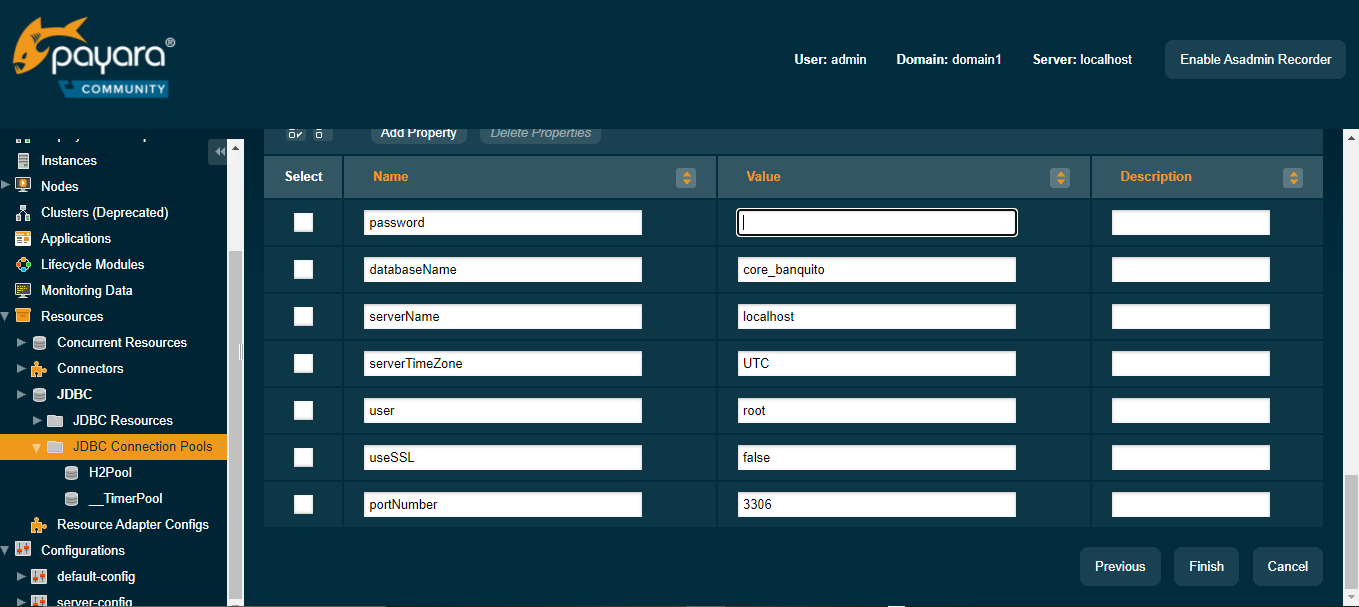


Figura 77. Connection Pool Properties

1. Dirigirse al Pool Connection Creado
2. Dar clic en Ping

Si da **Ping Succeded** quiere decir que la conexión fue exitosa

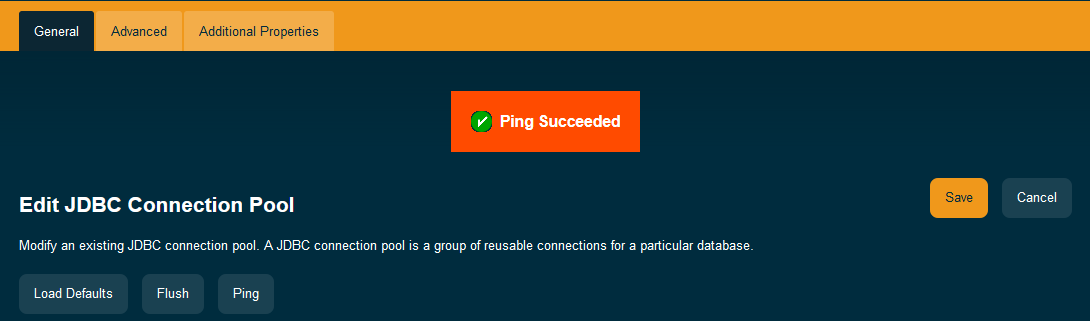


Figura 78. Propiedades del Pool de Conexiones.

## Comandos útiles dentro de Java

Dentro del desarrollo común en Java, existenvarios comandos útiles, que permiten poder optimizar tiempo al momento de escribir líneas de código:

Tabla 1. Resumen comandos útiles dentro de Java

|  |  |
| --- | --- |
| Comando | Función |
| Alt+insert | Generar código que se usa muy frecuentemente, por ejemplo constructores, getters, setters, etc. |
| Alt+Enter | Útil para enmendar errores de código |
| Mayus+ Alt+O | Útil cuando se trabaja con proyectos con varios archivos, permite buscar el nombre de un archivo concreto. |
| Alt+F7 | Se abre una ventana en la que se puede buscar dentro de un archivo una línea de sintaxis en específico. |
| Ctrl + f12 | Cuando se trabaja con archivos muy grandes, acceder a un esquema del archivo y a la documentación de java doc, siempre que el archivo la contenga. |
| Alt + f12 | Árbol de jerarquía de la clase donde nos encontramos en ese momento. |

Además, existe un sinnúmero de comandos dentro de la documentación de NetBeans que pueden resultar de interés al momento de escribir código y que sin duda serán de gran ayuda acorde a las necesidades del desarrollador.

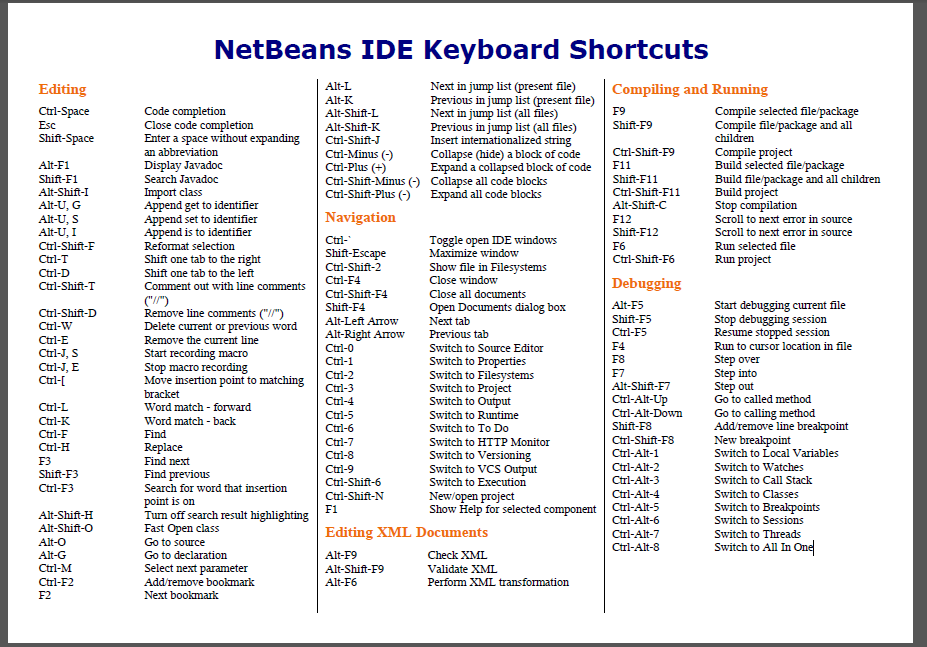


Figura 79. Comandos útiles dentro de Java

## CONCLUSIONES

* El servidor de aplicaciones GlassFish dentro del desarrollo en Java, ha sido reemplazado por Payara, ofreciendo mayor robustez, soporte y actualizaciones, permitiendo así desarrollar y testear aplicativos eficientemente.
* Java EE en sus versiones más recientes busca simplificar el desarrollo en J2EE que tradicionalmente ha sido muy complejo.

## RECOMENDACIONES

* Antes de realizar la instalación de Power Designer se debe tomar en cuenta ciertas características que debe poseer el equipo para que el software a ser instalado funcione de una manera óptima y no presente inconvenientes a la hora de su utilización, como requisitos mínimos de sistema:
  + Microsoft Windows 7, 8, 8.1 o 10, o en el caso de Microsoft Windows Server 2008 o 2012 (32 bits o 64 bits)
  + Procesador de 2 GHz
  + 3 GB de RAM
  + Adaptador gráfico SVGA o de mayor resolución y monitor de color compatible (800x600)
  + 1 GB de espacio en disco para la instalación de todos los módulos principales (y vídeos de demostración).
* NetBeans es uno de los entornos de programación más utilizados a la hora de construir programas en JAVA. Para descargar e instalar NetBeans es necesario haber instalado primero el JDK.
* MySQL es una de las bases de datos de código abierto más populares y utilizadas en todo el mundo, especialmente para el desarrollo de entornos web, aunque actualmente pertenece a Oracle, fue una empresa llamada MySQL AB la que inició el desarrollo de MySQL. Más tarde esta empresa fue adquirida por Sun Microsystem que a su vez fue comprada por Oracle, la actual propietaria. Este juego de compras de empresas seguro que te puede dar una idea de la importancia estratégica de MySQL para las compañías dedicadas a las bases de datos. MySQL, aunque puede utilizarse en una amplia gama de aplicaciones, se asocia más a menudo con las aplicaciones web y la publicación en línea.

## BIBLIOGRAFÍA

* Payara for Beginners—Adding Payara Server to NetBeans. (s. f.). Recuperado 24 de mayo de 2020, de <https://blog.payara.fish/adding-payara-server-to-netbeans>
* 01.- Curso Java EE 6. Configuración del Entorno de desarrollo (2012). Rcuperado el 24 de mayo de 2020 de: <https://www.youtube.com/watch?v=V23GNjnf438&t=73s&pbjreload=10>