

## Tarea 3: aplicación completa



Simón Vallejo  
Sofía Merino  
Raúl Requena  
Cristina Díaz  
Alejandro Aguayo



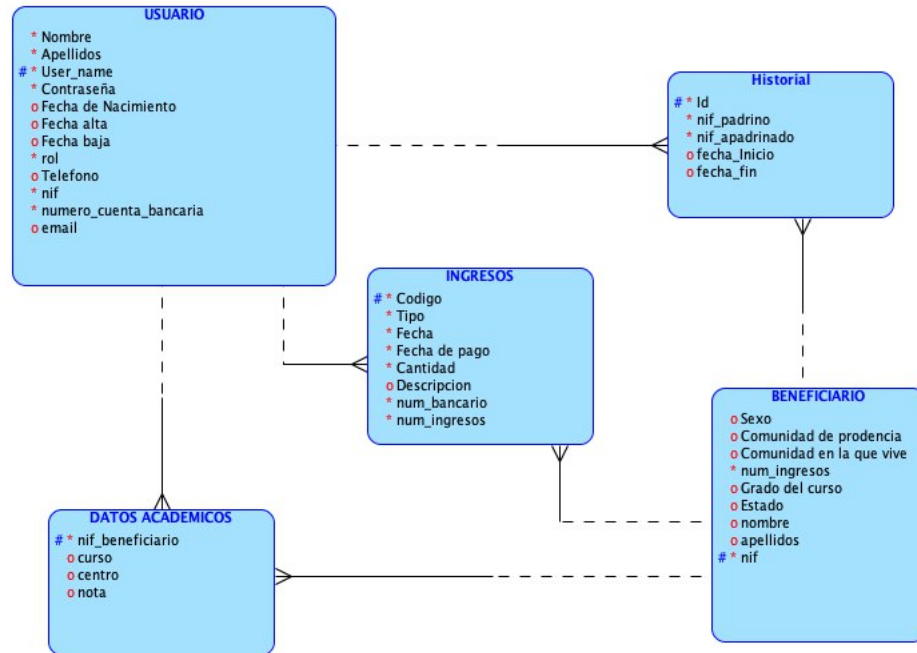
# Índice

Índice general	<b>1</b>
<b>1. Modificaciones a entregas anteriores</b>	<b>2</b>
1.1. Entrega 1 . . . . .	2
1.2. Entrega 2 . . . . .	6
<b>2. Entrega 3</b>	<b>22</b>
<b>3. URL a GIT</b>	<b>23</b>

# 1. Modificaciones a entregas anteriores

## 1.1. Entrega 1

### Modelo Entidad-Relación



Este es el modelo entidad-relación de nuestra aplicación. Debido a que el anterior modelo contenía múltiples fallos, nos hemos visto obligados a modificarlo y este es el resultado. Como vemos. Posee una entidad usuario con sus respectivos atributos y como clave primaria el nombre del usuario. Esta entidad está relacionada con las entidades Historial, ingresos y datos académicos.

Es muy importante el atributo rol de esta entidad ya que distingue entre socios y trabajadores. Por ejemplo, en el caso de la relación uno a muchos entre la entidad Usuario y la entidad Historial indica que un usuario puede tener muchos historiales pero un historial tiene que pertenecer a un solo usuario. Aquí, es donde el rol juega un papel muy importante, ya que en este caso solo tendrán historiales los que tengan el rol de socio.

El caso contrario para en el caso de los datos académicos, en el cual solo los trabajadores podrán añadir estos datos. La relación es uno a muchos ya que un trabajador podrá añadir varias notas a sus alumnos, pero una nota solo puede ser añadida por un trabajador y no por varios.

Con la entidad Datos Académicos lo que se pretende es almacenar las notas del

curso de cada alumno. En nuestro caso, hemos utilizado una nota global de todo el curso para evaluarlo. Otra opción era añadir un atributo llamado asignatura que fuera clave compuesta con el nif del usuario. Esta opción es mejor que la que hemos elegido, pero por problemas a la hora de implementarlo hemos tenido que optar por esta opción.

La entidad Historial se aspira a conseguir gestionar los apadrinamientos de los niños. Como clave primaria una id para cada historial, esta entidad almacena los NIFs tanto de los socios como de los niños para tenerlos relacionados.

La entidad ingresos gestiona las donaciones por parte de los padrinos a los Niños. A cada ingreso se le asigna un código aleatorio que será la clave primaria, además se almacenaran los datos de las cuentas tanto del socio como del niño, así como, la cantidad, la fecha de realización y la fecha en la que se realizara el pago. La relación uno a muchos entre Usuario e Ingresos indica que los usuarios pueden realizar varios ingresos, pero no pueden realizar un único ingreso entre varios usuarios. Mientras que la relación Ingresos-Beneficiario muestra que los niños pueden recibir varios ingresos pero un ingreso no puede ir destinado a varios niños.

Para terminar, la clase Beneficiario cuenta con el nif como clave primaria y con num\_ingresos como atributo importante ya que almacena la cuenta en la que el niño recibirá el dinero.

Los cambios del modelo con respecto a la entrega anterior, han sido los siguientes:

- Hemos añadido la entidad Usuario.
- Hemos añadido las relaciones en las entidades JPA.
- Modificación de la relación entre ingresos y socios para que un ingreso no pertenezca a varios socios.
- Getters/Setters incluidos en las entidades.

Con respecto a los requisitos anteriormente entregados, estos son los que se han conseguido implementar:

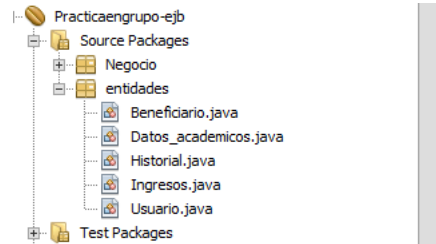
RF 1: Creación, lectura de los datos de usuarios y beneficiarios.

RF 2: Registro y lectura de ingresos.

RF 3: Gestión de datos académicos: Inserción de datos y notas, así como, la lectura de los distintos datos del alumno.

RF 4: Gestión de historial de apadrinamiento: Inserción de datos de los apadrinamientos.

## Java Persistence API (JPA)



En este apartado vamos a explicar todas las entidades creadas para la conexión de la base de datos.

Primero tenemos Beneficiario, clase que gestiona a los niños de la base de datos.

```

    ^
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long nif;
    @Column(nullable = false)
    private String nombre;
    @Column(nullable = false)
    private String apellidos;
    @Column(nullable = false)
    private String sexo;
    @Column(nullable = false)
    private String comunidadProcedencia;
    @Column(nullable = false)
    private String comunidadResidencia;
    @Column(nullable = false)
    private Integer numIngreso;
    @Column(nullable = false)
    private String grado;
    @Column(nullable = false)
    private String estado;
    @OneToOne(mappedBy = "beneficiario")
    private List<Historial> historiales;
    @OneToOne(mappedBy = "beneficiario")
    private List<Ingresos> ingresos;
    @OneToOne(mappedBy = "beneficiario")
    private List<Datos_academicos> datos_academicos;
```

En ella se describen los distintos atributos que tiene los niños, mencionando que ninguno puede ser nulo al ser metidos en la base de datos y el Nif debe ser único. Debajo de los atributos tenemos las relaciones de las distintas entidades en la base de datos, en este caso tiene relación uno a muchos con la entidad Historial, Ingresos y con Datos Académicos.

Los distintos get y set necesarios para cada atributo y sus propios métodos hashCode(), equals() y toString().

```

@Id
private String nif_beneficiario;
@Id
private String asignatura;
@Column(nullable = false)
private String curso;
@Column(nullable = false)
private String centro;
@Column(nullable = false)
private long nota;
@ManyToOne
private Usuario usuario;
@ManyToOne
private Beneficiario beneficiario;

```

Como segunda entidad tenemos Datos Académicos la cual es la encargada de guardar todos los datos relacionados con los estudios de los distintos niños. Donde el nif del niño y la asignatura son únicos. En cuanto a las relaciones con otras entidades tenemos una muchos a uno con Usuario y otra igual con beneficiario.

```

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
@Column(nullable = false)
private String nifPadrino;
@Column(nullable = false)
private Long nifApadrinado;
@Column(nullable = false)
private String fechaInicio;
@Column(nullable = false)
private String fechaFin;
@ManyToOne
private Beneficiario beneficiario;
@ManyToOne
private Usuario usuario;

```

La entidad Historial es una entidad intermedia entre las entidades Usuario y Beneficiario, la cual se utiliza para almacenar los niños apadrinados y sus padrinos.

```

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long codigo;
@Column(nullable = false)
private Integer numCuentasBancarias;
@Column(nullable = false)
private Integer numIngresos;
@Column(nullable = true)
private String tipo;
@Temporal(TemporalType.DATE)
@Column(nullable = true)
private Date fecha;
@Column(nullable = false)
private String fechaPago;
@Column(nullable = false)
private Integer cantidad;
private String description;
@ManyToOne
private Usuario usuario;
@ManyToOne
private Beneficiario beneficiario;

```

Estos son los atributos de la entidad Ingresos y como se han generado sus distintas relaciones.

```

    @Id
    private String username;
}

@Column(nullable = false)
private String nombre;
@Column(nullable = false)
private String apellidos;
private String email;
private String telefono;
@Column(nullable = false)
private String contrasenia;
@Column(nullable = false)
private String NIF;
private String fechaNacimiento;
private String fechaAlta;
private String fechaBaja;

private String numCuentaBancaria;
private String direccion;

private Rol rol;

@OneToMany(mappedBy = "usuario")
private List<Ingresos> ingresos;
@OneToMany(mappedBy = "usuario")
private List<Historial> historiales;
@OneToMany(mappedBy = "usuario")
private List<Datos_academicos> Datos_academicos;

```

Por último tenemos la entidad Usuario que gestiona los datos tanto de los socios como de los trabajadores de la asociación. Mencionando que el Usuario es la clave principal y por tanto debe ser único.

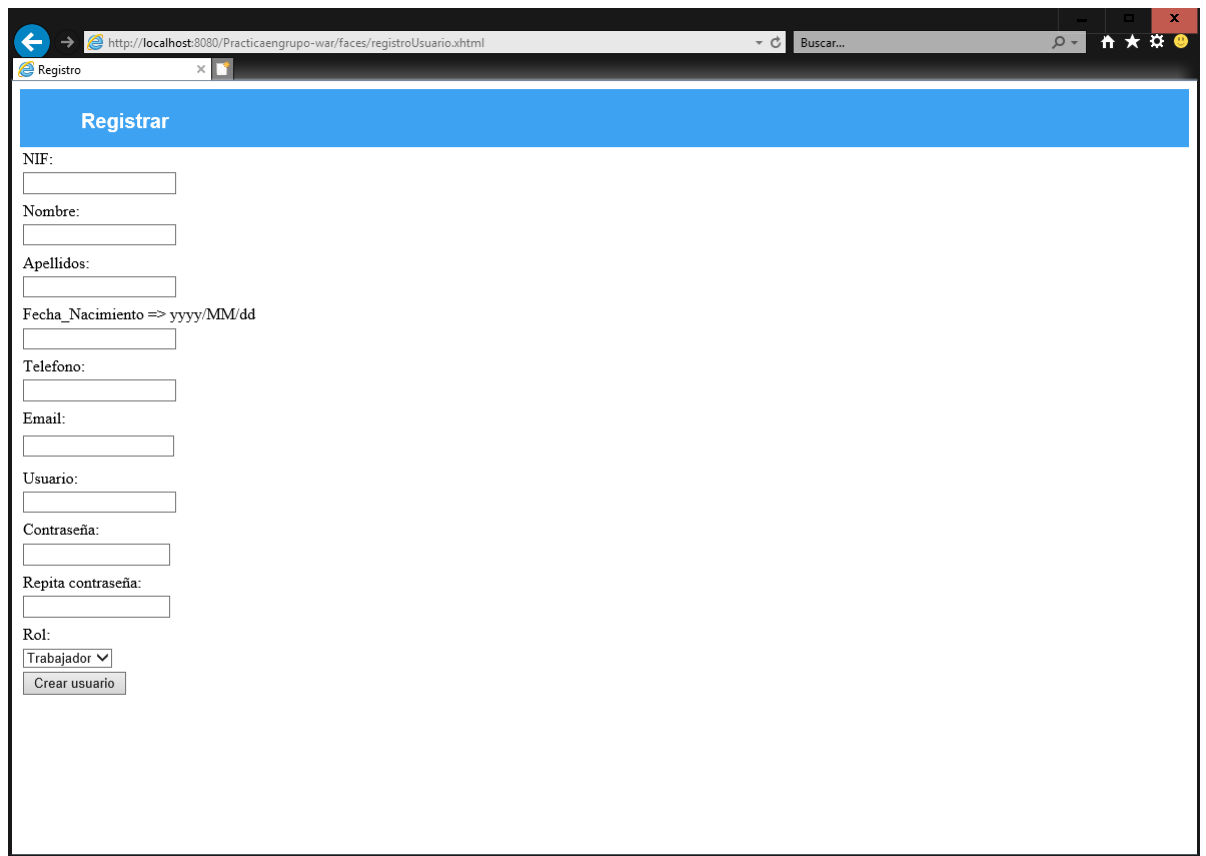
## 1.2. Entrega 2

Las navegación entre las diferentes páginas es la siguiente:





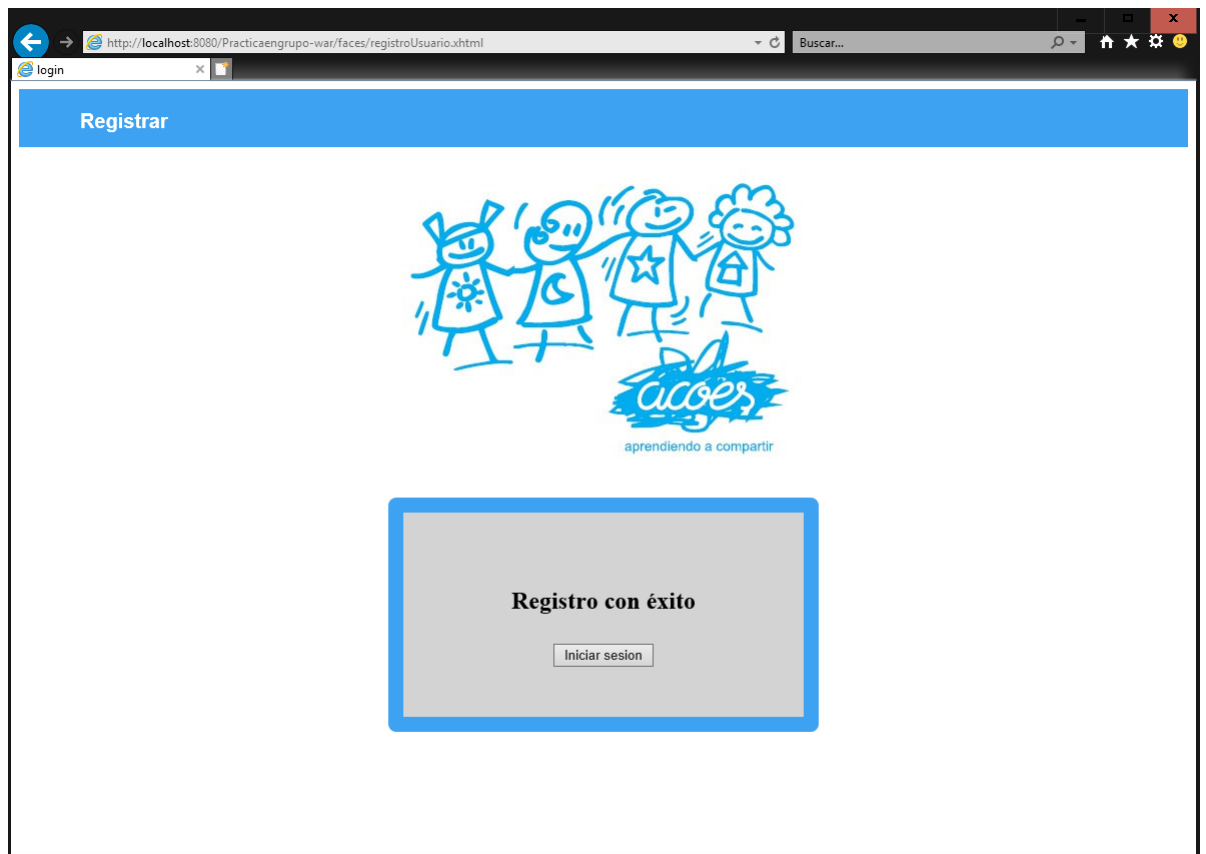
Empezamos en el índice, pudiendo iniciar sesión o registrarnos.



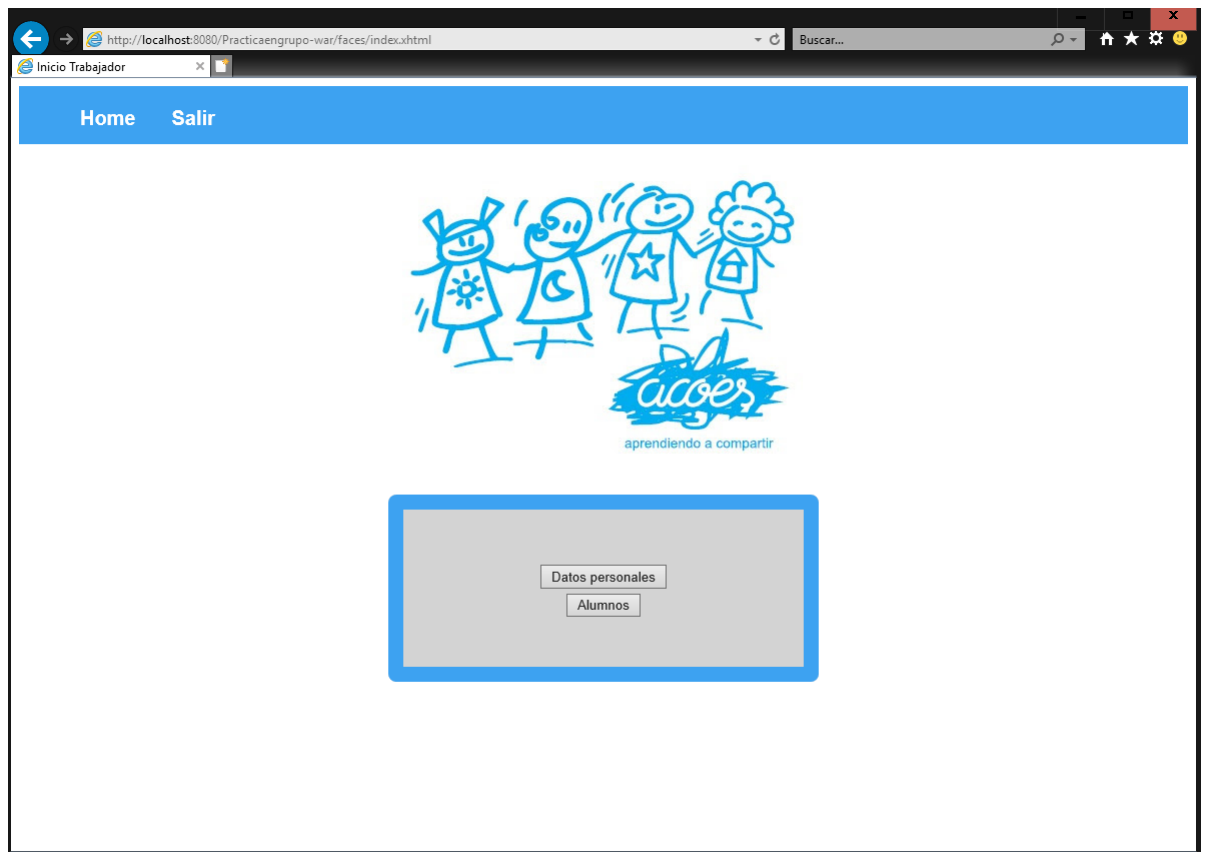
The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Practicaengrupo-war/faces/registroUsuario.xhtml`. The browser has a single tab titled 'Registro'. The page content features a blue header with the word 'Registrar' in white. Below the header, there is a registration form with the following fields and controls:

- NIF:
- Nombre:
- Apellidos:
- Fecha\_Nacimiento => yyyy/MM/dd:
- Telefono:
- Email:
- Usuario:
- Contraseña:
- Repita contraseña:
- Rol:
- 

Si al registrarnos, lo hacemos correctamente, podemos ver lo siguiente:



Si somos trabajadores, veremos esta pantalla:



Mientras que si somos socios, será:



Como trabajador, puedes añadir alumnos y sus calificaciones, y por supuesto, consultar la información respectiva a estos alumnos.

← → http://localhost:8080/Practicaengrupo-war/faces/anadirDatosPersonalesNino.xhtml Buscar...

Datos Personales

Home Salir

NIF:

Nombre:

Apellidos:

Sexo:

Comunidad de procedencia:

Comunidad en la que vive:

Cuenta de ingresos:

Grado del curso:

Estado:

← → http://localhost:8080/Practicaengrupo-war/faces/anadirCalificacion.xhtml Buscar...

Añadir Calificación

Home Salir

Nif Alumno:

Curso:

Centro:

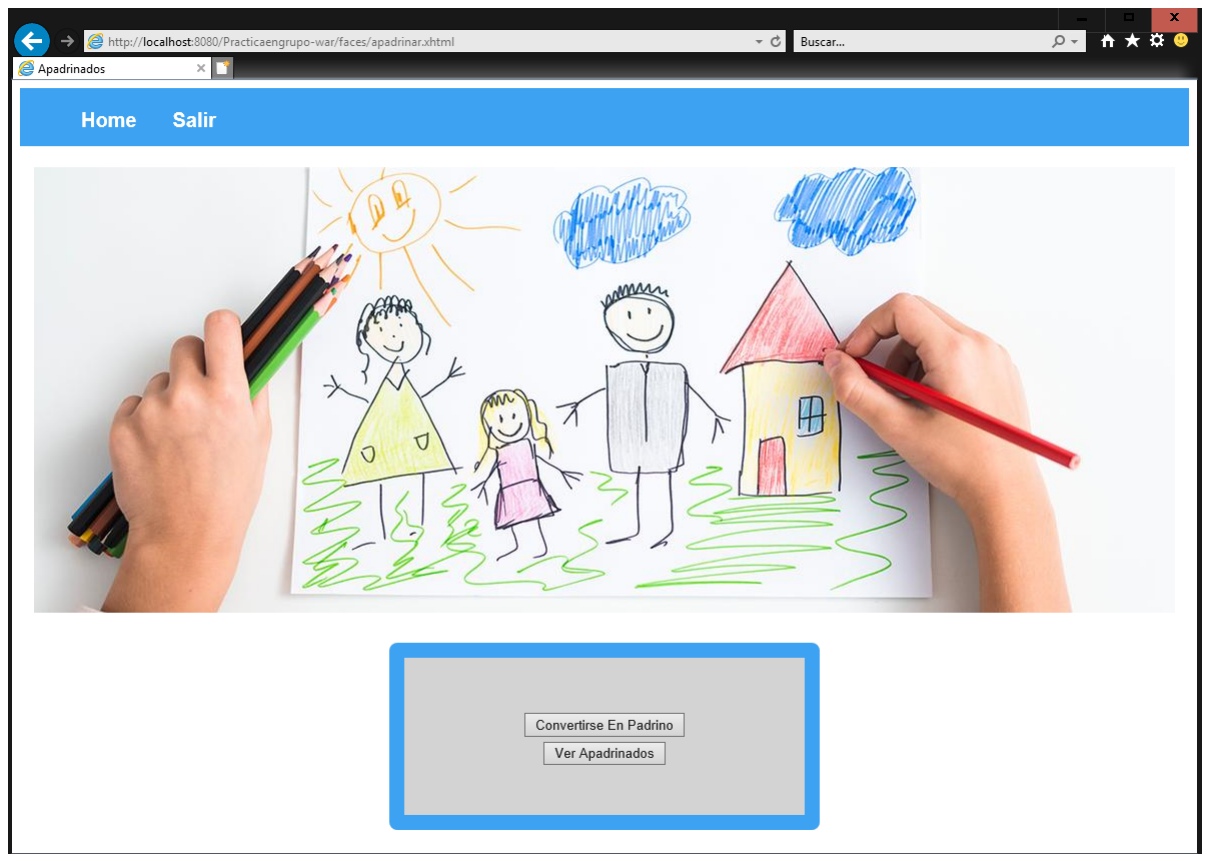
Nota:



capturas/VERDATOSYCALIFICACIONESDESEDETRABAJADORDELOSNINOS.PNG

Como socio, puedes apadrinar, ver los datos de tus apadrinados y enviarles dinero:





←


→

http://localhost:8080/Practicaengrupo-war/faces/darNifApadrinado.xhtml

Buscar...

Ver

Home Salir



El nif del niño apadrinado es:


Volver

← → http://localhost:8080/Practicaengrupo-war/faces/verApadrinados.xhtml

Apadrinados

Buscar...

Home Salir



Introduza el NIF del apadrinado:

Ver datos

Ver calificación

Donacion

Home Salir

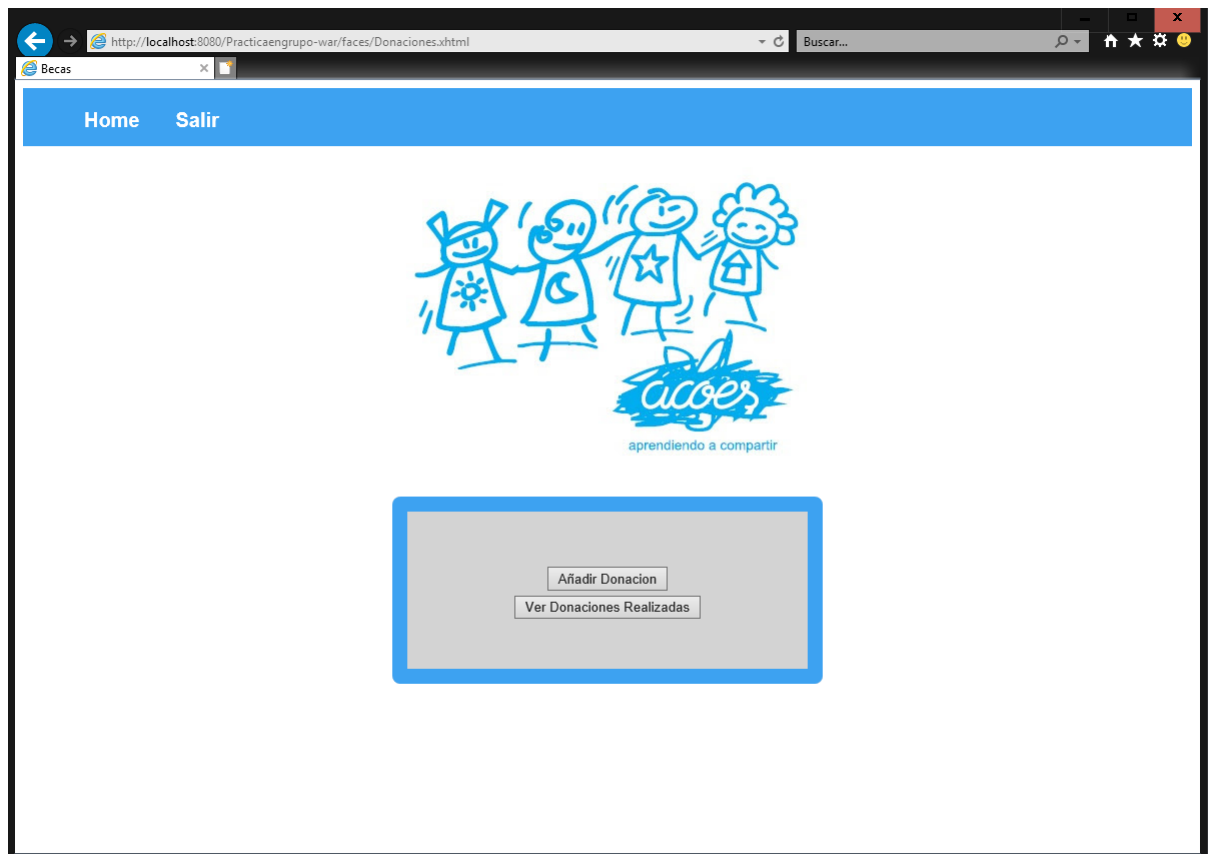
Cuenta Bancaria Usuario:

Cuenta de ingreso:

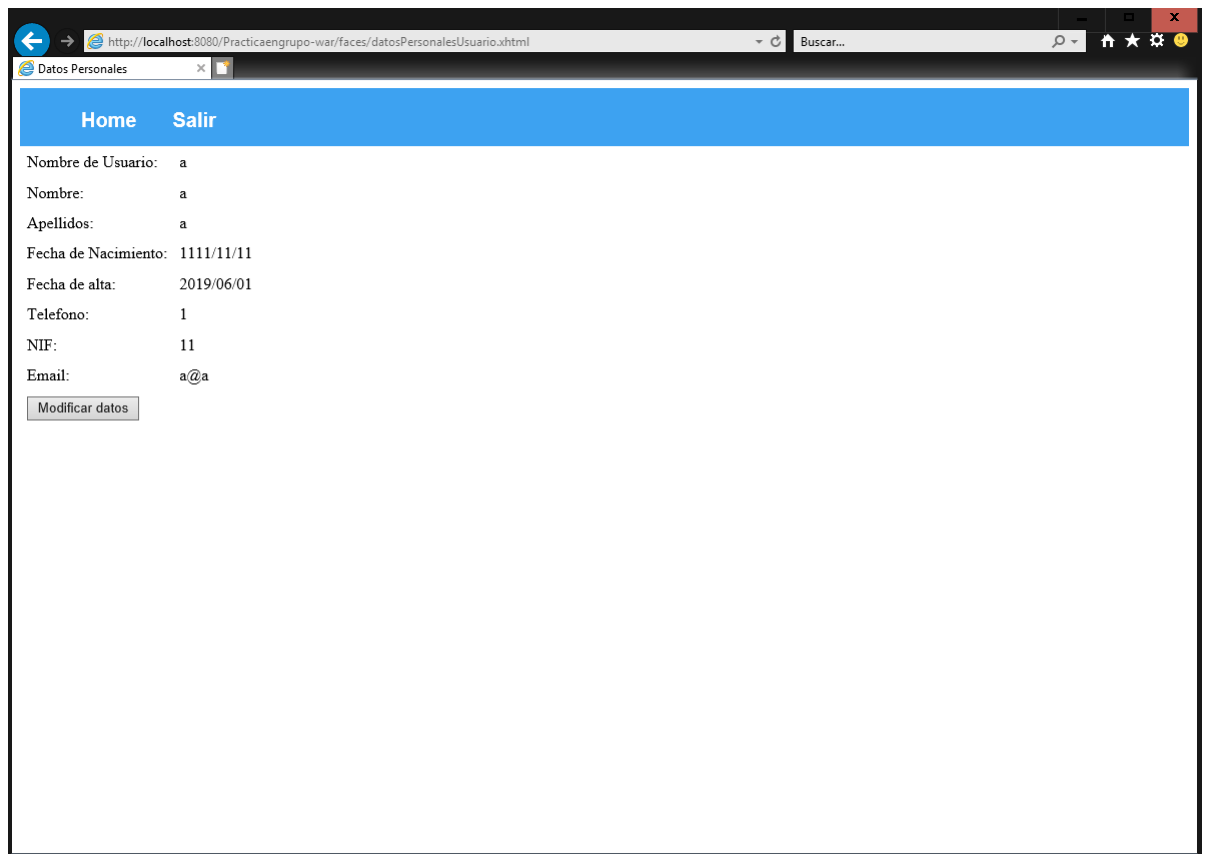
Cantidad:

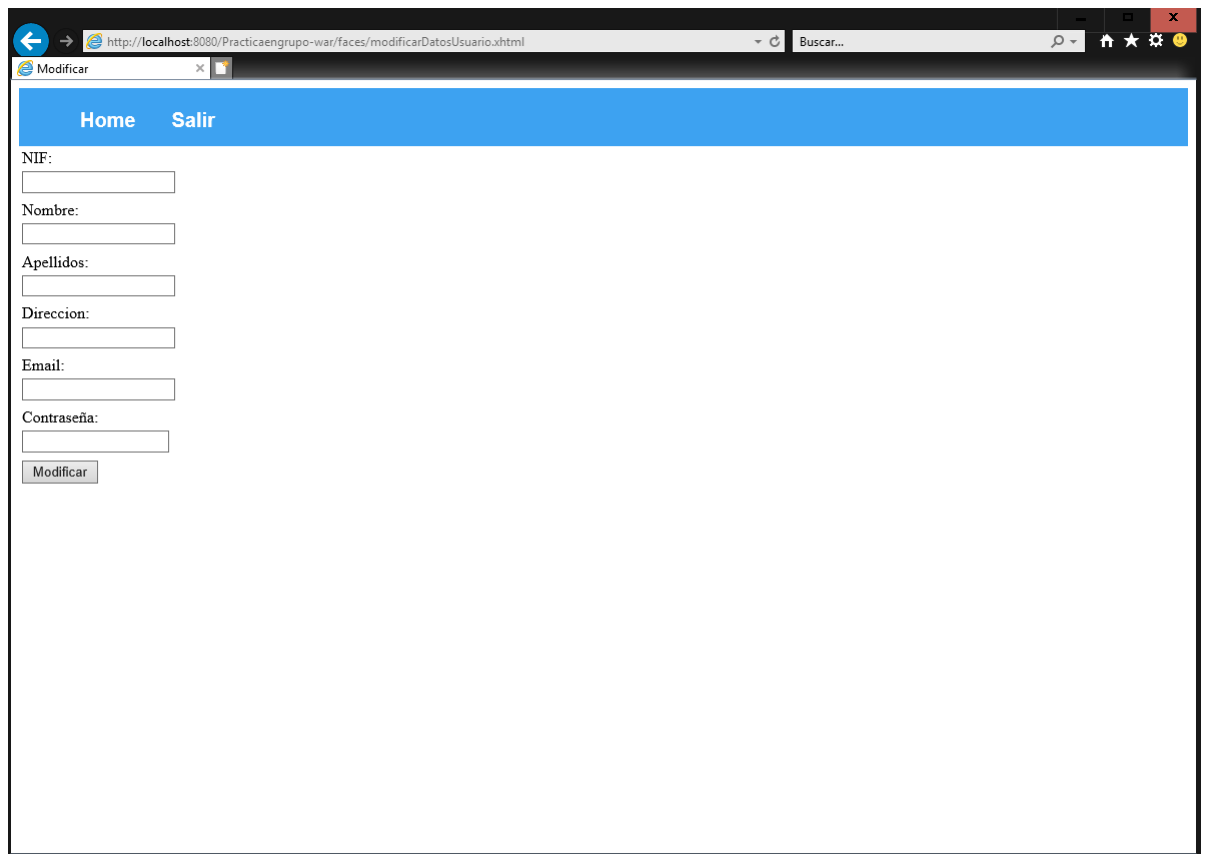
Anotaciones:

Hacer donación



Ambos pueden ver sus datos personales y modificarlos:





## Paquete ACOES

### ■ Control de Autorización

Tenemos la variable usuario que se utiliza para llamar a la entidad Usuario de la base de datos con sus correspondientes set y get.

También tenemos diferentes funciones como home() que lo que hace es devolver dependiendo del rol del usuario. Si no hay usuario nos lleva a la página de index.xhtml", si es un trabajador nos llevara a la página de inicioTrabajador.xhtml si es socio a inicioSocios.xhtml". Logout() no que hace es salir devolviendo a la página de index.xhtml poniendo a usuario a null Inicio() devuelve las pagina de inicio de los usuarios

### ■ Login

Tenemos las variables negocio que nos sirve para conectar con la capa de negocio , usuario y ctrl que llama a la clase anteriormente explicada. De cada variable se encuentra sus get y set.

En esta clase la única función que hay es autenticar() que lo que hace es logear al usuario que introduces. Esto se hace comprobando si el usuario

exite y mandándoselo al control de autorización para que nos diga a que pagina de inicio le corresponde con el rol que tiene ese usuario.

- **Nino**

Esta clase tiene las variable negocio, nino que llama a la entidad Benficiario de la base de datos , nifBuscar que se utiliza para guarda el nif del niño que quiere ver sus datos o nota y por ultimo da que llama a la entidad DatosAcademicosconsusrespectivosgetyset.

*Las funciones que hay son :*

- Beneficiario(): que carga el niño que queremos ver sus datos.
- datosAcademicos(): se encarga carga los datos académicos que queremos mostrar.
- registrar(): te envía a la pagina "verAlumDesdeProf.xhtml" llama a negocio para registrar al niño en la base de datos.
- mostrarDatos(): te manda a la pagina "datosPersonalesNino.xhtml" que lo que hace es mostrar los datos del niño que para ello con el nif.

## **Registro**

Esta clase contiene distintas variables con sus respectivos set y get. Lo más destacable de esta clase son los siguientes procedimientos y funciones:

- RegistrarUsuario(): comprueba que la contraseña y su repetir sean iguales y llama al procedimiento registrarUsuario() de la capa de negocio.
- RegistrarDonacion(): se encarga de registrar las donaciones realizadas llamando al procedimiento registrarDonacion() de la capa de negocio.
- RegistrarNino(): registra el beneficiario del mismo modo que las funciones anteriores.
- Apadrinar(): Se encarga de la gestión del apadrinamiento de los niños.
- anadirCalificacion(): Su misión es añadir la calificación a los datos académicos.
- modificarDatosUsuario(): modifica los datos del usuario activo.

## **2. Entrega 3**

### **Descripción de los EJB**

**AcoesException.java:** devuelve una excepción general.

**BeneficiarioInexistenteException.java:** devuelve una excepción al intentar buscar un niño que no pertenece a la base de datos.

**BeneficiarioRepetido.java:** devuelve una excepción al intentar registrar un niño que ya se encuentra en la base de datos.

**CalificacionRepetida.java:** devuelve una excepción al intentar añadir una calificación ya existente.



**ContraseniaInvalidaException.java:** devuelve una excepción al introducir una contraseña no válida.

**CuentaInactivaException.java:** devuelve una excepción al intentar ingresar con una cuenta no activa.

**CuentaInexistenteException.java:** devuelve una excepción al intentar ingresar con una cuenta inexistente.

**CuentaRepetidaException.java:** devuelve una excepción al intentar registrar una cuenta ya existente.

**Negocio.java:** interfaz que contiene los siguientes métodos:

- **registrarUsuario(Usuario u):** si el usuario recibido ya existe en la base de datos, devuelve una excepción. De lo contrario, lo crea.
- **compruebaLogin(Usuario u):** comprueba que el nombre de usuario y la contraseña del usuario recibido, y que pertenecen a un usuario que existe en la BBDD. En caso contrario, devuelve una excepción.
- **refrescarUsuario(Usuario u):** se realiza un `compruebaLogin(u)` y, si no devuelve una excepción, se devuelve una entidad `Usuario` con los datos existentes en la BBDD del usuario recibido.
- **regisNino(Beneficiario b):** añade al niño recibido a la base de datos.
- **refrescarNino(Beneficiario b):** se realiza un `compruebaNino(b)` y, si no devuelve una excepción, se devuelve una entidad `Beneficiario` con los datos existentes en la BBDD del niño recibido.
- **compruebaNino(Beneficiario b):** comprueba si el niño recibido se encuentra en la base de datos. En caso contrario, devuelve una excepción.
- **registrarDonacion(Ingresos i):** añade un nuevo ingreso.
- **registrarCalificacion(Datos academicos d):** añade una nueva calificación.
- **buscarNino(Long nif):** busca un niño en la base de datos por su NIF. Si no existe, devuelve una excepción.
- **modificarDatosUsuario(Usuario u):** actualiza los datos de un usuario.
- **buscarDA(Long nif):** busca los datos académicos de un niño mediante su NIF. Si no existen, devuelve una excepción.

**NegocioImpl.java:** desarrollo de los métodos definidos en `Negocio.java`.

### 3. URL a GIT

Este es el repositorio usado y el hash del commit es 91d8c261ff, aunque el commit posterior será con esta memoria actualizada.