

ACOES

SISTEMAS DE INFORMACIÓN PARA INTERNET

MEMORIA VISTA DE LA APLICACIÓN CON JSF

ACOES

Contenido

[INTRODUCCIÓN. 4](#_Toc7981550)

[Modificación JPA 4](#_Toc7981551)

[GRAFOS DE NAVEGACIÓN 4](#_Toc7981552)

[Actor: Usuario invitado (Sin registrar). 5](#_Toc7981553)

[Explicación. 6](#_Toc7981554)

[Requisitos cumplidos: 6](#_Toc7981555)

[Actor: Usuario registrado. 7](#_Toc7981556)

[Explicación. 8](#_Toc7981557)

[Requisitos cumplidos: 8](#_Toc7981558)

[Actor: Socio (Usuario verificado) 9](#_Toc7981559)

[Explicación. 10](#_Toc7981560)

[Requisitos cumplidos: 10](#_Toc7981561)

[Actor: Administrador 11](#_Toc7981562)

[Explicación. 11](#_Toc7981563)

[Requisitos cumplidos: 12](#_Toc7981564)

[BACKING BEANS Y PÁGINAS JSF 13](#_Toc7981565)

[Log-in: 13](#_Toc7981566)

[Administrador: 14](#_Toc7981567)

[Usuarios: 14](#_Toc7981568)

[Socios: 14](#_Toc7981569)

[Registro: 15](#_Toc7981570)

[Ser socio: 15](#_Toc7981571)

[Populorum 16](#_Toc7981572)

[Admin 17](#_Toc7981573)

[Ninios 17](#_Toc7981574)

[Conclusión. 17](#_Toc7981575)

[Fotos usuario invitado: 18](#_Toc7981576)

[Fotos usuario registrado: 19](#_Toc7981577)

[Fotos Socio: 23](#_Toc7981578)

[Fotos administrador: 27](#_Toc7981579)

#### MIEMBROS DEL GRUPO

Andrés Valentín Suárez Mediavilla - 77425032V

Sergio González Sicilia - 77225653W

Ignacio Pascual Gutiérrez - 54236255R

Katia Moreno Berrocal - 79038916Z

Rocío Montalvo Lafuente - 03138701Y

URL Repositorio (los archivos finales se encuentran en la carpeta: Tarea 2):

<https://github.com/Lilithsoul/ACOES>

Commit final del proyecto:

<https://github.com/Lilithsoul/ACOES/commit/bc3363786f050fbdfbb714cab188ec23fdff4bb3>

Dicho commit es la última modificación del proyecto, la carpeta del proyecto ACOES también se encuentra dentro de la tarea 2, llamada ‘TRABAJO GRUPAL’.

# INTRODUCCIÓN.

Este documento está elaborado con el propósito de actualizar toda la información referente al proyecto ACOES, incluyendo modificaciones respecto a las tareas anteriores, los nuevos grafos de navegación y la coherencia de las páginas JSF con los Backing Beans.

Aclarar que el contenido de la entrega refleja el funcionamiento de la página web y su navegación, pero no incluye dichas funcionalidades.

# Modificación JPA

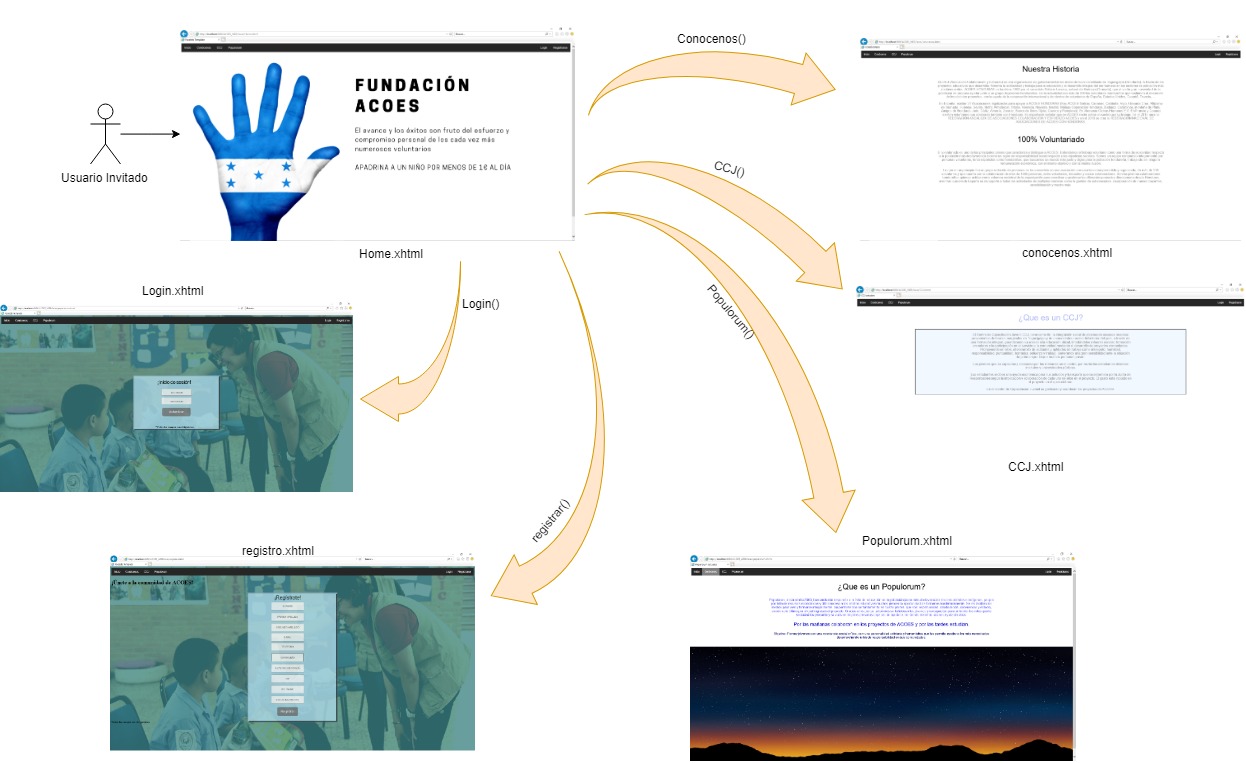
Se ha modificado una de las entidades JPA: JOVEN\_NIÑO añadiéndole un atributo de tipo boolean llamada apadrinado, debido a que será necesario para la función apadrinar.

# GRAFOS DE NAVEGACIÓN

\*Los requisitos desglosados se encuentran en el DGR aportado en la Tarea 2\*

\*Al final de la memoria se encuentran las fotos de todos los diagramas en el caso que no se puedan visualizar bien los diagramas\*

## Actor: Usuario invitado (Sin registrar).



### Explicación.

Un usuario que accede por primera vez a la página web tiene acceso a una vista de inicio con el eslogan de la asociación, desde esta página se permite el acceso a las vistas:

**Conócenos**: Donde se explica de forma superficial de qué trata la asociación.

**CCJ:** Con una breve explicación sobre lo que son los centros de capacitación juvenil.

**Populorum:** Con una breve explicación sobre lo que son los casas populorum donde se alojan los niños.

**Login:** Donde un usuario ya registrado puede acceder a su cuenta.

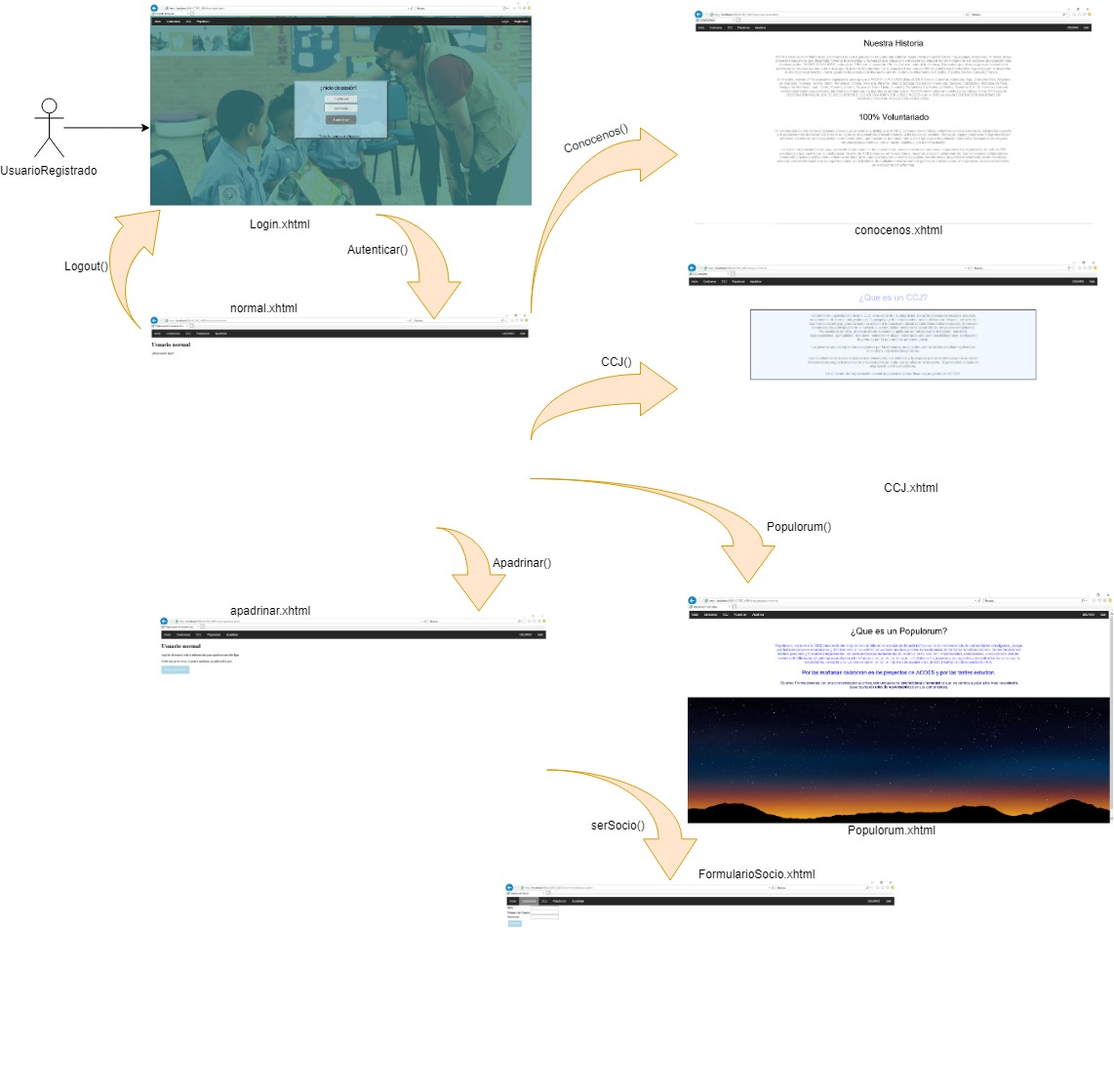
**Registrarse:** Donde un usuario que aún no posee una cuenta puede registrarse.

### Requisitos cumplidos:

Existencia de usuario invitado. (1.1)

Se proporciona un apartado para registro. (2.1)

## Actor: Usuario registrado.



### Explicación.

Un usuario registrado es aquel que aún no ha introducido su DNI, su número de tarjeta y dirección. Un usuario registrado desde la vista de login accede a su cuenta mediante el uso de sus credenciales, en la vista inicio se nos muestra actualmente una bienvenida muy básica, pero se planean agregar más funcionalidades, desde aquí se puede acceder a:

**Conócenos**: donde se explica de forma superficial de que trata la asociación.

**CCJ:** Con una breve explicación sobre lo que son los centros de capacitación juvenil.

**Populorum:** Con una breve explicación sobre lo que son los populorum.

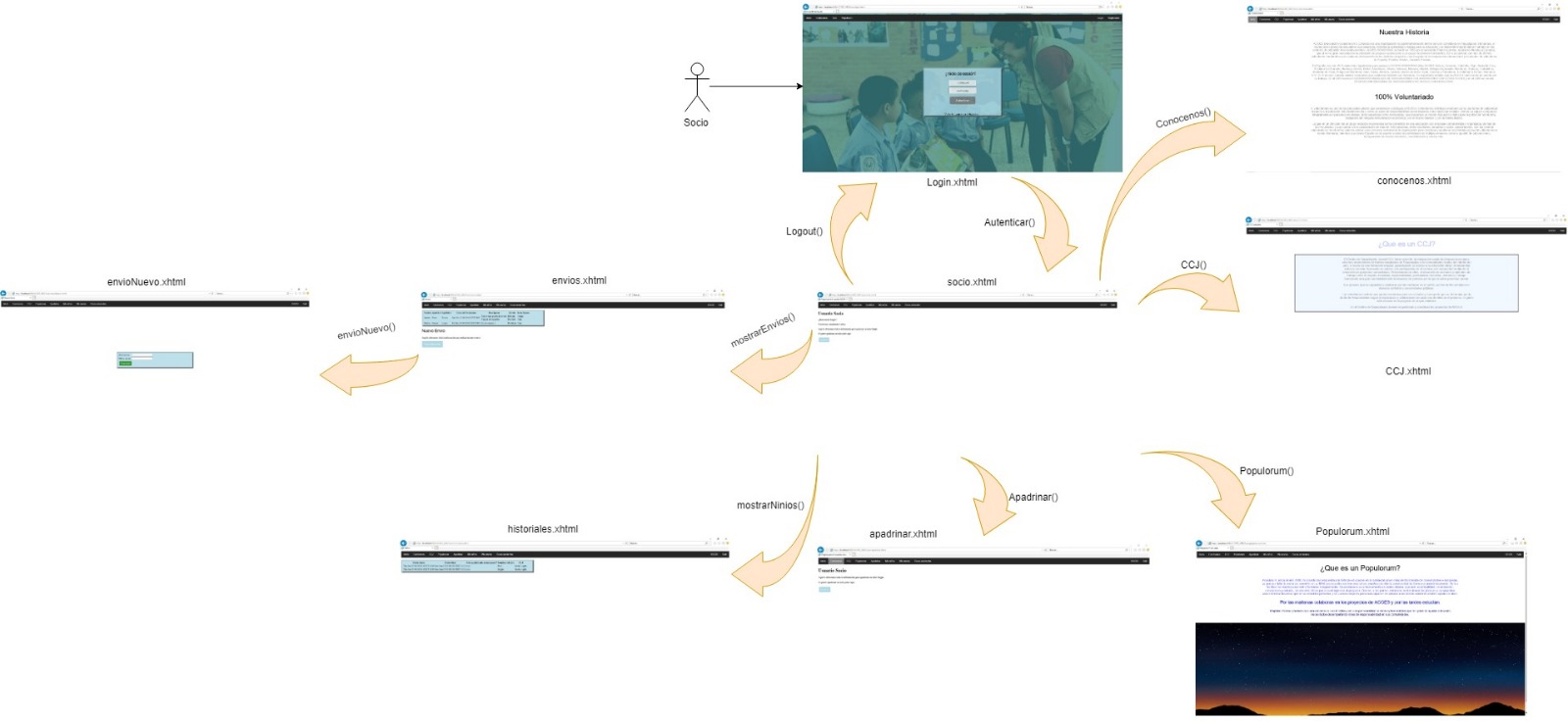
**Apadrinar:** Se da una breve explicación sobre la forma de apadrinar un niño junto a un botón para volverse socio y poder apadrinar.

**Salir:** Botón que permite cerrar sesión.

### Requisitos cumplidos:

Existencia de usuario registrado. (1.4)

Actor: Socio (Usuario verificado)



### Explicación.

Un socio es un usuario ya verificado con dirección, DNI y tarjeta asociada. Un socio desde la vista de login accede a su cuenta mediante el uso de sus credenciales, el socio posee como página de inicio una vista donde se le indican el número de apadrinados que tiene junto a un botón para apadrinar a algún niño si lo desea, desde esta vista puede acceder a:

**Conócenos**: donde se explica de forma superficial de que trata la asociación.

**CCJ:** Con una breve explicación sobre lo que son los centros de capacitación juvenil.

**Populorum:** Con una breve explicación sobre lo que son los populorum.

**Apadrinar:** Se da una breve explicación sobre la forma de apadrinar un niño junto a un botón de apadrinamiento.

**Mis niños:** Donde se podrá visualizar el historial de los apadrinamientos realizados por un socio.

**Mis envíos:** Donde se podrá visualizar el historial de los envíos realizados por unsocio.

**Becas:** Donde el socio podrá visualizar las becas existentes.

**Salir:** Botón que permite cerrar sesión.

### Requisitos cumplidos:

Existencia de socio. (1.2)

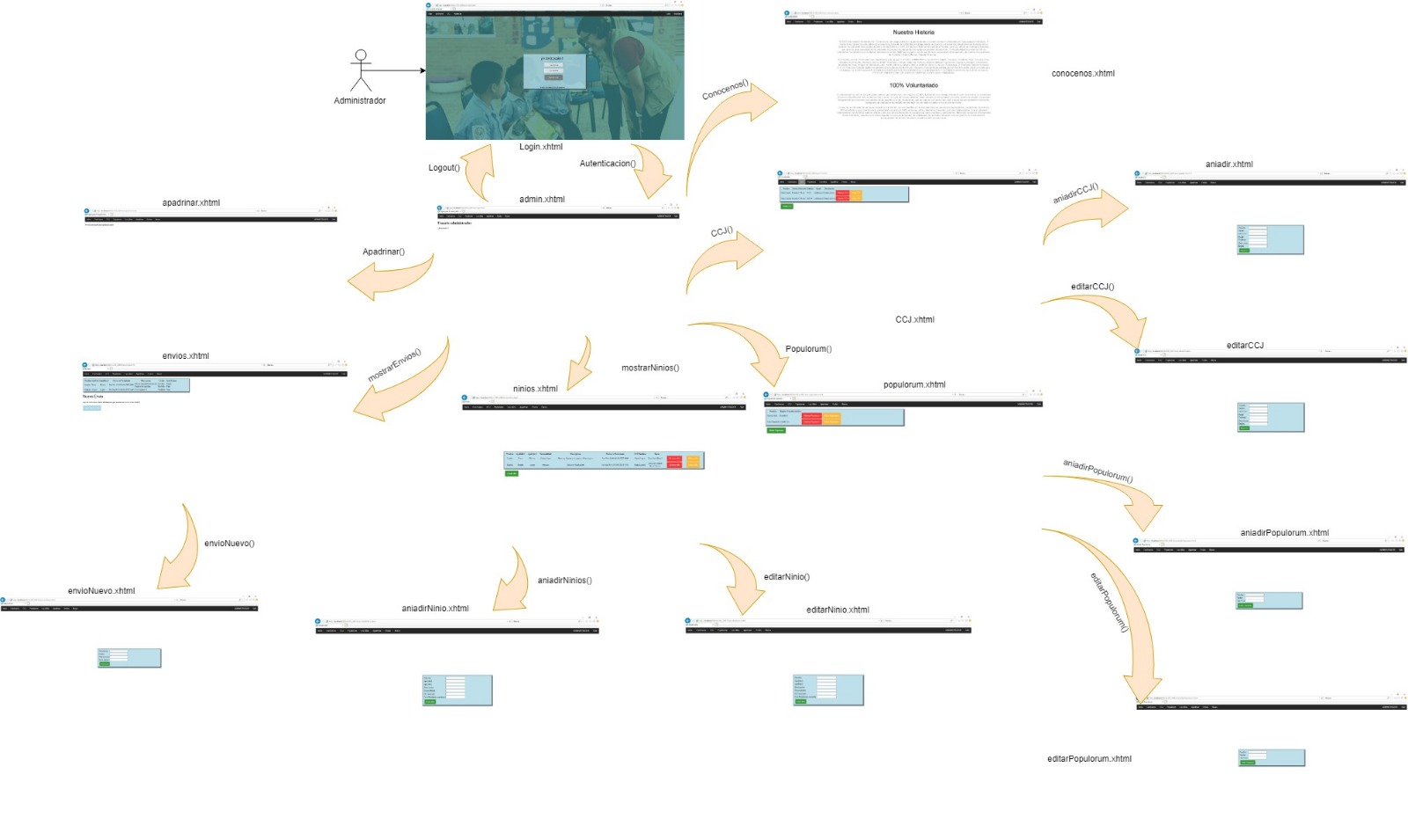
Visualización de becas. (2.2.1.5)

Visualización de CCJ de los apadrinados. (2.2.2.5)

Visualización de Populorum de los apadrinados. (2.2.3.5)

Visualización de niños. (2.2.4.5)

## Actor: Administrador



## Explicación.

Un administrador es un usuario con permisos especiales. Un administrador desde la vista de login accede a su cuenta mediante el uso de sus credenciales, este posee una bienvenida (también muy básica por el momento). Desde la página de inicio podemos acceder a:

**Conócenos**: donde se explica de forma superficial de que trata la asociación.

**CCJ:** Donde al administrador de permite eliminar editar o añadir CCJ que actualmente los botones no están operativos ya que aun no está implementada la base de datos.

**Populorum:** Donde existe una tabla con los diferentes populorum almacenados, donde más adelante se le permitirá, al administrador, añadir, editar o eliminar populorum.

**Apadrinar:** Muestra un mensaje de error ya que el administrador no puede apadrinar.

**Los niños:** Muestra a todos los niños registrados y al administrador se le permite, eliminar, añadir o editar a los niños. El botón “Eliminar Niño” borrará al niño seleccionado de la base de datos de ACOES.

**Añadir niño:** Posee un registro con todos los datos que se deben proporcionar para añadir un niño.

**Editar niño:** Posee un registro con todos los datos que se deben proporcionar para editar al niño seleccionado.

**Envíos:** Vista donde al administrador se le permite crear aparte de visualizar el historial de los ya creados.

**Crear Envío:** Posee un registro con los datos necesarios para crear un envío.

**Becas:** Donde el administrador podrá visualizar, crear, editar y eliminar becas. El botón “Eliminar Beca” borrará la beca seleccionada de la base de datos de ACOES.

**Crear Beca:** Posee un registro con los datos necesarios para crear una beca.

**Editar Beca**: Posee un registro con todos los datos que se deben proporcionar para editar la beca seleccionada.

**Salir:** Botón que permite cerrar sesión

### Requisitos cumplidos:

Existencia de administrador. (1.3)

Gestión de becas. (2.2.1)

Gestión CCJ. (2.2.2)

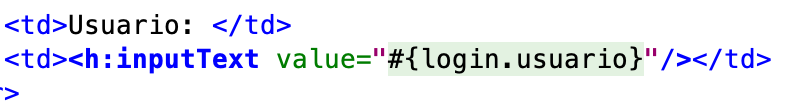
Gestión Populorum. (2.2.3)

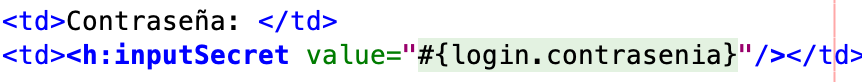
Gestión de niños. (2.2.4)

Gestión de envíos. (2.4)

# BACKING BEANS Y PÁGINAS JSF

Log-in:

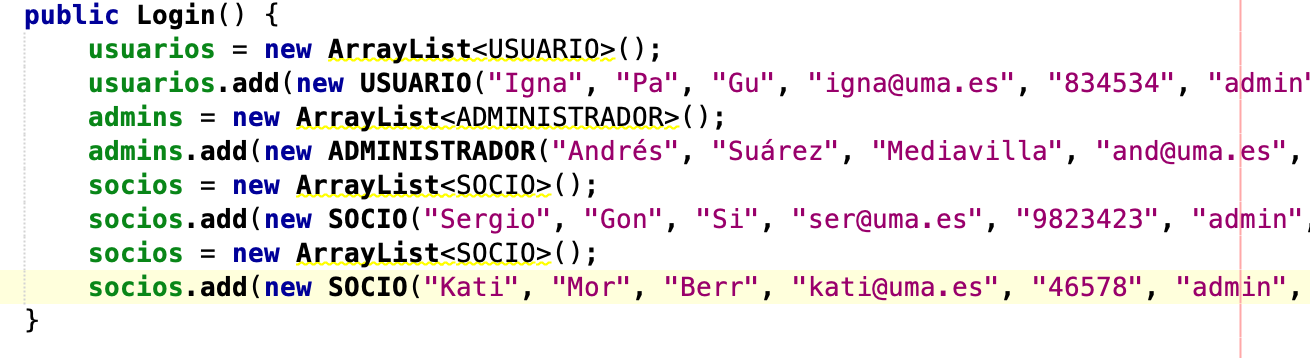


<!-- … -- >>

Tanto login.usuario como login.contrasenia se utilizan para almacenar como variable el usuario y la contraseña. Estos parámetros lo recibirá el constructor de la clase Login.

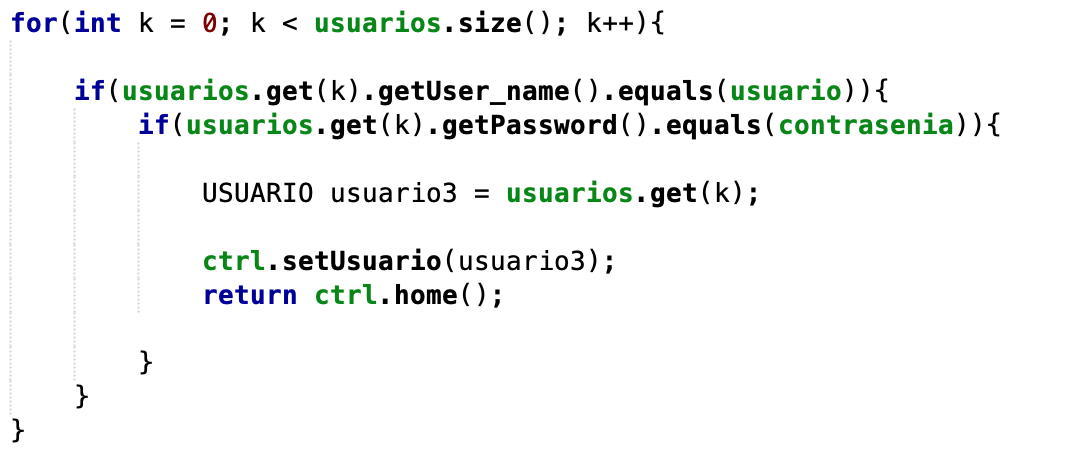
En nuestro caso tuvimos que introducir datos ficticios dado que la base de datos no está enlazada todavía. El primer paso es inyectar una variable de instancia de la clase ControlAutorización. Finalmente, se recorren los arrays de ADMINISTRADOR, SOCIO y USUARIO. Cuando se encuentra dónde está se le mostrará una vista y otra. En caso de no existir, saltará un mensaje de aviso.



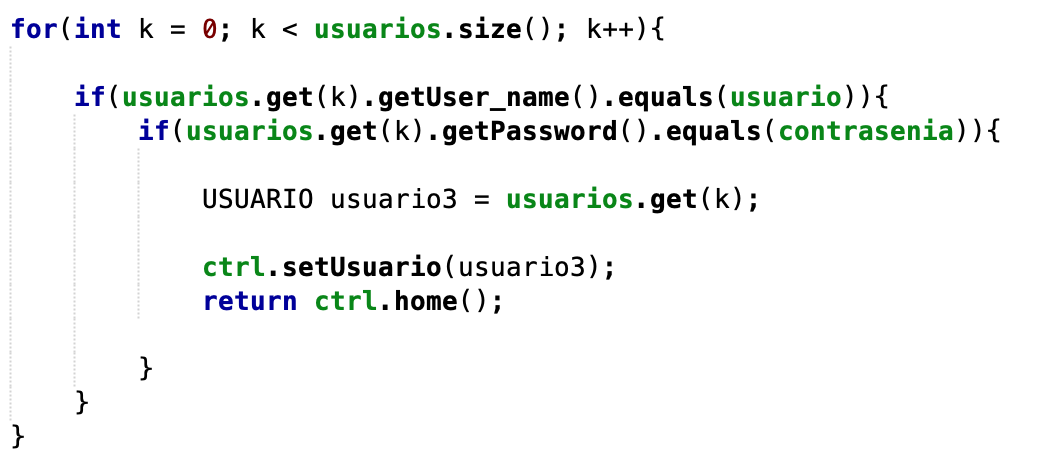


### Administrador:

### Usuarios:



### Socios:



Registro:

Al igual que en el apartado anterior, disponemos de una clase Registro donde almacenamos todos los campos de un nuevo usuario. Finalmente, al igual que en el login, tenemos un botón que llama a la función registrar de la clase Registro que hace efectiva esta operación.

Botón:

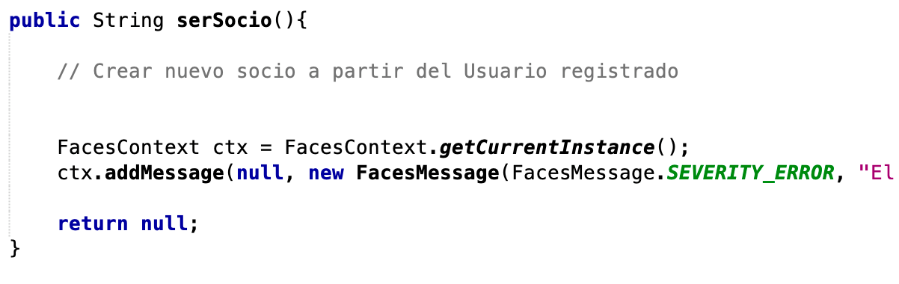
Ejemplo de como tomamos el nombre:

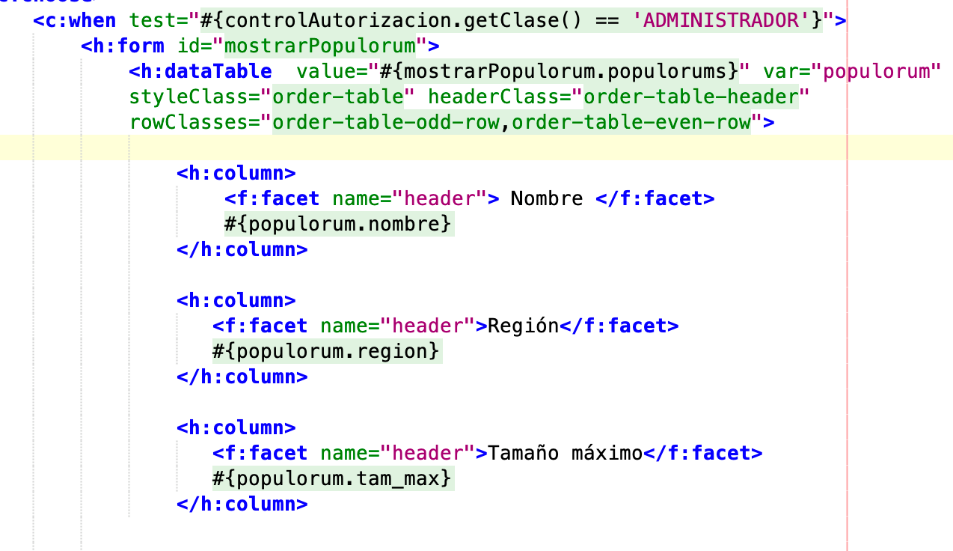
Método registrar (aplicable al tocar el botón):



Ser socio:

Al igual que en todo lo anterior, disponemos de una clase llamada: SerSocio. En esta guardamos los campos y finalmente en el método serSocio hacemos efectiva el registro.

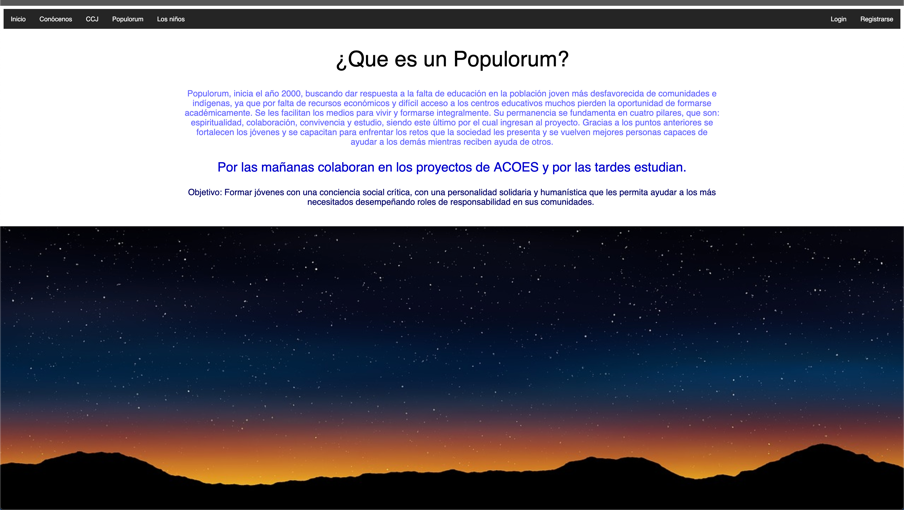


Populorum

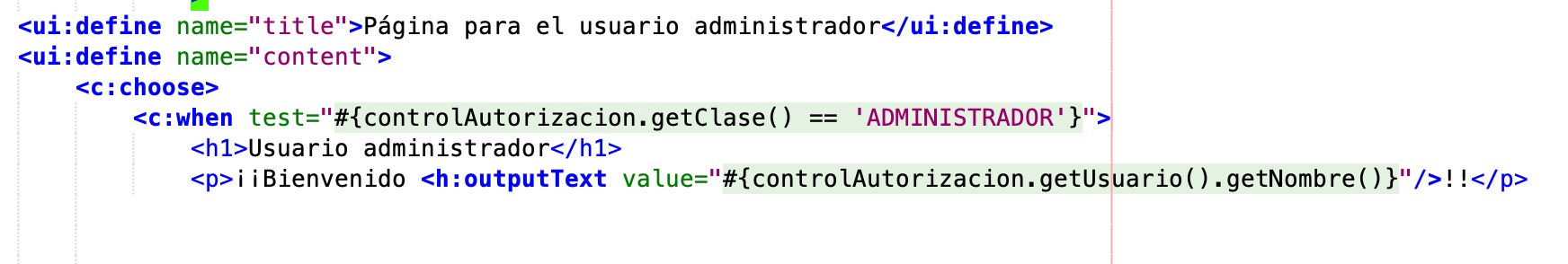
Antes de acceder a esta página comprobamos si el usuario que accede es administrador para introducir un nuevo centro.

En caso negativo, simplemente nos saldrá una breve explicación de lo que trata.

Cuando el usuario que accede no es administrador, esta es la vista.

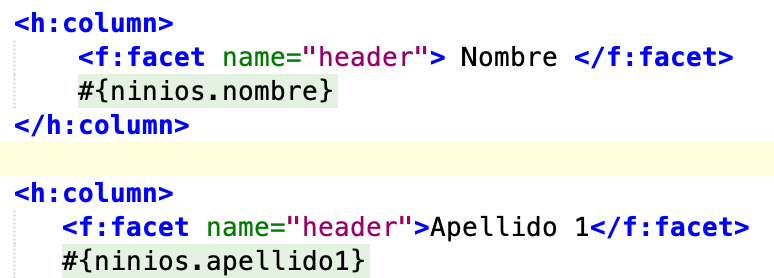


Admin

  
Cuando accedemos a esta página primero comprobamos que el usuario que intenta acceder sea administrador. Si es administrador, se nos muestra una vista, en caso negativo nos pedirá que iniciemos sesión como administrador.

Ninios

En la página ninios, se nos mostrará una lista de los niños que están disponibles para el apadrinamiento. La tabla se generará a medida que vayamos metiendo niños en la base de datos. Los parámetros se pasan de la siguiente manera:

Mediante los backing beans

#{ninios.Variable}

le pasamos la información.

Conclusión.

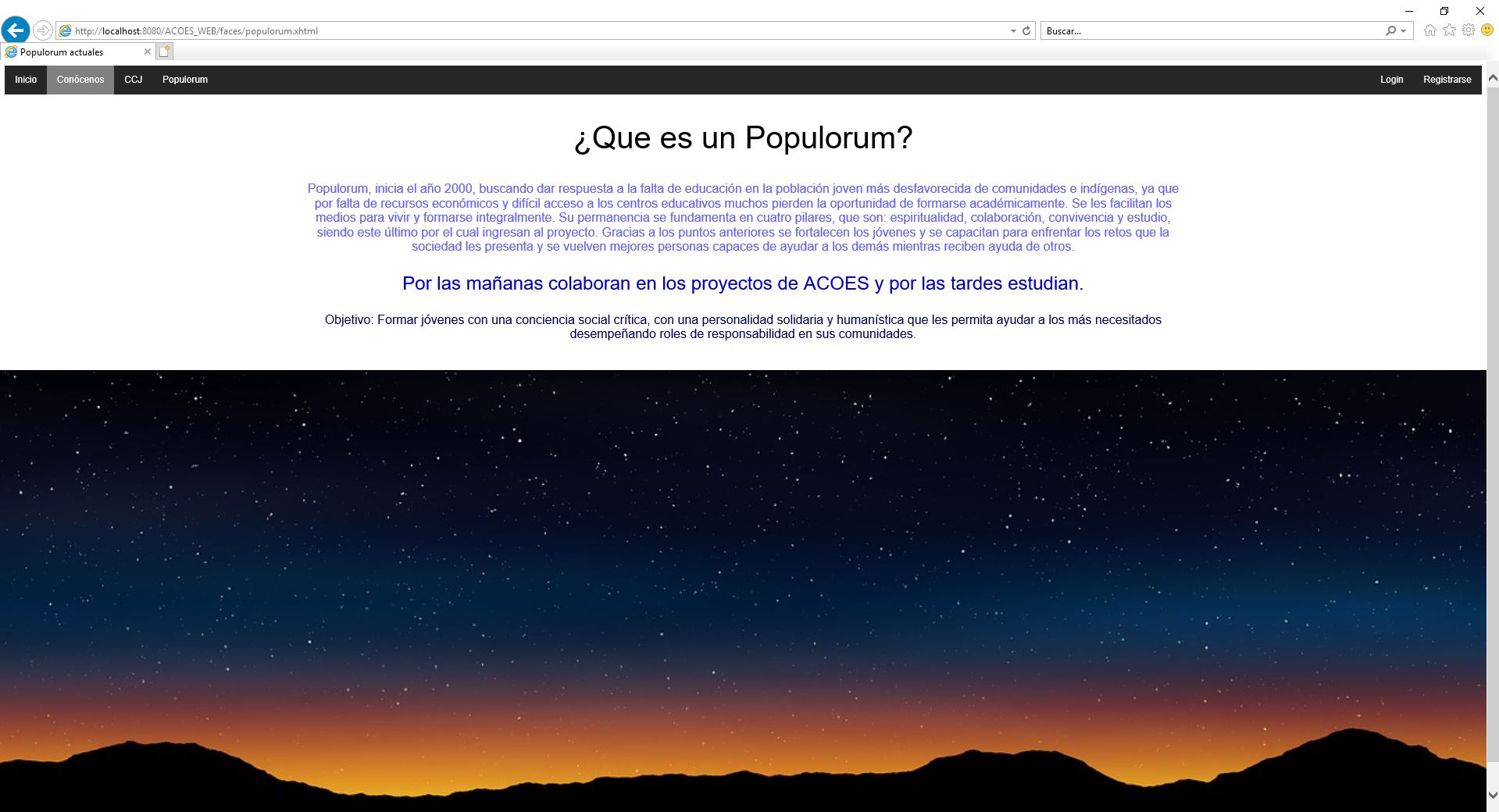
Las páginas están implementadas siguiendo la misma estructura.

1. Si se quiere mostrar un dato, se sigue lo siguiente:
   1. #{entidadJPA.parámetroAMostrar}
2. Si se quiere introducir un dato:
   1. #{controlador.entidadJPA.parametroAIntroducir}
   2. También es válido(en caso de que ya exista el usuario):
      1. #{empty controlAutorización.usuario} para ver que tipo de usuario nos pasan.
      2. #{controlador.entidadJPA}

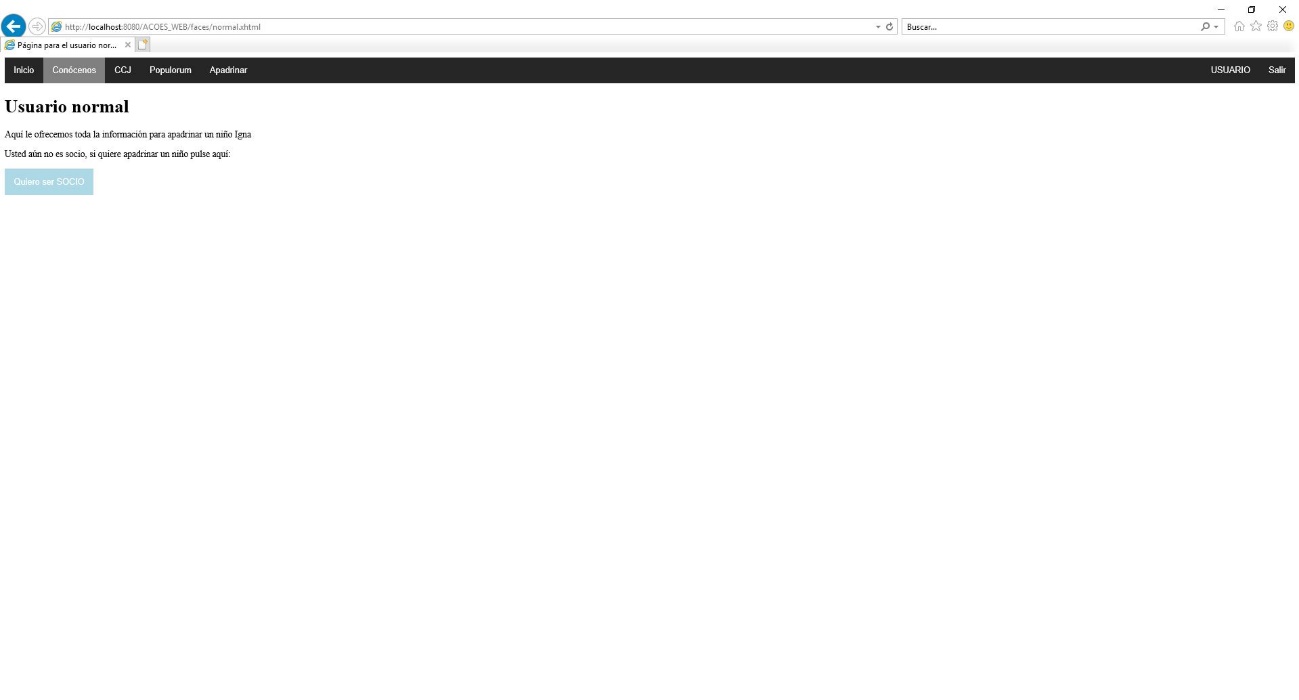
Después es necesario introducir un botón para que sean efectivos los datos introducidos. #{controlador.funcion()}

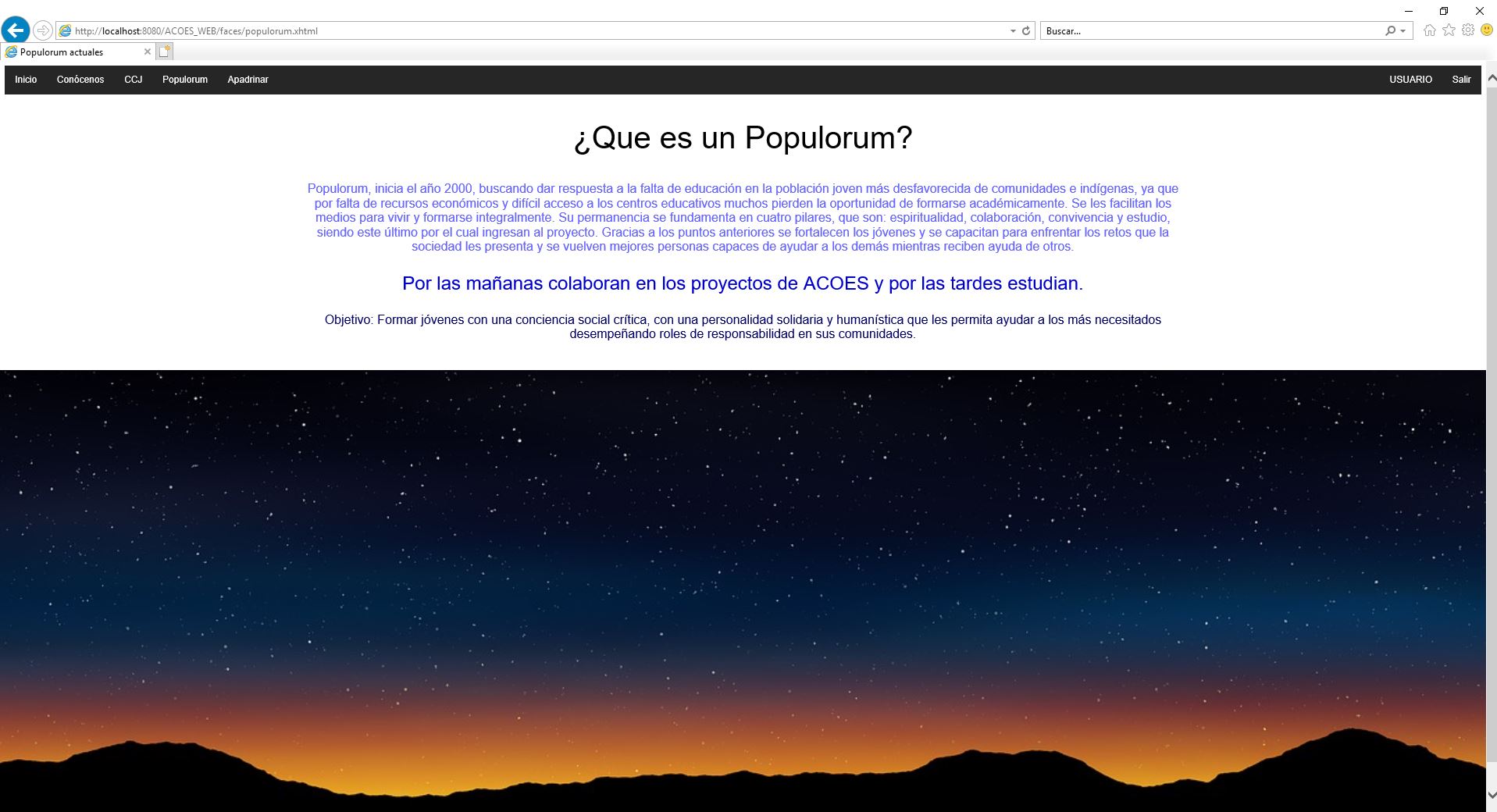
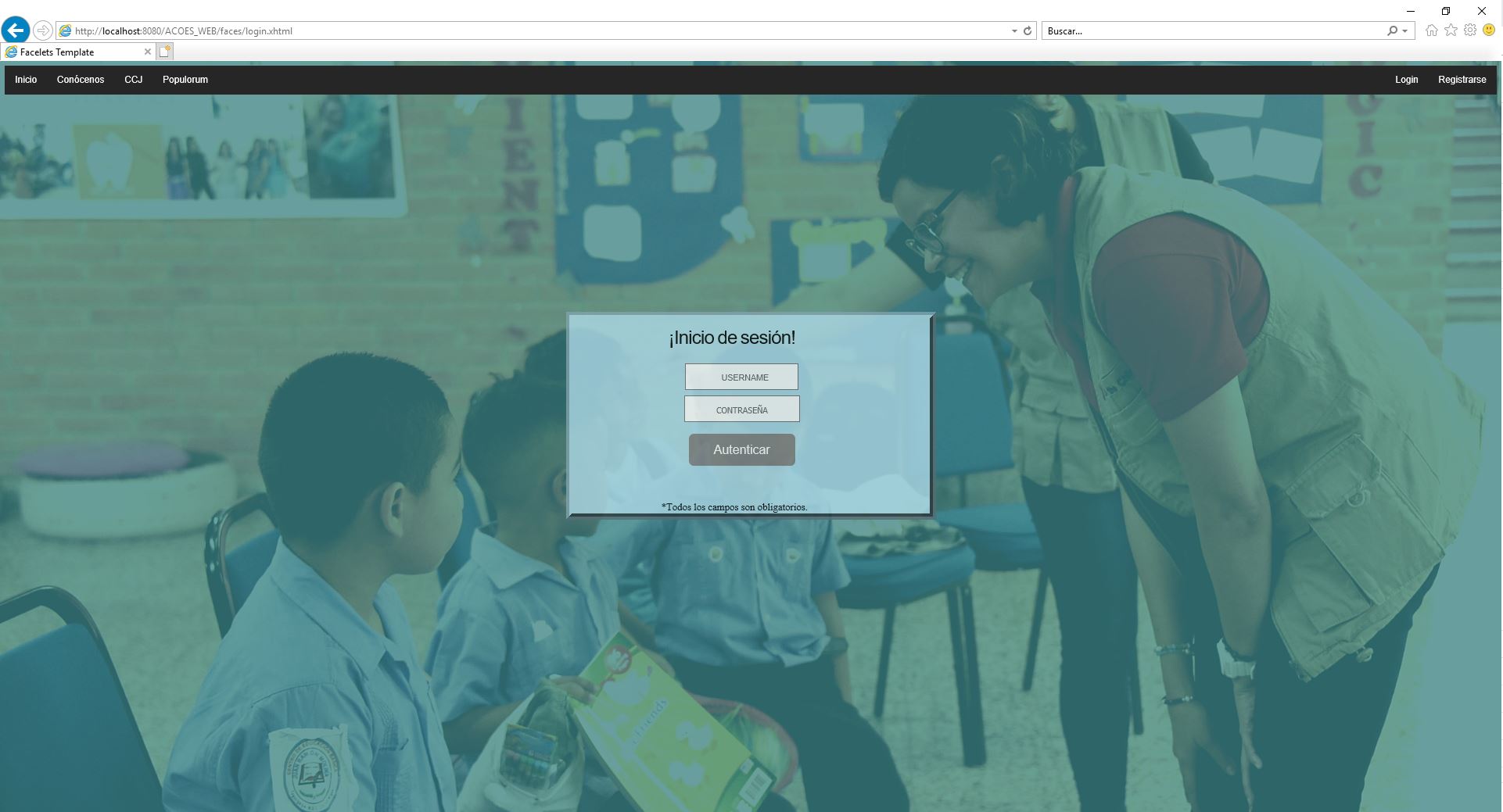
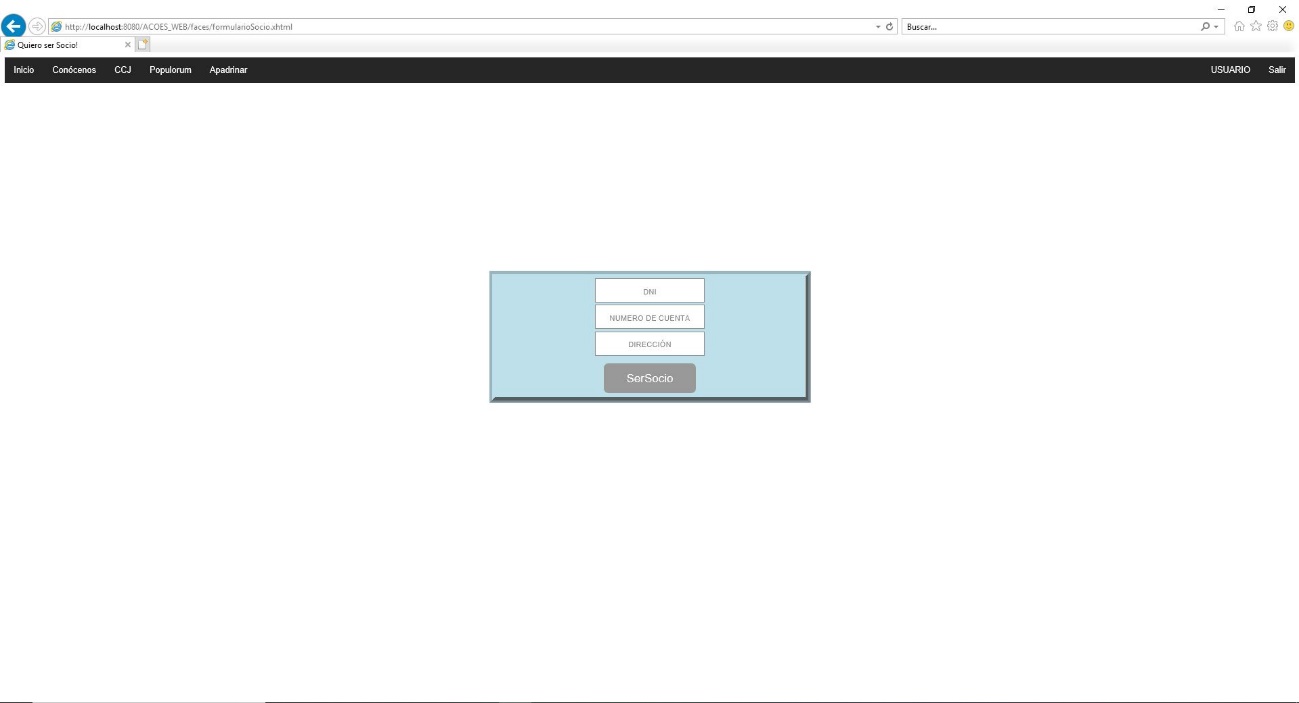
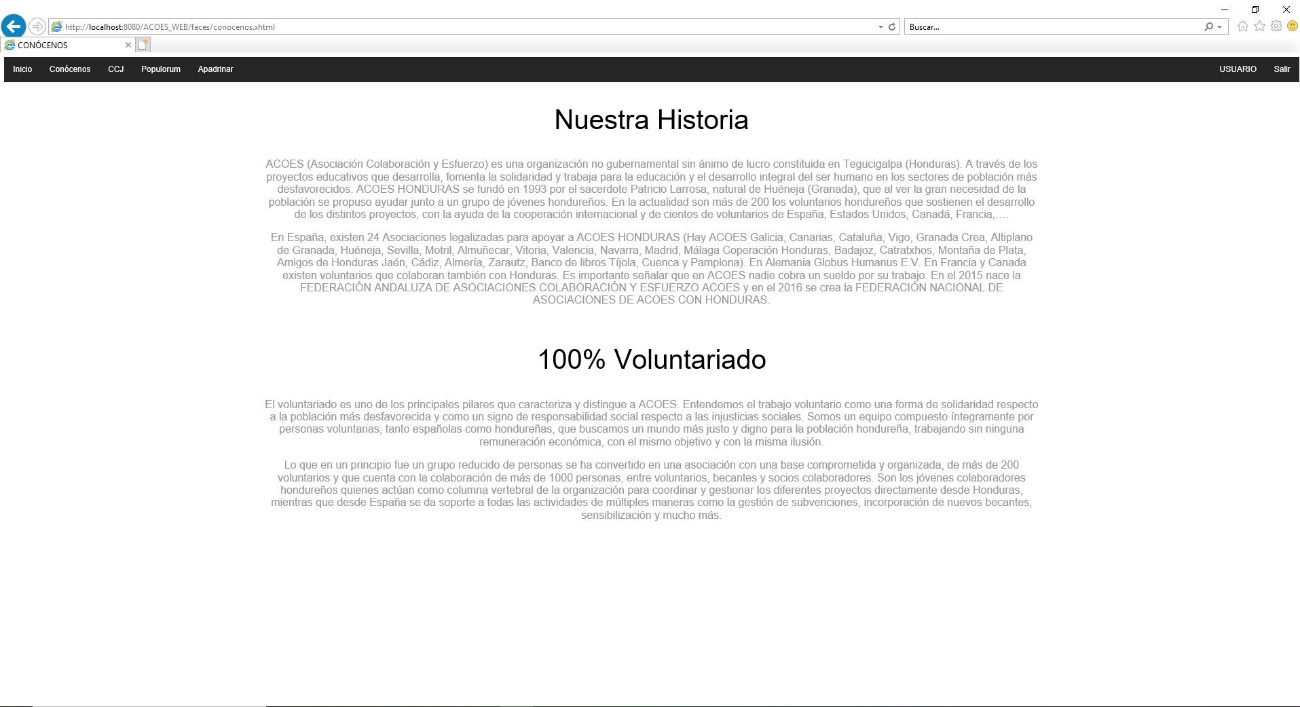
## Fotos usuario invitado:



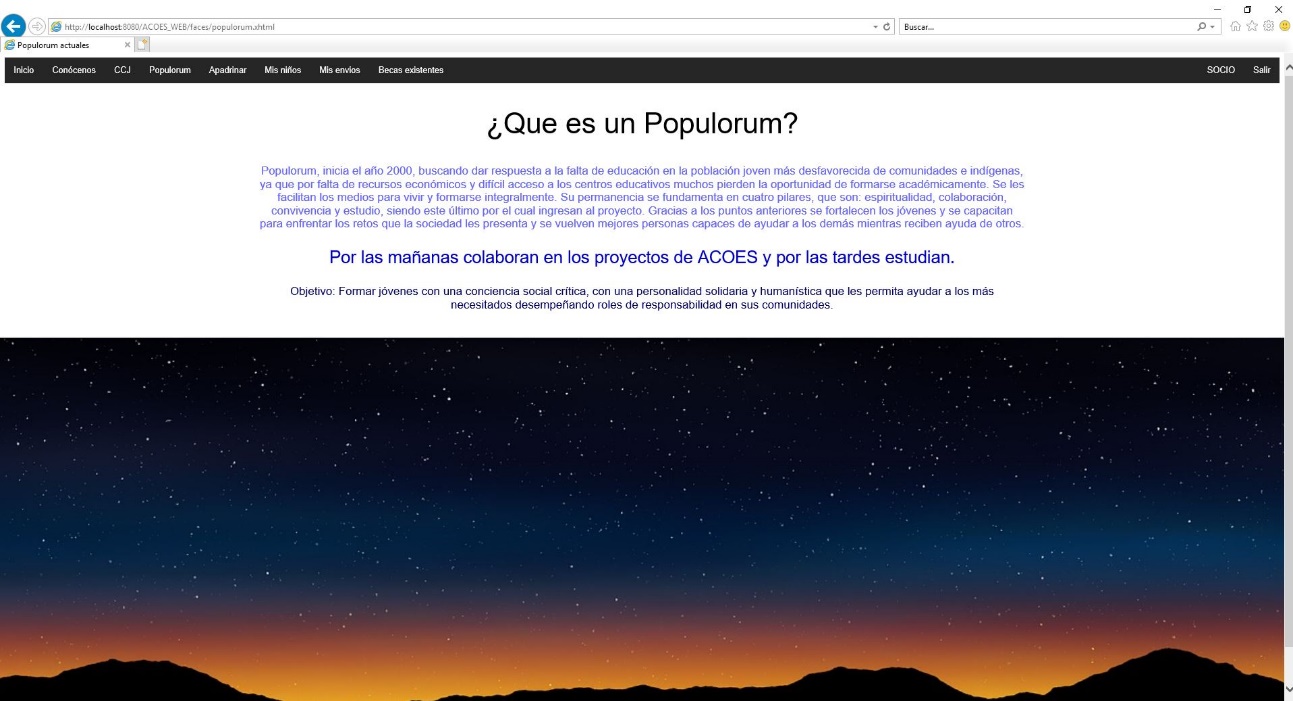
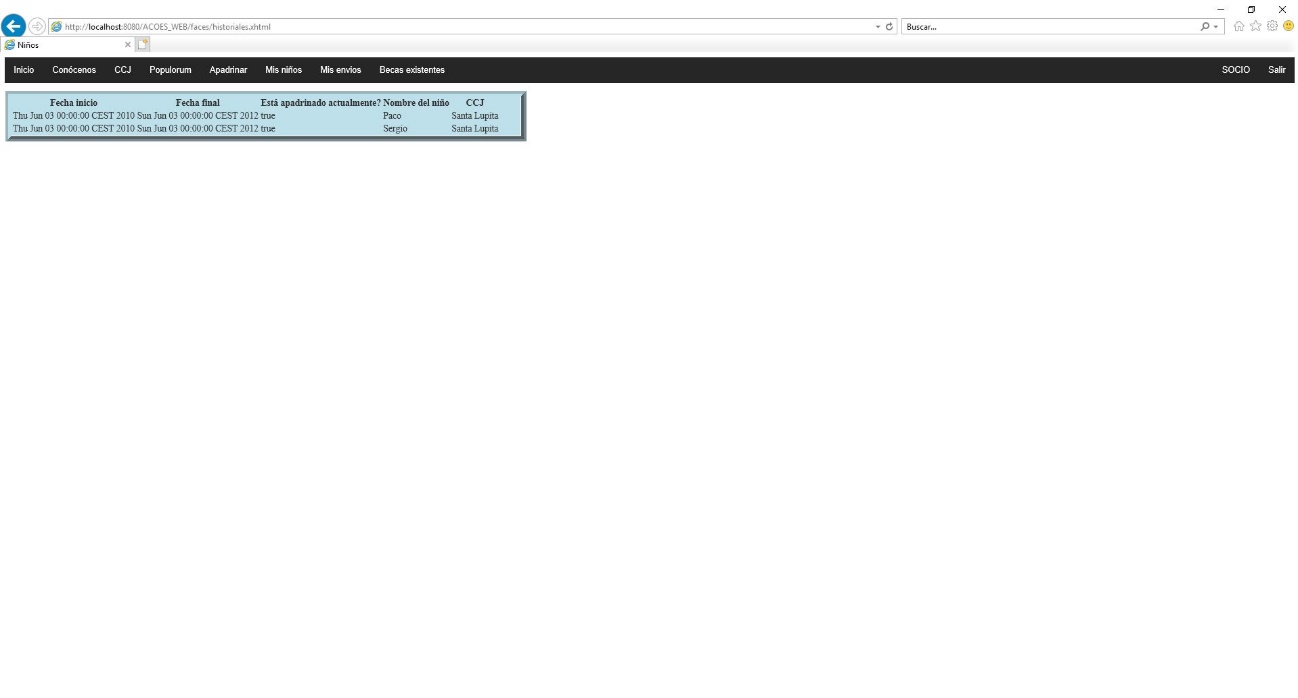
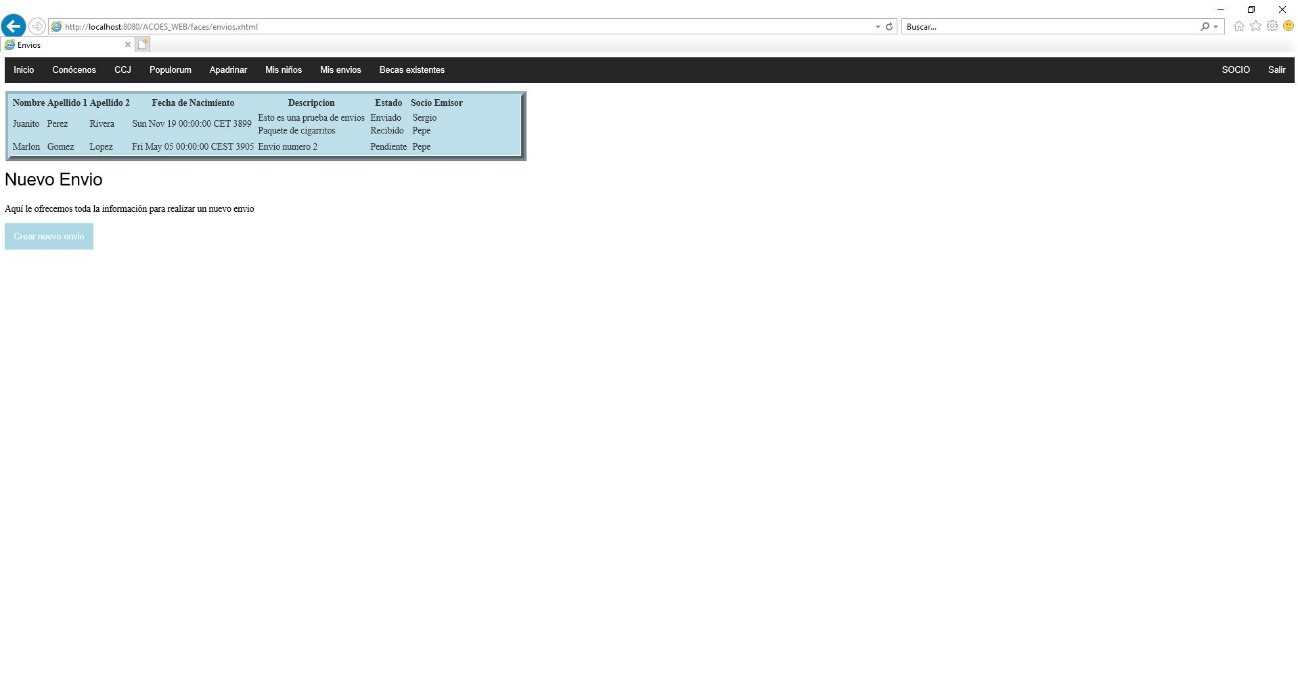
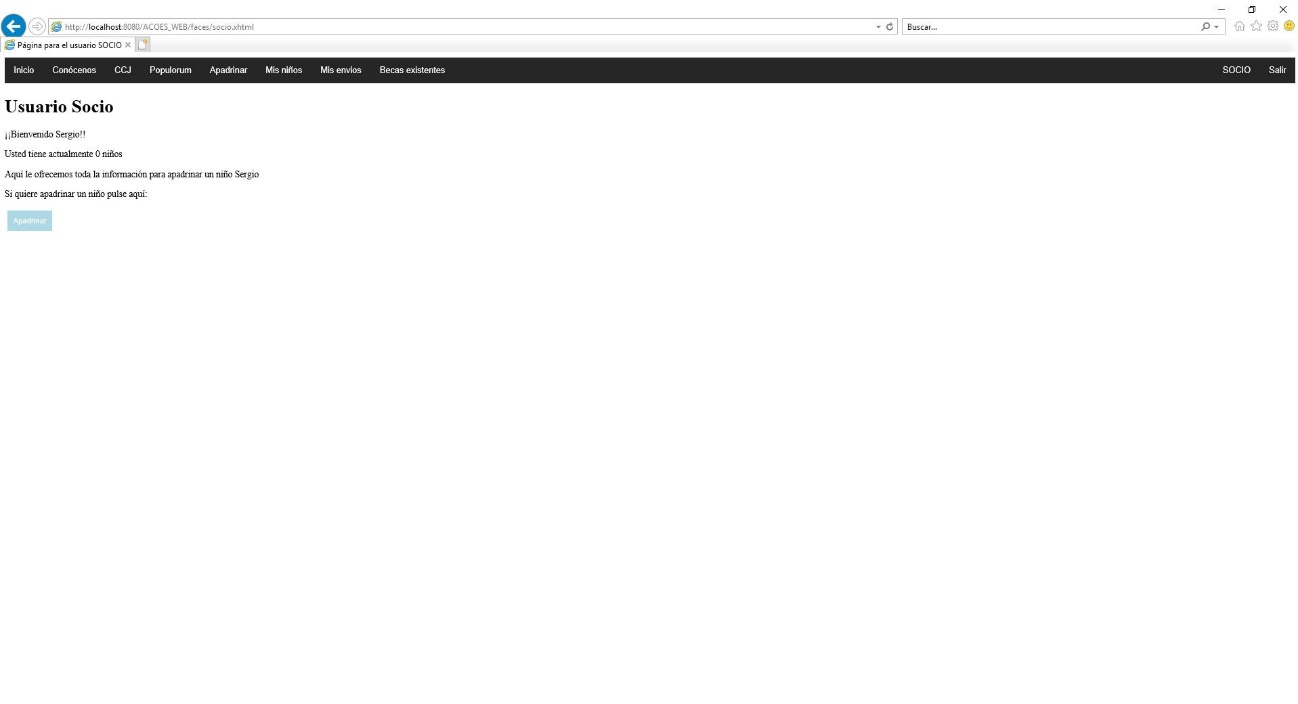
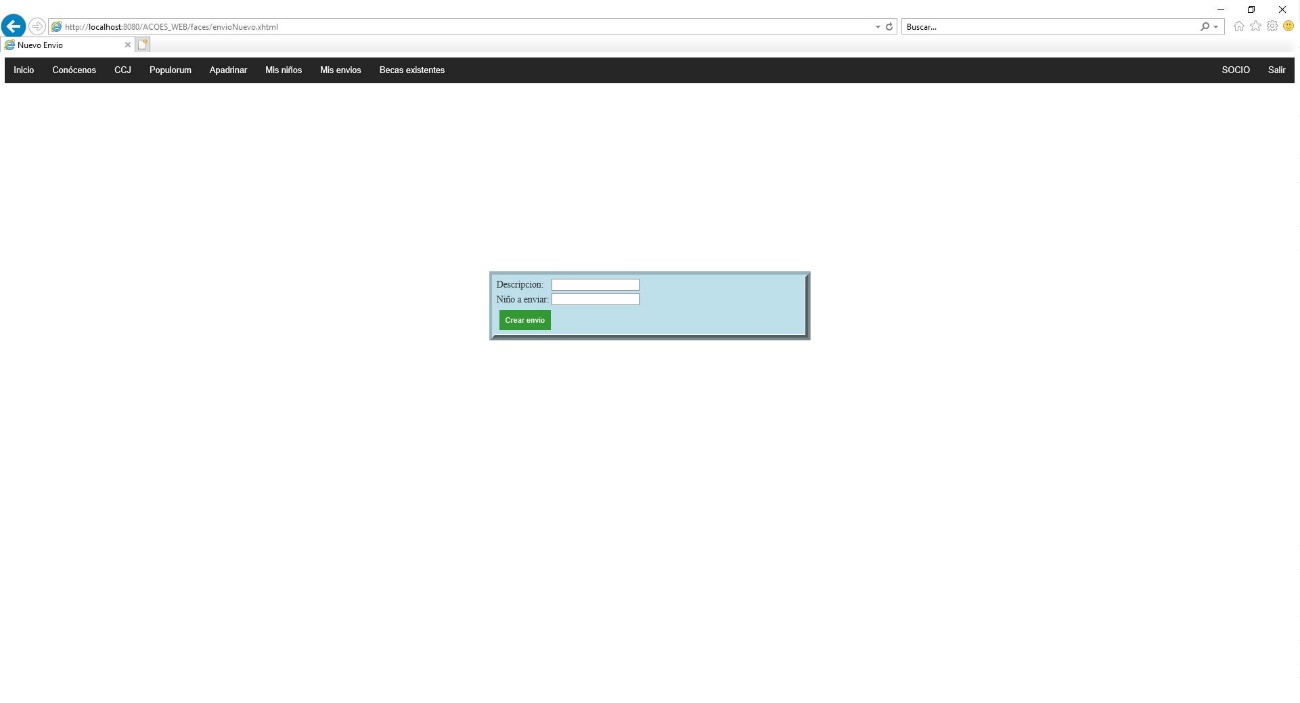
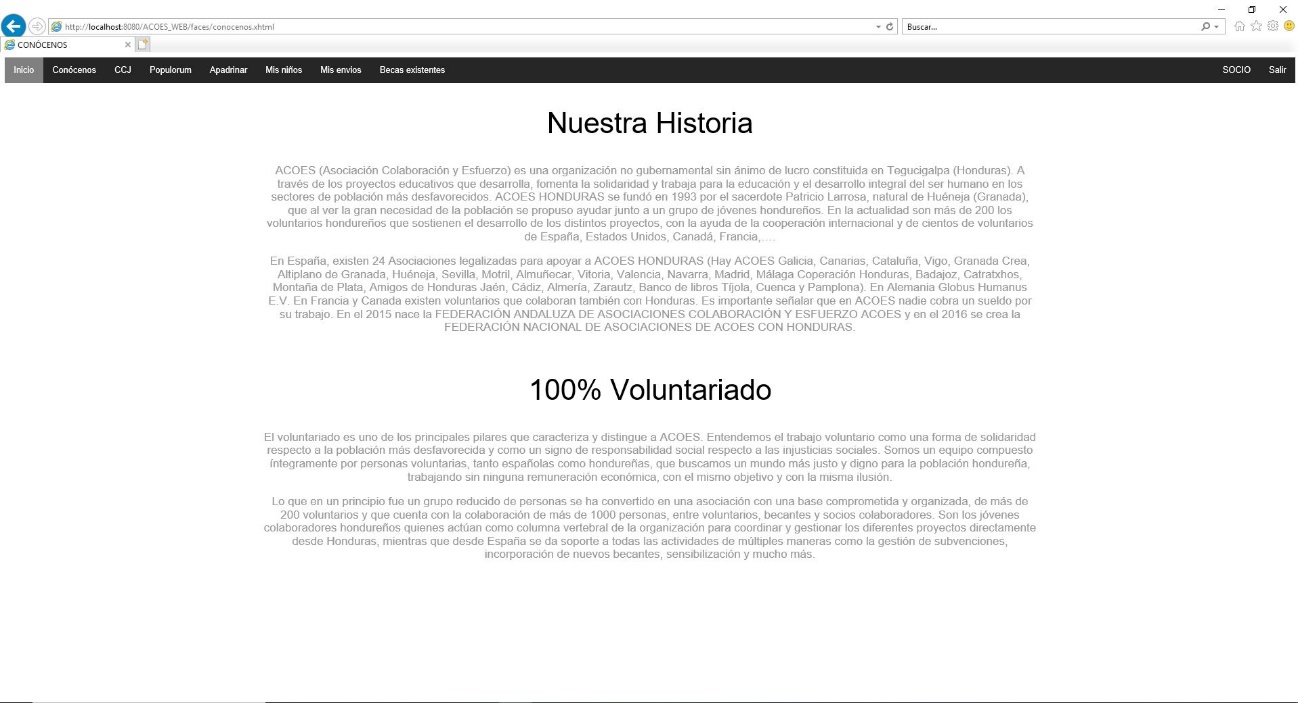
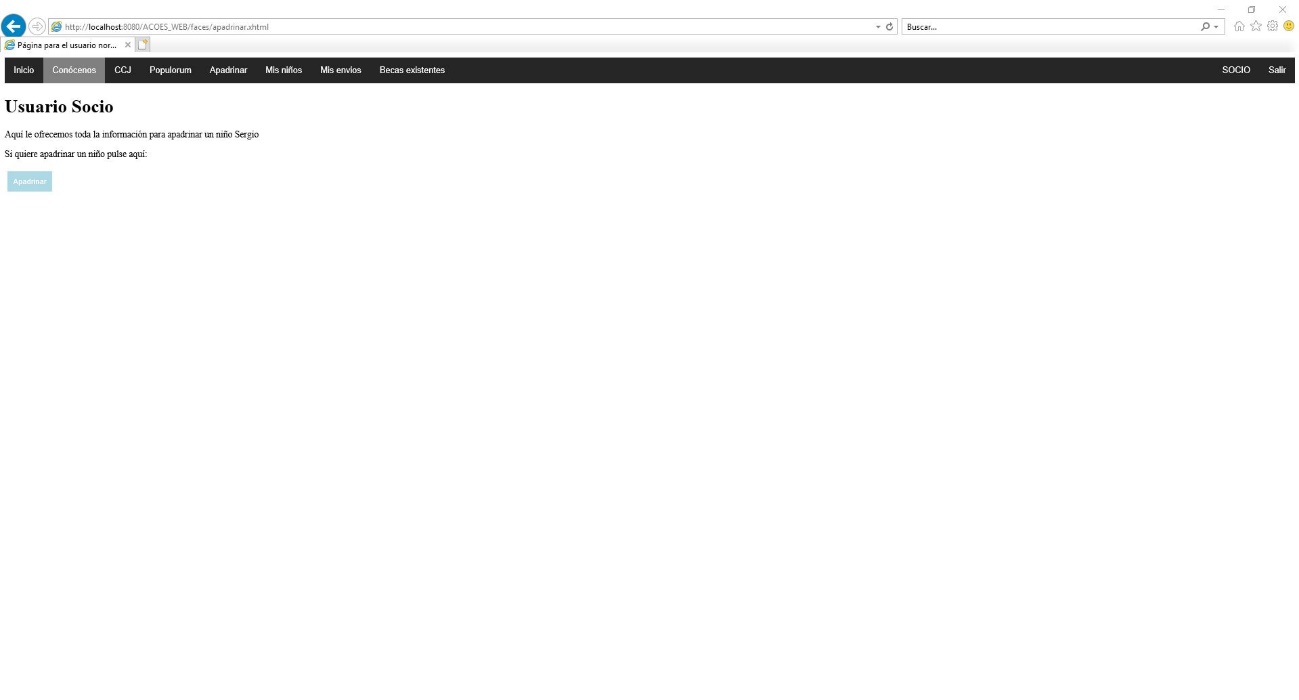


## Fotos usuario registrado**:**





## Fotos Socio:



## Fotos administrador:

