**Fundamentos de Arquitectura de Software**

**Departamento de Ciencias de la Computación y la Decisión**

**Facultad de Minas**

**Universidad Nacional de Colombia**

Proyecto Solución Mínima Viable

Versión: 2020.11.24 17:30

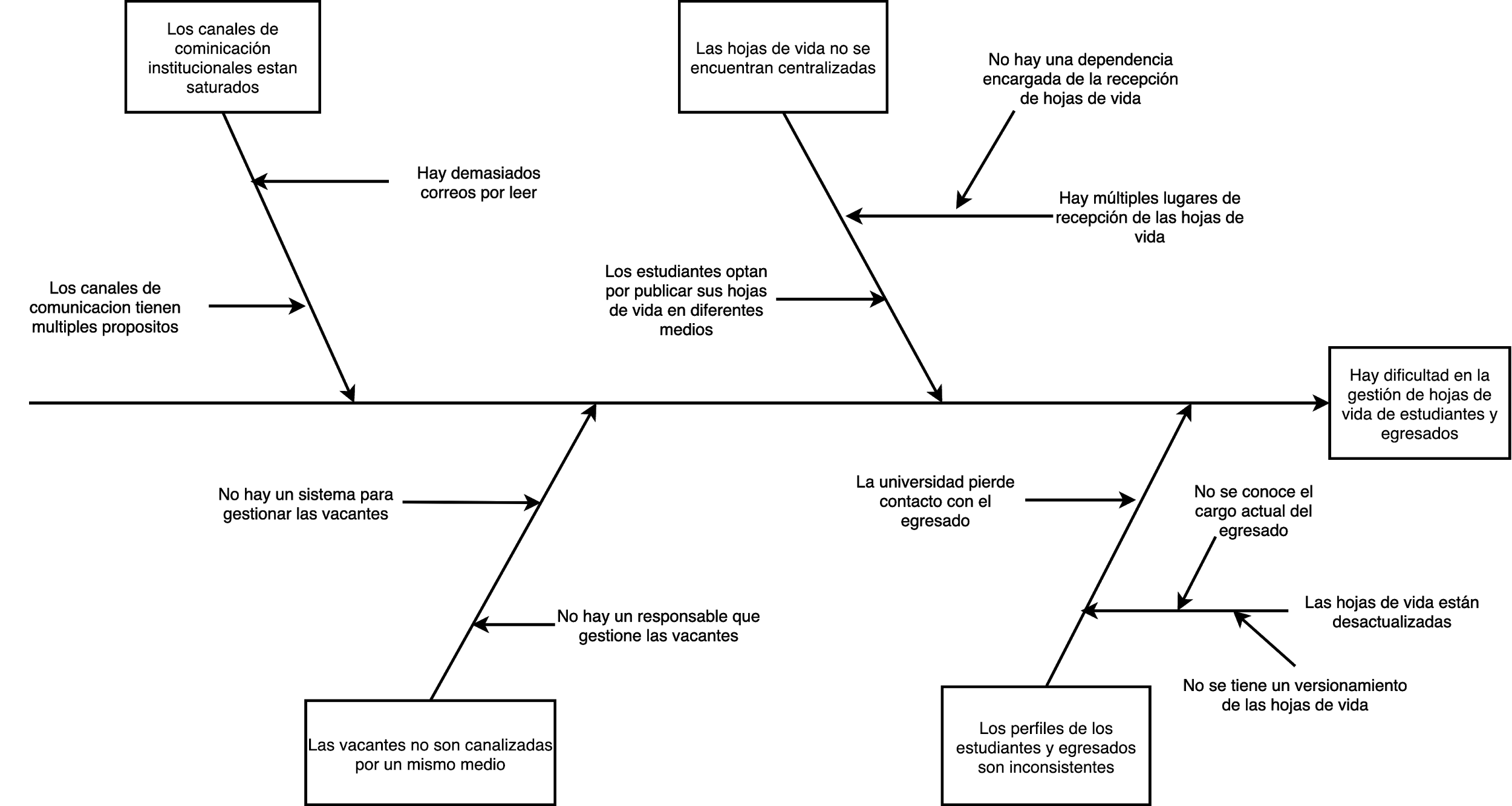
## Responsables

|  |  |  |
| --- | --- | --- |
|  | Nombre Completo | Rol |
| 1 | Jhojan Felipe Beltrán López |  |
| 2 | Julian Alejandro Rojas Betancur |  |
| 3 | Juan David Ruiz Echavarría |  |
| 4 | Jean Paul Yepes Guarnizo |  |
| 5 | Daniel Andrés Toto Aguirre |  |

Enlace de Github: <https://github.com/JulianRojasBet/hv_estudiantes>

Nota: Los diagramas que se plasman en este documento pueden ser encontrados en el Github compartido anteriormente. Pueden ser consultados en la carpeta *Diagramas* el archivo .drawio como la imagen correspondiente en formato .png

# Descripción y modelo Causa-Efecto del problema

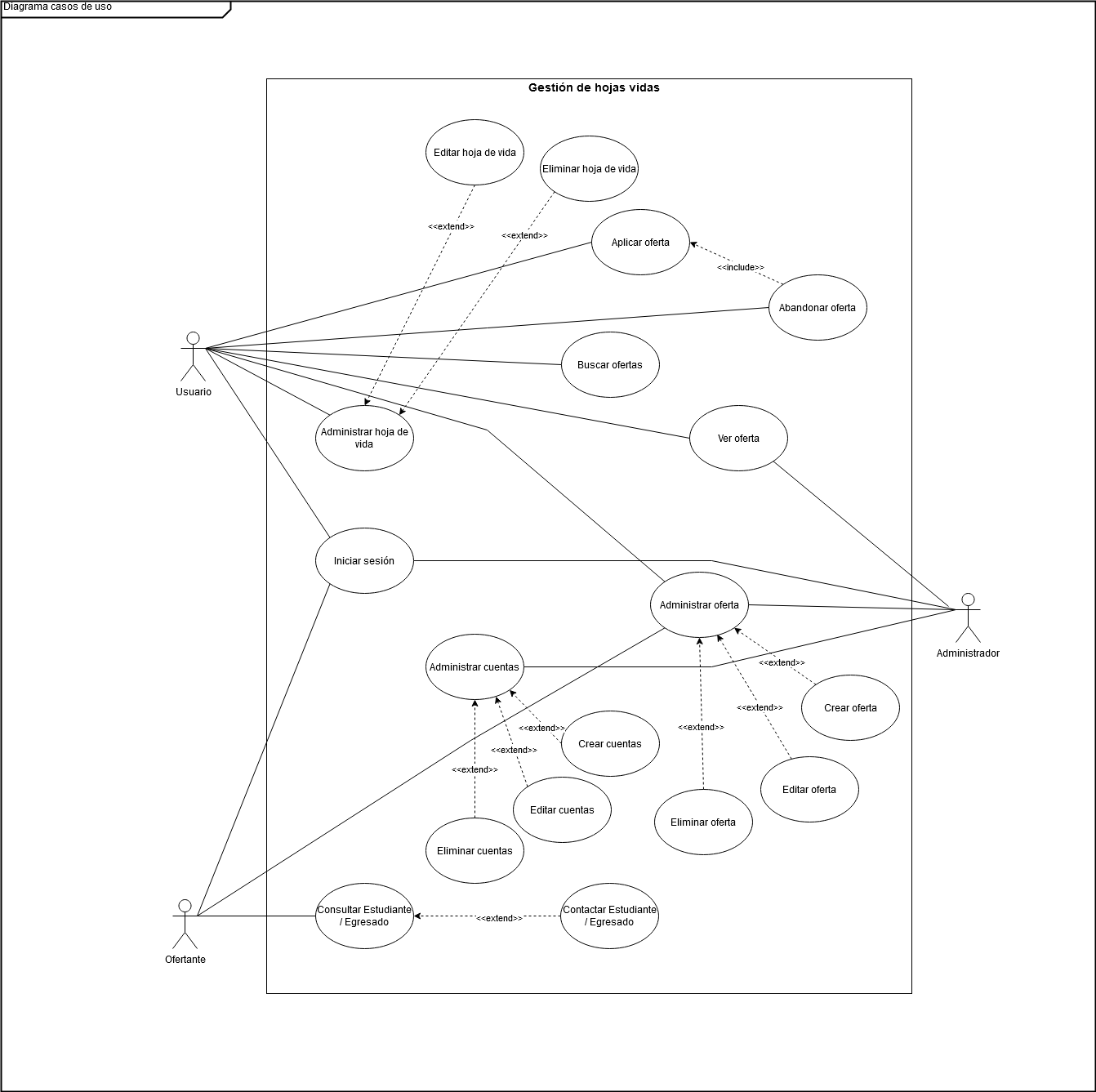


# Drivers arquitectónicos

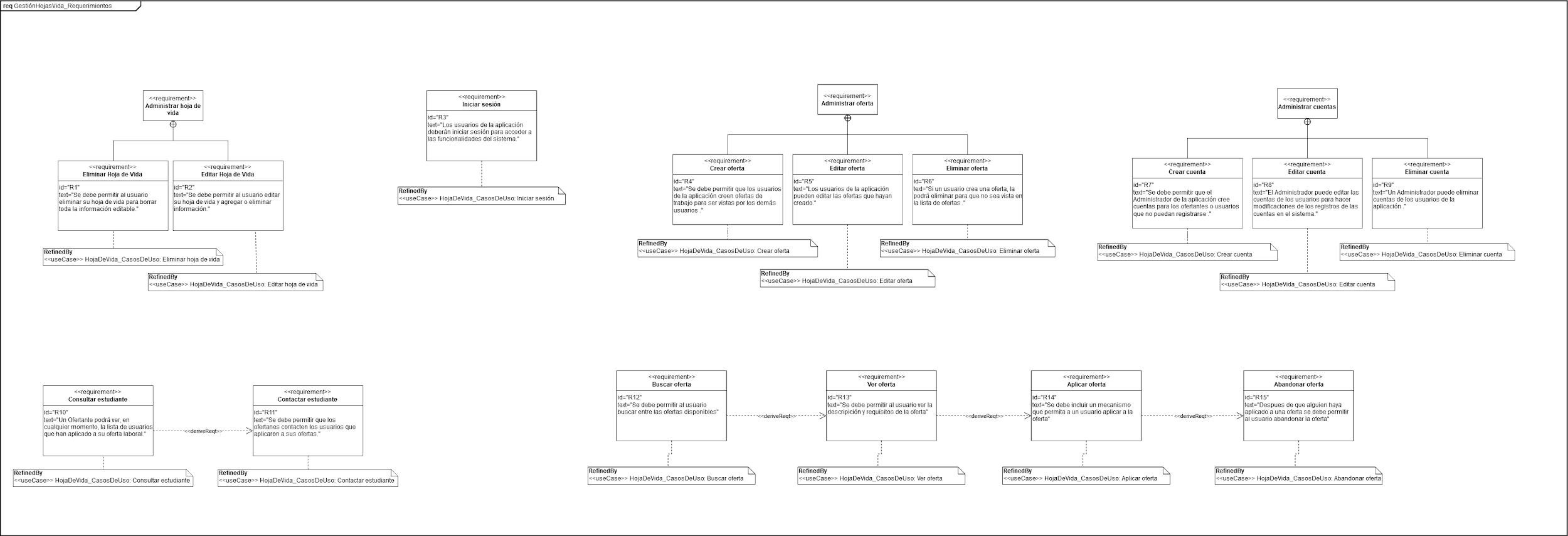
* 1. **Listado de Historias de Usuario**

Las historias de usuario fueron realizadas en Trello. Se encuentran disponibles en el repositorio compartido con el nombre de *Historias de usuario.pdf*

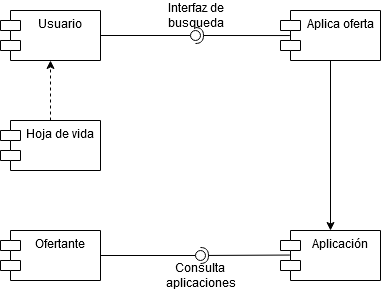
* 1. **Diagramas de Casos de Uso**

****

* 1. **Diagramas de Requisitos SYSML**

****

* 1. **Diagrama de Componentes UML**

****

# Modelo de Procesos del Prototipo

## Descripción verbal concreta del proceso

El usuario entra en la plataforma busca las ofertas, ve las ofertas que le interesan y aplican a la oferta. Puede editar su hoja de vida y agregar información relevante para ser expuesta a la hora de aplicar a una oferta.

Para el proceso de aplicación un ofertante debe crear una oferta, pero esta no se puede ver inmediatamente en la plataforma, se debe publicar la oferta para que los estudiantes puedan aplicar a la misma. Cuando se publica la oferta los estudiantes pueden aplicar la oferta y llega la fecha de vencimiento de la oferta y se estudian las hojas de vida y se seleccionan los candidatos.

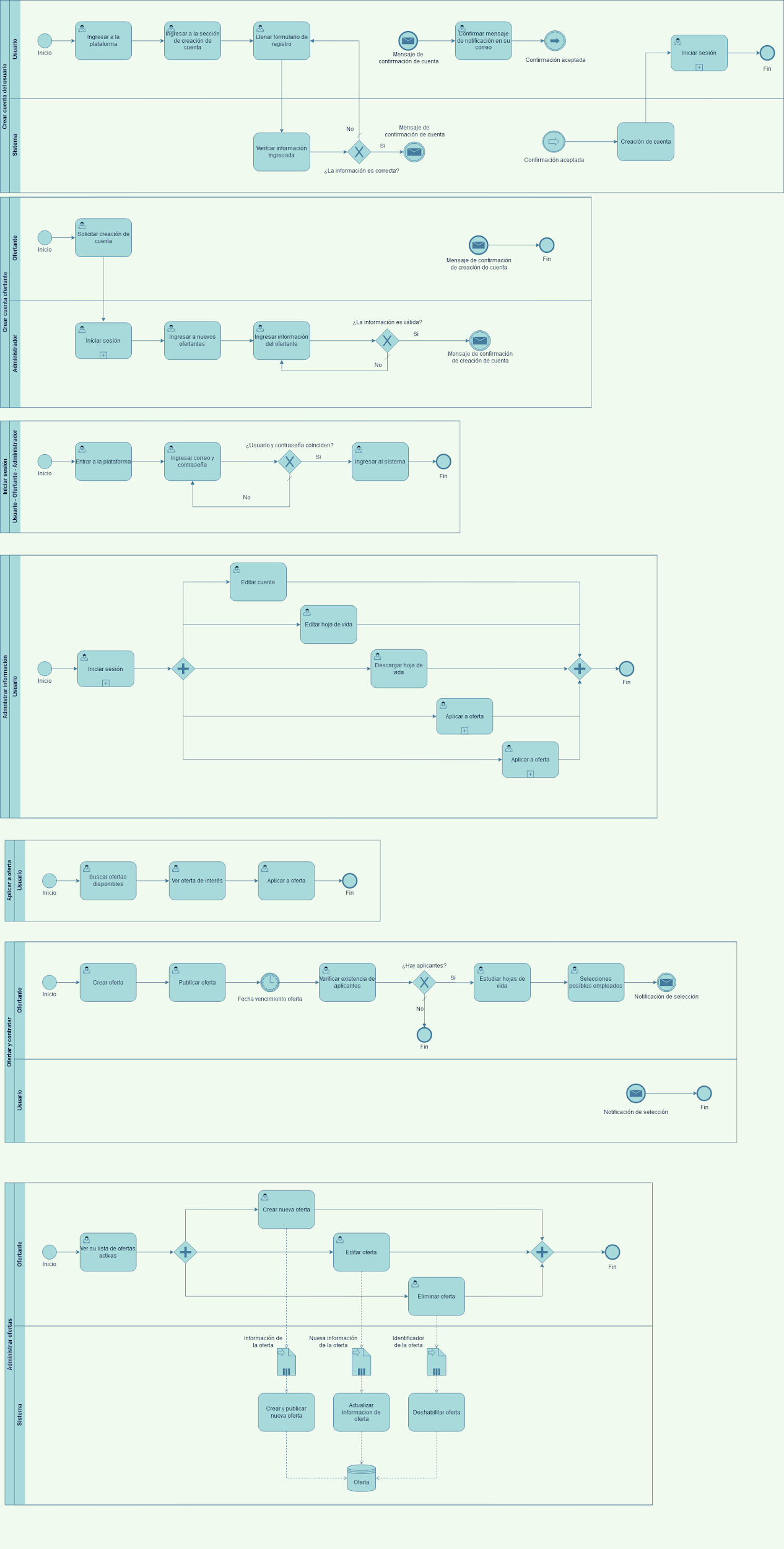
Además se tiene en cuenta de algunos inconvenientes que pueden surgir en la plataforma así que los ofertantes pueden editar o eliminar la oferta una vez se ha modificado.

Los administradores son los encargados de crear perfiles para ofertantes externos (empresas, organizaciones, terceros, etc.). Los estudiantes o egresados pueden también ofertar vacantes de trabajo.

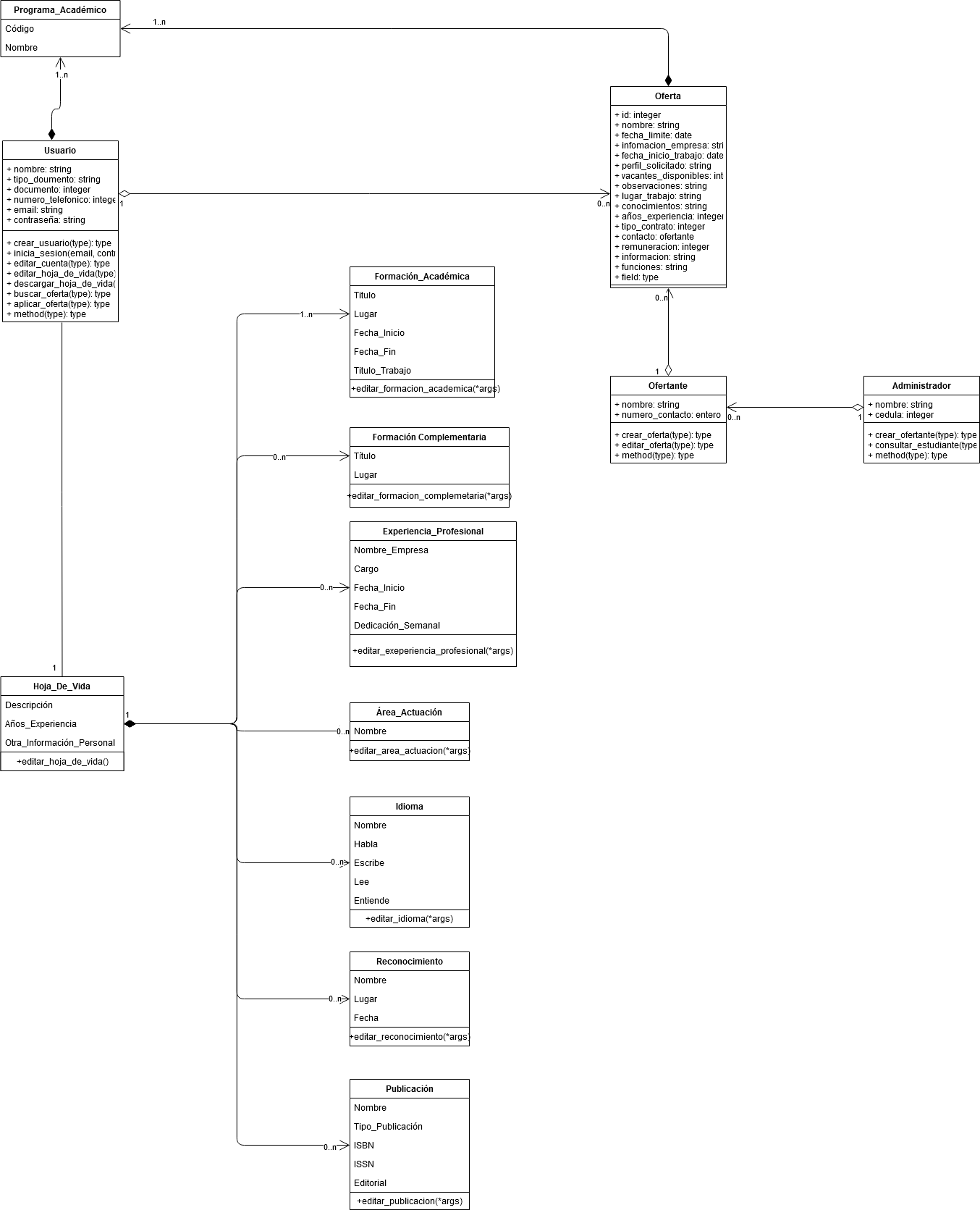
## Alcance del proceso

El alcance de la plataforma no incluye procesos de contratación de personal, está solo funcionará como intermediario para contactar ofertantes con personas interesadas en las vacantes. Los estudiantes de la Universidad Nacional podrán conseguir empleo o prácticas por cualquier otro medio, esta plataforma solo facilitará el proceso de que un ofertante y aplicante se comuniquen.

## Diagrama de procesos



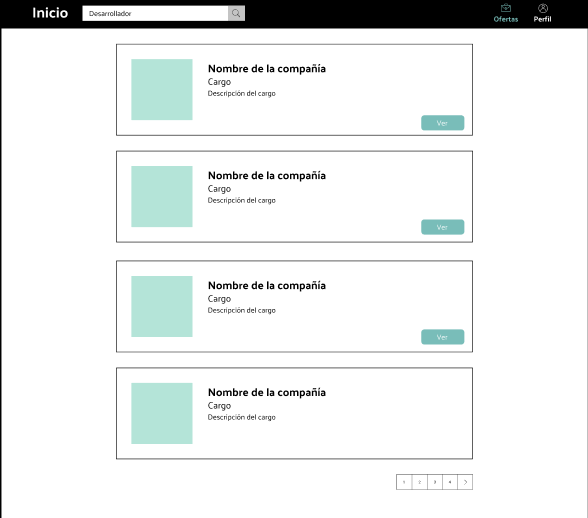
## Diagrama de Datos del Proceso

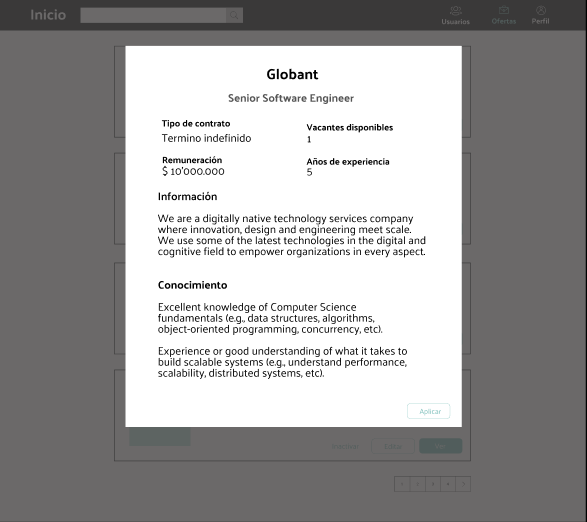


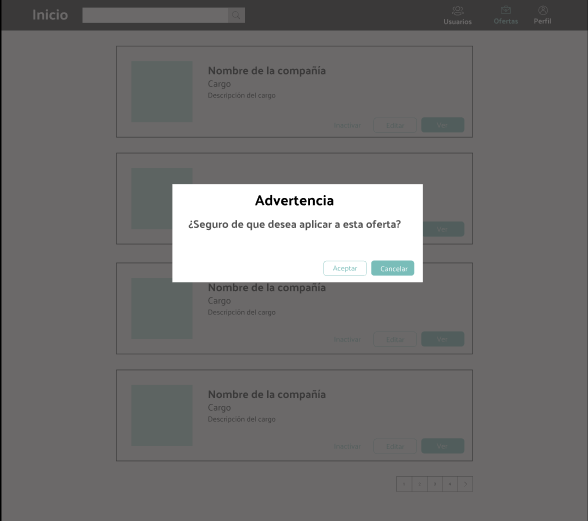
# Interfaces gráficas del Prototipo

## Formas por Actividad

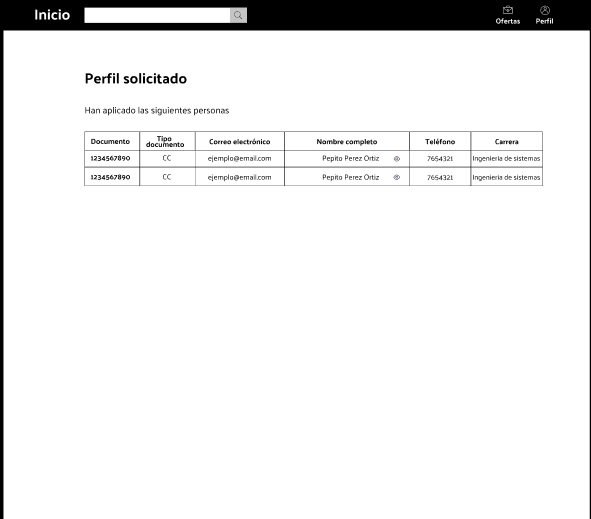
**Aplicar a oferta:**

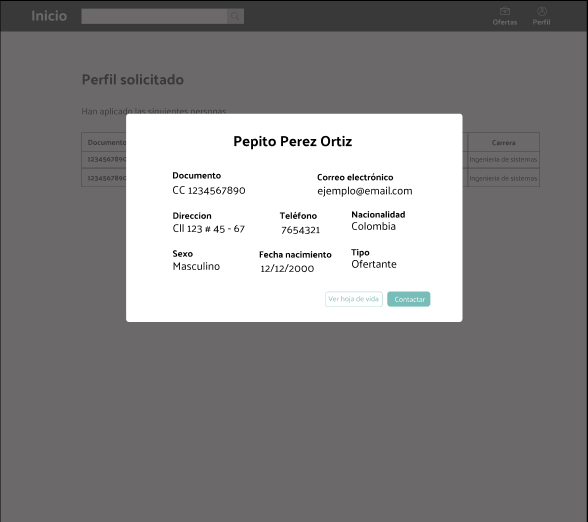




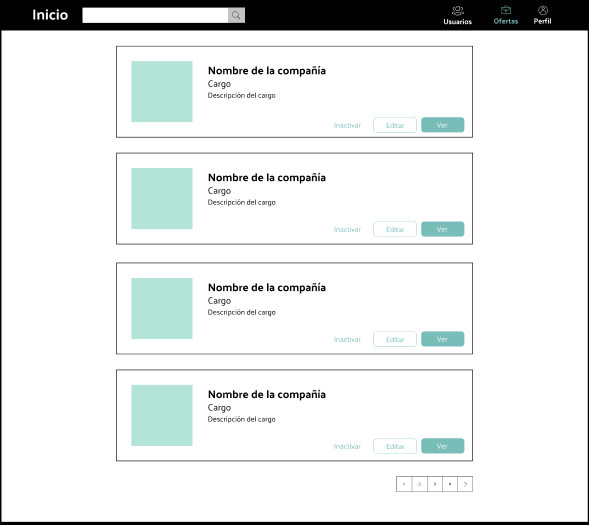


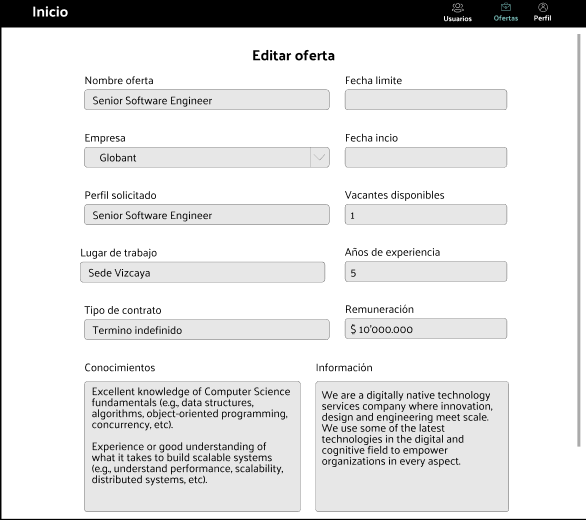
**Ofertante contacta a estudiante:**

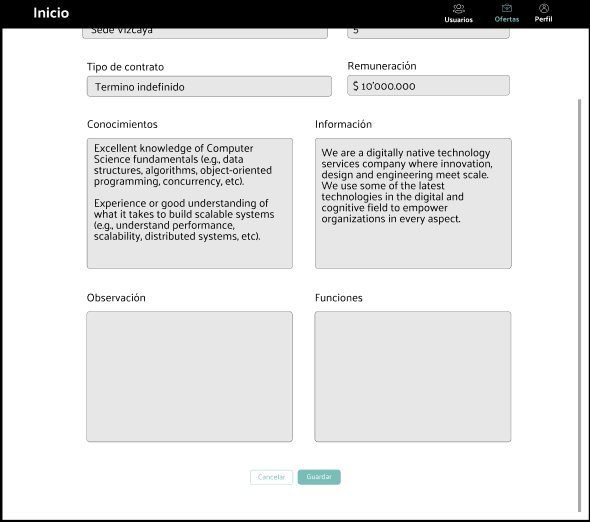


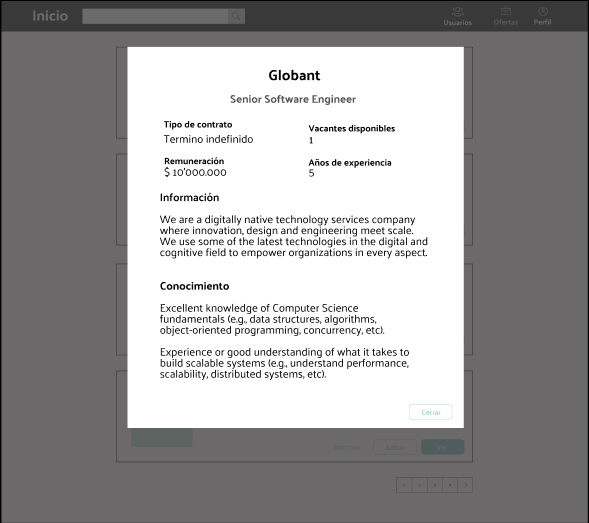


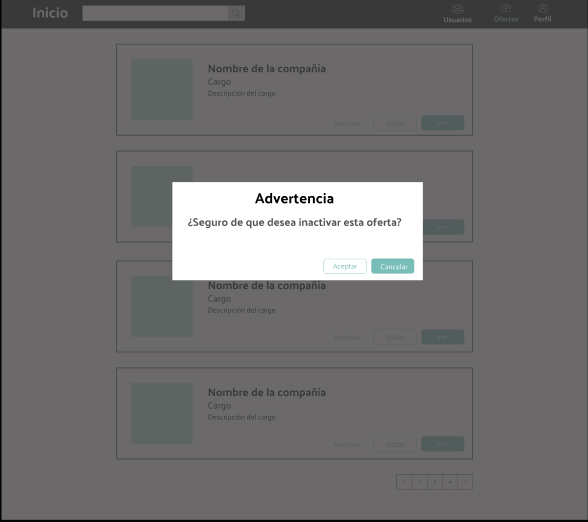
**Ofertante administra ofertas:**





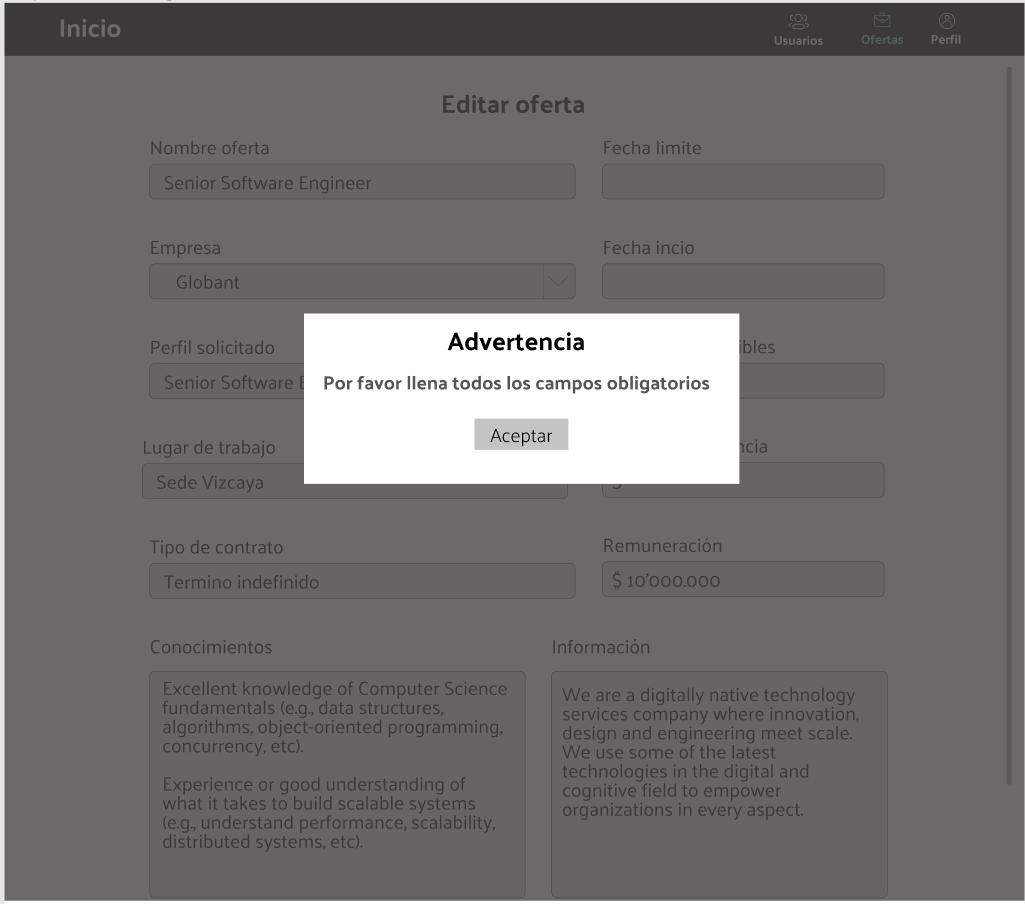


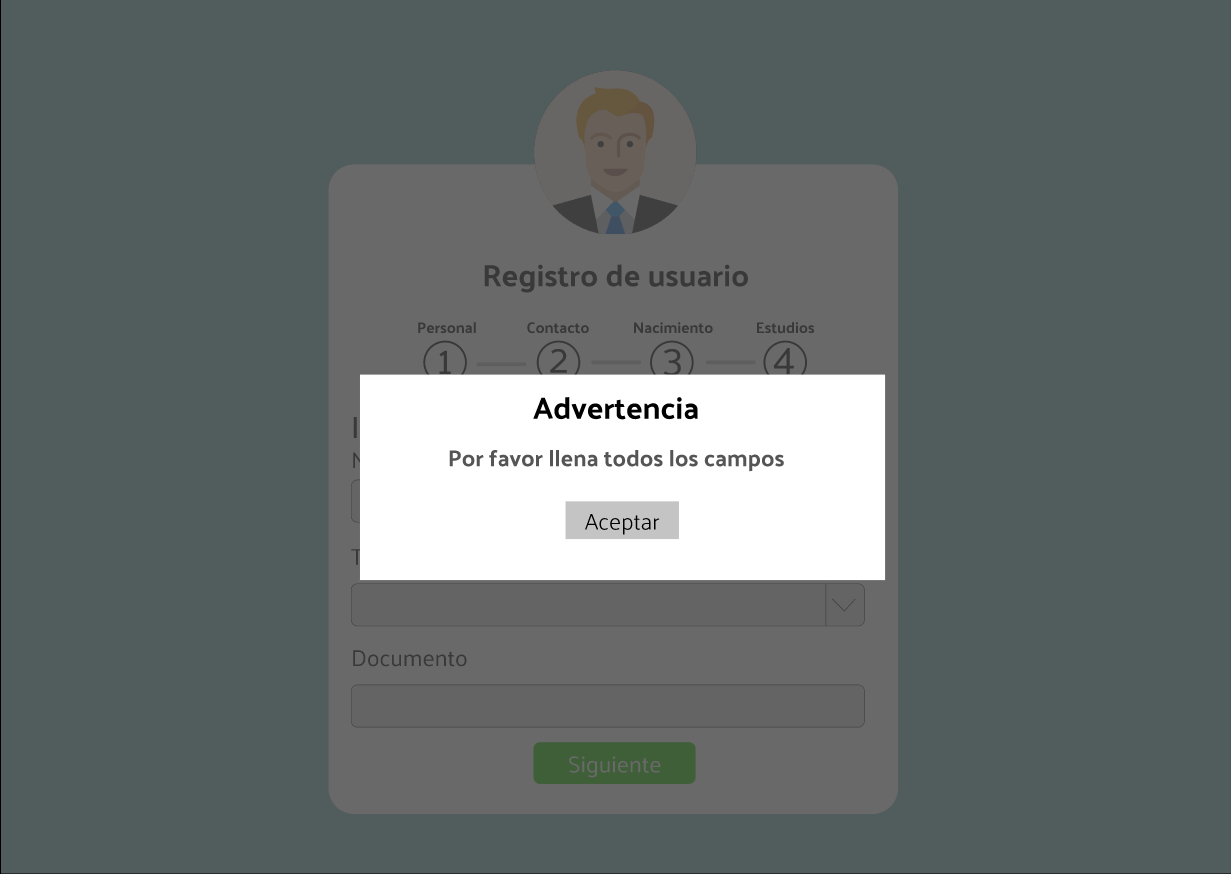


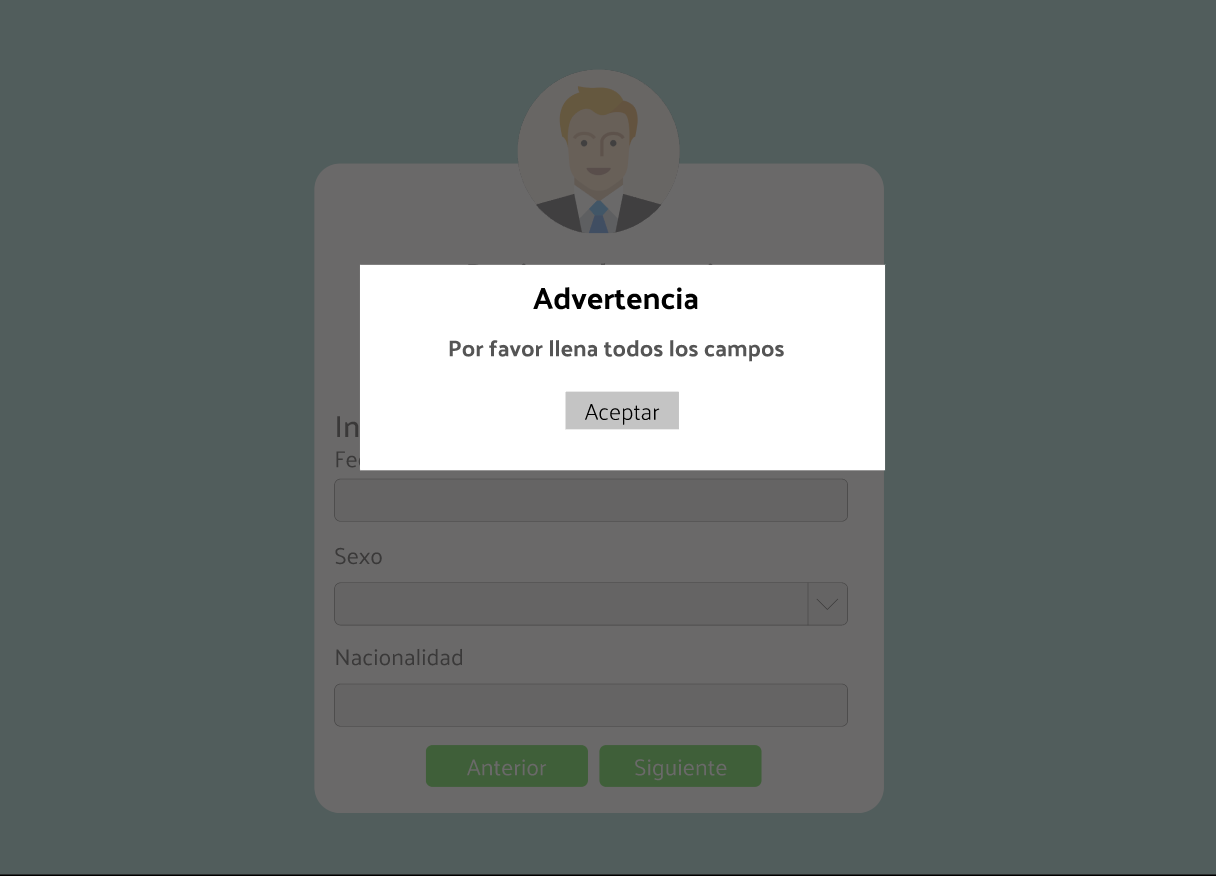


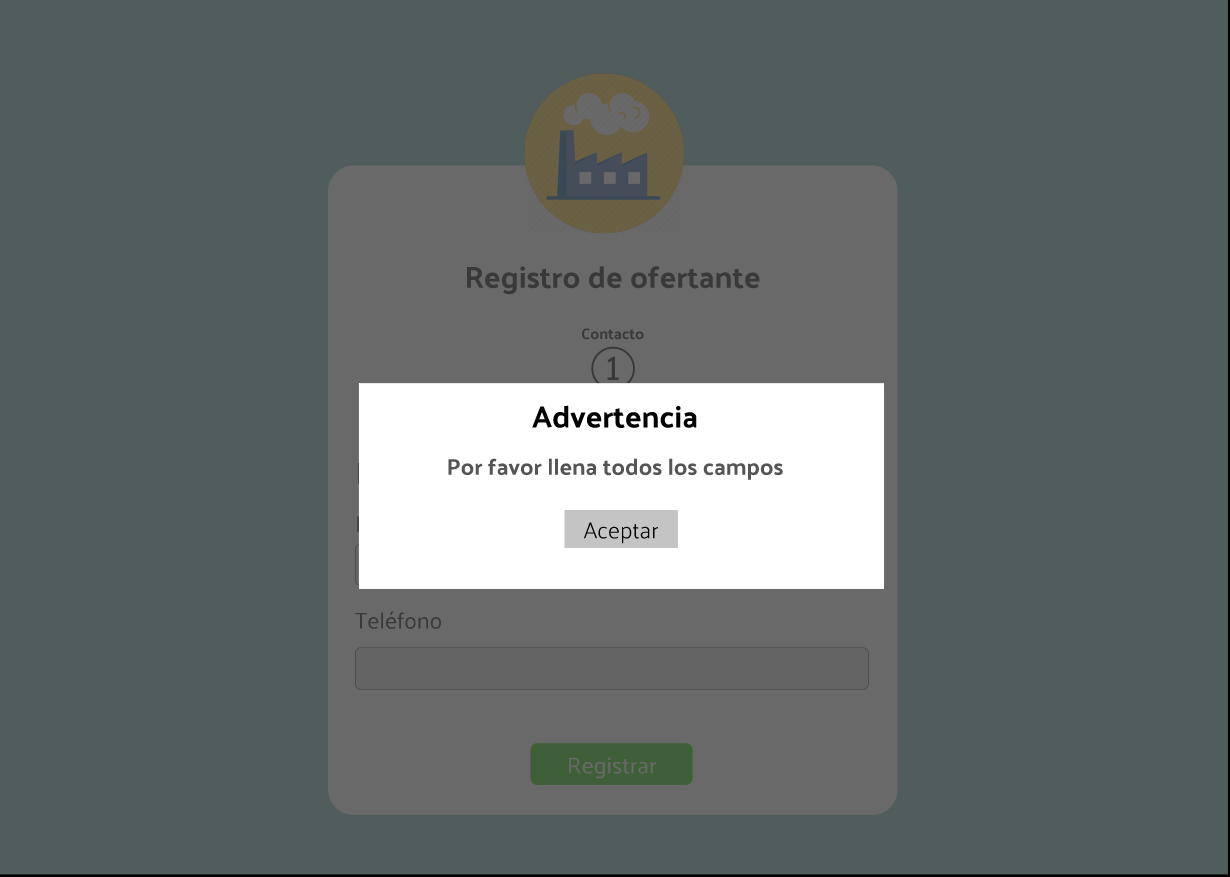
Las anteriores imágenes hacen referencia a las interfaces por actividad que fueron plasmadas en el diagrama de actividades. Podrá ver todas las interfaces del sistema en el archivo *Interfaces.pdf*, disponible en el repositorio compartido.

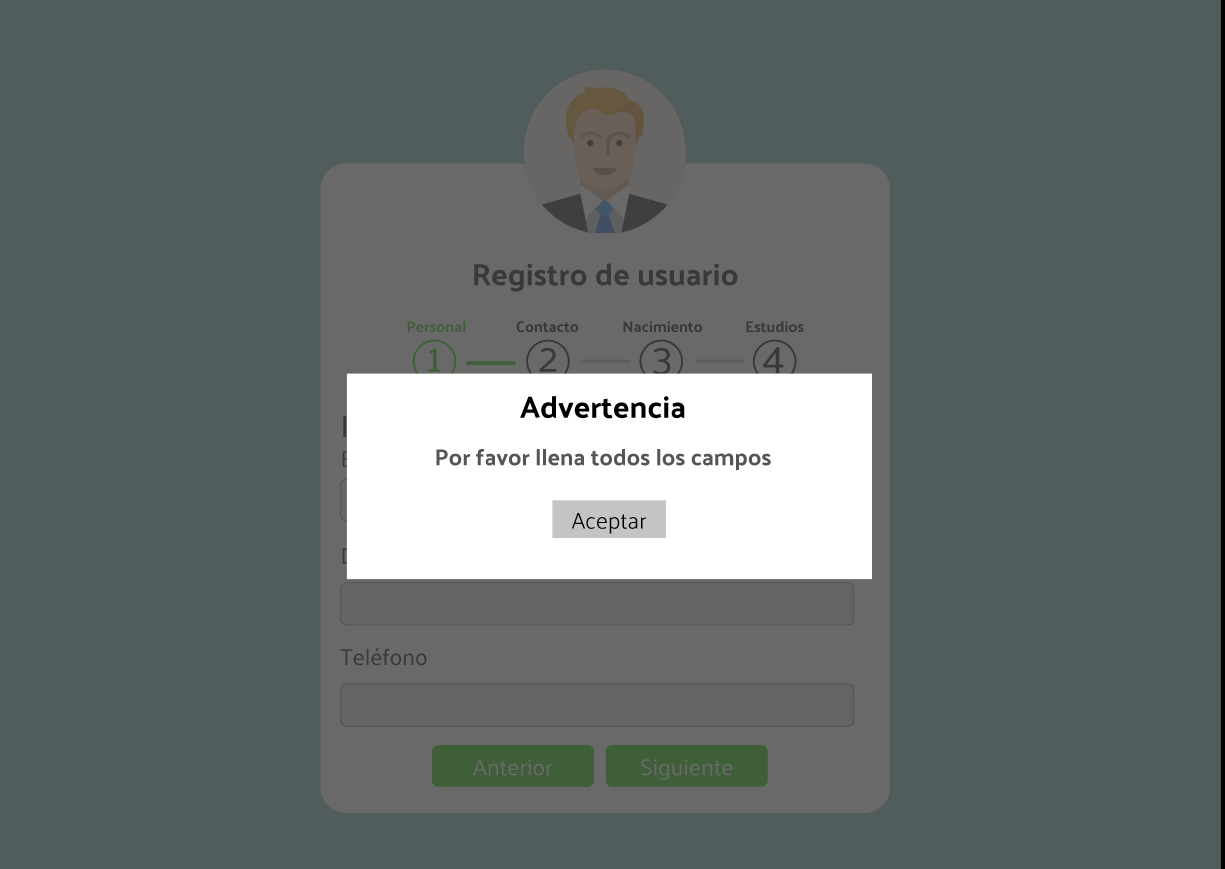
## Reglas de Negocio: Condiciones de Transición.

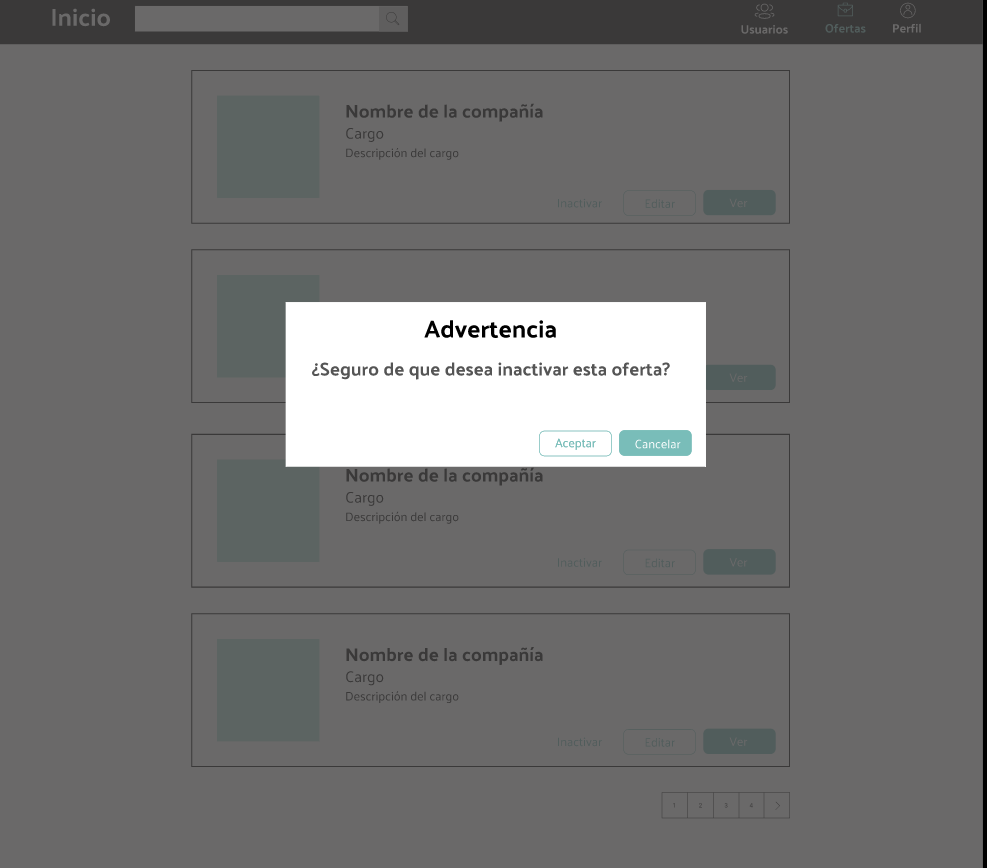
**

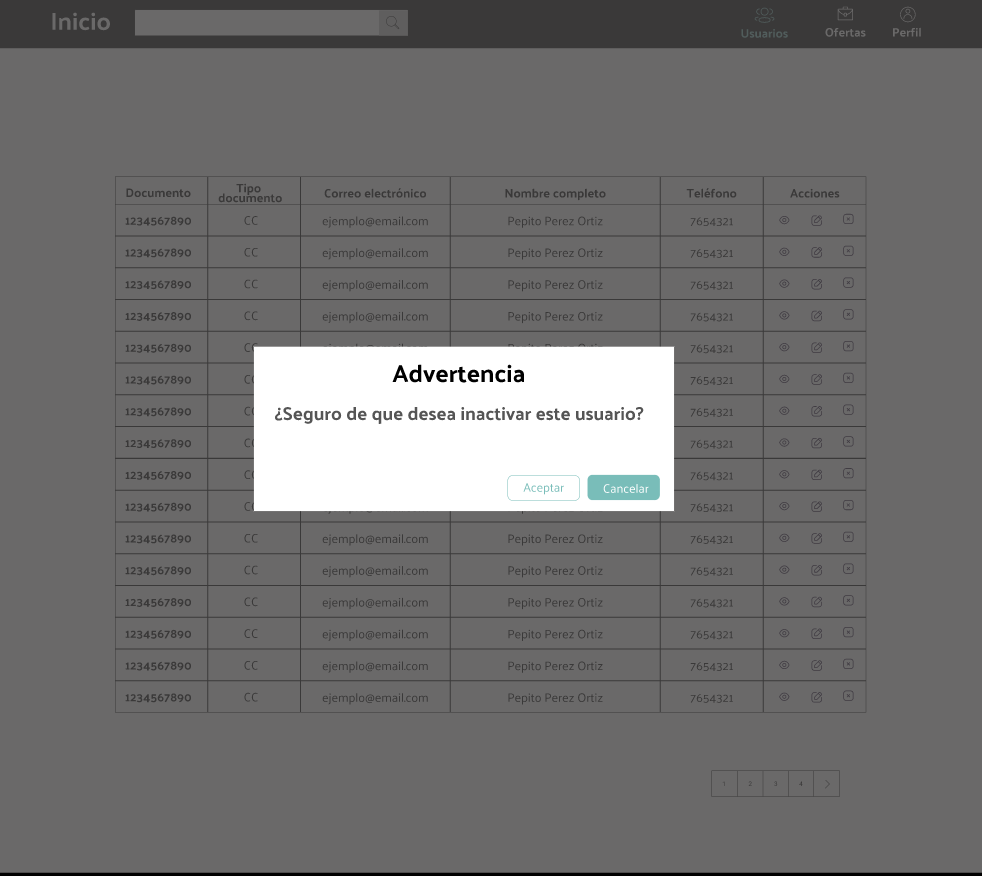
**

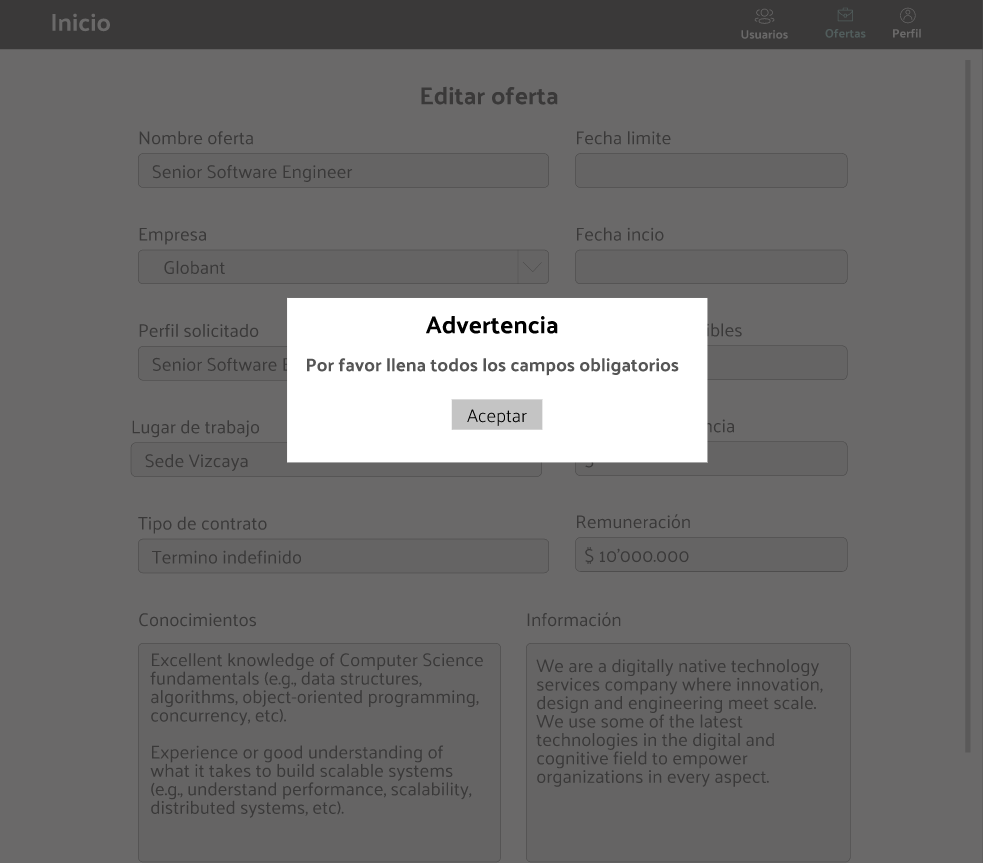
**

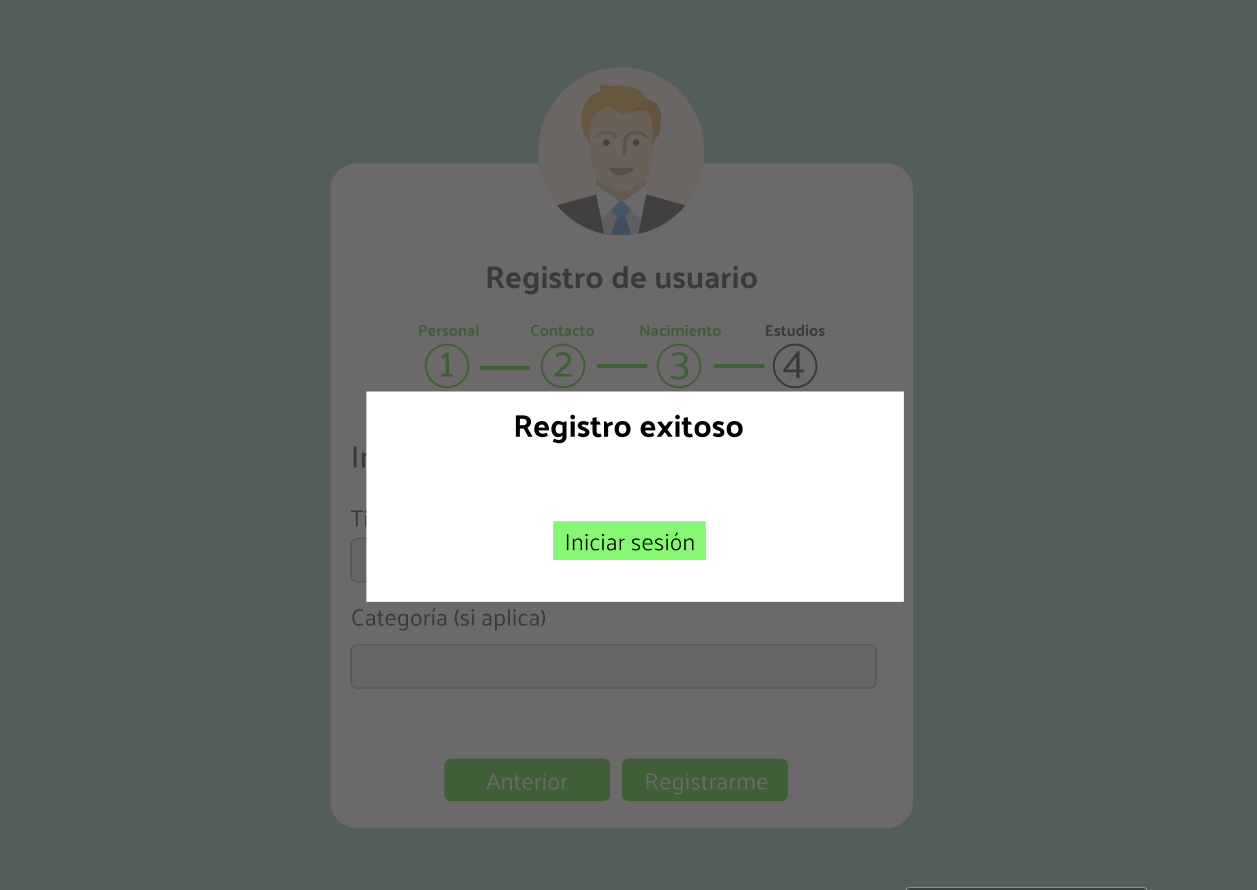
**

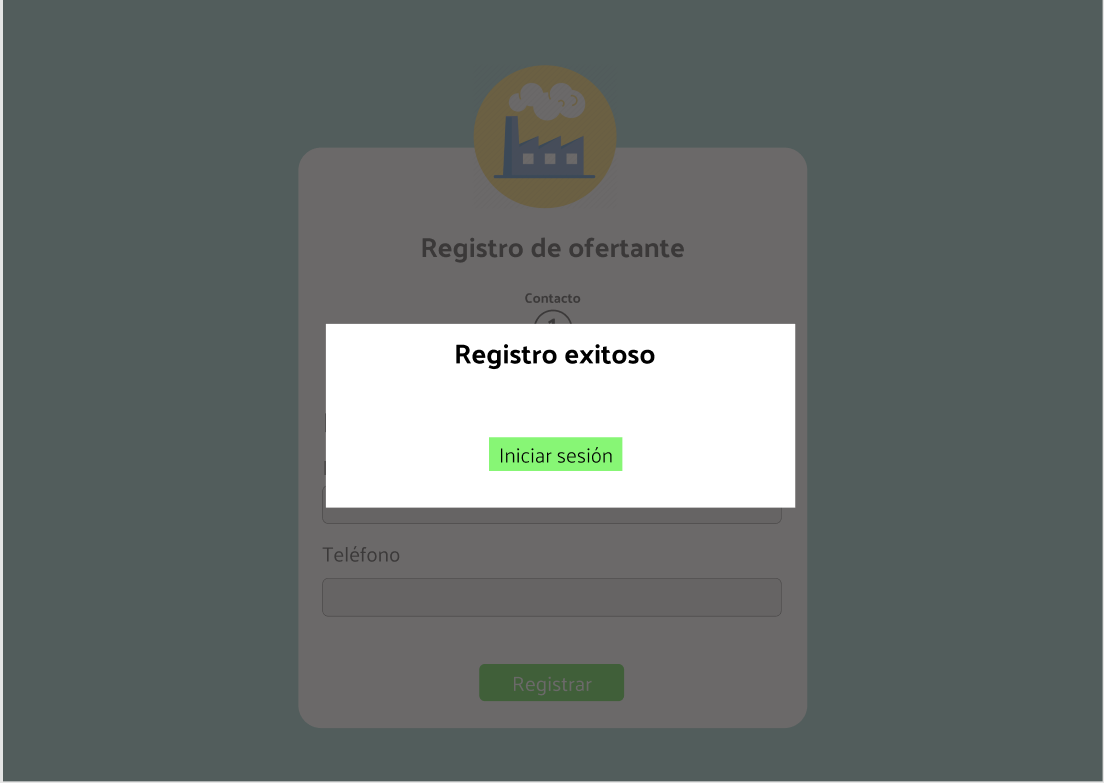
**

**

**

**

**

**

Además, se tiene que:

* Un usuario (estudiante o egresado) podrá hacer ofertas y por ende puede acceder a la interfaz de administración de ofertas.
* Un ofertante solo podrá entrar al sistema si le solicita, previamente, una cuenta al administrador.
* Un usuario solo puede ingresar al sistema si completa un registro exitoso.
* Cuando un usuario busca una oferta, entonces solo podrá ver las ofertas activas en ese momento.
* Si una oferta tiene aplicaciones al momento de vencer la convocatoria, entonces el ofertantes podrá acceder a la lista de aplicantes y contactarlos si así lo considera.

## Reglas de Negocio: Acciones de Actividad.

Las acciones de actividad ya fueron plasmadas en los **casos de uso**. Allí se muestran los requisitos funcionales y por ende no es necesario ponerlo en este punto.

# Participantes (asignación de Roles)

## Asignaciones

Se tienen tres roles: Usuario, Ofertante y Administrador. Cada rol podrá acceder a las siguientes interfaces:

El usuario podrá agregar eliminar y cambiar su propia hoja de vida, ver todas las ofertas creadas por los ofertantes (además de aplicar a las mismas o abandonarlas), y solicitar un cambio en su información básica a la administración del sitio.

El ofertante tendrá como principal actividad crear ofertas, también podrá modificarlas y eliminarlas, tendrá acceso a la información básica y hoja de vida de los usuarios que apliquen a sus ofertas y al igual que el usuario podrá solicitar un cambio en su información básica a la administración del sitio. Por supuesto, puede acceder a las interfaces para conocer los aplicantes a sus ofertas y contactarlas si lo considera necesario.

La administración del sitió podrá crear ofertantes ya que estos no se pueden crear así mismos como los usuarios, podrá editar la información básica de usuarios y oferentes y por último tiene completo poder sobre cualquier usuario, ofertante y oferta para eliminar estos.

# Integración con otras Aplicaciones

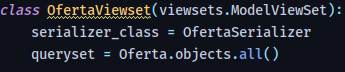
## Implementación de un servicio WEB

Para el servicio web se usa Python. En este lenguaje, usamos Django porque es un framework de fácil uso y orientado a la web. Además, se usa Django Rest Framework (DRF) para hacer los diferentes servicios web. Este framework se caracteriza por copiar muy poco código y se aprovecha de los modelos y las vistas, definidos en Django, para elaborar servicios de forma muy sencilla.

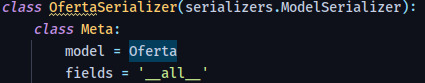
Modelo en Django:



Se define la vista de Oferta y observe que se especifica el Serializador que va a tener la vista. Ese serializador y la url permite definir el servicio web:



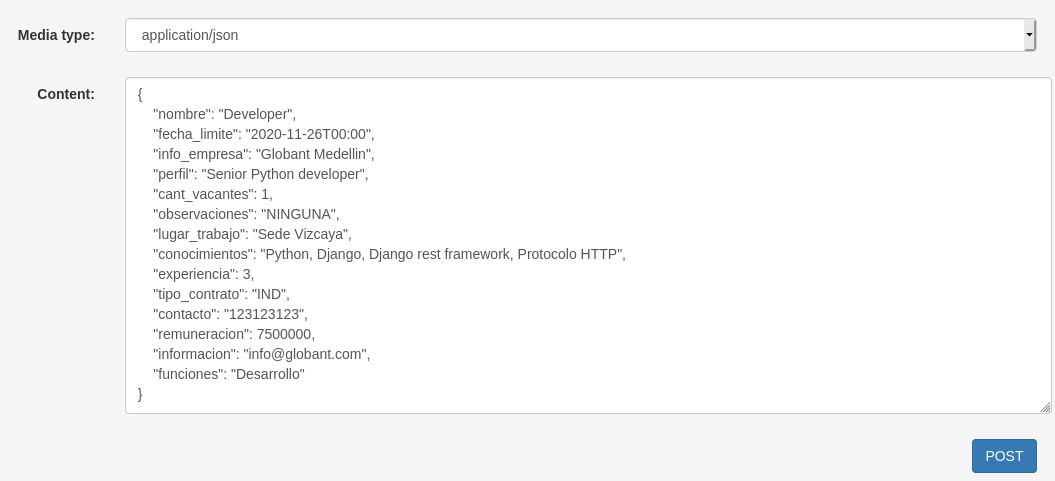
El serializador es algo muy sencillo y se ve de la siguiente manera:



Note que se define el modelo sobre el cual actúa, y en fields se definen los campos que se van a mostrar en el servicio web. Una vez se tiene eso, solo queda definir la ruta (api path) desde donde va a quedar accesible para ser llamado. En este caso se define la siguiente ruta:



Finalmente, el servicio recibe esta información y esto es lo que se debe mandar cuando se invoca:



## Invocar el servicio Web

Una vez se tiene el servicio implementado, ya puede ser invocado por Django o herramientas externas (como el desarrollo Front en otro lenguaje). Si se llama el servicio desde la propia Api de DRF se podría observar lo siguiente (es un get a la ruta previamente mostrada):



A continuación, se muestra un fragmento de código en Js que se usa para hacer peticiones al servicio web mediante Ajax:

function doAjaxSubmit(e) {

var form = $(this);

var btn = $(this.clk);

var method = (

btn.data('method') ||

form.data('method') ||

form.attr('method') || 'GET'

).toUpperCase();

if (method === 'GET') {

// GET requests can always use standard form submits.

return;

}

var contentType =

form.find('input[data-override="content-type"]').val() ||

form.find('select[data-override="content-type"] option:selected').text();

if (method === 'POST' && !contentType) {

// POST requests can use standard form submits, unless we have

// overridden the content type.

return;

}

e.preventDefault();

var url = form.attr('action');

var data;

if (contentType) {

data = form.find('[data-override="content"]').val() || ''

if (contentType === 'multipart/form-data') {

var boundaryCharNoSpace = "0-9A-Z'()+\_,-./:=?";

var boundaryChar = boundaryCharNoSpace + ' ';

var re = new RegExp('^--([' + boundaryChar + ']{0,69}[' + boundaryCharNoSpace + '])[\\s]\*?$', 'im');

var boundary = data.match(re);

if (boundary !== null) {

contentType += '; boundary="' + boundary[1] + '"';

}

data = data.replace(/\n/g, '\r\n');

}

} else {

contentType = form.attr('enctype') || form.attr('encoding')

if (contentType === 'multipart/form-data') {

if (!window.FormData) {

alert('Your browser does not support AJAX multipart form submissions');

return;

}

contentType = false;

data = new FormData(form[0]);

} else {

contentType = 'application/x-www-form-urlencoded; charset=UTF-8'

data = form.serialize();

}

}

var ret = $.ajax({

url: url,

method: method,

data: data,

contentType: contentType,

processData: false,

headers: {

'Accept': 'text/html; q=1.0, /'

},

});

ret.always(function(data, textStatus, jqXHR) {

if (textStatus != 'success') {

jqXHR = data;

}

var responseContentType = jqXHR.getResponseHeader("content-type") || "";

if (responseContentType.toLowerCase().indexOf('text/html') === 0) {

replaceDocument(jqXHR.responseText);

try {

history.replaceState({}, '', url);

scroll(0, 0);

} catch (err) {

// History API not supported, so redirect.

window.location = url;

}

} else {

// Not HTML content. We can't open this directly, so redirect.

window.location = url;

}

});

return ret;

}

Lo anterior, muestra una función que permite hacer peticiones al servicio web en cuestión. De allí, lo pedido se encuentra desde la línea 56 a la 65. Note que se hace uso de ajax para empaquetar las peticiones a la url definida.



# Ejecución del Producto Mínimo Viable

## Ejemplo de ejecución

Puede encontrar el video en el siguiente enlace: <https://drive.google.com/file/d/1eEutwfFfYgY7MnNfZqG2j_6zxpPDJHuY/view>