

OLIVER FABIAN STETCU

**DESARROLLO DE APLICACIONES
MULTIPLATAFORMA (DAM)**

ÍNDICE

1

- PORTADA (1)

2

- ÍNDICE (2-3)

3

- INTRODUCCIÓN (4)

4

- ANÁLISIS DE REQUISITOS (5-6)

- Objetivos del proyecto
- Alcance del proyecto
- Requisitos funcionales
- Requisitos no funcionales
- Planificación del proyecto

5

- NECESIDADES HARDWARE, SOFTWARE Y FINANCIACIÓN (7)

- Infraestructura hardware y software necesaria
- Recursos de interconexión
- Alternativas y mejoras posibles
- Presupuesto y financiación



6

- **DISEÑO DEL DIAGRAMA DE CASOS DE USO (8-10)**

- Obtención de los casos de uso
- Diagrama de casos de uso

7

- **DISEÑO DE LA INTERFAZ DE LA APLICACIÓN (MOCKUPS) (11-25)**

- Diseño de la aplicación web (si ocupa)
- Diseño de pantallas de la aplicación móvil

8

- **DISEÑO DE LA BASE DE DATOS (26-38)**

- Modelo entidad-relación
- Modelo relacional
- Creación de script para la base de datos y tablas
- Creación de script para insertar datos

9

- **ELEMENTOS ADICIONALES (39-47)**

- Sistema de Autenticación y Seguridad
- Sistema de gestión con altas, modificaciones, bajas, listar y búsqueda
- Tecnologías
- Varios niveles de acceso
- Posibles situaciones derivadas de los requerimientos

10

- **BIBLIOGRAFÍA Y ANEXOS (48)**



INTRODUCCIÓN

La gestión de **recursos humanos** es una tarea esencial en cualquier empresa y una de las funciones más importantes es la publicación y gestión de ofertas de trabajo. Actualmente, muchas empresas necesitan publicar ofertas de trabajo de manera rápida y efectiva.

Es por eso que he decidido brindar una **solución tecnológica** a este problema en concreto, desarrollando así una herramienta que permita a las empresas crear, editar, publicar y eliminar sus ofertas de manera más sencilla y eficiente, así como gestionar los candidatos que se postulen para las mismas.

El **objetivo principal** de esta aplicación es facilitar el proceso de publicación de ofertas de trabajo y la gestión de los candidatos para las empresas, permitiéndoles ahorrar tiempo y recursos en el proceso de selección, y permitiendo a los responsables de **recursos humanos** filtrar y seleccionar a los candidatos más adecuados para el puesto.

La aplicación contará con un **sistema de autenticación seguro**, así como distintos niveles de acceso y roles para los distintos usuarios, que permitirá a los usuarios acceder a la plataforma y gestionar las ofertas de trabajo según su rol en la empresa. Además, se implementarán sistemas de gestión que permitirán a los usuarios con roles determinados dar de alta nuevas ofertas, realizar modificaciones y bajas, buscar y listar las ofertas publicadas, entre otras funciones.

En este documento se presentarán los requisitos y especificaciones del proyecto, así como la planificación, necesidades de hardware y software, diseño de la base de datos, diagrama de clases, diseño de la interfaz y posibles mejoras.



ANÁLISIS DE REQUISITOS

Proyecto: Aplicación móvil de búsqueda y publicación de ofertas de trabajo.

Objetivo: Crear una plataforma para la búsqueda y publicación de ofertas de trabajo para facilitar la conexión entre empleadores y candidatos.

Requisitos Funcionales:

- Los **usuarios candidatos y empleadores** deben ser capaces de crear una cuenta y registrarse en la plataforma.
- Los **empleadores** deben ser capaces de publicar ofertas de trabajo con información detallada, incluyendo detalles y salario.
- Los **empleadores** deben ser capaces de modificar y eliminar ofertas de trabajo.
- Los **candidatos** deben ser capaces de buscar ofertas de trabajo por descripción, y aplicar a ellas mediante el envío de un CV.
- Los **candidatos** deben ser capaces de guardar ofertas de trabajo para un seguimiento posterior.
- Los **administradores** pueden gestionar y moderar las publicaciones de ofertas de trabajo. Así como la gestión y moderación de los usuarios.



Requisitos No Funcionales:

- La **plataforma** debe ser fácil de usar y navegar para los usuarios, así como tener una interfaz de usuario intuitiva.
- La **plataforma** debe ser segura y proteger la información personal y confidencial de los usuarios.
- La **plataforma** debe ser escalable y capaz de manejar un gran número de usuarios y ofertas de trabajo.
- La **plataforma** debe ser fácil de mantener y actualizar para garantizar que siga funcionando correctamente con el tiempo.
- La **plataforma** debe incluir una sección de ofertas destacadas, mostrando las ofertas más relevantes para los usuarios.
- La **plataforma** debe cargar rápidamente y ser eficiente en el uso de recursos.
- La **plataforma** debe cumplir con las regulaciones y leyes de protección de datos y privacidad.
- La **plataforma** debe ser accesible desde dispositivos móviles.

Planificación del Proyecto:

- Fase de investigación y análisis de requisitos (2 semanas).
- Fase de diseño y prototipado de la plataforma (4 semanas).
- Fase de desarrollo e implementación de la plataforma (8 semanas).
- Fase de pruebas y depuración (2 semanas).
- Fase de lanzamiento y puesta en marcha (1 semana).

Total de tiempo estimado para el proyecto: 17 semanas



HARDWARE, SOFTWARE Y FINANCIACIÓN

Para los recursos **Software** se ha utilizado **Android Studio** como herramienta de desarrollo, como lenguaje de programación **Java** para el desarrollo de la lógica de la aplicación y **XML** para el diseño de la interfaz gráfica de usuario. En cuanto a la infraestructura **Hardware**, se recomienda un dispositivo con un mínimo de 2 GB de RAM y 16 GB de almacenamiento interno para garantizar un buen rendimiento de la aplicación. Para correr la aplicación se necesitará un dispositivo móvil con sistema operativo **Android** versión 5.0 o superior.

Para la conexión de la aplicación con la base de datos, se ha optado por utilizar **SQLite**, que es una excelente opción para proyectos que requieren una base de datos ligera, fácil de integrar y que no requieren grandes volúmenes de datos. Para la interconexión de la infraestructura, se recomienda utilizar una conexión a Internet de alta velocidad para garantizar una rápida transmisión de datos.

Como alternativas al diseño elegido podría ser el uso de otras herramientas de desarrollo, como **React Native** o **Xamarin**, o el uso de otros servicios de base de datos en la nube, como **Amazon Web Services** o **Microsoft Azure**. Algunas de las posibles mejoras podrían incluir la implementación de funciones de inteligencia artificial para ofrecer recomendaciones personalizadas a los usuarios, o permitir a los usuarios buscar ofertas cercanas a su ubicación.

El **presupuesto** para el desarrollo de la aplicación dependerá de diversos factores como la complejidad de la aplicación, los costos de hardware y software y el diseño de la aplicación. Se podría optar por financiación a través de préstamos bancarios, o por una estrategia de monetización a través de la venta de la aplicación o de servicios adicionales dentro de la aplicación.



DIAGRAMA DE CASOS DE USO

Para obtener el **Diagrama de Casos de Uso** de nuestra aplicación primero debemos identificar los actores y las funcionalidades que estarán presentes en el sistema.

Actores:

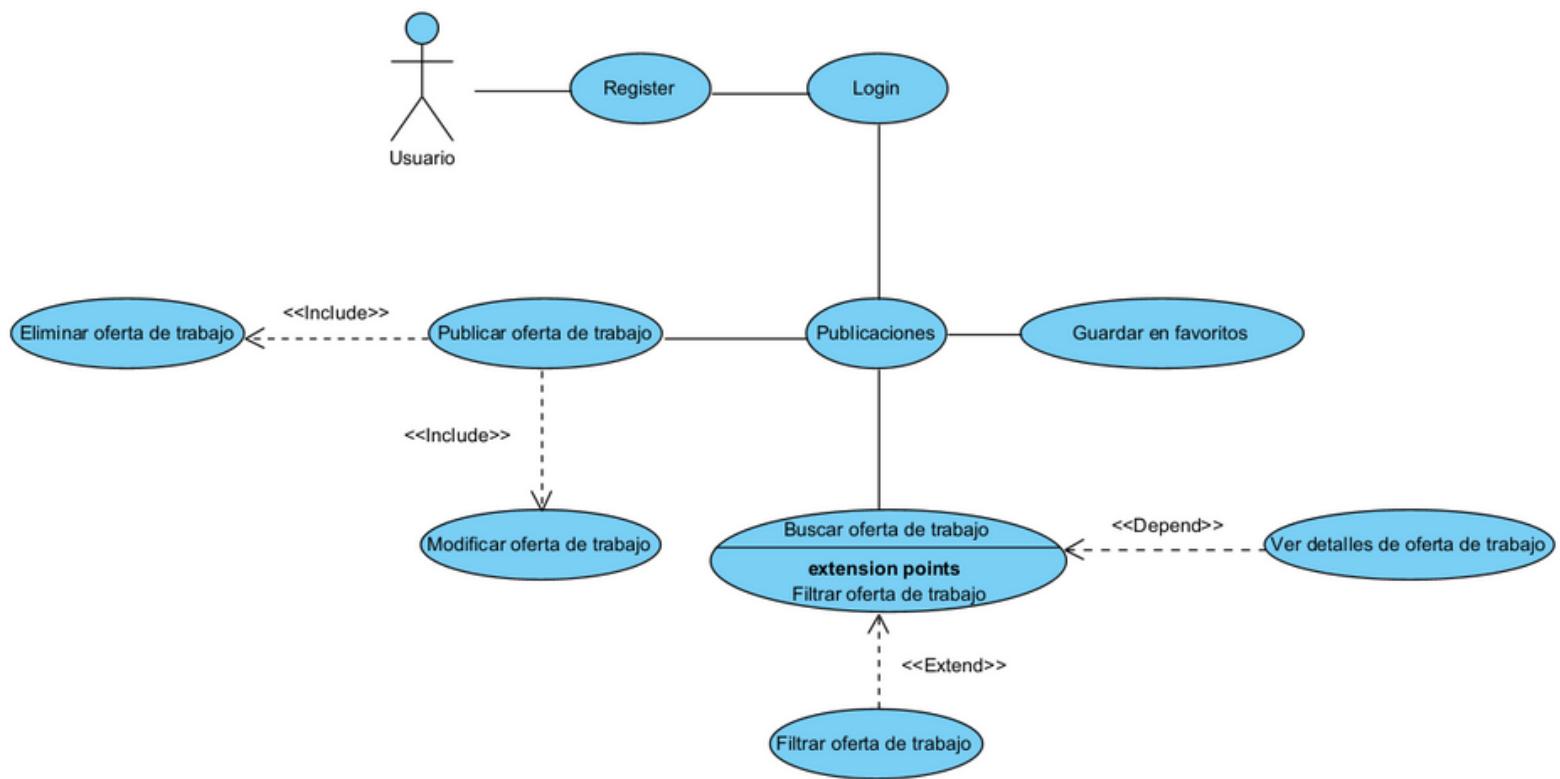
- **Usuario**: Representa a los usuarios de la plataforma, tanto **empleadores** como **candidatos** a ofertas de trabajo.

Funcionalidades:

- **Registro de usuario**: Los **empleadores** y **candidatos** podrán registrarse y loguearse en la plataforma para poder acceder a sus funcionalidades.
- **Publicar oferta de trabajo**: Los **empleadores** podrán publicar ofertas de trabajo con sus respectivos detalles.
- **Eliminar y Modificar oferta de trabajo**: Los **empleadores** podrán eliminar o modificar una oferta de trabajo publicada previamente.
- **Filtrar oferta de trabajo**: Los **candidatos** podrán filtrar las ofertas de trabajo según criterios específicos.
- **Ver detalles de oferta de trabajo**: Los **candidatos** podrán ver los detalles de una oferta de trabajo en particular.
- **Buscar ofertas de trabajo**: Los **candidatos** podrán buscar ofertas de trabajo que se ajusten a su perfil y preferencias.
- **Guardar en favoritos**: Los **candidatos** podrán guardar en favoritos una oferta de trabajo en particular.



Con estos elementos, podemos obtener el siguiente **Diagrama de Casos de Uso**:

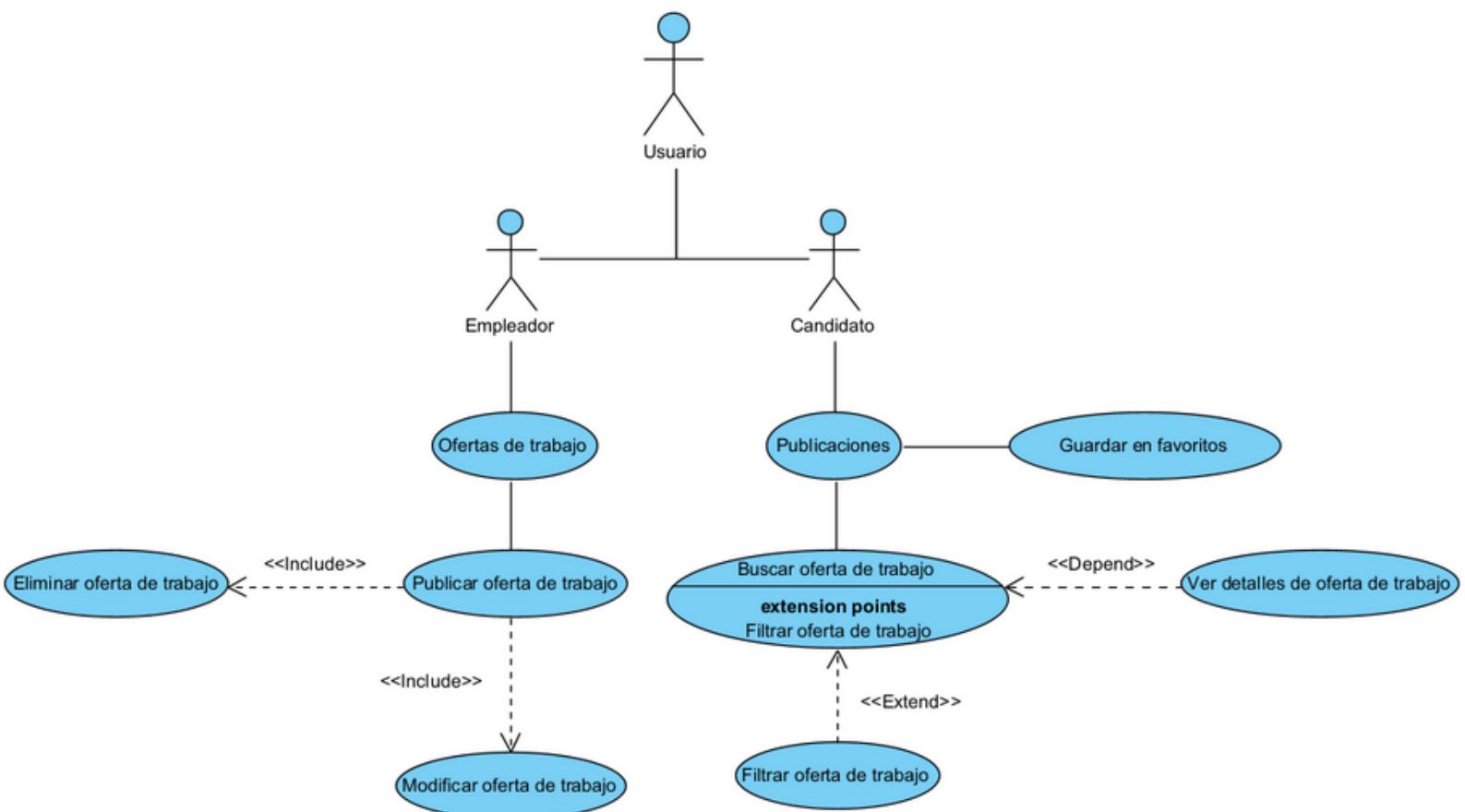


Las **relaciones y dependencias** son las siguientes:

- **Relación de inclusión:** La **relación de inclusión** indica que el caso de uso "**Publicar oferta de trabajo**" incluye al caso de uso "**Eliminar oferta de trabajo**". Es decir, para poder eliminar una oferta de trabajo, primero se debe haber publicado.
- **Relación de extensión:** La **relación de extensión** indica que el caso de uso "**Buscar oferta de trabajo**" puede extenderse a través del caso de uso "**Filtrar oferta de trabajo**". Es decir, el **usuario** puede optar por filtrar las ofertas de trabajo para obtener resultados más específicos.
- **Dependencia:** La **dependencia** indica que el caso de uso "**Ver detalles de oferta de trabajo**" depende del caso de uso "**Buscar oferta de trabajo**". Es decir, el **usuario** debe buscar una oferta de trabajo antes de poder ver sus detalles.



Teniendo en cuenta que nuestra aplicación integra **diferentes roles** (**administrador, empresa y candidato**), podemos hacer un diagrama de caso de uso parecido al anterior pero integrando estos roles, podría ser el siguiente:



En este **Diagrama de Caso de Uso** podemos observar lo siguiente:

- Los **empleadores** podrán publicar, modificar y eliminar ofertas de trabajo.
- Los **candidatos** podrán ver las publicaciones y guardarlas en favoritos, o buscar ofertas de trabajos filtrando por descripción, así como ver los detalles de dicha oferta.



DISEÑO DE LA INTERFAZ

He utilizado la herramienta **Ninjamock** para diseñar y desarrollar la interfaz de usuario de la aplicación móvil y el diseño de la aplicación web.

Ninjamock es una plataforma fácil de usar y altamente efectiva para crear prototipos interactivos de alta calidad que permiten visualizar y probar la funcionalidad y la estética de la aplicación antes de la implementación.

El **proceso de diseño** es una parte crucial del desarrollo de cualquier aplicación. El software ha permitido la creación de maquetas precisas, utilizando una amplia variedad de elementos personalizables, lo que me ha permitido asegurarme de que el diseño de la interfaz sea coherente y atractivo.

En resumen, la utilización de **Ninjamock** ha sido fundamental para el éxito de la aplicación, ya que me ha permitido crear una interfaz de usuario intuitiva y fácil de usar que esperamos que los usuarios encuentren atractiva y satisfactoria.

Estoy muy satisfecho con el resultado final y recomendaría **Ninjamock** a cualquiera que busque una herramienta eficaz y de alta calidad para el diseño de su aplicación.



NinjaMock



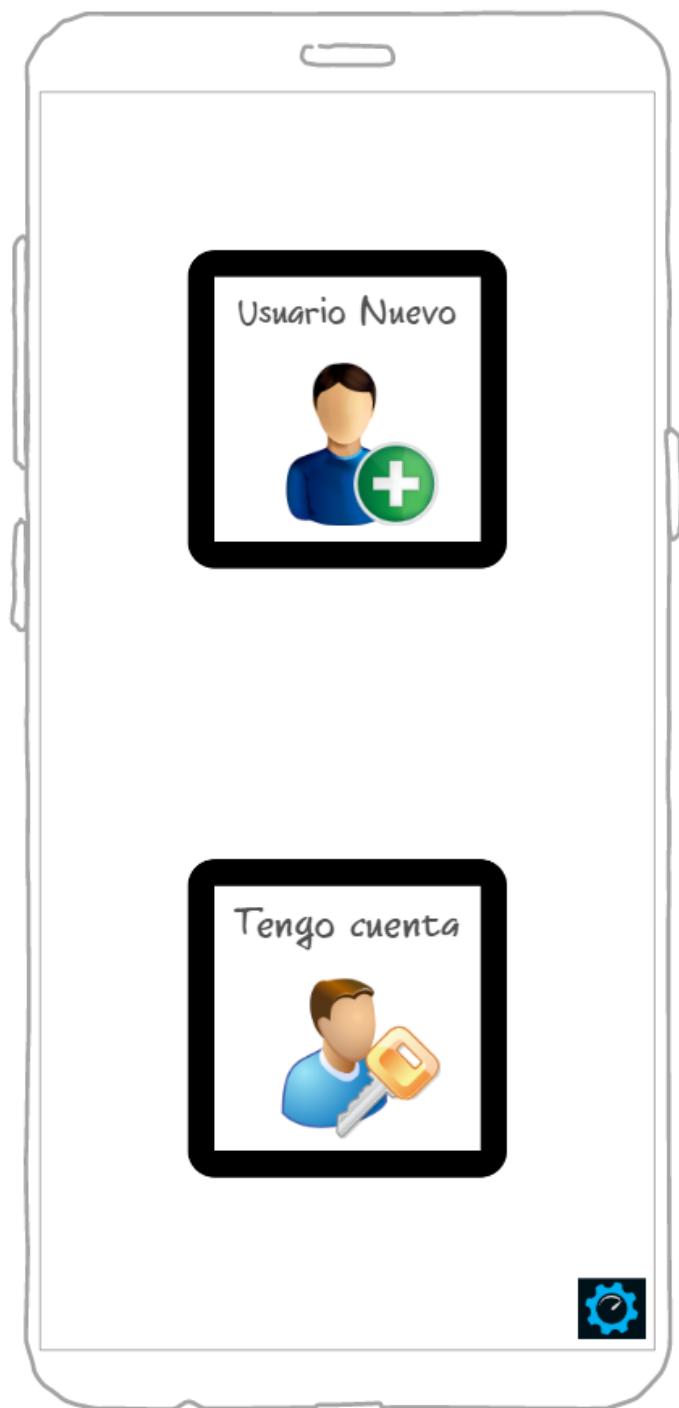
El diseño será adornado y estructurado de forma intuitiva en el proyecto (habrán cambios, este no es el diseño definitivo).

- Iniciamos en la **Pantalla Inicial** y pulsamos el botón para avanzar.
- Disponemos de 2 opciones: Pasar a hacer **login** si ya tenemos cuenta o **registrarnos** si somos un **nuevo usuario**.

- Pantalla Inicial -



- Elección-



◆ En caso de **tener cuenta**, introducimos nuestro **email** y **contraseña** para entrar.

◆ En caso de haber **olvidado la contraseña**, podemos utilizar el **email** (así como su confirmación) para restaurarla.

- Pantalla Login -



- Pop Reset Contraseña -



◆ Si no tenemos cuenta, podemos **registrarnos**:

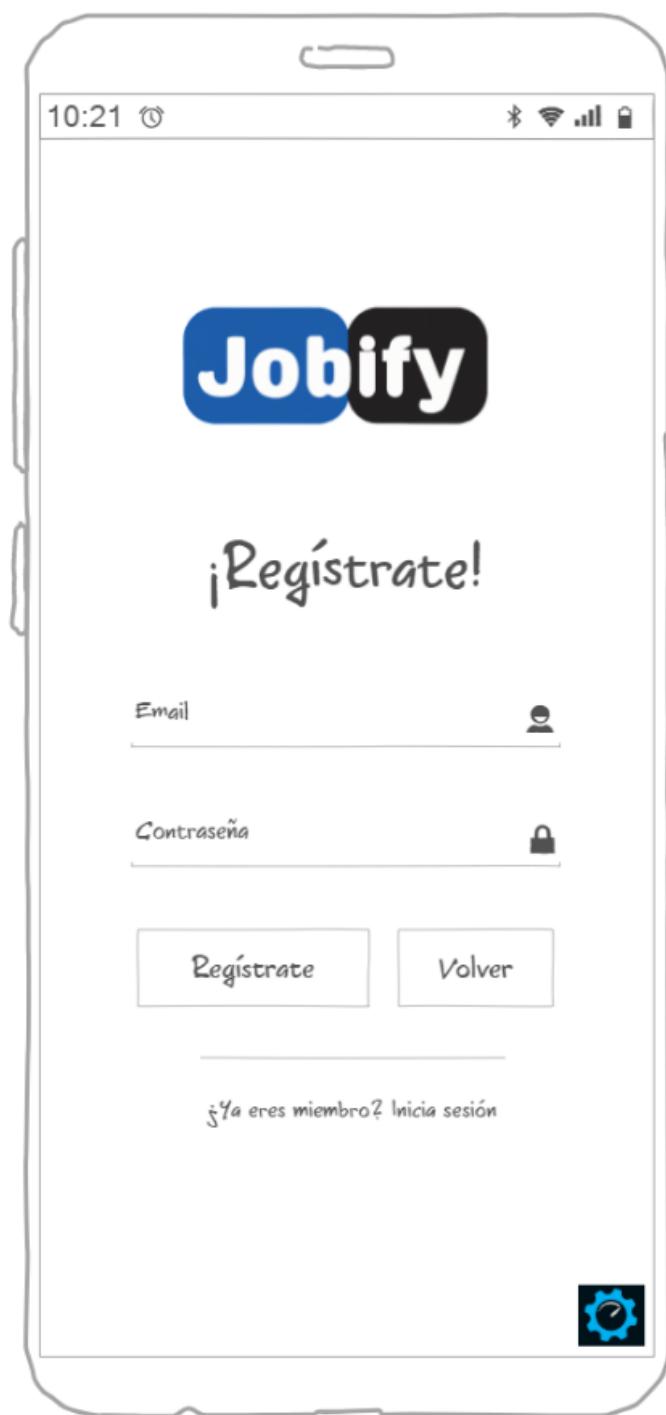
- ◆ Siendo usuario tipo **candidato**.
- ◆ Siendo usuario tipo **empresa**.

- Tipo Usuario Nuevo -



◆ Para ambas opciones, **introducimos los datos requeridos** y pulsamos en **registrarnos**, lo que nos lleva automáticamente a la pantalla de **login**.

- Registrar Candidato y Empresa -



 Una vez dentro, nos dirige a la **Pantalla de Ofertas Destacadas**.

◆ Podemos realizar las siguientes acciones:

- ◆ Ver los **detalles** de una oferta de trabajo.
- ◆ Hacer una **búsqueda** de una oferta de trabajo.
- ◆ Enviar **CV**.

- Pantalla Destacados -



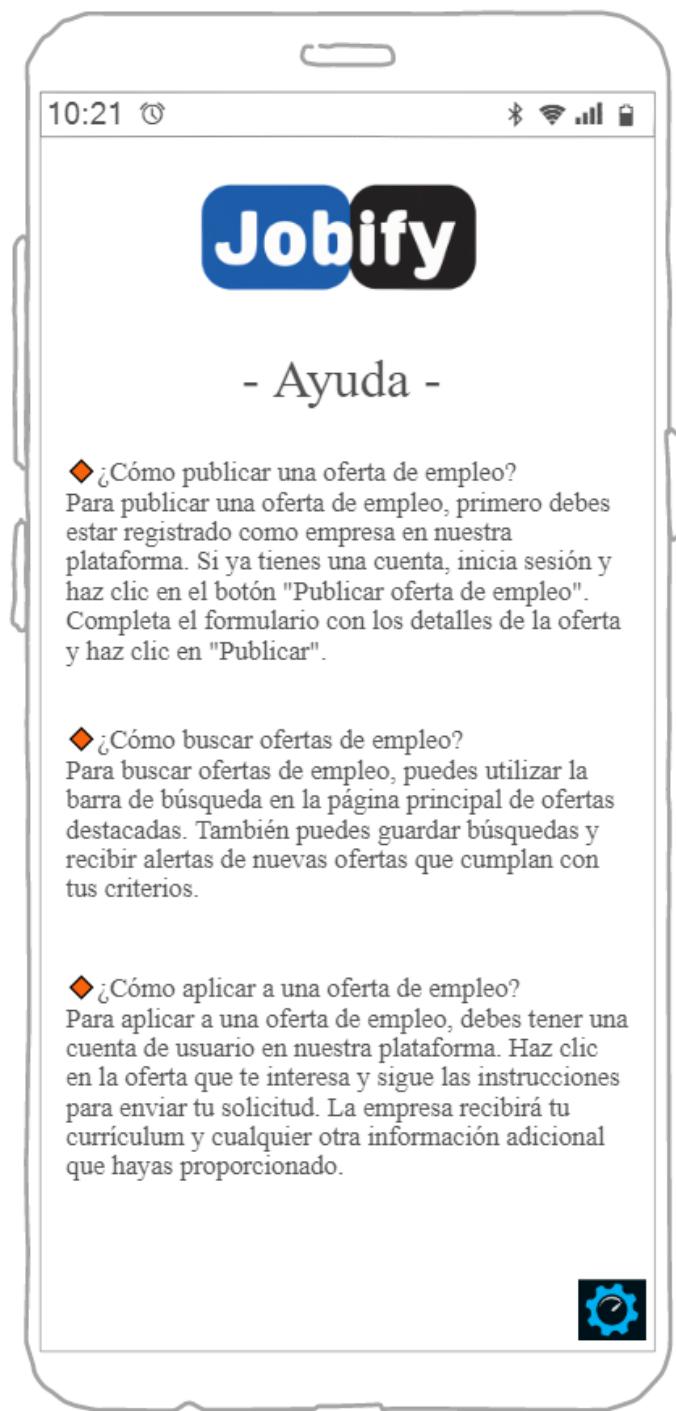
- Detalles Oferta -



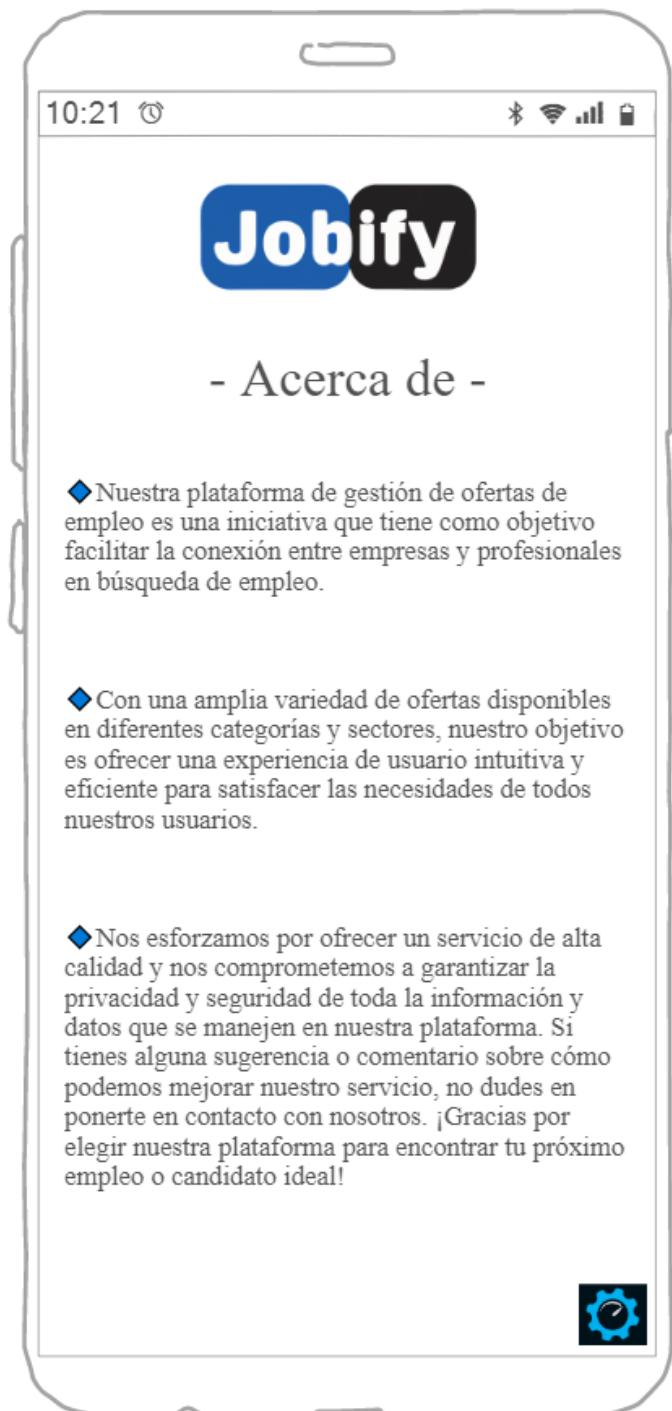
◆ Esta **pantalla**, además consta de un **menú** con la posibilidad de:

- ◆ Volver al **inicio** (**pantalla de ofertas destacadas**).
- ◆ Ver el **perfil** del usuario.
- ◆ **Ayuda**.
- ◆ **Acerca de**.
- ◆ **Cerrar sesión** (cierra la sesión existente y nos redirige a la pantalla de **inicio de sesión**)

- Ayuda -



- Acerca de -



◆ **Pantalla Perfil:**

◆ Para **cambiar nuestros datos** los editamos y pulsamos el botón de **guardar**.

◆ Para **borrar nuestra cuenta** pulsamos el botón de **borrar cuenta** (debemos confirmar la acción).

- Perfil -



- Pop Borrar Perfil -



◆ Para **cambiar la imagen**, pulsamos en ella y la seleccionamos desde la **cámara o la galería**.

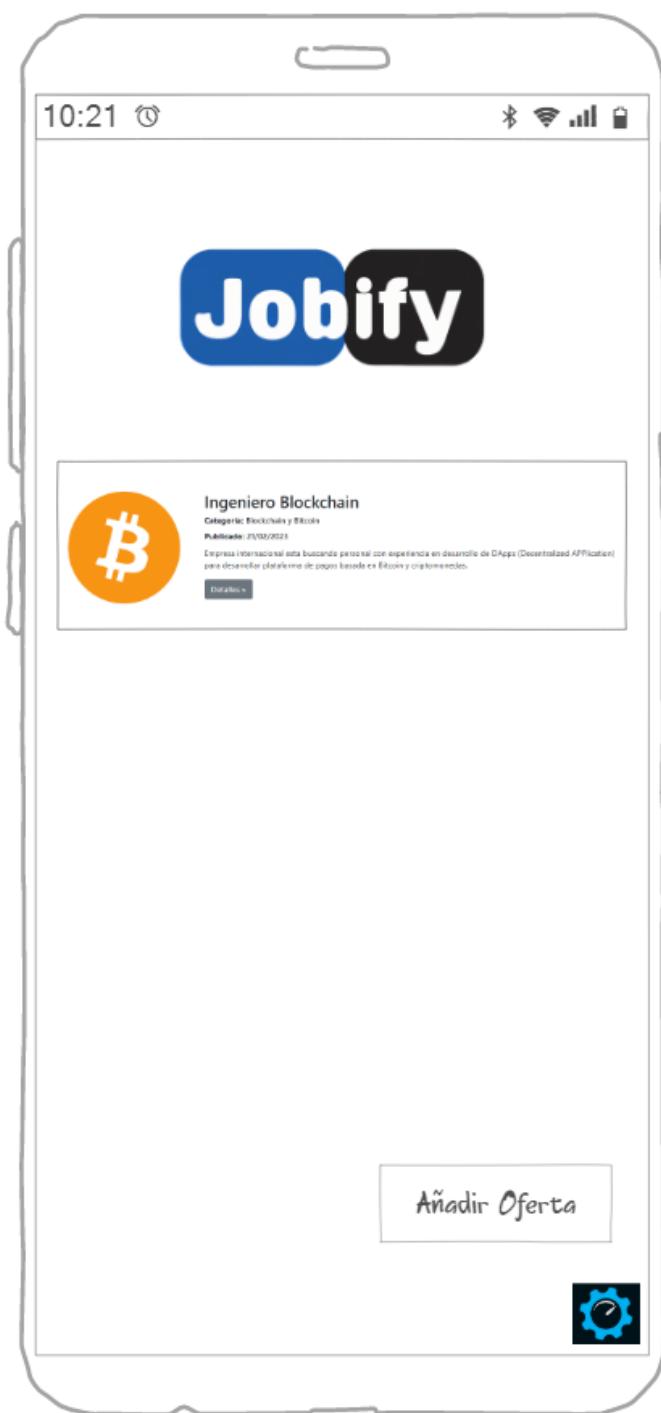
- Cambiar Imagen -



◆ Pantalla de Ofertas de Trabajo:

◆ Usamos el **botón inferior** para **añadir** una publicación nueva, rellenando los **campos requeridos** y dándole al botón de **guardar**.

- Pantalla ofertas -



- Añadir oferta -



- ◆ Eliminar la oferta de trabajo (debiendo hacer una segunda confirmación para borrarla).
- ◆ Actualizar los datos, que se abren en una ventana emergente y desde ahí podemos editarlos, cambiar la imagen y guardar los cambios.

- Actualizar oferta -



- Pop Borrar Oferta -

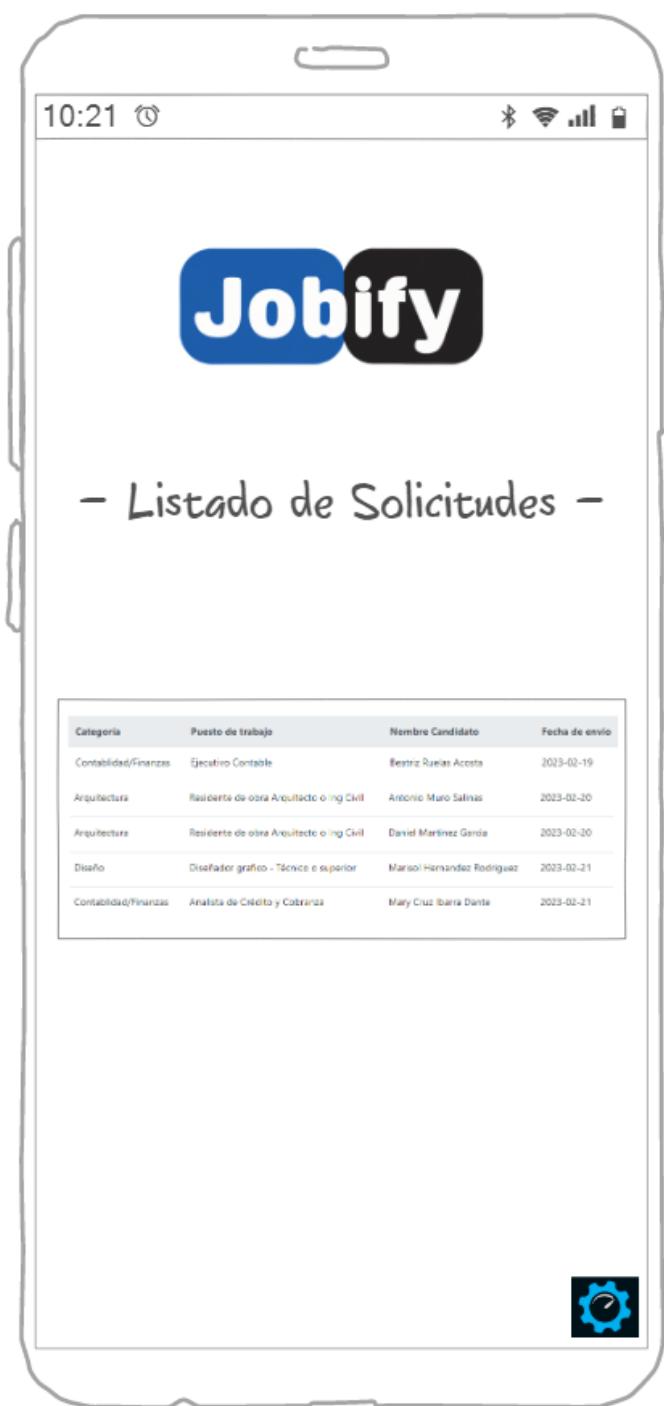


◆ Pantalla Solicituds:

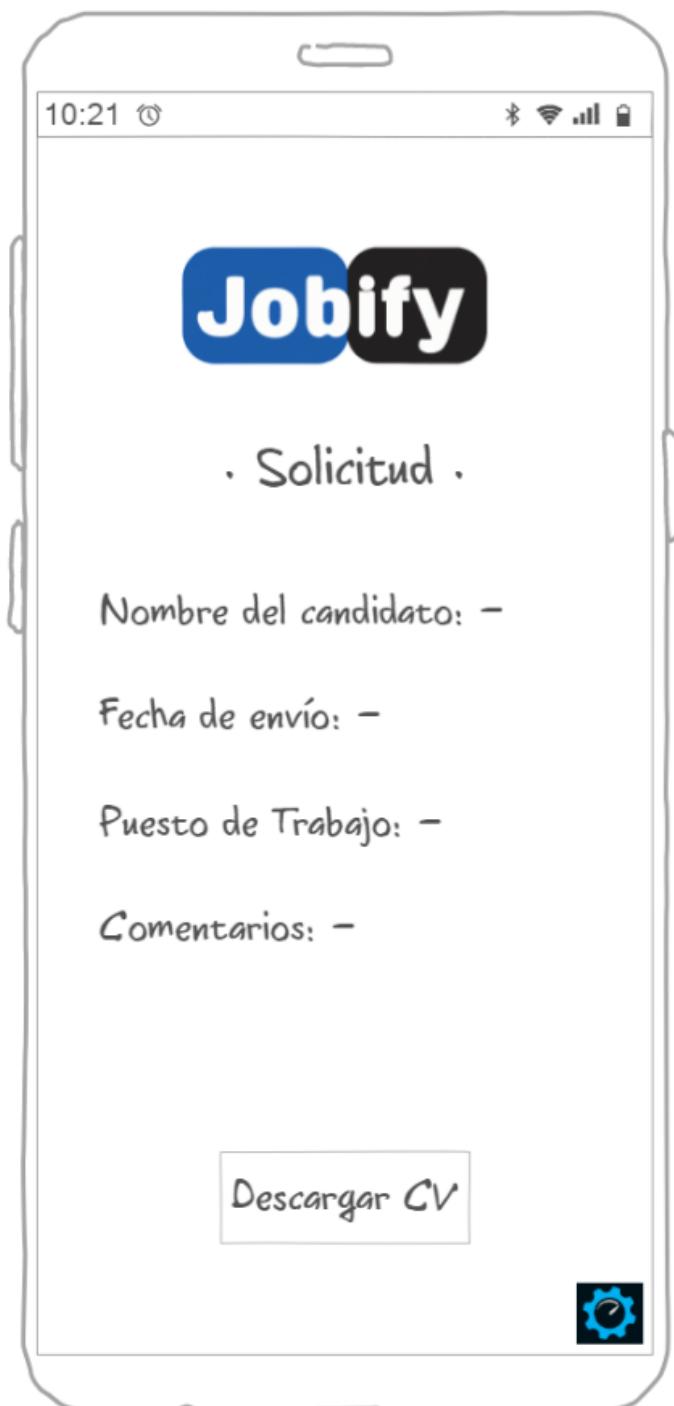
◆ Se muestra un **listado de solicitudes** de ofertas de trabajo enviadas por **usuarios candidatos**. Para cada una podemos ver el **nombre del candidato, la fecha de envío, los comentarios y el puesto de trabajo**

◆ Podemos **descargar el CV** del candidato.

- Listado Solicituds -



- Descargar CV -



◆ Pantalla Usuarios:

◆ Los administradores pueden hacer **cualquier cosa** en la aplicación, tanto **revisar y borrar ofertas, gestionar usuarios y permisos** de cada uno, **ver las solicitudes** (así como opciones adicionales en el menú)...

- Listado Usuarios -



- Gestión de Usuario -



◆ **Pantalla Usuarios:**

- ◆ La aplicación consta de **2 posibles usuarios (Candidato y Empresa)**, a parte del usuario **administrador**, entre estos nos podremos desplazar, cada uno con sus respectivos **permisos y roles**.
 - ◆ Los **usuarios candidatos** podrán **ver las ofertas** de trabajo, y si están interesados **mandar su CV**.
 - ◆ Los **usuarios empresa** podrán **añadir una nueva oferta de empleo, actualizarla o eliminarla**. También podrán **ver las solicitudes que tienen de candidatos a su oferta de trabajo**.

◆ Esta es una **plantilla base del diseño del proyecto**, de la cual se irá rediseñando y adaptando para optimizar la aplicación. Posibles implementaciones pensadas para un futuro:

- ◆ Mapas.
- ◆ Gráficos.
- ◆ Diálogos.
- ◆ Notificaciones.
- ◆ Tablas.
- ◆ ...



◆ **Resumen del flujo de actividad en la aplicación:**

1. La **aplicación** se abre en la **Pantalla Inicial** que contiene un botón que lleva a la **Pantalla de Registro o Inicio de Sesión**.
2. Si el **usuario** selecciona "Ya tengo una cuenta", se lleva a la **Pantalla de Login**.
3. Si el **usuario olvida su contraseña**, puede hacer click en el botón de "Olvidé mi contraseña" que muestra un **Pop-up** para introducir el **correo electrónico**, y si es válido, la aplicación **enviará un correo para restablecer la contraseña**.
4. Si el **usuario** selecciona "Soy un nuevo usuario", se le pide que seleccione entre "**candidato**" o "**empresa**" y rellene los datos necesarios en cada caso.
5. Después de completar el registro, la **aplicación** lleva al **usuario** a la **Pantalla de Destacados**, donde aparecen las **ofertas de empleo destacadas**. Desde allí, el **usuario** puede **ver los detalles de una oferta de empleo específica y enviar su CV**.
6. También se puede realizar una **búsqueda de ofertas de empleo** desde la **Pantalla de Destacados**.
7. En el **menú** hay opciones para volver a la **Pantalla de Destacados**, **ver el perfil del usuario**, **acceder a la ayuda**, **acceder a acerca de** o **cerrar sesión**.
8. La **Pantalla de Perfil** permite **editar el perfil del usuario** o **borrar su cuenta**. Si el **usuario** desea **eliminar su cuenta**, se muestra un **Pop-up** para **confirmarlo**.
9. La **Pantalla de Cámara o Galería** permite **cambiar la imagen del usuario**.
10. Si el **usuario** es una **empresa** puede **añadir, editar o borrar una oferta de empleo**, y **acceder a una pantalla para ver todas las solicitudes enviadas por los candidatos a esa empresa**.
11. Si el **usuario** es **administrador**, puede **acceder a una pantalla con un listado de usuarios para gestionarlos**.



BASE DE DATOS

Antes que nada vamos a recordar el significado de cada término:

- **Modelo entidad-relación (ER):** Es un **modelo conceptual** que representa las **entidades, atributos y relaciones** entre los datos de una organización. El **modelo E/R** describe cómo los datos están relacionados y cómo se pueden recuperar de una base de datos.
- **Modelo relacional:** Es un **modelo de datos** que representa la información en tablas compuestas por **filas y columnas**. Cada **tabla** representa una **entidad** y las **filas** contienen los **registros o instancias** de la **entidad**. Las **columnas** representan los **atributos o características** de la **entidad**.
- **Entidad:** Es un **objeto que existe independientemente** y puede ser **identificado** por una **clave única**. En una base de datos, una **entidad** se representa por una **tabla** (Ej: Usuario).
- **Relación:** Es una **asociación entre dos o más entidades**. En una **base de datos relacional**, la **relación** se representa mediante una **clave foránea que conecta las tablas**.



Con respecto a nuestra **base de datos**, tendríamos **6 tablas** en total.

- La tabla "**Categoría**" tiene información sobre las diferentes **categorías** de **vacantes** disponibles.
- La tabla "**Vacante**" contiene información sobre las **vacantes** disponibles (ofertas de trabajo disponibles).
- La tabla "**Solicitud**" contiene información sobre las **solicitudes** de los **usuarios** para una **vacante** en particular.
- La tabla "**Usuario**" contiene información sobre los **usuarios** registrados (pueden ser **candidato** o **empresa**).
- La tabla "**Empresa**" contiene información sobre los **usuarios** que están registrados como "**empresa**".
- La tabla "**Candidato**" contiene información sobre los **usuarios** que están registrados como "**candidato**".

El **Script de Creación de la base de datos** es el conjunto de **instrucciones SQL** que se utilizan para **crear la base de datos**, **tablas**, **columnas**, etc... En él, vamos a definir la **estructura de cada tabla**, incluyendo el **nombre de la tabla**, el **nombre de las columnas** y los **tipos de datos**, así como las **claves primarias y foráneas**.

El **Script para Insertar Datos** es el conjunto de **instrucciones SQL** que se utilizan para **agregar datos a la base de datos en las diferentes tablas**.



• *Modelo Entidad – Relación* •

En nuestro proyecto, las entidades son las siguientes:

- **Categoría**
- **Solicitud**
- **Usuario**
- **Vacante**
- **Empresa**
- **Candidato**

Cada **entidad** tiene sus propios **atributos**, que describen **características específicas de cada entidad**.

Por ejemplo, la entidad "**Categoría**" tiene los atributos "**id**" y "**nombre**". La entidad "**Vacante**" tiene los atributos "**id**", "**nombre**", "**fecha**", "**salario**", "**destacado**", "**imagen**" y "**detalles**".

Cada entidad tiene una clave principal, que es un **atributo** que **identifica de manera única cada registro en la tabla**. En nuestro proyecto, las **claves primarias** son:

- **Categoría**: idCategoria
- **Solicitud**: idSolicitud
- **Vacante**: idVacante
- **Usuario**: idUsuario
- **Empresa**: idUsuario (hereda de la entidad **Usuario**)
- **Candidato**: idUsuario (hereda de la entidad **Usuario**)



Además, existen relaciones o cardinalidades entre las entidades, que se representan en el diagrama E-R mediante líneas que conectan las entidades.

En nuestro proyecto, las relaciones son las siguientes:

- **Vacante y Solicitud:** Una **vacante** puede tener **cero o varias solicitudes**, y **cada solicitud** está asociada a **una vacante**. Esta es una **relación uno a muchos**, que se representa en el diagrama de forma que la **clave primaria de la tabla Vacante** se representa como **clave foránea en la tabla Solicitud**.
- **Categoría y Vacante:** Una **categoría** puede tener **cero o varias vacantes**, y **cada vacante** está asociada a **una categoría**. Esta es una **relación de muchos a uno**, que se representa en el diagrama de forma que la **clave primaria de la tabla Categoria** se representa como **clave foránea en la tabla Vacante**.
- **Candidato y Solicitud:** Un **usuario candidato** puede enviar **cero o varias solicitudes**, y **cada solicitud** está asociada a **un usuario candidato**. Esta es una **relación muchos a uno**, que se representa en el diagrama de forma que la **clave primaria de la tabla Candidato** se representa como **clave foránea en la tabla Solicitud**.
- **Especialización.** A la hora de registrar a un **usuario nuevo**, éste puede ser de tipo **empresa** o de tipo **candidato**. Para ello, cada uno de estos **hereda todos los atributos que tiene la entidad Usuario**, más los **atributos que presenta cada uno**. Además, ya sea **candidato** o **empresa** heredan el **id del usuario**.



En el **diagrama E-R**, se representan las **cardinalidades** de cada **relación** mediante notación específica, que indica el número de registros que pueden estar relacionados en ambas entidades.

Cada vez que un usuario se registra, se crea una entrada en la tabla "**Usuario**" con su información básica, y luego se crea una entrada correspondiente en la tabla "**Candidato**" o "**Empresa**", dependiendo de si el usuario se registra como **candidato** o **empresa**.

Las **entidades hijas** heredan todos los **atributos** de la entidad **padre**, incluyendo el atributo "**idUsuario**", que es una **clave primaria** de la entidad **padre**.

Por lo tanto, en este caso, el "**idUsuario**" que tienen las **entidades hijas** es el mismo "**id**" que **heredan de la entidad padre** para identificar de **forma única** a cada **candidato** o **empresa** en la **base de datos**. Cada una de las **entidades hijas** tendría además sus propios **atributos específicos**, como "**experiencia**", "**apellidos**" y "**educación**" para la entidad "**Candidato**", y "**sector**" y "**web**" para la entidad "**Empresa**".

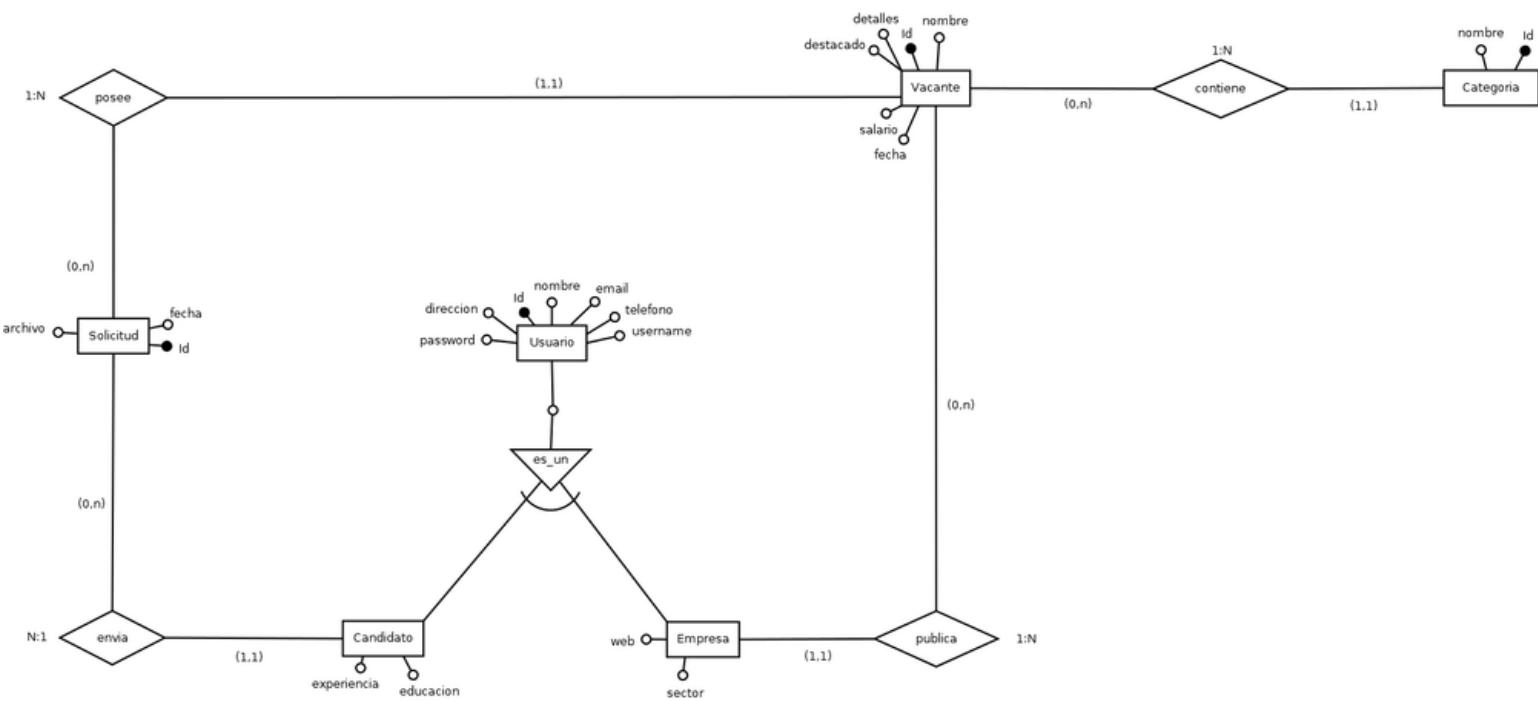
Para representar el **Diagrama Entidad-Relación** he utilizado los servicios que proporciona el software "**Día**".

Día es una **aplicación gratuita de código abierto** que se utiliza para **crear diagramas entidad/relación (ER)**.



La **aplicación** es fácil de usar y cuenta con una **interfaz gráfica de usuario intuitiva**. Con **Día**, se puede crear **diagramas entidad/relación** que ayudan a visualizar y diseñar la estructura de una base de datos. La **aplicación** ofrece **herramientas para crear entidades, atributos, relaciones y restricciones de integridad referencial**. Además, permite exportar los diagramas creados a diferentes formatos, como PNG, SVG, EPS, entre otros...

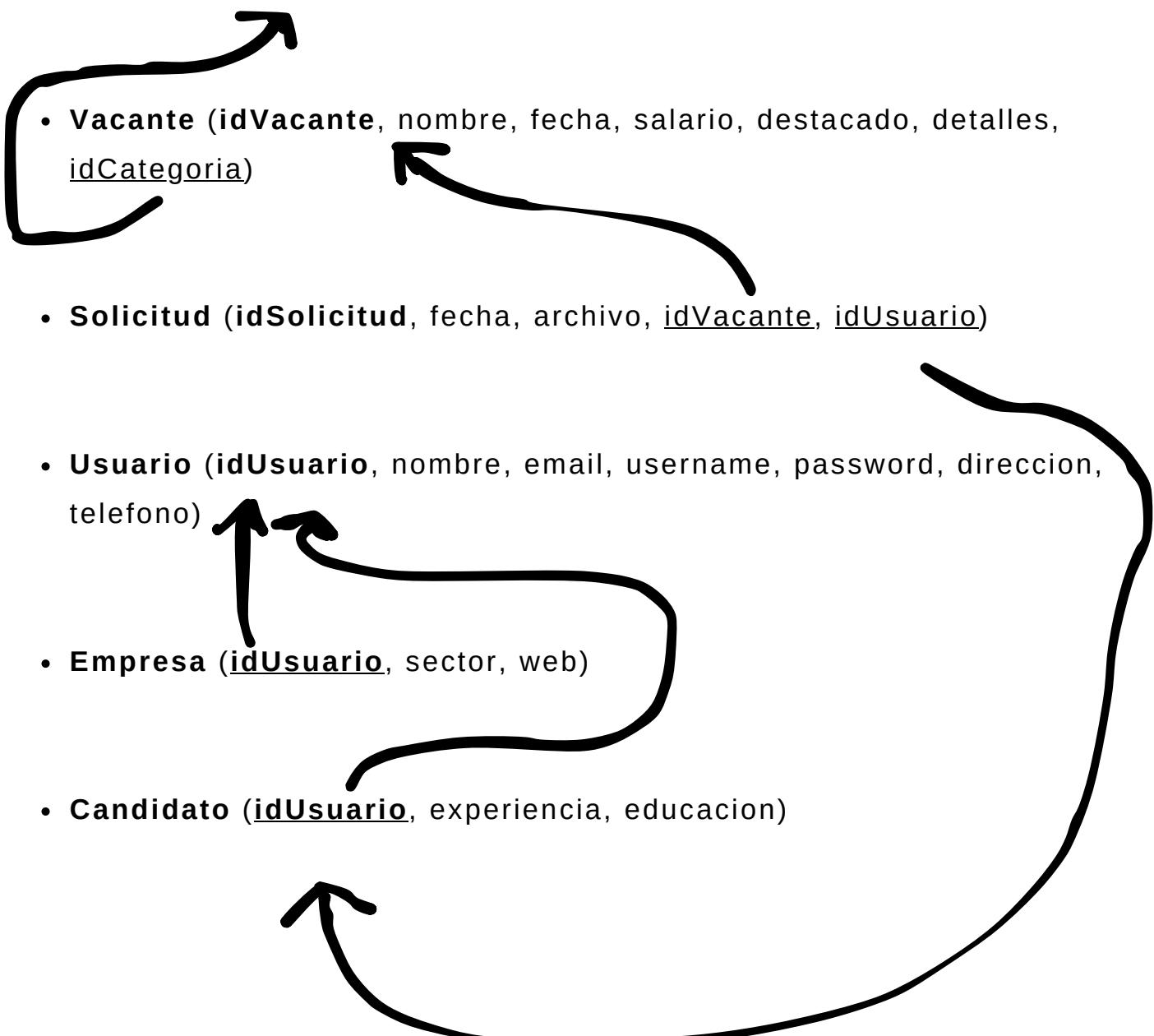
Así quedaría nuestro **diagrama Entidad/Relación**:



• *Modelo Relacional* •

Las tablas resultantes del **modelo entidad-relación** pasadas a **modelo relacional** son las siguientes:

- **Categoría** (**idCategoria**, nombre)



Donde:

- La tabla **Categoría** tiene como atributos: **idCategoria (clave primaria)** y **nombre**.
- La tabla **Vacante** tiene como atributos: **idVacante (clave primaria)**, **nombre**, **fecha**, **salario**, **destacado**, **detalles** e **idCategoria (clave foránea hacia Categoría)**.
- La tabla **Solicitud** tiene como atributos: **idSolicitud (clave primaria)**, **fecha**, **archivo**, **idVacante (clave foránea hacia Vacante)** e **idUsuario (clave foránea hacia Usuario)**.
- La tabla **Usuario** tiene como atributos: **idUsuario (clave primaria)**, **nombre**, **email**, **username**, **telefono**, **password** y **direccion**.
- La tabla **Candidato** hereda todos los atributos que tiene la entidad "**Usuario**", más sus atributos propios: **experiencia** y **educacion**.
- La tabla **Empresa** hereda todos los atributos que tiene la entidad "**Usuario**", más sus atributos propios: **web** y **sector**.



· Script de Creación de Tablas ·

Para crear una base de datos, se debe crear un script que contenga las **instrucciones SQL** necesarias para **crear las tablas y definir su estructura**. Lo he hecho en **MySQL**, pero en la aplicación lo implementaré con **SQLite**.

La primera **instrucción del script** debe ser la **creación de la base de datos** en sí misma, utilizando la siguiente sintaxis:

```
CREATE DATABASE empleosdb;  
USE empleosdb;
```

Una vez **creada la base de datos**, se deben **crear las tablas**.

Cada tabla se crea siguiendo la siguiente sintaxis:

```
DROP TABLE IF EXISTS `Categoria`;  
CREATE TABLE `Categoria` (  
    `id` int NOT NULL AUTO_INCREMENT,  
    `nombre` varchar(100) NOT NULL,  
    PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Se pueden agregar restricciones adicionales a cada columna, como **NOT NULL**, **UNIQUE**, **FOREIGN KEY**, entre otras...

La definición de la clave primaria es muy importante, ya que es la **columna** que **identificará de manera única** a cada **registro** en la **tabla**.



Es importante tener en cuenta que se pueden definir **múltiples columnas como clave primaria**, en cuyo caso se debe indicar una lista separada por comas de las columnas que conforman la clave primaria.

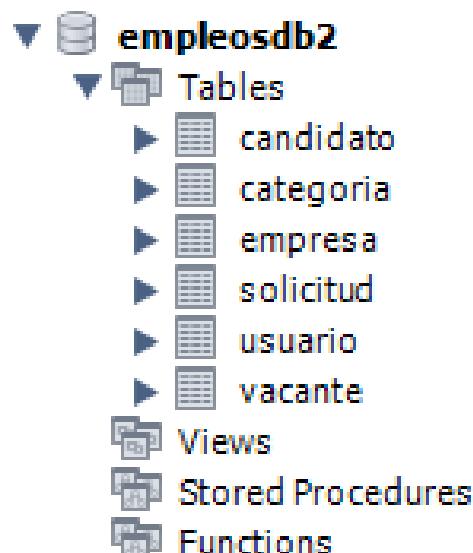
Una vez **creadas todas las tablas**, se pueden **crear las relaciones** entre ellas mediante la siguiente sintaxis:

```
CONSTRAINT `fk_vacante_categoria1` FOREIGN KEY (`idCategoria`) REFERENCES `Categoria` (`id`)
```

Esta sintaxis permite definir una **clave foránea** que hace referencia a la **clave primaria** de otra tabla. La **clave foránea** se define en la **tabla** que contiene la **relación**, y hace **referencia a la tabla** con la que se **relaciona**. En este caso, se debe especificar la **columna** que contiene la **clave foránea**, y la **tabla y columna** a la que hace **referencia**.

Una vez **creado el script**, se puede ejecutar en el gestor de base de datos para crear la base de datos y todas sus tablas.

Nos quedaría algo así:



Vamos a ver cómo han quedado las **tablas**.

La **tabla "Solicitud"** consta de la siguiente sintaxis:

```
DROP TABLE IF EXISTS `Solicitud`;
CREATE TABLE `Solicitud` (
  `id` int NOT NULL,
  `fecha` date NOT NULL,
  `archivo` BLOB NOT NULL,
  `idVacante` int NOT NULL,
  `idUsuario` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_Solicitud_Vacante1_idx` (`idVacante`),
  KEY `fk_Solicitud_Usuario1_idx` (`idUsuario`),
  CONSTRAINT `fk_Solicitud_Usuario1` FOREIGN KEY (`idUsuario`) REFERENCES `Candidato` (`usuario_id`),
  CONSTRAINT `fk_Solicitud_Vacante1` FOREIGN KEY (`idVacante`) REFERENCES `Vacante` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

La **tabla "Usuario"** consta de la siguiente sintaxis:

```
DROP TABLE IF EXISTS `Usuario`;
CREATE TABLE `Usuario` (
  `id` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) DEFAULT NULL,
  `email` varchar(100) NOT NULL,
  `username` varchar(45) DEFAULT NULL,
  `password` varchar(100) NOT NULL,
  `telefono` varchar(9) DEFAULT NULL,
  `direccion` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email_UNIQUE` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



• *Script de Inserción de Datos* •

Luego de tener la **estructura de la base de datos**, los **usuarios** y los **perfiles necesarios**, se debe proceder a crear el archivo ejecutable que contendrá el **script que insertará los datos en la base de datos**.

Este **archivo** puede ser un **archivo de texto** con extensión **.sql** que incluya las **instrucciones** necesarias para **insertar los datos en las tablas previamente creadas**.

Para realizar las **inserciones**, se deben tener en cuenta las **claves foráneas** que **relacionan las diferentes tablas**, asegurándose de **insertar primero los registros en las tablas padre para luego continuar con las hijas**.

Es importante **verificar que los datos ingresados cumplan con las restricciones definidas en la base de datos**, como la **integridad referencial** y las **restricciones de clave única**, entre otras.

Para la inserción en la tabla "**Categoría**" de los siguientes datos, la **sentencia SQL** sigue la siguiente sintaxis:

```
/* Inserto datos en la tabla "Categoria" */  
INSERT INTO `Categoria` (`id`, `nombre`) VALUES (1, 'Arquitectura');  
INSERT INTO `Categoria` (`id`, `nombre`) VALUES (2, 'Recursos Humanos');  
INSERT INTO `Categoria` (`id`, `nombre`) VALUES (3, 'Desarrollo de software');
```



Para la inserción en la tabla "Empresa" y "Candidato" de los siguientes datos, la **sentencia SQL** sigue la siguiente sintaxis:

```
/* Inserto datos en la tabla "Empresa" */  
INSERT INTO `Empresa` (`usuario_id`, `sector`, `web`) VALUES  
(2, 'Desarrollo', 'www.digital55.com');  
  
/* Inserto datos en la tabla "Candidato" */  
INSERT INTO `Candidato` (`usuario_id`, `experiencia`, `educacion`) VALUES  
(1, 'Programador Java', 'DAM - DAW');
```

Los **datos insertados** son los siguientes (mediante la siguiente consulta SQL):

```
SELECT * FROM empleosdb2.empresa;
```

	usuario_id	sector	web
▶	2	Desarrollo	www.digital55.com
●	NULL	NULL	NULL

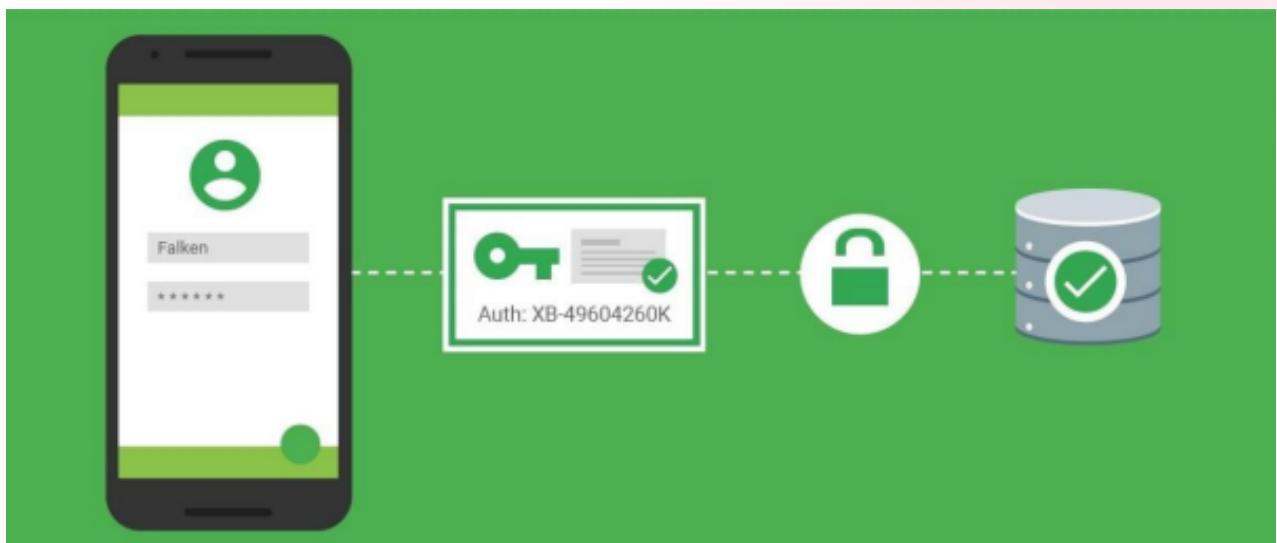
```
SELECT * FROM empleosdb2.candidato;
```

	usuario_id	experiencia	educacion
▶	1	Programador Java	DAM - DAW
●	NULL	NULL	NULL



SEGURIDAD

En cualquier proyecto de la actualidad se debe prestar mucha atención en la **seguridad**, para ello se tiene que **disponer de acceso como mínimo con usuario y contraseña**. En este caso, se **habilitará autenticación mediante correo electrónico y contraseña**.



Se utilizará el sistema de autenticación que nos ofrece **Google** llamado **Firebase Authentication**. En muchas ocasiones nos planteamos cómo acceder a un **servicio web** para tener nuestra aplicación trabajando con datos en la **nube**. Por ello surgió **Firebase**, para proveer una **API** donde **guardar y sincronizar los datos en la nube en tiempo real**. Uno de los aspectos que más hay que destacar es la asombrosa documentación que se puede consultar cuando accedemos a la plataforma.

Una vez que el **usuario inicie sesión por primera vez**, se **creará una cuenta de usuario nueva** y se la **vinculará con las credenciales**, es decir, el **email y la contraseña** del usuario con los que el **usuario inició sesión**.

Esta cuenta nueva se almacenará como parte del proyecto y se puede usar para identificar a un usuario en la app.



TECNOLOGÍA

Este proyecto ha sido desarrollado apoyándose en una herramienta especializada para el **desarrollo de aplicaciones Android**, es decir, con la ayuda de un **IDE**. Podría haber sido programado directamente mediante un editor de textos, pero este tipo de **herramientas facilitan la programación y el desarrollo de aplicaciones Android** facilitando en gran medida la **tarea del desarrollador**, ya que integran entre otros aspectos un sistema de ayuda para la construcción de interfaces gráficas de usuario lo cual agiliza mucho el trabajo. Estoy familiarizado con esta tecnología, utilizada durante el curso de **DAM**.

Android Studio

IDE oficial para la plataforma Android. Está basado en IntelliJ y pertenece a la empresa JetBrains. Admite el lenguaje Java. Proporciona emuladores Android.

Principales características de Android Studio:

- Renderización en tiempo real.
- Consola de Desarrollador: Consejos de optimización, ayuda para la traducción, estadísticas de uso...
- Soporte para construcción basada en **Gradle**.
- Refactorización específica de **Android** y arreglos rápidos.
- Plantillas para crear diseños comunes de Android y otros componentes.



Java

Se ha decidido utilizar este **lenguaje de programación** por varias razones. En primer lugar porque se había elegido previamente el **Sistema Operativo Android** y es un lenguaje nativo de éste.

Pero además de esta razón fundamental, me siento cómodo programando en **Java** gracias a la relativa experiencia alcanzada durante estos años cursando **DAM**.

Por otro lado, ha sido seleccionado este **lenguaje**, por las miles de librerías que nos ofrece, así como por la **numerosa documentación** que puede encontrarse.

Firebase Authentication

La **autenticación de usuarios** ha sido llevada a cabo mediante este servicio ofrecido por **Firebase** que utiliza sólo código del cliente.

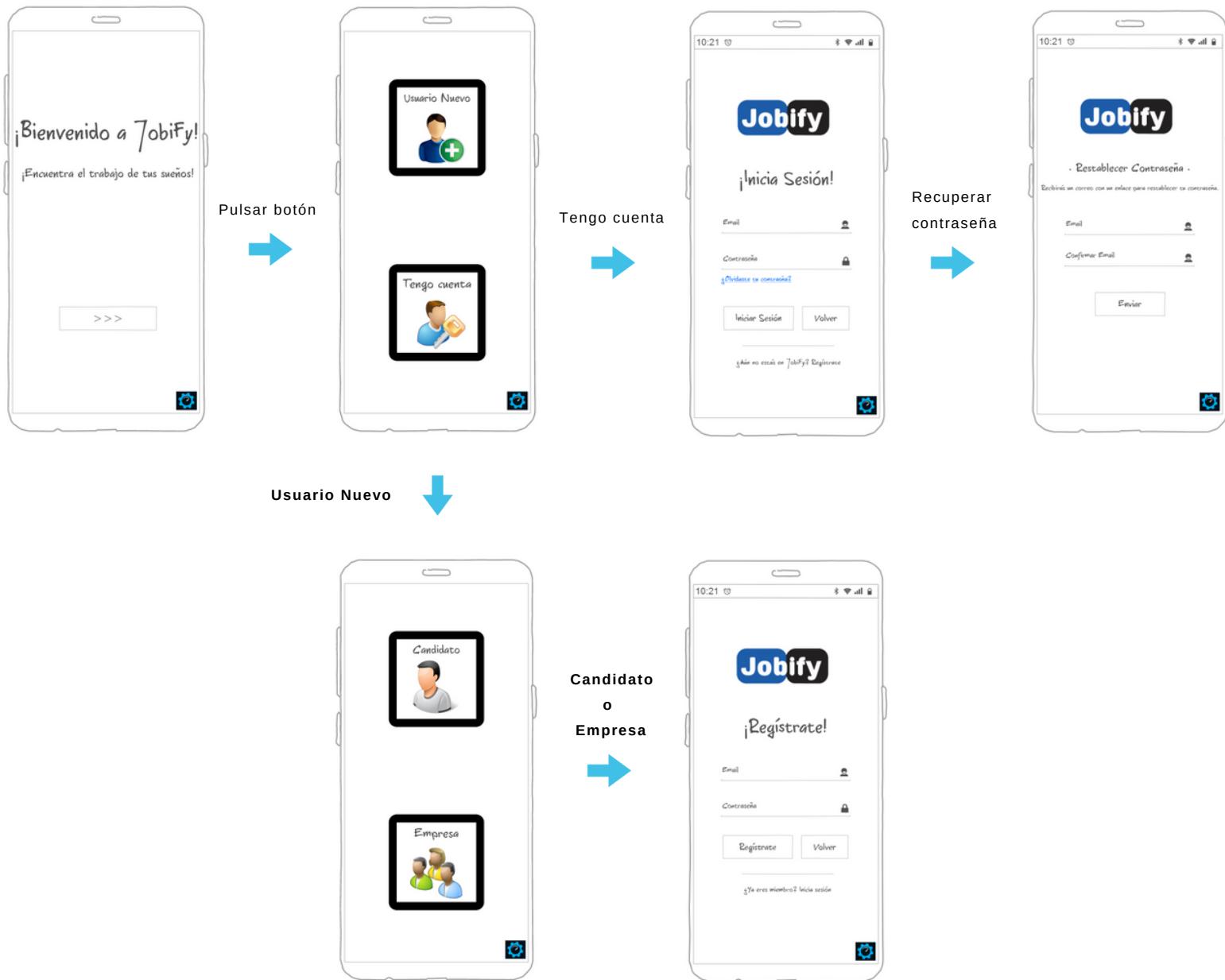


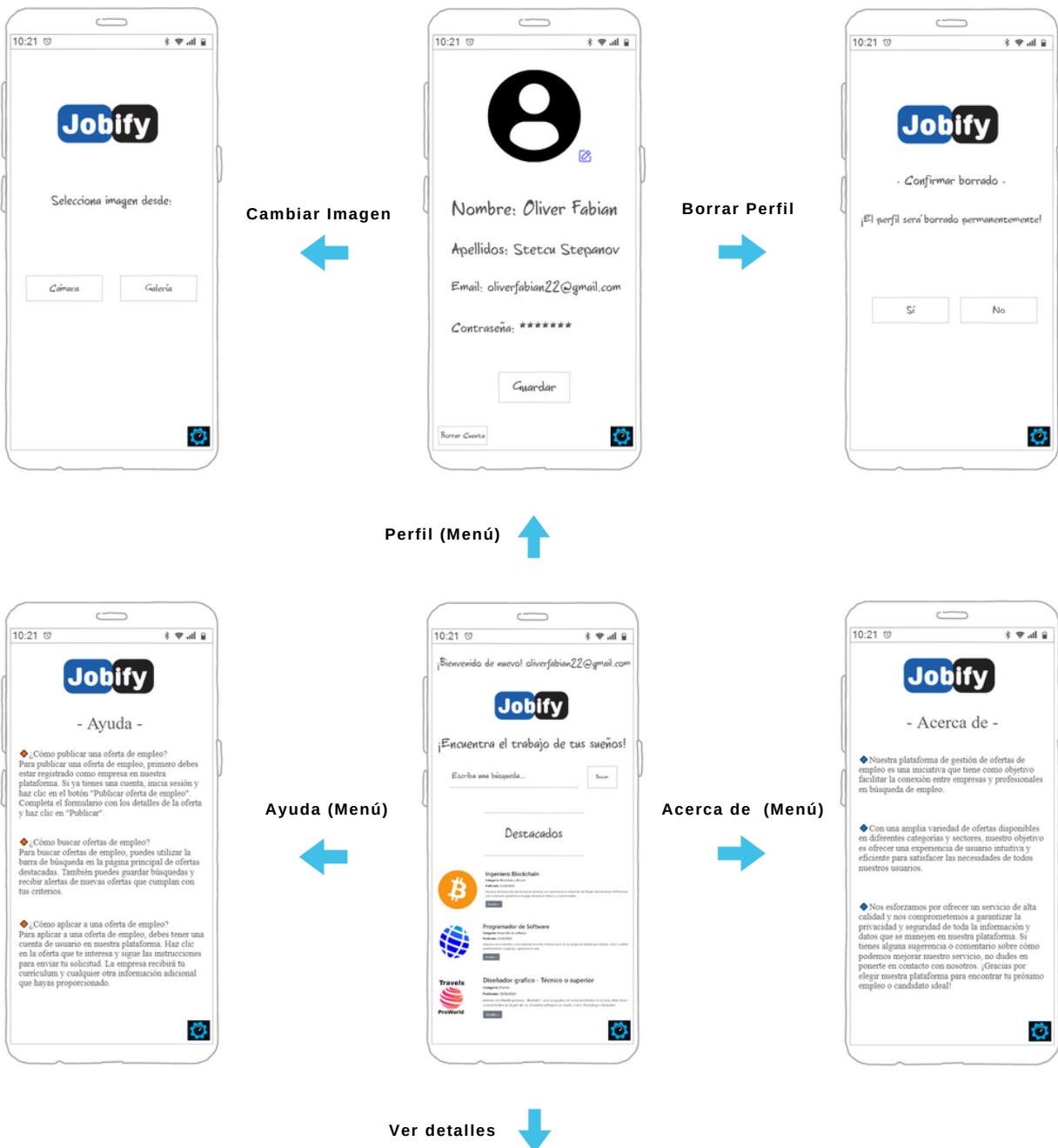
Firebase



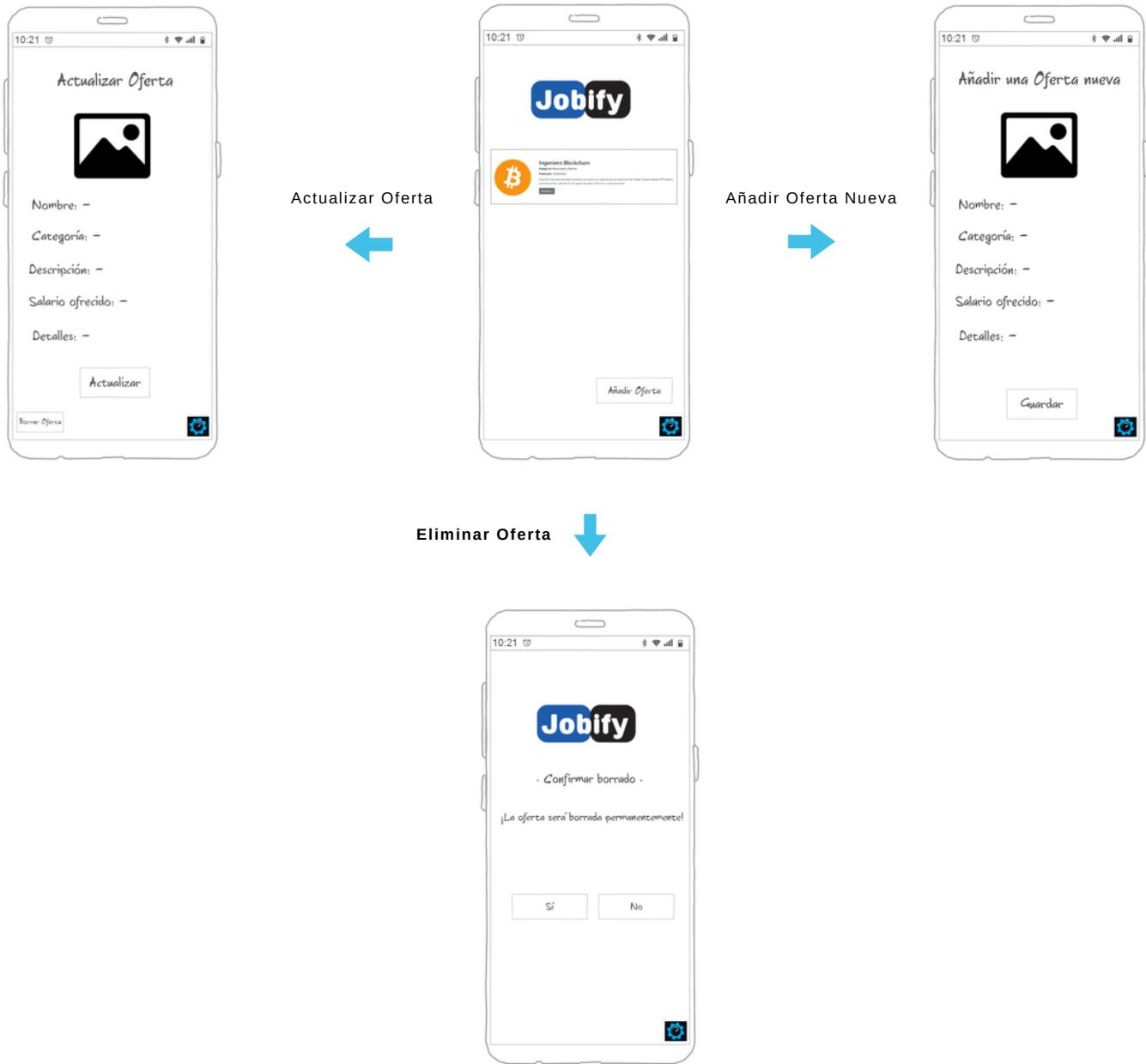
INTERFAZ COMPLETA

- La **Interfaz completa** con la **integración** de todas las **pantallas** y el **flujo** del mismo es la siguiente:





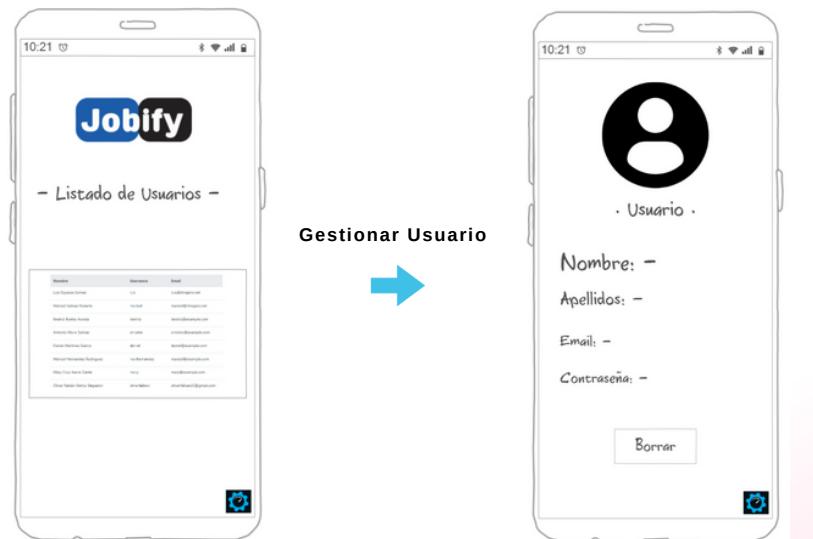
Ofertas de Trabajo (Menú)



Listado de Solicitudes (Menú)



Listado de Usuarios (Menú)

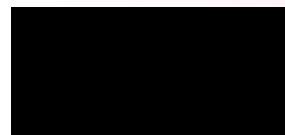


STYLESHEET

- La **paleta de colores** utilizada en esta aplicación es la siguiente:



• Primary



• Primary Text



• Primary Light



• Secondary Text



• Primary Dark



• Secondary



• Secondary Light



• Secondary Dark



GESTIÓN

Un **sistema de gestión** es una **herramienta que permite gestionar y controlar el proyecto** en particular. Este **sistema** puede estar compuesto por **varios módulos interconectados que trabajan juntos para lograr objetivos específicos**. Entre las **funcionalidades principales** que podrían ser incluidas en un **sistema de gestión**, se encuentran la **creación, modificación, eliminación, listado y búsqueda de datos**.

Algunas de estas **funcionalidades integradas en nuestro proyecto** podrían ser:

- Creación de una oferta de trabajo, solicitud, categoría, usuario...
- Modificación de una oferta de trabajo, solicitud, categoría, usuario...
- Eliminación de una oferta de trabajo, solicitud, categoría, usuario...
- Listar ofertas de trabajo, solicitud, categoría, usuario...
- Búsqueda de ofertas de trabajo, solicitud, categoría, usuario...

Además, podríamos tener una **funcionalidad para la gestión de usuarios y perfiles**. Por ejemplo, podríamos tener **diferentes perfiles de usuario (como administrador, empresa o candidato)** y cada perfil tendría **diferentes privilegios** (como la **capacidad de crear o eliminar ofertas de trabajo**).

También podríamos implementar una **función de autenticación y registro de usuario para garantizar que solo los usuarios autorizados puedan acceder al sistema**.



BIBLIOGRAFÍA Y ANEXOS

- <https://www.canva.com/>
- <https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/#:~:text=Generalmente%20hay%20dos%20tipos%20de,c%C3%B3mo%20debe%20comportarse%20el%20sistema.>
- <https://www.visual-paradigm.com/>
- <https://nijamock.com/>
- <https://www.freeiconspng.com/>
- <https://developer.android.com/training/index.html?hl=es>
- <https://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/>
- <https://color.adobe.com/es/create/color-wheel>
- <https://mybrandnewlogo.com/es/generador-de-paleta-de-colores>
- <http://stackoverflow.com/>
- <https://firebase.google.com/?hl=es-419>
- <https://www.youtube.com/watch?v=dpURgJ4HkMk&t=642s>
- <https://www.mysql.com/>
- <https://sqlite.org/index.html>
- <https://dbeaver.io/>
- https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n
- <https://www.java.com/es/>



OLIVER FABIAN STETCU

