ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE PWA DE SERVICIO DE BOLSA DE EMPLEO Y PASANTÍAS: CASO DE ESTUDIO, FIS - EPN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

WILMER VINICIO QUINGA CUICHAN

wilmer.quinga@epn.edu.ec

DIRECTOR: MSC. REGINA MARITZOL TENEMAZA VERA

maritzol.tenemaza@epn.edu.ec

Quito, enero 2022

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Wilmer Vinicio Quinga Cuichan, bajo mi supervisión.

REGINA MARITZOL TENEMAZA VERA

Firmado digitalmente por REGINA MARITZOL TENEMAZA VERA

Nombre de reconocimiento (DN): cn=REGINA MARITZOL TENEMAZA VERA, serialNumber=201221114438,

MSc. Maritzol Tenemaza

DIRECTOR DE PROYECTO

DECLARACIÓN

Yo, Wilmer Vinicio Quinga Cuichan, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional, y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Escuela Politécnica Nacional según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Wilmer Vinicio Quinga Cuichan

DEDICATORIA

Este trabajo de titulación está dedicada a mis padres, ya que, gracias a su esfuerzo y apoyo incondicional, me permitieron terminar esta meta.

Y para Leandro que es mi motor que me impulsa al éxito.

Vinicio Quinga

AGRADECIMIENTO

A mis padres por su amor incondicional, por enseñarme que el trabajo honesto siempre tiene su recompensa sin ustedes no lo hubiese logrado.

A mis hermanos Ivonne y Byron por siempre ayudarme en los momentos difíciles.

A mis padrinos que son como mis segundos padres, siempre me supieron aconsejar y ayudar para terminar esta meta.

A mi abuelo Belisario que siempre estuvo pendiente de mi bienestar.

A mi abuela Rosita que nos tuvo que dejar gracias por todo su cariño y ayuda.

A mi tutora MSc. Maritzol Tenemaza por su guía y colaboración en este proyecto.

Por último, agradezco a todos mis amigos, por el apoyo brindado en los momentos más difíciles.

Vinicio Quinga

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo es una solución al problema que surge en la facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional al no contar con un sistema de bolsa empleo por lo cual las empresas no pueden registrar sus requerimientos y los estudiantes no tienen un lugar para elegir opciones de pasantías o trabajos para fortalecer sus conocimientos. La solución propuesta es la construcción de un sistema de bolsa de empleo, ayudados por el marco de referencia SCRUM el cual permite realizar entregas parciales del producto para comprobar que se cumpla con los objetivos planteados. Se utiliza una arquitectura cliente – servidor de 2 capas para la construcción del servidor se usó el framework Sails el cual se conecta a la base de datos Mysql y su objetivo será administrar los datos para el cliente, se emplea el framework Ionic que emplea el lenguaje de programación Angular su objetivo será administrar la interfaz de usuario y la presentación de los datos. Como resultado se obtiene una aplicación web progresiva (PWA) el cual se adapta a cualquier tamaño de pantalla de dispositivo móvil y permite a las empresas crear las ofertas de empleo según sus necesidades y a los estudiantes egresados y graduados elegir opciones de pasantías o trabajos.

Palabras clave: Bolsa de empleo**,** pasantías, PWA, arquitectura cliente – servidor.

ABSTRACT

The following document is the solution to the problem about Job Exchange in the Faculty of Systems Engineering of Escuela Politécnica Nacional. The faculty does not have a Job Exchange system, due to enterprises cannot register their requirements and students do not have a place to choose internship options or jobs to strengthen their knowledge. The proposed solution is the construction of a job exchange system, supported by SCRUM reference framework, which allows partial deliveries of the product to verify that it meets the objectives set. A client architecture -2-layer server is used to build the server using Sails framework which connects to Mysql database and will aim to manage data for the client, using Ionic framework that works with Angular programming language its goal will be to manage user interface and data presentation. As a result, it gets a progressive web application (PWA) which adapts to any screen size of mobile devices and allows companies to create job offers according to their necessities for graduate students and graduates to choose options for internships or jobs.

Keywords: Job exchange, internships, PWA, client architecture - server.

1 INTRODUCCIÓN

1.1 Planteamiento del problema

Con el rápido progreso de la tecnología y con un mercado objetivo, la necesidad de aplicaciones web y/o móviles en cualquier negocio es inevitable [1]. La dependencia de los teléfonos inteligentes ha generado un crecimiento avanzado en el desarrollo de aplicaciones siendo cada vez más rápidas y con contenido personalizado [2].

Las empresas definen sus procesos de negocio con el apoyo de herramientas tecnológicas [3]. Los departamentos de recursos humanos se apoyan por aplicaciones como bolsas de empleo para el reclutamiento de nuevos funcionarios. Los trabajadores y las empresas utilizan las bolsas de empleo tanto para optar por una vacante de trabajo como para reclutar personal, respectivamente [3].

Los sistemas del manejo de bolsa de empleo se encargan de la gestión de ofertas laborales, donde los oferentes están en capacidad de escoger una opción de interés. Por otro lado, las empresas tendrán el beneficio de reclutar a los mejores candidatos verificando sus hojas de vida, reduciendo así sus gastos y el tiempo de ejecución [4] [5] .

La Escuela Politécnica Nacional como universidad pública referente en el Ecuador se ocupa de formar profesionales con altos estándares académicos y capaces de aportar al desarrollo del país. Por este motivo empresas públicas y privadas se interesan en contratar estudiantes en condición de pasantes o trabajadores.

La Escuela Politécnica Nacional si cuenta con un sistema de bolsa de empleo. Sin embargo, al momento la facultad de Ingeniería de Sistemas no dispone de un sistema de bolsa de empleo propio por esta razón surge la necesidad de crear la bolsa de empleo para facilitar que las empresas registren sus demandas laborales y los estudiantes elijan entre las opciones de pasantías o trabajos que aporten a enriquecer sus conocimientos.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar un sistema bolsa de empleo donde las empresas expongan sus requerimientos para contratar estudiantes egresados y/o graduados.

1.2.2 Objetivos Específicos

- Diseñar un sistema de bolsa de empleo
- Construir un sistema bolsa de empleo guiados por una metodología ágil.
- Integrar aspectos de usabilidad al diseño de interfaces de usuario donde las empresas registren sus requerimientos y estudiantes, egresados, graduados opten por plazas de pasantías y/o trabajos.
- Garantizar que el sistema sea una aplicación responsive para lo cual se aplicarán características de una PWA.

1.3 Marco Teórico

1.3.1 Aplicación web progresiva (PWA)

La aplicación web progresiva (PWA) es una tecnología que permite que un sitio web se comporte como si fuera una aplicación móvil. Está diseñada para aprovechar las funciones nativas que poseen los smartphones. Creada a partir de la tecnología web que contiene HTML, CSS y JavaScript, con una funcionalidad que compite con una aplicación móvil nativa [6].

A medida que el navegador se moderniza, incrementa progresivamente las funciones que están disponibles para el usuario, esta es la característica fundamental de las PWA [7].

Las ventajas que ofrecen son [8]:

- Progresiva. Funciona para todos los usuarios, sin importar la elección del navegador web porque está construida bajo la filosofía de mejora continua.
- Adaptable. Se pueden visualizar en cualquier pantalla ya sea de escritorio, móvil o Tablet.
- Modo aplicación. La visualización de la aplicación web en el dispositivo móvil será igual que una aplicación móvil nativa.
- Segura. Para ejecutar una PWA deben estar cargadas con el protocolo HTTPS para garantizar seguridad con los datos del usuario.
- Vinculable. Gracias a las características propias de los navegadores web móviles las PWA podrán instalarse y visualizarse en los dispositivos móviles como si fuera una aplicación móvil nativa.

Las PWA utilizan API web modernas como por ejemplo Service Workers y Web App Manifests. Estas 2 API tiene soporte completo en Chrome, Firefox, Safari y Microsoft Edge entre otros [9].

Service Workers es el actor más importante que permite tener aplicaciones fiables, rápidas y atractivas. Facilitando que las PWA funcionen sin conexión de internet cargando el contenido de la pantalla sin importar la conexión de red del usuario. Además, brindan un control detallado sobre el almacenamiento en cache a través de las API de JavaScript.

Web App Manifests permite que las PWA brinden la apariencia de un dispositivo móvil nativo, permitiendo especificar un icono, nombre de la aplicación, color de pantalla y su desempeño en pantallas de diferentes tamaños [9].

1.3.2 Metodología de desarrollo de software Scrum

Scrum es un marco de referencia para crear software complejo y entregarlo a tiempo, mediante la aplicación de un conjunto de directrices a seguir por el equipo de trabajo [10].

La misma que presenta las siguientes etapas [10] :

1. Establecer el product backlog o pila de producto el cual recopilará los requerimientos establecidos por el stakeholders. Al finalizar por cada requerimiento se establecerá el valor que aporta al cliente y el costo en esfuerzo estimado para completarlo. Además, se desarrollará un prototipo de interfaces de la aplicación y se modelará la base de datos para establecer el alcance del proyecto a desarrollar.

2. Planificación del Sprint. Es una reunión para establecer las tareas que se harán en el sprint y se genera el Sprint backlog.

3. Sprint. Es el periodo para codificar el software y probarlo siguiendo un conjunto de tareas seleccionadas en el backlog, por lo general se dispone de un periodo que puede ser de 15 días hasta 2 meses para este proceso, pero puede cambiar por las necesidades de cada equipo.

4. Daily: Reunión diaria donde se explicará las tareas realizadas y aquellas que se van a ejecutar, con el fin de evitar posibles errores en la ejecución del sprint.

5. Revisión del Sprint. Se entregará las funcionalidades realizadas, después de ser aprobadas se unirán al ambiente de desarrollo para tener un incremento estable y funcional.

6. Retrospectiva del Sprint. Se establecerá las debilidades y fortalezas en la ejecución del proyecto y se crea un plan de mejoras para el siguiente sprint.

El equipo Scrum está conformado por los siguientes roles:

1. Scrum Máster. Persona que lidera el equipo guiándoles para cumplir con el proceso de la metodología.

2. Product Owner. Su función es comunicarse con los dueños del producto obtener la visión del producto y comunicarle al equipo de desarrollo.

3. Development Teams. Grupo de profesionales que desarrollan el proyecto de manera conjunta.

1.3.3 APIS RESTFUL

Las APIs ofrecen una forma sencilla de conectar, integrar y ampliar un sistema, utilizan el protocolo HTTP en una API web. Utiliza las solicitudes HTTP para obtener, crear, actualizar y eliminar datos. La información que obtiene puede estar en diversos formatos XML, JSON, YAML entre otros [11].

Un servicio REST es un conjunto de restricciones que se debe tener en cuenta en la arquitectura del software [12].

1. Cliente – Servidor. El servidor se encarga de administrar los datos y el cliente maneja las interacciones con el usuario.
2. Sin estado. Cada petición enviada al servidor debe ser independiente y contener todos los parámetros solicitados
3. Cache. Administrar un almacenamiento de cache para evitar repetir peticiones.
4. Interfaz uniforme. Poseer una interfaz genérica para administrar las interacciones entre el servidor y el cliente.
5. Sistemas de capas. Disponer de varias capas para mejorar la escalabilidad, rendimiento y seguridad.

1.3.4 Arquitectura Cliente – Servidor

La arquitectura cliente servidor de 2 capas es la más simple en donde [13] [14]:

1. El cliente se encarga de administrar la interfaz de usuario, el control de entrada y la presentación de los datos.
2. EL servidor se encarga de administrar los datos. Creación, actualización, búsqueda y eliminación de datos.

En la comunicación entre el cliente y el servidor siempre es el cliente que solicita el servicio del servidor enviando una solicitud. Algunos beneficios:

- Flexibilidad en el software del cliente puede incluir cualquier sistema operativo siempre que disponga de un navegador web.
- Facilidad de mantenimiento de la aplicación cualquier actualización está disponible para todos los usuarios.
- Puede trabajar con cualquier dispositivo móvil porque el cliente solo necesita un navegador web para conectarse al servidor.

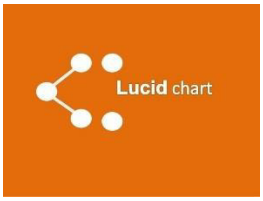
1.3.5 Herramientas, frameworks y librerías de desarrollo

Nombres	Descripción	Utilizado en
Nodejs	Es un entorno de ejecución de JavaScript orientado a eventos asincrónicos. Está diseñado para crear aplicaciones escalables [15].	Aplicación web.
NPM	Es un administrador de paquetes de software que utilizan los desarrolladores de código abierto para compartir y tomar prestados paquetes [16].	Aplicación web.

 IONIC	Es un marco de SDK frontend que permite crear aplicaciones híbridas utilizando la misma base de código, con la ayuda leguajes como HTML, CSS, JavaScript, Angular y TypeScript [17].	Aplicación web.
 ANGULAR	Es un marco basado en componentes para crear aplicaciones de una sola página eficientes y sofisticadas [18] .	Aplicación web.
 SAILS	Es un framework para nodejs incluye varias capas de abstracción para hacer un desarrollo fácil. Se puede conectar con diversas bases de datos como Postgres, Mongo, MySQL, Redis además nos facilita en gran medida el desarrollo de APIs REST [19].	Base de Datos.
 GIT	Es un sistema de control de versiones que permite realizar el seguimiento de los cambios que se realiza en los archivos. Dispone de un registro de lo que se realizó y puede volver a versiones específicas [20].	Aplicación web.
	Es un editor de código fuente ligero que dispone de potentes herramientas para el desarrollador.	

VISUAL STUDIO CODE	Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes [21].	Aplicación web y Base de datos.
WEBSTORN	Es un entorno de desarrollo integrado para codificar en JavaScript y sus tecnologías relacionadas. Permite que la experiencia de desarrollo sea más agradable automatiza el trabajo de rutina y realiza tareas complejas con facilidad [22].	Aplicación web y Base de datos.
POSTMAN	Es un cliente de API que facilita a los desarrolladores crear, compartir, probar y documentar APIs, ejecutando solicitudes HTTP simples y complejas [23].	Aplicación web.
NOTEPADD ++	Es un editor de código gratuito y un reemplazo del Bloc de notas dispone de una licencia pública general GNU [24].	Aplicación web y Base de datos.
DBEAVER COMMUNITY	Es una herramienta de base de datos universal y gratuita de código abierto. Compatible con cualquier	Base de Datos.

	base de datos que tenga un controlador JDBC [25].	
 CHROME	Es un navegador de Internet gratuito, dispone de sincronización con servicios y cuentas de Google, navegador con pestañas y traducción automática [26].	Aplicación web.
 MYSQL	Es un sistema de administración de bases de datos relacionales de código abierto con un modelo de cliente servidor [27].	Base de Datos.
 DOCKER	Es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones. Permite separar sus aplicaciones de su infraestructura para desarrollar software rápidamente [28].	Base de Datos.
 FIREBASE	Es un backend-as-a-Service (Baas) dispone de herramientas y servicios para ayudar a desarrollar aplicaciones de calidad. Está construida sobre la infraestructura de Google [29].	Aplicación web.

 LUCIDCHAR	Es una herramienta de diagramación basada en la web permite la colaboración de usuarios en tiempo real. Se puede realizar diagramas de flujo, maquetas para sitios web o aplicaciones móviles entre otras [30] .	Mockups.

2 METODOLOGÍA

2.1 Definición de roles

Tabla 1. Roles del equipo Scrum en el proyecto

Equipo Scrum	
Rol	**Responsable**
Scrum Máster	Maritzol Tenemaza
Product Owner	Maritzol Tenemaza
Development Team	Vinicio Quinga

2.2 Levantamiento de requerimientos

El análisis de los requerimientos es una de las principales etapas del desarrollo de software, donde se establecen las condiciones o capacidades que debe cumplir este sistema. Para obtener los requerimientos de la aplicación de bolsa de empleo se investigó el significado y las principales funcionalidades en distintos foros de internet, sistemas existentes en la web, paper, PDF entre, otros [31].

2.3 Aspectos de usabilidad

La usabilidad en la creación del software significa la capacidad de las aplicaciones para ser entendidas, aprendidas y usadas cuando se emplean en determinadas condiciones [32].

Para el presente proyecto se integrará los siguientes aspectos:

Capacidad de aprendizaje, significa la facilidad de comprender la funcionalidad y comportamiento del sistema, por ejemplo, el tiempo que le tomará al usuario utilizar por primera vez la interfaz y realizar operaciones básicas.

Protección contra errores de usuario, significa la capacidad del sistema para evitar que los usuarios cometan errores.

Para evaluar el grado de usabilidad se medirá de forma relativa, esto significa que el resultado no es ni bueno ni malo, sino que depende de las metas planteadas. La cual será que al menos del 70% de los usuarios puedan utilizar el sistema de bolsa de empleo la primera vez.

2.4 Historias Épicas

En la tabla 2 se identifican las historias épicas para el desarrollo del proyecto. El detalle de cada historia de puede encontrar en el anexo I.

Tabla 2. Historias épicas.

CÓDIGO	TÍTULO	PRIORIDAD
AH1	Implementar inicio de sesión.	Alta
AH2	Implementar módulo de estudiante – postulante encargado de gestionar datos del usuario.	Alta

AH3	Implementar módulo de empresa encargado de obtener datos del usuario tipo estudiante - postulante y publicar ofertas de empleo.	Alta
AH4	Implementar módulo de administrador encargado de gestionar la información de los usuarios tipo estudiante – postulante y empresa.	Media
AH5	Aplicación web progresiva (PWA)	Alta

2.5 Product Backlog

En la tabla 3 se puede visualizar el primer product backlog.

Tabla 3. Product Backlog inicial

PRODUCT BACKLOG				
HISTORIA ÉPICA	**HISTORIA DE USUARIO**			
	ID	**Título**	**Estimación (días)**	**Prioridad**
AH1	AH1-01	Inicio de sesión.	15	Alta
AH2	AH2-01	Buscador de ofertas de empleo.	5	Alta
	AH2-02	Registro de datos en hoja de vida del estudiante, egresado o graduado.	7	Alta

	AH2-03	Buscador de oferta de empleos seleccionados por el estudiante egresado o graduado.	3	Baja
AH3	AH3-01	Buscador de ofertas de empleo publicadas por las empresas.	5	Alta
	AH3-02	Registro de datos de oferta de empleo.	3	Alta
	AH3-03	Buscador de estudiantes, egresados o graduados.	5	Alta
	AH3-04	Registro de información de la empresa.	2	Baja
AH4	AH4-01	Registro de estudiantes, egresados o graduados que pertenezcan a la facultad de ingeniería en sistemas de la Escuela Politécnica Nacional.	4	Alta
	AH4-02	Listar a los usuarios tipo estudiantes – postulantes registrados en el sistema.	3	Baja
	AH4-03	Listar a los usuarios tipo empresas registrados en el sistema.	4	Baja
	AH4-04	Registro de datos de roles de usuario.	4	Alta

AH5	AH5-01	Aplicación web progresiva (PWA).	15	Alta

2.6 Definición de sprints

El análisis de los requerimientos dejo un total de 13 historias de usuario y se estableció un tiempo estimando de 3 meses para realizar el proyecto. Por esta razón se decidió desarrollarlo en 5 sprint, cada sprint tendrá la duración de 15 días.

2.7 Sprint 0

2.7.1 Sprint Planning

Objetivos del Sprint

- Diseñar arquitectura del sistema.
- Diseñar modelo lógico de base de datos.
- Crear Mockups con las interfaces de usuario del sistema
- Levantar los ambientes de desarrollo backend y frontend.

2.7.2 Ejecución Del Sprint

Arquitectura del sistema.

Para el desarrollo del software se establece la arquitectura cliente/servidor. El servidor administra los datos y permite realizar operación de búsqueda, creación, actualización y eliminación por medio de los servicios REST. El cliente administra las interacciones con el usuario y visualiza los datos, como se observa en la figura 1.

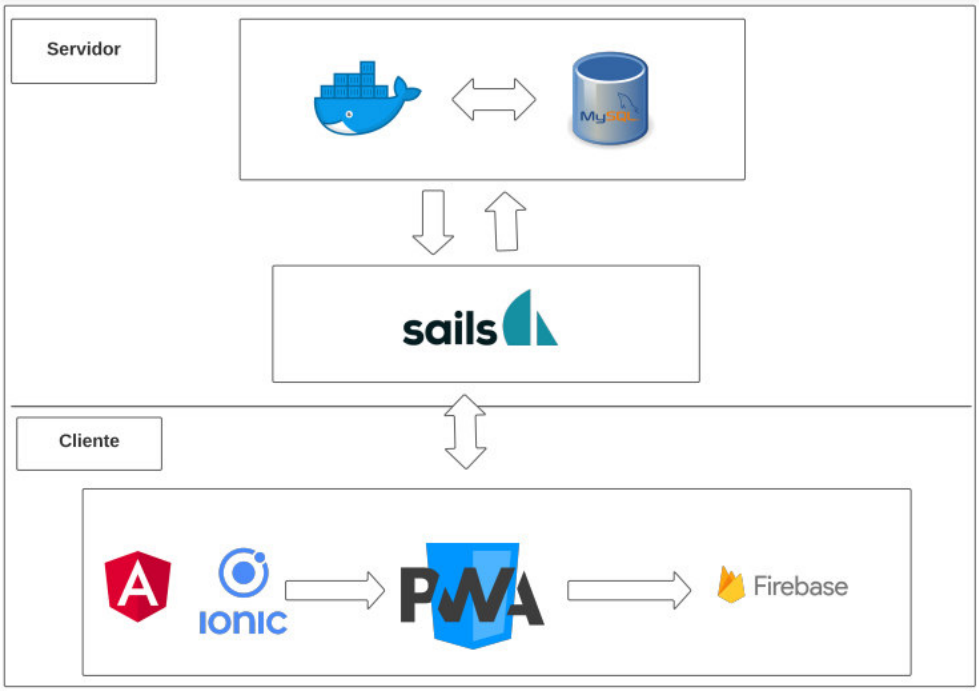


Figura 1. Arquitectura cliente/servidor.

Diseño del diagrama entidad relación de la base de datos

En el diagrama entidad relación se representa 11 entidades con sus propiedades e interrelaciones que fueron obtenidas mediante la investigación realizada a distintas sistema de búsqueda de empleo como se puede observar en la figura 2. En el anexo VII se encuentran con más detalle.

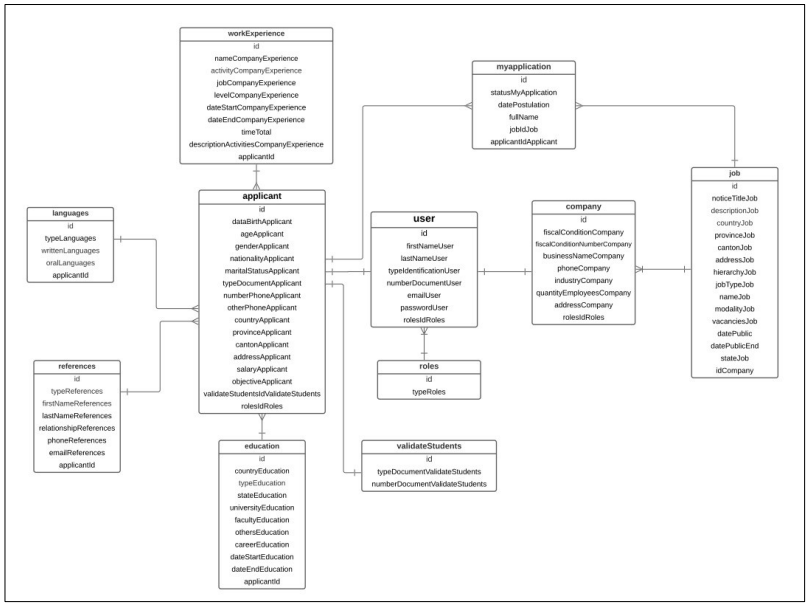


Figura 2. Diagrama de entidad relación.

Mockups del sistema

Para crear los mockups se utiliza la herramienta Lucidchart. En el anexo VIII se encuentran con más detalle.

Inicio de sesión. Se crea la pantalla para inicio sesión el cual solicita las credenciales del usuario y su rol específico. También dispone de las opciones de restablecer la contraseña y crear nuevo usuario, como se pude observa en la figura 3.

Figura 3. Diseño de interfaz para inicio de sesión.

Módulo del usuario tipo estudiante – postulante. Se crea las pantallas para los siguientes servicios un buscador de ofertas de empleo donde se obtendrán todos los datos registrados, un formulario con las características de una hoja de vida donde debe ingresar su información personal, un buscador de ofertas de empleo seleccionadas en donde se podrá ver el estado de esta.

Módulo del usuario tipo empresa. Se crea las pantallas para los siguientes servicios una pantalla para listar las ofertas de empleo publicadas, un formulario de registro de oferta de empleo donde se crea la publicación, un buscador del usuario tipo estudiante – postulante que permitirá obtener a todos los usuarios registrados, un formulario para editar la información de la empresa.

Repositorio GitHub

El proyecto se crea en el repositorio de GitHub con el nombre bolsaDeEmpleo donde se respalda la información del backend y frontend del proyecto. Además, la información de la base de datos como se puede observar en la figura 4.

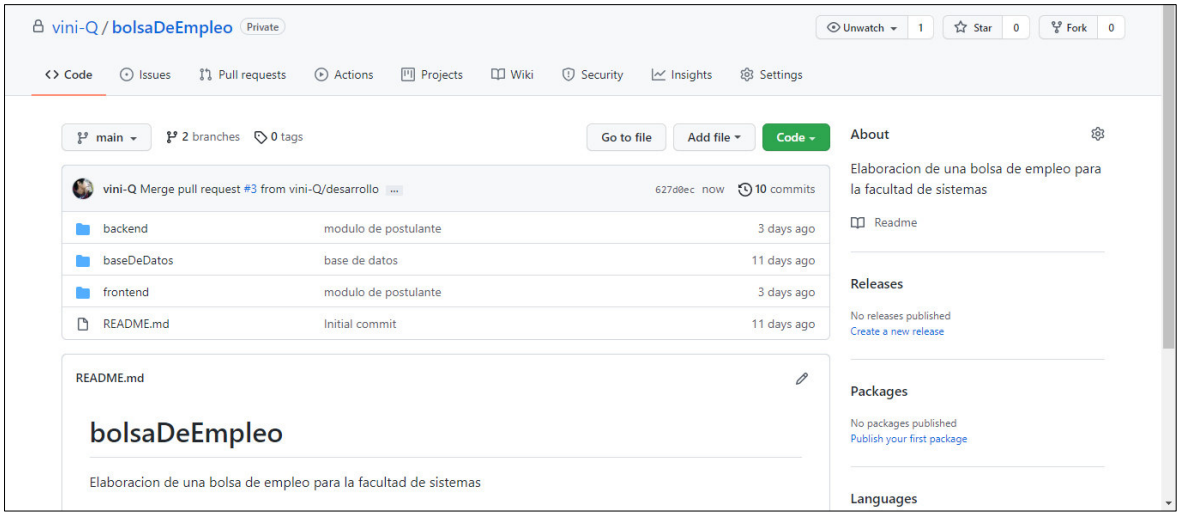


Figura 4. Repositorio de GitHub.

2.8 Sprint 1

2.8.1 Sprint Planning

Objetivo del Sprint

- Construir un servicio de inicio de sesión que permita registrar los datos del usuario.

Historias de Usuario

En la tabla 6 se lista la historia de usuario que fue creada para el primer sprint con su respectiva estimación de tiempo. En el anexo II se encuentran con más detalle.

Tabla 4. Historias de usuario escogidas para el Sprint 1

ID	Título	Estimación (días)	Estado
AH1-01	Inicio de sesión.	15	Por implementar

Sprint Backlog

Tabla 5. Sprint backlog del Sprint 1

SPRINT BACKLOG	
HISTORIA DE USUARIO	**TAREAS**
AH1-01	Configurar autenticación de correo electrónico en Firebase.
	Crear roles de usuario tipo estudiante – postulante, empresa y administrador.
	Implementar servicios para las entidades roles, user, validateStudents, company, authentication-firebase, send-page y alert-message.
	Implementar servicios para métodos de Firebase.
	Implementar CRUD con las entidades roles, user, validateStudents, company.
	Implementar modal para seleccionar usuario (Empresa, Estudiante - Postulante).
	Implementar formulario de validación y de registro de estudiante – postulante.
	Implementar formulario de registro de empresa.
	Implementar funcionalidad para registro de usuario en Firebase.

	Implementar funcionalidad con Firebase para envió de correo electrónico de confirmación para activar cuenta.
	Implementar estilos a formularios.
	Realizar pruebas funcionales.
	Implementar servicios authentication y auth.
	Implementar guards auth, auth-rol, auto-login e intro.
	Crear formulario inicio de sesión.
	Crear método de inicio de sesión en el cual se valida el correo electrónico, contraseña y nombre del rol.
	Implementar verificación para correo electrónico de confirmación.
	Implementar funcionalidad para guardar correo electrónico, id de rol e id de Firebase en local storage.
	Crear formulario para restablecer contraseña.
	Implementar búsqueda de correo electrónico para validar usuario registrado.
	Implementar funcionalidad con Firebase para actualizar contraseña de usuario.
	Realizar pruebas funcionales.

	Corregir errores encontrados
	Implementar estilos a formularios.

2.8.2 Ejecución del Sprint

Pantalla de inicio de sesión

El formulario inicio de sesión se encuentra en la figura 5 como se observa, se solicita el correo electrónico, contraseña y su rol. Los campos señalados son validados por el formulario reactivo en caso de error en los datos se visualiza un mensaje. Además, posee las opciones de crear nuevo usuario y restaurar contraseña.

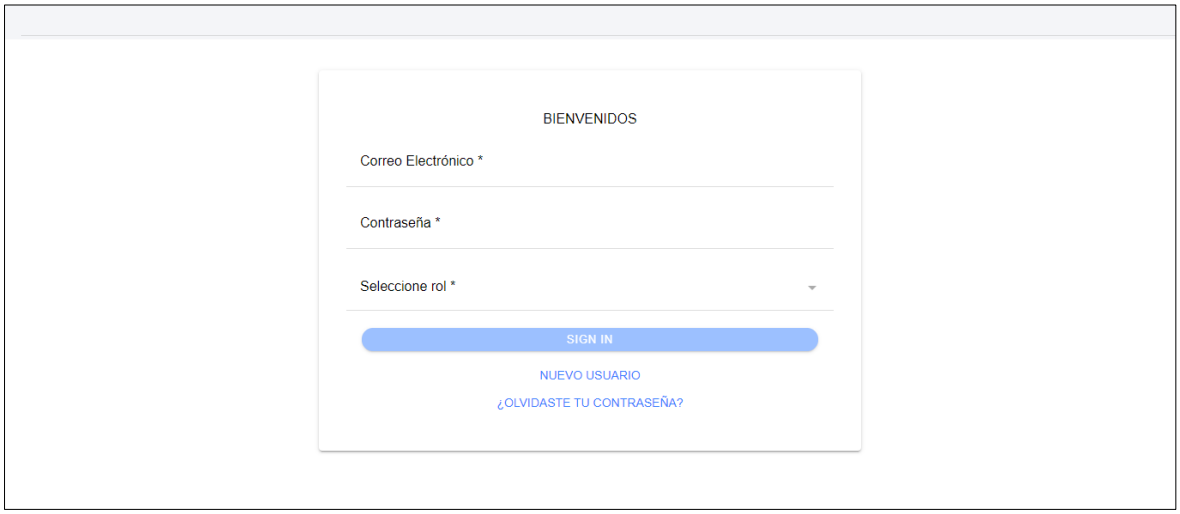


Figura 5. Interfaz de inicio de sesión.

Crear nuevo usuario

Para crear nuevo usuario existen 2 opciones como se observa en la figura 6:

- Usuario tipo empresa.
- Usuario tipo estudiante - postulante.

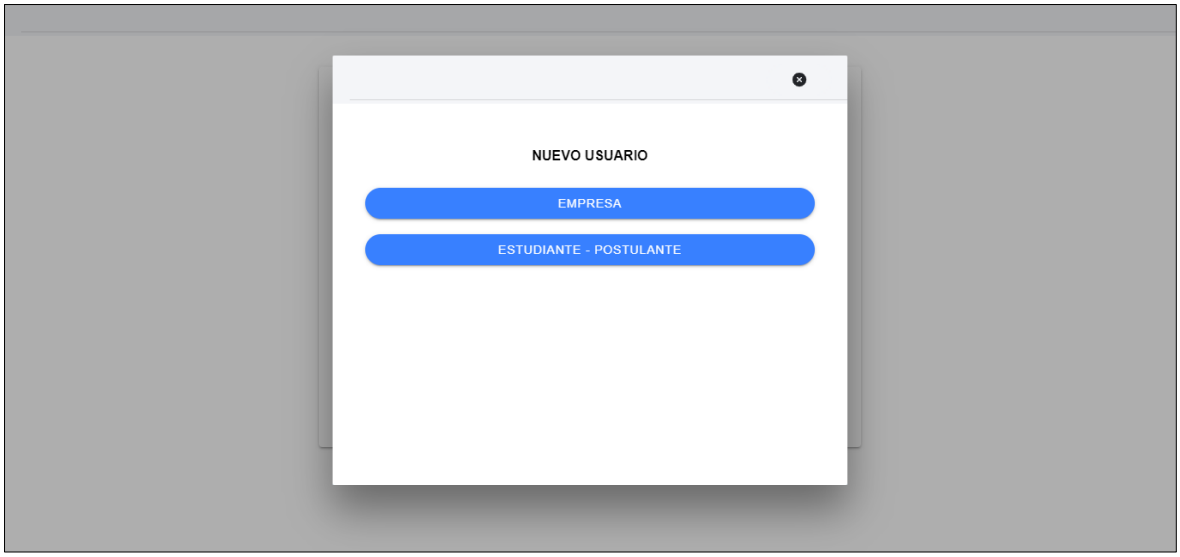


Figura 6. Interfaz para seleccionar rol.

Para los formularios de registro de la empresa y estudiante - postulante permite ingresar los datos de inicio de sesión y sus datos personales. Adicional para el registro del estudiante – postulante se verifica que pertenezcan a la facultad de Ingeniería de sistemas y se asegura que el correo electrónico sea único para un usuario.

Formulario Empresa

En la figura 7 se observa el formulario para el registro de la empresa, se solicita la información de inicio de sesión, los datos ingresados se validan por el formulario reactivo y si existiera algún error se visualiza un mensaje.

- Nombres: Campo requerido y no permite el ingreso de número y caracteres especiales.
- Correo electrónico: Campo requerido y valida que su sintaxis pertenezca a un correo electrónico.
- Contraseña: Campo requerido y valida que ingrese 5 caracteres como mínimo, al menos un número, una letra en mayúscula y una minúscula.
- Confirmar contraseña: campo requerido y valida que la información ingresada sea la misma que el campo contraseña.

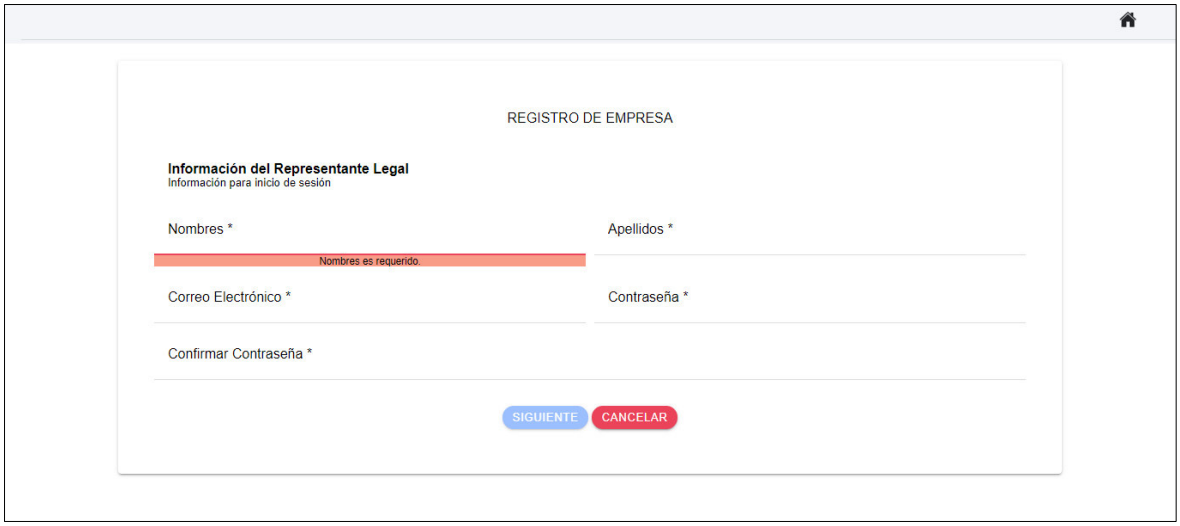


Figura 7. Interfaz de registro de empresa.

En los formularios de registro empresa y estudiante – postulante al finalizar el proceso, si no existe ningun error la informacion se guarda en el backend y en firebase. Además, se envia un mensaje de validación al correo electrónico registrado, sino se cumple este proceso no podrá ingresar al sistema. Si surge un problema con el mensaje de validación cuenta con la opción de reenviar dicho mensaje, como se puede ver en figura 8.

Figura 8. Mensaje de éxito para crear nuevo usuario.

Restablecer contraseña

Se implementa un formulario para restablecer la contraseña del usuario, solicitando su rol y correo electrónico. Este formulario actualiza la contraseña del backend y firebase, como se puede observar en la figura 9.

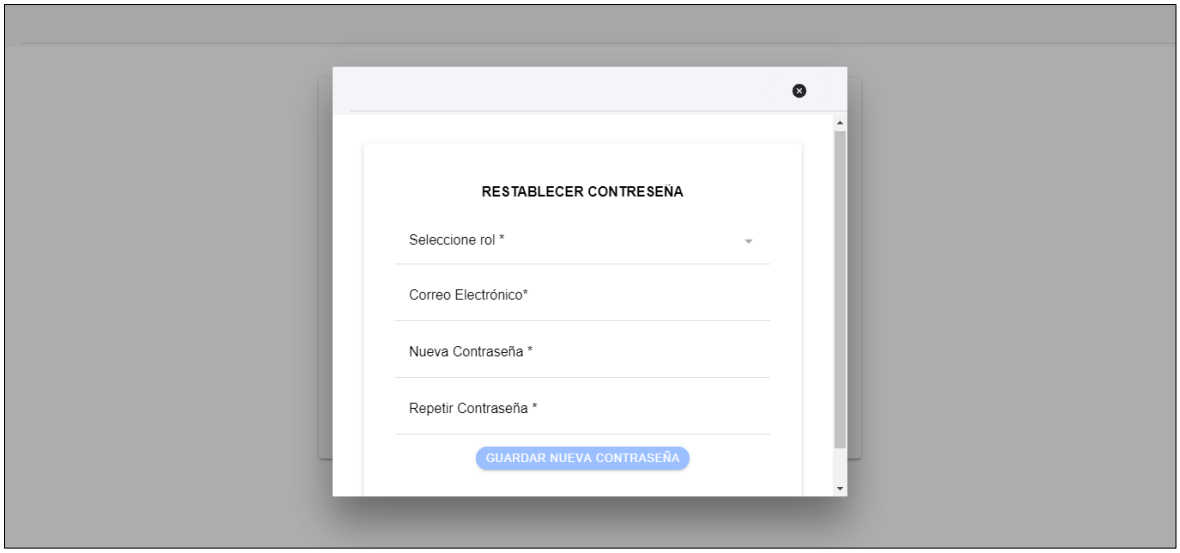


Figura 9. Interfaz para restablecer contraseña.

Protección de rutas

Se implementó GUARD tecnología de Angular para proteger las rutas. Estos GUARDS se ejecutan antes de cargar las rutas y verifica si se puede cargar. En este caso se utilizó canLoad que verifica antes de cargar los recursos de la ruta, como se observa en la figura 10.

```
path: 'applicant',
children:
  [
    {
      path: 'applicant-profile',
      loadChildren: () => import('./pages/applicant/applicant-profile/applicant-profile.module').then(m => m.ApplicantProfilePageModule)
      canLoad: [AuthGuard]
    },
```

Figura 10. Ruta de usuario tipo estudiante – postulante.

Información en backend

Los datos del usuario tipo empresa que se obtienen al consumir el servicio del API RESTfult es en formato JSON y se puede visualizar por medio de la herramienta Postman, como se puede observar en la figura 11. Estos datos se obtienen al guardar la información del formulario empresa:

- passwordUser: utiliza la biblioteca CryptoJS de JavaScript y por medio del algoritmo AES cifrar la contraseña del usuario.
- rolesIdUser: guarda el identificador del rol
- idCompany: guarda el identificador de tabla compañía.

```
[
  {
    "idApplicant": [],
    "idCompany": [
      {
        "createdAt": 1636782006038,
        "updatedAt": 1636782006038,
        "id": 2,
        "fiscalConditionCompany": "RUC",
        "fiscalConditionNumberCompany": "1723456544001",
        "businessNameCompany": "Prueba Empresa",
        "phoneCompany": "2265471",
        "industryCompany": "Informática / Tecnologia",
        "quantityEmployeesCompany": "",
        "addressCompany": " Site Center, Calle E, Quito 170157",
        "rolesIdRoles": 10
      }
    ],
    "createdAt": 1636781952827,
    "updatedAt": 1636782193645,
    "id": 10,
    "firstNameUser": "Carlos ",
    "lastNameUser": "Puente",
    "typeIdentificationUser": "",
    "numberDocumentUser": "",
    "emailUser": "vinicio_520@hotmail.com",
    "passwordUser": "U2FsdGVkX1+/n63wLh9m/xvBdq8nT685zqGe1/4Kfzc=",
    "rolesIdRoles": {
      "createdAt": 1636777171750,
      "updatedAt": 1636777171750,
      "id": 2,
      "typeRoles": "Empresa"
    }
  }
]
```


Figura 11. JSON con datos del usuario tipo empresa.

2.8.3 Sprint Review

En el cumplimiento del sprint se tuvo que solventar problemas con la implementación del servicio de guards, por lo que se investigó en la documentación de Angular y en diferentes foros de internet la implementación de este servicio. Además, se realizó la primera versión del inicio de sesión con la protección de las rutas utilizando guards.

Esta versión tiene 2 guards, el primer guards se utiliza para validar si el usuario ha iniciado sesión y dependiendo del resultado sea verdadero o falso se presenta la interfaz correspondiente. En el caso que sea verdadero significa que el usuario no ha iniciado sesión por lo que es dirigido a la pantalla de login, y en el caso que sea falso significa que el usuario ya ha iniciado sesión y tiene que dirigirse a las pantallas correspondiente de cada usuario.

En este segundo caso donde el resultado fue falso se presentó un error ya que el guards no tenía la ruta para dirigirse a las interfaces correspondientes. Para solucionar este problema se guardó en el localstorage el identificador del rol al cual pertenece cada usuario, y adicional se creó un método para validar y direccionar al usuario a sus respectivas pantallas.

El segundo guards se creó para proteger a las rutas y no permitir el ingreso de usuario que no estén logueados, este proceso no genero ningún problema. Solventado el error del primer guards se pudo aprobar las pruebas funcionales y continuar con las siguientes funciones planeadas para el sprint.

Pruebas de aceptación

Como se observa en la Tabla 8, la verificación de cada proceso se lo realizó mediante la aplicación de las pruebas de aceptación correspondientes.

Tabla 6. Pruebas de aceptación del Sprint 1

Historias de usuario	Criterios de aceptación	Cumplido
AH1-01	Dado el despliegue del formulario registro de empresa y estudiante – postulante cuando los campos requeridos estén llenos, entonces, los botones tipo submit se activarán.	Si
	Dado el despliegue del servicio del inicio de sesión cuando el usuario se loguea se verificará que su correo electrónico registrado este validado, entonces, le permitirá el ingreso al sistema.	Si
	Dado el despliegue del formulario de registro de empresa y estudiante – postulante cuando se proceda a guardar la información en el backend, entonces, se verifica que el correo electrónico no este registrado.	Si
	Dado el despliegue del inicio de sesión cuando el usuario se loguea y no este validado su correo electrónico, entonces, se permite reenviar el mensaje de validación.	Si
	Dado el despliegue del inicio de sesión cuando el usuario se loguea y no se genere ningún problema, entonces, se	Si

	guarda su información (email, id rol e id de Firebase) en el local storage.	
	Dado el despliegue del formulario restablecer contraseña cuando el usuario cambie de contraseña, entonces, la nueva información se actualizará en el backend y en firebase.	Si

Incremento obtenido

Registro de usuarios

En las figuras 12, 13, 14, se puede observar el procedimiento para el registro del usuario tipo empresa.

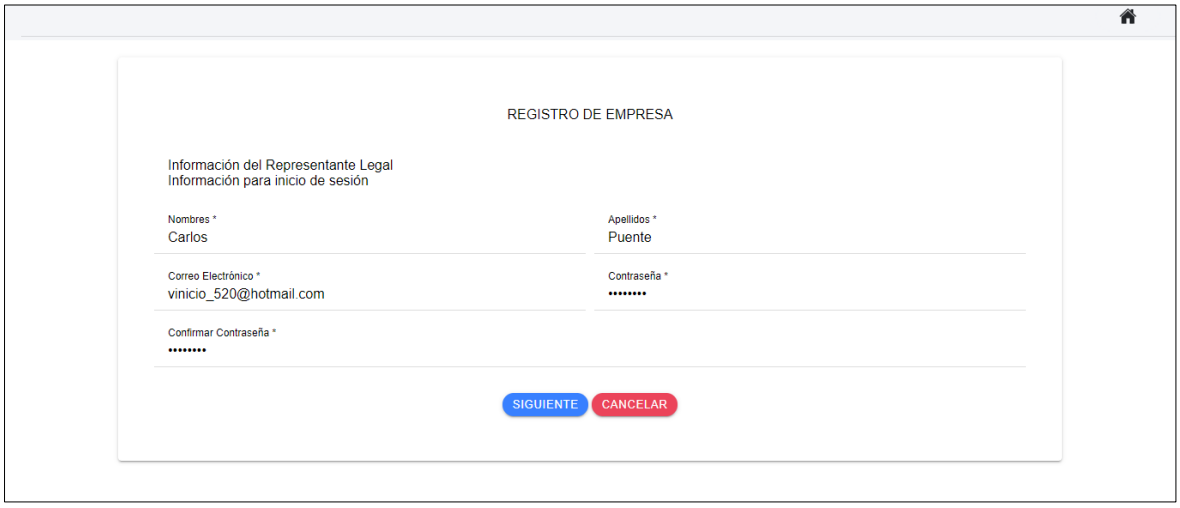


Figura 12. Formulario de registro de usuario tipo empresa, datos para inicio de sesión.

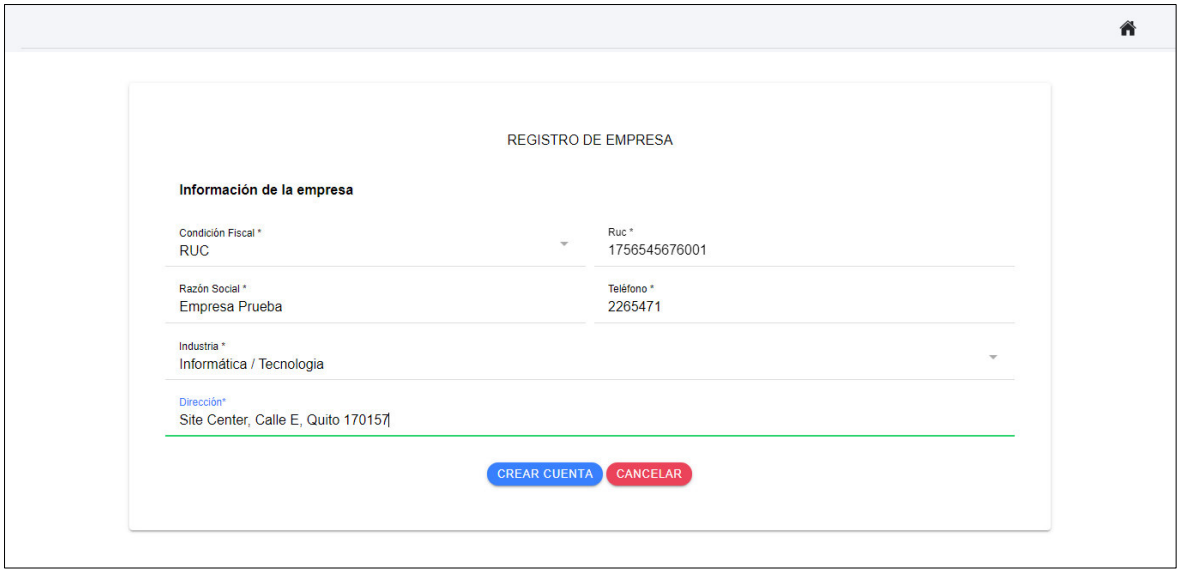


Figura 13. Formulario de registro de usuario tipo empresa, datos con la información de la empresa.

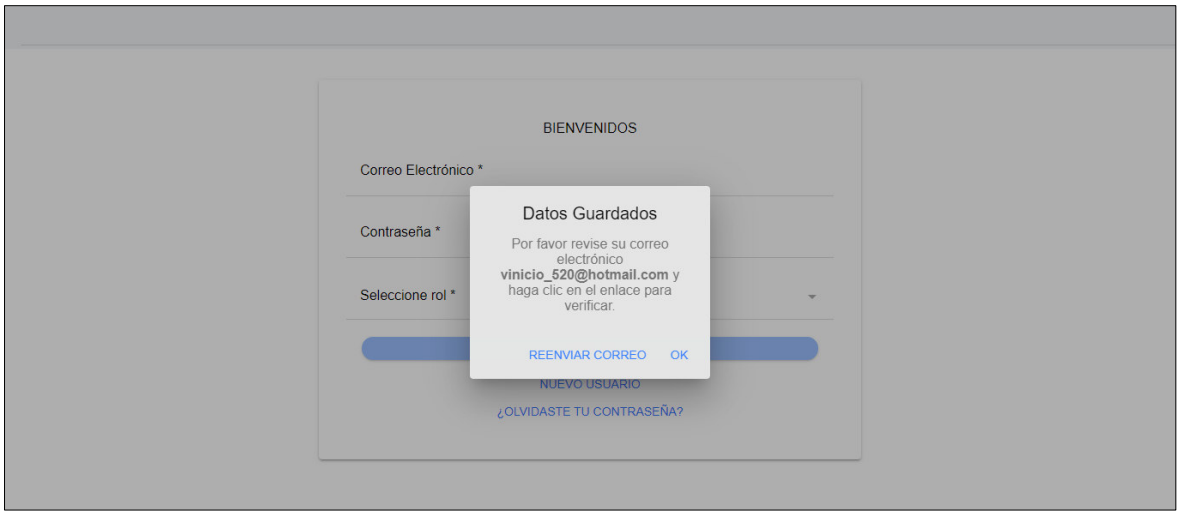


Figura 14. Mensaje de usuario creado correctamente, adicional opción para enviar mensaje de validación.

Inicio de sesión

En la figura 15 se observa el formulario de inicio de sesión, el cual solicita al usuario el correo electrónico, contraseña y rol, como campos requeridos. Adicional cuenta con los botones de registro de nuevo usuario y de restaurar contraseña.

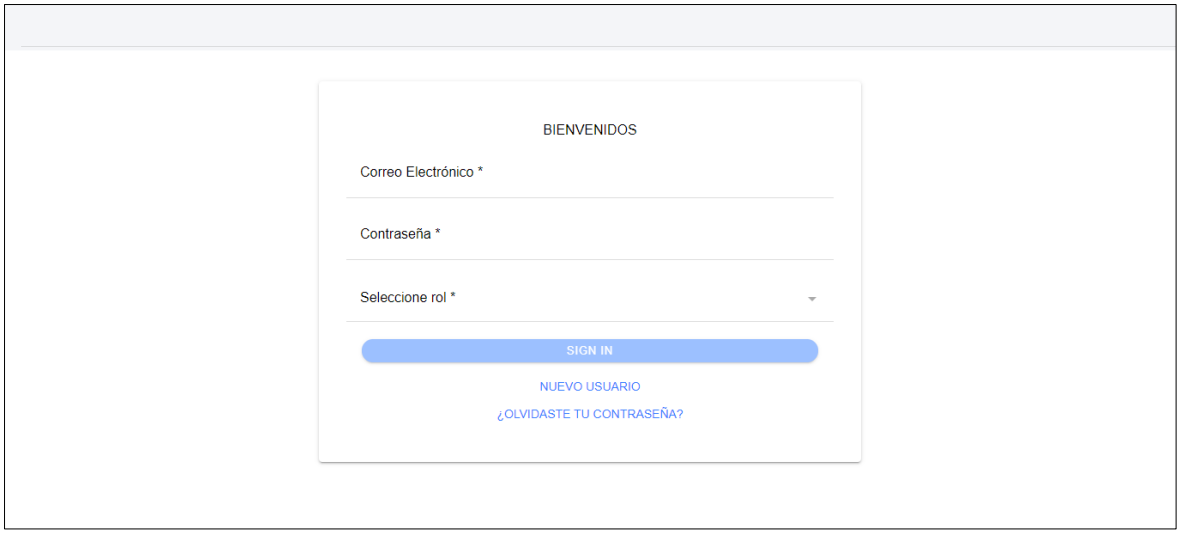


Figura 15. Formulario de inicio de sesión.

El servicio de inicio de sesión valida y guarda en la base de datos la información, también verifica si el correo electrónico se validó con el servicio de Firebase. Existen 2 casos de pruebas, como se observa en la figura 16, 17, 18,19.

Inicio de sesión exitoso.

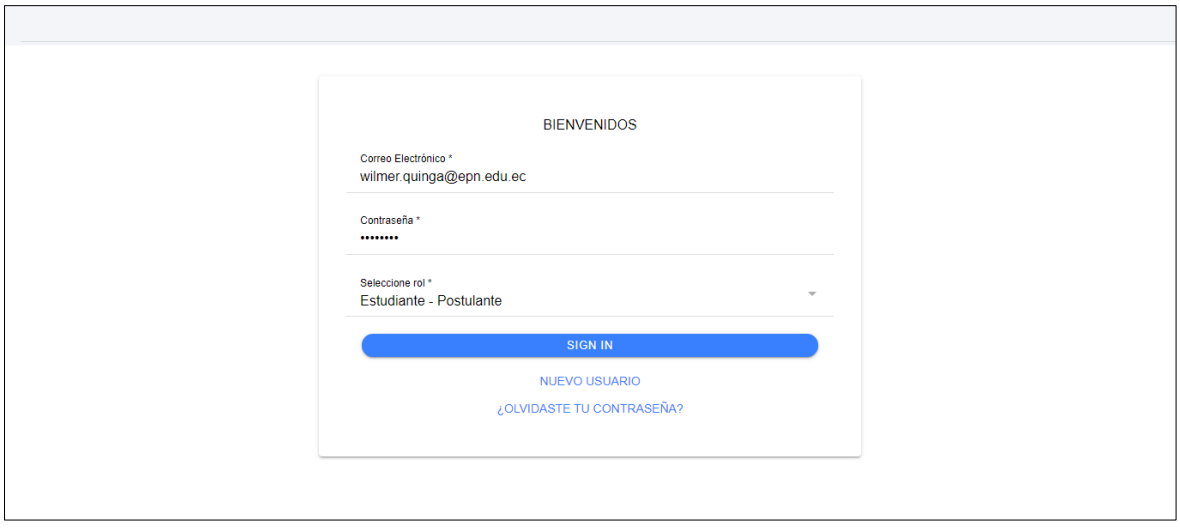


Figura 16. Caso de prueba exitoso inicio de sesión ingreso de datos.

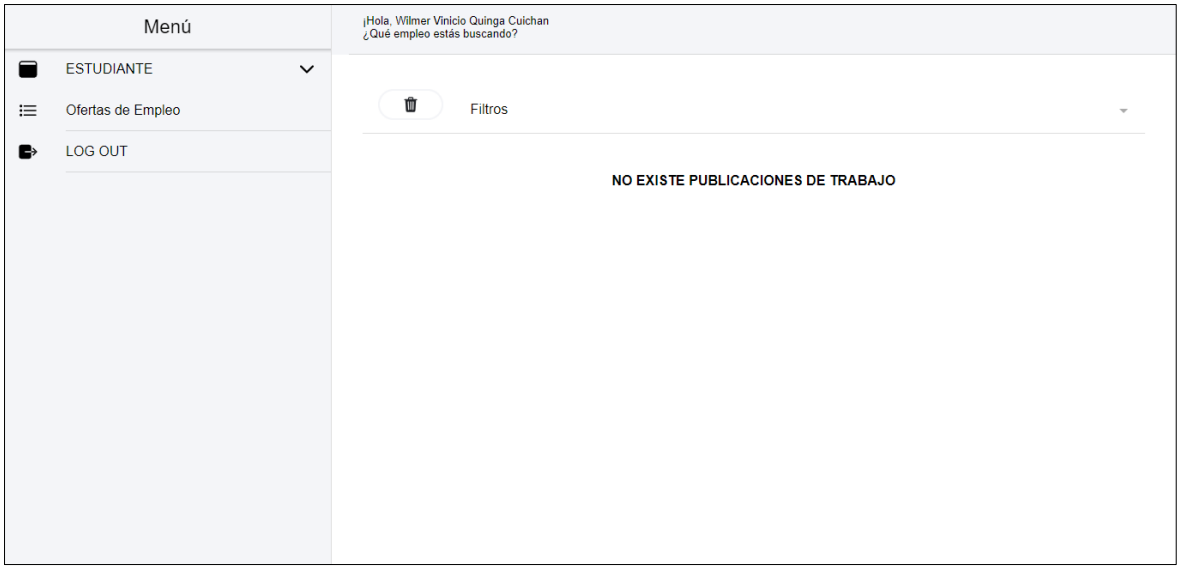


Figura 17. Caso de prueba exitoso inicio de sesión visualización del menú del usuario tipo estudiante - postulante.

Inicio de sesión fallida.

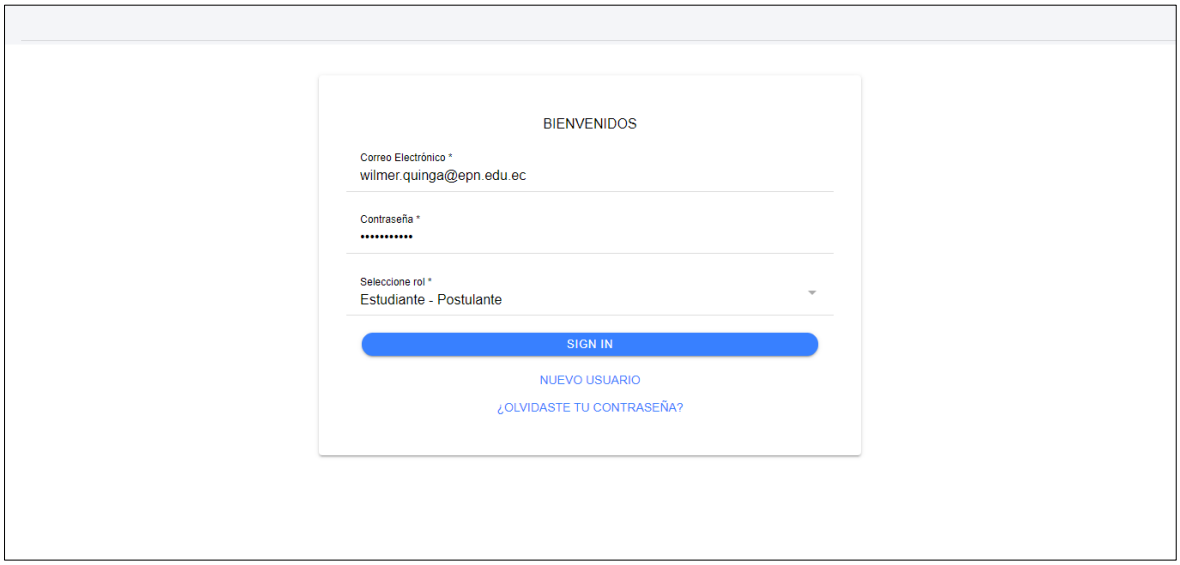


Figura 18. Caso de prueba fallida inicio de sesión ingreso de datos.

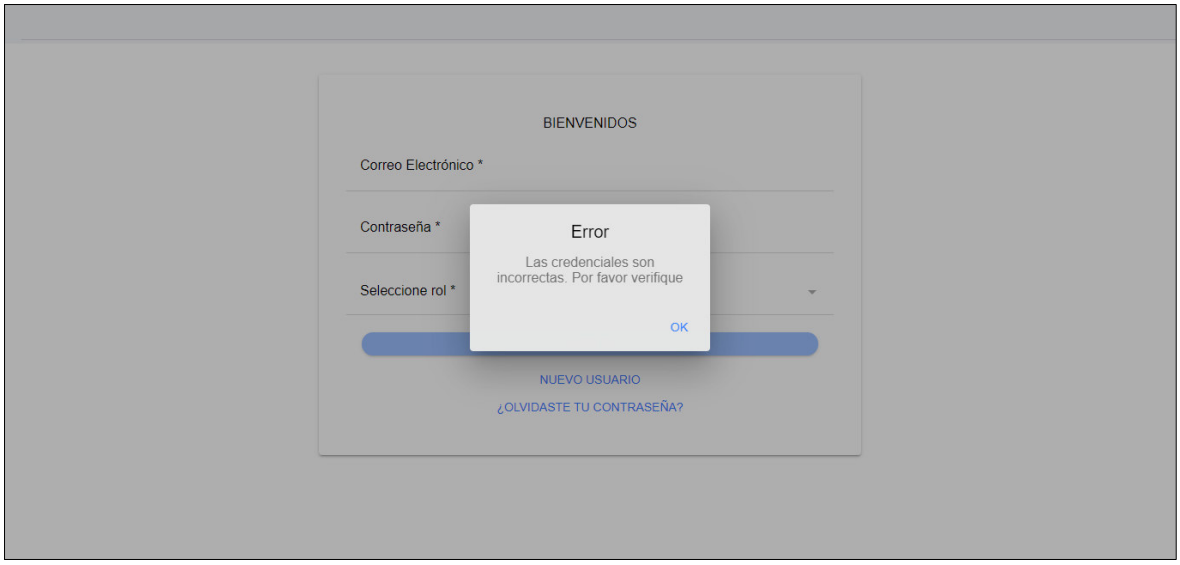


Figura 19. Caso de prueba fallida inicio de sesión.

2.8.4 Sprint Retrospective

¿Qué salió bien en la interacción?

La implementación del servicio de autenticación del correo electrónico con Firebase, se envía un email al correo electrónico registrado por el usuario, con el objetivo de verificar que el correo electrónico sea válido; la empresa o el estudiante - postulante deben de responder para activar su usuario.

¿Qué se puede mejorar?

Se puede reutilizar el componente que contiene el formulario de registro con la información de inicio de sesión, como se observa en la figura 14. Este componente se puede utilizar con los 2 usuarios empresas y estudiantes – postulantes. La información específica de cada usuario se puede ocultar o habilitar con un *ngIf obteniendo un componente dinámico.

2.9 Sprint 2

2.9.1 Sprint Planning

Objetivo del Sprint

Crear el módulo del usuario tipo estudiante – postulante con los siguientes servicios.

- Implementar un servicio de buscador de ofertas de empleo registradas por las empresas.

- Implementar un formulario para el registro de la hoja de vida del estudiante - postulante.
- Implementar un servicio de buscador de las ofertas de empleo seleccionadas por el usuario.

Historias de Usuario

En la tabla 10, se listan las historias de usuarios que fueron creadas para el segundo sprint con su respectiva estimación de tiempo. En el anexo III se encuentran con más detalle las historias de usuario.

Tabla 7. Historias de usuario escogidas para el Sprint 2.

ID	Título	Estimación (días)	Estado
AH2-01	Buscador de ofertas de empleo.	5	Por implementar
AH2-02	Registro de datos en hoja de vida del estudiante, egresado o graduado.	7	Por implementar
AH2-03	Buscador de oferta de empleo seleccionadas por el estudiante, egresado o graduado.	3	Por implementar

Sprint Backlog

Tabla 8. Sprint backlog del Sprint 2

SPRINT BACKLOG	
HISTORIA DE USUARIO	**TAREAS**
	Implementar servicio para las entidades job, myapplication.

AH2-01	Crear menú lateral izquierda para organizar funcionalidad del sistema.
	Crear una cabecera para todos los formularios, los títulos dependiendo de cada formulario.
	Crear formulario de búsqueda para las ofertas de empleo.
	Implementar método para obtener todas las ofertas de empleo.
	Implementar filtros de búsqueda por ejemplo fecha de publicación, modalidad de trabajo, jornada laboral, nivel de experiencia.
	Implementar un botón para limpiar los filtros seleccionados.
	Crear métodos para obtener información dependiendo de cada filtro.
	Crear modal para obtener más información de ofertas de empleo.
	Implementar botón de postulación en las ofertas de empleo.
	Guardar información de la fecha de postulación y el estado en la entidad myapplication
	Implementar estado habilitar y deshabilitar en botón de postulación.
	Implementar estilos en formularios.
	Validar información guardar en la base de datos.
	Realizar pruebas funcionales.

	Corregir errores encontrados
	Implementar estilos en los formularios.
AH2-02	Implementar servicios para las entidades education, language, reference, workExperience.
	Crear formulario para datos personales.
	Implementar método para calcular la edad del usuario.
	Implementar botón para visualizar y editar datos personales.
	Crear archivos con datos para comboBox de todo el sistema.
	Obtener los datos del formulario datos personales y realizar CRUD con las entidades user y applicant.
	Realizar pruebas funcionales.
	Validar información en la base de datos.
	Corregir errores encontrados.
	Crear formulario para datos de contacto.
	Implementar comboBox país, provincia y cantón. Con datos en cascada.
	Crear archivo con datos para comboBox.

	Implementar botón para visualizar y editar datos de contacto.
	Obtener los datos del formulario datos de contacto y realizar CRUD en la entidad applicant.
	Realizar pruebas funcionales.
	Validar información en la base de datos.
	Corregir errores encontrados.
	Crear formulario para preferencia salarial.
	Implementar botón para visualizar y editar datos de preferencia salarial.
	Obtener los datos del formulario preferencia salarial y realizar CRUD con la entidad applicant.
	Realizar pruebas funcionales.
	Validar información en la base de datos.
	Corregir errores encontrados.
	Crear formulario para objetivo laboral.
	Implementar botón para visualizar y editar datos de objetivo laboral.
	Implementar ingreso de 500 caracteres.

	Obtener los datos del formulario datos personales y realizar CRUD con la entidad applicant.
	Realizar pruebas funcionales.
	Validar información en la base de datos.
	Corregir errores encontrados.
	Crear formulario educación.
	Implementar botón para visualizar y editar datos del formulario educación.
	Implementar lógica para visualizar datos del formulario educación en forma de una tabla.
	Obtener los datos del formulario educación y realizar CRUD con las entidades applicant y education.
	Realizar pruebas funcionales.
	Validar información en la base de datos.
	Corregir errores encontrados.
	Crear formulario idiomas
	Implementar botón para visualizar y editar datos del formulario idiomas.

	Implementar lógica para visualizar datos del formulario idiomas en forma de una tabla.
	Crear datos para comboBox.
	Obtener los datos del formulario idiomas y realizar CRUD con las entidades applicant y language.
	Realizar pruebas funcionales.
	Validar información en la base de datos.
	Crear formulario experiencias laborales.
	Implementar botón para visualizar y editar datos del formulario experiencias laborales.
	Implementar lógica para visualizar datos del formulario experiencias laborales en forma de una tabla.
	Crear método para calcular tiempo entre 2 fechas.
	Implementar textArea para ingresar 250 caracteres.
	Obtener los datos del formulario idiomas y realizar CRUD con las entidades applicant y workExperience.
	Realizar pruebas funcionales.
	Validar información en la base de datos.

	Corregir errores encontrados.
	Crear formulario referencias.
	Implementar botón para visualizar y editar datos del formulario referencias.
	Crear método para visualizar relación laboral.
	Obtener los datos del formulario idiomas y realizar CRUD con las entidades applicant y references.
	Realizar pruebas funcionales.
	Corregir errores encontrados.
	Validar información en la base de datos.
	Implementar estilos en los formularios.
AH2-03	Crear formulario para listar las postulaciones realizadas.
	Implementar método para obtener todas las postulaciones realizadas
	Implementar filtros de búsqueda para las postulaciones. Todas, iniciadas, leídas y finalizadas.
	Crear métodos para obtener información dependiendo de cada filtro.
	Validar información guardar en la base de datos.

	Realizar pruebas funcionales.
	Corregir errores encontrados.
	Implementar estilos en el formulario.
	Implementar y pruebas con guards en rutas creadas.

2.9.2 Ejecución del Sprint

Buscador de ofertas de empleo.

Para obtener los datos de las ofertas de empleo se utiliza un servicio API RESTFul. En este caso solo se obtendrán los ítems que tengan estado de publicado, como se muestra en la figura 20.

```
listAllJobActive() {
  return this.httpClient.get( url: URL + '/job?stateJob=Publicado');
}
```

Figura 20. Servicio API RESTFul de la entidad job.

Los datos que se obtienen son configurados para visualizar en la pantalla, estos datos ofrecen detalles de la oferta de empleo adicional dispone de filtros y de un botón para más información, donde se realizará la postulación, como se observa en la figura 21.

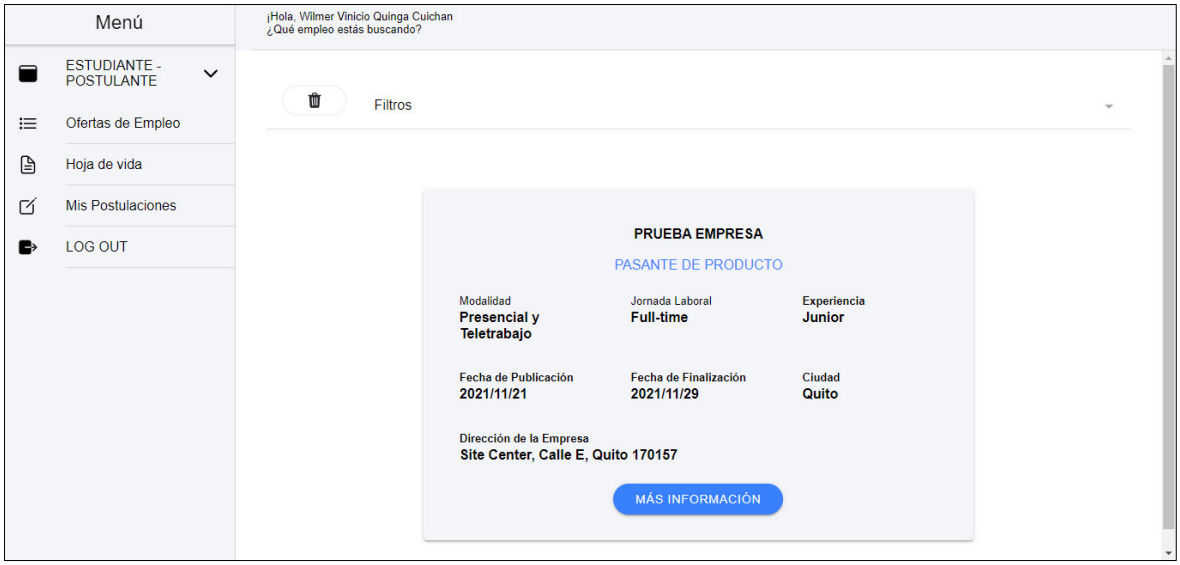


Figura 21. Pantalla de búsqueda de oferta de empleo.

Disponen de 5 tipos de filtros para obtener la información según el criterio del usuario. También existe un botón para limpiar los filtros seleccionados, como se puede observar en figura 24.

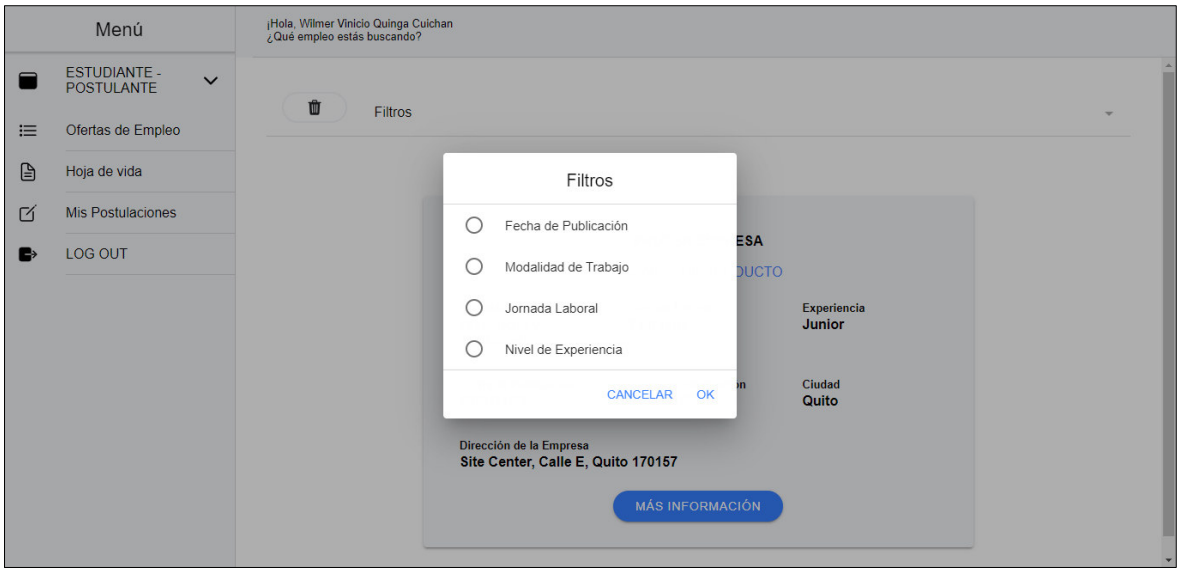


Figura 22. Tipos de filtros que dispone la interfaz de búsqueda de oferta de empleo.

El filtro fecha de publicación utiliza el formato aaaa/mm/dd en los datos. Para construir la consulta se utiliza Query Languages tecnología de Sails que interpreta una consulta y devuelve los datos encontrados, como se puede observar en la figura 23. En la parte izquierda se visualiza la consulta realizada.

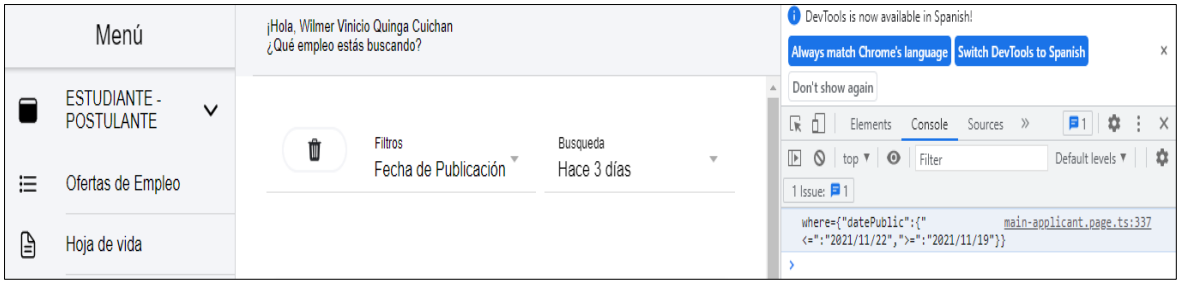


Figura 23. Filtro fecha de publicación.

El botón más información proporciona más detalles de la oferta de empleo adicional proporciona el botón de postulación que posee las siguientes validaciones:

- El usuario no se podrá postular si aún no ha llenado el formulario de hoja de vida.
- Solo podrá realizar una vez la postulación.
- No podrá eliminar la postulación realizada.

Si el usuario ya realizó la postulación, el botón se desactivará y publicará la fecha que realizó la solicitud, como se puede observar en la figura 24.

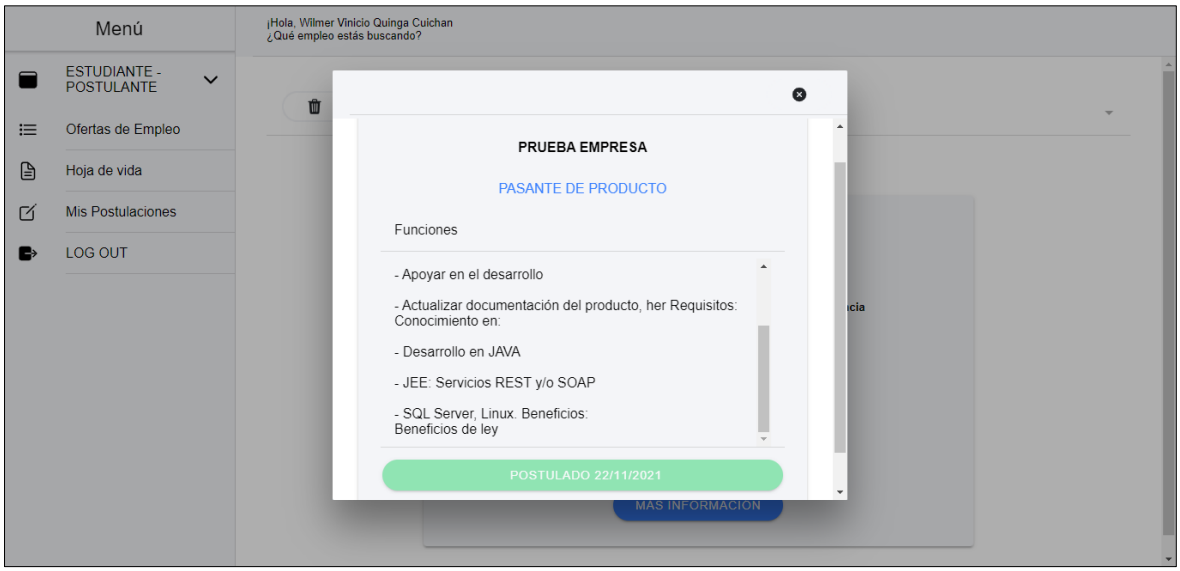


Figura 24. Interfaz de búsqueda de oferta de empleo visualizando el detalle.

Formulario Hoja de vida

En la figura 25, se observa el formulario hoja de vida del estudiante – postulante que utilizan las empresas para elegir al candidato idóneo.

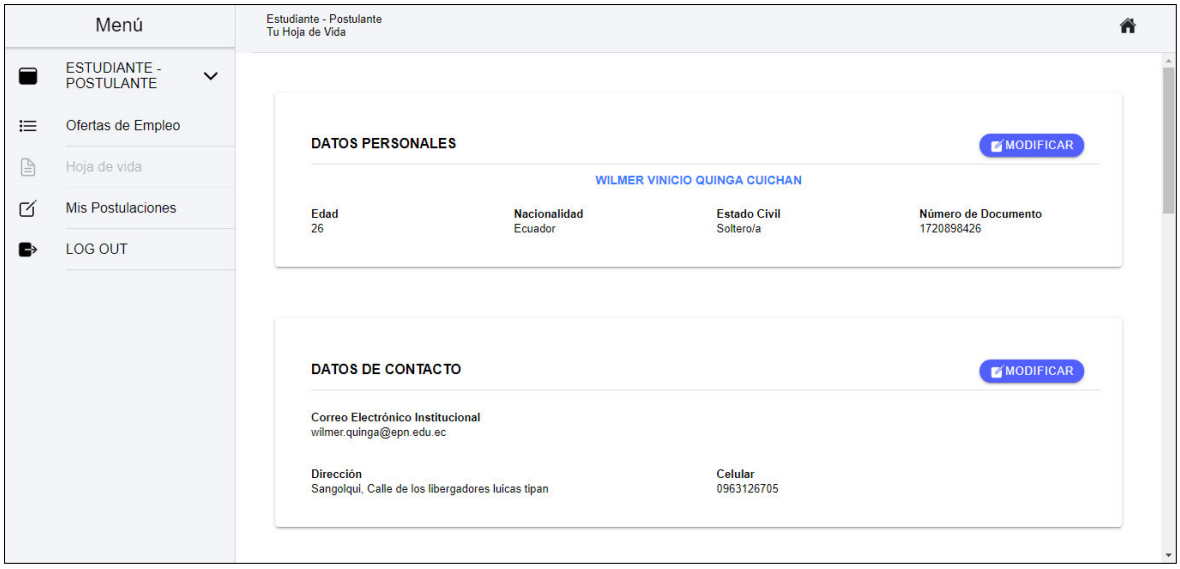


Figura 25. Interfaz con formulario de hoja de vida.

El formulario hoja de vida cuenta con 8 grupos de datos:

- Datos personales: Permite ingresar la información personal del usuario nombres y apellidos, fecha de nacimiento entre otras.
- Datos de contacto: Permite ingresar información mediante el cual el postulante sea localizado. Se solicita número de teléfono, correo electrónico entre otros.
- Preferencia salarial: Se solicita el salario que el usuario pretende recibir por la oferta de empleo.
- Objetivo: Permite ingresar un objetivo de trabajo para la oferta de empleo.
- Educación: Permite ingresar la información sobre la educación del usuario. Dispone opciones desde el colegio hasta doctorado.
- Idiomas: Permite ingresar los idiomas que maneje el postulante es un requisito que se toma mucho en cuenta las empresas.
- Experiencias laborales: Permite ingresar información sobre sus anteriores experiencias laborales, esta información es evaluada por el departamento de recursos humanos, dependiendo de los resultados se podría llevar a cabo el primer contacto con el usuario tipo estudiante – postulante.
- Referencias: Permite ingresar información de los datos de contacto de personas seleccionadas por el usuario tipo estudiante – postulante.

El botón modificar permite en el formulario visualizar los detalles de cada grupo de datos. Como se puede observar en la figura 26, el formulario datos personales solicita el llenado

de todos los campos, la información debe cumplir con las validaciones establecidas para activar el botón guardar.

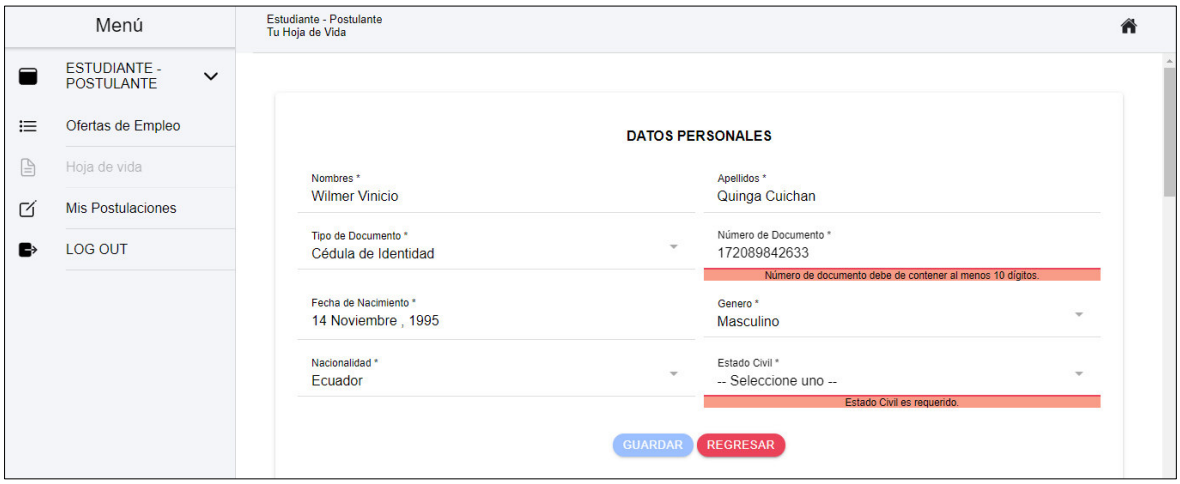


Figura 26. Interfaz con formulario de hoja de vida con detalle de datos personales.

El formulario idiomas en su componente se establece inputs (Es un decorador que establece al campo la propiedad de entrada de datos) con las propiedades GET y SET para detectar los nuevos valores recibidos. Como se puede observar en la figura 27. Adicional, en la página encargada de llamar al componente antes mencionado, se estableció la propiedad IONVIEWDIDENTER perteneciente al ciclo de vida IONIC el cual permite consultar datos al backend antes que la página sea cargada, como se observa en la figura 28. Esta configuración permite crear, actualizar, borrar datos y visualizar los nuevos valores sin la necesidad de recargar la página conservando la propiedad de una sola página de angular, como se puede observar en la figura 29.

```
@Input()
set dataLanguageInput(value: any[]) {
  this.dataLanguageModel = value;
}

get dataLanguageInput(): any[] {
  return this.dataLanguageModel;
}
```

Figura 27. Input con las propiedades GET y SET.

```
ionViewDidEnter() {
  this.getUser();
}
```

Figura 28. Evento Ionviewwillenter perteneciente al ciclo de vida de Ionic.

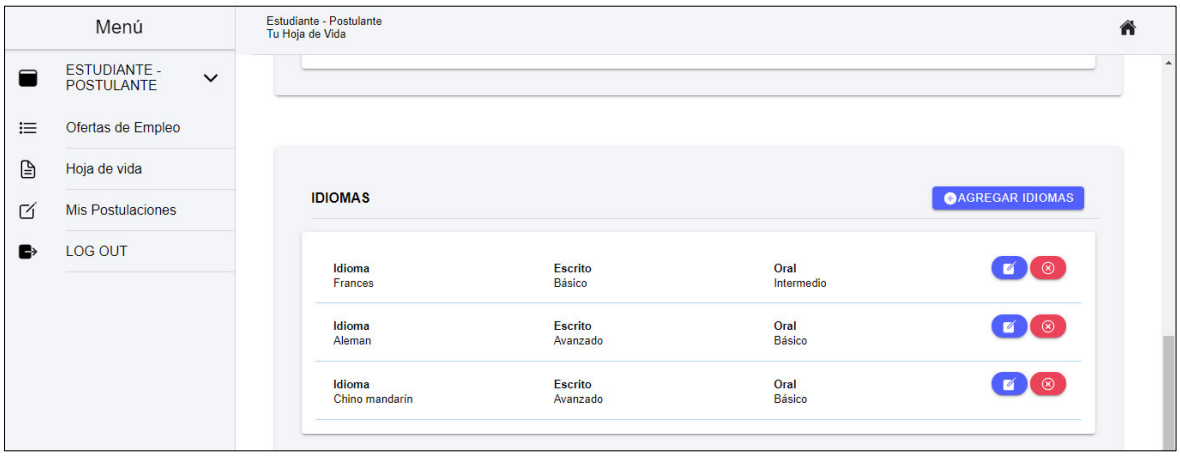


Figura 29. Pantalla del formulario hoja de vida grupo Idiomas.

Buscador de ofertas de empleo seleccionadas por el usuario

Para realizar la búsqueda se consume el servicio API RESTFul con la entidad myapplication como se observa en la figura 30, el atributo que recibe es el identificador del usuario de inicio sesión para obtener solo sus datos. Para realizar la consulta de los filtros se agrega el atributo estado, con las diferentes categorías iniciadas, leída y finalizadas, como se puede observar en la figura 31.

```
listPostulationIdApplicant(idApplicant: number) {
  return this.httpClient.get( url: URL + '/myapplication?applicantIdApplicant=' + idApplicant);
}
```

Figura 30. Servicio API RESTFul con la entidad myapplicantion – identificador user.

```
getSearchPostulation(idApplicant: number, status: string) {
  return this.httpClient.get( url: URL + '/myapplication?applicantIdApplicant=' + idApplicant + '&statusMyApplication=' + status);
}
```

Figura 31. Servicio Api Restful con la entidad myapplicantion con la propiedad estado.

Como se observa en la figura 32, el formulario donde se visualiza los datos de las ofertas de empleo seleccionadas por el usuario permite conocer el estado de la postulación.

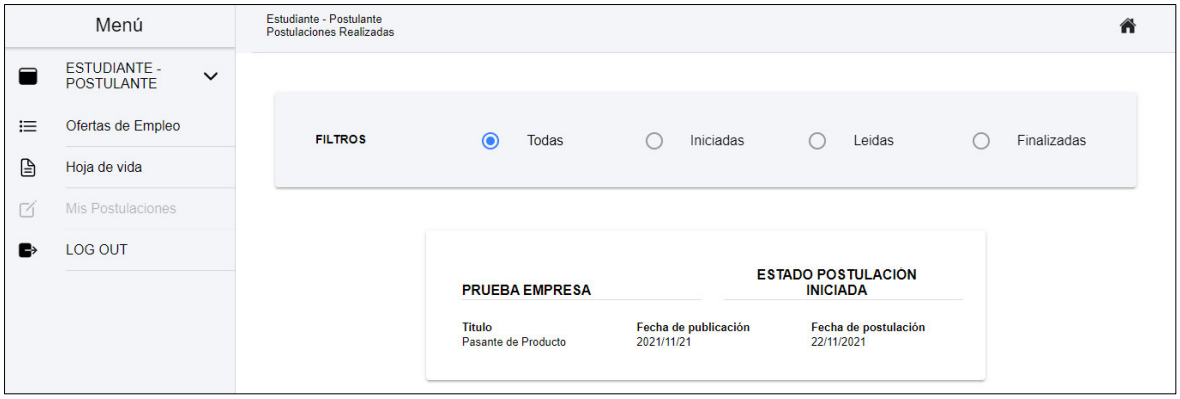


Figura 32. Interfaz búsqueda postulación realizada.

2.9.3 Sprint Review

El objetivo del sprint se cumplió con éxito respetando los tiempos establecidos, surgió un problema cuando actualizaba los datos en el backend, no reflejaba los cambios en la interfaz. Este problema se solucionó investigando documentación del ciclo de vida de IONIC.

Pruebas de aceptación

En la Tabla 12 se listan los criterios de aceptación cumplidos para el segundo *sprint*.

Tabla 9. Pruebas de aceptación del segundo Sprint.

Historia de Usuario	Criterios de aceptación	Cumplido
AH2-01	Dado el despliegue del formulario buscar de ofertas de empleo cuando el usuario ingrese a este servicio, entonces, se obtendrán las ofertas de empleo en estado activo.	Si
	Dado el despliegue del formulario buscador de ofertas de empleo cuando el usuario utiliza el servicio, entonces, se activarán las opciones para filtrar información.	Si
	Dado el despliegue del formulario buscador de ofertas de empleo cuando el usuario seleccione una oferta de empleo,	Si

	entonces, se visualizará un botón para obtener más información.	
	Dado el despliegue del formulario buscador de ofertas de empleo cuando el usuario realice la postulación, entonces, se validará que el usuario tenga en el formulario hoja de vida.	Si
AH2-02	Dado el despliegue del formulario hoja de vida cuando el usuario registre su información, entonces, el formulario permitirá guardar, visualizar y actualizar los datos.	Si
	Dado el despliegue del formulario hoja de vida cuando el usuario registre su información, entonces, el formulario tendrá los siguientes campos datos personales, datos de contacto, preferencia salarial, objetivo laboral, educación, idiomas, experiencia laboral y referencias.	Si
	Dado el despliegue del formulario hoja de vida cuando el usuario ingrese sus datos en los campos educación, idiomas, experiencia laboral, entonces, se habilitará el botón agregar más ítems.	Si
	Dado el despliegue del formulario hoja de vida cuando el usuario llene los campos obligatorios, entonces, se habilitarán los botones de tipo submit.	Si
AH2-03	Dado el despliegue del formulario buscador de las ofertas de empleo seleccionadas cuando el usuario ingrese a este servicio, entonces, se visualizarán solo las ofertas de empleo que el usuario selecciono o postulo.	Si

	Dado el despliegue del formulario buscador de las ofertas de empleo cuando el usuario ingrese a este servicio, entonces, se habilitarán las opciones para filtrar datos.	Si

Incremento obtenido

En este sprint se implementó todo el módulo del usuario tipo estudiante – postulante, para ingresar a este módulo debe iniciar sesión y podrá visualizar la siguiente pantalla como se puede observar en la figura 33, el servicio de buscador de ofertas de empleo dispone de todas las publicaciones con estado activo. Se implementó el botón más información, que dispone de los detalles de la oferta de empleo seleccionada. También dispone de la opción para realizar la postulación, como se puede observar en la figura 34, la postulación permite realizar el primer contacto entre el usuario tipo estudiante – postulante con el usuario tipo empresa. Y por último dispone de un filtro para obtener un conjunto de datos que cumpla con el criterio seleccionado.

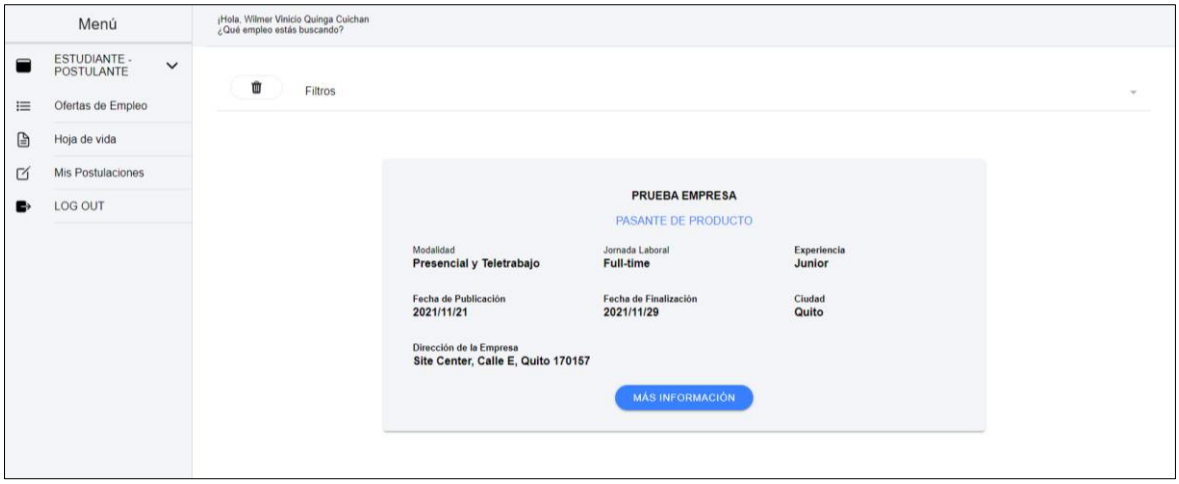


Figura 33. Pantalla de usuario - búsqueda de ofertas de empleo.

Figura 34. Botón de postulación.

Como se observa en la figura 35, el formulario hoja de vida dispone de los datos del usuario para que las empresas puedan seleccionar a un candidato a la oferta de empleo creada. El formulario cuenta con métodos para facilitar al usuario completar todos los campos requeridos por ejemplo calcular la edad del usuario, establecer datos en el comboBox cuidad, provincia, país y por último calcular los años, meses y días entre 2 fechas.

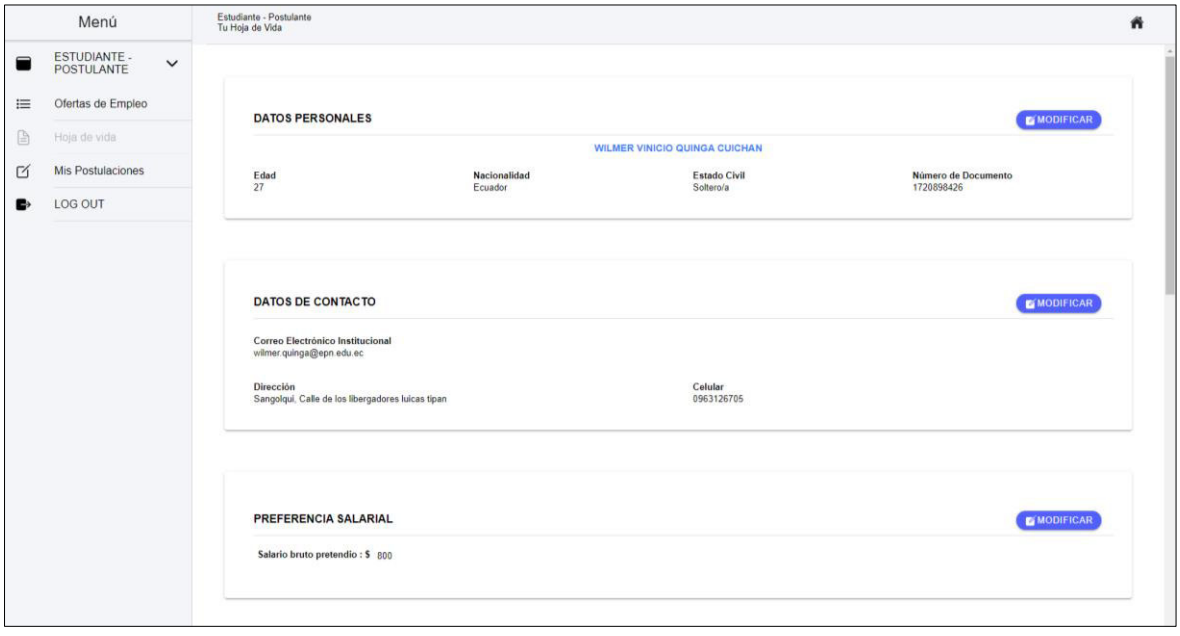


Figura 35. Pantalla de usuario - formulario con propiedades de hoja de vida.

El servicio de búsqueda de ofertas de empleo seleccionada obtiene todas las solicitudes realizadas por el usuario sin importar el estado. Esta información permite al usuario conocer en qué proceso se encuentra la oferta de empleo, como se puede observar en la figura 36.

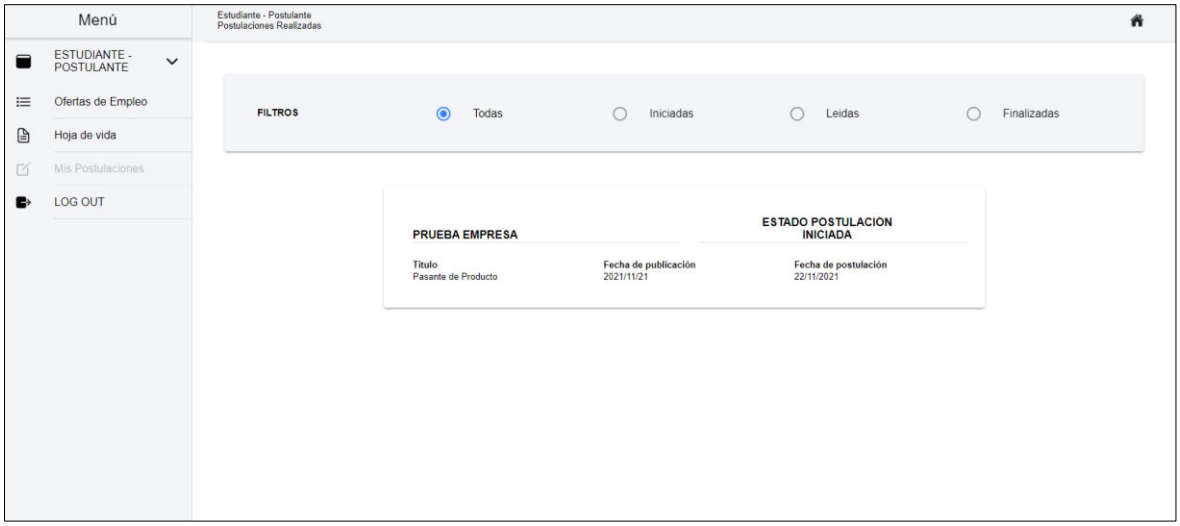


Figura 36. Interfaz de usuario - búsqueda de postulaciones realizadas.

2.9.4 Sprint Retrospective

¿Qué salió bien en la interacción?

Con el conocimiento obtenido en el primer sprint se realizó correctamente los formularios siguiendo el estándar planteado. Adicional, la detección de cambios configurada con el ciclo de vida de Ionic funciona correctamente y se estableció en todos los formularios creados.

¿Qué se puede mejorar?

La configuración responsive en dispositivos móviles no muestra en su totalidad las letras, se procede a investigar las propiedades de IONIC para resolver el problema.

2.10 Sprint 3

2.10.1 Sprint Planning

Objetivo del Sprint

- Implementar un formulario para registrar las ofertas de empleo
- Implementar un formulario para registrar la información de la empresa.

- Implementar servicio para listar todas las ofertas de empleo publicadas.
- Implementar un servicio para buscar los datos del usuario tipo estudiante - postulante.

Historias de Usuario

En la tabla 13 se listan las historias de usuarios que fueron creadas para el tercer sprint con su respectiva estimación de tiempo. En el anexo VI se encuentran con más detalle.

Tabla 10. Historias de usuario escogidas para el tercer sprint.

ID	Título	Estimación (días)	Estado
AH3-01	Buscador de ofertas de empleo publicadas por las empresas.	5	Por implementar
AH3-02	Registro de oferta de empleo.	3	Por implementar
AH3-03	Buscador de estudiantes, egresados o graduados.	5	Por implementar
AH3-04	Registro de información de la empresa.	2	Por implementar

Sprint Backlog

Tabla 11. Sprint backlog del tercer sprint.

SPRINT BACKLOG	
HISTORIA DE USUARIO	**TAREAS**

AH3-01	Crear página HTML para listar las ofertas de empleo publicadas.
	Implementar método para obtener todas las ofertas de empleo registradas.
	Implementar acciones en la tabla de editar y eliminar oferta de empleo, y visualizar a los usuarios postulados.
	Implementar métodos para seleccionar una oferta de empleo creada y editar la información.
	Crear un botón en el formulario crear postulación de empleo para finalizar oferta de empleo seleccionada.
	Crear formulario para obtener la información de la oferta de empleo creada y una lista de los usuarios que se postularon.
	Implementar métodos para obtener información de la oferta de empleo y de los usuarios postulados.
	Crear un botón para obtener la información del usuario.
	Crear método para obtener al usuario seleccionado y buscar sus datos.
	Implementar un modal con la información del usuario.
	Implementar un botón para imprimir en formato PDF la información del usuario.
	Realizar pruebas funcionales

	Arreglar errores encontrados
AH3-02	Implementar métodos en el servicio Job.
	Crear formulario crear oferta de empleo.
	Implementar textArea para permitir el ingreso de 2000 caracteres.
	Implementar método para obtener información de la empresa.
	Colocar el nombre de la empresa y dirección en los inputs correspondientes.
	Llenar a los comboBox con información respectiva.
	Implementar un método para obtener la fecha del día que se creó la oferta de empleo.
	Implementar CRUD con la información del formulario oferta de empleo en la entidad Job.
	Realizar pruebas funcionales.
	Validar información guardar en la base de datos.
	Arreglar errores encontrados.
	Implementar estilos en el formulario.
	Verificar información que se visualiza en el listado de ofertas laborales creadas.

AH3-03	Crear un buscador para obtener todos los usuarios registrados en el sistema.
	Implementar métodos para obtener información de los usuarios registrados.
	Seleccionar los datos del usuario que se visualizaran.
	Implementar filtros de búsqueda edad, lugar de residencia, pretensión salarial, género y educación.
	Crear métodos por cada filtro para obtener información.
	Implementar un botón para limpiar los filtros seleccionados.
	Implementar un botón para obtener más información del usuario
	Implementar un modal para visualizar la información del usuario.
	Implementar un botón para descargar la información en formato PDF.
	Realizar pruebas funcionales.
	Arreglar errores encontrados.
	Implementar estilos a los formularios.
	Crear formulario información de la empresa.
	Crear método para obtener información de la empresa.

AH3-04	Colocar los datos de la empresa en el input correspondiente.
	Con los datos del formulario de la empresa realizar CRUD con las entidades user y compamy.
	Realizar pruebas funcionales.
	Arreglar errores encontrados.
	Validar información registrada en la base de datos.
	Implementar estilos en el formulario.

2.10.2 Ejecución del Sprint

Servicio para listas las ofertas de empleo publicadas por cada empresa.

Para obtener los datos de las ofertas de empleo publicadas se configura el servicio del API RESTFul con la entidad job. En este caso se obtienen todos los datos sin importar el estado y se seleccionan los datos más importantes para su visualización como se puede, observar en la figura 37.

Figura 37. Pantalla de usuario – para listas ofertas de empleo publicadas.

La tabla de datos adicional dispone de 3 acciones:

- Editar: Permite modificar la oferta de empleo publicada y se habilitará un botón para finalizar el proceso de selección de nuevos empleados. El cual consiste en cambiar de estado a la oferta de empleo y el usuario de tipo estudiante – postulante no podrá visualizar la oferta.
- Visualizar. En la figura 38, se observa cómo se establece la ruta del formulario correspondiente que recibe el identificador de la oferta de empleo seleccionada. En la página post-notice-view se crea un método para obtener este identificador mediante un observable y realizar las consultas al backend, como se puede observar en la figura 39. Los datos que obtiene el método mencionado es la información de la oferta de empleo creada y un listado de todos los usuarios de tipo estudiante – postulante que participaron en la oferta de empleo, como se puede observar en la figura 40. También dispone de un botón para obtener el formulario hoja de vida y descargar esta información en formato PDF. Para este proceso se utilizó la biblioteca pdfMake es un complemento de generación de documentos PDF utilizando JavaScript [33].
- Eliminar. Esta opción borra la oferta de empleo creada mediante el servicio del API RESTful utilizando la propiedad DELETE como se puede observar en a figura 41, recibe el identificador de la oferta de empleo seleccionada.

Figura 38. Ruta configurada para recibir un identificador.

```
getIdJob() {
  const getIdPost = this.activatedRoute.params;
  this.obsGetIdPost = getIdPost
    .subscribe(
      next: (params : Params ) => {
        this.idPostNotice = Number(params.id);
        this.sendPageService.sendPostVIew(this.idPostNotice);
        const getJob = this.jobService.searchIdJob(params.id);
        this.obsGetJob = getJob
          .subscribe(
            next: (data : Object ) => {
              this.dataJobId = data as any;
              for (const i in this.dataJobId) {
                if (this.dataJobId.hasOwnProperty(i)) {
                  for (const l in this.dataJobId[i].idMyapplication) {
                    if (this.dataJobId[i].idMyapplication.hasOwnProperty(l)) {
                      this.getIdApplicant(this.dataJobId[i].idMyapplication[l].applicantIdApplicant);
                    }
                  }
                }
              }
            },
            error: (error) => {
              console.log(error);
            }
          );
      }
    );
}
```

Figura 39. Método para obtener el identificador enviado y obtener datos del backend.

Figura 40. Interfaz de usuario – postulaciones recibidas.

```
deleteJobId(idJob: number) {
  return this.httpClient.delete( url: URL + '/job/' + idJob);
}
```

Figura 41. API REST ful configurada para eliminar un registro de la entidad job.

Formulario para crear ofertas de empleo

La información requerida es validada por el formulario reactivo, como se puede observar en la figura 42. Si no se cumple todas las validaciones no se activará el botón crear aviso como se puede observar en la figura 43.

```
inputFormNoticePost = new FormGroup( controls: {
  titleJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  descriptionJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  countryJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  provinceJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  provinceJobsOthers: new FormControl( formState: ''),
  cantonJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  cantonJobsOthers: new FormControl( formState: ''),
  directionJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  hierarchyJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  typeJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  nameCompanyJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  modalityJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
  vacanciesJobs: new FormControl( formState: '',  validatorOrOpts: [Validators.required, CreateValidator.validOnlyNumbers]),
  dateFinPost: new FormControl( formState: '',  validatorOrOpts: [Validators.required])
});
```

Figura 42.Validación del formulario reactivo.

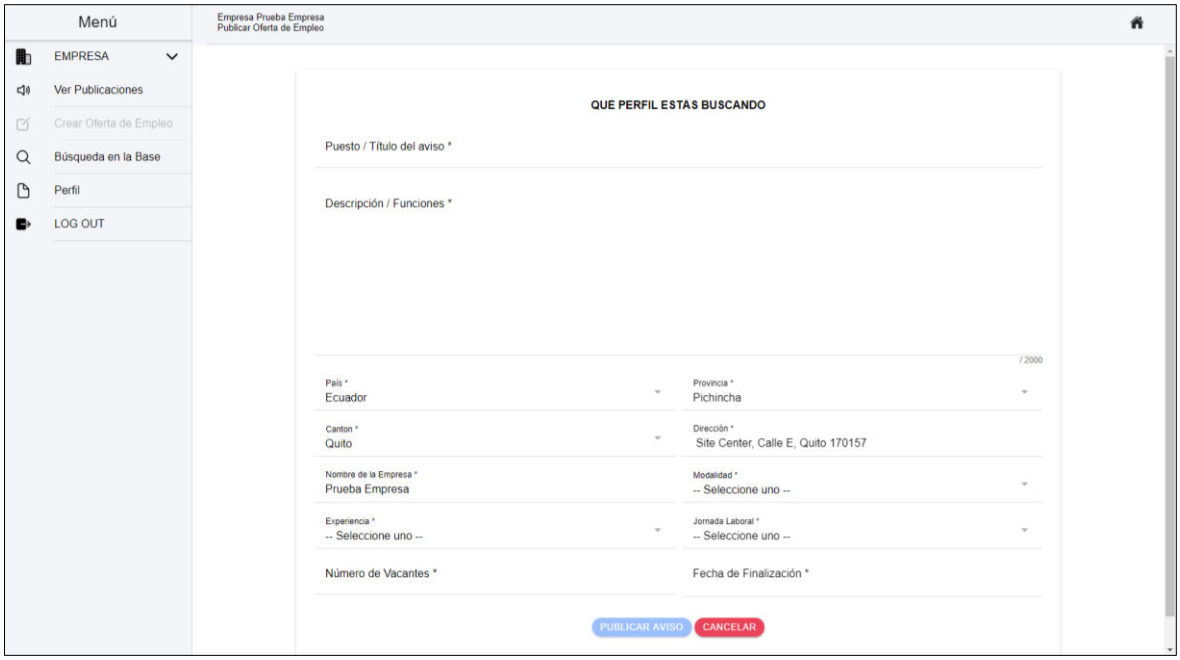


Figura 43. Pantalla de usuario – para crear ofertas de empleo.

Buscador de usuarios tipo estudiante – postulante

Esta funcionalidad se creó para que la empresa tenga disponible los datos de todos los usuarios tipo estudiante – postulantes registrados en el sistema. Para obtener esta información se configura el API RESTful con la propiedad GET para obtener todos los registros, como se puede observar en la figura 44.

Figura 44. Interfaz de usuario – buscado de usuarios estudiante – postulante.

La opción de filtros dispone de diversas categorías, como se observa en la figura 45, así se puede obtener un subconjunto de datos. Para este proceso se consume el servicio del Api RESTfull con la propiedad de query params.

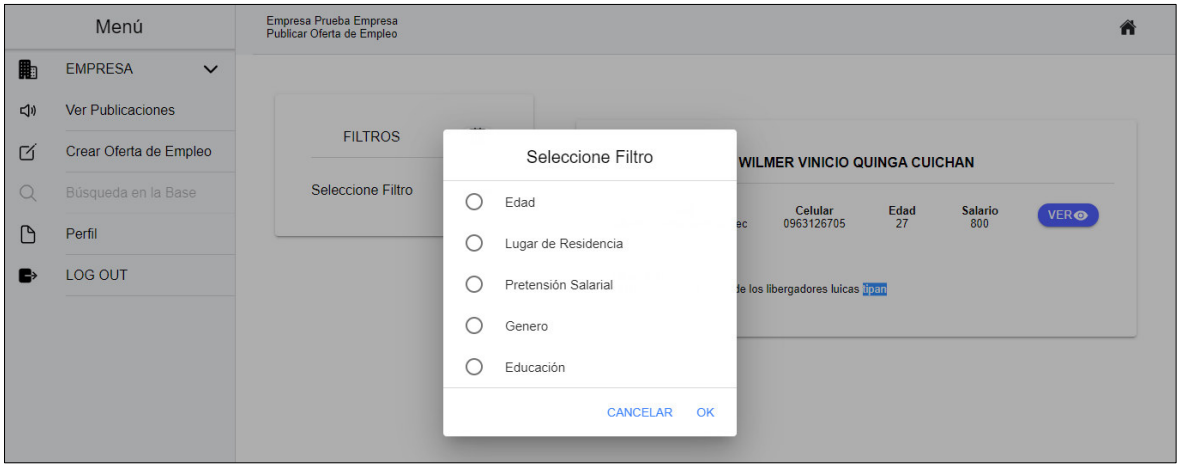


Figura 45. Pantalla de usuario – buscado de usuarios estudiante – postulante.

El formulario dispone de un botón para obtener toda la información del usuario seleccionado, como se puede observar en la figura 46. Para obtener estos datos el modal recibe el identificador del usuario y el nombre de la página que renderiza dicho modal, como se pueden observar en la figura 47. La página mencionada utiliza 8 componentes para presentar los datos del formulario hoja de vida, como se puede observar en la figura 48.

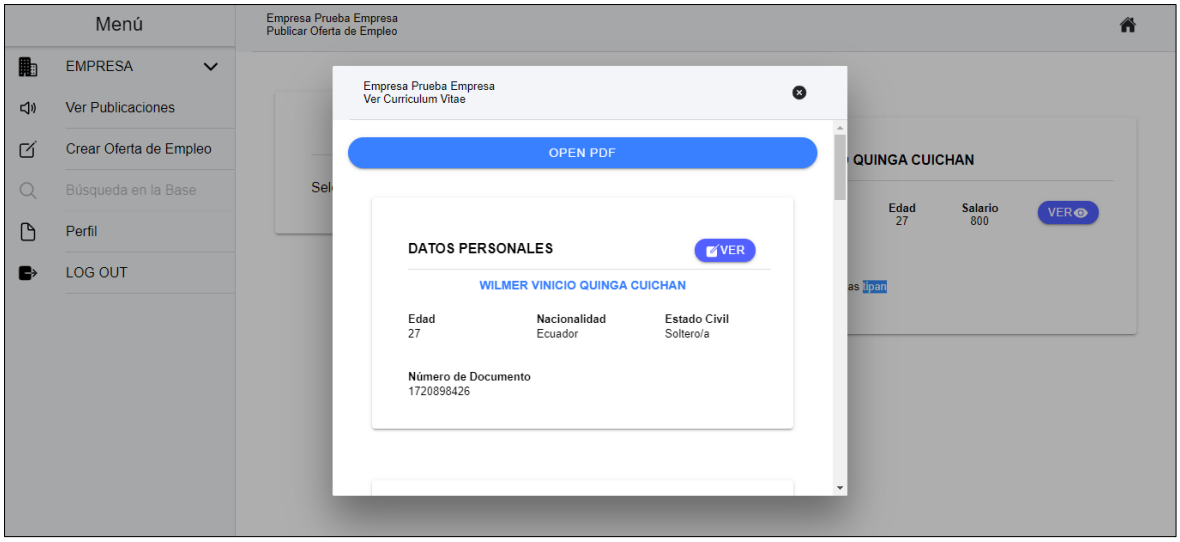


Figura 46. Pantalla de usuario – buscado de usuarios estudiante – postulante, información del usuario.

```
async presentModal(idUser) {
  const modal = await this.modalController.create({
    component: ViewProfilePostulantePage,
    cssClass: 'my-custom-class',
    componentProps: {
      idUserInput: idUser
    }
  });
  return await modal.present();
}
```

Figura 47. Método para abrir el modal en Ionic.

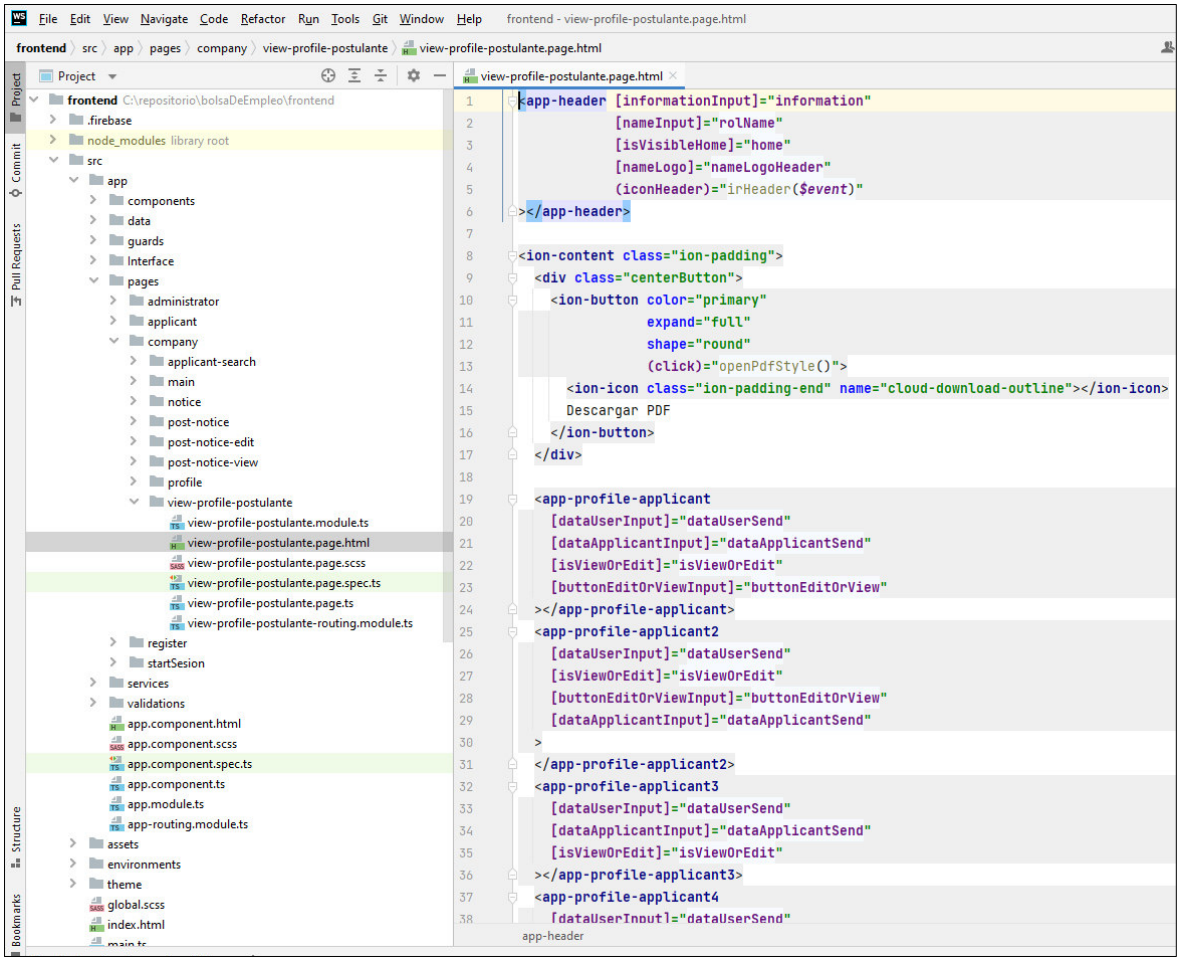


Figura 48. Página view-profile-postulante utiliza a los componentes que crear al formulario hoja de vida.

2.10.3 Sprint Review

Pruebas de aceptación

En la Tabla 15 se listan los criterios de aceptación cumplidos para el tercer sprint.

Tabla 12. Pruebas de aceptación del Sprint 3.

Historia de Usuario	Criterios de aceptación	Cumplido
	Dado el despliegue del formulario para obtener todas las ofertas de empleo publicadas cuando el usuario seleccione este formulario, entonces, se visualiza todas las ofertas de empleo sin importar su estado.	Si

AH3-01	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario guarde la información en el backend, entonces, se habilita la opción para actualizar los datos.	Si
	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario seleccione la opción de editar, entonces, se habilita la opción de finalizar oferta de empleo.	Si
	Dado el despliegue del formulario para obtener todas las ofertas de empleo creadas cuando el usuario seleccione una oferta, entonces, se habilita la opción para revisar que usuarios tipo estudiantes – postulantes optaron por seleccionar esta oferta de empleo.	Si
	Dado el despliegue del formulario para obtener todas las ofertas de empleo creadas cuando el usuario tipo empresa seleccione una oferta de empleo publicada y escoja a un usuario tipo estudiante – postulante que postulo, entonces, el sistema permite obtener todos los datos del usuario tipo estudiante - postulante y descargar la información en formato PDF.	Si
AH3-02	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario ingrese todos los campos obligatorios, entonces, se habilitan el botón de publicar aviso.	Si
	Dado el despliegue del formulario para registrar una oferta de empleo cuando llene los campos solicitados, entonces, el campo llamado descripción de funciones permite el ingreso de 2000 caracteres.	Si

	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario llene todos los campos requeridos, entonces, al momento de guardar los datos en el backend se incluye la fecha que realizo el registro.	Si
	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario llene los datos solicitados, entonces, en los camboBox de país, provincia y cantón se llena en cascada.	Si
AH3-03	Dado el despliegue del formulario para buscar a los usuarios estudiantes – postulantes cuando el usuario ocupe este servicio, entonces, se listan todos los datos registrados en el backend pertenecientes a los estudiantes – postulantes.	Si
	Dado el despliegue del formulario para buscar al usuario tipo estudiante – postulante cuando el usuario ocupe el servicio, entonces, se habilita la opción de filtrar datos por edad, lugar de residencia, pretensión salarial, género y educación.	Si
	Dado el despliegue del formulario para buscar al usuario tipo estudiantes – postulantes cuando la empresa seleccione a un estudiante – postulante, entonces, se abre un modal con toda la información del usuario seleccionado.	Si
	Dado el despliegue del formulario para buscar al usuario tipo estudiante – postulante cuando la empresa seleccione a un estudiante – postulante, entonces, puede descargar la información del usuario seleccionado en formato PDF.	Si
	Dado el despliegue del formulario para obtener datos de la empresa cuando el usuario ocupe este servicio, entonces, se visualiza la información registrada de la empresa.	Si

AH3-04	Dado el despliegue del formulario para obtener datos de la empresa cuando exista algún cambio en la información de la empresa, entonces, el sistema permite actualizar los datos deseados.	Si
	Dado el despliegue del formulario para obtener datos de la empresa cuando el usuario ingrese los datos obligatorios, entonces, se habilita el botón para actualizar la información.	Si

Incremento obtenido

En el tercer sprint se completó el módulo del usuario tipo empresa, la primera pantalla que se visualiza es la lista de las ofertas de empleo publicadas estas disponen de los servicios para actualizar y eliminar además de una opción para visualizar a los usuarios de tipo estudiante – postulante que realizaron la postulación y obtener sus datos, como se observar en la figura 49.

Figura 49. Pantalla de usuario para listar ofertas de empleo.

El siguiente servicio es el formulario para registrar la oferta de empleo aquí las empresas pueden crear sus solicitudes las veces que deseen. Este formulario tiene validaciones que se deben cumplir para realizar la publicación de la oferta de empleo, las cuales se visualizan en el módulo del usuario tipo estudiante – postulante, como se observa en la figura 50.

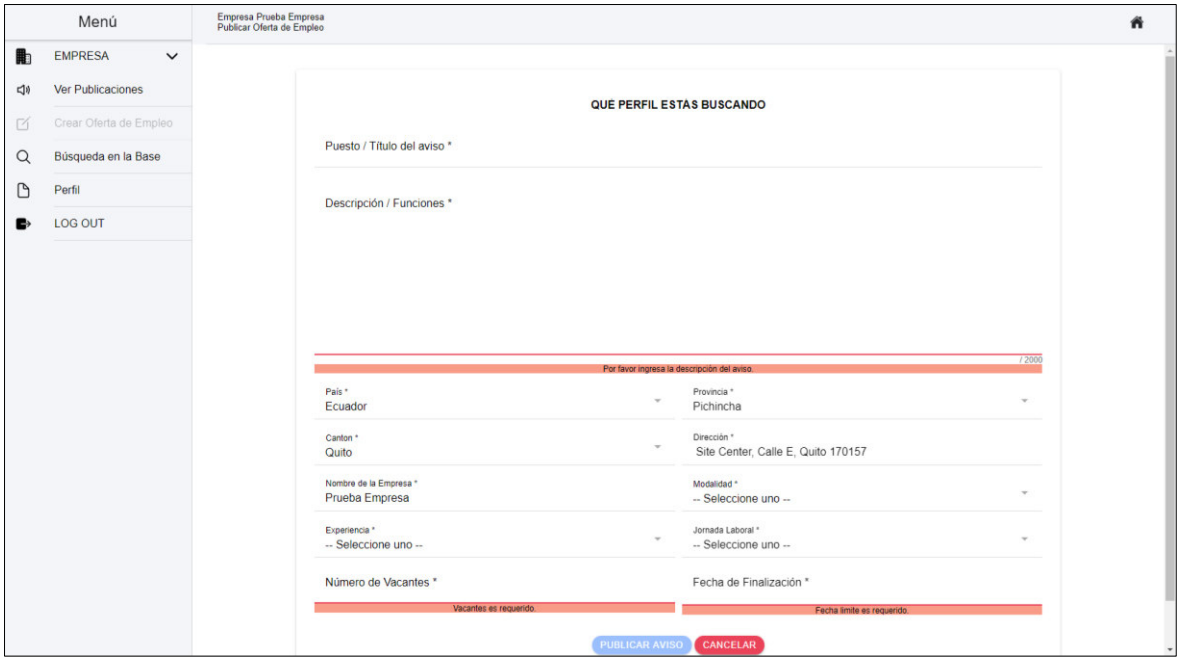


Figura 50. Pantalla de usuario – formulario de registro de oferta de empleo.

El servicio de búsqueda de usuario tipo estudiante – postulante es creado para que la empresa posea la información de todos los usuarios registrados. Eliminando el proceso de publicar ofertas de empleo, así la empresa ya posee la información de todos los usuarios registrados y con los filtros disponibles puede seleccionar a los candidatos ideales, como se puede observar en la figura 51.

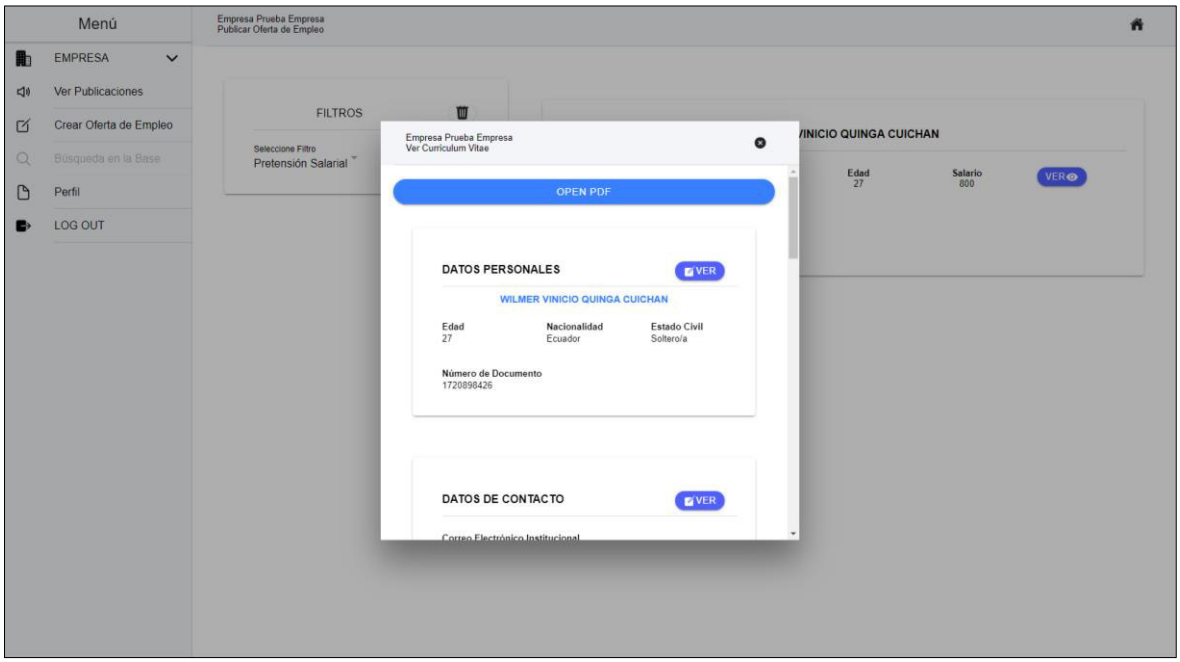


Figura 51. Pantalla de usuario – buscador de usuarios estudiante - postulante.

El último servicio es el formulario de empresa donde se puede editar sus datos esta información se utiliza para crear las ofertas de empleo, como se puede observar en la figura 52.

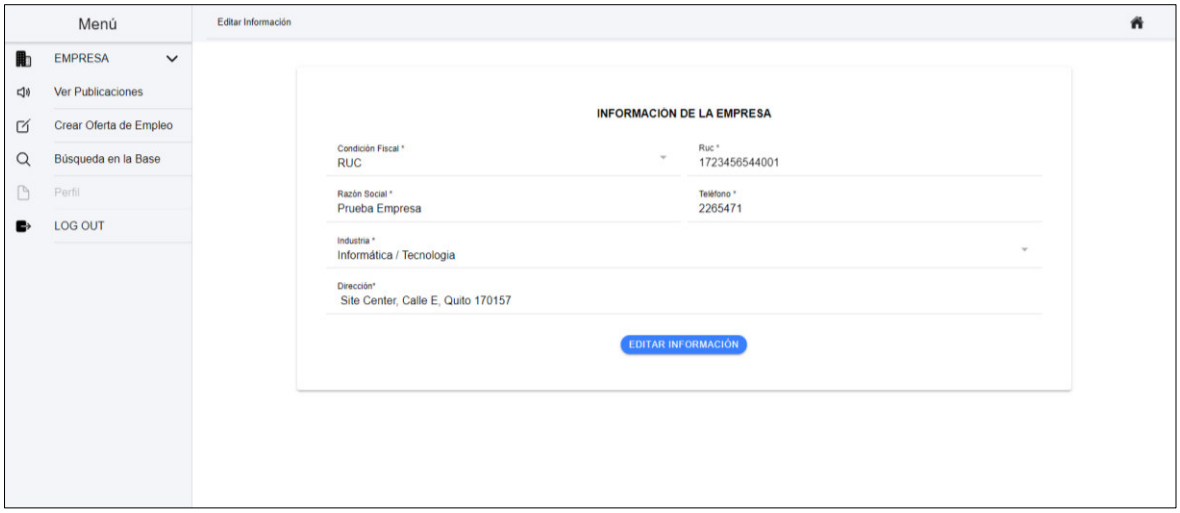


Figura 52. Pantalla de usuario – formulario registro de información de empresa.

2.10.4 Sprint Retrospective

¿Qué salió bien en la interacción?

En este sprint con los conocimientos obtenidos se pudo crear sin ningún problema los formularios con todas sus características y validaciones. Lo más importante fue la solución para obtener los datos del usuario tipo estudiante – postulante en el rol de usuario empresa.

¿Qué se puede mejorar?

La librería para descargar el formulario en formato PDF genera problemas al obtener datos de un array en este caso solo obtiene el último registro y no visualiza en todos los datos registrados. Se investiga más información sobre la librería pdfmake.

2.11 Sprint 4

2.11.1 Sprint Planning

Objetivo del Sprint

- Crear formulario para administrar datos de los usuarios registrados.
- Crear lógica para crear, actualizar, eliminar roles.

Historias de Usuario

En la tabla 16 se listan las historias de usuarios que fueron creadas para el cuarto sprint con su respectiva estimación de tiempo. En el anexo V se encuentran con más detalle.

Tabla 13. Historias de usuario escogidas para el cuarto sprint.

ID	Título	Estimación (días)	Estado
AH4-01	Registro de estudiantes, egresados o graduados que pertenezcan a la facultad de ingeniería en sistemas de la Escuela Politécnica Nacional.	4	Por implementar
AH4-02	Visualizar a los usuarios estudiantes – postulantes registrados en el sistema.	3	Por implementar
AH4-03	Visualizar a los usuarios empresas registrados en el sistema.	4	Por implementar
AH4-04	Registro de datos de roles de usuario.	4	Por implementar

Sprint Backlog

Tabla 14. Sprint backlog del cuarto sprint.

SPRINT BACKLOG	
HISTORIA DE USUARIO	**TAREAS**
	Crear menú lateral con enlace a los servicios que ofrece el menú administrador.

AH4-01	Configurar propiedad en el menú para que solo se visualice con el rol administrador.
	Crear método para capturar a todos los registros en la tabla validatestudents.
	Crear formulario para obtener los datos solicitados.
	Crear una tabla en el formulario creado y establecer los datos que se visualizan.
	Crear método para obtener los datos solicitudes por medio del API RESfult.
	Crear método para guardar la información del nuevo registro.
	Reutilizar formulario de validación de usuario tipo estudiante – postulante.
	Crear método para guardar la información del nuevo registro por medio del API RESfult.
	Realizar pruebas funcionales.
	Validar información guardar en la base de datos.
	Reparar errores encontrados.
	Implementar estilos en formularios.
	Añadir opción al menú del módulo.

AH4-02	Crear método para capturar a todos los registros en la tabla user. Crear lógica para solo obtener datos del usuario tipo estudiante – postulante.
	Crear formulario para obtener los datos solicitados.
	Crear una tabla en el formulario creado y establecer los datos que se visualizan.
	Crear método para obtener los datos solicitudes por medio del API RESfult.
	Realizar pruebas funcionales.
	Validar información obtenida de la base de datos.
	Reparar errores encontrados.
	Implementar estilos en formularios.
AH4-03	Crear menú lateral con enlace a los servicios que ofrece el menú administrador.
	Crear método para capturar a todos los registros en la tabla user. Crear lógica para solo obtener datos del usuario tipo tiempresa.
	Crear formulario para obtener los datos solicitados.
	Crear una tabla en el formulario creado y establecer los datos que se visualizan.

	Crear método para obtener los datos solicitudes por medio del API RESfult.
	Realizar pruebas funcionales.
	Validar información obtenida de la base de datos.
	Reparar errores encontrados.
	Implementar estilos en formularios.
AH4-04	Crear menú lateral con enlace a los servicios que ofrece el menú administrador.
	Configurar propiedad en el menú para que solo se visualice con el rol administrador.
	Crear método para capturar a todos los registros en la tabla roles.
	Crear formulario para obtener los datos solicitados.
	Crear una tabla en el formulario creado.
	Crear método para obtener los datos solicitudes por medio del API RESfult.
	Crear métodos para crear, editar, eliminar la información del nuevo registro.
	Crear formulario para obtener datos del nuevo registro.

	Crear método para guardar la información del nuevo registro por medio del API RESfult.
	Realizar pruebas funcionales.
	Validar información guardar en la base de datos.
	Reparar errores encontrados.
	Implementar estilos en formularios.

2.11.2 Ejecución Del Sprint

Menú

Cada módulo tiene un menú para seleccionar servicios. Se visualiza dependiendo del rol que se consulta en el componente principal. Al iniciar sesión se guarda en el local storage el id del rol, como se puede observar en la figura 53. Este id se puede consultar en cualquier parte del sistema.

```
if (this.getRoles.typeRoles === 'Estudiante - Postulante') {
  userObj = {
    email: dataUser.emailUser,
    rol: idRoles,
    idFireBase: idFirebase
  };
  this.authenticationService.saveStorage(userObj);
  this.router.navigateByUrl( url: '/applicant/main',  extras: {replaceUrl: true}).then(r => r);
```

Figura 53. Información que se guardar en el local storage.

En la página principal se crea el método asincrónico [34] que consume el servicio para obtener los datos del usuario, como se puede observar en la figura 54. Con el id el rol se realiza una búsqueda en el backend para obtener el nombre del rol seleccionado, como se puede observar en la figura 55. Primero se realiza una búsqueda para validar que existan roles creados y después se busca el id seleccionado.

```
getDataUser() {
  const getUser = this.authenticationService.getUser();
  this.obsGetUser = getUser
    .subscribe( next: (data) => {
        if (data) {
          this.email = data.email;
          this.searchRoles(data.rol);
        }
    });
}
```

Figura 54. Método asincrónico para obtener información del usuario que inicia sesión.

```
searchRoles(idRoles: number) {
  const getAllRoles = this.rolesServices.listAllRoles();
  this.obsGetAllRoles = getAllRoles
    .subscribe(
      next: (data : Object ) => {
        this.gerAllRoles = data as any;
        if (this.gerAllRoles.length > 0) {
          const searchRoles = this.rolesServices.listIdRoles(idRoles);
          this.obsSearchRoles = searchRoles
            .subscribe(
              next: (dataRoles : Object ) => {
                this.getRoles = dataRoles as any;
                if (this.getRoles.typeRoles === 'Estudiante - Postulante') {
                  this.router.navigateByUrl( url: '/applicant/main-applicant', extras: {replaceUrl: true}).then(r => r);
                } else if (this.getRoles.typeRoles === 'Empresa') {
                  this.router.navigateByUrl( url: '/company/notice', extras: {replaceUrl: true}).then(r => r);
                } else if (this.getRoles.typeRoles === 'Administrador') {
                  console.log('admin');
                }
              },
              error: (error) => {
                console.log(error);
              }
            );
        } else {
          this.router.navigateByUrl( url: '/login', extras: {replaceUrl: true}).then(r => r);
        }
      }
    );
}
```

Figura 55. Método asincrónico para obtener el nombre del rol.

Datos de usuarios estudiante - postulante activos

Para crear el formulario se diseña un método para obtener los datos de la tabla validatestudents por medio de un API RESTfult. Si no existen datos en esta tabla no se podrá registrar ningún usuario tipo estudiante – postulante. Esto sirve para validar que solo los usuarios que pertenezcan a la Facultad de Ingeniería en sistema de la Escuela Politécnica Nacional puedan utilizar el servicio de búsqueda de empleo.

Se crea un componente con el diseño HTML de la tabla para visualizar la información, los datos más importantes se seleccionan para la presentación. Se consume el componente mencionado en una página y por medio de una consulta de datos al backend se obtienen la información y es enviada a la tabla para su visualización. Como se puede observar en la figura 56, el método consume el servicio creado listAllActiveStudents() para obtener todos los datos del backend por medio de un observable [18] que maneja operaciones asincrónicas.

```
listStudentsValidate() {
  const observableBringAll = this.validateStudents.listAllActiveStudents();
  this.observableBringAllSub = observableBringAll
    .subscribe(
      next: (data : Object ) => {
        this.studentValidate = data as any;
      },
      error: (error) => {
        console.log(error);
      }
    );
}
```

Figura 56. Método asincrónico para obtener datos de la tabla validatestudents.

Registro de roles

Se crea 2 componentes, el primero se crea para establecer una tabla HTML, aquí se visualizan los datos, en el siguiente componente se crea un formulario para obtener la información del nuevo rol. La tabla creada permitirá seleccionar un registro y ejecutar las acciones de eliminar o actualizar su información. Como se puede observar en la figura 57 se establece un método para eliminar el dato seleccionado por medio de su id.

```
rolesDelete(id) {
  this.alertMessage.messageAlertConfirm( title: '¡ALERTA!',  messageData: '¿Desea eliminar?').then(
    value => {
      if (value === 'OK') {
        const deleteData = this.rolesServices.deleteRolesId(id);
        this.obsDelete = deleteData
          .subscribe(
            next: () => {
              const index = this.rolesArrayDelete.findIndex(u => u.id === id);
              this.rolesArrayDelete.splice(index, 1);
              this.listRolesRegister();
            },
            error: (error) => {
              console.error(error);
            }
          );
      }
    }
  );
}
```

Figura 57. Método asincrónico para eliminar información.

Para actualizar la información primero se obtiene el id seleccionado por medio de la clase params que pertenece a angular. Esta clase obtiene el id que viene en la ruta con la ayuda de un observable. Con el id se realiza una búsqueda en la tabla roles para obtener la información del rol seleccionado, como se puede observar en la figura 58.

```
getIdJob() {
  const getIdPost = this.activatedRoute.params;
  this.obsGetIdPost = getIdPost
    .subscribe(
      next: (params : Params ) => {
        this.idRoles = Number(params.id);
        const getJob = this.rolesServices.searchRolesId(this.idRoles);
        this.obsGetJob = getJob
          .subscribe(
            next: (data : Object ) => {
              this.getRoles = data as any;
            },
            error: (error) => {
              console.log(error);
            }
          );
      }
    );
}
```

Figura 58. Método asincrónico para obtener el id y realizar una búsqueda en el backend.

Al final. Como se puede observar en la figura 59, se realiza la actualización de los datos por medio de servicio editar roles que recibe el id del registro para actualizar y la nueva información.

```
editDataRoles(id, roles) {
  const editRoles = this.rolesServices.editRolId(id, roles);
  this.obsEditRoles = editRoles
    .subscribe(
      next: () => {
        this.alertMessage.presentAlertConfirm( title: 'Existo!!', messageData: 'Datos editados correctamente')
          .then(r => r);
        const url = ['/administrator', 'list-roles'];
        this.router.navigate(url).then(r => r);
      },
      error: (error) => {
        console.log(error);
      }
    );
}
```

Figura 59. Método asincrónico para actualizar información.

2.11.3 Sprint Review

Pruebas de aceptación

En la Tabla 18 se listan los criterios de aceptación cumplidos para el cuarto sprint.

Tabla 15. Pruebas de aceptación del cuarto sprint.

Historia de Usuario	Criterios de aceptación	Cumplido
AH4-01	Dado el despliegue del formulario para listas los datos de los usuarios activos cuando el usuario ocupe este servicio, entonces, se listan los datos de todos los usuarios que se podrán registrar como estudiante – postulante.	Si
	Dado el despliegue del formulario para listar los datos de los usuarios activos cuando el usuario ocupe este servicio, entonces, podrán registrar otros usuarios.	Si

AH4-02	Dado el despliegue del formulario para listar a los estudiantes – postulantes cuando el usuario ingrese a este servicio, entonces, se listan todos los usuarios de tipo estudiante - postulante registrados en el sistema.	Si
	Dado el despliegue del formulario para listar a los estudiantes – postulantes cuando el usuario ocupe este servicio en otros dispositivos con diferentes tamaños de pantalla, entonces, el formulario se adapta a cualquier tamaño de pantalla.	Si
AH4-03	Dado el despliegue del formulario para listar a las empresas cuando el usuario ingrese a este servicio, entonces, se listan a todos los usuarios de tipo empresa registrados en el sistema.	Si
	Dado el despliegue del formulario para listar a las empresas cuando el usuario ocupe este servicio en otros dispositivos con diferentes tamaños de pantalla, entonces, el formulario se adapta a cualquier tamaño de pantalla.	Si
AH4-04	Dado el despliegue del formulario para listar a los roles cuando el usuario ingrese a este servicio, entonces, se listan a todos roles registrados en el sistema.	Si
	Dado el despliegue del formulario para listar a los roles cuando el usuario seleccione a un rol, entonces, se habilitan las opciones para actualizar y eliminar el registro.	Si
	Dado el despliegue del formulario para listar a los roles cuando el usuario ocupe este servicio en otros dispositivos con diferentes tamaños de pantalla, entonces, el formulario se adapta a cualquier tamaña de pantalla	Si

Incremento obtenido

En este sprint se creó todo el módulo de administrador que cuenta con los servicios que se observar en la figura 60.

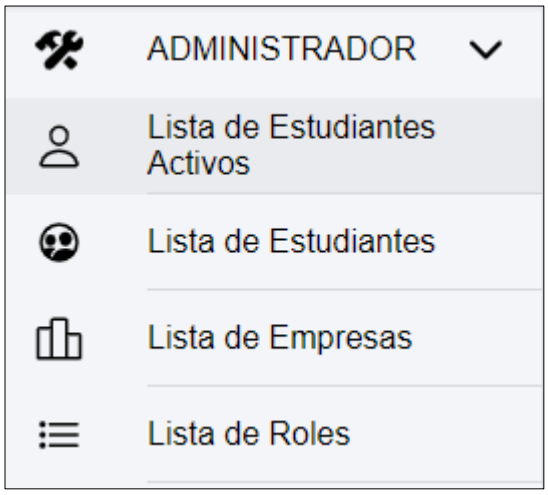


Figura 60. menú del módulo administrador.

En la figura 61, se observa el formulario, lista de estudiantes activos, el cual muestra la información de todos los usuarios que se pueden registrar como usuario de tipo estudiante – postulante. Además, es posible crear nuevos estudiantes a ser considerados como activos.

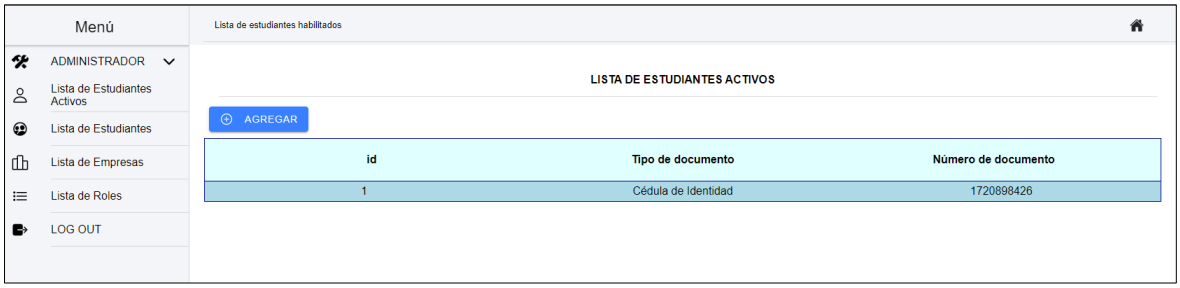


Figura 61. Pantalla para obtener los usuarios que se pueden registrar como estudiante - postulante.

El servicio de listar a los usuarios de tipo estudiantes – postulantes proporciona la información de todos los usuarios registrados en el sistema, esto sirve para llevar un control de los datos, como se observa en la figura 62.

Figura 62. Pantalla para obtener los usuarios estudiantes – postulantes registrados.

El servicio de listar a los usuarios tipo empresas suministra la información de todas las empresas registradas, esto sirve para llevar un control de los datos, como se observa en la figura 63.

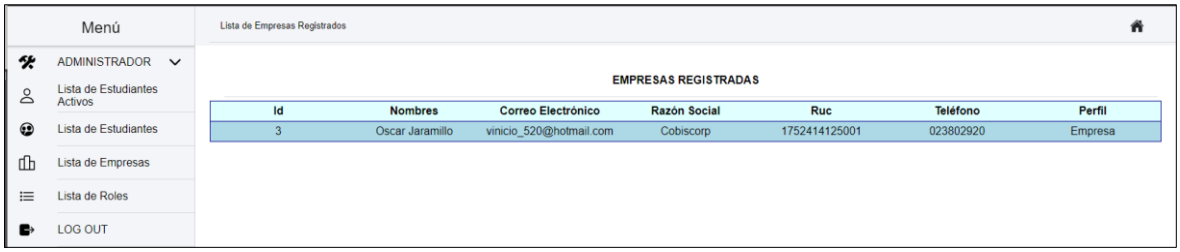


Figura 63. Pantalla para obtener los usuarios empresas registrados.

La figura 64, muestra la lista de roles que gestionan el sistema. Es posible crear nuevos roles o borrar roles.

Figura 64. Pantalla para obtener los roles creados.

2.11.4 Sprint Retrospective

¿Qué salió bien en la interacción?

En este sprint con los conocimientos obtenidos se pudo crear sin ningún problema los formularios con todas sus características y validaciones. Lo más importante fue crear los métodos para obtener información de la base de datos, se refactorizo el código para su correcto funcionamiento.

¿Qué se puede mejorar?

Lo que se puede mejorar en este sprint es el estilo responsive de las tablas creadas.

2.12 Sprint 5

2.12.1 Sprint Planning

Objetivo del Sprint

- Implementar al sistema características de una aplicación web progresiva (PWA)

Historias de Usuario

En la tabla 19 se lista la historia de usuario que fue receptada para el quinto sprint con su respectiva estimación de tiempo. En el anexo VI se encuentra con más detalle.

Tabla 16. Historias de usuario escogidas para el quinto sprint.

ID	Título	Estimación (días)	Estado
AH5-01	Aplicación web progresiva (PWA)	15	Por implementar

Sprint Backlog

Tabla 17. Sprint backlog del quinto sprint.

SPRINT BACKLOG	
HISTORIA DE USUARIO	**TAREAS**
	Investigar como convertir una aplicación de Ionic en PWA.
	Ejecutar el comando en consola para convertir la aplicación en una PWA.
	Instalar complementos de firebase para realizar deploying.

AH5-01	Ejecutar comandos en consola para construir la aplicación y encontrar errores en la programación.
	Reparar errores encontrados.
	Configurar hosting de firebase.
	Realizar el deploy de la aplicación.
	Realizar pruebas funcionales.

2.12.2 Ejecución Del Sprint

La información requerida se consultó de la documentación de Ionic [35].

1. Instalar el paquete de @angular/pwa que genera un service worker y un manifest archivos requeridos para el funcionamiento de una PWA, como se puede observar en las figuras 65 y 66.

```
PS C:\repositorio\bolsaDeEmpleo\frontend> ng add @angular/pwa
           @angular/pwa
Would you like to proceed? Yes
√ Package successfully installed.
CREATE ngsw-config.json (631 bytes)
CREATE src/manifest.webmanifest (1330 bytes)
UPDATE angular.json (5937 bytes)
UPDATE src/index.html (1338 bytes)
√ Packages installed successfully.
```

Figura 65. Proceso de instalación de paquete de pwa.

manifest.webmanifest | ngsw-config.json

```
{
  "$schema": "./node_modules/@angular/service-worker/config/schema.json",
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "empleo",
      "installMode": "prefetch",
      "resources": {
        "files": [
          "/favicon.ico",
          "/index.html",
          "/manifest.webmanifest",
          "/*.css",
          "/*.js"
        ]
      }
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [
          "/assets/**",
          "/*.(svg|cur|jpg|jpeg|png|apng|webp|avif|gif|otf|ttf|woff|woff2)"
        ]
      }
    }
  ]
}
```

Figura 66. Archivos de service worker y manifest.

2. Ejecutar el comando ionic build –prod –service-worker para verificar si existen errores en la programación. Como se puede observar en la figura 67 el comando no genero ningún error. Adicional se crea una carpeta www que contiene el build de la aplicación, como se puede observar en las figuras 68.

```
Terminal: Local | Local (2)
node_modules_ionic_core_dist_esm_ion-tab_2_entry_js.js                                  | -  |  9.62 kB
node_modules_ionic_core_dist_esm_ion-split-pane_entry_js.js                             | -  |  9.36 kB
src_app_pages_startSesion_verify-email_verify-email_module_ts.js                        | -  |  8.09 kB
node_modules_ionic_core_dist_esm_ion-spinner_entry_js.js                                | -  |  7.89 kB
src_app_pages_startSesion_recover-password_recover-password_module_ts-src_app_services_http_u-6d0ff8.js | -  |  6.86 kB
src_app_pages_administrator_main_main_module_ts.js                                      | -  |  6.81 kB
node_modules_ionic_core_dist_esm_ion-ripple-effect_entry_js.js                          | -  |  6.59 kB
node_modules_ionic_core_dist_esm_ion-avatar_3_entry_js.js                               | -  |  5.78 kB
node_modules_ionic_core_dist_esm_tap-click-cc1ae2b2_js.js                               | -  |  5.58 kB
node_modules_ionic_core_dist_esm_ion-backdrop_entry_js.js                               | -  |  3.42 kB
node_modules_ionic_core_dist_esm_swipe-back-fae97365_js.js                              | -  |  2.75 kB
node_modules_ionic_core_dist_esm_status-tap-5c3a5bca_js.js                              | -  |  2.51 kB
src_app_pages_applicant_post-notice-modal_post-notice-modal_module_ts.js                | -  |  2.39 kB
node_modules_ionic_core_dist_esm_focus-visible-f4ad4f1a_js.js                           | -  |  2.00 kB
node_modules_ionic_core_dist_esm_ion-text_entry_js.js                                   | -  |  1.72 kB

Build at:                        - Hash:                    - Time:      ms

Warning: C:\repositorio\bolsaDeEmpleo\frontend\src\app\pages\company\view-profile-postulante\view-profile-postulante.page.ts depends on 'pdfmake/build/pdfmake'. CommonJS or AMD dependencies can cause optimization bailouts.
For more info see: https://angular.io/guide/build#configuring-commonjs-dependencies

Warning: C:\repositorio\bolsaDeEmpleo\frontend\src\app\pages\register\company-user\company-user.page.ts depends on 'crypto-js'. CommonJS or AMD dependencies can cause optimization bailouts.
```

Figura 67. Proceso de build sin errores.

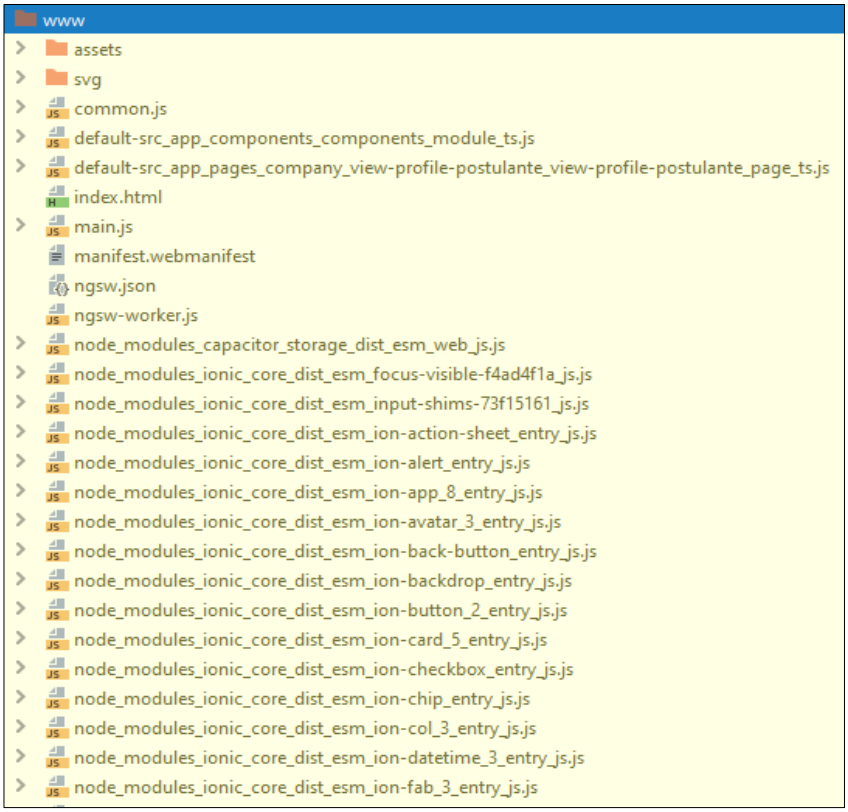


Figura 68. Carpeta con el build de la aplicación.

3. Iniciar firebase para configurar el hosting. Como se puede observar en la figura 69, se selecciona el servicio de hosting de firebase. Al terminar se obtiene el mensaje de éxito, como se puede observar en la figura 70. Y al final se crea un archivo firebase.json donde se configura la header, como se pude observa en la figura 71.

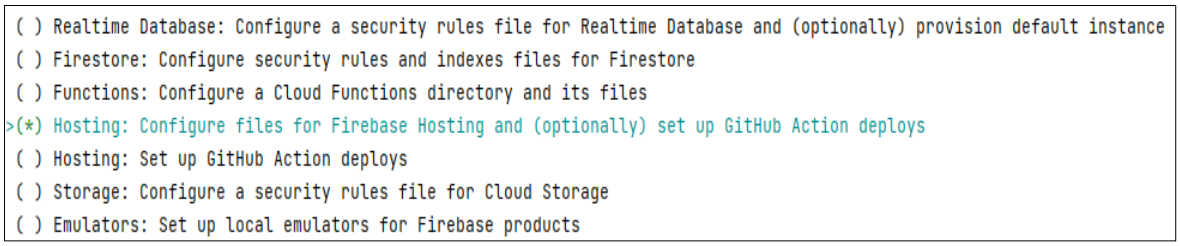


Figura 69. Proceso de configuración de hosting.

```
First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

i  Using project authenticationuser-18301 (authenticationUser)

? Set up automatic builds and deploys with GitHub? No
+  Wrote www/404.html
? File www/index.html already exists. Overwrite? No
i  Skipping write of www/index.html

i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...

+  Firebase initialization complete!
PS C:\repositorio\bolsaDeEmpleo\frontend>
```

Figura 70. Proceso de finalización para la configuración del hosting.

```
{
  "hosting": {
    "public": "www",
    "ignore": ["firebase.json", "**/.*", "**/node_modules/**"],
    "headers": [
      {
        "source": "/build/app/**",
        "headers": [
          {
            "key": "Cache-Control",
            "value": "public, max-age=31536000"
          }
        ]
      },
      {
        "source": "ngsw-worker.js",
        "headers": [
          {
            "key": "Cache-Control",
            "value": "no-cache"
          }
        ]
      }
    ]
  }
}
```

Figura 71. Archivo firebase.json creado al terminar la configuración.

4. Finalmente se realiza el deploy en firebase. El proceso de deploy se ejecutó con éxito, como se observa en la figura 72. Y por último se observa la pantalla de inicio de sesión, como se visualiza en la figura 73.

```
i  deploying hosting
i  hosting[authenticationuser-18301]: beginning deploy...
i  hosting[authenticationuser-18301]: found 1537 files in www
+  hosting[authenticationuser-18301]: file upload complete
i  hosting[authenticationuser-18301]: finalizing version...
+  hosting[authenticationuser-18301]: version finalized
i  hosting[authenticationuser-18301]: releasing new version...
+  hosting[authenticationuser-18301]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/authenticationuser-18301/overview
Hosting URL: https://authenticationuser-18301.web.app
PS C:\repositorio\bolsaDeEmpleo\frontend>
```

Figura 72. Proceso de deploy con éxito.

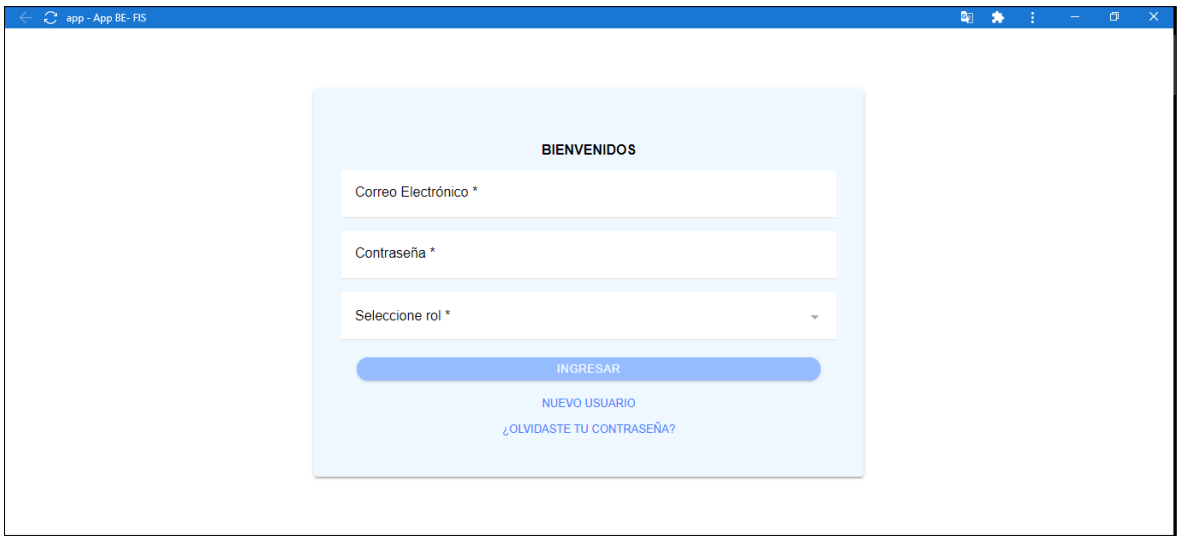


Figura 73. Interfaz de inicio de sesión en el hosting de firebase

2.12.3 Sprint Review

Pruebas de aceptación

En la Tabla 21 se listan los criterios de aceptación cumplidos para el quinto sprint.

Tabla 18. Pruebas de aceptación del quinto sprint.

Historia de Usuario	Criterios de aceptación	Cumplido
	Dado el despliegue de la aplicación cuando se configure el hosting de firebase, entonces, realizar el deploy de la aplicación.	Si

AH5-01	Dado el despliegue de la aplicación cuando se ingresa a la aplicación por medio del hosting de firebase, entonces, instalar la aplicación como si fuera una aplicación móvil.	Si
	Dado el despliegue de la aplicación cuando tenga las características de una PWA y se encuentre en el hosting de firebase, entonces, consumir el servicio de bolsa de empleo.	Si

Incremento obtenido

En este sprint se concluyó la construcción del sistema de bolsa de empleo con las características de una PWA. Además, se realizó el deploy en el hosting de firebase. La PWA permite instalar la aplicación en cualquier dispositivo móvil en este caso se instaló en la computadora portátil, como se pude observar en la figura 74.

Figura 74. Aplicación instalada en el dispositivo móvil.

El navegador Chrome permite instalar las aplicaciones PWA, como se puede observar en la figura 75. Se encontró algunos errores que se resolvieron en el transcurso del sprint y el resultado final es una aplicación de una sola página, como se pueden observar en figura 76.

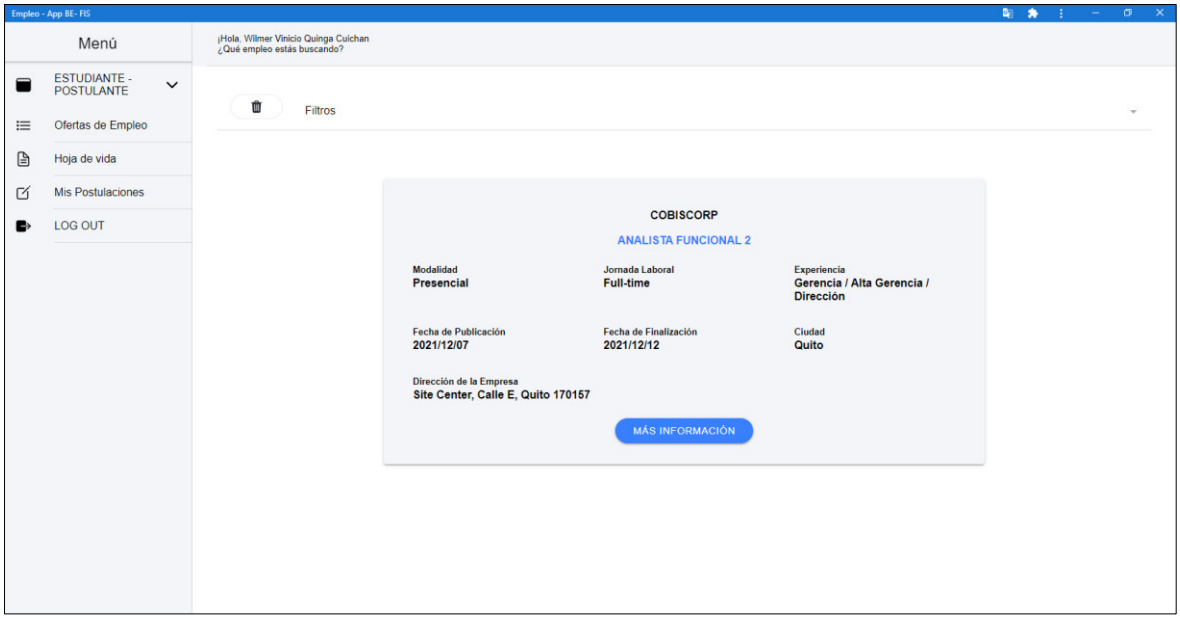


Figura 75. Pantalla principal del módulo estudiante - postulante.

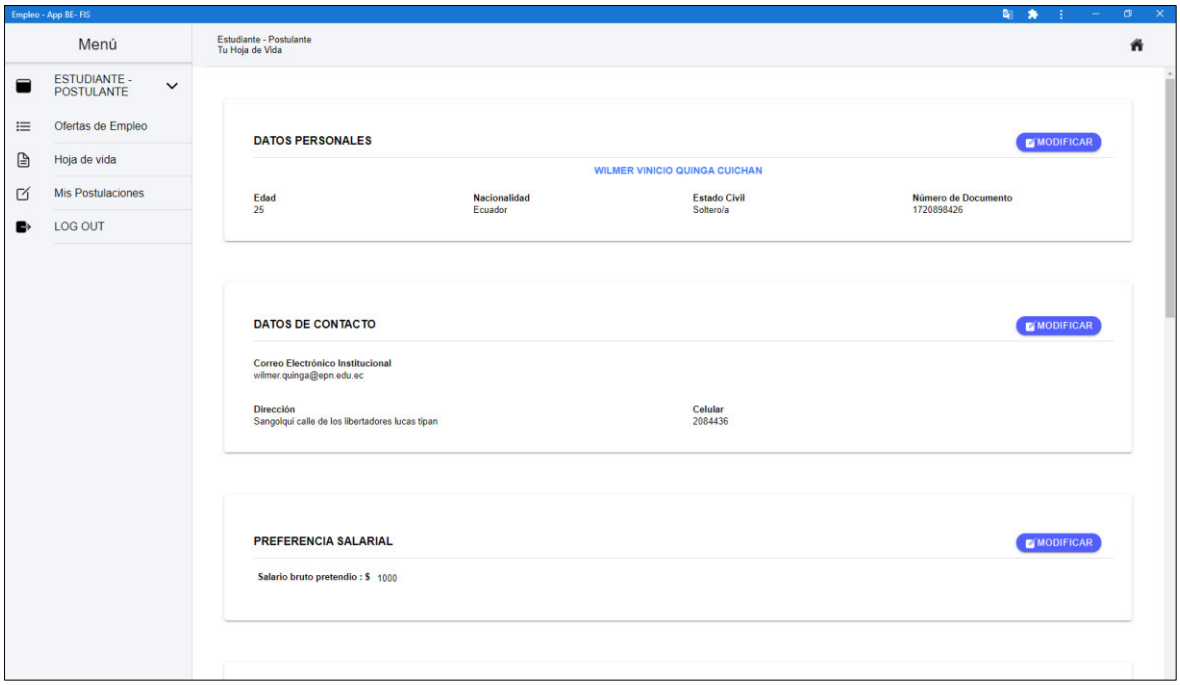


Figura 76. Pantalla del formulario hoja de vida.

Adaptación del Product Backlog

Tabla 19. Product Backlog adaptado tras finalizar el proyecto.

PRODUCT BACKLOG	
	HISTORIA DE USUARIO

HISTORIA ÉPICA	ID	Título	Estimación (días)	Estado
AH1	AH1-01	Inicio de sesión.	15	Terminado
AH2	AH2-01	Buscador de ofertas de empleo.	5	Terminado
	AH2-02	Registro de datos en hoja de vida del estudiante, egresado o graduado.	7	Terminado
	AH2-03	Buscador de oferta de empleo seleccionadas por el estudiante, egresado o graduado.	3	Terminado
AH3	AH3-01	Buscador de ofertas de empleo publicadas por las empresas.	5	Terminado
	AH3-02	Registro de oferta de empleo.	3	Terminado
	AH3-03	Buscador de estudiantes, egresados o graduados.	5	Terminado
	AH3-04	Registro de información de la empresa.	2	Terminado
	AH4-01	Registro de estudiantes, egresados o graduados que pertenezcan a la facultad de ingeniería en sistemas de la Escuela Politécnica Nacional.	4	Terminado

AH4	AH4-02	Visualizar a los usuarios estudiantes – postulantes registrados en el sistema.	3	Terminado
	AH4-03	Visualizar a los usuarios empresas registrados en el sistema.	4	Terminado
	AH4-04	Registro de datos de roles de usuario.	4	Terminado
AH5	AH5-01	Aplicación web progresiva (PWA).	15	Terminado

3 RESULTADOS Y DISCUSIÓN

3.1 Producto Final

El producto final del proyecto consta de 3 módulos:

Módulo usuario tipo empresa. Las empresas requieren contratar personal capacitado en este caso estudiantes de los últimos niveles, egresados y graduados de ingeniería en sistemas para formar parte de sus equipos de profesionales. La empresa tiene 2 opciones: publicar una oferta de empleo, como se puede observar en la figura 77, el formulario para el registro permite ingresar los diferentes datos para crear la oferta de empleo como dato importante la opción de descripción permite el ingreso de 2000 caracteres. Una vez publicada la oferta de empleo se comienza a recibir las postulaciones realizadas por el usuario tipo estudiante – postulante, como se observa en la figura 78. Además, visualizar sus datos y descargar en formato PDF.

La siguiente opción es la búsqueda en la base la cual dispone de todos los datos del usuario tipo estudiante – postulantes registrados en el sistema. La empresa por medio de los filtros establecidos puede realizar búsquedas y seleccionar a los candidatos idóneos, como se puede observar en la figura 79. La información de cada usuario tipo estudiante – postulante se puede visualizar y descargar en formato PDF.

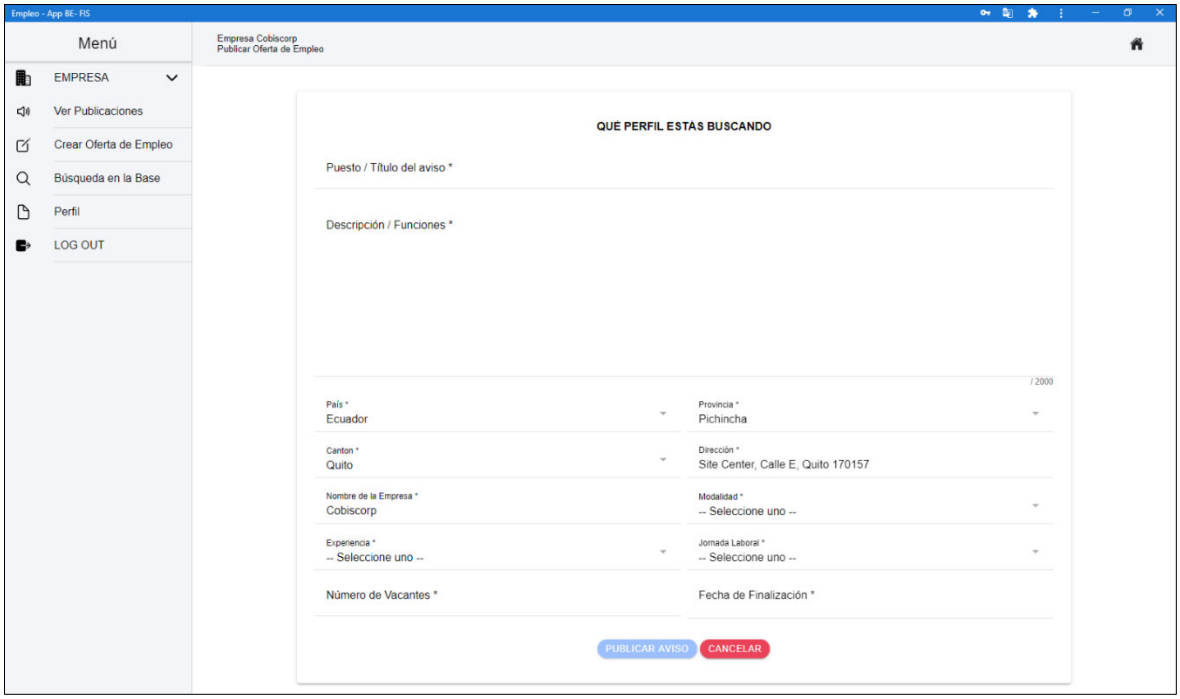


Figura 77. Pantalla de registro de oferta de empleo final

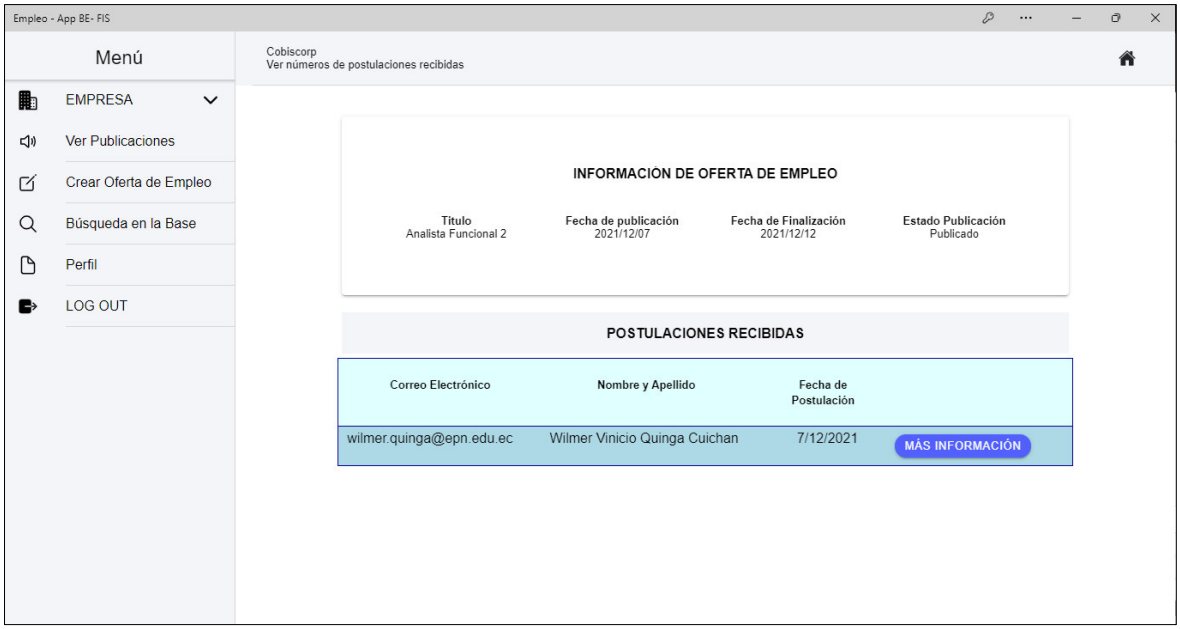


Figura 78. Pantalla que visualiza los usuarios tipo estudiante – postulante que postularon.

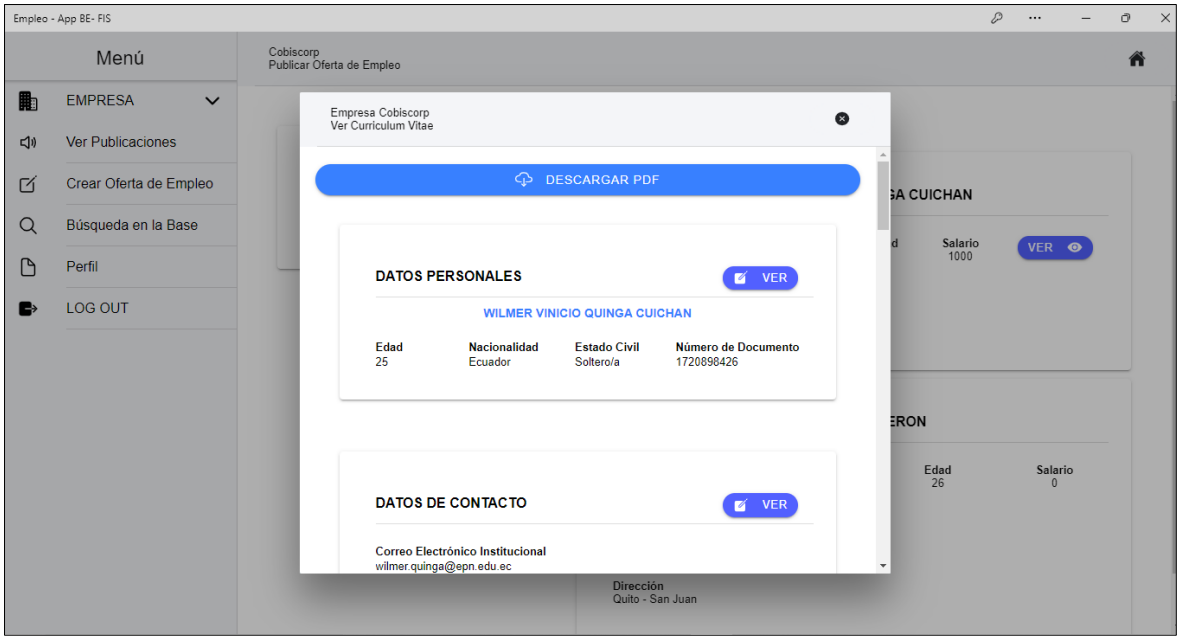
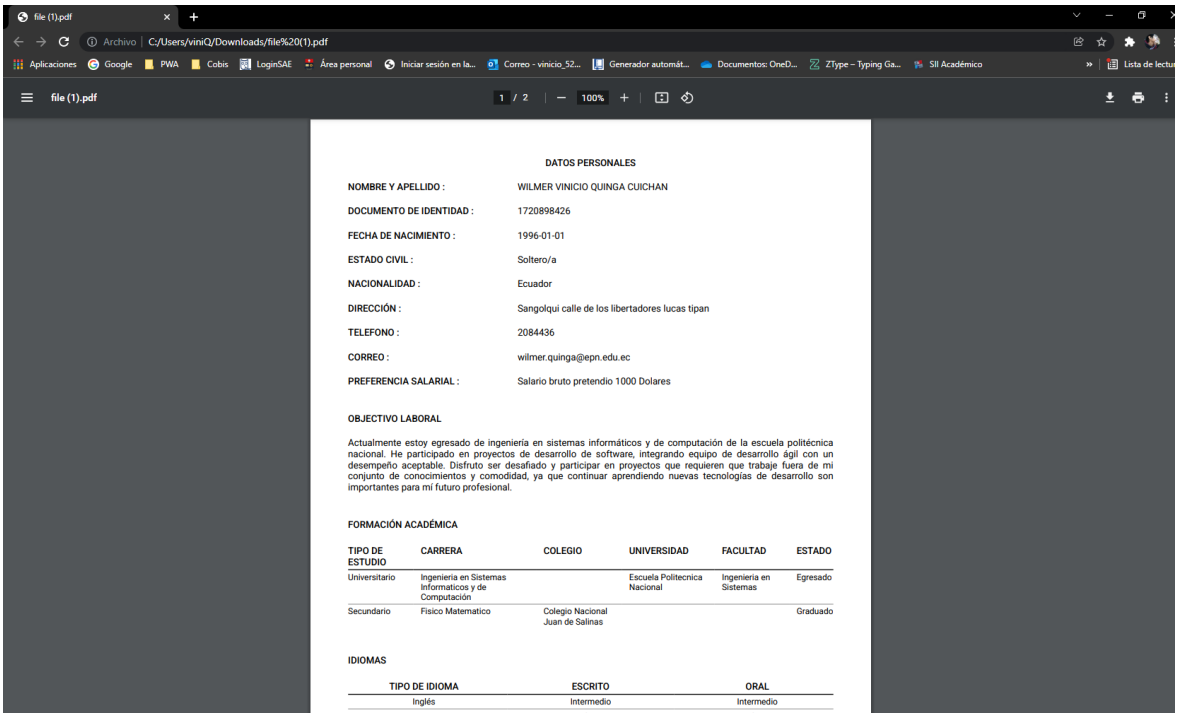


Figura 79. Pantalla para buscar usuarios tipo estudiantes – postulantes

Como se observa en la figura 81, el archivo de descarga con la información del formulario de hoja de vida del usuario tipo estudiante – postulante. Para más detalle revisar el anexo IX.



DATOS PERSONALES

NOMBRE Y APELLIDO :	WILMER VINICIO QUINGA CUICHAN
DOCUMENTO DE IDENTIDAD :	1720898426
FECHA DE NACIMIENTO :	1996-01-01
ESTADO CIVIL :	Soltero/a
NACIONALIDAD :	Ecuador
DIRECCIÓN :	Sangolquí calle de los libertadores lucas tipan
TELEFONO :	2084436
CORREO :	wilmer.quinga@epn.edu.ec
PREFERENCIA SALARIAL :	Salario bruto pretendio 1000 Dolares

OBJECTIVO LABORAL

Actualmente estoy egresado de ingeniería en sistemas informáticos y de computación de la escuela politécnica nacional. He participado en proyectos de desarrollo de software, integrando equipo de desarrollo ágil con un desempeño aceptable. Disfruto ser desafiado y participar en proyectos que requieren que trabaje fuera de mi conjunto de conocimientos y comodidad, ya que continuar aprendiendo nuevas tecnologías de desarrollo son importantes para mí futuro profesional.

FORMACIÓN ACADÉMICA

TIPO DE ESTUDIO	CARRERA	COLEGIO	UNIVERSIDAD	FACULTAD	ESTADO
Universitario	Ingenieria en Sistemas Informaticos y de Computación		Escuela Politecnica Nacional	Ingenieria en Sistemas	Egresado
Secundario	Fisico Matematico	Colegio Nacional Juan de Salinas			Graduado

IDIOMAS

TIPO DE IDIOMA	ESCRITO	ORAL
Inglés	Intermedio	Intermedio

Figura 80. Archivo PDF con formulario hoja de vidas

Módulo usuario tipo estudiante – postulante, el estudiante, egresado o graduado que pertenezca a la facultad de sistema de la Escuela Politécnica Nacional puede registrarse en el sistema de bolsa de empleo. La primera pantalla que visualiza es una lista de todas las ofertas de empleo disponibles, como se puede observar en la figura 81. El formulario dispone de filtros para obtener un subconjunto de ofertas de empleo. Cuando el usuario seleccioné una oferta de empleo tiene la opción para obtener más información en donde se especifica las habilidades y responsabilidad que el usuario debe de tener para postular en la oferta de empleo, como se pude ver en la figura 82.

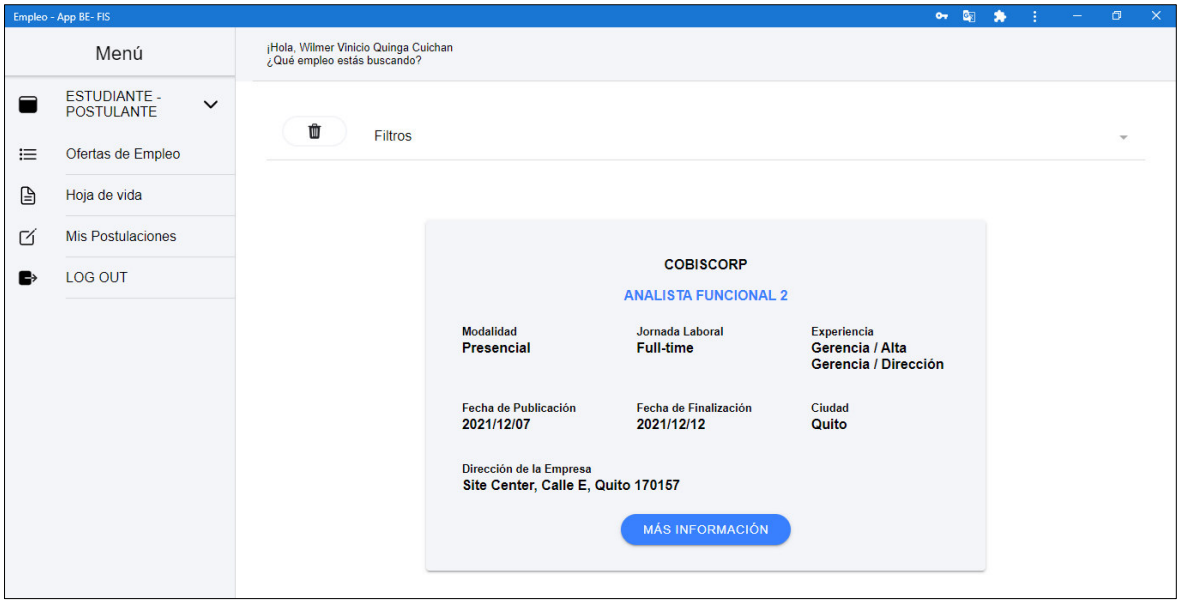


Figura 81. Pantalla donde se obtiene las ofertas de empleo disponibles.

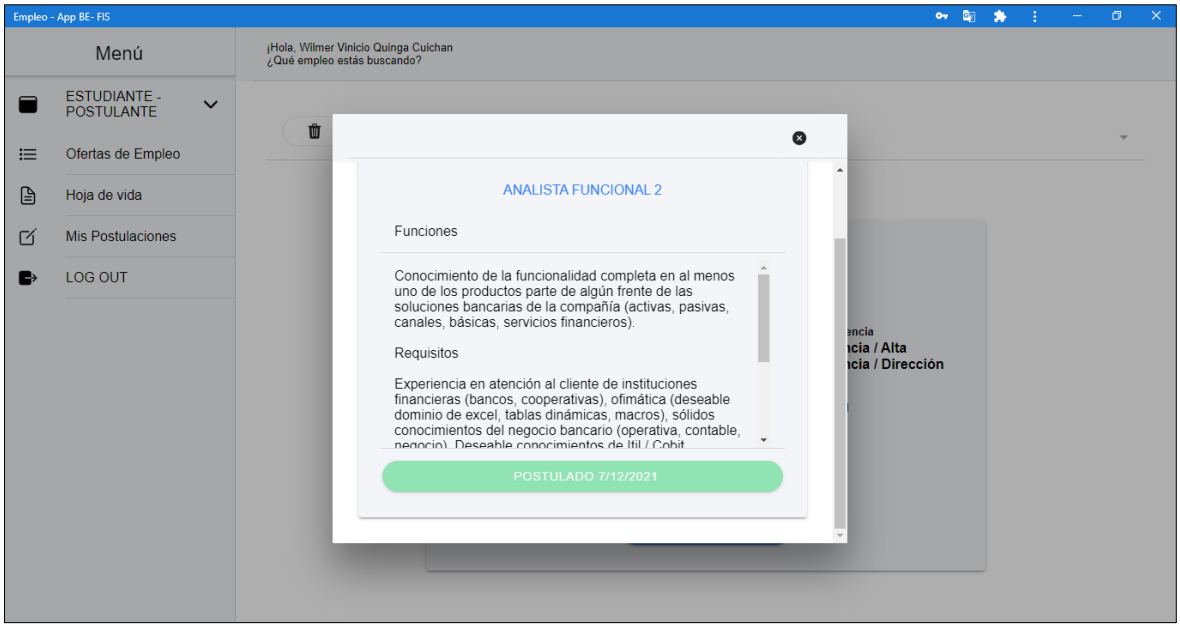


Figura 82. Detalle de oferta de empleo

El usuario puede postular en la oferta de empleo solo si lleno el formulario de hoja de vida, este formulario dispone la información principal del usuario que utilizara la empresa para seleccionar al candidato idóneo, como se puede observar en la figura 83.

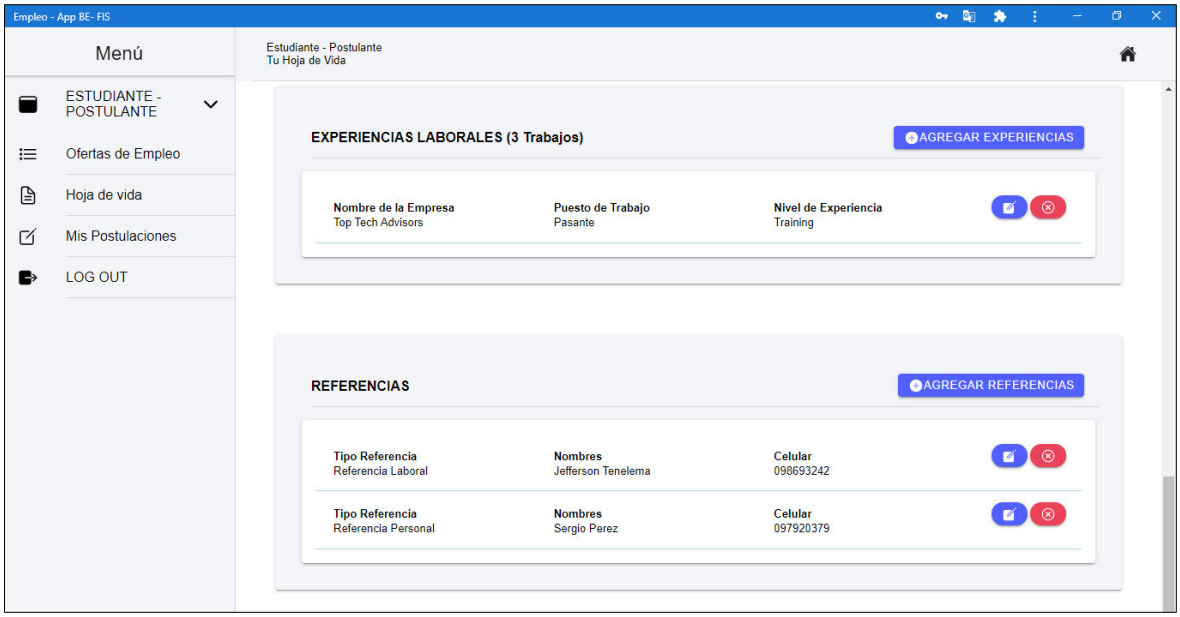


Figura 83. Formulario hoja de vida

Una vez realizada la postulación en la oferta de empleo el usuario tipo estudiante – postulante dispone de un servicio para visualizar el estado de la postulación, como se pude observar en la figura 84.

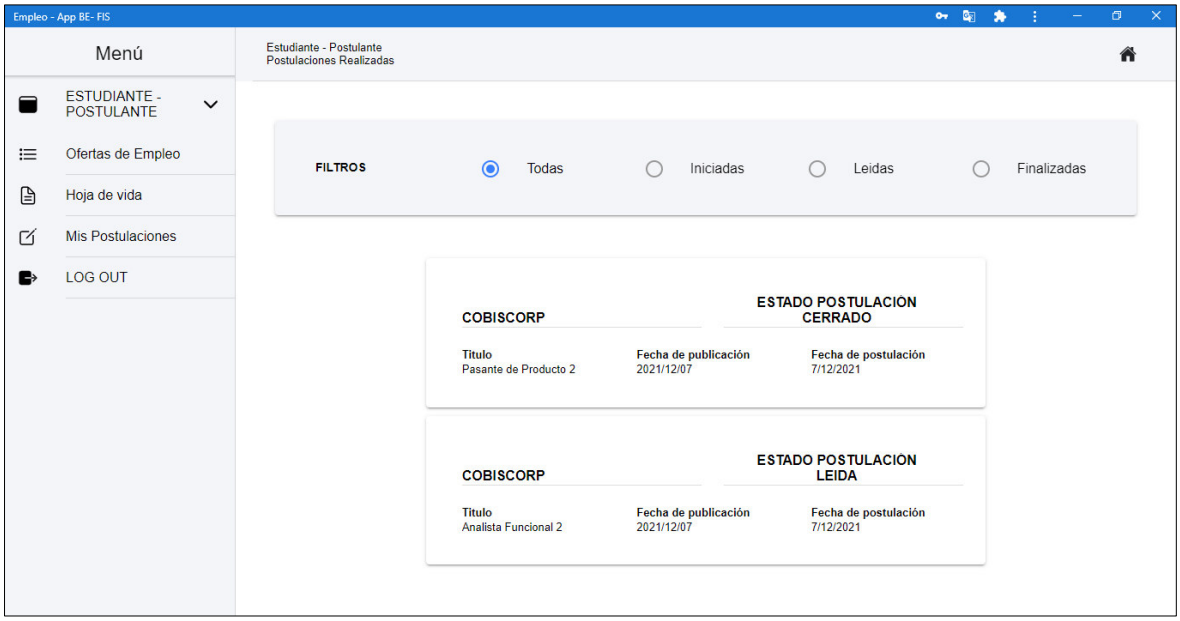


Figura 84. Pantalla donde se listarán las ofertas de empleo seleccionadas por el usuario

Módulo del usuario tipo administrador. El usuario tipo administrador tiene como función gestionar los datos registrados en el backend. En la pantalla para crear usuarios activos se registra a los usuarios que estén habilitados para registrarse como estudiante – postulante, el administrador antes de registrar los datos debe de verificar que el usuario pertenezca a la facultad de ingeniería de sistemas de la Escuela Politécnica Nacional, como se puede observar en la figura 85.

Figura 85. Usuarios habilitados para registrarse en el sistema de bolsa de empleo.

3.2 Pruebas con usuarios final

Se realizó pruebas del producto final con un total de 40 personas, 20 usuarios tipo estudiante – postulante donde intervinieron estudiantes de los últimos niveles, egresados y graduados de la facultad de ingeniería de sistemas de la Escuela Politécnica Nacional y 20 usuarios de tipo empresas en donde intervinieron personal de capital humano de Cobiscorp empresa desarrollo de software y de Conauto empresa de distribución de productos de mantenimiento automotriz. Se planteó un total de 6 preguntas a cada usuario 3 para la utilidad percibida y 3 para la facilidad de uso; la respuesta de cada pregunta se ajustó a una escala del 1 al 7 (donde 1 es la calificación mínima y 7 es la calificación máxima). Las preguntas realizadas y sus resultados se encuentran adjuntados en el Anexo X.

3.2.1 Facilidad de uso percibida

Los promedios obtenidos de cada pregunta se encuentran en la figura 86, del usuario tipo estudiante – postulante.

En la pregunta número uno se consultó sobre los datos para el registro del usuario son claros y precisos, se obtuvo un promedio de 6.75/7. Esto se debe a que se solicita solo los

datos esenciales para iniciar sesión adicional si el usuario comete un error en el registro de un dato se visualiza un mensaje de alerta.

En la pregunta número dos se consultó sobre la oferta de empleo, con sus requerimientos especificados fue fácil filtrar oportunidades de pasantía o trabajo, se obtuvo un promedio de 6.7/7. Esto debido a que el formulario de búsqueda de ofertas de empleo dispone de filtros. Para obtener las ofertas de empleo solo de pasantías o de trabajo se configura el filtro jornada laboral el cual dispone de las siguientes categorías: tiempo completo, tiempo parcial, pasantías entre otras. Seleccionado cualquier opción se puede encontrar la oferta de empleo deseada.

En la pregunta número tres se consultó sobre al usuario le fue fácil realizar la postulación en una oferta de empleo, se obtuvo un promedio de 6,6/7. Esto debido a que en la oferta de empleo publicada dispone de un botón de más información, el cual visualiza la opción para realizar la postulación, primero valida que el usuario tenga información en el formulario hoja de vida solo ahí podrá realizar la postulación.

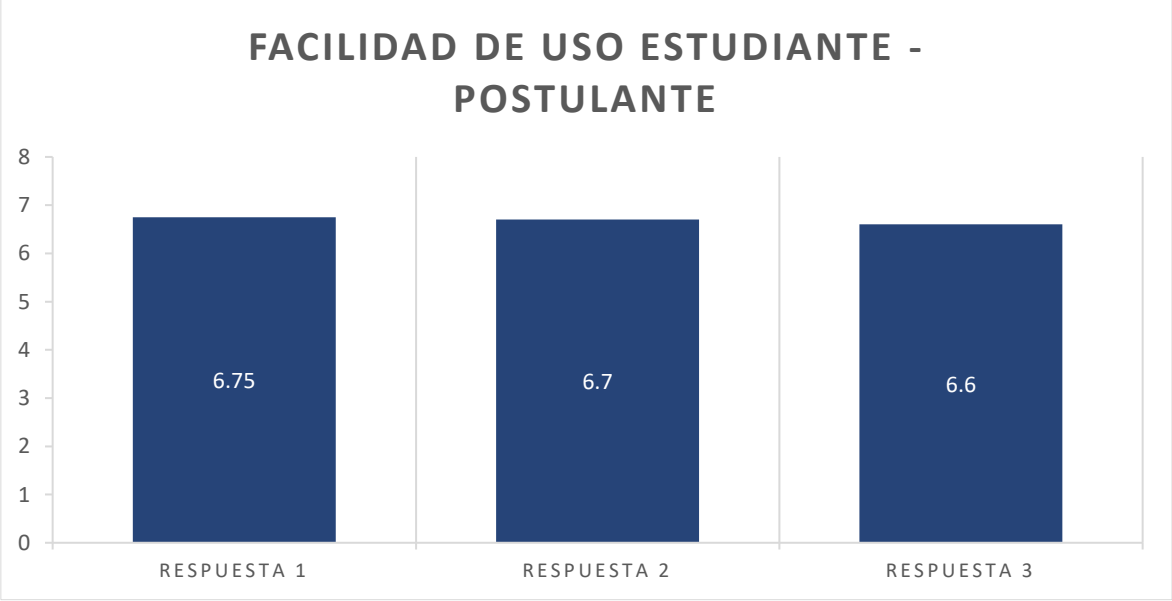


Figura 86. Promedio por pregunta de la facilidad de uso percibida en el producto final, usuario tipo Estudiante – Postulante.

Los promedios obtenidos de cada pregunta se encuentran en la figura 87, del usuario tipo empresa.

En la pregunta número uno, se consultó sobre los datos para el registro del usuario son claros y precisos, se obtuvo un promedio de 6.67/7. Esto se debe que se solicita al usuario

los datos esenciales para el inicio de sesión y los datos de la empresa son necesarios para validar el negocio, cualquier error en el llenado de los datos se visualiza un mensaje de alerta.

En la pregunta número dos, se consultó sobre los datos del formulario hoja de vida del usuario tipo estudiante – postulante, si facilito filtrar adecuados postulantes, se obtuvo un promedio de 6,42/7. Esto se debe, que con la información actual de trabajos anteriores, referencias personales, educación, sueldo pretendido y datos personales de la hoja de vida se puede obtener criterios suficientes para establecer el primer contacto con los candidatos.

En la pregunta número tres, se consultó sobre si al usuario tipo empresa le fue sencillo obtener los datos de los usuarios tipo estudiante – postulante que postularon en la oferta de empleo publicada, se obtuvo un promedio de 6,48/7. Esto debió a que la primera pantalla que se visualiza del usuario tipo empresa tiene un listado de las ofertas de empleo publicadas con la opción de visualizar a todo los usuarios tipo estudiantes – postulantes que postularon adicional, tiene la opción de visualizar su hoja de vida del postulante y descargar la información en formato PDF.

En las tres preguntas se obtuvieron un porcentaje alto porque, al momento de utilizar la aplicación por parte de los usuarios encuestados les resulto muy fácil entender los formularios y gracias a las interfaces intuitivas pudieron realizar los procesos solicitados sin ningún problema.

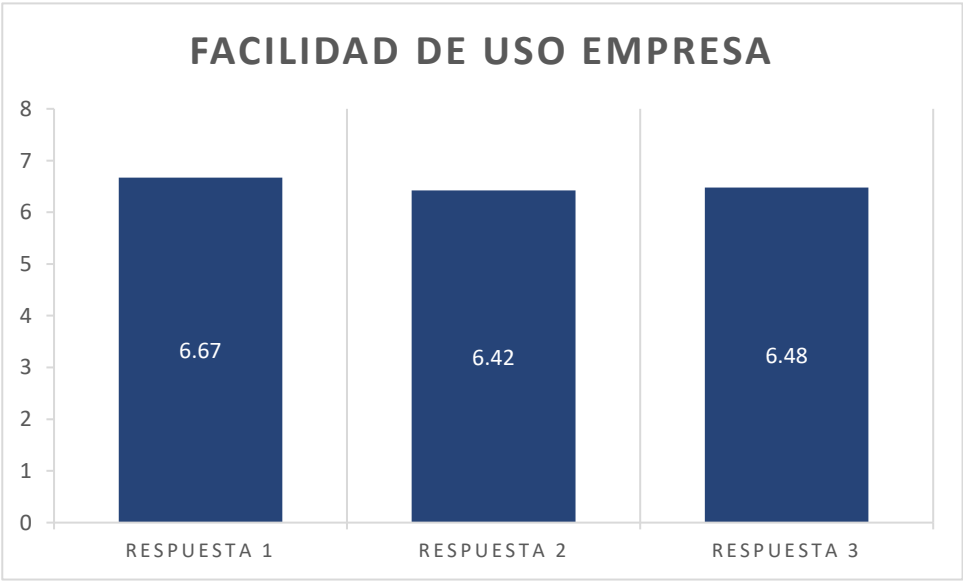


Figura 87. Promedio por pregunta de la facilidad de uso percibida en el producto final, usuario tipo empresa.

3.2.2 Utilidad percibida

Los promedios obtenidos de cada pregunta se encuentran en la figura 88, del usuario tipo estudiante - postulante.

En la pregunta número uno, se consultó sobre si la aplicación permitió encontrar una oferta de empleo de acuerdo con su nivel de experiencia, se obtuvo un promedio de 6,6/7. Esto debió a que el usuario puedo utilizar los filtros en el formulario de búsqueda de oferta empleo para encontrar a un determinado empleo.

En la pregunta número dos, se consultó sobre si la aplicación permitió encontrar una oferta de empleo de acuerdo con su disponibilidad de tiempo, se obtuvo un promedio de 6,75/7. Esto debió a la configuración de los filtros en el formulario de búsqueda de empleo, utilizando el filtro jornada laboral se puede elegir la oferta de empleo de acuerdo con su disponibilidad de tiempo.

En la pregunta número tres, se consultó sobre si los datos de la hoja de vida son suficientes para obtener una oferta de empleo, se obtuvo un promedio de 6,25/7. Esto debió a que el formulario hoja de vida solicita datos de los anteriores trabajos en los cuales el usuario tipo empresa puede tener un conocimiento del nivel de experiencia del usuario tipo estudiante – postulante y con los otros datos que se solicita en la hoja de vida se complementa la información para obtener el primer contacto con la empresa.

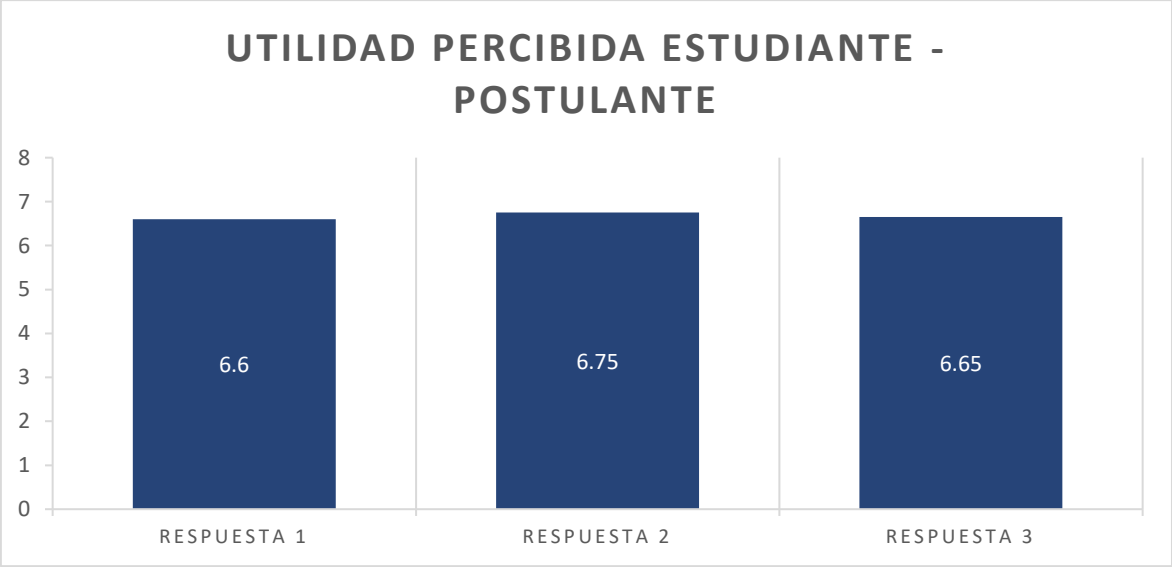


Figura 88. Promedio por pregunta de la utilidad percibida en el producto final, usuario tipo estudiante – postulante.

Los promedios obtenidos de cada pregunta se encuentran en la figura 89, del usuario tipo empresa.

En la pregunta número uno, se consultó sobre si la aplicación permitió encontrar a un candidato de acuerdo con los requerimientos propuestos, se obtuvo un promedio de 6,7/7. Esto debido a que la aplicación permite detallar correctamente los requerimientos para la nueva vacante y los usuarios tipo estudiante – postulante dependiendo de su nivel de experiencia pude postular en la oferta de empleo así la empresa puede seleccionar a los candidatos idóneos.

En la pregunta número dos, se consultó sobre la información de la hoja de vida perteneciente al usuario tipo estudiante – postulante es suficiente para reconocer las características esperadas de quien ocupará la vacante, se obtuvo un promedio de 6,7/7. Esto debido a que la hoja de vida perteneciente al usuario tipo estudiante – postulante posee información sobre los estudios realizados, sus anteriores trabajos y las referencias personales y con esta información se puede seleccionar a los postulantes con las características esperadas. Una sugerencia que salió de esta pregunta fue el adjuntar certificados de anteriores trabajos o de estudios los cuales se detalla en la sesión de recomendaciones para mejorar la aplicación.

En la pregunta número tres, se consultó sobre el formulario para crear una oferta de empleo permitió crear una publicación de trabajo con las necesidades de la empresa, se obtuvo un promedio de 6,65/7. Esto debió a que en el formulario para crear una oferta de empleo dispone de un campo llamado descripción de funciones en donde se puede detallar las necesidades de la empresa para la nueva vacante, también dispone de otros datos como nivel de experiencia, jornada laboral entre otras, para tener una oferta de empleo que cumpla con los requerimientos de la empresa.

En las tres preguntas se obtuvieron un porcentaje alto porque al momento de utilizar la aplicación por parte de los usuarios encuestados notaron que los procesos planteados pueden mejorar la etapa de contratar nuevo personal capacitado.

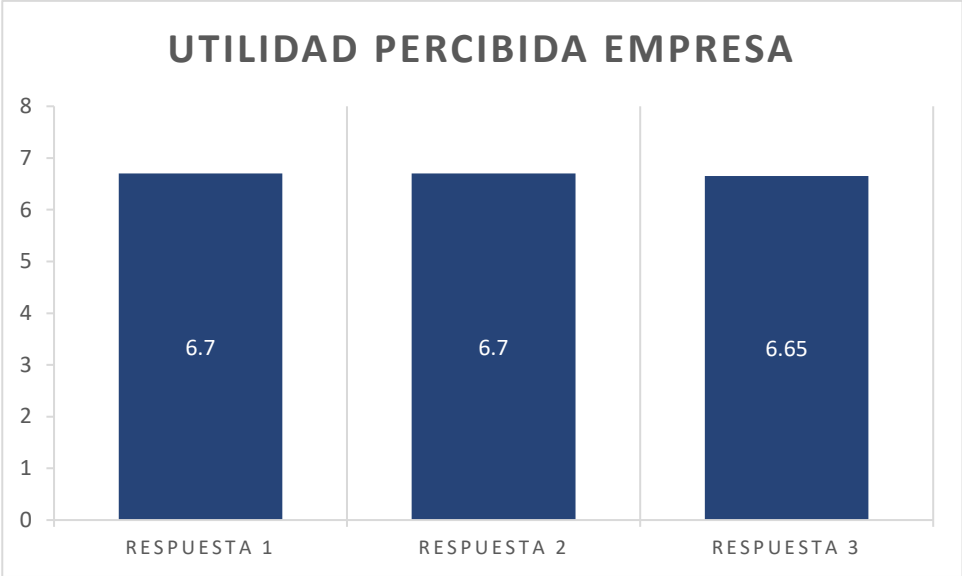


Figura 89. Promedio por pregunta de la utilidad percibida en el producto final, usuario tipo empresa.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se desarrollo un sistema que permite registrar a 2 usuarios, el primer usuario es de tipo empresa que permite registrar los datos de una compañía y publicar una oferta de empleo con sus requerimientos para contratar a un empleado y el segundo usuario es de tipo estudiante – postulante permite registrar los datos en un formulario llamado hoja de vida a los estudiantes, egresados y graduados y elegir una oferta de empleo acorde a su experiencia, disponibilidad de tiempo entre otros factores.
- El sistema se desarrolló con la guía del marco de referencia SCRUM el cual permitió crear un conjunto de tareas en la planeación de cada sprint sin ningún tipo de prioridad. Con lo cual el desarrollador determinará cual será la mejor secuencia para cumplir con el objetivo del Sprint, permitiendo así el avance rápido con las tareas sencillas y diseñando soluciones para las tareas complejas; a diferencia del marco de referencia XP en el cual el equipo si debe de seguir un orden de prioridad estricto en las tareas.
- El marco de referencia SCRUM posee un conjunto de directrices con las cuales se construyó el sistema generando un sprint con la duración de 15 días en los cuales se completó las tareas programadas y los corrigió los errores encontrados así al final de este se entregaban pequeños incrementos del software estables.

- La aplicación fue construida con el framework Ionic en el cual se configura una propiedad para que todas las pantallas se adapten a cualquier dispositivo móvil ayudado por el control grid que posee columnas y filas con diferentes puntos de adaptación según el tamaño de pantalla.
- La aplicación posee características de una aplicación web progresiva (PWA), las que permite al sistema adaptarse a cualquier tamaño de pantalla, con la ayuda del navegador web tiene la opción de instalarse como si fuera una aplicación móvil y por último la aplicación se puede instalar en diferentes tipos de sistemas operativos Windows, Android, IoS.
- Al momento de implementar el sistema de bolsa de empleo se realizó un levantamiento de requerimientos que permitió descubrir una gran variedad de funcionalidades las cuales fueron implementadas para cumplir con el objetivo principal del proyecto.
- Los aspectos de usabilidad integrados en la aplicación otorgaron a las interfaces del usuario elementos legibles y entendibles, coherencia en la navegación, una semántica acorde a los usuarios que utilizara el producto y opción de ayuda al momento de cometer un error. Estos parámetros generan una aplicación de alta calidad por lo que el usuario no tendrá inconvenientes al momento de utilizar el sistema de bolsa de empleo.

4.2 Recomendaciones

Para trabajos futuros se recomienda:

- Agregar más categorías de datos en la formulario hoja de vida que pertenece al usuario tipo estudiante – postulante así la empresa puede tener más factores de evaluación para contratar a un empleado.
- Agregar más características de una aplicación web progresiva como por ejemplo notificación push el cual alertará al usuario tipo estudiante – postulante sobre una nueva oferta de empleo. Y hacer que la aplicación trabaje sin conexión a internet con ayuda del service worker.
- Crear un servicio para eliminar el autocomple que en cada navegador web tiene su respectiva configuración así se elimina que se guarde información errónea los formularios.

- Crear una lógica para subir archivo a la aplicación. El usuario tipo empresa puede subir una imagen con los requerimientos de trabajo y el usuario tipo empresa puede subir sus certificados de trabajo o de estudio ayudando al proceso del sistema.
- Mejorar el formato de PDF de la hoja de vida del usuario tipo estudiante – postulante dependiendo de los requerimientos de las empresas.
- En el formulario para buscar a los usuarios tipo estudiante – postulante agregar más categorías en los filtros para que la empresa pueda seleccionar a los candidatos que cumplan con las necesidades de la empresa.
- Se recomienda usar librerías actualizadas, para evitar errores futuros en la funcionalidad de la aplicación. Con el servicio de firebase puede agregar más funcionalidades dependiendo de sus actualizaciones y de la necesidad de los usuarios.

5 Bibliografía

[1] A. Team, «appsbee,» 29 11 2018. [En línea]. Available: https://www.appsbee.com/blog/evolution-of-cross-platform-apps-in-present-day-business/. [Último acceso: 27 11 2021].

[2] A. MANCHANDA, «net solutions,» 08 12 2020. [En línea]. Available: https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019. [Último acceso: 28 11 2021].

[3] F. A. a. R. Tapia, Gestionando empresas en la sociedad de la información, Madrid, 2004.

[4] C. Proctor, «Zipjob,» 10 06 2021. [En línea]. Available: https://www.zipjob.com/blog/how-to-look-for-jobs/. [Último acceso: 28 11 2021].

[5] G. I. J. a. P. P. Aldás, Fundamentos de dirección de empresas, Madrid, 2016.

[6] T. T. Contributor, «TECHTARGET NETWORK,» 12 2017. [En línea]. Available: https://whatis.techtarget.com/definition/progressive-web-app-PWA. [Último acceso: 28 11 2021].

[7] D. Sheppard, Beginning Progressive Web App Development, Berkeley, 2017.

[8] C. P. a. M. L. F. Luna, PROGRAMACION WEB Full Stack 21 - Potenciar la faceta full stack: Desarrollo frontend y backend - Curso visual y práctico, 1st ed, : RedUsers, 2018.

[9] Ionic, «The Architect's Guide to Progressive Web Apps,» *Ionic,* pp. 1-15.

[10] T. Dimes, Conceptos basicos de scrum, Babelcube, 2015.

[11] M. Biehl, RESTful API Design, 2016.

[12] M. Massé, REST API, United State ofAmerica, 2012.

[13] R. Bandiera, Diseño e Desarrollo Web con Codeigniter 3, 2019.

[14] I. SOMMERVILLE, Ingeniería del software séptima edición, 2006.

[15] nodejs, «nodejs,» 2021. [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: 28 11 2021].

[16] npmjs, «npm Docs,» 2021. [En línea]. Available: https://docs.npmjs.com/about-npm. [Último acceso: 28 11 2021].

[17] ionicframework, «ionicframework,» 2020. [En línea]. Available: https://ionicframework.com/what-is-ionic#. [Último acceso: 28 11 2021].

[18] angular, «angular,» 2021. [En línea]. Available: https://angular.io/docs. [Último acceso: 28 11 2021].

[19] sailsjs, «sailsjs,» 2021. [En línea]. Available: https://sailsjs.com/documentation/concepts. [Último acceso: 28 11 2021].

[20] git-scm, «git-scm,» 2021. [En línea]. Available: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F. [Último acceso: 28 11 2021].

[21] code.visualstudio, «code.visualstudio,» 2021. [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: 28 11 2021].

[22] jetbrains, «webstorm,» 01 10 2021. [En línea]. Available: https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html. [Último acceso: 28 11 2021].

[23] J. Sharir, «Cómo utilizar Postman para la automatización de pruebas de API,» 24 03 2020. [En línea]. Available: https://www.blazemeter.com/blog/how-use-postman-manage-and-execute-your-apis. [Último acceso: 28 11 2021].

[24] notepad, «What is Notepad++,» 2021. [En línea]. Available: https://notepad-plus-plus.org/. [Último acceso: 28 11 2021].

[25] dbeaver, «dbeaver about,» 2021. [En línea]. Available: https://dbeaver.io/about/. [Último acceso: 28 11 2021].

[26] C. Hope, «Chrome,» 11 06 2021. [En línea]. Available: https://www.computerhope.com/jargon/c/chrome.htm. [Último acceso: 28 11 2021].

[27] Mysql, «What is MySQL: MySQL Explained For Beginners,» 19 08 2021. [En línea]. Available: https://www.hostinger.com/tutorials/what-is-mysql. [Último acceso: 28 11 2021].

[28] docker, «Docker overview,» 2021. [En línea]. Available: https://docs.docker.com/get-started/overview/. [Último acceso: 28 11 2021].

[29] E. Team, «What is Firebase?,» 2021. [En línea]. Available: https://www.educative.io/edpresso/what-is-firebase. [Último acceso: 28 11 2021].

[30] lucidchart, «LUCIDCHART HELP,» 2021. [En línea]. Available: https://www.lucidchart.com/pages/api_documentation. [Último acceso: 28 11 2021].

[31] N. A. A. Jamaludin, Requirements Engineering Project-Based Learinig Model Using the Electronic Learning Software Engineering System.

[32] J. D. C. Q. Andrés Fernando Solano Alegria, Evaluación Colaborativa de la usabilidad en el Desarrollo de Sistemas Software Interactivos, Santiago de Cali, 2015.

[33] «pdfmake,» 18 11 2021. [En línea]. Available: https://www.npmjs.com/package/pdfmake. [Último acceso: 18 12 2021].

[34] carlosazaustre, Desarrollo web ágil con angular js, 2015.

[35] ionicframework, «Progressive Web Apps in Angular,» 2021. [En línea]. Available: https://ionicframework.com/docs/angular/pwa. [Último acceso: 06 12 2021].

[36] J. D. C. Q. Andrés Fernando Solano Alegría, Evaluación colaborariva de la usabilidad en el desarrollo de sistemas software interactivos, Santiago de Cali, 2015.

6 ANEXOS

6.1 Anexo I: Historias Épicas

HISTORIA DE USUARIO ÉPICA	
ID: AH1	**Prioridad:** Alta
Título: Implementar inicio de sesión.	
Descripción: Como usuario, necesito ingresar al sistema para utilizar las funcionalidades del sistema.	

HISTORIA DE USUARIO ÉPICA	
ID: AH2	**Prioridad:** Alta
Título: Implementar módulo de estudiante – postulante encargado de gestionar datos del usuario.	
Descripción: Como estudiante – postulante, necesito registrar mis datos personales para postularme en las ofertas de empleo registradas en el sistema.	

HISTORIA DE USUARIO ÉPICA	
ID: AH3	**Prioridad:** Alta
Título: Implementar módulo de empresa encargado crear y publicar ofertas de empleo.	
Descripción: Como empresa, necesito crear ofertas de empleo para que los usuarios estudiantes – postulante se postulen.	

HISTORIA DE USUARIO ÉPICA	
ID: AH4	**Prioridad:** Media
Título: Implementar módulo de administrador encargado de gestionar la información de los usuarios estudiante – postulante y empresa.	
Descripción: Como administrador, necesito visualizar a los usuarios registrados para gestionar y validar sus datos.	

HISTORIA DE USUARIO ÉPICA	
ID: AH5	**Prioridad:** Alta
Título: Aplicación web progresiva (PWA).	

Descripción: Como usuario, necesito la funcionalidad de una PWA para instalar la aplicación en un dispositivo móvil.

6.2 Anexo II: Historias de Usuario del Sprint 1

HISTORIA DE USUARIO		
ID: AH1-01	**Días estimados:** 15	
Título: Inicio de sesión.		
Sprint N°: 1	**Prioridad:** Alta	**Estado:** Por implementar
Descripción: Como usuario, deseo ingresar mi correo electrónico y contraseña para ingresar al sistema.		
Criterios de aceptación	Dado el despliegue del formulario registro de empresa y estudiante – postulante cuando los campos requeridos estén llenos, entonces, los botones tipo submit se activarán.	
	Dado el despliegue del servicio del inicio de sesión cuando el usuario se loguea se verificará que su correo electrónico registrado este validado, entonces, le permitirá el ingreso al sistema.	
	Dado el despliegue del formulario de registro de empresa y estudiante – postulante cuando se proceda a guardar la información en el backend, entonces, se verifica que el correo electrónico no este registrado.	
	Dado el despliegue del inicio de sesión cuando el usuario se loguea y no este validado su correo electrónico, entonces, se permite reenviar el mensaje de validación.	
	Dado el despliegue del inicio de sesión cuando el usuario se loguea y no se genere ningún problema, entonces, se guarda su información (email, id rol e id de Firebase) en el local storage.	
	Dado el despliegue del formulario restablecer contraseña cuando el usuario cambie de contraseña, entonces, la nueva información se actualizará en el backend y en firebase.	

6.3 Anexo III: Historias de Usuario del Sprint 2

HISTORIA DE USUARIO		
ID: AH2-01	**Días estimados:** 5	
Título: Buscador de ofertas de empleo.		
Sprint N°: 2	**Prioridad:** Alta	**Estado:** Por implementar

Descripción: Como estudiante – postulante, deseo visualizar todas las ofertas de empleo para obtener la información y realizar la postulación.	
Criterios de aceptación	Dado el despliegue del formulario buscar de ofertas de empleo cuando el usuario ingrese a este servicio, entonces, se obtendrán las ofertas de empleo en estado activo.
	Dado el despliegue del formulario buscador de ofertas de empleo cuando el usuario utiliza el servicio, entonces, se activarán las opciones para filtrar información.
	Dado el despliegue del formulario buscador de ofertas de empleo cuando el usuario seleccione una oferta de empleo, entonces, se visualizará un botón para obtener más información.
	Dado el despliegue del formulario buscador de ofertas de empleo cuando el usuario realice la postulación, entonces, se validará que el usuario tenga en el formulario hoja de vida.

HISTORIA DE USUARIO		
ID: AH2-02		**Días estimados:** 5
Título: Registro de datos en hoja de vida del estudiante, egresado o graduado.		
Sprint N°: 2	**Prioridad:** Alta	**Estado:** Por implementar
Descripción: Como estudiante – postulante, deseo ingresar mis datos en un formulario con las características de una hoja de vida para que las empresas obtengan esta información.		
Criterios de aceptación	Dado el despliegue del formulario hoja de vida cuando el usuario registre su información, entonces, el formulario permitirá guardar, visualizar y actualizar los datos.	
	Dado el despliegue del formulario hoja de vida cuando el usuario registre su información, entonces, el formulario tendrá los siguientes campos datos personales, datos de contacto, preferencia salarial, objetivo laboral, educación, idiomas, experiencia laboral y referencias.	
	Dado el despliegue del formulario hoja de vida cuando el usuario ingrese sus datos en los campos educación, idiomas, experiencia laboral, entonces, se habilitará el botón agregar más ítems.	
	Dado el despliegue del formulario hoja de vida cuando el usuario llene los campos obligatorios, entonces, se habilitarán los botones de tipo submit.	

HISTORIA DE USUARIO		
ID: AH2-03	**Días estimados:** 5	
Título: Buscador de oferta de empleo seleccionadas por el estudiante, egresado o graduado.		
Sprint N°: 2	**Prioridad:** Media	**Estado:** Por implementar
Descripción: Como estudiante – postulante, deseo visualizar las ofertas de empleo que realice la postulación para conocer el estado de esta.		
Criterios de aceptación	Dado el despliegue del formulario buscador de las ofertas de empleo seleccionadas cuando el usuario ingrese a este servicio, entonces, se visualizarán solo las ofertas de empleo que el usuario selecciono o postulo.	
	Dado el despliegue del formulario buscador de las ofertas de empleo cuando el usuario ingrese a este servicio, entonces, se habilitarán las opciones para filtrar datos.	

6.4 Anexo IV: Historias de Usuario del Sprint 3

HISTORIA DE USUARIO		
ID: AH3-01	**Días estimados:** 20	
Título: Buscador de ofertas de empleo publicadas por las empresas.		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** En desarrollo
Descripción: Como empresa, deseo visualizar todas las ofertas de empleo para poder obtener por cada registro la información de los usuarios que se postularon.		
Criterios de aceptación	Dado el despliegue del formulario para obtener todas las ofertas de empleo publicadas cuando el usuario seleccione este formulario, entonces, se visualiza todas las ofertas de empleo sin importar su estado.	
	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario guarde la información en el backend, entonces, se habilita la opción para actualizar los datos.	
	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario seleccione la opción de editar, entonces, se habilita la opción de finalizar oferta de empleo.	
	Dado el despliegue del formulario para obtener todas las ofertas de empleo creadas cuando el usuario seleccione una oferta, entonces, se	

	habilita la opción para revisar que usuarios tipo estudiantes – postulantes optaron por seleccionar esta oferta de empleo.
	Dado el despliegue del formulario para obtener todas las ofertas de empleo creadas cuando el usuario tipo empresa seleccione una oferta de empleo publicada y escoja a un usuario tipo estudiante – postulante que postulo, entonces, el sistema permite obtener todos los datos del usuario tipo estudiante - postulante y descargar la información en formato PDF.

HISTORIA DE USUARIO		
ID: AH3-02	**Días estimados: 5**	
Título: Registro de datos de oferta de empleo.		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** Por implementar
Descripción: Como empresa, deseo crear una oferta de empleo para que los usuarios estudiantes – postulante se puedan postular.		
Criterios de aceptación	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario ingrese todos los campos obligatorios, entonces, se habilitan el botón de publicar aviso.	
	Dado el despliegue del formulario para registrar una oferta de empleo cuando llene los campos solicitados, entonces, el campo llamado descripción de funciones permite el ingreso de 2000 caracteres.	
	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario llene todos los campos requeridos, entonces, al momento de guardar los datos en el backend se incluye la fecha que realizo el registro.	
	Dado el despliegue del formulario para registrar una oferta de empleo cuando el usuario llene los datos solicitados, entonces, en los camboBox de país, provincia y cantón se llena en cascada.	

HISTORIA DE USUARIO		
ID: AH3-03	**Días estimados:** 10	
Título: Buscador de estudiantes, egresados o graduados.		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** Por implementar

Descripción: Como empresa, deseo obtener la información de todos los usuarios estudiante – postulante para obtener al candidato idóneo.	
Criterios de aceptación	Dado el despliegue del formulario para buscar a los usuarios estudiantes – postulantes cuando el usuario ocupe este servicio, entonces, se listan todos los datos registrados en el backend pertenecientes a los estudiantes – postulantes.
	Dado el despliegue del formulario para buscar al usuario tipo estudiante – postulante cuando el usuario ocupe el servicio, entonces, se habilita la opción de filtrar datos por edad, lugar de residencia, pretensión salarial, género y educación.
	Dado el despliegue del formulario para buscar al usuario tipo estudiantes – postulantes cuando la empresa seleccione a un estudiante – postulante, entonces, se abre un modal con toda la información del usuario seleccionado.
	Dado el despliegue del formulario para buscar al usuario tipo estudiante – postulante cuando la empresa seleccione a un estudiante – postulante, entonces, puede descargar la información del usuario seleccionado en formato PDF.

HISTORIA DE USUARIO		
ID: AH3-04	**Días estimados:** 5	
Título: Registro de información de la empresa.		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** Por implementar
Descripción: Como empresa, deseo registrar mi información para que sea visualizada en la oferta laboral.		
Criterios de aceptación	Dado el despliegue del formulario para obtener datos de la empresa cuando el usuario ocupe este servicio, entonces, se visualiza la información registrada de la empresa.	
	Dado el despliegue del formulario para obtener datos de la empresa cuando exista algún cambio en la información de la empresa, entonces, el sistema permite actualizar los datos deseados.	
	Dado el despliegue del formulario para obtener datos de la empresa cuando el usuario ingrese los datos obligatorios, entonces, se habilita el botón para actualizar la información.	

6.5 Anexo V: Historias de Usuario del Sprint 4

HISTORIA DE USUARIO		
ID: AH4-01	**Días estimados:** 4	
Título: Registro de estudiantes, egresados o graduados que pertenezcan a la facultad de ingeniería en sistemas de la Escuela Politécnica Nacional.		
Sprint N°: 4	**Prioridad:** Alta	**Estado:** Por implementar
Descripción: Como administrador, deseo visualizar los datos de los estudiantes que se podrán registrar en el sistema para llevar un control.		
Criterios de aceptación	Dado el despliegue del formulario para listas los datos de los usuarios activos cuando el usuario ocupe este servicio, entonces, se listan los datos de todos los usuarios que se podrán registrar como estudiante – postulante.	
	Dado el despliegue del formulario para listar los datos de los usuarios activos cuando el usuario ocupe este servicio, entonces, podrán registrar otros usuarios.	

HISTORIA DE USUARIO		
ID: AH4-02	**Días estimados:** 3	
Título: Listar a los usuarios tipo estudiantes – postulantes registrados en el sistema.		
Sprint N°: 4	**Prioridad:** Baja	**Estado:** Por implementar
Descripción: Como administrador, deseo visualizar los datos de los usuarios estudiantes – postulantes registrados en el sistema para llevar un control.		
Criterios de aceptación	Dado el despliegue del formulario para listar a los estudiantes – postulantes cuando el usuario ingrese a este servicio, entonces, se listan todos los usuarios de tipo estudiante - postulante registrados en el sistema.	
	Dado el despliegue del formulario para listar a los estudiantes – postulantes cuando el usuario ocupe este servicio en otros dispositivos con diferentes tamaños de pantalla, entonces, el formulario se adapta a cualquier tamaño de pantalla.	

HISTORIA DE USUARIO		
ID: AH4-03	**Días estimados:** 4	
Título: Listar a los usuarios tipo empresas registrados en el sistema.		
Sprint N°: 4	**Prioridad:** Baja	**Estado:** Por implementar
Descripción: Como administrador, deseo visualizar los datos de los usuarios estudiantes – postulantes registrados en el sistema para llevar un control.		
Criterios de aceptación	Dado el despliegue del formulario para listar a las empresas cuando el usuario ingrese a este servicio, entonces, se listan a todos los usuarios de tipo empresa registrados en el sistema.	
	Dado el despliegue del formulario para listar a las empresas cuando el usuario ocupe este servicio en otros dispositivos con diferentes tamaños de pantalla, entonces, el formulario se adapta a cualquier tamaño de pantalla.	

HISTORIA DE USUARIO		
ID: AH4-04	**Días estimados:** 4	
Título: Registro de datos de roles de usuario		
Sprint N°: 4	**Prioridad:** Alta	**Estado:** Por implementar
Descripción: Como administrador, deseo visualizar los datos de los roles registrados en el sistema para llevar un control.		
Criterios de aceptación	Dado el despliegue del formulario para listar a los roles cuando el usuario ingrese a este servicio, entonces, se listan a todos roles registrados en el sistema.	
	Dado el despliegue del formulario para listar a los roles cuando el usuario seleccione a un rol, entonces, se habilitan las opciones para actualizar y eliminar el registro.	
	Dado el despliegue del formulario para listar a los roles cuando el usuario ocupe este servicio en otros dispositivos con diferentes tamaños de pantalla, entonces, el formulario se adapta a cualquier tamaña de pantalla.	

6.6 Anexo VI: Historias de Usuario del Sprint 5

<table>
<tr><td colspan="3">HISTORIA DE USUARIO</td></tr>
<tr><td colspan="2">ID: AH5-01</td><td>Días estimados: 15</td></tr>
<tr><td colspan="3">Título: Aplicación web progresiva (PWA).</td></tr>
<tr><td>Sprint N°: 5</td><td>Prioridad: Alta</td><td>Estado: Por implementar</td></tr>
<tr><td colspan="3">Descripción: Como usuario, deseo utilizar las funcionalidades que ofrece la tecnología PWA para visualizar la aplicación en diferentes dispositivos.</td></tr>
<tr><td rowspan="3">Criterios de aceptación</td><td colspan="2">Dado el despliegue de la aplicación cuando se configure el hosting de firebase, entonces, realizar el deploy de la aplicación.</td></tr>
<tr><td colspan="2">Dado el despliegue de la aplicación cuando se ingresa a la aplicación por medio del hosting de firebase, entonces, instalar la aplicación como si fuera una aplicación móvil.</td></tr>
<tr><td colspan="2">Dado el despliegue de la aplicación cuando tenga las características de una PWA y se encuentre en el hosting de firebase, entonces, consumir el servicio de bolsa de empleo.</td></tr>
</table>

6.7 Anexo VII: Diseño de modelo lógico de base de datos

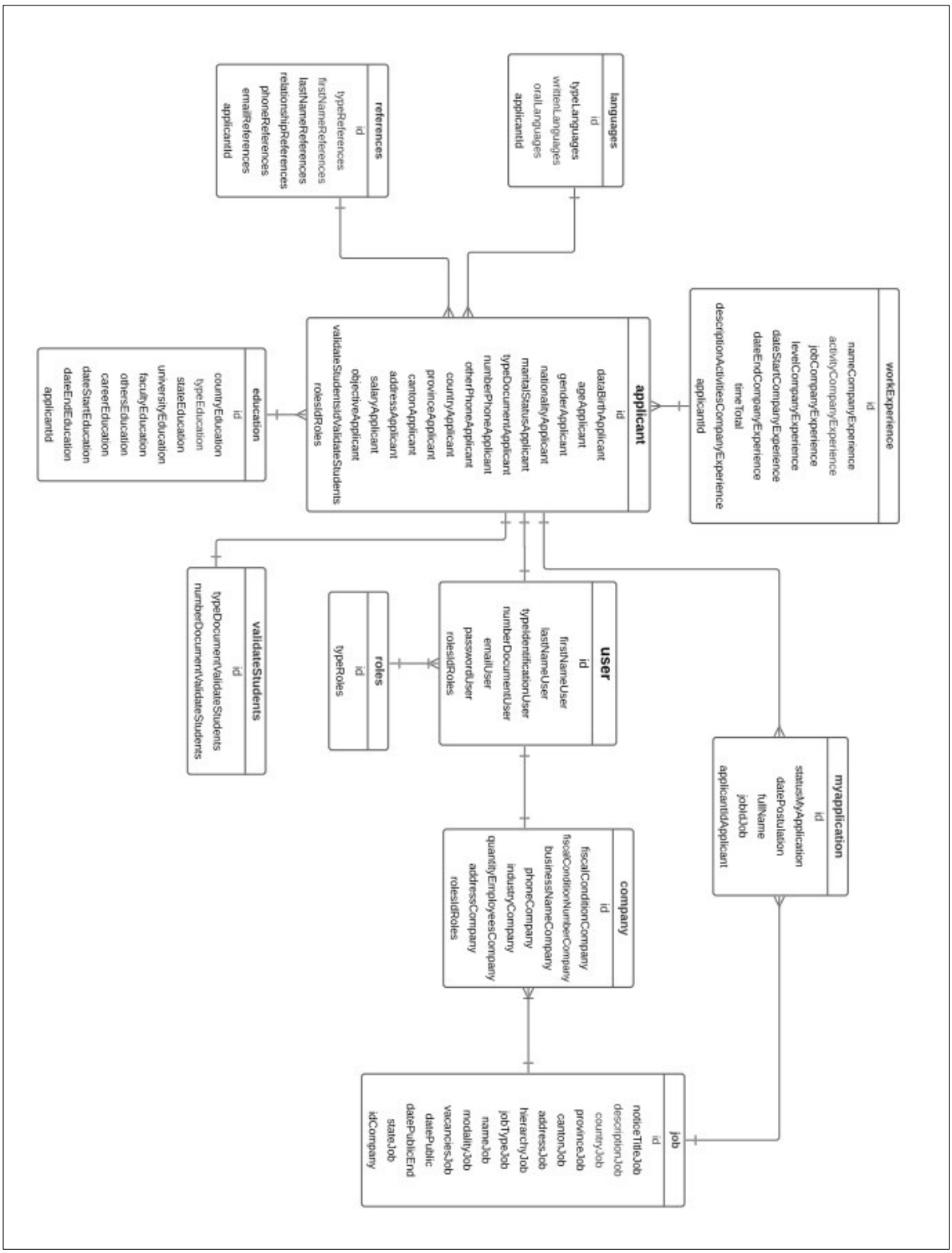


Figura 90. Modelo lógico de base de datos

6.8 Anexo VIII: Diseño de las interfaces de usuario final

Inicio de sesión

Figura 91. Diseño del formulario para realizar el inicio de sesión (Realizado en lucidchart)

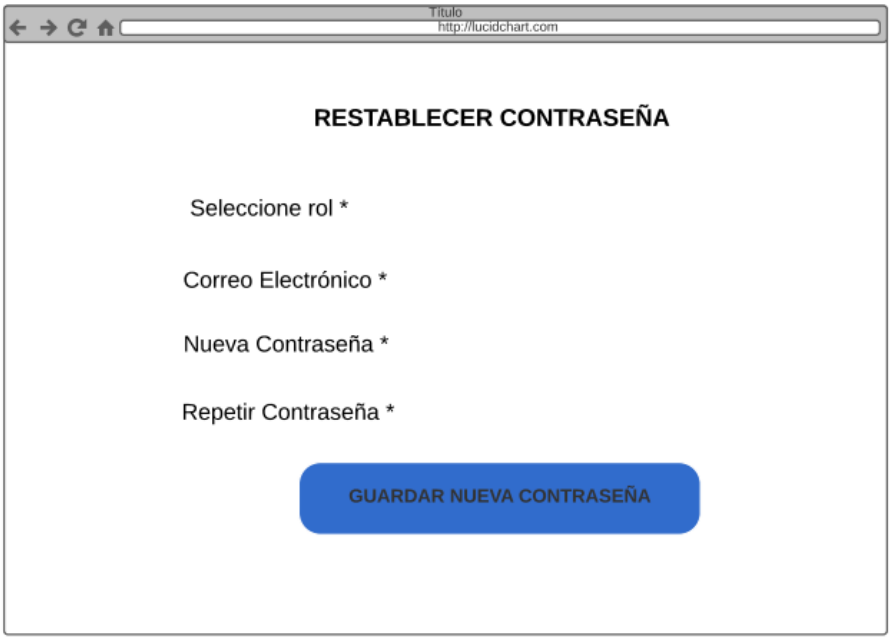


Figura 92. Diseño del formulario para restablecer contraseña (Realizado en lucidchart)

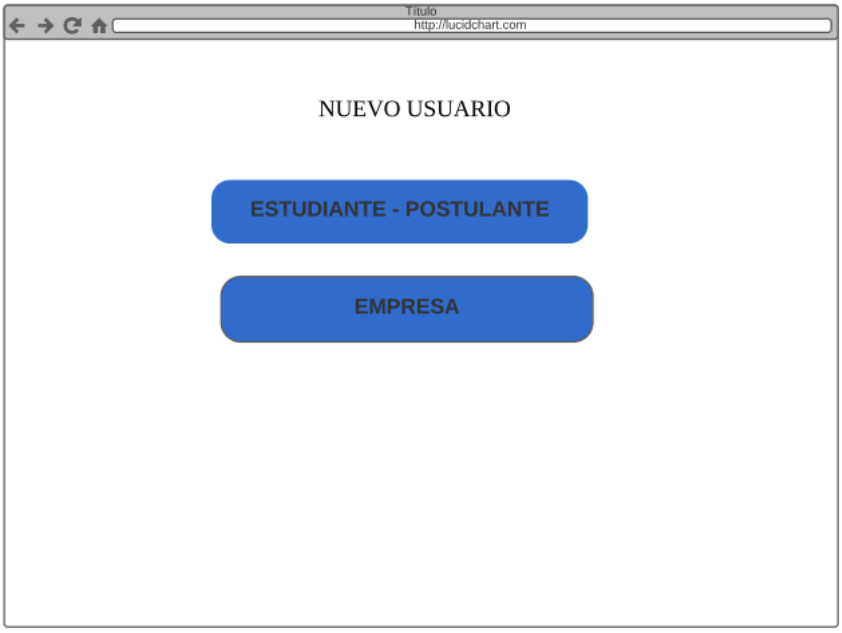


Figura 93. Diseño del formulario para seleccionar un usuario (Realizado en lucidchart)

Registro Usuario estudiante – postulante

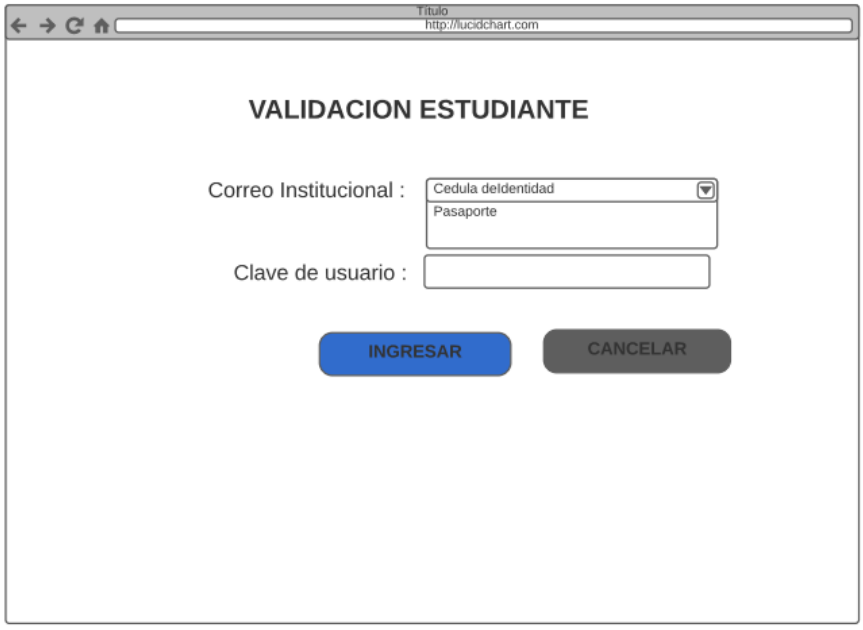


Figura 94. Diseño del formulario para validar usuario que pertenezca a la Facultad Ingeniera Sistemas de la EPN (Realizado en lucidchart)

Figura 95. Diseño del formulario para registrar estudiantes (Realizado en lucidchart)

Registro usuario empresa

Figura 96. Diseño del formulario para registrar empresa 1 (Realizado en lucidchart)

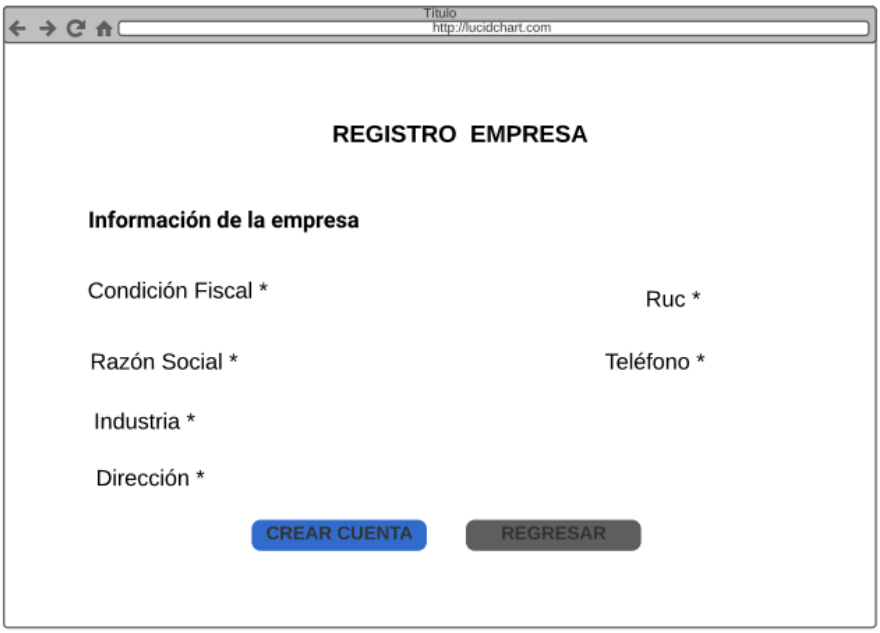


Figura 97. Diseño del formulario para registrar empresa 2 (Realizado en lucidchart)

Modulo usuario tipo estudiante – postulante

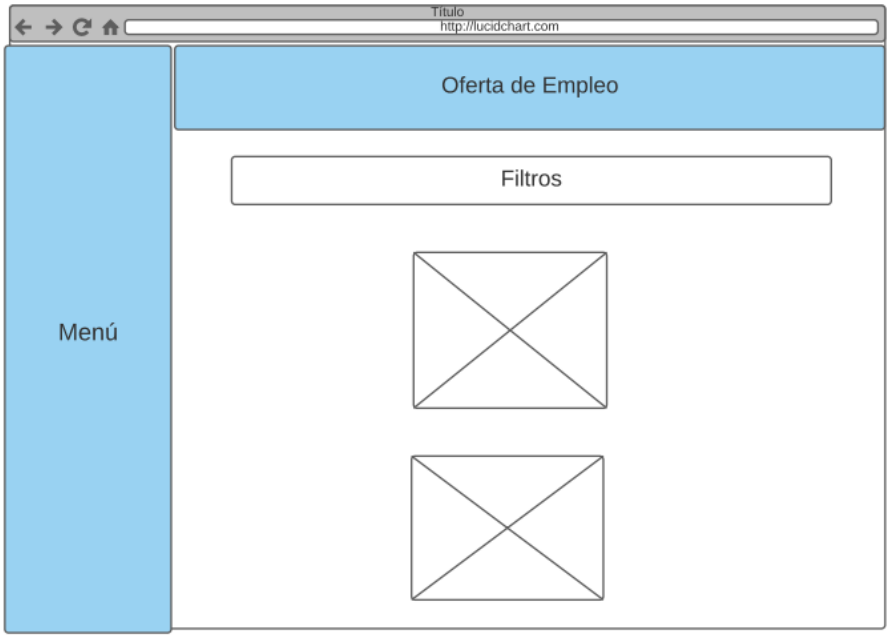


Figura 98. Diseño del formulario para obtener ofertas de empleo (Realizado en lucidchart)

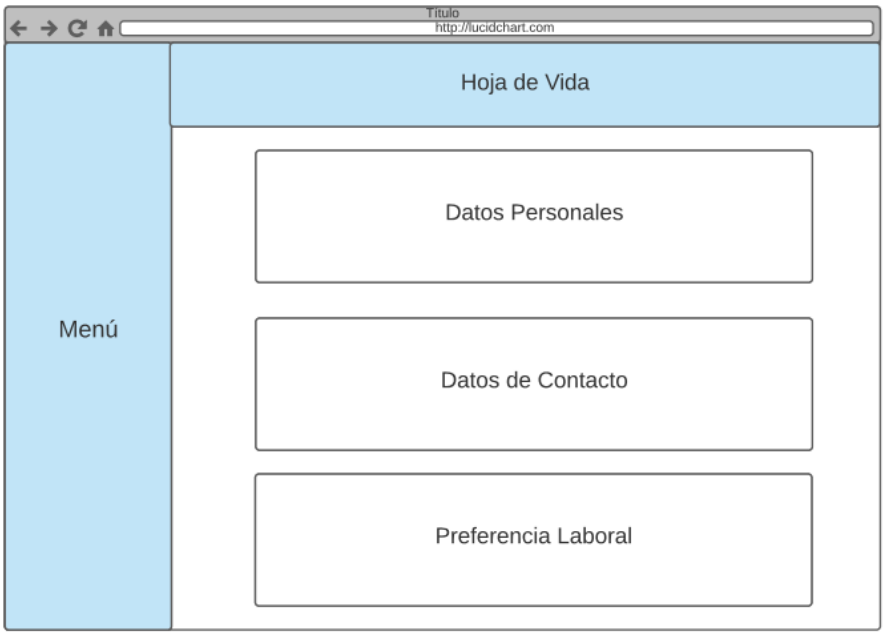


Figura 99. Diseño del formulario hoja de vida (Realizado en lucidchart)

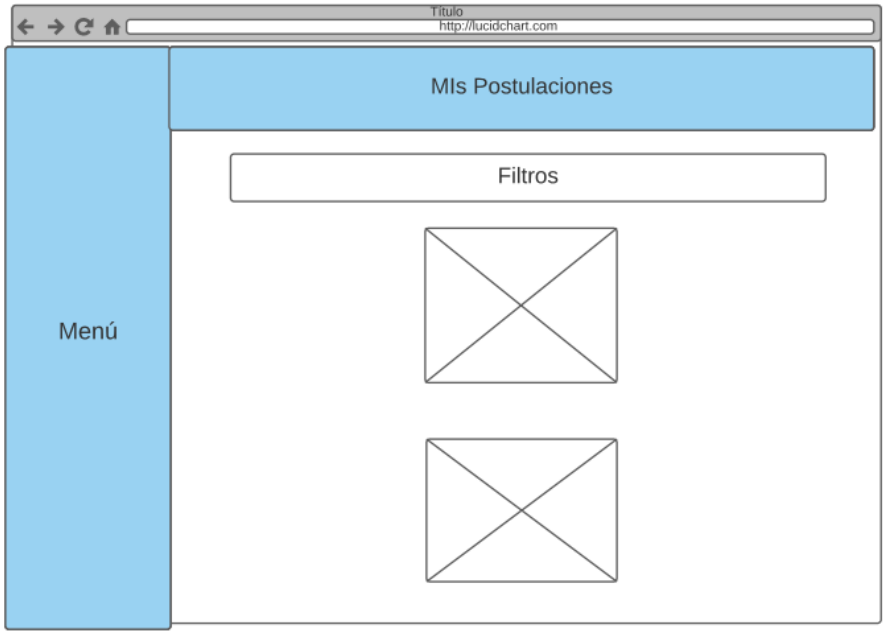


Figura 100. Diseño del formulario para obtener todas las postulaciones (Realizado en lucidchart)

Modulo usuario tipo empresa

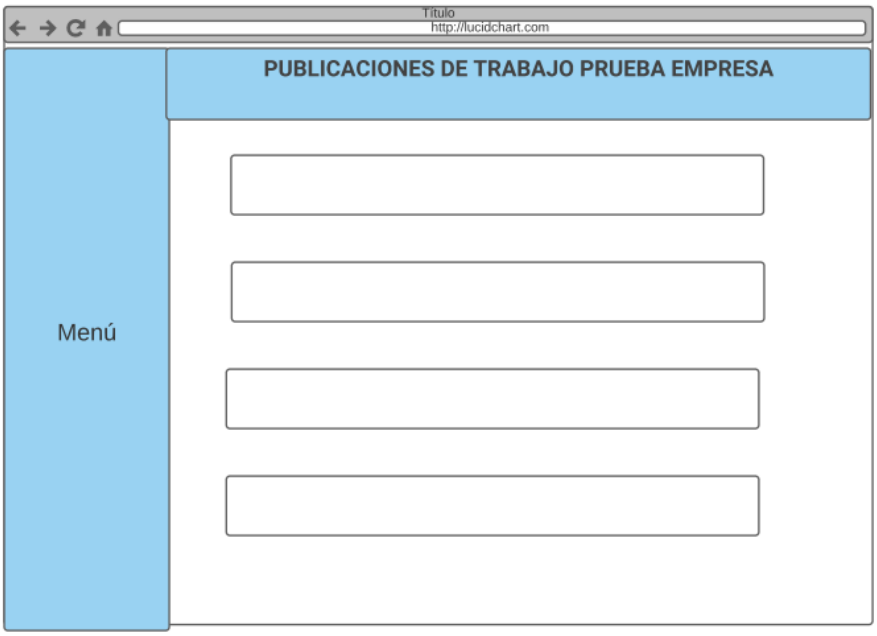


Figura 101. Diseño del formulario para obtener las ofertas de empleo publicadas (Realizado en lucidchart)

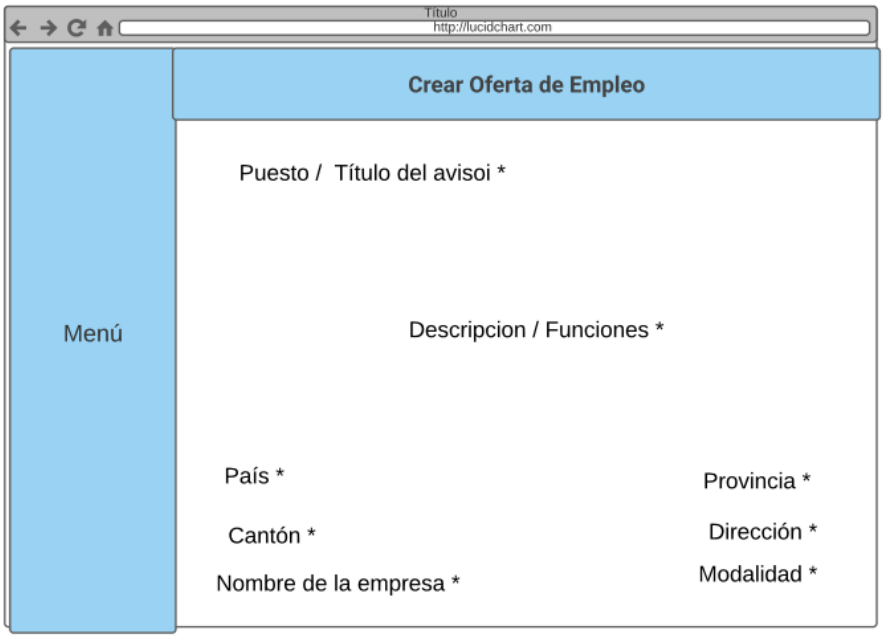


Figura 102. Diseño del formulario para crear las ofertas de empleo (Realizado en lucidchart)

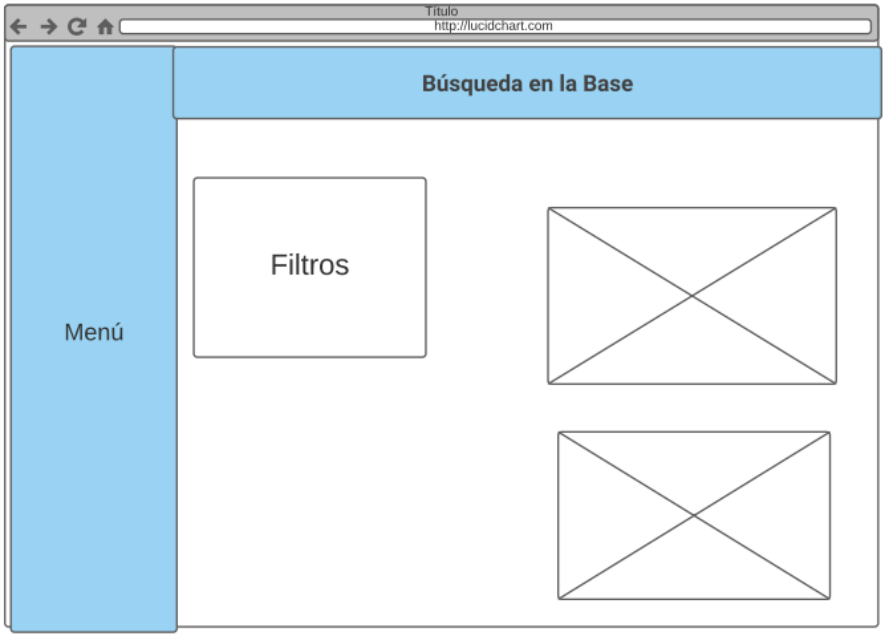


Figura 103. Diseño del formulario para obtener a los usuarios estudiante - postulante (Realizado en lucidchart)

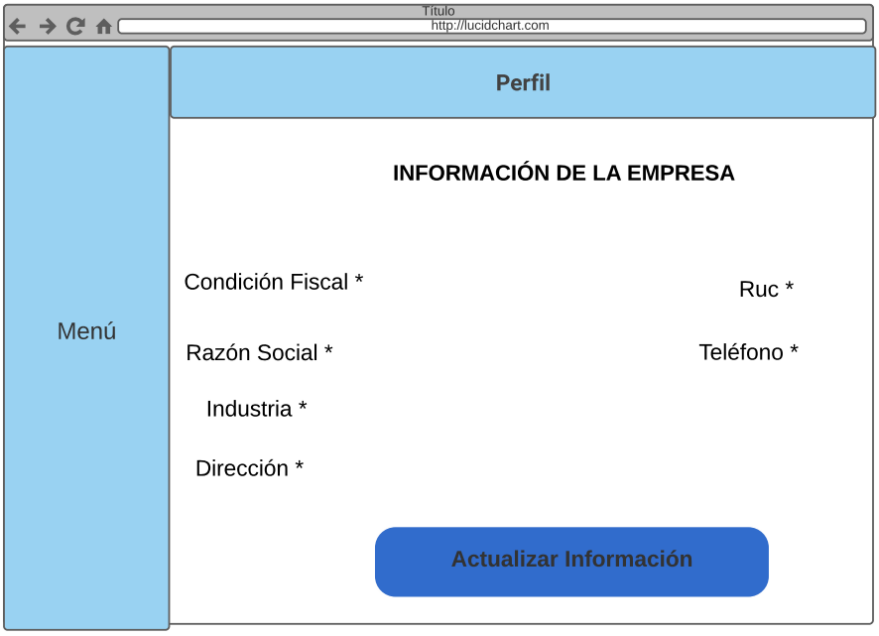
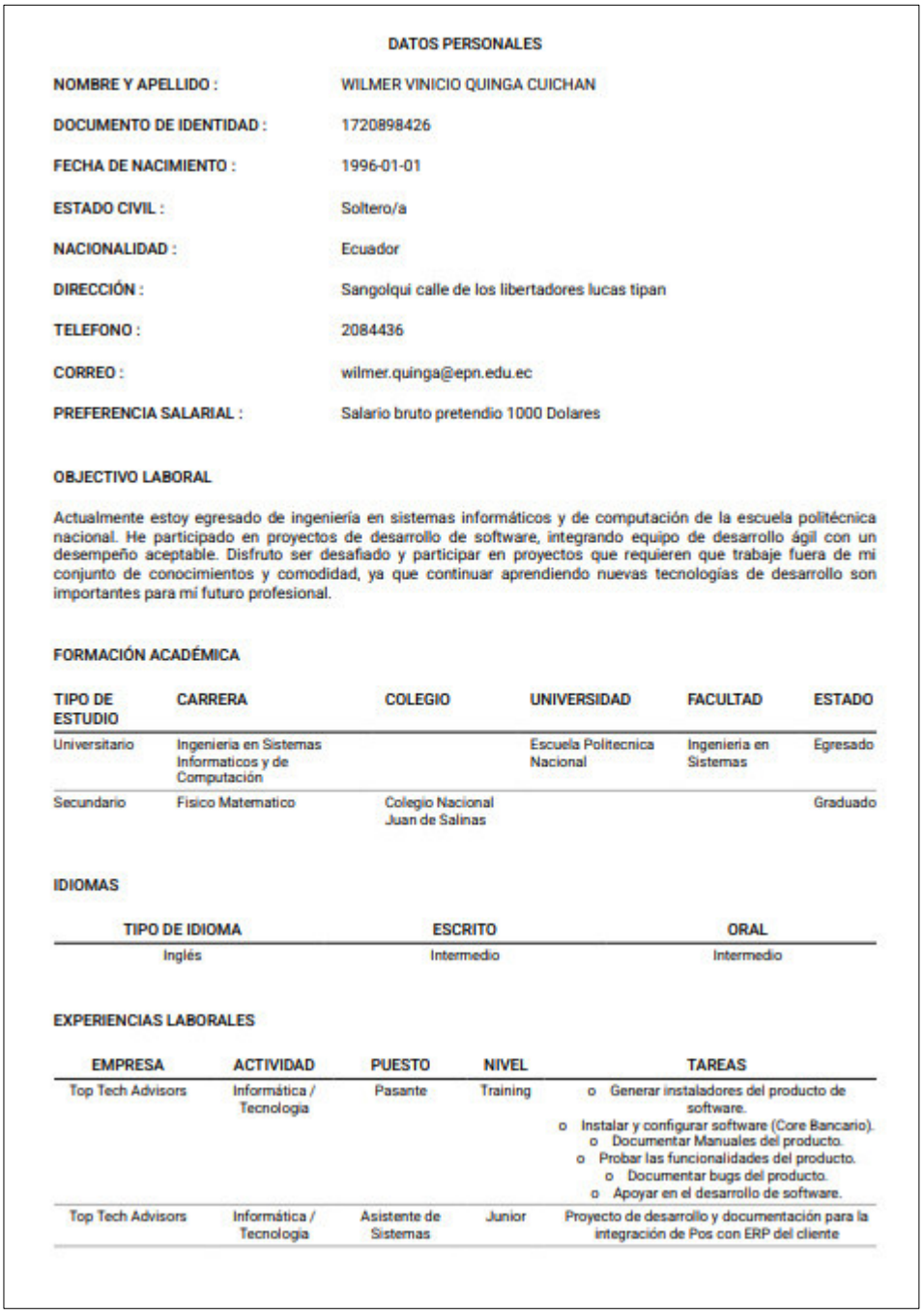


Figura 104. Diseño del formulario para actualizar información de la empresa (Realizado en lucidchart)

6.9 Anexo IX: Formato de hoja de vida



DATOS PERSONALES

NOMBRE Y APELLIDO :	WILMER VINICIO QUINGA CUICHAN
DOCUMENTO DE IDENTIDAD :	1720898426
FECHA DE NACIMIENTO :	1996-01-01
ESTADO CIVIL :	Soltero/a
NACIONALIDAD :	Ecuador
DIRECCIÓN :	Sangolquí calle de los libertadores lucas tipan
TELEFONO :	2084436
CORREO :	wilmer.quinga@epn.edu.ec
PREFERENCIA SALARIAL :	Salario bruto pretendio 1000 Dolares

OBJECTIVO LABORAL

Actualmente estoy egresado de ingeniería en sistemas informáticos y de computación de la escuela politécnica nacional. He participado en proyectos de desarrollo de software, integrando equipo de desarrollo ágil con un desempeño aceptable. Disfruto ser desafiado y participar en proyectos que requieren que trabaje fuera de mi conjunto de conocimientos y comodidad, ya que continuar aprendiendo nuevas tecnologías de desarrollo son importantes para mí futuro profesional.

FORMACIÓN ACADÉMICA

TIPO DE ESTUDIO	CARRERA	COLEGIO	UNIVERSIDAD	FACULTAD	ESTADO
Universitario	Ingenieria en Sistemas Informaticos y de Computación		Escuela Politecnica Nacional	Ingenieria en Sistemas	Egresado
Secundario	Fisico Matematico	Colegio Nacional Juan de Salinas			Graduado

IDIOMAS

TIPO DE IDIOMA	ESCRITO	ORAL
Inglés	Intermedio	Intermedio

EXPERIENCIAS LABORALES

EMPRESA	ACTIVIDAD	PUESTO	NIVEL	TAREAS
Top Tech Advisors	Informática / Tecnologia	Pasante	Training	o Generar instaladores del producto de software. o Instalar y configurar software (Core Bancario). o Documentar Manuales del producto. o Probar las funcionalidades del producto. o Documentar bugs del producto. o Apoyar en el desarrollo de software.
Top Tech Advisors	Informática / Tecnologia	Asistente de Sistemas	Junior	Proyecto de desarrollo y documentación para la integración de Pos con ERP del cliente

Figura 105. Formato hoja de vida 1

				permitiéndole que utilice el dispositivo en el pago de bines y servicios a través de tarjeta de crédito o débito.
SOLUCIONES TECNOLOGICAS LEADSOLUTIONS CIA. LTDA	Informática / Tecnología	Asistente / Ayudante/ Auxiliar / Instalador De Sistemas	Junior	• Documentación • Pruebas funcionales • Adaptación y optimización a Store Procedure (SP) en sql server. Para nuevas funcionalidades. • Sistema construido con backend en .net core 2.3 y frontend en angular.

REFERENCIAS PERSONALES

TIPO	RELACIÓN	NOMBRES	NÚMERO DE CELULAR	CORREO ELÉCTRONICO
Referencia Laboral	Compañera/o	Jefferson Tenelema	098693242	jtenelema@toptechadvisors.com
Referencia Personal		SergioPerez	097920379	chefperez01@hotmail.com

Figura 106. Formato hoja de vida 2

6.10 Anexo X: Encuestas Realizadas

6.10.1 Cuestionario de utilidad percibida estudiante – postulante

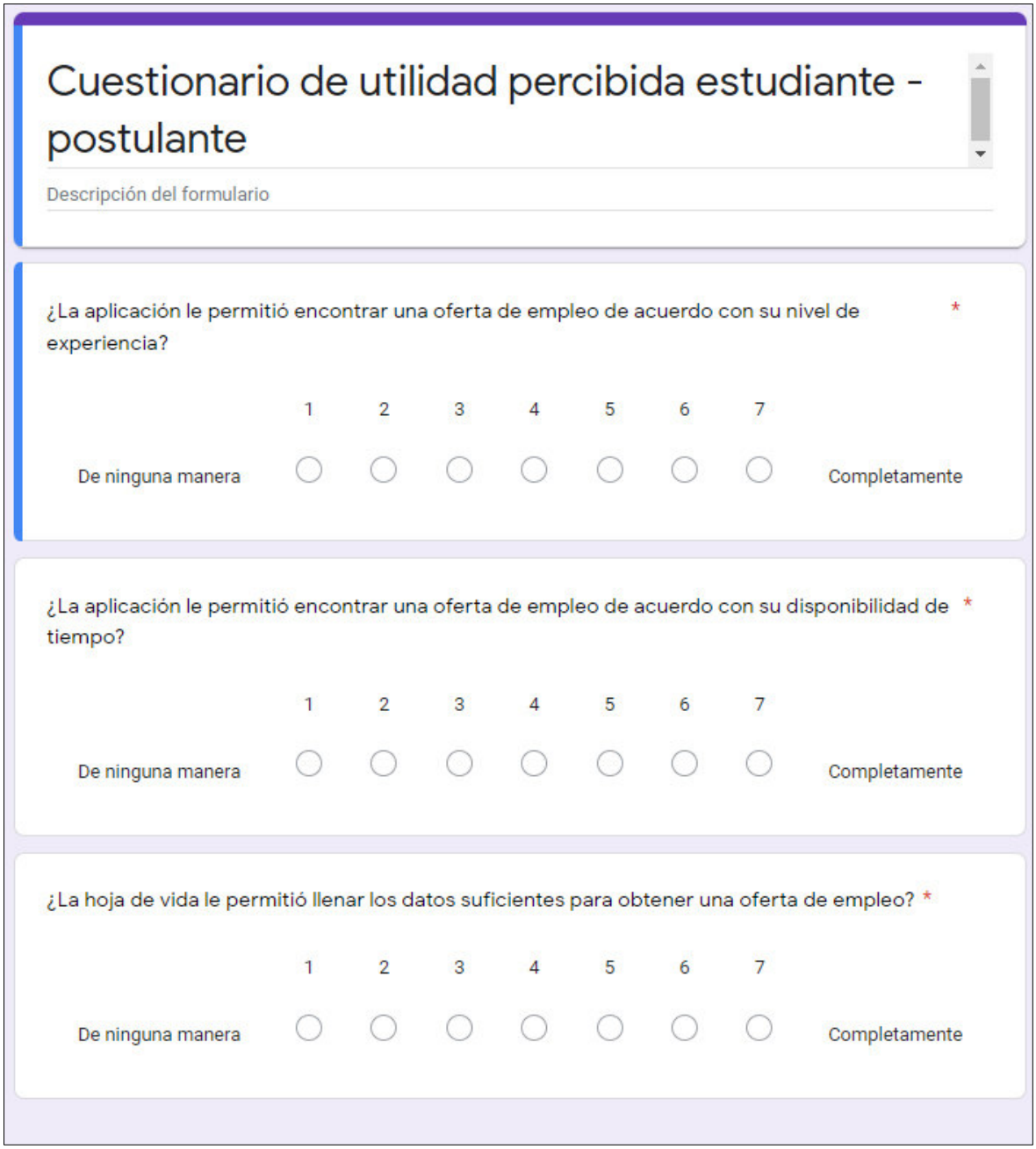


Figura 107. Cuestionario de utilidad percibida estudiante – postulante

6.10.2 Cuestionario de facilidad de uso percibida estudiante – postulante

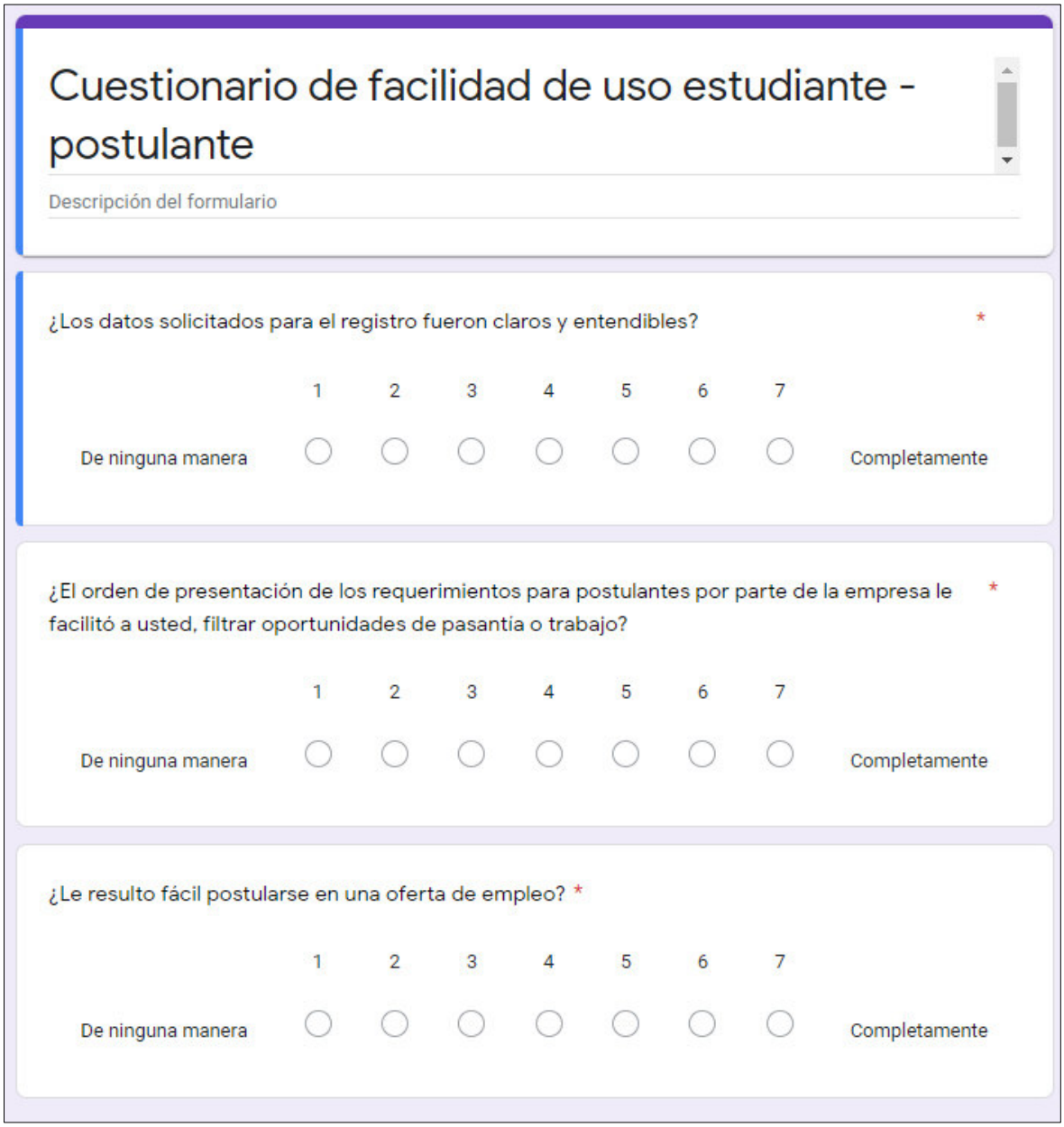


Figura 108. Cuestionario de facilidad de uso percibida estudiante – postulante

6.10.3 Cuestionario de utilidad percibida empresa

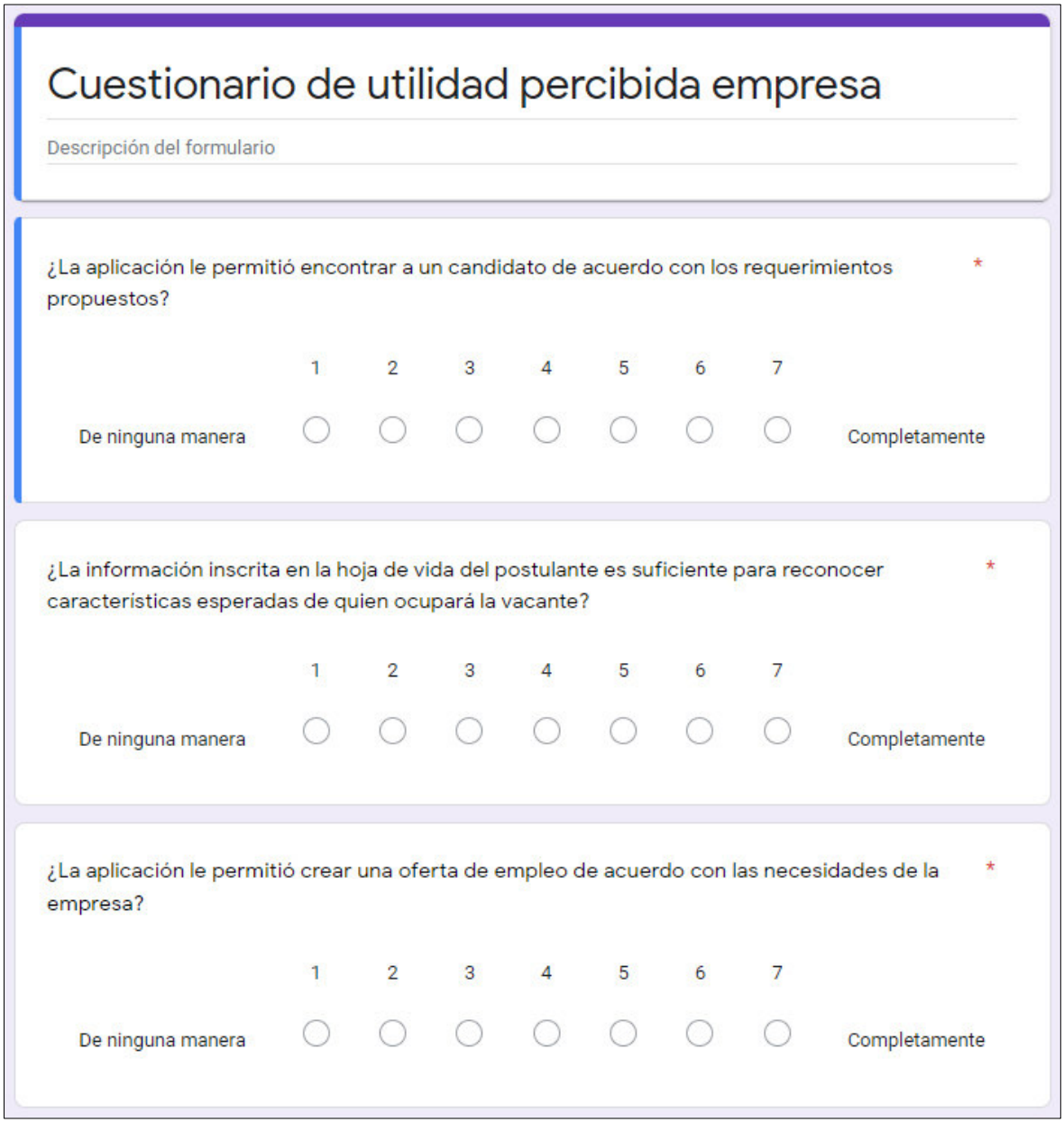


Figura 109. Cuestionario de utilidad percibida empresa

6.10.4 Cuestionario de facilidad de uso percibida empresa

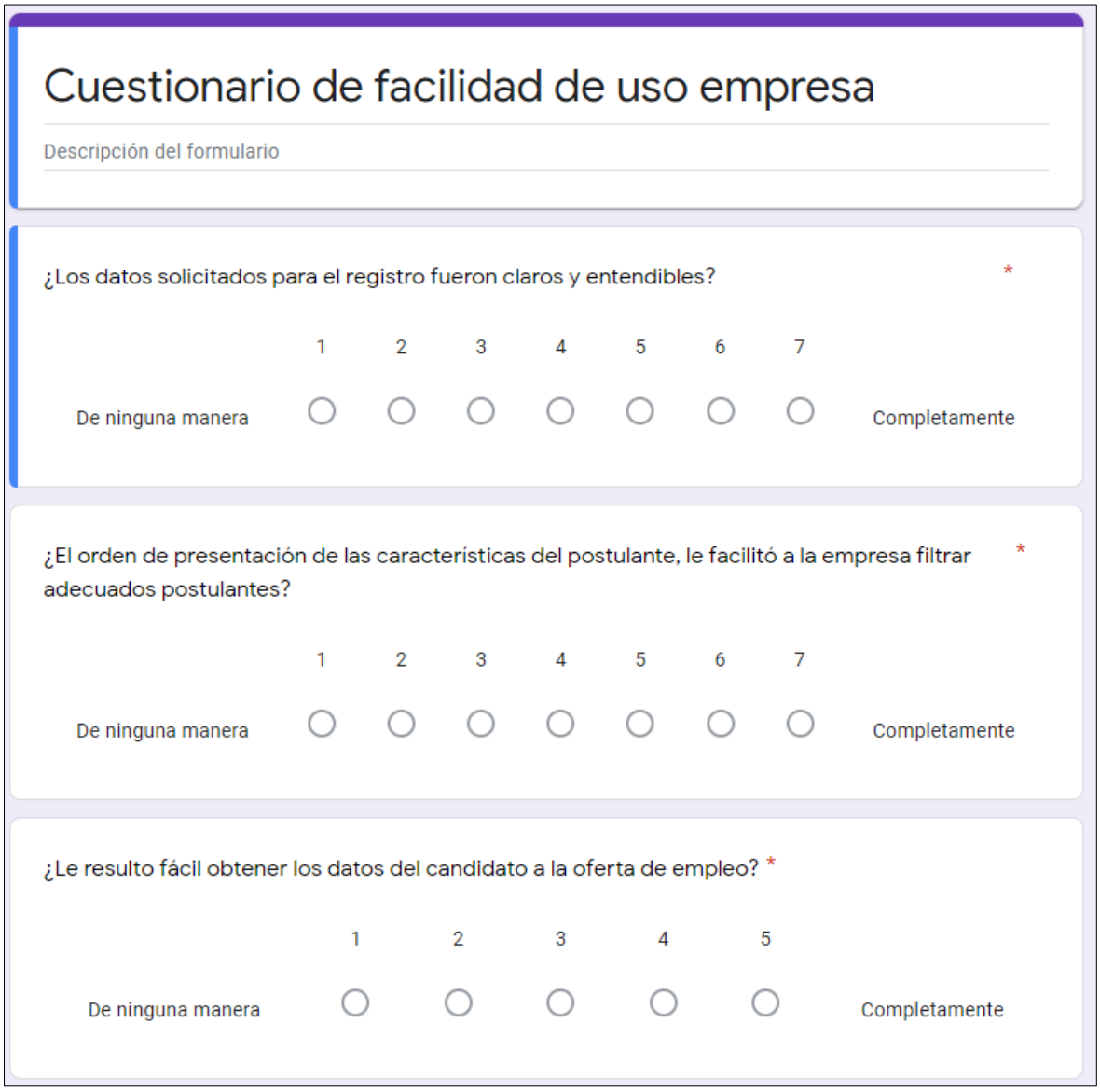


Figura 110. Cuestionario de facilidad de uso percibida empresa

Estudiante - Postulante

Facilidad de uso percibida

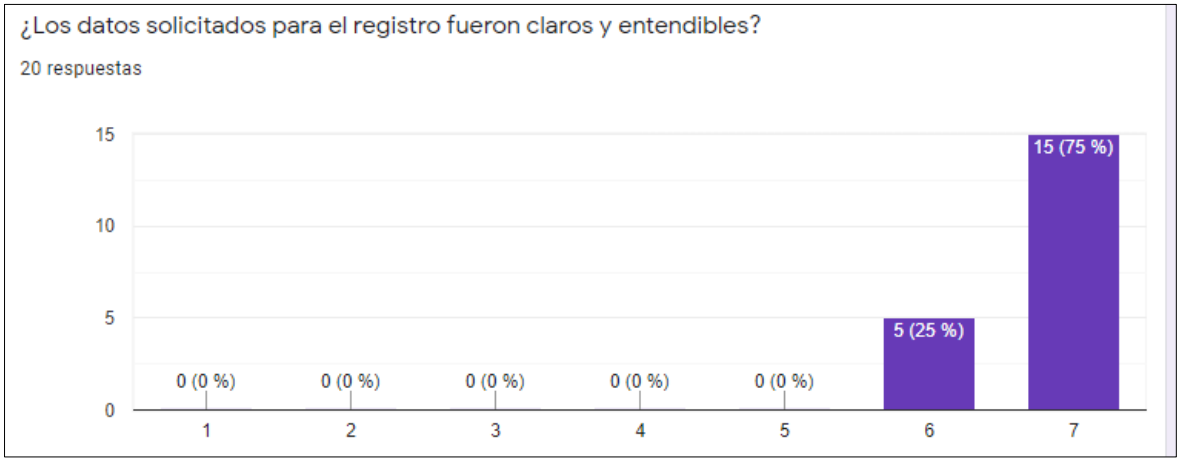


Figura 111. Respuesta a la pregunta 1 para facilidad de uso estudiante - postulante

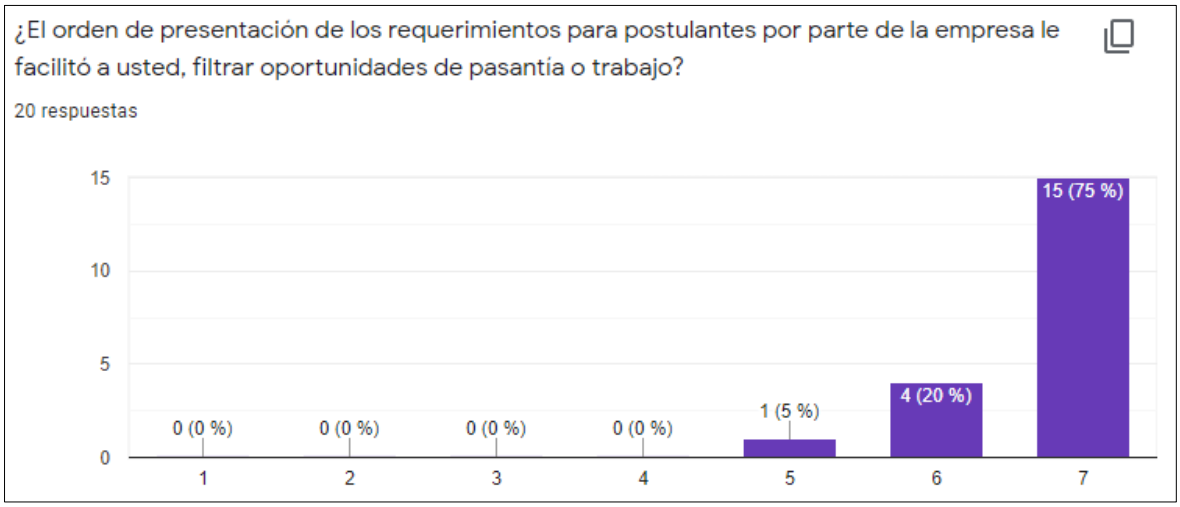


Figura 112. Respuesta a la pregunta 2 para facilidad de uso estudiante - postulante

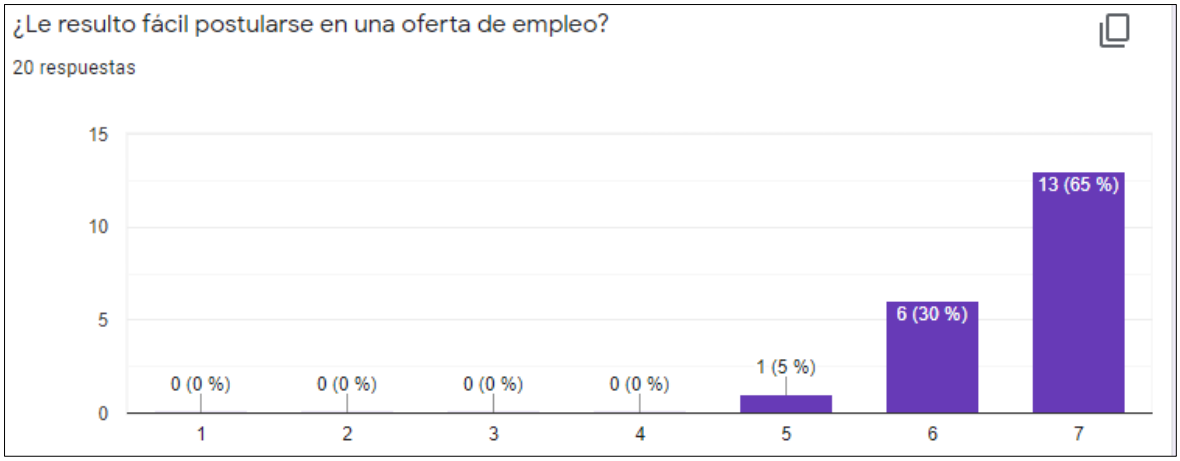


Figura 113. Respuesta a la pregunta 3 para facilidad de uso estudiante - postulante

Utilidad percibida

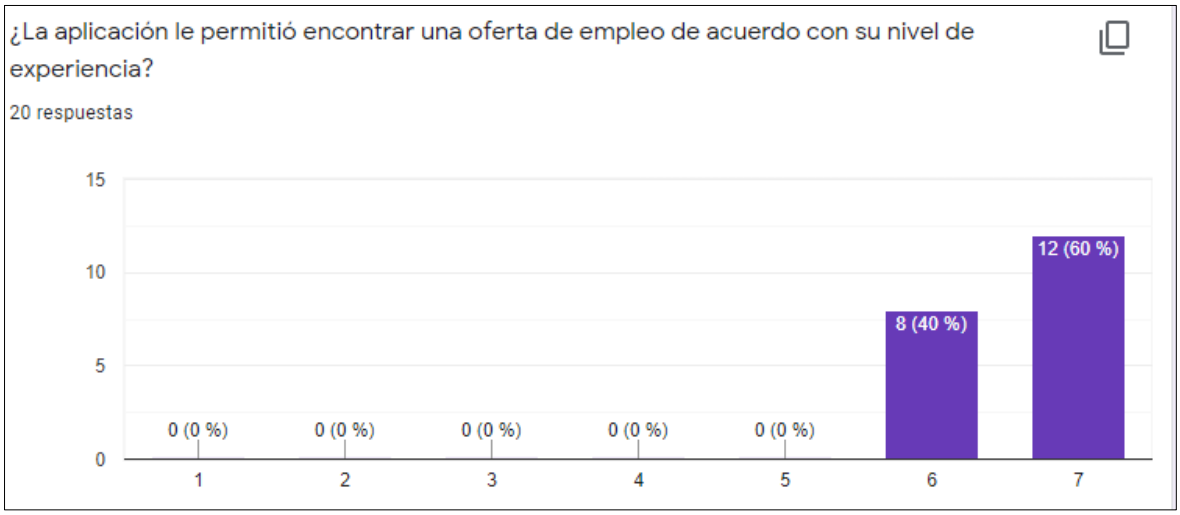


Figura 114. Respuesta de la pregunta 1 para utilidad percibida estudiante - postulante

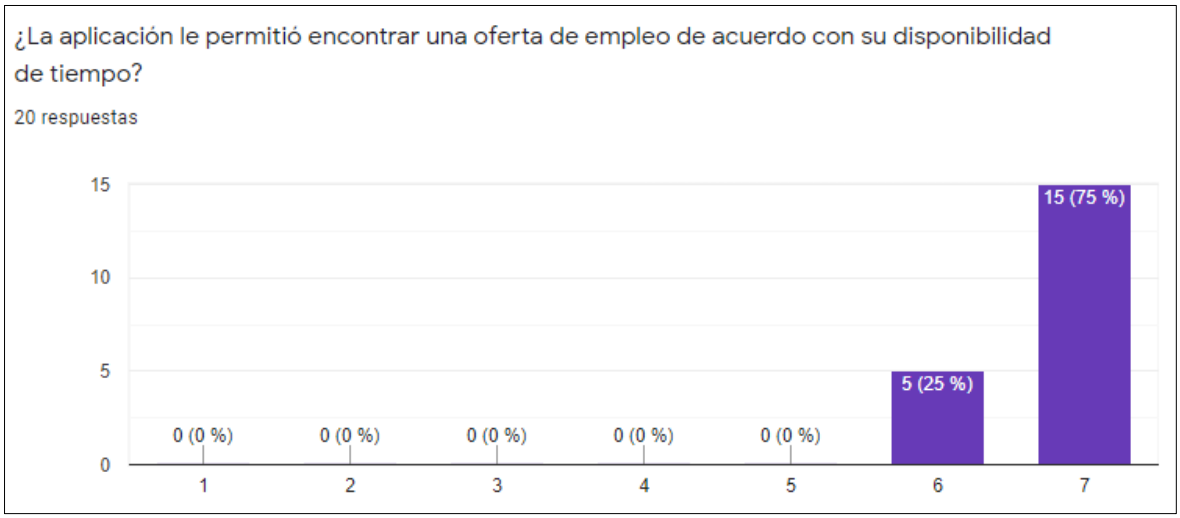


Figura 115. Respuesta de la pregunta 2 para utilidad percibida estudiante - postulante

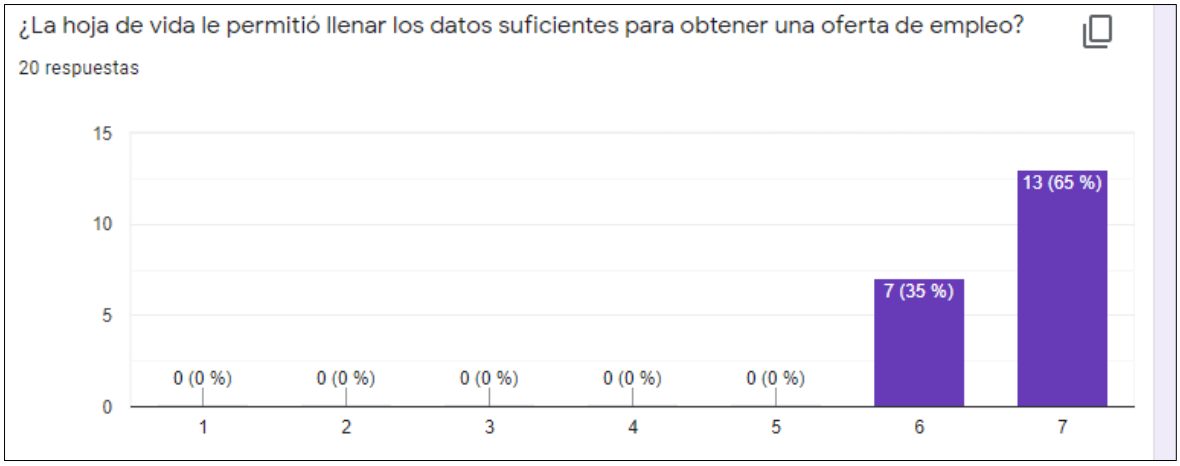


Figura 116. Respuesta de la pregunta 3 para utilidad percibida estudiante - postulante

Empresa

Utilidad percibida

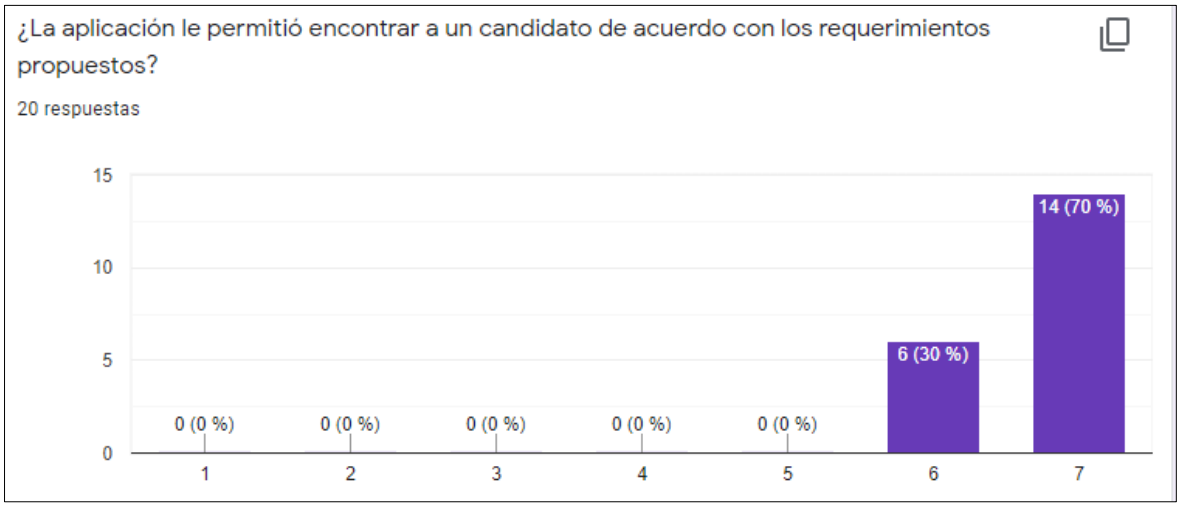


Figura 117. Respuesta de la pregunta 1 para utilidad percibida empresa

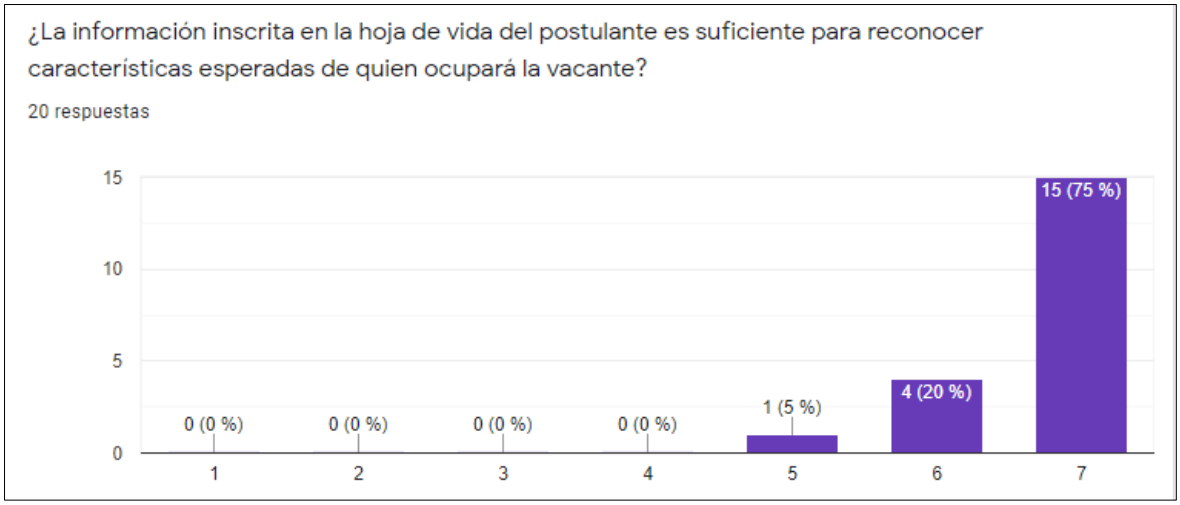


Figura 118. Respuesta de la pregunta 2 para utilidad percibida empresa

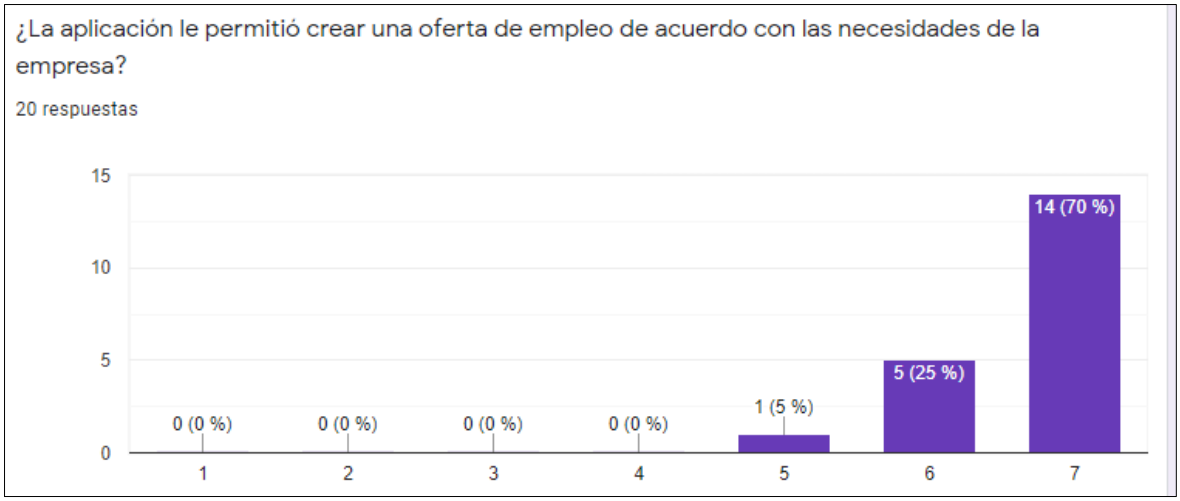


Figura 119. Respuesta de la pregunta 3 para utilidad percibida empresa

Facilidad de uso percibida

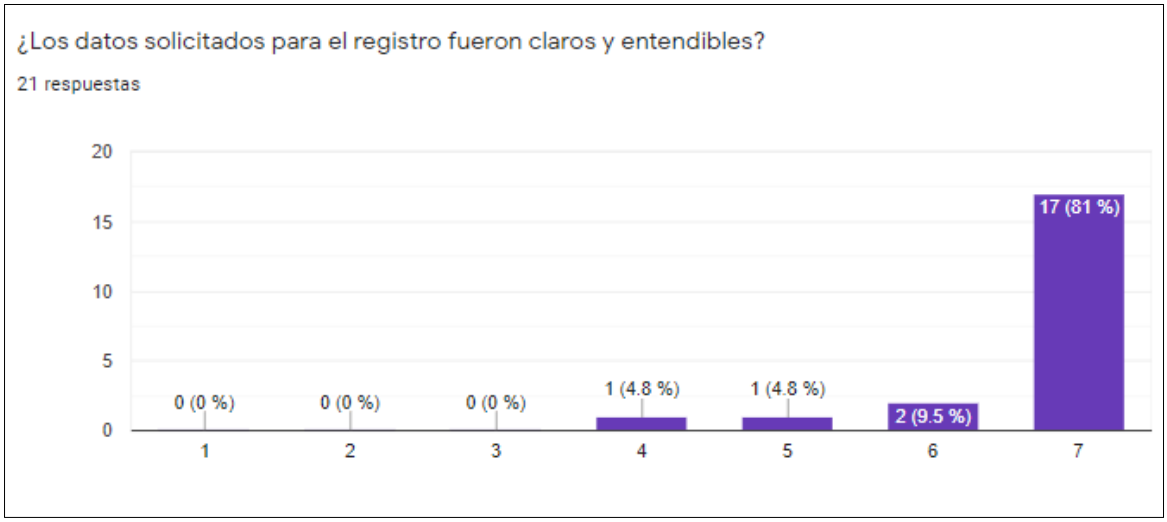


Figura 120. Respuesta de la pregunta 1 para facilidad de uso empresa

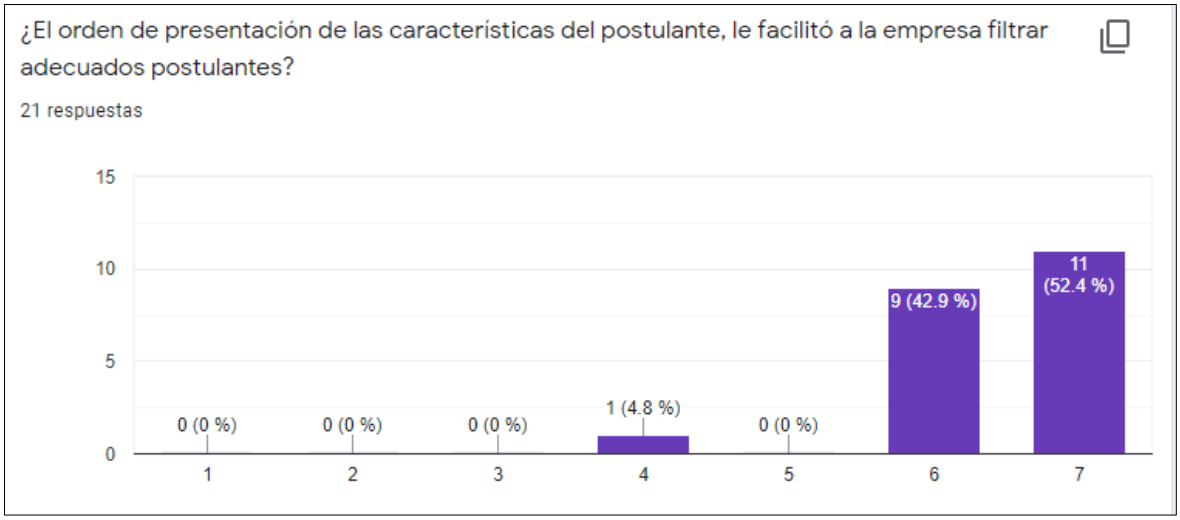


Figura 121. Respuesta de la pregunta 2 para facilidad de uso empresa

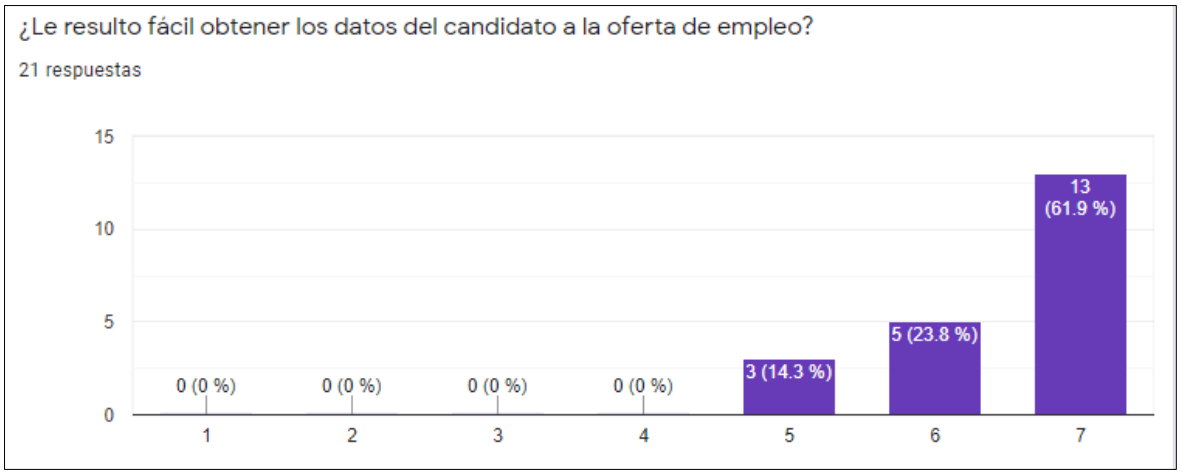


Figura 122. Respuesta de la pregunta 3 para facilidad de uso empresa.