Definición general del proyecto:

TalanSeeker es una herramienta para el manejo y tratamiento de datos de los talentos de la compañía (tanto contratados, como en proceso de entrevista).
La funcionalidad implementada añade un sistema de lectura de CVs (hojas de vida) que selecciona los datos más relevantes de los nuevos candidatos y es capaz de generar de forma automática CVs estandarizados para presentar la información disponible de cada uno.

Especificación de requerimientos:

TalanSeeker está desarrollada con Angular, NodeJS, Express y MongoDB, (MEAN) y complementada con la SDK de Firebase para el manejo de usuarios.
El sistema será desarrollado con las mismas tecnologías propuestas en el Stack de
TalanSeeker.

***Procedimientos* de instalación**:

Dentro de la carpeta raíz del proyecto, se requiere la instalación de de las dependencias necesarias para el funcionamiento tanto del front como del backend. Para ello, se debe ingresar a la carpeta llamada “backend” y ejecutar el comando “npm install” y se repite el mismo paso dentro de la carpeta “talanreader-web”. Lo cual instala todas las dependencias para el correcto funcionamiento del módulo.

Usuarios

Esta funcionalidad va dirigida para los consultores del área de talento humano, quienes tienen acceso a los curriculums de los nuevos colaboradores que ingresan a la empresa.

Objetivos

Objetivo General

- Desarrollar una interfaz fácil de aplicar para cargar y editar los curriculum de acuerdo al formato estándar y requerimientos de la compañía.

Objetivos Específicos

- Facilitar la extracción de información relevante de los curriculum de acuerdo al formato de la compañía.
- Entregar de forma automática y ágil un formato de CV estandarizado para presentar la información anteriormente extraída.
- Desarrollar una funcionalidad de fácil integración con la aplicación de TalanSeeker.
- Realizar la documentación completa del feature.

Procedimientos de desarrollo:

- Herramientas utilizadas:
    - backend: Se utilizaron varias librerías para el funcionamiento del módulo
        - Express: Librería utilizada para iniciar la aplicación web
        - Cors: Librería utilizada como protocolo de seguridad para las peticiones web
        - Nodemon: librería utilizada para la inicialización de la aplicación web desde el backend
        - Multer: librería utilizada para capturar el archivo pdf enviado desde el frontend
        - Pdf-parse: Librería utilizada para la extracción de la información de los archivos pdf. WARNING: esta librería requiere de la presencia de un archivo pdf dentro de una carpeta test/data/05-versions-space. Este archivo es un requerimiento para la correcta funcionalidad de la librería, más no hace parte del proyecto.
    - frontend:
        - Jspdf / jspdf-autotable: librería instalada sobre Angularjs, que permite descargar un archivo pdf de acuerdo al contenido representado en el frontend.

- Versiones:
    - Node: version 14.21.2
    - Angular CLI: version 15.1.3

- Planificación: una descripción global de la metodología utilizada para encarar y resolver el problema; por ejemplo: los pasos ejecutados a lo largo de la resolución del proyecto, a grandes rasgos.
- Planificación:

  Inicialmente, se planificó la funcionalidad del módulo y se subdividió en las siguientes funciones:

  - Extracción de la información: Identificamos el tipo de dato en el cual vamos a recibir la información del pdf.
  - Procesamiento de la información: Identificamos diferentes secciones dentro de los curriculums, por lo cual iniciamos una estandarización de la información, inicialmente se pone toda la información en minúsculas y adicional, se eliminan caracteres especiales como tildes o diéresis. Luego de esto se procede a realizar una tokenización y búsqueda de las secciones dentro de los tokens de la información.
  - Almacenamiento en archivo .JSON de la información: luego de tener la información dividida en secciones, se almacena en una estructura de datos de tipo objeto de javascript, y se almacena en un archivo de tipo .JSON.
  - Presentación de la información al usuario para la corrección: se presenta la información almacenada al usuario, en un formato prellenado, de tal forma que pueda modificar la información extraída por la aplicación.
  - Actualización del archivo json: Finalmente los datos corregidos son almacenados en el archivo .JSON
  - Descarga en PDF de la información: Como añadido, el usuario puede descargar un archivo pdf con la información que la aplicación extrajo, así como la información editada.

Arquitectura del sistema

El programa inicia su ejecución en el archivo index.js, encontrado en la carpeta backend.

- Index.js: Este inicia una aplicación web, esta configurada para que se ejecute en la ruta http://localhost:3001/api/consultor en el puerto 3001. Adicional, este modulo importa el módulo 'network.js', que contiene las rutas de peticiones.
- components/network.js: Este modulo maneja la petición post al endpoint /upload, recibe el archivo enviado en la petición, lo envía a la función que procesa la información y finalmente elimina localmente el archivo.
- components/controller.js: Este módulo hace uso de la librería de multer, con el fin de recibir el archivo enviado desde el frontend. La función que recibe el

archivo se ejecuta al momento de realizar la petición post en el endpoint /upload.

- reader/automate.js: Este módulo lee el pdf almacenado y lo pasa a la función encargada del procesamiento de la información que se encuentra en reader/process_data.js
- reader/process_data.js: Este módulo contiene las instrucciones para el procesamiento de la información y se encarga de llamar diferentes funciones, cada una encargada de procesar un segmento específico de la información. Finalmente, este módulo crea un objeto de javascript y lo almacena en un archivo .JSON.
    - Procesamiento de nombre, teléfono y correo electrónico - get_personal_data.js.
    - Procesamiento de experiencia académica - get_academic.js.
    - Experiencia laboral - get_work.js.
    - Habilidades - get_skills.js.

- reader/get_personal_data.js: Módulo encargado de la extracción de datos personales como el nombre del consultor, su correo electrónico y su teléfono celular del curriculum.
- reader/get_academic.js: Módulo encargado de la extracción de la sección o área académica del consultor.
- reader/get_work.js: Módulo encargado de la extracción de la sección o área de experiencia laboral del consultor.
- reader/get_skills.js: Módulo encargado de la extracción de skills o habilidades que refleja el consultor en su currículum.