UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una Herramienta de Generación de Currículums y Filtrado de Perfiles

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Adán Gurpegui, Javier

Tutor/a: Pelechano Ferragud, Vicente

CURSO ACADÉMICO: 2023/2024

RESUMEN

Este Trabajo de Fin de Grado se centra en la creación de una aplicación de escritorio que automatiza la generación de currículums a partir de perfiles de LinkedIn. La aplicación permitirá a los usuarios ingresar el nombre del perfil de LinkedIn y generar automáticamente un currículum completo utilizando la información disponible en dicho perfil, incluyendo detalles sobre educación, experiencia laboral, cursos realizados, entre otros.

Además, la aplicación ofrecerá funcionalidades de filtrado, permitiendo a los reclutadores buscar y seleccionar perfiles que cumplan con ciertos criterios específicos. Esto proporcionará una herramienta poderosa para las empresas en su proceso de reclutamiento de personal.

El desarrollo de la aplicación se llevará a cabo utilizando tecnologías como C# para el desarrollo de código y WPF para las interfaces, utilizando Visual Studio como entorno de desarrollo. Además, se implementarán scripts en Python para técnicas como web scraping, facilitando la extracción de datos de los perfiles de LinkedIn.

En resumen, este proyecto tiene como objetivo mejorar la eficiencia del proceso de reclutamiento al automatizar la creación de currículums y proporcionar herramientas avanzadas de filtrado basadas en perfiles de LinkedIn.

Palabras clave: LinkedIn, currículum, web scraping, aplicación de escritorio, automatización, filtros.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

CAPÍTULO 1
Introducción

LinkedIn es una **red social** fundada en diciembre de 2002 y lanzada en mayo de 2003. Está orientada al **uso empresarial, negocios y empleo**, con el objetivo de conectar a profesionales de diversas industrias para ayudarlos a establecer contactos, buscar empleo, reclutar talento y compartir conocimientos. Cada usuario tiene su **perfil** en el que libremente revela su **experiencia laboral, destrezas**, etc.

Sus principales características son la oportunidad de crear **perfiles profesionales**, establecer conexiones con distintos tipos de gente (amigos, compañeros de clase o de trabajo, profesionales del sector, etc.), **buscar empleo** a raíz de las ofertas que distintas empresas publican en la plataforma, tomar cursos en línea en distintas áreas, publicar contenidos e incluso debatir con otros usuarios.

LinkedIn es una **herramienta** que se ha convertido en **esencial** para **gestionar tu carrera**, ya que a través de la plataforma distintos reclutadores pueden contactarte para ofrecerte un mejor trabajo, tú mismo puedes optar a **nuevas oportunidades de trabajo** revisando las distintas ofertas que se publican, etc.

En resumen, LinkedIn es una **plataforma esencial** en el mundo profesional actual que prácticamente cualquier persona en el mercado laboral debería de tener y así poder gestionar su carrera, networking y aprendizaje continuo.

En cierta manera LinkedIn es una forma de tener un **currículum digital** en el que añades tu **experiencia laboral, educación, habilidades, logros**, etc. pero aun así cuando optas a distintas ofertas de trabajo un CV convencional sigue siendo necesario. Por ello, ¿por qué no generar un CV automáticamente a partir de tu propio perfil de LinkedIn?

Además, numerosas empresas usan LinkedIn para **reclutar nuevos talentos** en caso de encontrar un perfil que encaje con lo que buscan, por lo que ¿por qué no proporcionar una **herramienta** para **filtrar perfiles** y mostrar aquellos que cumplan unos filtros previamente dados?

1.1 Motivación

La idea de este TFG surgió debido a que el fin de mi grado universitario se acercaba, y pensando ya en comenzar mi carrera profesional, decidí crearme LinkedIn y completarlo con la información apropiada útil para este proceso de búsqueda de mi primer trabajo.

A partir de un CV anterior y con los cambios que debería introducir, empecé a rellenar la información que LinkedIn te permite en tu perfil, como son los estudios cursados, experiencia laboral, habilidades, etc.

Tras esto, empecé a navegar por la plataforma siguiendo perfiles interesantes y creando una red de contactos, a la vez que me creaba un nuevo CV ya que ofertas externas a LinkedIn me lo pedían.

Este proceso me pareció complicado, y en mi cabeza empezó a rondar la idea de **automatizar** la creación de un CV a partir del perfil de LinkedIn, ya que además en ese momento mi información estaba al día. Es por ello que surgió esta idea de TFG, ya que no conocía ninguna idea similar y le vi un **potencial interesante**, además de un reto que quería afrontar.

En este momento me surgieron varias ideas para crear la **solución**:

- Una extensión de **Google Chrome** que hubiese que instalar y que desde el perfil deseado y al pulsar un botón, generase un CV.

- Una **página web** que tuviese la opción de añadir el nombre de usuario y a partir de ahí generar el CV, e incluso añadir otras funcionalidades.

El **problema** que encontraba con ambas soluciones web es que necesitaría tener unos **conocimientos** que no dominaba, por lo que quizás invertía demasiado tiempo en adquirirlos, y eso me preocupaba.

Al contactar con el tutor, además de esto me sugirió ampliarlo y tener en cuenta también a los **reclutadores**, dándoles la funcionalidad de **filtrar perfiles** según características que pudiesen encontrarse en el perfil de LinkedIn y de esta forma facilitar esta función, y al comentarle mi preocupación, me sugirió la opción de crear una **aplicación de escritorio** con alguna tecnología que ya conociese. Esta opción me pareció la más correcta, y para añadir cierto factor de complejidad y forzarme a conocer algo nuevo, decidí usar WPF para la presentación de las interfaces.

En resumen, decidí usar C# ya que estaba familiarizado con el lenguaje gracias a proyectos que había tenido que desarrollar antes, pero en lugar de usar Windows Forms como ya había hecho previamente, ahora usar WPF y de esta manera hacer también una aplicación más potente gráficamente.

1.2 Objetivos del TFG

El principal objetivo del proyecto es desarrollar una **versión funcional** del mismo y ver si sería viable en un futuro. Por ello, me parece conveniente separar **objetivos funcionales** y **no funcionales**.

Objetivos funcionales

- Desarrollar una versión funcional del proyecto con sus dos principales funciones que pueda ser probada en cualquier momento, y dar **resultados aceptables**.

- Potencialmente, y si profesionalmente es viable, continuar su desarrollo y tratar de sacar **beneficio económico** de su implementación. También se valoraría qué haría falta para dar el salto al **mundo profesional** y el modelo de negocio a seguir.

Objetivos no funcionales

- Desarrollar una **interfaz potente e intuitiva** dominando WPF y sacando el máximo partido posible, además de continuar con la familiarización con C#, lenguaje utilizado en el ámbito profesional frecuentemente.

- Proporcionar una **solución útil** que pueda servir a mucha gente para hacer frente al problema encontrado.

1.3 Estructura del TFG

Esta memoria de TFG consta de doce secciones principales que a continuación se resumen brevemente para explicar qué se puede encontrar en cada una de ellas:

- En la primera sección encontramos la **Introducción** (sección actual) donde se da el contexto de la idea del proyecto, motivación y objetivos del TFG.

- En la segunda sección se hará un repaso del **Contexto tecnológico actual** para dar contexto y valorar otros productos que puedan estar relacionados con el proyecto actual, así como algunos detalles de los mismos.

- En la tercera sección hablaremos del **Proceso de software** aplicado en el desarrollo del proyecto basándonos en los conocimientos previamente adquiridos en el grado. Se dará una explicación de los distintos procesos existentes como introducción de esta sección, para después tratar en profundidad el elegido en el proyecto.

- En la cuarta sección se hará un repaso a la **Especificación de requisitos**. Comenzará con una explicación para continuar hablando de las técnicas aplicadas en el proyecto y finalizando con la explicación de los requisitos encontrados, tanto funcionales como no funcionales.

- En la quinta sección se presentará el **Modelado conceptual** de la aplicación. Para esta sección se han elegido una serie de diagramas que se presentarán dando una explicación de los mismos para después mostrar el caso concreto del diagrama aplicado en el proyecto.

- En la sexta sección encontraremos el **Diseño de la herramienta** que estará dividida en 3 subsecciones. La primera de ellas se usará para tratar los aspectos técnicos, como son las herramientas utilizadas y la arquitectura utilizada. Después se tratará el diseño, tanto desde el punto de vista de la interfaz como de las entidades, para finalizar hablando de los patrones de diseño utilizados.

- En la séptima sección se tratará la **Implementación**. Se hablará de las distintas tecnologías utilizadas así como de decisiones técnicas tomadas para la implementación de distintos aspectos en el proyecto.

- En la octava sección hablaremos de las **Pruebas y la calidad de software**. Se explicarán distintos conceptos relacionados y las técnicas aplicadas en la aplicación.

- En la novena sección hablaremos del **Mantenimiento y la gestión de software**. Nos centraremos en la herramienta utilizada, Azure DevOps, explicando la herramienta y su uso en el proyecto.

- En la décima sección se mostrará la **Aplicación final**. De esta forma, se mostrarán las pantallas finales reflejando la evolución de la interfaz desde el prototipado según las necesidades que se han ido encontrando durante la implementación.

- En la penúltima sección se hablará de posibles **Futuras mejoras** que se pueden introducir en la aplicación en el caso de que se siga trabajando en ella en el futuro. Estas mejoras nacen de ideas de posibles funcionalidades a incorporar que surgían durante la implementación.

- En la última sección de **Conclusiones** se valorará el resultado obtenido de la aplicación final así como la experiencia personal obtenida.

CAPÍTULO 2
Contexto tecnológico actual

LinkedCV es una aplicación que busca cubrir un hueco que no cubre ninguna aplicación en la actualidad. Por tanto, es complejo compararlo con otras aplicaciones, pero si dividimos la aplicación según funcionalidades, técnicas de obtención de datos, etc. podemos hacer comparaciones.

Durante esta sección se hablará de distintas **herramientas existentes** que han sido utilizadas o no, así como las distintas razones. Esto es importante ya que permite entender las razones de ciertas **decisiones** tomadas a la hora de crear la aplicación.

Para explicar todo esto, se dividirá el capítulo en dos subsecciones principales:

- La primera de ellas servirá para hablar de las distintas opciones existentes para **obtener** la **información de un perfil**, función necesaria para la funcionalidad de generar el currículum.
- En la segunda se hablará del **filtrado de perfiles** y de la opción elegida, correspondiente a la funcionalidad de filtrar los perfiles.

2.1 Obtención de la información del perfil

LinkedCV genera el currículum a partir de la información que se encuentra presente en los perfiles de LinkedIn del usuario. Para acceder a estos perfiles, la aplicación ofrece varias maneras enfocadas según el tipo de persona que seas: si eres un reclutador, te interesará **buscar un perfil** según unas características, pero si quieres generar el CV de un perfil concreto (el tuyo quizás) no hará falta filtrar perfiles, sino **navegar** directamente al perfil objetivo, por ello se puede acceder directamente proporcionando la URL del perfil.

Una vez se ha accedido al perfil deseado, llega la parte más importante, la **extracción** de la información, ya que si no no hay posibilidad de generar el CV.

Para esta función, se hizo un estudio de distintas herramientas que podrían ayudar al desarrollo del proyecto facilitando esta tarea.

2.1.1 linkedin_api

Linkedin_api fue la primera de las opciones que se encontró. Esta **API** ofrece una serie de métodos para navegar y obtener los datos de un perfil de LinkedIn.

Está diseñada para usarse en **Python**, y después de varias pruebas parecía una buena opción. Su instalación era como la de cualquier librería para Python, y su uso era sencillo.

En primer lugar, había que configurar la **inicialización** de LinkedIn (algo así como iniciar sesión) para posteriormente poder llamar a los métodos que necesitábamos.

```
import linkedin_api

# Reemplaza 'YOUR_API_KEY' y 'YOUR_API_SECRET' con tus credenciales
linkedin = linkedin_api.Linkedin('YOUR_API_KEY', 'YOUR_API_SECRET')
```

Imagen 1. Configuración linkedin_api.

Tras esta simple inicialización ya se podía hacer usos de los métodos. De hecho, había un método que directamente traía toda la información de un perfil dado. Este método era el método get_profile(user). Esta solución parecía buena, pero rápido aparecieron los primeros **problemas** hasta que finalmente, debido a que la API se encontraba **desactualizada**, se decidió buscar otras opciones mejores.

2.1.2 Opciones de pago

Una vez descartada la opción anterior, las siguientes opciones que podrían ser útiles en el proyecto eran de **pago**.

Había varias opciones de pago, con distintos precios pero que se quedaban fuera del proyecto ya que no se concebía la opción de pagar por este servicio. Algunas de estas opciones eran Prospeo, Proxycurl, lix o unipile.

Todas ellas son opciones similares que de una manera u otra ofrecen lo mismo pero al no contemplar la opción de pagar por algo así, se decidió **investigar** sobre distintas técnicas y métodos de extracción de información para cubrir este problema.

2.2 Filtrado de perfiles

Tras saber que no había ninguna API ni servicio que se pudiese usar para esta funcionalidad, se decidió investigar en la propia web de LinkedIn acerca de cómo filtrar perfiles según distintos parámetros.

La búsqueda consiguió su objetivo: se encontró la manera de **filtrar perfiles** dentro de la propia web de LinkedIn. Es una funcionalidad que ofrece LinkedIn pero es **compleja** de encontrar y posiblemente poco conocida, por lo que se podía explotar para hacer funcionar este requisito de la aplicación.

Por tanto, para esta funcionalidad, con las opciones del propio LinkedIn es suficiente para conseguir el objetivo buscado, solo hace falta **implementarlo** para que funcione correctamente.

CAPÍTULO 3
Proceso de software

Cuando desarrollamos un proceso software, existen distintas **metodologías** que seguir para mejorar el desempeño. Principalmente, hay dos grandes grupos de metodologías:

- **Clásicas o tradicionales**: se caracterizan por tener unas etapas de desarrollo que se realizan en un orden predefinido, completando cada etapa antes de pasar a la siguiente. También se realiza una documentación extensa y detallada con una planificación rígida, determinando elementos como el alcance, plazos o costos al principio del proyecto.

- **Ágiles**: este tipo de metodologías se realizan en ciclos cortos llamados iteraciones, consiguiendo un software funcional al final de cada ciclo. Los equipos tienen comunicación y colaboración continua, así como una fuerte integración de los stakeholders. Por último, son muy flexibles y se adaptan rápidamente a los requisitos y prioridades del cliente.

Por ello, para este proyecto he decidido elegir una **metodología ágil** ya que necesitaba una metodología **flexible** para gestionar correctamente cualquier posible cambio que pudiese aparecer y tener un producto funcional continuo para poder probarlo en cualquier reunión con el tutor.

Este tipo de metodología requiere cierta **disciplina** que gracias a los proyectos desarrollados en algunas asignaturas del grado he podido adquirir, por lo que sabía que no iba a ser difícil adaptarme a este requisito.

Dentro de las metodologías ágiles hay varias opciones entre las que se puede elegir. La primera gran distinción que me gustaría hacer es entre las metodologías ágiles estándar y las metodologías ágiles a gran escala. Estas últimas están adaptadas para proyectos grandes en empresas con muchos empleados o equipos. Para el proyecto en desarrollo, terminé eligiendo una metodología ágil estándar.

De entre las distintas opciones decidí escoger la metodología **Kanban**. Esta metodología se centra en la visualización del flujo de trabajo, limitando el trabajo en curso (WIP), promoviendo la mejora continua fomentando revisiones, y mejora del proceso mediante el análisis regular, mediante la gestión del flujo de trabajo teniendo un flujo constante de tareas.

Podemos encontrar los siguientes elementos clave en Kanban:

- **Tablero Kanban**: representa el flujo de trabajo mediante varias columnas, representando las tareas con tarjetas que van fluyendo de una etapa a otra.

- **Tarjetas Kanban**: representan unidades de trabajo conteniendo información relevante como descripción, fecha límite, etc.

- **Columnas**: Representan cada una de las distintas fases del proceso.

- **WIP**: Es la restricción de la cantidad de tareas en curso en una misma columna. El significado de las siglas es Work In Progress.

Las razones de esta elección se centran en la flexibilidad de Kanban para adaptarme a los distintos cambios que pudieran aparecer a lo largo del proyecto, la necesidad de tener una visión clara del estado

del trabajo con el uso de los tableros Kanban y mejorar el flujo del trabajo limitando el WIP. Por último, el tener una solución funcional en cualquier momento del desarrollo también me parece algo esencial para un proyecto de estas características.

Finalmente, a continuación, adjunto una muestra orientativa de un tablero Kanban:

Imagen 2. Modelo de tablero Kanban.

CAPÍTULO 4
Especificación de requisitos

La especificación de requisitos es la **primera** fase del desarrollo de la aplicación. En este capítulo se hablará de este proceso de una manera general, para dar contexto y explicar en qué consiste, para después aplicarlo para el caso concreto del proyecto.

Durante esta sección tenemos distintos objetivos:

- Explicar en **qué** consiste la especificación de requisitos, incluyendo las distintas fases de este proceso así como las distintas maneras de hacerlo.
- Explicar **cómo** se ha aplicado en el proyecto. Este proceso persigue los objetivos de definir las necesidades del cliente, establecer un acuerdo común, facilitar la planificación y el diseño y ayudar en la verificación y validación. Todos estos objetivos también se pretenden lograr en la definición de los requisitos que se hace en esta sección.

Este capítulo es **importante** ya que sirve como punto de partida del proyecto. Una vez se conocen las necesidades del usuario y los requisitos de la aplicación, se puede empezar a planificar el desarrollo y la implementación del proyecto, por lo que este capítulo será el punto de partida del mismo.

Para ello, en primer lugar explicaremos desde un punto de vista más teórico el proceso, con las distintas fases del mismo y las distintas maneras de definir los requisitos, para posteriormente explicar desde un punto de vista más práctico para definir los requisitos del proyecto en cuestión.

4.1 El proceso

La especificación de requisitos es una fase crucial en el desarrollo del software. Se define lo que el sistema debe hacer y las distintas restricciones que tendrá. El objetivo es capturar, analizar y documentar las necesidades y expectativas de los **stakeholders**. Es la base para las siguientes fases del desarrollo y ayudará a cumplir las expectativas y necesidades del cliente.

Este proceso consta de distintas fases:

- **Elicitación de requisitos**: esta fase se centra en la extracción de necesidades y preocupaciones de los stakeholders relacionadas con el sistema. En esta fase se pueden usar entrevistas, cuestionarios, observación, etc.
- **Análisis de requisitos**: en esta fase se evalúan y refinan los requisitos previamente elicitados para que sean completos, consistentes y realizables. Se pueden usar técnicas como prototipos, modelado de requisitos, etc.
- **Especificación de requisitos**: en esta fase los requisitos se documentan de manera detallada y formal. Se suelen dividir entre requisitos funcionales y no funcionales, pudiendo añadir casos de uso y diagramas para un mayor detalle.

- **Validación y verificación de requisitos**: en esta fase final, se asegura que los requisitos sean correctos y completos en relación a lo necesitado por los stakeholders. Las técnicas más comunes son la prueba de prototipos y las revisiones y auditorías.

Para la definición de requisitos existen distintas técnicas que pueden ser utilizadas, así como estándares que apoyan este proceso. A continuación, explico algunos de estas técnicas:

- **Historias de usuario**: consisten en descripciones breves de funcionalidades desde el punto de vista del usuario. Tienen un modelo "a seguir": "Como [tipo de usuario], quiero [acción] para [beneficio]". De esta forma, es más fácil entender el requisito desde el punto de vista del usuario y promueven la comunicación del equipo de desarrollo con los stakeholders.

- **Casos de uso**: consisten en descripciones de las interacciones del usuario con el sistema para lograr un objetivo. Tienen presente al actor, escenario, precondiciones, flujo de eventos, etc. y proporcionan un enfoque detallado, identificando todas las interacciones posibles ayudando a descubrir requisitos funcionales y no funcionales.

- **Escenarios y casos de prueba**: consisten en narrativas para describir cómo se comportará el sistema en situaciones particulares, con casos de prueba que detallan los pasos para verificar los requisitos. Están formados por la descripción del escenario, condiciones iniciales, pasos y resultados esperados. Ayudan a descubrir problemas antes de la implementación y a asegurar que una vez implementado, todo funcione como debería.

- **IEEE 830**: proporciona directrices para la creación de un documento de Especificación de Requisitos de Software, con introducción, descripción general y requisitos específicos. Esto ayuda a tener una estructura clara y estandarizada para una mayor facilidad de comprensión y comunicación entre todas las partes. En la siguiente sección, donde se definirán los requisitos, este estándar será utilizado.

4.2 Aplicación en el proyecto

A continuación, se hablará de los requisitos concretos del proyecto utilizando el **estándar IEEE 830** como guía para ello.

4.2.1 Introducción

El propósito de esta especificación de requisitos es dar una visión clara de las necesidades del usuario que la aplicación deberá cubrir de manera que la experiencia del usuario con la aplicación sea lo más positiva posible. Se usará sobre todo como guía en el desarrollo, pero también puede ser utilizado por todos aquellos que lean este documento para tener una visión más clara de lo que se espera de la aplicación.

El sistema se llamará **LinkedCV** porque básicamente consiste en, de cierta manera, añadir a LinkedIn la posibilidad de generar CV ya sea de un perfil encontrado mediante su url, o un perfil que cumple con unos filtros previamente dados. Se basa en dos funcionalidades clave:

- La **generación automática de CV** a partir del link a un perfil de LinkedIn.

- Aplicar **filtrado** sobre perfiles para encontrar el perfil que mejor se amolde a lo que el reclutador está buscando.

El principal objetivo es que el usuario obtenga ayuda de la aplicación de manera que le sea más fácil determinada necesidad: generar CV o buscar un perfil para una oferta de trabajo.

A continuación, se dará la descripción general del producto, apoyándose en la perspectiva del mismo, funciones, características de los usuarios, restricciones, suposiciones, etc. así como también se darán los requisitos que han sido detectados como necesarios para este proyecto.

4.2.2 Descripción general

El producto a desarrollar se trata de una **herramienta complementaria** a LinkedIn, por tanto, de cierta manera depende de LinkedIn, ya que si no no podría funcionar. Sin embargo, es una aplicación independiente de LinkedIn en el sentido de que no está ligada a él, a pesar de que haya que iniciar sesión en la página para que funcione.

Con la aplicación se pretende soportar la creación de distintos tipos de CV a partir de la información almacenada en LinkedIn para un usuario, así como permitir filtrar perfiles según unos parámetros que el usuario pueda rellenar en la aplicación.

Los usuarios para los que esta aplicación está enfocada son principalmente dos tipos:

- Usuarios que se encuentren **buscando trabajo** por lo que necesiten un CV, y ven buena opción inspirarse en su LinkedIn para crearlo.

- **Reclutadores** que busquen una persona con unas características determinadas para un puesto de trabajo.

Para la realización de la aplicación nos encontramos distintas restricciones que podrían entorpecer parcialmente el desarrollo de la aplicación. Es importante saber que LinkedIn, como medida de **seguridad**, para consultar la información de un perfil requiere tener una sesión iniciada, por lo que siempre habrá que iniciar sesión antes de empezar a usar la aplicación. Además, cierta información de LinkedIn puede ser establecida con distintos grados de privacidad, por lo que a veces, según el rol del usuario respecto de la persona de la que se genera el CV, se podrá acceder a dicha información o no.

Debido al conocimiento del desarrollador, hay que tener en cuenta que la aplicación se desarrollará bajo ciertas condiciones que limitarán su uso. Esto se debe a que la aplicación solo podrá usarse en **Windows**, tomando la suposición de que el usuario tendrá el sistema operativo en su equipo. De todas formas, este problema es considerado un problema menor ya que como podemos ver en el gráfico, Windows es el sistema operativo predominante en la actualidad.

También es importante saber que el usuario necesita iniciar sesión en la red social para empezar a usar la aplicación.

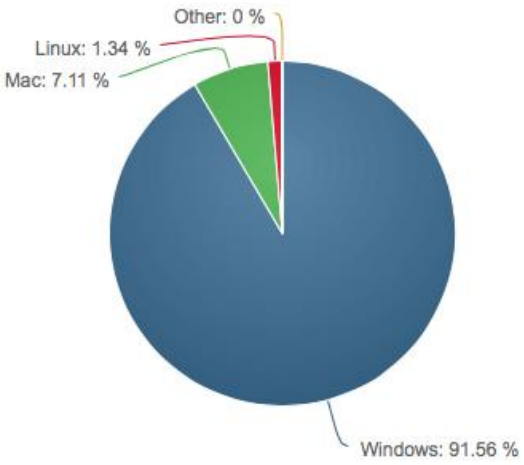


Imagen 3. Gráfica de uso de Sistemas Operativos.

Además, de cara al futuro la aplicación podría tener más funcionalidades como seleccionar distintas plantillas para la generación del CV, poder elegir la ausencia de alguna sección presente en el CV por defecto, más filtros a la hora de buscar perfiles, etc.

4.2.3 Requisitos específicos

Los requisitos del sistema serán divididos en dos distintos tipos: **requisitos funcionales** y **requisitos no funcionales**. Estos requisitos son definidos mediante **historias de usuario**, y también se les añaden **pruebas de aceptación**.

4.2.3.1 Requisitos funcionales

Los requisitos funcionales del sistema que han sido identificados son los siguientes:

- **Localizar el perfil a partir de una url**: Como usuario, quiero poder localizar un perfil tras insertar la url del mismo para posteriormente generar el CV.
    - *Prueba de aceptación*: Al entrar a la aplicación y seleccionar "Generar CV", el usuario debe poder insertar una url de LinkedIn y que la aplicación localice el perfil.
- **Elegir el tipo de CV deseado**: Como usuario, quiero que una vez mi perfil se localice pueda elegir el tipo de CV que deseo (con o sin foto) para posteriormente generarlo.
    - *Prueba de aceptación*: Tras localizar el perfil, el usuario debe poder elegir el tipo de CV que desea generar.
- **Generar CV**: Como usuario, quiero poder generar mi CV automáticamente y después guardarlo para poder usarlo.
    - *Prueba de aceptación*: Una vez se ha seleccionado el tipo de CV a generar, se ha de generar y el usuario ha de ser capaz de descargarlo o previsualizarlo.
- **Añadir filtros de búsqueda**: Como usuario (reclutador), quiero poder añadir filtros de búsqueda al estilo LinkedIn cuando quiera buscar perfiles para poder buscar el perfil indicado.

    - *Prueba de aceptación*: Al abrir la aplicación y seleccionar “filtrar perfiles”, el usuario ha de ser capaz de añadir distintos filtros que se aplicarán en la búsqueda para mostrar solo aquellos perfiles compatibles.

- **Listar perfiles compatibles**: Como usuario (reclutador), quiero que una vez haga la búsqueda reciba todos los perfiles compatibles en una lista para poder navegar y hacer distintas acciones.

    - *Prueba de aceptación*: Tras aplicar los filtros y hacer la búsqueda, el usuario ha de recibir todos los perfiles compatibles en forma de lista.

- **Navegar a un perfil compatible**: Como usuario (reclutador), quiero poder navegar y ver el perfil en LinkedIn de una persona con un perfil compatible con los filtros aplicados para saber más sobre él/ella.

    - *Prueba de aceptación*: Al tener todos los perfiles en lista, el usuario ha de ser capaz de navegar al perfil de LinkedIn de un usuario compatible.

- **Generar CV de un perfil compatible**: Como usuario (reclutador), quiero poder generar el CV automáticamente de una persona con perfil compatible para poder enviarlo al equipo de RRHH y que consideren su posible incorporación.

    - *Prueba de aceptación*: Al tener todos los perfiles en lista, el usuario ha de ser capaz de generar el CV de uno de esos perfiles, seleccionando el tipo de CV a generar y generándose, pudiéndose descargar o visualizar.

4.2.3.2 Requisitos no funcionales

Los requisitos no funcionales identificados son los siguientes:

- **Rendimiento**: la aplicación deberá proporcionar una respuesta en un tiempo razonable, sin tiempos de espera infinitos en caso de error.

    - *Prueba de aceptación*: en este caso, se hará una medición de cuánto se tarda en hacer ciertas acciones que involucren al script de Python, como generar un CV. El tiempo de ejecución deberá de ser menor de 30 segundos, al ser una operación compleja.

- **Usabilidad**: la aplicación deberá ser intuitiva y fácil de usar para todos los usuarios, independientemente de los conocimientos de cada uno en informática.

    - *Prueba de aceptación*: la aplicación podrá ser usada por usuarios reales que darán feedback mientras interactúan con la aplicación. El objetivo es que sean capaces de hacer las principales tareas sin necesidad de más información o ayuda.

- **Mantenibilidad**: el código deberá estar bien estructurado y ser legible para facilitar cualquier cambio que hubiera que introducir. Además incluirá comentarios explicativos para entender qué hace cada parte del código.

    - *Prueba de aceptación*: el código deberá ser revisado verificando que está bien documentado, a la vez que se evalúa la claridad y comprensibilidad del mismo, así como lo fácil que es introducir nuevas modificaciones o cambiar algo ya existente.

- **Portabilidad**: la aplicación debe ser fácil de instalar en cualquier sistema Windows mediante un asistente de instalación sencillo.

    - *Prueba de aceptación*: se ha de verificar que la aplicación se puede instalar y ejecutar correctamente en equipos Windows. Para ello, se probará a instalar y usar la aplicación en distintos equipos.

- **Fiabilidad**: la aplicación se deberá recuperar correctamente en caso de cualquier error que pudiera ocurrir tanto dentro de la aplicación como en el script de Python.

    - *Prueba de aceptación*: se someterá a fallos en el sistema para verificar la recuperación del mismo, por lo que en caso de error la aplicación ha de reaccionar correctamente. Por ejemplo, se simularán errores en la ejecución del script de Python.

CAPÍTULO 5
Modelado conceptual

La sección de modelado conceptual es una sección esencial. En ella, se permite representar de manera **abstracta** y **clara** los elementos fundamentales del sistema, sus interacciones y relaciones, por lo que mediante el uso de distintos tipos de diagramas se conseguirá todo esto.

Los principales objetivos de estos diagramas son facilitar la comprensión del sistema entre las distintas personas involucradas en el proyecto, identificar y clarificar los requisitos y proporcionar una base sólida para el diseño y la implementación, haciendo que este capítulo sea uno de los más importantes del proyecto.

El modelado conceptual será representado por 4 diagramas:

- Diagrama de clases.
- Diagrama de secuencia.
- Diagrama de casos de uso.
- Diagramas de flujo.

5.1 Diagrama de clases

El diagrama de clases (normalmente en UML) es un tipo de diagrama de **estructura estática** que describe la estructura de un sistema mostrando las clases del sistema, atributos, relaciones, etc.

- **Clases:** son las entidades principales del sistema. Se dibujan como rectángulos, con distintas secciones para añadir el nombre de la clase, los atributos y los métodos.
- **Atributos:** representan las propiedades o datos. La notación normalmente contiene la visibilidad, el nombre y el tipo de dato del atributo.
- **Métodos:** representan las funciones de la clase.
- **Relaciones:** hay distintos tipos de relaciones: asociación, agregación, composición, generalización o herencia, dependencia. Hay pequeñas diferencias entre ellas, así como la simbología también cambia.

En el caso del proyecto, se ha utilizado el diagrama de clases para representar y explicar el objeto en torno al cual gira todo. Este es el perfil del usuario. Este objeto se crea en base a la información que se extrae desde LinkedIn, y está formado por distintos atributos como se puede apreciar en la siguiente imagen. A partir de estos atributos se genera posteriormente el CV.

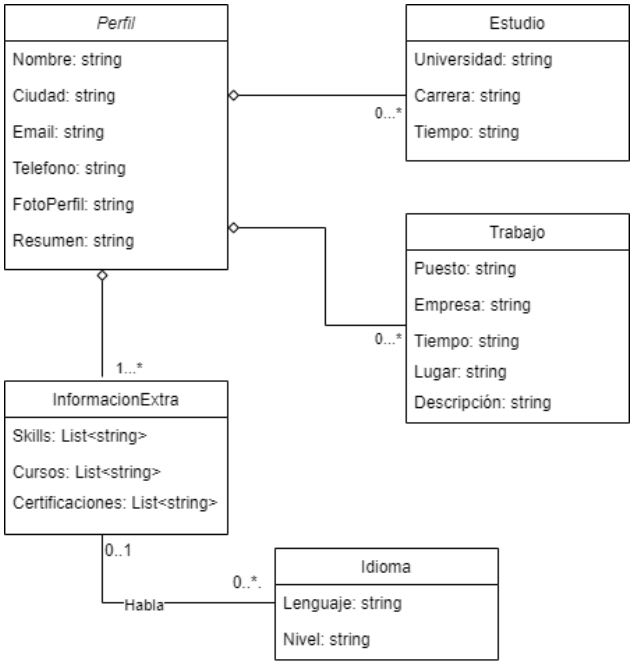


Imagen 4. Diagrama de clases de la entidad "Perfil".

5.2 Diagrama de secuencia

Este tipo de diagrama muestra la **interacción** de un conjunto de objetos en una aplicación a través del tipo, modelándose para cada caso de uso. Es muy útil para representar la lógica de los casos de uso y comprender cómo las partes del sistema colaboran para realizar una función.

Este diagrama está formado por distintos elementos:

- **Objetos:** representan las entidades que participan en la interacción.
- **Mensajes:** son la comunicación entre los objetos. Es una línea que va de un objeto a otro.
- **Activaciones:** tiempo durante el cual un objeto realiza una acción.

En el caso del proyecto, el diagrama de secuencia correspondiente es el siguiente:

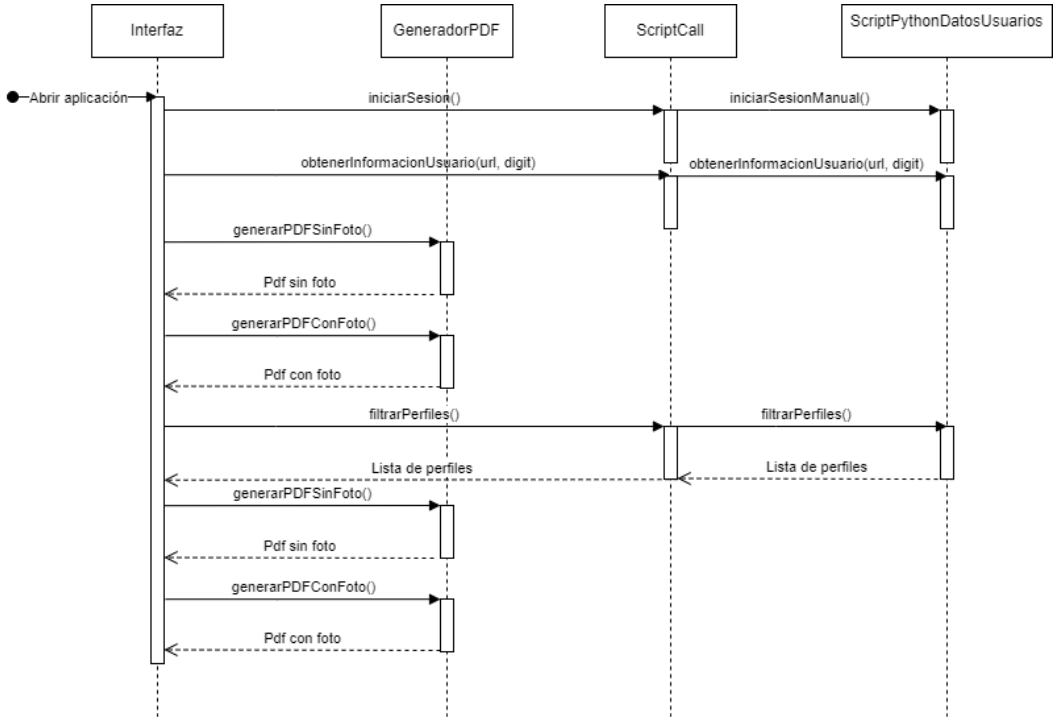


Imagen 5. Diagrama de secuencia del proyecto.

En él se puede apreciar los distintos casos de uso y como las llamadas se van sucediendo y la comunicación entre distintos objetos ocurre.

5.3 Diagrama de casos de uso

Este diagrama es utilizado para modelar y documentar los requisitos funcionales de un sistema. Representa las **interacciones** entre los actores y el sistema, mostrando los distintos casos de uso presentes en el sistema. Están formados por distintos elementos:

- **Actores:** representan las entidades que interactúan con el sistema.
- **Casos de uso:** representan las funcionalidades que el sistema ofrece a los actores.
- **Relaciones:** representan las distintas conexiones entre los actores y los casos de uso. Hay distintos tipos: asociación, inclusión, extensión, generalización.

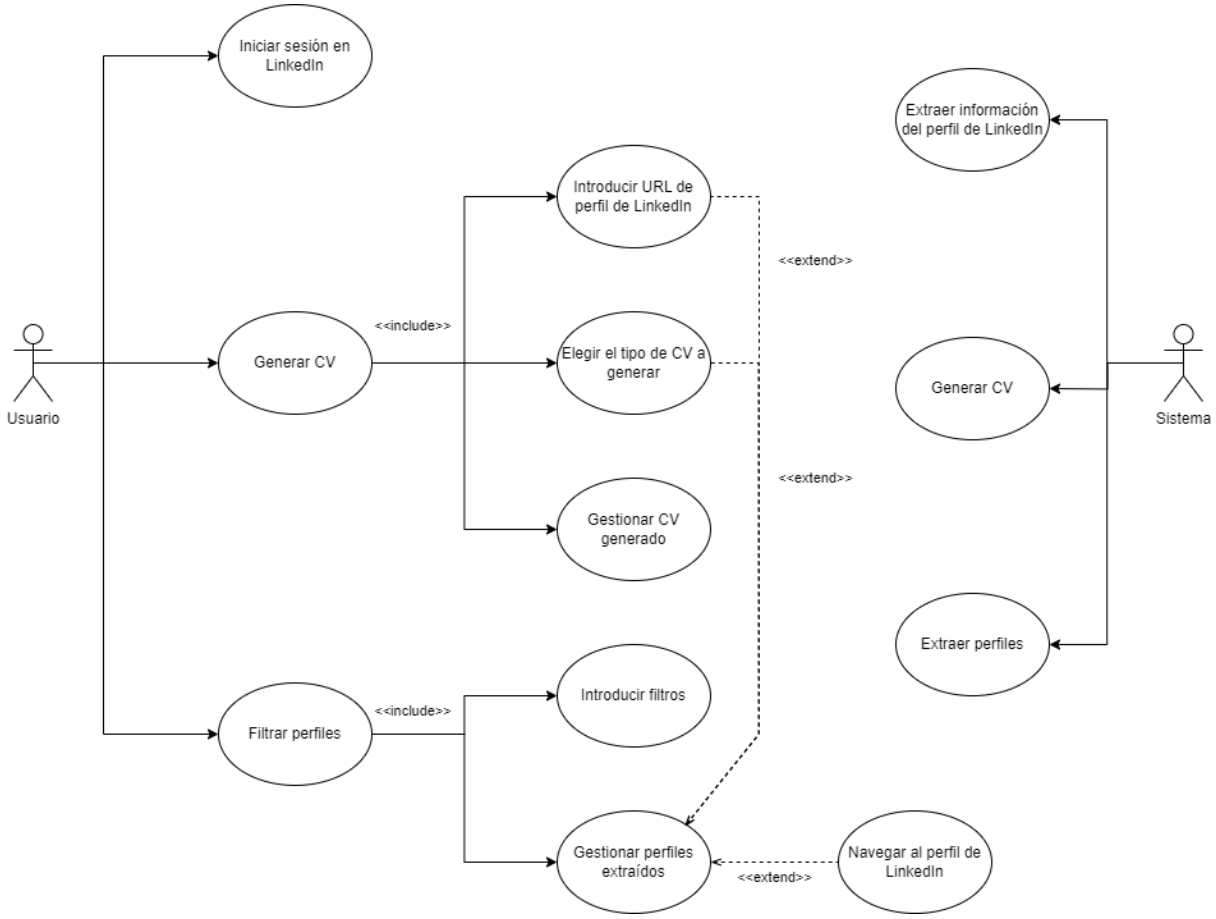


Imagen 6. Diagrama de casos de uso del proyecto.

5.4 Diagramas de flujo

El diagrama de flujo es la representación gráfica de un algoritmo o proceso, representando el **flujo de trabajo** paso a paso. Estos tipos de diagramas utilizan distintos símbolos con significados definidos que representan los pasos del proceso y el flujo de ejecución con flechas.

En el caso del proyecto, se usarán para representar el proceso de ejecución de los dos casos de uso principales: generar el currículum y filtrar perfiles.

Antes de comenzar, explicaré la distinta simbología empleada para clarificar el diagrama:

- **Comienzo/Final:** el comienzo y el final se representarán con la misma figura. Esta figura será un rectángulo de esquinas muy redondeadas de color gris oscuro.
- **Decisión:** las situaciones en las que se depende de la decisión del usuario se representarán con un rombo verde. En cada una de estas posibles decisiones se indicará el valor de la misma (si/no, verdadero/falso, etc.).
- **Proceso:** en determinados momentos de la aplicación la ejecución de un proceso toma lugar (cuando se genera un CV, por ejemplo). Esto se representará en el diagrama con un rectángulo de color violeta.

- **Input:** aquellos momentos en los que el usuario de la aplicación se espera que introduzca información en el sistema se representarán con un romboide de color azul.

Para generar el CV, tendríamos el siguiente diagrama de secuencia:

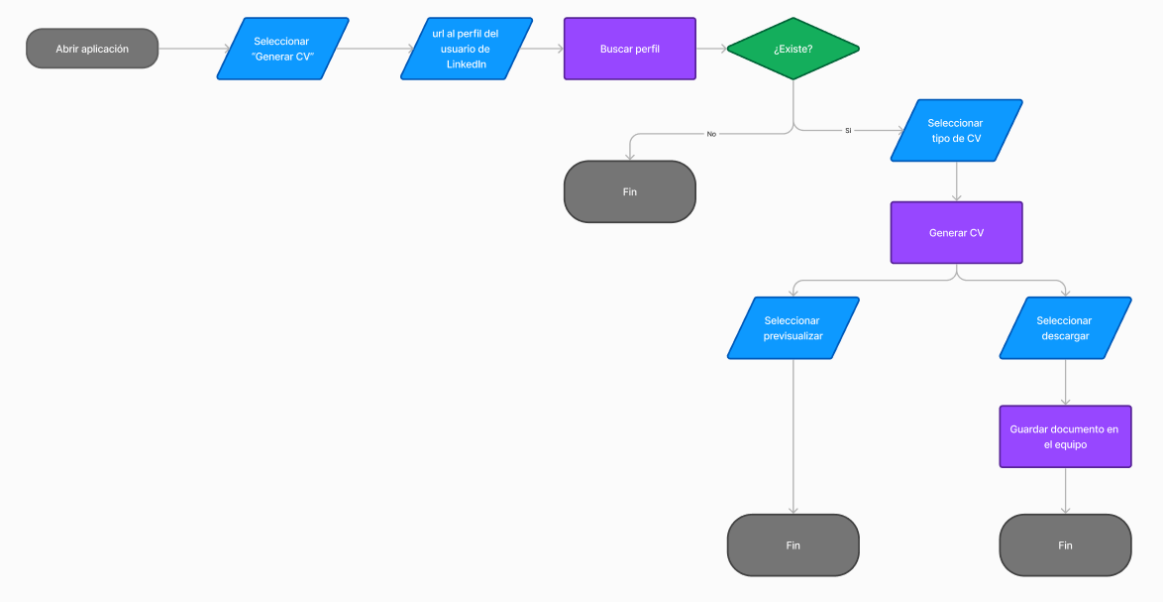


Imagen 7. Diagrama de flujo de la funcionalidad "Generar CV".

Para el caso de uso de filtrar los perfiles, tendríamos el siguiente:

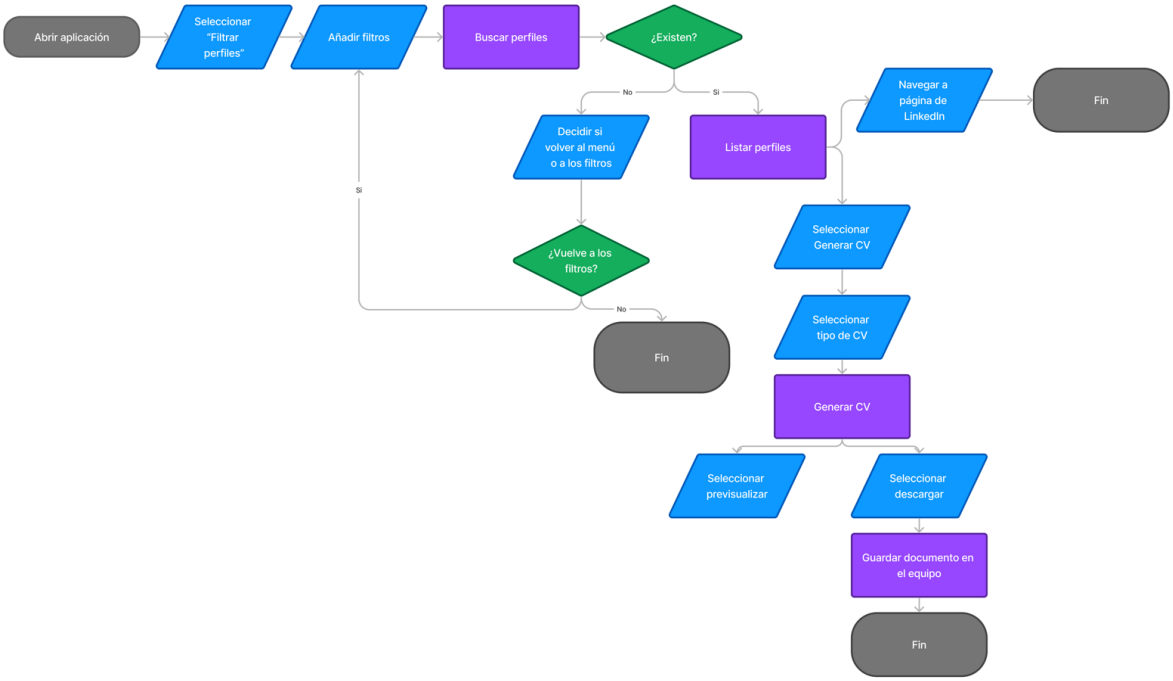


Imagen 8. Diagrama de flujo de la funcionalidad "Filtrar perfiles".

CAPÍTULO 6
Diseño de la herramienta

A lo largo de esta sección se abordará el diseño de la aplicación desde distintos puntos de vista, desde las herramientas o la arquitectura utilizada, hasta el prototipado de la aplicación o los patrones de diseño.

El objetivo de este capítulo es contextualizar el entorno alrededor del desarrollo de la aplicación. Por tanto, se hablará de los distintos lenguajes utilizados, arquitectura, interfaz e incluso de los patrones de diseño a utilizar, dejando así todo explicado.

Esta sección es la tercera pieza para completar toda la estructura de la aplicación. una vez ya conocemos los requisitos y la representación abstracta del proyecto mediante los distintos diagramas de la sección anterior, esta sección completará la información necesaria antes de comenzar el desarrollo hablando de las distintas tecnologías a utilizar, así como de otra información relevante previa a la implementación.

Esta sección se dividirá en distintos apartados como sigue:

- En primer lugar, se hablará de los **aspectos técnicos**, englobando tanto las distintas herramientas utilizadas como la arquitectura utilizada en el proyecto.
- A continuación, se hablará del **diseño**, incluyendo el prototipado de la interfaz, con la explicación de la función de cada una de las pantallas.
- Finalmente, se explicarán los **patrones** de diseño, explicando en concreto el patrón de diseño utilizado en el proyecto.

6.1 Aspectos técnicos

6.1.1 Herramientas utilizadas

En este proyecto, la aplicación se desarrolla completamente usando *Visual Studio 2022*. VS es un entorno de desarrollo integrado para Windows y macOS. Lo interesante y utilizado en este proyecto es su compatibilidad con el lenguaje *C#* y la posibilidad de compatibilidad con la plataforma *.NET*. Algunas características son:

- Es un editor de código **avanzado** con herramientas para la refactorización del mismo, así como la posibilidad de depuración con puntos de interrupción, seguimiento de la ejecución del programa, etc.
- Es fácilmente **integrable** con herramientas de desarrollo como *Azure DevOps*, algo también utilizado en este proyecto, por lo que es algo muy útil, además de soportar *Git* de forma nativa.
- Tiene herramientas integradas para escribir y ejecutar pruebas unitarias, pudiendo ver el resultado de cada prueba, así como las respuestas.

- Proporciona herramientas para el **desarrollo de interfaces** en el propio entorno de programación. En el caso del proyecto actual, se utiliza *WPF*, permitiendo el entorno la creación de interfaces ricas y responsivas sin la necesidad de escribir código manualmente.

Estas características eran muy interesantes de cara al proyecto, por lo que la elección de este IDE ayudaría en gran medida.

Además, el desarrollo del script de *Python* se produce en *Visual Studio Code*. Es un editor de código fuente gratuito, ligero y una herramienta ampliamente utilizada debido a su flexibilidad, extensibilidad y rendimiento. Algunas características son:

- Es un editor **rápido** y con **bajo consumo** de recursos, permitiendo una experiencia de usuario fluida.

- Soporta varios lenguajes de programación, y además mediante las extensiones del Marketplace se puede añadir nuevos.

- El marketplace de *VSCode* permite instalar una gran variedad de **extensiones** de manera que puede convertirse en un editor muy potente con gran variedad de herramientas muy variadas.

- Tiene la función de *Intellisense* incorporada para el autocompletado de código.

- Tanto depuración como testing están ambos integrados, permitiendo la ejecución de pruebas con puntos de interrupción, inspección de variables, etc.

- Permite el acceso a la línea de comandos a través de su terminal integrado, pudiendo ejecutar comandos de shell directamente desde el editor.

Por último, para la prototipación de las interfaces se ha usado *Figma*. Es una herramienta online de **diseño** que se ha vuelto muy popular en los últimos tiempos por su capacidad de poder trabajar conjuntamente en tiempo real. Las características que han hecho que sea una herramienta interesante para el proyecto son:

- Figma funciona totalmente **online**, por lo que no es necesario instalar nada en el equipo y permite acceder a cualquier proyecto desde cualquier lugar con conexión a internet.

- Permite crear **componentes reutilizables** asegurando la consistencia y la eficiencia en los proyectos. Además, ajusta automáticamente los elementos en función del contenido.

- Se pueden crear **prototipos** interactivos con transiciones, animaciones, etc simulando el comportamiento de la aplicación final permitiendo recibir feedback rápidamente del usuario.

- Guarda automáticamente el historial de versiones, por lo que en caso de un cambio que finalmente no fuera necesario, se puede restaurar una versión anterior sin mayor dificultad.

6.1.2 Arquitectura utilizada

La arquitectura que se ha decidido utilizar es una **arquitectura en capas**. Esta arquitectura es un estilo de diseño que organiza la aplicación en capas horizontales, cada una de las cuales con una responsabilidad específica y bien definida, facilitando el desarrollo, mantenimiento y escalabilidad. Inicialmente puede ser complejo diseñar y definir claramente cada capa, así como el rendimiento se

puede ver afectado por una sobrecarga de rendimiento. Sin embargo, para este proyecto, este tipo de arquitectura era el que mejor se amoldaba.

El modelo más típico es el modelo de tres capas. En este tipo de modelo, tenemos tres capas claramente separadas que se dividirán de la siguiente manera:

- **Capa de presentación**: es la capa en donde se desarrolla la **interfaz** de usuario. Es la parte de la aplicación con la que el usuario interactúa, donde introduce información, navega, etc.

- **Capa de lógica de negocio**: es la capa donde se procesan las **entradas** de usuario. Esta capa puede también llamar a otras funciones o extraer datos de la base de datos y, cuando sea necesario, da una información a la capa de presentación que servirá como respuesta.

- **Capa de datos**: es la capa donde se accede a los datos almacenados, ya sea en bases de datos, sistemas de archivos, etc. La capa de lógica hará llamadas a esta capa, recibiendo los datos que sean necesarios.

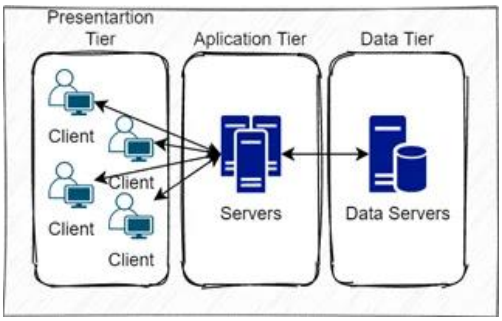


Imagen 9. Modelo de arquitectura de 3 capas.

En el caso de este proyecto, la arquitectura utilizada se asemeja un poco a este enfoque pero con alguna diferencia. A continuación se habla de las capas presentes en el proyecto:

- **Capa de presentación**: en el proyecto, la interfaz de usuario se desarrolla usando *WPF*. En este caso, el usuario navegará entre pantallas e introducirá información manualmente. Tanto para la creación de las interfaces como para su programación y gestión se usará Visual Studio.
- **Capa de lógica de negocio**: en el proyecto, procesa las entradas del usuario y a partir de los datos que obtiene llama al script de *Python* para obtener la información del perfil. Para esta capa se usará *C#* en *Visual Studio*.
- **Capa de acceso a datos**: en el proyecto, esta capa es **particular**. Como tal, no es una capa en la que se acceda a una base de datos. Lo que se hace es llamar a un **script** de Python que devolverá toda la información encontrada en el perfil de LinkedIn del usuario. Este script funciona gracias a técnicas de *web scraping* usando distintas bibliotecas Python que ayudan a esta tarea. Por ello, no es una capa de datos al uso, si no que tiene esa peculiaridad. En este caso se usará *Visual Studio Code* para la programación del script.

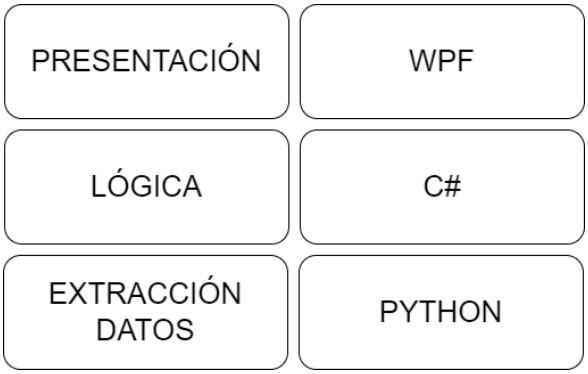


Imagen 10. Arquitectura del proyecto con las tecnologías utilizadas.

6.2 Diseño

En esta sección se describen las decisiones y enfoques técnicos tomados para crear la aplicación. Esta etapa es una etapa crítica, ya que establece la base para la implementación, asegurando que la aplicación cumpla con los requisitos definidos anteriormente.

Tras haber hablado de la arquitectura, ahora toca hablar de las distintas entidades presentes en la aplicación y de la estructura de las interfaces del usuario. Es importante ya que se quiere tener una interfaz intuitiva y fácil de usar para que los usuarios interactúen con la aplicación de manera eficiente y natural, y que se puedan agregar nuevas funcionalidades y adaptar la aplicación a futuros cambios.

6.2.1 Entidades presentes en la aplicación

Como ya se ha mencionado anteriormente, la aplicación no tiene base de datos, por lo que no se puede hablar de una base de datos y de las distintas tablas que forman parte de ella.

Sin embargo, la aplicación gira en torno a un objeto principal que es el perfil del usuario. Este objeto, construido a raíz de la información que se extrae desde LinkedIn, contiene distintos atributos que finalmente serán la base del CV a generar.

Por tanto, un posible esquema del objeto sería el siguiente:

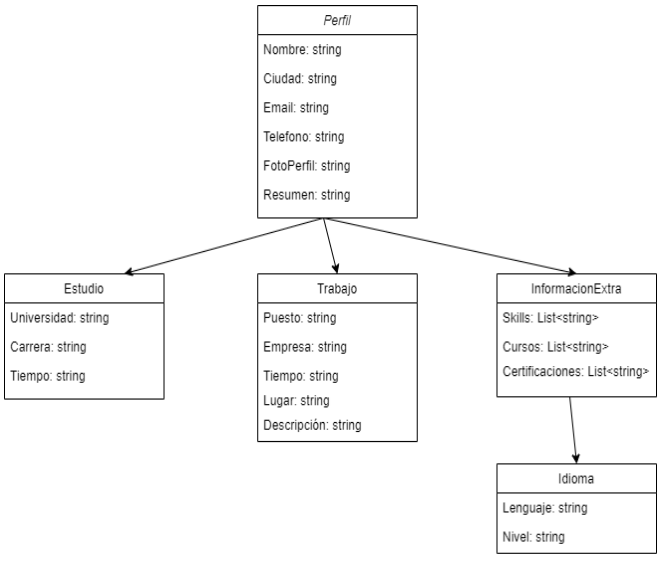


Imagen 11. Esquema del objeto "Perfil".

En esta imagen podemos ver como el perfil está formado por una serie de *strings* que le van a caracterizar, como son el nombre, teléfono, etc. además de otros objetos como el objeto Estudio y objeto Trabajo. Estos dos objetos serán de tipo lista (un usuario puede haber tenido más de un estudio y más de un trabajo) y cada uno de ellos irá caracterizado por determinados strings. Por último, habrá un objeto InformacionExtra que contendrá en listas distinta información adicional del usuario como podría ser las habilidades, cursos que ha realizado, certificaciones o lo distintos idiomas que puede hablar, estos últimos con el idioma que habla y el nivel.

6.2.2 Diseño de la interfaz

A continuación, se mostrarán los **prototipos** de las distintas interfaces que constituyen la aplicación. Es importante saber que puede ser que durante la implementación del proyecto puede ser que sufran cambios debido a la aparición de problemas no identificados que puedan surgir.

Según lo diseñado, al abrir la aplicación el usuario se encontrará con una pantalla con dos botones en la que elegirá a qué funcionalidad desea acceder. El pulsar cualquier botón le hará navegar a la pantalla correspondiente.

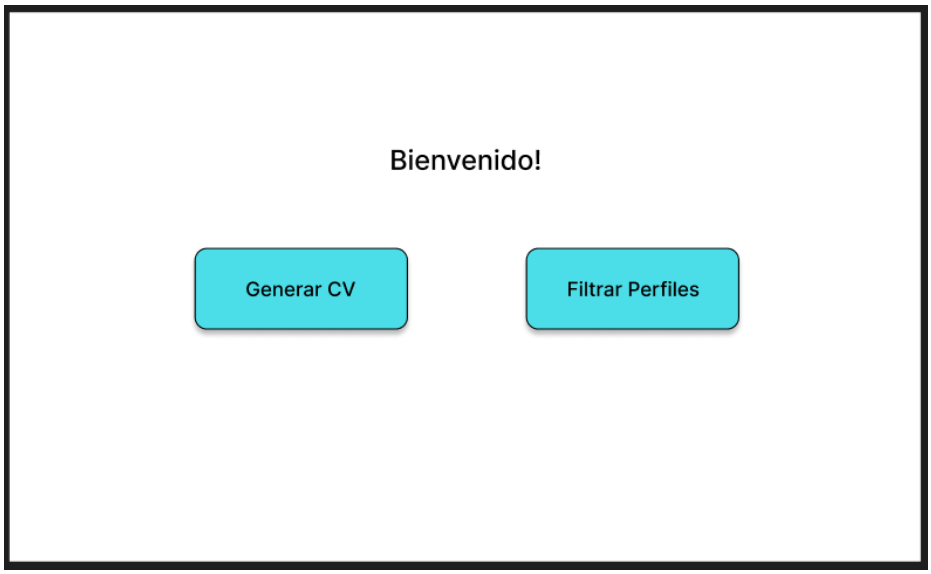


Imagen 12. Prototipo de la pantalla inicial.

En caso de pulsar el botón de “Generar CV”, el usuario se encontrará con una pantalla en la que tendrá un TextField para introducir el nombre del perfil del que desea generar el currículum. Este parámetro de entrada será el punto de partida del script para obtener toda la información necesaria.

Imagen 13. Prototipo de la pantalla de introducir el nombre.

Al pulsar el botón de búsqueda (en este caso la imagen de la lupa), aparecerá una pantalla de carga mientras el script se ejecuta. Esta pantalla estará unos segundos mostrándose, y automáticamente, se mostrará la siguiente pantalla.

Imagen 14. Prototipo de la pantalla de carga.

En caso de error, mostrará una pantalla diciendo que el perfil no ha sido encontrado y el usuario solo podrá volver a la pantalla inicial.

Imagen 15. Prototipo de la pantalla de error al buscar un perfil.

En caso de éxito, se mostrará una pantalla diciendo que el perfil ha sido encontrado, y el usuario podrá elegir entre volver al menú principal o navegar a la siguiente pantalla donde elegirá el tipo de CV a generar.

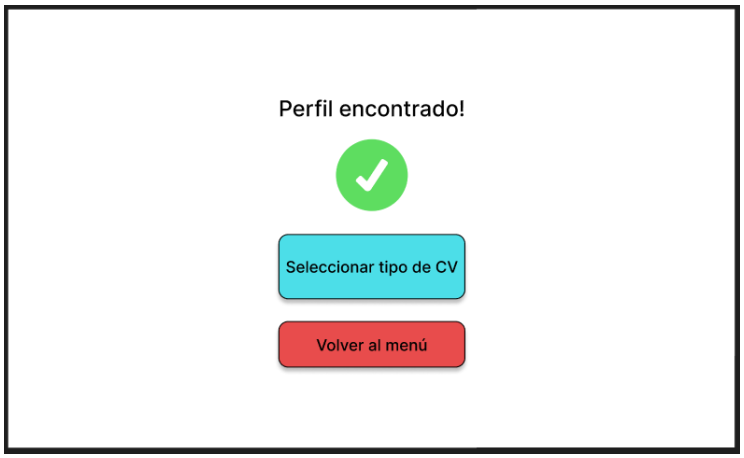


Imagen 16. Prototipo de la pantalla de éxito al buscar un perfil.

Al seleccionar el tipo de CV, en la primera versión de la aplicación encontraremos dos tipos, con foto y sin foto. El usuario elegirá uno de ellos y la aplicación lo generará.

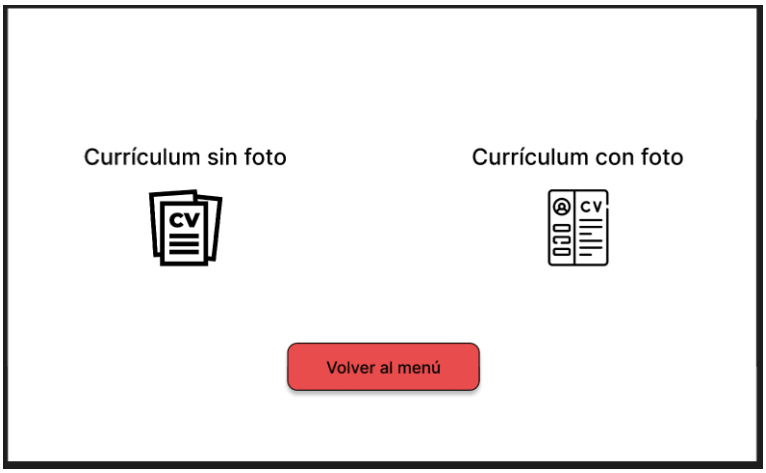


Imagen 17. Prototipo de la pantalla de selección del tipo de CV.

Tras esto, se mostrará una pantalla de carga similar a la anterior. Una vez se genere el CV el usuario podrá descargarlo o previsualizarlo antes de descargarlo.

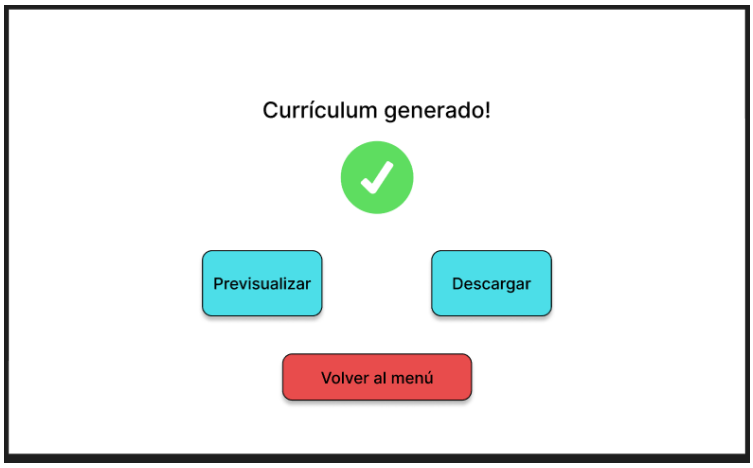


Imagen 18. Prototipo de la pantalla de éxito tras generar el CV.

En caso de seleccionar el botón de “Filtrar usuarios”, el usuario se encontrará con la siguiente pantalla. Está formada a modo de formulario, y el usuario deberá rellenar los campos que crea necesarios para encontrar los perfiles que sean compatibles. En este prototipo, el diseño aún no está claro ya que depende de la implementación los campos por los que se podrá filtrar.

Imagen 19. Prototipo de la pantalla de introducir los filtros.

Una vez aplicados los filtros, aparecerá una pantalla de búsqueda mientras el script encuentra los perfiles compatibles. Una vez los encuentre, mostrará una pantalla con una lista con todos aquellos perfiles que hayan sido compatibles (a falta de implementarlo, lo más seguro es que se pida un número máximo de personas para la lista). De cada elemento de la lista el usuario podrá navegar hasta el perfil de LinkedIn de la persona en la lista o generar el CV, por lo que se reutilizan las interfaces previamente explicadas.

Imagen 20. Prototipo de la pantalla de listar los perfiles encontrados.

6.2.3 Conclusiones del diseño

Tras el diseño de la interfaz, se pudieron alcanzar algunas conclusiones de pequeños cambios que habría que hacer en las interfaces finales debido a limitaciones en la implementación.

En primer lugar, habrá que estudiar la necesidad de que el usuario de cierta manera **inicie sesión** en LinkedIn, ya que para obtener los datos es necesario explorar el perfil, y para ello hay que iniciar sesión.

Además, en los filtros que podrá utilizar el usuario, dependerá de lo que necesite la implementación y de los filtros que pueda aplicar, y en base a eso la interfaz final se verá sometida a cambios.

Finalmente, también habrá que gestionar el **guardado** y la **previsualización** del currículum. Dependerá de la implementación si hará falta algún tipo de interfaz para ello o automáticamente con las aplicaciones por defecto de *Windows* podrán gestionarlo.

Por último, habrá que revisar que el usuario entienda el uso de imágenes en los botones, como la lupa para buscar. En caso de que cree problemas, habrá que buscar una solución, por ejemplo, cambiando la imagen por texto.

6.3 Patrones de diseño

Los patrones de diseño son **soluciones** habituales a problemas que ocurren con frecuencia en el diseño de software. Son conceptos para resolver problemas particulares que, a partir de unos detalles a seguir, se puede modificar al gusto del usuario para solucionar y hacerlo encajar con el código en el que se quiere usar.

Los patrones varían en complejidad, nivel de detalle y escala de aplicabilidad. Además, los patrones pueden clasificarse según su propósito. Hay tres grupos principales de patrones:

- **Patrones creacionales:** proporcionan mecanismos de creación de objetos que incrementan la flexibilidad y la reutilización de código existente. Por ejemplo, tenemos el Factory, Builder, Singleton, etc.

- **Patrones estructurales:** explican cómo ensamblar objetos y clases en estructuras más grandes a la vez que se mantiene la flexibilidad y eficiencia de la estructura. Por ejemplo, tenemos el Decorator, Facade, Adapter, etc.

- **Patrones de comportamiento:** se encargan de una comunicación efectiva y la asignación de responsabilidades entre objetos. Por ejemplo, tenemos el Command, Mediator, Chain of Responsibility, etc.

6.3.1 Patrón Singleton

El patrón empleado en el proyecto es el patrón **Singleton**. Este patrón es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de **acceso global** a esa instancia.

Para ello, el constructor por defecto se hará privado por lo que no se podrá usar el operador new con la clase Singleton, así como se creará un método estático de acceso a la instancia. En caso de que está instancia no tenga valor, se inicializará, y en las siguientes llamadas se devolverá este objeto almacenado en caché.

En el proyecto, este patrón se usa en repetidas ocasiones con distintas clases ya que al tener varias ventanas distintas, a menudo se da la situación de que en una ventana hace falta un objeto que se ha inicializado en una ventana anterior (por ejemplo, la url al perfil de LinkedIn hace falta en varias pantallas), por lo que se aplica el patrón Singleton para acceder siempre al mismo objeto sin que haga falta irlo pasando de ventana en ventana, sino que con solo llamar al método getInstance() se obtenga la instancia de ese objeto.

Por ejemplo, a continuación se muestra un ejemplo de la implementación de una clase Singleton en el proyecto.

```
public class InfoPerfil
{
    3 referencias
    public Perfil jsonInfo { get; set; }

    private static InfoPerfil infoPerfil;

    1 referencia
    private InfoPerfil()
    {
        jsonInfo = new Perfil();
    }

    2 referencias
    public static InfoPerfil getInstance()
    {
        if (infoPerfil == null)
        {
            infoPerfil = new InfoPerfil();
        }

        return infoPerfil;
    }
}
```

Imagen 21. Implementación del patrón Singleton en el proyecto.

En esta clase, podemos apreciar que solo hay un atributo, “jsonInfo”, pero también hay alguna otra clase Singleton en el proyecto con más atributos, pero la estructura es exactamente la misma.

CAPÍTULO 7
Implementación

Durante esta sección se hablará de distintos aspectos relacionados con la implementación de la aplicación, tanto las tecnologías utilizadas como soluciones a distintos problemas surgidos durante el desarrollo.

El objetivo de esta sección es explicar **cómo** se ha realizado la implementación del proyecto. Es una sección importante ya que una vez teniendo toda la información necesaria para realizar la implementación, explica cómo se ha realizado esta implementación para cubrir los requisitos del usuario, así como relacionar los distintos diagramas con el desarrollo.

Para ello, se dividirá el capítulo en distintas secciones. La primera de ellas hablará de las distintas tecnologías utilizadas en el proyecto. El resto de secciones se utilizarán para explicar distintas soluciones a problemas o necesidades surgidas durante la implementación, incluyendo fotos del código en algunas ocasiones.

7.1 Tecnologías utilizadas

Para la **implementación** de este proyecto se decidió utilizar **C#**. Es un lenguaje multiparadigma desarrollado y estandarizado por *Microsoft* como parte de su plataforma .NET. Este lenguaje es una evolución que *Microsoft* hizo tomando lo mejor de los lenguajes C y C++, y añadiendo funcionalidades inspirándose en otros lenguajes. Sus principales características son la sencillez, modernidad, seguridad, sistemas de tipos unificados, extensibilidad, versionable, compatible y eficiente.

Para la creación de la aplicación y su **interfaz** se ha utilizado **WPF**. Windows Presentation Foundation es un marco de interfaz de usuario que crea aplicaciones cliente de escritorio. Las interfaces se definen a través del lenguaje declarativo *XAML*. Fue lanzado por Microsoft como parte de .NET. La plataforma admite un amplio conjunto de características, como un modelo de aplicación, recursos, controles, etc. Ofrece una amplia infraestructura y potencia gráfica, con la que es posible desarrollar aplicaciones visualmente atractivas, con facilidades de interacción incluyendo animación, vídeo, audio, documentos, etc.

Para el d**esarrollo de algoritmos** de *web scraping* se utilizó **Python**. Este lenguaje es un lenguaje interpretado de alto nivel y multiparadigma, que hace hincapié en la legibilidad de su código. Es uno de los lenguajes más populares, por lo que tiene gran cantidad de recursos, tutoriales y soporte debido a su popularidad y comunidad.

Para el desarrollo de los scripts de web scraping se utilizaron principalmente dos librerías:

- **Selenium**: Selenium es un entorno de pruebas que se utiliza para comprobar si un software funciona correctamente, depurando casos de prueba que se pueden automatizar. Las acciones se pueden ejecutar punto a punto, referenciar a objetos DOM, etc.

Por ello, se decidió que con todas estas características se podría utilizar Selenium para navegar hasta el perfil de LinkedIn deseado, y en general, manejar toda la interacción con la página web de forma automática.

- **BeautifulSoup**: Esta biblioteca permite extraer contenido y transformar en datos utilizables en Python. Es muy popular, con una documentación completa y funcionalidades bien estructuradas, por lo que tiene una gran comunidad que ofrece soluciones variadas a la hora de utilizarla.

  Su funcionamiento consiste en la creación de un árbol con todos los elementos de la página web, del cual se extrae la información a través de distintos métodos.

Por tanto, el uso de estas dos librerías de manera conjunta es una excelente opción para el *web scraping*. Con Selenium se automatiza la navegación y la localización de los elementos de los que queremos sacar la información, filtrando a través de características como el id, para posteriormente, parsear el contenido de la página con BeautifulSoup y extraer la información requerida pudiendo organizarla y transformarla en datos manejables en Python.

7.2 Conexión Python - C#

Al usar un script de Python para extraer toda la información requerida del perfil de LinkedIn, era necesario conectar de alguna forma ese script con C# de tal forma que pudiese ser invocado cuando era necesario. Para ello, había distintas opciones con varios *paquetes NuGet* que se podían utilizar, pero de una u otra manera había problemas con ello.

Por tanto, lo que se decidió usar al final fue la creación de un **proceso** para la ejecución del fichero Python. Para recoger el resultado, la opción más viable era simplemente leer la salida por defecto, pero encontramos errores en la decodificación de ciertos caracteres (en primer lugar las tildes y ñ, y posteriormente caracteres especiales distintos a los latinos). Por ello, el proceso es crear un archivo en el cual escribir toda la respuesta, para después desde C# leer de ese archivo y obtener la respuesta.

```
// Crea un nuevo proceso
ProcessStartInfo psi = new ProcessStartInfo
{
    FileName = "python",
    Arguments = $"\"{pythonScript}\" \"{jsonData}\" \"{code}\"",
    RedirectStandardOutput = true,
    RedirectStandardError = true,
    UseShellExecute = false,
    CreateNoWindow = true,
    StandardOutputEncoding = Encoding.UTF8
};

Process process = new Process
{
    StartInfo = psi
};

// Ejecuta el proceso
process.Start();

string error = process.StandardError.ReadToEnd();

process.WaitForExit();
```

Imagen 22. Implementación de la creación de un proceso en C#.

El objeto de respuesta se decidió que fuese un *JSON*, ya que a fin de cuentas es un *String*, por lo que íbamos a tener pocos problemas, y además a la hora de transformarlo en un objeto de C#, con la característica "JsonProperty" era muy sencillo. El objeto que simboliza al usuario del cual se va a generar el CV tiene unas cuantas características que se pueden ver en la siguiente imagen:

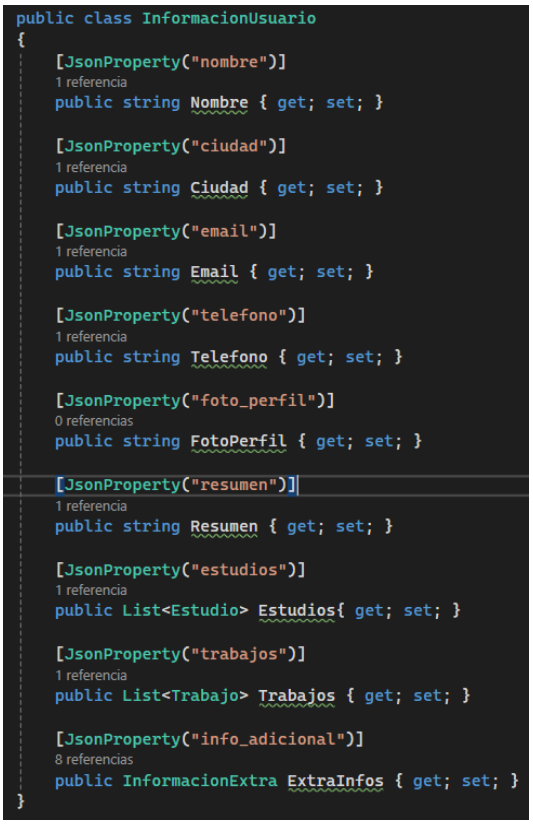


Imagen 23. Implementación de la deserialización de un JSON en C#.

7.3 Generación de PDF

Para la construcción del PDF del CV en C# existían distintos paquetes NuGet. El motivo de finalmente usar **QuestPDF** fue la posibilidad de previsualizar el PDF y modificarlo mientras los cambios se veían al momento, sin tener que reiniciar el programa.

QuestPDF es una **biblioteca** de código abierto para la creación de PDFs en .NET. Destaca por su enfoque declarativo y por permitir generar documentos complejos de manera programática.

Las principales características son su sintaxis declarativa que facilita la creación y mantenimiento de documentos, el diseño adaptativo que se ajusta automáticamente a diferentes tamaños de página y orientaciones, soporte para estilos y temas con distintas fuentes, colores, márgenes, etc. interactividad y vinculación con enlaces y anclas dentro del documento y la integración con .NET.

A continuación se puede ver una imagen del inicio de la creación del PDF:

```
Document.Create(contenedor =>
{
    contenedor.Page(pagina =>
    {
        pagina.Size(PageSizes.A4);

        pagina.Content().Padding(60).Column(columna =>
        {

            columna.Item().Text(usuario.Nombre).Bold().FontSize(18).AlignCenter().FontFamily(fontFamily);
```

Imagen 24. Código de inicio de la creación de un PDF con QuestPDF.

Para construir el currículum, se decidió dividirlo en tres partes. La primera parte hablaría de la formación académica, la segunda de la experiencia laboral y la tercera sería de información adicional, conteniendo información como cursos, certificados, idiomas, etc.

Para dividir cada una de estas secciones, manualmente se introduciría el título y una fina línea de subrayado, que en el documento generado quedaría como sigue:

EDUCACIÓN

Imagen 25. Apariencia de las divisiones de cada sección.

Posteriormente, en las dos primeras secciones se itera sobre la lista de elementos de cada sección. Como para los elementos de estas secciones existen objetos ya creados, cada elemento iterado contiene toda la información que se va a incluir de cada elemento.

En el caso de las formaciones: lugar de estudio (Universidad Politécnica de Valencia, por ejemplo), título de la formación (del grado universitario, por ejemplo), franja de tiempo en el que transcurre esa formación.

En el caso de las experiencias laborales, se incluye algo más de información: nombre de la empresa, puesto/función en la empresa, tiempo en la empresa y una descripción que se puede añadir cuando añades la experiencia laboral en el perfil, normalmente hablando de las tareas desarrolladas durante la experiencia.

Por último, la sección de información adicional incluirá (si existen) los idiomas, habilidades, cursos y certificaciones. Cada uno de ellos será presentado como un elemento precedido por "• " y el elemento del que se va a hablar (idiomas, por ejemplo) y a continuación toda la información que exista sobre ese elemento.

Toda esta información que se necesita para generar el CV se extrae mediante web scraping. Como ya se ha hablado, es una técnica para extraer información de las páginas web, y en este proyecto se hace uso de Selenium y BeautifulSoup para ello.

El proceso de extraer la información una vez el usuario de la aplicación ha introducido la URL de un perfil, o ha seleccionado uno de los perfiles listados es pasar desde C# la URL al script de Python como argumento de la llamada.

Una vez el script de Python ya tiene la URL, navega hasta el mismo y comienza a buscar la información en el perfil. Como podemos apreciar en la imagen, la página está dividida en secciones:

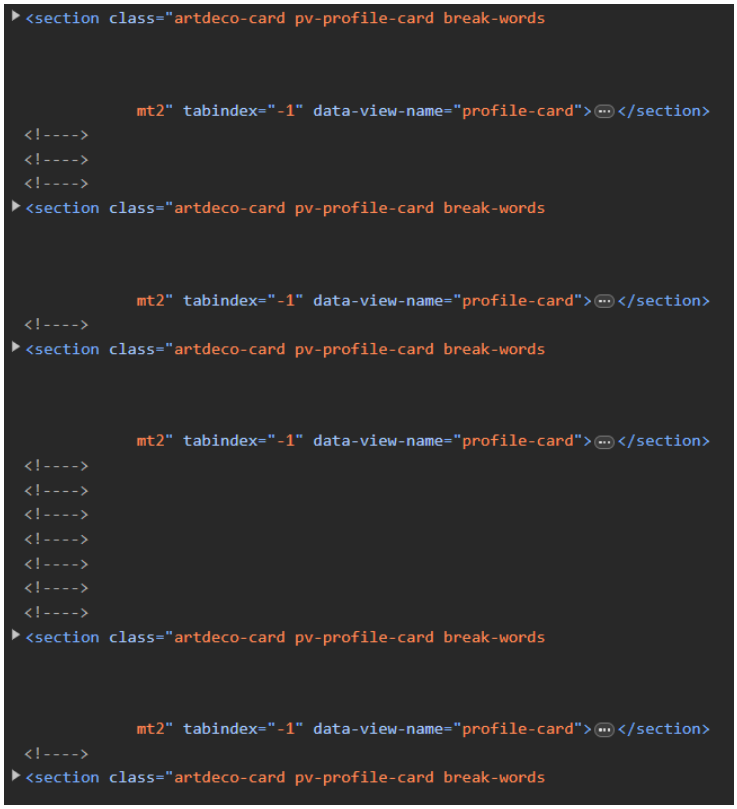


Imagen 26. Secciones en el HTML del perfil.

Cada una de estas secciones tiene un identificador que describe en qué consiste la sección (por ejemplo, “experience”), por lo que se puede comprobar que el identificador corresponde a una de las secciones de interés y entonces buscar la información correspondiente, ya que al haber muchas secciones hay secciones que no son interesantes.

Cuando hay mucha información de una de las secciones (por ejemplo hay muchos estudios) aparece algo como lo siguiente:

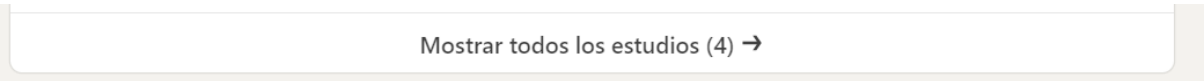


Imagen 27. Botón para ver toda la información de una sección.

Por tanto, como toda esa información es necesaria, con *Selenium* se comprueba la presencia o no de ese elemento, y en caso de estar presente, se hace clic en él y se extrae toda la información de la página a la que se navega. Para extraer la información se usa *BeautifulSoup*. Para ello, basta con inicializar el objeto BeautifulSoup parserizando el html del que queremos sacar información, y posteriormente usar el método find() o find_all(), pasándole como argumentos características para filtrar el elemento del que queremos sacar la información. A continuación se muestra un ejemplo:

```
listaEstudios_html = driver.page_source
soupList = BeautifulSoup(listaEstudios_html, 'html.parser')
div_items = soupList.find('div', attrs={"class": "pvs-list__container"})
items = div_items.find_all('li', attrs={"class": "artdeco-list__item"})
```

Imagen 28. Uso de los métodos de BeautifulSoup.

En este caso, tras haber pulsado en “Mostrar todos los estudios”, se estaban seleccionando todas las “tarjetas” que contenían información sobre alguna formación recibida por la persona del perfil. Como vemos, se pueden pasar distintos elementos como atributos y guardar en variables el resultado de la búsqueda.

Esto se hará para todas las secciones para las que sea necesario, ya que ninguna muestra toda la información en caso de que haya mucha.

La información extraída es aquella necesaria para la creación del documento, dejando parte de información sin extraer. Toda esta información se almacena en un *JSON* para que su acceso sea más sencillo, por lo que accediendo al atributo “nombre” del objeto, obtendremos directamente el nombre, por ejemplo.

7.4 Filtrado de perfiles

Para la funcionalidad del filtrado de perfiles, lo que se pretende en la aplicación es mostrar una **lista de perfiles** de acuerdo a unos filtros previamente introducidos por el usuario.

Para ello, se buscó en LinkedIn alguna forma ofrecida por la red social para filtrar perfiles. Tras un poco de búsqueda, se logró encontrar una función ofrecida por LinkedIn que podía ser útil. Por tanto, se buscaron los filtros más interesantes para un potencial usuario de la aplicación y se creó una pantalla para ello.

Además, se encontró la necesidad de dar la posibilidad al usuario de modificar los filtros utilizados. Para ello, se creó una pantalla navegable desde la pantalla de introducir los filtros en la que para cada tipo de filtro se muestra una lista de lo que se haya introducido, y el usuario puede seleccionar los elementos que quiera y eliminarlos.

Por último, también se da la opción al usuario de realizar la búsqueda según unas palabras clave (nombre, apellidos, título, etc.) por lo que en caso de no encontrar útiles los filtros anteriores, usar estas palabras clave.

Una vez sabiendo cómo el usuario iba a introducir toda la información, se gestionó la llamada al script de Python como ya se había hecho antes.

Mediante *Selenium*, se navega a la página de LinkedIn que iba a ser útil para filtrar los perfiles y se introduce la información. Una vez se buscan los perfiles que corresponden, la información guardada es el nombre del perfil que cumple los filtros y la URL de acceso al perfil.

Una vez la información ya ha vuelto a Visual Studio, se crea un objeto *ListBox* de WPF y se usan los nombres de los perfiles para mostrarlos en la lista. El usuario puede clicar en cualquier elemento de la lista y pulsar cualquiera de los botones de navegar al perfil o de generar el CV.

Para ambas acciones se utiliza la URL asociada al nombre. Para poder acceder a esta URL, se utiliza una opción que ofrece Visual Studio para la creación de las listas. A la lista a filtrar se le asigna una lista de objetos que representan cada uno de estos perfiles con el nombre y el link al perfil, y se dice que solo muestre el nombre. Cuando el usuario selecciona el elemento de la lista, está seleccionando el objeto en su totalidad, por lo que se puede acceder al link en la llamada, y seguir con la acción que procede. La forma de inicializar la lista es como sigue:

```
<ListBox x:Name="lbPerfilesFiltrados" FontFamily="Montserrat Medium" d:ItemsSource="{d:SampleData ItemCount=12}" 
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Grid Margin="2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"></ColumnDefinition>
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Text="{Binding nombre}"></TextBlock>
            </Grid>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Imagen 29. XAML de la creación de un ListBox.

Como se puede apreciar, en la declaración de la lista en el *XAML* se hace un *TextBlock* que está asociado con la propiedad nombre. Esto es permitido sin saber si el tipo de datos que se asociará a la lista tiene esta propiedad. Por lo tanto, en caso de que no tenga la propiedad se recibirá un error en la creación de la lista.

Para inicializar la lista con datos y asignar los elementos que deben ir en ella, se hará como sigue:

```
lbPerfilesFiltrados.ItemsSource = perfiles.perfilesFiltrados;
```

Imagen 30. Código para asignar elementos a la lista.

Asignando a la propiedad ItemsSource una lista, automáticamente se buscará la propiedad nombre de cada elemento de la lista para mostrarla. En este caso, “perfiles.perfilesFiltrados” es una lista de objetos que están formados por la propiedad nombre y por la propiedad url.

7.5 Otras consideraciones

Durante el desarrollo de la funcionalidad de generar CV a partir de la URL de un perfil de LinkedIn, hubo que ir matizando y cambiando algunas consideraciones que se habían tomado al inicio del proyecto que posteriormente no se pudieron implementar.

En primer lugar, LinkedIn requiere de tener una sesión iniciada para navegar por el perfil de un usuario. Esto no se tomó en cuenta en un inicio, por lo que hubo que decidir entre iniciar sesión manualmente al usar la aplicación (el usuario iniciaría su sesión) o tener unas credenciales de una cuenta vacía que solo se usaría para escapar de esta medida de seguridad. Finalmente, se acabó apostando porque el usuario iniciase sesión manualmente al abrir la aplicación, y guardando las cookies ya se podría usar esa sesión para el futuro, ya que iniciar sesión automáticamente con las herramientas que nos aporta Selenium hacía que LinkedIn hiciese comprobaciones de que el usuario no fuese un error por culpa de una actividad sospechosa, lo que generaba un nuevo problema. Esta decisión también se tomó ya que los usuarios pueden limitar el acceso a cierta información de tal forma que solo unos pocos puedan verla (por ejemplo, su red de contactos), por lo que podría pasar que cierta información que el usuario pudiese acceder desde su cuenta personal, con la aplicación fuera limitado ya que podría ser que esa información no estuviese disponible para la cuenta ficticia que se iba a usar.

En segundo lugar, en un inicio se planteó la posibilidad de usar una biblioteca Python llamada linkedin_api ya que brindaba solución al problema de sacar la información del perfil, pero al encontrarse desactualizada y tras muchas pruebas sin sacar realmente nada aprovechable, se decidió que lo mejor sería implementar un script desde cero para sacar la información.

En tercer lugar, también fue tarea complicada conseguir hacer funcionar la llamada al script de Python desde C#, ya que en un inicio se encontraron paquetes NuGet como IronPython que solucionaban esta tarea de forma muy sencilla, pero posteriormente la complejidad del script hizo que no se pudiese usar, ya que no incluía todas las bibliotecas Python requeridas. Tras un poco de investigación se encontró la solución de crear un nuevo proceso, que como previamente se ha explicado esa fue la decisión utilizada.

También hubo que gestionar el uso del script, ya que se decidió reunir todos los métodos de extracción de información en un mismo fichero Python, por lo que cuando se creaba un proceso para llamar al fichero había que gestionar cómo invocar el método deseado.

Para ello, se decidió pasar como primer parámetro un código y en función de su valor llamar a un método a otro. Esto se hizo de la siguiente manera:

```python
if __name__ == "__main__":
    # Leer el argumento pasado desde C#
    digit = sys.argv[1]
    # Estudiar causísticas
    if digit == "0":
        url = sys.argv[2]
        obtenerInformacionUsuario(url)
    elif digit == "1":
        inicioSesionManual()
    elif digit == "2":
        ubicaciones = json.loads(sys.argv[2])
        empresas_actuales = json.loads(sys.argv[3])
        empresas_anteriores = json.loads(sys.argv[4])
        educaciones = json.loads(sys.argv[5])
        sectores = json.loads(sys.argv[6])
        servicios = json.loads(sys.argv[7])

        filtrarPerfiles(ubicaciones, empresas_actuales, empresas_anteriores, educaciones, sectores, servicios, [])
    elif digit == "3":
        claves = sys.argv[2]
        filtrarPerfiles([], [], [], [], [], [], claves)
```

Imagen 31. Gestión del llamado del método.

Como podemos apreciar, se lee el primer argumento ya que siempre será el mismo, el código. Después, al analizar cada caso particular, según en qué situación estemos vendrán más argumentos que habrá que seguir leyendo conforme vaya tocando cada vez, y por ello, en el tercer caso por ejemplo, se leen tantos argumentos, mientras que en los demás casos no.

CAPÍTULO 8
Pruebas y calidad de software

Esta sección es una sección fundamental para asegurar que la aplicación desarrollada cumple con los requisitos especificados y funciona correctamente en todos los escenarios posibles. Se describirán las estrategias y métodos utilizados para la verificación y validación de software, asegurando la calidad y confiabilidad. Los objetivos son:

- **Verificar**: confirmar que el software ha sido desarrollado correctamente según las especificaciones técnicas.
- **Validar**: asegurar que el software cumple con las necesidades y expectativas del usuario final.
- **Mantener la calidad**: establecer procedimientos para mantener y mejorar la calidad del software a lo largo del tiempo.

Para la explicación completa de la sección, esta se dividirá en dos apartados siguiendo el título del capítulo. En un primer apartado se hablará de las pruebas realizadas, y en un segundo apartado de aspectos relacionados con la calidad de software.

8.1 Pruebas de aceptación

Estas pruebas se realizan para **validar** que el software cumple con los requisitos del usuario.

Consiste en la creación de casos de prueba basados en los requisitos funcionales definidos, ejecución de pruebas con usuarios reales y validación de los resultados contra las expectativas del usuario. Esto ya ha sido definido previamente junto a los requisitos, por lo que bastará para la aplicación de los mismos.

Para la realización de las pruebas, se ofreció la aplicación a un conjunto de usuarios con distintos conocimientos en informática y así poder recibir distintos tipos de feedback.

Un cambio importante introducido a raíz de estas pruebas fue la página de **gestión de filtros**. Algún usuario nos dijo que tras no encontrar ningún perfil interesante y querer “rebajar” el nivel de los filtros, no entendían de qué forma lo podían hacer estando la aplicación como estaba.

Por ello, se sugirió la creación de una pantalla en la que se mostrasen todos los filtros aplicados y directamente ahí hacer la gestión de los mismos, pudiendo eliminar algunos de ellos.

Dejando este matiz aparte, las pruebas fueron exitosas y se extrajo información útil. Las correspondientes a los requisitos funcionales se completaron correctamente. Para las de los no funcionales, se introdujeron algunos cambios, por ejemplo, para el tema del rendimiento.

El script de extracción de datos espera a que cargue la página en ocasiones de manera manual mediante un time.sleep(), por lo que el tiempo de espera se redujo con la intención de tener un mejor rendimiento.

Además, se completaron casos de manejo de errores que no habían sido gestionados, y se terminó de comentar el código de una manera explicativa para una mejor comprensión. De esta forma, las pruebas de aceptación fueron completadas correctamente.

8.2 Calidad de software

En búsqueda de una mayor calidad de software en la aplicación, se ha considerado el uso de una herramienta llamada *SonarQube*.

SonarQube es una plataforma para **evaluar** código fuente. Hace un **análisis estático** de código fuente con el uso de distintas herramientas. Comprueba el estilo, busca bugs potenciales, etc. obteniendo métricas que pueden ayudar a la calidad del código de un programa.

El primer análisis del código desarrollado mostró la siguiente imagen:

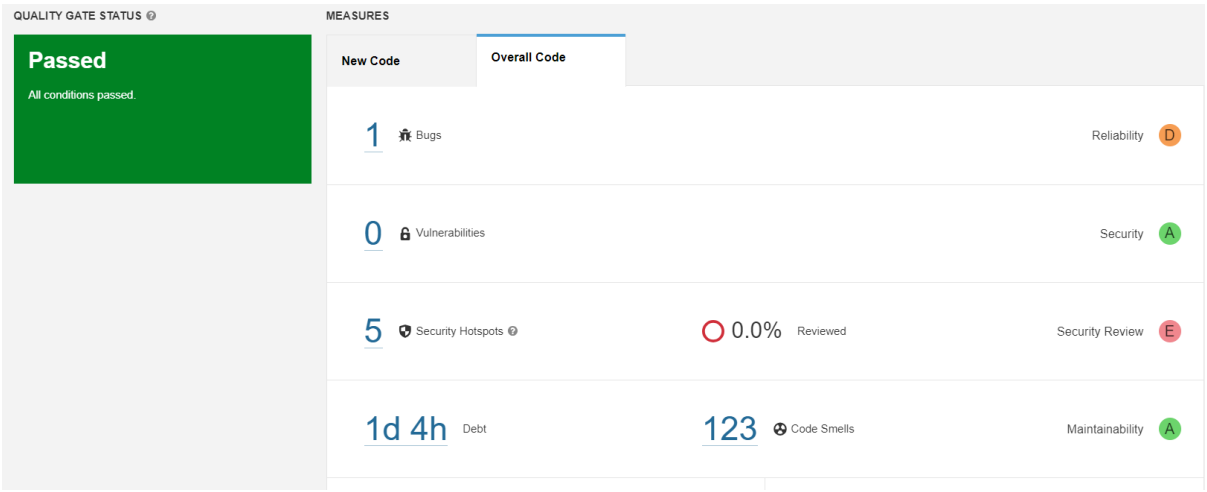


Imagen 32. Captura de SonarQube tras el primer análisis.

Como se puede apreciar, el análisis ha detectado un *bug*, lo que hace bajar claramente la nota en fiabilidad. Además, al no haber hecho pruebas unitarias y tener varios *Security Hotspots*, encontramos esa nota de "E" en Security Review. Por último, SonarQube encuentra 123 Code Smells.

Un **Code Smell** es un término utilizado en el desarrollo de software para hablar de indicios en el código que pueden sugerir un problema más profundo. No tienen por qué ser errores explícitos o defectos, sino que pueden ser señales de que algo en el diseño podría estar mal y necesitase refactorización.

Los Security Hotspots, que según el análisis son 5, son áreas en el código fuente que podrían ser vulnerables a ataques de seguridad. En el caso del análisis, los 5 nacen del mismo problema. Al crear el proceso en C#, como FileName se utiliza "python", por lo que se busca la ubicación del ejecutable en las variables de entorno del sistema, en concreto en la variable PATH. Es por ello que si un atacante puede tener las variables bajo control, tendríamos un problema de seguridad en este punto. Lo bueno de usar SonarQube es que toda esta información te la da el propio análisis, así como una solución al problema. Tras incluir la solución en el código, la creación del proceso quedaría como sigue:

```
ProcessStartInfo psi = new ProcessStartInfo
{
    FileName = @"C:\Python310\python.exe",
    Arguments = $"\"{scriptPath}\" {argumentos}",
    RedirectStandardOutput = true,
    RedirectStandardError = true,
    UseShellExecute = false,
    CreateNoWindow = true,
    StandardOutputEncoding = Encoding.UTF8
};
```

Imagen 33. Implementación de la creación de un proceso tras el cambio.

Además, tras realizar algunos cambios, el análisis de SonarQube quedaría como sigue:

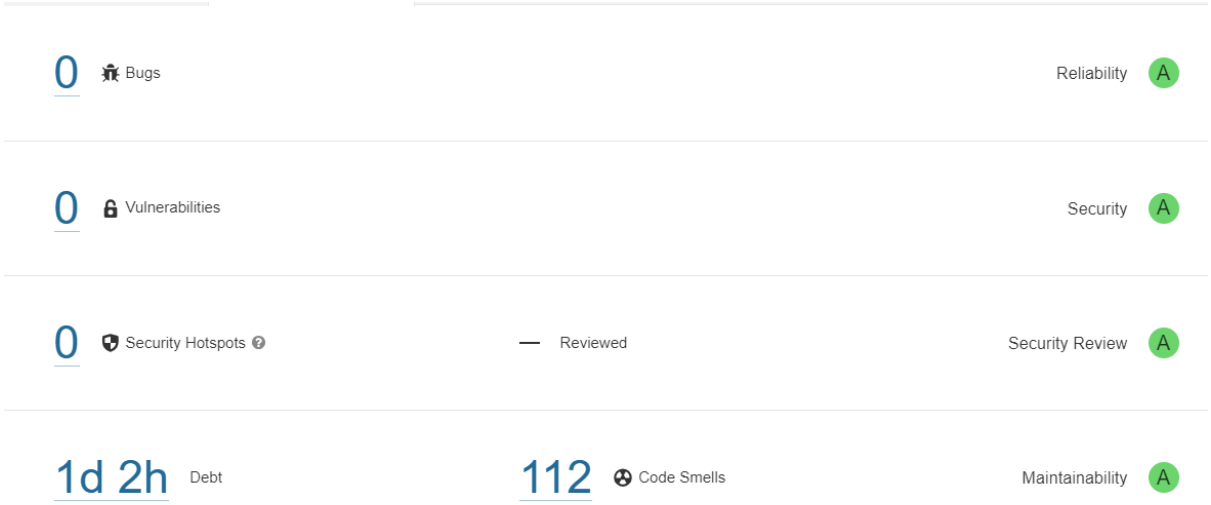


Imagen 34. Captura de SonarQube tras el segundo análisis.

Como podemos apreciar, el bug ha desaparecido y ahora en esa sección se ha obtenido también una "A". Además, se observa que algunos Code Smells han desaparecido, pero haría falta bastante más trabajo para dejarlos en 0. Ese trabajo estimado se refleja en Debt. Este término se utiliza para hablar de la deuda técnica acumulada en el proyecto, es decir, el costo adicional que puede suponer en un futuro las decisiones de desarrollo tomadas que a corto plazo parecían beneficiosas pero a largo plazo serán compromisos en términos de mantenibilidad, estabilidad, etc. A menudo surgen de compromisos técnicos que se deciden posponer o ignorar en el desarrollo.

CAPÍTULO 9
Mantenimiento y gestión de versiones

Durante esta sección se buscará explicar las herramientas utilizadas para la gestión de versiones y mantenimiento del código en el proyecto. Tras haber hablado previamente de la implementación, es importante explicar cómo ese código se ha gestionado.

El objetivo de esta sección será proporcionar una explicación clara de todo lo relacionado con este tema en relación al proyecto.

La gestión de versiones es algo muy importante en los proyectos de software, y por ello también es importante esta sección en la que se habla sobre esto. Para ello, en primer lugar se dará una extensa explicación de la herramienta utilizada, para después hablar del uso específico de la misma en el proyecto.

9.1 Azure DevOps: Visión general

Azure DevOps es una plataforma de servicios de desarrollo colaborativo. Proporciona un conjunto de herramientas para la gestión de proyectos, control de versiones, integración continua, entrega continua, etc. todo esto soportado mediante repositorios, pipelines, boards, etc.

Para el control de versiones encontramos los **repositorios Git**, gestionando eficientemente el código. En estos repositorios tenemos ramas para facilitar el desarrollo paralelo y la integración de nuevas características.

También existe la posibilidad de hacer commits y pull requests. En los commits, los cambios se pueden describir para tener una idea de qué se esta añadiendo. Mediante las pull requests, se puede revisar el código antes de integrarlo en la rama para asegurarse de que todo está tal y como debe.

Por último, también se pueden crear etiquetas para marcar versiones específicas de la solución.

Azure DevOps también ofrece herramientas para asegurar CI/CD mediante las pipelines. Hay principalmente dos tipos:

- **Build Pipelines**: sirven para la automatización de la compilación y prueba del código cada vez que se realiza un commit.
- **Release Pipelines**: sirven para la automatización del despliegue de la aplicación en diferentes entornos como desarrollo o producción.

En conclusión, Azure DevOps permite una gestión eficiente y organizada del ciclo de vida del desarrollo del software, mediante herramientas para el control de versiones, CI/CD, etc.

9.2 Azure DevOps: Uso específico

En el proyecto, Azure DevOps no ha sido exprimido para utilizar toda su funcionalidad, ya que en un proyecto de estas características, de tamaño reducido y con un solo programador no era necesario algunas de esas características.

A pesar de que en un primer momento se reflexionase sobre el uso de más características, finalmente sólo se usaron los **repositorios Git** para almacenar las versiones del código y subir los cambios introducidos en el mismo.

Además de servir como **copia de seguridad** del código, permitió que el cambiar de lugar de desarrollo no influyera en la disponibilidad del mismo, ya que con solo iniciar sesión en Azure DevOps ya se podía conectar Visual Studio con el repositorio y así tener todo el código disponible para probarlo, modificarlo, etc.

En la siguiente imagen se puede apreciar las distintas funcionalidades que nos aporta Azure DevOps y, en general, el proyecto del TFG.

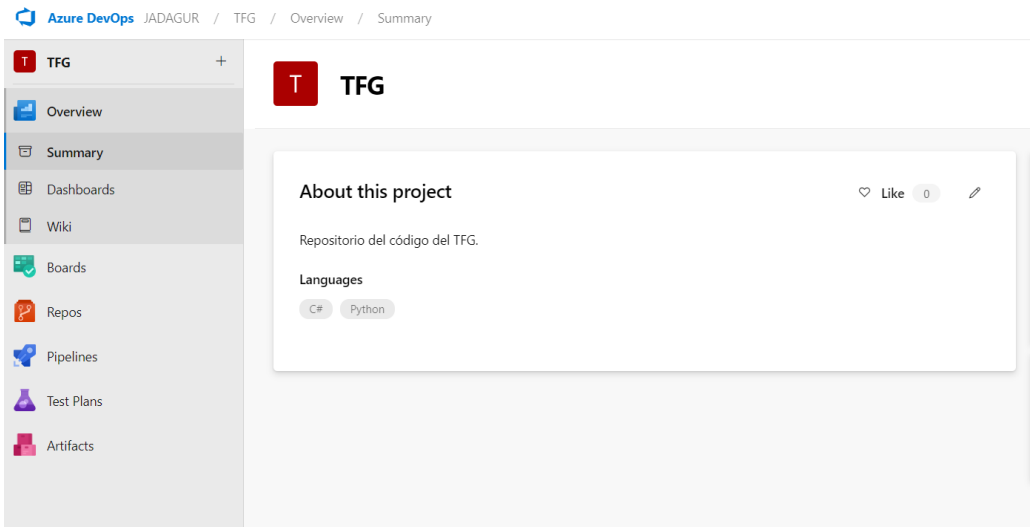


Imagen 35. Captura del proyecto de Azure DevOps.

CAPÍTULO 10
Aplicación final

En esta sección se proporcionará una vista de las distintas **interfaces** de la aplicación final. A pesar de tener unos prototipos previos, es común en cualquier proyecto que estos prototipos sean sometidos a cambios.

El objetivo de esta sección será explicar de manera completa la función de cada interfaz, así como aportar una captura de la misma. Esto es importante para conocer los distintos cambios que se han dado a lo largo del proceso de implementación debido a los problemas encontrados. Para ello, se irá haciendo un recorrido completo a lo largo de la aplicación hablando de cada una de las interfaces.

10.1 Inicio

Cuando el usuario inicia la aplicación, lo primero que encuentra es un botón que dice “Empezar”. Esta acción iniciará un buscador en la pantalla de login de LinkedIn. El usuario manualmente iniciará sesión con sus credenciales, y la aplicación extraerá las cookies. Esto se hace para posteriormente, cuando se necesiten obtener datos de LinkedIn, mediante las cookies podamos iniciar sesión automáticamente para no ser entorpecidos por la seguridad de la red social.

Imagen 36. Interfaz inicial de la aplicación.

10.2 Seleccionar acción

Esta pantalla, que no ha sufrido ningún cambio en comparación de los prototipos definidos previamente, servirá como puente desde el login hasta la acción que el usuario quiere llevar a cabo.

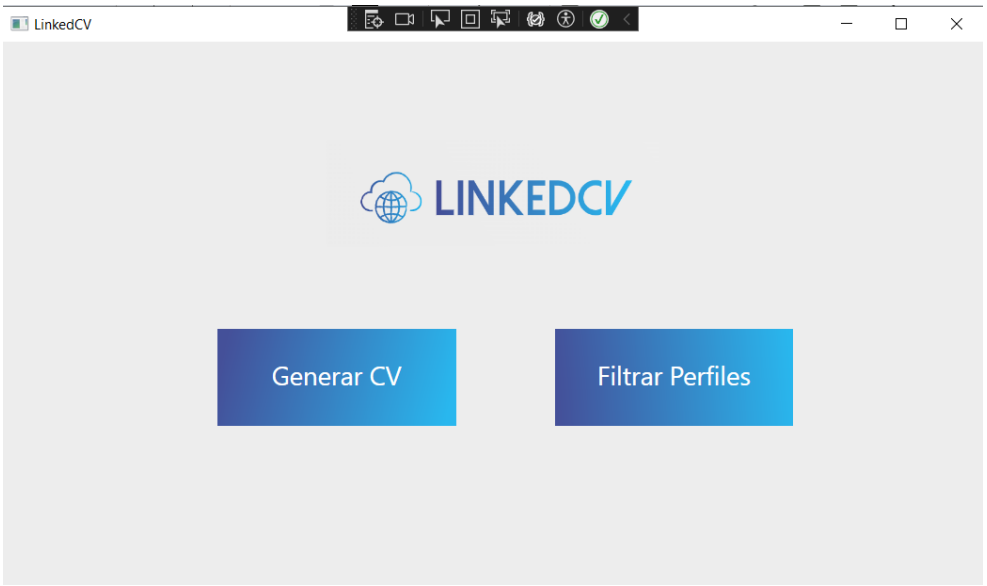


Imagen 37. Interfaz para seleccionar acción en la aplicación.

10.3 Generar CV

Esta pantalla mantiene un diseño muy similar al propuesto en la fase de prototipación. Sin embargo, en la implementación se decidió cambiar el parámetro dado por el usuario. En lugar de aportar el nombre del usuario del que se quiere generar el currículum, se aporta la URL al perfil del mismo. Con esto, nos ahorramos problemas que podrían surgir si el nombre aportado por el usuario no coincidía con el que el usuario del que se iba a generar el CV, por ejemplo, si el usuario solo tiene un apellido en LinkedIn y en la aplicación se introducen dos. Por ello, la interfaz quedaría como sigue:

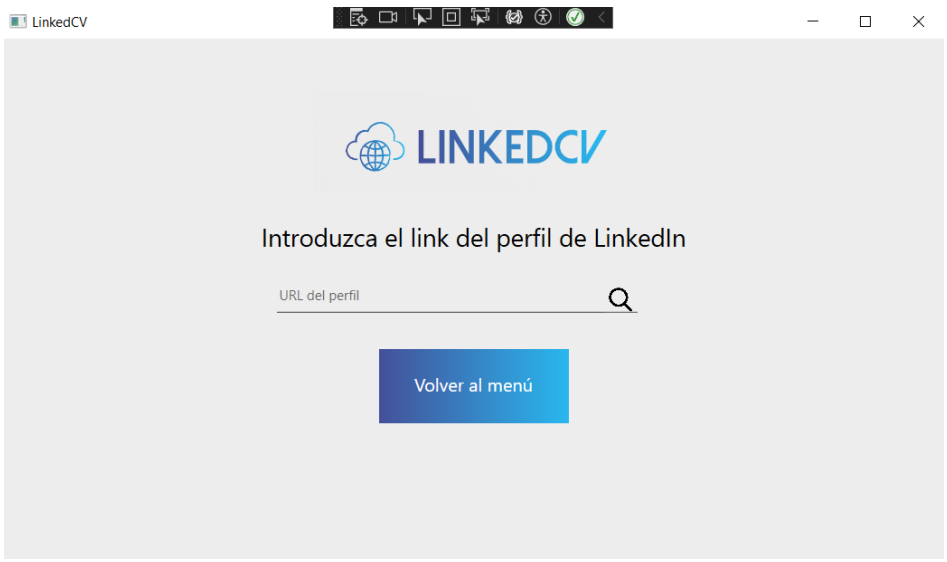


Imagen 38. Interfaz de generar CV en la aplicación.

10.4 Pantalla de carga

Esta pantalla se mostrará mientras se produce la extracción de los datos del usuario en el caso de que quiera generar un CV, y mientras se extraen los perfiles compatibles en el caso de que quiera filtrar una serie de perfiles. Esto se debe a que si no el usuario vería la aplicación congelada, y al ser un script que tarda unos pocos segundos en ejecutarse merecía la pena crear esta pantalla.

Una vez terminado el script, salta automáticamente a la siguiente pantalla.

Imagen 39. Pantalla de carga en la aplicación.

10.5 Elegir tipo de CV

Esta pantalla tiene la función de ofrecer al usuario los distintos tipos de CV que puede generar. En esta primera versión, las opciones serán con y sin foto. También puede navegar al menú.

En caso de seleccionar uno de los tipos de CV, aparecerá una pantalla de carga similar a las anteriores mientras se genera el CV, antes de navegar automáticamente a la siguiente pantalla.

Imagen 40. Interfaz de elegir el tipo de CV en la aplicación.

10.6 Gestionar CV generado

Una vez ya se haya generado el CV, el usuario tendrá la opción de decidir qué hacer. Podrá descargar el CV, o ver cómo ha quedado antes de decidir descargarlo. También podrá navegar al menú inicial.

Imagen 41. Interfaz de gestionar el CV en la aplicación.

10.7 Filtrar perfiles

En caso de pulsar el botón de filtrar perfiles en la pantalla de seleccionar acción, el usuario navegará hasta esta pantalla.

En ella encontrará una serie de campos a rellenar en función de aquello necesario por el usuario, y al finalizar seleccionará el botón de aplicar para entonces proceder a la función de filtrado, apareciendo una pantalla de carga similar a la mostrada anteriormente mientras se procesa el filtrado.

Los perfiles se podrán filtrar en función de distintos parámetros, como por ejemplo la ubicación, la empresa actual, las empresas anteriores, las instituciones educativas, el sector al que pertenece y las categorías de servicios.

En esta pantalla también encontrará una serie de botones para, en primer lugar, modificar los filtros que ha añadido (eliminar alguno de estos filtros) o filtrar los perfiles aportando una serie de palabras clave en vez de los campos que se le muestran en esta pantalla.

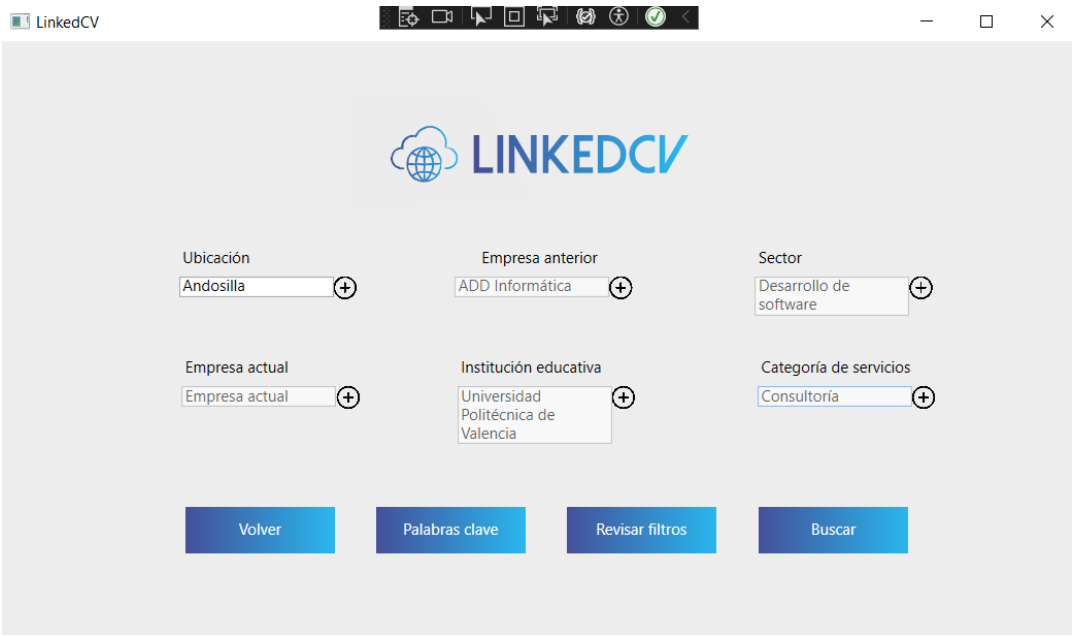


Imagen 42. Interfaz de añadir filtros en la aplicación.

10.8 Gestionar filtros

En esta pantalla habrá una lista con todos los filtros correspondientes a cada campo por el cual se puede filtrar, por lo que el usuario podrá eliminar aquellos filtros que no necesite.

En caso de pulsar el botón cancelar esta eliminación no se hará efectiva. Después, el usuario podrá volver a la pantalla de los filtros y seguir añadiendo filtros antes de buscar los perfiles que correspondan.

Imagen 43. Interfaz de gestionar filtros en la aplicación.

10.9 Palabras clave

En esta pantalla el usuario tendrá una serie de TextBox que rellenar con palabras clave relacionadas con distintos campos: nombre, apellidos, título, empresa y educación. Al pulsar el botón buscar, comenzará la búsqueda de los perfiles que puedan corresponder con esas palabras clave como se haría en el caso de que se buscase según los filtros.

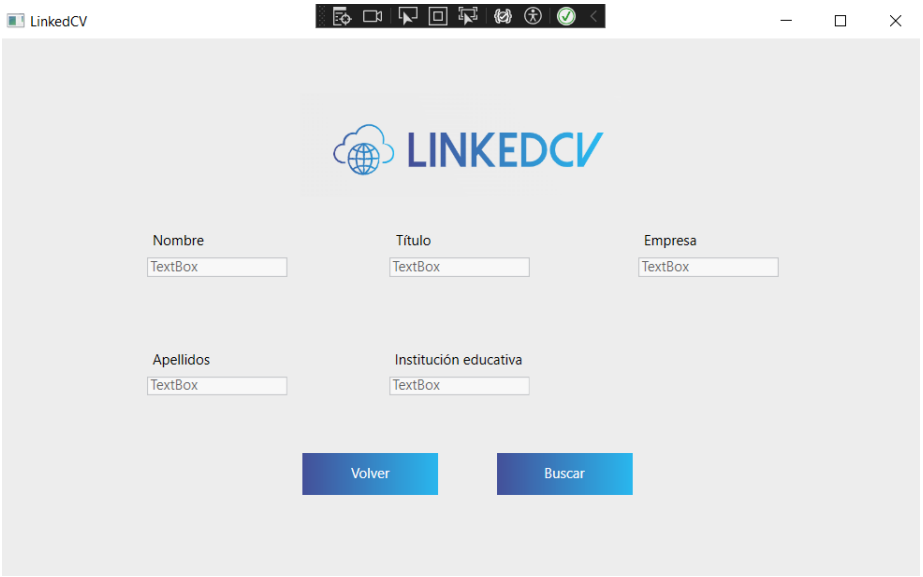


Imagen 44. Interfaz para introducir palabras clave en la aplicación.

10.10 Listar perfiles

Tras aplicar los filtros, todos los resultados compatibles serán mostrados en una lista con distintas tarjetas. En cada tarjeta aparecerá el nombre de cada perfil compatible, así como un botón para navegar hasta el perfil de LinkedIn del usuario, y otro para navegar hasta la pantalla de generar CV y proceder como ya se ha explicado antes.

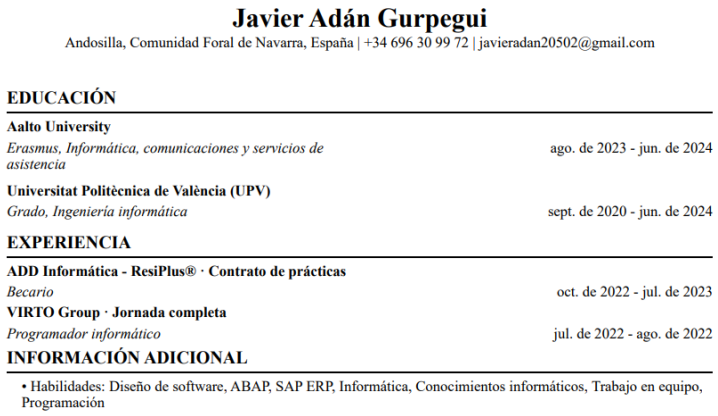
Imagen 45. Interfaz de listar perfiles en la aplicación.

10.11 Currículum generado

Como ya se ha explicado, el objetivo final de la aplicación consiste en la generación de un CV a partir de un perfil de LinkedIn, accedido mediante su URL o usando filtros.

La estructura de un currículum también ha sido explicada previamente, dividiéndose principalmente en educación, experiencia laboral e información adicional.

Por ejemplo, el currículum que se genera a partir de mi perfil se ve tal que así:



Javier Adán Gurpegui

Andosilla, Comunidad Foral de Navarra, España | +34 696 30 99 72 | javieradan20502@gmail.com

EDUCACIÓN

Aalto University
Erasmus, Informática, comunicaciones y servicios de asistencia — ago. de 2023 - jun. de 2024

Universitat Politècnica de València (UPV)
Grado, Ingeniería informática — sept. de 2020 - jun. de 2024

EXPERIENCIA

ADD Informática - ResiPlus® · Contrato de prácticas
Becario — oct. de 2022 - jul. de 2023

VIRTO Group · Jornada completa
Programador informático — jul. de 2022 - ago. de 2022

INFORMACIÓN ADICIONAL

• Habilidades: Diseño de software, ABAP, SAP ERP, Informática, Conocimientos informáticos, Trabajo en equipo, Programación

Imagen 46. Currículum generado por la aplicación.

Como se puede apreciar, al tener poca información añadida en el perfil de LinkedIn, el currículum obtenido es muy simple. En caso de tener descripciones para cada experiencia laboral, aparecería también en el currículum.

En mi caso, también falta información adicional como podrían ser los idiomas, certificaciones y cursos, teniendo en este caso un currículum más completo.

En el caso de generar el currículum con foto, se vería tal que así:

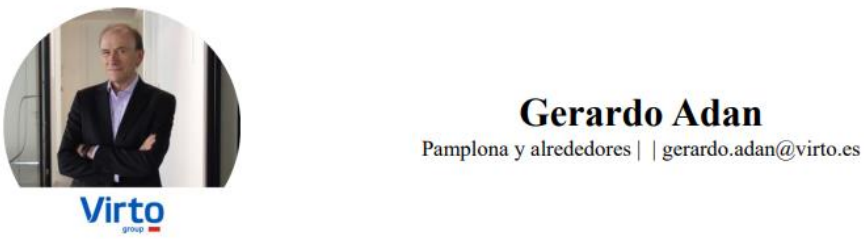


Imagen 47. Currículum con foto generado por la aplicación.

Como podemos ver, la foto se recorta en forma circular y la cabecera se desplaza ligeramente, mientras que lo demás queda exactamente igual.

CAPÍTULO 11
Futuras mejoras

Esta sección servirá para hablar de todo aquello que no ha podido ser completado debido a los problemas de tiempo y alcance del proyecto que se dan en cualquier proyecto de software, y más en uno como este con tiempo limitado.

El objetivo es explicar cuál sería el enfoque futuro para la aplicación, por dónde podría ir evolucionando con el objetivo final de crear una mejor aplicación para el usuario, siendo importante para dar contexto del estado actual de la aplicación respecto a las futuras mejoras que ya se contemplan introducir. Para ello, se irán introduciendo en distintos puntos las mejoras, explicando el por qué y qué cambios podría introducir en la aplicación.

11.1 Personalizar CV

Una de las mejoras que se podría implementar es dar a elegir al usuario las distintas partes que se incluyen en el CV. En esta mejora no se habla de los contenidos como tal, sino más bien de la estructura del currículum.

Con esta mejora, el usuario podría, antes de que se generase el currículum, elegir si incluir los estudios y el trabajo, no incluir los cursos realizados, incluir los idiomas, etc. así como el orden de los elementos presentes. Siempre y cuando suponiendo que toda esa información esté presente en su LinkedIn.

De esta forma, se podría obviar la pantalla de seleccionar el tipo de currículum, y ofrecer la foto como una opción de elemento a incluir en el currículum.

Así se ofrecería cierta personalización del currículum al usuario, haciendo que no todos los CV generados fueran iguales en estructura y diseño.

11.2 Plantillas de CV

Algo muy común a la hora de crear el CV es usar una plantilla de alguna página de diseño de documentos (Canva por ejemplo) para así tener un currículum más llamativo.

Esta mejora se centraría en ofrecer plantillas para la creación de los documentos, y el usuario podría elegir entre las proporcionadas por la aplicación.

También se podría valorar la opción de que el propio usuario incluyese la plantilla deseada, y generar el documento directamente sobre esa plantilla.

11.3 Búsqueda de empleo

Otra mejora que se podría incluir es dar la opción de buscar empleo directamente desde la aplicación.

Siguiendo un modelo similar al del filtrado de perfiles, se podría hacer una búsqueda de ofertas de trabajo. El usuario tendría que rellenar ciertos cambios con información relacionada con el empleo en cuestión, y la aplicación mostraría la lista de empleos relacionados con esos parámetros.

Después, la aplicación ofrecería la opción de navegar a la oferta o incluso postularse a la oferta directamente con solo pulsar un botón en la aplicación.

De esta forma, los usuarios podrían tanto generar el CV necesario para solicitar un trabajo, como directamente buscar ofertas que se amoldan con sus preferencias.

11.4 Roles

Esta mejora consiste en la limitación en el acceso a determinadas funcionalidades según el rol del usuario de la aplicación. Esto quiere decir que cuando el usuario inicia sesión en LinkedIn, se comprobaría si es por ejemplo un usuario buscando trabajo, reclutador, etc.

De esta manera, por ejemplo un reclutador sólo podría filtrar perfiles y generar CVs, mientras que un usuario buscando trabajo no tendría disponible la funcionalidad de filtrar perfiles.

Con esta modificación también se podría enfocar la aplicación de tal manera que se pudiese obtener rentabilidad de la misma.

En vez de obtener el rol de LinkedIn, cada usuario tendría un rol asociado a la suscripción que hubiera pagado, y de esa manera tendría acceso a unas funcionalidades u otras.

CAPÍTULO 12
Conclusiones

El objetivo de este proyecto fue desarrollar una herramienta de generación automática de CVs y filtrado de perfiles para facilitar estas tareas a reclutadores y a gente que pudiese necesitar un CV para buscar trabajo.

A través de la implementación de algoritmos de web scraping y con el uso de WPF y C# se ha conseguido desarrollar una aplicación cubriendo ambas funcionalidades clave.

La realización del mismo ha servido para afianzar conocimiento en distintas áreas relacionadas con el grado, así como recordar otras. También ha servido para demostrar la importancia del conocimiento de los distintos procesos software, la aplicación correcta del proceso software y el conocimiento de múltiples tecnologías entre otros.

Sin embargo, también se han encontrado múltiples desafíos durante su desarrollo, al tener que improvisar cuando ciertas decisiones tenían que ser modificadas debido a situaciones inesperadas, como el no funcionamiento de linkedin_api o la gestión de la llamada del script de Python.

Otras decisiones del desarrollo debido a los conocimientos previos han sido factores limitantes que se podrían mejorar. Uno de ellos, y en mi opinión el más importante, es la limitación de uso de la aplicación a usuarios Windows por el uso de WPF.

A pesar de todo esto, el resultado de la aplicación final es bueno y estoy contento con lo conseguido.

Bibliografía

(n.d.). Refactorización y patrones de diseño. https://refactoring.guru/es

(n.d.). QuestPDF. https://www.questpdf.com/

Amazon. (n.d.). *¿Qué es .Net? - Explicación de Dotnet - AWS*. Amazon AWS. https://aws.amazon.com/es/what-is/net/

Anderson, D. J. (2008, Diciembre 30). The Kanban Primer: A Cultural Evolution in Software. *2009-01*, 1-7. https://www.stickyminds.com/better-software-magazine/kanban-primer-cultural-evolution-software

Beautiful Soup : ¿cómo aprender a hacer web scraping en Python? (2022, septiembre 1). DataScientest.com. https://datascientest.com/es/beautiful-soup-aprender-web-scraping

CENTUM Solutions, S.L. (n.d.). *Captación y especificación de requisitos en proyectos de software*. CENTUM Digital. https://centum.com/captacion-y-especificacion-de-requisitos-en-proyectos-de-software/

▷ *Diagrama de casos de uso. Teoría y ejemplos*. (n.d.). diagramas UML. https://diagramasuml.com/casos-de-uso/

▷ *Diagrama de clases. Teoria y ejemplos.* (n.d.). diagramas UML. https://diagramasuml.com/diagrama-de-clases/?utm_content=cmp-true

▷ *Diagrama de secuencia*. (n.d.). diagramas UML. https://diagramasuml.com/secuencia/

Especificación de Requisitos según el estándar de IEEE 830. (2008). In Universidad Complutense de Madrid (Ed.), (pp. 1-9). https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

Figma. (n.d.). SaaS Rank. https://saasrank.es/producto/figma/

Figma, Inc. (2016). Figma: The Collaborative Interface Design Tool. https://www.figma.com/

Ibero, J. (n.d.). *Arquitectura Cliente/Servidor: modelo de 3 capas*. IberAsync.es. https://iberasync.es/arquitectura-cliente-servidor-modelo-de-3-capas/

Jibaru. (2023, May 13). *Cómo DESCARGAR e INSTALAR Sonarqube y ANALIZAR proyecto*. YouTube. https://www.youtube.com/watch?v=5UoygWLrBqo

Kinsta. (2022, July 28). *¿Qué Es el Web Scraping? Cómo Extraer Legalmente el Contenido de la Web*. Kinsta. https://kinsta.com/es/base-de-conocimiento/que-es-web-scraping/

LinkedIn API. (2020). Lix. https://lix-it.com/pages/linkedin-api

LinkedIn Email Finder API. (2023, November 3). Prospeo. https://prospeo.io/api/linkedin-email-finder

LinkedIn Messaging API: The Key to Next-Level Sales and Recruiting. (2020). Unipile. https://www.unipile.com/communication-api/messaging-api/linkedin-api/

MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS. (n.d.). In W. W. Royce (Ed.). https://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

Metodologías Agiles vs Tradicionales | POSITIVO. (n.d.). Agencia de Marketing Digital y Publicidad en Bilbao | POSITIVO. https://www.positivo.pro/blog/metodologias-agiles-vs-tradicionales/

Microsoft. (n.d.). Visual Studio: IDE y Editor de código para desarrolladores de software y Teams. https://visualstudio.microsoft.com/es/

Microsoft. (n.d.). *Azure DevOps Services*. Microsoft Azure. https://azure.microsoft.com/es-es/products/devops

Microsoft. (n.d.). *C# | Lenguaje de programación moderno y de código abierto para .NET*. Dot.Net. https://dotnet.microsoft.com/es-es/languages/csharp

Microsoft. (n.d.). *¿Qué es .NET? Una plataforma para desarrolladores de código abierto.* Dot.Net. https://dotnet.microsoft.com/es-es/learn/dotnet/what-is-dotnet

Microsoft. (2002). *Página oficial de LinkedIn*. LinkedIn. https://www.linkedin.com/feed/

Microsoft. (2015). *Visual Studio Code*. Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/

Microsoft. (2023, October 13). *Descripción de Windows Presentation Foundation - WPF .NET*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/desktop/wpf/overview/?view=netdesktop-8.0

Microsoft. (2024, June 13). *IDE de Visual Studio 2022: herramienta de programación para desarrolladores de software*. Visual Studio. https://visualstudio.microsoft.com/es/vs/

Ortego Delgado, D. (2017, marzo 29). *Qué es C#: Introducción*. OpenWebinars. https://openwebinars.net/blog/que-es-c-introduccion/

Patrón Singleton. (n.d.). Refactoring.Guru. https://refactoring.guru/es/design-patterns/singleton

Proxycurl. (2020). API for software engineers to build data-driven applications. https://nubela.co/proxycurl/

The Python programming language for .NET. (2023, Julio 12). IronPython.net /. https://ironpython.net/

Qué es la especificación de requisitos: definición, mejores herramientas y técnicas | Guía - Soluciones Visure. (n.d.). Visure Solutions. https://visuresolutions.com/es/blog/requirements-specification/

Qué es Selenium. (2023, February 22). Sentrio. https://sentrio.io/blog/que-es-selenium/

Qué es SonarQube: Verifica y analiza la calidad de tu código. (2021, December 15). Sentrio. https://sentrio.io/blog/que-es-sonarqube/

¿Qué es un diagrama de flujo? (n.d.). Lucidchart. https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-flujo

Quirk, T. (2018, Septiembre 9). *Documentación de linkedin_api*. GitHub. https://github.com/tomquirk/linkedin-api/

Rankin de los sistemas operativos más utilizados para PC... (2016, Junio). ResearchGate. https://www.researchgate.net/figure/Figura-1-Rankin-de-los-sistemas-operativos-mas-utilizados-para-PC-Fuente-Montes-2015_fig1_318359703

Schwaber, K. (n.d.). *What is Scrum?* Scrum.org. https://www.scrum.org/learning-series/what-is-scrum/

Steve Easterbrook. (2000). Requirements Engineering: A Roadmap. In B. Nuseibeh (Ed.), (pp. 1-10). https://www.cs.toronto.edu/~sme/papers/2000/ICSE2000.pdf

Web Scrape with Selenium and Beautiful Soup. (n.d.). Codecademy. https://www.codecademy.com/article/web-scrape-with-selenium-and-beautiful-soup

Ziabek, M. (2021, marzo 1). *QuestPDF/QuestPDF: QuestPDF is a modern open-source .NET library for PDF document generation. Offering comprehensive layout engine powered by concise and discoverable C# Fluent API. Easily generate PDF reports, invoices, exports, etc.* GitHub. https://github.com/QuestPDF/QuestPDF

Anexo

Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Tabla 1. Relación de los ODS con el proyecto.

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**x**
ODS 2. **Hambre cero.**				**x**
ODS 3. **Salud y bienestar.**				**x**
ODS 4. **Educación de calidad.**				**x**
ODS 5. **Igualdad de género.**		**x**		
ODS 6. **Agua limpia y saneamiento.**				**x**
ODS 7. **Energía asequible y no contaminante.**				**x**
ODS 8. **Trabajo decente y crecimiento económico.**	**x**			
ODS 9. **Industria, innovación e infraestructuras.**			**x**	
ODS 10. **Reducción de las desigualdades**.		**x**		
ODS 11. **Ciudades y comunidades sostenibles**.				**x**
ODS 12. **Producción y consumo responsables.**				**x**
ODS 13. **Acción por el clima.**				**x**
ODS 14. **Vida submarina.**				**x**
ODS 15. **Vida de ecosistemas terrestres.**				**x**
ODS 16. **Paz, justicia e instituciones sólidas.**				**x**
ODS 17. **Alianzas para lograr objetivos.**				**x**

El razonamiento para relacionar estas ODS con la aplicación es el siguiente:

- **ODS 5: Igualdad de género:** la funcionalidad de filtrado de perfiles no ofrece la opción de filtrar perfiles por género, ya que se considera que no es un filtro relevante a la hora de buscar un empleado para una vacante. De esta manera, se promueve la igualdad de género en las oportunidades de encontrar trabajo.

- **ODS 8: Trabajo decente y crecimiento económico:** con la aplicación los usuarios podrán generar el currículum para ellos mismos, con los que luego poder optar a trabajar.

  Además, al incluir una herramienta para que los reclutadores puedan buscar empleados para sus vacantes, damos la opción de que aquellas personas con LinkedIn puedan ser contactadas para ocupar una vacante de una empresa.

- **ODS 9: Industria, innovación e infraestructura:** la aplicación utiliza la tecnología para resolver problemas en la búsqueda de empleados con unas características concretas, mejorando la eficiencia y accesibilidad de los servicios laborales.

- **ODS 10: Reducción de las desigualdades:** la aplicación pretende contribuir a este problema mediante la opción de generación de currículums sin foto, lo que hace que cualquier persona, independientemente de cómo sea, pueda solicitar trabajo y que se le valore por su formación y experiencia y no por su apariencia, género, etc.