

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE



**PLAN DE ASEGURAMIENTO PARA PLATAFORMA DE
EMPLEOS PARA ESTUDIANTES DE EDUCACIÓN
SUPERIOR, STUTRABAJO**

FRANCISCO ANTONIO SÁNCHEZ VERGARA

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA**

Profesor Guía: Cristian Lara Valenzuela

Diciembre - 2024

DEDICATORIA

Dedicado a mi familia y familiares, ya que me han apoyado desde el comienzo para cumplir todos mis objetivos propuestos y ayudado en mis momentos más difíciles.

AGRADECIMIENTOS

El autor de este informe agradece a las siguientes personas por su apoyo en el desarrollo de este proyecto.

- A los estudiantes que participaron en las encuestas y entrevistas.
- Al equipo de desarrollo del portal por su dedicación y trabajo duro.

RESUMEN

Este proyecto consta del desarrollo de un portal de empleos de corto periodo de tiempo para estudiantes de Educación Superior en Chile, con el objetivo de proporcionarles oportunidades laborales que les permitan generar ingresos para cubrir necesidades básicas sin interferir sus responsabilidades académicas. La plataforma será construida con herramientas de diseño y programación web y los resultados apuntan a una mayor flexibilidad y accesibilidad en la búsqueda de empleos, contribuyendo a la reducción de deserción estudiantil por motivos económicos y mejorando la calidad de vida de los estudiantes al permitirles equilibrar sus responsabilidades académicas y laborales.

ABSTRACT

This project consists of the development of a short-term job portal for higher education students in Chile, with the objective of providing job opportunities that allow them to generate income to cover basic needs without interfering with their academic responsibilities. The platform will be built with web design and programming tools and the results aim to provide greater flexibility and accessibility in the search for jobs, contributing to the reduction of student desertion for economic reasons and improving the quality of life of students by allowing them to balance their academic and work responsibilities.

INDICE DE CONTENIDOS

RESUMEN	4
ABSTRACT	4
INDICE DE FIGURAS	8
INTRODUCCIÓN	11
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	12
1.1 Organización y contexto de trabajo	12
1.2 Situación actual	12
1.3 Actores o usuarios involucrados.....	13
1.4 Justificación y motivación	14
1.5 Competencia.....	15
1.6 Ventajas	16
1.7 Objetivos de la organización.....	17
1.7.1 Objetivo general	17
1.7.2 Objetivos específicos	17
1.8 Objetivo de la memoria.....	18
1.8.1 Limites del entregable	19
CAPÍTULO 2: MARCO CONCEPTUAL	20
2.1 Conceptos Técnicos	20
2.1.1 Arquitectura de Microservicios	20
2.1.2 <i>Google Cloud Platform (GCP)</i>	20
2.2 Metodologías	22
2.2.1 Desarrollo Ágil (<i>Scrum</i>)	22
2.3 Herramientas	24
2.3.1 <i>React y Vite</i>	24
2.3.2 <i>Visual Studio Code</i>	24
2.4 Ciberseguridad en la Plataforma	25

2.4.1 Referentes y estándares de Seguridad Utilizados.....	25
2.4.2 Enfoque en el Backend	26
2.4.3 Enfoque en el Frontend.....	27
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN.....	28
3.1 Descripción General de la Solución	28
3.2 Ciberseguridad en el Proyecto StuTrabajo.....	28
3.2.1 Introducción.....	28
3.2.2 Medidas de Seguridad.....	28
3.2.3 Políticas de Seguridad	30
3.2.4 Herramientas y Tecnologías de Seguridad.....	31
3.2.5 Implementación de estandarizaciones <i>iso 27002</i>	32
3.2.6 Mecanismos de seguridad en servicio en la nube	35
3.2.7 Trabajos a futuro	36
3.3 Resumen de otras áreas	38
3.3.1 Diseño de página web (desarrollado por Matías Castillo)	38
3.3.2 Descripción del Backend (desarrollado por Sebastián Poblete)	38
3.3.3 Descripción experiencia de usuario (UX) (desarrollado por Daniel Soto Benavides)	38
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	40
4.1 Seguridad de la contraseña.....	40
4.2 Manejo de sesiones.....	54
4.3 <i>Validation, sanitization and encoding</i>	66
4.4 <i>Data protection</i>	85
4.5 <i>API and web service</i>	89
4.6 <i>Configuration</i>	92
CAPÍTULO 5: CONCLUSIONES	104
5.1 Aprendizaje.....	104
5.2 Recomendaciones	104
5.3 Valor agregado a la organización.....	104

5.4 Reflexión.....	105
5.6 Conclusión	105
Referencias bibliográficas	106

INDICE DE FIGURAS

Figura 1: Visualizacion encuesta a estudiantes. Fuente: Elaboración propia.	14
Figura 2: Visualización propuesta arquitectura.	22
Figura 3: Visualización Sprint estudiantes.	23
Figura 4: Visualización Sprints solicitantes.	23
Figura 5: Visualización diagrama de proyectos OWASP.	26
Figura 6: Prueba de extensión de contraseña. Fuente: Elaboración propia.	42
Figura 7: Prueba de ingreso de 128 caracteres en contraseña.	43
Figura 8: Prueba de ingreso de exceso de caracteres de contraseña.	44
Figura 9: Prueba de ingreso variedad de caracteres en contraseña.	45
Figura 10: Prueba de cambio de contraseña.	46
Figura 11: Resultado de cambio de contraseña.	47
Figura 12: Prueba de cambio de contraseña antigua.	48
Figura 13: Resultado de cambio de contraseña antigua. Fuente: Elaboración propia.	49
Figura 14: Prueba de cambio de contraseña incorrecta.	49
Figura 15: Prueba de contraseña insegura.	51
Figura 16: Visualizacion de medor de contraseña. Fuente: Elaboración propia.	52
Figura 17: Visualizacion de contraseña enmascarada.	54
Figura 18: Prueba de generación de <i>token</i> Fuente: Elaboración propia.	55
Figura 19: Resultado de generación de <i>token</i> Fuente: Elaboración propia.	56
Figura 20: Visualización de bits de entropía de <i>token</i>	57
Figura 21: Prueba de invalidacion de <i>token</i> . Fuente: Elaboración propia.	59
Figura 22: Visualización inicio de sesión exitoso.	60
Figura 23: Visualización de sesión pasado 20 segundos.	61
Figura 24: Prueba de validación de codigo. Fuente: Elaboración propia.	63
Figura 25: Visualización de envío de codigo. Fuente: Elaboración propia.	63
Figura 26: Prueba de obtención de usuarios personales. Fuente: Elaboración propia.	65

Figura 27: Prueba de obtención de datos ajenos. Fuente: Elaboración propia.....	65
Figura 28: Prueba de contaminación de parametros. Fuente: Elaboración propia.	67
Figura 29: Prueba de error de campos no permitidos. Fuente: Elaboración propia.	68
Figura 30: Visualizacion lista de permitidos.	69
Figura 31: Prueba de validación de campos. Fuente: Elaboración propia.	70
Figura 32: Visualización de validación de campos. Fuente: Elaboración propia.	71
Figura 33: Prueba de obtención de archivos. Fuente: Elaboración propia.	74
Figura 34: Prueba de ingreso de archivos no permitidos. Fuente: Elaboración propia.	76
Figura 35: Obtención de error de archivos no permitidos. Fuente: Elaboración propia.	76
Figura 36: Prueba de ingreso de archivo no permitido.. Fuente: Elaboración propia.....	78
Figura 37: Visualización de error de archivo no permitido. Fuente: Elaboración propia.	78
Figura 38: Prueba de ingreso de codigo mediante formulario. Fuente: Elaboración propia.....	79
Figura 39: Prueba de inyección SQL. Fuente: Elaboración propia.	82
Figura 40: Visualizacion de inyección SQL. Fuente: Elaboración propia.	83
Figura 41: Prueba de inyección de comandos <i>PowerShell</i> . Fuente: Elaboración propia.	84
Figura 42: Prueba de eliminación de usuario. Fuente: Elaboración propia.	87
Figura 43: Visualización de términos y condiciones. Fuente: Elaboración propia.	88
Figura 44: Prueba de modificación de usuario ajeno. Fuente: Elaboración propia.	90
Figura 45: Visualización de error de modificación de usuario ajeno. Fuente: Elaboración propia. ..	90
Figura 46: Visualización de <i>tokens</i> adicionales. Fuente: Elaboración propia.	92
Figura 47: Visualización de encabezados no comprometedores. Fuente: Elaboración propia.	93
Figura 48: Visualizacion content-type. Fuente: Elaboración propia.	95
Figura 49: Visualizacion de cabecera CSP.	97
Figura 50: Visualización de cabecera nosniff.....	98
Figura 51: Visualización de cabecera <i>referer</i>	99
Figura 52: Visualización error <i>iframe</i>	101
Figura 53: Prueba metodos no permitidos.	102
Figura 54: Visualización encabezado <i>Origin</i>	103

INTRODUCCIÓN

La deserción estudiantil en las Instituciones de Educación Superior en Chile es un problema poco discutido que afecta tanto a los estudiantes como a las instituciones. Diversos estudios realizados por el Servicio de Información de Educación Superior (SIES), el Ministerio de Educación (MINEDUC)¹ y Psicomatrix² indican que, para el año 2023, hay 1.341.439 estudiantes matriculados en programas de Pregrado, Posgrado y Postítulo. De estos, el 21% abandona sus estudios por motivos económicos, vinculados a la falta de recursos para cubrir necesidades básicas como alojamiento, alimentación, transporte, matrícula, arancel, etc. Este problema surge de la incompatibilidad entre las responsabilidades académicas y laborales, lo que obliga a muchos estudiantes a abandonar sus estudios para trabajar. Además, aquellos que intentan equilibrar estudio y trabajo a menudo ven afectado negativamente su desarrollo académico y futuro profesional.

El presente proyecto aborda esta problemática mediante el desarrollo de un portal de empleos de corto periodo de tiempo exclusivamente para estudiantes de Educación Superior, que facilita la búsqueda de trabajos temporales adecuados a sus necesidades y disponibilidad. Esta solución no solo busca proporcionar una fuente de ingresos para los estudiantes, sino también apoyar su éxito académico y profesional a largo plazo.

Nos basaremos en un enfoque centrado en los usuarios del portal, comenzando con un análisis exhaustivo del problema para identificar los principales desafíos y necesidades de los estudiantes. Esto incluirá un estudio de las competencias existentes y la definición clara de objetivos y alcances. A continuación, abordaremos la parte conceptual, describiendo conceptos técnicos, metodologías de trabajo y herramientas que se utilizarán en la construcción de la propuesta. Posteriormente, presentaremos la propuesta de solución, detallando los elementos clave que conformarán la plataforma. Incluiremos también un plan de validación para demostrar la efectividad y pertinencia de la solución para nuestros usuarios. Finalmente, concluiremos con una reflexión sobre los aprendizajes obtenidos a lo largo del trabajo realizado y las implicaciones futuras para la continuidad y mejora de la plataforma.

¹ <https://www.mineduc.cl/>

² <https://psicomatrix.cl/abandono-de-los-estudios-datos-sobre-la-desercion-universitaria-en-chile/>

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 Organización y contexto de trabajo

El proyecto se desarrollará en el contexto de la plataforma **StuTrabajo**, una iniciativa destinada a brindar apoyo financiero a estudiantes de Educación Superior en Chile. La organización detrás de este proyecto está comprometida con la misión de reducir la deserción estudiantil por motivos económicos, facilitando a los estudiantes la posibilidad de generar ingresos que les permitan cubrir necesidades básicas como alojamiento, alimentos, transporte, matrícula, arancel, etc.

El proyecto se enmarca en varias líneas de la informática, destacándose el desarrollo de software, ciberseguridad, inteligencia artificial, aprendizaje automático, bases de datos, sistemas de información, experiencia de usuario, entre otras.

1.2 Situación actual

Actualmente, los estudiantes suelen recurrir a diferentes métodos para encontrar empleo, entre ellos:

- Portales de empleo genéricos: Utilizan plataformas que no están específicamente diseñadas para estudiantes y no siempre se ofrecen empleos con la flexibilidad necesaria.
- Redes sociales y preferencias: Se apoyan en redes sociales y recomendaciones de amigos o familiares para encontrar trabajos temporales, aunque esta estrategia no siempre resulta efectiva ni fiable.
- Ferias de empleo: Participación en ferias de empleo organizadas por las mismas instituciones o entidades externas, aunque las oportunidades son demasiado limitadas y no siempre se ajustan a sus necesidades de tiempo.
- Bolsas de trabajo universitarias: Algunas instituciones ofrecen bolsas de trabajo, pero estas suelen ser limitadas de alcance ya que existen pocos cupos para poder postular y no hay mucha diversidad de ofertas o por la cercanía a la ubicación del trabajo.

El proceso que los estudiantes siguen para encontrar empleos típicamente incluye:

- Búsqueda de empleo: buscan activamente empleos en múltiples plataformas, lo que puede ser un proceso largo y agotador.
- Aplicación y envío de CV: Una vez que encuentran la oportunidad, hacen envío de su currículum y esperar una respuesta, lo cual puede tardar días o incluso semanas.

- Entrevistas y selección: Si son seleccionados, pasan por entrevistas que pueden ser presenciales o virtuales, requiriendo una inversión adicional de tiempo.
- Negociación de horarios: Muchos estudiantes deben negociar los horarios con los solicitantes para asegurarse de que no interfieran con sus clases, lo cual no siempre es posible y provoca que el estudiante pierda la oportunidad de trabajar.

1.3 Actores o usuarios involucrados

Los principales actores involucrados en esta problemática son los estudiantes de Instituciones de Educación Superior y las personas interesadas en contratar sus servicios, ya sean solicitantes formales o individuos que requieran servicios específicos de forma temporal.

Los estudiantes buscan empleos temporales y flexibles que les permitan generar ingresos sin interferir con sus estudios. Actualmente, enfrentan diversas dificultades, como la falta de flexibilidad en los empleos disponibles, la escasa disponibilidad de plataformas especializadas en empleos temporales, y el largo y agotador proceso de búsqueda de empleo. Además, la inseguridad sobre la confiabilidad de los solicitantes y la calidad de los empleos es una preocupación constante. Según datos del Ministerio de Educación de Chile, hay aproximadamente 1.3 millones de estudiantes de Educación Superior en el país³.

Por otra parte, tenemos a los "solicitantes", que incluyen empresas, organizaciones y particulares que ofrecen empleos temporales y flexibles. Estos solicitantes enfrentan dificultades para encontrar estudiantes que se ajusten a las necesidades específicas del empleo, y el proceso de reclutamiento y selección puede ser largo y consume mucho tiempo. Además, la falta de información y confianza en la calidad de los candidatos es una barrera significativa.

³ <https://educacionsuperior.mineduc.cl/2023/07/11/informe-de-matricula-2023-la-matricula-en-educacion-superior-crecio-3-en-2023-y-suma-casi-140-mil-estudiantes-de-pueblos-originarios/>

1.4 Justificación y motivación

Si en el corto o mediano plazo no se solucionan las dificultades que enfrentan los estudiantes de Educación Superior en Chile para encontrar empleos temporales y flexibles, las consecuencias podrían ser significativas tanto a nivel individual como social.

A nivel individual, uno de los impactos más críticos es la posible deserción. Los estudiantes que no pueden equilibrar sus necesidades económicas con sus responsabilidades académicas pueden verse obligados a abandonar sus estudios. Según datos del Ministerio de Educación de Chile, la deserción universitaria alcanza un 23,1%, mientras que en institutos profesionales y centros de formación técnica las tasas son significativamente más altas, llegando al 32,4% y 34,3% respectivamente⁴. Además, la continua preocupación por la falta de ingresos puede generar altos niveles de estrés y ansiedad entre los estudiantes. Este estrés no solo afecta su bienestar emocional, sino también su rendimiento académico, creando un ciclo negativo que es difícil de romper.

12. ¿Cuáles fueron los motivos que te llevaron a pensar en abandonar tus estudios?



Figura 1: Visualización encuesta a estudiantes.

Fuente: Elaboración propia.

A nivel social, la deserción tiene un impacto directo en las instituciones de Educación Superior. Cuando los estudiantes abandonan sus estudios debido a dificultades económicas, estas instituciones sufren una reducción en sus tasas de retención y graduación, lo que afecta negativamente su reputación y acreditación. Las universidades, centros de formación técnica, institutos profesionales, dependen de las tasas de retención para mantener su financiación y apoyo gubernamental, y la deserción estudiantil puede resultar en menos recursos y oportunidades para todos los estudiantes.

⁴ https://bibliotecadigital.mineduc.cl/bitstream/handle/20.500.12365/4599/Desercionreingreso_2015_2019.pdf

Además, la falta de acceso a empleos temporales y flexibles también afecta a las instituciones de Educación Superior al limitar su capacidad para apoyar a sus estudiantes. Los estudiantes de bajos recursos, que son los más afectados por la necesidad de trabajar mientras estudian, pueden quedar en desventaja en comparación con sus pares que no enfrentan las mismas presiones económicas. Esto genera una brecha en el rendimiento académico y en las oportunidades de éxito profesional, lo que a su vez impacta negativamente en la misión educativa de estas instituciones.

Cada estudiante que no puede continuar su educación y no puede desarrollar su potencial completo representa una pérdida significativa para las instituciones de Educación Superior en términos de innovación y contribuciones al desarrollo académico. La fuga de talento también puede reducir la capacidad de estas instituciones para atraer financiamiento para investigación y desarrollo, así como para establecer colaboraciones con la industria y otros sectores. Por lo tanto, abordar estas dificultades es esencial no solo para el bienestar de los estudiantes, sino también para la sostenibilidad y crecimiento de las instituciones educativas en el país.

La motivación para llevar a cabo este proyecto se basa principalmente en contribuir a la reducción de deserción estudiantil causada por problemas económicos, equilibrar las responsabilidades académicas con lo laboral, reduciendo el estrés y mejorando el bienestar de los estudiantes y permitirles adquirir experiencia laboral valiosa, que mejorará sus perspectivas de empleo a largo plazo y su desarrollo profesional.

1.5 Competencia

En el contexto actual, existen varias soluciones y plataformas que intentan abordar el problema de la búsqueda de empleo para estudiantes de Educación Superior en Chile. Sin embargo, estas plataformas presentan limitaciones que impiden una solución efectiva y adaptada a las necesidades específicas de los estudiantes. A continuación, se presentan algunas de las principales competencias y se analiza por qué la solución propuesta por StuTrabajo sería mejor.

- Aunque los portales de empleo actuales ofrecen una gran cantidad de ofertas laborales, no están específicamente diseñados para estudiantes y, por lo tanto, no siempre proporcionan la flexibilidad necesaria. Los empleos disponibles en estos sitios suelen requerir un compromiso de tiempo completo o fijo, lo que no se ajusta a los horarios variables y exigencias académicas de los estudiantes. Además, estos portales no suelen ofrecer mecanismos para asegurar que las ofertas sean temporales o de corto plazo, lo cual es una necesidad clave para los estudiantes.

- Los estudiantes a menudo recurren a redes sociales y referencias personales para encontrar trabajos temporales. Aunque esta estrategia puede ser útil, es inherentemente informal y no garantiza la seguridad ni la calidad de los empleos.
- En otros países, se han desarrollado plataformas especializadas que abordan problemas similares. Por ejemplo, en Estados Unidos y Europa, existen plataformas como *Handshake*⁵ y *StudentJob*⁶ que se centran en conectar a estudiantes o graduados con empleos flexibles y ayudándoles a construir su carrera. Sin embargo, estas soluciones están adaptadas a los contextos y necesidades de sus respectivos países, y pueden no ser directamente aplicables al contexto chileno sin adaptaciones significativas.

1.6 Ventajas

StuTrabajo ofrece características diseñadas específicamente para abordar las necesidades únicas de los estudiantes en Chile:

- **Especialización en Estudiantes:** StuTrabajo está específicamente diseñada para atender las necesidades de los estudiantes de Educación Superior en Chile. Esto significa que todas las funcionalidades y características de la plataforma están adaptadas para proporcionar la flexibilidad y temporalidad necesarias.
- **Match Interactivo:** Utiliza algoritmos de emparejamiento que conectan automáticamente a los estudiantes con ofertas laborales que se ajustan a sus perfiles y disponibilidad. Esto con la intención de reducir el tiempo y esfuerzo necesarios para encontrar un empleo adecuado.
- **Perfiles Dinámicos y Valoraciones:** Los estudiantes pueden crear perfiles detallados y recibir valoraciones de solicitantes, lo que mejora la transparencia y confianza en el proceso de contratación. Los solicitantes también se benefician al poder evaluar la calidad y adecuación de los candidatos antes de contratarlos.
- **Alertas:** La plataforma envía notificaciones inmediatas por correo electrónico, previa autorización del estudiante, para informar sobre nuevas oportunidades laborales relevantes. Esto garantiza que los estudiantes puedan aprovechar cada oferta sin preocuparse por perder opciones importantes. Además, los solicitantes

⁵ <https://joinhandshake.com/about/>

⁶ <https://www.studentjob.co.uk/about-studentjob>

también reciben alertas cuando sus ofertas generan postulaciones, facilitando una comunicación ágil y efectiva entre ambas partes.

- **Seguridad y Confiabilidad:** La plataforma garantiza un entorno seguro y confiable mediante la implementación de medidas de seguridad robustas en base a referentes como OWASP⁷ y estándares como ISO 27002⁸.

StuTrabajo se presenta como una solución diseñada específicamente para atender las necesidades únicas de los estudiantes de Educación Superior en Chile. Con un enfoque integral, funcional y seguro, la plataforma facilita la conexión entre estudiantes y solicitantes, promoviendo oportunidades laborales adaptadas a las exigencias académicas y profesionales de este grupo.

1.7 Objetivos de la organización

1.7.1 Objetivo general

Desarrollar una plataforma tecnológica integral que facilite a los estudiantes de Educación Superior en Chile encontrar empleos temporales y flexibles, permitiéndoles generar ingresos sin comprometer sus responsabilidades académicas, y contribuyendo así a reducir la deserción por motivos económicos.

1.7.2 Objetivos específicos

- Desarrollar una plataforma web y móvil que permita a los estudiantes registrarse, crear perfiles personalizados, buscar empleos temporales y flexibles, y postular a ofertas laborales. Asimismo, ofrecer a los solicitantes la posibilidad de publicar ofertas de trabajo adaptadas a horarios reducidos o jornadas no extensivas, respondiendo a las necesidades específicas de ambos grupos.
- Desarrollar algoritmos de match interactivo que conecten automáticamente a los estudiantes con ofertas laborales que se ajusten a sus perfiles y disponibilidad.
- Implementar un sistema de valoraciones donde los solicitantes puedan calificar a los estudiantes y viceversa, mejorando la transparencia y confianza en la plataforma.
- Trabajar en conjunto con universidades para promover la plataforma entre los estudiantes.

⁷ <https://OWASP.org/about/>

⁸ <https://www.iso.org/standard/75652.HTML>

- Diseñar e implementar una estrategia de marketing para atraer usuarios a la plataforma, destacando sus beneficios y funcionalidades únicas.
- Utilizar redes sociales, eventos y alianzas estratégicas para aumentar la visibilidad y adopción de la plataforma.
- Recopilar *feedback* de los usuarios y realizar mejoras continuas en las funcionalidades y la experiencia de usuario.

1.8 Objetivo de la memoria

El principal objetivo de esta memoria es proporcionar un conocimiento claro y estructurado sobre los estándares de seguridad aplicables al desarrollo de plataformas en las que interactúan múltiples usuarios, como el portal de empleo desarrollado en este proyecto. Para lograrlo, la memoria se centra en la implementación de estándares específicos que garanticen la protección de la información sensible, asegurando una base sólida para la interacción entre estudiantes y solicitantes.

¿Qué se hará?

Se analizarán, seleccionarán e implementarán estándares de seguridad relevantes para garantizar la protección de los datos sensibles y minimizar los riesgos asociados al uso de la plataforma. También se documentará el proceso de toma de decisiones y las prácticas de seguridad adoptadas.

¿Cómo se hará?

La memoria utilizará un enfoque basado en el análisis de estándares internacionales reconocidos, como *OWASP ASVS*, la norma *ISO/IEC 27002* y el referente *OWASP CheatSheet*. Se evaluará su aplicabilidad y se implementarán aquellos que se alineen con las necesidades específicas del portal de empleo.

¿Para qué se hará?

El propósito es garantizar que la plataforma sea confiable y segura para sus usuarios, protegiendo sus datos sensibles y fomentando una interacción segura entre las partes involucradas. Además, la memoria busca servir como un recurso de referencia para la organización.

1.8.1 Limites del entregable

Aunque esta memoria aborda la implementación de estándares de seguridad, no incluye la aplicación exhaustiva de toda la serie de estándares disponibles. Esto se debe a que no todos los estándares se ajustan a los requerimientos y el enfoque particular de la plataforma desarrollada. Los estándares seleccionados fueron cuidadosamente elegidos en función de su relevancia para las necesidades del proyecto, priorizando aquellos que abordan aspectos esenciales como la protección de datos sensibles, la interacción segura entre usuarios y entre el usuario con la plataforma. Este enfoque limitado garantiza una solución práctica y adecuada al alcance del proyecto, sin perder de vista los principios fundamentales de ciberseguridad.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1 Conceptos Técnicos

2.1.1 Arquitectura de Microservicios

La arquitectura de microservicios es un estilo de diseño software para la web de nuestro proyecto, donde los componentes de una aplicación se distribuyen en servicios pequeños, independientes y desplegados de forma autónoma. Cada microservicio tiene sus funciones específicas y que se comunican con los otros servicios a través de las *APIs*.

Ventajas:

- **Escalabilidad:** Permite la escalabilidad de los componentes según la demanda.
- **Mantenibilidad:** Facilita el mantenimiento y actualización de cada servicio sin afectar el resto de los componentes del sistema.
- **Despliegue Independiente:** Los microservicios pueden ser desplegado de forma independiente

Este capítulo se centra en los fundamentos conceptuales y las ventajas generales de la arquitectura de microservicios. Los detalles específicos sobre su implementación y configuración en el proyecto, incluyendo la lógica del Backend y la integración con las bases de datos, se desarrollan en la sección **3.3.2 Descripción del Backend**, a cargo de Sebastián Poblete. Allí se detalla cómo esta arquitectura se traduce en una solución escalable y eficiente para las necesidades del sistema.

2.1.2 Google Cloud Platform (GCP)

GCP es una suite de servicios en la nube que ofrece *Google*, proporcionando infraestructura, almacenamiento y servicios de *machine learning*. Es especialmente útil para proyectos que requieren escalabilidad y fiabilidad.

Componentes Clave:

- **Google Kubernetes Engine (GKE):** Para la orquestación de contenedores.
- **Google Cloud SQL:** Bases de datos relacionales gestionadas.

Propuesta de Arquitectura del Proyecto

Para este proyecto, se propone una arquitectura basada en microservicios que aprovecha las capacidades de *GKE* para implementar, gestionar y escalar las aplicaciones. La solución

se construye sobre un clúster de *Kubernetes* alojado en *GCP*, utilizando un enfoque modular para garantizar flexibilidad, escalabilidad y mantenibilidad.

Elementos clave de la arquitectura:

1. Frontend:

- Accesible desde navegadores web y dispositivos móviles a través de aplicaciones *React Native*.
- La comunicación con los servicios Backend se realiza mediante protocolos *HTTPS* para garantizar seguridad.

2. API Gateway:

- Actúa como un punto de entrada único para todas las solicitudes hacia los microservicios.
- Maneja la autenticación, el enrutamiento de solicitudes y la integración entre los servicios Frontend y Backend.

3. Backend:

- Basado en microservicios desarrollados en *Django*, cada uno dedicado a una funcionalidad específica:
 - **Jobs Posting Service:** Maneja la publicación de empleos.
 - **Jobs Feed Service:** Gestiona la visualización de ofertas para los usuarios.
 - **Notifications Service:** Coordina el envío de notificaciones a los usuarios.
 - **Chat Service:** Facilita la comunicación en tiempo real entre solicitantes y estudiantes.
 - **Users Service:** Administra la autenticación, los perfiles de usuario y la recuperación de contraseñas.

4. Bases de Datos:

- Se utilizan bases de datos relacionales *MySQL* para almacenar de manera eficiente la información asociada a cada servicio. Por ejemplo, ofertas de trabajo, mensajes de chat y perfiles de usuario.

5. Escalabilidad y Gestión:

- Los contenedores de los microservicios se implementan y orquestan en el clúster de *GKE*, que ajusta automáticamente los recursos según las necesidades del sistema.

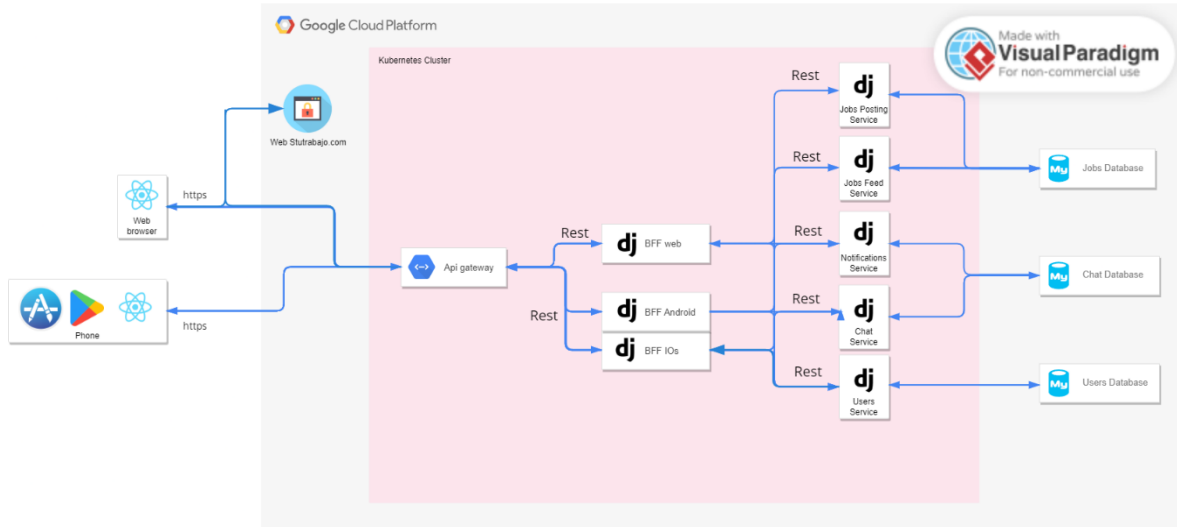


Figura 2: Visualización propuesta arquitectura.

Fuente: Elaboración propia.

2.2 Metodologías

2.2.1 Desarrollo Ágil (*Scrum*)

Scrum es una metodología ágil para la gestión y desarrollo de proyectos de software. Se basa en la formación de equipos autoorganizados y multifuncionales, trabajando en *sprints* cortos para entregar incrementos funcionales del producto.

Principios

- **Iterativo e Incremental:** Desarrollo en ciclos cortos (*sprints*) con entregas parciales pero funcionales.
- **Colaboración:** Comunicación constante entre los miembros del equipo y las partes interesadas.
- **Adaptabilidad:** Capacidad para responder a cambios y ajustar el plan de trabajo según las necesidades emergentes.

Implementación de *Scrum* en el proyecto

En el desarrollo de la plataforma, *Scrum* ha sido fundamental para gestionar las diversas funcionalidades del proyecto, como la postulación a empleos, el emparejamiento (match)

entre estudiantes y empleadores, el sistema de chat, y la recuperación de contraseñas. Cada una de estas características se traduce en historias de usuario claramente definidas y priorizadas en el *Product Backlog*, como se observa en la tabla presentada.

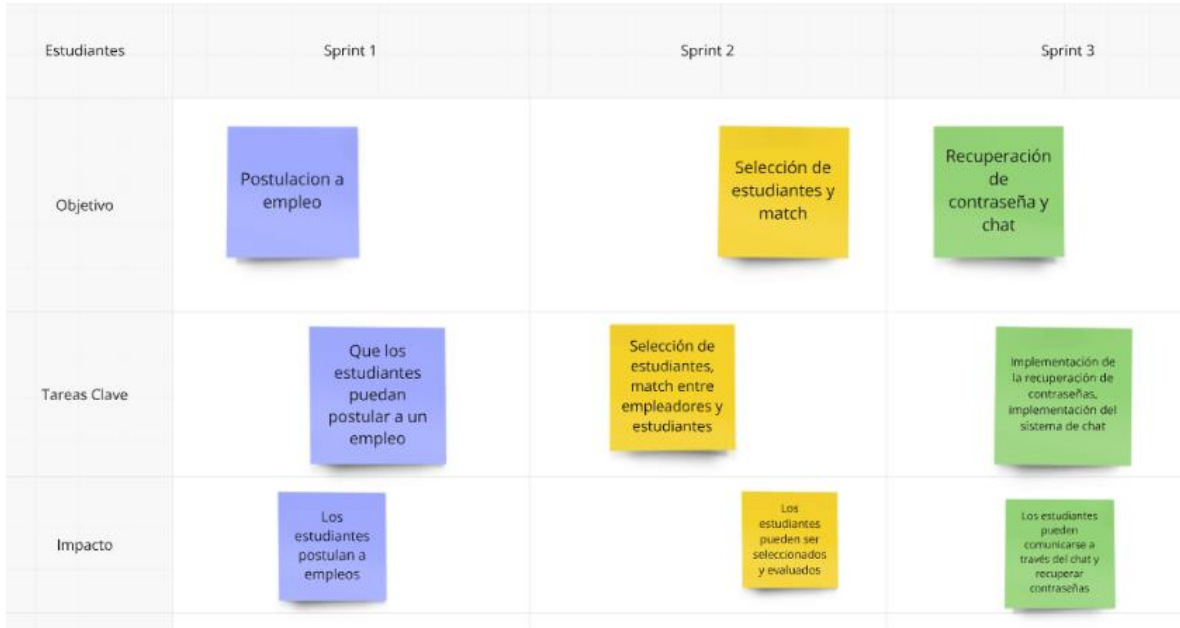


Figura 3: Visualización Sprint estudiantes.

Fuente: Elaboración propia.

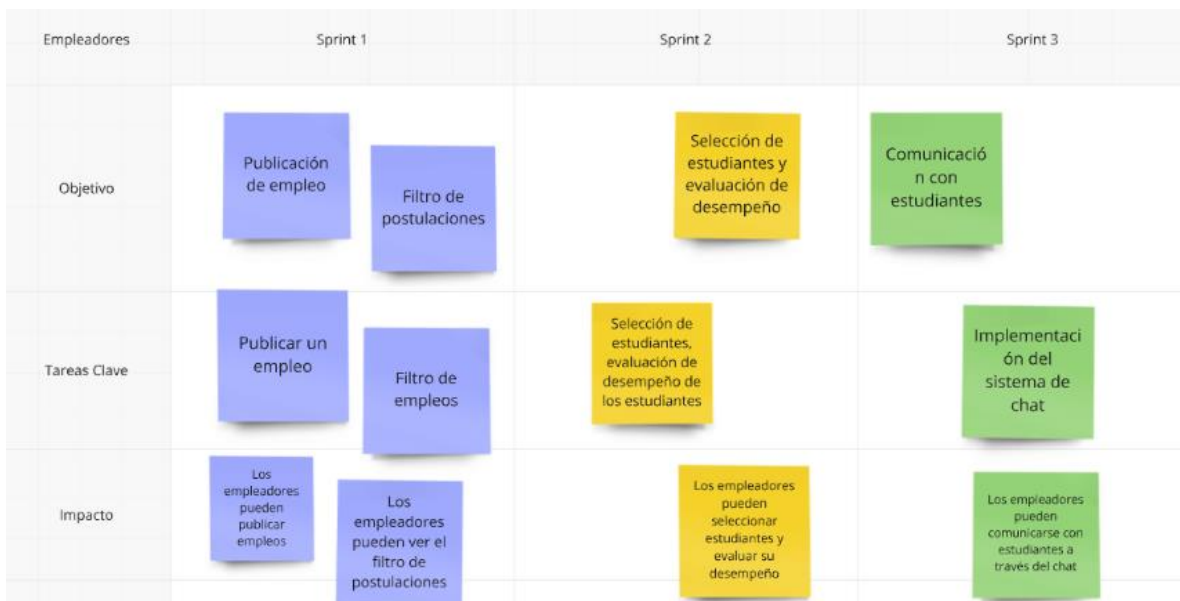


Figura 4: Visualización Sprints solicitantes.

Fuente: Elaboración propia.

Nota aclaratoria: En la figura, se utiliza el término "Empleadores" para referirse a una de las partes involucradas en la planificación. Sin embargo, el término correcto es "Solicitantes", ya que refleja de manera

más adecuada el rol de quienes publican y gestionan las ofertas de empleo en el contexto del proyecto. Esta aclaración se hace con el propósito de mantener consistencia en la terminología empleada a lo largo del informe.

2.3 Herramientas

2.3.1 *React* y *Vite*

React es una biblioteca de *JavaScript* para construir interfaces de usuario, desarrollada por Facebook. *Vite* es un entorno de desarrollo Frontend que permite una configuración mínima y una compilación rápida, ideal para proyectos que utilizan *React*.

Ventajas:

- ***React***: Componentes reutilizables, *virtual DOM* para rendimiento mejorado.
- ***Vite***: Desarrollo rápido y eficiente, recarga en caliente.

Este apartado aborda los fundamentos y beneficios de *React* y *Vite* en el desarrollo de interfaces de usuario. Los detalles específicos de su implementación, incluyendo el diseño visual y los flujos de navegación intuitivos de la página web, se desarrollan en la sección **3.3.1 Diseño de página web**, a cargo de Matías Castillo. Allí se profundiza en cómo estas herramientas se integraron para lograr un *Frontend* atractivo, responsivo y accesible.

2.3.2 *Visual Studio Code*

Visual Studio Code (*VS Code*) es un editor de código fuente desarrollado por *Microsoft*. Lo usan desarrolladores por su rendimiento, extensibilidad y soporte para múltiples lenguajes de programación. *VS Code* proporciona un entorno de desarrollo versátil que mejora la productividad y la eficiencia en la codificación.

Características Principales:

- **Extensibilidad**: *VS Code* tiene una vasta colección de extensiones que permiten a los desarrolladores agregar soporte para nuevos lenguajes, depuradores, integraciones con herramientas de desarrollo y servicios en la nube, entre otros. Esto permite personalizar el entorno de trabajo según las necesidades específicas del proyecto.
- ***IntelliSense***: Proporciona autocompletado de código, resaltado de sintaxis y sugerencias de código basadas en el contexto, lo cual mejora la velocidad y precisión de la codificación.
- **Depuración Integrada**: Incluye herramientas de depuración que permiten establecer puntos de interrupción, inspeccionar variables, y realizar seguimiento del flujo de ejecución del código, todo dentro del editor.

- **Terminal Integrado:** *VS Code* incluye un terminal integrado, permitiendo a los desarrolladores ejecutar comandos de *shell* sin salir del entorno de desarrollo.
- **Editor Multilenguaje:** Soporte para una amplia variedad de lenguajes de programación y scripts, incluyendo, pero no limitado a *JavaScript*, *TypeScript*, *Python*, *C++*, *Go* y *Rust*.

Ventajas de *VS Code*:

- **Ligero y Rápido:** *VS Code* es conocido por ser un editor ligero que se inicia rápidamente y consume menos recursos del sistema en comparación con otros entornos de desarrollo integrados (*IDEs*).
- **Multiplataforma:** Está disponible para *Windows*, *macOS* y *Linux*, proporcionando un entorno de desarrollo consistente en todas las plataformas.
- **Personalización:** Los desarrolladores pueden personalizar su entorno de trabajo utilizando temas, configuraciones personalizadas y atajos de teclado.
- **Comunidad Activa:** Una comunidad activa de desarrolladores contribuye regularmente con extensiones, temas y soluciones a problemas comunes, enriqueciendo el ecosistema de *VS Code*.

2.4 Ciberseguridad en la Plataforma

La ciberseguridad es un pilar fundamental en el diseño de esta plataforma, dado que maneja información sensible, como perfiles de usuario, comunicaciones y datos relacionados con ofertas laborales. En este capítulo, se presentan los referentes y el estándar utilizados para establecer las bases de seguridad del sistema, asegurando un enfoque estructurado para la protección de datos y la seguridad del usuario.

Para garantizar la seguridad de la plataforma, se han adoptado estándares reconocidos como *OWASP ASVS*, *CheatSheet Series* e *ISO/IEC 27002*, que proporcionan lineamientos específicos para mitigar riesgos y establecer controles efectivos. Estas herramientas sirvieron como guía conceptual para definir las medidas necesarias que se describen de manera detallada en el capítulo siguiente.

2.4.1 Referentes y estándares de Seguridad Utilizados

- **OWASP ASVS (*Application Security Verification Standard*)⁹:** Este referente estructurado fue elegido para guiar la implementación y verificación de controles

⁹ <https://owasp.org/www-project-application-security-verification-standard/>

- **Autenticación y autorización:** Se implementaron protocolos seguros para garantizar que solo usuarios autenticados y con permisos específicos puedan acceder a ciertas funcionalidades.
- **Protección contra inyecciones:** Se utilizaron consultas parametrizadas y validaciones en las entradas de usuario para evitar ataques de inyección *SQL* y otras amenazas relacionadas.

2.4.3 Enfoque en el Frontend

Aunque el Frontend es más susceptible a ser manipulado por los usuarios, se implementaron medidas como:

- Validación básica de entradas para mejorar la experiencia del usuario.
- Delegación de la seguridad crítica al *Backend*, minimizando riesgos al no manejar datos sensibles directamente en el *Frontend*.

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

3.1 Descripción General de la Solución

El proyecto se centra en abordar la deserción estudiantil por razones económicas creando un portal de empleo temporal orientado específicamente a estudiantes de educación superior. Este portal pretende dar oportunidades de trabajo flexibles adaptadas a los horarios académicos de los estudiantes, permitiéndoles generar ingresos y continuar sus estudios.

Por otro lado, la presente memoria se enfoca en implementar una solución que no solo cumpla con los objetivos funcionales del portal, sino que también asegure la protección de la información de los usuarios. Para ello, se adoptan estándares y prácticas de seguridad reconocidas, diseñadas para prevenir vulnerabilidades y garantizar la confidencialidad, integridad y disponibilidad de los datos. Este enfoque busca brindar una plataforma confiable, en la que tanto estudiantes como organizaciones puedan interactuar de manera segura, fortaleciendo así la confianza en el sistema.

3.2 Ciberseguridad en el Proyecto StuTrabajo

3.2.1 Introducción

En este capítulo, se describen de manera técnica las medidas y prácticas implementadas para proteger la plataforma, asegurando la confidencialidad, integridad y disponibilidad de los datos. Estas medidas se basan en el referente y estándar mencionados previamente, como **OWASP CheatSheet Series** e **ISO/IEC 27002**, en el caso de la norma iso se tomó en cuenta una menor cantidad de recomendaciones en comparación al referente de **OWASP** y se tomaron en cuenta las más relevantes para este contexto. Con estas medidas se enfoca en garantizar la seguridad tanto del Backend como del flujo de datos entre servicios Frontend y Backend.

3.2.2 Medidas de Seguridad

Para garantizar la seguridad de la plataforma StuTrabajo, se han implementado diversas medidas de seguridad tanto a nivel de software como de infraestructura. Entre ellas se incluyen:

Autenticación y Autorización

Utilización de **OAuth** para gestionar la autenticación y autorización de los usuarios, asegurando que solo individuos autorizados puedan acceder a los diferentes recursos y funcionalidades de la plataforma.

Características principales

- **Proveedores de identidad:** Se permite a los usuarios iniciar sesión utilizando servicios de terceros confiables, como Google, eliminando la necesidad de crear

credenciales sensibles del usuario y dedicando los recursos de la base de datos en la personalización del usuario.

- **Tokens de acceso:** Una vez autenticado el usuario se genera un *token* de acceso temporal que permite la interacción segura con los recursos autorizados. Este *token* incluye información sobre el alcance y tiempo de validez.
- **Authorization code Flow:** La autenticación sigue el flujo estándar recomendado por *OAuth*, donde las credenciales del usuario nunca son compartidas directamente con la plataforma, minimizando riesgos de seguridad.
- **Renovación de tokens:** Los *tokens* de acceso tienen una duración limitada y pueden ser renovados mediante un *token* de actualización para mantener la seguridad.

Requisitos del uso de *OAuth*

- **Configuración de proveedores:** Se debe realizar una configuración previa de los proveedores con la plataforma de *OAuth*, definiendo los *redirect URIS*, el flujo de trabajo de la *API* de *OAuth* para realizar el procedimiento completo del registro o inicio de sesión del usuario.

Encriptación de Datos

Implementación de encriptación de datos tanto en tránsito como en reposo. Esto incluye *HTTPS* para la comunicación segura entre cliente y servidor y la encriptación de datos sensibles almacenados en las bases de datos dentro de la nube *Google Cloud*.

Características principales

- **Protocolo HTTPS:** Todas las comunicaciones entre el cliente y el servidor utilizan el protocolo *HTTPS*, garantizando una conexión segura mediante el uso de certificados *SSL/TLS* a través de la herramienta de software gratuita *Certbot*.
- **Prevención de ataques:** La encriptación en tránsito protege a la plataforma contra ataques de tipo *Man-in-the-Middle* y asegura que la información no pueda ser interceptada o manipulada.

Protección contra Ataques de Fuerza Bruta

Integración de mecanismos de detección y mitigación de ataques de fuerza bruta, tales como el uso de captchas y la limitación de intentos de inicio de sesión.

Características principales

- **Restricción de intentos:** Se establece un límite máximo de intentos fallidos consecutivos de inicio de sesión por usuario, establecido en 4 intentos.

- **Bloqueo temporal:** Una vez superado el límite de intentos, la cuenta del usuario se bloquea temporalmente por un período predefinido (15 segundos).

Requisitos de implementación

- **Configuración de límite de intentos:** Los límites y tiempos de bloqueo se configuran desde el Backend para evitar manipulaciones mal intencionadas por parte del usuario final.
- **Política de desbloqueo:** Las cuentas bloqueadas pueden ser desbloqueadas automáticamente tras el período de bloqueo.

3.2.3 Políticas de Seguridad

Además de las medidas técnicas, StuTrabajo también ha adoptado políticas de seguridad rigurosas y programas de capacitación para todos los desarrolladores y administradores de la plataforma. Estas políticas y programas incluyen:

3.2.3.1 Políticas de Contraseñas Seguras.

Requerimientos estrictos para la creación y gestión de contraseñas, promoviendo el uso de contraseñas fuertes y la implementación de autenticación de dos factores (2FA). También asegurando que el usuario cree una contraseña que no haya sido comprometida mediante el API de *Pwned Passwords*.

Requisitos de contraseñas fuertes

- **Longitud mínima:** Las contraseñas deben tener una longitud mínima de 8 caracteres según *OWASP*. Sin embargo, se recomienda aceptar contraseñas de hasta 64 caracteres para soportar frases de contraseñas más seguras
- **Composición:** Se debe permitir al usuario ingresar todo tipo de caracteres en una contraseña, para que no haya un patrón reconocible, tales como:
 - 1 letra mayúscula
 - 1 letra minúscula
 - 1 número
 - 1 carácter especial, por ejemplo, @, #, \$
- **Evitar patrones predecibles:** No se debe permitir contraseñas comunes como "123456", "password". Las contraseñas se enviarán a un API para su revisión y en caso de estar comprometida se le requerirá cambiarla.

- **Fortaleza visual:** Se debe proporcionar un indicador visual de la fortaleza de la contraseña en tiempo real durante la creación, utilizando métricas como:
 - Longitud
 - Diversidad de caracteres
 - Ausencia de patrones comunes
- **Consulta a la API:** Los primeros 5 caracteres del hash SHA-1 generado tras crear la contraseña se enviarán al *endpoint* “GET [HTTPS://API.pwnedpasswords.com/range/{5 primeros caracteres del hash}](https://api.pwnedpasswords.com/range/{5 primeros caracteres del hash})” para luego devolver una lista de hashes truncados (los restantes 35 caracteres) junto al número de veces que cada hash ha sido comprometido. Esto sirve para comparar el hash actual de la contraseña y compararlo con los resultados devueltos por la API, si se encuentra una coincidencia, se notifica al usuario que esa contraseña ha sido comprometida, por lo que debe crear una nueva.

Gestión segura de contraseñas

- **Almacenamiento seguro de contraseñas:** Las contraseñas no se almacenan en texto plano, ya que se usan algoritmos de *hashing* SHA-256 (por defecto en *Django*). Esto sucede una vez pasa la prueba del API

3.2.4 Herramientas y Tecnologías de Seguridad

El proyecto StuTrabajo utiliza una variedad de herramientas y tecnologías para fortalecer su postura de seguridad, tales como:

3.2.4.1 Firewall de Aplicaciones Web (WAF)

Protección contra ataques comunes en aplicaciones web, incluyendo inyecciones SQL, Cross-Site Scripting (XSS), y otros tipos de ataques.

Características principales

- **Inyecciones SQL:** La protección contra inyecciones SQL se implementan consultas preparadas con parámetros en lugar de concatenar directamente las entradas del usuario en las consultas SQL. Asegurando así que el usuario no pueda alterar la estructura de la consulta.
- **Cross-site Request Forgery:** Se generarán y asociarán *tokens CSRF* a las sesiones de los usuarios. Estos *tokens* se incluirán en cada formulario HTML que realice una acción en el servidor. El servidor validará este *token* antes de procesar la solicitud.
- **Cabeceras HTTP customizadas:** Se validará que las solicitudes que modifiquen datos incluyan cabeceras *X-requested-With* o *Origin* correctas. Las solicitudes que no tengan estas cabeceras se rechazarán automáticamente.

3.2.4.2 Indicador visual de seguridad

El proceso de validación de entradas por parte del usuario es una parte crítica para evitar errores de seguridad, como la inyección de datos maliciosos o la introducción de información no válida.

- **Validación en tiempo real de entradas de usuario:** Utilizando eventos de *JavaScript* como *onchange*, se validará el formato y contenido de los datos conforme el usuario los ingresa. Cualquier error o dato no permitido generará un mensaje de alerta junto al campo correspondiente, para evitar ingresos de código malicioso dentro del formulario.
 - **Proporcionar retroalimentación positiva al usuario**
Cuando los campos se validen correctamente, se mostrará un mensaje de éxito en verde para indicar que los datos ingresados son válidos.
 - **Desactivar el envío del formulario hasta que no existan errores**
Se desactivará el botón de envío del formulario mientras haya campos con errores. El botón solo se habilitará cuando todos los campos hayan sido validados correctamente.

3.2.5 Implementación de estandarizaciones *iso 27002*

3.2.5.1 Enmascaramiento de datos

El enmascaramiento de datos es una técnica utilizada para proteger la privacidad al ocultar o modificar datos personales sensibles en documentos y bases de datos.

Técnicas utilizadas:

- **Hashing:** *Django* utiliza por defecto el algoritmo *PBKDF2* con un *hash SHA-256*. Este enfoque incluye una sal única para cada contraseña y un número configurable de iteraciones para aumentar la complejidad computacional del *hashing*.
- **Enmascarado de caracteres para proteger la información del usuario:** Se enmascara partes críticas de la información del usuario, tales como correo electrónico, dirección, comuna. A excepción de la contraseña ya que recibe un hash para ocultar sus caracteres. Se realiza este enmascaramiento desde el Backend para evitar filtraciones de los datos durante el trayecto hacia el dispositivo del usuario.

Se aplica enmascaramiento desde el Backend para evitar que usuarios no autorizados puedan visualizar la información real de los usuarios y para minimizar los riesgos de filtraciones durante la transmisión de datos hacia el cliente. Este

enfoque garantiza la seguridad y confidencialidad de los datos en tránsito y en el dispositivo del usuario.

Enfoque Técnico

- Correo electrónico: Solo se muestra parcialmente, por ejemplo: j***@ejemplo.com.
- Dirección: Se enmascara completamente, ejemplo: *****.
- Comuna: Puede mostrarse de manera truncada o redondeada a un nivel más genérico, como la región.

Exclusión de Contraseñas:

Las contraseñas no se enmascaran, ya que se almacenan en el *Backend* como un **hash criptográfico** generado mediante **PBKDF2 con SHA-256** (o el algoritmo configurado en *Django*). Esto asegura que las contraseñas reales nunca se envían ni almacenan en texto plano.

3.2.5.2 Eliminación de información sensible

La eliminación de información sensible es un procedimiento fundamental para garantizar la protección de los datos de los usuarios y prevenir su uso indebido en caso de una filtración. Este proceso debe asegurar que los datos eliminados sean completamente irrecuperables. Asimismo, es esencial proporcionar a los usuarios la facultad de solicitar y ejercer el derecho a la eliminación de sus datos personales en cualquier momento, respetando los principios de transparencia y control sobre su información.

1. Selección del Método de Eliminación

Para garantizar la eliminación segura de la información, se ha implementado un mecanismo basado en las recomendaciones de la norma *ISO*. Este proceso permite a los usuarios solicitar la eliminación de sus datos personales a través de un botón visual denominado “Eliminar cuenta”. Al utilizar esta opción, se procede a eliminar de forma segura la mayor parte de la información asociada al usuario almacenada en la base de datos, asegurando el cumplimiento de estándares de seguridad y normativas aplicables.

2. Mantenimiento de evidencia

Para garantizar la transparencia y trazabilidad del proceso de eliminación, se generarán registros que se almacenarán en la base de datos para futuras auditorías. Estos registros incluirán:

- Identificador del usuario eliminado (sin incluir el contenido del dato sensible).

- Fecha y hora de la eliminación.
- Usuario o proceso que ejecutó la eliminación.
- Resultado de la operación (éxito o error).
- Ejemplo de un log en formato *JSON*:

```
{ "timestamp": "2024-12-14T15:00:00Z", "action": "DELETE", "record_id": "12345",  
  "user": "admin", "result": "success" }
```

3.2.5.3 autenticación segura:

Para garantizar que los usuarios se autenticuen de manera segura en la plataforma y puedan acceder a sus servicios con protección adecuada, se implementarán los siguientes métodos de autenticación.

Factores Clave de Autenticación

Para garantizar la seguridad de la información sensible, se requiere al menos dos factores de autenticación en eventos donde el usuario intente acceder a datos críticos. Esto no se limita únicamente al inicio de sesión, sino que se aplica específicamente en acciones que implican la obtención o manejo de información confidencial del usuario. Los factores de autenticación pueden incluir:

- Algo que el usuario sabe (contraseña o PIN).
- Algo que el usuario tiene (un *token* físico o código enviado por SMS/Email).
- Algo que el usuario es (huella dactilar, reconocimiento facial).

De estos factores de autenticación se implementará el código enviado por email para casos, por ejemplo, cambio de contraseña.

No revelar información sensible antes de la autenticación: Los datos sensibles, como el correo electrónico completo o los datos de perfil, no se exponen hasta que la autenticación se realice con éxito.

Ejemplo: Mostrar un mensaje general como “Acceso denegado” en lugar de indicar si el correo electrónico o la contraseña es incorrecta, o decir genéricamente “correo o contraseña incorrectos”.

Alertas Generalizadas: Los mensajes de error no deben dar información adicional que facilite ataques.

- a. Correcto: “Credenciales inválidas.”
- b. Incorrecto: “La contraseña es incorrecta para este usuario.”

No Brindar Mensajes de Ayuda: El sistema no proporciona sugerencias ni pistas durante el proceso de autenticación, reduciendo así el riesgo de ataques de fuerza bruta o ingeniería social.

3.2.6 Mecanismos de seguridad en servicio en la nube

3.2.6.1 Gestión de identidades y accesos (IAM)

principio de privilegio mínimo

Este principio asegura que cada usuario, servicio o aplicación tenga únicamente los permisos necesarios para realizar sus tareas, limitando el impacto de accesos malintencionados o errores humanos.

Dentro de Google Cloud, los permisos se configuraron para que los usuarios solo puedan acceder a los recursos estrictamente necesarios.

Se evitó asignar permisos excesivos, como roles predeterminados que abarcan un amplio rango de permisos (*Editor*, *Owner*).

Roles y políticas

Se definieron roles específicos dentro de Google Cloud para cubrir las necesidades del proyecto:

Usuario Administrador: Permisos avanzados para gestionar recursos dentro del clúster de *Kubernetes*, incluida la creación, modificación y eliminación de recursos.

Acceso limitado a configuraciones de *Google Cloud* fuera del ámbito del clúster.

Usuario Visitante: Permisos restringidos a lectura y modificaciones limitadas en recursos de *Kubernetes* (por ejemplo, consultar *pods* o *logs*).

Restricciones Identificadas de IAM en AWS

Durante la implementación y configuración de mecanismos de seguridad en el entorno de *Amazon Web Services (AWS)*, se identificaron restricciones relacionadas con la gestión de identidades y accesos (IAM). Estas limitaciones afectaron directamente la capacidad para modificar o crear configuraciones específicas en el sistema IAM.

Falta de permisos administrativos:

No se contaba con privilegios suficientes para crear, modificar o eliminar roles, políticas o usuarios dentro de *AWS IAM*.

Las acciones disponibles se limitaron a lectura de configuraciones existentes y el uso de los roles asignados previamente por el administrador del entorno.

Cualquier modificación en políticas o roles requería la intervención de un administrador con privilegios completos en la cuenta de *AWS*, lo cual era imposible al tratarse de una cuenta de laboratorio.

3.2.7 Trabajos a futuro

En esta sección se identifican las actividades pendientes y las acciones proyectadas que serán fundamentales para fortalecer el desarrollo y la seguridad del proyecto *StuTrabajo*. Estas actividades no han sido implementadas en su totalidad, pero son prioritarias para garantizar la sostenibilidad, eficiencia y seguridad de la plataforma:

3.2.7.1 Políticas de Privacidad de Datos

Adopción de políticas de privacidad que aseguren el manejo responsable y ético de los datos personales, cumpliendo con las normativas locales e internacionales de protección de datos

3.2.7.2 Implementación de la Capacitación en Ciberseguridad

- Descripción: Desarrollo e implementación del programa de capacitación en ciberseguridad planteado en la sección 3.2.3.2, con el objetivo de concienciar a todos los involucrados sobre las mejores prácticas y los protocolos de respuesta ante incidentes.
- Resultados Esperados: Reducción de riesgos asociados a amenazas cibernéticas, como ataques de phishing, brechas de datos y accesos no autorizados.

3.2.7.3 Sistemas de Detección de Intrusos (IDS) y Sistemas de Prevención de Intrusos (IPS)

Descripción: Para reforzar la seguridad de la plataforma *StuTrabajo*. Estas herramientas serán clave para identificar y mitigar accesos no autorizados y ataques maliciosos en tiempo real.

Objetivos:

1. Detección Proactiva: Identificar actividades sospechosas o patrones de comportamiento que puedan indicar un intento de intrusión.
2. Prevención Activa: Bloquear automáticamente amenazas detectadas antes de que puedan comprometer los sistemas.
3. Monitoreo Continuo: Proveer un análisis constante del tráfico de red para identificar vulnerabilidades emergentes.

3.2.7.4 Monitoreo y Auditoría

Establecimiento de sistemas de monitoreo y auditoría para detectar y responder a actividades sospechosas o no autorizadas. Esto incluye el registro de eventos críticos y la generación de alertas en caso de incidentes de seguridad.

Sistemas de monitoreo

1. Registro centralizado de eventos

Implementación de un sistema de *logging* centralizado para consolidar todos los registros de actividades críticas, como:

- Accesos a la plataforma
- Modificaciones en configuraciones sensibles
- Intentos fallidos de autenticación

Herramientas de servicios en la nube recomendadas:

- *Amazon CloudWatch (AWS)* o *Google Cloud Logging (GCP)*: Servicios nativos para gestionar logs en la nube.

Supervisión del tráfico de red

Utilización de sistemas como *IDS (Intrusion Detection Systems)* y análisis de tráfico para identificar actividades no autorizadas.

Configuración de alertas para accesos a puertos no permitidos mediante *AWS CloudWatch Logs* o *VPC Flow Logs* de Google Cloud.

3.2.7.5 Segmentación de Red del *Backend*

Implementación de segmentación de red en el Backend para aislar y proteger los distintos microservicios. Esto incluye la utilización de nubes privadas virtuales (*VPC*) y firewalls para limitar la comunicación entre los microservicios únicamente a lo necesario, minimizando la superficie de ataque y mejorando la seguridad general del sistema.

3.3 Resumen de otras áreas

3.3.1 Diseño de página web (desarrollado por Matías Castillo)

En esta área se desarrolla la interfaz de usuario (*Frontend*) del portal de empleo destinado a la plataforma web. La labor implica no solo la creación de una apariencia visual atractiva, sino también el establecimiento de un flujo de navegación intuitivo que permita a los estudiantes encontrar de manera sencilla oportunidades laborales que se ajusten a sus perfiles, intereses o necesidades. Para ello, se ha recurrido a la integración de tecnologías web modernas, como *frameworks* de *JavaScript* (*React*, *Angular* o *Vue*) y librerías de estilos (*Bootstrap*, *Material UI* u otras), que facilitan la creación de componentes reutilizables y elementos gráficos responsivos. Además, se ha tenido especial cuidado en la optimización del rendimiento y en el cumplimiento de estándares de accesibilidad, asegurando que el sitio pueda ser utilizado por personas con diferentes capacidades y dispositivos. El objetivo final es generar una experiencia agradable, confiable y eficiente, que motive la permanencia de los usuarios, favorezca la interacción con el contenido y mejore la tasa de conversión de las postulaciones a empleos.

3.3.2 Descripción del Backend (desarrollado por Sebastián Poblete)

En esta área se desarrolla y consolida la lógica de negocio que sostiene la plataforma, funcionando como el núcleo que conecta las interfaces de usuario (tanto web como móvil) con las bases de datos y los servicios externos necesarios. El Backend se ha diseñado bajo una arquitectura escalable basada en microservicios, lo que permite distribuir las cargas de trabajo en diferentes módulos independientes, favoreciendo la mantención, la actualización y la resiliencia del sistema. Esta aproximación no solo facilita la adición de nuevas funcionalidades sin impactar el rendimiento global, sino que también brinda la flexibilidad necesaria para adaptarse a cambios en la demanda de usuarios. Asimismo, se han implementado protocolos de seguridad y cifrado de datos, junto con un sistema robusto de manejo de errores y auditorías, garantizando de este modo que la información personal y las operaciones realizadas dentro de la plataforma se mantengan íntegras y protegidas. De esta forma, el Backend actúa como el soporte fundamental que asegura la estabilidad, la eficiencia y la confiabilidad del ecosistema de servicios.

3.3.3 Descripción experiencia de usuario (UX) (desarrollado por Daniel Soto

Benavides)

En esta área se abordan aspectos estratégicos y metodológicos para optimizar la experiencia de usuario dentro de la plataforma. La UX no se limita a la apariencia o la navegabilidad, sino que considera todo el recorrido del usuario, desde que ingresa al portal hasta que completa una acción específica, como encontrar una oportunidad laboral relevante o postularse a una vacante. Para ello, se han implementado buenas prácticas de arquitectura de la información, flujos de interacción coherentes, y un diseño visual que fomente la confianza y la satisfacción. Además, se han utilizado técnicas de investigación de

usuarios, como encuestas, entrevistas y pruebas de usabilidad, con el fin de comprender las necesidades, motivaciones y comportamientos del público objetivo. De esta forma, las decisiones de diseño no se toman de manera aislada, sino respaldadas por datos empíricos y análisis cualitativos. El resultado es una plataforma en la que la facilidad de uso, la accesibilidad, la relevancia del contenido y la estética contribuyen de manera integral a una experiencia de usuario sólida, capaz de fidelizar a los visitantes y potenciar el cumplimiento de los objetivos del portal.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

Para garantizar una validación rigurosa y estandarizada de la solución propuesta, se ha seleccionado el Estándar de Verificación de Seguridad de Aplicaciones (ASVS) versión 4.0.3 de *OWASP* como marco de referencia principal. Este estándar representa una base establecida por la comunidad internacional para la verificación de la seguridad en aplicaciones web.

Descripción del Estándar

El *OWASP ASVS 4.0.3* es un marco de trabajo que establece tres niveles de verificación de seguridad:

Nivel 1 (L1): Verificaciones básicas de seguridad.

Nivel 2 (L2): Aplicaciones que manejan datos sensibles.

Nivel 3 (L3): Aplicaciones críticas que requieren el máximo nivel de confianza.

El estándar *OWASP ASVS* incluye una amplia variedad de controles diseñados para abordar diferentes contextos de seguridad, como arquitectura, criptografía, manejo de errores, entre otros. En esta memoria, se han seleccionado las secciones relacionadas con seguridad de contraseñas, manejo de sesiones, validación, sanitización y codificación, protección de datos, *API* y servicios web, y configuración, ya que son las más relevantes para los objetivos y necesidades del proyecto.

A continuación, se presenta el análisis de la validación de la solución implementada, basado en pruebas alineadas con las funcionalidades desarrolladas tanto en el Backend como en el Frontend de la plataforma web.

4.1 Seguridad de la contraseña

La seguridad de las contraseñas es uno de los pilares fundamentales para proteger las aplicaciones web de accesos no autorizados. En el contexto de este proyecto, donde estudiantes y solicitantes interactúan en un portal de empleo, garantizar la integridad de las credenciales de los usuarios es crucial para mantener la confianza y prevenir posibles brechas de seguridad.

Esta sección se centra en la implementación de controles relacionados con la gestión de contraseñas, basados en los lineamientos establecidos por *OWASP ASVS* en su nivel 1, diseñado para aplicaciones que manejan datos sensibles.

El propósito de implementar estos controles es minimizar el riesgo de comprometer cuentas de usuario y garantizar que las contraseñas almacenadas en el sistema cumplan con altos estándares de seguridad, contribuyendo así a la protección integral de la plataforma.

Password Security

4.1.1 Se verifica que la contraseña de los usuarios tenga al menos 12 caracteres en extensión (considerar también espacios).

Motivo

La implementación de una longitud mínima de 12 caracteres para las contraseñas es una medida esencial para aumentar la seguridad de las credenciales de los usuarios. Las contraseñas más largas son significativamente más difíciles de comprometer mediante ataques de técnicas de descifrado automatizado como ataques de fuerza bruta, ya que incrementan exponencialmente el tiempo necesario para adivinar combinaciones posibles.

Ejecución

Información de la prueba			
Historia de usuario	Registro de usuario	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Una invalidación de extensión de contraseña exitosa			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso incompleto de contraseña	Error señalando extensión insuficiente de caracteres	Aparición del error	Éxito

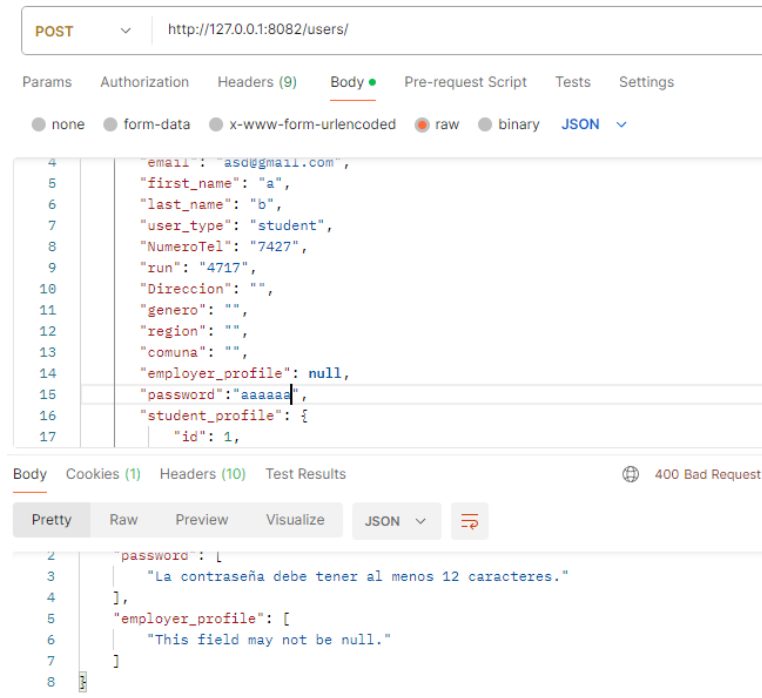


Figura 6: Prueba de extensión de contraseña.

Fuente: Elaboración propia.

4.1.2 Se verifica que las contraseñas de al menos 64 caracteres estén permitidas, y las contraseñas mayores a 128 caracteres sean denegadas.

Motivo

Permitir un rango amplio de longitudes de contraseñas, desde un mínimo de 12 caracteres hasta un máximo de 128 caracteres, es esencial para garantizar la flexibilidad y la seguridad en la creación de credenciales. Se asegura de que las contraseñas de al menos 64 caracteres sean aceptadas por el sistema, fomentando el uso de frases clave más complejas y seguras, que son significativamente más resistentes a ataques de fuerza bruta.

Asimismo, establecer un límite superior de 128 caracteres evita posibles problemas de rendimiento y almacenamiento en sistemas que procesan contraseñas utilizando algoritmos de *hashing*. Contraseñas extremadamente largas también podrían ser utilizadas maliciosamente para sobrecargar recursos del sistema o evadir validaciones.

Ejecución

Información de la prueba			
Historia de usuario	Registro de usuario	Prerrequisitos	Ninguno

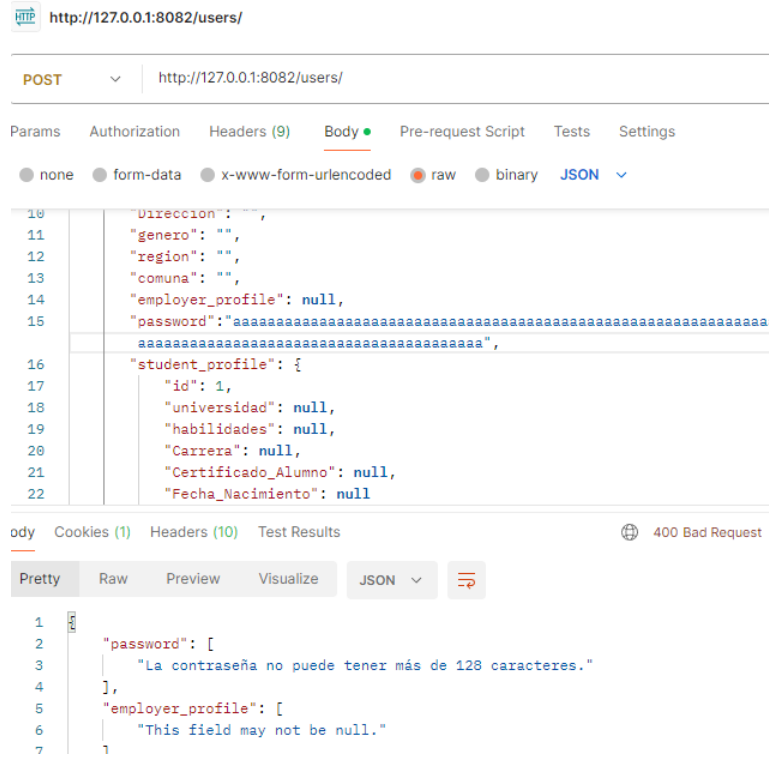


Figura 8: Prueba de ingreso de exceso de caracteres de contraseña.

Fuente: Elaboración propia.

4.1.4 Se verifica que todo tipo de carácter, incluido emojis sean permitidos en las contraseñas.

Motivo

Permitir el uso de todo tipo de caracteres en las contraseñas, incluidos emojis, es una medida que refuerza la seguridad y flexibilidad en la creación de credenciales. Este control fomenta contraseñas más impredecibles y únicas, lo que dificulta su descifrado mediante ataques de fuerza bruta, ya que amplía el conjunto de caracteres posibles que un atacante debe considerar, requiriendo que use más poder computacional y tiempo para intentar obtener la contraseña del usuario, dándole tiempo de reaccionar mediante la notificación de múltiples intentos de inicio de sesión y permitirle cambiar de contraseña antes de que sea descubierta.

Ejecución

Información de la prueba			
Historia de usuario	Registro de usuario	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			

- Validación exitosa			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de múltiples caracteres distintos	Omisión del error	Omisión del error	Éxito

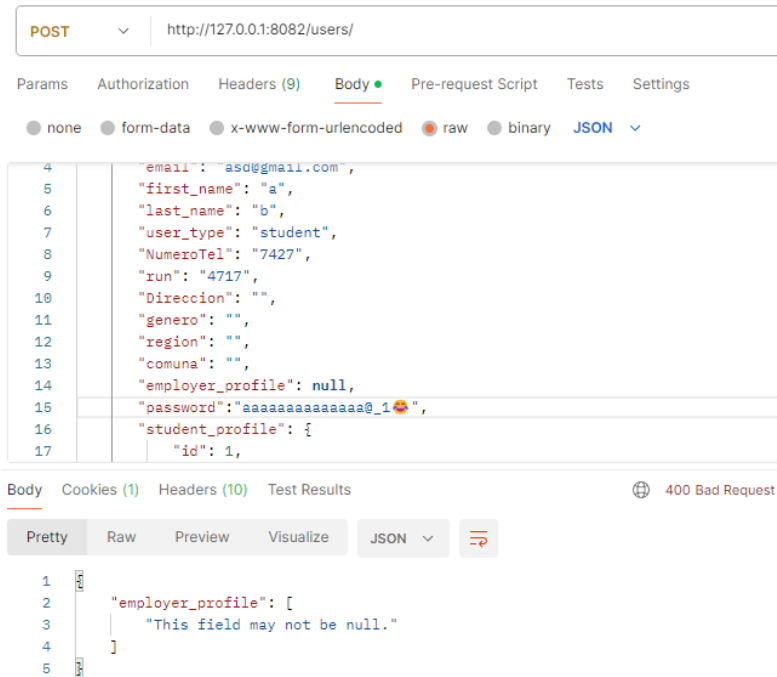


Figura 9: Prueba de ingreso variedad de caracteres en contraseña.

Fuente: Elaboración propia.

4.1.5 Se verifica que los usuarios puedan cambiar de contraseña.

Motivo

Permitir a los usuarios cambiar su contraseña es un control fundamental para garantizar la seguridad de sus cuentas y proteger la información sensible en caso de que las credenciales actuales sean comprometidas. Este mecanismo es crucial para mitigar riesgos relacionados con ataques de fuerza bruta, *phishing* o accesos no autorizados, ya que otorga a los usuarios la capacidad de responder rápidamente ante sospechas de vulneración.

Ejecución

Información de la prueba

Historia de usuario	Cambio de contraseña	Prerrequisitos	Tener una sesión iniciada
Dependencias	Tener una cuenta existente		
Resumen resultados			
- Cambio de contraseña exitoso			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de la contraseña antigua y la nueva junto con su confirmación	Modificación exitosa de contraseña	Modificación exitosa de contraseña	Éxito

localhost:8082/change-password/

Django REST framework

solicitante

Change Password

OPTIONS

GET /change-password/

HTTP 405 Method Not Allowed
Allow: PUT, OPTIONS
Content-Type: application/json
Vary: Accept
{
"detail": "Method \"GET\" not allowed."
}

Raw data HTML form

Old password 12345678910111213
New password 123456789101112
Confirm password 123456789101112
PUT

Figura 10: Prueba de cambio de contraseña.
Fuente: Elaboración propia.

Change Password

PUT /change-password/

HTTP 200 OK
 Allow: PUT, OPTIONS
 Content-Type: application/json
 Vary: Accept

```
{
  "message": "Contraseña actualizada exitosamente."
}
```

Raw data HTML form

Old password

New password

Confirm password

PUT

Figura 11: Resultado de cambio de contraseña.
 Fuente: Elaboración propia.

4.1.6 Se verifica que el cambio de contraseña requiera la contraseña actual y la nueva.

Motivo

Esta práctica mitiga riesgos como accesos indebidos por parte de usuarios malintencionados que ya tengan acceso temporal a la sesión activa del usuario o que intenten aprovechar vulnerabilidades en la autenticación. Al requerir la contraseña actual, el sistema añade una capa de verificación que dificulta el uso indebido de la funcionalidad de cambio de contraseña.

Ejecución

Información de la prueba			
Historia de usuario	Cambio de contraseña	Prerrequisitos	Tener sesión iniciada
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Modificación exitosa de contraseña - Detección exitosa de contraseña incorrecta			

Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de contraseña actual y nueva junto con su confirmación	Modificación exitosa de contraseña	Modificación exitosa de contraseña	Éxito
Ingreso incorrecto de contraseña actual, pero ingreso correcto de contraseña nueva junto con su confirmación	Detección exitosa de contraseña actual incorrecta	Detección exitosa de contraseña actual incorrecta	Éxito

localhost:8082/change-password/

Django REST framework

solicitante

Change Password

OPTIONS

GET /change-password/

```

HTTP 405 Method Not Allowed
Allow: PUT, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "detail": "Method \"GET\" not allowed."
}

```

Raw data

HTML form

Old password

12345678910111213

New password

123456789101112

Confirm password

123456789101112

PUT

Figura 12: Prueba de cambio de contraseña antigua.

Fuente: Elaboración propia.

The screenshot shows a web browser at the URL `localhost:8082/change-password/`. The page title is "Change Password". Below the title, the HTTP method is `PUT /change-password/`. The response status is `HTTP 200 OK`. The response headers are: `Allow: PUT, OPTIONS`, `Content-Type: application/json`, and `Vary: Accept`. The response body is a JSON object: `{ "message": "Contraseña actualizada exitosamente." }`. At the bottom, there is a form with three input fields: "Old password", "New password", and "Confirm password". A "PUT" button is located at the bottom right of the form.

Figura 13: Resultado de cambio de contraseña antigua.

Fuente: Elaboración propia.

The screenshot shows a web browser at the URL `localhost:8082/change-password/`. The page title is "Change Password". Below the title, the HTTP method is `PUT /change-password/`. The response status is `HTTP 400 Bad Request`. The response headers are: `Allow: PUT, OPTIONS`, `Content-Type: application/json`, and `Vary: Accept`. The response body is a JSON object: `{ "old_password": "Contraseña actual incorrecta." }`. At the bottom, there is a form with three input fields: "Old password", "New password", and "Confirm password". A "PUT" button is located at the bottom right of the form.

Figura 14: Prueba de cambio de contraseña incorrecta.

Fuente: Elaboración propia.

4.1.7 Se verifica que las contraseñas ingresadas durante el registro de cuenta, inicio de sesión y cambio de contraseña se comparen contra una lista de contraseñas comprometidas, como las 1,000 o 10,000 contraseñas más comunes que cumplan con la política de contraseñas del sistema. Si se utiliza una API externa para realizar esta comprobación, debe garantizarse que la contraseña en texto plano no se envíe, utilizando mecanismos como pruebas de conocimiento cero. En caso de que la contraseña esté comprometida, la aplicación debe exigir al usuario que cree una nueva contraseña que no haya sido comprometida.

Motivo

Comparar las contraseñas ingresadas por los usuarios durante el registro, inicio de sesión y cambio de contraseña contra una lista de contraseñas comprometidas es una medida fundamental para prevenir el uso de credenciales débiles o expuestas. Muchas contraseñas comunes o previamente comprometidas en violaciones de seguridad están disponibles públicamente en bases de datos, lo que facilita a los atacantes realizar ataques de fuerza bruta o de diccionario.

Implementar este control ayuda a garantizar que los usuarios no utilicen contraseñas fácilmente predecibles, como las 1,000 o 10,000 más comunes que cumplan con la política del sistema. Además, al exigir una contraseña no comprometida, se reduce significativamente el riesgo de que cuentas de usuario sean vulneradas.

Ejecución

Información de la prueba			
Historia de usuario	Registro de usuario	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de registro con contraseña vulnerable detectado exitosamente			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de una contraseña vulnerable	Alerta de contraseña comprometida	Alerta de contraseña comprometida	Éxito

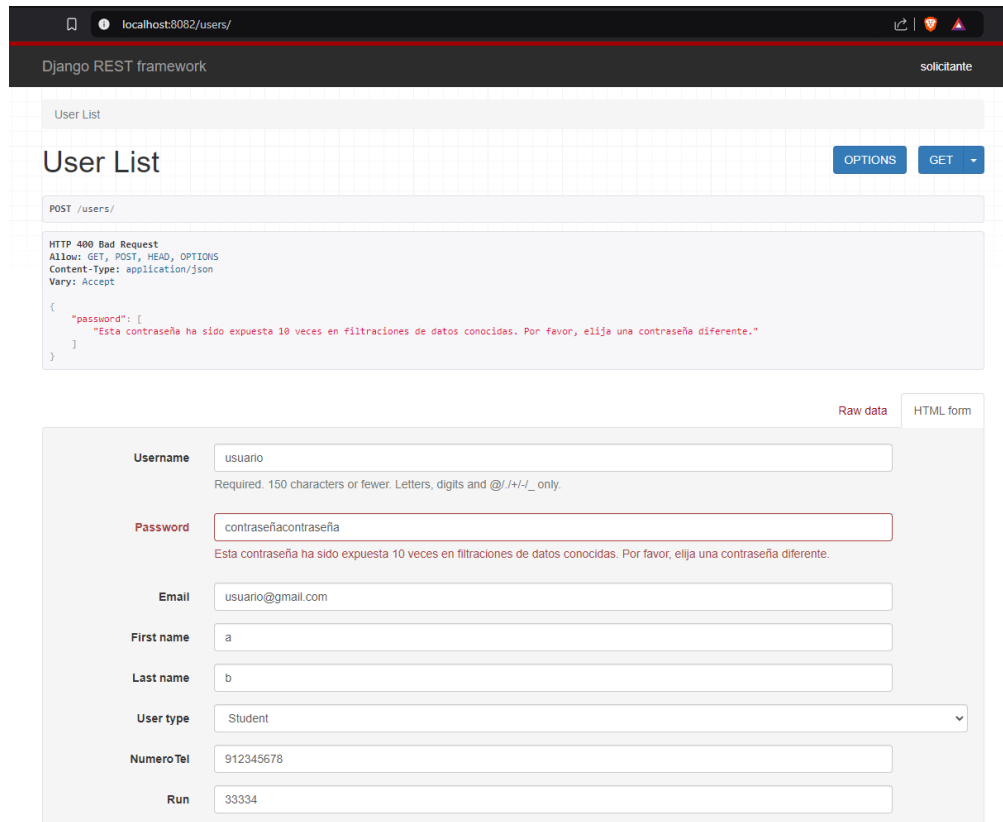


Figura 15: Prueba de contraseña insegura.

Fuente: Elaboración propia.

4.1.8 Se verifica que haya un “medidor” de fuerza de la contraseña para ayudar a los usuarios a establecer una contraseña fuerte.

Motivo

Esta funcionalidad no solo fomenta la adopción de contraseñas más seguras, sino que también reduce el riesgo de compromisos derivados de contraseñas débiles. Al incluir un medidor que evalúe criterios como la longitud, la complejidad (uso de mayúsculas, minúsculas, números y caracteres especiales) y la imprevisibilidad de la contraseña, el sistema mejora proactivamente la postura de seguridad de los usuarios.

Ejecución

Crea tu cuenta en StuTrabajo

1 Tipo de usuario 2 Datos personales 3 Credenciales 4 Ubicación 5 Información adicional

Correo Electrónico
correo@gmail.com

Contraseña
iil6pL4nJSs]

Segura

Confirma tu contraseña
.....

ANTERIOR SIGUIENTE

Figura 16: Visualización de medidor de contraseña.

Fuente: Elaboración propia.

4.1.9 Se verifica que no haya reglas limitando el tipo de caracteres posibles. No debería haber requerimientos de caracteres en mayúscula o minúscula o caracteres especiales.

Motivo

Las restricciones estrictas en el tipo de caracteres pueden llevar a la frustración de los usuarios y al uso de patrones previsibles que cumplen los requisitos mínimos pero que resultan vulnerables a ataques de fuerza bruta o diccionario. En lugar de imponer reglas restrictivas, se debe alentar el uso de contraseñas más largas, como frases de paso, que ofrecen mayor seguridad debido a su mayor entropía, mientras son más fáciles de recordar para los usuarios.

Ejecución

No se ha implementado restricciones del formato de la contraseña, únicamente su extensión.

4.1.10 Se verifica que no se obligue al usuario a cambiar periódicamente su contraseña durante determinado período ni establecer restricciones de historial de contraseñas.

Motivo

En lugar de imponer cambios periódicos, es más efectivo garantizar que los usuarios creen contraseñas sólidas desde el principio y proporcionar mecanismos para detectar y responder a compromisos de seguridad reales, como notificar al usuario si su contraseña en el momento de su creación aparece en listas de contraseñas comprometidas. Además, la eliminación de restricciones de historial fomenta que los usuarios elijan contraseñas que sean verdaderamente seguras y fáciles de recordar, en lugar de forzar combinaciones complejas pero predecibles para “terminar rápido la creación del usuario”.

Ejecución

No se ha implementado políticas que obliguen al usuario a realizar cambios de manera periódica su contraseña.

4.1.11 Se verifica que la funcionalidad de “pegar” dentro de los buscadores, gestores de contraseña externos estén permitidos.

Motivo

Los gestores de contraseñas son herramientas recomendadas para generar y almacenar contraseñas únicas y robustas, pero para funcionar correctamente requieren la capacidad de pegar las credenciales en los campos de entrada. Restringir esta funcionalidad no mejora la seguridad y, en cambio, puede empujar a los usuarios a adoptar prácticas menos seguras, como reutilizar contraseñas débiles o emplear combinaciones fáciles de recordar.

Ejecución

No se ha implementado alguna funcionalidad para limitar esta acción.

4.1.12 Se verifica que el usuario puede escoger entre ver temporalmente la contraseña totalmente desenmascarada, o temporalmente ver los últimos caracteres.

Motivo

Como medida adicional de seguridad, se restringe la visibilidad de la contraseña únicamente al proceso de registro o cambio de contraseña. Esto garantiza que las contraseñas no puedan ser visualizadas desde otras páginas o contextos, evitando exposiciones innecesarias de información sensible y mitigando posibles riesgos de manipulación en otros flujos de la aplicación.

Ejecución

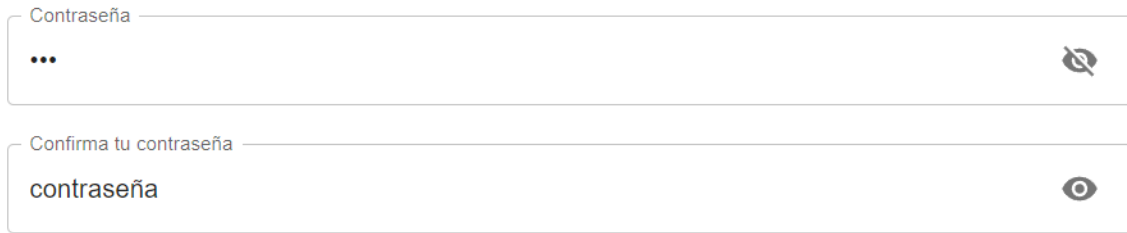
El formulario consta de dos campos de entrada de texto. El primer campo está etiquetado como 'Contraseña' y contiene tres puntos de relleno para ocultar el texto. El segundo campo está etiquetado como 'Confirma tu contraseña' y contiene el texto 'contraseña'. Ambos campos tienen un icono de ojo en el extremo derecho para alternar entre ver y ocultar el contenido.

Figura 17: Visualización de contraseña enmascarada.

Fuente: Elaboración propia.

4.2 Manejo de sesiones

El manejo de sesiones es un aspecto fundamental para garantizar la seguridad y continuidad de las interacciones autenticadas en el portal de empleo. Una gestión adecuada protege contra riesgos como el secuestro de sesión y ataques *CSRF*, asegurando que los *tokens* de sesión sean seguros y configurados correctamente. Dado que la plataforma maneja datos sensibles de usuarios y operaciones críticas, implementar medidas robustas en el manejo de sesiones refuerza la confianza de los usuarios y asegura un entorno confiable y seguro.

Fundamental Session Management Security

4.2.1 Se verifica que la aplicación nunca revela la sesión de *token* en los parámetros de *URL*.

Motivo

Revelar *tokens* de sesión en los parámetros de *URL* representa un riesgo significativo para la seguridad, ya que las *URLs* pueden ser registradas en historiales del navegador, *logs* del servidor o *proxies* intermedios, exponiendo los *tokens* a posibles actores malintencionados.

Ejecución

No se realizan envíos de *tokens* a través de parámetros.

Session Binding

4.2.2 Se verifica que la aplicación genera un nuevo *token* de sesión cuando un usuario se autentica.

Motivo

La generación de un nuevo *token* de sesión al momento de autenticarse es una práctica fundamental para garantizar la seguridad de las sesiones de usuario. Este control previene ataques como el *Session Fixation*, donde un atacante intenta reutilizar un *token* de sesión previamente conocido o preestablecido para acceder a una cuenta de usuario sin autorización.

Ejecución

26bbb569eb36e2be78ebcc669a8522ac83a34116

9e8f332923722f1fedefca362a48f72d317dfc3e

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Tener una cuenta creada		
Resumen resultados			
- 2 inicios de sesión para comprobar <i>token</i>			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso y cierre de credenciales 2 veces	<i>Tokens</i> distintos	<i>Tokens</i> distintos	Éxito

Login

Check the credentials and return the REST Token if the credentials are valid and authenticated.

Calls Django Auth login method to register User ID in Django session framework

Accept the following POST parameters: username, password
Return the REST Framework Token Object's key.

```
POST /auth/login/
```

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "key": "9e8f332923722f1fedefca362a48f72d317dfc3e"
}
```

Key

9e8f332923722f1fedefca362a48f72d317dfc3e

Figura 18: Prueba de generación de *token*

Fuente: Elaboración propia.

Login

Check the credentials and return the REST Token if the credentials are valid and authenticated.
Calls Django Auth login method to register User ID in Django session framework

Accept the following POST parameters: username, password
Return the REST Framework Token Object's key.

POST /auth/login/

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "key": "cdab48522b94822233d372ee202ba1dfce22cb00"
}
```

Key

cdab48522b94822233d372ee202ba1dfce22cb00

Figura 19: Resultado de generación de *token*

Fuente: Elaboración propia.

4.2.3 Se verifica que el *token* de sesión posee al menos 64 bits de entropía.

Motivo

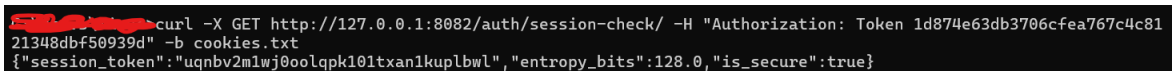
Hay que asegurar que los *tokens* de sesión posean al menos 64 bits de entropía es una medida crítica para proteger las sesiones contra ataques como el *Session Prediction*. La entropía de un *token* se refiere al grado de aleatoriedad que tiene, lo que determina la dificultad de adivinarlo o recrearlo mediante ataques automatizados.

Un *token* con baja entropía es más susceptible a ser adivinado, permitiendo que un atacante pueda hacerse pasar por un usuario legítimo. Por el contrario, un *token* con al menos 64 bits de entropía genera combinaciones suficientes para que sea prácticamente imposible predecirlo dentro de un tiempo razonable, incluso utilizando técnicas avanzadas como fuerza bruta.

Ejecución

Información de la prueba

Historia de usuario	Verificación entropía	Prerrequisitos	Tener la cuenta iniciada
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Obtención de bits de entropía de <i>token</i> de sesión			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de comando con <i>token</i> de autorización	Obtención de bits de entropía	Obtención de bits de entropía	Éxito



```

curl -X GET http://127.0.0.1:8082/auth/session-check/ -H "Authorization: Token 1d874e63db3706cfea767c4c8121348dbf50939d" -b cookies.txt
{"session_token": "uqnbv2m1wj0oolqpk101txanlkuplbwl", "entropy_bits": 128.0, "is_secure": true}

```

Figura 20: Visualización de bits de entropía de *token*.

Fuente: Elaboración propia.

4.2.4 Se verifica que la aplicación solo guarda sesiones de *token* en el buscador usando métodos seguros tales como cookies aseguradas apropiadamente.

Motivo

Garantizar que los *tokens* de sesión se almacenen en el navegador utilizando métodos seguros, como cookies configuradas correctamente, es crucial para proteger las sesiones de usuario contra accesos no autorizados. Los *tokens* de sesión son esenciales para mantener a los usuarios autenticados, pero si no se almacenan de manera segura, pueden ser robados o manipulados, exponiendo a la plataforma a riesgos como secuestro de sesiones o ataques de *Cross-Site Scripting* (XSS).

Ejecución

Irrealizable debido a entorno de trabajo local, por lo que no se puede habilitar conexiones *HTTPS*.

Session Termination

4.2.5 Se verifica que el *logout* y expiración invalida el *token* de sesión, o cualquier otro identificador que permita la autenticación.

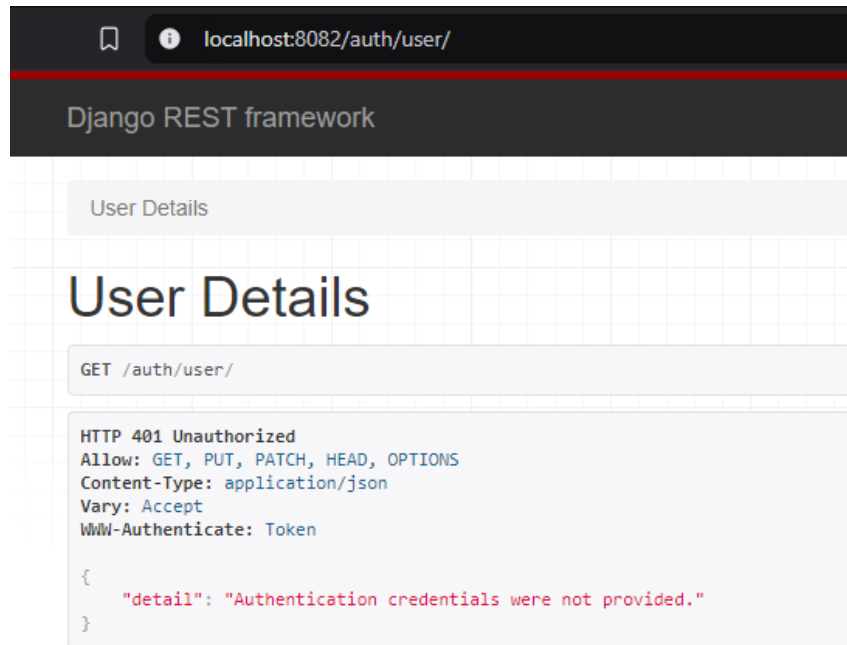
Motivo

El control de invalidación de *tokens* de sesión durante el *logout* o la expiración es esencial para garantizar que las sesiones finalizadas no puedan ser reutilizadas de forma malintencionada. Esto protege contra accesos no autorizados que podrían aprovechar *tokens* activos tras el cierre de sesión. Implementar este control asegura que cualquier identificador de autenticación quede completamente invalidado al finalizar la sesión, reforzando la seguridad de las interacciones y protegiendo los datos sensibles de los usuarios.

Ejecución

Token generado: 34a34a994c44b92adad7139039edb9fb50b495e3

Información de la prueba			
Historia de usuario	Cierre e inicio de sesión	Prerrequisitos	Tener la cuenta iniciada
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Intento de obtención de credenciales después de cierre de sesión			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
POST /auth/login/ POST /auth/logout/ GET /auth/user/	Error de credenciales de autenticación	Error de credenciales de autenticación	Éxito

Figura 21: Prueba de invalidacion de *token*.

Fuente: Elaboración propia.

4.2.6 Si un autenticador permite al usuario mantenerse iniciado, se verifica que la re-autenticación ocurra periódicamente tanto cuando el usuario esté activo o en un periodo de desconexión.

Motivo

En caso de que un autenticador permita al usuario mantenerse conectado, se implemente un mecanismo de re-autenticación periódica para minimizar riesgos de acceso no autorizado. La periodicidad de la re-autenticación protege las sesiones activas de posibles compromisos, incluso si el usuario está desconectado o no supervisa activamente su dispositivo. Esto refuerza la seguridad del sistema, garantizando que el acceso continuo esté sujeto a verificaciones regulares que validen la legitimidad de la sesión.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Tener una cuenta creada		
Resumen resultados			

- Expiración de contraseña en 2 horas			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de credenciales y espera de < 20s	Vencimiento de <i>token</i> de autenticación	Vencimiento de <i>token</i> de autenticación	Éxito

Login

Check the credentials and return the REST Token if the credentials are valid and authenticated. Calls Django Auth login r

Accept the following POST parameters: username, password Return the REST Framework Token Object's key.

POST /auth/login/

HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "key": "1d874e63db3706cfea767c4c8121348dbf50939d"
}
```

Key 1d874e63db3706cfea767c4c8121348dbf50939d

Figura 22: Visualización inicio de sesión exitoso.

Fuente: Elaboración propia.

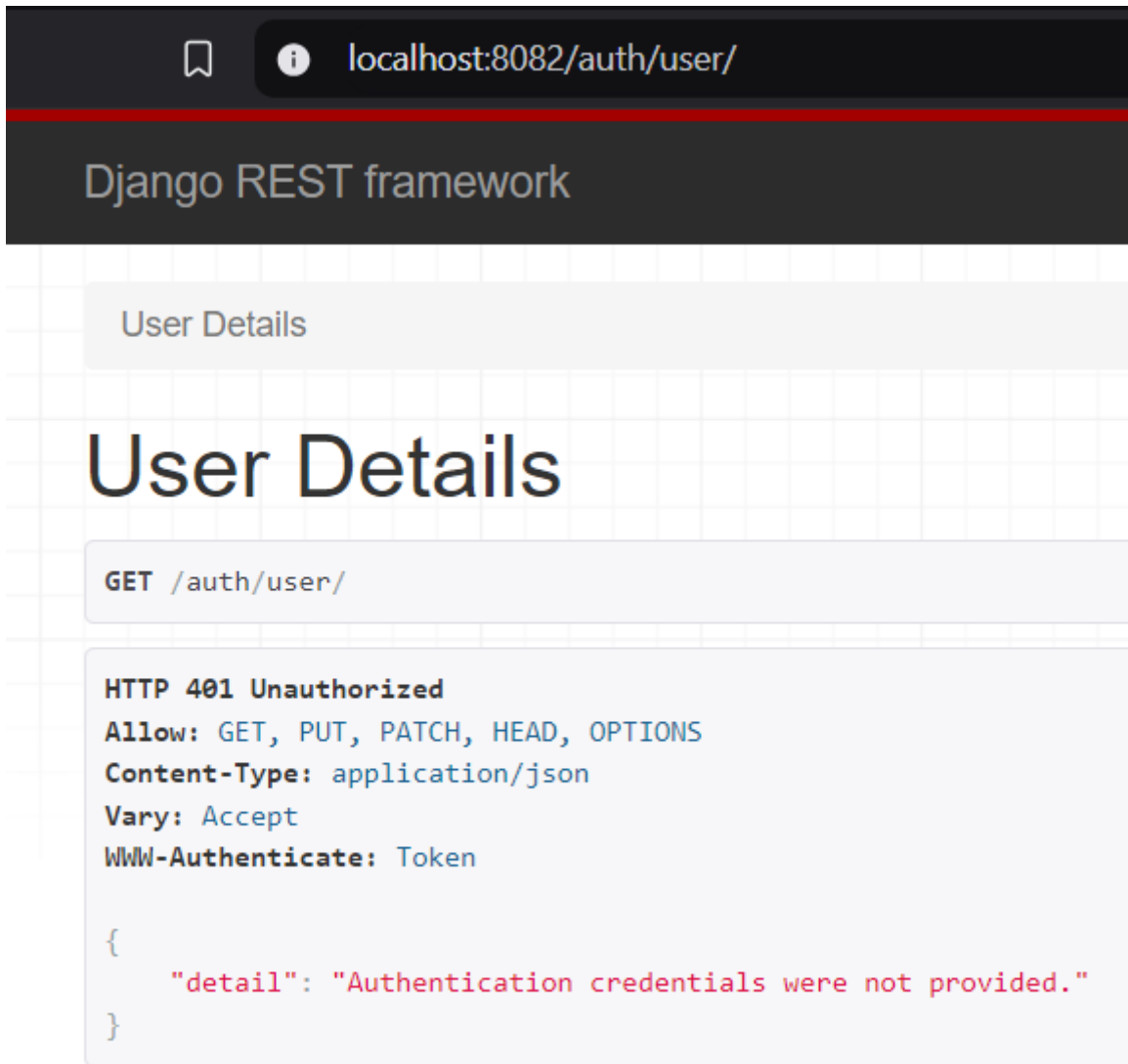


Figura 23: Visualización de sesión pasado 20 segundos.

Fuente: Elaboración propia.

Cookie-based Session Management

4.2.7 Se verifica que los *tokens* de sesión basada en cookies tienen el atributo “*Secure*” establecido.

Motivo

Sin el atributo ‘*Secure*’, los *tokens* de sesión pueden ser interceptados por atacantes que monitorean la red, comprometiendo la confidencialidad de la sesión y permitiendo ataques como el Session Hijacking. Al garantizar que los *tokens* solo viajen en conexiones seguras, se refuerza la protección de las credenciales de usuario y la integridad de las sesiones.

Ejecución

No se cuenta con un certificado SSL habilitado para la aplicación, por lo que no se ha implementado el atributo 'Secure' en las cookies de sesión y *CSRF*. El atributo 'Secure' solo puede ser activado en entornos que usen *HTTPS*, ya que garantiza que las cookies se transmitan de forma segura solo a través de conexiones cifradas. Dado que la aplicación no está configurada para usar *HTTPS*, el atributo 'Secure' no se aplica en este momento.

Defenses Against Session Management Exploits

4.2.8 Se verifica que la aplicación asegura un completo, valido inicio de sesión o requiere re-autenticacion o verificación secundaria antes de permitir cualquier transacción sensible o modificación de la cuenta.

Motivo

Exigir una re-autenticación o una verificación secundaria antes de permitir transacciones sensibles o modificaciones en la cuenta es una medida crítica para prevenir accesos no autorizados y proteger la integridad de la información del usuario. Este control asegura que cualquier acción significativa, como cambiar contraseñas, actualizar datos personales o realizar transacciones, solo pueda ser ejecutada por el propietario legítimo de la cuenta.

Esta práctica es fundamental para mitigar riesgos derivados de accesos indebidos, como aquellos causados por sesiones activas dejadas sin supervisión o por ataques de sesión *hijacking*. Además, proporciona una capa adicional de seguridad en el caso de dispositivos compartidos o entornos no confiables.

Ejecución

Información de la prueba			
Historia de usuario	Cambio de contraseña	Prerrequisitos	Haber iniciado sesión
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Tras obtención de código de verificación se cambia exitosamente la contraseña.			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Credencial de contraseña y código de verificación	Cambio exitoso en la contraseña	Cambio exitoso en la contraseña	Éxito

Change Password

PUT /change-password/

HTTP 200 OK

Allow: GET, PUT, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "message": "Contraseña actualizada correctamente"
}
```

Old password	
New password	
Confirm password	
Verification code	

Figura 24: Prueba de validación de código.

Fuente: Elaboración propia.

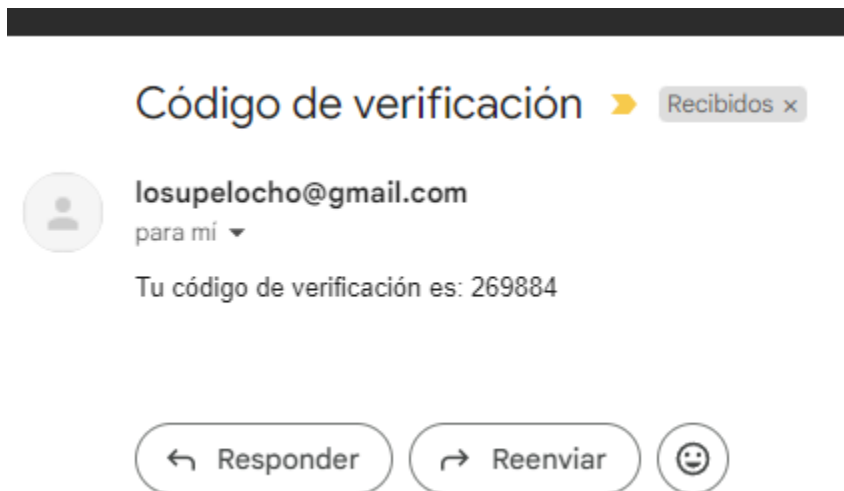


Figura 25: Visualización de envío de código.

Fuente: Elaboración propia.

Operation Level Access Control

4.2.9 Se verifica que los datos sensibles y las API estén protegidos contra ataques de Referencia Directa Insegura a Objetos (IDOR), que tienen como objetivo la creación, lectura, actualización y eliminación de registros, como crear o actualizar el registro de otra persona, ver los registros de todos o eliminar todos los registros.

Motivo

Proteger los datos sensibles y las API contra ataques IDOR es esencial para prevenir accesos o modificaciones no autorizadas de registros en la aplicación. Este tipo de ataque ocurre cuando un atacante manipula identificadores en las solicitudes a las API para obtener acceso a datos o realizar acciones sobre objetos que no le pertenecen. Por ejemplo, podría intentar visualizar registros de otros usuarios, modificar datos sensibles o eliminar información crítica del sistema.

Ejecución

Información de la prueba			
Historia de usuario	Obtención de datos ajenos	Prerrequisitos	Haber iniciado sesión
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Intentos de obtención de datos de distintos usuarios			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
GET /users/20/ (propio ID)	Ver todos los datos personales.	Datos mostrados.	Éxito
GET /users/8/ (ajeno ID)	Ver solo datos públicos del usuario.	Solo públicos visibles.	Éxito

```
GET /users/20/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 20,
  "username": "correo",
  "first_name": "a",
  "last_name": "sanchez",
  "user_type": "student",
  "region": "1",
  "comuna": "a",
  "employer_profile": null,
  "student_profile": {
    "id": 8,
    "universidad": "universdiada",
    "habilidades": "a",
    "Carrera": "a",
    "Certificado_Alumno": null,
    "Fecha_Nacimiento": "2024-11-26",
    "imagen": null,
    "id_usuario": 20,
    "username": "correo",
    "email": "correo@correo.com",
    "first_name": "a"
  },
  "email": "correo@correo.com",
  "NumeroTel": "912345678",
  "Direccion": "a"
}
```

Figura 26: Prueba de obtención de usuarios personales.

Fuente: Elaboración propia.

```
GET /users/8/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 8,
  "username": "a",
  "first_name": "a",
  "last_name": "a",
  "user_type": "admin",
  "region": "a",
  "comuna": "a",
  "employer_profile": null,
  "student_profile": null
}
```

Figura 27: Prueba de obtención de datos ajenos.

Fuente: Elaboración propia.

4.2.10 Se verifica que la aplicación o el framework implementen un mecanismo sólido de anti-CSRF para proteger las funcionalidades autenticadas, y que se utilicen medidas efectivas de anti-automatización o anti-CSRF para proteger las funcionalidades no autenticadas.

Motivo

Los ataques *CSRF* representan una amenaza significativa para aplicaciones web, permitiendo que un atacante realice acciones no autorizadas en nombre de un usuario autenticado sin su conocimiento. Para proteger las funcionalidades autenticadas, es fundamental implementar mecanismos sólidos anti-*CSRF* que garanticen que las solicitudes provengan de fuentes legítimas y autorizadas.

Ejecución

Django incluye un middleware llamado *CSRFViewMiddleware*, que protege la plataforma contra ataques de *Cross-Site Request Forgery (CSRF)*. Este *middleware* asegura que cada solicitud que requiera autenticación o realice cambios en el servidor (como solicitudes *POST*, *PUT* o *DELETE*) debe incluir un *token CSRF*. Este *token* es único por usuario y sesión, y actúa como una capa adicional de seguridad para verificar que las solicitudes provienen de fuentes legítimas, evitando acciones maliciosas desde otros sitios o scripts externos.

4.3 Validation, sanitization and encoding

La validación, sanitización y codificación de datos son esenciales para proteger el portal de empleo contra vulnerabilidades como inyecciones y *Cross-Site Scripting (XSS)*, asegurando que las entradas cumplan con los formatos esperados y que las salidas sean seguras según el contexto. Implementar estas medidas garantiza la integridad de los datos procesados, refuerza la seguridad de la plataforma y proporciona una experiencia confiable para los usuarios.

InPUT Validation

4.3.1 Se verifica que la aplicación tenga defensas contra ataques de contaminación de parámetros de HTTP, especialmente si el *framework* de la aplicación no diferencia entre las fuentes de los parámetros de la solicitud (*GET*, *POST*, *cookies*, *headers* o variables de entorno).

Motivo

Este tipo de ataque ocurre cuando un atacante manipula parámetros *HTTP* provenientes de diferentes fuentes, como solicitudes *GET*, *POST*, *cookies*, *headers* o variables de entorno, para alterar el comportamiento de la aplicación o ejecutar acciones maliciosas.

Ejecución

Información de la prueba			
Historia de usuario	Intento de contaminación de parámetros	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de obtención de 2 usuarios distintos a la vez			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
GET /users/?id=8&id=4	Error 400, parametros duplicados detectados	Error 400, parametros duplicados detectados	Éxito

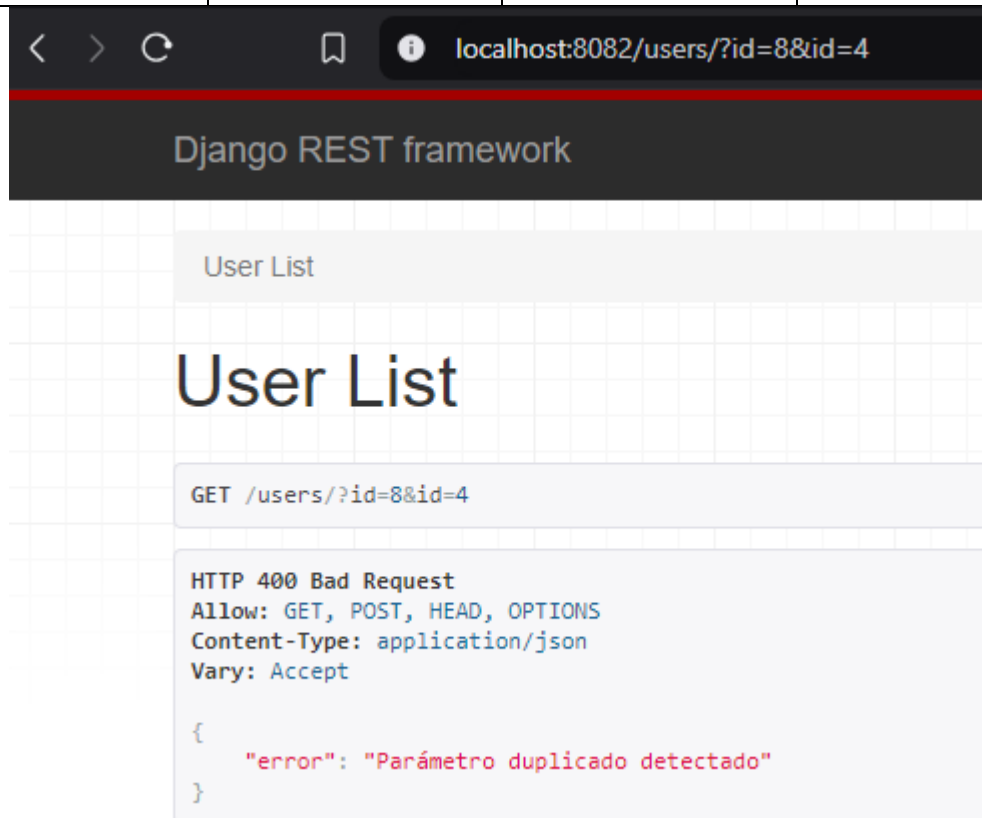


Figura 28: Prueba de contaminación de parámetros.

Fuente: Elaboración propia.

4.3.2 Se verificar que los frameworks protejan contra ataques de asignación masiva de parámetros, o que la aplicación cuente con contramedidas para evitar la asignación no segura de parámetros, como marcar campos como privados o utilizar mecanismos similares.

Motivo

Prevenir ataques de asignación masiva de parámetros es esencial para evitar que un atacante manipule y sobrescriba atributos sensibles o no autorizados en la aplicación. Este tipo de ataque ocurre cuando un sistema permite asignar directamente datos provenientes de solicitudes *HTTP* a los objetos o modelos de la aplicación, sin validar adecuadamente los campos que pueden ser actualizados o asignados.

Ejecución

Información de la prueba			
Historia de usuario	Ingreso de campos inexistentes	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de registro de usuario con campos inexistentes			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de campo unallowed_field con valor ThisShouldNotBeAllowed	'Este campo no está permitido.'	'Este campo no está permitido.'	Éxito

```
self.assertIn('unallowed_field', response.data, "El serializer debería indicar que unallowed_field no es válido")
AssertionError: 'unallowed_field' not found in {'Numerotel': ErrorDetail(string='Este campo no está permitido.', code='invalid')} : El serializer debería indicar que unallowed_field no es válido
o
-----
Ran 1 test in 0.010s

FAILED (failures=1)
Destroying test database for alias 'default'...
```

Figura 29: Prueba de error de campos no permitidos.

Fuente: Elaboración propia.

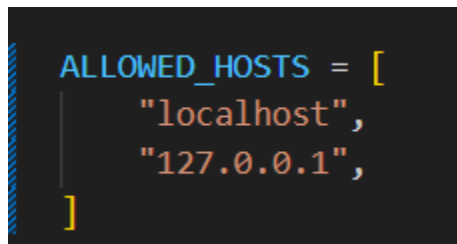
4.3.3 Se verifica que todas las entradas (campos de formularios *HTML*, solicitudes *REST*, parámetros de *URL*, encabezados *HTTP*, cookies, archivos por lotes, fuentes *RSS*, etc.) sean validadas utilizando validación positiva (listas de permitidos o *allow lists*).

Motivo

Validar todas las entradas a la aplicación utilizando listas de permitidos es una medida esencial para proteger el sistema contra vulnerabilidades relacionadas con datos no confiables o malintencionados. Este enfoque asegura que solo los valores previamente definidos y esperados sean aceptados, minimizando el riesgo de ataques como inyección *SQL*, inyección de comandos, *Cross-Site Scripting* y otros que explotan entradas mal gestionadas.

Ejecución

Esta solución no es viable de implementar en este momento, ya que requiere de un DNS configurado para que la prueba sea significativa. Dado que actualmente no se dispone de una configuración de DNS, se propone como alternativa implementar una lista de permitidos basada en localhost como equivalente funcional.

A screenshot of a code editor with a dark background. It shows a Python list definition for ALLOWED_HOSTS. The text is: ALLOWED_HOSTS = ["localhost", "127.0.0.1",]. The list is enclosed in square brackets, and each item is a string in quotes, separated by commas. The variable name is in blue, and the strings are in orange.

```
ALLOWED_HOSTS = [  
    "localhost",  
    "127.0.0.1",  
]
```

Figura 30: Visualización lista de permitidos.

Fuente: Elaboración propia.

4.3.4 Se verifica que los datos estructurados sean fuertemente tipados y validados contra un esquema definido, incluyendo caracteres permitidos, longitud y patrón (por ejemplo, números de tarjetas de crédito, direcciones de correo electrónico, números de teléfono o validación de que dos campos relacionados sean razonables, como comprobar que el suburbio y el código postal coincidan).

Motivo

Garantizar que los datos estructurados sean fuertemente tipados y validados contra un esquema definido es crucial para proteger la aplicación contra entradas maliciosas, errores de procesamiento y vulnerabilidades relacionadas con datos no controlados. El motivo de este control es asegurar que los datos enviados por los usuarios cumplan con formatos específicos, como caracteres permitidos, longitud, patrones predefinidos y relaciones lógicas entre campos, evitando inconsistencias o ataques como inyecciones de código.

Ejecución

Información de la prueba			
Historia de usuario	Creación de usuario	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de creación de usuario con valores inseguros o incorrectos			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de contraseña insegura junto con número y rut incompletos	Error indicando formato ideal y contraseña insegura	Error indicando formato ideal y contraseña insegura	Éxito

User List

POST /users/

```

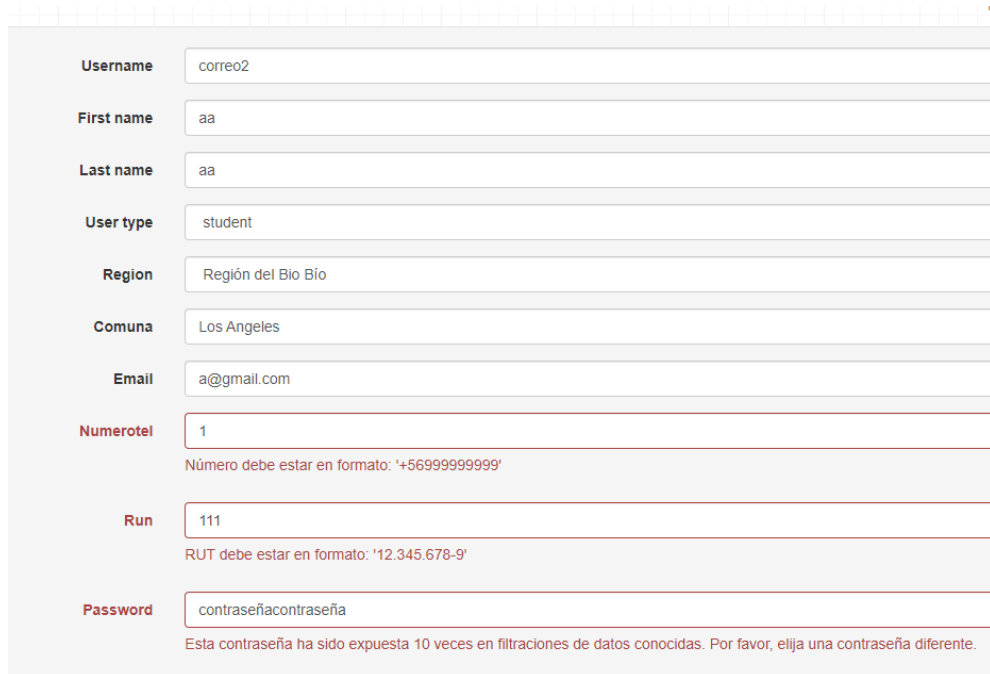
HTTP 400 Bad Request
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "NumeroTel": [
    "Número debe estar en formato: '+56999999999'"
  ],
  "run": [
    "RUT debe estar en formato: '12.345.678-9'"
  ],
  "password": [
    "Esta contraseña ha sido expuesta 10 veces en filtraciones de datos conocidas. Por favor, elija una contraseña diferente."
  ]
}

```

Figura 31: Prueba de validación de campos.

Fuente: Elaboración propia.



The image shows a registration form with the following fields and values:

- Username:** correo2
- First name:** aa
- Last name:** aa
- User type:** student
- Region:** Región del Bío Bío
- Comuna:** Los Angeles
- Email:** a@gmail.com
- Numerotel:** 1 (Error: Número debe estar en formato: '+56999999999')
- Run:** 111 (Error: RUT debe estar en formato: '12.345.678-9')
- Password:** contraseñacontraseña (Error: Esta contraseña ha sido expuesta 10 veces en filtraciones de datos conocidas. Por favor, elija una contraseña diferente.)

Figura 32: Visualización de validación de campos.

Fuente: Elaboración propia.

4.3.5 Se verifica que las redirecciones y reenviados de *URL* solo permitan destinos que aparezcan en una lista de permitidos (*allow list*), o que se muestre una advertencia al redirigir a contenido potencialmente no confiable.

Motivo

Controlar las redirecciones y reenviados de *URL* es fundamental para proteger la aplicación contra ataques como *phishing* y *Open Redirect*, en los que un atacante manipula las redirecciones para dirigir a los usuarios a sitios maliciosos.

Ejecución

Esta solución no se puede implementar en su totalidad, ya que requiere una lista de permitidos configurada con *DOM*inios válidos. Como no se dispone actualmente de dicha configuración, su equivalente funcional es implementar una lista de permitidos que limite las redirecciones a rutas internas del *localhost*.

Sanitization and Sandboxing

4.3.6 Se verifica que toda entrada de *HTML* no confiable proveniente de editores *WYSIWYG* o similares sea debidamente saneada utilizando una librería de sanitización de *HTML* o una característica del *framework*.

Motivo

Las entradas de *HTML* generadas a través de editores *WYSIWYG* (*What You See Is What You Get*) pueden incluir contenido malicioso si no se gestionan adecuadamente. Permitir que los usuarios ingresen *HTML* sin un saneamiento riguroso expone a la aplicación a vulnerabilidades como *Cross-Site Scripting*, que pueden ser explotadas para ejecutar scripts maliciosos, robar información confidencial o comprometer la experiencia de otros usuarios.

El saneamiento de entradas *HTML* no confiables es fundamental para eliminar elementos o atributos peligrosos, como scripts en línea, enlaces a recursos no seguros o etiquetas que puedan alterar el comportamiento del sistema. Utilizar librerías especializadas de sanitización o características integradas del *framework* asegura que solo se permita contenido *HTML* seguro y válido.

Ejecución

No se utiliza algún editor *WYSIWYG* durante la creación de *HTML*.

4.3.7 Se verifica que los datos no estructurados sean saneados para aplicar medidas de seguridad como la restricción de caracteres y la longitud.

Motivo

Los datos no estructurados, como texto libre o datos cargados por los usuarios, representan un riesgo significativo si no se procesan adecuadamente, ya que pueden contener contenido malicioso o no cumplir con los límites esperados en cuanto a longitud y caracteres permitidos. Este control asegura que los datos no estructurados sean saneados antes de ser almacenados o procesados, eliminando elementos peligrosos y aplicando restricciones claras.

El saneamiento incluye la validación de caracteres permitidos, la eliminación de contenido que no cumpla con las reglas definidas y el establecimiento de límites de longitud para evitar ataques como inyecciones de código, ataques de desbordamiento de *buffer* o inyecciones de *scripts* maliciosos.

Ejecución

No se utilizan datos no estructurados en la obtención ni procesamiento de información.

4.3.8 Se verifica que la aplicación sanee la entrada del usuario antes de pasarla a los sistemas de correo para proteger contra inyecciones SMTP o IMAP.

Motivo

La importancia del saneamiento de las entradas de los formularios antes de pasarlo al sistema de correo es debido a las inyecciones *SMTP* o *IMAP*, ya que ocurren cuando un atacante incluye comandos maliciosos en campos como el destinatario, el asunto o el cuerpo del correo, aprovechando la falta de validación o saneamiento para manipular el comportamiento del sistema. Estas inyecciones pueden permitir el envío de correos no

deseados (spam), la suplantación de identidad o el acceso a información confidencial almacenada en el servidor de correo.

Ejecución

No se ha implementado un sistema de correos en el que el usuario pueda ingresar texto

4.3.9 Se verifica que la aplicación evite el uso de *eval()* u otras características de ejecución de código dinámico. Cuando no haya alternativa, cualquier entrada del usuario que se incluya debe ser saneada o confinada en un entorno seguro (*sandbox*) antes de ser ejecutada.

Motivo

Evitar el uso de funciones como *eval()* y otras características de ejecución de código dinámico es crucial para proteger la aplicación contra vulnerabilidades que puedan derivar en la ejecución de código malicioso. Estas funciones son inherentemente peligrosas porque permiten interpretar y ejecutar código arbitrario en tiempo de ejecución, lo que puede ser explotado por un atacante para comprometer el sistema, en el caso de requerir estas características, se requiere del saneamiento o uso de un entorno seguro antes de ser procesadas las entradas del usuario.

Ejecución

No se ha implementado el uso de *eval()* en los microservicios o en la interfaz de usuario.

4.3.10 Se verifica que la aplicación se protege contra ataques de inyección en plantillas (*template injection*) asegurándose de que cualquier entrada del usuario que se incluya sea saneada o confinada en un entorno seguro (*sandbox*).

Motivo

Los ataques de inyección en plantillas ocurren cuando un atacante manipula entradas del usuario para inyectar código malicioso en plantillas que son procesadas por el sistema. Este tipo de ataque puede permitir la ejecución de código arbitrario en el servidor o el cliente, comprometiendo la seguridad de la aplicación, los datos de los usuarios y la funcionalidad general del sistema.

Ejecución

Al igual que las 2 siguientes evaluaciones, el usuario no puede subir archivos que no sean *.jpeg*, *.png*, *.pdf*, *.jpg* y no se realiza ninguna clase de procesamiento con estas imágenes.

4.3.11 Se verifica que la aplicación se protege contra ataques *SSRF (Server-side Request Forgery)*, validando o saneando los datos no confiables o los metadatos de archivos y campos de entrada de *URL*, y utilizando listas de permitidos (*allow lists*) para protocolos, *DOMinios*, rutas y puertos.

Motivo

Los ataques *SSRF* explotan la capacidad de una aplicación para realizar solicitudes *HTTP* desde el servidor. Un atacante puede manipular entradas como *URL* o metadatos de archivos para engañar al servidor y realizar solicitudes maliciosas a recursos internos o externos, lo que podría exponer información sensible, comprometer servicios internos del propio sistema y ya no tan solo de la plataforma.

Ejecución

No hay lógica dentro de la plataforma donde el usuario pueda manipular un parámetro o brindar una *URL* maliciosa, ni tampoco puede acceder a archivos del sistema operativo del servidor.

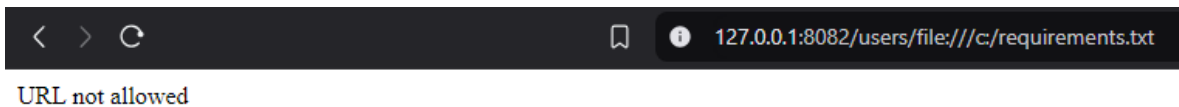


Figura 33: Prueba de obtención de archivos.

Fuente: Elaboración propia.

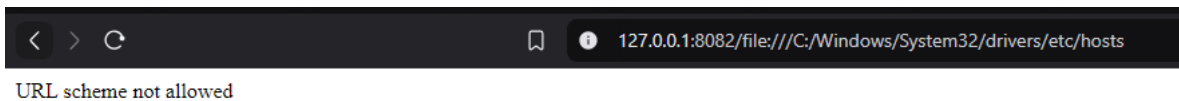


Figura 17.2: Prueba de obtención de datos de *host*.

Fuente: Elaboración propia.

4.3.12 Se verifica que la aplicación sanea, desactiva o aísla (*sandbox*) el contenido programable de *Scalable Vector Graphics (SVG)* proporcionado por el usuario, especialmente en lo que respecta a *XSS (Cross-Site Scripting)* que proviene de scripts en línea y el uso de *foreignObject*.

Motivo

El manejo de contenido *SVG* proporcionado por los usuarios introduce un riesgo significativo de vulnerabilidades de *XSS*, especialmente cuando los archivos *SVG* incluyen *scripts* en línea o elementos como *foreignObject*. Estas vulnerabilidades pueden ser explotadas por atacantes para inyectar código malicioso en la aplicación, comprometiendo tanto la seguridad de los datos como la integridad del sistema.

El *SVG* es un formato que combina gráficos vectoriales con capacidades programables, lo que lo convierte en una herramienta poderosa pero también en un vector potencial de ataques si no se gestiona adecuadamente. Saneando, desactivando o aislando el contenido

SVG, se evita que el código malicioso ejecutado dentro de un archivo SVG afecte a la aplicación o a los usuarios.

Ejecución

Información de la prueba			
Historia de usuario	Crear usuario	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de crear un usuario estudiante con svg malicioso			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de svg malicioso	Error indicando tipo de archivos permitidos	Error indicando tipo de archivos permitidos	Éxito

Figura 34: Prueba de ingreso de archivos no permitidos.
Fuente: Elaboración propia.

```

    ],
    "student_profile": {
      "Certificado_Alumno": [
        "Tipo de archivo no permitido. Use JPG, PNG o PDF"
      ]
    }
  }
}

```

Figura 35: Obtención de error de archivos no permitidos.
Fuente: Elaboración propia.

4.3.13 Se verifica que la aplicación sanea, desactiva o aísla (*sandbox*) el contenido programable o de lenguaje de plantillas de expresiones proporcionado por el usuario, como *Markdown*, *CSS* o hojas de estilo *XSL*, *BBCode* o contenido similar.

Motivo

Permitir a los usuarios proporcionar contenido programable o basado en lenguajes de plantillas, como *Markdown*, *CSS*, hojas de estilo *XSL*, *BBCode*, entre otros, introduce riesgos significativos de ejecución de código malicioso dentro de la aplicación. Estos lenguajes permiten la personalización y mejora del contenido visual, pero también pueden ser utilizados por atacantes para inyectar scripts o estilos manipulados con el objetivo de comprometer la seguridad del sistema o de los datos de los usuarios.

Ejecución

Información de la prueba			
Historia de usuario	Creación de empleo	Prerrequisitos	Tener cuenta solicitante
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Denegación de intento de subir un archivo no permitido de forma exitosa			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de archivo .pbix	Error tipo de archivo no permitido	Error tipo de archivo no permitido	Éxito

Figura 36: Prueba de ingreso de archivo no permitido..

Fuente: Elaboración propia.

```

data:
  imagen: ['Tipo de archivo no permitido. Use JPG, JPEG o PNG']
  [[Prototype]]: Object
    
```

Figura 37: Visualización de error de archivo no permitido.

Fuente: Elaboración propia.

OutPUT Encoding and Injection Prevention

4.3.14 Se verifica que la codificación de salida sea relevante para el intérprete y el contexto requerido. Por ejemplo, usar codificaciones específicamente para valores *HTML*, *SMTP*, y otros según lo requiera el contexto, especialmente para entradas no confiables (por ejemplo, nombres con caracteres *Unicode* o apóstrofes *ʼ* o *O'Hara*).

Motivo

Implementar una codificación de salida específica para el intérprete y el contexto requerido es esencial para proteger la aplicación contra ataques basados en la manipulación de datos no confiables, como XSS, inyecciones de comandos o alteraciones de formatos de datos. Cuando la salida no se codifica correctamente para el entorno en el que será utilizada (por

ejemplo, *HTML*, *SMTP*, *JSON*), se abre la posibilidad de que entradas maliciosas sean ejecutadas en lugar de tratarse como datos inofensivos.

Ejecución

Información de la prueba			
Historia de usuario	Creación empleo	Prerrequisitos	Tener cuenta solicitante
Dependencias	Tener cuenta creada		
Resumen resultados			
- Intento de ingresar código malicioso a través del formulario de creación de empleo			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de código alerta e ingreso de código alerta presionable	Codigo inutilizable	Codigo inutilizable	Éxito

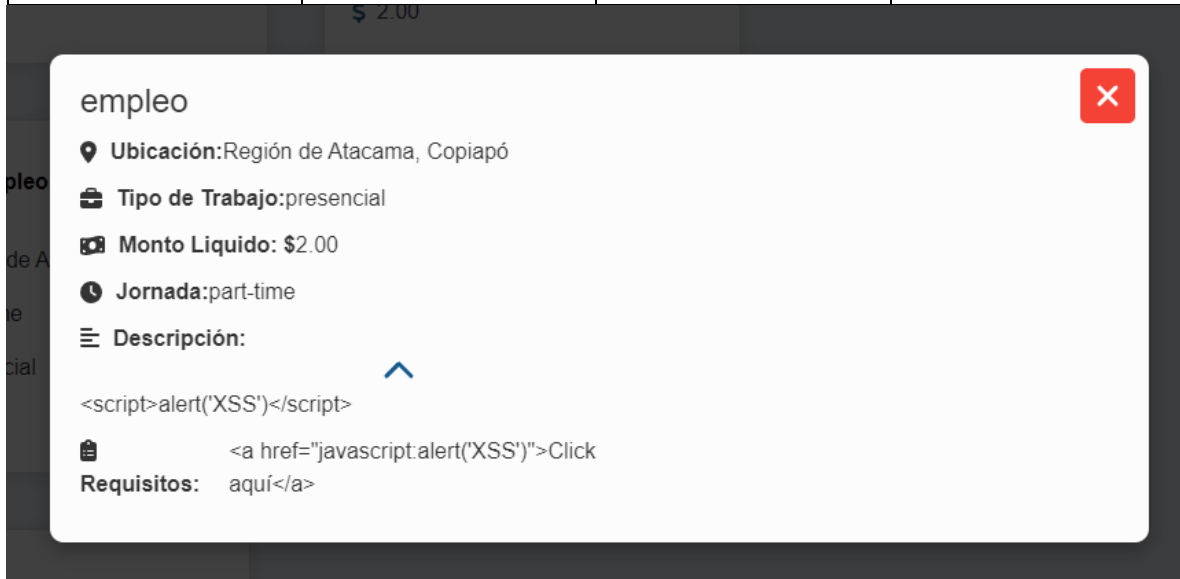


Figura 38: Prueba de ingreso de código mediante formulario.

Fuente: Elaboración propia.

4.3.15 Se verifica que la codificación de salida preserve el conjunto de caracteres y la configuración regional seleccionados por el usuario, de manera que cualquier punto de carácter *Unicode* sea válido y manejado de forma segura.

Motivo

Garantizar que la codificación de salida preserve el conjunto de caracteres y la configuración regional seleccionados por el usuario es esencial para evitar problemas de compatibilidad, pérdida de datos y vulnerabilidades de seguridad relacionadas con el manejo inadecuado de caracteres *Unicode*. Al asegurarse de que cualquier punto de carácter *Unicode* sea válido y manejado de manera segura, se protege la integridad de los datos y la experiencia del usuario, especialmente en aplicaciones que procesan entradas multilingües o sensibles a la configuración regional.

Cuando no se manejan correctamente los caracteres *Unicode*, pueden surgir vulnerabilidades como inyecciones de código malicioso, interpretaciones erróneas de datos o corrupción de texto, lo que pone en riesgo tanto la seguridad como la funcionalidad del sistema. Preservar el conjunto de caracteres asegura que los datos se procesen de manera consistente, sin introducir errores o brechas de seguridad.

Ejecución

React maneja todo dato ingresado por el usuario como texto (a menos que se indique explícitamente como un tipo numérico o especial), lo que garantiza que cualquier carácter *Unicode*, incluyendo caracteres especiales o internacionales, se muestre de forma segura como texto plano, sin que se ejecute ningún código malicioso.

4.3.16 Se verifica que la escapatoria de salida consiente del contexto, preferentemente automatizada o, en el peor de los casos, manual, proteja contra los ataques de XSS reflejados, almacenados y basados en el *DOM*.

Motivo

Los ataques de XSS, ya sean reflejados, almacenados o basados en el *DOM*, representan una de las vulnerabilidades más comunes y críticas en aplicaciones web. Estos ataques permiten a un atacante inyectar y ejecutar scripts maliciosos en el navegador de la víctima, comprometiendo la información del usuario, la integridad de la aplicación y la experiencia del usuario.

Ejecución

React maneja de manera automática los escapados de salida, por lo que, aunque se ingrese código malicioso a través de algún formulario, no se activará ya que se manipula como texto y no como script.

4.3.17 Se verifica que la selección de datos o consultas a la base de datos (por ejemplo, *SQL*, *HQL*, *ORM*, *NoSQL*) utilicen consultas parametrizadas, *ORMs*, marcos de entidades, o estén de otro modo protegidas contra ataques de inyección en la base de datos.

Motivo

Los ataques de inyección en bases de datos, como inyección *SQL* o *NoSQL*, son una de las vulnerabilidades más críticas y explotadas en aplicaciones web. Estos ataques ocurren cuando un atacante manipula entradas del usuario para alterar las consultas a la base de datos, lo que puede resultar en accesos no autorizados, filtraciones de datos sensibles o manipulación de información almacenada.

Ejecución

El *ORM* de *Django* implementado en el proyecto genera consultas *SQL* de manera segura y automáticamente parametriza las entradas proporcionadas por los usuarios, lo que ayuda a proteger contra ataques de inyección de *SQL*.

4.3.18 Se verifica que, en los casos donde no se utilicen mecanismos parametrizados o más seguros, se emplee codificación de salida específica para el contexto para proteger contra ataques de inyección, como el uso de escape de *SQL* para proteger contra inyecciones *SQL*.

Motivo

Cuando no se utilizan mecanismos parametrizados para manejar datos en una aplicación, como consultas *SQL* parametrizadas o *APIs* seguras, es fundamental implementar una codificación de salida específica para el contexto. Este control protege contra ataques de inyección, como la inyección *SQL*, que pueden comprometer bases de datos y exponer o alterar información sensible.

Ejecución

No se realizan consultas fuera de la herramienta *ORM* de *Django*.

4.3.19 Se verifica que la aplicación se protege contra ataques de inyección *JSON*, ataques de evaluación de *JSON* (*JSON eval*) y la evaluación de expresiones *JavaScript*.

Motivo

La evaluación directa de *JSON* (*JSON eval*) o expresiones *JavaScript* sin la debida validación y sanitización representa un riesgo significativo, ya que estas prácticas pueden permitir la ejecución de código malintencionado en el servidor o en el cliente, comprometiendo tanto los datos como la funcionalidad de la aplicación.

Ejecución

Información de la prueba			
Historia de usuario	inyección <i>SQL</i>	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de inyección <i>SQL</i> a través de la contraseña o username			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de código select * from user;	Registro de usuario sin revelar información	Registro de usuario sin revelar información	Éxito

POST http://127.0.0.1:8082/users/

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```

1  {
2    "username": "intruso1",
3    "first_name": "intruso",
4    "last_name": "intruso",
5    "user_type": "student",
6    "region": "Región del Bio Bio",
7    "comuna": "Los Angeles",
8    "email": "tramo@gmail.com",
9    "NumeroTel": "+56999999999",
10   "run": "12.345.678-5",
11   "password": "select * from user;",
12   "employer_profile": {
13     "Nombre_Compnia": null,
14     "Compnia_rut": null,
15     "Descripcion": null
  
```

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize

```

{"id":37,"username":"intruso1","first_name":"intruso","last_name":"intruso","user_type":"student"}
  
```

Figura 39: Prueba de inyección SQL.

Fuente: Elaboración propia.

```
    },  
    {  
      "id": 36,  
      "username": "select * from user;",  
      "first_name": "intruso",  
      "last_name": "intruso",  
      "user_type": "student"  
    },  
    {  
      "id": 37,  
      "username": "intruso1",  
      "first_name": "intruso",  
      "last_name": "intruso",  
      "user_type": "student"  
    },  
    {  
      "id": 38,  
      "username": "intruso2",  
      "first_name": "intruso",  
      "last_name": "intruso",  
      "user_type": "student"  
    },  
    {  
      "id": 40,  
      "username": "intruso3",  
      "first_name": "intruso",  
      "last_name": "intruso",  
      "user_type": "student"  
    }  
  ]
```

Figura 40: Visualización de inyección SQL.

Fuente: Elaboración propia.

4.3.20 Se verifica que la aplicación se protege contra vulnerabilidades de inyección *LDAP*, o que se han implementado controles de seguridad específicos para prevenir la inyección *LDAP*.

Motivo

Proteger la aplicación contra vulnerabilidades de inyección *LDAP* (*Lightweight Directory Access Protocol*) es esencial para evitar que los atacantes manipulen las consultas a los directorios de autenticación y autorización. Este tipo de ataque ocurre cuando los datos ingresados por el usuario no son validados correctamente y se incluyen directamente en las consultas *LDAP*, permitiendo al atacante alterar la lógica de estas consultas para acceder a información sensible, modificar datos o incluso tomar el control de las cuentas de usuario.

Ejecución

En nuestra plataforma, no se utiliza *LDAP* para la autenticación ni para la gestión de usuarios, grupos o directorios. En lugar de *LDAP*, estamos utilizando métodos de autenticación basados en *tokens* para gestionar la seguridad de los usuarios.

4.3.21 Se verifica que la aplicación se protege contra la inyección de comandos del sistema operativo (OS) y que las llamadas al sistema operativo utilizan consultas parametrizadas o emplean codificación contextual de la salida de la línea de comandos.

Motivo

Proteger la aplicación contra vulnerabilidades de inyección de comandos del sistema operativo (*OS Command Injection*) es fundamental para prevenir que un atacante ejecute comandos arbitrarios en el sistema host. Este tipo de ataque ocurre cuando los datos proporcionados por el usuario son inadecuadamente validados y son incluidos directamente en llamadas al sistema operativo, permitiendo que un atacante manipule o comprometa el entorno de ejecución.

Ejecución

La plataforma no pasa a entornos donde se puedan ejecutar comandos de sistema, y además de que las entradas se sanitizan junto con la realización de escapados de entradas.



Not Found

The requested resource was not found on this server.

Figura 41: Prueba de inyección de comandos *PowerShell*.

Fuente: Elaboración propia.

4.3.22 Se verifica que la aplicación se protege contra ataques de inclusión de archivos locales (LFI) o inclusión de archivos remotos (RFI).

Motivo

Los ataques de inclusión de archivos locales (*LFI*) y remotos (*RFI*) son vulnerabilidades críticas que permiten a un atacante cargar y ejecutar archivos maliciosos en el servidor de la aplicación. Estos ataques ocurren cuando una aplicación no valida o controla adecuadamente las rutas de archivos proporcionadas por los usuarios, permitiendo que se carguen archivos locales no autorizados o se incluyan archivos remotos desde fuentes externas.

En el caso de *LFI*, los atacantes pueden acceder a archivos confidenciales del servidor, como configuraciones o claves, mientras que en *RFI* pueden cargar scripts maliciosos desde un servidor externo para comprometer el sistema o robar datos. Estas vulnerabilidades representan un riesgo significativo para la integridad, confidencialidad y disponibilidad de la aplicación.

Ejecución

No se ha implementado algún tipo de lectura de archivo ni para acceder a archivos específicos, la plataforma se limita específicamente a almacenar y entregar archivos *.pdf*, *.png*, *.jpg*, *.jpeg*

4.3.23 Se verifica que la aplicación se protege contra ataques de inyección *XPath* o inyección *XML*.

Motivo

En una inyección *XPath*, un atacante puede alterar consultas utilizadas para recuperar información de documentos *XML*, permitiendo, por ejemplo, acceder a registros que deberían estar restringidos. Por otro lado, una inyección *XML* puede explotar elementos maliciosos en documentos *XML* para realizar acciones como ataques de ejecución de código malicioso.

Ejecución

No se manipulan datos en *XML*, únicamente en *JSON*.

4.4 Data protection

La protección de datos es un componente clave para garantizar la seguridad y privacidad de la información sensible procesada en el portal de empleo. Implementar controles efectivos asegura que los datos de los usuarios, como información personal y profesional, estén protegidos contra accesos no autorizados, manipulaciones o filtraciones. Estas medidas no solo cumplen con los estándares de seguridad, sino que también refuerzan la confianza de los usuarios en la plataforma al garantizar un manejo seguro y responsable de su información.

Sensitive Private Data

4.4.1 Se verifica que los datos sensibles se envíen al servidor en el cuerpo del mensaje *HTTP* o en los encabezados, y que los parámetros de la cadena de consultas de cualquier verbo *HTTP* no contengan datos sensibles.

Motivo

Garantizar que los datos sensibles se transmitan al servidor de manera segura es fundamental para proteger la información personal y confidencial de los usuarios frente a exposiciones no autorizadas.

Ejecución

En la plataforma, todos los datos sensibles, como credenciales de usuario, contraseñas y *tokens* de autenticación se envían siempre de forma segura a través del cuerpo del mensaje *HTTP*, utilizando los métodos *HTTP* apropiados como *POST*, *PUT* o *DELETE*. En ningún momento, los datos sensibles son transmitidos a través de la cadena de consulta de la *URL*.

4.4.2 Se verifica que los usuarios tengan un método para remover o exportar sus datos bajo demanda.

Motivo

Permitir la exportación de datos garantiza que los usuarios puedan obtener una copia de su información personal en un formato estructurado y legible, mientras que la opción de remover datos asegura que puedan solicitar la eliminación de información que ya no desean mantener en la plataforma. Estas capacidades refuerzan la transparencia de la aplicación y brindan a los usuarios la seguridad de que sus datos están bajo su control.

Ejecución

Información de la prueba			
Historia de usuario	Eliminación de usuario	Prerrequisitos	Haber iniciado sesión
Dependencias	Tener cuenta creada		
Resumen resultados			
- Intento de eliminación de cuenta exitosa			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Presión de botón para borrar cuenta	Estatus 204	Estatus 204	Éxito

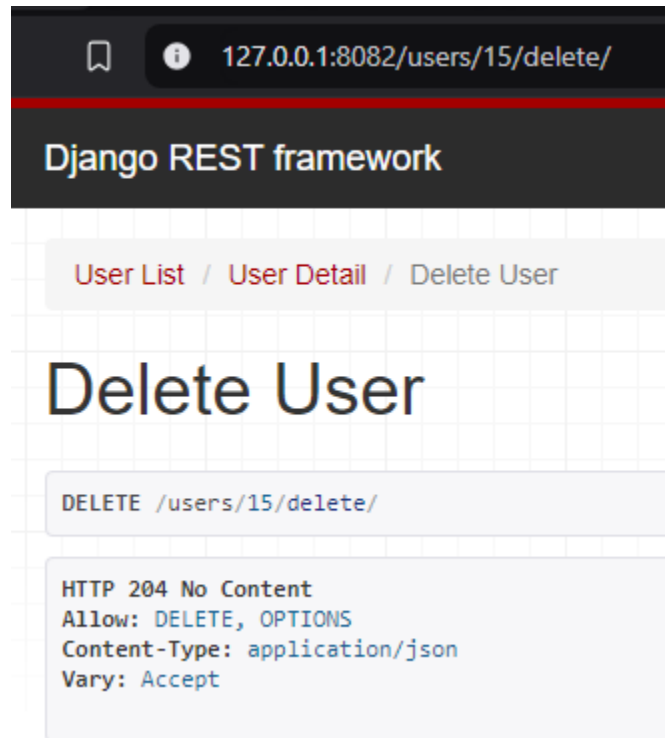


Figura 42: Prueba de eliminación de usuario.

Fuente: Elaboración propia.

4.4.3 Se verifica que se proporcione a los usuarios un lenguaje claro respecto a la recolección y el uso de la información personal proporcionada, y que los usuarios hayan dado su consentimiento explícito (*opt-in*) para el uso de esos datos antes de que sean utilizados de cualquier manera.

Motivo

Proporcionar a los usuarios un lenguaje claro sobre la recolección y el uso de su información personal, así como garantizar que otorguen su consentimiento explícito (*opt-in*) antes de utilizar sus datos, es fundamental para respetar los derechos de privacidad.

Ejecución



Figura 43: Visualización de términos y condiciones.

Fuente: Elaboración propia.

4.4.4 Se verifica que todos los datos sensibles creados y procesados por la aplicación hayan sido identificados y garantizar que exista una política establecida sobre cómo manejar esos datos sensibles.

Motivo

La identificación de todos los datos sensibles creados y procesados por la aplicación, junto con el establecimiento de una política clara para su manejo, es fundamental para garantizar la protección de la información y el cumplimiento de las mejores prácticas de seguridad.

Ejecución

En nuestra plataforma, hemos identificado los datos sensibles como aquellos que incluyen credenciales de usuario (como contraseñas y *tokens* de autenticación), datos personales (como nombre, correo electrónico, dirección, datos universitarios o de solicitante) y cualquier otra información que pueda comprometer la privacidad de los usuarios

Para garantizar la protección de estos datos, hemos implementado una política de manejo de datos sensibles que cubre aspectos clave como el acceso restringido basado en roles y autenticación robusta, la eliminación segura de los datos cuando ya no son necesarios y la

diferenciación entre datos públicos y privados para no comprometer información sensible de los usuarios al momento de reflejarse su postulación en caso de ser estudiantes o al mostrar su empleo en caso de ser el solicitante.

Se deja únicamente como datos públicos los siguientes: 'id', 'username', 'first_name', 'last_name', 'user_type', 'region', 'comuna'.

4.5 API and web service

Las API y los servicios web son fundamentales para la interacción entre sistemas y usuarios, pero también representan un punto crítico para la seguridad de la plataforma. Implementar controles efectivos en esta área garantiza que las comunicaciones sean seguras, que solo se permitan solicitudes legítimas y que los datos intercambiados estén protegidos contra ataques malintencionados. Estas medidas refuerzan la integridad y confiabilidad del portal de empleo, asegurando que las funcionalidades ofrecidas sean seguras y eficaces.

RESTful Web Service

4.5.1 Se verifica que los métodos HTTP RESTful habilitados sean una opción válida para el usuario o la acción, como evitar que usuarios normales utilicen los métodos DELETE o PUT en API o recursos protegidos.

Motivo

Controlar y restringir los métodos HTTP RESTful habilitados para usuarios y acciones específicas es fundamental para proteger la integridad y seguridad de las APIs y los recursos protegidos. Métodos como **DELETE** o **PUT**, si no se gestionan adecuadamente, pueden ser utilizados indebidamente por usuarios no autorizados, lo que podría derivar en la eliminación o modificación de datos críticos en la aplicación.

Ejecución

Información de la prueba			
Historia de usuario	Modificación usuario	Prerrequisitos	Haber iniciado sesión
Dependencias	Tener cuenta creada		
Resumen resultados			
- Intento fallido de modificar datos de un usuario ajeno - Intento fallido de eliminar una cuenta ajena			
Detalle resultados			

Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Alteración de datos personales del otro usuario	Error de permisos	Error de permisos	Éxito
Ingreso del identificador del otro usuario	Error de permisos	Error de permisos	Éxito

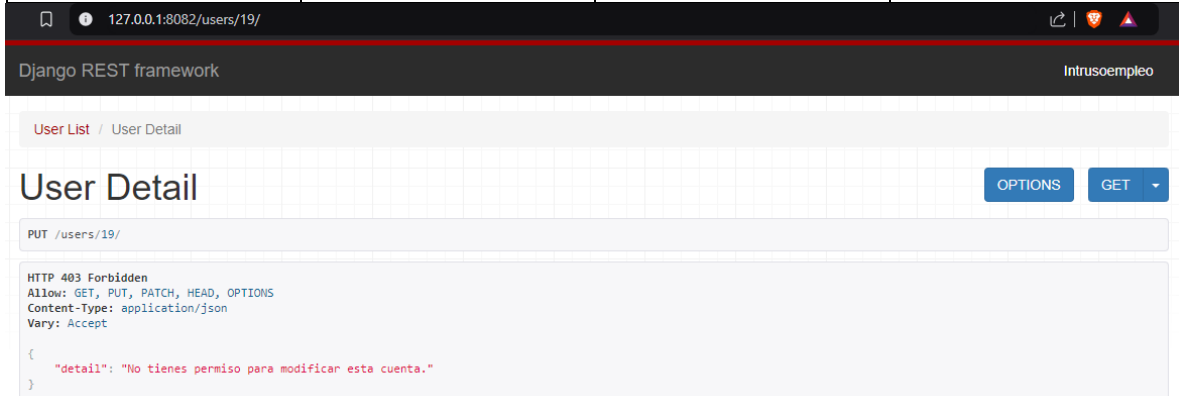


Figura 44: Prueba de modificación de usuario ajeno.

Fuente: Elaboración propia.

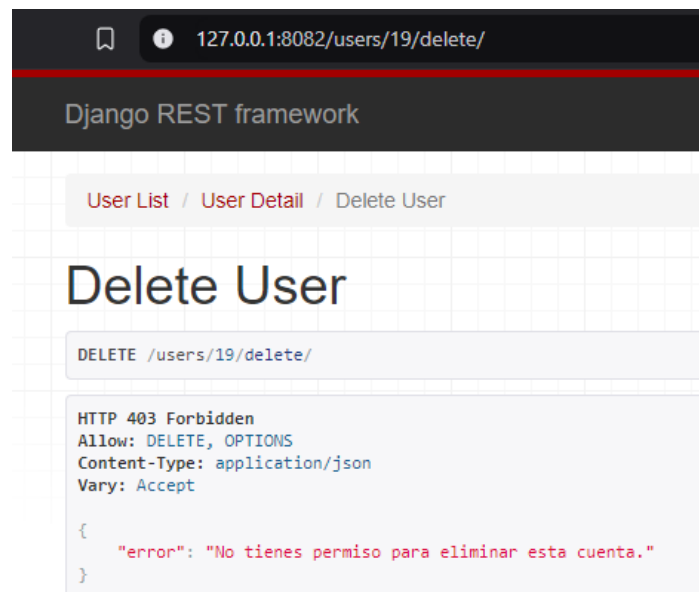


Figura 45: Visualización de error de modificación de usuario ajeno.

Fuente: Elaboración propia.

4.5.2 Se verifica que se haya implementado una validación de esquema *JSON* y que se haya verificado antes de aceptar la entada.

Motivo

La validación del esquema permite definir reglas claras sobre los campos esperados, sus tipos de datos, longitud, formato y posibles valores, asegurando que cualquier entrada que no cumpla con estos criterios sea rechazada. Esto es particularmente importante en *APIs RESTful*, donde el intercambio de datos estructurados en formato *JSON* es fundamental para la comunicación entre el cliente y el servidor.

Ejecución

Todo formulario que se envíe del *React* a *Django* se valida su formato (mostrado en figuras anteriores) y se hace una segunda validación por parte de *Django* en caso de que el usuario burle alguna restricción.

4.5.3 Se verifica que los servicios web *RESTful* que utilizan cookies estén protegidos contra ataques de *Cross-Site Request Forgery (CSRF)* mediante el uso de al menos uno o más de los siguientes mecanismos: patrón de doble envío de cookies, nonces de *CSRF* o comprobaciones del encabezado de solicitud *Origin*.

Motivo

Los ataques de *CSRF* son una amenaza crítica en aplicaciones que utilizan cookies para autenticar solicitudes, ya que pueden permitir que un atacante realice acciones malintencionadas en nombre de un usuario autenticado sin su consentimiento. Estos ataques explotan la confianza que un servidor tiene en las cookies del navegador del usuario y pueden derivar en accesos no autorizados, modificaciones de datos o acciones perjudiciales en la aplicación.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Obtención de <i>token</i> de sesión y <i>CSRF</i> al iniciar sesión			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?

Ingreso de datos para iniciar sesión		Inicio de sesión con <i>tokens</i> adicionales	Inicio de sesión con <i>tokens</i> adicionales	Éxito

Figura 46: Visualización de *tokens* adicionales.

Fuente: Elaboración propia.

4.6 Configuration

La configuración adecuada de la aplicación y sus componentes es esencial para garantizar un entorno seguro y resiliente. Implementar controles en esta área asegura que se eliminen configuraciones por defecto inseguras, se deshabiliten características innecesarias y se apliquen políticas que refuercen la seguridad. En el contexto del portal de empleo, una configuración robusta protege contra vulnerabilidades comunes y asegura que la plataforma funcione de manera confiable y segura, reforzando la confianza de los usuarios en el sistema.

Unintended Security Disclosure

4.6.1 Se verifica que los modos de depuración del servidor web o de la aplicación y del marco de la aplicación estén deshabilitados en producción para eliminar características de depuración, consolas de desarrollador y divulgaciones de seguridad no deseadas.

Motivo

Deshabilitar las características de depuración en producción elimina la posibilidad de divulgaciones accidentales que puedan exponer vulnerabilidades o información confidencial. Esto incluye consolas de desarrollador, mensajes detallados de error y herramientas que podrían ser manipuladas para acceder a datos protegidos o realizar acciones no autorizadas.

Ejecución

No se ha lanzado a producción por lo que no es necesario realizar estas configuraciones.

4.6.2 Se verifica que los encabezados *HTTP* o cualquier parte de la respuesta *HTTP* no expongan información detallada de las versiones de los componentes del sistema.

Motivo

Evitar que los encabezados *HTTP* o cualquier parte de la respuesta *HTTP* expongan información detallada sobre las versiones de los componentes del sistema es crucial para reducir el riesgo de explotación de vulnerabilidades conocidas. Cuando un servidor web,

framework o biblioteca divulga su versión exacta en las respuestas *HTTP*, un atacante puede utilizar esta información para identificar vulnerabilidades específicas asociadas con esa versión y lanzar ataques dirigidos.

Ejecución

Información de la prueba			
Historia de usuario	Revisión encabezados	Prerrequisitos	Haber iniciado sesión
Dependencias	Tener cuenta creada		
Resumen resultados			
- Se realiza una petición simple para analizar encabezados			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo ?
cURL HTTP://localhost:5173/trabajos?menu=student	Encabezados genericos	Encabezados genericos	Éxito

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: text/html
Cache-Control: no-cache
Etag: W/"2bd-cE57KYKyemENLafjN24rioigMwk"
Date: Sat, 21 Dec 2024 07:55:59 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

Figura 47: Visualización de encabezados no comprometedores.

Fuente: Elaboración propia.

HTTP Security Headers

4.6.3 Se verifica que cada respuesta *HTTP* contenga un encabezado *Content-Type*. Además, especificar un conjunto de caracteres seguro (por ejemplo, UTF-8, ISO-8859-1) si los tipos de contenido son *text/**, */+xml* y *application/xml*. El contenido debe coincidir con el encabezado *Content-Type* proporcionado.

Motivo

Garantizar que cada respuesta *HTTP* incluya un encabezado *Content-Type* apropiado y que el contenido corresponda con el tipo especificado es fundamental para prevenir ataques como *XSS* y otros problemas derivados de interpretaciones incorrectas del contenido por parte del cliente. Este control asegura que los navegadores y otras aplicaciones que consumen las respuestas *HTTP* procesen el contenido de manera predecible y segura.

Ejecución

Información de la prueba			
Historia de usuario	Peticiones simples	Prerrequisitos	Haber iniciado sesión
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Realización de peticiones simples para ver encabezados			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Realización de peticiones simples a la plataforma	Aparición de encabezados esperados	Aparición de encabezados esperados	Éxito

Request URL:	http://127.0.0.1:8082/auth/login/
Request Method:	POST
Status Code:	200 OK
Remote Address:	127.0.0.1:8082
Referrer Policy:	strict-origin-when-cross-origin
▼ Response Headers ☐ Raw	
Access-Control-Allow-Credentials:	true
Access-Control-Allow-Origin:	http://localhost:5173
Allow:	POST, OPTIONS
Content-Length:	50
Content-Type:	application/json
Cross-Origin-Opener-Policy:	same-origin
Date:	Sat, 21 Dec 2024 08:35:18 GMT
Referrer-Policy:	same-origin
Server:	WSGIServer/0.2 CPython/3.10.11

Figura 48: Visualización content-type.

Fuente: Elaboración propia.

4.6.4 Se verifica que todas las respuestas de la API contengan un encabezado *Content-Disposition: attachment; filename="api.json"* (o el nombre de archivo apropiado para el tipo de contenido).

Motivo

Se debería asegurar que todas las respuestas de la API incluyan un encabezado *Content-Disposition* con la configuración *attachment* y un nombre de archivo apropiado es fundamental para proteger a los usuarios contra ataques como XSS y otros riesgos asociados con la visualización no controlada de datos. Este encabezado fuerza al navegador o cliente a descargar el contenido en lugar de interpretarlo directamente, reduciendo la posibilidad de que contenido malicioso se ejecute en el navegador del usuario.

Ejecución

No existe una funcionalidad que permita al usuario descargar contenido de la plataforma.

4.6.5 Se verifica que se haya implementado un encabezado de respuesta *Content Security Policy (CSP)* que ayude a mitigar el impacto de los ataques XSS, como las vulnerabilidades de inyección de *HTML*, *DOM*, *JSON* y *JavaScript*.

Motivo

Una política *CSP* correctamente configurada puede prevenir la ejecución de scripts no autorizados, limitar las fuentes de datos externas y controlar el comportamiento del contenido activo en el navegador. Por ejemplo, puede restringir el uso de *scripts* en línea (*inline scripts*), permitir solo recursos cargados desde dominios confiables o bloquear elementos de medios provenientes de fuentes desconocidas.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Obtención de cabecera CSP tras iniciar sesión			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de credenciales	Obtención de cabecera CSP	Obtención de cabecera CSP	Éxito

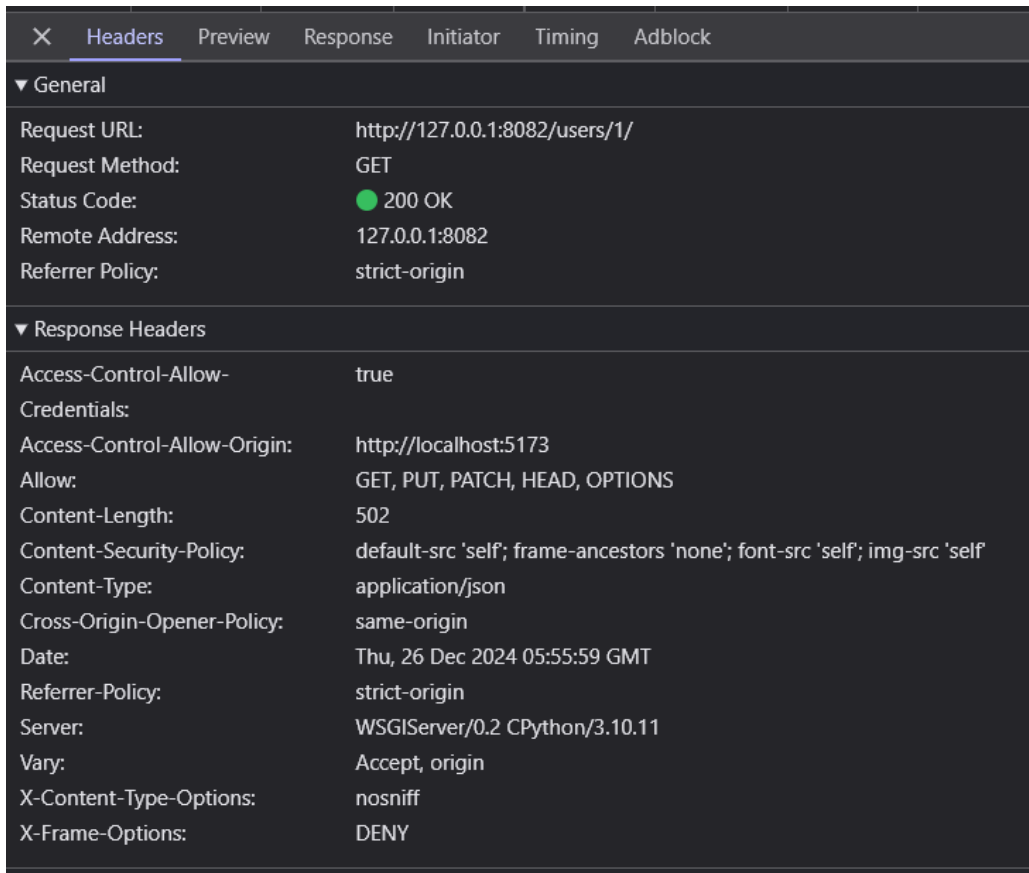


Figura 49: Visualización de cabecera CSP.

Fuente: Elaboración propia.

4.6.6 Se verifica que todas las respuestas contienen un *X-Content-Type-Options: nosniff* header.

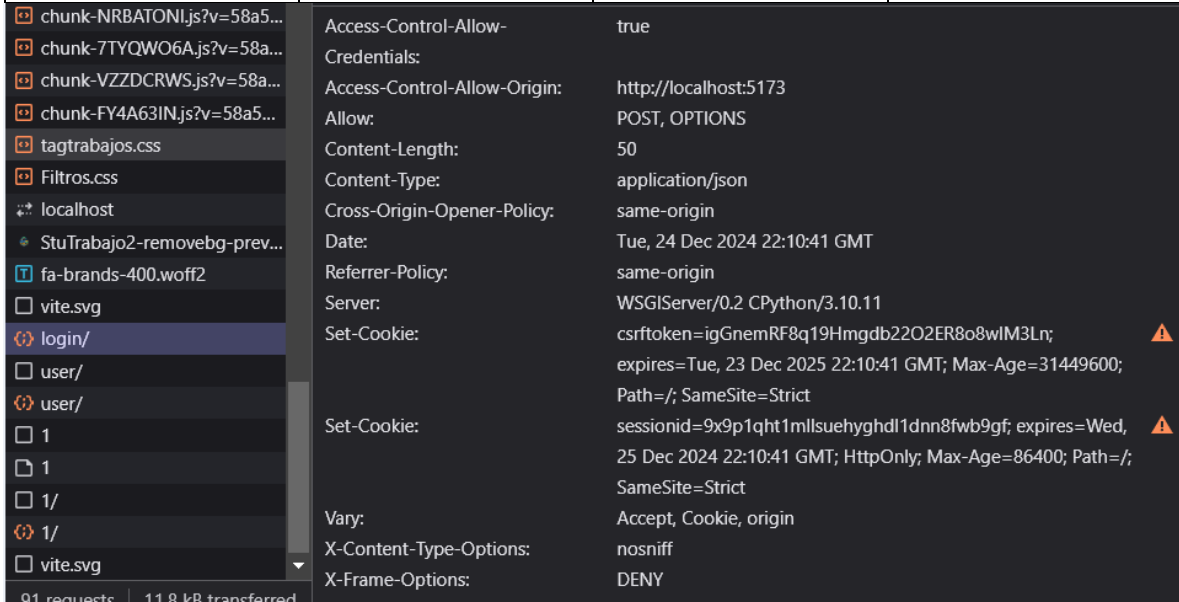
Motivo

Al incluir el encabezado *nosniff*, los navegadores son instruidos para adherirse estrictamente al tipo de contenido definido en el encabezado *Content-Type* de la respuesta, evitando la ejecución de contenido no esperado. Esto es importante para recursos como *scripts*, hojas de estilo o contenido *JSON* que podrían ser explotados si son mal interpretados.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Tener una cuenta creada		
Resumen resultados			

- Obtención de cabecera <i>X-Content-Type-Options: nosniff</i>			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Credenciales de usuario	Cabecera de <i>X-Content-Type-Options: nosniff</i> obtenida	Cabecera de <i>X-Content-Type-Options: nosniff</i> obtenida	Éxito



The screenshot shows the 'login/' endpoint in the browser's developer tools. The response headers are as follows:

- Access-Control-Allow-Credentials: true
- Access-Control-Allow-Origin: http://localhost:5173
- Allow: POST, OPTIONS
- Content-Length: 50
- Content-Type: application/json
- Cross-Origin-Opener-Policy: same-origin
- Date: Tue, 24 Dec 2024 22:10:41 GMT
- Referrer-Policy: same-origin
- Server: WSGIServer/0.2 CPython/3.10.11
- Set-Cookie: csrftoken=igGnemRF8q19Hmgdb22O2ER8o8wIM3Ln; expires=Tue, 23 Dec 2025 22:10:41 GMT; Max-Age=31449600; Path=/; SameSite=Strict
- Set-Cookie: sessionid=9x9p1qht1mllsuehyghdl1dnn8fwb9gf; expires=Wed, 25 Dec 2024 22:10:41 GMT; HttpOnly; Max-Age=86400; Path=/; SameSite=Strict
- Vary: Accept, Cookie, origin
- X-Content-Type-Options: nosniff
- X-Frame-Options: DENY

Figura 50: Visualización de cabecera nosniff.

Fuente: Elaboración propia.

4.6.7 Se verifica que se incluya un encabezado *Strict-Transport-Security* en todas las respuestas y para todos los subdominios, como *Strict-Transport-Security: max-age=15724800; includeSubdomains*.

Motivo

El encabezado *Strict-Transport-Security (HSTS)* es esencial para garantizar que las conexiones entre los clientes y el servidor se realicen exclusivamente a través de *HTTPS*, eliminando la posibilidad de que datos sensibles sean transmitidos por canales inseguros como *HTTP*. Este encabezado instruye a los navegadores para que, durante un periodo especificado (por ejemplo, 15724800 segundos, equivalente a 6 meses), se conecten únicamente mediante *HTTPS*, incluso si el usuario intenta acceder a través de *HTTP*.

Ejecución

No se puede implementar el encabezado *Strict-Transport-Security* en este momento ya que la plataforma no tiene *HTTPS* habilitado.

4.6.8 Se verifica que se incluya un encabezado adecuado *Referrer-Policy* para evitar exponer información sensible en la *URL* a través del encabezado *Referer* a partes no confiables.

Motivo

El encabezado *Referrer-Policy* es fundamental para evitar que información sensible incluida en las *URL*, como *tokens*, identificadores de sesión o parámetros relacionados con datos personales, sea expuesta a partes no confiables a través del encabezado *Referer*. Este encabezado permite a los desarrolladores controlar qué información, si alguna, debe enviarse a sitios externos cuando un usuario navega fuera de la aplicación.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Tener una cuenta creada		
Resumen resultados			
- Un ingreso de sesión observándose la cabecera <i>referer</i>			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Credenciales de usuario	Cabecera <i>referer</i>	Cabecera <i>referer</i>	Éxito

localhost

tagtrabajos.css

Filtros.css

chunk-FY4A63IN.js?v=58a57c9d

chunk-VZZDCRWS.js?v=58a57c9d

StuTrabajo2-removebg-preview.png

fa-brands-400.woff2

vite.svg

login/

login/

user/

user/

1

1/

1/

1

vite.svg

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br, zstd

Accept-Language: es-ES,es

Authorization: Token 1d874e63db3706cfea767c4c8121348dbf50939d

Connection: keep-alive

Content-Length: 69

Content-Type: application/json;charset=UTF-8

Host: 127.0.0.1:8082

Origin: http://localhost:5173

Referer: http://localhost:5173/

Sec-Ch-UA: "Brave";v="131", "Chromium";v="131", "Not_A_Brand";v="24"

Sec-Ch-UA-Mobile: ?0

Sec-Ch-UA-Platform: "Windows"

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

Sec-Fetch-Site: cross-site

Sec-Gpc: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36

92 requests | 11.4 kB transferred | 12.7 MB resources | Finish: 6.1

Figura 51: Visualización de cabecera *referer*.

Fuente: Elaboración propia.

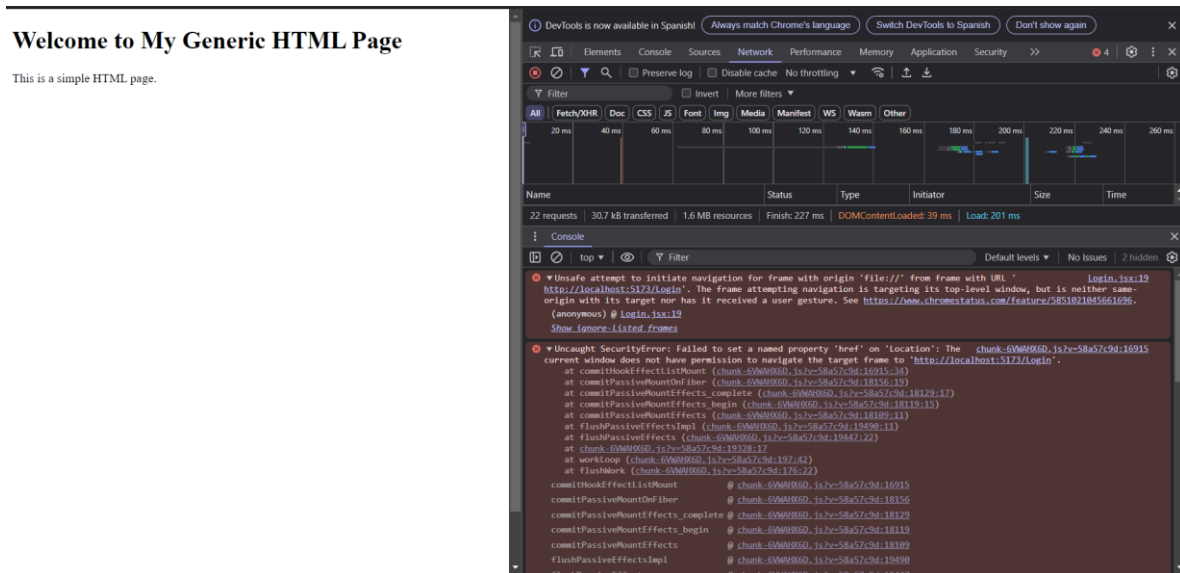
4.6.9 Se verifica que el contenido de una aplicación web no pueda ser incrustado en un sitio de terceros de forma predeterminada y que la incrustación de los recursos exactos solo se permita cuando sea necesario, utilizando los encabezados de respuesta adecuados Content-Security-Policy: frame-ancestor y X-Frame-Options.

Motivo

Evitar que el contenido de una aplicación web sea incrustado en sitios de terceros de forma predeterminada es esencial para protegerla contra ataques como *Clickjacking*, en los que un atacante incrusta la aplicación en un marco invisible dentro de su propio sitio, engañando a los usuarios para que realicen acciones no deseadas. Este tipo de ataque puede comprometer la seguridad de los datos y las interacciones dentro de la plataforma.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Haber creado una cuenta		
Resumen resultados			
- Intento de inicio de sesión a través de un <i>iframe</i> ajeno			
Detalle resultados			
Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de credenciales	Bloqueo de <i>cors</i> por parte de <i>Django</i>	Bloqueo de <i>cors</i> por parte de <i>Django</i>	Éxito

Figura 52: Visualización error *iframe*.

Fuente: Elaboración propia.

HTTP Request Header Validation

4.6.10 Se verifica que el servidor de la aplicación solo acepte los métodos *HTTP* utilizados por la aplicación/API, incluyendo *pre-flight OPTIONS*, y *logs/alerts* sobre cualquier solicitud que no sea válida para el contexto de la aplicación.

Motivo

Restringir los métodos *HTTP* aceptados por el servidor a aquellos utilizados específicamente por la aplicación o *API* es esencial para reducir la superficie de ataque y prevenir el mal uso de solicitudes no autorizadas. Este control garantiza que únicamente los métodos necesarios, como *GET*, *POST*, *PUT* o *DELETE*, estén habilitados, mientras que otros métodos innecesarios o potencialmente peligrosos, como *TRACE* o *CONNECT*, sean rechazados.

Ejecución

Información de la prueba			
Historia de usuario	Petición de métodos incorrectos	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Prueba fallida de uso de métodos no habilitados			
Detalle resultados			

Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ninguna	Estatus 405 para métodos no permitidos	Estatus 405 para métodos no permitidos	Éxito

```

py test Userservice.tests.HTTPMethodsTest
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
405 status encontrado para patch
405 status encontrado para trace
200 status encontrado para options
F
=====
FAIL: test_disallowed_methods (Userservice.tests.HTTPMethodsTest)
Test that disallowed methods return 405 Method Not Allowed
=====

```

Figura 53: Prueba metodos no permitidos.

Fuente: Elaboración propia.

4.6.11 Se verifica que el encabezado *Origin* proporcionado no se utilice para la autenticación o decisiones de control de acceso, ya que el encabezado *Origin* puede ser fácilmente modificado por un atacante.

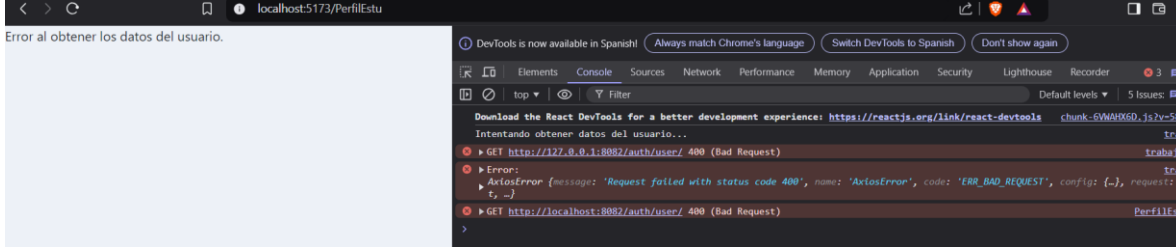
Motivo

La autenticación y el control de acceso deben basarse en mecanismos más seguros, como *tokens*, sesiones autenticadas o validaciones estrictas de credenciales. Utilizar el encabezado *Origin* de forma inapropiada introduce un riesgo significativo de *bypass* de controles de seguridad, comprometiendo la integridad del sistema y la confidencialidad de los datos.

Ejecución

Información de la prueba			
Historia de usuario	Inicio de sesión	Prerrequisitos	Ninguno
Dependencias	Ninguno		
Resumen resultados			
- Intento de acceso a página de empleos sin <i>token</i> de autenticación			
Detalle resultados			

Entradas de usuario	Resultado esperado	Resultado actual	Éxito/fallo?
Ingreso de parámetro "PerfilEstu"	Error de obtención de credenciales de usuario	Error de obtención de credenciales de usuario	Éxito


Figura 54: Visualización encabezado *Origin*.

Fuente: Elaboración propia.

4.6.12 Se verifica que el encabezado *Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin* utilice una lista de permitidos estricta de dominios y subdominios de confianza para hacer la comparación y que no soporte el origen "null".

Motivo

El encabezado *Access-Control-Allow-Origin*, parte de la política de *CORS*, es fundamental para controlar qué dominios pueden interactuar con los recursos de una aplicación web a través de solicitudes *cross-origin*. Este control garantiza que solo una lista estricta de dominios y *subdominios* confiables esté permitida para acceder a la *API* o recursos del sistema, evitando que orígenes no autorizados o maliciosos realicen solicitudes que puedan comprometer la seguridad del sistema.

Permitir orígenes amplios, como el uso de un comodín (*) o aceptar el origen *null*, introduce un riesgo significativo, ya que habilita cualquier dominio o incluso contextos sin origen definido (como archivos locales) para interactuar con la *API*. Esto puede ser explotado por atacantes para realizar solicitudes *cross-origin* no autorizadas, comprometiendo datos sensibles o manipulando recursos del sistema.

Ejecución

Como no se cuenta con una lista de dominios en funcionamiento no se puede implementar esta verificación.

CAPÍTULO 5: CONCLUSIONES

5.1 Aprendizaje

La realización de este proyecto no solo permitió abordar un problema relevante como la deserción estudiantil, sino que también fue una oportunidad para aplicar estándares de ciberseguridad en un contexto práctico, lo cual fortaleció mi conocimiento en este campo. Durante el desarrollo, aprendí herramientas modernas como *React* y servicios de *AWS* junto con *GCP*, y a seleccionar e implementar estándares de seguridad relevantes en base a las necesidades específicas de la plataforma, como los propuestos por *OWASP CheatSheet*, *OWASP ASVS* y la *ISO/IEC 27002*.

Además, comprendí la importancia de ajustar los estándares y herramientas a las necesidades específicas de la plataforma, priorizando aquellas que realmente aportaran valor. Esto me permitió desarrollar un enfoque más crítico hacia la toma de decisiones técnicas y de seguridad.

5.2 Recomendaciones

Priorizar la seguridad desde el inicio: Incluir estándares de ciberseguridad desde las fases iniciales del proyecto permite mitigar riesgos desde el diseño, en lugar de aplicar soluciones correctivas más costosas en términos de tiempo.

Seleccionar herramientas adaptadas al proyecto: No todas las herramientas o estándares son necesarias; la clave está en elegir aquellas que mejor se alineen con los objetivos y limitaciones del proyecto debido a los enfoques que tienen los estándares.

Documentar el proceso: Mantener registros detallados de las decisiones y las implementaciones facilita la transferencia de conocimiento.

5.3 Valor agregado a la organización

La implementación de este proyecto y la memoria asociada proporcionaron una base sólida para que la organización pueda ofrecer un portal de empleo confiable, accesible y seguro. Las principales mejoras incluyen:

- **Incremento en la confianza del usuario:** La aplicación de estándares de seguridad ha garantizado la protección de la información sensible, brindando tranquilidad tanto a los estudiantes como a las organizaciones que interactúan en la plataforma.
- **Mejora en la gestión de proyectos:** La integración de metodologías ágiles ha optimizado los procesos de desarrollo, estableciendo una dinámica de trabajo más eficiente y organizada.

5.4 Reflexión

La metodología Scrum fue un pilar clave para organizar el trabajo en *sprints* manejables, pero también requirió un compromiso constante para coordinar tareas y priorizar funcionalidades. Entre las herramientas utilizadas, *React* y *Vite* demostraron ser valiosas para el desarrollo rápido del *Frontend*, mientras que los experimentos en *AWS* y *GCP* simplificaron la gestión de recursos en la nube con sus respectivos usos.

Sin embargo, enfrenté desafíos como la necesidad de adaptar estándares de seguridad a un proyecto con limitaciones de tiempo y alcance. Esto me enseñó a identificar lo esencial y a optimizar las decisiones técnicas sin comprometer la calidad del sistema.

5.6 Conclusión

En resumen, esta memoria no solo permitió desarrollar un portal de empleo funcional y seguro, sino que también dejó un legado de conocimiento para la organización, mejorando sus procesos y brindando un modelo para futuros proyectos. Este trabajo representa un paso significativo hacia la creación de soluciones tecnológicas accesibles y confiables, demostrando que la integración de estándares de seguridad y buenas prácticas técnicas puede generar un impacto positivo tanto en el proyecto como en los usuarios que confían en él.

REFERENCIAS BIBLIOGRÁFICAS

Foundation, O. (s.f). *OWASP Cheat Sheet Series*. Obtenido de
[HTTPS://cheatsheetseries.OWASP.org/cheatsheets/Authentication_Cheat_Sheet.HTML#
mplement-proper-password-strength-controls](HTTPS://cheatsheetseries.OWASP.org/cheatsheets/Authentication_Cheat_Sheet.HTML#implement-proper-password-strength-controls)

Foundation, T. O. (Octubre de 2021). *Application Security Verification Standard 4.0.3*. Obtenido de
<HTTPS://OWASP.org/www-project-application-security-verification-standard/>

Foundation, T. O. (2021). *Application Security Verification Standard 4.0.3*. OWASP Foundation.

ISO/IEC. (2022). *SO/IEC 27002:2022 Information security, cybersecurity and privacy protection — Information security controls*. Geneva: ISO/IEC.