UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE PLATAFORMA PARA GESTIÓN DE RECLUTAMIENTO Y SELECCIÓN DE POSTULANTES PARA EMPRESA DE PRESTACIÓN DE SERVICIOS DE CUIDADOS MÉDICOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

CRISTÓBAL ANTONIO DOMÍNGUEZ RIVERA

PROFESOR GUÍA:
ADOLFO CARRASCO ACOSTA

MIEMBROS DE LA COMISIÓN:
NANCY HITSCHFELD KAHLER
SANDRA DE LA FUENTE GONZÁLEZ

SANTIAGO DE CHILE
2022

Resumen

MyraSalud, es una empresa dedicada al suministro de personal de salud para cuidados, principalmente en dependencias externas. Con esto, requieren de la gestión del personal de manera rápida, con tal de cubrir rápidamente los requerimientos, y a su vez, evitar mayores perjuicios.

De esta forma, su unidad de reclutamiento y selección de personal, se encarga de elegir al personal idóneo para una oportunidad de trabajo determinada, asegurando por un lado a un buen profesional o técnico, y por el otro, un costo razonable.

Así, MyraSalud dispone de un vasto registro de personas, ya sean profesionales o técnicos, con lo cual debe suplir toda clase de requerimientos. Pero, este registro es difícil de manejar y no funciona de manera óptima, a la vez que tiene bastantes oportunidades de mejora.

De esta manera, se estudia la unidad de reclutamiento y selección, y el proceso propiamente tal, con tal de determinar una propuesta de solución, a la vez que se establece una relación con el cliente para favorecer el trabajo.

Con esto, se propone el desarrollo de una plataforma `WEB`, para estructurar este proceso, con lo cual se recurre al framework `Django`, para el desarrollo de esta, a la vez que se propone un esquema de desarrollo, que considera reuniones frecuentes con el cliente.

En particular, se lleva a cabo la construcción de una aplicación con usuarios registrables mediante cuentas de usuario, verificación y acceso homologable con servicios externos como, redes sociales, etc. A su vez, los usuarios registrados disponen de interfaces y formularios para registrar postulantes de trabajo y oportunidades laborales, y su posterior asignación, de acuerdo a la implementación de un algoritmo de búsqueda y priorización.

Durante el desarrollo se dan ciertos contratiempos, desafíos y aprendizajes, en los cuales se experimenta con distintas tecnologías para mejorar el despliegue de la aplicación como para mejorar el desarrollo en `Django`.

Finalmente, se logra construir un aplicación, de despliegue simplificado, pero no se logra poner en producción, aunque de todas maneras, se valida en gran manera, generando un producto extensible.

A mis padres y a mi hermano, que me han apoyado desde el inicio.

Agradecimientos

Agradezco, en primer lugar a mi familia, a mis padres que siempre me han apoyado y a mi hermano que desde que puedo recordar, ha estado ahí, como compañía, como inspiración y como colaborador muchas veces en lo que me he propuesto.

Agradezco también a mis amigos que me han acompañado por todas estas etapas, y que me han ayudado particularmente en esta parte de la vida.

Agradezco igualmente a mi profesor guía, que siempre estuvo dispuesto, en lo académico, como en lo particular.

Agradezco a su vez, al equipo de MyraSalud, que me recibió gratamente, y que las veces que nos reunimos prestaron absoluta disposición y contribuyeron notablemente.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

MyraSalud es una empresa chilena dedicada a ofrecer servicios de cuidado de pacientes a domicilio o en instalaciones propias. El cuidado ofrecido a cada usuario generalmente requiere de la atención de profesionales de la salud, muchas veces con atención las 24 horas del día o, al menos, en una frecuencia considerable, por lo que es fundamental asegurar a cada persona la disponibilidad del personal de salud requerido. Por otro lado, en distintas ocasiones, se requiere de atenciones de urgencia, con lo cual se hace necesario generar el contacto con un profesional, con la mayor brevedad posible.

Con esto, se acarrean bastantes problemas, ya que la disponibilidad está sujeta a múltiples factores, ya sean personales por parte de cada profesional, aptitudes requeridas que no se encuentren disponibles o motivos externos que generen una menor disponibilidad del perfil requerido. Con esto se pone en riesgo la salud y oportunidad de los cuidados a los pacientes que confían en la organización, a la vez que se compromete la reputación de esta. Cabe señalar que también puede existir un perjuicio económico para la organización, debido a multas o recargos en el caso de clientes con contratos particulares.

Para ayudar a mitigar eventuales situaciones problemáticas, se requiere de mantener y gestionar de manera ágil la información de cada profesional, con tal de mantener un registro robusto de profesionales disponibles para atender cada una de las variadas necesidades y poder asignar más fácil y rápidamente un profesional disponible hacia un nuevo paciente, sobretodo en casos cuando surgen eventualidades, y por ende, se requiere del personal inmediatamente.

Considerando los tipos de indisponibilidades que pueda presentar cada profesional, es necesario contar con información detallada y actualizada, con tal de asignar el profesional indicado al paciente y servicio requerido, tomando en cuenta aptitudes técnicas, psicológicas, etc., con el fin de minimizar problemas técnicos y/o sociales al momento de que el profesional concurra a la atención.

1.1. Justificación de la memoria

La organización, en su función de suministrar profesionales a las distintas demandas y servicios, depende de sus propios registros de manera tal que, de acuerdo a una selección de aptitudes y características propias de un servicio a atender, se asigna al profesional que mejor cumpla y se adapte a tales requerimientos, además de velar por la continuidad de la persona asignada a con cierto paciente.

Por otro lado, para tener mayores posibilidades de poder cubrir los requisitos, se deben ampliar de manera constante los registros, con tal de poder reunir a la mayor cantidad de profesionales disponibles, consiguiendo así, a los mejores profesionales para una mayor cantidad de requerimientos.

Lo anterior supone dos tareas no triviales, dado que en primer lugar se debe poder seleccionar de manera rápida y eficiente a un profesional de acuerdo a un conjunto de características, mientras que por otro lado, se debe registrar a la mayor cantidad de profesionales, de manera consistente.

Es así como surge la idea de llevar a cabo un proyecto de desarrollo de una plataforma de reclutamiento y selección de personal, dado que, en primer lugar se requiere de una plataforma con la que los usuarios finales puedan interactuar, por otro lado, un sistema para la gestión de la información robusto, y capaz de resolver las problemáticas dadas, considerando los intereses y procesos preestablecidos por la empresa, y también, la consideración de los datos y la información valiosa, previamente recabada por el área de reclutamiento, y que debe aprovecharse en este nuevo desarrollo.

1.2. Objetivos

1.2.1. Objetivo general

En esta memoria, se va a desarrollar un sistema informático que supla las necesidades requeridas en el área de reclutamiento y selección de personal, facilitando y agilizando el trabajo que actualmente se realiza.

Para cumplir con el objetivo se requiere de un estudio de la unidad de reclutamiento y selección, de acuerdo a su dinámicas, flujos de trabajo y procesos, además de consultar la opinión del personal encargado, incluyendo a las futuras usuarias.

Por otro lado, es necesario realizar un análisis acabado de las tecnologías de desarrollo e implementación de software para desarrollar un producto de calidad, robusto y que se adapte a las condiciones de la organización y los usuarios finales, de tal manera que sea un aporte para la gestión de la empresa. A la vez, se estudiarán distintas alternativas, con tal de verificar la factibilidad y facilidad de implementación que presenten.

Además, es necesario el respaldo y contacto con el cliente en la medida que se avanza con el

desarrollo, por lo que surge la necesidad de llevar a cabo un desarrollo mediante metodologías ágiles, con lo cual se contará con la constante supervisión y evaluación del cliente, que es parte fundamental en el desarrollo de esta plataforma.

Por último, la revisión y validación por parte del estudiante memorista y el cliente, es necesaria para garantizar la cobertura de las necesidades y requerimientos del proyecto.

1.2.2. Objetivos Específicos

1. Identificar los requerimientos de la aplicación.
2. Desarrollar una plataforma `WEB` para la sección y asignación de postulantes.
3. Validar la plataforma `WEB` con distintos tipos de usuarios.

1.3. Metodología

Para el desarrollo del proyecto, se procederá siguiendo una serie de pasos como se muestra a continuación.

1. Desarrollar un estudio de la unidad de reclutamiento y selección para comprender a cabalidad sus necesidades y requerimientos.
2. Analizar la capacidad e infraestructura técnica disponible en la unidad.
3. Estudiar el estado del arte en el desarrollo de plataformas similares.
4. Comparar alternativas existentes ya construidas.
5. Indagar plataformas de hosting en la nube.
6. Generar interfaz, para estudiar usabilidad.
7. Añadir módulo de manejo de documentos
8. Desarrollar el proyecto utilizando metodologías ágiles.
9. Migrar la información actual que posee el área de reclutamiento y selección al nuevo desarrollo.
10. Capacitar al personal para la utilización de la plataforma.
11. Constatar el funcionamiento y utilidad de la plataforma con los usuarios finales y cliente.

1.4. Contenido de la memoria

En el capítulo 2, titulado "Descripción del problema", se abordan los aspectos generales del área de reclutamiento y selección en MyraSalud. A su vez se describen las entidades fundamentales involucradas en el posterior diseño del sistema.

En el siguiente capítulo llamado "Análisis y diseño de la solución", se describe el estudio realizado para la definir los lineamientos del desarrollo de la plataforma, desde el tipo de plataforma a desarrollar, la infraestructura del despliegue y los aspecto fundamentales tomados en consideración para los requerimientos.

En el capítulo 4, "Implementación de la solución", se expone el desarrollo llevado a cabo, a la vez que se describen los aspectos generales, tomados en cuenta para abordar los requerimientos.

En el capítulo 5, "Validación de la solución", se muestran algunos casos de utilización de la plataforma, con una vista general de las interfaces implementadas.

Finalmente, en la Conclusión se indican las generalidades del trabajo realizado, los aspectos aprendidos, y los posibles desarrollos futuros.

Capítulo 2

Descripción del problema

MyraSalud, como empresa que ofrece servicios de cuidado de pacientes, depende de su área de reclutamiento y selección para elegir a los profesionales que llevarán a cabo estos servicios. El área correspondiente está a cargo de dos personas, quienes publican ofertas laborales en sitios diversos, para luego recopilar los datos de los profesionales y técnicos que acceden a ellas.

La información de cada candidato es registrada manualmente en una hoja de cálculo, donde se registra información que va desde los datos personales como también datos correspondientes a preguntas de índole psicológico-laboral estandarizadas. A su vez, dado el contexto actual, se registra información respectiva a la pandemia de COVID 19.

De esta forma, al registrar a cada postulante de una oportunidad laboral, se consideran sus datos no solo para aquella oportunidad, sino que también para futuras oportunidades, de forma tal que si sus cualidades son compatibles con un futuro requerimiento, entonces sea un candidato a ser contactado por parte del equipo de selección.

Con esto, el registro en la hoja de cálculo se puebla con datos e información en extremo importante, ya que de esta información depende la ventaja competitiva de la organización, debido a que al contar con mas datos de profesionales calificados, mas fácilmente se encuentra al profesional adecuado para cada requerimiento.

El problema general de este esquema es que por un lado, la hoja de cálculo tiene bastantes desventajas como la incapacidad de resguardar los datos de forma segura, la vulnerabilidad frente a la perdida de información, o la imposibilidad de mantener la consistencia de los datos.

A su vez, el sistema actual basado en la planilla no considera el hecho de que una vez se comienza con la recopilación de los datos del postulante, a su vez se debe considerar el guardado de documentos, debido a que cada postulante debe presentar documentación que va desde a la identidad, hasta la acreditación de sus habilidades, lo cual el sistema actual no puede resolver, con lo cual se debe almacenar la información manualmente.

2.1. Área de reclutamiento y selección

El área de reclutamiento y selección, compuesta por dos encargadas, fue estudiada, de tal manera que, en un principio, se determinaron dos entidades fundamentales en su gestión, estas son la **oportunidad laboral**, y el **postulante**. Estas dos entidades, son las que interactúan entre sí y por ende se analizaron a fondo para poder determinar sus ciclos completos desde que se crean hasta que expiran, si ese fuese el caso. A continuación, se definirán cada una de estas entidades:

2.2. Oportunidad Laboral

La oportunidad laboral representa a un requerimiento de un servicio médico, el cual debe ser atendido por uno o más profesionales. Cada oportunidad laboral se define por una serie de parámetros, entre ellos:

1. Fecha creación de oportunidad
2. Cantidad requerida (de postulantes)
3. Ciudad o comuna
4. Turno
5. Tipo de contrato
6. Sueldo líquido
7. Quién lo solicita

Las oportunidades laborales a su vez se clasifican en dos tipos: SOS y FULL, es decir, oportunidades que se requieren de manera urgente, y las que se ejercen de manera extendida.

A su vez, de la entidad de oportunidad laboral, se puede identificar un ciclo de vida, donde en primer lugar, se establecen los requerimientos, luego se busca el personal, para luego terminar el ciclo con las eventuales contrataciones. El proceso se representa en el siguiente diagrama 2.1:

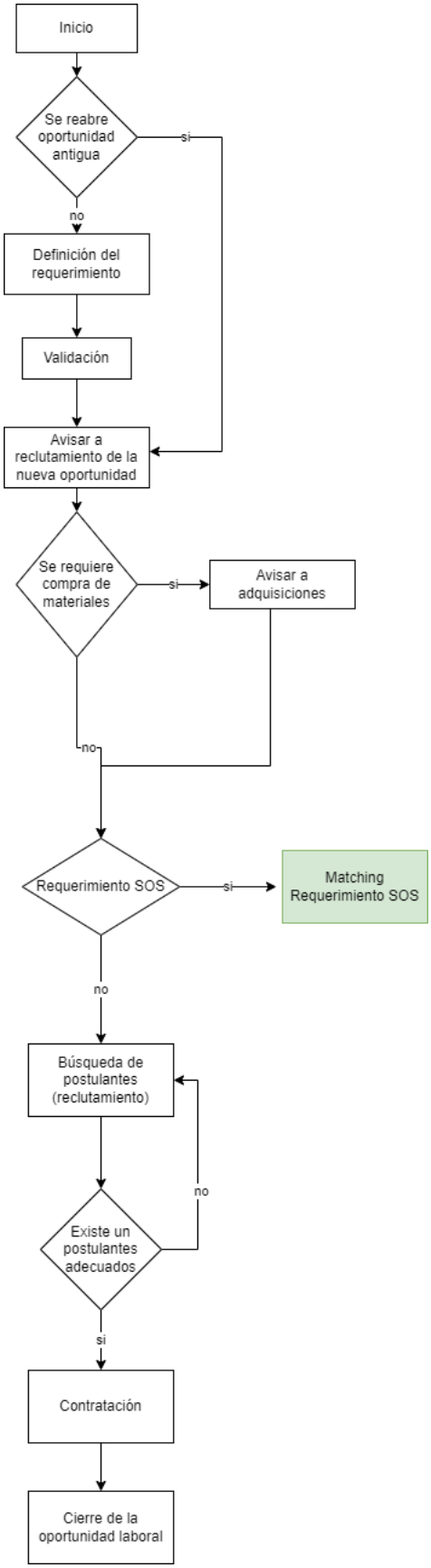

Figura 2.1: Diagrama del ciclo de vida de la oportunidad laboral.

2.3. Postulante

La siguiente entidad relevante es el postulante, es decir, el personal de salud que dada una oportunidad laboral, pretende tomar el puesto disponible. Los principales atributos que definen a un postulante son:

1. Teléfono
2. Correo electrónico
3. RUN
4. Dirección
5. Habilidades técnicas
6. Preguntas psico-laborales

Respecto a las preguntas psico-laborales, estas pueden variar con el tiempo, con lo cual, se requiere que se puedan añadir o eliminar preguntas del formulario.

Por otro lado, la entidad postulante tiene un ciclo de vida, que a diferencia de la oportunidad laboral, tiende a ser de tiempo de expiración indefinido (Figura 2.2):

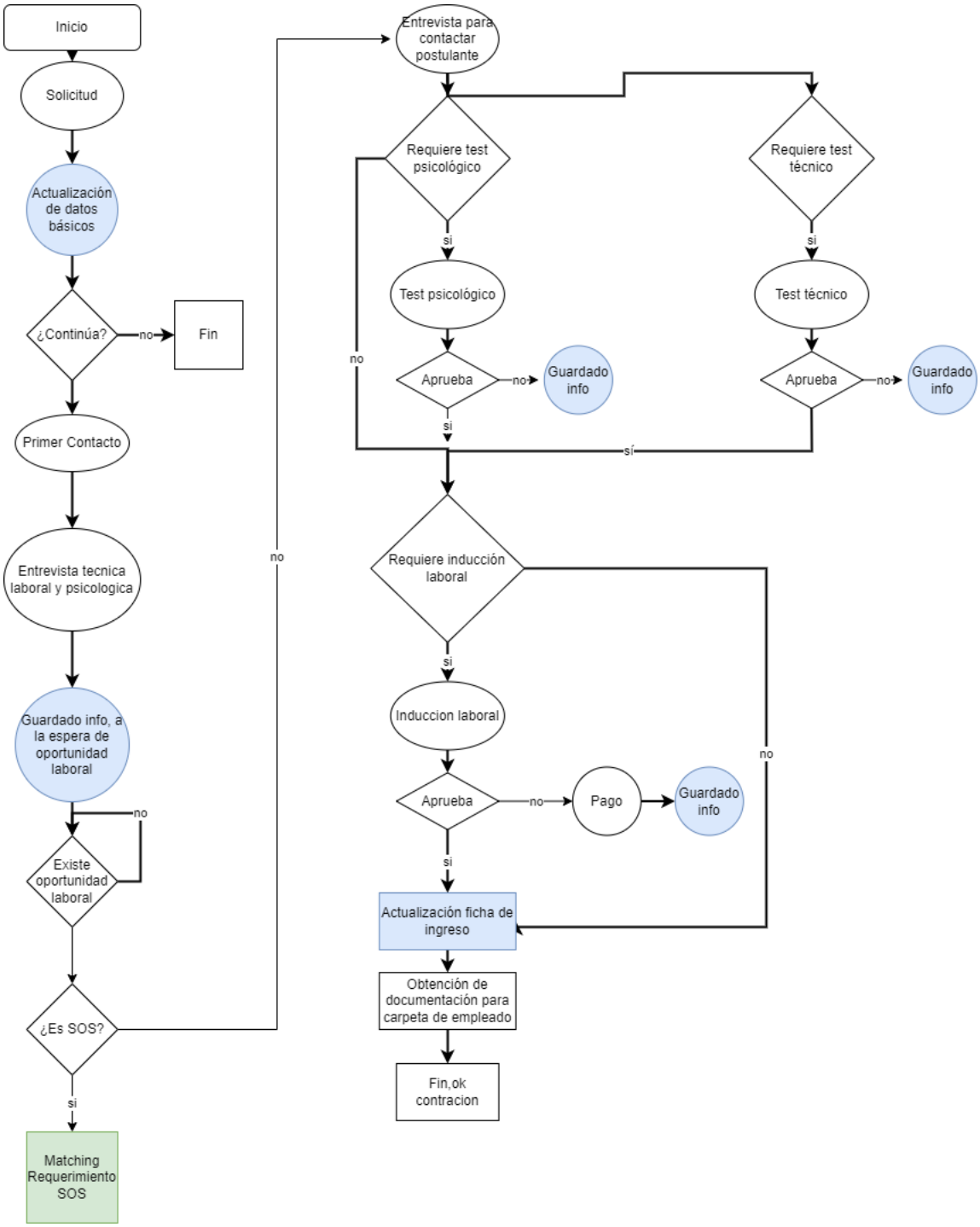

Figura 2.2: Diagrama del ciclo de vida del postulante.

2.4. Interacción Oportunidad-Postulante

Para cada oportunidad, se busca entre los postulantes, quienes tengan las cualidades técnicas adecuadas, dentro de un conjunto definido de cualidades, por ejemplo:

1. Traqueotomía
2. Gastrostomía
3. Sondeo vesical
4. Inyecciones intramusculares
5. Sonda nasogástrica
6. Manejo de parapléjicos/postrados
7. Uso de tecle

Con lo cual, si dado el caso, una oportunidad requiere de un profesional con las habilidades de traqueotomía y gastrostomía, la plataforma deberá facilitar la búsqueda entregando profesionales que cumplan esas habilidades.

A su vez, las oportunidades laborales, al ser clasificadas ente `FULL TIME` o `SOS`, requieren de un trato distinto. El diagrama 2.3 gráfica en términos generales la interacción con oportunidades SOS.

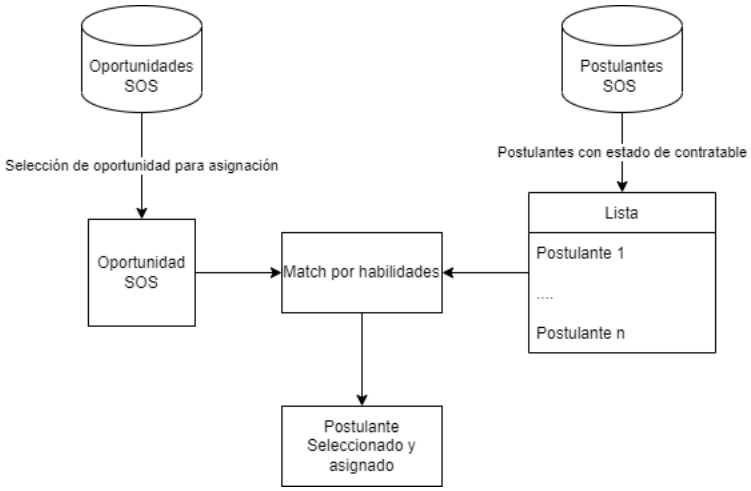

Figura 2.3: Matching oportunidades SOS

Prioridad a postulantes

A su vez, dentro del proceso de contacto con postulantes, en particular en las oportunidades `SOS`, la organización detectó que muchas veces, dado que los postulantes conocen el proceso de selección y los salarios que en teoría podrían llegar a pagarse por un servicio,

evitan contestar los llamados de forma afirmativa y esperan que los salarios de las ofertas empiecen a incrementarse.

Con lo cual, debido a la urgencia del servicio, se termina llegando siempre al limite superior del sueldo preestablecido, por tanto, sumado a las presiones sobre el personal de reclutamiento, se termina generando un perjuicio tanto para la empresa como para el cliente final, que no podrá acceder a un servicio urgente a un precio razonable.

De esta forma, se plantea al memorista alguna forma de resolver o mitigar parcialmente el problema.

Capítulo 3

Análisis y Diseño de la solución

En el siguiente capítulo se detalla la forma y los procedimientos con los cuales fue concebida la solución al problema de la unidad de reclutamiento y selección de MyraSalud, pasando por decisiones técnicas de implementación, hasta los asuntos propios de la metodología de trabajo de la unidad en la organización.

3.1. Tipo de Plataforma

La unidad de reclutamiento y selección, en un principio, basa su registro de postulantes y oportunidades laborales en una hoja de cálculo de `Google`[1]. Esta solución, fue un avance significativo en la gestión de las postulaciones, ya que si bien una planilla como tal puede presentar evidentes deficiencias a la hora de almacenar y manejar información, el hecho de migrar datos físicos a una planilla que incluso permite el uso concurrente, representó una mejora y un avance notable en las labores de gestión.

De esta manera, en primer lugar se justifica el uso de herramientas informáticas para solucionar el problema, ya que, por un lado, han representado una mejora evidente, aunque de todas maneras, con algunas deficiencias propias de una hoja de cálculo, y para lo cual se puede desarrollar una solución más enfocada y que aborde de mejor manera el problema de la selección de postulantes.

Con esto, surge la idea de implementar una plataforma `WEB`, la cual presenta grandes ventajas frente a la hoja de cálculo, dentro de las cuales se puede señalar:

1. Acceso: Una plataforma `WEB` permite el acceso por distintos usuarios de la organización, de manera simultánea, desde distintos terminales, con lo cual, por un lado, se replica la concurrencia de la hoja de cálculo, además de que se evitan las inconsistencias, ya que, a priori, las celdas de una hoja de cálculo no previenen las modificaciones accidentales frente a un esquema que procure que los datos mantengan un sentido lógico, de esta forma, los datos de la organización no se verían en riesgo.

[1] `https://www.google.com/intl/es/sheets/about/`

2. Gestión de datos: Generalmente, las plataformas web constan de una base de datos, para almacenar la información de cada una de las entidades propias de la actividad relacionada a la plataforma, con lo cual la información es guardada y manejada de forma segura, manteniendo la consistencia. De esta forma, la organización adquiere un mayor control y seguridad de sus datos.

3. Seguridad: El acceso de los usuarios es restringido, con lo cual se evita el acceso indeseado de terceros que podrían tener intenciones maliciosas, como el robo de datos valiosos de técnicos y pŕofesionales que ninguna otra organización posee, y con esto, comprometer la propuesta de valor de la empresa. Además, esto implica la implementación de un sistema de usuarios que permitiría el control y manejo de las atribuciones de los usuarios, a nivel de lógica del negocio.

4. Interfaz: La plataforma WEB permite la implementación de interfaces las cuales son una ventaja *per se*, ya que permiten un despliegue de la información más amigable para los usuarios de la organización, a la vez que presentan las funcionalidades de manera acotada. Además, el uso y la capacitación de un nuevo usuario se facilita ya que, en lugar de la hoja de cálculo, no se requiere de la intervención directa de los datos, sino que los registros y vistas se llevan a cabo por medio de formularios, tablas, etc., con lo cual la adaptación para un usuario principiante se facilita.

De esta manera, la implementación de una plataforma `WEB` para la solución de la plataforma de reclutamiento y selección, en conformidad con el cliente y todo lo descrito anteriormente, se indica como la alternativa a proceder, de forma tal que se puedan registrar postulantes con su información respectiva, agregar oportunidades laborales, para luego asignar a los postulantes con su respectiva oportunidad laboral, una vez sean seleccionados y contratados.

3.2. Alojamiento de la plataforma

Teniendo en cuenta la decisión de implementar una plataforma `WEB`, se hace necesario pensar en cómo y dónde se va a alojar la plataforma, es decir, el `hosting`. Para esto, se hace imperativa la indagación de distintos servicios de `hosting`, en particular, servicios de `cloud computing`, dado que el cliente señala que un `hosting` local dentro de las instalaciones de la empresa no sería conveniente, tomando en cuenta que han ocurrido robos a las dependencias físicas de la empresa.

Dentro de los servicios analizados para esto, se cuentan[2]:

1. Amazon web services (AWS)
2. Google Cloud
3. Microsoft Azure

[2] `https://www.avenga.com/magazine/top-cloud-service-providers/`

4. Oracle Cloud

Para comparar estos servicios, los factores de interés corresponden a la existencia de una instancia gratuita, el tiempo de esta, y los costos de los servicios en el caso de que se requiera aumentar la capacidad. Con esto se obtuvo:

	AWS	Google Cloud	Azure	Oracle Cloud
Instancia gratuita	1 año	90 días	30 días	Ilimitado
Descripción instancia	1 GB RAM	32GB almacenamiento 200 dólares en servicios	300 dólares en servicios 24 GB RAM	32GB almacenamiento

Tabla 3.1: Comparación servicios de `Cloud Computing`

Dado que se requiere de los servicios a largo plazo, tanto como para el desarrollo, las alternativas que más tiempo de prueba ofrecen son: `AWS` y `Oracle Cloud`

Con esto, para realizar pruebas y desarrollar la plataforma se utilizará `AWS`, ya que, si bien ofrece un hardware bastante limitado, brinda un año de servicio gratuito, y en el caso de requerir más potencia y/o capacidad, los planes mensuales comienzan en 7 dólares aproximadamente[3] mientras que en `Oracle` se disponen de planes desde 47 dólares [4], es así como `AWS` es la opción más conveniente, respecto a las restricciones presupuestarias sugeridas por el cliente. De todas formas, dependiendo de las pruebas a realizar en la etapa de desarrollo, si se requiere, se harán pruebas con el servicio de `Oracle` en su versión gratuita. En resumen, se elige el servicio de `Amazon` porque es más escalable y económico en la eventual puesta en producción, y por ende, para la etapa de producción, se utilizará la misma alternativa.

Con esto, se hace necesario pensar en alternativas y herramientas para facilitar y acelerar el despliegue, teniendo en cuenta circunstancias donde se tenga que migrar la aplicación, y de esta forma, evitar un eventual gasto de tiempo excesivo cada vez que se tenga que realizar un nuevo despliegue sobre una nueva plataforma y/o servicio de `hosting`, ya sea en la etapa de desarrollo tanto como de producción, tomando en cuenta a su vez, que la plataforma en plena producción requeriría de depliegues ágiles en tales casos.

3.3. `Stack` de desarrollo

Una vez decidido el desarrollo de una plataforma `WEB`, es necesaria la decisión de las herramientas y recursos informáticos a utilizar, esto es, lenguajes de programación, `frameworks`, gestores de bases de datos, bibliotecas, etc. De esta forma, a continuación se abordaran los asuntos correspondientes a las decisiones sobre el `stack` de desarrollo.

[3] `https://aws.amazon.com/es/ec2/pricing/reserved-instances/pricing/`
[4] `https://www.oracle.com/cl/cloud/cost-estimator.html`

3.3.1. `Framework` para desarrollo `WEB`

Para desarrollar una plataforma `WEB`, existen distintas tecnologías, lenguajes de programación, y eventualmente `frameworks`, los cuales pretenden facilitar la labor de programación, ya que disponen de una amplia gama de funcionalidades y procesos ya programados, con lo cual no habría que destinar tiempo al desarrollo de funcionalidades que por un lado, ya fueron programadas, están realizadas por profesionales y por ende, acabarían incentivando la utilización de buenas prácticas de programación en el proyecto.

De esta manera, el uso de un `framework` se debe considerar, y con esto, a su vez se debe tener en cuenta el lenguaje de programación compatible, bibliotecas anexas y gestores de base de datos que interactúen de la manera adecuada con el `framework` seleccionado. Entre los `frameworks` más utilizados se encuentran:

1. React
2. Vue
3. Flask
4. Django
5. Ruby on Rails
6. Laravel

De esta forma, para seleccionar un `framework` conveniente para el desarrollo de este proyecto, en particular para la construcción del `Backend`, se limita la lista a solo estos: `Flask`, `Django`, `Ruby on Rails` y Laravel, dado que el resto se concentra en `Frontend`.

Dentro de estos últimos, un factor fundamental para su selección es el lenguaje de programación en el que están basados, ya que no se tiene dominio de todos ellos, y en particular, puede ser conveniente la utilización de un lenguaje por un motivo similar al expuesto en el ítem anterior: actualmente hay lenguajes más populares, y más activos frente a problemas recurrentes.

De esta forma, se tiene lo siguiente: `Flask` y `Django` se basan en `Python`, `Ruby on Rails` se basa en `Ruby` y `Laravel` se basa en `PHP`. Con esto, se debe reconocer que dentro de los lenguajes expuestos, actualmente son más populares `Python` y `Ruby`, frente a `PHP`. Por otro lado, `Python` es de mayor dominio del memorista frente a `Ruby`, además de que `Python` ostenta una mayor popularidad frente a `Ruby` lo cual sugiere una mayor facilidad de mantenimiento de la plataforma. Con esto, se reduce la lista a solo dos `frameworks` para un uso conveniente: `Flask` y `Django`.

Así, entre `Flask` y `Django` , por un lado `Flask` se usa generalmente para el desarrollo de `APIs`, mientras, que `Django` se usa para plataformas WEB en general, incluyendo `APIs`, permitiendo de mejor forma el desarrollo de `frontend`. A su vez `Django` agiliza el desarrollo de distintos aspectos que serán vistos más adelante.

Con esto es pertinente señalar que `Django` es el `framework` que mejor se adecúa a un proyecto como este.

3.3.2. Django

Django[5] es un framework de desarrollo web que permite la construcción de plataformas seguras, y de forma ágil y mantenible. Entre sus cualidades, está escrito en Python, lenguaje muy utilizado y conocido por el estudiante y con vasta documentación y desarrollos actualizados. Por otro lado, Django brinda una ORM[6] que permite un manejo más simplificado sin tener que realizar un manejo exhaustivo de consultas SQL en la base de datos.

Por otro lado, ya se tiene cierta noción de este framework, lo cual es una clara ventaja frente a otras herramientas, a la vez que el framework consta de una basta documentación y una importante comunidad desarrolladora. Para el proyecto se utilizará la versión 4.0.

3.3.3. Gestor de base de datos

El framework Django requiere de una base de datos para almacenar cada uno de los objetos dentro de los modelos que implementa. Dada las características del proyecto y del mismo framework, se recurre en primer lugar, a un sistema de base de datos basado en SQL, dado su extensión y popularidad, su compatibilidad con Django y la variedad de motores de base de datos SQL disponibles. Dentro de los motores más populares SQL se encuentran[7]:

1. Oracle
2. MySQL
3. Microsoft SQL Server
4. PostgreSQL

Django, dentro de su documentación, permite la interacción con todos esos motores, pero en particular, presenta una gran afinidad, a su vez de una gran cantidad de desarrollo enfocado en el motor PostgreSQL[8], con lo cual, para el desarrollo del proyecto se utilizará ese gestor de base de datos.

3.3.4. WSGI

WSGI, sigla para Web Server Gateway Interface, es una especificación que describe como un servidor WEB se comunica con aplicaciones WEB, y como las aplicaciones WEB podrán ser concatenadas para procesar una request [2].

[5] https://www.djangoproject.com/
[6] Object Relational Mapping: Mapeo de entidades de la base de datos a objetos.
[7] https://db-engines.com/en/ranking
[8] https://docs.djangoproject.com/en/4.0/ref/contrib/postgres/

Para las plataformas basadas en Django, las interfaces WSGI más utilizadas en ambientes de producción son Gunicorn[9] y uWSGI[10]

Dado que Gunicorn es más utilizado y posee una documentación más acabada, se selecciona esta interfaz.

3.3.5. Reverse proxy

Añadir un Reverse proxy ofrece dos ventajas principales[11], relacionadas con la seguridad como son los certificados SSL, y la inhabilidad de conocer el IP del servidor directamente.

Para este proyecto, se utilizará NGINX, dada la actualidad de desarrollo que posee, su orientación hacia la implementación de Reverse proxy frente a alternativas como APACHE, entre otras.

3.3.6. Túnel HTTP

Ya sea para una etapa de desarrollo como para de producción, un túnel HTTP es útil, sobre todo si el servidor no cuenta con una IP propia. A su vez, estos túneles ofrecen la implementación de certificados SSL de forma automática, con lo cual, se puede acceder de manera más simple a servicios externos (ej: autenticación de usuarios), a la vez que un sitio certificado ofrece una barrera de seguridad ampliamente exigida.

Para esto, se probarán dos servicios de túnel HTTP: Ngrok[12] y Cloudflare[13]. La decisión frente a cual de estos servicios utilizar se tomará en la etapa de desarrollo, dado que, a priori, el servicio ofrecido es muy similar, es decir, túnel HTTP con dirección asignada aleatoriamente.

3.4. Plataforma

3.4.1. Autenticación de usuarios

En la unidad de reclutamiento y selección de MyraSalud, sus integrantes generalmente utilizan cuentas de Google, ya sea para correo electrónico, como para acceder a la hoja de cálculo compartida, base del sistema actual.

Es por esto que, tomando en cuenta este último punto, se decide integrar en el sistema la biblioteca django-allauth, la cual, por medio del protocolo OAuth 2.0[14], permite el inicio

[9] https://gunicorn.org/#docs
[10] https://uwsgi-docs.readthedocs.io/en/latest/
[11] https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/
[12] https://ngrok.com/docs
[13] https://www.cloudflare.com/products/tunnel/
[14] https://oauth.net/2/

de sesión, es decir, la autenticación de usuarios, por medio de las cuentas asociadas a servicios externos, en este caso, `Google`.

A su vez, la biblioteca ofrece, las vistas e interfaces medianamente desarrolladas, tal que se pueden adaptar a las interfaces propias del proyecto.

Sumado a lo anterior, la biblioteca también ofrece la posibilidad de manejar cuentas no asociadas a servicios externos con lo cual, no se pierde esta capacidad, y tampoco se obliga a los eventuales usuarios a crearse cuentas en servicios que podrían no utilizar.

De esta forma, `django-allauth` pretende facilitar los accesos, sin perder a la vez capacidades previamente contempladas, a la vez que usa un protocolo seguro y validado para los accesos.

Para el proyecto, se utiliza la versión 4.0.3, última versión disponible en Q2 2022.

3.4.2. Entidades

Oportunidades laborales

Dentro del estudio realizado en la unidad de reclutamiento de MyraSalud, se definió claramente una de las unidades, la oportunidad laboral, la cual consiste básicamente en los trabajos o requerimientos de profesionales/técnicos para una atención de salud. Para definirla, se requieren de distintos campos, tanto de información propia de un empleo, como de información respecto de las habilidades médicas o técnicas requeridas par tal atención.

Las oportunidades laborales se definieron bajo los siguientes parámetros:

1. Nombre: Nombre de la postulación. (varchar)
2. Cantidad de Postulantes requerida (integer)
3. Turno: Tipo de turno requerido para el cuidado médico. (varchar)
4. Comuna: Localización del requerimiento médico.
5. Usuario solicitante: Usuario de la plataforma que crea el requerimiento.
6. Fecha del requerimiento: Fecha límite para la contratación.(timestamp)
7. Descripción: Descripción de la oferta (varchar)
8. Sueldo mínimo (integer)
9. Sueldo máximo (integer)
10. Fecha de la entrevista técnica (timestamp)
11. Tipo de Contrato (varchar)
12. Requiere celular (boolean)

13. Requiere notebook (boolean)
14. Requiere uniforme (boolean)
15. Dependencia: Usuario de la plataforma responsable el requerimiento.
16. Fecha de inscripción: Fecha del registro de la oportunidad laboral. (timestamp)
17. Status (abierta o cerrada)
18. Oportunidad SOS: Determina si la oportunidad es FULL TIME o SOS. (boolean)
19. Habilidades Técnicas: Lista de habilidades médicas.

Postulantes

Al igual que con la entidad anterior, los postulantes se definieron una vez se estudió la unidad de reclutamiento, a la vez que el sistema actual de información que ellos utilizan. Los campos definidos recorren desde información personal, hasta habilidades técnicas. Finalmente, los postulantes quedaron definidos como sigue a continuación:

1. Nombre completo: Nombre del postulante (varchar)
2. RUN (varchar)
3. Teléfono 1 (varchar)
4. Teléfono 2 (varchar)
5. email (varchar)
6. Fecha de nacimiento (date)
7. Dirección (varchar)
8. Comuna
9. Licencia de Conducir: Tipo de licencia. (Integer)
10. Años de experiencia (Integer)
11. Sexo (varchar)
12. Nacionalidad (varchar)
13. Status: Contratable, no contratable, contratado. (Integer)
14. No contratable: Mensaje adicional describiendo la condición de "no contratable". (varchar)
15. Información adicional. (varchar)
16. Habilidades Técnicas: Listado de habilidades médicas dentro de las cuales se seleccionan las que cumple el postulante.
17. SOS/FULL TIME. (boolean)
18. Preguntas psico-laborales: Listado de preguntas adicionales. (lista de varchars)
19. Puntaje de respuesta: Prioridad de acuerdo a la tasa de respuesta afirmativa. (integer)

Habilidades Técnicas

A su vez, se definen las habilidades técnicas dentro de las cuales, para cada postulante, se seleccionan las que cumplen. Las habilidades definidas a priori son las siguientes:

1. Traqueotomía
2. Gastrostomía
3. Sondeo vesical
4. Sonda Foley
5. Colostomía
6. Cistostomía
7. Estimulación anal
8. Inyecciones intramusculares
9. Sonda nasogástrica
10. Manejo de parapléjicos/postrados
11. Uso de tecle
12. Equipo SARS para daño medular
13. Transferencia

Además, cabe la posibilidad de añadir más habilidades, dado el carácter de entidad independiente, es decir, no depende de los modelos de oportunidad ni postulante.

Preguntas Psico-laborales

De manera similar a las habilidades técnicas, existen preguntas psico-laborales, las cuales, a priori, se definen como las siguientes:

1. ¿Cómo actuarías si estás frente a un caso de violencia intrafamiliar?
2. Si sufro la agresión por parte de un paciente debo
3. Entorno familiar
4. ¿Cómo te describes en breves palabras?

Al igual que con las habilidades técnicas, se pueden añadir más preguntas, o eliminarlas de acuerdo al contexto y circunstancias, por ejemplo, agregar preguntas sobre COVID-19.

3.4.3. Interfaces

A continuación, se presentarán algunos diagramas a modo de bosquejo, elaborados con el fin de presentar las ideas respecto al diseño preliminar de las interfaces, tanto como referencia para el cliente, como para el estudiante memorista.

Oportunidades laborales

Para la creación/edición de oportunidades laborales, se requiere de formularios, que consideren campos de texto o numéricos, a la vez que para seleccionar las habilidades técnicas propias de la oportunidad, se propone un tipo de interfaz donde se puedan agregar múltiples habilidades técnicas, con una prioridad determinada.

Para esto se pretende utilizar una multiselección de ítems como se muestra en la siguiente imagen:

Figura 3.1: Ejemplo de multiselección de ítems.

Postulantes

Para la creación/edición de postulantes, se crearán las interfaces respectivas, donde, para la selección de habilidades técnicas, se utilizará una lista de `checkboxes`. Para el resto de campos se utilizarán campos numéricos o de texto según corresponda y su disposición será decidida posteriormente.

Asignación de Postulantes a Oportunidades Laborales

Para la asignación de postulantes a la oportunidades laborales, se expone un esquema (Figura 3.2), donde para una oportunidad en específico, se listan postulantes posibles y postulantes seleccionados.

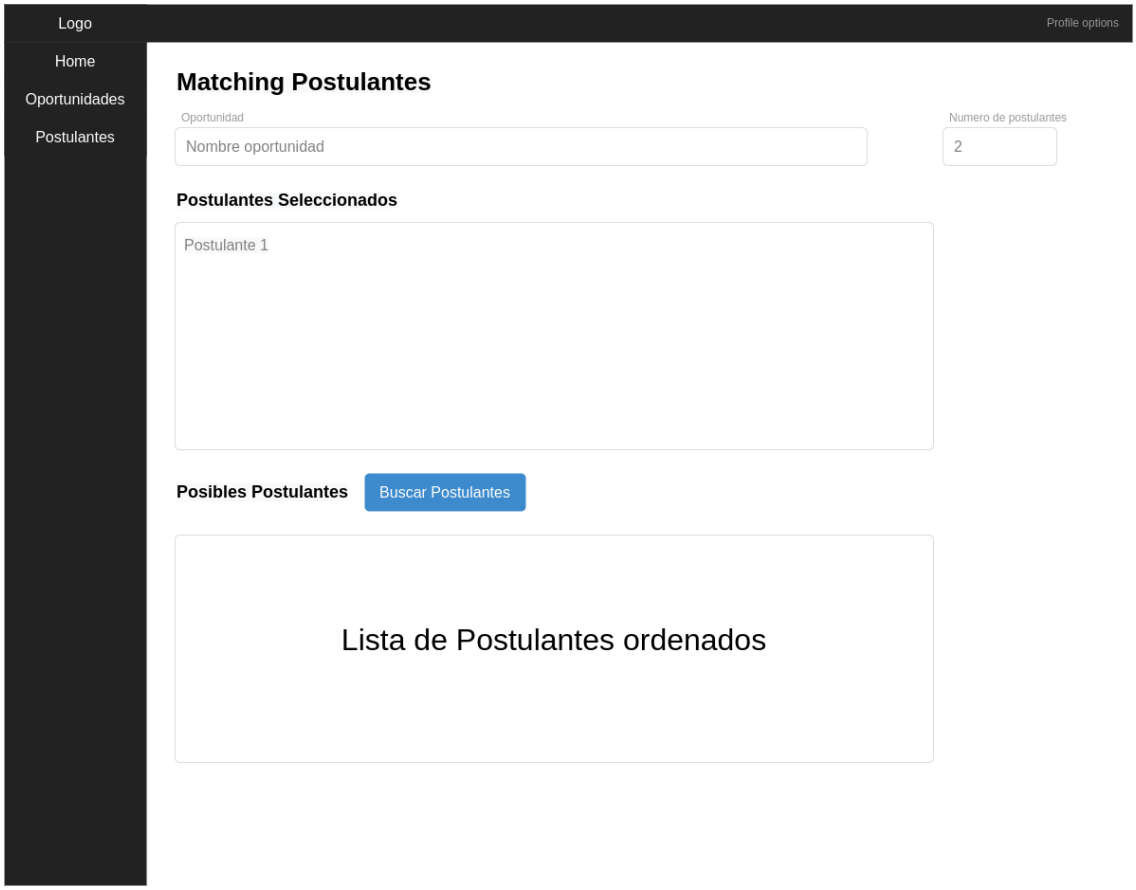

Figura 3.2: Diagrama de vista de asignación de postulantes

3.5. Optimización por puntaje de respuesta

Uno de los problemas señalados en el capítulo anterior fue la incapacidad por parte de la organización para negociar pagos razonables frente a requerimientos urgentes, dada la asimetría de información frente a las oportunidades laborales.

Frente a esta situación, tras discusiones con el cliente, el memorista propone un sistema de puntajes de acuerdo a las respuestas positivas frente a los requerimientos. Esto se implementaría de la siguiente manera:

1. Inicialmente, cada postulante comienza con un puntaje igual a cero.
2. Si para cierta oportunidad, un postulante contesta afirmativamente una vez se contacta, correspondería incrementar este puntaje en 1.
3. De lo contrario, si el postulante contesta de forma negativa, se decrementa este valor, llegando eventualmente a valores negativos, sin que esto represente un problema.

De esta manera, cuando luego se seleccione al postulante para cierta oportunidad, este valor actuaría como un valor de prioridad, con el cual se puede ordenar a los postulantes aptos para la oportunidad, beneficiando a los que mejor responden a las oportunidades.

3.6. Sistema de gestión de archivos

Para cada postulante, se requiere del almacenamiento de la documentación, ya sea documentos de identidad, hasta la acreditación de las habilidades y capacidades.

Dado que lo ideal sería que los documentos fuesen accesibles, visibles y gestionables, el estudiante memorista propone la idea de utilizar `software` de terceros, `open-source`, de forma tal que se proporcione una herramienta de gestión de documentos (EDMS[15]), sin consumir excesivo tiempo de desarrollo.

Dentro de los `softwares` de gestión documental `open-source` más populares, se encuentran:

1. OpenKM
2. OpenDocMan
3. LogicalDoc
4. Krystal DMS
5. Kimios
6. Mayan EDMS
7. SeedDMS
8. Alfresco community edition

Analizando los `softwares`, con el propósito de vincularlo a la herramienta de desarrollo, destaca `Mayan EDMS`[4], la cual está programada sobre `Django`, recibe soporte en la actualidad, es bastante utilizada debido a las características anteriores, además de que cuenta con una `API` con la cual, a priori, se podría interactuar.

Además, esta herramienta cuenta con un formato basado en gabinetes, con lo cual, se establecería un gabinete definido con la documentación necesaria para cada postulante

Por otro lado, el despliegue de la plataforma se simplifica, debido a que cuenta con la opción de `deployment` mediante Docker y Docker-compose. Con esto, se preselecciona esta herramienta para su análisis y eventual implementación.

[15]EDMS: Electronic Document Management System

Capítulo 4

Implementación de la solución

4.1. Plataforma

4.1.1. Autenticación

Como se señaló en el capítulo anterior, para la autenticación de usuarios, se recurrió a la librería `django-allauth`, la cual admite múltiples servicios y plataformas externas para validar usuarios. Para este caso en particular, se enlazó el servicio de autenticación de `Google`, el cual, por medio de un `Token` generado dentro de la consola de `Google Cloud Services`, se enlaza el servicio de autenticación, de forma tal que el `login` de la plataforma redirige al `login` de `Google`.

Dentro de la implementación, surgieron distintas dificultades, en particular, el servicio de autenticación requiere de una dirección de re-direccionamiento certificada bajo SSL, lo cual, tomando en cuenta la fase de desarrollo, no estaba previsto.

Luego, mediante el uso de túneles, se pudo acceder a una dirección certificada. Por otro lado, mediante los ensayos y pruebas del `login`, se experimentó, otro problema, el cual, corresponde a que el servicio de autenticación de `Google` tiene cierta demora cada vez que se inscribe una nueva dirección de re-direccionamiento, lo cual en un principio se desconocía, pero que luego fue señalado y superado, es decir, fue algo que se consideró para los sucesivos cambios de dirección de re-direccionamiento, otorgando un tiempo de aseguramiento (10 minutos) tras el cual ya no se experimentaba el problema.

Descripción del sistema de autenticación

Aparte del `login`, el sistema de autenticación, permite la recuperación de claves, cambio de claves, configuración del `login` externo, vale decir, mediante `Google`, entre otras funcionalidades propias de un sistema de autenticación de usuarios, con lo cual se obtiene un sistema integral de autenticación.

A su vez, el cambio y recuperación de contraseña se realizan mediante el envió de un correo electrónico al email vinculado a la cuenta, mediante un servidor de correo electrónico configurable. Durante las pruebas, se utilizó un servidor `smtp`, gestionado por el memorista.

A su vez, cabe señalar que `Django` posee servidores "`Dummy`" de correo electrónico, es decir, servidores con funcionalidad orientada a las pruebas, ya sea guardando los correos en lugar de enviarlos o simplemente no realizando ninguna acción al momento de dar la orden, todo esto con el propósito de no saturar una eventual bandeja de entrada de un correo electrónico con correos de prueba cuando no fuese estrictamente necesario.

Interfaz de login

A continuación, un ejemplo de la interfaz de login:

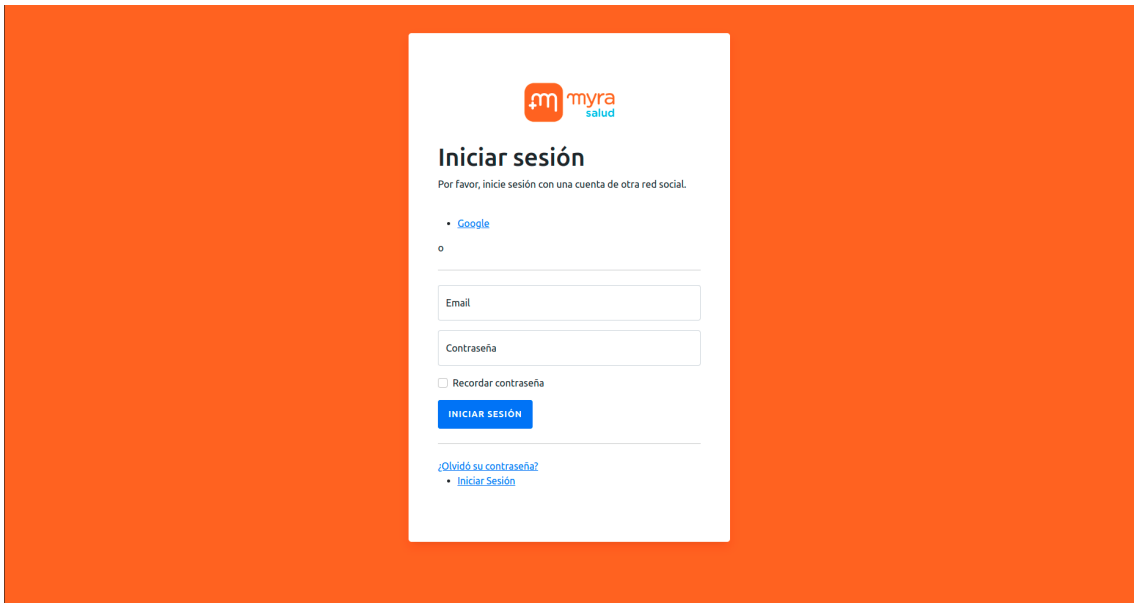

Figura 4.1: Vista del Login

4.1.2. Entidades

Postulantes

Esta entidad, definida anteriormente, cuenta con una id como llave primaria, pero además su RUN debe ser único.

Para el campo de licencia de conducir (driver license), existen opciones definidas, mapeadas a valores enteros. Estas son:

1. No tiene/No informa

2. Autos
3. Motos
4. Taxis
5. Transporte privado 17 personas
6. Transporte privado ilimitado y escolares
7. Carga hasta 3500 Kg
8. Carga ilimitado
9. Maquinaria
10. Tracción animal
11. Vehículos policiales bomberos y FFAA

A su vez, para el campo de `status`, existen tres opciones:

1. Contratable
2. No Contratable
3. Contratado

Donde, al seleccionar la opción "No contratable", desbloquea un campo de texto nuevo para justificar este estado.

A su vez, cada postulante posee habilidades técnicas, para lo cual, se aplica una relación de tipo `Many to Many`, es decir, cada postulante se relaciona con varias habilidades, y viceversa.

Por último, cada postulante cuenta con preguntas psico-laborales, para lo cual, también se utiliza una relación de tipo `Many to Many`, pero esta vez, con un campo adicional correspondiente al contenido de la respuesta (texto). De esta forma, por un lado, se pueden relacionar múltiples preguntas con múltiples postulantes, a la vez que se pueden añadir preguntas sin perturbar los registros anteriores de cada postulante.

Esto se logra, por un lado, con la eliminación "en cascada" de cada pregunta en el caso de que se requieran eliminar, y en el caso de que se agreguen preguntas, estas se agregan por defecto con campos vacíos y de manera postergada ("`lazy`"), con lo cual no se debe revisar cada registro de postulante cada vez que se añada una pregunta.

A su vez, se señala que la facultad para añadir/eliminar preguntas es propia de un administrador, por lo cual se limita solo a este tipo de usuarios, por medio de la interfaz de administración de `Django`. Una tabla con los campos de esta entidad, descritos a nivel de base de datos, se expone a continuación.

Nombre	Tipo
name:	varchar(100)
run:	varchar(12)
telephone_1:	varchar(15)
telephone_2:	varchar(15)
email:	varchar(255)
birthday:	date
address:	varchar(50)
driver_license:	integer
experience_years:	integer
sex:	integer
nationality:	varchar(20)
status:	integer
additional_info:	varchar(300)
comuna_id:	bigint
SOS_FULL:	bigint
status_message:	varchar(300)
availability_score:	integer

Tabla 4.1: Campos de la entidad "Postulante"

Oportunidades laborales

La entidad de oportunidades laborales, cuenta con un campo correspondiente al `ID`, para su identificación. Para identificar la oportunidad por parte de los usuarios de la plataforma, se utiliza el nombre.

A cada oportunidad, se le define la cantidad de postulantes requerido (`required qty`). A su vez, se le define un turno correspondiente (horarios de los profesionales/técnicos una vez seleccionados), la fecha del requerimiento, descripción, rango de salario (mínimo/máximo), fecha de la entrevista técnica, tipo de contrato, entre otros.

La oportunidad tiene un campo `status`, correspondiente a si la oportunidad esta abierta o cerrada. A su vez, se especifica si la oportunidad es del tipo `Full time` o `SOS`. Mediante llaves foráneas se enlaza con una comuna respectiva para definir su locación.

Cabe señalar que los eventuales postulantes no se agregan directamente a esta entidad ya que se requiere de una relación `Many to many`, con lo cual se usa una entidad tercera para este propósito.

En la tabla 4.2, se especifican los campos de la entidad.

Nombre	Tipo
name:	varchar(100)
required_qty:	integer
shift:	varchar(100)
date_required:	timestamp with time zone
description:	varchar(300)
min_salary:	integer
max_salary:	integer
technical_interview:	date
contract_type:	varchar(100)
cellphone_required:	boolean
notebook_required:	boolean
uniform_required:	boolean
date_added:	timestamp with time zone
status:	boolean
sos:	boolean
comuna_id:	bigint
dependency_id:	bigint
requestor_id:	bigint

Tabla 4.2: Campos de la entidad "Oportunidad laboral"

Preguntas a Postulantes

Esta entidad modela las preguntas que se realizan a cada postulante, de esta forma se define, por medio de un campo "`question`", referente a la pregunta en sí. Para enlazar cada pregunta a cada uno de los postulantes, se utiliza una entidad diferente, llamada "`ApplicantQuestionAnswer`", debido a que se requiere de una relación `Many to Many`, a la vez que esta entidad auxiliar guarda la respuesta a cada pregunta.

Al modelar de esta manera las preguntas a los postulantes, se facilita el agregado/borrado de preguntas, dado que al eliminar una pregunta, se eliminan las respuestas asociadas, y por el contrario, al agregar una pregunta, al no requerir de un valor por defecto (vacío), no es necesario repasar cada pregunta agregando una respuesta respectiva, solo hasta cuando se revisa al postulante en cuestión, es decir, se procede con `laziness`[1].

Habilidades Técnicas

De manera similar al campo anterior, las habilidades técnicas se modelan como entidad una especial, lo que permite agregarlas o bórralas con facilidad. Estas se relacionan con cada postulante mediante una relación `Many to many`, con lo cual `Django`, crea automáticamente una entidad intermedia para llevar a cabo esta vinculación.

[1]Laziness: Refiere a la evaluación perezosa.

Se señala que la creación y borrado de las preguntas se realiza por medio de usuarios administradores en el perfil de administración de `Django`, con tal de limitar su acceso.

Usuarios

Si bien, `Django` automáticamente designa una clase dedicada para la gestión de usuarios, esta es bastante básica y limitada con lo cual se hace necesaria su modificación. De todas maneras, para este caso en particular, aunque las modificaciones son limitadas, se crea una clase nueva para el manejo de usuarios. Esto se sugiere como una buena práctica, dado que, comúnmente se le realizan modificaciones a la gestión de usuarios, y llevarla a cabo una vez avanzado el proyecto es un proceso complejo, dado que se deben modificar múltiples secciones del código.

Con esto, se siguió este camino, dado que, en primer lugar, se utilizó un servicio de `login` distinto, con lo cual, se adaptó el modelo de usuarios a uno donde se identificase a los usuarios principalmente por su correo electrónico. Además se declara la distinción entre usuarios administradores y no administradores.

En definitiva, este "`override`" que se le realiza al modelo de usuarios, facilita futuras modificaciones, dado que esta reimplementación se hizo en el instante correcto, evitando futuros problemas.

Locaciones

Las locaciones son gestionadas por dos modelos: Regiones y Comunas. La regiones se definen por sí solas, con un `ID` y su nombre respectivo. Estas corresponden a las regiones de Chile. A su vez, las comunas se definen con nombre e `ID`, además de que dependen, mediante llave foránea, de las regiones.

Las oportunidades laborales, y los postulantes, esta ligados a una comuna, y por ende, indirectamente, a una región.

Resumen de base de datos

La base de datos no fue creada directamente por el estudiante memorista, ya que esta es formulada por `Django`, mediante su `ORM`. De todas formas, las definiciones son llevadas a cabo por medio de modelos de `Django` que son básicamente un mapeo de lo que serian las entidades `SQL` equivalentes. De todas formas, se adjunta un diagrama de la base de datos generada en el anexo. Ver anexo [2]

4.1.3. Interfaces

Las interfaces fueron desarrolladas mediante herramientas (lenguajes) como `CSS, HTML` y `JS` además del framework `Bootstrap`, en su versión 5.0[2]

Figura 4.2: Ejemplo de la interfaz

`Django`, en su arquitectura `MVT`[3], facilita la generación de las interfaces mediante una suerte de `HTML` "enriquecido", donde se facilita el uso de interpolación de texto para generar las vistas con la información requerida. De la misma forma, se implementa el uso de herencia de `HTML`, de la misma manera, interpolando secciones de texto definidas con anterioridad.

Postulantes

Para la creación y edición de postulantes, se establece un formulario como se ve en la figura 4.3.

El formulario, dispone de validación de RUN, y las regiones y comunas se cargan mediante una consulta `Ajax`. Las habilidades técnicas se seleccionan a través de una lista de `checkboxes`, como se ve en la figura 4.4

Oportunidades Laborales

Al igual que con los postulantes, para las oportunidades laborales se tienen formularios para la creación y edición como se ve en la figura 4.5. A su vez, la selección de habilidades

[2] `https://getbootstrap.com/docs/5.0/getting-started/introduction/`

[3] MVT: Model-View-Template, variación del MVC, que hace énfasis en la generación de los `templates` por lado del servidor.

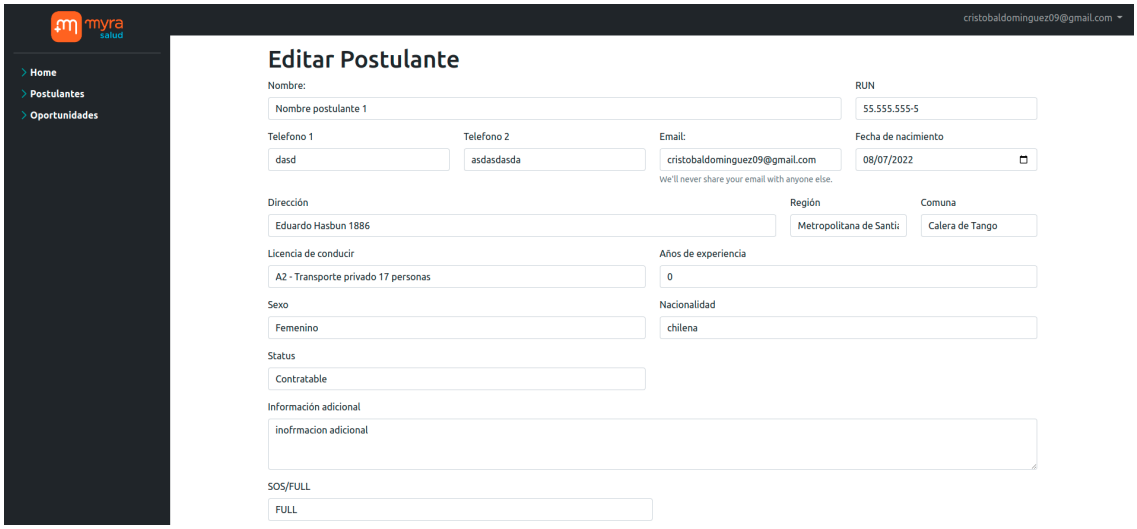

Figura 4.3: Formulario de edición de postulante

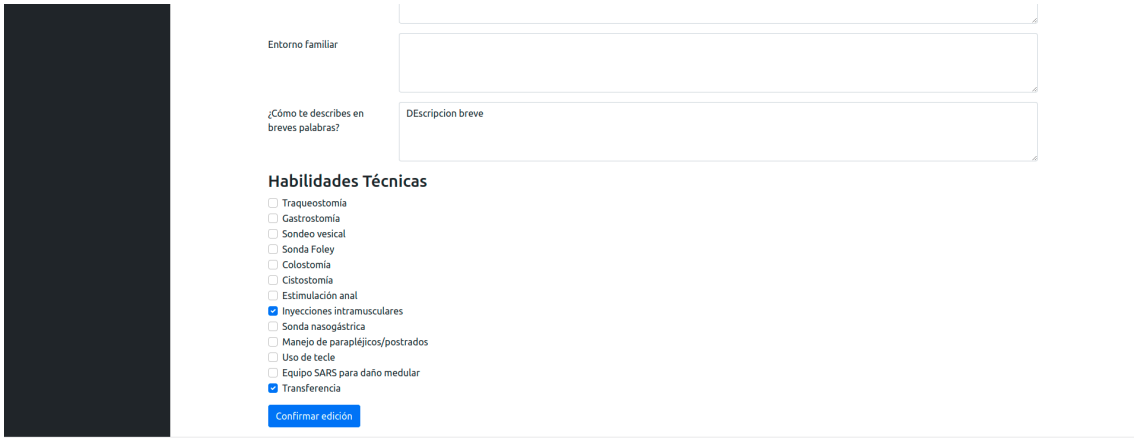

Figura 4.4: Selección de habilidades técnicas para un postulante

.

técnicas por orden, se hace mediante una campo de multiselección, como se ve en la figura 4.6.

Asignación de Postulantes a oportunidades laborales

Para la asignación de postulantes a las oportunidades laborales, se establece una interfaz de `Matching` de postulantes, donde para una oportunidad en particular, se pueden buscar postulantes de acuerdo al criterio de búsqueda definido en el siguiente capítulo. La figura 4.7 muestra a la interfaz.

Figura 4.5: Formulario de edición de oportunidad laboral

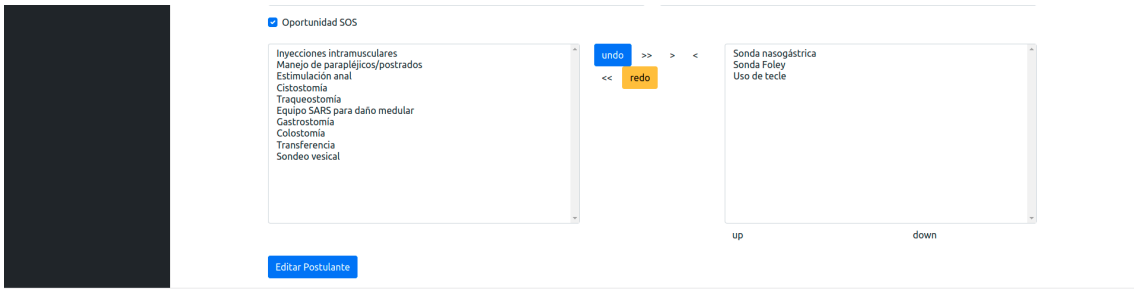

Figura 4.6: Multiselección de habilidades.

4.1.4. Selección de postulantes

La selección de postulantes implica un algoritmo de consultas un poco complejo, pero que se describirá a continuación. Cabe señalar que en `Django` generalmente no se programan las consultas `SQL`, sino que se usan métodos propios de la `ORM`, con lo cual no se describirán consultas de este tipo. El algoritmo se describe a continuación.

Para una oportunidad laboral definida:

1. Se seleccionan los postulantes que no estén contratados, en particular, los contratables e incluso, los no contratables.
2. Luego se filtran los que pertenecen a la misma región que la oportunidad.
3. Después, para cada habilidad técnica, se filtran los postulantes que la cumplan, esto genera grupos de postulantes para cada habilidad.
4. En seguida, se agrupan estos conjuntos, y se ordenan por el mayor numero de repeticiones, lo cual ordena a los postulantes por la cantidad de habilidades que coinciden.
5. Finalmente se ordenan por el puntaje de respuesta, priorizando a los que tienen mejor puntaje dentro de los que comparten igual numero de habilidades.

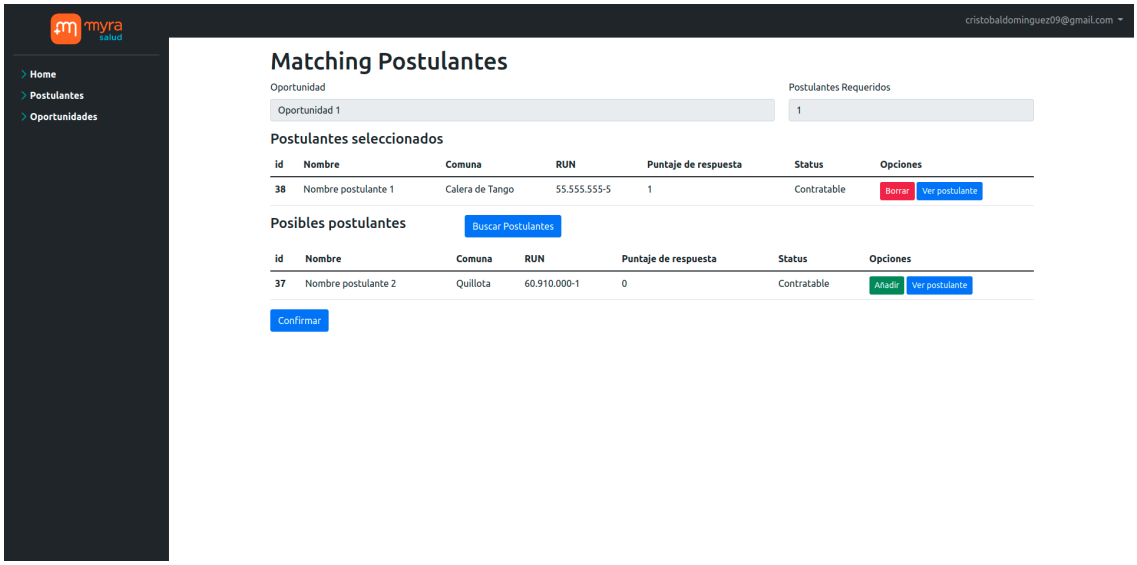

Figura 4.7: `Matching` de Postulantes.

De esta forma, se obtiene un listado de candidatos que deben contactarse, y luego agregarse eventualmente. Este método de selección fue propuesto por el estudiante memorista y aceptado por el cliente.

4.1.5. Optimización por puntaje de respuesta

La implementación de este ítem consiste simplemente en un campo adicional, numérico, que se puede modificar en el apartado de edición de cada postulante. De esta forma se consigue una implementación simple y suficiente para poder calificar la respuesta de los postulantes. Luego, este parámetro numérico se utiliza en la priorización de postulante para las eventuales oportunidades laborales.

4.2. Uso de contenedores

En la fase de diseño de la plataforma, se planteó la necesidad de agilizar los despliegues de esta, es decir, las puestas en marcha cada vez que se cambiase de `hosting` o infraestructura. Es por esto que se considera la `containerización` de la aplicación.

De esta forma, se utilizó `Docker` y `Docker Compose`, de forma tal que mediante `Docker Compose`, herramienta que simplifica el lanzamiento de distintos contenedores simultáneamente, se pudiese automatizar el `deployment` de la aplicación, en diferentes contextos y `hardware`.

Así, se crearon los siguientes contenedores:

1. Django-app: Contenedor con la aplicación de `Django`, y `WSGI Gunicorn`.

2. Nginx: Contenedor con `reverse-proxy` Nginx.
3. Postgres: Contenedor con el gestor de base de datos `PostgreSQL`.
4. CloudFlared: Contenedor para pruebas, usado como túnel para acceso remoto mediante dirección `WEB`.

Para el contenedor de Django-app se automatizó el proceso de poblamiento de base de datos para los datos estáticos, por ejemplo regiones y comunas, mediante la opción `load-data` de `Django`. A su vez se automatiza la creación del superusuario administrador para acceder a los datos.

En el contenedor `Postgres` se automatiza la creación de la base de datos y el usuario de base de datos de forma tal que luego el contenedor de Django-app migra los modelos a la recién creada.

Luego se inicia el contenedor de `Nginx`, el cual crea el `reverse-proxy` entre la aplicación de `Django` y el posterior túnel de `Cloudflare`. Este último, lanza un túnel entre `reverse-proxy` y un servidor gratuito de `Cloudflare`, de forma tal que desde el exterior se pueda acceder a la plataforma, mediante una dirección `WEB`.

Cabe señalar que el propósito del túnel es poder lanzar la aplicación al exterior, en servidores limitados por redes `CG-NAT` o similares, donde no se posee una dirección de `IP` propia, necesaria para un servidor.

Si bien en un principio, se lanzó la aplicación en una instancia gratuita de AWS con `IP` propia (no requiere túnel), el `hardware` ofrecido era limitado, además de que el tiempo de uso gratuito fue excedido. Además, existieron problemas con la obtención de certificados SSL. De todas formas, no se descarta su uso, siempre y cuando se acceda a un instancia de mejores características, y a su vez, el uso de un túnel `Cloudflare` no se desperdiciaría, ya que, con este servicio, se facilita la asignación de direcciones `WEB`, en caso de requerir un cambio.

4.3. Mayan EDMS

En esta sección, se describe el uso y pruebas que se le hizo el gestor de documentos `Mayan EDMS`.

En resumen, `Mayan EDMS`, cuenta con instalación mediante `Docker Compose`, con lo cual pudo ser desplegado de esta manera. A su vez, utiliza `PostgreSQL`, con lo cual se gana esa compatibilidad con lo previamente implementado.

Dentro de lo que se alcanzó a experimentar, se encuentra la `API`, con la cual se pudieron crear gabinetes, entre otras cosas. En la figura 4.8, se ve un ejemplo de la documentación de la `API`, la cual se accede desde la aplicación desplegada.

De todas formas, no se logró integrar de manera completa, debido a falta de tiempo, de todas formas, el uso del sistema fue validado por el cliente.

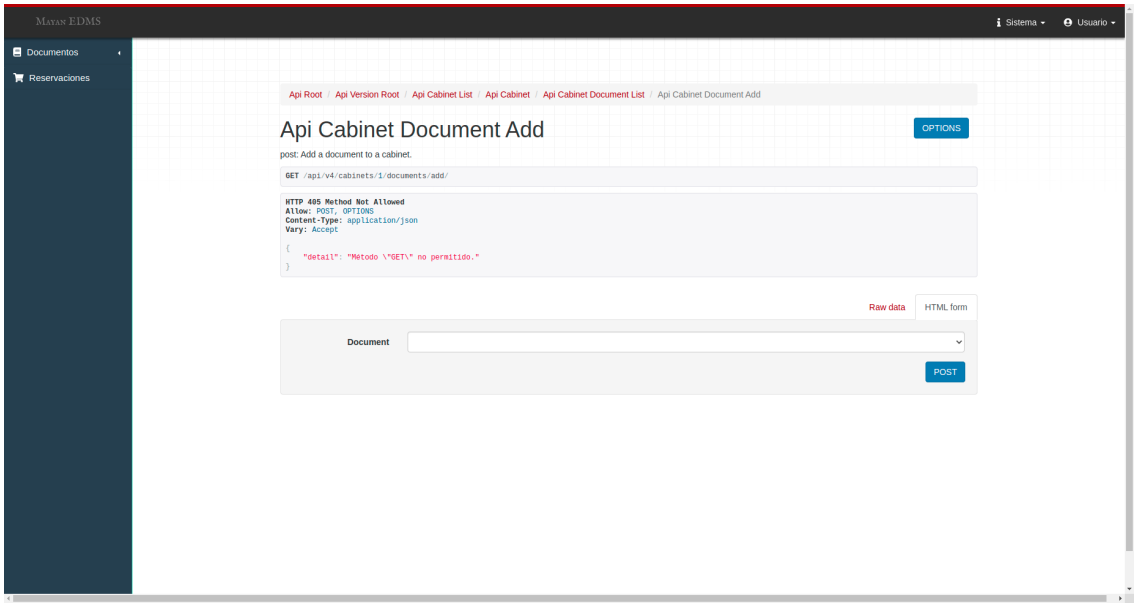

Figura 4.8: Ejemplo de Mayan EDMS

Capítulo 5

Validación de la solución

A continuación, se presentarán casos de uso con procesos relevantes dentro de la plataforma. Cabe señalar que estas dinámicas fueron validadas por el cliente, salvo pequeños criterios estéticos.

5.1. Postulantes

Se presenta el caso de uso correspondiente a la creación de un postulante, con su formulario respectivo. En primer lugar, mediante el menú lateral (`sidebar`), se accede a la opción "Añadir Postulante", tal como se observa en la figura 5.1.

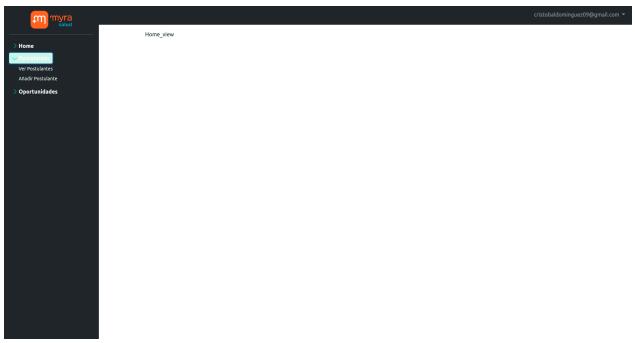

Figura 5.1: Opción "Crear Postulante" en el `Sidebar`.

Luego, se observa un formulario completo, salvo que se notifica un error, en este caso, se pretendía agregar un usuario con un RUN ya existente con lo cual, no se puede concluir la agregación del postulante. (Figura 5.2).

Finalmente, se corrige el error, observándose así, el nuevo postulante al final de la lista (Figura 5.3).

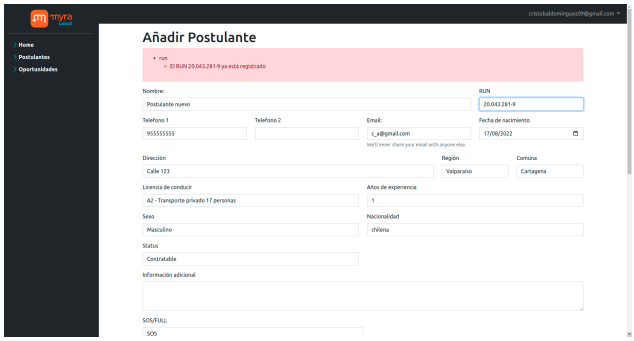

Figura 5.2: Error de RUN ya existente.

Figura 5.3: Nuevo postulante en lista.

5.2. Oportunidades laborales

Ahora, se expone el caso de uso correspondiente a la edición de una oportunidad laboral, con su formulario respectivo. A través del `sidebar`, se accede a la opción "Ver Oportunidades", tal como se observa en la figura 5.4.

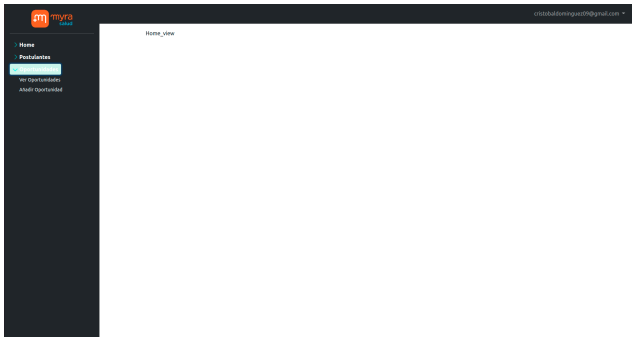

Figura 5.4: Ver Oportunidades en `Sidebar`.

En seguida, se muestra una oportunidad en el listado, la cual se procederá a editar mediante el botón correspondiente. (Figura 5.5).

Finalmente, se observa el menú de edición, donde se procede a añadir habilidades técnicas al postulante (Figura 5.6).

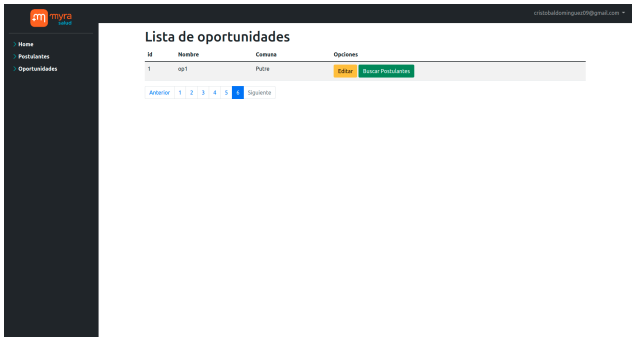

Figura 5.5: Editar postulante en lista.

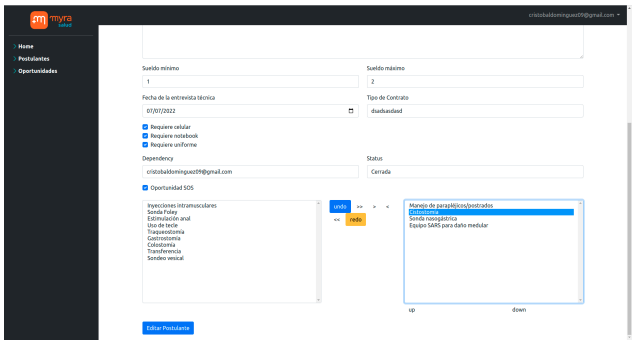

Figura 5.6: Edición de postulante.

5.3. Búsqueda de postulantes

Para el `matching` de postulantes con una oportunidad laboral, se muestra el caso de uso que procede como se expone a continuación.

En primer lugar, se observa la vista del `matching` propiamente tal (Figura 5.7, donde para la "Oportunidad 1", no se tiene ningún postulante seleccionado.

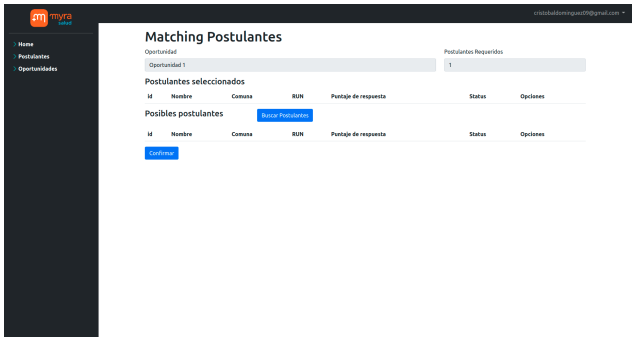

Figura 5.7: Vista inicial `matching` de postulantes

Después, se presiona el botón de búsqueda, el cual, por medio del algoritmo señalado en un capítulo anterior, reúne a los postulantes correspondientes (Figura 5.8).

Y finalmente, se observa la asignación propiamente tal, por medio del botón añadir, donde la oportunidad de `ID` 61 es asignada, para luego ser confirmada con el botón al fondo del

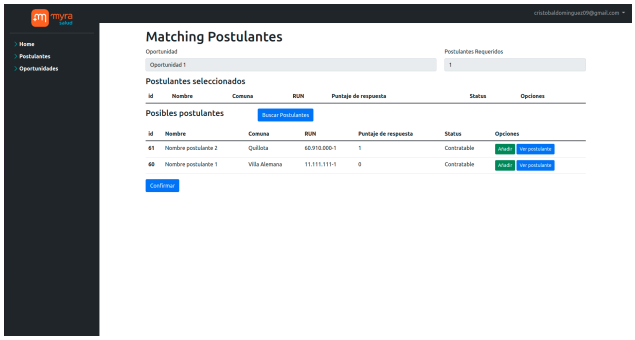

Figura 5.8: Búsqueda de postulantes para la oportunidad.

formulario (Figura 5.9).

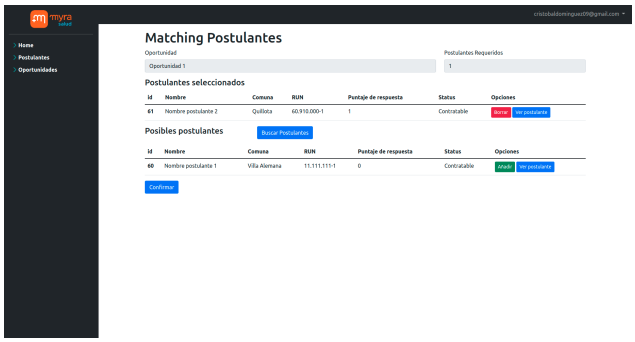

Figura 5.9: Asignación de postulantes para la oportunidad.

5.4. Perfil de administrador

Algunas funciones como añadir preguntas al formulario de postulantes, se limitan al perfil de administrador de `Django`, con lo cual, solo usuarios con tal privilegio, pueden hacer este tipo de modificación. En la imagen 5.10, se observa un ejemplo del perfil de administración.

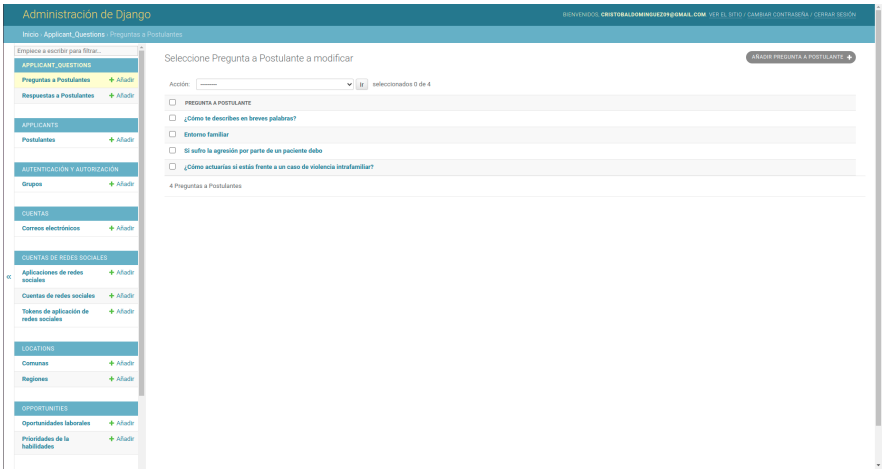

Figura 5.10: Perfil de administración de `Django`: Añadir pregunta de postulante

Capítulo 6

Conclusión

En este capítulo, se presentan los principales hitos, logros, aprendizajes, comentarios y proyecciones del proyecto realizado y relatado en los capítulos anteriores.

6.1. Generalidades

El proyecto, se concluye por el momento, con una plataforma que permite el el manejo de postulantes, oportunidades laborales, y su vínculo, pero que todavía carece de la gestión de documentos.

La relación con el cliente fue buena, pero no fue con la frecuencia óptima, con lo cual se indujeron demoras y retrasos, en el desarrollo.

6.2. Comunicación con el cliente

Para un proyecto de estas características, donde se está creando una plataforma para la gestión de una unidad de reclutamiento, es esencial la comunicación con el cliente, debido a que esta plataforma desempeñaría un papel fundamental para este, a la vez que el desarrollo iría orientado de mejor manera hacia los intereses del cliente, reduciendo la cantidad de contratiempos en la medida que el cliente se conforma con el desarrollo.

En este caso, la comunicación con el cliente no fue la óptima, debido a que por una parte, el estudiante memorista experimentó una serie de contratiempos en las fases de desarrollo, y por el otro, el cliente tenía una agenda bastante ocupada, lo que muchas veces dilataba los ciclos de intercambio y retroalimentación necesarios para el desarrollo de un proyecto como este.

La comunicación, representa un eslabón fundamental, ya que, de cierta manera, otorga cierta agilidad al desarrollo en la medida que este último se adapta de mejor manera a los intereses, estáticos o cambiantes que podría tener el cliente. De todas maneras, dentro de la

comunicación que se pudo conseguir, esta fue bastante fructífera, desde la etapa de concepción del proyecto, hasta las fases de desarrollo, como fue el caso de la discusión del sistema de puntaje de respuesta, el cual, se realizó en un intercambio multidisciplinario, por un lado con el estudiante memorista, y por parte del cliente, representado por una de las encargadas del área de reclutamiento y selección (psicóloga), y la gerencia de MyraSalud (abogado).

6.3. Conocimientos previos

En lo que respecta al estudiante memorista, este contaba con conocimientos en tecnologías `WEB`, lo cual sirvió en gran medida al desarrollo de esta plataforma, pero, de todas maneras, este no fue del todo suficiente, con lo cual se tuvo que estudiar bastante sobre las tecnologías, en particular, `Django`, que si bien, se conocía, y se había trabajado con este, en las fases de diseño y programación del proyecto, se recurrió a distintos recursos a modo de estudio y referencia, dado que se experimentaron distintas situaciones donde, al no poder asegurar una solución convincente, se recurrió tanto a la documentación de `Django`, como de otros recursos y librerías implementadas, como por ejemplo `AJAX`, `multiselect`, entre otras.

Por el lado del despliegue y puesta en marcha de plataformas `WEB`, el conocimiento previo, se limitaba al ámbito teórico, es decir, nunca se había puesto en marcha una plataforma `WEB` desde cero, personalmente. De esta forma, también se indagaron recursos, bibliografía, etc. de forma tal que se logró adquirir conocimiento, aunque con el debido contratiempo.

En conclusión, si bien el conocimiento era limitado, se contaba con cierta base y fundamentos, de forma tal que se pudo avanzar de manera considerable en el proyecto a pesar de nunca haber realizado un desarrollo de una magnitud comparable.

6.4. Aprendizajes

6.4.1. `Django`

Como piedra angular del proyecto, `Django` es una herramienta bastante capaz, que sin duda simplificó muchos aspectos de la programación envuelta en el proyecto. A pesar de que el dominio previo de esta herramienta era, en parte limitado, durante el transcurso del proyecto se adquirió un conocimiento y dominio importante.

En esta línea, gran parte de lo aprendido, se hizo en base a la documentación oficial de `Django`, como también de cursos orientados al manejo profesional de este `framework`, con lo cual, se promovieron las buenas prácticas de desarrollo, independiente del resultado final. En síntesis, si se hubiera dispuesto de todo el conocimiento adquirido al principio del proyecto, el resultado se hubiera visto mejorado.

6.4.2. Autenticación externa

Por medio de la librería `django-allauth`, se conoció un área nueva, importante en el desarrollo de una gran parte de las aplicaciones `WEB`, como es la autenticación. En este caso, se experimentó bastante con la autenticación por medio de servicios externos, en particular, la de `Google`, sobre el protocolo **OAuth2**.

Con esto, se pudo vincular exitosamente la plataforma a la autenticación de `Google`, logrando que los usuarios pudiesen acceder desde sus propias cuentas a la nueva plataforma, sin tener que memorizar otra clave.

En términos de aprendizaje, se pudo conocer en cierto nivel, los servicios que ofrece la `API` de la nube de `Google`, a la vez que se pudo presenciar, en simultáneo, la actualización de los protocolos de seguridad que se están migrando a **OAuth2**, y que en su momento fueron un pequeño problema, que pudo ser sorteado.

6.4.3. Infraestructura de la plataforma

Como se puede deducir de los puntos anteriores, en el apartado de la infraestructura de la plataforma se aprendió de igual manera, pasando por tecnologías como `Nginx`, `gunicorn`, entre otras. Estos elementos son vitales para el despliegue de una aplicación `WEB`, y se experimentaron desde perspectivas, la clásica, correspondiente a la puesta en marcha de cada servicio, en una infraestructura determinada, con sus respectivas configuraciones, y a través de contenedores.

De esta manera, se hizo una pasada, casi pedagógica, desde las formas fundamentales, hasta lo mas actual, con lo cual se pudo reforzar el conocimiento previo, que se limitaba en gran medida a lo teórico.

6.4.4. `AWS`

Dada la intención de experimentar con los desarrollos en la nube, se hicieron los principales experimentos y despliegues en una instancia gratuita de `Amazon Web Services (AWS)`, con lo cual se pudo experimentar los `deployments` en servidores, sin interfaz gráfica, por medio de conexiones `SSH`. Con esto se pudo tener una experiencia realista de puesta en marcha de una aplicación de estas características, a la vez que por medio del uso de repositorios `GitHub`, se hicieron actualizaciones sobre estos despliegues de manera dinámica.

6.4.5. `Docker`

El uso de contenedores, no estaba previsto desde un comienzo, pero dado el cambio de infraestructuras, y las migraciones experimentadas y por experimentar, se decidió la utilización de contenedores, en particular, los ofrecidos por `Docker`.

Si bien, al comienzo, se tenía cierta reticencia al uso de esta tecnología, dado que se desconocía el uso de recursos de esta, luego se decidió experimentar con esta, dadas las contantes migraciones de la aplicación. De esta manera, se obtuvieron muy buenos resultados, dado que `Docker` a través de `Docker Compose`, permite orquestar múltiples contenedores, a la vez que los depliegues se pueden automatizar completamente, con lo cual se pudo hacer una aplicación autocontenida, de despliegue automático, y resiliente, ya que `Docker` cuenta con la reinicialización de contenedores en caso de fallo lo cual es otra ventaja de esta tecnología.

De esta manera, se considerará el uso de `Docker` en proyectos posteriores, tomando en cuenta que es, básicamente, un estándar de la industria en la actualidad.

6.5. Trabajo futuro

Esta sección es bastante importante debido a que, si bien el proyecto no pudo concluirse de la mejor manera, de todas formas podrían realizarse mejoras posteriormente para un desarrollo completo, a la vez que lo que ya se encuentra desarrollado funcionaría como un buen producto viable e incrementable, debido a la cantidad de trabajo realizado y al esmero en seguir buenas practicas de desarrollo.

En lo particular, existen dos ítems fundamentales en lo que respecta a trabajo futuro, los cuales corresponden a la implementación del sistema de gestión de archivos, y la optimización de la búsqueda de postulantes.

Para el primero, dado que solo se experimentó con la plataforma de gestión de documentos, esta debería enlazarse de manera apropiada a la plataforma por medio de su `API`.

Para el segundo, dada que la búsqueda se realiza por medio de la `ORM` de `Django`, podría experimentar ciertas latencias, con lo cual, debería ser necesario un análisis, de las consultas, y si es necesario, una implementación directamente en `SQL`, permitida por `Django`.

Bibliografía

[1] NGINX org. Nginx. https://nginx.org/en/.

[2] WSGI org. What is WSGI? https://wsgi.readthedocs.io/en/latest/what.html, 2006.

[3] Raymond Penners. Django-allauth. https://django-allauth.readthedocs.io/en/latest/, 2017.

[4] Roberto Rosario. Mayan-edms documentation. https://docs.mayan-edms.com/, 2011.

Anexo

Base de datos

Figura 1: Diagrama de base de datos de la aplicación, parte 1.

Figura 2: Diagrama de base de datos de la aplicación, parte 2.