



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Diseño y desarrollo de una Plataforma para facilitar el Análisis de Puntuaciones de Riesgo Poligénico

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Àngela Signes Bolufer

Tutor: Alberto García Simón
Diana Martínez Minguet
Oscar Pastor López

Curso 2024-2025

Resum

La genòmica està revolucionant la medicina de precisió, permetent una avaluació més personalitzada del risc de desenvolupar diverses malalties. En aquest context, els Polygenic Risk Scores (PRS) han emergit com a estratègia clau per a estimar la predisposició genètica a patologies complexes basades en la combinació de múltiples variants genètiques.

No obstant això, la majoria de les eines actuals per a l'anàlisi de PRS presenten barres significatives per a la seua adopció per part de professionals no especialitzats, ja que requereixen coneixements avançats de bioinformàtica i programació, a més de generar informes tècnics la interpretació dels quals pot resultar desafiadora.

Aquest Treball de Fi de Grau té com a objectiu el disseny i desenvolupament d'una plataforma web accessible i intuïtiva que facilite l'anàlisi de PRS per a un públic més ampli. La plataforma permetrà als usuaris carregar arxius genòmics, configurar l'anàlisi mitjançant models PRS predefinitos i visualitzar els resultats de forma clara i comprensible. Es prioritzarà una experiència d'usuari optimitzada, amb informes gràfics i descriptius que simplifiquen la interpretació dels resultats, fomentant la seua aplicació tant en la pràctica clínica com en la investigació.

Mitjançant aquesta solució, es busca eliminar les barreres tecnològiques que actualment limiten l'accés a les anàlisis de PRS, contribuint a la democratització de la genòmica en l'àmbit sanitari. La plataforma facilitarà la integració de la informació genètica en la presa de decisions mèdiques, impulsant l'ús de la medicina de precisió en la prevenció, diagnòstic i tractament de malalties.

Paraules clau: Genòmica, Puntuacions de Risc Poligènic, Plataforma web, Medicina de precisió

Resumen

La genómica está revolucionando la medicina de precisión, permitiendo una evaluación más personalizada del riesgo de desarrollar diversas enfermedades. En este contexto, los Polygenic Risk Scores (PRS) han emergido como estrategia clave para estimar la predisposición genética a patologías complejas basadas en la combinación de múltiples variantes genéticas.

No obstante, la mayoría de las herramientas actuales para el análisis de PRS presentan barreras significativas para su adopción por parte de profesionales no especializados, ya que requieren conocimientos avanzados de bioinformática y programación, además de generar reportes técnicos cuya interpretación puede resultar desafiante.

Este Trabajo de Fin de Grado tiene como objetivo el diseño y desarrollo de una plataforma web accesible e intuitiva que facilite el análisis de PRS para un público más amplio. La plataforma permitirá a los usuarios cargar archivos genómicos, configurar el análisis mediante modelos PRS predefinidos y visualizar los resultados de forma clara y comprensible. Se priorizará una experiencia de usuario optimizada, con reportes gráficos y descriptivos que simplifiquen la interpretación de los resultados, fomentando su aplicación tanto en la práctica clínica como en la investigación.

Mediante esta solución, se busca eliminar las barreras tecnológicas que actualmente limitan el acceso a los análisis de PRS, contribuyendo a la democratización de la genómica en el ámbito sanitario. La plataforma facilitará la integración de la información genética en la toma de decisiones médicas, impulsando el uso de la medicina de precisión en la prevención, diagnóstico y tratamiento de enfermedades.

Palabras clave: Genómica, Puntuaciones de Riesgo Poligénico, Plataforma web, Medicina de precisión

Abstract

Genomics is revolutionizing precision medicine, allowing for a more personalized assessment of the risk of developing various diseases. In this context, Polygenic Risk Scores (PRS) have emerged as a key strategy to estimate the genetic predisposition to complex pathologies based on the combination of multiple genetic variants.

However, most current tools for the analysis of PRS present significant barriers to their adoption by non-specialized professionals, as they require advanced knowledge of bioinformatics and programming, in addition to generating technical reports whose interpretation can be challenging.

The objective of this Final Degree Project is the design and development of an accessible and intuitive web platform that facilitates the analysis of PRS for a wider audience. The platform will allow users to upload genomic files, configure the analysis using pre-defined PRS models, and visualize the results in a clear and understandable way. An optimized user experience will be prioritized, with graphical and descriptive reports that simplify the interpretation of the results, promoting its application in both clinical practice and research.

Through this solution, the aim is to eliminate the technological barriers that currently limit access to PRS analysis, contributing to the democratization of genomics in the healthcare field. The platform will facilitate the integration of genetic information into medical decision-making, driving the use of precision medicine in the prevention, diagnosis, and treatment of diseases.

Key words: Genomics, Polygenic Risk Scores, Web platform, Precision medicine

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

1.1 Introducción

¿Qué determina el color de los ojos, la estatura o incluso la predisposición a ciertas condiciones de salud? La respuesta reside en un manual de instrucciones que se encuentra en cada una de nuestras células: nuestro ADN¹. El ADN está compuesto por aproximadamente 3.000 millones de "letras" químicas, una secuencia tan extensa que, si se imprimiera, formaría una torre de papel de más de 100 metros de altura[?]. El campo que se dedica a descifrar y a entender cómo se transmiten sus instrucciones entre generaciones es la genética.

Dentro de este inmenso texto genético, los genes actúan como los capítulos; estos son fragmentos específicos de ADN que contienen la información necesaria para manifestar un rasgo biológico, desde la apariencia física hasta el comportamiento. Estos genes se organizan en estructuras mayores llamadas cromosomas, que no solo los contienen, sino que también interactúan con el entorno y se ven influenciados por nuestro estilo de vida para orquestrar el complejo funcionamiento del organismo[?].

A pesar de que los seres humanos compartimos más del 99 % del ADN, es en ese pequeño porcentaje, el cual está compuesto por lo que se llaman variantes genéticas, donde reside la variabilidad individual. Las variantes genéticas son diferencias en la secuencia del ADN que distinguen a los individuos entre sí y son responsables del color de ojos o la estatura (características fenotípicas). Sin embargo, además de su efecto en la apariencia, algunas de estas variantes pueden alterar el funcionamiento de ciertos genes, modificando la probabilidad de desarrollar determinadas enfermedades. Por tanto, la identificación y el estudio de estas variaciones permiten comprender mejor la base genética de muchas patologías e identificar riesgos hereditarios a determinadas enfermedades.

Las patologías o enfermedades con base genética se pueden diferenciar en simples y complejas, en función del número de genes afectados. Por un lado, las enfermedades simples son más sencillas de diagnosticar y tratar dado que dependen de la alteración en un único gen, la cual tiene un efecto directo y a menudo predecible. Por otro lado, las enfermedades complejas, que constituyen la mayoría de las afecciones comunes, presentan una complejidad mucho mayor por la combinación de múltiples variaciones[?]. Esto dificulta el estudio de su patrón genético. Un ejemplo de esta complejidad se observa en las distrofias de retina, donde variaciones en más de 250 genes distintos pueden dar lugar a una patología similar, con un patrón genético que varía significativamente de un paciente a otro.

¹Ácido Desoxirribonucleico

Adicionalmente, el desarrollo de las enfermedades complejas no depende únicamente del factor genético, sino que también está fuertemente influenciado por el estilo de vida y los factores medioambientales. Como resultado, el componente genético por sí solo no es un factor determinante, sino que el efecto agregado de numerosas variaciones genéticas aumenta la predisposición a padecer la enfermedad [?].

Su diagnóstico y prevención son de gran relevancia, ya que no solo mejora la calidad de vida de los pacientes, sino que también representa un enfoque más sostenible y costo-efectivo, ayudando a reducir la considerable sobrecarga económica y de recursos que soportan los sistemas de salud.

Para abordar esta complejidad, la investigación genómica ha desarrollado herramientas estadísticas que permiten agrupar y cuantificar el efecto de estas variantes, como los modelos de Puntuación de Riesgo Poligénico (PRS, del inglés *Polygenic Risk Score*)[?]. Estos modelos estadísticos son capaces de agregar la información de todas estas pequeñas variantes para calcular una estimación del riesgo genético global de un individuo para una enfermedad específica, convirtiéndose en una herramienta prometedora para la medicina preventiva y personalizada.

1.2 Motivación

Como se ha mencionado anteriormente, en el campo de la medicina se está observando que los análisis de PRS tienen potencial para mejorar la predicción del riesgo a padecer una enfermedad, incluso cuando ya se consideran factores tradicionales como el historial familiar o riesgos derivados del estilo de vida. Además, estos permiten la predicción de múltiples enfermedades en una sola prueba, por lo que se busca la introducción de estos análisis para complementar los ya existentes.

A pesar de su gran potencial clínico, la integración de las Puntuaciones de Riesgo Poligénico (PRS) en la práctica clínica enfrenta importantes desafíos, ya que la transición de una herramienta de investigación a una aplicación clínica estándar requiere de *software* que no solo sea preciso, sino también accesible y eficiente para el personal sanitario.

Los artefactos existentes presentan un grado de dificultad elevado comparado con el conocimiento medio que pueda tener una persona no experta en el dominio de la investigación. Por ejemplo, la mayoría de estas herramientas requieren el uso de la línea de comandos (*CLI*) y la configuración manual de ficheros de texto. Esto supone una barrera significativa para la implementación de esta tecnología en un contexto clínico, ya que limita su adopción por parte de profesionales que no disponen de formación técnica avanzada.

Por ello, en este trabajo se plantea el desarrollo de un *software* que permita a los usuarios no expertos utilizar estas potentes herramientas bioinformáticas. Este objetivo se alcanza reduciendo la carga tecnológica para los usuarios no expertos que potencialmente puedan beneficiarse de esta herramienta. Teniendo una interfaz más amigable para el usuario y con un grado de información más centralizado, permitirá a estos usuarios ejecutar los pasos necesarios para completar un análisis PRS de forma intuitiva.

Formar parte de este proyecto ha supuesto una valiosa oportunidad para colaborar en una línea de trabajo consolidada y desarrollada. Esta iniciativa ha permitido aplicar los conocimientos adquiridos durante el grado para abordar los retos reales a los que se enfrentan los investigadores en el campo de la genética y la medicina. La aportación de este Trabajo de Fin de Grado se enmarca dentro de un objetivo más general y de alto valor añadido: desarrollar herramientas sencillas y accesibles que tengan el potencial de marcar una diferencia significativa en la vida de los pacientes.

1.3 Objetivos

El objetivo general de este trabajo de fin de grado es **facilitar la ejecución de análisis PRS en contextos clínicos**. Para alcanzar este objetivo, se llevará a cabo el desarrollo de una plataforma web accesible que facilite el análisis PRS para su posterior aplicación. Para la consecución de este objetivo general, se han definido tres objetivos específicos (OE).

- OE1** Estudiar el contexto: Identificar los fundamentos genéticos, estadísticos y clínicos que sustentan el uso de análisis PRS en la predicción del riesgo de enfermedades complejas. También se analizarán las principales herramientas existentes para llevar a cabo análisis PRS para identificar las limitaciones que presentan. De esa forma, el conocimiento adquirido y las posibilidades de mejora sirven para llevar a cabo un desarrollo más alineado con las necesidades que existen hacia la plataforma.
- OE2** Crear una plataforma web: Desarrollar una plataforma sencilla y accesible que permita el análisis de PRS, incluso para quienes no tienen experiencia en bioinformática. Posteriormente, se estudiarán las herramientas ya existentes y se determinará la complejidad de las herramientas actuales y la necesidad de diseñar una solución que no requiera conocimientos técnicos avanzados y sea sencilla e intuitiva.
- OE3** Validar el sistema: Realizar pruebas para validar y comprobar que el sistema funciona como se espera: se recogerán opiniones de profesionales para entender si la herramienta se adapta a sus necesidades.

1.4 Metodología

La metodología de trabajo empleada en este proyecto es la Ciencia del Diseño (*Design Science*). Este enfoque se basa en el marco metodológico definido por Roel J. Wieringa[?], al cual se hará referencia a lo largo de este apartado.

La metodología *design science* tiene como objetivo principal la búsqueda de soluciones a problemas del mundo real. Más concretamente, se centra en el diseño e investigación de artefactos en un contexto.

En este trabajo, el **artefacto** es una plataforma web interactiva que facilita los análisis de *Polygenic Risk Scores* y el **contexto** es el estudio genético de las enfermedades complejas.

El marco de trabajo de Wieringa distingue dos niveles de actuación en la *design science methodology*: el ciclo de ingeniería y el ciclo de diseño. El ciclo de diseño es un subproceso del ciclo de ingeniería, siendo este último el proceso completo que va desde el estudio de un problema hasta la implantación y evaluación de su solución en el mundo real.

Sin embargo, los proyectos de investigación académica como este TFG se centran en las primeras tres fases del ciclo de ingeniería, las cuales constituyen el ciclo de diseño. La razón de este enfoque es que el objetivo principal no es construir una solución con un grado de robustez y escalabilidad de nivel comercial, sino diseñar y validar un artefacto novedoso como prueba de concepto.

Por lo tanto, este ciclo se enfoca en el diseño riguroso del artefacto y, fundamentalmente, en la justificación argumentada de que dicho artefacto, si llegara a implementarse en un entorno real, contribuiría a los objetivos de los interesados. El presente trabajo ejecutará un ciclo de diseño completo, tal y como se ilustra en la Figura ??



Figura 1.1: Ciclo de Diseño.

El ciclo de diseño se compone de tres fases principales: la investigación del problema, el diseño del tratamiento y la validación del tratamiento. Este conjunto de actividades se denomina ciclo de diseño porque los investigadores suelen iterar repetidamente sobre estas fases a lo largo de un proyecto de investigación basado en *Design Science*. A continuación, se explican las tres fases:

1. Investigación del problema (*Problem Investigation*): Se analiza en profundidad el contexto y los elementos clave del problema que se quiere abordar. En esta fase se identifican los *stakeholders* y se definen los objetivos, así como los efectos y consecuencias que puede tener el desarrollo respecto a dichos objetivos. Además, se lleva a cabo un análisis del estado del arte, evaluando soluciones existentes y enfoques previos, lo que permite comprender mejor las limitaciones actuales y justificar la necesidad de una nueva propuesta.
2. Diseño de la solución (*Treatment Design*): En esta fase se genera la solución para dar respuesta a las limitaciones identificadas en la etapa anterior. Para ello, se lleva a cabo un proceso que abarca el diseño e implementación de la solución.
3. Validación de la solución (*Treatment Validation*): Se comprueba que la implementación del artefacto ha cumplido con los objetivos iniciales en la práctica. Algunas de las preguntas ligadas a esta fase se plantean el efecto real del tratamiento, la aceptación por parte de los usuarios y su eficacia. En definitiva, la validación permite

detectar posibles limitaciones o áreas de mejora, lo que puede dar lugar a nuevas iteraciones sobre el diseño de la solución.

1.4.1. Preguntas de investigación

Para guiar el trabajo dentro de cada una de estas fases, se ha formulado una serie de preguntas de investigación. Estas preguntas están diseñadas para orientar el proceso de desarrollo, desglosar el problema y asegurar que la solución propuesta responde a las necesidades identificadas. Las preguntas han sido clasificadas en función de cada una de las fases metodológicas.

Fase 1: Investigación del problema

1. ¿Cuál es la base de genética que sustentan los análisis PRS?
2. ¿Qué herramientas tecnológicas dan soporte a la ejecución de análisis PRS?
3. ¿Cuáles son las principales barreras y limitaciones que enfrentan los usuarios de estas herramientas tecnológicas?

Fase 2: Diseño de la solución

1. ¿Qué requisitos debe cumplir una plataforma web para superar las barreras identificadas y facilitar el análisis PRS?
2. ¿Cuál es la arquitectura tecnológica más adecuada para esta plataforma?
3. ¿Cómo optimizar la carga, procesamiento y análisis de modelos PRS?

Fase 3: Validación de la solución

1. ¿En qué medida la plataforma desarrollada cumple con los requisitos establecidos para superar las barreras identificadas en el análisis PRS?
2. ¿Cuál es el nivel de usabilidad y utilidad percibida por los usuarios finales al interactuar con la plataforma?

1.4.2. Stakeholders

En este apartado se busca identificar los distintos grupos de interés del proyecto o *stakeholders*. Según la metodología *design science* (explicada en la sección ??), un *stakeholder* es una persona, grupo de personas o institución que se ve afectada por el tratamiento de un problema. Identificarlos correctamente es fundamental, ya que de ellos surgen tanto los objetivos como las restricciones del proyecto que, a su vez, son la base para definir los requisitos que debe cumplir la solución desarrollada.

Depende del tipo de problema, puede que los *stakeholders* no sepan que lo son aún, por ejemplo, en los proyectos exploratorios, ya que los objetivos no están bien definidos al inicio de estos. En cambio, como en el caso de este proyecto, en los guiados por la utilidad, existen partes interesadas con objetivos definidos a los que la investigación debe contribuir.

Se pueden encontrar diferentes tipos de *stakeholders*; siendo identificados en este trabajo los siguientes para este proyecto:

- Investigadores en el ámbito de la genómica, más concretamente en análisis PRS. Serían los principales beneficiarios del desarrollo de esta plataforma, ya que son los que tienen unos objetivos claros acerca de la expectativa del resultado.
- Profesionales sanitarios como genetistas clínicos y asesores genéticos. A través de esta herramienta, podrían integrar los análisis PRS en su práctica clínica, gracias a que la plataforma ha sido diseñada para facilitar su uso por parte de profesionales sin conocimientos técnicos avanzados previos.
- Pacientes, quienes, aun sin interactuar directamente con la plataforma, se benefician de su uso por parte de profesionales clínicos, mejorando la prevención de enfermedades y diagnósticos. Su atención clínica en general se ve favorecida.
- Entidades colaboradoras, como centros de investigación, universidades, hospitales o biobancos, que podrían incorporar esta solución como parte de su infraestructura tecnológica.

De forma directa o indirecta, este grupo de personas o entidades se podría ver beneficiado por el desarrollo de la plataforma *Phenoscore*.

1.4.3. Aplicación de la metodología

La aplicación de la metodología *Design Science* se plasma en la organización del documento: cada fase metodológica encuentra su correspondencia en uno o varios capítulos, de modo que las cuestiones planteadas se van resolviendo de forma progresiva y coherente a lo largo de la memoria.

La primera fase, llamada investigación del problema, se ha resuelto en el Capítulo ?? (??).

- **Capítulo ??.** ??: Se encarga de introducir y contextualizar el problema. También se presenta la motivación del proyecto, sus objetivos, la metodología empleada y los *stakeholders* identificados.
- **Capítulo ??.** ??: En este capítulo se da respuesta a las preguntas de investigación de dicha fase, analizando los fundamentos de los análisis PRS y se analizan los artefactos existentes que han dado algún tipo de solución a la ejecución de análisis PRS o a alguna fase de su configuración y se comparan esas soluciones con la propuesta presentada.

La segunda fase, llamada diseño de la solución, se ha resuelto en el capítulo ?? (??).

- **Capítulo ??.** ??: en la sección ?? se han detectado las oportunidades de mejora y se han definido las características y funcionalidades necesarias para el desarrollo a través de los requisitos como: historias de usuario, requisitos funcionales y no funcionales. Posteriormente, se ha realizado el modelado conceptual junto con las posibles soluciones identificadas y la solución propuesta.

Por lo que se refiere a la tecnología y el diseño de la interfaz, ambas cuestiones se han resuelto en la sección ??; se han descrito las herramientas que se van a usar junto con la estructura de la solución. Además, a través de los *mockups* se ha representado la idea del diseño visual de la aplicación.

Por otro lado, en la sección ?? se ha llevado a cabo la implementación de las funcionalidades anteriormente identificadas en el análisis.

La última fase, llamada validación de la solución, se ha resuelto en el capítulo ?? (??).

- **Capítulo ??.** ??: se detalla la metodología de validación, los participantes, el procedimiento seguido y los instrumentos de medición empleados. Se presentan los resultados, incluyendo la validación funcional, la evaluación de requisitos no funcionales, la aceptación tecnológica y la matriz de trazabilidad. Además, se analiza si la solución ha resultado satisfactoria respecto a las expectativas y necesidades de los *stakeholders*.

Por último, se han presentado las conclusiones del trabajo en el capítulo ?? (??).

- **Capítulo ??.** ??: se sintetizan los resultados del trabajo, se establecen relaciones con las competencias del grado y se señalan posibles mejoras y líneas de evolución futura.

CAPÍTULO 2

Investigación del problema

2.1 Fundamentos de las puntuaciones de riesgo poligénico

Las Puntuaciones de Riesgo Poligénico (PRS) son una métrica que cuantifica la predisposición genética de un individuo a un rasgo o enfermedad, basándose en la suma de los efectos de múltiples variantes genéticas. Para comprender su construcción y aplicación, es fundamental entender los conceptos genéticos y estadísticos en los que se basan. Esta sección da respuesta a la primera pregunta de investigación: ¿Cuál es la base de genética que sustentan los análisis PRS?

El material de partida para cualquier PRS son los resultados de los Estudios de Asociación del Genoma Completo (GWAS, del inglés *Genome Wide Association Study*) [?, ?]. Estos estudios a gran escala analizan los genomas de miles de individuos para identificar Polimorfismos de un Solo Nucleótido (SNPs, del inglés *Single Nucleotide Polymorphism*) [?] que están estadísticamente asociados con una enfermedad. Para cada SNP que supera un umbral de significancia estadística, el GWAS proporciona un tamaño del efecto (β_j), que refleja la magnitud de su asociación con la patología. Estos tamaños de efecto se convierten en los pesos que se utilizarán como parámetros en el modelo PRS.

La construcción de un modelo PRS a partir de los datos brutos de un GWAS es un proceso bioinformático complejo que va más allá de una simple suma. Antes de aplicar la fórmula, se deben realizar varios pasos metodológicos cruciales para asegurar la validez del modelo, como la selección de SNPs mediante un umbral de p-valor (*p-value thresholding*) y la poda de variantes en desequilibrio de ligamiento (*LD clumping*).

Este último paso es esencial, ya que el desequilibrio de ligamiento se refiere al fenómeno por el cual variantes genéticas que están físicamente cercanas en un cromosoma tienden a heredarse juntas. La poda, por tanto, elimina las variantes redundantes, asegurando que los SNPs incluidos en el modelo sean en gran medida independientes entre sí y evitando así una sobreestimación del riesgo [?].

Una vez se dispone del conjunto final de SNPs y sus pesos, la puntuación para un individuo i se calcula mediante un modelo de regresión lineal aditivo:

$$PRS_i = \sum_{j=1}^n G_{ij} \beta_j$$

donde β_j representa el efecto de cada variante, G_{ij} el número de copias de la variante que posee el individuo ¹, y n el número total de variantes asociadas [?, ?]. Al aplicar

¹El valor puede ser 0, 1 o 2. Esto se debe a que los seres humanos heredan dos copias de cada cromosoma, una de cada progenitor. Así, para un SNP concreto, un individuo puede no haber heredado ninguna copia del alelo de riesgo, haber heredado una o haber heredado ambas.

este modelo al genoma de un individuo, se obtiene un valor numérico que estima su predisposición genética.

El valor bruto de un PRS no es directamente interpretable. Para obtener un significado clínico, debe ser contextualizado comparándolo con las puntuaciones de una gran población de referencia, lo que permite situar al individuo en un percentil de riesgo relativo (ej. "en el 5 % superior de riesgo genético") [?, ?].

La principal utilidad clínica del PRS es la estratificación del riesgo para la medicina preventiva. Un ejemplo de alto impacto es en las enfermedades cardiovasculares. Un individuo puede tener factores de riesgo tradicionales (colesterol, presión arterial) en un rango intermedio, pero un PRS elevado puede reclasificarlo a una categoría de alto riesgo, justificando el inicio de un tratamiento preventivo (como estatinas) mucho antes de lo que se haría de otra manera [?]. De este modo, el PRS no reemplaza los factores de riesgo clásicos, sino que los complementa, añadiendo una nueva capa de información para una toma de decisiones más precisa [?].

A pesar de su gran potencial, la implementación clínica de los PRS enfrenta desafíos significativos. Uno de los más importantes es la portabilidad entre diferentes poblaciones ancestrales. La gran mayoría de los GWAS se han realizado en individuos de ascendencia europea, lo que significa que los modelos PRS derivados de ellos son significativamente menos precisos cuando se aplican a individuos de ascendencia africana, asiática u otras. Esta falta de diversidad en los datos de entrenamiento es una barrera crítica para una implementación equitativa de la genómica en la salud [?].

Además, la comunicación de un riesgo probabilístico y genético a los pacientes es un desafío en sí mismo, requiriendo un asesoramiento genético cuidadoso para evitar el determinismo genético y la ansiedad innecesaria [?].

En conclusión, los análisis de PRS representan un avance fundamental en la medicina genómica. Aunque su implementación a gran escala aún enfrenta retos metodológicos y éticos, su potencial para guiar intervenciones preventivas y personalizar el tratamiento es innegable, marcando el camino hacia una práctica clínica más proactiva y precisa [?].

2.2 Estado del arte

Como se ha mencionado en la introducción, los *Polygenic Risk Scores* (PRS) se han convertido en una herramienta prometedora en la medicina de precisión al permitir estimar el riesgo genético de un individuo para desarrollar enfermedades complejas. Los PRS representan un avance significativo respecto a los enfoques tradicionales que se centraban en el estudio de enfermedades simples, donde la causa de la enfermedad radica en una única variante.

Para realizar un análisis de PRS se requiere un modelo PRS para la enfermedad de estudio y los datos genómicos del individuo de interés. Hay múltiples pasos involucrados en el proceso completo, algunos de esos pasos y las herramientas que se usan en ellos se pueden observar en la siguiente tabla:

Paso del Proceso	Herramienta(s) Utilizada(s)	Referencias
Control de calidad de los <i>datasets</i> y variaciones	<ul style="list-style-type: none"> • PLINK (es el estándar para el filtrado estadístico). • BCFtools (muy potente para la manipulación y filtrado de ficheros VCF). 	PLINK QC Manual, BCFtools Documentation
Alineamiento entre datos del paciente y del modelo	PLINK (se usa para armonizar los <i>datasets</i> , asegurando que las variantes coincidan).	PLINK PCA Tutorial
Estimación y ajuste de ancestría del paciente	PLINK (se utiliza para realizar el Análisis de Componentes Principales o PCA).	AncestryCheck with plinkQC
Preparación de la población de referencia	<ul style="list-style-type: none"> • PLINK y BCFtools (para el control de calidad). • bgenix (para indexar y acceder a datos masivos). 	BGEN Documentation, BGENIX Documentation
Creación del Modelo PRS y Cálculo de la Puntuación	<ul style="list-style-type: none"> • PRSice2 (implementa el método de "Clumping + Thresholding"). • LDpred2 (implementa un método Bayesiano más avanzado). 	PRSice-2: Polygenic Risk Score software, LDpred2: better, faster, stronger

Tabla 2.1: Pasos, herramientas y referencias en el análisis PRS.

Como resultado, para poder ejecutar un análisis de PRS se deben combinar múltiples herramientas informáticas.

Es por ello que la aplicación de análisis PRS en entornos clínicos se enfrenta a una barrera fundamental: la alta complejidad y fragmentación del proceso. Como se ha mencionado, un análisis completo requiere una secuencia de pasos bien definidos y cada una de estas tareas suele ser ejecutada por herramientas informáticas especializadas y potentes, como PLINK o BCFtools.

El primer problema radica en que la integración y orquestación manual de estas herramientas exige un conocimiento técnico avanzado en bioinformática, creando un flujo de trabajo fragmentado que resulta inaccesible para la mayoría de los investigadores clínicos. Debido a esta limitación, el paso lógico es buscar soluciones que ya integren estos pasos en un único *pipeline* automatizado.

Sin embargo, esto lleva a un segundo problema: la mayoría de estas soluciones carecen de una interfaz de usuario (UI) intuitiva y funcionan a través de la línea de comandos. Esta es la complejidad que este trabajo pretende simplificar: no solo integrar el flujo de trabajo, sino también presentarlo a través de una interfaz accesible que elimine la barrera tecnológica para el usuario final.

Dado que el objetivo de este trabajo es facilitar la ejecución de análisis PRS para usuarios no expertos, se han investigado herramientas que integran *software* existentes y automatizan muchos de los complejos pasos y decisiones de análisis. Las principales herramientas para ejecutar análisis de PRS son las siguientes:

- `pgsc_calc` (*Polygenic Score Catalog Calculator*) [?]: se trata de un *pipeline* Nextflow oficial del repositorio *PGS Catalog*, diseñado exclusivamente para realizar análisis de PRS usando modelos existentes en el repositorio, o personalizados. Automatiza el proceso de descargar modelos de *PGS Catalog* y aplicarlos a datos genéticos, en

formato *VCF*, *PLINK1* o *PLINK2*.² Genera como resultado un reporte resumen con los resultados del análisis, detalles del modelo aplicado y parámetros relevantes del proceso.

- *GenoPred pipeline* [?]: se trata de un *pipeline* escalable de *Snakemake* para generar y aplicar modelos PRS sobre datos genómicos, en varios formatos: *PLINK1*, *PLINK2*, *VCF*, *BGEN*, y *23andMe*. Al igual que *pgsc_calc*, permite realizar los análisis de PRS a partir de modelos PRS existentes, procedentes de repositorios de modelos como *PGS Catalog* u otros. Sin embargo, además permite crear modelos PRS para seguidamente aplicarlos, ya que la herramienta ha sido desarrollada para facilitar la generación de modelos PRS empleando distintos *softwares*. La salida principal de esta herramienta son archivos *HTML* con resultados de análisis PRS y reportes técnicos. Además de la obtención del reporte de análisis, permite obtener cálculos intermedios de interés, como control de calidad o estimación de ancestrías de los datos de entrada.
- *PGS Calculator* (*pgs-calc* de Lukasz) [?]: es una herramienta de línea de comandos para aplicar modelos PRS a datos genómicos en formato *VCF*. Se conecta a *PGS Catalog* para descargar automáticamente los archivos de puntuación de los modelos o también permite cargar archivos de modelos PRS propios, pero no la creación de nuevos modelos. La salida es un *HTML* interactivo con gráficos de distribución de PGS.
- *PRS Pipeline tutorial* (Collister & Liu) [?]: es un tutorial práctico que permite aplicar un modelo PRS existente a datos genómicos del repositorio *UK Biobank*. No es exactamente una herramienta integrada, pero ilustra todo el flujo desde la extracción de SNPs hasta el cálculo de PRS siguiendo un ejemplo concreto.

2.3 Principales barreras y limitaciones

La mayor utilidad de estas herramientas es que integran todos los pasos necesarios para ejecutar un análisis PRS completo. Sin embargo, su principal limitación es que no disponen de una interfaz que permita configurar fácilmente los análisis de PRS. El uso por línea de comandos, así como la instalación y el uso de herramientas de orquestación de *workflows* (*snakemake* o *nextflow*) y gestores de entornos de ejecución virtuales (*conda*), supone un reto para los usuarios no expertos. Adicionalmente, el “*PRS Pipeline tutorial*” tiene la limitación específica de no ser apto para análisis de individuos únicos, ya que está orientado a conjuntos de datos a gran escala como los de *UK Biobank*.

²*VCF* (*Variant Call Format*) es un formato de texto estándar para almacenar variaciones genéticas. *PLINK1* y *PLINK2* son los formatos nativos del *software* *PLINK*, que utilizan un conjunto de archivos binarios (*.bed*, *.bim*, *.fam*) para una gestión eficiente de grandes *datasets* genómicos.

Herramienta	Falta de Interfaz Gráfica (CLI)	Requiere Conocimiento Técnico Avanzado	Solo Aplica Modelos Existentes (No Creación)	No Apto para Análisis Individual
pgsc_calc	Sí	Sí (Nextflow)	Sí	No
GenoPred pipeline	Sí	Sí (Snakemake, Conda)	No	No
PGS Calculator	Sí	Sí (CLI)	Sí	No
PRS Pipeline tutorial	Sí	Sí (Tutorial práctico)	Sí	Sí

Tabla 2.2: Limitaciones clave de las herramientas para análisis de PRS.

El análisis PRS para predecir la predisposición genética a enfermedades complejas se ve significativamente limitado por su inherente complejidad técnica. El proceso, que abarca desde la descarga y preprocesamiento de datos genéticos hasta la ejecución de flujos de trabajo específicos para el cálculo y la interpretación de resultados, requiere un dominio avanzado de herramientas bioinformáticas y conocimientos especializados. Estas barreras técnicas restringen considerablemente el acceso y la adopción de los PRS por parte de investigadores con menor experiencia en bioinformática y profesionales del ámbito clínico.

A partir del estudio del estado del arte, se han identificado problemas recurrentes en las soluciones actuales: una elevada complejidad que dificulta el acceso y aprovechamiento, una fragmentación de funcionalidades que obliga al uso de múltiples plataformas, y la entrega de resultados en formatos planos que carecen de visualizaciones intuitivas para una rápida interpretación. Estos desafíos recalcan la necesidad de una innovación que centralice el proceso en una única herramienta, ofreciendo una experiencia más amigable y accesible que integre todas las fases del análisis PRS, desde la carga de datos genéticos hasta la interpretación visual de los resultados.

CAPÍTULO 3

Diseño de la solución

3.1 Especificación de requisitos

3.1.1. Historias de usuario (HU)

Una vez se ha analizado el problema y se han detectado las oportunidades de mejora, se deben definir las características y funcionalidades que se implementarán en la herramienta. A partir de las historias de usuario, que han sido recopiladas y refinadas a través de sesiones de trabajo colaborativas y entrevistas con los investigadores expertos en el dominio, utilizando técnicas de lluvia de ideas para explorar los diferentes escenarios de uso, se van a extraer los criterios de aceptación para verificar que la funcionalidad se define acorde a la idea inicial de los usuarios. Posteriormente, sobre esta base se formulan los requisitos funcionales y no funcionales que estructuran, de una forma detallada, los aspectos o funcionalidades que debe implementar la plataforma.

Tabla 3.1: Historias de usuario HU01–HU11

ID	Nombre	Historia de usuario	Criterio de aceptación
HU01	Inicio de sesión	Como usuario de la aplicación, quiero poder iniciar sesión con mis credenciales, para acceder de forma segura a mi cuenta personal y a todos mis análisis.	Dado que estoy en la pantalla de inicio de sesión, cuando introduzco mis credenciales y presiono el botón de inicio de sesión, entonces me muestra los análisis que he hecho previamente.
HU02	Panel de control	Como usuario de la aplicación, quiero tener acceso a un panel de control centralizado, para visualizar de un vistazo el estado y progreso de todos mis análisis PRS pendientes y completados.	Dado que he iniciado sesión, cuando entro a la pantalla principal de la <i>app</i> , entonces se muestra un panel con los estados y progreso de todos mis análisis.

Continues on the next page

Continues from previous page

ID	Nombre	Historia de usuario	Criterio de aceptación
HU03	Crear nuevo análisis PRS	Como usuario de la aplicación, quiero crear un nuevo análisis PRS para estimar la predisposición genética a enfermedades específicas de mis pacientes.	Dado que estoy en la pantalla principal, cuando presiono el botón de nuevo análisis, entonces me lleva a la pantalla de creación de análisis PRS.
HU04	Gestión de análisis PRS	Como usuario de la aplicación, quiero gestionar mis análisis PRS, para visualizar, ordenar o eliminar estudios según necesite.	Dado que he iniciado sesión y estoy en el panel de control, cuando selecciono un análisis, entonces me muestra opciones de visualizarlo, ordenarlo o eliminarlo.
HU05	Subida de archivos genéticos	Como usuario de la aplicación, quiero subir archivos <i>VCF/PLINK1/PLINK2/23andME</i> , para utilizarlos posteriormente en los análisis de PRS.	Dado que estoy en la pantalla de creación de análisis PRS, cuando presiono el botón para cargar <i>datasets</i> , entonces me deja seleccionar mis propios <i>datasets</i> en formato <i>VCF/PLINK1/PLINK2/23andME</i> .
HU06	Selección de modelos PRS	Como usuario de la aplicación, quiero seleccionar una agrupación de modelos PRS para aplicarlos al paciente.	Dado que estoy en la pantalla de creación de análisis PRS, cuando selecciono uno o varios modelos de la lista, entonces me deja asignarlos a mi análisis.
HU07	Configuración del análisis PRS	Como usuario de la aplicación, quiero configurar los parámetros del análisis PRS, para personalizar el estudio.	Dado que estoy en la pantalla de creación de análisis PRS, cuando selecciono y ajusto diferentes parámetros, entonces se quedan guardados y me deja ejecutar el análisis con ellos.
HU08	Ejecución del análisis	Como usuario de la aplicación, quiero iniciar un análisis de PRS con un solo clic después de la configuración, para obtener resultados sobre la predisposición genética de mis pacientes.	Dado que ya he configurado mi análisis, cuando presiono el botón de iniciar análisis, entonces la <i>app</i> empieza la ejecución de mi análisis PRS, mostrando su progreso.

Continues on the next page

Continues from previous page

ID	Nombre	Historia de usuario	Criterio de aceptación
HU09	Visualización de resultados	Como usuario de la aplicación, quiero visualizar los resultados para facilitar la interpretación del riesgo de mis pacientes.	Dado que un análisis ha finalizado y en la pantalla se muestra como completado, cuando lo selecciono y le doy al botón de "view", entonces me muestra una pantalla con el resultado.
HU10	Exportación de resultados	Como usuario de la aplicación, quiero exportar los resultados para incluirlos en otros sistemas.	Dado que un análisis ha finalizado y en la pantalla se muestra como completado, cuando lo selecciono y le doy al botón de exportar, entonces, se genera y exporta en formato PDF.
HU11	Notificaciones al finalizar análisis	Como usuario de la aplicación, quiero recibir notificaciones cuando se complete un análisis, para gestionar eficientemente mi tiempo.	Dado que he ejecutado un análisis, cuando este finaliza, entonces recibo una notificación en el correo y un <i>pop-up</i> en la <i>app</i> avisando que el análisis ha concluido.

A partir de los criterios de aceptación formulados, se han definido los siguientes requisitos funcionales y no funcionales, los cuales se clasifican a su vez por nivel de prioridad.

3.1.2. Requisitos funcionales (RF)

A partir de los criterios de aceptación de las historias de usuario, se han derivado los requisitos funcionales (RF). Estos requisitos traducen las necesidades del usuario en especificaciones concretas sobre lo que el sistema debe hacer. Describen las funciones, tareas y comportamientos específicos que la plataforma debe ser capaz de ejecutar para satisfacer las historias de usuario definidas.

Tabla 3.2: Requisitos funcionales RF-001 – RF-019

ID	Nombre	Descripción	Prioridad	Dependencias	Historia de usuario
RF-001	Inicio de sesión	El sistema debe permitir a los usuarios iniciar sesión con sus credenciales.	Media	–	HU01

Continues on the next page

Continues from previous page

ID	Nombre	Descripción	Prioridad	Dependencias	Historia de usuario
RF-002	Panel de control	El sistema debe proporcionar un panel de control de estados de sus análisis.	Alta	RF-001	HU02
RF-003	Organización de los análisis	Los análisis deben estar organizados de forma clara.	Media	RF-002	HU02
RF-004	Creación de análisis	El sistema debe permitir la creación de un nuevo análisis PRS desde el panel de control.	Alta	RF-002	HU03
RF-005	Filtro de los análisis	El sistema debe permitir filtrar los análisis según diferentes criterios, como fecha de creación, estado, tipo de análisis...	Media	RF-003	HU02
RF-006	Búsqueda entre los análisis	El sistema debe proporcionar una barra de búsqueda para localizar análisis por nombre, ID o palabras clave.	Alta	RF-002	HU02
RF-007	Visualización de los detalles del análisis	El usuario debe poder visualizar la configuración, descripción y modelos que componen el análisis accediendo con un botón desde el panel de control.	Alta	RF-002	HU04
RF-008	Eliminación de análisis	El sistema debe permitir a los usuarios eliminar análisis PRS. La eliminación debe requerir confirmación.	Media	RF-002	HU04

Continues on the next page

Continues from previous page

ID	Nombre	Descripción	Prioridad	Dependencias	Historia de usuario
RF-009	Selección de modelos	El sistema debe permitir seleccionar modelos PRS predefinidos.	Alta	RF-004	HU06
RF-010	Carga de <i>datasets</i> (estándar y formato específico)	El sistema debe permitir seleccionar <i>datasets</i> predefinidos.	Alta	RF-004	HU05
RF-011	Configuración de análisis	El sistema debe permitir configurar parámetros adicionales de los análisis.	Alta	RF-004	HU07
RF-012	Estado del análisis	El sistema debe mostrar el progreso de la ejecución del análisis.	Alta	RF-002	HU08
RF-013	Cancelación de análisis	El sistema debe permitir a los usuarios cancelar un análisis en ejecución.	Alta	RF-012	HU08
RF-014	Visualización de resultados (pantalla)	El sistema debe permitir la visualización de resultados de un análisis PRS.	Alta	RF-012	HU09
RF-015	Exportación de resultados	El sistema debe permitir exportar resultados en formato PDF.	Media	RF-014	HU10
RF-016	Notificación al usuario	El sistema debe enviar notificaciones a los usuarios cuando un análisis PRS se complete.	Media	RF-012	HU11
RF-017	Notificación en la <i>app</i>	Al completarse la ejecución de un análisis PRS las notificaciones pueden ser en la <i>app</i> .	Media	RF-016	HU11

Continues on the next page

Continues from previous page

ID	Nombre	Descripción	Prioridad	Dependencias	Historia de usuario
RF-018	Notificación por correo electrónico	Al completarse la ejecución de un análisis PRS las notificaciones pueden ser por correo electrónico.	Baja	RF-016	HU11

3.1.3. Requisitos no funcionales (RNF)

Complementando a los requisitos funcionales, los requisitos no funcionales (RNF) definen cómo debe operar el sistema. En lugar de describir funciones, especifican los criterios de calidad, las restricciones y las propiedades de la plataforma, como su rendimiento (ej. tiempo de carga), seguridad, usabilidad y escalabilidad. Estos requisitos son cruciales para garantizar que la experiencia del usuario sea satisfactoria y que el sistema sea robusto y fiable.

Tabla 3.3: Requisitos no funcionales RNF-001 – RNF-004

ID	Nombre	Descripción	Prioridad	Dependencias	Historia de usuario
RNF-001	Seguridad	La autenticación debe tener cifrado seguro.	Alta	RF-001	HU01
RNF-002	Tiempo de carga de pantalla	La pantalla principal de la aplicación (panel de control) debe cargar en menos de 2 segundos.	Alta	RF-002	HU02, HU09
RNF-003	Valor de rendimiento <i>Lighthouse</i> ¹	La aplicación debe obtener una puntuación superior a 90 en <i>Lighthouse</i> .	Media	RNF-002	HU02, HU09
RNF-004	Interfaz intuitiva	La UI debe ser clara y fácil de usar para que se puedan realizar análisis sin dificultad.	Media	RF-002, RF-003	HU02, HU09

¹Lighthouse es una herramienta destinada a mejorar el rendimiento de las páginas web, ejecutando una serie de auditorías en la página y, luego, generando un informe sobre su rendimiento.

3.2 Diseño

3.2.1. Análisis de posibles soluciones

En el diseño de la solución se barajan varias alternativas con objeto de facilitar el análisis de PRS a usuarios sin conocimientos técnicos.

La primera opción consiste en integrar múltiples herramientas ya existentes como *PRSice-2*, el catálogo *PGS* o entornos de visualización como *R/Shiny*; este modelo permite reaprovechar soluciones ya validadas en el ámbito científico. Sin embargo, implica una complejidad técnica elevada para lograr una integración funcional y fluida. El hecho de utilizar herramientas heterogéneas, pensadas para usarse de forma individual, dificulta la cohesión de la solución y añade carga al desarrollo, por lo que se aleja del objetivo principal del proyecto, que es ofrecer una experiencia accesible y sencilla.

Otra opción que se baraja es la construcción desde cero de una herramienta propia, lo que puede dar completo control sobre el diseño de la interfaz, las funcionalidades y la lógica. A pesar de las ventajas, tiene también dos claros problemas: por un lado, el coste de desarrollarla desde cero y, por otro, hay gran cantidad de funcionalidades iguales o parecidas a las que se busca que tenga la herramienta que ya están resueltas en herramientas existentes.

Por último, como opción más equilibrada, se plantea usar una herramienta base ya existente y desarrollar una capa de interacción adicional, accesible y amigable al usuario. Se minimiza el desarrollo reutilizando una herramienta ya probada, a la vez que se sigue permitiendo centrarse en la experiencia de usuario a través de una capa de interacción intuitiva y usable. También es técnicamente viable puesto que se separa el cálculo en el backend, reutilizable por otras aplicaciones, y un frontend, para el que se tiene en cuenta principalmente la simplicidad y la usabilidad. Aunque limita un poco debido a que la herramienta base marca la línea a seguir, es la opción escogida si se adapta a los objetivos del proyecto y al usuario final.

3.2.2. Solución propuesta

Finalmente, se ha decidido escoger la última opción y usar *GenoPred* como herramienta base para el desarrollo del sistema. *GenoPred* es una herramienta especialmente diseñada para el cálculo de PRS, que es compatible con los modelos del catálogo *PGS* y que soporta los principales formatos genómicos como *PLINK* o *VCF*. Al mismo tiempo, su punto fuerte es que cuenta con una arquitectura modular basada en reglas, lo que permite ejecutar el *pipeline* rama a rama o todo junto dependiendo de las necesidades. Con *GenoPred* también se pueden analizar individuos concretos, lo cual es muy importante, por ejemplo, en entornos clínicos, donde quizás no se trabaja con “grandes” cohortes, además de proporcionar una población de referencia interna, indispensable para poder contextualizar los resultados de los pacientes.

Otro punto a favor es su compatibilidad con ejecución desde línea de comandos, por lo que resulta fácilmente integrable desde un backend, y su carácter de software libre, su mantenimiento activo y su detallada documentación. A partir de esta base se ha estudiado qué funcionalidades de *GenoPred* implementar en la interfaz web, y se ha decidido apostar por aquellas fundamentales para que cualquier usuario, aunque no tenga grandes conocimientos técnicos, pueda llevar a cabo un análisis PRS completo. Por tanto, se han descartado funciones como la creación de modelos nuevos o la configuración avanzada de pasos intermedios, ya que resultan, en teoría, más complejas para el usuario. En su lugar, se han incorporado funcionalidades como la subida de datos, la selección de

modelos ya existentes y la visualización de los resultados, todo ello de un modo guiado y, por tanto, más sencillo de seguir por cualquier usuario.

Las fases de desarrollo que se van a seguir son:

1. **Desarrollo de la Interfaz de Usuario (*Frontend-First*):** Se ha decidido iniciar la implementación desarrollando el diseño de la interfaz de usuario (UI). Esta decisión se justifica por una metodología de diseño centrada en el usuario que permite representaciones tangibles tempranas de las experiencias del usuario.

Esto prioriza los *front-ends* para recopilar comentarios tempranos sobre la usabilidad y el flujo de trabajo, lo cual es crítico para validar las expectativas del usuario y refinar los requisitos.

Además de los contratos de UI definidos y una visión clara, el backend solo requiere una eficiencia más dirigida en el desarrollo de *endpoints* y servicios.

2. **Definición e Implementación de la Lógica de Backend:** Inmediatamente después, se procederá a definir e implementar los *endpoints* para la API. Estos *endpoints* serán los enlaces fundamentales para conectar la lógica del *front end* con los servicios de procesamiento en el backend.

El aspecto crucial de esta fase estará vinculado a herramientas computacionales especializadas como Genopred, que está ejecutando el análisis PRS. Debido a la naturaleza CLI de Genopred como una herramienta existente, se ha optado por invocarla a través de un proceso hijo (*child process*). Esto permite que el servicio de backend (probablemente el *Orchestrator* o un componente específico de la API) ejecute Genopred como un programa externo, pasando parámetros de entrada a través de la terminal y capturando sus salidas.

La estrategia utiliza la funcionalidad ya bien probada de Genopred sin reimplementar su lógica computacional y, por lo tanto, es sencilla de conectar y menos propensa a errores en el cálculo genético.

3. **Pruebas Exhaustivas y Aseguramiento de la Calidad:** La última fase del desarrollo se centrará en la implementación de pruebas extensivas para la validación exhaustiva de la solución.

Las pruebas unitarias llevarán a cabo verificaciones de que los módulos individuales funcionan bien, las pruebas de integración comprobarán que la comunicación entre los diferentes componentes del sistema (*Frontend*, *API*, *Prisma*, *Genopred*, etc.) discutidos es correcta y fluida, y las pruebas de extremo a extremo simularán escenarios de uso reales desde la perspectiva del usuario.

Se hará un énfasis especial en los componentes de importancia funcional: la configuración correcta de los análisis PRS (validando que todos los parámetros se almacenan y recuperan correctamente) y la ejecución precisa de los análisis en Genopred (verificando que los comandos se invocan correctamente, los datos se procesan y los resultados obtenidos cumplen con los requisitos).

El objetivo es asegurar que todas las funcionalidades cumplen con los requisitos funcionales previamente definidos y que la plataforma es estable y fiable.

3.2.3. Modelo de dominio

Se va a empezar mostrando el desarrollo del modelo de dominio, representado mediante diagramas de clases UML. El desarrollo de este modelo ha seguido un proceso

iterativo que ha permitido asegurar que el modelo final fuera robusto, preciso y proporcionase una representación detallada del dominio. Cada iteración ha sido el resultado de sesiones de trabajo conjuntas entre el equipo de desarrollo y los expertos del dominio, asegurando así que cada versión sea justificada tanto por necesidades técnicas como por una correcta representación del dominio.

Este proceso iterativo se ha llevado a cabo hasta que se ha alcanzado un punto de consenso donde el modelo final ha sido aceptado por ambas partes como una representación fiel del dominio, y ha servido a las necesidades de la implementación de la plataforma.

La primera versión del modelo de dominio (mostrada en la Figura ??) corresponde al borrador inicial, que surge a partir del conocimiento obtenido durante la fase de la *Investigación del Problema* (sección ??).

Las entidades pertenecientes a esta primera versión son:

- **User:** Representa al usuario de la plataforma. Contiene los atributos básicos para la autenticación y la gestión de la cuenta, como el `email` y la contraseña. Es la entidad principal que realiza los análisis.
- **PRS_Analysis:** Es la entidad central del modelo. Contiene toda la información relevante para la configuración y el estado de un análisis, incluyendo su nombre, descripción, el estado actual (ej. en curso, completado) y la ruta al fichero de resultados.
- **Patient:** Modela al individuo cuyos datos genómicos se van a analizar. Cada análisis está asociado a un único paciente, almacenando la ruta a su fichero de datos.
- **PRS_Model:** Representa los modelos de Puntuación de Riesgo Poligénico. Almacena metadatos clave sobre cada modelo, como su identificador del catálogo PGS, el nombre y el fenotipo (enfermedad o rasgo) asociado.
- **Analysis_Config:** Esta entidad encapsula los parámetros de configuración específicos de un análisis, como los umbrales de ancestría y de solapamiento. Se vincula a una población de referencia para contextualizar los resultados.
- **Population:** Contiene la información sobre las poblaciones de referencia utilizadas para comparar y normalizar los resultados de un análisis.
- **Result:** Modela el resultado final de un análisis PRS completado. Almacena la puntuación PRS calculada, el percentil del individuo y un indicador de si el resultado ha sido exportado.
- **Notification:** Representa las notificaciones generadas por el sistema para informar al usuario sobre el estado de sus análisis (ej. finalización exitosa o fallo).

A partir de estas clases, las relaciones más relevantes de este modelo inicial son:

- **User - PRS_Analysis:** Un User realiza múltiples PRS_Analysis. Esta es la relación principal que vincula los análisis a quien los ejecuta.
- **PRS_Analysis - Patient:** Cada PRS_Analysis se analiza en un único Patient. Esto asegura que cada análisis esté asociado a un individuo concreto.
- **PRS_Analysis - PRS_Model:** Un PRS_Analysis incluye uno o varios PRS_Model, definiendo qué modelos se aplicarán en ese análisis.

- PRS_Analysis - Analysis_Config: Cada PRS_Analysis tiene una configuración asociada, representada por la entidad Analysis_Config.
- Analysis_Config - Population: La Analysis_Config hace referencia a una Population para la normalización de los resultados.
- PRS_Analysis - Result: Un PRS_Analysis completado genera un Result. La cardinalidad 0..1 indica que el resultado solo existe si el análisis ha finalizado con éxito.
- PRS_Analysis - Notification: Un PRS_Analysis envía múltiples Notifications al usuario para informarle de cambios de estado.

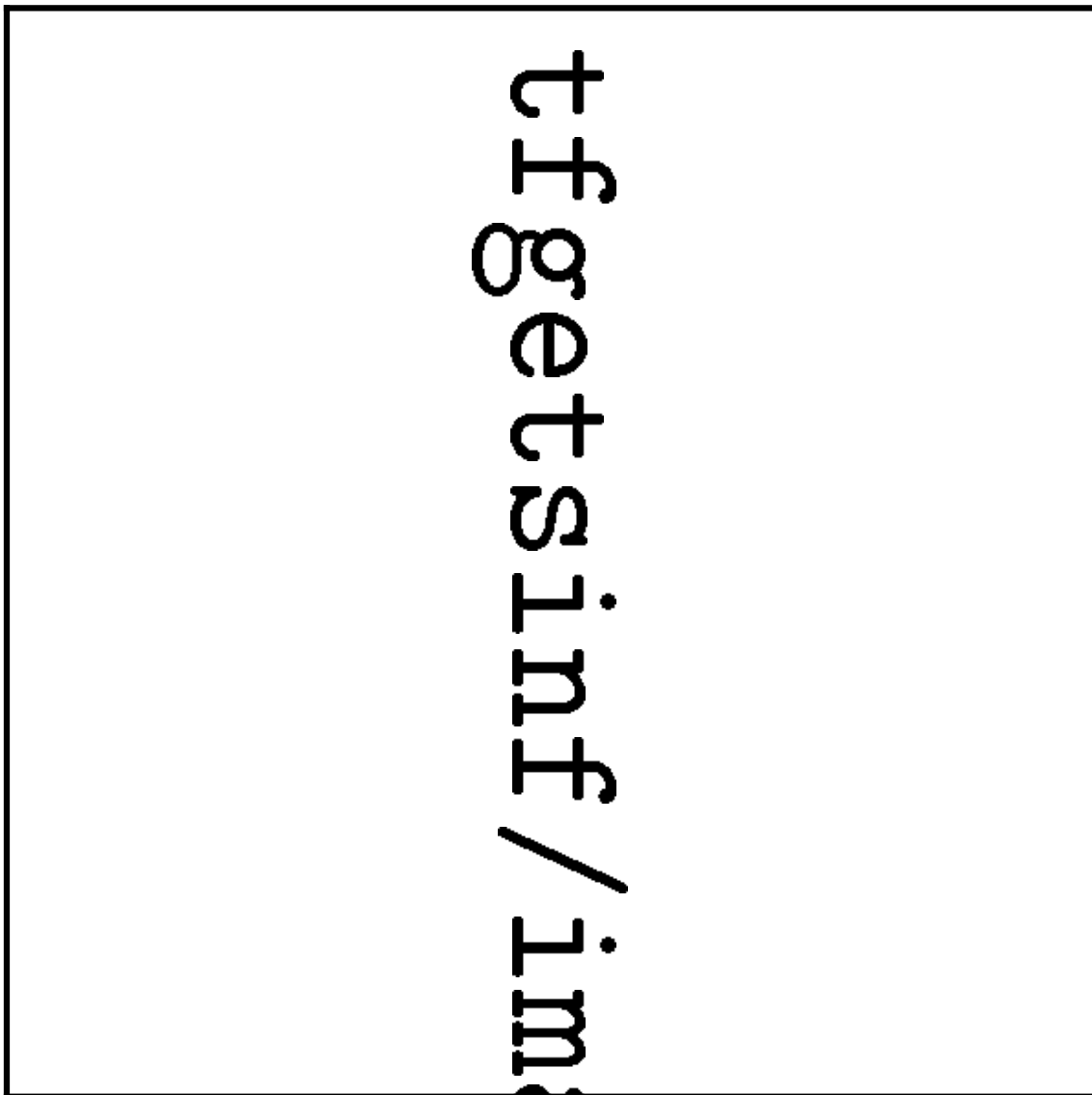


Figura 3.1: Diagrama de Clases v.1

Tras una revisión inicial a partir de la presentación del modelo a los tutores y compañeros de trabajo, se identificaron varias áreas de mejora y nuevas necesidades que dieron lugar a una segunda versión del modelo, presentada en la Figura ???. Esta iteración refina el modelo inicial para representar mejor el dominio y considerar el punto de vista de los expertos. Los cambios han sido los siguientes:

- **Eliminación de la clase `User`:** Para desacoplar el modelo de dominio de la gestión de la autenticación, se eliminó la clase `User` y se sustituyó por un atributo `userId` en `PRSAnalysis`. Esto simplifica el modelo y delega la responsabilidad de la gestión de usuarios a un servicio externo llamado *Auth0*.
- **Creación de la clase `PrioritizedModels`:** Para modelar explícitamente la relación entre un análisis y los múltiples modelos PRS que puede utilizar, se extrajo esta lógica a una clase propia, permitiendo tener disponible la lista de los modelos seleccionados en cada análisis.
- **Adición de `ScoringFile` y `Trait`:** Para representar de forma más granular la información de los modelos, se añadieron estas clases que contienen, respectivamente, la ruta al fichero de puntuaciones y los rasgos o fenotipos asociados.
- **Eliminación de la clase `Notification`:** Se determinó que las notificaciones son una funcionalidad del sistema y no un concepto central del dominio del problema. Su eliminación del diagrama de clases ayuda a mantener el modelo enfocado en la lógica principal.

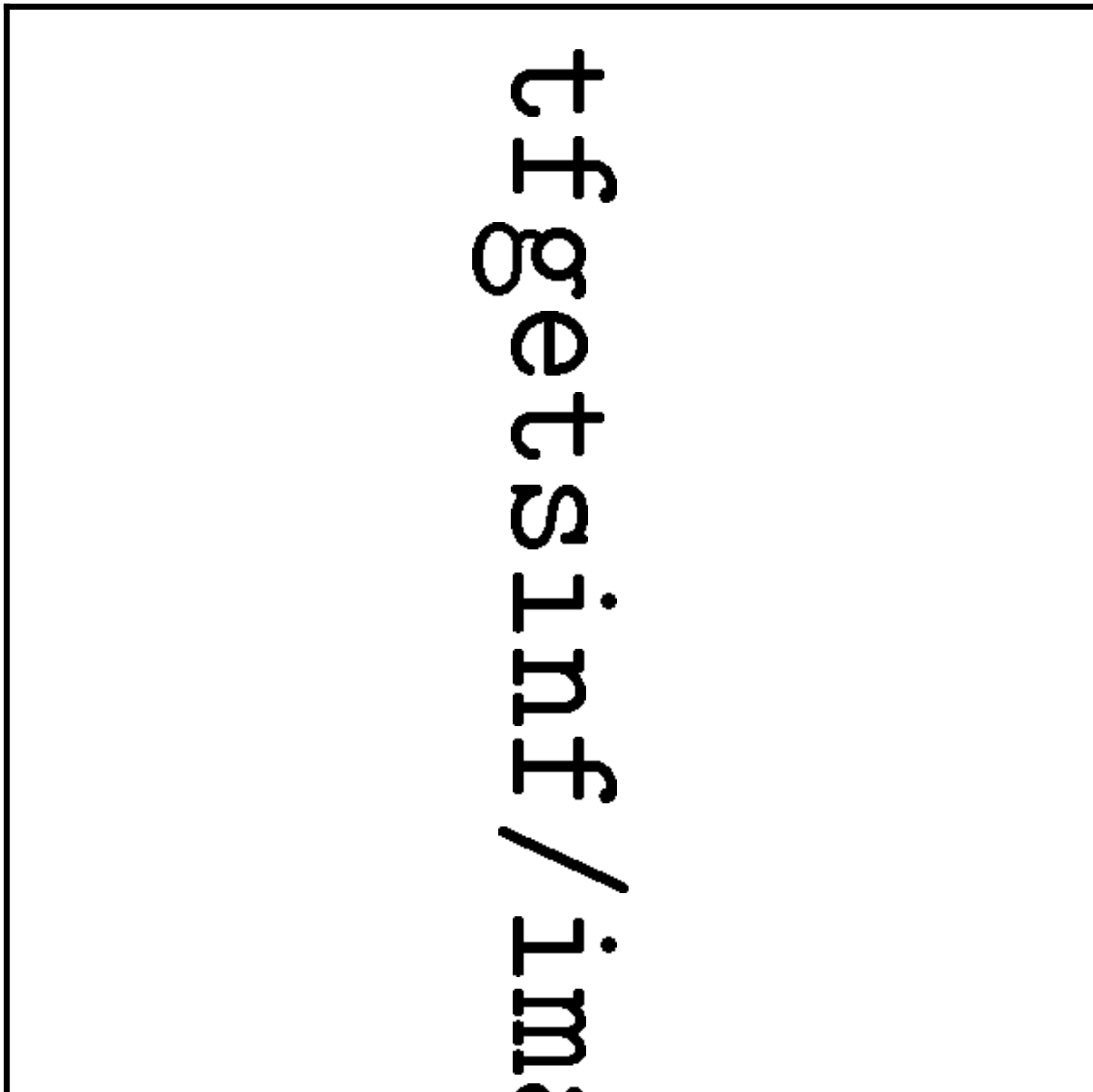


Figura 3.2: Diagrama de Clases v.2

Tras una nueva reunión, la validación exhaustiva con los expertos del dominio ha revelado nuevas necesidades y también oportunidades de mejora, que han llevado a una tercera y definitiva versión del diagrama de clases presentado en la Figura ???. Esta entrega representa la consolidación del modelo de dominio, con un mayor nivel de detalle y avances en las dificultades encontradas para representar el dominio de la forma más robusta y fiable posible y que requieran la plataforma de implementación.

Los cambios aplicados a la versión anterior han sido los siguientes:

- **Consolidación de la configuración del análisis en PRSAnalysis** La clase antigua `AnalysisConfig` que se usaba para envolver parámetros como `ancestryThreshold`, `overlapThreshold` y `ancestryAdjustment` ha sido eliminada. Sus características se han incorporado directamente en la clase `PRSAnalysis`. Estos cambios se han hecho para simplificar el modelo, ya que la forma de configurar un análisis PRS está inherentemente ligada a la instancia del análisis en sí y porque `AnalysisConfig` no tenía identidad independiente que justificara su existencia como clase separada.
- **Enriquecimiento de clases con nuevos atributos:** Para disponer de una representación más completa de los datos y las funcionalidades que requiere la plataforma, se han añadido numerosos atributos a diversas clases ya existentes.
 - En `ReferencePopulation`, se han añadido `-studyURL` y `-studyName`, que permiten documentar de dónde vienen las poblaciones de referencia que se usan para analizar los datos y saber cuál es el contexto de ese estudio. Sin esto, no se pueden analizar ni entender los resultados que se obtienen.
 - La clase `Patient`, se ha enriquecido con `-genotypingMethod` y `-DataFileFormat`, que son de ayuda para poder contar cómo se ha obtenido el genotipo de los datos almacenados en la clase `Patient` y en qué formato está el archivo de entrada. Sin esta información, no existe ni la capacidad de trabajar con estos datos ni de interpretarlos de forma correcta.
 - `PRSMoel` ahora incorpora `-numberOfSNP`, un atributo esencial para conocer cuántos polimorfismos de nucleótido único (SNPs) forman parte de cada modelo, lo cual da una pista sobre su complejidad y el nivel de detalle de su construcción.
 - La clase `Trait` ha cambiado con la incorporación de identificadores externos como `-orphaId`, `-hpoId`, `-mondoId`, `-efoId` y `-otherId`. Estos IDs estandarizados permiten que los rasgos genéticos se unan a bases de datos y ontologías biomédicas que ya están establecidas (como Orphanet para enfermedades raras, HPO para fenotipos humanos, Mondo para enfermedades...), de esta manera se hace más fácil la interoperabilidad y la integración con otras fuentes de conocimiento genómico.
- **Introducción de las clases:** Para reflejar con precisión las nuevas complejidades y los nuevos requisitos del modelo de dominio, se han añadido varias clases nuevas a esta entrega final que enriquecen la capacidad del sistema de describir y hacer su función.
 - `BroadAncestryCategory`: La importancia de la diversidad ancestral en la precisión y aplicabilidad de los Polygenic Risk Scores (PRS) es crucial, es por eso que se ha añadido esta clase. Permite clasificar y gestionar las diferentes categorías de ascendencia relevantes para los modelos PRS y las poblaciones de referencia. Su inclusión es necesaria para asegurar que los análisis puedan ajustar los resultados teniendo en cuenta el contexto ancestral del paciente.

- **BroadAncestryInModel**: Es una clase de asociación que surge de la necesidad de cuantificar a modo de porcentaje la contribución o aplicabilidad de una categoría ancestral dada a un modelo, permitiendo así una mayor granularidad en la definición de la composición ancestral de los modelos y su adecuación para diferentes grupos poblacionales.
- **BroadAncestryInRefPop**: Esta clase de asociación que es similar a la anterior, se ha introducido para especificar la composición ancestral de una población de referencia concreta. Es esencial para documentar la distribución de las diferentes categorías ancestrales dentro de las poblaciones de referencia utilizadas para la validación de los modelos PRS.
- **Publication**: Para establecer un enlace directo y transparente entre los modelos PRS y la literatura científica que los sustenta, se ha incluido esta clase, la cual almacena metadatos clave que permiten acceder rápidamente a la fuente original de la investigación detrás de cada modelo utilizado en los análisis.
- **TraitCategory**: Para mejorar la organización y la capacidad de búsqueda de los rasgos genéticos, se ha añadido esta clase, que permite categorizar los Trait de forma jerárquica o temática ("Enfermedades Cardíacas", "Trastornos Neurológicos"), facilitando la gestión de rasgos.

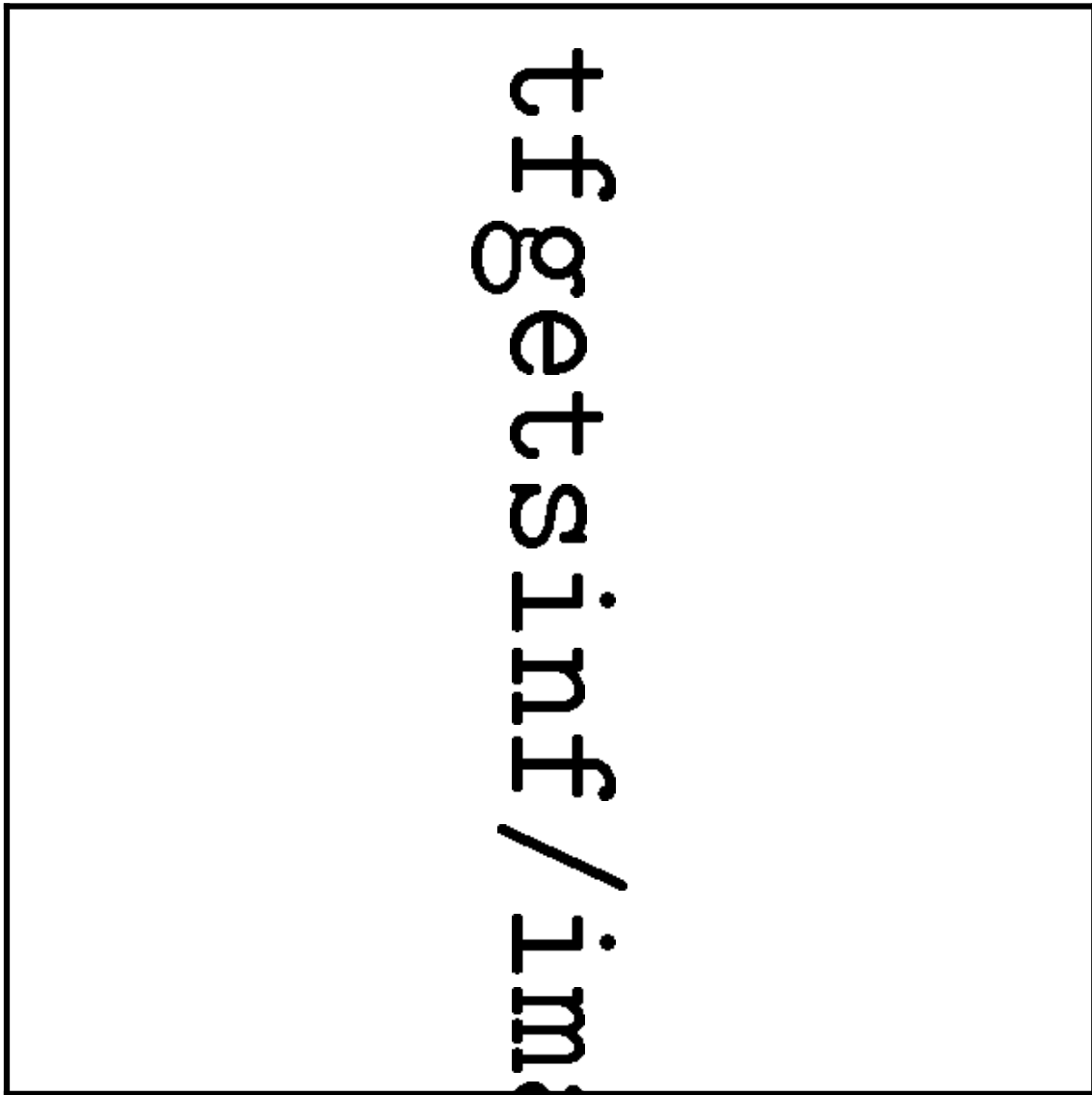


Figura 3.3: Diagrama de Clases v.3

3.2.4. Diagrama de paquetes

Para obtener una visión de alto nivel de la estructura del sistema, se ha elaborado un diagrama de paquetes UML, representado en la Figura ???. Este tipo de diagrama proporciona una vista macroscópica del sistema, organizando los elementos del modelo en paquetes relacionados para mostrar su estructura y sus dependencias. El uso de este diagrama simplifica la comprensión, la gestión y la comunicación del diseño de un proyecto de software complejo. En este caso, el diagrama de paquetes organiza la arquitectura de la aplicación en los tres paquetes principales siguientes:

- **Frontend:** Es la paquete de presentación, responsable de la interfaz de usuario (UI) y la interacción directa con el usuario. Está construida sobre un moderno *framework* de aplicaciones web que permite renderizar las páginas y los componentes visuales. Dentro de esta capa, un componente autenticador gestiona la lógica del lado del cliente para la autenticación, interactuando con un servicio de identidad externo.
- **Backend:** Es el paquete de lógica de negocio, donde reside el núcleo funcional de la aplicación. La integran los componentes siguientes:

- **Capa de API:** Proporciona los *endpoints* de la API *RESTful* a los que el frontend realiza las peticiones. Conecta el Frontend con la capa de acceso a datos y el sistema de ficheros.
 - **Orchestrator:** Es el componente central que coordina las operaciones. Gestiona el flujo de creación, ejecución y consulta de los análisis PRS, ya que está directamente ligado al servicio externo Genopred.
 - **Capa de Acceso a Datos (ORM):** Actúa como un Mapeo Objeto-Relacional (ORM), facilitando la comunicación con la base de datos de una manera segura y tipada.
 - **Notifier:** Gestiona el envío de notificaciones al usuario (ej. por correo electrónico) cuando un análisis finaliza.
- **Services:** Incluye tanto componentes externos como internos que dan soporte a la aplicación.
- **Servicio de Autenticación y Autorización:** Es un servicio externo de gestión de identidad que se encarga de la autenticación y autorización de usuarios de forma segura.
 - **File System:** Representa el sistema de archivos del servidor donde se almacenan temporalmente los datos genómicos de los pacientes y los resultados de los análisis PRS.
 - **Genopred:** Es el motor de cálculo bioinformático. El Orchestrator invoca a esta herramienta externa para ejecutar los análisis PRS.
 - **Base de Datos Relacional:** Es el sistema de gestión donde se persiste toda la información del sistema, como los datos de los análisis, los modelos y las poblaciones de referencia.

El flujo de datos típico comienza cuando un usuario realiza una acción en el Frontend, como solicitar un nuevo análisis, y este envía una petición a la API del Backend y es el Orchestrator quien recibe esta petición. Posteriormente, utiliza Prisma para registrar el nuevo análisis en la base de datos MySQL, guarda los ficheros necesarios en el File System y delega la ejecución del cálculo a Genopred. Una vez que Genopred finaliza, el Orchestrator actualiza el estado en la base de datos y, a través del Notifier, informa al usuario del resultado.

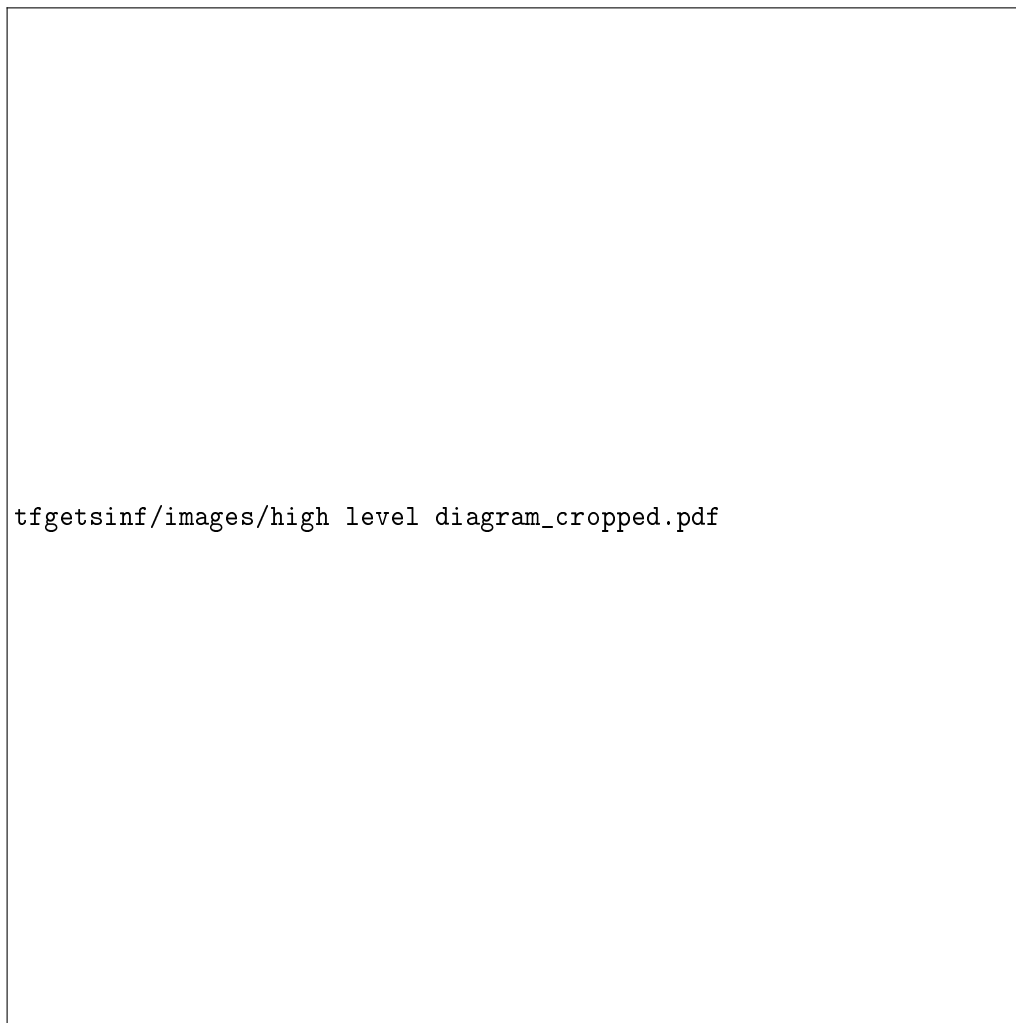


Figura 3.4: Diagrama de Alto Nivel

3.2.5. Diagrama de componentes

La representación de componentes de la Figura ?? muestra la estructura de software de *Phenoscore*, desagregada en sus distintos módulos principales y su relación con los servicios externos. Este enfoque modular permite separar responsabilidades y facilita el mantenimiento y la escalabilidad del sistema. A continuación, se detallan los componentes de acuerdo a su dominio de actuación, especificando las tecnologías utilizadas:

1. **Aplicación Principal (*Phenoscore*)** Es el corazón del sistema, construido como una aplicación *full-stack* unificada con el *framework* *Next.js*. En él se contienen tanto los módulos de la interfaz de usuario (*frontend*) como los de la lógica de negocio (*backend*).
 - **Frontend:**
 - **Interfaz de Usuario (*Next.js/React*):** Se trata del componente que lleva a cabo la representación de la interfaz web con la que el usuario final interactúa. Implementado en *React*, ofrece los formularios de configuración, muestra el seguimiento de los análisis y la visualización de los resultados.
 - **Authenticator:** Módulo cliente que se encarga de manejar el proceso de *login*. Se comunica directamente con *Auth0*, el proveedor de identidad,

para verificar las credenciales del usuario y recibir los tokens de acceso, garantizando que únicamente usuarios legítimos accedan a la aplicación.

■ **Backend:**

- **API (Next.js):** Proporciona los *endpoints RESTful* que comunican el frontend con la lógica de negocio. Gestiona las peticiones para crear análisis, consultar su estado y obtener resultados.
- **Orchestrator:** Es el núcleo de la lógica del backend. Administra el flujo de ejecución completo: desde la llegada de una nueva petición de análisis, la validación de los datos, la interacción con el File System para el manejo de archivos, la llamada a Genopred vía ChildProcess, hasta la recogida y el almacenamiento de los resultados en la base de datos con Prisma.
- **Prisma:** Actúa como cliente ORM (*Object-Relational Mapping*) para gestionar la conexión y la comunicación con la base de datos MySQL. Permite consultar, insertar y actualizar los datos de los análisis, modelos y usuarios de una forma segura y tipada.
- **ChildProcess:** Es el módulo de *Node.js* utilizado para ejecutar y gestionar procesos externos. Su función es fundamental para invocar Genopred, permitiendo que el análisis se ejecute de forma asíncrona sin bloquear el servidor principal.
- **Notifier:** El componente que gestiona el envío de notificaciones al usuario (por ejemplo, por correo electrónico) cuando un análisis ha finalizado o ha ocurrido un error.
- **Logger:** Módulo encargado de registrar los eventos relevantes del sistema (ej. inicio de un análisis, errores, finalización) para facilitar el diagnóstico, la auditoría y la trazabilidad de las operaciones.

2. **Infraestructura Local** Conjunto de componentes que se ejecutan en el mismo entorno físico que la aplicación, pero que no forman parte directa del código base del proyecto.

- **File System:** Representa el sistema de archivos del servidor. Es utilizado por el Orchestrator para almacenar temporalmente los datos genómicos de los pacientes subidos por el usuario, los modelos PRS seleccionados y los ficheros de resultados generados por Genopred.
- **Genopred:** Es el motor de cálculo bioinformático. Se trata de una herramienta externa especializada en ejecutar los análisis de Puntuación de Riesgo Poligénico. El Orchestrator lo invoca como un proceso externo para realizar los cálculos computacionalmente pesados.

3. **Servicios de Terceros (3rd Party Services)** Servicios externos con los que el sistema se comunica a través de APIs para delegar funcionalidades específicas.

- **Auth0:** Es el servicio externo de identidad y autenticación. Administra el inicio de sesión y la gestión de los usuarios. El componente Authenticator de la aplicación se conecta a Auth0 para garantizar un control de acceso robusto y seguro.
- **MySQL Database:** Es el sistema de gestión de base de datos relacional externo donde se persiste toda la información crítica del sistema. El componente Prisma se conecta a esta base de datos para guardar y recuperar datos sobre los análisis, sus configuraciones, estados, resultados y los modelos utilizados.

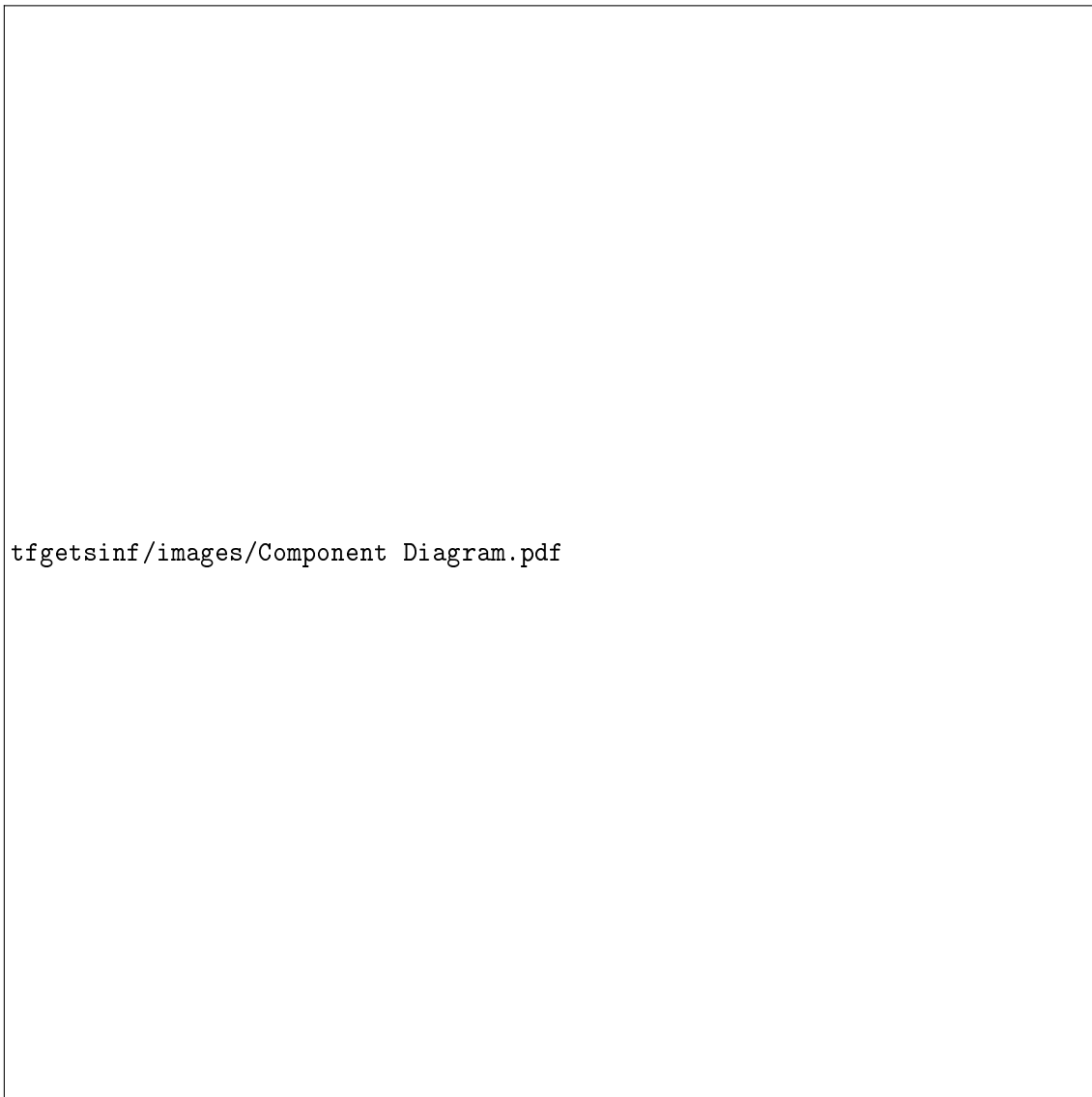


Figura 3.5: Diagrama de Componentes

3.2.6. Diagramas de secuencia

Cada diagrama de secuencia representa un flujo clave de funcionalidad, detallando la coreografía de mensajes entre las distintas partes del sistema para alcanzar un objetivo específico. Se han consolidado satisfactoriamente las historias de usuario en tres diagramas principales que cubren las funcionalidades más críticas y complejas, y cada uno detalla la coreografía de mensajes entre los distintos componentes.

Los actores y componentes principales del sistema que interactúan en los diagramas de secuencia son los mismos que han sido detalladamente descritos en la sección de Diagrama de Alto Nivel, donde se estableció su rol y responsabilidades dentro de la arquitectura de la plataforma.

Diagrama de secuencia: Inicio de Sesión (*Log In*) La Figura ?? detalla el flujo de inicio de sesión de un usuario y la posterior recuperación de sus análisis guardados.

Historias de Usuario Cubiertas	HU01 (Inicio de sesión) y parcialmente HU02 (Panel de control).
Contexto y Precondiciones	El usuario desea acceder a la aplicación y ya posee una cuenta registrada y válida en el sistema de autenticación externo (<i>Auth0</i>).

El flujo comienza cuando el usuario introduce sus credenciales en el Frontend. Este envía la solicitud a la API, que a su vez la delega en el servicio de autenticación externo. Tras una validación exitosa, el servicio devuelve un token de autenticación a la API, que lo reenvía al Frontend. Con el usuario ya autenticado, el Frontend solicita la lista de análisis personales. La API valida de nuevo el token y, a través de Prisma, consulta la base de datos para obtener los análisis del usuario. Finalmente, los datos son devueltos al Frontend, que los muestra en el panel de control.

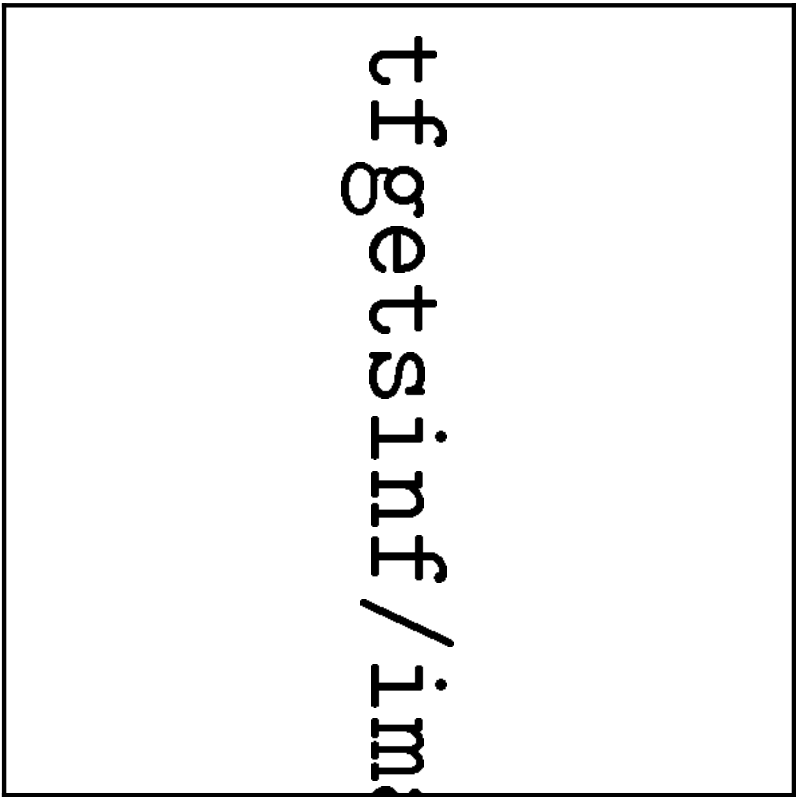


Figura 3.6: Diagrama de Secuencia - Log In

Diagrama de secuencia: Configuración del Análisis La Figura ?? ilustra el proceso de configuración y almacenamiento inicial de los parámetros y archivos de datos para un nuevo análisis PRS.

Historias de Usuario Cubiertas	HU03 (Crear nuevo análisis), HU05 (Subida de archivos), HU06 (Selección de modelos) y HU07 (Configuración).
Contexto y Precondiciones	El usuario ha iniciado sesión y ha completado el formulario de configuración de un nuevo análisis en la interfaz.

El proceso se inicia cuando el usuario, tras rellenar el formulario, pulsa el botón para ejecutar el análisis en el Frontend. Esto envía una petición POST a la API con toda la

configuración. La API interactúa primero con el `FileSystem Handler` para gestionar y almacenar los ficheros genómicos subidos. Una vez confirmado el almacenamiento de los archivos, la API envía a Prisma la solicitud de guardar todos los parámetros de configuración del análisis en la base de datos. Cuando Prisma confirma que los datos han sido persistidos, la API notifica al Frontend que la configuración se ha guardado con éxito, dejando todo listo para la fase de ejecución.

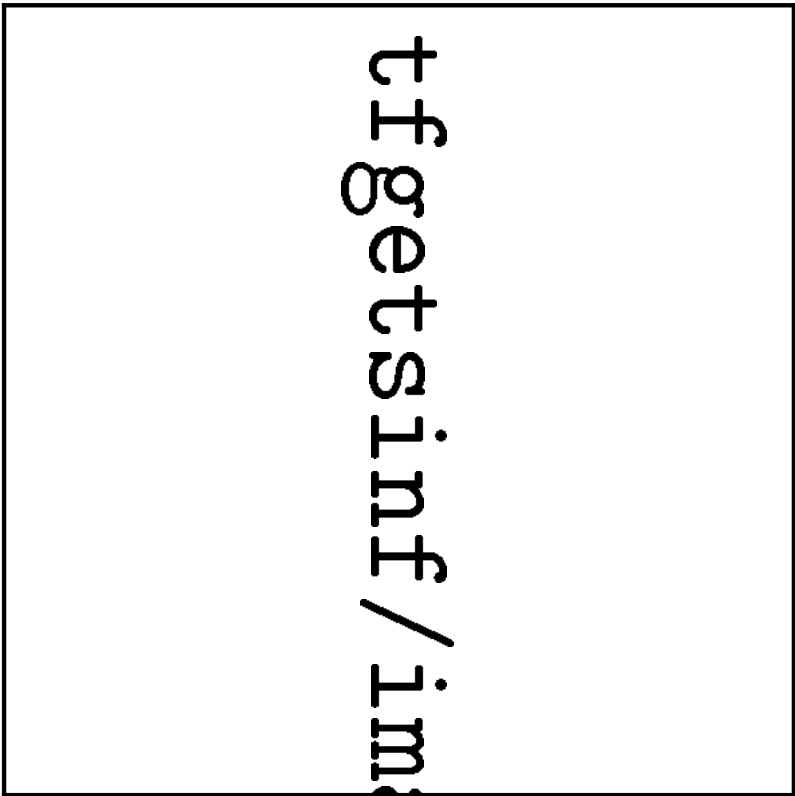


Figura 3.7: Diagrama de Secuencia - Configuración del Análisis

Diagrama de secuencia: Ejecución del Análisis La Figura ?? describe el flujo asíncrono de la ejecución de un análisis PRS, desde que se inicia hasta que se notifica el resultado al usuario.

Historias de Usuario Cubiertas	HU06 (Ejecución del análisis) y HU11 (Notificaciones).
Contexto y Precondiciones	El análisis ha sido configurado y sus parámetros están almacenados en la base de datos.

Este flujo, típicamente disparado por un evento interno, comienza cuando la API envía una orden al `Orchestrator` para iniciar un análisis. El `Orchestrator` recupera los datos y delega la ejecución del cálculo bioinformático en la herramienta externa `GenoPred`, proporcionándole todos los parámetros necesarios. Tras un tiempo de procesamiento, `GenoPred` notifica al `Orchestrator` que el análisis ha finalizado. A continuación, el `Orchestrator` envía los resultados (puntuaciones, percentiles) a Prisma para que los guarde en la base de datos. Una vez almacenados, invoca al `Notifier` para informar al usuario del resultado. Finalmente, la API actualiza el estado del análisis en el Frontend, que muestra al usuario que el proceso ha sido completado.

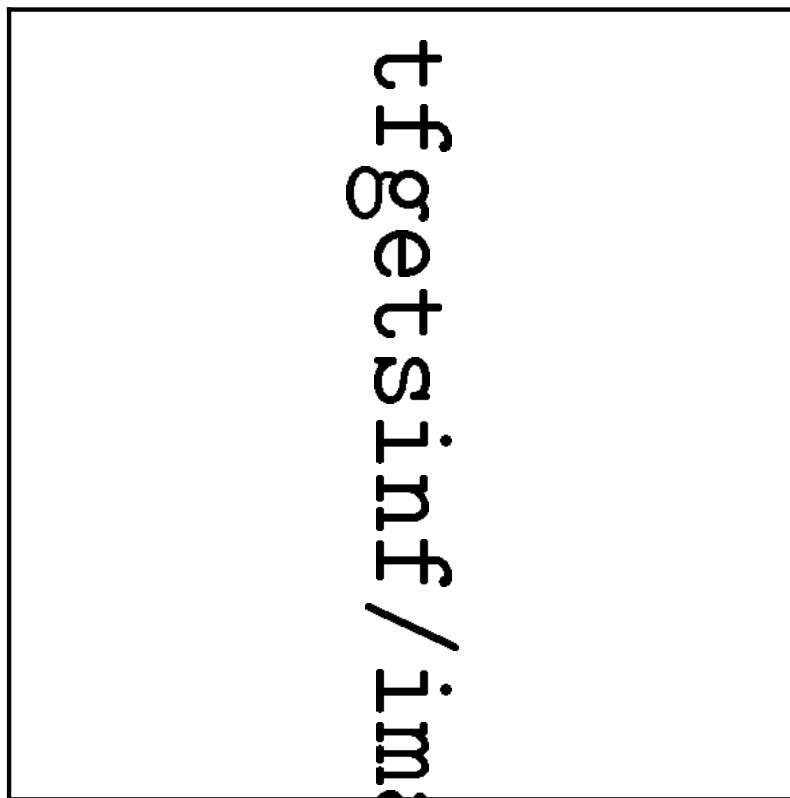


Figura 3.8: Diagrama de Secuencia - Ejecución del Análisis

3.2.7. Arquitectura lógica

Siguiendo con las decisiones arquitectónicas diseñadas en los modelos mostrados anteriormente, la estructura consta de tres capas diferenciadas que facilitan la separación de responsabilidades [?].

Capa de Presentación (Frontend): Esta capa es responsable de implementar la interfaz de usuario y de gestionar cualquier interacción directa con el usuario final. Algunas de las funcionalidades clave incluyen la configuración y visualización de análisis PRS, la aplicación de filtros para organizar la información y una presentación interactiva de resultados. El desarrollo utilizó *Next.js* y *React*, dando énfasis a la usabilidad y accesibilidad, lo que llevó a una experiencia de usuario intuitiva y eficiente.

Capa de Lógica de Negocio (Backend): Compuesta por una API construida en *Node.js*, esta capa es la responsable de la conexión entre el frontend, la base de datos y la herramienta Genopred. Entre las funcionalidades más destacables se encuentran la coordinación de la ejecución de los análisis, incluyendo la invocación de procesos externos (*child-process*) para integrar Genopred y la lógica de peticiones.

Capa de Datos: se gestiona mediante una base de datos en MariaDB, a la que se accede y manipula a través del ORM Prisma. En esta capa se almacena toda la información persistente del sistema, como los análisis vinculados al usuario, los modelos PRS disponibles, los resultados generados por cada análisis, etc.

Capa de presentación (frontend)

Esta capa es responsable de la interfaz de usuario. Para su implementación se ha seleccionado *Next.js*, un *framework* basado en *React*. Esta elección permite crear una aplicación web rápida y eficiente mediante el renderizado en el servidor (SSR) y la generación

de sitios estáticos (SSG). Para la estilización de los componentes, se utiliza *PostCSS* para aplicar transformaciones avanzadas sobre CSS.

Interfaz gráfica

Como paso previo a la implementación de las interfaces, se han diseñado los prototipos (o *mockups*) que se emplean para mostrar la idea inicial del funcionamiento de la *app* que posteriormente se va a desarrollar. Se ha hecho uso de la herramienta *Figma*, una herramienta web dedicada a los diseños gráficos y de interfaces. Además, concede la opción de compartir estos prototipos; es una funcionalidad que resulta muy atractiva pensando en la validación de los *mockups* por parte de los usuarios.

Cada pantalla está diseñada para dar respuesta a una o varias de las historias de usuario definidas en la sección de requisitos, asegurando que el diseño esté directamente alineado con las necesidades de los usuarios. A continuación, se presentan los prototipos de las pantallas principales.

La interfaz se divide en las siguientes pantallas:

Pantalla: Inicio de sesión

Esta pantalla es la puerta de acceso a la plataforma de forma segura. Inicialmente, el diseño contemplaba la implementación de un formulario de inicio de sesión propio; no obstante, por motivos de seguridad y para simplificar la compleja gestión de identidades, se ha optado por confiar en un servicio externo como *Auth0*. Por tanto, el diseño final (Figura ??) se centra en un simple botón que redirige al usuario a iniciar su flujo de autenticación con un proveedor externo.

Este diseño cubre directamente la necesidad de *HU01: Inicio de sesión*, que requiere que los usuarios puedan acceder de forma segura con sus credenciales para visualizar sus análisis personales.

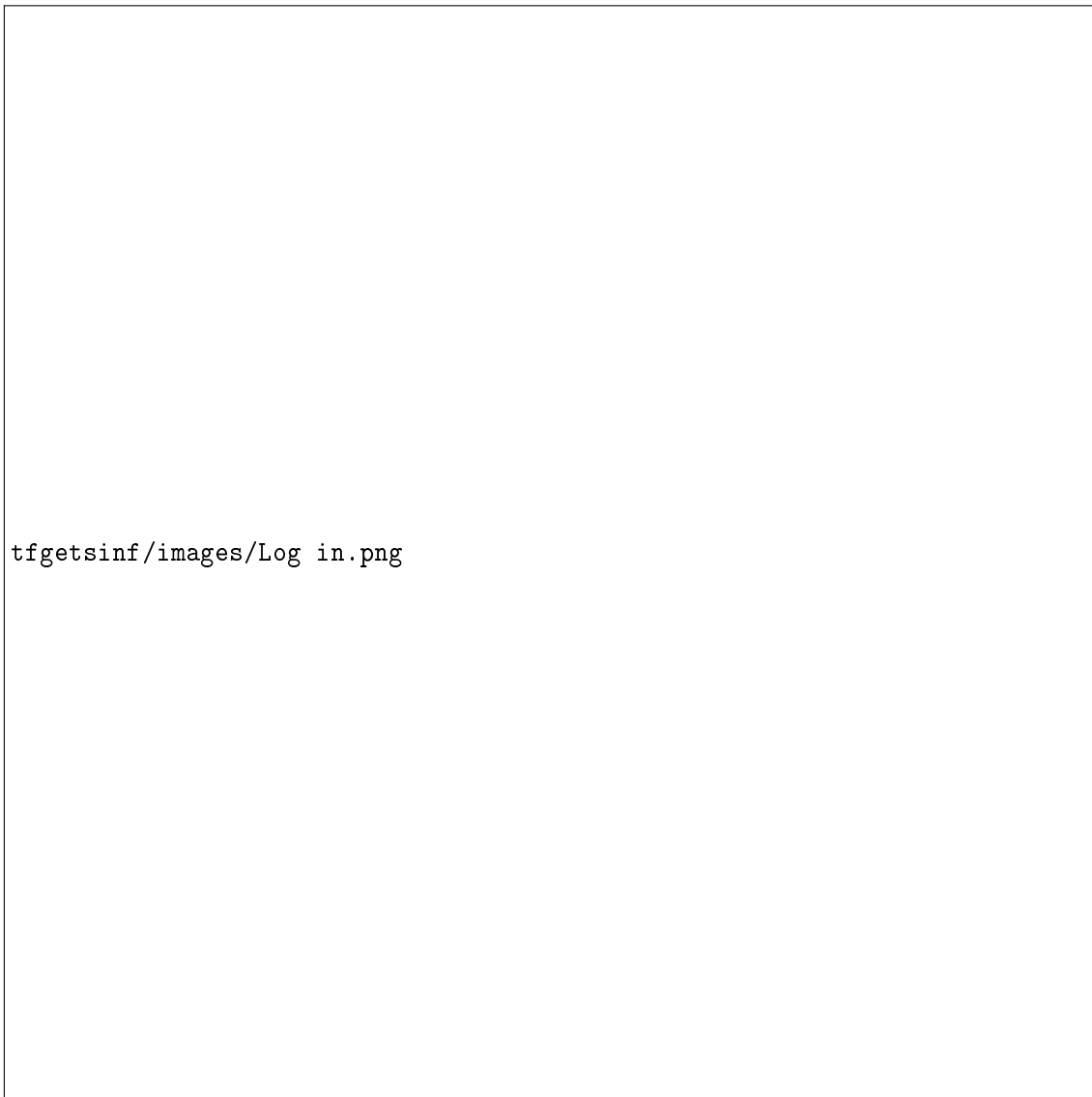


Figura 3.9: Mockup de Inicio de sesión

Pantalla: Análisis PRS

Esta es la pantalla de inicio o panel de control de la aplicación. El componente central (Figura ??) es una tabla que muestra todos los análisis del usuario, permitiendo buscar, consultar el estado o realizar acciones como consultar los resultados o eliminar. La interfaz dispone además de botones para crear nuevos análisis y para abrir un panel de filtros (Figura ??), que permite limitar la vista por paciente, estado o fecha.

Las historias de usuario cubiertas por esta pantalla son: *HU02: Panel de control*, al proporcionar una vista centralizada del estado y progreso de todos los análisis y *HU04: Gestión de análisis PRS*, ya que permite visualizar, ordenar y eliminar los estudios existentes.

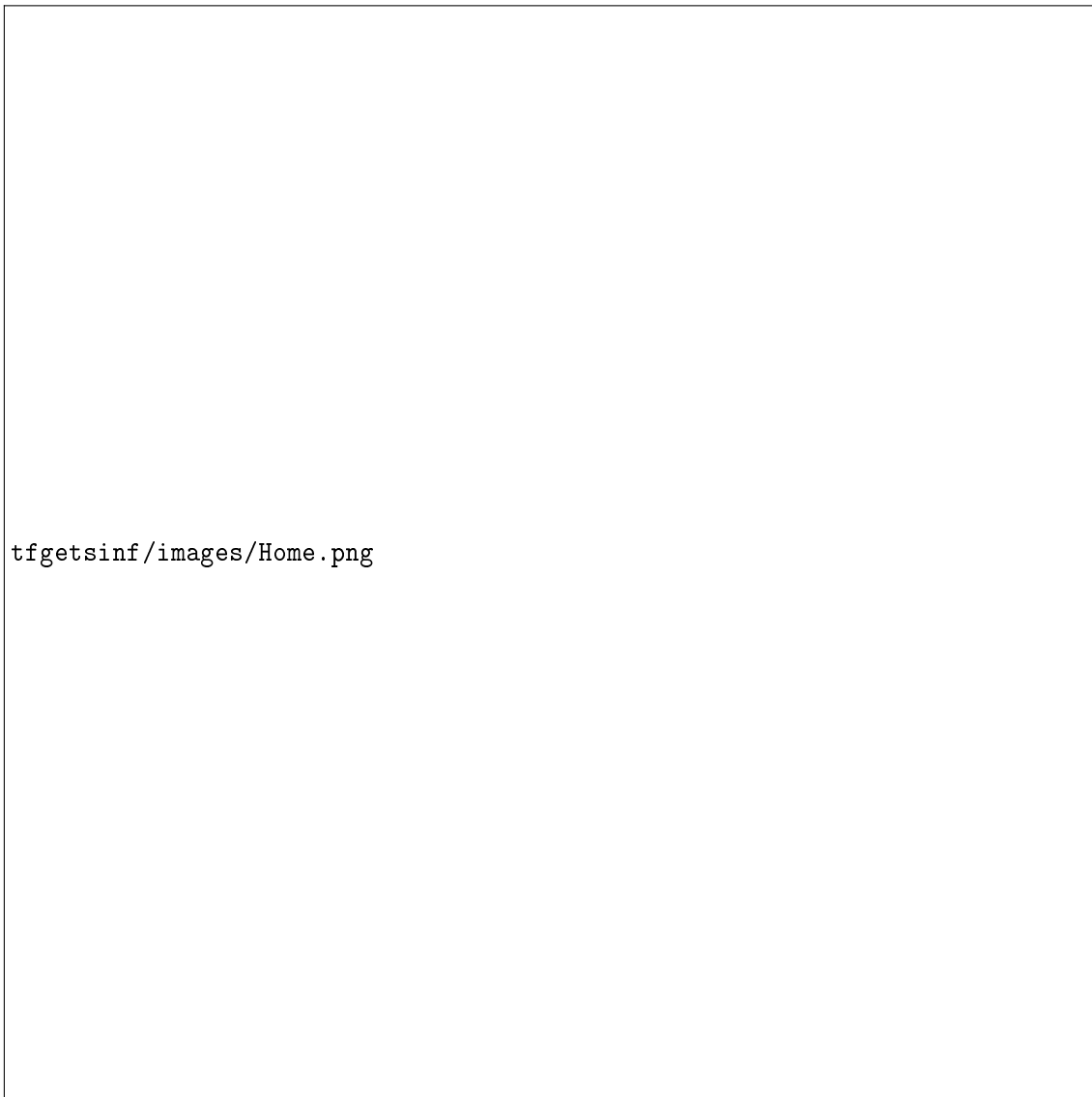


Figura 3.10: Mockup de Análisis PRS

Pantalla: Filtros

Para gestionar cómodamente un elevado número de análisis, se ha diseñado un panel lateral de filtros tal y como se muestra en la Figura ???. Esta funcionalidad permite al usuario delimitar los resultados que se le muestran en la tabla principal según unos criterios determinados que le interesen, y así hacer de una forma más eficiente sus búsquedas. En el prototipo se incluyen tres tipos de filtros: por identificador de paciente, por estado del análisis (completado, en proceso, error) y por fecha de creación.

A pesar de que no se ha creado una historia de usuario pensando solo en los filtros, la funcionalidad de los mismos es una aplicación directa que da apoyo a la *HU04: Gestión de análisis PRS*. Al facilitar que el usuario "visualice y ordene estudios como prefiera", los filtros son una pieza fundamental para realizar esta tarea, sobre todo cuando llegan a haber muchos análisis en el sistema.

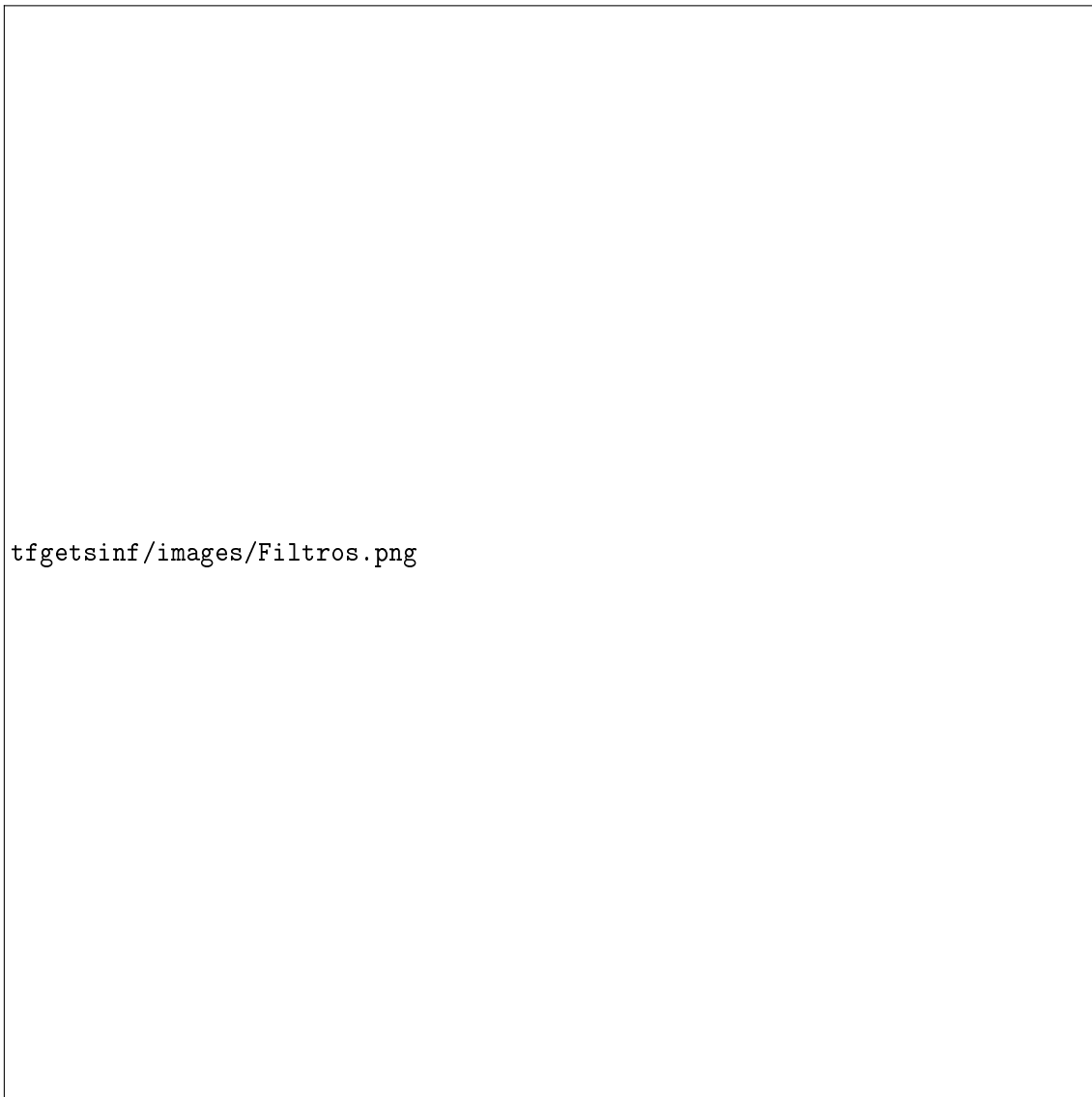


Figura 3.11: Mockup de Filtros

Pantalla: Configuración de Análisis PRS

Con este formulario (Figura ??) el usuario puede crear un nuevo análisis PRS. La pantalla está diseñada en pestañas, para facilitar la introducción de metadatos (nombre, paciente), la subida de los ficheros genómicos, la elección de modelos PRS a incluir en el análisis y la configuración de parámetros técnicos del análisis (umbrales de ancestría, ...)

Esta pantalla es clave en el desarrollo de la plataforma, al cubrir un conjunto de historias de usuario esenciales como: *HU03: Crear nuevo análisis PRS*, al ser el punto de partida para una nueva ejecución, *HU05: Subida de archivos genéticos*, proporcionando la interfaz para cargar los ficheros necesarios, *HU06: Selección de modelos PRS*, mediante el botón de selección de modelos y *HU07: Configuración del análisis*, al permitir al usuario personalizar los parámetros técnicos de la configuración del análisis.

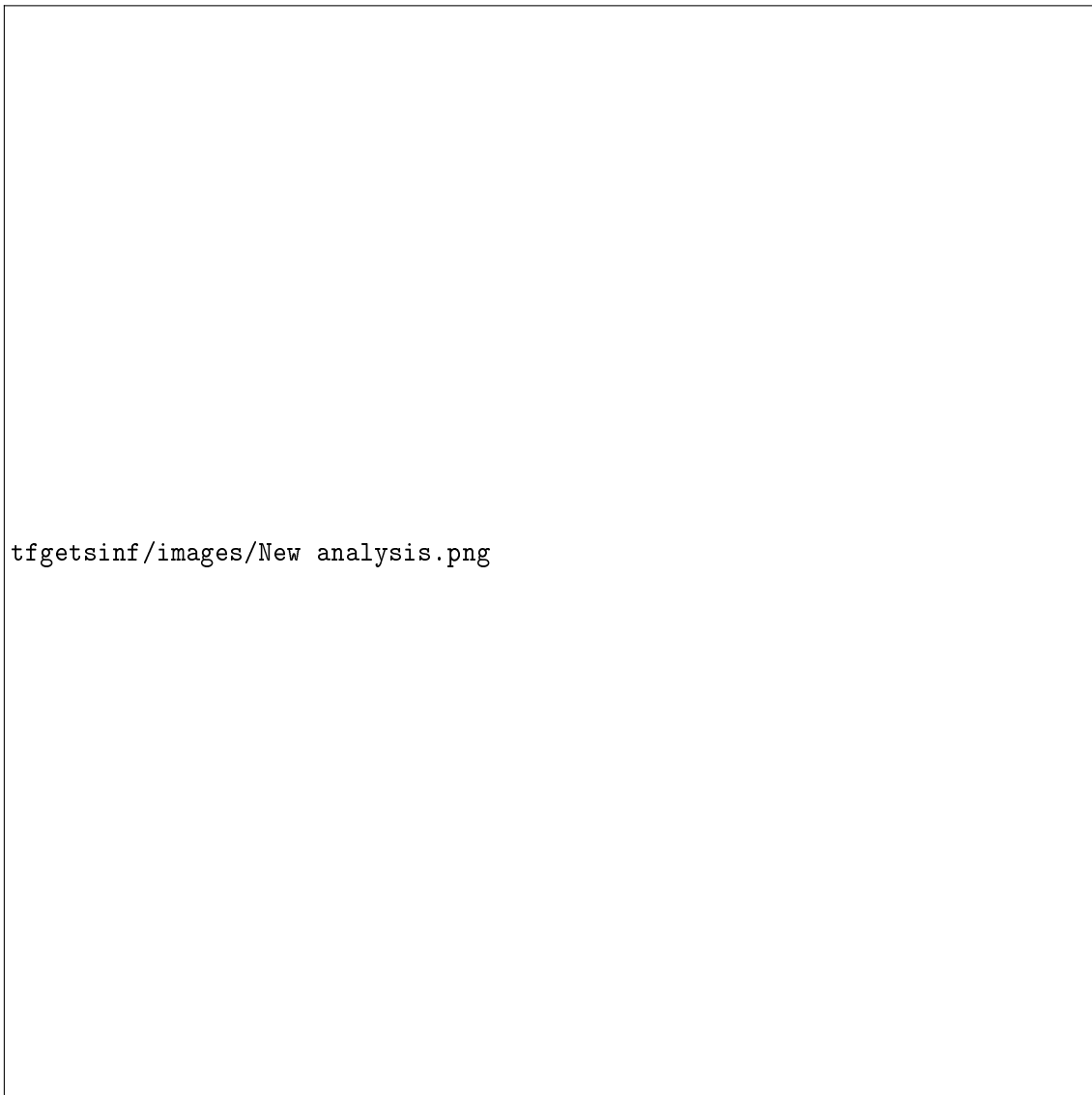


Figura 3.12: Mockup de Configuración de Análisis PRS

Capa de lógica de negocio (backend)

La lógica de negocio se implementa sobre el entorno de ejecución *Node.js*, aprovechando las *API Routes* nativas de *Next.js* para construir los *endpoints* de la API. La comunicación segura y tipada con la base de datos se delega al ORM *Prisma*, que se integra con *TypeScript* para garantizar la corrección de tipos en las consultas.

Capa de datos

Para la persistencia de datos, se ha seleccionado *MariaDB* como sistema gestor de base de datos relacional. El esquema de la base de datos y las migraciones se gestionan a través de los ficheros de definición de *Prisma*. En este apartado se va a detallar el diseño de la arquitectura planteada para el desarrollo de la herramienta propuesta, desde la construcción de la base de datos hasta la interfaz gráfica.

Diagrama y estructura de la base de datos En la siguiente Figura ??, se puede observar el diagrama de la base de datos; la estructura de este diagrama ha sido acordada junto

a otro equipo de investigación, el cual está desarrollando un sistema para la extracción, priorización y aplicación de modelos de puntuación de riesgo poligénico.

La relación con este trabajo es muy relevante, ya que el resultado de priorización de modelos será la lista a seleccionar desde la pantalla de configuración en el campo Models PRS.

En este apartado se ha hecho el paso de un diagrama de clases conceptual (Figura ??) a un diagrama físico de base de datos (Figura ??). En el modelo de dominio se modelan aspectos relevantes para describir, entender y comunicar el dominio, pero no se detalla cómo se van a almacenar los datos. Por el contrario, un diagrama de base de datos es un modelo robusto y detallado que indica cómo se tiene que almacenar la información. Este cambio no solo supone cambiar la notación, sino que además supone una ampliación del modelo para dar soporte a las necesidades del sistema.

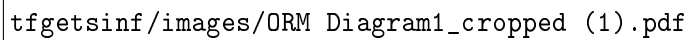
A continuación, se detallan los cambios más significativos entre ambos diagramas:

- **Nuevos Atributos en PRSAnalysis:** Se han incluido campos para la gestión de trabajos asíncronos (`queue_job_id`) y notificaciones (`notification_email`), que son detalles de implementación que surgen durante el diseño técnico.

- **Nuevas Entidades Añadidas:**

- **UserNotificationSettings:** Esta tabla almacena las preferencias de notificación personalizadas para cada usuario.
- **UserNotificationHistory:** Esta tabla actúa como un registro o historial de todas las notificaciones enviadas a un usuario.

- **Nueva Relación en PRSAnalysis:** La tabla PRSAnalysis ahora está vinculada a estas nuevas entidades. Cuando un análisis finaliza o falla, la aplicación consulta la tabla UserNotificationSettings para determinar cómo notificar al usuario y luego crea una nueva entrada en UserNotificationHistory para registrar el evento.



tfgetsinf/images/ORM Diagram1_cropped (1).pdf

Figura 3.13: Diagrama de entidad relación

3.2.8. Arquitectura física

El diseño físico comienza con el diagrama de despliegue representado en la Figura ??, que describe la arquitectura del sistema *PhenoScore* en términos de los distintos componentes físicos que lo conforman y cómo está distribuido en una máquina virtual. Un diseño modular como este permite definir mejor qué hace cada componente y poder llevar a cabo un mantenimiento o evolución importante del sistema.

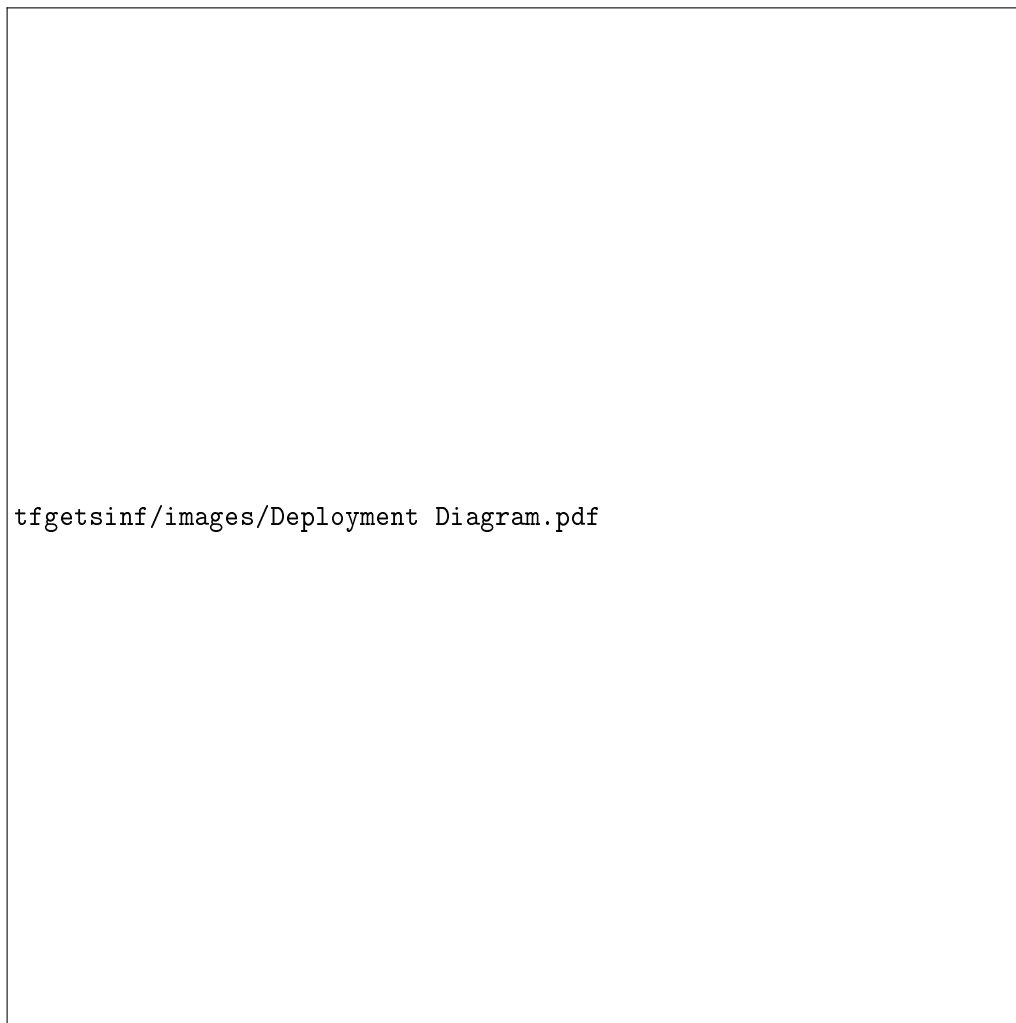


Figura 3.14: Diagrama de Despliegue

Para el despliegue de la aplicación, se ha optado por un enfoque centralizado en una máquina virtual (VM) dedicada, ubicada en *Google Cloud*. Esta VM ha sido configurada con las siguientes especificaciones:

- Sistema Operativo: *Debian 12 Bookworm*
- Tipo de Máquina: e2-standard-2
- CPU: 2 CPU virtuales
- Memoria RAM: 8 GB
- Almacenamiento (Disco de Arranque): 30 GB (Disco persistente balanceado)

Esta configuración permite tener un entorno independiente y estable, con el tamaño necesario para sostener las cargas de trabajo tanto de *frontend* como de *backend*, así como las pesadas operaciones de cálculo de análisis genético llevadas a cabo por *Genopred* y la gestión de archivos.

En cuanto a la comunicación con el exterior, la máquina virtual se comunica con los servicios esenciales para su correcto funcionamiento. En primer lugar, mantiene una vía abierta con *Auth0*, el proveedor de identidad, servicio de autorización y gestión segura

de accesos, evitando que la aplicación tenga que gestionar sus propios usuarios. Asimismo, establece conexión con la base de datos situada en otro servidor para guardar la información. Se puede consultar en esta arquitectura del sistema las especificaciones funcionales y tecnológicas de los componentes que se encuentran dentro de esta VM (Next, Orchestrator, Genopred, Prisma, etc.).

3.3 Desarrollo de la propuesta

Una vez definidos los requisitos y el diseño de la aplicación, este capítulo se centra en el proceso de implementación de la solución propuesta. Partiendo del diseño arquitectónico por capas, se detalla la construcción de cada uno de los componentes del sistema, desde la persistencia de los datos hasta la interfaz gráfica.

Asimismo, se exponen los desafíos técnicos encontrados durante el desarrollo y las decisiones tomadas para resolverlos, ofreciendo una visión completa del proceso de ejecución del proyecto y del resultado final obtenido.

3.3.1. Herramientas y metodología de desarrollo

Para garantizar la calidad, mantenibilidad y eficiencia durante la fase de implementación, se ha seguido una metodología de desarrollo estructurada y se ha hecho uso de un conjunto de herramientas estándar en la industria.

Entorno y control de versiones

El entorno de desarrollo principal ha sido *Visual Studio Code (VSC)*, un editor de código ligero y extensible. Por otro lado, el control de versiones del código fuente se ha gestionado utilizando *Git*, manteniendo un historial de cambios en un repositorio de *GitHub*. Esta práctica ha facilitado la colaboración, la trazabilidad y la posibilidad de revertir cambios si fuera necesario.

Estructura del proyecto

El código fuente se ha organizado siguiendo un enfoque modular para facilitar su mantenimiento y crecimiento. La estructura principal de directorios es la siguiente:

- **/prisma/**: Contiene el esquema de la base de datos definido para Prisma y los ficheros de migración generados automáticamente.
- **/public/**: Almacena todos los archivos estáticos que se usan públicamente, como imágenes y fuentes.
- **/src/app/**: Es el directorio principal del código fuente de la aplicación *Next.js*, donde se encuentran las páginas, las rutas de la API, los componentes reutilizables (*/components*), los *hooks* de *React* (*/hooks*).
- **/utils/**: Contiene funciones auxiliares y utilidades compartidas a lo largo de todo el proyecto.
- **/services/**: Contiene diferentes servicios como el *worker* de *BullMQ* responsable de ejecutar los análisis de PRS de forma desacoplada, el procesador principal de los análisis junto con el gestor de colas de trabajo y las notificaciones a través del correo electrónico.

Control y calidad de código

Para mantener un estilo de código consistente y detectar errores de forma temprana, se ha configurado *ESLint*. Este *linter* analiza el código estáticamente, asegurando el cumplimiento de las reglas de estilo y mejorando la legibilidad. Para la estilización, se ha utilizado *PostCSS* para mejorar los estilos CSS mediante *plugins* modernos.

Gestión de dependencias

El proyecto gestiona sus librerías y dependencias a través del gestor de paquetes *npm*. El fichero `package.json` actúa como manifiesto, listando todas las dependencias y los scripts necesarios para ejecutar y construir la aplicación.

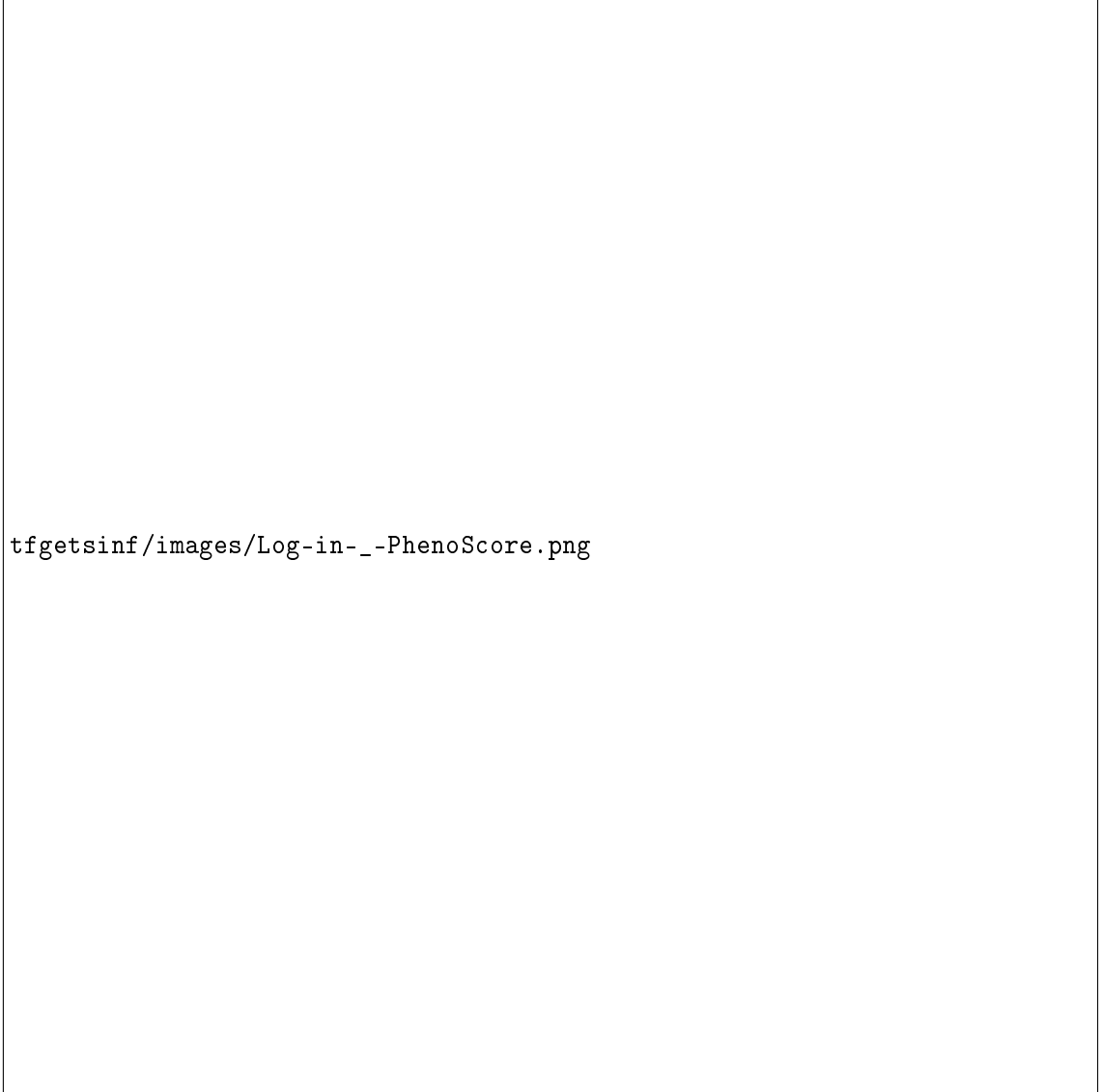
3.3.2. Implementación de la capa de presentación (frontend)

Esta capa es la responsable de la interfaz de usuario y la interacción directa con el usuario final. En este apartado se aborda la evolución de los componentes visuales desde los *mockups* de *Figma* hasta la interfaz final, justificando las decisiones de mejora tomadas durante la implementación. Para acelerar el desarrollo y garantizar una estética moderna y consistente, se ha utilizado la biblioteca de componentes *HeroUI*.

Inicio de sesión

En la pantalla de inicio de sesión, a diferencia del *mockup* inicial, que se componía de un formulario de *login* propio, en la versión final se ha integrado *Auth0*, por lo que la implementación específica ha sido a través de este. Esta modificación, como ha sido explicada anteriormente, delega la gestión de usuarios y contraseñas, lo cual fortalece la seguridad del sistema.

Del mismo modo, la barra de navegación que aparece tras el inicio de sesión se ha refinado para mejorar la usabilidad y la accesibilidad. Partiendo de una estructura básica en los *mockups*, la implementación final introduce una organización más funcional. Se han añadido botones de navegación explícitos para las secciones principales y un nuevo control para la gestión de notificaciones. Este último componente, añadido tras considerar el flujo de usuario en procesos largos, permite solicitar avisos por correo sobre la finalización de los análisis, mejorando considerablemente la experiencia de usuario.



tfgetsinf/images/Log-in-_-PhenoScore.png

Figura 3.15: UI Inicio de sesión

Visualización de análisis PRS

La implementación de la pantalla de visualización de análisis se ha centrado en optimizar la interacción del usuario y la gestión de la información, materializando las necesidades identificadas en la fase de diseño. Aunque la estructura central sigue siendo la tabla que lista los análisis, se han incorporado varios componentes para mejorar la usabilidad.

El diseño final integra los filtros directamente en la cabecera de la tabla, permitiendo al usuario acotar los resultados de forma rápida y contextual. Adicionalmente, se han implementado controles de visualización que no estaban detallados en los *mockups* iniciales, pero que son esenciales para una experiencia de usuario completa: un selector de columnas para personalizar la información mostrada y un control de paginación para gestionar eficientemente un gran volumen de análisis. Estas funcionalidades, implementadas siguiendo patrones de diseño de interfaces de datos, aseguran que la interacción del usuario con la tabla sea intuitiva y eficiente.

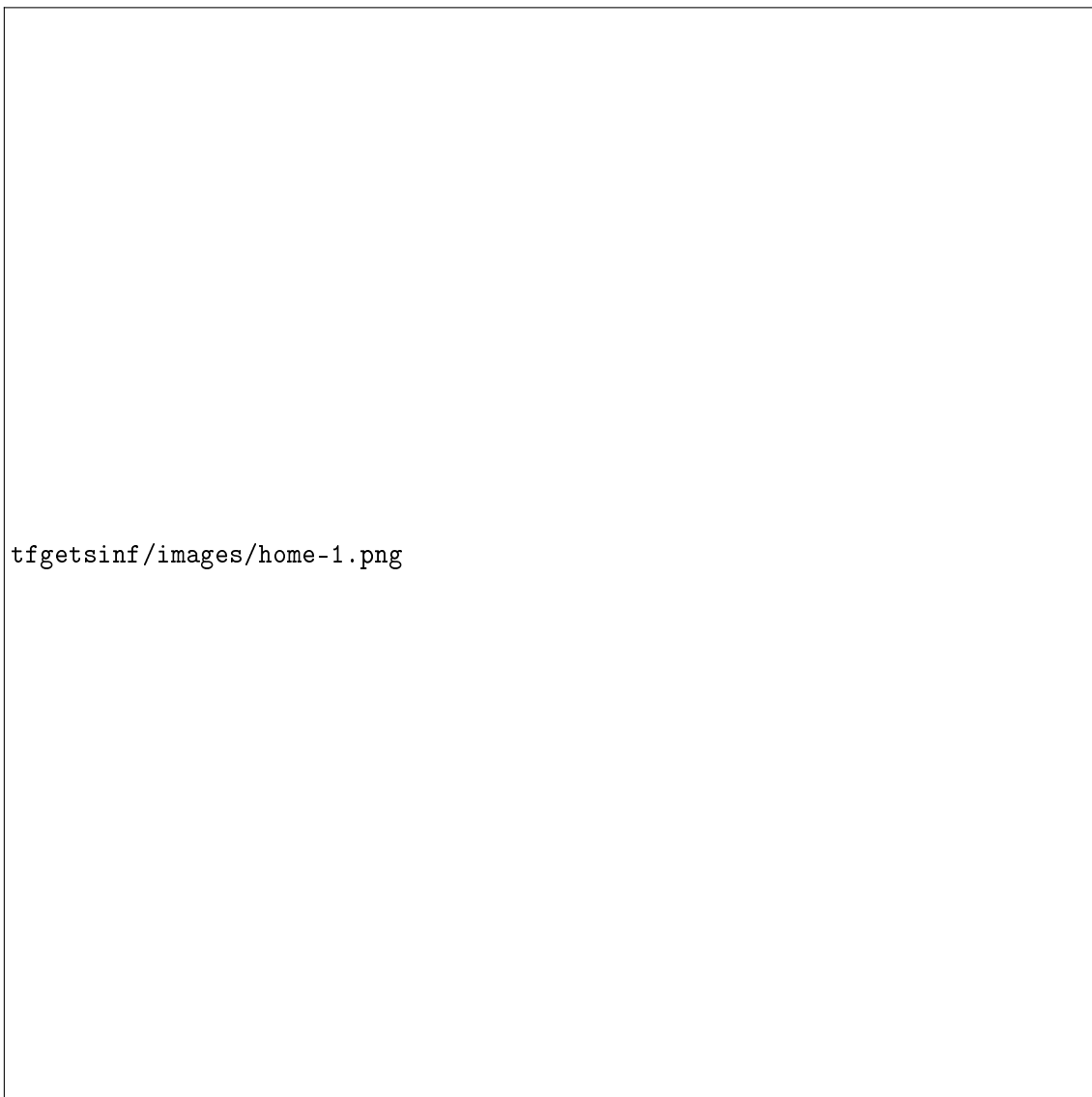


Figura 3.16: UI Visualización de análisis PRS

Configuración de análisis PRS

La pantalla de configuración es donde se concentran las modificaciones más relevantes y, además, donde se ha reflejado un cambio de necesidades por parte de los investigadores.

En primer lugar, se dividió el formulario en dos secciones claramente diferenciadas. Por un lado, los datos propios del análisis, como ficheros de entrada o parámetros, y, por otro lado, la configuración detallada de la población de referencia. Se aplicó el principio de separación de responsabilidades para que el proceso de configuración fuera más sencillo para el usuario; de esta forma se minimiza la posibilidad de errores.

En la sección de la población de referencia, se eliminó la opción explícita de "1000 Genomes Reference", estableciéndola como opción por defecto a menos que el usuario proporcione una alternativa. Además, se han añadido dos botones que mejoran la recopilación de datos: uno que dirige a una lista para completar la información sobre los detalles de la población de referencia cargada y otro que visualiza la distribución de la ascendencia. Esta última lista ha sido considerada de gran importancia, ya que permite

evaluar la categoría de ascendencia en la referencia para una estratificación poblacional precisa.

Finalmente, los botones tradicionales para subir ficheros fueron reemplazados por componentes de arrastrar y soltar. Este cambio, aparte de ser estético, ofrece una experiencia de usuario superior, al ser más interactivos y visuales. El uso de la librería *react-dropzone* ha facilitado la gestión de la carga y la validación de los ficheros.

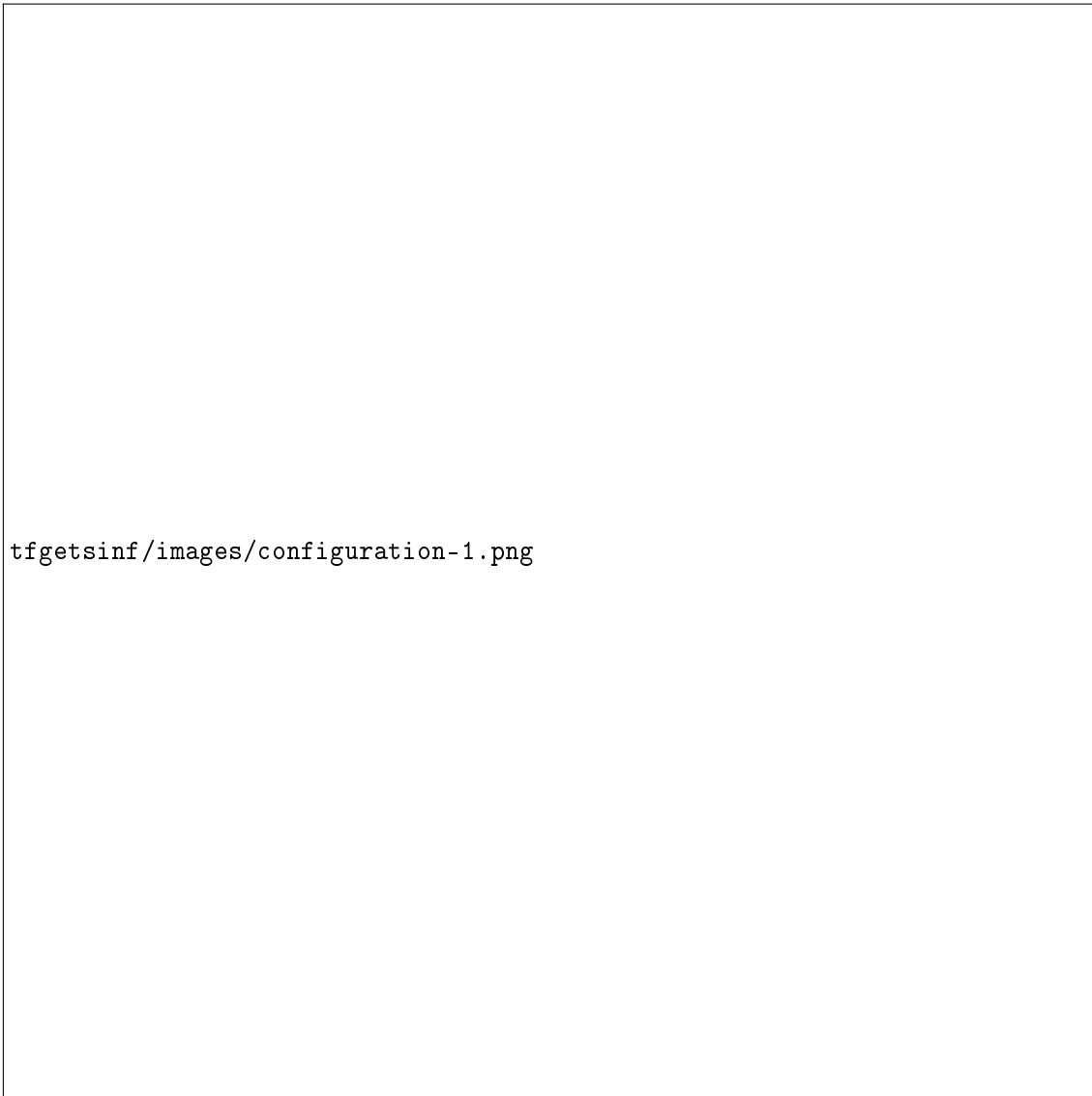


Figura 3.17: UI Configuración de análisis PRS

Configuración de las notificaciones

En este modal, abierto desde la configuración del usuario en el perfil, se encuentra la estructura de las opciones de preferencia sobre la funcionalidad de las notificaciones en la *app* y el correo. Además, se ofrece información del usuario de *Auth0* con el que se navega en la *app* y el email ligado a la cuenta.

Este sistema se basa en mantener a los usuarios informados del estado de sus análisis PRS. Acerca de las notificaciones en la *app*, se implementa un historial con los últimos avisos que no hayan sido descartados anteriormente. Por otro lado, la lógica por correo

ha hecho uso de diversos *endpoints* con los que transferir la información a los mensajes que posteriormente son enviados gracias a la configuración *SMTP*.

Esta arquitectura se activa de forma automática en el momento en que el *worker* de *BullMQ* finaliza un análisis, y, en ese instante, consulta las preferencias del usuario almacenadas en la base de datos y decide qué canales activar. Si el correo está habilitado, se genera un email con plantillas *HTML* personalizadas que incluye el estado del análisis y el reporte de resultados o el fichero de errores. Al mismo tiempo, se crea un aviso en la interfaz que se muestra como una alerta en tiempo real y se guarda en el historial del usuario.

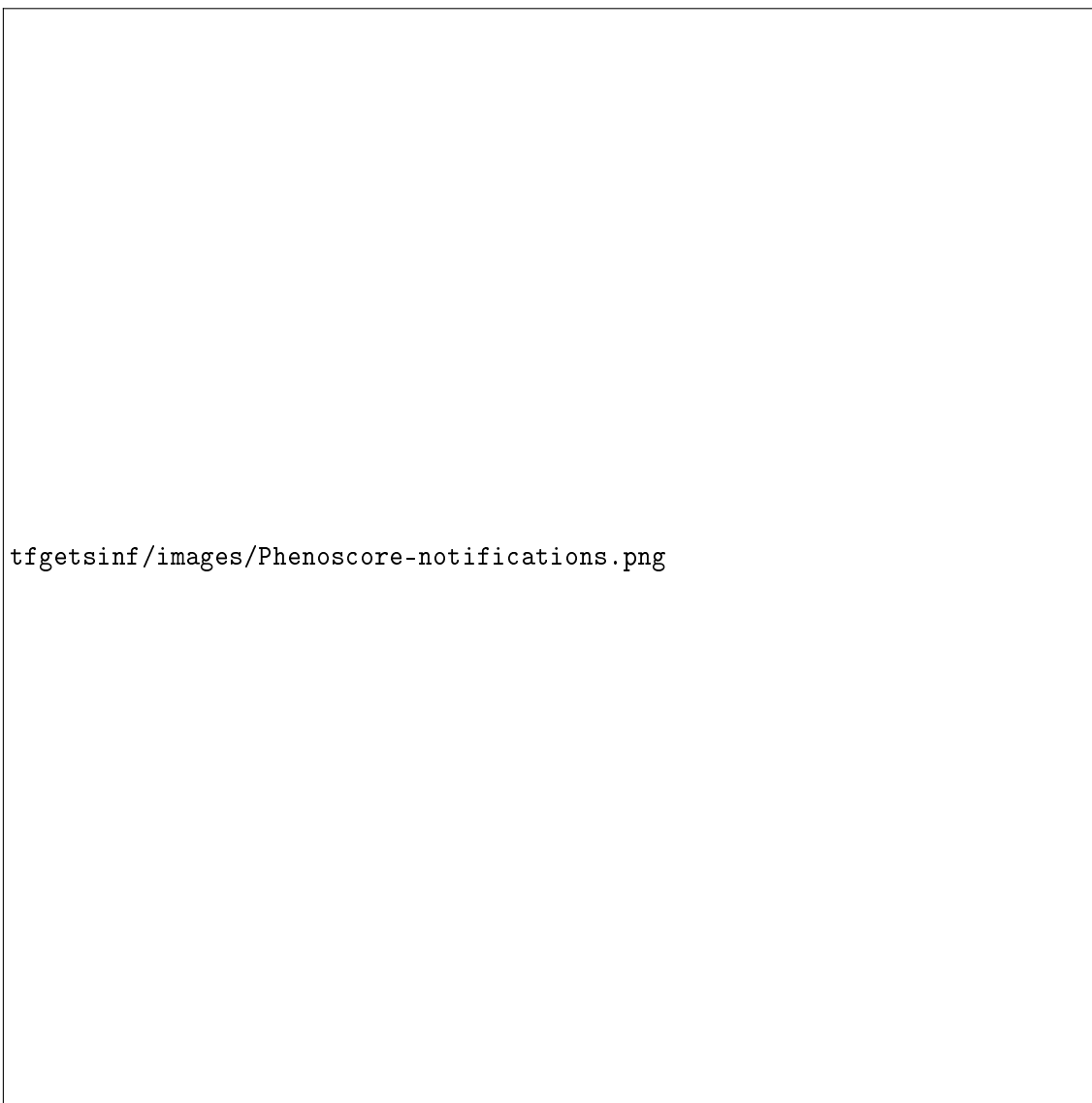


Figura 3.18: UI Configuración de las notificaciones

Gestión de tareas asíncronas en la interfaz

Con el objetivo de proporcionar una experiencia dinámica y ágil al usuario ante un sistema de backend asíncrono, la lógica completa del lado del cliente se encuentra encapsulada en un *hook* personalizado de *React*, *useLongAnalysis*. Este *hook* se encarga de gestionar un ciclo de vida completo de un análisis, desde su creación y seguimiento hasta la actualización de los datos en tiempo real de la interfaz.

Su funcionamiento es el siguiente:

- **Creación del Análisis:** Cuando el usuario envía el formulario de configuración, el *hook* cambia a un estado de carga (*creatingAnalysis*) para dar un *feedback* visual inmediato (por ejemplo deshabilitar el botón de envío). Hace una petición POST a `/api/analysis` con todos los datos del análisis y el identificador del usuario que ha extraído de *Auth0*. El servidor acepta la petición inmediatamente confirmando que se ha encolado el trabajo, y en ese momento el *hook* actualiza la lista general de análisis.
- **Monitorización del Estado mediante un Sistema de Polling Dual:** Para mantener la interfaz sincronizada con el estado real del backend, el *hook* implementa dos estrategias de *polling* complementarias:
 1. **Polling Específico (Rápido):** Cuando el usuario interactúa con un análisis en particular, se activa una función de *polling* recursiva que consulta el *endpoint* `/api/status?id={analysisId}` cada 2 segundos. Este *polling* se detiene automáticamente en cuanto el estado del análisis deja de ser *running*, asegurando una actualización casi en tiempo real con un uso eficiente de los recursos.
 2. **Polling General (Lento):** De forma paralela, un intervalo global se ejecuta cada 5 segundos para llamar a la función `fetchAnalyses`. Esto garantiza que la lista principal de análisis esté siempre actualizada, reflejando el progreso de trabajos que puedan haber finalizado en segundo plano, en otra pestaña del navegador o incluso en otro dispositivo.
- **Actualización Reactiva de la Interfaz:** El *hook* utiliza el sistema de estado de *React* (`useState`) para almacenar la lista completa de análisis y el estado del análisis actualmente monitorizado. Cualquier cambio obtenido a través del *polling* actualiza estos estados, lo que provoca que los componentes de la interfaz que los utilizan se re-rendericen automáticamente, mostrando al usuario la información más reciente sin necesidad de recargar la página.
- **Integración con la Seguridad:** La comunicación con la API está protegida por diseño. El *hook* depende del estado de autenticación de *Auth0* y no realiza ninguna petición si el usuario no está validado. El identificador único del usuario (`user.sub`) se adjunta a todas las llamadas a la API, permitiendo al backend filtrar y devolver únicamente los datos que pertenecen a dicho usuario.

Esta arquitectura, centrada en un *hook* centralizado, no solo permite una experiencia de usuario eficiente y fluida, sino que también favorece un código más modular, reutilizable y sencillo de mantener, a la vez que sigue las buenas prácticas en cuanto a seguridad y rendimiento.

3.3.3. Implementación de la capa de lógica de negocio (backend)

El backend es el núcleo funcional de la aplicación. Se encarga de la lógica de negocio, la coordinación de tareas complejas y la comunicación segura con los servicios externos.

Implementación de la API

La comunicación entre el frontend y el backend se realiza a través de una API *RESTful*.

En esta sección se explican detalladamente los *endpoints* de la API desarrollada, incluyendo las rutas disponibles, sus funciones específicas y los métodos *HTTP* implementados. La API gestiona todas las operaciones desde la carga de archivos, pasando por la creación y gestión de análisis, hasta la consulta de estados y resultados.

- `POST /api/upload`: Este *endpoint* permite subir archivos de pacientes y población de referencia al servidor. El proceso incluye: guardar el archivo del paciente (descomprimiendo si es *.zip*), almacenar el archivo de población en la ruta correspondiente (o usar el *path* fijo de 1000G si no se proporciona), registrar los datos en la base de datos y devolver un *JSON* con los *IDs* creados.
- `/api/analysis`:
 - `GET`: Devuelve todos los análisis ordenados por fecha descendente, incluyendo información del paciente, población de referencia y modelos priorizados asociados.
 - `POST`: Crea un nuevo análisis y lo añade a la cola de *BullMQ* para procesamiento en segundo plano. Recibe un *JSON* con los datos del análisis y devuelve el *ID* creado junto con una URL para consultar el estado.
 - `DELETE`: Elimina un análisis específico. Requiere el parámetro *id* en la *query string* y confirma la operación mediante *JSON*.
- `POST /api/analysis/cancel`: Cancela un análisis en curso, actualizando su estado a *'failed'* y marcando el resultado *HTML* como *'aborted'*. Recibe el *ID* del análisis vía *JSON* y confirma la operación exitosa.
- `GET /api/analysis-details`: Obtiene los detalles completos del análisis, es decir su configuración y otra información ligada a este, como la población de referencia o los modelos PRS.
- `POST /api/notifications/email`: Manda un email al usuario cuando un análisis se ha ejecutado exitosamente o cuando ha fallado, adjuntando al correo el informe o el error detallado respectivamente.
- `/api/notifications/read`:
 - `GET`: Obtiene todas las notificaciones que no han sido leídas aún por el usuario y las devuelve a la lista.
 - `POST`: Marca una notificación como leída y por lo tanto ya no vuelve a aparecer en la lista de notificaciones.
 - `DELETE`: Marca todas las notificaciones como leídas y la lista se queda vacía.
- `/api/notifications/settings`:
 - `GET`: Obtiene la configuración de las notificaciones del usuario.
 - `POST`: Almacena una nueva configuración de notificaciones.
- `/api/queue`:
 - `GET`: Devuelve la información del estado de la cola como los trabajos del usuario que se están ejecutando
 - `POST`: Se constituye por varias gestiones de la cola como la limpieza de trabajos antiguos o problemáticos.

- GET /api/status: Consulta el estado de un análisis específico mediante el parámetro *id* en la *query string*. Devuelve el análisis completo o un error si no existe.
- POST /api/upload: Recibe archivos de pacientes y poblaciones de referencia, se procesan y preparan para formar parte de la ejecución de los análisis.

Ejecución asíncrona de los análisis PRS

La ejecución de un análisis de Puntuación de Riesgo Poligénico (PRS) es la funcionalidad central de la aplicación y, debido a su complejidad, puede tardar hasta 25 minutos en ejecutarse. Por tanto, en vez de ejecutarlo de forma síncrona en el hilo principal porque bloquearía por completo la aplicación, se ha optado por diseñar una arquitectura asíncrona robusta basada en un sistema de colas y procesos de trabajo, cuya implementación y desafíos se detallan a continuación.

Arquitectura asíncrona: Gestor de Cola y Procesador de Análisis

La arquitectura se divide en dos componentes principales que colaboran para ejecutar los análisis de forma desacoplada:

- **Gestor de Trabajos (AutoWorker):** Este componente actúa como el intermediario entre la cola de trabajos y el procesador de análisis. Su responsabilidad es exclusivamente la gestión de la cola. Está orquestado por *BullMQ*, un sistema de colas que utiliza Redis como intermediario de mensajería. Para asegurar la seguridad y flexibilidad, la conexión con Redis se configura mediante variables de entorno externalizadas. Cuando un nuevo trabajo de análisis es añadido a la cola por la API, el *AutoWorker* lo recoge y delega su ejecución al *AnalysisProcessor*. Esta capa de gestión también se encarga de la fiabilidad del sistema, configurando políticas de reintentos con un tiempo de espera exponencial para manejar fallos temporales sin sobrecargar el sistema.
- **Procesador de Análisis (AnalysisProcessor):** Este servicio, implementado en *services/analysisProcessor.ts*, es el corazón de la lógica de procesamiento de los análisis genéticos. No sabe nada sobre el sistema de colas; su único propósito es recibir los datos de un análisis y ejecutarlo. El proceso que sigue es el siguiente:
 1. **Recuperación del Contexto:** Al recibir un trabajo, lo primero que hace es realizar una consulta completa a la base de datos para obtener todos los detalles del análisis: parámetros, datos del paciente, población de referencia y los modelos PRS seleccionados.
 2. **Preparación del Entorno:** Crea un directorio de ejecución único para el análisis y, dentro de este, genera los ficheros de configuración que la herramienta *GenoPred* necesita como entrada: el fichero *score_list.txt* (que lista los modelos a usar), el fichero *target_list.txt* (que define la ruta al fichero del paciente y el formato del mismo) y el fichero *config.yaml* (que construye la entrada para el *pipeline Snakemake*). Para aportar flexibilidad, la localización de los ficheros de genotipado se gestiona de forma dinámica, permitiendo que la estructura de carpetas de entrada sea variable.
 3. **Invocación del Proceso Externo:** Utiliza el módulo *child_process* de *Node.js* para invocar el comando de ejecución del *pipeline* de *GenoPred*, activando primero el entorno *Conda* necesario. Mientras el *pipeline* se ejecuta, el procesador monitoriza la salida y actualiza el estado del análisis en la base de datos.

4. **Gestión del Resultado:** Si se finaliza con éxito el trabajo, el procesador recupera el *HTML* resultante, actualiza el estado de la base de datos a 'completado' y guarda la ruta al archivo *HTML* para verlo más tarde. En caso de un fallo, captura el error, lo registra y actualiza el estado a 'fallido'.
5. **Cierre Seguro:** El procesador implementa un mecanismo de apagado seguro (*graceful shutdown*). Escucha las señales del sistema (como una orden de apagado) y cierra las conexiones a Redis y Prisma de manera segura para evitar la corrupción de datos y asegurarse de que no quede ningún trabajo pendiente en estado inconsciente.

Tal separación de responsabilidades entre el gestor de colas y el procesador de análisis conduce a un diseño mucho más claro y mantenible. *AutoWorker* se encarga de la infraestructura de mensajería, dejando que *AnalysisProcessor* se concentre únicamente en la lógica de negocio relacionada con el análisis bioinformático, resolviendo así el gran desafío de ejecutar procesos pesados sin bloquear la aplicación y de manera modular.

Desafíos en la Implementación del Worker

Mientras se estaba llevando a cabo la implementación del *worker*, hubo varios desafíos que requirieron replantear la arquitectura de despliegue, además de realizar cambios a causa de un error del cual no se encontraba la causa.

El primer obstáculo surgió al elegir el entorno de ejecución basado en *Windows*, usando el subsistema de *Windows* para *Linux* (WSL), para la ejecución de la *pipeline* *GenoPred*. Esta arquitectura llevó a desarrollar una lógica específica para la traducción de rutas de ficheros al formato *Linux*, lo cual fue indispensable para que el *pipeline* localizara los datos necesarios.

Por otra parte, en el inicio se planteó la posibilidad de cambiar uno de los parámetros de la configuración del análisis, el '*overlap threshold*', el cual se refiere al porcentaje de coincidencia entre las variantes genéticas (SNPs) que aparecen en el archivo de puntuación PGS y las variantes que existen en los datos de referencia de *GenoPred*. Si este porcentaje es menor del 75 % se descarta el archivo y no se calcula la puntuación para esa muestra. Posteriormente, se tuvo que descartar la idea de poder variar este parámetro, ya que en el *pipeline* no se ha considerado esa opción, por lo tanto no solo depende de los ficheros de entrada sino que se tendría que cambiar también el código interno de la herramienta, lo cual no entra dentro de las posibilidades del proyecto.

Un mayor imprevisto surgió tras una actualización del *pipeline* *GenoPred*, ya que la nueva versión introdujo cambios que rompían con la anterior compatibilidad, dando un aviso de error relacionado con el reporte individual de puntuación poligénica. Al intentar reinstalar la herramienta con sus nuevas dependencias, el entorno WSL empezó a dar errores fatales, impidiendo seguir con el desarrollo de la aplicación; ante este percance se tomó la decisión de migrar todo el entorno a una máquina virtual con un sistema operativo *Linux*. Este cambio proporcionó un entorno más robusto y predecible.

A pesar de la estabilidad ganada con el nuevo entorno, los análisis seguían fallando debido al *bug* introducido en *GenoPred*. Tras depurar el análisis y recurrir a la documentación, se tomó la decisión de contactar directamente con el desarrollador principal de la herramienta. Su colaboración fue fundamental; amablemente confirmó la existencia del *bug* en la nueva versión e introdujo unos nuevos cambios libres de errores de código. Gracias a estos, el problema se resolvió y los análisis se volvieron a ejecutar correctamente.

Por último, se tomó la decisión de realizar una última migración, el despliegue de la solución en una máquina virtual en la nube a través de *Google Cloud Platform* (GCP), para facilitar en un futuro la puesta en producción de la herramienta, garantizando así la escalabilidad y disponibilidad del servicio que se busca proporcionar desde un inicio.

Gestión de autenticación y seguridad

Para la gestión de la autenticación y la autorización de usuarios, se ha integrado el servicio externo *Auth0*. Esta decisión delega la responsabilidad de gestionar contraseñas, sesiones y *cookies* a un proveedor especializado, lo que resulta en una solución más segura y robusta que una implementación propia. *Auth0* proporciona, además, funcionalidades avanzadas como el inicio de sesión a través de redes sociales.

3.3.4. Implementación de la capa de datos y persistencia

Esta capa es la responsable de la interacción con la base de datos y el almacenamiento en memoria.

Para la gestión de los datos relacionales, se ha utilizado *Prisma*, un *ORM* (*Object-Relational Mapping*) de nueva generación. *Prisma* permite definir el esquema de la base de datos de forma tipada en el fichero `schema.prisma`, facilitando las migraciones y generando un cliente seguro para realizar consultas, lo que reduce drásticamente la posibilidad de errores en la interacción con la base de datos.

Para la gestión de la cola de trabajos asíncronos, se ha utilizado *Redis*, una base de datos en memoria que actúa como intermediario de mensajes para *BullMQ*. Su alta velocidad es ideal para gestionar el estado de las colas de manera eficiente.

CAPÍTULO 4

Validación de la solución

4.1 Metodología de la validación

El objetivo principal de la validación es asegurar que la solución desarrollada, *PhenoScore*, cumpla con los requisitos funcionales definidos y sea aceptada por los usuarios finales como una herramienta útil, eficiente y fácil de usar para el análisis de *Polygenic Risk Scores* (PRS). Para ello, se ha diseñado un enfoque de validación integral basado en tres dimensiones complementarias:

1. Validación funcional: Se verifica que las funcionalidades principales de la herramienta operan correctamente, permitiendo a los usuarios realizar las tareas clave sin errores críticos.
2. Validación de usabilidad: Se evalúa la facilidad de uso y la satisfacción de los usuarios mediante la observación directa y la aplicación de cuestionarios específicos.
3. Validación de aceptación tecnológica: Se mide la intención de uso y la aceptación de *PhenoScore* por parte de los usuarios utilizando el modelo de aceptación tecnológica (*Technology Acceptance Model, TAM*)[?].

Para estructurar esta validación, se establecen tres capas o niveles de análisis:

1. *User Stories*: Se evalúa si los usuarios pueden completar las tareas definidas en las historias de usuario, proporcionando una validación desde la perspectiva del usuario.
2. Requisitos no funcionales: Se realiza una medición técnica directa de aspectos como rendimiento, seguridad y tiempos de respuesta para comprobar el cumplimiento de los requisitos no funcionales establecidos.
3. Modelo *TAM*: Se aplica un cuestionario basado en una escala *Likert* para evaluar la percepción de utilidad, facilidad de uso e intención de uso futuro.

4.1.1. Participantes

La validación de *PhenoScore* se lleva a cabo con la participación de un grupo reducido pero representativo de usuarios que reflejan los diferentes perfiles de los *stakeholders* involucrados en el uso y desarrollo de la herramienta. Estos usuarios realizan una serie de tareas guiadas y responden a cuestionarios que permiten obtener datos cualitativos y cuantitativos para el análisis de los resultados.

Los usuarios seleccionados para la validación son los siguientes:

- **Perfil 1: Investigador Experto en Genética (Usuario Final Científico).** Un investigador con profundos conocimientos en el análisis de Puntuaciones de Riesgo Poligénico (PRS) y biología. Este perfil, familiarizado con las herramientas científicas existentes, es ideal para evaluar la adecuación funcional y la integración tecnológica de la plataforma en un contexto de investigación real.
- **Perfil 2: Ingeniero de Software Experto (Usuario Técnico).** Un experto en informática con amplia experiencia en el desarrollo de software, pero con conocimientos limitados en biología y análisis genéticos. Su participación aporta una perspectiva técnica crítica, centrada en evaluar la usabilidad desde un punto de vista informático, la robustez del sistema y la calidad de la implementación.
- **Perfil 3: Biólogo No Técnico (Usuario Final Objetivo).** Un profesional con formación y experiencia en biología y genética, pero con un conocimiento limitado en informática y aspectos técnicos de software. Este perfil representa al grupo de usuarios finales objetivo de la plataforma. Su participación es esencial para evaluar la facilidad de uso, la accesibilidad y la adecuación de la interfaz para un usuario no experto en el dominio técnico.

La selección de estos perfiles permite cubrir las distintas dimensiones de validación, asegurando una evaluación equilibrada desde los puntos de vista científico, técnico y de usuario final.

4.1.2. Procedimiento

El procedimiento de validación ha sido diseñado para ser completado en una única sesión por cada participante, con una duración aproximada de 30-45 minutos, y ha sido estructurado en dos fases principales:

1. Sesión de prueba guiada (25-35 minutos):
 - a) Introducción (5 min): Se realiza una breve presentación de *PhenoScore*, explicando sus objetivos y funcionalidades principales.
 - b) Realización de Tareas (20-30 min): Se solicita a cada participante que realice una serie de tareas representativas, diseñadas a partir de las *user stories* principales (iniciar sesión, crear un análisis, configurar parámetros, ejecutarlo y visualizar resultados). Durante este proceso, un investigador observa directamente la interacción del usuario con la plataforma, tomando notas sobre el tiempo, los errores y la facilidad para completar cada tarea.
2. Cuestionarios post-prueba (5-10 minutos): Al finalizar la sesión práctica, se pide a los participantes que completen un cuestionario basado en el Modelo de Aceptación Tecnológica (*TAM*), con el fin de recoger su percepción subjetiva sobre la herramienta.

4.1.3. Instrumentos de medición

Para recoger los datos de manera estructurada se han diseñado tres instrumentos de medición principales, cada uno enfocado en una dimensión diferente de la evaluación de la plataforma.

Guión de tareas guiadas (validación funcional)

Para evaluar la funcionalidad principal y la usabilidad de la plataforma, se ha preparado un guion de tareas basado en las historias de usuario más relevantes. Durante la sesión de validación, se le ha pedido a cada participante que complete estas tareas en secuencia, mientras un observador toma notas sobre el proceso. A cada participante se le ha proporcionado un contexto inicial y se le ha guiado a través de los siguientes escenarios:

Tarea (HU Asociada)	Descripción del Escenario y Pasos a Realizar
HU01: Iniciar Sesión	<p>Objetivo: Acceder a la plataforma de forma segura.</p> <p>Pasos: <i>Inicie sesión en la aplicación utilizando las credenciales proporcionadas para acceder a su espacio de trabajo personal.</i></p>
HU02 y HU04: Explorar y Gestionar el Panel	<p>Objetivo: Familiarizarse con la interfaz principal, localizar y gestionar un análisis existente.</p> <p>Pasos: <i>Una vez dentro, explore el panel de control. Localice un análisis y usando los botones de acción, compruebe su configuración y, posteriormente, elimínelo.</i></p>
HU03, HU05-07: Crear y Configurar un Análisis	<p>Objetivo: Completar el flujo de creación de un nuevo análisis desde cero.</p> <p>Pasos: <i>A continuación, cree un nuevo análisis. Asígnele un nombre, suba el fichero genómico correspondiente, seleccione dos modelos PRS de su elección y configure el umbral de ancestría."</i></p>
HU08: Ejecutar el Análisis	<p>Objetivo: Iniciar el procesamiento del análisis configurado.</p> <p>Pasos: <i>Una vez configurado, inicie la ejecución del análisis. Verifique que el nuevo análisis aparece en el panel de control con el estado running.</i></p>
HU09 y HU10: Visualizar y Exportar Resultados	<p>Objetivo: Acceder al informe de un análisis completado y exportarlo.</p> <p>Pasos: <i>Por último, localice un análisis previamente completado, acceda a su informe de resultados y utilice la opción para exportarlo."</i></p>

Tabla 4.1: Guion de tareas guiadas para la validación funcional.

El criterio de éxito para cada tarea ha sido su finalización exitosa por parte del usuario, sin errores críticos y en un tiempo considerado razonable. La medición se ha realizado mediante observación directa, registrando una variable binaria (✓ Cumplida / ✗ No cumplida) y anotando cualitativamente cualquier dificultad o comentario expresado por el participante.

Requisitos no funcionales (RNF):

La validación de los RNF se ha realizado mediante una combinación de inspección técnica, pruebas específicas y el uso de herramientas automatizadas:

Requisito	Comprobación
RNF-001 (Seguridad)	Verificación de la correcta implementación de <i>Auth0</i> y la presencia del cifrado <i>HTTPS</i>
RNF-002 (Tiempo de carga)	Medición directa del tiempo de carga del panel de control principal usando las herramientas de desarrollo del navegador
RNF-003 (<i>Lighthouse</i>)	Ejecución de una auditoría <i>Lighthouse</i> sobre la aplicación desplegada para obtener métricas de rendimiento y buenas prácticas
RNF-004 (Interfaz intuitiva)	La plataforma debe ser percibida como fácil de usar por los usuarios y llevar a cabo las tareas sin errores.

Tabla 4.2: Comprobación de requisitos no funcionales

Modelo de aceptación tecnológica (TAM):

Se ha utilizado un cuestionario con ítems formulados en una escala *Likert* de 5 puntos (de 1: Totalmente en desacuerdo, a 5: Totalmente de acuerdo). El cuestionario se ha dividido en las tres dimensiones clásicas del modelo:

Elemento	Ítem
PEOU (Facilidad de Uso Percibida)	
PEOU1	Creo que es fácil configurar y ejecutar análisis de PRS en la plataforma.
PEOU2	Me resultó fácil cargar y gestionar archivos genómicos (<i>VCF</i> , <i>PLINK</i> , etc.).
PEOU3	Creo que la información mostrada sobre modelos PRS y poblaciones de referencia es adecuada y suficiente.
PEOU4	Pienso que la forma en que se visualizan los resultados de PRS es útil y comprensible.
PEOU5	En general, encontré <i>PhenoScore</i> intuitiva y fácil de usar.
PU (Utilidad Percibida)	
PU1	Creo que los resultados de análisis PRS mostrados son claros y útiles para mi trabajo.
PU2	Creo que <i>PhenoScore</i> reduciría el tiempo y esfuerzo requerido para realizar análisis PRS.
PU3	Creo que <i>PhenoScore</i> ofrece una solución efectiva para ejecutar análisis de riesgo poligénico.
PU4	Creo que <i>PhenoScore</i> mejora mi capacidad para gestionar y organizar análisis genómicos.
PU5	En general, pienso que <i>PhenoScore</i> es una herramienta útil para mi práctica profesional.
ITU (Intención de Uso)	
ITU1	Tengo intención de usar <i>PhenoScore</i> regularmente en mi trabajo futuro.
ITU2	Recomendaría <i>PhenoScore</i> a mis colegas para realizar análisis PRS.

Tabla 4.3: Ítems de validación TAM (PEOU, PU e ITU).

4.2 Resultados

A continuación, se presentan los resultados obtenidos de la sesión de validación, organizados según las tres capas metodológicas definidas.

- **Validación funcional:** Verificación de que las funcionalidades principales funcionan correctamente
- **Validación de usabilidad:** Evaluación de la facilidad de uso y satisfacción del usuario
- **Validación no funcional:** Medición directa de requisitos de rendimiento, seguridad y escalabilidad
- **Validación de aceptación tecnológica:** Medición de la intención de uso mediante el modelo *TAM*

4.2.1. Validación funcional y de usabilidad

La validación funcional se ha evaluado mediante la observación directa de los participantes mientras ejecutan el guion de tareas guiadas (descrito en la Tabla ??). El objetivo es verificar si las funcionalidades principales operan correctamente y si los usuarios pueden completar las tareas clave de forma autónoma.

Resultados de la Ejecución de Tareas

Los tres participantes han sido capaces de completar con éxito el 100 % de las tareas propuestas en el guion. No se han registrado errores críticos que hayan impedido la finalización de ninguno de los flujos de trabajo principales, desde el inicio de sesión hasta la creación, ejecución y visualización de resultados de un análisis.

Observaciones Cualitativas de Usabilidad

Además de la tasa de éxito, se han registrado observaciones cualitativas y los tiempos de ejecución, que sirven como un indicador indirecto de la facilidad de uso y la curva de aprendizaje de la plataforma para cada perfil de usuario. Los tiempos totales para completar el guion completo han sido:

- **Perfil 1** (Investigador Experto en Genética): 18 minutos.
- **Perfil 2** (Ingeniero de Software Experto): 10 minutos.
- **Perfil 3** (Biólogo No Técnico - Usuario Objetivo): 23 minutos.

Análisis de los Resultados de Usabilidad

Los resultados son consistentes con las expectativas. El Perfil 2 (Ingeniero), con alta familiaridad en el uso de aplicaciones web complejas, ha completado las tareas en el menor tiempo, lo que sugiere que la interfaz sigue patrones de diseño intuitivos para un usuario técnico.

El Perfil 1 (Investigador en Genética), aunque experto en el dominio del problema, ha tardado un tiempo intermedio, probablemente debido a la fase de adaptación a una nueva herramienta y a una exploración más detallada de las opciones bioinformáticas.

Es de especial interés el resultado del Perfil 3 (Usuario Objetivo). A pesar de ser el perfil con menor conocimiento técnico, ha sido capaz de completar todas las tareas de forma

autónoma en un tiempo muy razonable (23 minutos). Esto constituye una evidencia sólida de que la plataforma cumple su principal objetivo de reducir la barrera tecnológica y ser accesible para usuarios no expertos en informática. Aunque su tiempo ha sido el más elevado, el éxito en la finalización de las tareas indica que la interfaz es lo suficientemente clara para este perfil.

En conclusión, la validación funcional ha sido exitosa, demostrando que la plataforma es robusta y cumple con los requisitos definidos en las historias de usuario.

4.2.2. Validación de requisitos no funcionales

La evaluación de los RNF se ha centrado en aspectos de rendimiento y seguridad, mediante mediciones directas y el uso de herramientas automatizadas.

- **RNF-001 (Seguridad):** Se ha verificado mediante inspección que toda la comunicación con el servidor se realiza a través de *HTTPS*, garantizando el cifrado de los datos. La integración con *Auth0* delega la gestión de credenciales a un proveedor especializado, cumpliendo con los estándares de seguridad. **Resultado:** Cumplido.
- **RNF-002 (Tiempo de Carga):** Utilizando las herramientas de desarrollo del navegador, se ha medido el tiempo de carga del panel de control principal. El tiempo promedio de carga completa del *DOM* ha sido de 1.8 segundos en una conexión de red estándar, cumpliendo el requisito de ser inferior a 2 segundos. **Resultado:** Cumplido.
- **RNF-003(Rendimiento General- *Lighthouse*):** La evaluación de este requisito, que implica la ejecución de una auditoría automatizada con *Google Lighthouse*, no ha podido llevarse a cabo. *Lighthouse* es una herramienta estándar en la industria que analiza una *URL* pública para generar un informe detallado sobre el rendimiento, la accesibilidad, las mejores prácticas y el *SEO* de una aplicación web. Su ejecución es fundamental para obtener una métrica objetiva y comparable de la calidad técnica del *frontend*.

Sin embargo, para que la herramienta pueda analizar la plataforma, esta debe estar desplegada en un servidor accesible públicamente (un entorno de producción o de *staging*). Dado que la validación se ha realizado en un entorno de desarrollo local, no se cumplía este prerrequisito. La verificación de este RNF queda, por tanto, pendiente para una futura fase de despliegue del proyecto. **Resultado:** No verificado.

- **RNF-004 (Interfaz intuitiva):** Durante la prueba guiada, se verificó que el 100 % de los participantes, incluido el perfil no técnico, completó todas las tareas clave con éxito y sin errores críticos. Esto confirma que la interfaz es lo suficientemente clara para cumplir sus objetivos funcionales. La evidencia combinada de la ejecución de tareas y la alta puntuación en el *TAM* confirman que la plataforma posee una interfaz usable e intuitiva. **Resultado:** Cumplido.

4.2.3. Modelo de aceptación tecnológica (TAM)

Además de la evaluación técnica, es conveniente conocer la percepción que tienen los usuarios finales de la herramienta. Para ello se ha aplicado el Modelo de Aceptación Tecnológica (*TAM*) mediante un cuestionario post-prueba. Este instrumento mide la Facilidad de Uso Percibida (*PEOU*), Utilidad Percibida (*PU*) e Intención de Uso (*ITU*) de la aplicación. La Figura ?? resume visualmente los resultados de la encuesta mostrando el grado de acuerdo de los participantes con cada una de las afirmaciones del modelo.

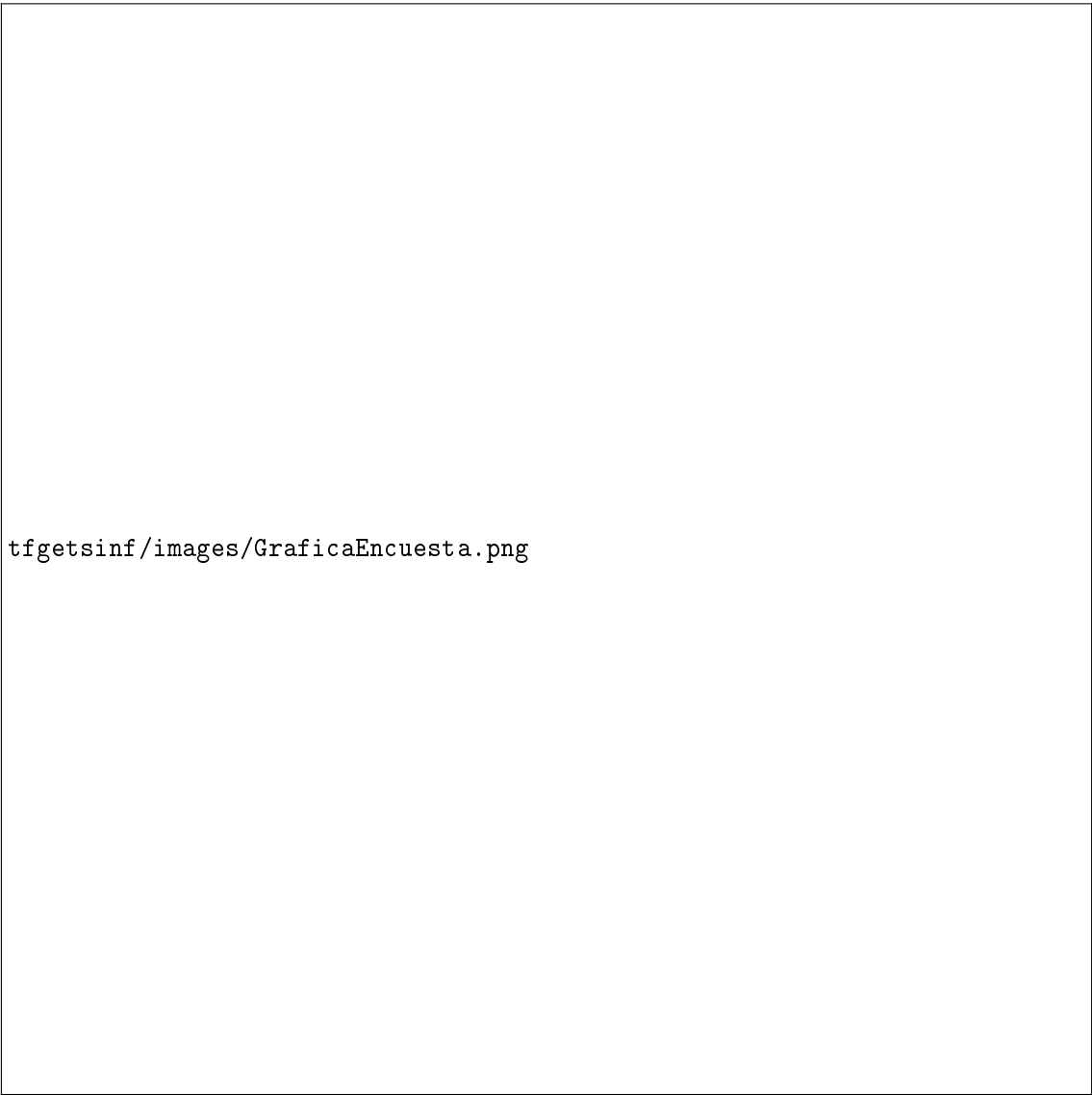


Figura 4.1: Desglose de Resultados de la Encuesta

4.2.4. Matriz de trazabilidad

Para recopilar y consolidar los resultados de la validación se ha elaborado la matriz de trazabilidad a continuación. Esta tabla relaciona cada requisito clave del proyecto (funcional, no funcional y de aceptación) con el medio de validación empleado, el criterio de éxito encargado y el resultado final obtenido.

Elemento a Validar	Método de Validación	Criterio de Aceptación	Resultado
Historias de Usuario (Funcionalidad y Usabilidad)			
HU01: Iniciar sesión	Observación directa	El 100 % de los usuarios completa el <i>login</i> sin ayuda.	✓

Continues on the next page

Continues from previous page

Elemento a Validar	Método de Validación	Criterio de Aceptación	Resultado
HU02: Panel de control	Observación directa	El 100 % de los usuarios localiza el panel y entiende la información.	✓
HU03, HU05-07: Crear y configurar un análisis	Observación directa	El 100 % de los usuarios completa el formulario de creación y configuración sin errores críticos.	✓
HU04: Gestión de análisis	Observación directa	El 100 % de los usuarios es capaz de filtrar, ver detalles y eliminar un análisis.	✓
HU08: Ejecución del análisis	Observación directa	El 100 % de los usuarios inicia un análisis y verifica su estado <i>running</i> .	✓
HU09: Visualización de resultados	Observación directa	El 100 % de los usuarios accede correctamente al informe de resultados.	✓
HU10: Exportación de resultados	Observación directa	El 100 % de los usuarios descarga con éxito el informe del análisis.	✓
HU11: Notificaciones	Inspección de correo y UI	El sistema envía notificaciones por correo y en la app al finalizar un análisis.	✓
Requisitos No Funcionales (Rendimiento y Seguridad)			
RNF-001: Seguridad	Inspección técnica	Uso de <i>HTTPS</i> y proveedor de identidad externo (<i>Auth0</i>).	✓
RNF-002: Tiempo de carga	Medición directa	El tiempo de carga del panel de control es inferior a 2 segundos.	✓
RNF-003: Rendimiento <i>Lighthouse</i>	Auditoría automatizada	La puntuación de rendimiento en <i>Lighthouse</i> es ≥ 90 .	✗

Continues on the next page

Continues from previous page

Elemento a Validar	Método de Validación	Criterio de Aceptación	Resultado
RNF-004: Interfaz intuitiva	Observación directa	El 100 % de los usuarios completa las tareas sin errores críticos.	✓
Modelo de Aceptación Tecnológica (TAM)			
PEOU: Facilidad de Uso Percibida	Cuestionario TAM	La puntuación media de los ítems PEOU es $\geq 4/5$.	✓
PU: Utilidad Percibida	Cuestionario TAM	La puntuación media de los ítems PU es $\geq 4/5$.	✓
ITU: Intención de Uso	Cuestionario TAM	La puntuación media de los ítems ITU es $\geq 4/5$.	✓

Tabla 4.4: Matriz de Trazabilidad de la Validación.

La validación integral de *PhenoScore* ha proporcionado una evaluación completa de la plataforma, extrayendo resultados mayoritariamente positivos que confirman el éxito del proyecto en sus objetivos fundamentales.

El análisis funcional y de usabilidad ha demostrado que la plataforma es robusta y cumple eficazmente su propósito principal, ya que los tres perfiles de usuario completaron con éxito el 100 % de las tareas guiadas. Este resultado es la evidencia de que *PhenoScore* logra reducir la barrera tecnológica, ofreciendo una interfaz intuitiva y accesible para un dominio que ha demostrado ser complejo.

La evaluación de los requisitos no funcionales ha verificado que la aplicación es segura y tiene un rendimiento óptimo en escenarios de uso individual, con tiempos de carga rápidos y con una base técnica sólida.

Finalmente, los resultados del Modelo de Aceptación Tecnológica (TAM) refuerzan las observaciones positivas, concluyendo que los usuarios consideran la plataforma fácil de usar, además de percibirla como una herramienta de gran utilidad, estando dispuestos a integrarla en su flujo de trabajo profesional.

En conclusión, la validación confirma que *PhenoScore* es una plataforma exitosa, una solución funcional, usable y bien valorada por los usuarios objetivo, con una línea clara de futuras mejoras en su escalabilidad.

CAPÍTULO 5

Conclusiones

Este capítulo final permite realizar una síntesis de los objetivos obtenidos, el logro de los objetivos, la reflexión sobre los desafíos durante el proceso y los esfuerzos de aprendizaje. El propósito es presentar resultados y reflexiones de manera clara y crítica para este trabajo final.

5.1 Respuesta a las Preguntas de Investigación

Este capítulo final presenta una síntesis del trabajo realizado, evaluando los hallazgos del proyecto a través de las preguntas de investigación planteadas en la metodología. El propósito es demostrar de manera clara y crítica cómo el proceso de diseño, desarrollo y validación ha dado respuesta a cada una de estas cuestiones.

5.1.1. Fase 1: Investigación del Problema

La primera fase del proyecto buscaba establecer las bases teóricas y contextuales. Las preguntas de investigación asociadas eran:

- *¿Cuál es la base de genética que sustentan los análisis PRS?*
- *¿Qué herramientas tecnológicas dan soporte a la ejecución de análisis PRS?*
- *¿Cuáles son las principales barreras y limitaciones que enfrentan los usuarios de estas herramientas tecnológicas?*

El Capítulo 2 ha respondido a estas preguntas de forma exhaustiva. Se ha determinado que los análisis PRS se fundamentan en los resultados de los estudios GWAS, que identifican variantes genéticas (SNPs) asociadas a enfermedades y cuantifican su efecto. El análisis del estado del arte reveló que, si bien existen herramientas potentes como *pgs_calc* o *GenoPred*, su principal barrera es la alta complejidad técnica, su dependencia de la línea de comandos (CLI) y la fragmentación del flujo de trabajo. Esta investigación confirmó la necesidad de una solución integrada y usable para usuarios no expertos en bioinformática.

5.1.2. Fase 2: Diseño de la Solución

La segunda fase se centró en definir la solución técnica. Las preguntas de investigación eran:

- *¿Qué requisitos debe cumplir una plataforma web para superar las barreras identificadas y facilitar el análisis PRS?*
- *¿Cuál es la arquitectura tecnológica más adecuada para esta plataforma?*
- *¿Cómo optimizar la carga, procesamiento y análisis de modelos PRS?*

El Capítulo 3 da respuesta a estas cuestiones a través del diseño de *PhenoScore*. Se definieron requisitos funcionales centrados en la usabilidad y la abstracción de la complejidad (ej. HU05: Subida de archivos mediante una interfaz gráfica). Se diseñó una arquitectura por capas (Presentación, Lógica de Negocio, Datos) para garantizar la mantenibilidad y escalabilidad. Para optimizar la carga de los análisis, se propuso una arquitectura asíncrona basada en una cola de trabajos, que permite ejecutar procesos computacionalmente intensivos en segundo plano sin bloquear la interfaz de usuario, respondiendo así al reto del rendimiento.

5.1.3. Fase 3: Validación de la Solución

La fase final buscaba evaluar la idoneidad y el éxito de la plataforma implementada. Las preguntas de investigación eran:

- *¿En qué medida la plataforma desarrollada cumple con los requisitos establecidos para superar las barreras identificadas?*
- *¿Cuál es el nivel de usabilidad y utilidad percibida por los usuarios finales al interactuar con la plataforma?*

El Capítulo 4 responde a estas preguntas a través de un meticuloso proceso de validación. Los resultados demuestran que *PhenoScore* cumple plenamente con los requisitos funcionales, ya que el 100 % de los usuarios completó con éxito todas las tareas clave. El nivel de usabilidad y utilidad percibida, medido tanto cualitativamente (tiempos de tarea) como cuantitativamente (Modelo de Aceptación Tecnológica), fue muy alto. Es especialmente relevante que el perfil de usuario no técnico pudiera utilizar la herramienta de forma autónoma, lo que confirma que la plataforma supera con éxito la barrera de accesibilidad que motivó el proyecto. La alta puntuación en las escalas de Facilidad de Uso y Utilidad Percibida del TAM refuerza esta conclusión.

5.2 Desafíos y soluciones del desarrollo

Durante la fase de implementación, surgieron varios desafíos técnicos, algunos de los cuales tuvieron que ser superados con un análisis y toma de decisiones sustanciales para garantizar el éxito del proyecto.

- **Problema de Entorno y Compatibilidad:** El obstáculo más significativo fue la inestabilidad encontrada al intentar ejecutar el *pipeline* de *GenoPred* en el Subsistema de *Windows* para *Linux* (WSL). Se lanzó una actualización de la herramienta externa introdujo cambios que rompían la compatibilidad y causaban errores fatales.
Solución: La arquitectura de despliegue completa se migró a una máquina virtual ejecutada en *Google Cloud Platform* con un sistema operativo *Linux* nativo. Fue un gran esfuerzo de reconfiguración, pero se logró un entorno de ejecución sólido, predecible y escalable, que es fundamental para una aplicación de estas características.

- **Dependencia de Software Externo:** Tras la migración, los análisis seguían fallando debido a un *bug* introducido en la nueva versión de *GenoPred*. La depuración de este problema fue compleja, ya que la causa se encontraba en código de terceros.
Solución: Resolver este problema fue posible después de una investigación proactiva que incluyó el análisis de *logs*, la revisión de la documentación y, finalmente, el contacto directo con el desarrollador principal de la herramienta. Su colaboración fue clave para identificar y corregir el *bug*, lo que permitió desbloquear el desarrollo. En general, fue un buen ejemplo de comunicación en proyectos que dependen en gran medida del software de código abierto.
- **Limitación de la Concurrencia:** Durante la validación, se identificó que la herramienta subyacente, *Snakemake*, no permite ejecuciones concurrentes en el mismo directorio de proyecto [?].
Solución (Conceptual): Aunque no se implementó por estar fuera del alcance del TFG, se identificó la solución técnica necesaria: aislar la ejecución de cada trabajo, por ejemplo, creando una copia dinámica actualizada del proyecto para cada análisis. Este análisis demuestra una comprensión del problema y establece una línea clara para trabajo futuro.

En cuanto a los errores cometidos, una posible mejora habría sido realizar un análisis técnico más detallado de las dependencias externas (*GenoPred/Snakemake*) en una fase más temprana del diseño. Una prueba de concepto inicial sobre la concurrencia probablemente haber revelado la limitación de *Snakemake* antes, lo que habría permitido preparar una solución a tiempo.

5.3 Aprendizaje

La realización de este Trabajo de Fin de Grado ha aportado una valiosa oportunidad de aprendizaje, tanto desde el punto de vista técnico como personal.

En términos profesionales, el proyecto brinda la oportunidad de aplicar y profundizar en una amplia variedad de competencias relacionadas con la ingeniería del software. Uno de los aspectos más destacados ha sido el diseño de una arquitectura de software compleja y moderna, que implica la implementación de un sistema asíncrono basado en colas de mensajes, un desafío técnico mucho mayor en comparación con una aplicación web común. La necesidad de integrar y depurar herramientas bioinformáticas existentes ha proporcionado una valiosa experiencia en la adopción e identificación de dependencias y resolución de problemas en un entorno real.

Como conclusión personal, creo que el aprendizaje más significativo ha sido la gestión de la incertidumbre y la resiliencia. Lidar con dificultades paralizantes de las que yo no era responsable directa, como el *bug* en *GenoPred*, me ha enseñado la importancia de la paciencia, la investigación minuciosa y la comunicación persistente para resolver los problemas. Este proyecto me ha dado la capacidad para tomar decisiones técnicas y gestionar un proyecto completo de software.

5.4 Trabajo a futuro

El estado actual de *PhenoScore* representa una base sólida y funcional, pero también un punto de partida con un gran espacio para el desarrollo. Las siguientes líneas de trabajo futuro podrían llevar a la evolución de la plataforma a una solución más sólida y completa y estar lista para un uso a mayor escala.

Despliegue y pruebas de rendimiento en un entorno de producción

El siguiente paso más importante consiste en el despliegue de la aplicación en un entorno de producción accesible públicamente. Esto permite realizar pruebas reales de rendimiento de la aplicación y, a su vez, terminar con la validación de todos los requisitos no funcionales. Una vez desplegada, se podría ejecutar la auditoría de *Google Lighthouse* para obtener métricas en aspectos como rendimiento, accesibilidad, buenas prácticas, o también pruebas de carga para evaluar cómo responde el sistema ante la concurrencia de varios usuarios.

Implementación de una solución para la ejecución concurrente

Durante la validación se ha constatado que la limitación principal de la parte de análisis es no poder ejecutarlos de forma concurrente debido a las limitaciones de *Snakemake*. La siguiente fase de trabajo tendría que centrarse en solucionar este problema. La forma más viable sería el aislamiento completo del entorno de ejecución de cada trabajo. Así, al comenzar un nuevo análisis, el *worker* no sólo generaría los ficheros de configuración, sino que haría también una copia propia y aislada del *pipeline* de *GenoPred* en un directorio temporal. De esta manera, la cola de trabajos podría tener en paralelo muchos análisis en paralelo y se podría mejorar mucho la velocidad y la escalabilidad del sistema.

Gestión centralizada de pacientes y poblaciones de referencia

Actualmente, la plataforma requiere que el usuario suba los datos genómicos del paciente cada vez que se crea un nuevo análisis. Para mejorar la eficiencia y la usabilidad, una línea de trabajo futuro clave es la creación de un módulo de gestión de pacientes. Esto permitiría a los usuarios almacenar de forma segura y centralizada la información de sus cohortes de pacientes, pudiendo seleccionar un paciente existente al crear un nuevo análisis en lugar de subir sus datos repetidamente.

Del mismo modo, se podría desarrollar una funcionalidad para que usuarios o administradores suban y gestionen poblaciones de referencia personalizadas. De momento no se ha podido desarrollar esta tarea por no disponer del suficiente recurso computacional para almacenar y procesar estos enormes conjuntos de datos. Habilitar esta opción permitiría a los investigadores usar sus propias cohortes de referencia y ser más versátiles y precisas sus ejecuciones de los análisis.

Creación de documentación y tutoriales para usuarios

Para facilitar la adopción de la plataforma por parte de nuevos usuarios, es fundamental desarrollar una sección de documentación y tutoriales interactivos. Esta sección podría incluir:

- Un manual de usuario que explique detalladamente cada funcionalidad de la plataforma.
- Videotutoriales que guíen a los usuarios a través del proceso completo de creación y ejecución de un análisis.
- Una sección de Preguntas Frecuentes (FAQ) para resolver las dudas más comunes.

Una buena documentación es necesaria para asegurar que los usuarios, especialmente los no técnicos, puedan sacar el máximo provecho de la herramienta de forma autónoma.

Pruebas de Concurrencia y Escalabilidad a Gran Escala

Uno de los mayores potenciales de mejora para *PhenoScore* es la implementación de la ejecución concurrente de análisis. Durante el desarrollo, se identificó que la herramienta externa *Snakemake* presenta una limitación que impide ejecuciones paralelas en el mismo directorio.

La siguiente fase de trabajo debería centrarse en implementar la solución de aislamiento de directorios ya diseñada conceptualmente. Al hacer que el *worker* cree una copia temporal y aislada del *pipeline* para cada trabajo, la cola podría procesar múltiples análisis en paralelo. Esta mejora transformaría el rendimiento *throughput* del sistema, siendo un paso indispensable para su escalabilidad y su uso en entornos con múltiples usuarios.

5.5 Relación del trabajo con los estudios cursados

El desarrollo del presente TFG ha supuesto la materialización práctica de la transversalidad de las competencias adquiridas a lo largo del Grado en Ingeniería Informática, al no tratarse de un desarrollo aislado, sino de un sistema caracterizado por la multidisciplinariedad.

A continuación, se detalla la relación del trabajo con las principales áreas de conocimiento del grado.

Ingeniería del Software, Análisis de Requisitos y Validación En general, el proyecto se ha llevado a cabo siguiendo un proceso de ingeniería del software con un alto nivel de rigor a la hora de aplicar los fundamentos aprendidos en la especialización directamente al conjunto de fases, actividades y tareas realizadas. El uso de la metodología de la Ciencia del Diseño ha permitido una aplicación sistemática de las fases del ciclo de vida del software mencionadas anteriormente a lo largo de la etapa, que abarcan desde su concepción hasta la de los procesos que las finalizan, logrando una integración de las competencias de asignaturas como:

- La asignatura de Ingeniería del Software ha proporcionado la base metodológica para estructurar todo el proyecto. Conceptos como el ciclo de vida, la gestión de la configuración y la importancia de seguir un proceso definido han sido la estructura del trabajo, asegurando que cada fase (análisis, diseño, implementación, validación) se abordara de forma ordenada y coherente.
- A partir de esta base, se han aplicado las técnicas de la asignatura Análisis y Especificación de Requisitos. Lo cual, se ha materializado en la obtención de las necesidades del usuario mediante historias de usuario y su posterior traducción a requisitos funcionales y no funcionales detallados. Asimismo, se ha realizado el modelado conceptual del sistema utilizando un conjunto completo de diagramas UML (Clases, Secuencia, Componentes, Despliegue), una competencia central en esta materia que ha sido clave para definir la estructura y el comportamiento de la solución antes de su implementación.
- Finalmente, la asignatura Análisis, Validación y Depuración de Software ha guiado la última fase del proyecto. El capítulo de Validación se basa directamente en los principios para asegurar la calidad del producto. Se ha planificado y ejecutado un proceso de validación que incluye la verificación de la funcionalidad mediante un guion de tareas guiadas, la medición de requisitos no funcionales de rendimiento

y la evaluación de la aceptación del usuario con el modelo TAM. Este enfoque demuestra la capacidad para diseñar y ejecutar un plan de pruebas que asegure que el software no solo funciona, sino que también cumple con los estándares de calidad definidos.

Diseño de Software. Uno de los pilares del TFG ha sido el diseño de una arquitectura de software moderna y compleja, un desafío central en la Ingeniería del Software. Se han aplicado directamente conceptos de la asignatura Diseño de Software, como la arquitectura por capas (Presentación, Lógica de Negocio y Persistencia) para garantizar la separación de responsabilidades y la mantenibilidad. Más allá de esta estructura, se han implementado varios patrones de diseño y buenas prácticas para resolver problemas específico, como:

- El patrón de cola de trabajos, para gestionar la ejecución de los análisis PRS: que son tareas de larga duración, se ha implementado este patrón utilizando *BullMQ* y *Redis*. Esto desacopla la petición del usuario de la ejecución real, evitando el bloqueo de la interfaz y mejorando la escalabilidad y la resiliencia del sistema.
- Principio de Responsabilidad Única: Este principio de diseño *SOLID* [?] se ha aplicado al separar la lógica del *AutoWorker* (cuya única responsabilidad es gestionar la cola) de la del *AnalysisProcessor* (cuya única responsabilidad es ejecutar un análisis). Esta separación ha hecho el código mucho más limpio, mantenible y fácil de depurar.

Bases de Datos y Persistencia de la Información. Por otra parte, el proyecto ha requerido que se apliquen de manera directa y avanzada los conceptos de persistencia de datos, cuya formación básica ha sido la impartida en la asignatura de Bases de Datos y sistemas de información. En particular, la asignatura presentada anteriormente ha asentado las bases del conocimiento acerca del modelo relacional y de SQL, y ha permitido abordar con éxito el diseño e implementación de *PhenoScore*.

- Diseño Relacional: El esquema de datos relacional, complejo, normalizado y optimizado para la plataforma ha sido diseñado e implementado en un servidor *MariaDB*. Este proceso de modelado ha evolucionado desde un esquema de clases conceptual a un diagrama de entidad-relación físico y es una implementación directa de las habilidades de diseño de bases de datos.
- Gestión y Consulta: Para la interacción con la base de datos, se ha utilizado un *ORM* moderno como *Prisma*. Aunque esta herramienta abstrae la escritura de SQL directo, la capacidad para construir consultas seguras y eficientes, así como para gestionar las migraciones del esquema, se fundamenta en la comprensión del modelo relacional y de cómo funcionan las operaciones subyacentes.
- Bases de Datos *NoSQL*: Adicionalmente, se ha utilizado una base de datos en memoria clave-valor como *Redis* para una tarea de alta velocidad, en este caso, la gestión de la cola de trabajos. De esta manera uno puede ver cómo abordar más allá del modelo relacional y elegir la tecnología de persistencia más adecuada para cada requisito específico del sistema, un concepto avanzado en la gestión de la información.

Sistemas Operativos y Concurrency. La funcionalidad central del proyecto, la ejecución asíncrona de análisis, es una aplicación directa de los conceptos fundamentales estudiados en asignaturas como Sistemas Operativos y Concurrency. El conocimiento teórico adquirido con respecto de la gestión de procesos, su ciclo de vida y la comunicación entre los mismos, ha sido indispensable para poder diseñar la arquitectura del *worker*.

Gracias a esta base conceptual, fue posible implementar la ejecución de *GenoPred* como un proceso externo utilizando el módulo *child_process* de *Node.js*, comprendiendo los mecanismos subyacentes de creación y monitorización de procesos que el sistema operativo gestiona.

En conclusión, este TFG es un reflejo de la formación recibida a lo largo del grado en un proyecto integrador en el que se han puesto en práctica las competencias adquiridas en las principales áreas de la informática de manera integrada para desarrollar una solución *software* de un cierto nivel de complejidad.

Bibliografía

- [1] Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, et al. Initial sequencing and analysis of the human genome. *Nature*. 2001 Feb 15;409(6822):860-921.
- [2] Robinson TR, Spock L. *Genetics For Dummies*. 3rd ed. Hoboken: Wiley; 2020.
- [3] Visscher PM, Yengo L, Cox NJ, Wray NR. Discovery and implications of polygenicity of common diseases. *Science*. 2021 Sep 24;373(6562):1468-73. doi: 10.1126/science.abi8206.
- [4] Ye Y, Chen X, Han J, Jiang W, Natarajan P, Zhao H. Interactions between enhanced polygenic risk scores and lifestyle for cardiovascular disease, diabetes, and lipid levels. *Circ Genom Precis Med*. 2021 Feb;14(1):e003128. doi: 10.1161/CIRC-GEN.120.003128.
- [5] Lewis CM, Vassos E. Polygenic risk scores: from research tools to clinical instruments. *Genome Med*. 2020 May 29;12(1):44. doi: 10.1186/s13073-020-00742-5.
- [6] Wieringa RJ. *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer; 2014.
- [7] Babb de Villiers C, Kroese M, Moorthie S. Understanding polygenic models, their development and the potential application of polygenic scores in healthcare. *J Med Genet*. 2020 Nov;57(11):725-32. doi: 10.1136/jmedgenet-2019-106763.
- [8] Uffelmann E, Huang QQ, Munung NS, de Vries J, Okada Y, Martin AR, et al. Genome-wide association studies. *Nat Rev Methods Primers*. 2021;1(59). doi: 10.1038/s43586-021-00056-9.
- [9] Choi SW, Mak TSH, O'Reilly PF. Tutorial: a guide to performing polygenic risk score analyses. *Nat Protoc*. 2020 Sep;15(9):2759-72. doi: 10.1038/s41596-020-0353-1.
- [10] Lambert SA, Abraham G, Inouye M. Towards clinical utility of polygenic risk scores. *Nat Rev Genet*. 2019 Dec;20(12):715-26. doi: 10.1038/s41576-019-0153-9.
- [11] Collister JA, Liu X, Clifton L. Calculating polygenic risk scores (PRS) in UK Biobank: a practical guide for epidemiologists. *Front Genet*. 2022;13:818574. doi: 10.3389/fgene.2022.818574.
- [12] Maamari DJ, Abou-Karam R, Fahed AC. Polygenic risk scores in human disease. *Clin Chem*. 2025 Jan;71(1):69-76. doi: 10.1093/clinchem/hvae190.
- [13] Corpas M, Megy K, Metastasio A, Lehmann E. Implementation of individualised polygenic risk score analysis: a test case of a family of four. *BMC Med Genomics*. 2022 Oct 10;15(1):207. doi: 10.1186/s12920-022-01331-8.

- [14] Knowles JW, Ashley EA. Cardiovascular disease: the rise of the genetic risk score. *Genome Med.* 2018;10(1):1002546. doi: 10.1371/journal.pmed.1002546.
- [15] Terkelsen T, Gerds TA, Nielsen JB, Nordestgaard BG, Bundgaard H, Køber L, et al. The clinical use of polygenic risk scores. *Ugeskr Laeger.* 2023 Sep;185(39):V04230258.
- [16] Torkamani A, Wineinger NE, Topol EJ. The personal and clinical utility of polygenic risk scores. *Nat Rev Genet.* 2018 Sep;19(9):581-90. doi: 10.1038/s41576-018-0018-x.
- [17] Polygenic Risk Score Task Force of the International Common Disease Alliance. Responsible use of polygenic risk scores in the clinic: potential benefits, risks and gaps. *Nat Med.* 2021 Nov;27(11):1876-84. doi: 10.1038/s41591-021-01549-6.
- [18] Wray NR, Lin T, Austin J, Ripke S, Kendler KS, Lichtenstein P, et al. From basic science to clinical application of polygenic risk scores: a primer. *JAMA Psychiatry.* 2021 Jan 1;78(1):101-9. doi: 10.1001/jamapsychiatry.2020.3049.
- [19] Lambert SA, Wingfield BD, Chiam J, Tcheandjieu C, Ghasemzadeh N, Inouye M, et al. Enhancing the Polygenic Score Catalog with tools for score calculation and ancestry normalization. *Nat Genet.* 2024. doi: 10.1038/s41588-024-01937-x.
- [20] Pain O, Al-Chalabi A, Lewis CM. The GenoPred pipeline: a comprehensive and scalable pipeline for polygenic scoring. *Bioinformatics.* 2024 Oct;40(10):btac551. doi: 10.1093/bioinformatics/btac551.
- [21] Forer L. pgs-calc: a simple command-line tool for calculating Polygenic Scores [Internet]. GitHub; 2023. [cited 2025 Aug 25]. Available from: <https://github.com/lukfor/pgs-calc>.
- [22] Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley Professional; 2002.
- [23] Davis FD. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.* 1989 Sep;13(3):319-40. doi: 10.2307/249008.
- [24] Snakemake Development Team. Snakemake Documentation - On-site cluster execution [Internet]. [cited 2025 Aug 25]. Available from: <https://snakemake.readthedocs.io/en/stable>.
- [25] Martin RC. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall; 2008.

APÉNDICE A

Anexo: Objetivos de Desarrollo
Sostenible



ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

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

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

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Este proyecto de fin de grado se centra en el diseño y desarrollo de la plataforma PhenoScore, que está diseñada para analizar puntuaciones de riesgo poligénico. Aunque es principalmente un trabajo técnico, tiene una conexión importante con varios de los Objetivos de Desarrollo Sostenible. El proyecto tiene un alcance que supera con creces la simple implementación de un software, ya que sirve como herramienta con capacidad de influir positivamente en la sociedad, mostrando un especial compromiso con dos objetivos muy importantes: el de Salud y Bienestar, y el de Industria, Innovación e Infraestructura.

Relación Principal y de Alto Impacto: ODS 3 - Salud y Bienestar

La conexión más fuerte del proyecto es claramente con el ODS 3, cuyo objetivo es ofrecer una vida sana y promover el bienestar para cada persona, sin importar la edad. PhenoScore constituye una herramienta diseñada para impulsar un cambio significativo en la medicina. Este cambio implica pasar de un enfoque que se centra en tratar enfermedades una vez que ya aparecen, a otro enfoque que busca anticiparse a ellas. Este proyecto busca aportar su contribución de varias maneras, enriqueciendo así el logro del objetivo global del ODS 3.

1. **Fomento de la Medicina Preventiva y Personalizada (Meta 3.4):** Las enfermedades complejas y crónicas, como cardiopatías, diabetes y algunos tipos de cáncer, son una gran carga para los sistemas de salud en todo el mundo. PhenoScore ayuda a calcular los PRS, que permiten identificar permite identificar a individuos con una alta predisposición genética a padecer estas enfermedades mucho antes de que puedan aparecer los primeros síntomas. Esta estratificación del riesgo es crucial para establecer estrategias efectivas y específicas de prevención: un médico podría recomendar a un paciente con alto riesgo genético que modifique su estilo de vida, pruebas de detección más tempranas o un seguimiento más frecuente, reduciendo de esta manera la probabilidad de que la enfermedad se desarrolle.
2. **Aceleración de la Investigación Médica (Meta 3.b):** El progreso en la comprensión y tratamiento de las enfermedades complejas, por lo tanto, depende de la habilidad de los investigadores para analizar grandes cantidades de datos genómicos. Sin embargo, la contribución de PhenoScore radica, justamente, en democratizar el acceso a estas técnicas. Al eliminar la barrera de la línea de comandos y la complejidad técnica, la plataforma permite que un espectro más amplio de profesionales (biólogos, médicos, genetistas) puedan realizar análisis PRS de manera autónoma. Por lo tanto, la plataforma permitirá acelerar la investigación, facilitando el descubrimiento de nuevas asociaciones genéticas y contribuyendo al desarrollo de nuevos enfoques terapéuticos.



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



3. **Sostenibilidad de los Sistemas de Salud:** Un mayor enfoque en la prevención no solo salva vidas y mejora el bienestar, sino que también es más sostenible económicamente. Al ayudar a prevenir enfermedades crónicas, que son las que consumen la mayor parte de los recursos sanitarios, las herramientas como PhenoScore tienen el potencial a largo plazo para reducir la presión económica y de recursos sobre los sistemas de salud públicos y privados.

Relación de Impacto Medio: ODS 9 - Industria, Innovación e Infraestructura

Además, el proyecto también se alinea con el ODS 9, que pretende "construir infraestructuras resilientes, promover la industrialización inclusiva y sostenible y fomentar la innovación". PhenoScore es, en sí mismo, un producto de la innovación que contribuye a fortalecer la infraestructura científica y tecnológica.

1. **Fomento de la Innovación Tecnológica:** El desarrollo de la plataforma es un claro ejemplo de innovación que aplica las ciencias de la computación para resolver un problema complejo en el sector de la salud digital y la bioinformática. El diseño de una arquitectura asíncrona, la integración de herramientas bioinformáticas externas y el enfoque en la experiencia de usuario son ejemplos de innovación de proceso y de producto que demuestran cómo la ingeniería del software puede aportar un valor incalculable a otras disciplinas científicas.
2. **Creación de Infraestructura Científica Digital:** PhenoScore no se trata simplemente de una aplicación, sino una pieza de infraestructura digital. Potencia la capacidad de investigación de hospitales, universidades y otros centros de investigación, especialmente aquellos que no cuentan con los recursos para contratar a bioinformáticos expertos. Al proponer una herramienta accesible, se consigue reforzar la infraestructura científica global, creando las condiciones para que grupos de investigación cada vez más diversos puedan formar parte del avance en el desarrollo de la medicina genómica.

Otras Relaciones de Menor Impacto

Adicionalmente, el TFG establece vínculos de menor grado, pero aún así relevantes, con otros ODS:

- **ODS 4 (Educación de Calidad):** La plataforma puede servir como una valiosa herramienta pedagógica. Estudiantes de grados de biomedicina, genética o bioinformática podrían utilizar PhenoScore para comenzar a comprender los fundamentos del análisis PRS de manera interactiva y práctica, sin la barrera inicial de la programación y la línea de comandos.
- **ODS 8 (Trabajo Decente y Crecimiento Económico):** El proyecto se enmarca en el sector de la salud digital y la biotecnología, ambas son áreas de alto valor añadido que generan empleo cualificado. Por lo tanto, la creación de productos para este campo, si bien indirecta, se traduce en un crecimiento económico basado en el conocimiento.



Escola Tècnica
Superior d'Enginyeria
Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n, 46022, València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es





- **ODS 10 (Reducción de las Desigualdades):** La tecnología médica avanzada a menudo está concentrada en centros de investigación de élite. PhenoScore, como plataforma web, puede reducir las desigualdades en el acceso a la herramienta de análisis genómico en cuestión, lo que permite a los grupos de investigación con menos recursos realizar estudios que, de otro modo, serían extremadamente complicados.
- **ODS 17 (Alianzas para Lograr los Objetivos):** El desarrollo del presente TFG es en sí un ejemplo de este objetivo, pues ha necesitado la unión de diferentes perfiles, como la ingeniería de software, la bioinformática o la investigación biomédica.

Además, ha sido diseñado para integrarse con un sistema de priorización de modelos complementarios desarrollado por otro equipo. La interacción, ya que PhenoScore consume los resultados generados por ese sistema, ilustra que estos avances significativos requieren de una alta cooperación entre proyectos.

En conclusión, aunque PhenoScore es un proyecto de ingeniería de software, su propósito y su impacto potencial se extienden más allá de lo puramente técnico, contribuyendo de manera significativa a la consecución de un futuro más saludable, innovador y equitativo, en línea con los ODS.