Facultad de Ingeniería
Escuela de Ingeniería Informática

DESARROLLO DE UN SISTEMA DE PRIORIZACIÓN DE MODELOS DE PUNTUACIÓN DE RIESGO POLIGÉNICO

Por

Valentina Gallardo Reinoso

Trabajo realizado para optar al Título de
INGENIERA CIVIL INFORMÁTICA
Prof. Guía: René Noël López
Prof. Co-Referente: Diana Martínez-Minguet
Agosto 2025

Resumen

Este trabajo aborda el desarrollo de un sistema para la extracción, priorización y futura aplicación de modelos de puntuación de riesgo poligénico. Estas puntuaciones constituyen una herramienta clave en la medicina personalizada, ya que permiten estimar la predisposición genética de un individuo al desarrollo de enfermedades complejas. Sin embargo, la ausencia de estandarización y la alta variabilidad entre los modelos disponibles dificulta su comparación, selección y uso clínico efectivo.

El principal problema identificado es la necesidad de contar con una herramienta que permita mejorar la selección y priorización de modelos PRS, considerando criterios objetivos y clínicamente relevantes. En respuesta a esta necesidad, se desarrollo un sistema compuesto por 2 módulos: un módulo de extracción automatizada desde repositorios especializados, un módulo selección y priorización basado en criterios como la ancestría, el poder discriminativo y la asociación de riesgo. Entre los resultados más relevantes se encuentra el desarrollo de un sistema funcional capaz de sistematizar el proceso de priorización, facilitando la toma de decisiones clínicas y reduciendo la carga técnica asociada a la evaluación manual de modelos. Esto se traduce en una mejora en la precisión, eficiencia y trazabilidad de los análisis genómicos.

El sistema desarrollado representa un avance tangible para la bioinformática y la medicina personalizada, ya que pone a disposición una herramienta flexible y escalable que puede adaptarse a diferentes entornos clínicos y de investigación.

Agradecimientos

Quiero expresar mi más profundo agradecimiento a mi **madre**, mi pilar fundamental y la mejor persona que Dios pudo darme. Gracias por nunca dejarme sola, por cada reto que me diste sabiendo que era para mi bien. Todo lo que he logrado en este camino es por ti y para ti, porque sin tu amor, sacrificio y ejemplo no estaría aquí hoy.

A mi **hermano**, el mejor compañero de vida que me pudo tocar. No sé qué haría sin ti. Gracias por estar siempre presente, por tus retos, consejos y por tu apoyo incondicional. Cada paso que damos lo damos juntos, y eso es algo que valoro profundamente porque esto también es por y para ti.

A mi familia en general, por ser mi red de apoyo incondicional. Gracias por sus consejos, su paciencia y sus palabras de aliento en los momentos más difíciles. Su respaldo constante me dio la fortaleza para superar cada desafío y la motivación para seguir adelante en este camino.

A mis amigos **Vayhron y Aaron**, compañeros inseparables desde nuestro regreso a la universidad después de la pandemia. Siempre buscamos la manera de permanecer unidos, incluso cuando nos decían que "se hacen de cuatro o en pareja". Fuimos un gran equipo, el gran "Equipo 7". Gracias infinitas por su apoyo, por estar presentes en las malas notas y en las buenas, en los trabajos más difíciles y por ayudarnos mutuamente en todo momento. Por las incontables risas que nunca faltaron en clases que nos hicieron más llevadero el camino.

A mi cuñada **Fernanda**, que junto con Emilia y Felipe Alonso llegaron a mi vida en un momento muy complicado, y desde entonces se han convertido en lo mejor que me pudo pasar. Gracias por estar siempre ahí, sin importar las circunstancias, y por cada palabra y gesto que me ayudaron a seguir adelante. Siempre tendrán un lugar especial en mi corazón.

A mi profesor guía **René** y a mi co-referente **Diana**, por confiarme este proyecto, por su paciencia y por darme la oportunidad de adentrarme en un mundo nuevo que al principio no entendía. Hoy puedo decir que este desafío me enseñó mucho y que, a pesar de las dificultades, logramos sacarlo adelante.

Finalmente, a todas las personas que llegaron al principio o al final de este camino, gracias por brindarme su apoyo y por darme alguna palabra de aliento durante este trayecto. Cada gesto, consejo o ayuda, por pequeño que pareciera, ha sido fundamental para alcanzar esta meta.

Índice general

Resumen III

Agradecimientos IV

1. Introducción 1

2. Marco Conceptual y Estado del Arte 3
- 2.1. Marco conceptual . . . 3
  - 2.1.1. PGS catalog . . . 5
  - 2.1.2. Bibliografía: guías y recomendaciones . . . 5
  - 2.1.3. Genopred . . . 6
- 2.2. Estado del arte . . . 6

3. Definición del Problema y Análisis de Requerimientos 11
- 3.1. Formulación del Problema . . . 11
- 3.2. Solución Propuesta . . . 12
  - 3.2.1. Importancia del trabajo . . . 13
- 3.3. Objetivos . . . 14
  - 3.3.1. Objetivo General . . . 14
  - 3.3.2. Objetivos Específicos . . . 14
- 3.4. Metodología . . . 14
- 3.5. Especificación de Requerimientos . . . 15
  - 3.5.1. Situación Actual (AS IS) . . . 16
  - 3.5.2. Situación Propuesta (TO BE) . . . 17
  - 3.5.3. Requerimientos funcionales . . . 18
  - 3.5.4. Requerimientos no funcionales . . . 19
- 3.6. Funcionalidades del sistema . . . 21
  - 3.6.1. Diagramas de casos de uso . . . 21
  - 3.6.2. Casos de Uso . . . 22
  - 3.6.3. Diagrama de Secuencia . . . 24
  - 3.6.4. Modelo conceptual . . . 25

4. Diseño **28**
4.1. Diseño Arquitectónico . . . 28
4.1.1. Tecnologías utilizadas . . . 28
4.1.2. Flujo de datos / Vista de alto nivel . . . 29
4.2. Diseño Lógico . . . 30
4.2.1. Diagrama de despliegue . . . 30
4.2.2. Diagrama de componentes . . . 32
4.2.3. Diagrama de paquetes . . . 33
4.2.4. Diagrama de clases . . . 36
4.3. Diseño de datos . . . 38
4.3.1. Modelo relacional . . . 38
4.3.2. Diccionario de Datos . . . 40
4.4. Diseño de Interfaz . . . 41
4.4.1. Arquitectura de la Información . . . 41
4.4.2. Prototipos de Interfaces Gráficas . . . 42

5. Implementación **45**
5.1. Software utilizado . . . 45
5.2. Hardware utilizado . . . 47
5.3. Lenguajes de programación . . . 48
5.4. Estrategia de implementación . . . 49
5.5. Interfaces de Usuario . . . 51

6. Pruebas **54**
6.1. Contexto de pruebas . . . 54
6.1.1. Características . . . 54
6.1.2. Estructura del ETL . . . 55
6.2. Pruebas Unitarias . . . 57
6.3. Pruebas de Integración . . . 57
6.3.1. Pruebas de Integración entre Base de Datos y Backend . . . 57
6.3.2. Pruebas de Integración entre Backend y Frontend . . . 58
6.4. Pruebas de Sistema . . . 59
6.5. Pruebas con Usuarios . . . 60
6.6. Pruebas de Aceptación . . . 61

7. Implantación **64**
7.0.1. Preparación de Ambiente . . . 64
7.0.2. Manual de Usuario . . . 65
7.0.3. Documentación de desarrollo . . . 65

8. Conclusiones **78**

Bibliografía **80**

A. Anexo 1 **83**

B. Anexo 2 **97**
B.0.1. Pruebas Unitarias para la clase Trait 97
B.0.2. Pruebas Unitarias para la clase TraitCategory 98
B.0.3. Pruebas Unitarias para la clase Publications 99
B.0.4. Pruebas Unitarias para la clase PRSModel 99
B.0.5. Pruebas Unitarias para la clase BroadAncestryCategory 100
B.0.6. Pruebas Unitarias para la clase PopulationSample_Development . . 101
B.0.7. Pruebas unitarias para la clase ModelEvaluation 103
B.0.8. Pruebas Unitarias para la clase BroadAncestryInModel 104

C. Anexo 3 **106**
C.1. Introducción 106
C.2. Pantalla de Phenotype Selector 106
C.2.1. Selección de Ancestría 107
C.2.2. Triat Category 108
C.2.3. Visualización Gráfica de Categorías de Rasgo 109
C.2.4. Selección de Rasgos y Modelos 110
C.2.5. Resumen de Modelos Seleccionados 110
C.2.6. Acciones Finales: Ir a Priorizar o Cancelar 111
C.3. Módulo de Priorización de Modelos PRS 111
C.3.1. Selección de Ancestría de Priorización 112
C.3.2. Visualización del Ranking de Modelos 113
C.3.3. Selección y Ordenamiento Final 114
C.3.4. Manejo de Ausencia de Modelos para una Ancestría 115

Índice de figuras

2.1. Modelo conceptual de modelos PRS [1] 5

3.1. Diagrama de solución propuesta. El módulo de aplicación se encuentra fuera del alcance de este proyecto y será desarrollado por un tercero encargado del análisis genético clínico. 13
3.2. Diagrama metodología secuencial [2]. 15
3.3. Diagrama BPMN de la situación actual (AS IS). 17
3.4. Situación propuesta (TO BE). Se destacan las actividades automatizadas por el sistema desarrollado: obtención y priorización de modelos PRS. 18
3.5. Diagrama de casos de uso. 21
3.6. Diagrama de secuencia. 25
3.7. Modelo conceptual de modelos PRS [1] 27

4.1. Vista de alto nivel 30
4.2. Diagrama de despliegue 31
4.3. Diagrama de componentes del sistema PRS 33
4.4. Estructura de paquetes del sistema 34
4.5. Estructura de paquetes del sistema 36
4.6. Diagrama de Clases [1]. 38
4.7. Modelo relacional de la base de datos del sistema 40
4.8. Mockup de la pantalla `Phenotype Selector` 43
4.9. Mockup de la pantalla `Prioritization Module` 44

5.1. Vista de alto nivel de las tecnologías utilizadas en el sistema 47
5.2. Estrategia de implementación del sistema 51
5.3. Interfaz para selección de características fenotípicas. 52
5.4. Interfaz para la priorización de modelos PRS. 53

6.1. Estructura proceso ETL. 56

C.1. Pantalla Phenotype Selector 107
C.2. Selección de ancestría 108

C.3. Selección de ancestría . . . 108
C.4. Filtrado por categoría de rasgo . . . 109
C.5. Gráfico de distribución de Trait Categories con número de modelos . . . 109
C.6. Selección de rasgos con modelos asociados . . . 110
C.7. Resumen de traits seleccionados y cantidad total de modelos PRS a priorizar 110
C.8. Botones de acción: Cancelar o avanzar a priorización . . . 111
C.9. Botones de acción: Cancelar o avanzar a priorización . . . 112
C.10. Selección de ancestría para priorización . . . 113
C.11. Selección de ancestría para priorización . . . 113
C.12. Tabla de priorización de modelos . . . 114
C.13. Ordenamiento personalizado de modelos . . . 115
C.14. Ordenamiento personalizado de modelos . . . 115

Índice de tablas

2.1. Estado del arte: herramientas de análisis PGS. 9

3.1. Requerimientos funcionales del sistema. 19
3.2. Requerimientos No Funcionales del Sistema 20
3.3. Caso de Uso: Extraer datos de Modelos PRS 23
3.4. Caso de Uso: Priorizar Modelos PRS 24

6.1. Relación entre endpoints ETL, endpoints del catálogo PGS y clases del modelo conceptual 56
6.2. Pruebas de integración entre Backend y Base de Datos 57
6.3. Pruebas de integración entre Backend y Frontend 58
6.4. Pruebas de sistema 59
6.5. Pruebas de Usuario. 61
6.6. Casos de prueba de aceptación 62

A.1. Definición de la tabla PerformanceMetric 83
A.2. Definición de la tabla PerformanceMetricEvaluation 84
A.3. Definición de la tabla ModelEvaluation. 85
A.4. Definición de la tabla Publication. 85
A.5. Definición de la tabla EvaluationPopulationSamples. 87
A.6. Definición de la tabla BroadAncestryCategory. 87
A.7. Definición de la tabla BroadAncestryInRefPop. 88
A.8. Definición de la tabla DevelopmentPopulationSample. 89
A.9. Definición de la tabla ReferencePopulation. 90
A.10. Definición de la tabla DevelopmentPopulationSample. 91
A.11. Definición de la tabla BroadAncestryModel. 92
A.12. Definición de la tabla PRSModel. 92
A.13. Definición de la tabla PrioritizedModel. 93
A.14. Definición de la tabla ScoringFile. 93
A.15. Definición de la tabla PRSModelToTrait. 94
A.16. Definición de la tabla Trait. 95
A.17. Definición de la tabla TraitToTraitCategory. 95

A.18. Definición de la tabla TraitCategory. 96

B.1. Pruebas Unitarias clase *Trait*. 97
B.2. Pruebas Unitarias clase *TraitCategory*. 98
B.3. Pruebas Unitarias clase *Publication*. 99
B.4. Pruebas Unitarias *PRSModel*. 100
B.5. Pruebas Unitarias clase *BroadAncestryCategory*. 101
B.6. Pruebas Unitarias de la clase *PopulationSample_Development*. 102
B.7. Pruebas Unitarias de la clase *ModelEvaluation*. 104
B.8. Pruebas unitarias de la clase *BroadAncestryInModel*. 105

Capítulo 1

Introducción

En los últimos años, las puntuaciones de riesgo poligénico (PRS, por sus siglas en inglés: Polygenic Risk Scores) y las puntuaciones de riesgo genético (GRS, Genetic Risk Scores) [3] han ganado un papel destacado en la medicina personalizada. Estas herramientas permiten estimar la predisposición genética de un individuo a desarrollar enfermedades complejas, basándose en la agregación de variantes genéticas distribuidas en todo el genoma [4]. Esta capacidad predictiva resulta fundamental para apoyar la toma de decisiones clínicas y orientar tratamientos más precisos y personalizados.

Un modelo PRS se define como una combinación de variantes genéticas y sus respectivos pesos, calculados mediante algoritmos y análisis estadísticos aplicados a estudios genómicos poblacionales [5]. Su correcta aplicación permite identificar de forma más eficiente el riesgo genético de un paciente. Sin embargo, uno de los desafíos actuales es la correcta selección del modelo PRS más adecuado entre cientos de opciones disponibles en repositorios públicos. Una mala elección puede afectar negativamente la precisión del análisis y, en consecuencia, las decisiones clínicas que se deriven de él.

El proceso de selección se ve dificultado por dos grandes factores. En primer lugar, la falta de estandarización entre los modelos PRS y la heterogeneidad en su desarrollo dificulta su comparación directa [6]. En segundo lugar, existe una alta variabilidad en las características disponibles por cada modelo, lo que exige un análisis exhaustivo de metadatos para determinar cuál se ajusta mejor al perfil genético y ancestral del paciente. Aunque existen herramientas automatizadas para aplicar modelos PRS, aún persiste la necesidad de sistemas capaces de filtrar, priorizar y seleccionar modelos de forma efectiva y personalizada.

Para abordar esta problemática, este trabajo propone el desarrollo de un sistema compuesto por dos módulos principales: uno de extracción y otro de priorización de modelos PRS. El sistema aprovecha técnicas de ingeniería de software para automatizar la recopilación, transformación y almacenamiento de información proveniente de *PGS Catalog*, y ofrece una interfaz interactiva para que el usuario pueda seleccionar y priorizar modelos según características específicas como rasgo, ancestría, potencia discriminativa y año de

publicación.

Durante el desarrollo del proyecto se presentó una reestructuración tecnológica significativa: originalmente se contempló la extracción de datos mediante scripts en Python y pandas; sin embargo, se optó por implementar un proceso ETL directamente desde el backend en Node.js por su mayor eficiencia, robustez y escalabilidad en la manipulación de datos. Este cambio implicó acotar el alcance inicial, enfocándose exclusivamente en la preparación del sistema base para su posterior integración con plataformas de análisis genómico. La etapa de aplicación clínica de los modelos será responsabilidad de los investigadores del proyecto en la Universidad Politécnica de Valencia, quienes utilizarán este sistema como base para el análisis personalizado de riesgo genético en sus líneas de investigación.

El sistema resultante constituye una herramienta modular, escalable y reutilizable que facilita la gestión, selección y priorización de modelos PRS. Sus beneficios incluyen una mejor organización de la información genómica, mayor velocidad en la toma de decisiones clínicas fundamentadas y una contribución concreta al ecosistema de la medicina personalizada.

Este trabajo pretende aportar al área de la bioinformática mediante el desarrollo de una solución práctica y eficiente para el uso de modelos PRS. Sus principales contribuciones incluyen:

- La implementación de un sistema capaz de integrar, transformar y mostrar modelos PRS de forma clara y navegable.
- La priorización flexible de modelos basados en múltiples criterios relevantes.
- La preparación de una base técnica robusta para integrarse con futuras plataformas de análisis genómico automatizado.

El informe se estructura de la siguiente manera: el Capítulo 2 presenta el marco conceptual y el estado del arte; el Capítulo 3 describe el problema abordado, los objetivos y la solución propuesta; el Capítulo 4 detalla el diseño del sistema, incluyendo casos de uso, diagramas y modelo conceptual; el Capítulo 5 describe el desarrollo, la arquitectura tecnológica y las funcionalidades implementadas; el Capítulo 6 presenta las pruebas realizadas; el Capítulo 7 aborda el proceso de implantación del sistema, incluyendo consideraciones de integración y despliegue; y finalmente, el Capítulo 8 expone las conclusiones y proyecciones futuras del proyecto.

Capítulo 2

Marco Conceptual y Estado del Arte

En este capítulo, la Sección 2.1 presenta el marco conceptual de los modelos PRS (Polygenic Risk Scores), basado en el trabajo de Diana Martinez estudiante de doctorado en la Universidad Politécnica de Valencia, España. Dado que el modelo conceptual fue desarrollado en inglés, se ha mantenido en ese idioma para asegurar la consistencia con investigaciones previas.

La Sección 2.2 expone el estado del arte, revisando los avances más recientes en el uso de modelos PRS y las herramientas disponibles, alineando este trabajo con estudios actuales en la materia.

2.1. Marco conceptual

- **PRS Model**: Representa un modelo de puntuación de riesgo poligénico. Está identificado por un nombre, un identificador en PGS catalog (`PGSC_id`) y otro en la publicación científica donde se describe (`pub_id`). Un modelo PRS define el uso previsto y el resultado esperado en relación con un fenotipo determinado.

- **Phenotype**: Representa el rasgo observable o medible de una enfermedad. Se identifica mediante un nombre y un `EFO_id`, que corresponde al identificador de la Ontología Factorial Experimental (EFO). Un **PRS Model** puede estar vinculado a uno o varios fenotipos.

- **Scoring Set**: Este conjunto incluye las variantes genéticas necesarias para calcular el PRS a través del número de variantes (`n_snp`) y el genoma de referencia (`assembly`). Está asociado con el **PRS Model** y es esencial para realizar un análisis PRS.

- **Patient**: Es el individuo al que se le aplica el **PRS Model** para obtener una medida de predisposición genética (`PRS`). Se identifica mediante un `patient_id` y su

ascendencia, la cual influye en la interpretación de los resultados.

- **PRS Analysis**: Describe el proceso mediante el cual se aplica un **PRS Model** a un paciente para calcular su PRS. Está vinculado a un conjunto de puntuación (**Scoring Set**) y proporciona la interpretación de los resultados.
- **Ancestry Distribution**: Define la distribución de ascendencia para una muestra poblacional. Se compone de una lista de categorías de ascendencia (`categories`) y sus porcentajes (`percentages`), lo cual es relevante para entender la aplicabilidad de un **PRS Model** en diferentes poblaciones.
- **Population Sample**: Representa un conjunto de datos utilizado en el desarrollo o evaluación de un **PRS Model**. Puede describirse mediante detalles sobre cómo se generó la muestra o características demográficas de la población, como la edad y el sexo.
- **GWAS SumStat**: Son estadísticas de resumen obtenidas de estudios de asociación de genoma completo (GWAS). Estas muestras contienen un número de variantes (`n_variants`) y están asociadas a conjuntos de datos a nivel individual.
- **Indiv-Lev Data**: Representa datos a nivel individual utilizados en estudios genéticos. Incluye el número de individuos (`n_individuals`) y el conjunto de datos (`cohort`) del cual se obtuvo la muestra.
- **Development**: Representa los conjuntos de datos para el desarrollo de un **PRS Model**. Incluye un conjunto de datos **Base**, el cual es obligatorio, y conjuntos de **Tuning** opcionales, utilizados para ajustar u optimizar el modelo.
- **Evaluation**: Representa el conjunto de datos para la evaluación de un **PRS Model** mediante una **Performance Metric**.
- **Performance Metric**: Define una métrica utilizada para evaluar el desempeño de un **PRS Model**. Tiene un nombre y una descripción de lo que mide, como la capacidad discriminatoria del modelo.
- **Metric Evaluation**: Es el resultado de la evaluación de una **Performance Metric**. Está vinculado tanto a la métrica evaluada como al **PRS Model** y proporciona un resultado numérico (`result`) junto con una descripción del mismo.

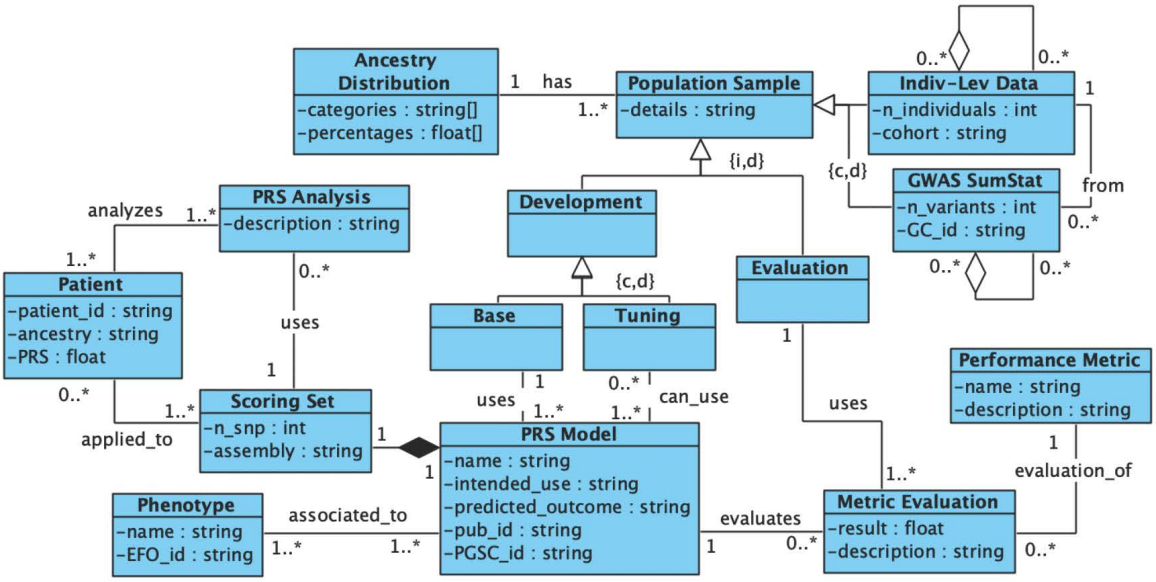


Figura 2.1: Modelo conceptual de modelos PRS [1]

2.1.1. PGS catalog

Repositorios como **PGS Catalog** [3] (Polygenic Score Catalog) es una base de datos especializada que compila información sobre **modelos PRS**, que son herramientas utilizadas para predecir la susceptibilidad de un individuo a ciertas enfermedades y condiciones de salud basándose en su perfil genético. Esta fuente de información es crucial en la investigación genética y la medicina personalizada, ya que permite a los investigadores y profesionales de la salud acceder a datos estandarizados sobre diferentes modelos PRS.
En el contexto de este trabajo, repositorios como **PGS Catalog** sirve como fuente primaria para el módulo de extracción de la solución propuesta, garantizando que los datos procesados provengan de un recurso confiable y actualizado.

2.1.2. Bibliografía: guías y recomendaciones

Dada la complejidad del dominio, especialmente para investigadores externos o clínicos, la literatura científica ofrece guías y recomendaciones para facilitar el uso de los modelos PRS y su adecuación al tipo de análisis. Ejemplos de estas guías son las proporcionadas por Lambert et al. [7], que proporcionan un marco para la estandarización del reporte de modelos PRS, o en Collister et al. [5], que buscan orientar a los ususarios en la correcta implementación de los modelos PRS en diferentes contextos. No obstante, estas recomendaciones son limitadas y carecen de una metodología clara, lo que impide una

automatización completa del proceso. A pesar de ello, pueden servir como una base sólida para el desarrollo de métodos que permitan su automatización y estandarización en el futuro. En este trabajo, dichas guías y recomendaciones sirven como base para el desarrollo del algoritmo de priorización necesario para el segundo módulo de la solución propuesta.

2.1.3. Genopred

GenoPred [8, 9] es una plataforma avanzada de análisis genético diseñada para aplicar Modelos PRS. Destacada por su capacidad para procesar grandes volúmenes de datos genéticos provenientes de fuentes como el UK Biobank y estudios de asociación a nivel genómico (GWAS), utilizando algoritmos de aprendizaje automático para asegurar la escalabilidad y reproducibilidad de los análisis. El objetivo principal de GenoPred es facilitar el uso de modelos PRS en investigaciones genéticas, permitiendo a los investigadores evaluar el riesgo genético de enfermedades complejas de manera eficiente.

2.2. Estado del arte

El Polygenic Risk Score (PRS) se utiliza ampliamente para estimar el riesgo genético de una persona a desarrollar diversas enfermedades. A lo largo de los últimos años, se han desarrollado varias herramientas que permiten tanto la generación como la aplicación de modelos PRS. En esta sección, se analizan las principales herramientas disponibles, centrándonos en sus utilidades, interoperabilidad, soporte del dominio y tipo de interfaz.

Una de las herramientas más completas es **GenoPred** [10, 9], que destaca por su capacidad de generar y aplicar modelos PRS. Esto la convierte en una opción versátil, ya que no solo permite el análisis PRS, sino también la creación de nuevos modelos basados en datos genómicos. Además, GenoPred es compatible con PGS catalog, lo que facilita el acceso a modelos validados por la comunidad científica. Sin embargo, al utilizar una interfaz de línea de comandos, requiere conocimientos técnicos para su manejo, lo que puede ser una barrera para usuarios menos experimentados.

En un nivel similar está **pgscalc** [11] una herramienta diseñada principalmente para aplicar modelos preexistentes de PGS Catalog. Aunque no genera modelos nuevos, es ideal para aquellos investigadores que buscan trabajar con modelos ya establecidos y probados. Al igual que GenoPred, pgscalc ofrece reportes visuales de los resultados, facilitando su interpretación. Sin embargo, comparte la limitación de requerir conocimientos de línea de comandos para su uso eficiente.

PGS Calculator [12] es otra herramienta que se centra en la aplicación de modelos PRS existentes. Aunque carece de la capacidad para generar nuevos modelos, es una opción confiable para aquellos que buscan evaluar riesgos genéticos utilizando modelos previamente desarrollados. Como las herramientas anteriores, PGS Calculator utiliza PGS Catalog para

acceder a modelos validados y es multiplataforma, lo que garantiza su adaptabilidad en diferentes entornos de investigación. Sin embargo, también se enfrenta a la misma limitación de usabilidad al no ofrecer una interfaz gráfica.

Por otro lado, **The Polygenic Index (PGI) Repository** [13] y **Post-GWAS Polygenic Risk Scores (PRS) Pipeline**[14] se centran más en la generación de modelos PRS a partir de datos de estudios genómicos (GWAS), pero no permiten la aplicación de modelos preexistentes ni ofrecen reportes visuales. Esto las convierte en herramientas más especializadas para investigadores avanzados que buscan desarrollar nuevos modelos a partir de datos crudos. A pesar de su interoperabilidad y flexibilidad como herramientas de código abierto, su falta de integración con PGS Catalog y la ausencia de una interfaz gráfica limitan su accesibilidad para usuarios menos experimentados en el campo de la genómica.

Herramientas como **PRS Knowledge Base** [15] proporcionan un equilibrio entre generación y aplicación de modelos, con la ventaja adicional de ofrecer tanto una interfaz gráfica como de línea de comandos. Aunque no prioriza modelos automáticamente, su soporte en la interpretación de resultados la hace más accesible para usuarios que no tienen una experiencia profunda en análisis PRS. Sin embargo, la falta de integración con PGS Catalog sigue siendo una limitación comparada con otras opciones más robustas.

PRS Tutorial [16] es una guía enfocada en ayudar a los usuarios a construir y ejecutar un pipeline para el análisis de PRS. Aunque no genera o aplica modelos de PRS por sí misma, esta herramienta sirve como un excelente recurso para investigadores que deseen implementar sus propias herramientas de análisis de PRS. La guía proporciona instrucciones claras sobre cómo realizar análisis PRS basados en datos GWAS, y es multiplataforma y de código abierto. Sin embargo, al ser un tutorial más que una aplicación directa, no incluye la visualización de resultados ni la priorización de modelos.

Por último la herramienta **GWA tutorial** [17] ofrece una introducción similar, aunque más enfocada en los pasos iniciales del análisis GWAS y cómo preparar los datos genéticos para la implementación de análisis de PRS. Este tutorial también ayuda a construir una herramienta desde cero para calcular PRS, pero no proporciona una solución preempaquetada. Al igual que PRS-Tutorial, su enfoque está en la enseñanza y orientación para el desarrollo de pipelines personalizados, lo que la convierte en una opción útil para aquellos que buscan más control sobre el proceso. Sin embargo, no incluye características como la integración directa con PGS Catalog o la visualización automatizada de resultados.

La Tabla 2.1 resume las características relevantes de estas herramientas para el presente trabajo. Estas característicasse listan a continucación.

- **Utilidades:** Se refiere a las capacidades y funciones que ofrece cada herramienta. En esta categoría se evalúa si la herramienta permite **generación y aplicación de modelos PRS (GAM)**, es decir, si es capaz tanto de crear nuevos modelos PRS como de aplicarlos en datos genéticos. También se analiza si es compatible con la **aplicación de modelos PRS preexistentes (AMP)**, lo que significa que puede utilizar modelos PRS externos sin necesidad de generarlos nuevamente. Finalmente, se evalúa si la

herramienta proporciona **visualización de resultados (VR)**, presentando los datos en forma de reporte o gráficos interpretables.

- **Interoperabilidad:** Se refiere a la capacidad de la herramienta para integrarse con repositorios de modelos PRS y su adaptabilidad a diferentes plataformas tecnológicas. En esta categoría se evalúa la **compatibilidad con PGS catalog (CPC)**, es decir, si la herramienta puede acceder y utilizar modelos PRS almacenados en este repositorio reconocido. También se analiza si la herramienta es **multiplataforma (MP)**, lo que indica si puede operar en diversos sistemas operativos y entornos tecnológicos. Por último, se considera si la herramienta es de **código abierto (CA)**, permitiendo a los usuarios acceder y modificar su código fuente.

- **Soporte del Dominio:** Se refiere a la facilidad que ofrece la herramienta para ser utilizada por usuarios que no son expertos en el ámbito de la investigación en PRS. Esta categoría incluye la **priorización de modelos PRS (PMP)**, que facilita a los usuarios seleccionar los modelos más relevantes y útiles para sus investigaciones. Además, se evalúa si la herramienta proporciona **asistencia en la interpretación de resultados de PRS (AIR)**, lo cual ayuda a los usuarios a entender y contextualizar los resultados obtenidos, mejorando así la accesibilidad y la aplicabilidad de los análisis realizados.

- **Interfaz:** El tipo de interfaz afecta la complejidad de uso de la herramienta. Una **interfaz de línea de comandos (ILC)** requiere conocimientos técnicos, mientras que una **interfaz gráfica (IG)** facilita la interacción, especialmente para usuarios con menos experiencia en informática.

	Utilidades			Interoperabilidad			Soporte Dominio		Interfaz	
Herramienta	**GAM**	**AMP**	**VR**	**CPC**	**MP**	**CA**	**PMP**	**AIR**	**ILC**	**IG**
GenoPred	SÍ	SÍ	SÍ	SÍ	SÍ	SÍ	NO	SÍ	SÍ	NO
pgs_calc	NO	SÍ	SÍ	SÍ	SÍ	SÍ	NO	NO	SÍ	NO
PGS Calculator	NO	SÍ	SÍ	SÍ	SÍ	SÍ	NO	NO	SÍ	NO
The Polygenic Index (PGI) Repository	SÍ	NO	NO	NO	SÍ	SÍ	NO	NO	SÍ	NO
Post-GWAS polygenic risk scores (PRS) pipeline	SÍ	NO	NO	NO	SÍ	SÍ	NO		SÍ	NO
PRS Knowledge Base	SÍ	NO	SÍ	NO	SÍ	SÍ	NO	NO	SÍ	SÍ
PRS Pipeline*	NO	SÍ	NO	SÍ	SÍ	SÍ	NO	NO	SÍ	NO
PRS-Tutorial*	SÍ	NO	NO	NO	SÍ	SÍ	NO	NO	SÍ	NO
GWA_tutorial*	SÍ	NO	NO	NO	NO	SÍ	NO	NO	SÍ	NO

Tabla 2.1: Estado del arte: herramientas de análisis PGS.

Como se observa en la Tabla 2.1, no hay una herramienta que brinde soporte completo al proceso priorizado de análisis con modelos PGS. Entre las opciones disponibles, GenoPred es la alternativa más integral para ser incorporada en el desarrollo del sistema. Sin embargo, aún requiere la extracción, transformación y priorización de los modelos a analizar, así como su integración en una aplicación web que facilite su uso para el usuario final.

Capítulo 3

Definición del Problema y Análisis de Requerimientos

En esta sección se abordarán aspectos clave para el desarrollo del sistema, incluyendo la formulación del problema, la propuesta de solución, los objetivos del proyecto, la metodología empleada, los requerimientos del sistema y sus funcionalidades principales. Este análisis permite establecer una base sólida para el diseño e implementación de una herramienta que integre, priorice y analice modelos PRS a partir de datos públicos, asegurando su utilidad y alineación con las necesidades de los usuarios.

3.1. Formulación del Problema

El problema surge cuando un clínico debe seleccionar un modelo PRS adecuado y aplicarlo al genoma del paciente. En el proceso de selección, se presentan dos problemas principales. El primer problema es la falta de estandarización y la variabilidad entre modelos PRS lo que presenta un problema significativo para la comparación entre modelos. Aunque hay diversos modelos disponibles en la literatura, siendo PGS Catalog uno de los repositorios más conocidos [3], la ausencia de estandarización dificulta la selección del modelo más adecuado. Para mejorar la estandarización del dominio se ha propuesto como solución un modelo conceptual para la precisa conceptualización del dominio. El estudio [1] presenta un modelo conceptual ilustrado en la Figura 3.7 que delimita el marco de trabajo de la investigación en PRS dirigido a clínicos y epidemiólogos, tomando como piedra angular los Modelo PRS y los factores que toman parte en el proceso de comparación y selección de modelos. Además se definen características como el fenotipo de interés, información sobre los datasets para generar los modelos (número de individuos, ancestría, etc), las métricas de rendimiento evaluadas para los modelos.

El segundo problema es la priorización de modelos PRS que se complica debido al gran número de características que deben considerarse. Aunque la literatura ofrece

recomendaciones para la priorización de modelos [5], no existe un enfoque metodológico claro para manejar esta priorización de manera efectiva. Esto añade una capa adicional de dificultad al proceso de selección.

Finalmente para aplicar el modelo PRS seleccionado, esto es realizar un análisis de PRS, se requiere de un software (conocido como pipeline) que procese los datos genómicos del paciente y aplique un Modelo PRS para analizar la predisposición genética a cierta enfermedad. El desarrollo de estos pipelines es costoso y requiere de habilidades bioinformáticas específicas del dominio de los PRSs. Afortunadamente, existen varias herramientas para el análisis de PRS como [9, 7, 18]. Estas herramientas son útiles porque agilizan el proceso de análisis, sin embargo, suelen requerir conocimiento especializado del dominio, así como habilidades informáticas avanzadas, para funcionar correctamente. Además, aunque permiten seleccionar el modelo PRS deseado, no ofrecen ningún tipo de recomendación o clasificación de modelos PRS en función de su adecuación al análisis.

La selección inadecuada del modelo PRS puede comprometer la fiabilidad de los resultados del análisis, lo que afecta negativamente la precisión del diagnóstico y las decisiones clínicas. Por lo tanto, es crucial optimizar los resultados del análisis y facilitar la implementación de modelos PRS en la práctica clínica para mejorar la precisión diagnóstica y la calidad del cuidado del paciente.

3.2. Solución Propuesta

Para abordar la problemática presentada, este trabajo de tesis se enfocó en el desarrollo de un sistema de **extracción y priorización de modelos PRS**, mediante técnicas de ingeniería de software. El objetivo principal es gestionar eficientemente la información y sistematizar el proceso de priorización de modelos PRS, con el fin de utilizar los modelos priorizados en un proceso de análisis posterior, que se implementa como un módulo independiente.

El proyecto se compone de dos módulos principales, como se muestra en la Figura 3.7: **extracción** y **priorización**.

- **Extracción**: Este módulo fue desarrollado en `Node.js` y se implementó siguiendo el enfoque *ETL* (Extract, Transform, Load). El sistema realiza la extracción de datos desde el repositorio del *PGS Catalog* [3], transformando posteriormente la información mediante procesos como el filtrado de datos irrelevantes, la estandarización de nombres de columnas y el mapeo de relaciones. Finalmente, los datos son cargados en una base de datos estructurada, permitiendo su uso eficiente en los procesos posteriores de priorización y análisis.

- **Priorización**: En este módulo se consideraron múltiples metadatos de los modelos PRS para poder compararlos, como ancestría del modelo, métricas evaluadas como

AUC y OR, y ancestría del dataset de evaluación, y se realizaron cálculos y agrupaciones en base a estos metadatos. Se realizaron cálculos y agrupaciones según criterios como la ancestría, el rasgo o el tipo de muestra poblacional. A partir de estos calculos, se generó un sistema de ranking que permite priorizar los modelos más adecuados para el análisis genético. Este proceso considera tanto la calidad del modelo como su aplicabilidad a poblaciones específicas, facilitando su uso clínico y orientando el trabajo de los consultores genéticos.

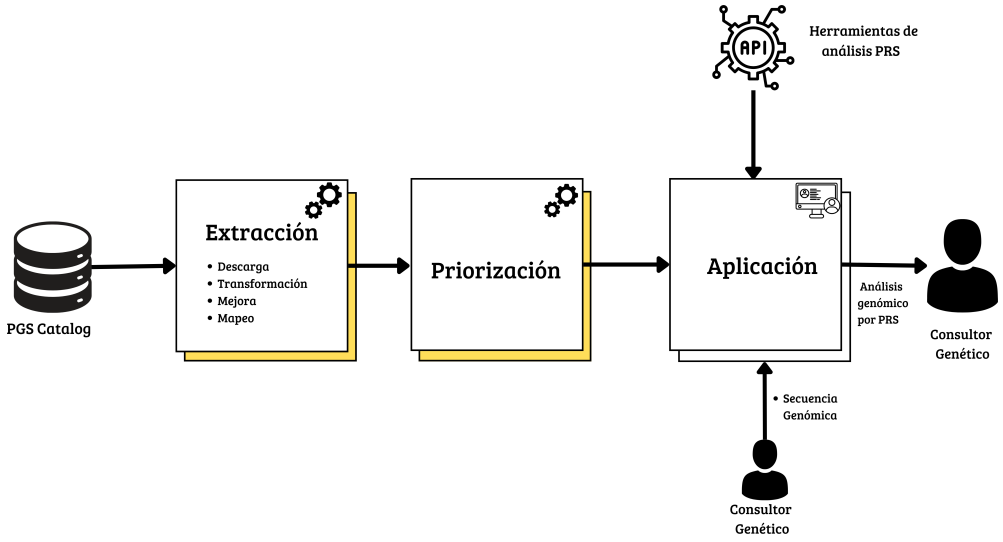


Figura 3.1: Diagrama de solución propuesta. El módulo de aplicación se encuentra fuera del alcance de este proyecto y será desarrollado por un tercero encargado del análisis genético clínico.

3.2.1. Importancia del trabajo

Este trabajo es de gran relevancia tanto desde un punto de vista científico como social. En el ámbito social, su impacto se refleja en la mejora de la atención médica personalizada, especialmente para enfermedades poligénicas como las cardiovasculares y varios tipos de cáncer. Al optimizar el análisis de PRS, mediante una mejor identificación y selección de los modelos más adecuados al contexto de análisis, se facilita una identificación más precisa de los riesgos genéticos, lo que permite a los profesionales de la salud tomar decisiones mejor fundamentadas y adaptadas a las características genómicas de cada paciente. Esto se

traduce en un beneficio directo para la sociedad, ya que contribuye a una atención sanitaria más personalizada y preventiva, mejorando así la calidad de vida de los pacientes.

Desde el enfoque científico, este trabajo representa una contribución significativa al perfeccionamiento de los análisis PRS, abordando desafíos clave como la falta de estandarización y la variabilidad entre los modelos existentes. Al superar estas limitaciones, se aumenta la precisión en la predicción de predisposiciones genéticas, lo que no solo mejora las prácticas actuales, sino que también impulsa nuevas investigaciones y desarrollos tecnológicos en el campo de la genética.

3.3. Objetivos

3.3.1. Objetivo General

Desarrollar un sistema para la priorización de modelos PRS con la finalidad de mejorar la efectividad de los análisis de predisposición genética de enfermedades complejas.

3.3.2. Objetivos Específicos

1. Analizar el modelo conceptual del marco de trabajo de los PRS para identificar requerimientos de información de modelos PRS y del proceso de priorización de modelos para el análisis PRS.

2. Desarrollar un proceso de extracción, transformación y carga de datos de modelos PRS a la base de datos PEGASUS.

3. Desarrollar una aplicación para la selección y priorización de modelos PRS para la preparación de un análisis de riesgo poligénico.

4. Validar la aplicación mediante pruebas unitarias del proceso ETL y de aceptación de usuarios expertos en el análisis PRS.

3.4. Metodología

Para el desarrollo de este proyecto se adoptó la **metodología secuencial**, también conocida como *modelo en cascada*, según lo propuesto por Pressman [2]. Esta metodología estructura el proceso de desarrollo en una serie de fases claramente definidas, donde cada etapa depende de la finalización de la anterior, como se ilustra en la Figura 3.2. Aunque el modelo sigue un flujo lineal, incorpora puntos de retroalimentación entre fases, permitiendo revisar y ajustar entregables antes de avanzar, lo que incrementa la calidad y alineación del sistema con los objetivos definidos.

En la fase de Comunicación, se establecen las bases del proyecto, incluyendo la comprensión de los objetivos y los resultados esperados. Además, se realiza una recopilación exhaustiva de los requerimientos del sistema, definiendo las funcionalidades necesarias para cumplir con las expectativas establecidas. Durante la Planificación, se identifican los hitos clave que marcarán el progreso del proyecto. Se establecen fechas y plazos para las entregas parciales y la finalización de cada fase. En la fase de Diseño, se detallan los procesos del proyecto, como la extracción de datos, la priorización de modelos y la aplicación de modelos PRS. Esta etapa implica crear un plan detallado sobre cómo se llevarán a cabo estas actividades y qué recursos serán necesarios. La Construcción es donde se desarrolla e implementa el código, seguido de pruebas exhaustivas para asegurar que cumple con los requisitos establecidos. Finalmente, en la fase de Despliegue, se realiza la entrega del proyecto final. Tras la entrega, se recopila feedback para identificar posibles mejoras y realizar ajustes futuros.

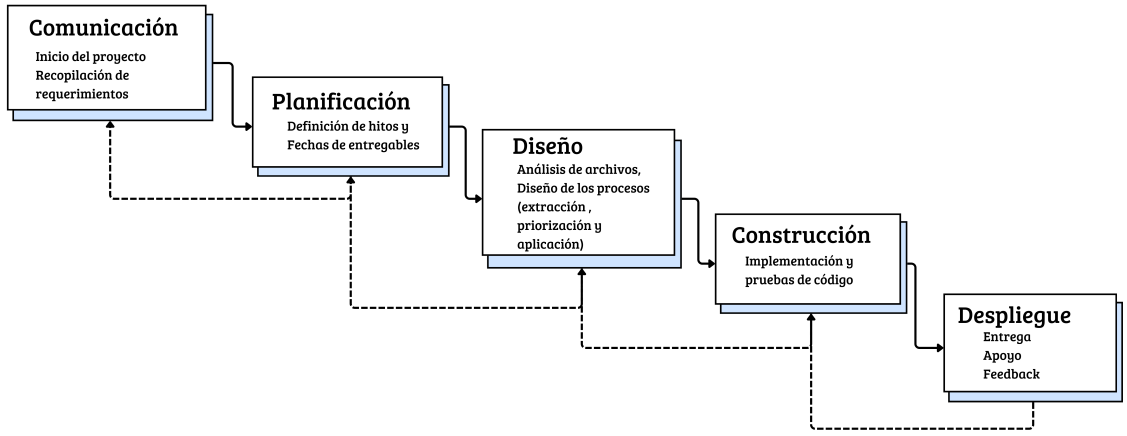


Figura 3.2: Diagrama metodología secuencial [2].

3.5. Especificación de Requerimientos

En esta sección se presenta el proceso de análisis genómico tal como se realiza actualmente (AS IS) y la visión futura esperada con la implementación del nuevo sistema propuesto (TO BE). A continuación, se describen ambos escenarios con sus respectivos diagramas de procesos de negocio.

3.5.1. Situación Actual (AS IS)

En la situación actual, el proceso de análisis de riesgo poligénico es llevado a cabo de manera secuencial y con un alto grado de intervención manual por parte de diversos actores, como se observa en la Figura 3.3. El proceso se inicia a partir de la solicitud médica y se extiende hasta la entrega e interpretación de resultados por parte del equipo clínico.

A continuación, se describen las principales actividades identificadas en el flujo actual:

- **BP01 – Solicitar análisis de riesgo poligénico:** El proceso comienza cuando un médico solicita la evaluación del riesgo genético de un paciente. Esta solicitud es gestionada por un equipo de asesoría genética o investigadores especializados.
- **BP02 – Identificar modelos para el análisis:** El investigador busca modelos PRS disponibles en repositorios públicos como *PGS Catalog*, *Cancer PRS Web* o *PGS Depot*, considerando el rasgo o enfermedad de interés.
- **BP03 – Priorizar modelos PRS:** La priorización de modelos se realiza manualmente, tomando en cuenta factores como la ancestría del paciente, el tipo de estudio, la población utilizada en el desarrollo del modelo y las métricas de evaluación publicadas.
- **BP04 – Realizar análisis genómico:** Una vez priorizado el modelo, se procede a realizar el análisis genómico sobre la muestra del paciente, generando las variantes necesarias para aplicar el modelo PRS.
- **BP05 – Analizar riesgo aplicando modelos PRS:** Se ejecuta el modelo PRS priorizado sobre los datos del paciente, calculando la puntuación de riesgo poligénico asociada.
- **BP06 – Reportar resultados de análisis genómico:** Los resultados obtenidos son consolidados en un reporte técnico que incluye la interpretación preliminar del riesgo genético.
- **BP07 – Interpretar resultados:** Un especialista en asesoría genética interpreta clínicamente los resultados, considerando el contexto del paciente y los datos del informe.
- **BP08 – Recibir análisis de riesgo poligénico:** Finalmente, el médico recibe el informe interpretado y lo comunica al paciente como parte del proceso clínico.

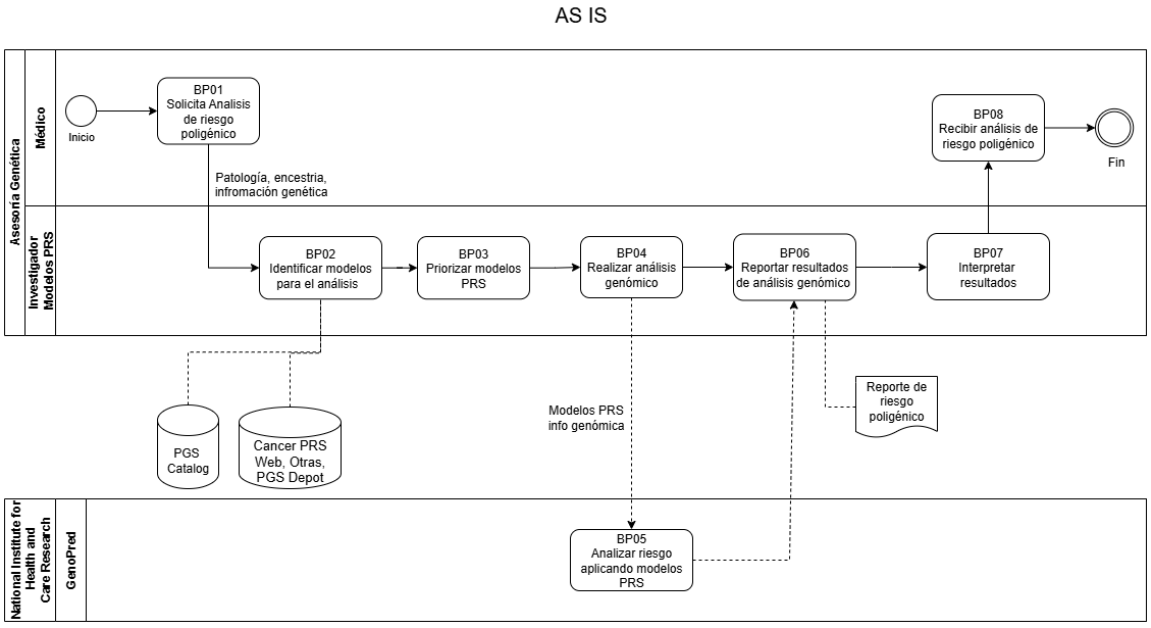


Figura 3.3: Diagrama BPMN de la situación actual (AS IS).

3.5.2. Situación Propuesta (TO BE)

En la situación propuesta, se automatizan dos de las etapas clave del flujo de análisis de riesgo poligénico: la obtención de modelos PRS desde repositorios y su priorización para el análisis. Estas actividades, previamente realizadas de forma manual, son abordadas mediante el desarrollo de un sistema que sistematiza el acceso, filtrado y ordenamiento de los modelos de puntuación de riesgo poligénico (PGS), como se muestra en la Figura 3.4.

- **Obtener modelos de análisis PGS:** El sistema realiza una extracción automatizada desde el repositorio público *PGS Catalog*, utilizando un proceso de integración tipo *ETL* (Extract, Transform, Load). Este proceso permite obtener modelos PRS relevantes, transformar y depurar los datos, y finalmente cargarlos en una base de datos diseñada desde cero, construida en base a un modelo conceptual específico. Esta automatización garantiza la calidad y estructuración de los datos, reduce tiempos de búsqueda, y evita omisiones en la recolección de modelos disponibles para el análisis.

- **Priorizar modelos PGS para hacer análisis:** Una vez obtenidos los modelos, se aplica un algoritmo de priorización basado en criterios definidos, tales como la ancestría del paciente, las métricas de desempeño del modelo (como AUROC u OR), y la población utilizada en su validación. El sistema calcula un puntaje de priorización

y entrega un ranking de modelos adecuados, facilitando la selección de aquellos que presentan mayor utilidad clínica para el contexto del análisis.

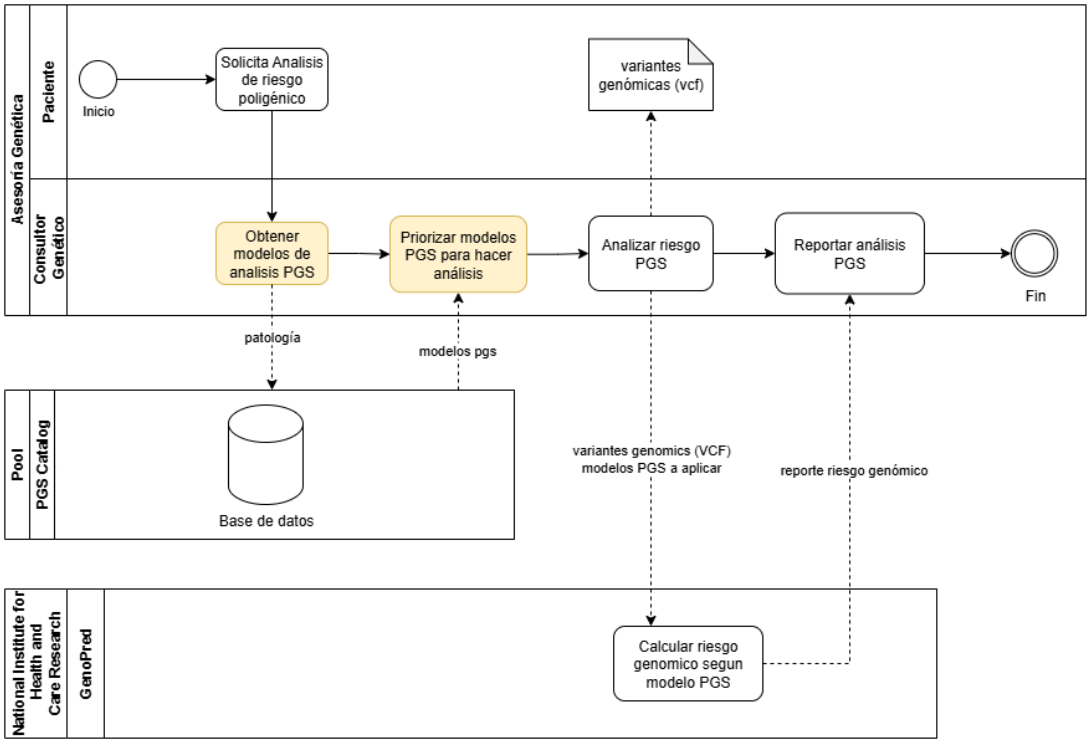


Figura 3.4: Situación propuesta (TO BE). Se destacan las actividades automatizadas por el sistema desarrollado: obtención y priorización de modelos PRS.

3.5.3. Requerimientos funcionales

El proceso anterior va a ser soportado por un sistema de información a través de las siguientes funcionalidades.

ID	Requerimiento Funcional	Descripción	Prioridad
BP02-RF01	Identificar modelos para el análisis	El sistema debe permitir la extracción de información desde PGS Catalog sobre los modelos disponibles para realizar la priorización correspondiente.	Alta
BP02-RF02	Identificar modelos para el análisis	El sistema debe facilitar el proceso de transformación de los datos extraídos del PGS Catalog.	Alta
BP02-RF03	Identificar modelos para el análisis	El sistema debe permitir un proceso de mejora de los datos para garantizar su calidad y relevancia.	Alta
BP02-RF04	Identificar modelos para el análisis	El sistema debe permitir la realización de mapeo de los datos extraídos, asegurando que se alineen adecuadamente con los requerimientos del análisis.	Alta
BP03-RF05.	Priorizar modelos PRS	El sistema debe permitir priorizar los modelos PRS para un mismo fenotipo de acuerdo a diversos criterios tales como la ancestría, las métricas evaluación del modelo.	Alta

Tabla 3.1: Requerimientos funcionales del sistema.

3.5.4. Requerimientos no funcionales

En esta sección se presentan los requerimientos no funcionales del sistema, alineados con la norma ISO/IEC 25010 [19]. Estos requerimientos definen criterios esenciales de calidad que no están directamente relacionados con las funcionalidades del sistema, pero que influyen significativamente en su desempeño, fiabilidad, usabilidad, seguridad y mantenimiento.

ID	Requerimiento No Funcional	Descripción	Prioridad
RNF01	Usabilidad	La interfaz del sistema debe ser intuitiva y fácil de utilizar, especialmente para consultores genéticos, sin necesidad de capacitación técnica avanzada.	Alta
RNF02	Rendimiento	El sistema debe ejecutar el proceso de priorización de modelos PRS en menos de 5 minutos por consulta, considerando una entrada promedio de 20 modelos.	Alta
RNF03	Seguridad	Toda la información sensible, incluyendo configuraciones y resultados de priorización, debe estar protegida mediante autenticación y cifrado de datos en tránsito y en reposo.	Muy Alta
RNF04	Mantenibilidad	El sistema debe permitir la actualización de modelos y ajustes al algoritmo de priorización sin necesidad de interrupciones en la operación general.	Media
RNF05	Portabilidad	El sistema debe poder ser desplegado tanto en entornos locales (Linux, Windows) como en la nube, utilizando contenedores (Docker).	Media
RNF06	Interoperabilidad	El sistema debe ser capaz de conectarse automáticamente con el repositorio público *PGS Catalog* para extraer y actualizar modelos PRS mediante el proceso ETL.	Alta

Tabla 3.2: Requerimientos No Funcionales del Sistema

3.6. Funcionalidades del sistema

En esta sección se describen las principales capacidades del sistema desarrollado, abarcando tanto el proceso de extracción y gestión de datos como la priorización de modelos PRS para su posterior análisis. Para ello, se presentan los diagramas y descripciones formales de los casos de uso, que permiten comprender las interacciones entre los actores y el sistema, así como los flujos de trabajo implicados. También se incluyen diagramas de secuencia y el modelo conceptual que sustentan la lógica y organización de la información.

3.6.1. Diagramas de casos de uso

El siguiente diagrama (ver Figura 3.5) representa los principales casos de uso del sistema de extracción y priorización de modelos PRS. Se ilustran las interacciones entre los actores externos y el sistema, en particular aquellos que participan en la obtención de modelos desde el *PGS Catalog* y en la priorización de dichos modelos para su posterior análisis clínico.

En esta versión del sistema, los procesos han sido automatizados en gran parte, por lo que el rol del sistema es actuar como intermediario entre la fuente de datos (PGS Catalog) y el consultor genético, facilitando la selección de modelos adecuados. También se contempla la integración con una herramienta externa de análisis PRS.

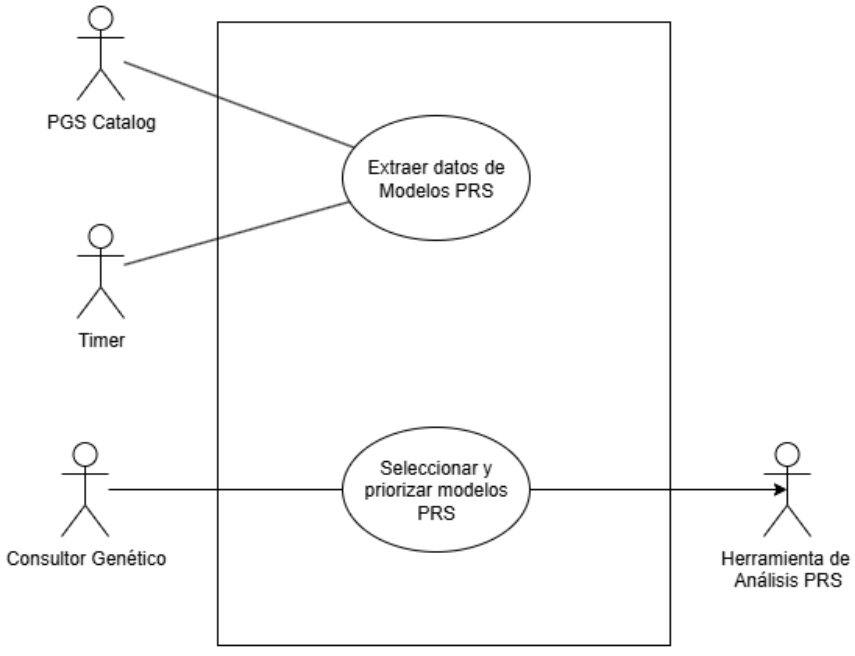


Figura 3.5: Diagrama de casos de uso.

3.6.2. Casos de Uso

A continuación, se describen los principales casos de uso asociados al sistema:

- **Extraer datos de modelos PRS:** El sistema obtiene los modelos desde el repositorio *PGS Catalog*, ejecutando un proceso ETL automatizado que transforma y carga la información en una base de datos local estructurada.
- **Seleccionar y priorizar modelos PRS:** El consultor genético accede al sistema para seleccionar los modelos que desea evaluar, y el sistema genera automáticamente un ranking basado en métricas como AUC, OR y grado de coincidencia con la ancestría del paciente.
- **Integración con herramienta de análisis PRS:** Una vez priorizados, los modelos pueden ser utilizados por herramientas externas que aplican las puntuaciones de riesgo genómico a las variantes (VCF) del paciente.

Cada uno de estos casos de uso puede ser detallado posteriormente mediante descripciones formales, incluyendo precondiciones, flujo de eventos, excepciones y poscondiciones, con el fin de documentar el comportamiento esperado del sistema en distintos escenarios.

Caso de Uso	**Extraer datos de Modelos PRS**
Descripción	El sistema ejecuta un proceso ETL para obtener los modelos PRS desde el repositorio público *PGS Catalog*. A través de este proceso se realiza la descarga, transformación y carga de datos hacia una base de datos estructurada basada en un modelo conceptual propio. Además, durante la fase de transformación, se generan campos adicionales para enriquecer la información y facilitar los procesos de priorización.
Precondición	El catálogo PGS debe estar disponible y accesible vía HTTP.
Secuencia Normal	1. El sistema se conecta automáticamente al catálogo *PGS Catalog*. 2. Descarga los archivos CSV de metadatos (modelos, muestras, métricas, publicaciones, etc.). 3. Se realiza la transformación de los datos para adaptarlos al modelo conceptual definido. 4. Durante la transformación se crean campos adicionales que no existen en los archivos originales, como: ▪ **SUM1:** Total de individuos en las muestras de desarrollo (`n_indiv`) para un modelo PRS. ▪ **SUM2:** Total de individuos agrupados por *Broad Ancestry* (por símbolo) asociados a las muestras de desarrollo de ese modelo. ▪ **Porcentaje de representación por ancestría:** Se calcula como `(SUM2 / SUM1) * 100`. 5. Se insertan los modelos transformados y enriquecidos en la base de datos propia, manteniendo integridad referencial entre tablas.
Poscondición	Los modelos PRS han sido correctamente descargados, transformados y cargados en la base de datos local. La información cuenta con campos enriquecidos que permiten análisis más específicos.
Excepciones	Si no se puede acceder al catálogo PGS, se muestra un mensaje de error y se detiene el proceso de extracción.

Tabla 3.3: Caso de Uso: Extraer datos de Modelos PRS

Caso de Uso	**Priorizar Modelos PRS**
Descripción	El sistema realiza una priorización de los modelos PRS previamente extraídos, considerando criterios como ancestría, métricas de desempeño y patología objetivo.
Precondición	Los modelos PRS deben haber sido previamente extraídos y cargados en la base de datos.
Secuencia Normal	1. El consultor genético accede al sistema. 2. Selecciona los fenotipos de interés. 3. El sistema aplica el algoritmo de priorización definido. 4. Se genera un ranking de modelos PRS priorizados.
Poscondición	Los modelos PRS han sido ordenados según criterios relevantes y están listos para ser utilizados en fases posteriores.
Excepciones	Si no hay modelos en la base de datos o falta algún criterio, se notifica al usuario.

Tabla 3.4: Caso de Uso: Priorizar Modelos PRS

.

3.6.3. Diagrama de Secuencia

El siguiente diagrama de secuencia (ver Figura 3.6) describe el flujo de interacción entre los principales actores del sistema durante el proceso de extracción y priorización de modelos PRS. Este proceso involucra tres participantes clave: el **consultor genómico**, el **sistema** desarrollado y el repositorio externo **PGS catalog**.

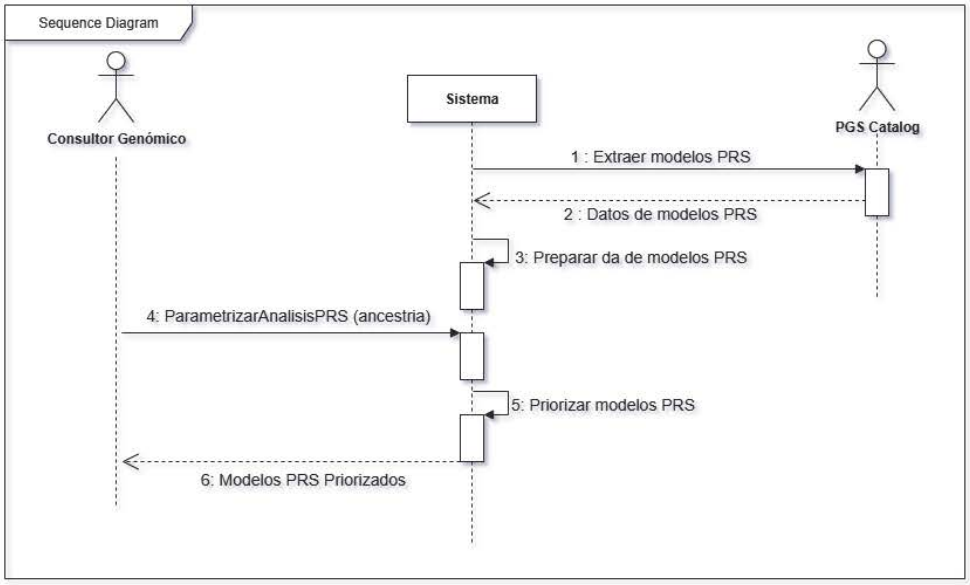


Figura 3.6: Diagrama de secuencia.

El proceso comienza cuando el sistema realiza una solicitud al *PGS catalog* para extraer los modelos PRS disponibles (paso 1). Una vez recibidos los datos (paso 2), el sistema transforma y prepara la información de acuerdo con el modelo conceptual definido (paso 3). En esta etapa se calculan campos adicionales como el total de individuos por modelo (SUM1), el total por ancestría (SUM2), y el porcentaje de representación ancestral para cada modelo.

Posteriormente, el **Consultor Genómico** parametriza el análisis, por ejemplo, indicando la ancestría del paciente (paso 4). Con esta información, el sistema procede a ejecutar el algoritmo de priorización (paso 5), generando una lista de modelos PRS ordenados según criterios relevantes. Finalmente, el sistema devuelve al consultor la lista de **modelos PRS priorizados** (paso 6), los cuales pueden ser utilizados en un módulo posterior de análisis (fuera del alcance del presente trabajo).

3.6.4. Modelo conceptual

El modelo conceptual desarrollado para este trabajo representa de forma estructurada los elementos clave involucrados en la extracción, priorización y análisis de modelos PRS. Este modelo permite visualizar cómo se relacionan las entidades necesarias para construir una solución funcional y escalable dentro del dominio de la bioinformática y la medicina personalizada.

El proceso comienza con los modelos PRS (`PRSModel`), los cuales agrupan información sobre su origen, publicación asociada, número de variantes genéticas (`SNPs`) utilizadas,

y las características del archivo de puntuación correspondiente (ScoringFile). Cada modelo está vinculado a uno o varios rasgos fenotípicos (Trait), que a su vez están organizados en categorías (TraitCategory) para facilitar su navegación.

El modelo conceptual también incluye la caracterización de las poblaciones utilizadas tanto en el desarrollo como en la evaluación de los modelos. Las muestras poblacionales de desarrollo (DevelopmentPopulationSample) y evaluación (EvaluationPopulationSample) contienen datos demográficos relevantes como el número de individuos, proporción de sexo, ancestría, edad y cohortes asociadas. Estas muestras están relacionadas con las categorías de ancestría (BroadAncestryCategory) y su representación porcentual en los modelos y poblaciones de referencia (BroadAncestryInModel, BroadAncestryInRefPop).

Para evaluar el desempeño de los modelos, se utilizan métricas de rendimiento (PerformanceMetric) cuyas evaluaciones específicas se almacenan en PerformanceMetricEvaluation. Estas permiten analizar la capacidad predictiva de los modelos a través de medidas como el AUC o el Odds Ratio, siendo fundamentales para su priorización.

Los modelos también están asociados a publicaciones científicas (Publication) que respaldan su validez y aplicación. A su vez, las poblaciones de referencia utilizadas en la interpretación clínica se modelan en la entidad ReferencePopulation, proporcionando contexto genómico.

Finalmente, los usuarios pueden realizar análisis sobre su información genómica a través de instancias de PRSAnalysis, seleccionando modelos priorizados (PrioritizedModel) en base a su ancestría y rasgos de interés. Los resultados derivados del análisis se almacenan en PRSResult, completando así el ciclo funcional del sistema.

Este modelo conceptual permite una representación integral del dominio, asegurando la correcta integración entre los módulos de extracción, priorización y análisis genómico.

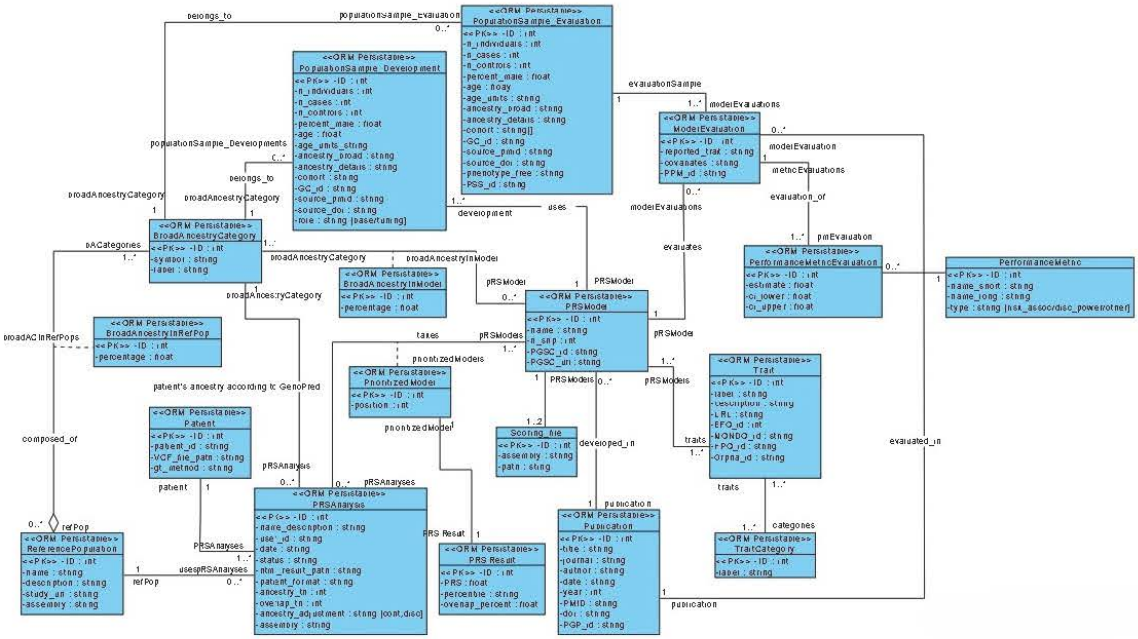


Figura 3.7: Modelo conceptual de modelos PRS [1]

Capítulo 4

Diseño

En este capítulo se presenta el diseño arquitectónico de la solución, describiendo las decisiones estructurales que permiten integrar de forma coherente el flujo ETL, la capa de servicios y la interfaz web. Se detallan las tecnologías seleccionadas, el flujo de datos a nivel alto, y la organización lógica y física de los componentes (despliegue, componentes, paquetes y datos). El objetivo es proveer una base modular y escalable que facilite el mantenimiento, la extensión del sistema y su adopción en distintos entornos.

4.1. Diseño Arquitectónico

El diseño arquitectónico define la estructura y organización de los componentes del sistema, así como las interacciones entre ellos. En este capítulo se describe la arquitectura propuesta para integrar de forma coherente el proceso ETL, el backend y la interfaz web. Además, se detallan las tecnologías empleadas y la vista de alto nivel del sistema, con el fin de comprender su funcionamiento global.

4.1.1. Tecnologías utilizadas

- **MySQL [20]:** Motor de base de datos relacional utilizado para almacenar la información estructurada extraída del *PGS Catalog*. La base de datos fue diseñada desde cero a partir de un modelo conceptual, permitiendo una gestión eficiente de modelos PRS, métricas, ancestrías, publicaciones, entre otros elementos clave del sistema.
- **Node.js [21]:** Plataforma utilizada en dos componentes principales del sistema: (1) para implementar el proceso *ETL* que extrae, transforma y carga los datos desde el *PGS Catalog* hacia la base de datos; y (2) para desarrollar el backend de la aplicación web, que expone una API encargada de procesar consultas, manejar filtros y aplicar el algoritmo de priorización de modelos PRS.

- **Prisma [22]:** ORM (Object-Relational Mapping) utilizado junto con Node.js para facilitar las operaciones sobre la base de datos MySQL. Permite definir esquemas relacionales, hacer migraciones automáticas y realizar consultas complejas de forma eficiente y tipada.
- **Next.js [23]:** Framework utilizado para el desarrollo del frontend de la aplicación web. A través de su arquitectura basada en componentes y renderizado híbrido (SSR y CSR), permite visualizar interactivamente la selección de fenotipos y los modelos PRS priorizados, presentando al usuario final un ranking detallado y ordenado de modelos según criterios definidos.
- **PGS Catalog [3]:** Fuente oficial de datos que provee información sobre modelos de puntuación de riesgo poligénico. Su estructura en archivos CSV descargables permite la automatización del proceso de carga y análisis.

4.1.2. Flujo de datos / Vista de alto nivel

La arquitectura del sistema se basa en un flujo de trabajo estructurado y automatizado, orientado a la extracción, transformación, almacenamiento y priorización de modelos PRS. En la Figura 4.1, se presenta una vista de alto nivel del sistema.

El proceso comienza con un **timer** que activa automáticamente el módulo **ETL**, desarrollado en **Node.js**, encargado de extraer los datos desde el repositorio público **PGS Catalog**. Este módulo descarga y transforma los datos, y luego los inserta en una base de datos relacional implementada en **MySQL**, utilizando **Prisma** como ORM para asegurar integridad y flexibilidad en las operaciones sobre los datos.

El **backend**, también desarrollado en **Node.js**, expone una **API REST** que permite consultar y priorizar los modelos PRS almacenados. Esta capa intermedia se encarga de ejecutar el algoritmo de priorización, realizar filtrados según los criterios definidos, y preparar los datos para su visualización.

El **frontend**, desarrollado en **Next.js**, consume esta API y presenta una interfaz interactiva donde los usuarios pueden seleccionar fenotipos, visualizar modelos PRS priorizados y explorar información relevante de manera amigable. La combinación de estas tecnologías permite una experiencia fluida y escalable para el análisis de riesgo genético.

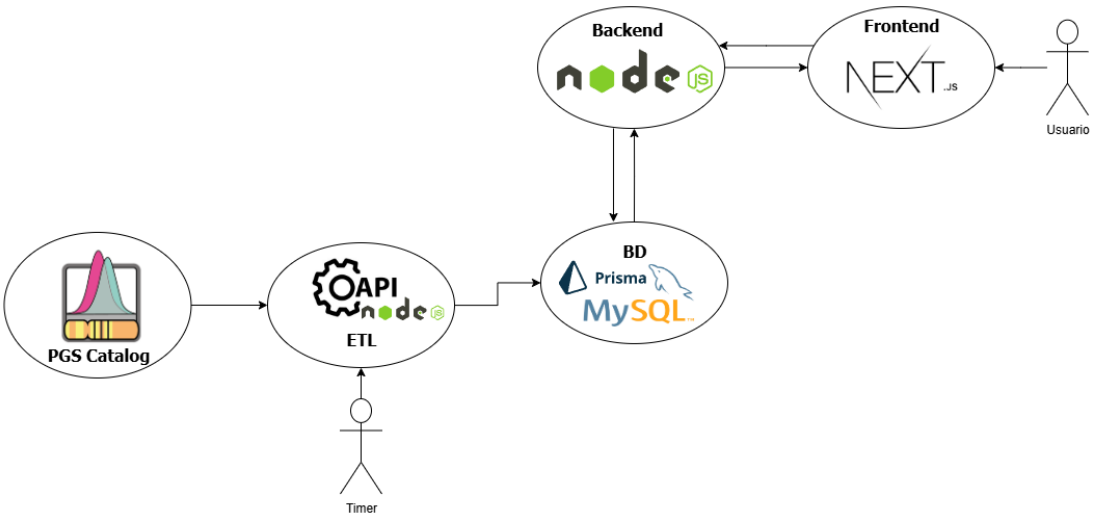


Figura 4.1: Vista de alto nivel

4.2. Diseño Lógico

El diseño lógico define la estructura interna del sistema, detallando cómo se organizan y relacionan sus componentes para cumplir con los requisitos funcionales y no funcionales establecidos. En esta sección se presentan los diagramas y descripciones de los elementos clave, incluyendo la distribución física de los componentes, su interacción mediante servicios y APIs, la organización modular en paquetes, y la representación detallada de las entidades y sus relaciones a través del diagrama de clases.

4.2.1. Diagrama de despliegue

El diagrama de despliegue representado en la Figura 4.2 muestra cómo se distribuyen físicamente los componentes del sistema, así como sus interacciones. A continuación, se describen sus principales elementos:

Componentes principales y su función

- **Consultor Genómico**: Usuario que interactúa con el sistema a través de una interfaz web. Es quien consulta, visualiza y prioriza modelos PRS mediante el frontend.
- **Timer**: Representa un agente automatizado que activa periódicamente el proceso de extracción y carga desde el repositorio externo.
- **Servidor Web**: Es el núcleo funcional del sistema y aloja los siguientes componentes:

- **Controlador ETL (Node.js)**: Módulo que coordina y orquesta la ejecución del proceso ETL, siendo activado por el temporizador.
- **Backend ETL (Node.js)**: Se encarga de realizar la extracción de datos desde el repositorio *PGS Catalog*, su transformación y almacenamiento en la base de datos.
- **Backend PRS system (Node.js)**: Provee servicios y lógica de negocio para la priorización y gestión de modelos PRS, interactuando tanto con la base de datos como con el frontend.
- **Frontend PRS system (react-next)**: Interfaz gráfica que permite al Consultor Genómico visualizar los datos, seleccionar modelos, aplicar filtros y ver resultados priorizados.

- **Repositorio de modelos (PGS Catalog)**: Fuente externa de datos de puntuación de riesgo poligénico. El sistema se conecta mediante solicitudes HTTP para obtener los modelos.
- **Base de Datos (MySQL)**: Motor de base de datos relacional que almacena los modelos obtenidos y transformados por el sistema ETL. También mantiene la información necesaria para el funcionamiento del sistema de priorización.

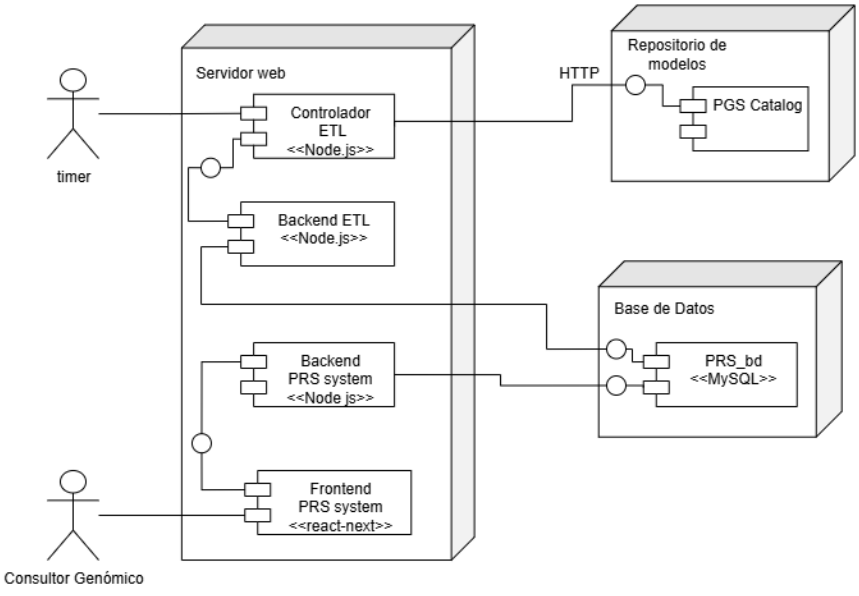


Figura 4.2: Diagrama de despliegue

.

4.2.2. Diagrama de componentes

El diagrama de componentes presentado en la Figura 4.3 muestra la estructura y comunicación entre los distintos módulos funcionales del sistema, los cuales colaboran en la extracción, transformación y visualización de modelos de riesgo poligénico. A continuación, se detallan sus elementos principales:

- **PGS catalog**:
  - Repositorio externo accesible vía HTTP que contiene modelos de puntuación de riesgo poligénico.
  - Proporciona los datos de entrada al sistema a través del módulo de extracción.
- **Controlador ETL («Node.js»)**:
  - Encargado de coordinar el proceso ETL (extracción, transformación y carga) activado por un temporizador.
  - Inicia la conexión con el PGS Catalog para obtener los datos de los modelos PRS.
- **Backend ETL («Node.js»)**:
  - Ejecuta la lógica del proceso ETL, realizando la transformación de los datos obtenidos y su almacenamiento en la base de datos.
- **PRS_bd («MySQL»)**:
  - Base de datos relacional que almacena los modelos procesados, evaluaciones y metadatos relacionados.
  - Accedida tanto por los módulos ETL como por el sistema de priorización.
- **Backend PRS system («Node.js»)**:
  - Proporciona servicios al frontend para consultar y priorizar modelos almacenados.
  - Gestiona la lógica de negocio y la interacción con la base de datos.
- **Frontend PRS system («react-next»)**:
  - Componente de interfaz de usuario desarrollado con React y Next.js.
  - Permite al usuario explorar, filtrar y visualizar los modelos PRS priorizados.

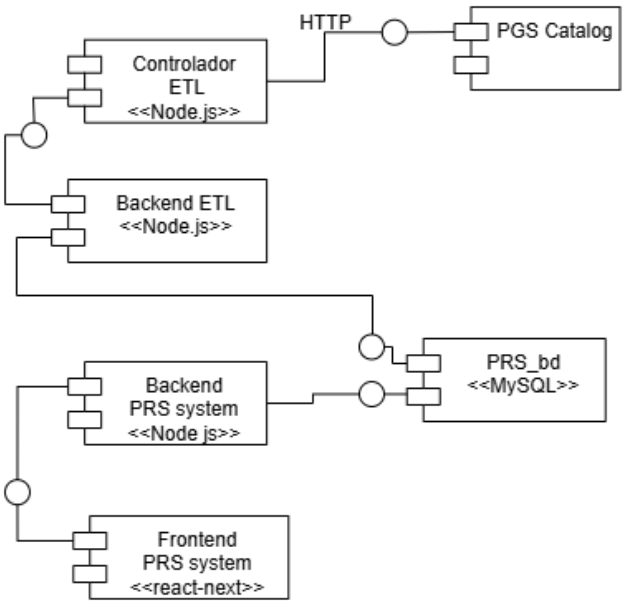


Figura 4.3: Diagrama de componentes del sistema PRS

.

4.2.3. Diagrama de paquetes

Proyecto ETL

El proyecto ETL forma parte central de la arquitectura del sistema y está implementado dentro del directorio `src/`. Este módulo se encarga de orquestar la extracción, transformación y carga de datos genómicos desde el repositorio `PGS Catalog`, con el propósito de almacenar modelos PRS estructurados en la base de datos interna del sistema.

La estructura del proyecto, representada en la Figura 4.4, sigue una organización modular orientada a servicios y componentes reutilizables. A continuación, se describen sus principales carpetas y archivos:

- **`app/`**: Contiene la estructura de la aplicación principal en Next.js, incluyendo la definición del layout, configuración global y punto de entrada de la interfaz.
  - `layout.tsx`, `page.tsx`, `providers.tsx`: Componentes raíz que definen la estructura y el contexto de la aplicación.
  - `globals.css`, `favicon.ico`: Recursos globales de estilo e íconos.
  - `api/`, `components/`, `hooks/`: Directorios preparados para organizar las rutas de API internas, componentes visuales y hooks reutilizables.

- `repositories/`: Incluye lógica de inserción de datos, como `insertData.ts`.
- `services/`: Contiene servicios clave del flujo ETL.
  - `etlService.ts`: Orquestador principal del proceso ETL.
  - `externalApi.ts`: Encargado de conectarse con fuentes externas como el PGS Catalog.
  - `transformer.ts`: Aplica transformaciones a los datos antes de ser almacenados.

- **`pages/api/etl/`**: Este directorio implementa los endpoints de la API REST que exponen y gestionan los recursos relacionados con el modelo PRS.
  - Incluye archivos como `traits.ts`, `traitCategories.ts`, `prsModels.ts`, entre otros, que permiten el acceso a los datos transformados.

- **`utils/`**: Contiene utilidades auxiliares usadas por diversos módulos del sistema.
  - `broadAncestryMapping.ts`, `prisma.ts`, `logging.ts`: Funciones de mapeo, conexión a la base de datos y manejo de logs.
  - `*.json`: Archivos de respuesta usados para pruebas o desarrollo.

Esta organización modular y orientada a servicios facilita el desarrollo, prueba y despliegue del flujo ETL, permitiendo que el sistema escale y se integre con nuevas fuentes de datos o lógicas de transformación sin comprometer la mantenibilidad del proyecto.

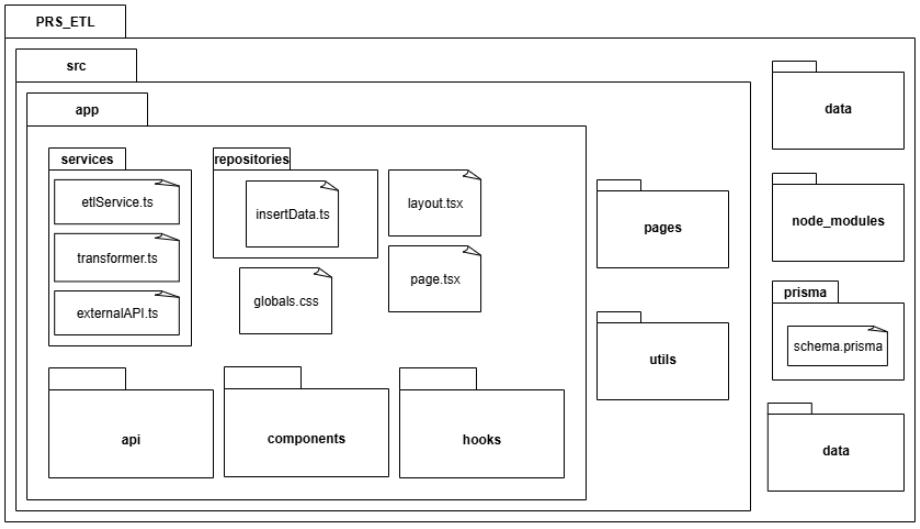


Figura 4.4: Estructura de paquetes del sistema

Proyecto PRS Models

El diagrama de paquetes presentado en la Figura 4.5 representa la estructura modular del sistema `mi-proyecto`, dividido en dos niveles principales: **backend** y **frontend**. Esta organización refleja una arquitectura clara orientada a responsabilidades separadas, facilitando el mantenimiento y escalabilidad del sistema.

- **Backend**: Este paquete contiene la lógica del servidor y está desarrollado en Node.js. Incluye los siguientes submódulos:
  - `package.json`: Define las dependencias y scripts del backend.
  - `prisma/schema.prisma`: Archivo que define el esquema de base de datos y su conexión mediante Prisma ORM.
  - `src/`:
    - `app.js`, `server.js`, `swaggerConfig.js`: Archivos principales de configuración, servidor y documentación.
    - `controllers/`: Módulo que contiene la lógica de control para los distintos endpoints.
    - `routes/`: Define las rutas de la API que conectan con los controladores.
- **Frontend**: El frontend está desarrollado con React y Next.js. Este paquete proporciona la interfaz de usuario que consume los servicios del backend.
  - `package.json`: Archivo de configuración del entorno de desarrollo del frontend.
  - `jsconfig.json`, `next.config.mjs`, `postcss.config.mjs`: Archivos de configuración específicos de Next.js y PostCSS.
  - `app/`: Contiene las páginas de la aplicación generadas por Next.js.
  - `components/`: Agrupa los componentes reutilizables y específicos de la interfaz.
    - `prioritization/`: Componentes relacionados con la tabla de priorización de modelos PRS, incluyendo funcionalidades de ordenamiento, filtros, paginación y reordenamiento por *drag-and-drop*.
    - `ui/`: Componentes visuales reutilizables como botones, tarjetas o inputs, utilizados en distintas partes del sistema.
  - `public/`: Recursos estáticos como imágenes y archivos accesibles directamente desde el navegador.

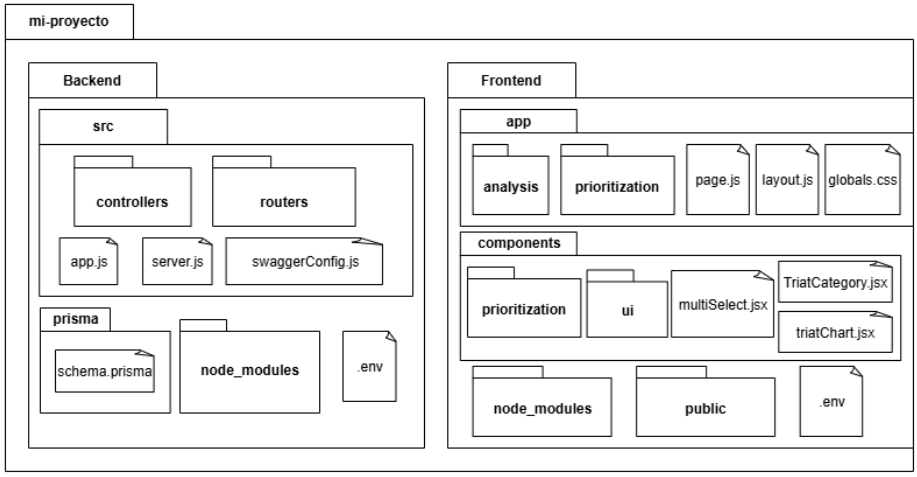


Figura 4.5: Estructura de paquetes del sistema

4.2.4. Diagrama de clases

El diagrama de clases que se presenta en la figura 4.6 modela la estructura de un sistema para el análisis de puntuaciones de riesgo poligénico y sus relaciones entre diferentes entidades.

1. **Paciente y análisis PRS**:
   - La clase **Patient** representa a los individuos analizados, los cuales tienen atributos como `patient_id`, `ancestry` y `PRS`.
   - Estos pacientes son analizados a través de instancias de la clase **PRS Analysis**, que describe el proceso de evaluación del paciente en función de las puntuaciones de riesgo.
   - Cada **PRS Analysis** utiliza un conjunto de datos llamado **Scoring Set**, que contiene información sobre variantes genéticas (`n_snp`) y el `assembly` genómico asociado.
2. **Modelos de PRS y fenotipos**:
   - El **Scoring Set** está vinculado a un **PRS Model**, que representa un modelo genómico específico con atributos como `name`, `intended_use` y `predicted_outcome`.
   - Los modelos PRS están asociados a uno o más **Phenotype**, que describen características genéticas o resultados esperados, definidos por atributos como `name` y `EFO_id`.

3. **Evaluación y métricas de desempeño**:
    - Los modelos PRS son evaluados a través de instancias de la clase **Evaluation**, que utiliza métricas de desempeño representadas por la clase **Performance Metric**.
    - Los resultados de las evaluaciones son almacenados en **Metric Evaluation**, con atributos como `result` y `description`, proporcionando un análisis detallado del desempeño del modelo.
4. **Distribución y datos de población**:
    - Las muestras de población son modeladas por la clase **Population Sample**, que contiene información detallada sobre los individuos analizados (`details`).
    - Estas muestras están vinculadas a la clase **Ancestry Distribution**, que describe la distribución de categorías ancestrales (`categories`) y sus porcentajes (`percentages`).
    - Además, **Population Sample** puede incluir datos de nivel individual (**Indiv-Lev Data**) o estadísticas de resumen (**GWAS SumStat**), que describen variantes genéticas y cohortes.
5. **Desarrollo y ajuste de modelos**:
    - La clase **Development** modela el proceso de desarrollo de los modelos PRS y tiene dos subclases principales: **Base** y **Tuning**.
    - **Base** representa modelos iniciales o básicos, mientras que **Tuning** incluye ajustes adicionales realizados en los modelos para optimizar su desempeño.

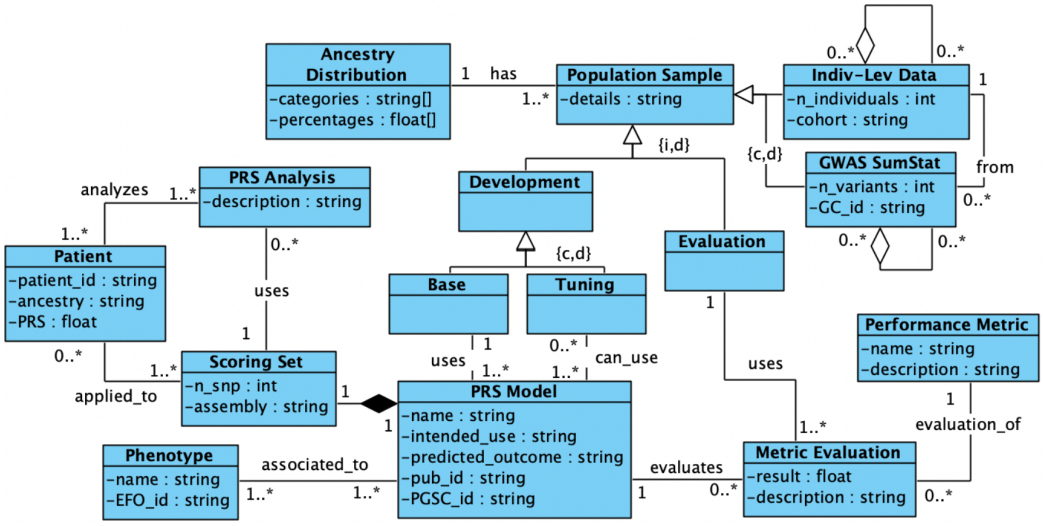


Figura 4.6: Diagrama de Clases [1].

4.3. Diseño de datos

El diseño de datos define la estructura y organización de la información necesaria para el funcionamiento del sistema, asegurando su integridad, consistencia y facilidad de acceso. Este apartado presenta el modelo relacional que sustenta la base de datos, así como el diccionario de datos, que documenta de manera detallada cada una de las tablas, atributos y relaciones empleadas. El objetivo es proporcionar una representación clara de cómo se almacenan, vinculan y gestionan los datos utilizados en la extracción, priorización de modelos PRS.

4.3.1. Modelo relacional

El modelo relacional mostrado en la Figura 4.7 representa la estructura lógica de la base de datos diseñada para gestionar la información relacionada con los modelos PRS, publicaciones, evaluaciones poblacionales, análisis de usuarios y su priorización. Este modelo sustenta la arquitectura del sistema y permite una integración eficiente de los distintos módulos.

1. **Modelos PRS y sus relaciones:**
   - La tabla `PRSModel` almacena la información principal de los modelos PRS, como el nombre, número de variantes y el identificador PGSC.
   - Cada modelo puede estar relacionado con múltiples `Trait`, mediante la tabla intermedia `PRSModelToTrait`.

- También se vinculan a publicaciones mediante el campo `publicationId`.
- Los modelos se relacionan con datos ancestrales a través de `BroadAncestryInModel`, indicando el porcentaje de validación por ancestría.

2. **Evaluación de modelos y muestras poblacionales:**

   - Las evaluaciones se almacenan en `ModelEvaluation`, que enlaza un modelo con una muestra evaluada.
   - Las muestras evaluadas se describen en `EvaluationPopulationSample`, incluyendo número de individuos, casos, controles, edad, género, ancestría, cohorte, y referencias a fuentes GWAS o DOI.
   - Por cada evaluación, se pueden almacenar métricas de desempeño (AUC, CI) en `PerformanceMetricEvaluation`, vinculadas a `PerformanceMetric`.

3. **Publicaciones y datos de referencia:**

   - La tabla `Publication` guarda la información bibliográfica de los estudios asociados a los modelos (PMID, DOI, año, autor).
   - Las tablas `ScoringFile` y `ReferencePopulation` almacenan archivos y metadatos usados para interpretar resultados.
   - También se registran las ancestrías por población de referencia en `BroadAncestryInRefPop`.

4. **Organización jerárquica de traits:**

   - Los rasgos (`Trait`) se agrupan en categorías mediante `TraitToTraitCategory`, y estas últimas están definidas en `TraitCategory`.
   - Esto permite organizar los modelos de forma semántica según el dominio biomédico al que pertenecen.

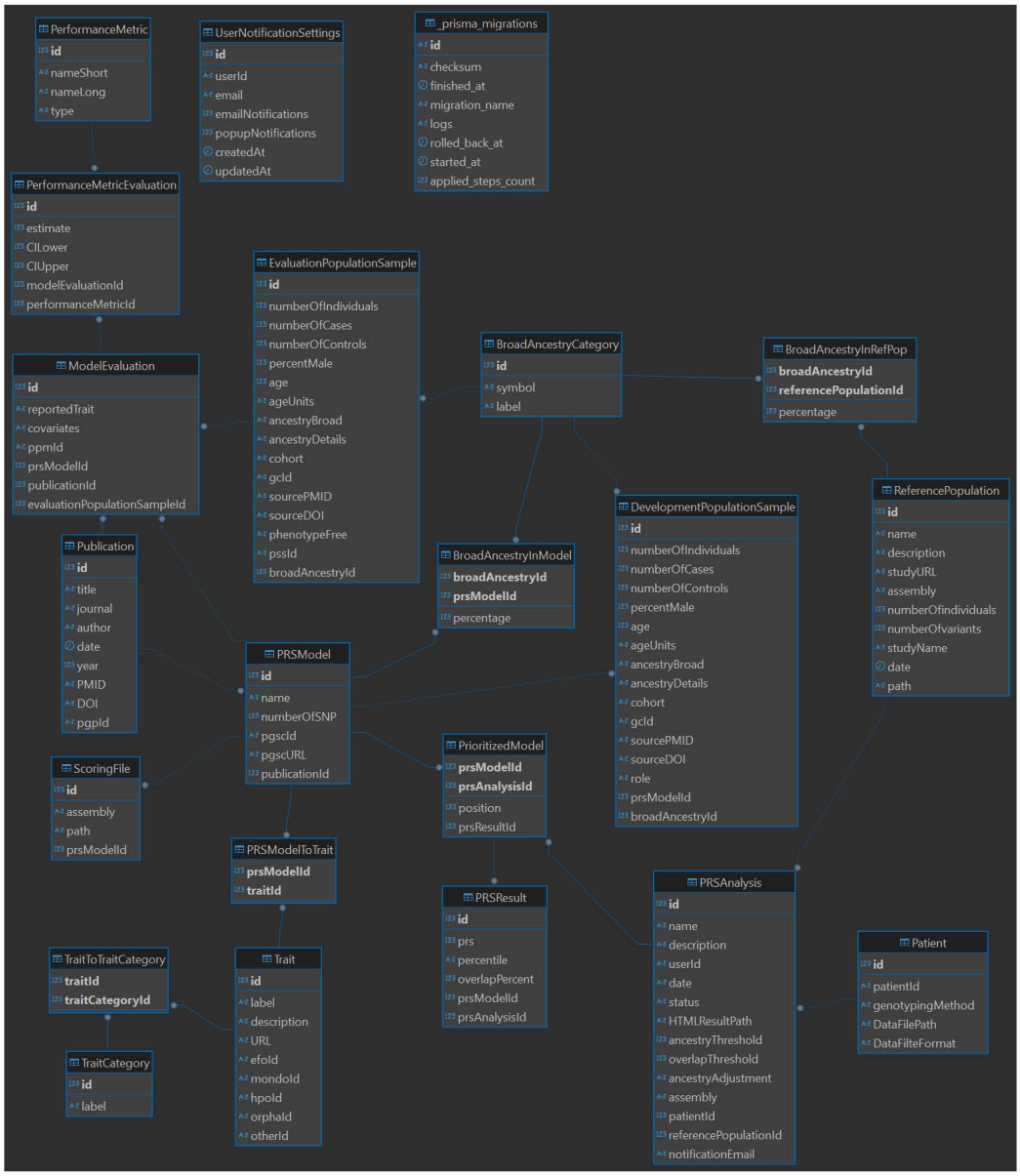


Figura 4.7: Modelo relacional de la base de datos del sistema

4.3.2. Diccionario de Datos

El diccionario de datos describe la estructura de las tablas que conforman la base de datos del sistema, especificando para cada una de ellas sus atributos, tipo de dato, significado y restricciones. Esta documentación permite comprender cómo se organiza y relaciona la información necesaria para la extracción, priorización y análisis de modelos PRS. Las tablas

detalladas del diccionario de datos se encuentran en el Anexo A.

4.4. Diseño de Interfaz

El diseño de la interfaz se enfocó en ofrecer una experiencia clara e intuitiva, guiando al usuario desde la selección inicial de datos hasta la priorización final de modelos PRS. Se incorporaron gráficos, tablas dinámicas y controles interactivos que permiten filtrar, ordenar y seleccionar información en tiempo real, facilitando un análisis ágil y reduciendo la posibilidad de errores.

4.4.1. Arquitectura de la Información

La arquitectura de la información del sistema se diseñó para organizar y estructurar las funcionalidades principales, asegurando que los usuarios puedan interactuar de manera intuitiva y eficiente con las herramientas disponibles. A continuación, se presenta la estructura de categorías y jerarquías del sistema:

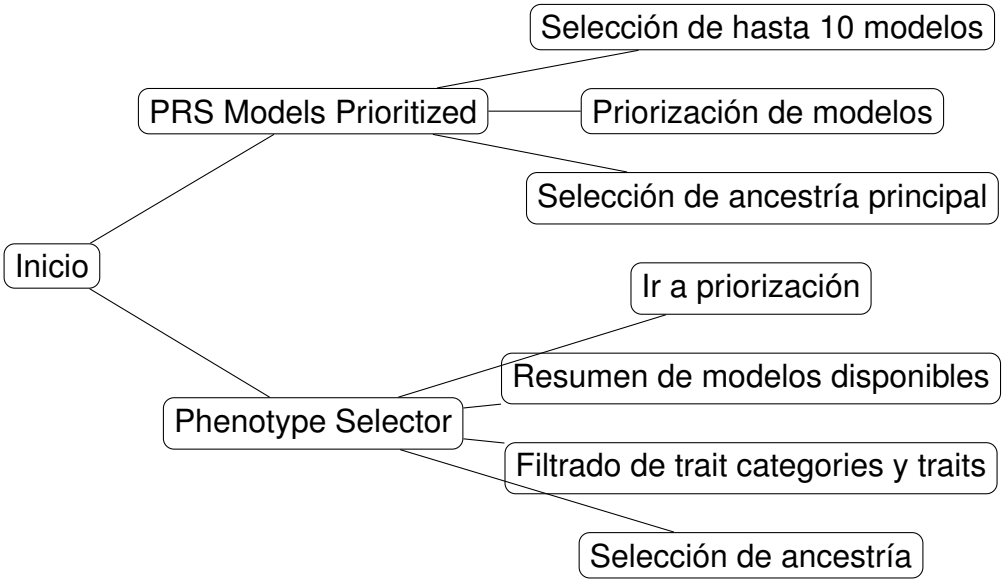
1. Categorías principales

El sistema está dividido en las siguientes categorías principales:

- **Phenotype Selector**: Página de entrada que permite el acceso directo a las funcionalidades clave del sistema. Aquí se encuentra el componente `Phenotype Selector`, donde el usuario puede seleccionar una ancestría de interés desde un menú interactivo. Esta selección filtra dinámicamente las categorías de rasgos (`trait categories`) y sus respectivos rasgos (`traits`), los cuales pueden ser seleccionados para su análisis. En la parte inferior se muestra el total de modelos PRS disponibles para la combinación seleccionada. Al hacer clic en el botón `Go to Prioritization`, el usuario es dirigido a la siguiente etapa.
- **Prioritization Module**: Página donde el usuario debe seleccionar una ancestría principal que se aplicará a los modelos o rasgos elegidos previamente. Una vez realizada esta selección, el sistema ejecuta el proceso de priorización, mostrando una tabla ordenada según un puntaje calculado. En esta tabla se pueden seleccionar hasta un máximo de 10 modelos, los cuales serán utilizados en la etapa de análisis.

2. Jerarquías

La estructura jerárquica del sistema sigue el siguiente esquema:



4.4.2. Prototipos de Interfaces Gráficas

1. Pantalla: Phenotype Selector

La pantalla `Phenotype Selector`, como se observa en la Figura 4.8, permite al usuario explorar y seleccionar rasgos (traits) asociados a categorías específicas de acuerdo con una ancestría determinada. Esta interfaz sirve como paso inicial en el flujo de priorización.

- **Selección de ancestría**: A través de un menú desplegable, el usuario puede elegir una o mas ancestrías o dejar activa la opción `All` para ver todos los modelos disponibles.
- **Visualización por categoría**: A la izquierda se muestra un gráfico tipo *donut* que indica la cantidad de modelos por ancestría, y una lista de categorías de rasgos (`Trait Category`) con su respectivo número de modelos disponibles.
- **Selección de traits**: Al seleccionar una categoría, se habilita una lista de traits relacionados que pueden ser seleccionados de manera individual. Los traits seleccionados se resumen en la parte inferior junto con el total de modelos disponibles para esas combinaciones.
- **Acción**: Al presionar el botón `Go to Model Prioritization`, el sistema dirige al usuario a la siguiente etapa, donde se realiza la priorización.

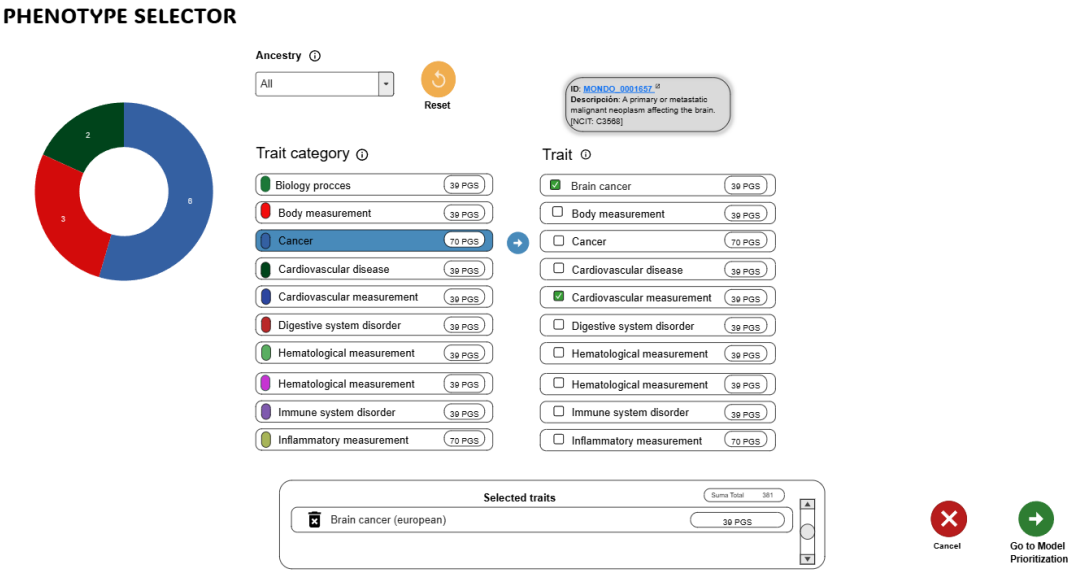


Figura 4.8: Mockup de la pantalla `Phenotype Selector`

2. Pantalla: Prioritization Module

La pantalla `Prioritization Module`, ilustrada en la Figura 4.9, permite ejecutar el proceso de priorización sobre los modelos seleccionados previamente.

- **Selección de ancestría**: El usuario debe elegir una ancestría principal sobre la cual se aplicará el proceso de priorización. Esta selección afecta directamente el puntaje final de cada modelo.
- **Tabla de modelos**: Se muestra una tabla comparativa con los modelos filtrados, detallando información clave como:
  - `Model ID` y `Trait`.
  - Composición de ancestría (representada gráficamente).
  - Número de SNPs y detalles de muestras de desarrollo.
  - Métricas de evaluación: asociación de riesgo, poder discriminatorio, año de publicación y enlace a PubMed.
- **Selección y análisis**: El usuario puede seleccionar hasta un máximo de 10 modelos para su análisis. Una vez hecha la selección, se debe presionar el botón `Run Analysis` para continuar con la siguiente etapa.

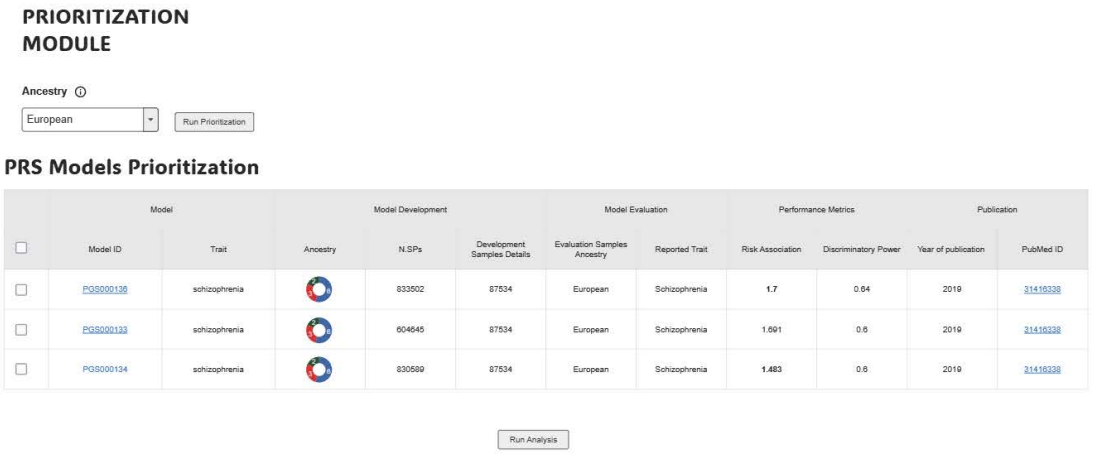


Figura 4.9: Mockup de la pantalla `Prioritization Module`

Capítulo 5

Implementación

En este capítulo se detalla el proceso de implementación de la solución propuesta, abarcando desde la preparación del entorno de trabajo hasta la construcción de los distintos módulos que conforman el sistema. Se describe la selección de tecnologías, herramientas y lenguajes de programación empleados, así como la estrategia de desarrollo aplicada para integrar de forma coherente los componentes de extracción, priorización y visualización de modelos PRS. Además, se presentan las interfaces de usuario diseñadas para facilitar la interacción y el análisis de la información procesada.

5.1. Software utilizado

En el desarrollo de este proyecto se utilizaron tecnologías de software modernas y de código abierto, seleccionadas por su eficiencia, escalabilidad y compatibilidad para el tratamiento de datos genómicos y el desarrollo de interfaces interactivas. A continuación, se describen y justifican las principales herramientas utilizadas:

MySQL

MySQL[20] es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto. Se utilizó para almacenar de manera estructurada los modelos PRS, sus evaluaciones y metadatos asociados. MySQL fue elegido por su robustez, rendimiento en consultas complejas y compatibilidad con Prisma y Node.js, lo cual facilita su integración en entornos full stack modernos.

Node.js

Node.js [21] es un entorno de ejecución basado en JavaScript que permite desarrollar aplicaciones del lado del servidor. En este proyecto se utiliza como motor del backend,

gestionando las rutas de la API, la lógica de negocio y la conexión con la base de datos. Su modelo asincrónico y orientado a eventos lo hace ideal para manejar múltiples solicitudes concurrentes de forma eficiente.

Prisma

Prisma [22] es un ORM (Object-Relational Mapping) para Node.js y TypeScript que facilita la interacción con bases de datos SQL mediante un lenguaje declarativo. Prisma se utilizó para definir el esquema de la base de datos, generar migraciones y realizar consultas tipadas de forma segura. Su integración con TypeScript mejora la productividad y reduce errores en tiempo de desarrollo.

Next.js

Next.js [23] es un framework basado en React que permite la creación de aplicaciones web con renderizado híbrido (SSR y CSR). Se empleó en el frontend del proyecto para estructurar las páginas, manejar rutas y mejorar el rendimiento de carga. Su soporte integrado para API routes también facilita la extensión de funcionalidades del lado del servidor si se requiere.

React.js

React [24] es una biblioteca de JavaScript para la construcción de interfaces de usuario. En este sistema se utilizó para implementar componentes interactivos, como el Phenotype Selector y el Prioritization Module, que permiten al usuario seleccionar rasgos, aplicar filtros y visualizar modelos PRS de manera dinámica. React fue seleccionado por su enfoque modular basado en componentes reutilizables, lo que mejora la mantenibilidad del código y la experiencia del usuario.

Swagger.js

Swagger.js [25] es un conjunto de herramientas de código abierto que facilita la documentación y prueba de APIs. En este proyecto se empleó para generar y mantener la documentación interactiva de los endpoints desarrollados en el backend, permitiendo a los usuarios y desarrolladores explorar las rutas disponibles, sus parámetros y respuestas esperadas. Su uso mejora la comunicación entre el equipo de desarrollo y facilita la integración con otros sistemas.

Visual Studio Code

Visual Studio Code (VS Code) [26] es un editor de código multiplataforma y gratuito desarrollado por Microsoft. Se utilizó como entorno principal de desarrollo debido a su ligereza, amplia comunidad de extensiones y soporte para múltiples lenguajes y frameworks. En este proyecto, su integración con herramientas como ESLint, Prettier y el depurador integrado permitió optimizar el flujo de trabajo, garantizando un código más limpio y fácil de mantener.

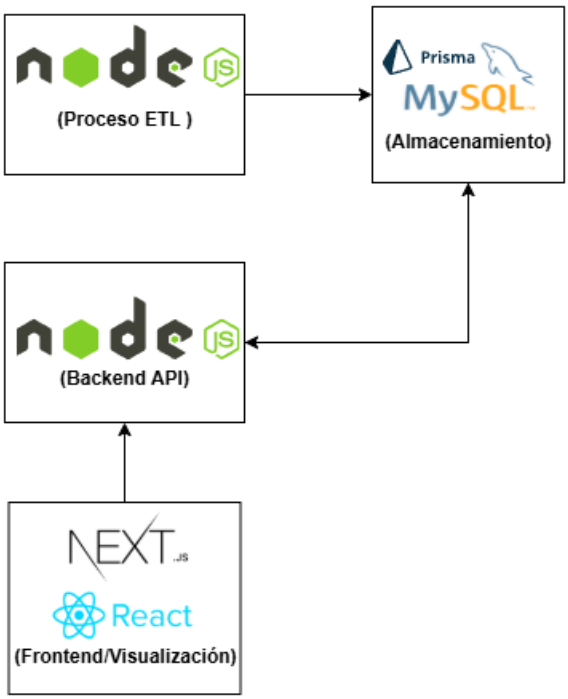


Figura 5.1: Vista de alto nivel de las tecnologías utilizadas en el sistema

5.2. Hardware utilizado

El hardware utilizado para el desarrollo será mi computador personal, con un procesador **AMD Ryzen 7 4000 series**, 8 núcleos y 8 procesadores lógicos, con un sistema de almacenamiento SSD y memoria RAM adecuada para manejar los entornos de desarrollo. Desde este equipo se desarrollarán los scripts de extracción, transformación, mejora y priorización de los modelos PRS que serán obtenidos desde el repositorio **PGS Catalog**, empleando herramientas como **Node.js** y bases de datos gestionadas mediante **MySQL** y **Prisma**. Asimismo, en este mismo equipo se implementará la aplicación web para el

análisis de los modelos PRS, desarrollando tanto el backend (con Node.js) como la interfaz de usuario (con React.js y Next.js)

5.3. Lenguajes de programación

En el desarrollo del sistema se utilizaron varios lenguajes de programación, cada uno con un propósito específico en la arquitectura general de la aplicación. A continuación, se detallan los principales lenguajes utilizados y la justificación de su elección.

TypeScript

TypeScript [27] es un lenguaje de programación de alto nivel, gratuito y de código abierto, que se basa en JavaScript e introduce tipado estático opcional. Fue utilizado tanto en el backend como en el frontend del proyecto, facilitando el desarrollo robusto con mayor control de tipos y mejor mantenibilidad del código.

- **Escalabilidad:** TypeScript es ideal para proyectos grandes y escalables, ya que permite organizar mejor el código mediante clases, interfaces y módulos, lo que facilita el mantenimiento a largo plazo.
- **Integración con Frameworks:** Ofrece excelente soporte para frameworks modernos como `React` y `Next.js`, mejorando la integración, la experiencia de desarrollo y reduciendo errores en tiempo de compilación.

HTML

HTML (HyperText Markup Language) [28] es el lenguaje estándar de marcado utilizado para estructurar contenido en la web. En este proyecto, se emplea en combinación con React para definir la estructura básica de la interfaz de usuario.

- **Estructuración del contenido:** HTML permite definir encabezados, párrafos, listas, formularios, tablas y otros elementos fundamentales para la presentación y disposición del contenido.
- **Compatibilidad con tecnologías web:** Se integra perfectamente con CSS y JavaScript/TypeScript, permitiendo el desarrollo de interfaces interactivas y responsivas.

JavaScript

JavaScript [29] es el lenguaje base sobre el cual se construye TypeScript. Aunque en este proyecto se utilizó principalmente TypeScript, JavaScript sigue siendo fundamental, ya que gran parte del ecosistema de librerías y frameworks, así como funcionalidades del navegador, están escritas en este lenguaje.

- **Interactividad en el frontend:** Permite manejar eventos, actualizar dinámicamente el DOM y crear componentes reactivos.
- **Compatibilidad total:** Todo el código TypeScript es transpilado a JavaScript para su ejecución en el navegador o en Node.js.

SQL

SQL (Structured Query Language) [30] es el lenguaje estándar utilizado para gestionar y manipular bases de datos relacionales. En este proyecto se utiliza a través de Prisma ORM para definir esquemas, realizar migraciones y ejecutar consultas sobre la base de datos MySQL.

- **Consultas estructuradas:** Permite recuperar y modificar datos con precisión mediante operaciones como `SELECT`, `INSERT`, `UPDATE` y `DELETE`.
- **Integración con ORM:** Prisma traduce operaciones definidas en TypeScript a consultas SQL seguras y eficientes.

5.4. Estrategia de implementación

La estrategia de implementación del sistema se estructuró en tres etapas fundamentales: la etapa de extracción de datos, la etapa de priorización de modelos PRS y el desarrollo de la aplicación como tal. Esta división permite abordar el desarrollo de forma progresiva y modular, asegurando la integración fluida entre componentes. La Figura 5.2 resume visualmente este flujo.

1° Desarrollo de la etapa de extracción

Esta fase inicial se enfocó en sentar las bases para el procesamiento de los datos genómicos provenientes del repositorio PGS Catalog.

- **Recopilación de información**: Se investigó sobre análisis de riesgo poligénico y la estructura de los modelos disponibles en el repositorio PGS Catalog.

- **Desarrollo de la API de extracción**: Se implementó un sistema para extraer, transformar y cargar los modelos PRS en la base de datos, validando su estructura y calidad según el modelo conceptual diseñado.
- **Validación**: Se realizó un control de calidad sobre los datos cargados para asegurar consistencia con el modelo de dominio.

2° Desarrollo de la etapa de priorización

En esta fase se centró en establecer el criterio de evaluación y construcción del sistema de priorización.

- **Creación del endpoint de priorización**: Se implementó la lógica necesaria en el backend para recibir los traits seleccionados y calcular un puntaje de prioridad para los modelos PRS.

3° Desarrollo de la aplicación

Esta última etapa integró todos los elementos desarrollados anteriormente en una aplicación interactiva y funcional.

- **Backend**: Se desarrolló una API REST para gestionar las operaciones de priorización y comunicación con la base de datos. Para facilitar la comunicación entre módulos sin acoplamiento directo, se implementó el patrón de diseño **Mediator** [31], el cual centraliza y coordina la interacción entre componentes.
- **Frontend**: Se desarrolló una interfaz moderna utilizando React.js y Next.js. Se aplicó el patrón de diseño **Container-Presenter** [32], que permite separar la lógica de negocio (contenedor) de la lógica de presentación (componente presentador), facilitando la mantenibilidad y reutilización de componentes visuales.

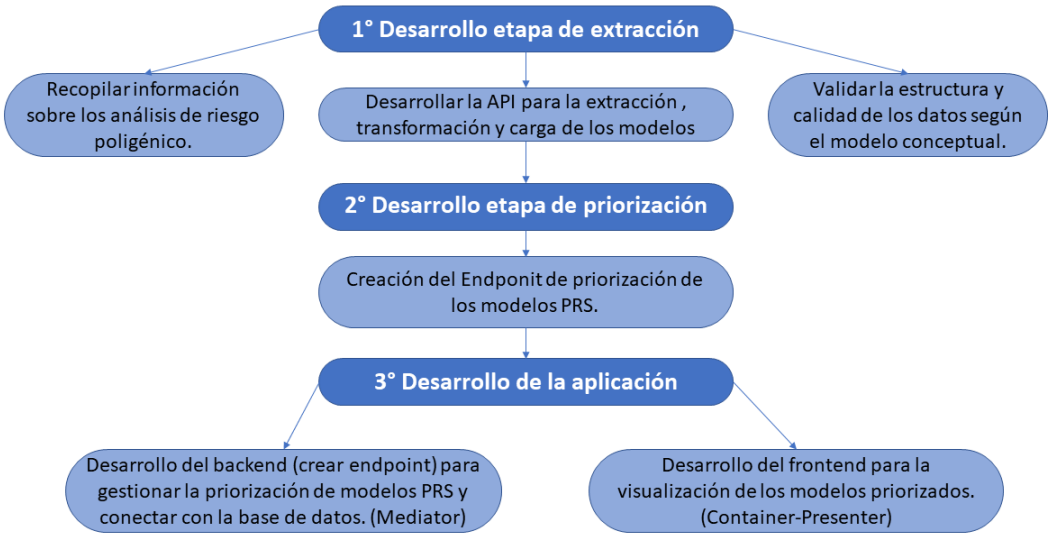


Figura 5.2: Estrategia de implementación del sistema

5.5. Interfaces de Usuario

Las interfaces desarrolladas para el sistema permiten una interacción fluida, guiando al usuario desde la selección de características genéticas hasta la priorización de modelos PRS.

Página de selección de características fenotípicas

La Figura 5.3 muestra la interfaz de selección de características en la etapa inicial del análisis. El usuario puede:

- Seleccionar una o más ancestrías disponibles.
- Visualizar la distribución de categorías de características (*trait categories*) mediante un gráfico circular.
- Filtrar y seleccionar características específicas (*traits*) dentro de cada categoría.
- Visualizar una descripción detallada de cada característica.

- Consultar el número total de modelos asociados a las características seleccionadas.
- Avanzar al proceso de priorización de modelos presionando `Go to Prioritization`.

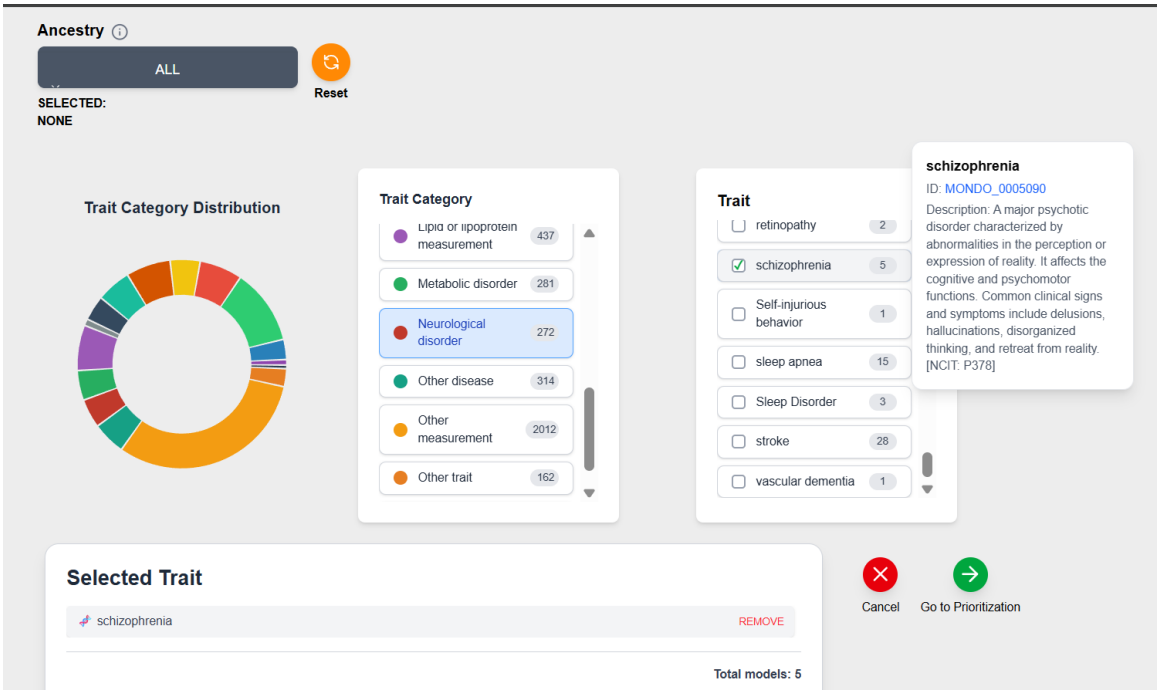


Figura 5.3: Interfaz para selección de características fenotípicas.

Página de priorización de modelos PRS

En la Figura 5.4 se presenta la interfaz donde se ejecuta el proceso de priorización. El usuario debe:

- Elegir una ancestría principal que se usará como referencia para calcular la afinidad de los modelos.
- Presionar el botón `Run Prioritization` para ejecutar el algoritmo.
- Observar los modelos ordenados según la puntuación de priorización, evaluando características como:
  - Asociación de riesgo (`OR`).

    - Poder discriminatorio (`AUROC/C-index`).
    - Año de publicación y enlace al artículo original (`PMID`).

- Reordenar manualmente los modelos en base a las columnas que permiten ordenamiento, como `Trait`, `Risk Association (OR)`, `Discriminatory Power (AUROC/C-index)` o `Year`.
- Seleccionar hasta 10 modelos para su posterior análisis.
- Iniciar el análisis con el botón `Run Analysis`.

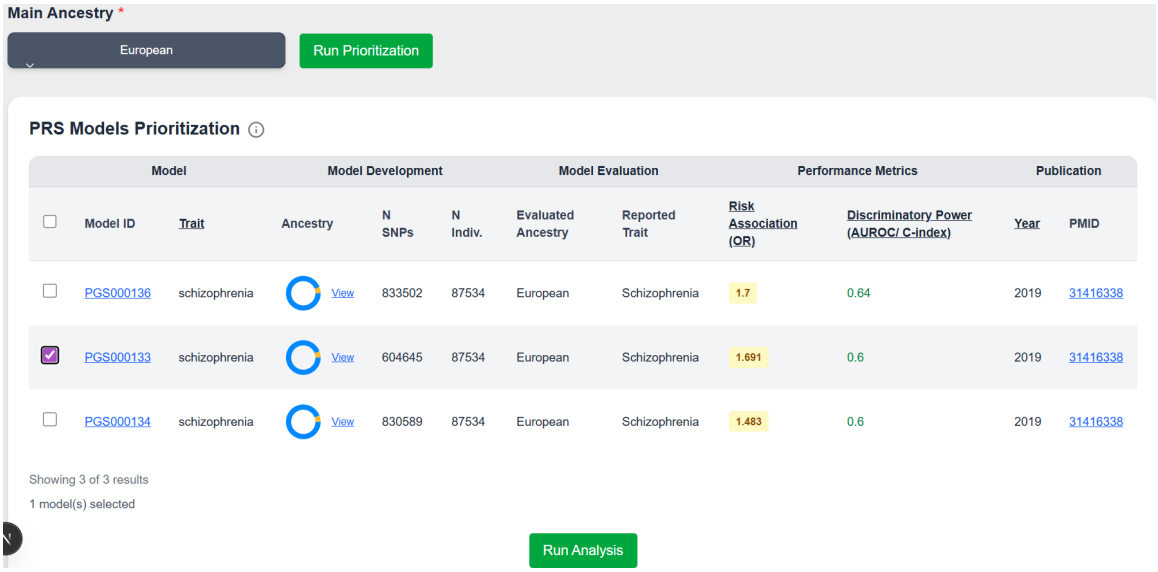


Figura 5.4: Interfaz para la priorización de modelos PRS.

Capítulo 6

Pruebas

Las pruebas realizadas tuvieron como objetivo validar el correcto funcionamiento y la estabilidad de todos los componentes del sistema desarrollado, abarcando tanto el módulo ETL como la aplicación web. Se diseñaron y ejecutaron casos de prueba que evaluaron la precisión de los datos extraídos, transformados e insertados desde el PGS Catalog hacia la base de datos, así como la interacción, visualización y manipulación de esta información en la interfaz de usuario.

6.1. Contexto de pruebas

Para validar la eficacia y robustez del sistema ETL desarrollado, se realizaron diversas pruebas unitarias orientadas a comprobar la correcta extracción, transformación e inserción de datos en la base de datos, así como la integridad de las relaciones entre entidades.

El enfoque principal está en evaluar el proceso ETL (Extract, Transform, Load), el cual permite integrar datos desde una fuente externa (PGS Catalog) hacia una base de datos propia, a través de una serie de transformaciones estructurales.

6.1.1. Características

El proceso ETL fue construido con las siguientes tecnologías:

1. **Next.js [23] (TypeScript)**: Framework utilizado para desarrollar tanto la interfaz como el backend del proyecto. Se basa en React y permite funcionalidades como SSR (Server Side Rendering) y generación de rutas dinámicas.

2. **Prisma ORM [22]**: Herramienta de mapeo objeto-relacional que permite la interacción con la base de datos de forma sencilla, eficiente y segura. Facilita las consultas complejas y mantiene la integridad del esquema.

3. **MySQL** [20]: Motor de base de datos relacional donde se almacenan los datos procesados. La estructura de la base fue generada a partir del modelo conceptual definido previamente.

6.1.2. Estructura del ETL

La estructura del proceso ETL se compone de varios servicios organizados en una arquitectura modular, que permite escalar o reutilizar partes del flujo según la fuente de datos. El proceso puede iniciarse mediante una petición HTTP desde herramientas como el navegador, `curl` o Postman.

A continuación se describe el flujo general:

- **EndPoint**: Es el punto de entrada del sistema. Recibe la solicitud para iniciar el proceso ETL.
- **etlService**: Servicio central encargado de coordinar el flujo. Invoca los siguientes subcomponentes:
  - **externalAPI**: Extrae los datos desde la API externa (PGS Catalog).
  - **transformer**: Realiza la transformación de los datos recibidos en formato JSON a una estructura más manejable y alineada con el modelo de la base de datos.
  - **insertData**: Inserta los datos transformados en la base de datos MySQL.

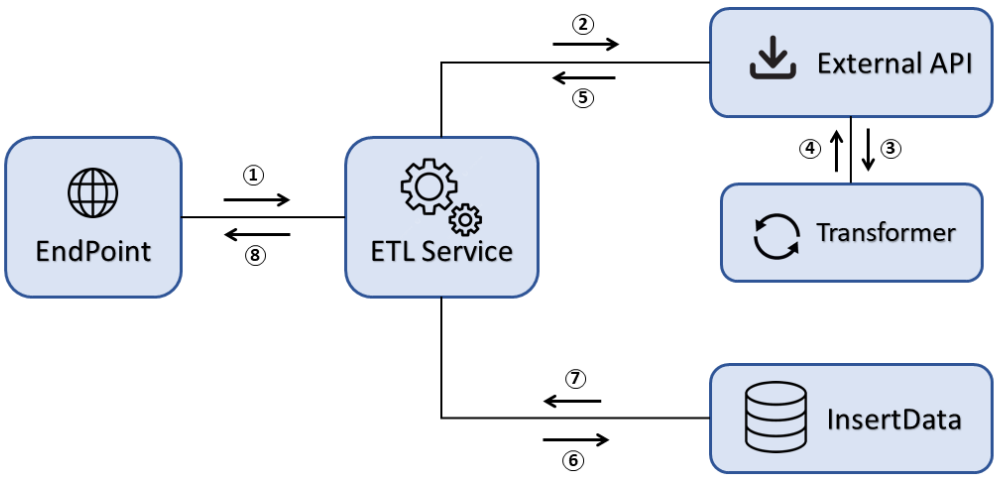


Figura 6.1: Estructura proceso ETL.

Este diseño modular permite replicar el proceso ETL para distintas fuentes de datos, manteniendo una estructura clara y reutilizable.

Los ETL desarrollados y las clases del modelo conceptual afectadas:

ETL	**Endpoint PGS Catalog**	**Clases del modelo conceptual**
NEXT_PUBLIC_API_URL/etl/traits	/trait/all	Traits
NEXT_PUBLIC_API_URL/etl/traitCategories	/trait_category/all	Trait Category
NEXT_PUBLIC_API_URL/etl/publications	/publication/all	Publication
NEXT_PUBLIC_API_URL/etl/prsModels	/score/all	PRSModel
NEXT_PUBLIC_API_URL/etl/prsModelTraits	/score/all	PRSModel, Trait
NEXT_PUBLIC_API_URL/etl/broadAncestryCategories	/ancestry_categories/	BroadAncestryCategory
NEXT_PUBLIC_API_URL/etl/developmentSamples	/score/all	PopulationSample_Development
NEXT_PUBLIC_API_URL/etl/modelEvauations	/performance/all	PopulationSample_Evaluation, PerformanceMetricEvaluation, PopulationSample_Evaluation
NEXT_PUBLIC_API_URL/etl/BroadancestryInMdoel	/broadancestryInMdoel/all	BroadancestryInMdoel

Tabla 6.1: Relación entre endpoints ETL, endpoints del catálogo PGS y clases del modelo conceptual

6.2. Pruebas Unitarias

Las pruebas unitarias se diseñaron para verificar el funcionamiento correcto y aislado de las entidades y relaciones que componen el modelo de datos del sistema. Cada prueba evaluó la integridad de la información, la consistencia con las restricciones del esquema y la correcta inserción y vinculación de datos extraídos desde el PGS Catalog.

El detalle completo de estas pruebas, junto con sus resultados por ciclo de ejecución, se encuentra documentado en el Anexo B.

6.3. Pruebas de Integración

Con el objetivo de verificar la correcta comunicación y funcionamiento conjunto de los distintos componentes del sistema, se llevaron a cabo pruebas de integración en múltiples ciclos de desarrollo. Estas pruebas se realizaron evaluando la interacción entre la base de datos, el backend y el frontend, con el fin de detectar y corregir errores de conexión, mapeo de datos o renderizado en la interfaz.

Para cada ciclo, se ejecutaron casos de prueba predefinidos que cubrían escenarios de consulta, transformación y visualización de información, documentando las observaciones y mejoras implementadas. Los resultados obtenidos permitieron validar que los datos fluyen correctamente desde la base de datos hasta la interfaz de usuario, asegurando la coherencia y estabilidad del sistema antes de las pruebas de sistema.

6.3.1. Pruebas de Integración entre Base de Datos y Backend

Las siguientes pruebas validan que los controladores implementados en Node.js utilizando Prisma permiten una correcta interacción con la base de datos MySQL. Se presentan observaciones obtenidas a lo largo de tres ciclos de desarrollo y prueba.

Endpoint	**Objetivo**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
`GET /api/traits`	Obtener lista de traits junto a la cantidad de modelos PRS asociados	La consulta retornaba datos incompletos porque no se había incluido la relación con la tabla intermedia.	Se agregó correctamente la relación con `prsModels`. Se comenzó a visualizar el número de modelos.	Verificado que todos los traits se recuperan correctamente con su campo `pgss`.
`GET /api/trait Categories`	Obtener categorías de traits con el total de modelos PRS asociados a los traits de cada categoría	Error inicial en el mapeo de relaciones entre categorías, traits y modelos.	Se corrigió el acceso a `trait.prsModels`. El conteo se generó correctamente en el backend.	Se confirmó la precisión del conteo de modelos por categoría. Se mantuvo estable.
`GET /api/ancestries`	Obtener todas las categorías de ancestría sin relaciones adicionales	Consulta ejecutada correctamente desde el inicio. Sin errores.	Sin cambios necesarios. Datos retornados correctamente.	Validación completada. Se usa directamente en el frontend.

Tabla 6.2: Pruebas de integración entre Backend y Base de Datos

6.3.2. Pruebas de Integración entre Backend y Frontend

Estas pruebas validan que los componentes del frontend consumen correctamente los datos proporcionados por el backend y que los resultados se renderizan adecuadamente en la interfaz. A continuación, se detallan los objetivos y resultados obtenidos en los distintos ciclos de validación.

Componente Frontend	**Objetivo**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
`TraitCategory.jsx`	Renderizar dinámicamente las categorías de traits y mostrar la cantidad de modelos PRS asociados	El fetch fallaba por error de CORS y estructura incorrecta del JSON.	Se configuró correctamente CORS y se adaptó el mapeo de datos. Los datos se recibían pero no se renderizaban del todo.	Los datos se mostraban correctamente en el frontend, con su conteo de modelos. Interacción establecida.
`BroadAncestry Selector.jsx`	Cargar dinámicamente las opciones de ancestría desde la base de datos y permitir multiselección	El componente no obtenía datos por un error en la URL del endpoint.	Se corrigió la URL y la función 'fetch', comenzando a mostrar opciones sin orden.	Opciones renderizadas ordenadamente, con capacidad de selección múltiple. Validación completa.
`Trait.jsx`	Mostrar en una tabla todos los traits con su número de modelos PRS asociados, de forma paginada y ordenada	El frontend no mostraba datos. La respuesta JSON era vacía por errores de consulta.	Se depuró la consulta en el backend y se ajustó el campo `pgss`. Los datos comenzaron a mostrarse parcialmente.	Se visualizó correctamente la tabla de traits y se habilitó paginación. El componente funcionaba como se esperaba.

Tabla 6.3: Pruebas de integración entre Backend y Frontend

6.4. Pruebas de Sistema

Con el fin de asegurar la correcta operación e integración de los distintos componentes del sistema desarrollado, se realizaron pruebas de sistema orientadas a evaluar tanto el backend como el frontend. Estas pruebas permitieron verificar el cumplimiento de los requisitos funcionales, la coherencia en la visualización de datos y la interacción dinámica entre los elementos de la interfaz. A continuación, se presentan los principales escenarios evaluados, junto con sus respectivos objetivos y resultados obtenidos.

Escenario	**Objetivo**	**Resultado Obtenido**
Selección de ancestrías	Validar que el usuario pueda seleccionar hasta 4 categorías de ancestría (o ninguna), y que esta selección actualice dinámicamente los datos mostrados	El componente de selección de ancestrías funcionó correctamente. Al seleccionar ancestrías, se filtraron las categorías de traits asociadas. Si no se seleccionó ninguna, se mostraron todos los datos disponibles.
Visualización de todas las categorías de traits y sus traits	Verificar que el sistema muestre correctamente todas las categorías de traits y los traits correspondientes, incluso sin selección previa de ancestrías	El sistema mostró correctamente todas las categorías y los traits asociados, incluso sin seleccionar ancestrías.
Selección de traits para análisis	Validar que el usuario pueda seleccionar traits específicos para avanzar en el análisis personalizado	El usuario pudo seleccionar traits específicos desde la interfaz para continuar con el análisis posterior.
Filtrado de categorías de traits	Comprobar que, en base a la selección de ancestrías, se actualicen correctamente las categorías de traits disponibles en el sistema	Las categorías de traits se actualizaron dinámicamente según la ancestría seleccionada. El usuario pudo ver el número de modelos PRS por categoría y acceder a los traits asociados.
Visualización de gráfico circular de categorías	Confirmar que se muestre un gráfico circular representando el total de traits por categoría de forma clara y precisa	El gráfico circular se visualizó correctamente, diferenciando cada categoría con su respectiva proporción de traits.
Despliegue de información en pop-up de traits	Verificar que cada trait permita acceder a una descripción emergente con sus ontologías asociadas, y que los enlaces dirijan a información externa	Cada trait mostró un pop-up con su descripción y las ontologías relacionadas. El enlace a la ontología redirige correctamente al sitio oficial.

Tabla 6.4: Pruebas de sistema

6.5. Pruebas con Usuarios

Se realizaron pruebas funcionales y de experiencia de usuario con personas que, si bien no son consultores genómicos, cuentan con un perfil cercano al dominio, ya que poseen conocimientos sobre qué son los modelos PRS y cómo se aplican en distintos contextos. Este nivel de familiaridad permitió evaluar la funcionalidad del sistema y su usabilidad desde la perspectiva de usuarios con un conocimiento intermedio en la materia, pero sin una especialización clínica formal.

Durante el proceso de evaluación, se identificaron diversos errores y oportunidades de mejora, principalmente relacionados con la interfaz de usuario y la presentación de la información. La Tabla **??** contiene los hallazgos más relevantes.

Componente	**Problema detectado**	**Solución aplicada**
Trait Category (gráfico)	No se entendía fácilmente su función	Se añadió texto explicativo y ajustó tamaño visual
Trait Category (etiqueta)	Solo mostraba nombres cortos	Se configuró para mostrar el nombre completo
Trait (checkbox)	Los cuadrados eran de distinto tamaño	Se estandarizó el tamaño de todos los selectores
Ancestry (selector)	El selector se desalineaba al elegir más de 2 ancestrías	Se corrigió el comportamiento del selector para mantener posición fija
Pop-up en Ancestry	Falta de instrucción sobre cómo seleccionar	Se agregó la información correspondiente de cómo seleccionar las ancestrías
Selected Trait	El recuadro estaba a la derecha y causaba molestias visuales	Se movió a la izquierda; botones `Cancel` y `Go to` a la derecha
Trait Category (etiqueta de nros de modelos)	Mostraba número de fenotipos en vez del total de modelos que tenía ese trait category	Se ajustó para mostrar cantidad total de modelos que contiene un trait category
Prioritization (sin ancestría)	Si no se elegía ancestría antes, no se mostraban resultados	Se modificó para mostrar todas las ancestrías por defecto

Tabla 6.5 – continuación

Componente	Problema detectado	Solución aplicada
Botón info en Prioritization	Faltaba explicación sobre el componente	Se añadió botón informativo con ayuda contextual
Main Ancestry	No estaba claramente marcado como obligatorio	Se agregó un asterisco $\star$ para indicar campo requerido
Reset order	Ubicación poco visible del botón	Se reubicó arriba a la derecha con ícono y texto aclaratorio
Run Analysis	Botón no se desactivaba si no había selección o si excedía el límite	Se corrigió para permanecer inactivo si no hay modelos seleccionados o hay más de 10 modelos seleccionados
Tabla (desplazamiento)	No se adaptaba bien horizontalmente	Se permitió desplazamiento horizontal de la tabla
Filtrado en tabla	Faltaba encabezado explicativo	Se añadió fila superior acorde al diseño de *ProtoPrio*
Columnas clave	No se destacaban visualmente	Se subrayaron columnas importantes para filtrar
Tabla de Prioritization	Los valores de AUROC y OR eran incorrectos o estaban vacíos	Se corrigió el proceso de extracción desde el backend para asegurar que se muestren los valores reales y válidos

Tabla 6.5: Pruebas de Usuario.

6.6. Pruebas de Aceptación

Durante las pruebas de aceptación, se trabajó con dos usuarios que, si bien no son expertos o consultores genéticos, cuentan con un perfil cercano al dominio, ya que poseen conocimientos sobre qué son los modelos PRS y cómo se aplican. Esto permitió evaluar

la funcionalidad del sistema desde la perspectiva de usuarios con un nivel intermedio de familiaridad con el tema, asegurando que la interfaz y el flujo de trabajo fueran comprensibles para un público más amplio.

Las pruebas se enfocaron en validar el correcto funcionamiento del sistema en distintos escenarios de uso, definiendo casos de prueba específicos para evaluar el algoritmo de priorización, el manejo de datos y la interfaz de usuario. La Tabla 6.6 resume los principales casos de prueba y sus objetivos.

TEST	**Objetivo del caso de prueba**
Ranking con múltiples modelos con puntajes distintos	Validar que el sistema ordene correctamente los modelos según su puntuación de priorización.
Empates entre modelos	Verificar el manejo de empates aplicando las reglas de desempate definidas.
Un solo modelo válido	Evaluar el comportamiento cuando solo un modelo cumple con los criterios mínimos establecidos.
Sin modelos válidos para la ancestría seleccionada	Comprobar la respuesta del sistema cuando no existen modelos que cumplan los criterios para la ancestría elegida.
Modelos con puntaje idéntico	Validar que el ordenamiento sea consistente cuando todos los modelos presentan el mismo puntaje.

Tabla 6.6: Casos de prueba de aceptación

Durante estas pruebas, se identificaron distintos errores y oportunidades de mejora en la interfaz y el comportamiento del sistema. Una de las observaciones más relevantes fue que la tabla de priorización mostraba resultados incorrectos debido a errores en la extracción de los valores de las métricas clave **AUROC** y **OR**. Este problema se debía a una falla en el backend al identificar correctamente los valores más representativos de cada modelo. Como solución, se revisó y corrigió el proceso de consulta y transformación de estos datos, garantizando así que cada modelo priorizado muestre información precisa.

Adicionalmente, se realizaron las siguientes correcciones:

- Mejora del comportamiento del selector de ancestrías al elegir múltiples opciones.
- Reubicación de los botones de navegación para mayor claridad.
- Ajuste de los componentes visuales para mantener el diseño responsivo sin solapamientos.

- Inclusión de información contextual y marcadores visuales (asterisco *) para campos obligatorios.
- Corrección de errores en el filtrado por *trait categories* y en la visualización de resultados.

Estas modificaciones permitieron cumplir los criterios de aceptación establecidos, garantizando que el sistema fuera funcional, confiable y comprensible para los usuarios finales.

Capítulo 7

Implantación

En esta sección se describe el proceso llevado a cabo para poner en funcionamiento el sistema desarrollado, asegurando que pueda ser instalado, configurado y ejecutado correctamente tanto en entornos de desarrollo como de producción. Se incluyen las indicaciones para la preparación del ambiente, la documentación de usuario y la documentación técnica necesaria para garantizar la reproducibilidad del sistema y facilitar su mantenimiento. Asimismo, se presentan las instrucciones y repositorios oficiales que contienen los pasos detallados para la instalación, configuración de la base de datos y despliegue de los servicios que conforman la plataforma.

7.0.1. Preparación de Ambiente

Para asegurar la reproducibilidad del sistema y permitir la ejecución de las pruebas por terceros, se documentó la preparación del ambiente en los repositorios oficiales del proyecto disponibles en GitHub:

- **Repositorio base (migración para entrega a la Universidad Politécnica de Valencia):** `https://github.com/algarsi3/base_project/priotitization`
- **Frontend (implementación actual en Next.js):** `https://github.com/valegallardo001/PRSfrontend`
- **Backend (implementación actual en Node.js/Prisma):** `https://github.com/valegallardo001/PRSbackend`

En dichos repositorios se detallan los pasos necesarios para levantar el entorno completo, incluyendo:

- Instalación de dependencias mediante `npm install`.

- Configuración de variables de entorno en un archivo `.env`.
- Inicialización de la base de datos MySQL utilizando Prisma (`npx prisma migrate dev`).
- Ejecución del backend con `npm run dev`.
- Ejecución del frontend mediante Next.js con `npm run dev`.

7.0.2. Manual de Usuario

El Manual de Usuario proporciona las instrucciones necesarias para la correcta utilización del sistema, incluyendo la descripción de cada módulo, el flujo de trabajo recomendado y las acciones disponibles para el usuario. Este documento está acompañado de capturas de pantalla y ejemplos prácticos que facilitan la comprensión de las funcionalidades principales, como la selección de rasgos y la priorización de modelos PRS. El contenido completo del manual se encuentra en el Anexo C.

7.0.3. Documentación de desarrollo

Documentación del proceso ETL

Las funciones ETL se encuentran implementadas en el archivo `src/services/etlService.ts`. Cada función representa un proceso encargado de la extracción, transformación y carga de una entidad específica del Catálogo PGS hacia la base de datos del sistema.

A continuación se documentan cada uno de estos procesos:

runTraitETL()

- **Objetivo:** Extrae todos los traits del catálogo, los transforma al formato interno y los inserta en la base de datos.
- **Fuente:** API externa de traits.
- **Salida:** Inserciones en la tabla `Trait`.
- **Log generado:** `traits_log.txt`

runTraitCategoryETL()

- **Objetivo:** Extrae las categorías de traits y sus relaciones con traits, luego las inserta.
- **Transformaciones:** Asocia traits a categorías.
- **Salida:** Tablas `TraitCategory` y relaciones.
- **Log generado:** `trait_category_log.txt`

runPRSModelETL()

- **Objetivo:** Descarga e inserta modelos PRS de forma incremental.
- **Lógica:** Batches transformados e insertados uno a uno.
- **Salida:** Tabla `PRSModel`.
- **Log generado:** `prsmodel_log.txt`

runPublicationETL()

- **Objetivo:** Extrae e inserta publicaciones asociadas a modelos PRS.
- **Fuente:** API de publicaciones.
- **Log generado:** `publications_log.txt`

runPRSModelTraitRelationETL()

- **Objetivo:** Establece relaciones entre modelos PRS y traits asociados.
- **Entrada:** Modelos PRS con sus `traitIds`.
- **Salida:** Relaciones en la tabla intermedia.
- **Log generado:** `prsmodel_traits_log.txt`

runDevelopmentSamplesETL()

- **Objetivo:** Extrae e inserta muestras de desarrollo (*DevelopmentPopulationSamples*).
- **Lógica:** Usa el mapeo `BROAD_ANCESTRY_MAPPING` para normalizar ancestrías.
- **Salida:** Tabla `DevelopmentPopulationSample`.
- **Log generado:** `development_samples_log.txt`

runBroadAncestryCategoryETL()

- **Objetivo:** Inserta las categorías de ancestría a partir de una fuente externa.
- **Transformación:** Estandariza símbolos y nombres.
- **Log generado:** `broad_ancestry_category_log.txt`

runBroadAncestryInModelETL()

- **Objetivo:** Calcula la representación porcentual de cada categoría de ancestría en los modelos PRS.
- **Lógica:** Agrupa muestras por `broadAncestryId` y calcula porcentajes.
- **Salida:** Tabla intermedia de `PRSModel BroadAncestryCategory`.
- **Log generado:** `broadAncestryInModel_log.txt`

runModelEvaluationETL()

- **Objetivo:** Inserta evaluaciones de desempeño (`ModelEvaluations`) de los modelos PRS.
- **Transformaciones:** Crea `EvaluationSamples`, normaliza ancestrías, y vincula métricas de desempeño.
- **Log generado:** `modelEvaluation_log.txt`

runSelectedModelEvaluationsETL(ppmIds)

- **Objetivo:** Inserta evaluaciones de desempeño solo para un subconjunto de IDs seleccionados.
- **Uso:** Interno para pruebas o cargas parciales.
- **Log generado:** `modelEvaluation_selected_log.txt`

Funciones de extracción desde la API externa (externalApi.ts)

El módulo `externalApi.ts` implementa funciones para la obtención de datos desde la API pública del Catálogo PGS (`https://www.pgscatalog.org/rest`). A continuación se documentan las funciones utilizadas:

fetchAllTraits()

- **Objetivo:** Recuperar todos los traits disponibles.
- **URL base:** `/trait/all`
- **Tipo de retorno:** Arreglo de `TraitJson` (con campos `id`, `label`, `description`, `url`).
- **Notas:** Implementa paginación automática.

fetchAllTraitCategories()

- **Objetivo:** Obtener todas las categorías de traits.
- **URL base:** `/trait_category/all`
- **Tipo de retorno:** Arreglo no tipado (`any[]`).

fetchAllPRSModelsIncremental(processFn)

- **Objetivo:** Descargar modelos PRS de forma incremental por lotes.
- **URL base:** `/score/all`
- **Parámetro:** Función de procesamiento por lote (`processFn`).
- **Manejo de errores:** Espera automática ante error `429 Too Many Requests`.

fetchAllPublicationsIncremental(processFn)

- **Objetivo:** Obtener publicaciones relacionadas a los modelos PRS.
- **URL base:** `/publication/all`
- **Parámetro:** Función de procesamiento para cada lote.
- **Notas:** Incluye reintento con espera en caso de sobrecarga de la API.

fetchBroadAncestryCategories()

- **Objetivo:** Obtener todas las categorías de ancestría amplia.
- **URL base:** `/ancestry_categories/`
- **Tipo de retorno:** Objeto tipo `Record<string, any>`.

fetchAllModelEvaluationsIncremental(onBatch, startPpmId?)

- **Objetivo:** Descargar evaluaciones de desempeño (performance) para modelos PRS.
- **URL base:** `/performance/all`
- **Parámetros:**
  - `onBatch`: función que recibe cada lote de datos.
  - `startPpmId` (opcional): permite reanudar desde un ID específico.
- **Notas:** Reanuda desde un punto específico si se proporciona `startPpmId`.

fetchModelEvaluationsByIds(ppmIds)

- **Objetivo:** Recuperar evaluaciones específicas por su ID.
- **URL base:** `/performance/{ppmId}`
- **Parámetro:** Arreglo de strings (`ppmIds`) con los identificadores a consultar.
- **Manejo de errores:** Imprime advertencias y continúa si alguna petición falla.

Funciones de transformación (transformer.ts)

Las funciones definidas en este módulo procesan los datos crudos obtenidos desde la API pública y los transforman a estructuras más uniformes y apropiadas para ser almacenadas en la base de datos o manipuladas internamente.

transformTrait(raw)

- **Entrada:** Objeto `TraitJson`.
- **Salida:** Objeto con los campos `label`, `description`, `url`, e identificadores categorizados por prefijo (EFO, MONDO, HP, OBA, u otros).
- **Uso:** Clasifica el ID del trait según su origen semántico.

transformAllTraits(traits)

- **Entrada:** Arreglo de objetos `TraitJson`.
- **Salida:** Arreglo transformado con uso de `transformTrait`.

transformTraitCategory(raw)

- **Entrada:** Objeto con `label` y arreglo `efotraits`.
- **Salida:** Objeto con `label` y lista de `traitEfoIds`.

transformAllTraitCategories(raw)

- **Entrada:** Objeto con campo `results` conteniendo `TraitCategoryJson[]`.
- **Salida:** Arreglo de categorías transformadas.

transformPRSModel(raw)

- **Entrada:** Objeto con datos del modelo PRS.
- **Salida:** Objeto con nombre, número de SNPs, ID, URL, y PMID de publicación.
- **Nota:** Valida si el campo `PMID` está presente.

extractPRSModelTraitRelations(raw)

- **Entrada:** Objeto del modelo PRS.
- **Salida:** Relación entre modelo PRS y lista de IDs de traits asociados.

transformAllPRSModels(rawData)

- **Entrada:** Objeto con campo `results`.
- **Salida:** Arreglo de modelos transformados con `transformPRSModel`.
- **Manejo de errores:** Valida que los resultados sean válidos antes de continuar.

transformPublication(raw)

- **Entrada:** Publicación en crudo desde el Catálogo PGS.
- **Salida:** Objeto con título, revista, autores, fecha, DOI y PMID (o nulo si es `bioRxiv` o `medRxiv`).

extractDevelopmentPopulationSamples(raw)

- **Entrada:** Objeto con campos `samples_variants` y `samples_training`.
- **Salida:** Arreglo de muestras poblacionales (tipo BASE o TUNING), incluyendo edad, género, cohorte y ancestría.

transformBroadAncestryCategories(rawData)

- **Entrada:** Objeto de pares `symbol: valor`.
- **Salida:** Arreglo con símbolo y etiqueta de categoría de ancestría.

transformModelEvaluation(raw)

- **Entrada:** Evaluación de modelo cruda.
- **Salida:** Objeto con ID de evaluación, trait reportado, covariables, ID de modelo y publicación.

transformEvaluationSample(raw)

- **Entrada:** Evaluación con conjunto de muestras.
- **Salida:** Objeto que representa una muestra con detalles de individuos, cohorte, ancestría, edad, y fuentes.

Documentación de los endpoints de la página web

La documentación de los endpoints de la API fue generada automáticamente utilizando la herramienta Swagger UI [25]. Esta proporciona una interfaz gráfica interactiva para consultar y probar los endpoints, así como una exportación en PDF que se incorpora a continuación:

Documentación de la API 1.0.0 OAS 3.0

API para gestionar ancestrías, traits y categorías de traits.

Broad Ancestry Operaciones relacionadas con las ancestrías amplias

GET `/api/ancestries` Obtener todas las categorías de ancestría

Parameters Try it out

No parameters

Responses

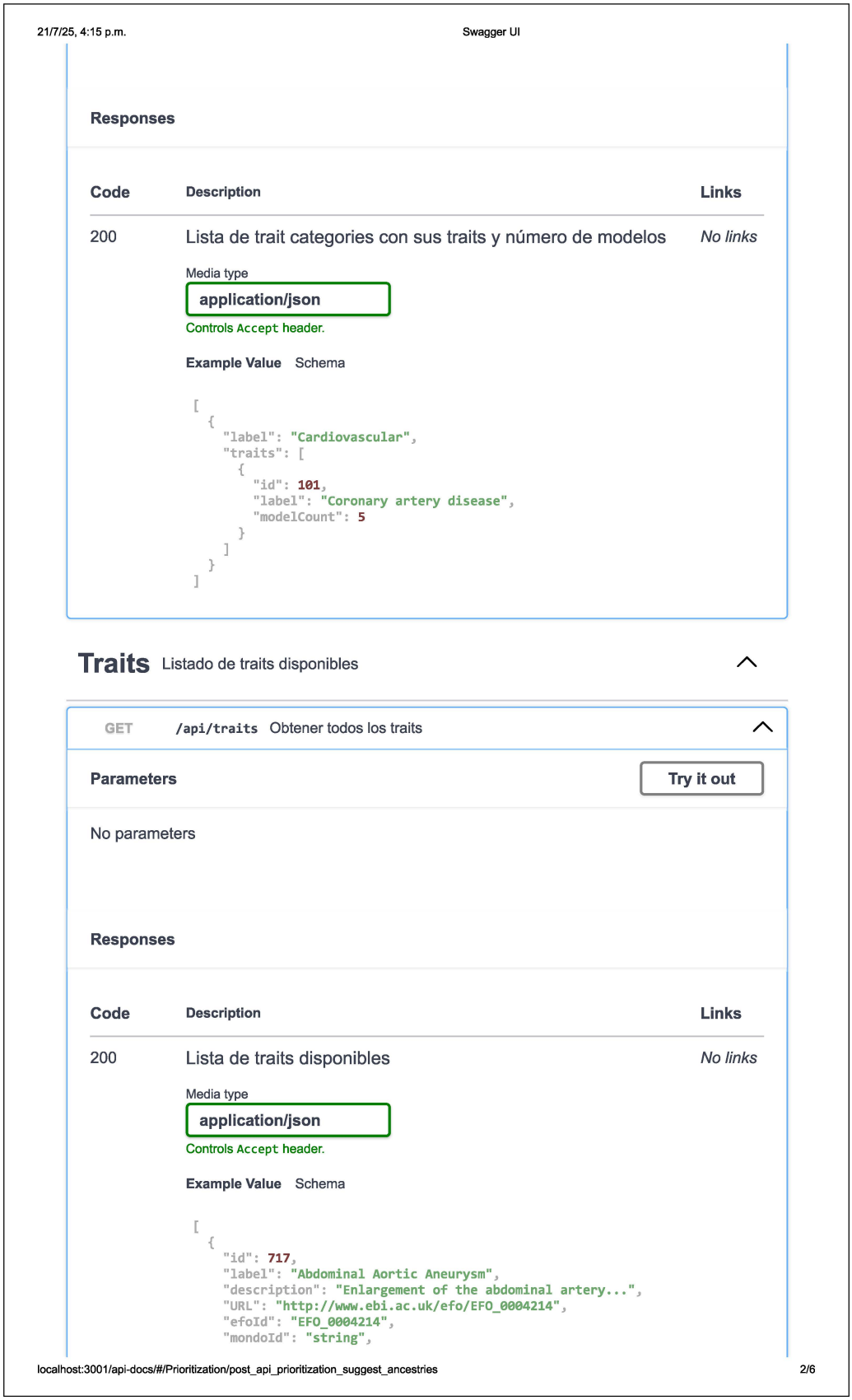
Code	Description	Links
200	Lista de categorías de ancestría Media type: application/json Controls Accept header. **Example Value** Schema `[ { "id": 1, "label": "European", "symbol": "EUR" } ]`	*No links*
500	Error interno al obtener las categorías	*No links*

Trait Category Categorías de traits con conteo de modelos

GET `/api/trait-categories`
Obtener todas las trait categories con su conteo de modelos

Parameters Try it out

No parameters



Responses

Code	Description	Links
200	Lista de trait categories con sus traits y número de modelos Media type: application/json Controls Accept header. **Example Value** Schema	*No links*

```
[
  {
    "label": "Cardiovascular",
    "traits": [
      {
        "id": 101,
        "label": "Coronary artery disease",
        "modelCount": 5
      }
    ]
  }
]
```

Traits Listado de traits disponibles

GET `/api/traits` Obtener todos los traits

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	Lista de traits disponibles Media type: application/json Controls Accept header. **Example Value** Schema	*No links*

```
[
  {
    "id": 717,
    "label": "Abdominal Aortic Aneurysm",
    "description": "Enlargement of the abdominal artery...",
    "URL": "http://www.ebi.ac.uk/efo/EFO_0004214",
    "efoId": "EFO_0004214",
    "mondoId": "string",
```

Code	Description	Links


```
    "hpoId": "string",
    "orphaId": "string",
    "otherId": "string"
  }
]
```


Prioritization Endpoints para priorización de modelos PRS

POST `/api/prioritization/prioritize`
Prioriza modelos PRS según traits seleccionados y una ancestría principal

Parameters Try it out

No parameters

Request body required application/json

Example Value Schema


```
{
  "selectedTraitIds": [
    101,
    102
  ],
  "mainAncestrySymbol": "EUR"
}
```


Responses

Code	Description	Links
200	Lista de modelos PRS priorizados Media type: application/json Controls Accept header. **Example Value** Schema	*No links*


```
[
  {
    "modelId": 0,
    "pgscId": "string",
    "name": "string",
    "ancestryScore": 0,
    "or": 0,
    "auroc": 0,
    "finalScore": 0,
    "trait_label": "string",
    "ancestry": "string",
    "num_snps": 0,
    "dev_sample": 0,
    "eval_ancestry": "string",
    "reported_trait": "string",
    "year": "string",
    "pubmed_id": "string"
```

Code	Description	Links
	}]	
400	Datos de entrada incompletos o no válidos	*No links*
500	Error interno en la priorización	*No links*

POST `/api/prioritization/suggest-ancestries`
Sugiere ancestrías disponibles para traits seleccionados

Parameters Try it out

No parameters

Request body required application/json

Example Value Schema

```
{
  "selectedTraitIds": [
    "EFO_0001645",
    "MONDO_001"
  ]
}
```

Responses

Code	Description	Links
200	Lista de ancestrías sugeridas Media type application/json Controls `Accept` header. **Example Value** Schema `[ { "symbol": "EUR", "label": "European" } ]`	*No links*
400	No se proporcionaron trait IDs	*No links*
500	Error interno del servidor	*No links*

Schemas

BroadAncestryCategory {

```
id
    integer
example: 1
label
    string
example: European
symbol
    string
example: EUR
}
```

TraitCategory {

```
label
    string
example: Cardiovascular
traits
    [   {
    id
        integer
example: 101
    label
        string
example: Coronary artery
disease
    modelCount
        integer
example: 5
 }]
}
```

TraitCategoryGrouped {

```
label
    string
example: Cardiovascular
totalPrs
    integer
example: 28
traits
    [   {...}]
}
```

Trait {

```
id
    integer
example: 717
label
    string
example: Abdominal Aortic Aneurysm
description
    string
example: Enlargement of the abdominal artery...
URL
    string
example: http://www.ebi.ac.uk/efo/EFO_0004214
efoId
    string
example: EFO_0004214
mondoId
    string
nullable: true
hpoId
    string
nullable: true
orphaId
    string
nullable: true
otherId
    string
nullable: true
}
```

PRSModelPrioritized {

modelId	integer *example: 128*
pgscId	string *example: PGS000128*
name	string *example: Coronary artery disease*
ancestryScore	number *example: 1*
or	number *example: 2.3*
auroc	number *example: 0.76*
finalScore	number *example: 0.88*
trait_label	string *example: CAD*
ancestry	string *example: EUR (European) 90%*
num_snps	integer *example: 3251*
dev_sample	integer *example: 14500*
eval_ancestry	string *example: European*
reported_trait	string *example: Heart disease*
year	string *example: 2023*
pubmed_id	string *example: PMID:12345678*

}

Capítulo 8

Conclusiones

Este trabajo presentó el diseño e implementación de una plataforma web para la extracción y priorización de modelos de riesgo poligénico, basada en datos públicos provenientes del *PGS Catalog*. La propuesta se desarrolló en torno a una arquitectura modular que integró componentes *frontend* interactivos, un *backend* eficiente y un proceso ETL especializado.

Durante el desarrollo se logró cumplir con los objetivos principales propuestos: extraer modelos PRS desde una fuente externa confiable, transformarlos y almacenarlos en una base de datos estructurada, y poner a disposición del usuario final un sistema amigable que permite seleccionar rasgos de interés, visualizar información complementaria y priorizar modelos en base a criterios relevantes como la ancestría, el poder discriminativo (AUC/OR) y el año de publicación. La herramienta permite al usuario tomar decisiones informadas mediante visualizaciones, filtros personalizados, ordenamientos y un ranking final de modelos PRS seleccionados.

Uno de los principales desafíos encontrados a lo largo del trabajo fue el cambio de tecnología en el proceso de extracción de datos. En una primera etapa, la estrategia utilizada consistía en obtener los datos mediante scripts en Python utilizando la biblioteca *pandas*, lo que permitió un prototipado inicial efectivo. Sin embargo, este enfoque resultó limitado en cuanto a eficiencia, escalabilidad y tiempos de ejecución, especialmente al trabajar con grandes volúmenes de datos desde endpoints REST. Por esta razón, se decidió migrar todo el proceso de extracción y transformación hacia el *backend* en Node.js, permitiendo así un manejo asincrónico más ágil y una integración directa con la lógica de negocio de la aplicación. Esta decisión técnica implicó rehacer buena parte del módulo de extracción, pero redundó en una solución mucho más robusta, rápida y escalable.

Otra dificultad significativa fue la necesidad de acotar el alcance original del sistema. Si bien inicialmente se contemplaba una integración completa con una plataforma mayor de análisis genómico, se optó por centrar este trabajo en el desarrollo y validación del módulo de priorización, con vistas a su futura integración. Esta decisión permitió concentrar los esfuerzos en la construcción de una interfaz intuitiva, la implementación de la lógica de

ranking personalizada y la validación de las funcionalidades críticas del sistema.

Este proyecto forma parte de una iniciativa más amplia liderada por el Centro de Investigación PROS de la Universidad Politécnica de Valencia. El módulo desarrollado aquí constituye uno de los pilares de esa plataforma integral, enfocándose específicamente en la etapa de selección y priorización de modelos PRS. Posteriormente, los investigadores del centro serán los encargados de desarrollar e integrar la parte de análisis genético y clínico, lo que permitirá unificar todos los componentes en un solo sistema. Esta futura integración dotará a la plataforma de capacidades completas, desde la exploración de modelos hasta su evaluación aplicada en contextos reales.

Si bien el sistema permite priorizar modelos en base a información pública y criterios configurables, aún existen desafíos relacionados con la estandarización de métricas, la diversidad poblacional representada en los modelos PRS disponibles y la interpretación clínica de los resultados. Además de estas limitaciones inherentes al dominio de los modelos PRS, también se identifican ciertas limitaciones técnicas del sistema desarrollado. Por ejemplo, actualmente no se implementa la visualización jerárquica de rasgos (*traits*), lo que impide mostrar relaciones de dependencia entre rasgos principales y secundarios (*child traits*), una funcionalidad que estuvo considerada en etapas iniciales. Asimismo, el sistema de actualización de datos no es completamente automático, por lo que requiere una ejecución manual del proceso ETL para reflejar nuevos modelos y actualizaciones del catálogo original, lo que podría limitar la actualización continua de la información disponible para los usuarios.

Como proyecciones, se contempla la extensión del sistema hacia nuevas fuentes de datos, la incorporación de medidas de calidad adicionales para los modelos (como la replicabilidad o el tamaño de muestra), y una integración más estrecha con herramientas de análisis genético aplicadas a cohortes locales. También se prevé incorporar autenticación de usuarios, generación de reportes descargables y funcionalidades para el análisis comparativo entre múltiples modelos para un mismo rasgo o condición.

En conclusión, el trabajo desarrollado constituye un avance importante hacia la creación de herramientas accesibles que permitan aprovechar el valor de los modelos de riesgo poligénico en contextos de investigación y eventualmente en aplicaciones clínicas. Pese a las dificultades técnicas enfrentadas, la solución propuesta demuestra ser efectiva y con gran potencial de crecimiento, destacándose por su enfoque modular, su adaptabilidad a nuevos criterios de evaluación y su capacidad para integrarse a sistemas mayores orientados al análisis de datos genómicos.

Bibliografía

[1] D. Martinez-Minguet and O. Pastor, "Conceptual modeling for polygenic risk score research: Improving domain understanding and clinical utility," in International Conference on Advanced Information Systems Engineering. Springer, 2024, pp. 3–10.

[2] S. Roger, "Pressman, ingeniería del software un enfoque práctico séptima edición," México DF: McGraw Hill, pp. 1–74, 2010.

[3] PGS Catalog, "PGS Catalog: A database of published polygenic scores," https://www.pgscatalog.org/, 2024.

[4] N. Wray et al., "From basic science to clinical application of polygenic risk scores: A primer," JAMA psychiatry, vol. 78, 09 2020.

[5] J. Collister et al., "Calculating Polygenic Risk Scores (PRS) in UK Biobank: A practical guide for epidemiologists," Frontiers in Genetics, vol. 13, 02 2022.

[6] S. W. S. Choi et al., "Tutorial: a guide to performing polygenic risk score analyses," Nature Protocols, vol. 15, 07 2020.

[7] S. A. Lambert, B. Wingfield, J. T. Gibson, L. Gil, S. Ramachandran, F. Yvon, S. Saverimuttu, E. Tinsley, E. Lewis, S. C. Ritchie et al., "The polygenic score catalog: new functionality and tools to enable fair research," medRxiv, pp. 2024–05, 2024.

[8] J. Smith, J. Doe, and L. Zhang, "Genopred: A computational pipeline for polygenic risk scoring," Journal of Computational Genetics, vol. 15, no. 2, pp. 123–145, 2024.

[9] O. Pain, A. Al-Chalabi, and C. Lewis, "The genopred pipeline: A comprehensive and scalable pipeline for polygenic scoring." medRxiv, pp. 2024–06, 2024.

[10] National Institute for Health and Care Research, "Genopred - homepage," https://opain.github.io/GenoPred/.

[11] P. S. P. C. team, "pgsc_calc: a reproducible workflow to calculate polygenic scores — Polygenic Score (PGS) Catalog Calculator documentation," https://pgsc-calc.readthedocs.io/en/latest/index.html.

[12] M. U. o. I. Lukas Forer, Institute of Genetic Epidemiology, "GitHub - lukfor/pgs-calc: Applying polygenic scores (PGS) on imputed genotypes, howpublished = https://github.com/lukfor/pgs-calc."

[13] J. Becker et al., "Github - aysuo/pgi_repo: A repository for polygenic indexes," https://github.com/aysuo/PGI_Repo.

[14] P. Palmer et al., "GitHub - lifebit-ai/prs: Polygenic Risk Scores Pipeline to calculate genetic risk score for any given phenotype/trait (using PRSice)," https://github.com/lifebit-ai/prs.

[15] X. L. Jennifer Collister, "PRS Pipeline," https://2cjenn.github.io/PRS_Pipeline/#exe, March 2022.

[16] "Basic tutorial for polygenic risk score analyses," https://choishingwan.github.io/PRS-Tutorial/.

[17] "GitHub - MareesAT/GWA_tutorial: A comprehensive tutorial about GWAS and PRS," https://github.com/MareesAT/GWA_tutorial/.

[18] M. Page et al., "The polygenic risk score knowledge base offers a centralized online repository for calculating and contextualizing polygenic risk scores," Communications biology, vol. 5, no. 1, p. 899, 2022.

[19] "ISO 25010," https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[20] "Mysql :: Mysql documentation." [Online]. Available: https://dev.mysql.com/doc/

[21] "Index | node.js v24.5.0 documentation." [Online]. Available: https://nodejs.org/docs/latest/api/

[22] "Prisma | simplify working and interacting with databases." [Online]. Available: https://www.prisma.io/

[23] true, "Next.js by vercel - the react framework." [Online]. Available: https://nextjs.org/

[24] "React." [Online]. Available: https://react.dev/

[25] "Rest api documentation tool | swagger ui." [Online]. Available: https://swagger.io/tools/swagger-ui/

[26] Microsoft, “Documentation for visual studio code,” 11 2021. [Online]. Available: https://code.visualstudio.com/docs

[27] “Typescript: Documentation - typescript for javascript programmers.” [Online]. Available: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

[28] “<html> - html: Lenguaje de etiquetas de hipertexto | mdn,” 6 2025. [Online]. Available: https://developer.mozilla.org/es/docs/Web/HTML/Reference/Elements/html

[29] “Javascript - javascript en español.” [Online]. Available: https://lenguajejs.com/javascript/

[30] “¿qué es sql? - explicación de lenguaje de consulta estructurado (sql) - aws.” [Online]. Available: https://aws.amazon.com/es/what-is/sql/

[31] “Mediator.” [Online]. Available: https://refactoring.guru/es/design-patterns/mediator

[32] “Container/presentational pattern.” [Online]. Available: https://www.patterns.dev/react/presentational-container-pattern/

Apéndice A

Anexo 1

Tabla: PerformanceMetric

Esta tabla almacena el catálogo de métricas utilizadas para evaluar el rendimiento de los modelos PRS, como AUC o OR.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la métrica de desempeño.	INTEGER	Clave primaria, autoincremental.
nameShort	Nombre corto de la métrica (ej. AUC, OR).	VARCHAR(20)	No nulo.
nameLong	Nombre largo de la métrica.	TEXT	No nulo.
type	Tipo de métrica (ej. asociación, discriminación).	VARCHAR(50)	No nulo.

Tabla A.1: Definición de la tabla PerformanceMetric

.

Tabla: PerformanceMetricEvaluation

Contiene los resultados de la evaluación de las métricas definidas en PerformanceMetric para un modelo específico.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la evaluación de la métrica.	INTEGER	Clave primaria, autoincremental.
estimate	Valor estimado de la métrica (por ejemplo, AUC o OR).	FLOAT	No nulo.
CILower	Límite inferior del intervalo de confianza.	FLOAT	Opcional.
CIUpper	Límite superior del intervalo de confianza.	FLOAT	Opcional.
modelEvaluationId	Referencia a la evaluación del modelo (ModelEvaluation).	INTEGER	Clave foránea, no nulo.
performanceMetricId	Referencia a la métrica utilizada PerformanceMetric.	INTEGER	Clave foránea, no nulo.

Tabla A.2: Definición de la tabla PerformanceMetricEvaluation

.

Tabla: ModelEvaluation

Registra las evaluaciones completas realizadas a un modelo PRS, incluyendo la población evaluada, la publicación asociada y las métricas calculadas.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la evaluación del modelo.	INTEGER	Clave primaria, autoincremental.
reportedTrait	Descripción del rasgo informado en el estudio.	TEXT	Opcional.
covariates	Covariables utilizadas en la evaluación.	TEXT	Opcional.
ppmId	Identificador externo del conjunto de métricas del modelo.	VARCHAR(50)	Único, opcional.
prsModelId	Referencia al modelo PRS evaluado.	INTEGER	Clave foránea, no nulo.

Tabla A.3 – continuación

Atributo	Descripción	Tipo de dato	Restricciones
publicationId	Referencia a la publicación científica asociada.	INTEGER	Clave foránea, no nulo.
evaluationPopulationSampleId	Referencia a la muestra poblacional evaluada.	INTEGER	Clave foránea, no nulo.

Tabla A.3: Definición de la tabla ModelEvaluation.

Tabla: Publication

Almacena información sobre los estudios científicos de los cuales se obtuvieron los modelos PRS o sus evaluaciones.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la publicación.	INTEGER	Clave primaria, autoincremental.
title	Título del estudio publicado.	TEXT	No nulo.
journal	Revista científica donde fue publicada.	VARCHAR(150)	No nulo.
author	Autor principal o primer autor del estudio.	VARCHAR(100)	No nulo.
date	Fecha completa de publicación.	DATE	Opcional.
year	Año de publicación.	INTEGER	No nulo.
PMID	Identificador PubMed del estudio.	VARCHAR(20)	Único, opcional.
DOI	Identificador digital del artículo (DOI).	VARCHAR(100)	Único, opcional.
pgpId	Identificador externo vinculado desde la fuente de datos original.	VARCHAR(50)	Opcional.

Tabla A.4: Definición de la tabla Publication.

Tabla: EvaluationPopulationSamples

Describe las características de las muestras poblacionales utilizadas en la evaluación de los modelos PRS.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la muestra poblacional evaluada.	INTEGER	Clave primaria, autoincremental.
numberOfIndividuals	Número total de individuos en la muestra.	INTEGER	No nulo.
numberOfCases	Número de individuos con el fenotipo de interés.	INTEGER	Opcional.
numberOfControls	Número de individuos sin el fenotipo de interés.	INTEGER	Opcional.
percentMale	Porcentaje de individuos de sexo masculino.	FLOAT	Opcional.
age	Edad promedio de los individuos en la muestra.	FLOAT	Opcional.
ageUnits	Unidad de medida para la edad (e.g., años).	VARCHAR(20)	Opcional.
ancestryBroad	Categoría general de ancestría (e.g., EUR, AFR).	VARCHAR(10)	No nulo.
ancestryDetails	Descripción más específica de la ancestría.	TEXT	Opcional.
cohort	Nombre del o los cohortes incluidos en la muestra.	TEXT	Opcional.
gcId	Identificador del catálogo GWAS asociado.	VARCHAR(50)	Opcional.
sourcePMID	Identificador PubMed de la fuente de la muestra.	VARCHAR(20)	Opcional.

Tabla A.5 – continuación

Atributo	Descripción	Tipo de dato	Restricciones
sourceDOI	Identificador DOI de la fuente.	VARCHAR(100)	Opcional.
phenotypeFree	Descripción libre del fenotipo evaluado.	TEXT	Opcional.
pssId	Identificador de la muestra de evaluación (PSS).	VARCHAR(50)	Opcional.
broadAncestryId	Clave foránea a la categoría de ancestría.	INTEGER	Relación con BroadAncestryCategory.

Tabla A.5: Definición de la tabla EvaluationPopulationSamples.

Tabla: BroadAncestryCategory

Contiene la clasificación general de ancestrías utilizadas en el sistema (por ejemplo, EUR, AFR, SAS).

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la categoría de ancestría.	INTEGER	Clave primaria, autoincremental.
symbol	Código corto de la categoría (e.g., EUR, AFR, SAS).	VARCHAR(10)	Único, no nulo.
label	Descripción larga de la categoría de ancestría.	VARCHAR(100)	No nulo.

Tabla A.6: Definición de la tabla BroadAncestryCategory.

Tabla: BroadAncestryInRefPop

Relaciona una categoría de ancestría con una población de referencia, incluyendo el porcentaje representativo.

Atributo	Descripción	Tipo de dato	Restricciones
broadAncestryId	Referencia a la categoría de ancestría amplia asociada.	INTEGER	Clave foránea a BroadAncestryCategory, no nulo.
referencePopulationId	Referencia a la población de referencia.	INTEGER	Clave foránea a ReferencePopulation, no nulo.
percentage	Porcentaje representativo de la ancestría en la población de referencia.	FLOAT	No nulo.

Tabla A.7: Definición de la tabla BroadAncestryInRefPop.

Tabla: DevelopmentPopulationSample

Registra las muestras poblacionales utilizadas en el desarrollo de los modelos PRS, incluyendo tamaño muestral, ancestría y rol.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la muestra poblacional.	INTEGER	Clave primaria, autoincremental.
numberOfIndividuals	Número total de individuos en la muestra.	INTEGER	No nulo.
numberOfCases	Número de casos en la muestra.	INTEGER	Opcional.
numberOfControls	Número de controles en la muestra.	INTEGER	Opcional.
percentMale	Porcentaje de individuos masculinos.	FLOAT	Opcional.
age	Edad estimada promedio de la muestra.	FLOAT	Opcional.
ageUnits	Unidades utilizadas para la edad (años, meses, etc.).	VARCHAR(20)	Opcional.
ancestryBroad	Ancestría amplia declarada para la muestra.	VARCHAR(100)	No nulo.

ancestryDetails	Detalles específicos de la ancestría.	TEXT	Opcional.
cohort	Nombre(s) de cohorte(s) a la que pertenece la muestra.	TEXT	Opcional.
gcId	Identificador del estudio GWAS en el catálogo.	VARCHAR(50)	Opcional.
sourcePMID	PMID de la fuente de datos.	VARCHAR(20)	Opcional.
sourceDOI	DOI de la fuente de datos.	TEXT	Opcional.
role	Rol de la muestra en el modelo (por ejemplo, 'BASE' o 'TUNING').	VARCHAR(20)	No nulo.
prsModelId	Clave foránea al modelo PRS asociado.	INTEGER	Relación con PRSModel.
broadAncestryId	Clave foránea a la categoría de ancestría amplia.	INTEGER	Relación con BroadAncestryCategory.

Tabla A.8: Definición de la tabla DevelopmentPopulationSample.

Tabla: ReferencePopulation

Define las poblaciones de referencia utilizadas para validar o contextualizar los modelos PRS.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la población de referencia.	INTEGER	Clave primaria, autoincremental.
name	Nombre identificador de la población de referencia.	VARCHAR(100)	No nulo.
description	Descripción general de la población.	TEXT	Opcional.
studyURL	URL al estudio o fuente de datos.	TEXT	Opcional.
assembly	Ensamblaje genómico utilizado en el estudio.	VARCHAR(50)	Opcional.
numberOfindividuals	Número total de individuos en la población.	INTEGER	No nulo.
numberOfvariants	Número de variantes consideradas.	INTEGER	No nulo.
studyName	Nombre del estudio que originó la población de referencia.	VARCHAR(100)	Opcional.
date	Fecha del estudio.	DATE	Opcional.
path	Ruta al archivo de referencia genómica.	TEXT	Opcional.

Tabla A.9: Definición de la tabla ReferencePopulation.

Tabla: DevelopmentPopulationSample

Registra las muestras poblacionales utilizadas en el desarrollo de los modelos PRS, incluyendo tamaño muestral, ancestría y rol.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único de la muestra poblacional.	INTEGER	Clave primaria, autoincremental.
numberOfIndividuals	Número total de individuos en la muestra.	INTEGER	No nulo.
numberOfCases	Número de casos en la muestra.	INTEGER	Opcional.
numberOfControls	Número de controles en la muestra.	INTEGER	Opcional.
percentMale	Porcentaje de individuos masculinos.	FLOAT	Opcional.

Tabla A.10 – continuación

Atributo	Descripción	Tipo de dato	Restricciones
age	Edad estimada promedio de la muestra.	FLOAT	Opcional.
ageUnits	Unidades utilizadas para la edad (años, meses, etc.).	VARCHAR(20)	Opcional.
ancestryBroad	Ancestría amplia declarada para la muestra.	VARCHAR(100)	No nulo.
ancestryDetails	Detalles específicos de la ancestría.	TEXT	Opcional.
cohort	Nombre(s) de cohorte(s) a la que pertenece la muestra.	TEXT	Opcional.
gcId	Identificador del estudio GWAS en el catálogo.	VARCHAR(50)	Opcional.
sourcePMID	PMID de la fuente de datos.	VARCHAR(20)	Opcional.
sourceDOI	DOI de la fuente de datos.	TEXT	Opcional.
role	Rol de la muestra en el modelo (por ejemplo, 'BASE' o 'TUNING').	VARCHAR(20)	No nulo.
prsModelId	Clave foránea al modelo PRS asociado.	INTEGER	Relación con PRSModel.
broadAncestryId	Clave foránea a la categoría de ancestría amplia.	INTEGER	Relación con BroadAncestryCategory.

Tabla A.10: Definición de la tabla DevelopmentPopulationSample.

Tabla: BroadAncestryModel

Asocia un modelo PRS con una categoría de ancestría y el porcentaje de validación correspondiente.

Atributo	Descripción	Tipo de dato	Restricciones
broadAncestryId	Clave foránea a la categoría de ancestría amplia evaluada.	INTEGER	Relación con BroadAncestryCategory.
prsModelId	Clave foránea al modelo PRS evaluado.	INTEGER	Relación con PRSModel.
percentage	Porcentaje de validación del modelo en la ancestría correspondiente.	FLOAT	Valor entre 0 y 100.

Tabla A.11: Definición de la tabla BroadAncestryModel.

Tabla: PRSModel

Contiene los metadatos principales de cada modelo PRS almacenado en el sistema.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único del modelo PRS.	INTEGER	Clave primaria.
name	Nombre del modelo PRS.	VARCHAR(255)	No nulo.
numberOfSNP	Número de variantes SNP incluidas en el modelo.	INTEGER	No nulo.
pgscId	Identificador oficial en el catálogo PGSC.	VARCHAR(50)	No nulo, único.
pgscURL	Enlace directo al modelo PRS en el catálogo PGSC.	TEXT	No nulo.
publicationId	Clave foránea a la publicación asociada.	INTEGER	Relación con `Publication`.

Tabla A.12: Definición de la tabla PRSModel.

Tabla: PrioritizedModel

Registra los modelos seleccionados y ordenados tras el proceso de priorización, incluyendo su posición en el ranking.

Atributo	Descripción	Tipo de dato	Restricciones
prsModelId	Clave foránea al modelo PRS seleccionado.	INTEGER	Relación con `PRSModel`.
prsAnalysisId	Clave foránea al análisis PRS al que pertenece el modelo.	INTEGER	Relación con `PRSAnalysis`.
position	Posición o ranking del modelo en la priorización.	INTEGER	No nulo.
prsResultId	Clave foránea al resultado del modelo.	INTEGER	Relación con `PRSResult`.

Tabla A.13: Definición de la tabla PrioritizedModel.

Tabla: ScoringFile

Guarda la información de los archivos de puntuación asociados a cada modelo PRS.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único del archivo de puntuación.	INTEGER	Clave primaria, autoincremental.
assembly	Ensamblaje genómico utilizado (ej: GRCh37, GRCh38).	VARCHAR(50)	No nulo.
path	Ruta del archivo de puntuación en el sistema o repositorio.	TEXT	No nulo.
prsModelId	Clave foránea al modelo PRS al que pertenece el archivo.	INTEGER	Relación con `PRSModel`.

Tabla A.14: Definición de la tabla ScoringFile.

Tabla: PRSModelToTrait

Establece la relación entre los modelos PRS y los rasgos (traits) que predicen.

Atributo	Descripción	Tipo de dato	Restricciones
traitId	Clave foránea que hace referencia a un rasgo (trait) asociado al modelo PRS.	INTEGER	Clave primaria compuesta, relacionado con `Trait`.
prsModelId	Clave foránea que hace referencia al modelo PRS relacionado con el rasgo.	INTEGER	Clave primaria compuesta, relacionado con `PRSModel`.

Tabla A.15: Definición de la tabla PRSModelToTrait.

Tabla: Trait

Almacena los rasgos biomédicos que pueden ser asociados a uno o más modelos PRS.

Atributo	Descripción	Tipo de dato	Restricciones
id	Identificador único del rasgo.	INTEGER	Clave primaria, autoincremental.
label	Nombre del rasgo biomédico.	VARCHAR(255)	No nulo.
description	Descripción opcional del rasgo.	TEXT	Opcional.
URL	Enlace a la ontología del rasgo.	TEXT	Opcional.
efoId	Identificador del rasgo en el sistema EFO (Experimental Factor Ontology).	VARCHAR(50)	Opcional.
mondoId	Identificador MONDO del rasgo.	VARCHAR(50)	Opcional.
hpoId	Identificador del rasgo en Human Phenotype Ontology (HPO).	VARCHAR(50)	Opcional.
orphaId	Identificador ORPHA del rasgo.	VARCHAR(50)	Opcional.
otherId	Otro identificador alternativo si no pertenece a los sistemas anteriores.	VARCHAR(50)	Opcional.

Tabla A.16: Definición de la tabla Trait.

Tabla: TraitToTraitCategory

Relaciona cada rasgo con una o varias categorías de rasgos.

Atributo	Descripción	Tipo de dato	Restricciones
traitCategoryId	Identificador de la categoría de rasgo.	INTEGER	Clave foránea a `TraitCategory`.
traitId	Identificador del rasgo biomédico.	INTEGER	Clave foránea a `Trait`.

Tabla A.17: Definición de la tabla TraitToTraitCategory.

Tabla: TraitCategory

Contiene la lista de categorías de rasgos biomédicos manejadas en el sistema.

Atributo	**Descripción**	**Tipo de dato**	**Restricciones**
id	Identificador único de la categoría de rasgos.	INTEGER	Clave primaria, autoincremental.
label	Nombre descriptivo de la categoría de rasgos.	VARCHAR(255)	No nulo, único.

Tabla A.18: Definición de la tabla TraitCategory.

Apéndice B

Anexo 2

B.0.1. Pruebas Unitarias para la clase Trait

Validó la existencia de registros en la tabla Trait asegurando que los datos importados coincidan con la fuente original.

Nombre de la Prueba	**Tablas Involucradas**	**Objetivo**	**Resultado Esperado**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
Existencia de Traits	Trait	Verificar que haya datos en la tabla Trait	656 filas	No se cargaron los datos inicialmente porque el efoId era un string en la consulta, pero estaba almacenado como entero en la BD.	Se cargaron 50 filas porque la columna description tenía poco espacio. Se tuvo que aumentar el tamaño del campo a VARCHAR(3000).	Finalmente se pudieron cargar todos los datos correctamente.

Tabla B.1: Pruebas Unitarias clase *Trait*.

B.0.2. Pruebas Unitarias para la clase TraitCategory

Comprobó la correcta carga de categorías, la ausencia de traits sin categoría asignada y la ausencia de categorías vacías, garantizando integridad entre *Trait*, *TraitCategory* y la tabla intermedia.

Nombre de la Prueba	**Tablas Involucradas**	**Objetivo**	**Resultado Esperado**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
Existencia de Categorías	TraitCategory	Verificar que haya datos en TraitCategory	16 filas	Todos los datos se cargaron correctamente sin problemas.	La carga de datos fue estable en este ciclo.	Sin cambios. Se cargaron los datos correctamente.
Traits sin categoría	Trait, TraitToTraitCategory	Detectar traits sin categoría asignada	0 filas	Se encontraron 5 traits sin categorías asignadas. Esto se debió a que algunos traits fueron ingresados sin categorías asignadas.	Se corrigió la asignación de categorías y se encontraron solo 1 trait sin categoría.	Todos los traits ahora tienen categoría asignada, no se encontraron filas.
Categorías sin traits	TraitCategory, TraitToTraitCategory	Detectar categorías sin traits asignados	0 filas	Se encontró 1 categoría sin ningún trait asignado.	Se solucionó el problema y ahora todas las categorías tienen al menos un trait.	No se encontraron categorías sin traits asignados.

Tabla B.2: Pruebas Unitarias clase *TraitCategory*.

B.0.3. Pruebas Unitarias para la clase Publications

Verificó la correcta inserción de todas las publicaciones, asegurando que campos como `title` tuvieran tamaño suficiente y que la carga no fallara por restricciones de longitud.

Nombre de la Prueba	**Tablas Involucradas**	**Objetivo**	**Resultado Esperado**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
Inserción de datos en Publication	Publication	Verificar correcta carga de publicaciones	Inserción de todas las publicaciones esperadas	No se insertaron publicaciones debido a que el campo `title` estaba limitado a `VARCHAR(191)`.	Se aumentó el tamaño del campo a `VARCHAR(1000)` y se insertaron todas las publicaciones.	Todas las publicaciones fueron insertadas sin problemas.

Tabla B.3: Pruebas Unitarias clase *Publication*.

B.0.4. Pruebas Unitarias para la clase PRSModel

Validó que todos los modelos PRS fueran insertados correctamente y asociados a publicaciones válidas, además de comprobar que cada modelo estuviera relacionado a traits existentes.

Nombre de la Prueba	Tablas Involucradas	Objetivo	Resultado Esperado	Ciclo 1	Ciclo 2	Ciclo 3
Inserción de datos en PRSModel	PRSModel, Publication	5000 filas	Inserción exitosa de modelos relacionados a publicaciones válidas	No se insertaron modelos porque primero se debían insertar las publicaciones (Publication).	Se insertaron algunos modelos, pero en ciertos casos falló porque no se encontró la publicación con el PMID correspondiente.	Se insertaron correctamente todos los modelos PRS.
Relación entre PRSModel y Trait	PRSModel, Trait	Verificar que todos los modelos PRS estén correctamente relacionados con traits válidos	Todos los modelos relacionados exitosamente a traits existentes	No se cargaron todos los datos debido a errores en las relaciones. Algunos modelos hacen referencia a traits cuyo `orpha_id` aún no existe.	Se corrigieron algunos traits y se cargaron más relaciones, aunque persistían errores en modelos con traits aún no registrados.	Todas las relaciones entre modelos y traits fueron exitosamente establecidas.

Tabla B.4: Pruebas Unitarias *PRSModel*.

B.0.5. Pruebas Unitarias para la clase BroadAncestryCategory

Aseguró la carga completa y correcta de todas las categorías de ancestría definidas, cumpliendo con las restricciones de unicidad y no nulidad.

Nombre de la Prueba	Tablas Involucradas	Objetivo	Resultado Esperado	Ciclo 1	Ciclo 2	Ciclo 3
Inserción de datos en BroadAncestryCategory	Broad Ancestry Category	45 filas	Inserción exitosa de todas las categorías esperadas	Todos los datos fueron insertados correctamente sin ningún inconveniente.	Sin cambios, los datos ya estaban correctamente insertados desde el ciclo anterior.	No se requirieron modificaciones, todo en orden.

Tabla B.5: Pruebas Unitarias clase *BroadAncestryCategory*.

B.0.6. Pruebas Unitarias para la clase PopulationSample_Development

Evaluó la correcta inserción de muestras de desarrollo, el cumplimiento de longitudes en campos, la presencia de todos los valores obligatorios y la estandarización de las ancestrías para que coincidieran con las categorías definidas.

Nombre de la Prueba	Tablas Involucradas	Objetivo	Resultado Esperado	Ciclo 1	Ciclo 2	Ciclo 3
Inserción de datos en Population Sample_ Development	Population Sample_ Development	Verificar la correcta inserción de los datos en la tabla	Todos los datos insertados correctamente	Falló la carga de datos por errores en campos obligatorios, valores de longitud y valores de ancestría no mapeados.	Se ajustaron los valores de longitud, se completaron campos nulos y se corrigieron valores de ancestría.	Todos los datos fueron insertados sin errores.

Validación del tamaño del campo cohort	PopulationSample_Development	Verificar que los valores en la columna cohort no excedan el tamaño permitido	Todos los valores insertados sin errores por longitud	Se detectaron errores por longitud excesiva en el campo cohort.	Se ajustó el tamaño del campo y se reintentó la carga.	Todos los valores fueron insertados correctamente.
Validación de campos obligatorios	PopulationSample_Development	Asegurar que los campos requeridos tengan valores válidos	Todos los registros contienen los valores obligatorios	Se encontraron valores nulos en el campo obligatorio numberOfIndividuals.	Se corrigieron los registros con valores faltantes.	Todos los registros contienen información completa.
Verificación de ancestría mapeada	PopulationSample_Development, BroadAncestryCategory	Verificar que los valores de ancestría estén correctamente mapeados en el sistema	Todos los valores de ancestría coinciden con los definidos en BroadAncestryCategory	Algunos valores de ancestría no coincidían con los registros de referencia.	Se estandarizaron los valores de ancestría.	Todos los valores fueron correctamente mapeados.

Tabla B.6: Pruebas Unitarias de la clase *PopulationSample_Development*.

B.0.7. Pruebas unitarias para la clase ModelEvaluation

Verificó la inserción de evaluaciones de modelos junto con sus métricas y poblaciones asociadas, además de confirmar que todas estuvieran vinculadas a publicaciones válidas.

Nombre de la Prueba	**Tablas Involucradas**	**Objetivo**	**Resultado Esperado**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
Inserción en ModelEvaluation y dependencias	Model Evaluation, PerformanceMetric, PerformanceMetricEvaluation, EvaluationPopulationSample	Verificar que al insertar registros en Model Evaluation, se generen correctamente los datos asociados en las tablas relacionadas	Todos los datos insertados correctamente con sus relaciones respectivas	La carga falló porque las ancestrías necesarias en Broad Ancestry Category aún no estaban insertadas.	Se insertaron las ancestrías faltantes, pero algunos modelos aún fallaban por no tener publicación asociada en Publication.	Todos los datos se insertaron correctamente. Las publicaciones fueron completadas y las relaciones se establecieron exitosamente.

Tabla B.7 – continuación

Nombre de la Prueba	Tablas Involucradas	Objetivo	Resultado Esperado	Ciclo 1	Ciclo 2	Ciclo 3
Validación de publicación asociada (PubMed)	Model Evaluation, Publication	Verificar que cada modelo evaluado esté correctamente relacionado con una publicación válida	Todos los registros deben tener un `pubmedId` válido	Se detectaron modelos sin publicación asociada. Se omitieron registros por error de clave foránea.	Se identificaron y registraron los modelos faltantes. Se preparó una solución para insertar publicaciones dummy o excluirlos.	Todos los modelos evaluados tienen publicación válida y se insertaron correctamente.

Tabla B.7: Pruebas Unitarias de la clase *ModelEvaluation*.

B.0.8. Pruebas Unitarias para la clase BroadAncestryInModel

Comprobó la correcta relación entre modelos PRS y categorías de ancestría, asegurando que no hubiera errores de integridad referencial y que los datos permanecieran consistentes entre ciclos.

Nombre de la Prueba	**Tablas Involucradas**	**Objetivo**	**Resultado Esperado**	**Ciclo 1**	**Ciclo 2**	**Ciclo 3**
Inserción de datos en BroadAncestryInModel	Broad Ancestry InModel, PRSModel, BroadAncestryCategory	Validar que se inserten correctamente los registros que relacionan modelos PRS con ancestrías amplias	Todos los registros insertados correctamente sin errores ni conflictos de claves	Los datos se insertaron correctamente desde el primer intento. No se presentaron errores de validación ni de integridad referencial.	Sin cambios. Los datos se mantuvieron consistentes y correctamente relacionados. Se confirmó la integridad de los datos y relaciones.	Se revisaron nuevamente todas las relaciones y no se detectaron modificaciones necesarias. La estructura y consistencia de los datos se mantuvo intacta tras pruebas adicionales.

Tabla B.8: Pruebas unitarias de la clase *BroadAncestryInModel*.

Apéndice C

Anexo 3

Manual de Usuario del Sistema Web de Priorización de Modelos PRS

C.1. Introducción

Este manual describe el uso de la plataforma web desarrollada para la selección, filtrado y priorización de modelos de puntuación de riesgo poligénico (PRS). La plataforma permite al usuario explorar y comparar modelos PRS en función de características genéticas, categorías de rasgos y grupos ancestrales.

C.2. Pantalla de Phenotype Selector

La primera vista del sistema corresponde al **Phenotype Selector**. Aquí el usuario debe realizar una serie de selecciones para personalizar el análisis:

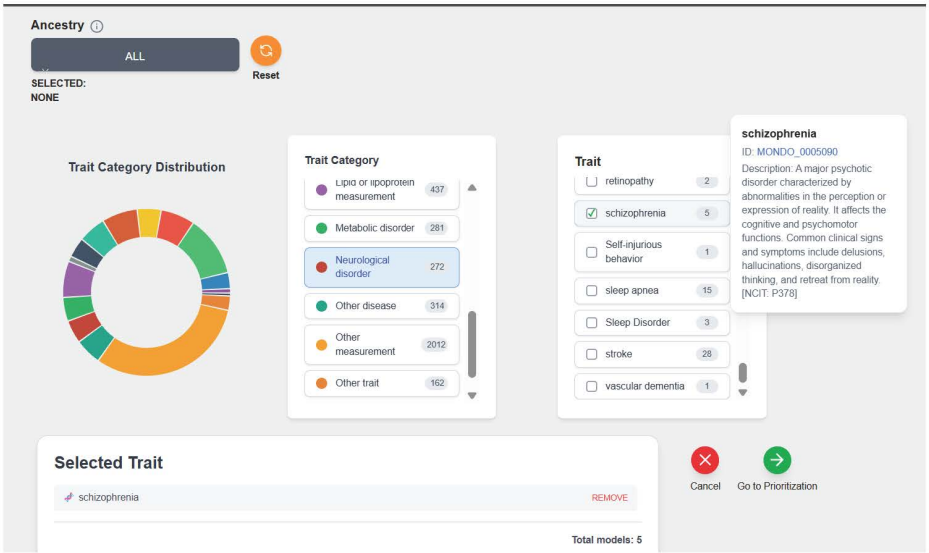


Figura C.1: Pantalla Phenotype Selector

C.2.1. Selección de Ancestría

El primer paso del proceso consiste en seleccionar una o más **ancestrías de interés** mediante el componente de *multiselección*, el cual permite escoger muchas ancestrías. Esta selección es crucial, ya que determina los *trait categories* y *traits* que serán visualizados a continuación para su análisis.

Además, el sistema cuenta con un botón de **Reset** que se utiliza para **reiniciar completamente la interfaz**. Al presionar este botón, se eliminan todas las ancestrías seleccionadas, junto con las categorías de rasgos y rasgos seleccionados, permitiendo al usuario comenzar el proceso desde cero.

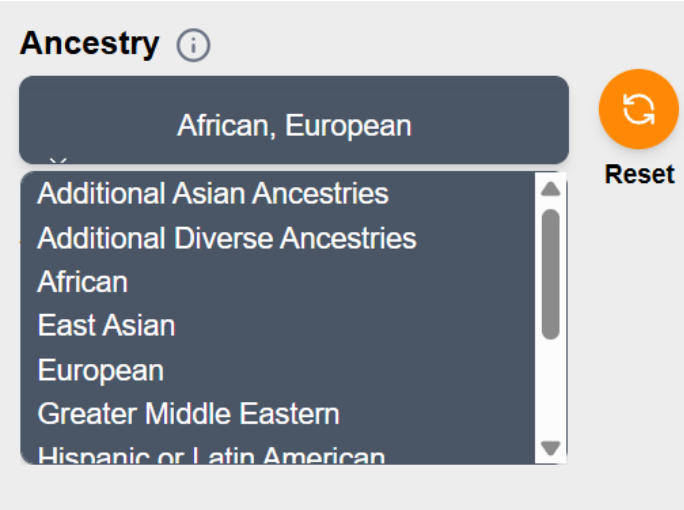


Figura C.2: Selección de ancestría

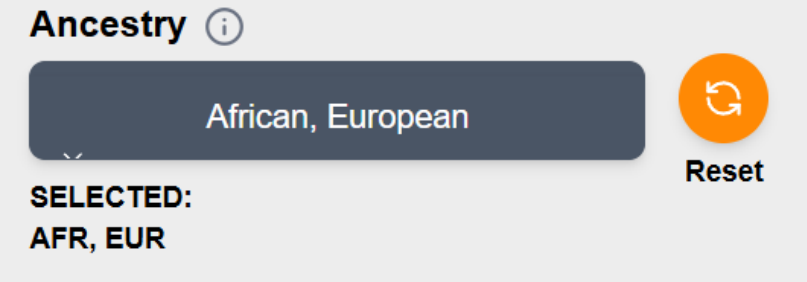


Figura C.3: Selección de ancestría

Esta selección filtrará las categorías de rasgos disponibles para que sólo se muestren aquellas que contienen modelos PRS desarrollados para la(s) ancestría(s) seleccionada(s).

C.2.2. Triat Category

Luego de elegir la ancestría, se filtran las categorías de rasgo (*Trait Categories*) relevantes. El usuario puede hacer clic sobre una categoría para desplegar los rasgos específicos asociados.

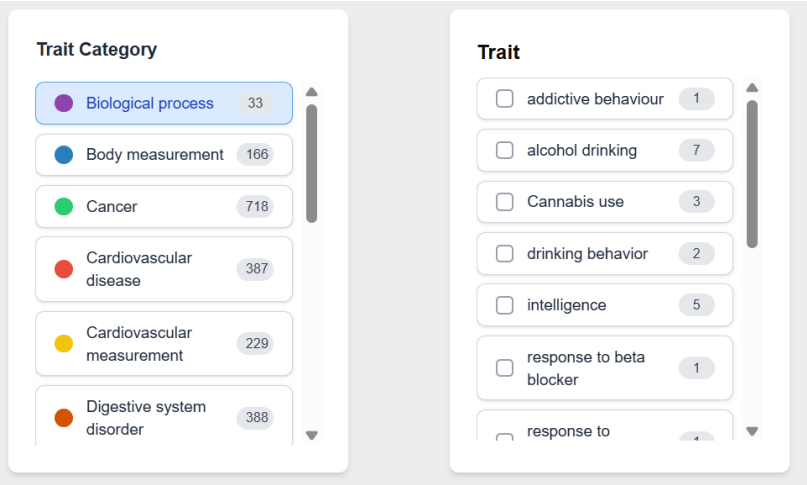


Figura C.4: Filtrado por categoría de rasgo

C.2.3. Visualización Gráfica de Categorías de Rasgo

Además del listado textual, el sistema proporciona un gráfico interactivo donde se visualizan las categorías de rasgos y la cantidad de modelos PRS asociados a cada una. Esto permite al usuario tener una vista general de la distribución de modelos por categoría, facilitando la selección informada.

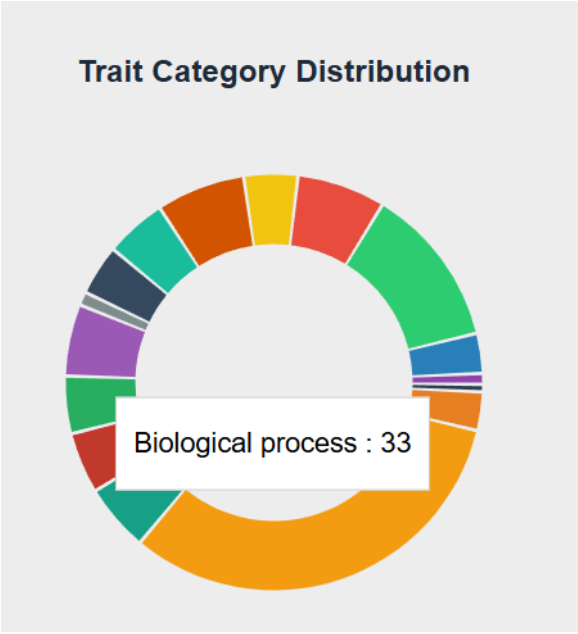


Figura C.5: Gráfico de distribución de Trait Categories con número de modelos

C.2.4. Selección de Rasgos y Modelos

Cada rasgo mostrado permite ser seleccionado junto a los modelos PRS asociados. Al pasar el cursor sobre un rasgo, se despliega un *pop-up* informativo que contiene una breve descripción del rasgo y un enlace directo a su correspondiente ontología, lo que permite al usuario obtener información adicional antes de tomar una decisión.

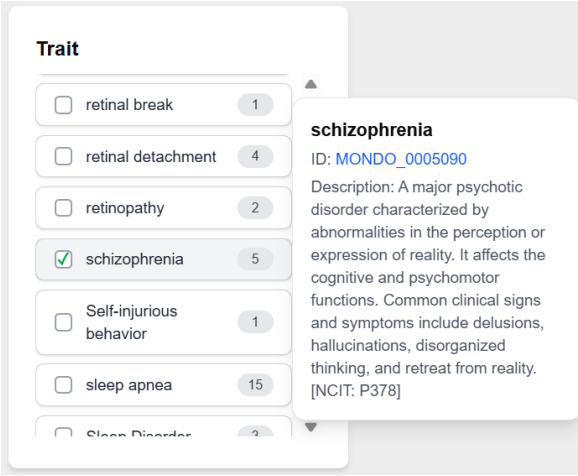


Figura C.6: Selección de rasgos con modelos asociados

Una vez que el usuario ha seleccionado los modelos deseados, puede hacer clic en el botón para continuar al módulo de priorización.

C.2.5. Resumen de Modelos Seleccionados

Una vez que el usuario ha seleccionado uno o más rasgos de interés se ponen en un cuadro de resumen. Este módulo muestra los **traits seleccionados**, cada uno acompañado del número de modelos PRS asociados que serán considerados en la etapa de priorización.

Además, cada rasgo dentro del resumen cuenta con una opción para ser eliminado individualmente mediante el botón *REMOVE*, permitiendo al usuario afinar su selección antes de continuar.

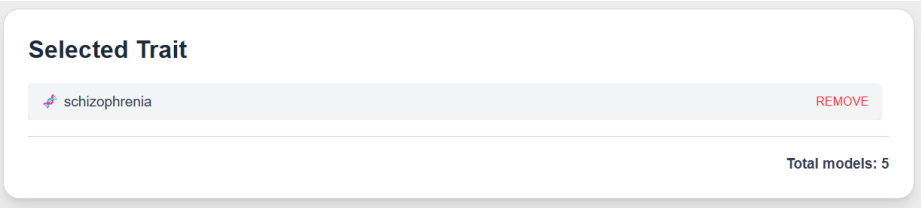


Figura C.7: Resumen de traits seleccionados y cantidad total de modelos PRS a priorizar

C.2.6. Acciones Finales: Ir a Priorizar o Cancelar

Una vez seleccionados los rasgos y modelos deseados, el sistema presenta dos opciones principales de acción:

- **Cancel:** Permite reiniciar el proceso. Al hacer clic en este botón, se eliminan todas las ancestrías seleccionadas, todos los traits y modelos seleccionados, devolviendo al usuario al estado inicial del sistema.
- **Go to Prioritization:** Dirige al usuario hacia el módulo de priorización, donde podrá aplicar criterios específicos para obtener un ranking de los modelos PRS seleccionados.

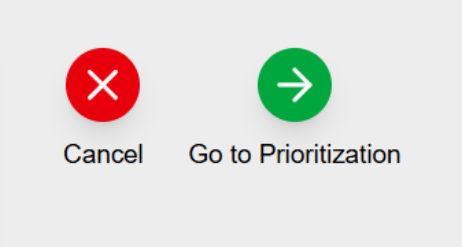


Figura C.8: Botones de acción: Cancelar o avanzar a priorización

C.3. Módulo de Priorización de Modelos PRS

En esta sección, el usuario verá una tabla con los modelos seleccionados y podrá iniciar el proceso de priorización.

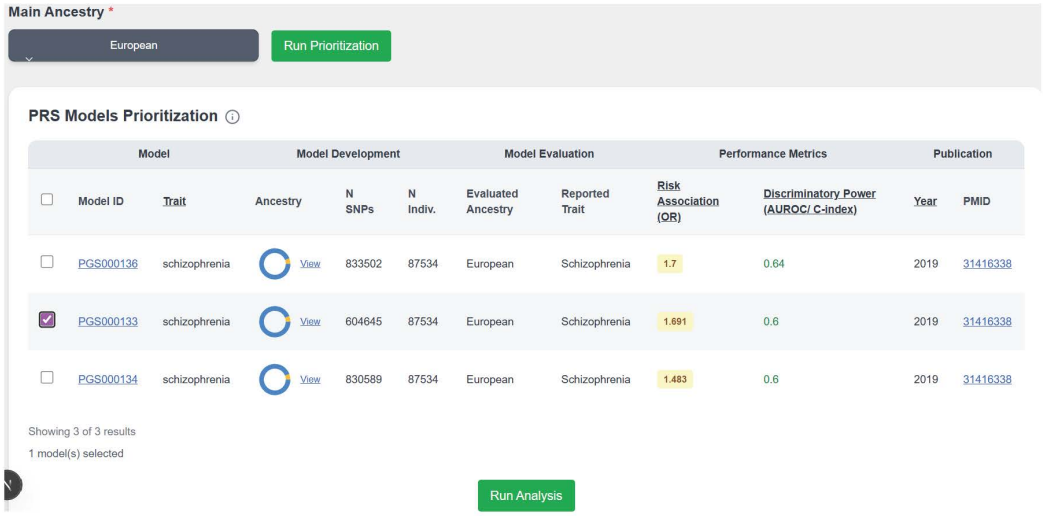


Figura C.9: Botones de acción: Cancelar o avanzar a priorización

C.3.1. Selección de Ancestría de Priorización

Para priorizar los modelos seleccionados, es necesario elegir una ancestría principal. Esta elección influye directamente en el cálculo del puntaje de prioridad de cada modelo, ya que se evalúa su compatibilidad con la ancestría seleccionada.

Una vez realizada la selección, se debe presionar el botón **Run Prioritization** para ejecutar el proceso de priorización y obtener el ranking correspondiente de los modelos PRS.

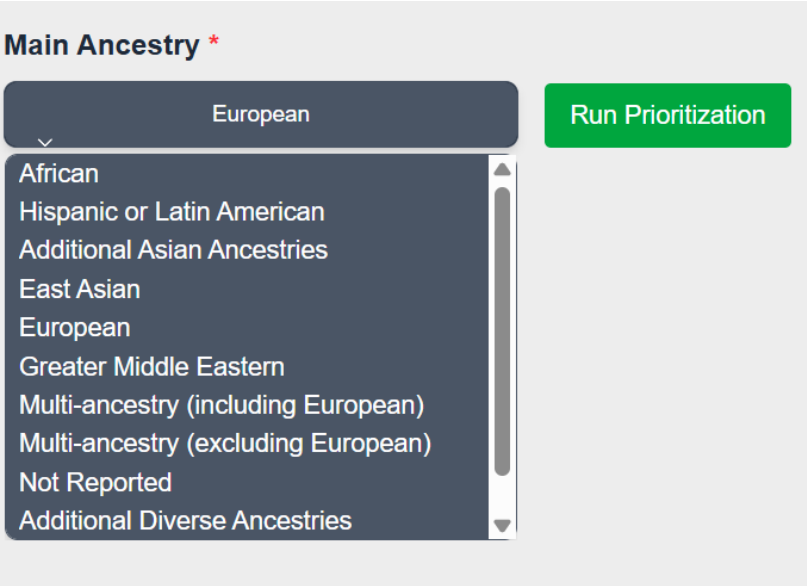


Figura C.10: Selección de ancestría para priorización

En caso de que el usuario haya seleccionado una o más ancestrías en el módulo **Phenotype Selector**, estas serán automáticamente visibles al ingresar a la página del **Módulo de Priorización**. Esta funcionalidad permite mantener la coherencia en el análisis, facilitando la continuidad del flujo de trabajo sin necesidad de repetir la selección de ancestrías.

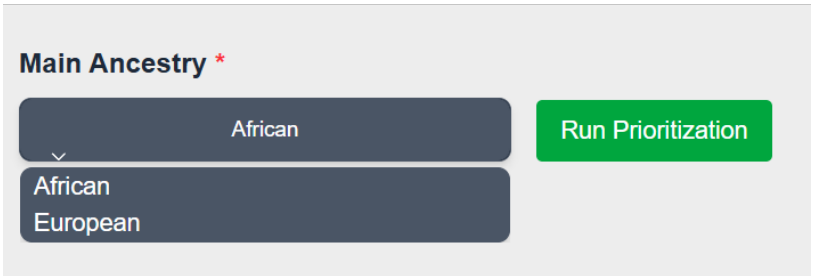


Figura C.11: Selección de ancestría para priorización

C.3.2. Visualización del Ranking de Modelos

Una vez realizada la selección, el sistema genera automáticamente un ranking de modelos PRS según criterios definidos: coincidencia de ancestría, poder discriminativo

(AUC/C-index) y asociación de riesgo (Odds Ratio).

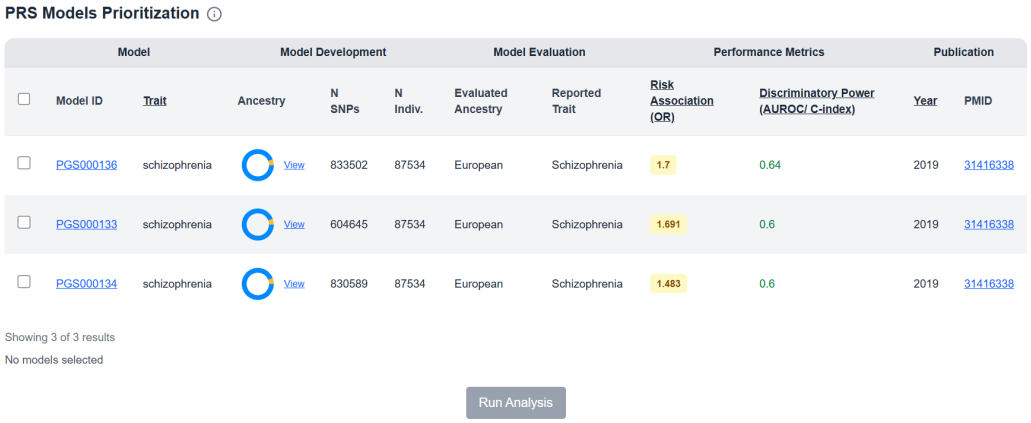


Figura C.12: Tabla de priorización de modelos

C.3.3. Selección y Ordenamiento Final

En la tabla de modelos priorizados, ubicada en el módulo de priorización, el usuario puede seleccionar hasta un máximo de **10 modelos** para su análisis final. Esta tabla permite ordenar los modelos por diferentes columnas, tales como: **rasgo**, **asociación de riesgo**, **poder discriminativo** y **año de publicación**, lo que facilita la comparación y evaluación de los modelos según distintos criterios.

Adicionalmente, se permite el **reordenamiento manual** de los modelos seleccionados mediante la funcionalidad de *arrastrar y soltar* (drag and drop). Este reordenamiento solo está disponible cuando no hay una ordenación activa aplicada en la tabla.

Para facilitar la restauración del orden original de los modelos, existe un botón **Reset Order** ubicado en la esquina superior izquierda de la tabla, el cual permite deshacer cualquier reordenamiento manual y volver al orden de priorización por puntaje.

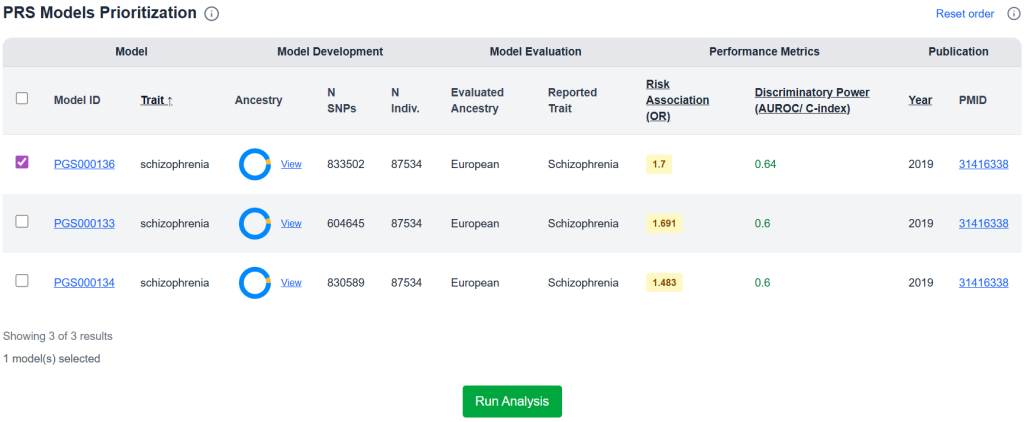


Figura C.13: Ordenamiento personalizado de modelos

C.3.4. Manejo de Ausencia de Modelos para una Ancestría

En situaciones donde el usuario selecciona una ancestría principal para realizar la priorización, pero no existen modelos PRS validados para dicha combinación de rasgo y ancestría, el sistema despliega un mensaje de advertencia visible en pantalla.

Este mensaje, mostrado en rojo, indica de manera clara que no se encontraron modelos disponibles para la ancestría elegida, como se ilustra en la Figura C.14. Esta retroalimentación evita resultados vacíos y orienta al usuario a realizar una nueva selección.

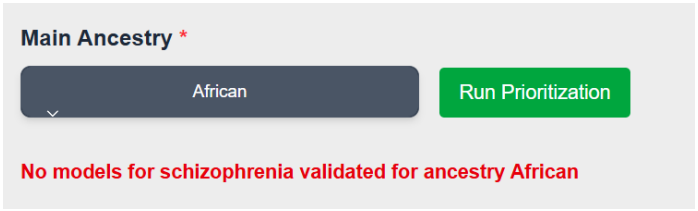


Figura C.14: Ordenamiento personalizado de modelos